A reconfigurable computing system for an autonomous sailboat

Jose C. Alves, Tiago M. Ramos, Nuno A. Cruz
Department of Electrical and Computer Engineering
University of Porto
Porto, Portugal
{jca,ee06254,nacruz}@fe.up.pt

Abstract

This paper presents the computing infrastructure used in an autonomous unmanned small-scale sailboat. The system is based on a FPGA and includes custom designed interfaces for the various sensors and actuators used in the sailboat. The central processing unit is a 32-bit 50 MHz RISC microprocessor implemented as a soft IP core in the FPGA. The computing system runs uClinux, a simplified version of the popular Linux operating system. The usage of a reconfigurable platform enables a quick reconfiguration of the logic circuit implemented in the FPGA, facilitating the development stage and allowing a dynamic switch among different implementations, according to the navigation requirements and environmental conditions.

Introduction

The FEUP[1] autonomous sailboat (FASt) is a small sailing yacht capable of fully autonomous navigation through a predefined set of waypoints. The boat was custom designed and built as a response to the Microtransat challenge (`www.microtransat.org`) and offers a flexible platform for various applications like data acquisition of oceanographic or atmospheric variables, wild life tracking, surveillance and also as support platforms for cooperative navigation with autonomous underwater vehicles (Curtin et al., 1993) (AUVs).

The computing system designed for FASt is based on a FPGA (Field-Programmable Gate Array), providing a flexible reconfigurable computing platform (Compton and Hauck, 2002). The system implemented in the FPGA includes a RISC 32-bit central processor surrounded by a set of custom designed peripheral digital systems that run the processes responsible for the interface with sensors and actuators. This allows the integration of almost all the custom digital electronics into a single chip, simplifying significantly the design of the control software by alleviating the processor from low-level interfacing and data processing tasks.

FPGAs

FPGAs are commercial integrated circuits that can be configured by the end user to perform any arbitrary digital system. The common configuration technology used in present FPGAs is based on SRAM and provides a virtually infinite number of re-configurations in very short times (tens to hundreds of milliseconds). Cutting-edge FPGA devices offer capacities equivalent to a few millions of logic gates, including on-chip memory blocks exceeding 10 Mbit, functional blocks optimized for signal-processing applications, gigabit transceivers and even embedded high-performance processors. This is now a mature digital technology that offers flexible and efficient platforms for targeting complex and high-performance digital systems, without incurring in the high costs and long turnaround times of silicon fabrication.

An attractive of the FPGA technology is the ability to quickly modify the digital system implemented in the chip. Different configuration files can reside in low cost off-chip low cost memories and loaded

[1] Faculdade de Engenharia da Universidade do Porto (School of Engineering of the University of Porto)

Total length (LOA)	2,50 m
Maximum width (beam)	0,67 m
Draft	1,25 m
Displacement	45 kg
Sail area	3,7 m^2

Table 1: Main dimensions of the FEUP Autonomous Sailboat - FASt

into the FPGA to configure a completely different system. Some FPGA families even allow partial reconfigurations without disturbing the rest of the chip. This is particularly interesting for applications where the processing requirements of some localized parts may vary along the time.

The FASt project

The FASt project was launched at FEUP in the beginning of 2007 to participate in the Microtransat competition (`http://www.fe.up.pt/fast`). Although the Microtransat rules allow a maximum length of 4 m and the theoretical maximum speed of a boat is proportional to the square root of its length, we decided for a 2,5 m long mono-hull. This was determined after scaling down in length and displacement some real oceanic modern sailing boats, keeping the final weight not far from the 40 kg mark in order to facilitate the launch and transportation. The design was developed with the free version of the boat design software DelftShip (`http://www.delftship.org`, former FreeShip). Main dimensions of FASt are presented in table 1.

