Boise State University
ScholarWorks

Electrical and Computer Engineering Faculty Publications and Presentations

Department of Electrical and Computer Engineering

1-7-2008

Hardware/Software Codesign in a Compact Ion Mobility Spectrometer Sensor Systemfor Subsurface Contaminant Detection

Sin Ming Loo
Boise State University

Jonathan P. Cole
Boise State University

Molly M. Gribb
Boise State University

This document was originally published by Hindawi in *EURASIP Journal on Embedded Systems*. Copyright restrictions may apply. DOI: 10.1155/2008/137295

Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2008, Article ID 137295, 8 pages
doi:10.1155/2008/137295

Research Article

Hardware/Software Codesign in a Compact Ion Mobility Spectrometer Sensor System for Subsurface Contaminant Detection

Sin Ming Loo,[1] Jonathan P. Cole,[1] and Molly M. Gribb[2]

[1] *Hartman Systems Integration Laboratory, Department of Electrical and Computer Engineering, Boise State University, Boise, ID 83725, USA*
[2] *Center for Environmental Sensing, Department of Civil Engineering, Boise State University, Boise, ID 83725, USA*

Correspondence should be addressed to Sin Ming Loo, smloo@boisestate.edu

Received 6 August 2007; Revised 3 December 2007; Accepted 7 January 2008

Recommended by Miriam Leeser

A field-programmable-gate-array-(FPGA-) based data acquisition and control system was designed in a hardware/software codesign environment using an embedded Xilinx Microblaze soft-core processor for use with a subsurface ion mobility spectrometer (IMS) system, designed for detection of gaseous volatile organic compounds (VOCs). An FPGA is used to accelerate the digital signal processing algorithms and provide accurate timing and control. An embedded soft-core processor is used to ease development by implementing nontime critical portions of the design in software. The design was successfully implemented using a low-cost, off-the-shelf Xilinx Spartan-III FPGA and supporting digital and analog electronics.

Copyright © 2008 Sin Ming Loo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Ion mobility spectrometry (IMS) is an analytical technique for gas phase analysis of chemical compounds in laboratory environments; more recently, this method has been used in field applications to rapidly detect chemical warfare agents, explosives, and narcotics [1, 2]. Ion mobility spectrometry is used to separate and quantify ions based on the drift of ions at ambient pressure under the influence of an electric field against a counter-flowing neutral drift gas. Gas samples to be analyzed are moved into a reaction region where the sample molecules are ionized using one of several ionization methods (in our case a Ni^{63} foil is used). The product ions produced are then introduced to the drift region of the IMS in pulses by opening a gate for a specified time interval (called the ion gate pulse width, or pulse width). Once in the drift region, the ions are separated according to their mobilities, which are dependent on the size-to-charge ratio of the ions. When the ions reach the detector, a small current is generated that is amplified and sent to the data acquisition system. This current is recorded as a function of drift time, which is used to identify each analyte; the relative size of the current peak is used to determine concentration.

The focus of this paper is on the use of a hardware/software codesign technique and an FPGA for control of a small IMS sensor system designed for real-time detection of gaseous VOCs in the subsurface [3–5]. The areas of effort include the use of an FPGA to implement IMS data acquisition and processing, sensor and sampling module control, digital filtering to remove noise from the IMS data, and digital signal processing for detecting current peaks for identification of VOCs. Section 2 provides an overview of the IMS system, Section 3 describes the hardware/software codesign implementation details, and in Section 4, results of system validation tests are discussed. Finally, a summary is presented in Section 5.

2. IMS SENSOR SYSTEM

The sensor system is comprised of downhole subsystem and uphole components (Figure 1). The uphole subsystem includes a power supply, supply monitoring and communication, and the sensor drift and carrier gas supply. The downhole subsystem is housed in a 4.4-cm diameter, and 1.22-m long cylindrical steel casing that can be lowered down an open borehole (currently deployable to approximately 10 m)

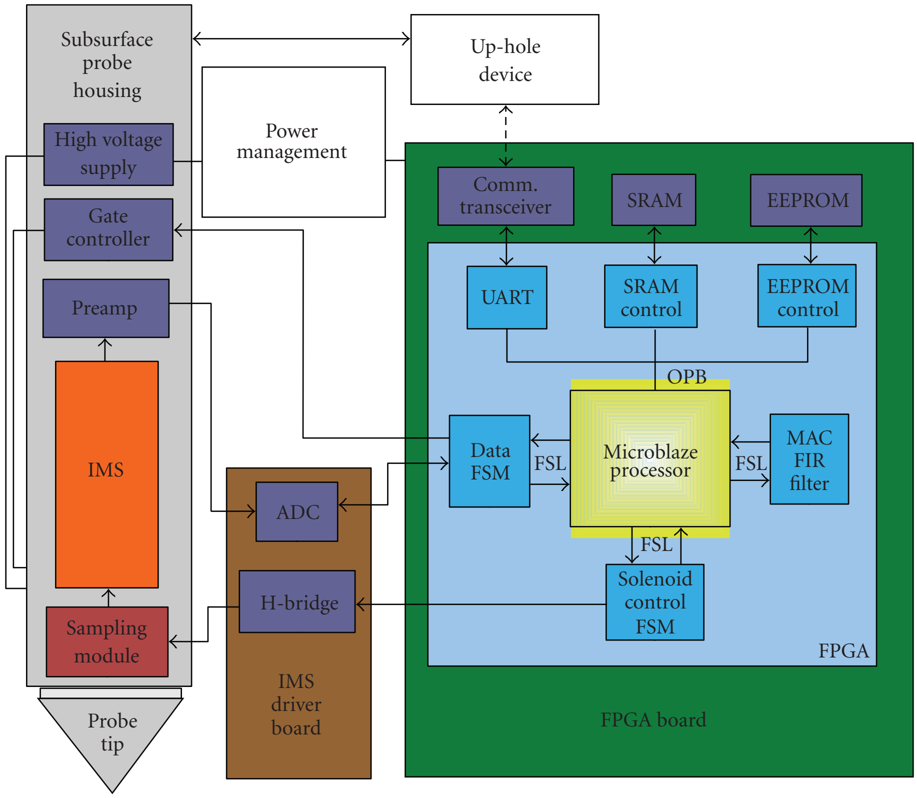

Figure 1: IMS hardware/software system block diagram.

