18

Dynamic Wi-Fi Reconfigurable FPGA Based Platform for Intelligent Traffic Systems

Mihai Hulea, George Dan Moiş and Silviu Folea
Technical University of Cluj-Napoca, Romania

1. Introduction

This chapter proposes a software and hardware platform based on a FPGA board to which a Wi-Fi communication device has been added in order to make remote wireless reconfiguration possible. This feature introduces a high level of flexibility allowing the development of applications which can quickly adapt to changes in environmental conditions and which can react to unexpected events with high speed. The capabilities introduced by wireless technology and reconfigurable systems are important in road traffic control systems, which are characterized by continuous parameter variation and unexpected event and incident occurrence.

Intelligent Transportation System (ITS) is the term commonly used to describe the employment of electronic devices for the management of road traffic and other types of transportation networks for improving decision making by operators and users. There are many new available technologies which can be used for increasing the efficiency of road systems and many cities are introducing ITS models as pilot schemes to test their effectiveness.

Reconfigurable hardware offers many benefits and this led to its use in the ITS field also. Traffic light systems implemented on FPGAs were developed and examples can be found in the literature (El-Medany & Hussain, 2007; Zhenggang et al., 2009). This research led to the conclusion that FPGA based devices can be a good solution for this type of systems conferring them scalability, adaptability and stability, and increasing their efficiency (Zhenggang et al., 2009). FPGAs had also been used in vehicle-to-vehicle communication (V2V), the authors of (Sander et al., 2009) presenting a V2V communication system based on this technology and special mechanisms that exploit its benefits. The result consists of a modular and flexible framework for software routines which in the same time supports critical tasks with special hardware accelerators. The modularity achieved with the help of FPGA technology allows the system to react to changes in the environmental conditions or in the user demands.

2. Background

2.1 FPGA reconfigurable systems

FPGA is the acronym for Field Programmable Gate Array and it represents a silicon chip which has an internal scheme that can be “changed”. Together with the class of

programmable logic devices (PLDs), without a clearly defined distinction among them, they form the programmable Application-Specific Integrated Circuit (ASIC) group of devices (Smith, 1997).

A FPGA is a device that consists of an array of basic logic cells that can be configured after fabrication using a certain programming technology. The logic cells are interconnected by wires and switch boxes with each other and with special input/output blocks (Mitra et al. 1998). Other important components include, but are not limited to multipliers, embedded block RAM and digital clock managers. The configurable logic blocks, or logic cells, are made up of two basic components: flip-flops and look-up tables (LUTs). The LUT is a storage block having the capacity of one bit that acts as a programmable gate. The input represents the address lines of the storing element and the output represents the value contained at this address. The input and output data can be synchronized with the clock signals. When programming the FPGA, one actually specifies the logic function of each basic logic cell and configures the connections represented by the switch boxes within the device. FPGA devices can be divided in two main classes: one time programmable FPGAs which are based on an antifuse approach and reconfigurable FPGAs which usually employ SRAM cells (Lyke, 2002). The SRAM cells contained in the device will not keep the stored information when the supply power is turned off and this way the configuration will be lost. Depending on its generation, the FPGA allows static or dynamic reconfiguration. In the case of static reconfiguration, the FPGA is programmed in the compiling phase and it cannot be reconfigured during operation, while in the second case, the FPGA can be reconfigured at any point in the lifetime of the application. Furthermore, dynamic reconfiguration can be total, when the entire device is reconfigured, or partial, when only a part of an operating FPGA is reconfigured.

By using the hardware to its maximum capabilities and due to their truly parallel nature, FPGAs are able to assure better performance as compared to conventional processors (National Instruments[1], 2011). They can be used in digital signal processing (DSP) systems where they can provide more complete solutions than the ones represented by traditional DSP implementations (programmable digital signal processors or ASICs) (Tessier & Burleson, 2001). Despite being slower than ASICs, the hardwired circuits realized by FPGAs have a speed advantage of several orders of magnitude over the software solutions on general purpose processors (Reis & Jess, 2004**)**. Other applications which employ FPGA devices include space and defence systems (Tessier & Burleson, 2001; Altera, 2011), prototyping systems (Banovic, 2005), medical systems (Altera, 2010) and many other relatively new developed markets such as language recognition, bioinformatics (HPCWire, 2009), cryptography (Prasanna & Dandalis, 2007) etc.

The capacity of reprogramming the hardware during operation allowed programmable logic to take over a central role in digital systems. The most important feature that results from the ability to reconfigure digital circuits is flexibility (Compton & Hauck, 2002). Among other things, it provides the possibility that a system might adapt to changing operating contexts or to unforeseen events (Lyke, 2002) in order to be able to fulfil its originally assigned tasks. Although this important feature, the flexibility of "on-field" re-programming without going through re-fabrication with a new design (Xilinx, 2010), along with the FPGA based systems' low-cost and short time-to-market come at a considerable price, namely significant area overhead, increased delay and power consumption (Kuon et al., 2007), this did not affect their development and large scale usage.

Systems that make use of user-reprogrammable logic elements such as FPGAs were studied (Patel & Moallem, 2010; Kalte, 2002) being well suited for applications with high

dependability (Straka & Kotasek, 2008); Bolchini, et al., 2007). Reconfigurability allows the implementation of mechanisms that improve overall system performance and dependability: in the field hardware upgrades, runtime reconfiguration, adaptive hardware algorithms, continuous service applications and system adaptation to unexpected events and environmental conditions.

The utilization of FPGAs in applications where the reach to the target device is impractical and difficult through wires (e.g. the system is not stationary, a portable embedded system) (Andretzky, 2005) lead to the need for the wireless configuration of FPGAs. In the literature we can find several implementations for the configuration process (Maye, 2005; Adly et al. 2010). The work in (Maye, 2005) achieves the wireless configuration of FPGA based components of a Modular Robot through Bluetooth. This is possible because each one of the modules that make up the system contains a FPGA board and a Bluetooth board. The authors in (Adly et al. 2010) developed a flash memory configuration controller for configuring a FPGA wirelessly from a flash memory. The design of a programmable chip working as a configuration controller which receives configuration bits through a wireless link, stores them into a flash memory and afterwards recalls them to configure the FPGA was developed in this case.

2.2 LabVIEW development platform

LabVIEW, the short name for Laboratory Virtual Instrument Engineering Workbench, is a graphically based programming language developed by National Instruments (Bitter et al., 2006). An appropriate description is given by Simon Hogg: "LabVIEW is a highly productive development environment for creating custom applications that interact with real-world data or signals in fields such as science and engineering" (National Instruments[1], 2010). LabVIEW, as a programming language, is a powerful tool that is ideal for test and measurement (T&M), automation, instrument control, data acquisition, and data analysis applications (Bitter et al., 2006).

The LabVIEW software development environment consists of several valuable components such as G Programming, Hardware Support, Analysis and Technical Code Libraries, UI Components and Reporting Tools and Models of Computation, which reduce the amount of time and people needed for project completion (National Instruments[1], 2010). It provides a number of add-on modules for extending the graphical development platform to target a wide range of hardware devices. One of these add-ons is the NI LabVIEW FPGA Module which makes custom measurement and hardware control possible without prior knowledge of low-level hardware description languages or board-level design. It uses LabVIEW VIs for defining the FPGA logic instead of low-level languages such as very high speed integrated circuit hardware description language (VHDL) (National Instruments, 2007). By using this module, applications that employ field-programmable gate arrays on NI reconfigurable I/O (RIO) hardware can be developed (National Instruments[2], 2011). The parallelism and data flow in LabVIEW are very similar to the ones in a FPGA implementation, therefore running LabVIEW code on FPGAs leads to true, simultaneous, parallel processing (National Instruments, 2007; National Instruments[2], 2011).

