Reconfigurable and Modular Mobile Robotics Platform for Remote Experiments

http://dx.doi.org/10.3991/ijoe.v9i3.2554

Yannick Verbelen[1], Pieter Taelman, An Braeken[1] and Abdellah Touhafi[12]
[1] Eramsus University College Brussels, Brussels, Belgium
[2] Vrije Universiteit Brussel, Brussels, Belgium

***Abstract*—In this paper a hardware and software architecture based on a modular approach for a reconfigurable mobile robot is developed with intended use as remote experiment. A Field Programmable Gate Array (FPGA) was chosen as control unit for the robot and allows the implementation of custom hardware and software by the user. FPGA configurations are downloaded to the robot using a selectable wireless interface. Users can take control of all elementary robot hardware, and an on-system recovery mechanism is included to ensure a fallback to a golden image in case of errors in downloaded bit streams.**

***Index Terms*—Modular robotics, FPGA, remote reconfiguration.**

I. INTRODUCTION

The decreased costs and wide availability of stepper motors and brushless DC motors (BLDC) have led to the introduction of robots in a wide range of engineering fields outside conventional mechanics and mechatronics, including electric and electronic engineering. The relative ease with which modern educational robots can be programmed have also opened opportunities towards informatics and ICT, among others. An immediate output of the growing interests of academic research in robotics is the organization of various robot competitions. Since 2004 the RoboCup initiative spawned a yearly showdown for robots in different size and weight categories, and has since evolved into an incubator of novel mechatronics technologies.

Currently, 15 academic institutions worldwide support one or more RoboCup teams [19], continuously trying to improve their robots to climb in the yearly RoboCup rankings. The game competition element is named as one of the decisive factors in motivating students to participate in intra or extra curricular robotics activities.

The wide interest of students opens up opportunities for long running robotics projects such as RoboCup participation. The Technical Universities of Delft and Eindhoven in the Netherlands have demonstrated that student teams are able to improve robots year after year, and do not consider it a limitation of creativity to work on the improvement of existing prototypes.

Unfortunately, the practical organization of both intra and extra curricular robotics activities is limited by the available robot hardware. During courses there must ideally be as many robots available as students to allow them to work independently on the programming of the robots. Aside from the initial construction costs of the robots, significant resources are also required for maintenance and space required for docking stations and play fields.

The availability of the robots, limited to the campus area, poses another problem. With the exception of simulation tools, students enrolled in robot oriented courses are unable to practice or work on their projects at home. This is a fatal drawback in the engineering context, as students need time to familiarize themselves with complex subjects. This time is not sufficiently available during the regular courses, and as such it is a must that learning tools are at the student's disposal from their home residences.

Since less robots are typically available than there are students signed in for a course, students cannot be given a personal robot for the entire duration of the course which can span from 4 months up to an entire academic year. Secondly, transportation of equipment on metro or bus is not an ideal situation due to the size and fragile nature of this material. Hence, the only option left is to reside the robots on the campus and make them physically accessible to students during course hours, as well as remotely available as remote experiments.

In this publication, the design and implementation of a reconfigurable mobile robotics platform tuned for application in remote experiments is presented. Section II summarizes previous work in the field and identifies the limitations of current implementations. Section III presents the proposed reconfigurable architecture, and section IV elaborates the reconfiguration mechanics. After conclusions in section V the paper is concluded with proposals for future work in section VI.

II. PREVIOUS WORK

Partially thanks to the popularity of the RoboCup initiative, a large number of academic institutions around the world have established research groups dedicated to the construction of football robots [3, 10, 15].

Unsurprisingly, 78% of countries in the humanoid football world cup top 10 also earned a stable position in the Robocup rankings, thus reflecting a nationwide interest in the subject. As part of the initiatives to popularize the research for RoboCup related technology, work is also being done in the field of remote reconfiguration of robot platforms [17] and how they can be optimized for e-learning in the form of remote experiments [5].

Currently, several robotics related remote experiments have been deployed by various institutions [16], including remote experiments with FPGA based robot architectures [9]. The to date existing solutions offer students the opportunity to upload and test custom firmware (hardware or

software) by reconfiguring a microcontroller or FPGA in a static robot architecture. Although this suffices for robotics study on an introductory level, mastering the topic requires students to be able to modify the hardware of the robot itself outside the digital constraints of the physical FPGA. This allows students to acquire experience with the practical limitations of electronic and mechanic building blocks.

The novelty of our work is in the expansion of hardware customizability of the robot, providing a framework of separate robot building blocks called *modules*. Furthermore a transparent mechanism for wireless reconfiguration and control of the robot is proposed.

III. MODULAR MOBILE ROBOTICS ARCHITECTURE

The high level of robot hardware complexity is a prohibitive factor in the understanding of robotics. In a classic remote experiment, this problem is bypassed by pulling an additional user layer over the top layer architecture. This approach simplifies the system for the user, and allows the control unit (typically a remote experiment server) to retain control over the experiment. However, when designing a robot architecture that must be usable on both a shallow and intermediate level, it is necessary to define a set of constraints for hardware firmware to avoid conflicts. This design process naturally evolves in the creation of a modular design. To meet the preset requirements, such modular design must be built up from one or more core components and a variable set of expansion modules performing input and output as well as signal processing.