or inserted into the soil using telescopic drilling techniques (Figure 2). The downhole subsystem includes the sampling module (used to draw in gaseous samples from the soil for introduction to the sensor), IMS sensor, preamplifier, high voltage power supply, power manager, and an FPGA responsible for communication, control, and data acquisition and processing. During the prototyping stage, the FPGA is placed uphole, but will ultimately reside with the downhole components. A compact Spartan-III FPGA board has been designed to be inserted into the IMS probe housing as shown in Figure 2.

3. IMS HARDWARE/SOFTWARE CODESIGN

Previously, an embedded data acquisition (DA) system using a Microchip PIC18F452 microcontroller (PIC) was used for IMS system control [6, 7]. The PIC-based DA system provided a small, low-cost, and low-power solution capable of meeting the needs of the system. The DA system took advantage of the resources of the PICby using the on-chip analog-to-digital conversion (ADC) to digitize the analog signal provided by the IMS and the on-chip memory to store the data. However, the PIC-based DA system had many limitations, including limited memory (1536 bytes), low-sampling rate (10 kHz) and resolution (10-bit) ADC, only 24 digital inputs/outputs (I/Os), and lack of digital signal processing (DSP) power. These limitations led to very limited research growth potential for the IMS system. When the DA requirements were re-evaluated, three of the most important criteria identified were performance, accurate timing control, and expandability.

To overcome some of the limitations of PIC microcontroller, a low-cost Xilinx Spartan-III FPGA was chosen to be the main processing element of the IMS sensor system. The FPGA provides a high degree of flexibility and scalability, good high speed (DSP) performance, and a larger number of I/O pins (over 200 I/Os). Flexibility in the IMS system was a high priority, since neither the IMS sensor nor the system itself had been fully optimized. The high performance FPGA DSP allows for upgrades and higher-order digital filtering techniques that can help improve the signal-to-noise ratio (SNR) and, therefore, the resolving power of the sensor.

3.1. IMS control stages

The process of controlling the IMS sensor system can be divided into the following five control stages: *initialization*, *sampling*, *data acquisition*, *postprocessing*, and *data transmission*. In the initialization stage, the drift and carrier gas flow rates are measured and if they are out of their desired operating ranges, they are adjusted accordingly. In addition, the ion gate pulse width (PulseWidth) and data scan time (i.e., ScanTime, length of time over which ion currents are collected at the detector) are set. The IMS system is then ready to begin the sampling stage, during which a gaseous sample is extracted from the soil with the sampling module and introduced into the ionization region of the IMS sensor. In the data acquisition stage, the ion gate is cycled open and closed

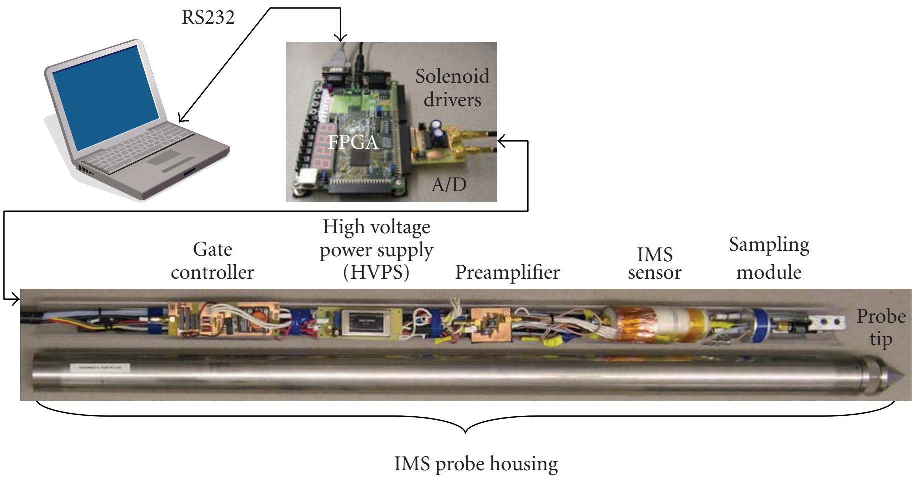

FIGURE 2: IMS sensor with supporting electronics, metal probe housing, Spartan-III Starter Kit FPGA, and computer.

for 200 microsecond intervals (PulseWidth), and ions travel through the drift tube until they strike the detector. The ion currents are amplified, digitized, and stored in memory. This data is collected repeatedly and averaged to eliminate noise. If desired, current data is digitally filtered to further reduce the amount of noise in the data and/or reduce the number of data samples required to represent the signal. Once the data are in memory, the postprocessing stage occurs when further filtering may be applied and peak times and areas are calculated. Once this stage is complete, the data are transmitted to the uphole subsystem.

3.2. Hardware and software control

The FPGA (along with the necessary external circuitry) must initiate the various IMS control stage transitions, control the ion gate controller and sampling module, collect, amplify, digitize, and process the ion current measurements, perform desired postprocessing of the data (including ion peak calculations for identification and quantification of analytes), and finally, communicate with the uphole subsystem. The roles of the FPGA, the hardware modules that are implemented by the FPGA, and most importantly the Xilinx Microblaze microprocessor, are shown in the IMS system diagram (Figure 1). It is interesting to note that the system control naturally breaks down into hardware (those requiring speed and accurate timing control) and software (postprocessing for rapid algorithm changes/updates). Portions requiring tight timing requirements were implemented with VHDL and interfaced to the Microblaze through fast simplex link (FSL). The overall operation of IMS hardware and software control can be visualized with a task graph with six tasks; see Figure 3 (init: initialization, samp: sampling, da: data acquisition, post: postprocessing, and com: communication). The control is sequential in nature, thus hardware is used where performance and accurate timing are required.