The LabVIEW FPGA VIs are created on a computer and afterwards compiled and run on reconfigurable hardware. Because the compiling time can range from minutes to hours depending on design complexity, LabVIEW provides a FPGA Device Emulator for

executing the VI on the Windows development computer for verification purposes. While the typical FPGA design flow includes program creation, translating design files (design entry), synthesizing and mapping design elements to device resources, placing and routing design resources, performing timing analysis and generating programming file (Altera, 2009), LabVIEW offers an intuitive and simplified solution for this action. The development process for creating FPGA applications in LabVIEW consists of: the creation of the FPGA VI, emulation on PC for testing if necessary, compilation to FPGA and the creation of host VIs for communicating with the reconfigurable hardware (National Instruments², 2010). Actually, the configuration of the FPGA device's operation is performed by programming in LabVIEW. Another useful characteristic is represented by the fact that when a specific FPGA device is targeted, LabVIEW displays only the functions available in the FPGA (National Instruments², 2010).

3. Wi-Fi FPGA based reconfigurable platform for ITS

3.1 General system overview

The general structure of the proposed platform is presented in the figure bellow. As it can be seen, the platform is composed of the two main components: the FPGA board at which level the application's logic algorithms are implemented and the Tag4M board which has Wi-Fi communication capabilities and which executes the reconfiguration and communication algorithms needed for making remote access possible. The presented platform can be installed on vehicles or on the road traffic infrastructure equipment (like traffic lights and dynamic signs). The presented platform can communicate with a Configuration Server which is responsible for performing the reconfiguration operations in order to set-up new applications at the FPGA platform level. It can also communicate with Traffic Servers in order to report traffic data or to receive commands.

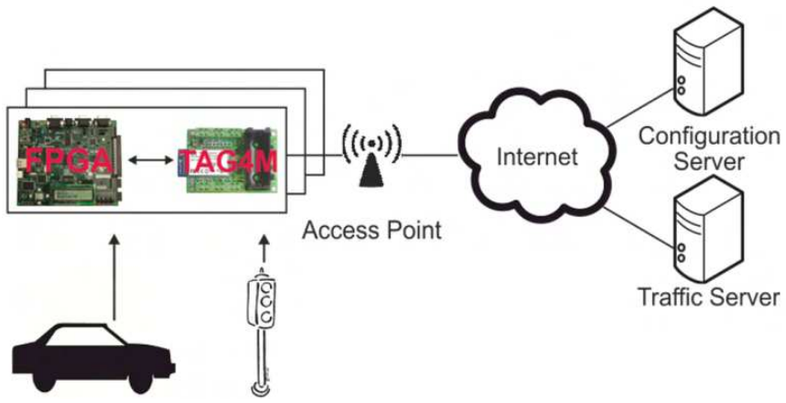


Fig. 1. General system structure

By attaching the Tag4M Wi-Fi extension to the FPGA board a versatile computing device with communication capabilities which has numerous applications in the ITS field is obtained. This provides the FPGA device not only with the capacity to communicate over the Internet with remote servers or other devices but also the possibility to dynamically change its functionality depending on the current application's requirements.

This flexibility is particularly important in dynamic environments represented by road traffic systems. A FPGA platform installed in a vehicle can play multiple roles and the same hardware can be used in multiple traffic scenarios without the need of having direct access to the device installed in the field or on the vehicle. The device can be easily accessed by using remote servers to which it communicates and which perform the configuration procedures. While the vehicle is entering the highway, a configuration server can load into the vehicle device an application for automatic toll collection eliminating the need for the vehicles to slow down while passing the checkpoints. When the vehicle exits the highway the previously loaded toll collection program can be replaced with a road traffic control module which makes the vehicle capable of communicating with other vehicles or with infrastructure equipments in order to provide peers information for achieving traffic fluidization. At a later time, when the vehicle is entering a parking zone the on-board FPGA platform can be loaded with a parking advisory system in order to guide the driver to the closest parking place and to keep track of the parking time.
The same FPGA equipment can be used as the infrastructure controller for managing the traffic lights and other dynamic traffic signs. By using its Wi-Fi communication capabilities it is easy to integrate these pieces of equipment into the distributed road traffic monitoring and control system.

3.2 Wi-Fi Tag4M device

The Tag4M device which is used for FPGA reconfiguration is presented in Figure 2 (scale 1:1). This is a Wi-Fi RFID (Radio-Frequency Identification) active device with measurement capabilities. By attaching sensors to its I/O terminal blocks in a similar manner as for a data acquisition device, the user can build wireless proof-of-concept sensor solutions for a wide range of applications. The system has the advantage of reduced dimensions (4.7 cm x 7.0 cm) and of a limited weight of 50 g and can run on battery power, making it a portable solution. It is a complete Wi-Fi and networking solution, incorporating a 32-bit CPU, a memory unit, an eCos real-time operating system and a UDP or TCP/IP stack. Other included components are the analogical sensor interface, the power management unit, the hardware cryptographic accelerator and the real time clock (G2 Microsystems[1], 2008).

Fig. 2. Tag4M Data Acquisition System

The hardware architecture of the device is presented in Figure 3. In this version, 8 analogical and 10 digital channels and 2 serial ports are available. A general interface was implemented to allow the acquisitions from different sources like: 3-axis accelerations, 3-axis gyroscopes, magnetic sensors and more other parameters, which can be combined for obtaining a wide range of configurations (G2 Microsystems[3], 2008).