Figure 1 depicts an overview of the total architecture, consisting of 3 main parts. These parts include the FPGA carrier card, consisting of the central processing core unit and corresponding auxiliary core hardware, the wideband bus system, and the IO modules. Each of them are described in the following paragraphs.

A. FPGA carrier card

Since a central processing core unit, together with some essential auxiliary core hardware are an integral part of any setup because they are required for the correct operation of an FPGA, they can be integrated on a single module. This module is the logic center of the robot, and was named the *carrier card*. We first explain these two parts in more detail before giving more technical information on the type of carrier card we used.

1) Central processing core unit

For the design of the RoboCup SSL robots, an FPGA was elected as the primary core component due to its high grade of reconfigurability and ease of use. The chosen Xilinx Spartan 6 FPGA (XC6SLX16 CSG324-2C) is built on 45 nm technology [25] with improved 6 input LUTs to enhance implementation of combinatorial logic. The XC6SLX16 contains 14,579 logic slices, with 4 LUTs and 8 flipflops per slice [25]. This relatively high reconfigurable capacity allows the implementation of complex sequential combinatorial logic. In addition, slices can also be used to implement softcore processing units into the FPGA. The FPGA optimized softcore processor, the Microblaze, requires around 10^3 LUTs and an equal amount of flipflops. This translates into a capacity for several Microblazes as well as on chip bus interconnection logic, which in turn opens up opportunities for high performance calculations such as parallel computing or video stream processing from cameras on the robot. Furthermore it is also possible to implement the 8 bit Picoblaze microprocessor, which is just like the 32 bit Microblaze limited to Xilinx FPGAs. The Picoblaze requires less than 200 slices [6], effectively removing a number of limitations in almost all practical situations due to the large (2278 slices) reconfigurable area available on the Spartan 6.

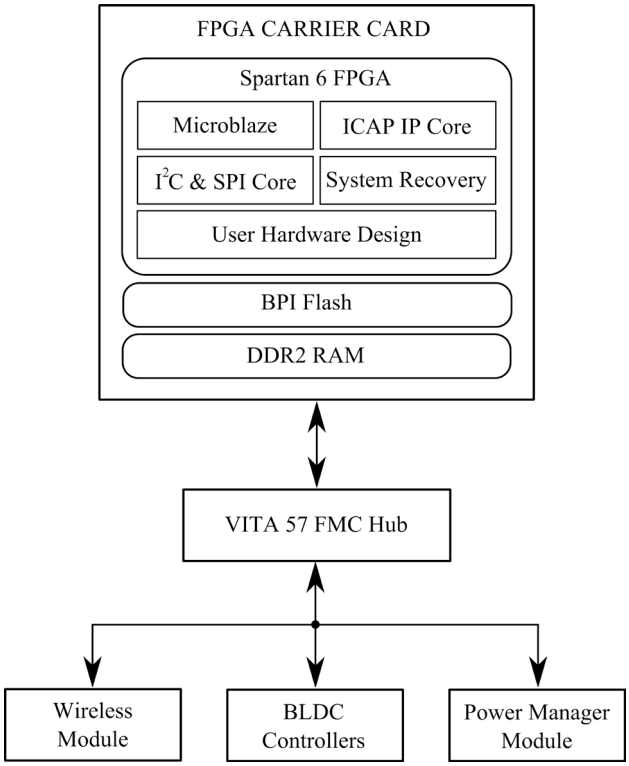

Figure 1. Architecture of the system, showing the FMC Mezzanine Hub connecting the FPGA carrier card with IO modules.

A drawback of the choice for an FPGA based robot core is the increased power consumption in comparison to a dedicated hardcore microcontroller with lower customizability [8]. However, in this application the higher power consumption is not critical because

- the duration of a RoboCup match is limited to 20 minutes [18];
- a module for automatic return to the battery recharge station and charge controller is implemented;
- a scheduler to select an operational robot is provided on the robot controller server, making the recharge procedure of batteries transparant to the user.

2) Auxiliary core hardware

Aside from the central processing unit, the Xilinx Spartan 6 FPGA, 4 additional core building blocks are required to ensure the operational stability of the central processing unit. These building blocks include volatile and non-volatile memory, an internal reconfiguration unit, and a fallback routine for automated system recovery.

The 576 kB of RAM available on the FPGA dye itself is unlikely to be sufficient for both system and user applications, especially for implementations of Harvard architectures [25]. A unit of 128 MB of DDR2 RAM has been directly attached to the Spartan 6 FPGA to relieve this problem. This approach allows the flexibility to choose which critical information is stored on the FPGA dye itself

in high speed block RAM, and which less critical data may be stored externally in the DDR2 memory.

The Spartan 6 FPGA does not have internal non-volatile memory to store a configuration from which it can reconfigure itself on power up. Hence, a portion of on board storage must be non-volatile to store a bootloader for user designs on one hand, and a golden image for fallback on the other. The size of this non-volatile memory must be proportional to the reconfigurable logic of the FPGA since bit file size scales up together with the size of reconfigurable FPGA logic. For the Spartan 6 XC6SLX16, a non-volatile memory of around 4 MB is required per bit file. To accommodate multiple bit files along with bootloader and golden image, a 16 MB parallel flash memory (BPI) has been chosen for the robot as seen in fig. 1.