The init task is responsible for checking environmental variables including drift gas flow rates, and receiving parameters from the uphole system. This task has been implemented in C since parsing through a data packet can be done much more easily and efficiently in software. The samp task is implemented in VHDL since accurate timing can be achieved in hardware (by counting clocks). The ionized sample molecules do not move into the drift region until the IMS gate opens during the data acquisition stage (da task). The third task, da, involves opening and closing the IMS gate at discrete intervals (PulseWidth) and allowing groups of ions to travel through the IMS where they eventually strike the collector. This collection process is repeated, and the data is averaged together to eliminate any anomalies. Much of the da task is implemented in VHDL with moderately complex control in C. In the post task, the data can again be filtered digitally (implemented in VHDL) to further remove any noise from the data, and analysis methods can be applied to calculate the peak locations and the area under the peaks, to identify and quantify the compounds present. For example, with peak detection algorithm, the ability to change number of scans that are averaged is extremely important during the research phase of the sensor system development program. Hardware implementations with state machines were used because accurate timing control is critical for some tasks and settings (e.g., InjectTime, ExtractTime, PulseWidth, and ScanTime) of the sensor system. Finally, for the com task, data are passed from the downhole device to the uphole device. Details on the uphole device and the communication protocol have not yet been fully specified. However, it is assumed that communication with the uphole device will be achieved via a communication standard called recommended standard 232 (RS232). This standard was chosen because of its wide use in industry and ease of implementation. Figure 4 contains additional details related to Figure 3.

3.2.1. Control stages transitions

Transitions between control stages are achieved in the FPGA as a combination of software implemented in C running on an embedded Microblaze soft-core processor, and two separate hardware finite state machines (FSMs) written in VHDL.

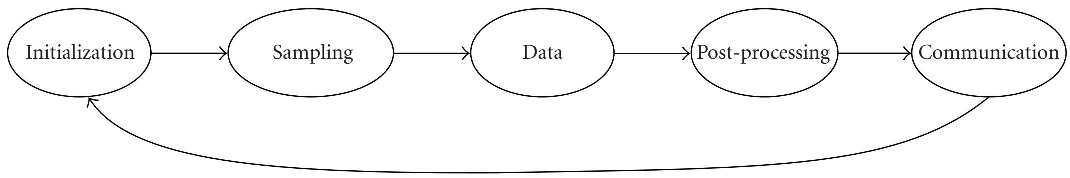

Figure 3: IMS control task graph.

The software control program remains in an idle state until it receives a command from the uphole device. It will then parse out the desired commands and data and begin the IMS sensor data collection by sending commands to the appropriate hardware. Figure 4 shows the flow diagram of the overall control program. Due to the large amount of RAMrequired to store and process the IMS data, the code is designed to run in external 1 Mb static random access memory (SRAM), as opposed to the internal block RAM(BRAM) of the FPGA. The Microblaze controls the SRAMusing an external memory controller (EMC) IP core provided by Xilinx EDK, which attaches to the Microblaze via the on-chip peripheral bus (OPB).

A bootloader is used to run the software control in SRAM. The software control program data is placed in the Xilinx platform flash programmable read only memory (PROM) along with the bit stream that is used to configure the FPGA. The bootloader program runs in BRAM, and during power-up of the system it copies program data from the PROM and places it in SRAM. The bootloader then jumps to the start of the SRAM and executes the program. The bootloader program and the methods used for extracting the data from the PROM are based on a sample design provided by Xilinx [8]. The PROM interface design is shown in Figure 5. The logic in Figure 5 includes clock to the PROM, enable, and initialize signal to indicate start of read. Since XCF02S is a serial PROM, DIN is only one-bit width.

The first hardware FSM (referred to hereafter as the DA FSM) generates the ion gate pulse width (PulseWidth), collects the ion current data from the IMS sensor, and sends it to the Microblaze where it is stored in SRAM. The DA FSMprovides the software control program (running in Microblaze) with new ion current data points as they become available. The DA FSM communicates with Microblaze via two FSL busses, one for sending and the other for receiving data. These high-speed, unidirectional links allow the DA FSM to pass data fast enough for the Microblaze to receive and process it before receiving the next data packet. The FSL also allows the Microblaze to send the FSM a single 32-bit parameter representing the settings for the current IMS test parameters (ScanTime, PulseWidth, etc.), which is decoded by the DA FSM. A state machine diagram of the DA FSM is shown in Figure 6. The state machine uses ClkCnt = 9 to transition to the send pulse state to ensure that the state machine is synchronized properly. The ClkCnt counter is used to generate the ADC clk (which is always running), and checking for ClkCnt = 9 synchronizes the start of the gate pulse with the rising edge of the clock. This also helps ensure that gate pulse width is always the same width by making the gate pulse starts and stops on the same number of counts every time.

The sampling module FSM controls the timing of the actuation of solenoid valves in the sampling module that extract samples from the soil and then inject them into the IMS ionization region. This FSM also communicates with the Microblaze via two FSL busses. The software control program sends data to the sampling module control FSM as a single 32-bit word representing the extraction and injection sequence for the valves. The FSM stays in IDLE state until a start signal is received. At this time FSM proceeds to extract state and the sample is extracted from the soil. Once enough sample has been extracted, FSM moves to inject state where the sample is introduced to the IMS sensor. A single bit is sent to the Microblaze via FSL when the sampling module FSM has completed its task. A high-level state machine diagram of this FSM is shown in Figure 7.