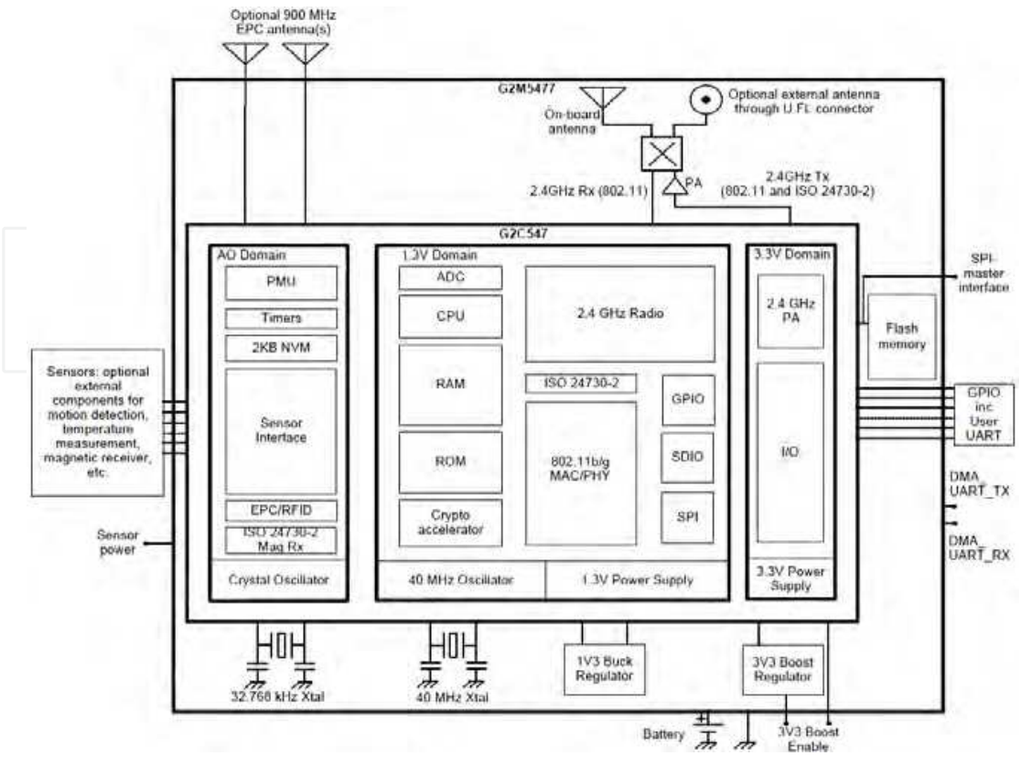


Fig. 3. The Hardware Architecture of the Tag4M Data Acquisition System

3.3 Wi-Fi Tag4M software stack

In this section, the software stack installed on the Wi-Fi Tag4M is described. The device is powered by a G2C547 module from G2 Microsystem (bought by Roving Networks). It has an eCos real-time operating system installed on ROM which boots up when the system is powered on (G2 Microsystems², 2008).

The eCos operating system runtime provides full preemptibility, low latency, synchronization primitives, scheduling policies and interrupt handling mechanisms which are useful in real-time applications. The embedded applications can also take advantage of other eCos functionalities such as device drivers, memory management mechanisms, exception handling, math libraries and many more others. The OS also includes all the necessary tools for developers, namely: build tools, compilers, assemblers, linkers, debugger and simulators.

Programming microcontroller devices raise a unique set of challenges in general and the Tag4M device is not an exception. First, there are several points of failure. The Wi-Fi Tag4M device itself may fail. There can be transient or permanent communication errors between connected devices and the microcontroller device like overlapping signals, accidental disconnection etc. The C Application on the Wi-Fi Tag4M device itself may fail, but this is unlikely to happen. A communication failure between the Wi-Fi Tag4M device and the application server could also occur. An application for the Tag4M device needs to handle these multiple points of failure and hide its complexity from the end user.

Secondly, device programming requires much more debugging and testing iterations than normal application development because of the direct low-level interaction with the underlying hardware.

The application needs to provide robust error handling capabilities. Automatic error correction codes must be incorporated for correcting several types of transient errors. This minimizes maintenance requirements and user intervention.

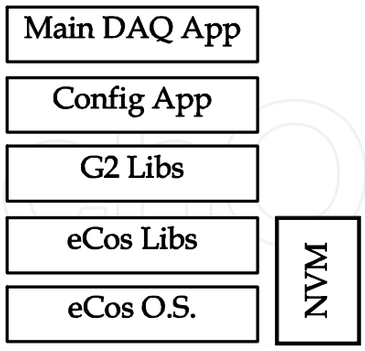


Fig. 4. Tag4M Device Software Stack

As it is shown in Figure 4, the Wi-Fi Tag4M device software stack is composed of the eCos operating system, eCos libraries for accessing low level services, the G2 library for accessing sensors and measurement services and two user applications (Main DAQ App and Config App) which are stored in the non-volatile memory (NVM).

3.4 FPGA configuration process

The prototype architecture represented by Figure 5 uses a Spartan-3E Starter Kit board which can be configured in serial mode (Xilinx, 1998). The upgrade of the design in the field over a wireless network is made possible by using the Tag4M as a microcontroller and as a Wi-Fi receiver. The signals used are the DIO lines 0 to 4 of the Tag4M and the FPGA IO lines (-PROGRAMM, CCLK, DIN, -INT and DONE).

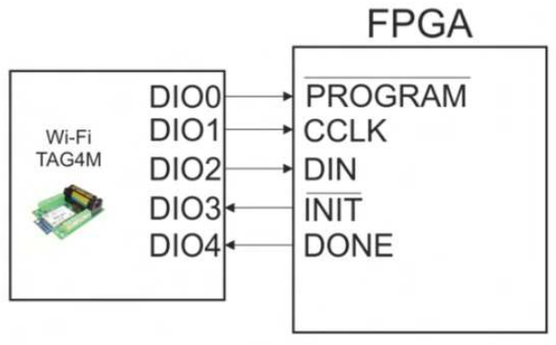


Fig. 5. The connection between the FPGA and the Tag4M device

Spartan-3E

The Spartan-3E Starter Kit Board we used provides a powerful development platform for designs targeting the Spartan 3E FPGA produced by Xilinx. It features the Xilinx XC3S500E FPGA having 500K gates. Some of the Spartan-3E FPGA family features include: parallel NOR Flash configuration, SPI Serial Flash Configuration, Master and Slave Serial configuration, etc. The FPGA chip on the board can provide up to 232 user-I/O pins and over 10000 logic cells in a 320-pin FBGA package (Xilinx, 2006).

Configuration Process

Serial mode was chosen for configuration because of the reduced number of interface signals needed (a minimum of four: DIN, CCLK, -PROGRAM and -INIT) and because of the relative ease of implementation. The microcontroller that initiates the configuration process and which sends configuration data, the Tag4M, acts as the "master" and the Spartan-3E board plays the role of the "slave" device, being the one receiving data.
A brief description of the configuration process is given in the following paragraphs. The four steps that make up the configuration flow are: the clearing of the configuration memory, initialization, configuration and start-up.

Clearing Configuration Memory

The first step is performed either automatically at system power-up or by applying a Low-level pulse to the -PROGRAM input. This is useful when the "master" initiates the configuration at any time during device operation. When -PROGRAM changes to High, the last clearing operation takes place. After this step, in the initialization phase, -INIT goes High to confirm that the memory is cleared. It is not recommended for -PROGRAM to be held low for more than 500µs and it should not be used for delaying the configuration process. The entrance in the configuration step can be delayed by holding the -INIT signal Low.

Initialization

At this point, the selected configuration mode is identified by sampling the mode pins which indicate the slave serial mode (M[g_SR_2:0]=<1:1:1>). Now, the device can pass to the configuration step. By holding -INIT low, the entry to the configuration step can be delayed.

Configuration

After -INIT goes High, the controller begins to send data on the DIN line from the memory block along with clock pulses, this action taking place in the configuration phase. A small pause having the duration between 55 and 275µs is needed before sending clock signals and data bits. After the last bits for configuration are sent, some additional clock cycles are needed, and then the Start-Up step takes place and the Spartan device starts normal functioning (Xilinx, 2009).

Start-Up

The events taking place during Start-Up are: the DONE pin goes High, the I/Os go active and the Global Set/Reset (GSR) Net is released. The DONE pin is optional because the "master" knows the number of configuration bits it has to send, but it can be used as an indicator for problems with configuration: its failure to go High means that an error had occurred (Xilinx, 1998). In this phase, several additional clock cycles must be provided, for synchronizing the events that take place at this time.
The following figure presents the general configuration process of the Spartan-3 devices (Xilinx, 2009).
When a FPGA board installed on a vehicle or on the road side needs to be configured, a *Start Programming* message is sent to the associated Tag4M device, which in turn initializes the configuration procedure for the FPGA device. As a result of this first set of messages exchanged, the Configuration Server receives an ACK message if the FPGA board is ready for receiving the new program or ERR in case an error occurred. In the case in which the ACK message was received, the Configuration Server will decompose the program in small packages and will send them sequentially over Wi-Fi to the Tag4M device, and from there to the FPGA board. In the case in which a package is lost (and an ERR message received), the

respective package will be resent. The programming procedure will end with a *Stop Programming* command.