During remote experiments the FPGA will be frequently reconfigured with custom embedded user hardware and software. To accomplish this reconfiguration, there are 2 possible techniques: full reconfiguration and partial reconfiguration. During a full reconfiguration cycle, a process similar to the initial configuration over JTAG is executed, during which the FPGA is powered down to allow the new configuration to be loaded. This approach has the drawback that the FPGA will draw a current that is typically double or triple of its nominal operation current [4], thus requiring an appropriate power supply able to handle these current spikes. A workaround for this issue is provided by partial reconfiguration. During partial reconfiguration, independent banks within the FPGA are powered down and reconfigured individually. This has the advantage of eliminating the power surge, and leaving the other banks of the FPGA online during reconfiguration. When using this approach, it is possible to locate the robot's platform firmware in one or more banks and (re)configure the user design in other banks, simultaneously transmitting feedback on the (re)configuration process to the user from the banks that are still operational. Despite being an interesting technique, partial reconfiguration is currently not implemented due to its higher complexity in comparison with full reconfiguration [23].

Reconfiguration requires access to the FPGA configuration mechanics from inside the FPGA when the reconfiguration is done independently by the robot. This access is provided by the ICAP (Internal Configuration Access Port) [12]. For this robotics application, a custom ICAP IP core has been developed based on the Virtex 7 ICAP since a similar configuration does not yet exist for Spartan 6.

In the event of a faulty bit stream or an interruption during the configuration process, the FPGA will be stuck in an incorrectly configured state. This may potentially disable the robot's platform firmware, and prohibit further reconfigurations. To prevent these situations, it is necessary to implement an automatic mechanic for system recovery. High end FPGAs like the Spartan 6 already include dedicated logic for interfacing with serial or parallel flash to reconfigure themselves without external control [11], and will automatically fall back to a backup procedure when reconfiguration with the intended bit stream fails. When failure is detected, the FPGA is reconfigured with a golden bit stream stored on a fixed address space in the connected non volatile memory [14].

The automated system recovery mechanism increases the reliability of the robot as a remote experiment because it prevents failure of robot operation in case faulty bit streams are downloaded into the FPGA or unknown factors interrupt the regular reconfiguration process.

B. Wideband bus system

As a result of the complex, expensive and time consuming nature of the design of a custom FPGA board including all the above named features, the Xilinx SP601 Development Kit was chosen as an off the shelf low cost carrier card [26].

The SP601 is equipped with various IO interfaces, the most significant being the VITA 57 interconnection standard. This standard makes use of a high pin count connector with either 160 pins (LPC) or 400 pins (HPC). The availability of such a high number of parallel IO lines is leading the VITA 57 standard to continue its evolution to the standard for FPGA mezzanine card topologies. It should be noted that from the designated pin count (160 or 400) about 50% is effectively available for free assignment by the user—the others being reserved for power, protocol signaling and JTAG [21].

For an academic application such as a remote experiment setup, the VITA 57 FMC standard provides both advantages and disadvantages. Starting with advantages, the high amount of user assignable pins in comparison with traditional header connectors, provides the opportunity to distribute control and data signals from the FPGA board to all parts of the robot. This useful functionality is provided in a single connector, thus eliminating the need of different cable types on the robot. Both the FMC connectors and the fitting flat cables are becoming more widely available from different manufacturers as the integration of the standard in FPGA mezzanine applications progresses [2, 20]. A notable problem with the flat cables that became apparent during testing however is the limited flexibility of said cables. This makes it often challenging to integrate the electronic design within the strictly size constrained robot. Figure 2 shows the FMC mezzanine hub card with a flex cable connecting an IO module. In the pictured setup the hub is not connected to the FPGA carrier card, as evidenced by the empty male *Carrier* FMC connector.

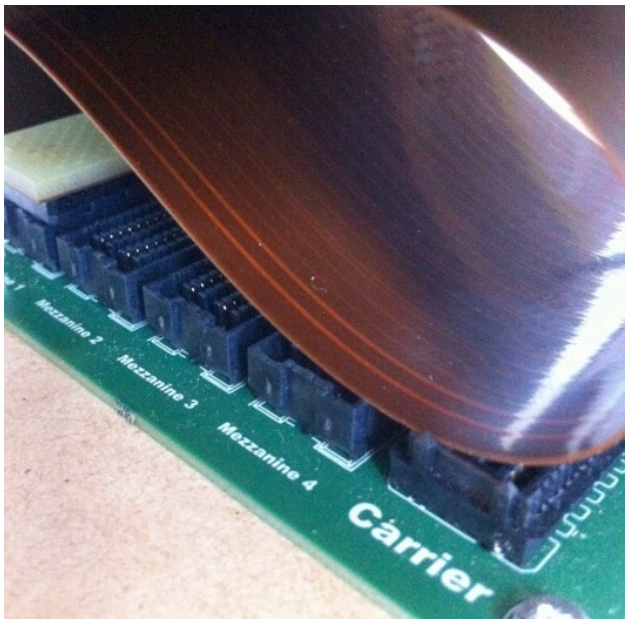

Figure 2. FMC hub card following the VITA 57 standard. Up to 4 IO modules may be plugged into the carrier card. This custom carrier was developed as part of the research.