3.2.2. *IMS gate controller*

The IMS gate controller is a digital device that controls the flow of ions from the ionization region into the drift region of the IMS sensor. The FPGA interfaces with the gate controller via a single digital signal. When the signal is a digital "low" the IMS gate is closed, and when the signal is a digital "high" the IMS gate is open. A single, brief opening and closing of IMS gate is generated in the FPGA by counting clock cycles using a countup counter, implemented as hardware in the FPGA. A countup counter is designed to increment on every rising edge of a clock. The counter value needed to generate a desired pulse width is equal to the desired ion gate pulse width multiplied by the system clock frequency. The accuracy of the timing is limited only by the frequency of the system clock and the jitter of the signal. Our system clock runs at 75 MHz using a digital clock manager (DCM) module, which makes it possible to generate pulses (PulseWidth) with an accuracy of 13.33 nanoseconds, or the duration of a single clock cycle. If greater accuracy is required, the system clock frequency can be raised using a DCM. The ion gate pulse width is part of the hardware DA FSM and is designed to provide the user with the option of selecting the desired ion gate pulse width. This flexibility was especially valuable during the prototype development stage of the IMS system when the ideal pulse width had not yet been determined.

3.2.3. *Sampling module*

The sampling module extracts a volume of gas samples from the soil to be introduced to the IMS sensor for analysis using

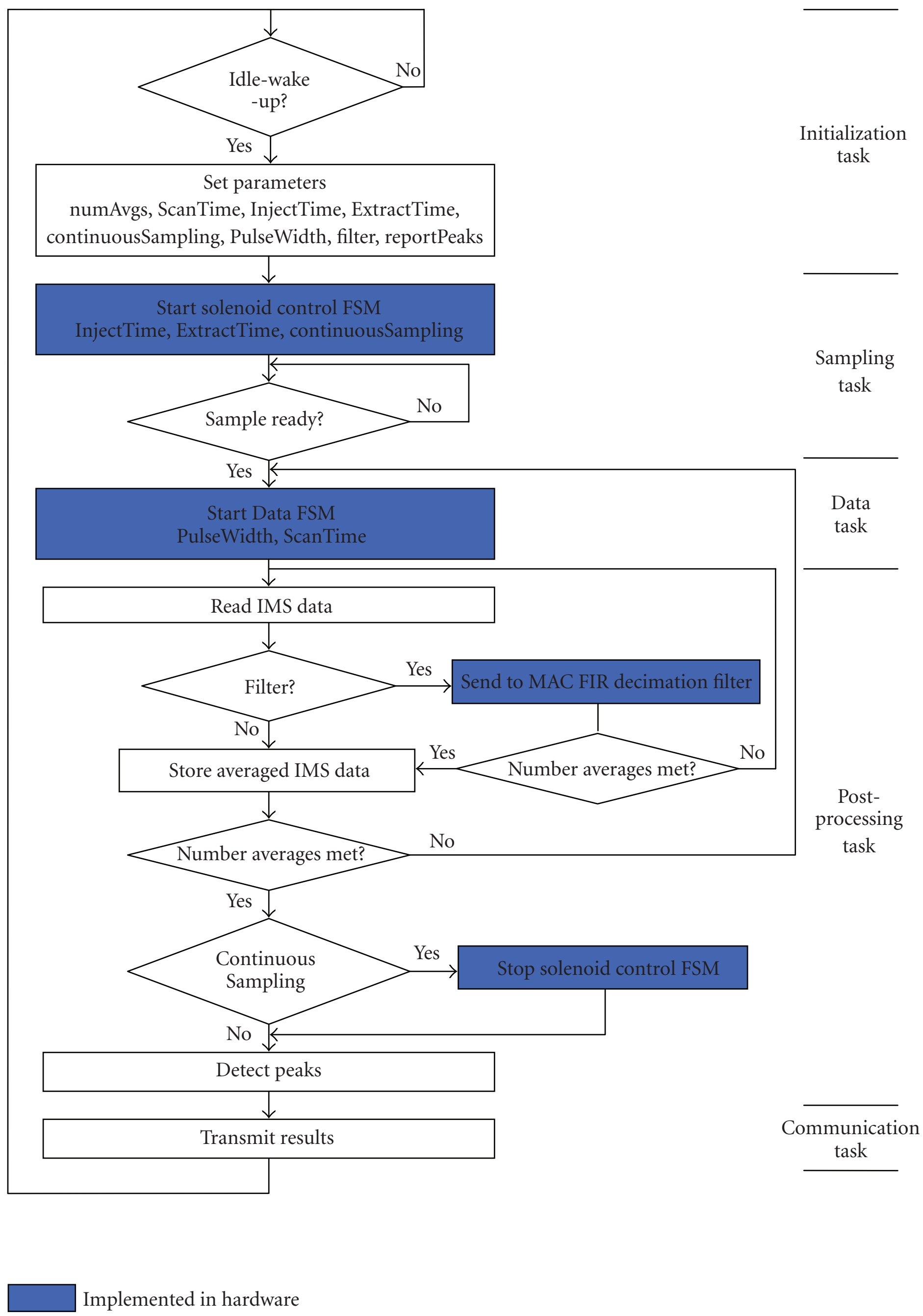

Figure 4: Overall control program.

latching solenoid valves [9]; it can be run in “single injection mode” in which a single sample is extracted, injected, and analyzed or in continuous mode in which a continuous stream of sample is injected into the IMS for analysis. These valves are controlled by swapping the high and low terminals and sending 20-millisecond, 5-V pulses to change the open/close state. As these valves could not be driven directly from the FPGA, which is only capable of driving 24 mA of current at a maximum of 3.75 V, a half-H bridge (Texas Instruments SN754410NE) is used to drive the valves. The control signals for the valves are generated by the sampling module FSMas described above. The timing of state transitions is achieved by counting clock cycles with a countup counter in the same manner as for the gate controller. The sampling module control code was written in VHDL and is designed to allow for flexibility in the duration of the extraction and injection states, so that the system can be optimized for differing environmental conditions. A single 32-bit control word contains extraction time (ExtractTime), injection time (InjectTime), and continuous or single injection mode.