FASt hull construction

The boat was built by a team of students and professors of FEUP, with the support of a local kayak builder (Elio Kayaks, `http://www.elio-kayaks.com`) that executed the fabrication of the parts in composite materials and gave a valuable help during all the phases of the process. The hull has been fabricated using a sandwich of carbon fibre in the outer layer, a low density honey comb core in the middle and a inner layer of fibreglass, pressed with vacuum during the cure of the epoxy resin. The hull was reinforced internally at the points subject to the major mechanical forces: the attachment of the keel, foot of mast and the points where the shrouds connect to the hull. Figure 1 illustrates some stages of the construction process.

Figure 1: The construction of FASt: (top-left to bottom-right) assembly of the frames; strip planking; the mould; filling the mould with the layers of fibre and core materials; the final boat.

Electronic system

The electronic system used in FASt is assembled with various modules, some of them custom built for this application. Figure 2 presents a block diagram with the general organization of the system and the approximate physical location inside the hull.

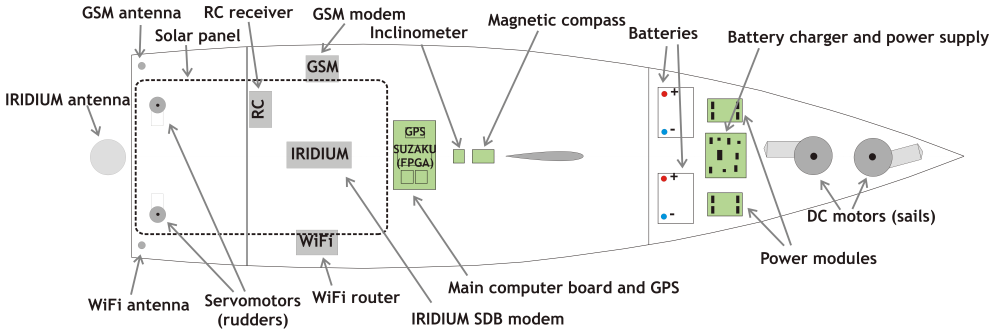

Figure 2: The organization of the FASt electronic system and its distribution inside the hull.

The onboard computer is implemented in a small FPGA-based board (Atmark-Techno, 2006). The design configured in the FPGA includes a 32-bit RISC microprocessor running at a 50 MHz maximum clock frequency (Microblaze (Xilinx, 2004)), surrounded by various custom designed dedicated circuits. The organization of the computing system is detailed in the next sections.

The communications section includes a WiFi router (LinkSys WRT54GC), GSM modem (Siemens MC35), a conventional RC receiver used in radio-commanded models. A IRIDIUM SBD modem (model 9601) will be added later for long course navigation.

Sensors include the wind vane and anemometer, boom position, inclinometer, digital compass (Honeywell HMR3300), GPS (uBlox RCB-4H), voltage monitors, ambient light sensor, interior temperature and a set of moisture sensors distributed is various locations inside the hull.

Regarding the actuators, one DC motor controls the sail's position and two standard RC servos provide independent control of the two rudders. The DC motor used to control the sail position has a gearbox that naturally locks the motor's shaft when it is unpowered. As the sail angle only needs to be adjusted to set a new course or when the wind direction changes, this feature translates in an important saving of electric energy when comparing to other combinations motor-gearbox that need power to react to the the sail sheet force.

Finally, the power generation and control section includes a 45 Wp solar panel (Solara SM160M), two 95 Wh Li-ion batteries, battery management module and a ATX power supply. This power system was included as an integrated solution provided by OceanServer (`http://www.ocean-server.com`).

The computing platform

The computer board used in FASt is a commercial system from Suzaku (SZ130 (Atmark-Techno, 2006)), built around a Xilinx FPGA, model Spartan3E S1200. The system has 32 MB of SDRAM, 8 MB of SPI flash memory, serial interface and Ethernet. A total of 86 digital I/O pins are available for the user application, directly connected to FPGA pins and distributed in edge connectors around the board. The FPGA is only partially occupied by the base project (processor and essential peripherals), leaving roughly more than 1 million equivalent logic gates available for user logic. Figure 3 depicts the organization of the SZ130 board and the reference project implemented in the FPGA that is included with the development kit.