The use of the FMC connection standard also shows notable disadvantages. Aside from the relatively high cost of connectors and cables (in comparison with the SP601 FPGA board), the high density of the pins together with their arrangement in 4 rows requires high resolution PCBs. Experiments have shown that it is challenging to reflow even the LPC version of the connector to a lab etched PCB without short circuits. A professional grade PCB with solder mask and tented vias is almost a requirement for successful mounting of the connector. This drawback limits the speed with which students can develop custom hardware for the robot. It also increases the price of the hardware with 300 – 500%, due to the high cost of PCB fab manufactured PCBs.

Once mounted, the connector and flat cable interconnection system proved to be a reliable solution for building a modular robot design. The connector provides good electrical contacts between PCBs as well as the potential to use differential signaling and ground return paths. During the construction and testing of the prototype, not a single connector or connection failed. Additionally, the design of both the connector and its footprint prevents wrong orientation (unlike classic 2.54 mm headers), avoiding short circuits and connection errors.

The large pin count of the FMC connector has the primary advantage of providing connectivity for dedicated buses running in parallel, as well as reserved user pins. Although it is possible for modules to reserve a dedicated IO line directly connected to the FPGA (for example to deploy a custom driver in VHDL), the preferred method of communication between the FPGA and the hardware modules is through standard buses. Data lines for the 2 most common embedded buses, I²C and SPI, have been reserved and are driven by the FPGA as master by means of SPI and I²C IP cores. More data lines may be reserved for additional buses when needed.

Dedicated JTAG pins are already reserved in the FMC specification [20, 21] for JTAG, which follows from the intended use of FMC in FPGA mezzanine configurations. On FPGA side, JTAG may be implemented as an IP core unrelated to the Microblaze processor as it is implementable as a single state machine [1]. JTAG data lines of all modules are connected using a chain pattern (e.g. TDI line of a module connected to the TDO line of the previous module) on an FMC interconnection board. This makes the implementation of JTAG transparent for any connected modules, and allows the execution of a Boundary Scan by the FPGA to identify connected module. This is a convenient way of both detecting the presence of IO modules and configuring/programming them, as well as debugging.

In the current version, implementation of JTAG is not a mandatory requirement for modules. Furthermore, debugging and configuring/programming modules wirelessly is not yet implemented in the prototype of the robot.

C. Basic IO modules

Hardware related to movement capability, batteries and wireless communication, is included as IO modules rather than integrated hardware on the FPGA carrier card. This approach has 4 advantages:

- outdated hardware components can be easily upgraded;
- malfunctioning hardware can be replaced without requiring tools or soldering;
- students can replace hardware with their own versions;
- modules can be disconnected for debugging purposes.

Each basic IO module is discussed below.

1) Wireless communication module

The wireless communication module is equipped with a double XBee socket [22] on which up to 2 XBees can be mounted. Both are relayed to the FPGA directly since a wireless interface for the robot is a core feature for which the reservation of a small number of pins on the FMC connector is acceptable. Depending on the power budget attributed to wireless communication, transmission speeds vary from 9.6 kbps for XBee-PRO XSC up to 65 Mbps for XBee Wi-Fi [7]. Either XBee can be selected as the preferred communication medium by means of an I²C interface. This could for example allow fast firmware transfer over Wi-Fi at the cost of increased power consumption, while after reconfiguration the robot is controlled remotely over a much slower and less power hungry wireless interface.

2) BLDC controller module

The primary propulsion mechanism of the robot consists of 4 brushless DC motors (BLDC) of type Maxon EC32, with a power of 15 W when driven with a 24 V DC voltage. This is in line with the specifications of other teams, also using Maxon BLDC motors with powers between 15 W and 50 W (*MRL* and *Immortals*, both Iranian teams). The motors drive custom designed omnidirectional wheels through an external gear. The control of the three motor coils is left to a TMC603 BLDC driver from Trinamic. This IC implements break-before-make logic, a high side driver for N-MOS half bridges, and an integrated current sensor by measuring the voltage drop over the internal channel resistance of the bridge MOSFETs. This boosts motor performance since ohmic losses in current sense resistors are eliminated.

All motor control signals are routed over dedicated IO pins on the FMC connector to prevent congestion of the general purpose IO buses, and to allow modules to probe the actions performed by the motors.

3) Power management module

The robot is equipped with Lithium Cobalt rechargeable batteries in 18650 form factor, with a total capacity of 22.4\,Ah at a voltage of 3.6\,V. The power management module is responsible for the coordination of the power path and for charging the batteries when the robot is positioned in its docking station. The module incorporates low battery detection, and signals this together with various other status flags to the FPGA over I^{2}C. The FPGA firmware then transmits this status information wirelessly back to the server, which accordingly notifies the user over the web interface.

Three additional daughter boards can be clicked on the power management module in piggyback mounting style. These boards perform DC/DC conversion from a 25.9V battery voltage to 24V for the BLDC motors, 5V for powering general purpose hardware, and 3.3V for powering the FPGA. The DC/DC conversion to 2.5V and 1.8V for the FPGA's VCCINT and VCCAUX power supply is done on the FPGA carrier card itself to minimize the voltage drop over PCB traces and cables.