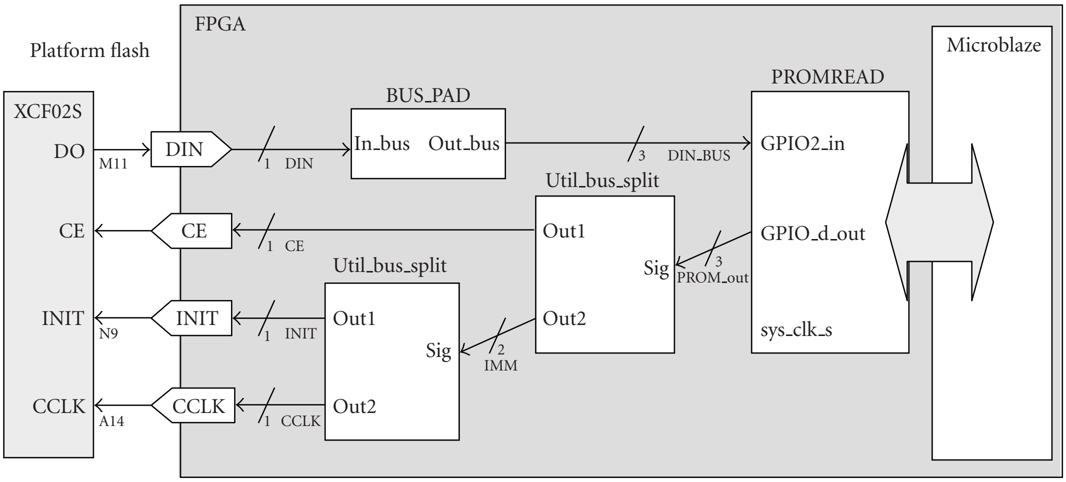

Figure 5: Logic for bootloader.

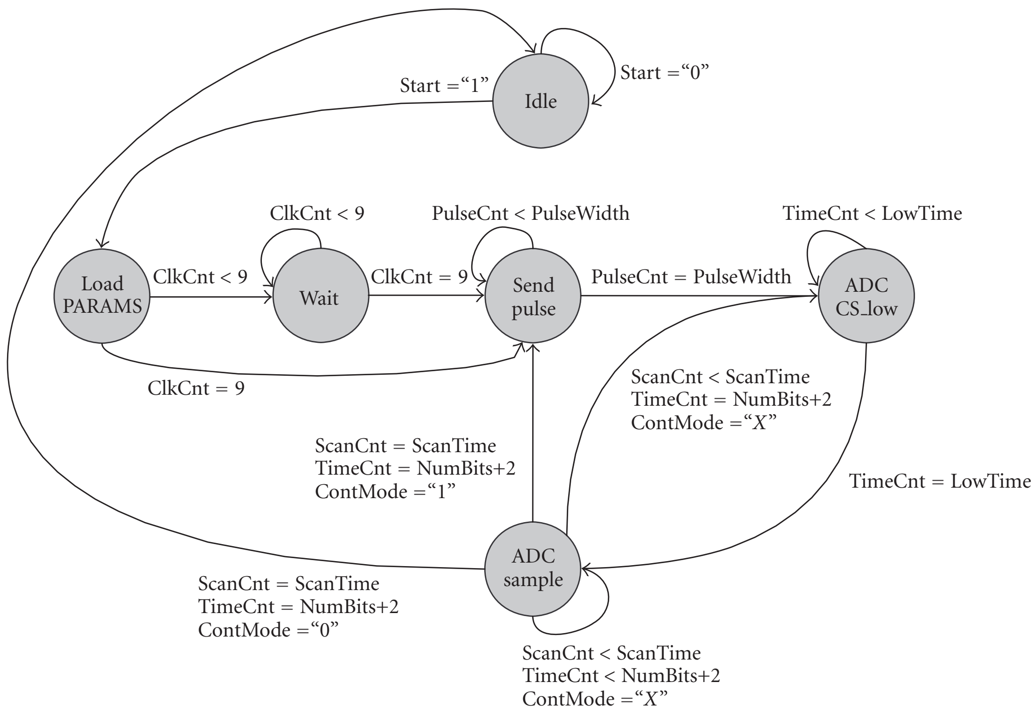

Figure 6: DA FSM state machine diagram.

3.2.4. Data acquisition and processing

The data acquisition process involves the following stages: amplification, digitization, and processing of the ion current. These stages are described below.

Preamplification

An inverting opamp configuration with a 10^9 gain factor was used to amplify the subnano Amp ion current signal from the IMS and convert it to a voltage before ADC.

Analog-to-digital conversion

For ease of implementation, an off-the-shelf successive-approximation register (SAR) ADCwas used for ADC (ADS7818, TI). All noise filtering is applied during postprocessing to ensure that as much of the IMS signal remains intact as possible. The ADS7818 interfaces with the FPGA via a 3-wire serial peripheral interface (SPI) protocol. The data are collected from the ADS7818 in a 12 bit shift-register. The DA FSMcollects data from the ADS7818 and sends them to the Microblaze via FSL for processing and storage in SRAM every 2.1 microseconds (ScanTime). It is noted that ADS7818 has

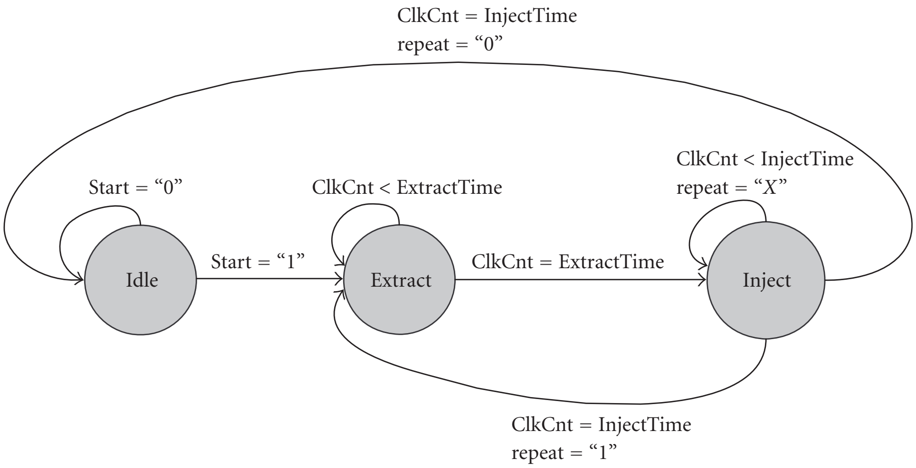

Figure 7: Sampling module FSM state machine diagram.

top sampling rate of 500 K samples/second. Although this is likely not needed for the system, it can be used to enhance noise filtering by pushing the noise frequency further out in the frequency spectrum, thus making the noise easier to filter.