The system runs uCLinux (`www.uclinux.org`), a version of the popular Linux operating system that has been modified for embedded applications running in processors with no memory management unit (MMU). The operating system provides an interactive console through a standard RS232 port, a structured file system, multitasking and a TCP/IP stack.

This board constitutes a convenient platform for building a digital integrated system, providing a combined hardware/software solution for a given problem. A design may include virtually any custom

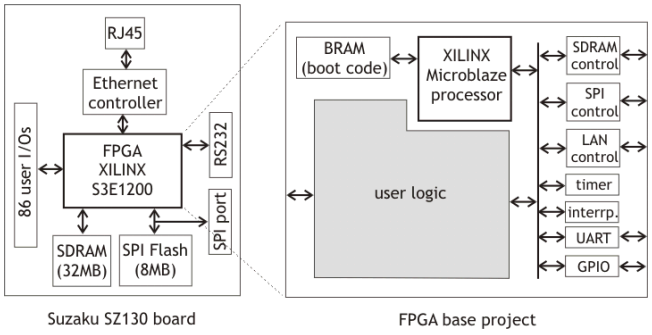

Figure 3: Organization of the Suzaku SZ130 board.

designed digital circuit that may fit into the FPGA's free resources. Combining sequential software execution in a conventional processor with custom processing by dedicated hardware modules can alleviate significantly the computing load of the central microprocessor. Besides simplifying the development of the software because various concurrent processing tasks are done transparently in hardware, this approach also permits to reduce the processor's clock frequency and thus the overall power consumption.

Software development

The software for this machine is developed in ANSI C and compiled with a customized version of `gcc`. The programs running on the uCLinux operating system can make use of the common Linux standard libraries, including TCP/IP communication, file I/O and file system management. During development, evaluation and debug, the support of the uCLinux operating system is very convenient because it eases the implementation of network communication processes, file management and multitasking of different program's parts.

Hardware development

The development of the digital system implemented in the FPGA is done with the EDK/ISE software tools from XILINX. The XILINX EDK (Embedded Development Kit) is a design tool that builds a combined hardware/software design, targeted to a XILINX FPGA-based board. The hardware part is assembled with pre-designed parametrizable modules (microprocessor, SDRAM interface, USART, etc.) and user designed components. The software component is built as a C program that is later embedded with the FPGA configuration data. The ISE tool suite performs the complete digital design flow for XILINX FPGAs and translates the digital modules produced by EDK into the final bitstream used to configure the FPGA. A user's circuit is specified in standard hardware description languages (Verilog or VHDL) and integrated into the system's top-level description.

FPGA reconfiguration

When the system is powered up, the FPGA is configured with the data stored in the flash memory. Once the configuration is completed, the I/O buffers associated to the FPGA pins are enabled and the system implemented in the FPGA starts running. The Microblaze microprocessor then runs the code stored in the FPGA internal memories. To start uCLinux, a boot loader is executed to load into a RAM disk the file system image stored into the flash and then boot the system Linux kernel.

The reconfiguration of the FPGA can be done under control of software from a regular file stored in the local file system or in a http server within range. The running software can thus choose a bitstream from a batch of pre-built configurations, copy it to the flash memory and issue a `reboot` command to restart the system with a different FPGA configuration. This unique feature of FPGA-based systems allows to change the digital circuit played by the FPGA, according to different processing needs that may be driven by several factors (eg. the availability of energy or environmental conditions).

The FASt computing system

The FASt computing system is implemented in the FPGA of the Suzaku board. Besides the central Microblaze processor, the system includes various dedicated controllers for interfacing the sensors and actuators used in the sailboat, some of them with custom computing capability. Figure 4 presents a simplified view of the organization of the system.