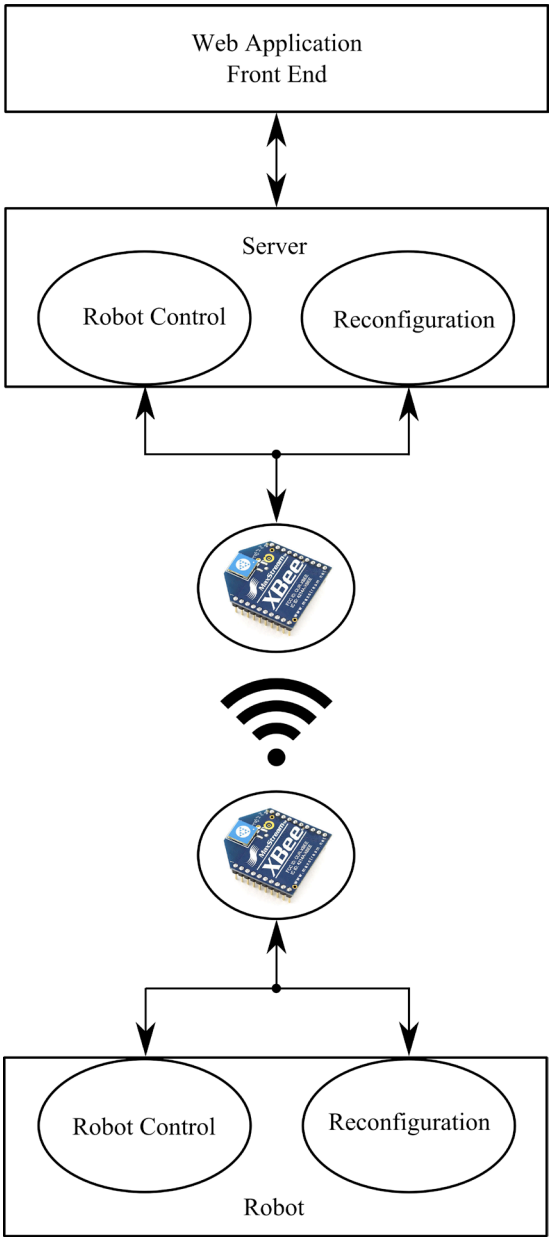

Figure 3. Layout of the wireless reconfiguration mechanism, showing a split architecture for robot control and remote reconfiguration.

IV. WIRELESS REMOTE FIRMWARE RECONFIGURATION

When the wireless IO module is connected to the FPGA carrier card, the firmware of the robot can be dynamically reconfigured. Fig. 3 shows the layout of the wireless configuration mechanism between server and robot through a web application front end. This includes wireless transmission of a new bit stream for the FPGA, incorporating both embedded hardware and software to be run on the FPGA's embedded softcore processor, the Microblaze. Reconfiguration is performed in 3 steps: synthesis, firmware transfer and rebooting the FPGA. These steps are elaborated in the following paragraphs.

A. Remote synthesis

Before the robot can be reconfigured, firmware matching the targeted FPGA must be prepared. Users provide code to be run on the robot, as well as the VHDL description of optional embedded hardware (IP cores). This combination is treated as the *user application*, in contrast with the platform firmware necessary to ensure continuity of the robot's operation as remote experiment.

The user application is uploaded to a server system using a web interface, which allows students to upload VHDL and C files. The uploaded information is checked by the server to ensure compatibility with the robot design, and then merged together with the platform firmware to obtain the VHDL description of a design ready for synthesis. Synthesis on the server has 2 notable advantages:

- no synthesis tools need to be installed on the user's computer, e.g. VHDL and C code can be written in a simple text editor;
- the larger system resources of a server system speed up the synthesis process and do not slow down the user's computer.

During synthesis, feedback from the Xilinx synthesis tool is caught by the parent web application and sent to the client. This allows the user to review any errors and obtain an overview of their design.

After synthesis, the hardware structure of the FPGA is retrieved from a predefined user constraints file (UCF), which describes the pin connections of the FPGA to FMC connector and on board memory. If the user reserved any additional pins on the FMC bus, these constraints are translated to nets and merged together with the UCF file. The web application then proceeds with placing, mapping and routing to transform the logic on register transfer level (RTL) generated in the synthesis process to a hardware layout. If any problem occurs in these steps, they are again notified to the user.

Finally, the user's C code is compiled and merged together with the embedded FPGA hardware into a bit file, which optionally can be downloaded from the server by the user. The bit file is stored in a repository linked to the user's session, and converted into the Intel MCS-86 Hexadecimal Object (MCS) format required for compatibility with the target flash memory on the robot [22].

B. MCS file transfer

When new a new user application or source code has been assembled into MCS format, the server initiates a reconfiguration cycle by sending a request to the robot. This causes the robot to enter a reconfiguration state and reserve on board memory to store the new MCS file. The robot then starts listening for data and confirms its state to the server. The server responds by splitting up the MCS file in records and sequentially transmitting these records to the robot using the selected XBee interface.