Real-time processing

To improve analyte identification ability, it is important to remove as much of the noise in the ion current readings as possible without removing valid content. Therefore, multiple scans of the IMS spectra are averaged to filter out system noise. After data collected by the hardware FSM are sent to Microblaze, averages are performed in real time and stored in external SRAM. The number of scans to be averaged is a variable selected by the user via the uphole device.

The second area of real-time processing performed in the FPGA and Microblaze is decimation filtering. The decimator is designed using finite impulse response (FIR) filtering techniques. The filter coefficients used in this design were calculated using Matlab (MathWorks, Natick , MA) filter design tools [10]. The user selects the windowing method used to calculate the coefficients, the number of taps, the cut-off frequency and filter gain, and whether symmetric coefficients are desired. The Hamming window was used, and a 65-order (65 taps) filter was selected with a gain of 1 and a cut-off equal to 10 kHz. Since IMS signal has bandwidth of less than 10 kHz, a decimation factor of 10 was chosen to bring the sampling rate down to ∼50 kHz, as this is an appropriate sampling rate for the IMS. The design tool was configured to generate the coefficients as unsigned 16-bit values to facilitate the integration of the coefficients into FPGA filter implementation. In this implementation, oversampling (500 K samples/second and decimated to 50 kHz) is used as an attempt to reduce noise due to the antenna created by the wiring in the probe and support cables to the uphole system components.

The decimator is implemented in this design using the Xilinx Logicore multiply accumulate finite impulse response v5.1 (MACFIR) core attached to Microblaze via a send FSL and a receive FSL [11]. The MACFIR is a highly configurable and highly efficient core that is included with the Xilinx ISE tools.

Postprocessing

Postprocessing of IMS data may be performed in the FPGA prior to sending the data uphole, or after the data have been collected by the uphole device. Once the noise has been removed from the signal, the data can be processed to identify the analytes present in the sample. An important step in this identification process is peak detection. The peak detection algorithm is implemented as a software routine written in C running on Microblaze. The detection algorithm allows the number of nearest neighbors to be passed in as an input parameter to allow the routine to be altered according to the types of compounds being detected, the widths of the peaks being detected, and so that the number of unwanted peak detections that occur due to noise can be reduced. In addition, a noise floor threshold parameter is used that limits peaks detected to those occurring above that threshold. This is necessary even after applying noise filtration techniques because a certain amount of noise will nearly always be allowed to pass through the noise filter. The algorithm for the autodetermination of the noise floor threshold uses the initial ∼1 millisecond of data to determine the threshold by storing the largest value during this interval as the threshold. This method assumes that no peaks occur during this interval, and the detected current is only due to DC offset and noise. If this is not the case, then the threshold would need to be provided directly or the algorithm can be altered as needed.

Once the IMS sensor system is fully characterized and calibrated, a table of mobility values will be stored in the system. The peak locations (along with pressure and temperature measurements) will allow analytes to be identified by calculating the reduced ion mobility values associated with the peaks and comparing them to the values stored in the table. The results will then be sent to the uphole device.

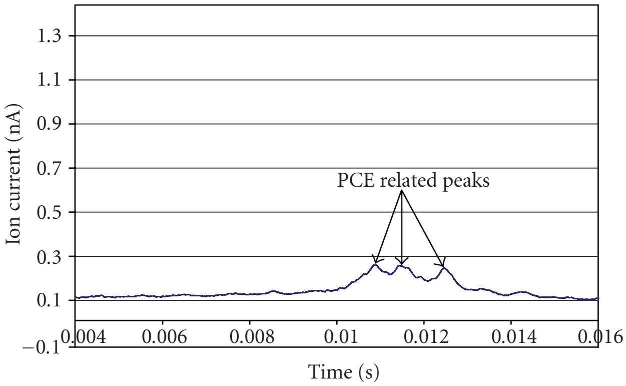

Figure 8: 15 PCE waveform collected using FPGA and ADC (512 averages).

4. SYSTEM VERIFICATION

A step-by-step approach to system verification was taken. Each component of the design was verified on an individual basis, then additional components were successively combined until the entire design was verified. Figure 8 shows perchloroethylene (PCE) data captured by the IMS system under laboratory conditions. The sampling module was set to continuously extract and inject the PCE sample gas with extraction and injection times both equal to 0.5 second. An ion gate pulse width of 200 microseconds and scan time of 19.6 milliseconds was used. As the PCE sample was injected into the IMS, three distinct peaks characteristic of PCE became apparent. The data was collected using the design as described in Section 3 with 512 averages. This test showed that the FPGA is capable of successfully gathering data from the IMS while operating the sampling module. Further complete sensor system characterization is currently underway.

5. SUMMARY

An FPGA data acquisition and control system for a compact IMS sensor system that was designed for subsurface use was successfully designed, implemented, and tested. The functionality of the design was verified in a laboratory environment with PCE and in the field (not shown). The design provides accurate control and timing to the IMS gate controller and sampling module while simultaneously collecting IMS data with an external ADC. The system, even though it is very sequential in nature, was designed in a true hardware/software codesign environment with the use of Xilinx Microblaze soft processor core; it takes advantage of the hardware resources of the FPGA to employ real-time digital signal processing to reduce noise in the data which eases further processing of the data. The design makes use of an embedded soft-core processor to provide a high-level software interaction to the system and implement a peak detection algorithm. Since the FPGA can be reconfigured easily, the design is flexible enough to allow for future changes and improvements.