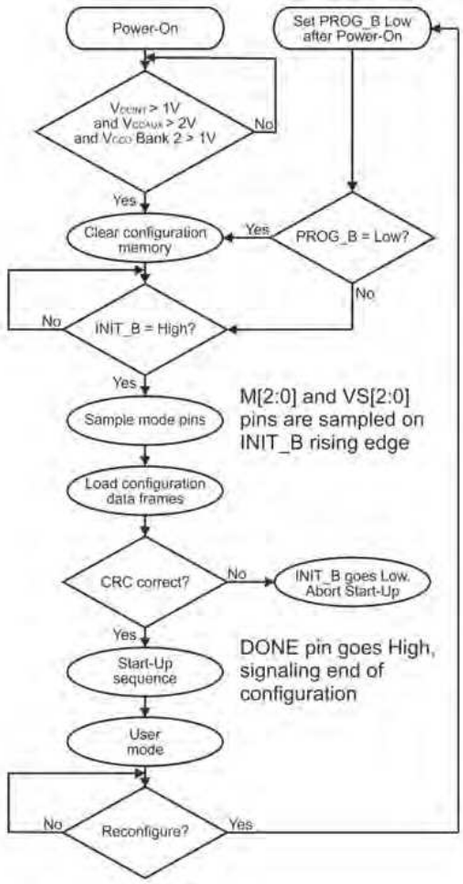


Fig. 6. General configuration process (Xilinx, 2009)

The messages exchanged between the entities involved in the FPGA configuration process is presented in the following figure.

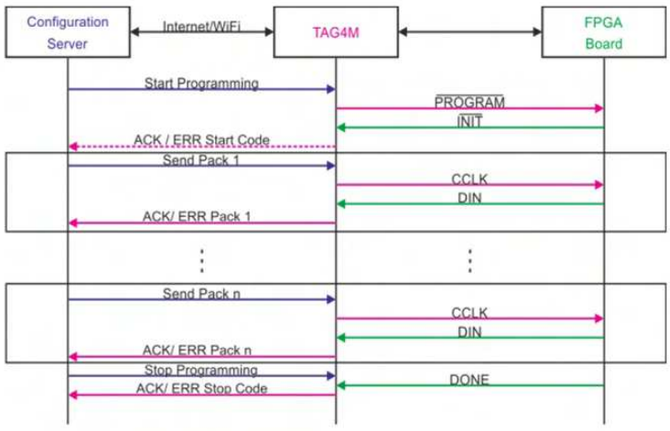


Fig. 7. Description of the communication protocol

The application installed on the Tag4M device is responsible for performing the FPGA programming procedure. The application has an event-driven architecture with a main loop (presented in Table 1) in which events are handled.

```
activateAnalogDigitalLines();
startDHCP();
whiel(1){
        evntID = getNextEvent();
        case START_PROGRAMMING:
                startFpgaProgramming();
                postEvent(REPORTING_EVENT);
                break;
        case PACKAGE_RECEIVED:
                fpgaPackageSend();
                postEvent(REPORTING_EVENT);
                break;
        case STOP_PROGRAMMING:
                fpgaStopProgramming();
                postEvent(REPORTING_EVENT);
                break;
        case REPORTING_EVENT:
                sendReport();
                break;
        case FIND_ACCESSPOINT_EVENT:
                associate();
                break;
        case DHCP_COMPLEAT:
                dhcpConfigure();
                initializeMeasureTimer(frequency);
        case DHCP_RENEW:
                dhcpConfigure();
                break;
        case WATCHDOG_RESET:
                restartApplication();
}
```

Table 1. Events executed in the main application thread

Without entering into implementation details, the basic activities implemented at Tag4M level are presented in this paragraph. When the application is started, the first action performed is the search for an available access point (AP). If one AP is found, the application will acquire a dynamic IP address and then will wait for messages from the Configuration Server. When a new message is received by the Tag4M device, it will be decoded, and depending of its content a new event will be fired. For example, when a *Start Programming* message is received this will result in firing a START_PROGRAMMING event which will be detected and handled by the event loop described in the table above. As described in the *Configuration* section, the FPGA configuration bits are sent on the rising edge of the CCLK signal (the clock is generated on the DIO1 line) using the FPGA DIN line (which is connected to the Tag4M DIO2 line). The bit sending process is implemented in the *fpgaPackageSend()* function.

4. Applications of the Wi-Fi FPGA platform for ITS

4.1 Detecting vehicle location using Wi-Fi RSSI

The simplicity and cost efficiency of the Received Signal Strength Indicator (RSSI) based localization makes it the best candidate for specific applications like tracking systems and dynamic networks where precision is not crucial.

In the following paragraphs an application implemented in LabVIEW for vehicle localisation using a FPGA and a Tag4M device is presented.

A "scan" operation which finds all access points and reads the corresponding RSSI values is implemented at Tag4M level. These values are packed and sent to the FPGA using a serial communication link where the localisation algorithm is implemented.

Table 2 presents a scan operation result. In this case four APs are detected. The corresponding RSSI values (in dBm) measured by the Tag4M are reported for every AP.

```
> scan
>
RSSI Scan Results: 4
  Time        SSID        Ch Ad-Hoc    Sec WPS    MAC Address     ERP WMM  R
SSI Supported Rates (Mbit/s, *Mandatory)        Crypto Suites       CW Max/Min

AP_SCAN,Hawk,-57 dBm,
AP_SCAN,Helicopter,-50 dBm,
AP_SCAN,tag4m,-68 dBm,
AP_SCAN,witagserver,-45 dBm,
```

Table 2. A sequence of results for the "scan" operation executed on the Tag4M device

The program for reading the RSSI values from the Tag4M device was implemented in LabVIEW2010. Three APs (which are placed on a crossroad) named witagserver, Hawk and Tag4M, are displayed by the application in this case.

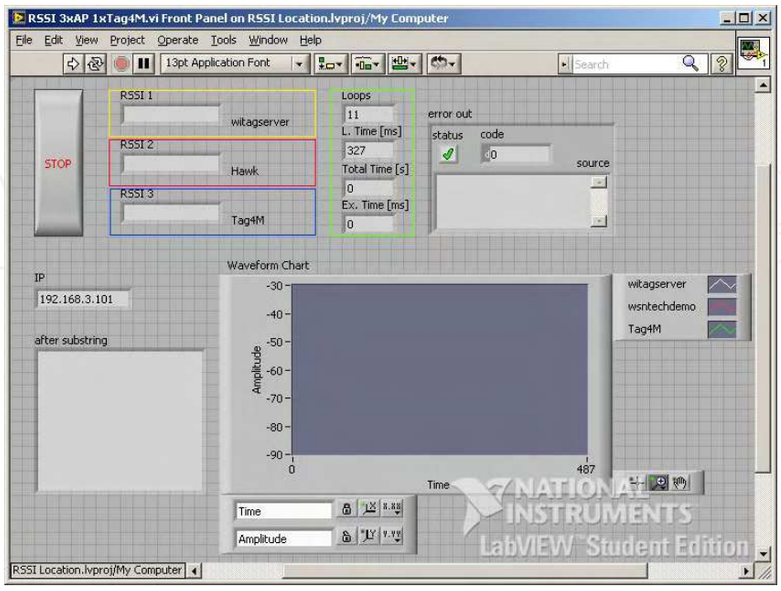


Fig. 8. Front Panel of the program for reading RSSI values

The data received by the program are parsed and only data from APs which we are interested in are processed. The RSSI values are converted to distance values using formula (1).
The value of the received signal strength is a function of the transmitted power and the distance between the sender and the receiver. The received signal strength will decrease when the distance increases as the following equation shows (Aamodt, 2006).

$$RSSI = -(10n \cdot \log_{10} d + A) \tag{1}$$

Where:
- n represents the signal propagation constant, also named propagation exponent,
- d represents the distance from the sender,
- A represents the received signal strength at a distance of one meter.