A simplified ZIP compression is applied to the MCS file to speed up the transmission process. Every record is assigned a serial number and a hash. If the hash is 0, then the contents of the record are checked to verify the data within. If the entire record is 0, then it is skipped and not transmitted to the robot. From the serial numbers of individual records, the robot is able to identify which records have not been sent and thus 0, and automatically writes a zero record to the local memory. Since the transmission of the firmware is the bottleneck in the reconfiguration process [24], this technique speeds up the perceived responsiveness of the robot for the user. Depending on the performance of the server (the decisive factor for synthesis), a gain of 50% may be achieved for small designs.

Because of technical reasons, the transmission of the MCS file records to the robot is not handled by the web application. Control over this section of the reconfiguration process is handed over to a C# application, directly controlling a wireless transmitter. This is conform literature [24].

C. FPGA reconfiguration

Once the complete MCS data has been stored on the robot, the Internal Configuration Access Port (ICAP) is configured with the necessary information to execute the reconfiguration process. This includes the memory addresses of the new firmware in flash memory, together with the address of a fallback image (*golden image*). The ICAP module in the Xilinx EDK is only implementable on Virtex class FPGAs [22], requiring an adaptation to be developed for the Spartan class FPGAs. This was implemented as an IP core, which addresses the standard Xilinx ICAP IP core. Finally, a software command is sent to the Microblaze to initiate the reconfiguration cycle. If necessary in case of larger designs, additional software is subsequently downloaded as an SREC file.

D. Wireless reconfiguration performance

Because of the ZIP compression optimization described above, the reconfiguration speed depends on the actual contents of the bit stream that is sent to the FPGA. The platform firmware, acting as the operating system of the robot from the user's perspective, already consumes 87% of the FPGA's reconfigurable area. This corresponds to 1,993 slices from a total of 2278 available slices on the XC6SLX16 Spartan 6 FPGA, from which ca. 1,324 are spent on the Microblaze processor and the remaining 669 slices on IP cores to support the various hardware on the robot. Figure 4 shows the resource usage on the Spartan 6 FPGA in comparison with the total available FPGA resources. Note that only slices are used as a resource parameter since in this application most other resources such as IOBs etc. will remain constant. Configuring the design with the IEEE 802.15.4 wireless standard with a maximum flow of 250 kbits/s requires a total of 10 minutes and 30 seconds. Erasing the flash memory to store the MCS file takes another 17 s, bringing the total reconfiguration time to 10 minutes and 47 seconds since the FPGA reconfiguration itself is 2 orders of magnitude faster. It must be noted that the exact reconfiguration time T equals the amount of bits to reconfigure (design size dependent) N, divided by the CCLK frequency f_{CCLK} and the number of bits being read per clock cycle n:

$$T = \frac{N}{f_{CCLK} \cdot n}$$

Although Xilinx confirms only a 50% accuracy interval for f_{CCLK}, reconfiguration will always be done in less than a second for the Spartan 6 on board CCLK even for completely occupied FPGAs. Since this is clearly not the most important bottleneck in the current application, applying a more stable external f_{CCLK} does not result in any noticeable performance gain and was accordingly omitted.

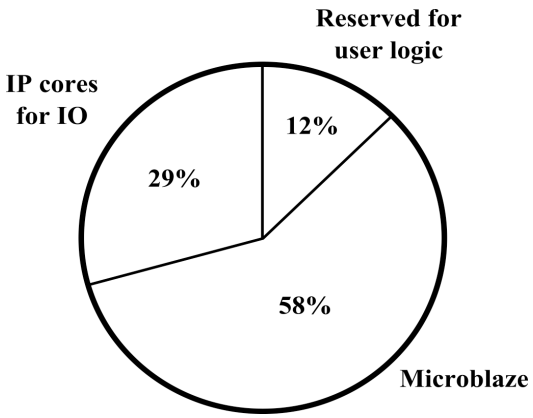

Figure 4. Breakdown of available FPGA resources over microblaze (58 %), auxiliary IP cores for IO (29 %) and 12 % (285 slices) available for user designs.

When extra user logic (VDHL as IP core) or software is added to the design, the MCS file transfer time will be increased. Experimental results confirm that transfer times rise proportionally with used FPGA reconfigurable area, as seen in figure 5. If only 10 slices of the FPGA's reconfigurable area are used (ca. 1% of the available area), the transfer time of the design corresponds mostly to the structural overhead of the FPGA, and was measured to be around 15s. When increasing the used area, transfer time also increases linearly, with ca. one additional second per 4 additional slices.

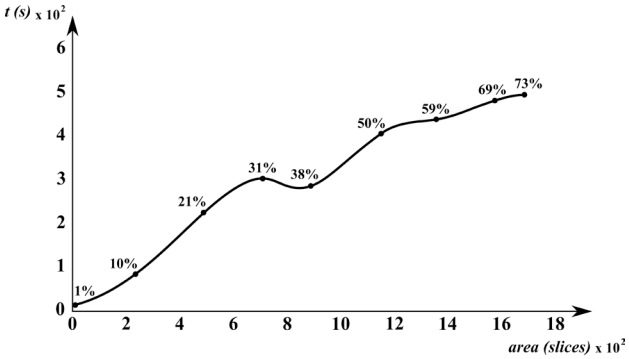

Figure 5. Time required for wireless transmission of designs with various occupied are. The approximately linear correlation is an indication for an efficient compression algorithm, preventing empty blocks from being transmitted. The design size in every measurement is situated as a percentage of the FPGA's available slice resources.