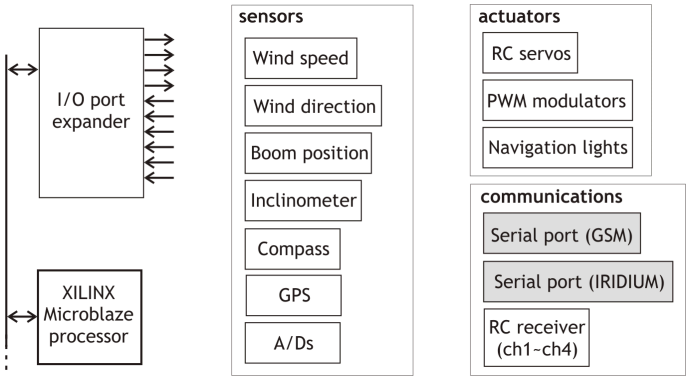

Figure 4: Simplified view of the organization of the FASt computing system implemented in the FPGA. All the sensors and actuators are accessed by the microprocessor via dedicated interfaces. The two shaded blocks are standard serial port controllers modules from the EDK library.

The global strategy adopted during the design of this system was to create a set of autonomous interfaces capable of delivering to the software the data retrieved from the sensors in a format easy to use by the a software application. Besides the implementation of each sensor's specific low level interface protocol, this includes parsing messages, filtering, and also units conversion.

The access to the peripherals from the Microblaze is done through a port expander. This module uses only one pair of 32-bit memory-mapped bidirectional ports and makes available for the rest of the circuit a set of 16 output ports and 16 inputs ports (each 32 bits wide). This is implemented as a finite-state machine that interprets a set of commands issued by the microprocessor as memory writes and reads.

Sensors

The wind direction interface reads the AS5040 sensor sampled at 50 Hz and averaged using a sliding window of 64 samples (the boom position sensor is built with the same chip and is interfaced by another instance of this module). The interface with the wind speed sensor outputs the number of clock periods during one revolution of the cup rotor, sampled at 10 Hz and also averaged by a 64 tap mean filter.

The inclinometer reads the PWM outputs of the 2-axis accelerometer and returns two integers that are proportional to the X and Y acceleration. The roll angle in computed in software.

The magnetic compass provides data packets with the heading, roll and pitch angles in ASCII format. Although the ASCII format is easy to interface with a software parser, it is also important to have this data directly available to the hardware system in numeric digital format. This allows for future integration of hardware control processes that directly link the heading information to the controllers of the steering servos. The interface with the compass implements a parser of the ASCII messages and extracts the relevant data in binary format. The GPS interface is done in a similar way, extracting the relevant data (lat/lon, speed, course and status) from the binary protocol output (uBlox UBX protocol (u-blox, 2008)).

The interface with the radio-control receiver is done by 4 instances of the same module, one for each channel of the receiver. The standard control signal used in RC receivers and servos is a 50 Hz digital signal, where the high time defines the position of the servo (ranging from 0,8 ms to 2,2 ms). Each receiver module measures the high time of the corresponding channel and converts it to a two's complement 10 bit integer.

4-input LUTs	7.314	(42%)
Flip-flops	3.997	(23%)
Occupied slices	4.730	(54%)
LUTs used as route-thru	324	(1,8%)
LUTs used as SRAM	256	(1,5%)
LUTs used as shift-registers	796	(4,6%)
Block RAMs	14	(50%)
Dedicated multipliers	8	(28%)
Equivalent gate count	1.078.257	

Table 2: Summary of the FPGA Spartan 3E 1200 occupation. LUT stands for Look-up table and is the elementary configurable block in Spartan 3E FPGA: a 16 bit SRAM implementing any 4-input combinational logic function

The system includes two A/D converters with 6 analogue inputs, plus two additional channels that monitor the 3,3 V supply and the chip temperature (AD7795). This is interfaced by a controller that implements the SPI protocol and provides to the computing system a simpler interface to select A/D channels and read conversion results.

Actuators

The two servos controlling the rudders are driven by independent hardware modules. A hardware multiplexer selects the source of data that is routed to these servos: this can be the output of the RC channel 1 (to use the left-right stick) or the data sent by the software application running on the processor.

The sail sheet of both sails is commanded by the same DC motor equipped with a multi-turn potentiometer for position feedback. This motor is controlled by a PWM modulator that drives one power bridge with MOSFET power transistors. The PWM control module includes a low-pass filter applied to the input data, to smooth the accelerations and limit the current draw.