The distance values are grouped into an array which is sent to another application using a Shared Variable. This option offers the possibility for the user to read the location from his application which runs locally on a Touch Panel Computer (TPC) installed on the car or on a mobile phone.
The program for localization using the Wi-Fi FPGA platform was implemented in LabVIEW2010. The application displays the distances between the Wi-Fi FPGA placed in a car and the three APs placed on a crossroad. The number of APs can be increased for obtaining a more accurate position estimate. The distances between the Wi-Fi FPGA and each AP are represented as circles having the centre in the AP locations which have previously known coordinates.

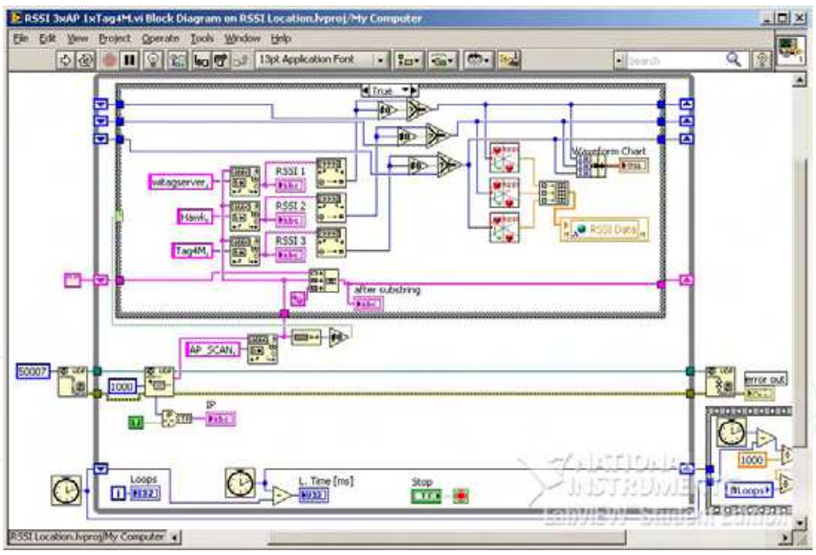


Fig. 9. Block Diagram for the Tag4M RSSI value reading program

The application was tested for only one crossroad because of the difficulty of placing a larger number of APs on a road. The car's position is given by the point at the intersection of the three circles.
The distance data are received from the program that reads the RSSI values using a Shared Variable. The local applications which run on a TPC or mobile phone convert the data to graphical representations which are afterwards displayed.

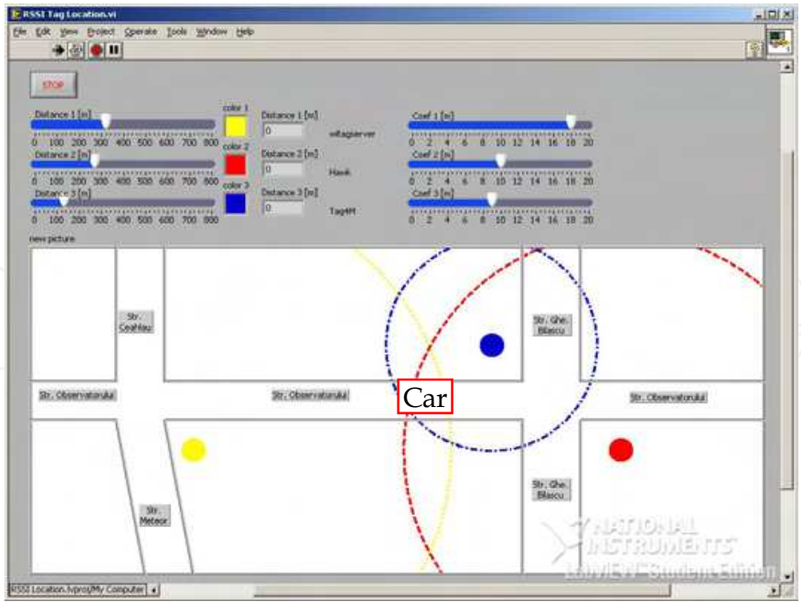


Fig. 10. Front Panel of the localisation program

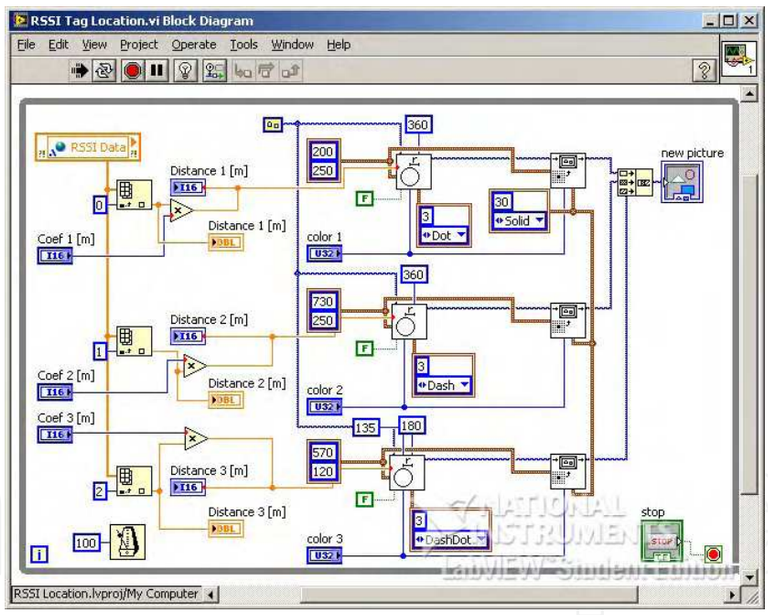


Fig. 11. Block Diagram of the localisation program

4.2 Detecting vehicle movement using accelerometer and the gyroscope sensors

Another application of the presented FPGA platform can be used for detecting the dynamic vehicle parameters. By attaching a gyroscope sensor and a 3-axis accelerometer to the FPGA board, the implementation of an application for determining the vehicle's state (moving or stopped, running on a straight road or making a turn) is possible. A LabVIEW application was implemented and installed on a FPGA in order to determine the vehicle's dynamic behaviour.

A correlation between the signals from a 3-axis accelerometer and a 2-axis gyroscope is realized in order to determine the state of the car. The signals acquired by the accelerometer and gyroscope sensors were filtered implementing a Butterworth filter (Figure 12) which has an essential characteristic: a smooth and monotonically decreasing frequency response (National Instruments, 2009). The Butterworth filter is available in LabVIEW FPGA, and the Express VI from the Help documentation and configure panel is presented in Figure 12 along with the settings. LabVIEW allows the operative and rapid change of the filter parameters and the usage of other types of filters.