Despite being the industry standard for low power, short distance RF communication, the bandwidth provided by the IEEE 802.15.4 standard is much too slow to be realistically usable in remote experiment applications with FPGAs. For this reason, the current implementation is currently extended with the IEEE 802.11 standard which allows transfer speeds up to 65 Mbits/s. This makes it possible to reduce transfer times with a factor 260, allowing a design for a completely filled Spartan 6 FPGA to be transmitted in ca. 3 s instead of the current 11 minutes and 40 seconds. A drawback of this approach, however, is the increased complexity of the Microblaze design due to the overhead required for IEEE 802.11 implementation. Hence, a faster transfer speed comes at the cost of less available FPGA area for the user.

V. CONCLUSIONS

The design methodology for a modular reconfigurable robotics platform was discussed with a special focus on applications in educational academic environments. The proposed architecture centers around a Spartan 6 FPGA as reconfigurable unit, with multiple auxiliary modules attached to it. The VITA 57 FMC standard for FPGA mezzanine architectures was selected as an interconnection interface, and various communication buses were defined.

A wireless interface using the XBee standard was implemented to communicate between a central reconfiguration server and the robot. A web application allows users to upload C and VHDL source files, which are remotely compiled and synthesized. A compression technique is applied to the synthesized firmware to lower the amount of data to be remotely transmitted. Finally, the FPGA reconfigures itself using the embedded ICAP. A webcam is positioned overhead to provide the user visual feedback on the robot's operation by using a video stream in the web application.

Fig. 6 gives an impression of the physical implementation of the features described in this work.

The reconfigurable robotics platform presented in this paper is currently in the development state of an early project outline, with only a single prototype completed. A group of students in electronic engineering will work on finishing a basic team of 3 robots and identify existing bottlenecks in the assembly process. Four Bachelor students are working in parallel on the development of custom hardware modules for ultrasonic ranging and navigation in the context of their Bachelor theses.

A battalion of reconfigurable robots is planned to be integrated in informatics courses in the first Bachelor Electronic Engineering in the near future as an introduction to programming. In higher years of the electronics specialization, students will work in parallel on the improvement of the next generation of robots. There is also potential to teach students the basics of VHDL using the developed FPGA platform.

Consequently there is more work to be done on the deployment of the robots to make them ready for use in a course. This primarily comprises of increasing the reliability of the robots as well as streamlining the ease of reconfiguration using the web interface. This also includes testing for robustness, as constant failure of robots will stall any course making use of them.

Finally, attention must be directed towards the speed optimization of reconfiguration since this currently takes too long to be practically usable. Further integration of high speed wireless communication protocols and combined with efficient compression algorithms should be able to push the reconfiguration time under a few minutes, unlike the 10 – 15 minutes required by the current generation.

Figure 6. Overview of an early robot prototype. At the top level, a capacitor array is provided to store energy for the shooting mechanism of the robot. White parts are manufactured in ABS on a 3D printer. FPGA carrier card, batteries and FMC flat cables are omitted for clarity.

ACKNOWLEDGMENT

This work has been partially realised with contributions of the following students of the 3rd bachelor electronic engineering: Alexander Bulezyuk, Matthias Carlier, Joris de Hoog, Pieter Leemans, Frank Vanbever and Steven Vanden Branden.

The research above was partially funded by the Prins Filip Fonds in Belgium. It also makes use of Secure Techniques for the Remote Reconfiguration of Embedded Systems developed in the STRES project which was financially supported by IWT, the Flemish Agency for Innovation by Science and Technology, under the Tetra program.

REFERENCES

[1] Amontec, *JTAG Interface: Simple Introduction*, Application Note AN004, v1.0, 2005.

[2] ANSI/VITA 57, *American National Standard for FPGA Mezzanine Card (FMC) Standard*, ANSI/VITA 57, 2008.

[3] BUIU Catalin, DUMITRACHE Ioan, MIHAI Florin and GILLET Denis, *Experiments in Remote Collective Robotics. New Pedagogical Scenarios*, Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2004, p. 3836 - 3841, Lugano, Switzerland, 2004.

[4] BURSKY Dave, *FPGA Startup and Configuration Issues*, ED Online 10506, Actel, 2009.

[5] CEREZO Eva, BALDASSARRI Sandra, PINA Alfredo, HUIZI Lore, *Learning Robotics via Web: Remote Experiment Systems for Distance Training*, Computing and Systems Engineering Department, Univeristy of Zaragoza, Spain.

[6] Core Technologies, *Soft CPU Cores for FPGA*, http://www.1-core.com/library/digital/soft-cpu-cores/soft-cpu-cores.pdf, p. 2, accessed online June 27, 2012.

[7] Digi, *Xbee Wireless RF Modules,* http://www.digi.com/xbee/, accessed online July 6, 2012.

[8] GARDINER Koji, *Comparing Power Consumption of FPGAs with Customizable Microcontrollers*, Standord University, Electronic Engineering Journal, 2008.

[9] KHAMIS Alaa M., RIVERO D. M., RODRIGUEZ Franscisco J., SALICHS Miguel A., *Pattern-based Architecture for Building Mobile Robotics Remote Laboratories*, Department of System Engineering and Automation, Carlos III University of Madrid, Madrid, Spain, IEEE International Conference on Robotics and Automation 2003, Proceedings. ICRA '03, 2003.