At this writing, this is the first use of FPGA in an IMS system which has yielded a system that is easily expandable and reconfigurable. These attributes are highly desirable, so that as the needs of the IMS research program change, the hardware can keep up with it. Throughout this research, design, and implementation process, we ran into many software (development tool) bugs, which were resolved in the latest software releases. However, upgrading the software would cause problems in other portions of our design. We found the flexibility gained through the use of the FPGA and Microblaze a mixed blessing. The system is very configurable, but due to software problems, it was often difficult to determine if a problem was due to hardware or software elements of the system.

ACKNOWLEDGMENT

Funding of this project by EPA Award nos. X97031101-0 and X97031102-0 is gratefully acknowledged.

REFERENCES

[1] F. W. Karasek, “Plasma chromatography,” *Analytical Chemistry*, vol. 46, no. 8, pp. 710A–720A, 1974.

[2] G. A. Eiceman and Z. Karpas, *Ion mobility spectrometry*, CRC Press, Boca Raton, Fla, USA, 1994.

[3] A. B. Kanu, H. H. Hill, M. M. Gribb, and R. N. Walters, “A small ion mobility spectrometer sensor for detecting environmental soil-gas contaminants,” *Journal of Environmental Monitoring*, vol. 9, no. 1, pp. 51–60, 2007.

[4] D. Sevier, M. M. Gribb, R. N. Walters, et al., “An in-situ ion mobility spectrometer sensor system for detecting gaseous VOCs in the Vadose zone,” in *Proceedings of the 4th International Conference on Unsaturated Soils*, G. A. Miller, C. E. Zapata, S. L. Houston, and D. G. Fredlund, Eds., vol. 147, pp. 225–234, ASCE Publications, Carefree, Ariz, USA, April 2006.

[5] J. Cole, “On the use of a field programmable gate array in a compact ion mobility spectrometer sensor system for subsurface volatile organic compound detection,” M.S. thesis, Department of Electrical and Computer Engineering, Boise State University, Boise, Idaho, USA, 2007.

[6] W. A. Prouty, “Embedded system design for multi-purpose sensors to detect and analyze environmental contaminants,” M.S. thesis, Department of Electrical and Computer Engineering, Boise State University, Boise, Idaho, USA, 2003.

[7] “Microchip PIC18F452 System Specification,” Microchip Technology, October 2006, http://www.microchip.com.

[8] Xilinx PROM Application Note (XAPP 483), August 2006, http://www.xilinx.com/bvdocs/appnotes/xapp483.pdf.

[9] K. P. Ryan, “A gas sampling module for a subsurface ion mobility spectrometer,” M.S. thesis, Department of Civil Engineering, Boise State University, Boise, Idaho, USA, 2006.

[10] MathWorks, Matlab Filter Design Toolbox, October 2006, http://www.mathworks.com.

[11] Xilinx MAC FIR Filter (V5.1), March 2007, http://www.xilinx.com/ipcenter/catalog/logicore/docs/mac_fir.pdf.

EURASIP Journal on Wireless Communications and Networking

Special Issue on
Network Coding for Wireless Networks

Call for Papers

The main idea in network coding was introduced in 2000 by Ahlswede et al. With network coding, an intermediate node in a network cannot only forward its incoming packets but also encode them. It has been shown that the use of network coding can enhance the performance of wired networks significantly. Recent work indicates that network coding can also offer significant benefits for wireless networks.

Communications over wireless channels are error-prone and unpredictable due to fading, mobility, and intermittent connectivity. Moreover, in wireless networks, transmissions are broadcast and can be overhead by neighbors, which is treated in current systems as interference. Finally, security poses new challenges in wireless networks, where both passive and active attacks have quite different premises than in wired networks. Ideas in network coding promise to help toward all these issues, allowing to gracefully add redundancy to combat errors, take advantage of the broadcast nature of the wireless medium and achieve opportunistic diversity, exploit interference rather than be limited by it, and provide secure network communication against adversarial attacks.

In this special issue, we are interested in original research articles which can carry the momentum further and take the wireless network coding research to the next level. The areas of interest include novel network code designs and algorithms, new applications of wireless network coding, network coding capacity, and performance analysis. In addition to original research articles, we are open to review articles. The following list indicates topics of interest which is by no means exhaustive:

- Network codes and algorithms for wireless networks
- Physical layer network coding
- Joint source coding and network coding
- Graph codes and network coding
- Reduced complexity decoding for network coding
- Secure network coding
- Capacity and fundamental bounds on network coding performance
- Cross-layer optimization and network coding
- Energy-efficient network coding
- TCP, routing, MAC, or scheduling algorithms for network codes
- Wireless network coding for multimedia application
- Wireless network coding for bio-medical application

Before submission authors should carefully read over the journal's Author Guidelines, which are located at http://www.hindawi.com/journals/wcn/guidelines.html. Prospective authors should submit an electronic copy of their complete manuscript through the journal Manuscript Tracking System at http://mts.hindawi.com/ according to the following timetable:

Manuscript Due	October 1, 2009
First Round of Reviews	January 1, 2010
Publication Date	April 1, 2010

Lead Guest Editor

Heung-No Lee, Department of Information and Communications, Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea; heungno@gist.ac.kr

Guest Editors

Sae-Young Chung, School of EECS, KAIST, Daejeon, South Korea; sychung@ee.kaist.ac.kr

Christina Fragouli, School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland; christina.fragouli@epfl.ch

Zhi-Hong Mao, ECE/Bio-Medical Dept., the University of Pittsburgh, Pittsburgh, PA, USA; zhm4@pitt.edu

Special Issue on Design, Implementation, and Evaluation of Wireless Sensor Network Systems

Call for Papers

The recent advances in embedded software/hardware design have enabled large-scale and cost-effective deployment of Wireless Sensor Networks (WSNs). Such a network consists of many small sensor nodes with sensing, control, data processing, communications, and networking capabilities. The wireless sensor networks have a broad spectrum applications ranging from wild life monitoring and battlefield surveillance to border control and disaster relief, and have attracted significant interests from both academy and industry.