Implementation

Current design, as represented in figure 4, uses less than 50% of the XC3S1200E FPGA resources and corresponds approximately to 1 million equivalent logic gates. All the modules support the maximum clock frequency of 50 MHz allowed for the Microblaze processor, although most of them are allowed to run with much lower clock frequencies. Table 2 summarizes the occupation of the FPGA resources.

Power consumption issues

Electric power consumption is one of the great concerns in an autonomous sailboat. For a small boat, the reasonable sources of electric energy for long term navigation are photovoltaic panels and wind turbines. Best solution would be a combination of both but, as far as we know, the commercially available wind generators are too large and heavy for our sailboat. In both cases, the availability of energy always depends on the weather conditions which still have a high degree of uncertainty. For this reason, the electronic system must consume the lowest possible energy and whenever possible adapt its behaviour to the power budget available at each stage.

According to our first estimates, the computing system will account for more than 50% of the total energy consumed by the system, assuming a continuous operation with the maximum power consumption measured for the present configuration. This represents aproximately 500 mA for the 3,3 V supply (1,65 W, including 100 mW for the digital compass and the wind sensors) running the microprocessor at the maximum frequency of 50 MHz with the Ethernet port enabled. Disconecting the Ethernet cable puts the Ethernet controller in power down mode and reduces the power consumption by approximatley 200 mW; lowering the clock frequency to 25 MHz saves more 100 mW.

Operating modes

The FASt computing system supports three different modes of operation. The selection of the control mode is done by the throttle level of the radio-command: middle for radio-commanded mode, front for WiFi semi-autonomous control and rear for fully autonomous sailing. When the radio transmitter is switched off, the boat enters automatically the fully autonomous mode, and checks for the presence of the RC radio signal at periodic intervals (this is currently set to 1 minute).

In radio-commanded control mode, the control is totally done through the radio-command, using only two control sticks: left-right turn and sheet control. The semi-autonomous interprets commands sent from a control program running in a PC in the range of the WiFi signal (for example, to setup a course, tack or jibe). In the fully autonomous mode, FASt starts navigation to reach a set of pre-programmed waypoints.

Conclusions

This paper presented a FPGA-based reconfigurable electronic system used in an autonomous sailing boat. The FPGA implements a RISC microprocessor surrounded by several custom designed peripherals that interface with the sensors and actuators used in the sailboat. The hardware reconfigurability feature of the FPGA enables a short design iteration and fast in-circuit reconfigurations of the running hardware. This may be exploited for reducing the energy consumption by adapting the control and computing logic circuits to the specific requirements of navigation, for given wind and sea conditions.

Acknowledgements

The authors would like to thank the Department of Electrical and Computing Engineering (DEEC) of the School of Engineering of the University of Porto, Portugal (FEUP), for the financial support of this project.

References

Atmark-Techno (2006). SUZAKU-SZ130-U00, Hardware Manual (English version) V1.0.2, [online]. Atmark-Techno [cited 30 april 2008]. Available from World Wide Web: <http://www.atmark-techno.com>.

Compton, K. and Hauck, S. (2002). Reconfigurable computing: A survey of systems and software. *ACM Computing Surveys*, 34(2):171–210.

Curtin, T., Bellingham, J., Catapovic, J., and Webb, D. (1993). Autonomous oceanographic sampling networks. *Oceanography*, 6(3):86–94.

u-blox (2008). u-blox 5 NMEA, UBX Protocol Specification, Public Release [online]. u-blox AG, Switzerland [cited 30 april 2008]. Available from World Wide Web: <http://www.u-blox.com/customersupport/gps.g5/u-blox5_Protocol_Specifications(GPS.G5-X-07036).pdf>.

Xilinx (2004). Microblaze Processor Reference Guide, EDK v6.2 [online]. Xilinx [cited 30 april 2008]. Available from World Wide Web: <http://www.xilinx.com/support/documentation/ip_documentation/microblaze.pdf>.