[10] KHAMIS Alaa M., RODRIGUEZ Francisco J. and SALICHS Miguel A., *Remote Interaction with Mobile Robots*, Department of

Systems Engineering and Automation, Carlos III University of Madrid, Madrid, Spain, published in Autonomous Robots, 2003, vol. 15, no. 3, p. 267-281, 2003.

[11] KONRAD M., BONNES U., BURANDT C., EICHHORN R., ENDERS J. and PIETRALLA N., *A Digital Base-band RF Control System*, TU Darmstadt, Darmstadt, Germany, Proceedings of ICALEPCS 2011, Grenoble, France, 2011.

[12] LAI V., DIESSEL O., *ICAP-I: A reusable interface for the internal reconfiguration of Xilinx FPGAs*, International Conference on Field-Programmable Technology, p. 357 - 362, 2009.

[13] MARIN Raul, LEON German, WIRZ Raul, SALES Jorge and CLAVER Jose M., *Remote Programming of Network Robots Within the UJI Industrial Robotics Telelaboratory: FPGA Vision and SNRP Network Protocol*, IEEE Transactions on Industrial Electronics, vol. 56, no. 12, 2009.

[14] NORTON Andy, MULLINS Jeff, MINCHER Dick, *Blade Management Controller Rides FPGA Embedded Processor*, CloudShield Technologies Inc., Xcell Journal: Solutions for a Programmable World, vol. 69, p. 14 – 17, 2009.

[15] POPESCU Dorin, SELISTEANU Dan, DINULESCU Ionut and LAZAR Daniela, *Development of online experiments for a mobile robot via Internet*, ICAI'08, 978-960-6766-77-0, World Scientific and Engineering Academy and Society (WSEAS), Bucharest, Romania, p. 519-524, 2008.

[16] POPESCU Dorin, SELISTEANU Dan, *Real Robot Remote Control versus Simulations*, University of Applied Sciences, FH Ravensburg-Weingarten, Germany.

[17] PRASHANT Kumar Tripathi, JIDHU Mohan M., GANGADHARAN K.V., *Design and Implementation of Web based Remote Laboratory for Engineering Education*, Department of Mechanical Engineering, National Institute of Technology Karnataka, Surathkal, Mangalore, India, International Journal of Engineering and Technology vol. 2, no. 2, ISSN 2049-3444, 2012.

[18] RoboCup SSL, *Laws of the RoboCup Small Size League 2012*, http://small-size.informatik.uni-bremen.de/_media/rules:ssl-rules-2012.pdf, p. 16, accessed online June 27, 2012.

[19] RUIZ-DEL-SOLAR Javier, CHOWN Eric, PLOEGER Paul G. et al., *RoboCup 2010: Robot Soccer World Cup XIV*, Lecture Notes in Computer Science, vol. 6556, 1st Edition, 2011.

[20] Samtec, *VITA 57 FMC*, http://www.samtec.com/search/vita57fmc.aspx, accessed online 2012.

[21] SEELAM Raj, *I/O Design Flexibility with the FPGA Mezzanine Card (FMC)*, Xilinx White Paper: Virtex-6 and Spartan-6 FPGAs, 2009.

[22] TAELMAN, Pieter, *Ontwikkeling van een draadloos herconfigureerbaar mobiel pedagogisch roboticaplatform*, unpublished, 2012.

[23] TAN Heng, DEMARA Ronald F., EJNIOUI Abdel, SATTLER Jason D., *Complexity and Performance Evaluation of Two Partial Reconfiguration Interfaces on FPGAs: a Case Study*, World Congress in Computer Science, Computer Engineering and Applied Computing, Las Vegas, Nevada, USA, 2006.

[24] VERBELEN Yannick., BRAEKEN An., KUBERA Serge., TOUHAFI Abdellah., MENTENS Nele., and VLIEGEN Jo., *Implementation of a Server Architecture for Secure Reconfiguration of Embedded Systems*, ARPN Journal of Systems and Software, vol.1, no. 4, 2011.

[25] Xilinx, *Xilinx Spartan 6 Overview*, http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf, accessed online June 27, 2012.

[26] Xilinx, *Spartan 6 FPGA SP601 Evaluation Kit*, http://www.xilinx.com/products/boards-and-kits/EK-S6-SP601G.htm, accessed online June 29, 2012.

AUTHORS

Yannick Verbelen is with the Erasmus University College Brussels and the Vrije Universiteit Brussel and prepares a PhD. His research focuses on embedded systems, energy harvesting and mechatronics (e-mail: yannick.verbelen@ehb.be).

Pieter Taeman graduated at the Erasmus University College Brussels as M.Sc (e-mail: pieter.taelman@gmail.com).

An Braeken graduated with a PhD in Mathematics and currently teaches at the Erasmus University College Brussels. Her research focuses on cryptography and parallel computing (e-mail: an.braeken@ehb.be).

Abdellah Touhafi is with the Erasmus University College Brussels and the Vrije Universiteit Brussel and teaches advanced electronics and embedded design. (e-mail: abdellah.touhafi@ehb.be).

Received 11 March 2013. Published as resubmitted by the authors 12 June 2013