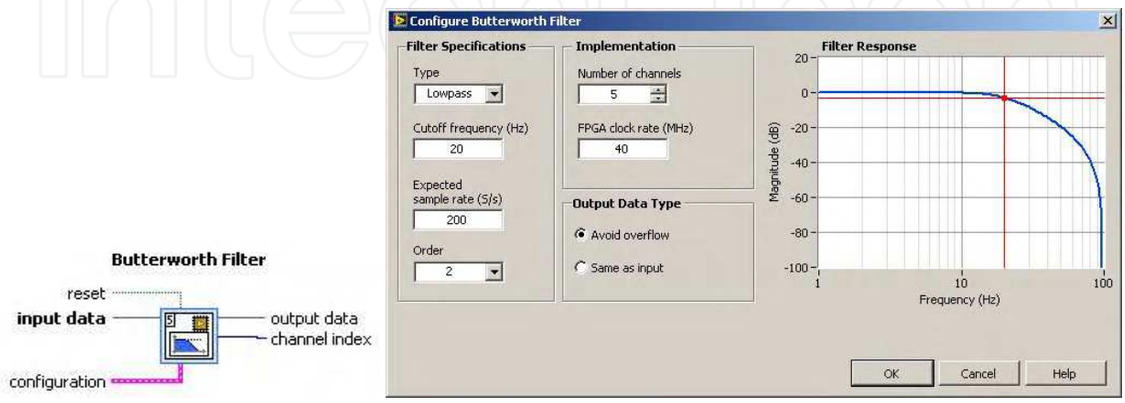


Fig. 12. Butterworth filter, FPGA implementation

3-Axis Acceleration Sensor

The acceleration sensor (Figure 13) allows the measurement of static or dynamic acceleration (in ±3g range) on three axes. The ADXL330, iMEMS type, from Analog Devices was chosen to be used for this purpose.

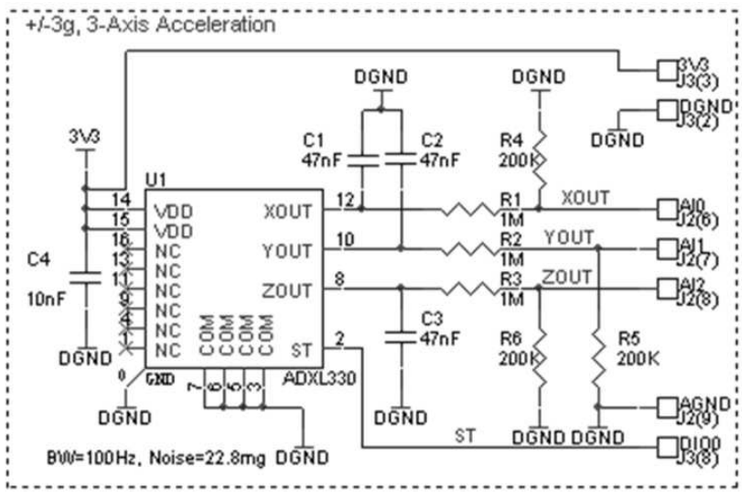


Fig. 13. Acceleration sensor scheme

External components are used for establishing the period of the output signal between 2 and 1000 ms, and the frequency band is limited to values between 0.5 and 1.6 kHz. The typical noise level is 280 µg/√Hz rms and allows the acquisition of signals under a 5 mg level (Analog Devices, 2006).

2-Axis Gyroscope Sensor

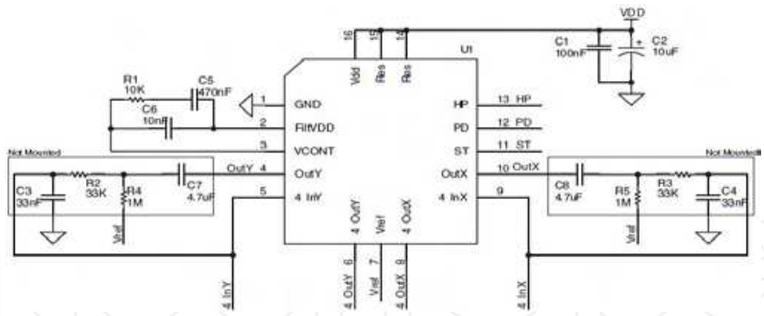


Fig. 14. Gyroscope sensor scheme

The gyroscope chosen to be used (LPR530AL) integrates one actuator and one accelerometer in a single micro machined structure. It is based on the Coriolis principle and it is able to react when an angular rate is applied to the sensing element which is kept in continuous oscillating movement (STMicroelectronics, 2009). The electric scheme is presented in Figure 14. It has a full scale of ±300 °/s and it is capable of detecting rates of up to 140 Hz with a -3 dB bandwidth.

A series of tests has been performed in order to determine a correlation between the vehicle's movement and the signal generated by the two sensors.

The program for displaying data acquired by the Wi-Fi FPGA platform from the 3-axis accelerometer and the 2-axis gyroscope was implemented in LabVIEW2010 and the application panel is presented in Figure 15. The application displays the signals received from the two sensors.