A wireless sensor node generally has limited storage and computation capabilities, as well as severely constrained power supplies, and the networks often operate in harsh unattended environments. Successful design and deployment of wireless sensor networks thus call for technology advances and integrations in diverse fields including embedded hardware manufacturing and signal processing as well as wireless communications and networking across all layers. We have seen the initial and incremental deployment of real sensor networks in the past decade, for example, the ZebraNet for wildlife tracking, the CitySense for weather and air pollutants reporting, and the Sensormap portal for generic monitoring services, to name but a few; yet the full potentials of such networks in the real world remain to be explored and demonstrated, which involves numerous practical challenges in diverse aspects.

This special issue aims to summarize the latest development in the design, implementation, and evaluation of wireless sensor systems. Topics of interest include but are not limited to:

- Practical architecture and protocol design for sensor communications and networking
- Management, monitoring, and diagnosis of sensor networks
- Implementation and measurement of experimental systems and testbeds
- Experience with real-world deployments of wireless sensor networks
- Interactions with ubiquitous communication and networking infrastructures
- Novel killer applications of wireless sensor networks

We solicit original unpublished research papers only. Papers previously published in conference/workshop proceedings can be considered, but should be substantially extended.

Before submission authors should carefully read over the journal's Author Guidelines, which are located at http://www.hindawi.com/journals/wcn/guidelines.html. Prospective authors should submit an electronic copy of their complete manuscript through the journal Manuscript Tracking System at http://mts.hindawi.com/ according to the following timetable:

Manuscript Due	February 15, 2010
First Round of Reviews	May 15, 2010
Publication Date	August 15, 2010

Lead Guest Editor

Jiangchuan Liu, School of Computing Science, Simon Fraser University, British Columbia, Canada; jcliu@cs.sfu.ca

Guest Editors

Jiannong Cao, Department of Computing, Hong Kong Polytechnic University, Hong Kong; csjcao@comp.polyu.edu.hk

Xiang-Yang Li, Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA; xli@cs.iit.edu

Limin Sun, Institute of Software, Chinese Academy of Science, Beijing, China; sunlimin@ios.cn

Dan Wang, Department of Computing, Hong Kong Polytechnic University, Hong Kong; csdwang@comp.polyu.edu.hk

Edith C.-H. Ngai, Department of Information Technology, Uppsala University, Sweden; edith.ngai@it.uu.se

EURASIP Journal on Image and Video Processing

Special Issue on Dependable Semantic Inference

Call for Papers

After many years of exciting research, the field of multimedia information retrieval (MIR) has become mature enough to enter a new development phase—the phase in which MIR technology is made ready to get adopted in practical solutions and realistic application scenarios. High users' expectations in such scenarios require high dependability of MIR systems. For example, in view of the paradigm "getting the content I like, anytime and anyplace" the service of consumer-oriented MIR solutions (e.g., a PVR, mobile video, music retrieval, web search) will need to be at least as dependable as turning a TV set on and off. Dependability plays even a more critical role in automated surveillance solutions relying on MIR technology to analyze recorded scenes and events and alert the authorities when necessary.

This special issue addresses the dependability of those critical parts of MIR systems dealing with semantic inference. Semantic inference stands for the theories and algorithms designed to relate multimedia data to semantic-level descriptors to allow content-based search, retrieval, and management of data. An increase in semantic inference dependability could be achieved in several ways. For instance, better understanding of the processes underlying semantic concept detection could help forecast, prevent, or correct possible semantic inference errors. Furthermore, the theory of using redundancy for building reliable structures from less reliable components could be applied to integrate "isolated" semantic inference algorithms into a network characterized by distributed and collaborative intelligence (e.g., a social/P2P network) and let them benefit from the processes taking place in such a network (e.g., tagging, collaborative filtering).

The goal of this special issue is to gather high-quality and original contributions that reach beyond conventional ideas and approaches and make substantial steps towards dependable, practically deployable semantic inference theories and algorithms.

Topics of interest include (but are not limited to):

- Theory and algorithms of robust, generic, and scalable semantic inference
- Self-learning and interactive learning for online adaptable semantic inference
- Exploration of applicability scope and theoretical performance limits of semantic inference algorithms
- Modeling of system confidence in its semantic inference performance
- Evaluation of semantic inference dependability using standard dependability criteria
- Matching user/context requirements to dependability criteria (e.g., mobile user, user at home, etc.)
- Modeling synergies between different semantic inference mechanisms (e.g., content analysis, indexing through user interaction, collaborative filtering)
- Synergetic integration of content analysis, user actions (e.g., tagging, interaction with content) and user/device collaboration (e.g., in social/P2P networks)

Authors should follow the EURASIP Journal on Image and Video Processing manuscript format described at http://www.hindawi.com/journals/ivp/. Prospective authors should submit an electronic copy of their complete manuscripts through the journal Manuscript Tracking System at http://mts.hindawi.com/, according to the following timetable:

Manuscript Due	December 1, 2009
First Round of Reviews	March 1, 2010
Publication Date	June 1, 2010

Guest Editors

Alan Hanjalic, Delft University of Technology, 2600 AA Delft, The Netherlands; a.hanjalic@tudelft.nl

Tat-Seng Chua, National University of Singapore, Singapore 119077; chuats@comp.nus.edu.sg

Edward Chang, Google Inc., China; University of California, Santa Barbara, CA 93106, USA; echang@ece.ucsb.edu

Ramesh Jain, University of California, Irvine, CA 92697, USA; jain@ics.uci.edu