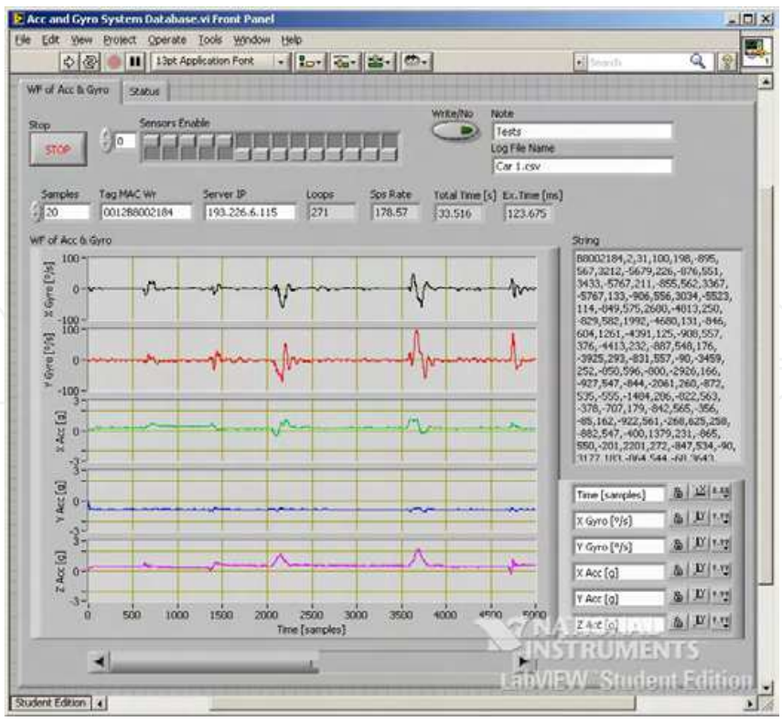


Fig. 15. Front Panel of the "Display Data" program

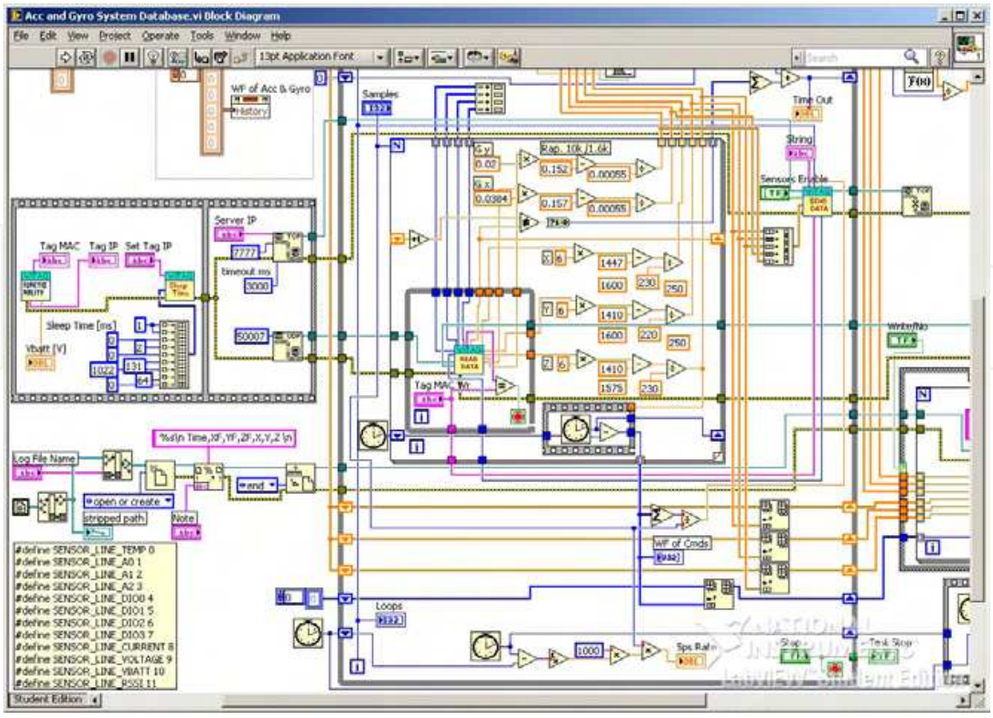


Fig. 16. Block Diagram of the "Display Data" program

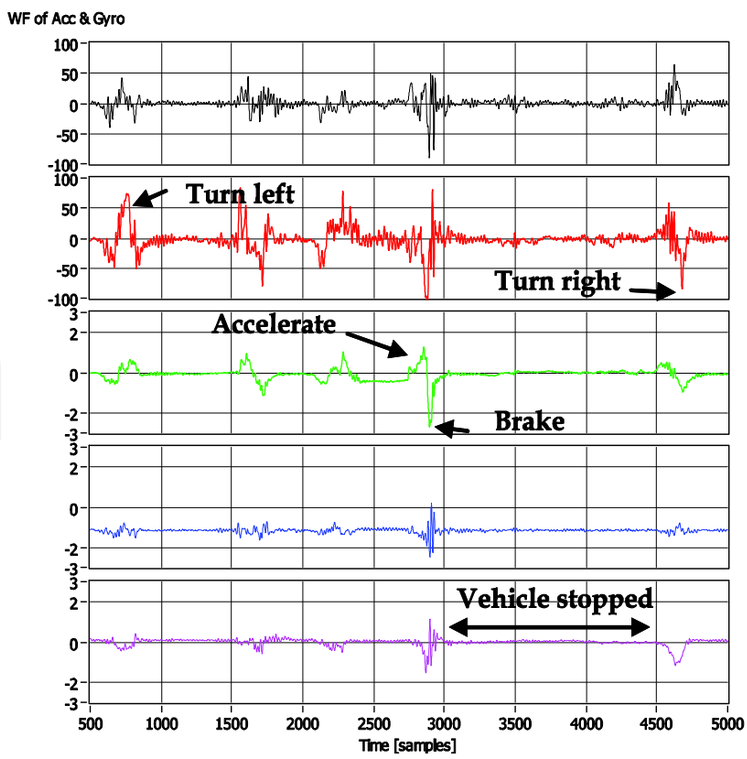


Fig. 17. Accelerometer and Gyroscope signals

In this case, the Tag4M device is used as a serial RS232 to Wi-Fi gate between the FPGA board (Spartan-3E Starter Kit) and the PC application that runs on the server. The block diagram of the PC application is presented in Figure 16.
One initialization step is necessary for reading the data from the Tag4M device. During this stage, the Tag4M sends a package containing the IP received through DHCP from the AP. This IP is used in the application for sending commands to the Tag4M device after the initialization step. The data are read in a loop and are validated if they are received from a previously known MAC address which is used as a validation mask. The latency determined after performing a number of experiments lies between the value of 5 and 20 milliseconds and it depends on the RSSI values. The values of the latency are greater in case of a poor signal.
Figure 17 presents the experimental results obtained from an IVU device installed on a vehicle performing a series of manoeuvres in a test field.
The length of the "Vehicle stopped" period is of 1500 samples which represent 7.5 seconds at a rate of acquisition of 200 Sps.

5. Conclusion

The first part of this paper presents a solution for remote Wi-Fi FPGA reconfiguration using Tag4M devices. The resulting system represents a versatile equipment with multiple applications in various fields, including ITS.
In the second part two applications developed for a FPGA platform that can be used in road traffic systems are presented. The first application exemplifies the way in which a vehicle can be tracked by using access points installed in the field and by using the Tag4M device which can read the RSSI values for each of them. The approximate of the vehicle's location is computed by applying a triangulation algorithm.
In the second application, the dynamic behaviour of a vehicle is determined by using a gyroscope and an accelerometer sensor attached to a FPGA board.
The work outlines the advantages provided to the developer and to the user by the employment of the FPGA technology and the features of the LabVIEW programming language in an Intelligent Transportation System. The reconfigurable systems' flexibility and the simplified way of creating programs for FPGAs by using the LabVIEW platform lead to a system that allows facile in the field hardware upgrades, runtime reconfiguration and adaptation to unexpected events or to changing environmental conditions.

6. References

Aamodt, K. (2006). CC2431 Location Engine, Application Note AN042, from http://focus.tij.co.jp/jp/lit/an/swra095/swra095.pdf

Adly, I.; Ragai, H. F.; Al-Henawy, A. & Shehata, K. A. (2010). Wireless Configuration Controller Design for FPGAs in Software Defined Radios, *The Online Journal on Electronics and Electrical Engineering (OJEEE)*, vol. 2, no. 3, pp. 293 – 297.

Altera (2009). AN 307: Altera Design Flow for Xilinx Users, from: http://www.altera.com/literature/an/an307.pdf.

Altera (2010). Medical Imaging Implementation Using FPGAs, *White Paper*, from http://www.altera.com/literature/wp/wp-medical.pdf

Altera (2011). Military End Market, from

http://www.altera.com/end-markets/military-aerospace/mil-index.html

Analog Devices (2006). Small, Low Power, 3-Axis 3g MEMS Accelerometer", Technical Report, from http://www.analog.com/en/mems-sensors/inertial-sensors/adxl330/-products/product.html

Andretzky, B. (2005). FPGAs Build Bridges To Wireless Connectivity, from http://electronicdesign.com/article/embedded/fpgas-build-bridges-to-wireless-connectivity9820.aspx

Banovic, K.; Khalid, M.A.S. & Abdel-Raheem, E. (2005). FPGA-Based Rapid Prototyping of DSP systems, *48th Midwest Symposium on Circuits and Systems*, pp. 647 – 650, Covington, KY.

Bitter, R.; Mohiuddin, T. & Nawrocki, M. (2006). *LabVIEW Advanced Programming Technique*, CRC Press, second edition.

Bolchini, C.; Miele, A. & Santambrogio, M. D. (2007). TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs, *Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pp.87 – 95, Rome, Italy.

Compton, K. & Hauck, S. (2002). Reconfigurable Computing: A Survey of Systems and Software, *ACM Computing Surveys*, vol. 34, no. 2, pp. 171 – 210.

El-Medany, W.M. & Hussain, M.R. (2007). FPGA-Based Advanced Real Traffic Light Controller System Design, *4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2007)*, pp. 100 – 105, Dortmund, Germany.

G2 Microsystems[1] (2008). G2C547 SoC, Technical Report, from www.g2microsystems.com

G2 Microsystems[2] (2008). G2C547 Software, Example Application. Technical Report, from www.g2microsystems.com.

G2 Microsystems[3] (2008). G2M5477 Wi-Fi Module Data Sheet, Technical Report, from www.g2microsystems.com.

HPCWire (2009). FPGA cluster accelerates bioinformatics application by 5000×, from http://www.hpcwire.com/offthewire/FPGA-Cluster-Accelerates-Bioinformatics-Application-by-5000X- 69612762.html

Kalte, H.; Langen, D.; Vonnahme, E.; Brinkmann, A. & Rückert, U. (2002). Dynamically Reconfigurable System-on-Programmable-Chip, *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pp. 235 – 242, Canary Islands, Spain.

Kuon, I.; Tessier, R. & Rose, J. (2007). FPGA Architecture - Survey and Challenges, *Foundations and Trends® in Electronic Design Automation*, vol. 2, no.2, pp. 135 – 253.

Lyke, J. (2002). Reconfigurable Systems: A Generalization of Reconfigurable Computational Strategies for Space Systems, *2002 IEEE Aerospace Conference Proceedings*, vol. 4, pp. 4-1935 – 4-1950.

Maye, J.; Supervisors: Upegui, A. & Prof. Ijspeert, A. J. (2005). Bluetooth Configuration of an FPGA: An Application to Modular Robotics, *Semester Project*, from http://birg.epfl.ch/webdav/site/birg/users/147507/public/semester/report.pdf

Mitra S.; Shirvani, P. P. & McCluskey, E.J. (1998). Fault Location in FPGA-Based Reconfigurable Systems, *IEEE Intl. High Level Design Validation and Test Workshop*.

National Instruments (2007). *CompactRIO™ and LabVIEW™ Development Fundamentals Course Manual*, Course Software Version 8.2.

National Instruments (2009). Working with LabVIEW Filtering VIs and the LabVIEW Digital Filter Design Toolkit Vis, , from http://zone.ni.com/devzone/cda/tut/p/id/4851

National Instruments[1] (2010). *What is LabVIEW?*, from http://zone.ni.com/devzone/cda/pub/p/id/1141

National Instruments[1] (2011). FPGA Technology, from http://www.ni.com/fpga_technology/

National Instruments[2] (2010), *Building Programmable Automation Controllers with LabVIEW FPGA*, Tutorial, from: http://zone.ni.com/devzone/cda/tut/p/id/3068

National Instruments[2] (2011). NI LabVIEW FPGA, from http://www.ni.com/fpga/

Patel, P. & Moallem, M. (2010). Reconfigurable system for real-time embedded control applications, *IET Control Theory and Applications*, issue 11, pp. 2506 – 2515.

Prasanna, V. K. & Dandalis, A. (2007). FPGA-based Cryptography for Internet Security, *Online Symposium for Electronics Engineers (OSEE)*, pp. 1– 6.

Reis, R. & Jess, Jochen A. G. (2004). *Design Of System On A Chip: Devices & Components*, Kluwer Academic Publishers.

Sander, O.; Glas, B.; Roth, C.; Becker, J. & Muller-Glaser, K. D. (2009). Design of a Vehicle-to-Vehicle Communication System on Reconfigurable Hardware, *International Conference on Field-Programmable Technology (FPT 2009)*, pp. 14 – 21, Sydney, NSW, Australia.

Smith, M. J. S. (1997). *Application-Specific Integrated Circuits*", Addison-Wesley Pub Co.

STMicroelectronics (2009). LPR530AP MEMS motion sensor: dual axis pitch and roll 300/s analog gyroscope, Technical Report, from http://www.st.com/internet/com/-TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00237209.pdf

Straka, M. & Kotasek, Z. (2008). Design of FPGA-Based Dependable Systems, *Proceedings of the 4th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pp. 240 – 247, Znojmo, Czech Republic.

Tessier, R. & Burleson, W. (2001). Reconfigurable Computing for Digital Signal Processing - A Survey, *Journal of VLSI Signal Processing*, vol. 28, issue 1/2, pp. 7 – 27.

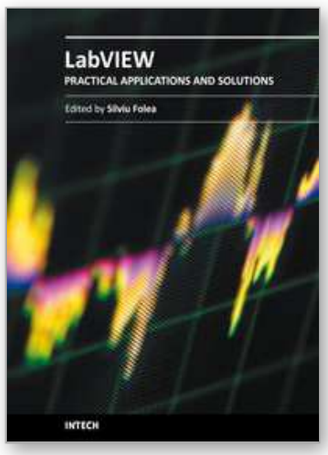
Xilinx (1998). The Low-Cost, Efficient Serial Configuration of Spartan FPGAs, XAPP098, Application Note by Kim Goldblatt, November 13 (Version 1.0), from http://www.xilinx.com/support/documentation/application_notes/xapp098.pdf

Xilinx (2006). Spartan-3E Starter Kit User Guide, User Guide, (Version 1.0), from http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf

Xilinx (2009). Spartan-3E FPGA Family: Data Sheet, Product Specification, August 26 (Version 3.8), from http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

Xilinx (2010). Partial Reconfiguration User Guide, from http://www.xilinx.com/-support/documentation/sw_manuals/xilinx12_1/ug702.pdf

Zhenggang, L.; Jialong, X.; Mingyun, Z.; Jun, Y. & Hongwei, D. (2009). FPGA-Based Dual-Mode Traffic Light System Design, *1st International Conference on Information Science and Engineering (ICISE)*, pp. 558 – 561, Nanjing, China.



Practical Applications and Solutions Using LabVIEW™ Software

Edited by Dr. Silviu Folea

ISBN 978-953-307-650-8
Hard cover, 472 pages
Publisher InTech
Published online 01, August, 2011
Published in print edition August, 2011

The book consists of 21 chapters which present interesting applications implemented using the LabVIEW environment, belonging to several distinct fields such as engineering, fault diagnosis, medicine, remote access laboratory, internet communications, chemistry, physics, etc. The virtual instruments designed and implemented in LabVIEW provide the advantages of being more intuitive, of reducing the implementation time and of being portable. The audience for this book includes PhD students, researchers, engineers and professionals who are interested in finding out new tools developed using LabVIEW. Some chapters present interesting ideas and very detailed solutions which offer the immediate possibility of making fast innovations and of generating better products for the market. The effort made by all the scientists who contributed to editing this book was significant and as a result new and viable applications were presented.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen