Issues Surrounding Moving a Successful Business Oriented Test Driven Development Process into the Environment of Real-time Embedded Systems

Michael Smith[1,2], Andrew Beaudin[3], James Miller[4], Craig Steinback[3], Syed Islam[1,5], Marc Poulin[3]

[1]*Department of Electrical and Computer Engineering, University of Calgary, Canada*
email:{ Mike.Smith, Isls} @ucalgary.ca

[2]*Department of Radiology, University of Calgary, Canada*
email:Mike.Smith@ucalgary.ca

[3]*Department of Physiology and Pharmacology, University of Calgary, Canada.*
email:{abeaudin, csteinba, poulin} @ucalgary.ca

[4]*Department of Electrical and Computer Engineering, University of Alberta, Canada*
email: jm @ece.ualberta.ca

[5]*Department of Electrical and Communication Engineering, Universiti Teanaga Nasional, Malaysia*
email: zahidul @uniten.edu.my

***Abstract*—Recent industrial surveys indicate that the development of embedded medical instruments is frequently undertaken by programmers with no formal training in testing who have been hired by firms with no formal software process. This is highly problematic given the need for 'defect-free' software in the medical field. The problem is compounded when the project requires software optimization to meet required real-time performance. We have identified two Agile methodologies, test driven development (TDD) and eXtreme-Programming (XP), as providing appropriate paths to the development of more reliable embedded systems. TDD, successfully used in the business world, essentially consists of 4 steps: (1) consulting with the customer to identify requirements, (2) writing tests to show when those requirements are met, (3) writing code to satisfy the tests and (4) adjusting the code and requirements now that further knowledge of the system is available. In the XP inspired life-cycle, we have described the advantages of repeating the TDD process as the medical device development proceeds from (A) envisagement through (B) algorithm prototyping in MATLAB and onto (C) initial implementation in C / C++ before (D) release of a real-time embedded product. In this case study we examine the advantages and disadvantages of applying TDD and the XP-inspired lifecycle when extending the capabilities of an existing echo Doppler imaging system to permit the determination of several blood flow indices. We propose an extension to the XP-inspired lifecycle to include the development and use of custom hardware, digital signal processing accelerators, when existing software optimization techniques are found to be inadequate.**

***Keywords* – eXtreme-Programming (XP) Inspired Software Development Process, Test Driven Development, Defect Free Development of Medical Devices, DSP accelerators**

I INTRODUCTION

Failure of medical devices has considerable business financial consequences with, unfortunately, related patient injuries commonly reported: e.g. [1, 2]. A recent survey of European firms [3] attempted to identify possible root causes for the failures. They found that only 50% of the 57 companies surveyed followed a defined software process. Further the majority of software developers in 64% of the companies had backgrounds which lacked the formal training in testing so necessary in producing reliable, 'defect free' medical devices.

Qiao [4] suggests that this problem exists, in part, because of the emphasis placed on what constitutes "important knowledge" during university training for biomedical engineers. For example, the "Biomedical Technology and Devices Handbook" [5], a comprehensive reference and biomedical engineering (BME) text book, contains 32 chapters on medical background and common techniques, but zero content on implementing clinical measurements in real devices. "The Biomedical Engineering Handbook" [6] was recommended for consultants, clinical engineers, undergraduate and BME graduate students in the *IEEE Engineering in Medicine and Biology*

Magazine [7]. This handbook includes eighteen instrument design examples but nothing about the development of the all important software that drives biomedical instruments today. Even when some focus is placed on software development, e.g. [8], it is typical that the suggested approach mirrors a more than 40 year old software process, the Waterfall model.

We have been reasonably successful in adapting some of the newer software processes successfully used by the business community into the realm of embedded system development. Our first attempt was the introduction into an undergraduate embedded system development course of one of the Agile methodologies, test driven development (TDD) [9]. We agree with Mugridge [10] that the probable reason for the success of this approach with undergraduates is that TDD is analogous to many aspects of the scientific process; setting hypotheses, doing experiments, and adjusting the process based on what has been discovered. Thus we were essentially just formalizing and simplifying the application of something with which the students were already familiar, rather than asking them to adopt something new.

We generalized this concept by proposing an eXtreme Programming (XP) inspired lifecycle to handle the development of biomedical devices from the initial proposal by the customer (clinical research team) to commercialization [11, 12]. Through the case studies presented in this paper, we intend to overcome one of the standard problems of moving a successful idea from one domain to another:

> *Everybody agrees that the concept is good, but there is insufficient mentorship to show how to actually apply the techniques in the new context.*

The remainder of the paper is organized as follows. Section 2 briefly describes this lifecycle, including a proposed extension to cover software / hardware co-design components. Section 3 provides a brief description of a software / hardware extension to an existing echo Doppler imaging system to permit the real-time evaluation of a number of blood-flow indices. Sections 4 and 5 provide a detailed case study on using the proposed lifecycle to produce the revised system described in Section 3. The paper concludes with a discussion of the practical issues in attempting to adapt this technique into the software-hardware co-design environment of medical device development.

II PRODUCT LIFE CYCLE FOR BIOMEDICAL DEVICES

The first stage of our proposed XP-inspired lifecycle [11, 12] is the envisioning of the embedded product. The customers or end users, described as domain knowledgeable personnel in [11], specify their ideas in terms of a combination of acceptance tests and user stories. Tools such as the frameworks for integrated testing (Fit / Fitnesse, [13]) are useful in allowing customers to produce executable test statements; but do not move cleanly into the new domain. In particular the current test vectors supported by the framework, business oriented uses of strings and numbers, are not particularly useful in handling signals and images.

Standard XP processes would now use the customer developed acceptance tests to drive a final production stage [14]. We take a different route with the Stage 1 tests now driving a high level prototyping second stage of the XP-inspired life cycle. Essentially we anticipate the development of an initial prototype requiring the identification of numerical algorithms that define the system's functionality followed by a MATLAB implementation. New test cases capture additional understanding of the system. However, while the domain technical specialist (e.g. BME graduate students) may have considerable experience in MATLAB code production, formalized testing is commonly an overlooked skill. In [12], we suggested that the availability of automated testing framework(s) capable of supporting TDD through all stages of the device development would assist in alleviating this issue.

Stage 3 involves moving the MATLAB code, and tests developed during Stages 1 and 2 into a high level compiled language such as C or C++. We originally completed the XP-inspired lifecycle with Stage 4: the development of the production system itself. We recognized that the system would need to be adapted to meet the final hardware environment with its limited memory resources and interactions with device specific peripherals. However we believed that this was essentially a software process involving: (A) rewriting inefficient programming constructs, (B) using DSP specific *#pragma* statement to direct architecturally aware compilers, and (C) rewriting critical components in low-level, highly customized machine code. We felt that the conflicts between 'refactoring code for speed' and 'refactoring code for maintainability' would be adequately handled by the wide range of tests developed during all four stages of the XP-inspired life cycle.

However, recent trends in microprocessor design depart from 'customized high-speed machine code' and towards 'on chip accelerators' which are specifically designed to offload many key DSP functions from the processor core. We are therefore proposing, for these newer processors, that a fifth stage be added to the medical device development lifecycle that is more targeted towards testing related to hardware / software co-design issues.

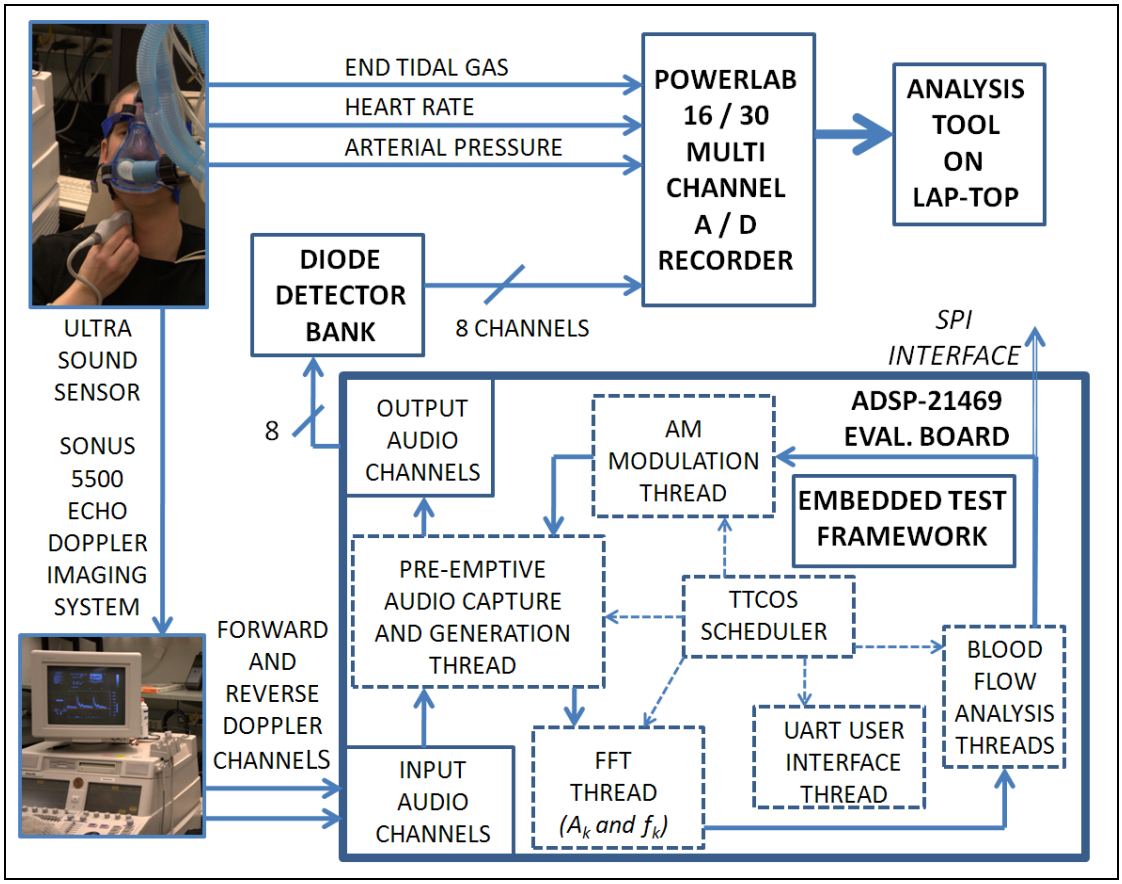

Figure 1: Schematic showing the SHARC ADSP-21469 based real-time ultra-sound analysis embedded system required to provide real-time blood velocity analysis of the Doppler shifted signals obtained by combining the forward and reverse signals outputted through the audio channels of the Sonos 5500 imaging system. A multi-channel AM modulation thread encodes the digital blood flow analysis information within 8 analog audio streams. These are processed by a diode detector bank before presentation to a multichannel ADInstruments PowerLab multi-channel A/D recorder which does not support any digital interface, e.g. the 4 wire SPI protocol support by the SHARC.

III Medical Device Description

A test subject is seated in a semi-supine resting position connected to a system used to manipulate the levels of oxygen and carbon dioxide the subject breathes in order to increase and decrease blood flow through different vascular beds (e.g. cerebral and brachial). Continuous measurements of end-tidal partial pressures of CO_2 and O_2, heart rate (3-lead electrocardiogram), and finger blood pressures are sampled by an ADInstruments Powerlab 16/30 A/D multi-channel recorder. The sampled signals are then subjected to multi-variate analysis by a custom written program running on a lap-top (Windows 7).

Figure 1 provides a schematic of the SHARC Ultra-Sound ANalysis system (SUSAN) which is used to provide a reliable combination of these existing measurements with new blood velocity indices from the right brachial artery calculated from the backscattered Doppler signals measured using a Philips Sonos 5500 clinical Echo Doppler Imaging System.

The ultra-sound signal from the Sonos 5500 sensor is reflected by the red blood cells moving through the monitored blood vessel. The backscattered signal is frequency down-shifted by the Sonos 5500 to produce two audio channels. One channel contains the Doppler shifted signal in the 0 to –10 kHz range related to velocity of blood moving towards the transmitting ultrasound probe (i.e., forward / positive blood velocity); and the second channel contains the 0 to +10 kHz signal related to blood flow away from the transmitting probe (i.e., reverse / negative blood velocity). The SUSAN system must capture and perform frequency analysis of these Doppler signals, before calculating a variety of blood flow related indices. These operations must be performed in real time for the blood flow information to remain synchronized with the other collected physiological measurements.

SUSAN was implemented on an Analog Devices ADSP-21649 SHARC evaluation board [15]. The key features of this board include two input and output eight audio channels and a high speed core capable of performing single-cycle SIMD floating point operations. There are a series of digital processing accelerators available to boost the system's

real-time capability; all designed into a low-cost evaluation board originally intended to demonstrate consumer home theatre applications. Software development was simplified because of the availability of 21469 versions of (A) a Time Triggered Co-Operative Scheduler (TTCOS [16]), and (B) an embedded testing framework [11, 12]; both developed in-house for embedded system courses and research applications at the University of Calgary [17].

Based on a preliminary MATLAB investigation (XPI-Stage 2), the SUSAN software architecture was constructed as follows:

(A) Capture of the Sonos 5500 forward and reverse frequency signals into audio buffers by a high priority (pre-emptive) thread;

(B) A digital signal processing thread to generate the amplitude (A_k) of each frequency component (f_k) from windowed versions of the audio buffers using a discrete Fourier transform implemented using software (XPI-Stage 4) or hardware variants of the fast Fourier transform (FFT) (XPI-Stage 5);

(C) Other analysis threads generated (i) a real-time physiologically accurate representation of the maximum frequency envelope ($f_{ENVELOPE}$) satisfying

$$\sum_{k=0}^{k=f_{ENVELOPE}} |A_k| < \quad frac \sum_{k=0}^{k=f_{MAX}} |A_k|$$

where *frac* is the range 0.7 to 0.95, (ii) a global power weighted average (PWA) waveform,

$$PWA = \sum_{k=-f_{MAX}}^{k=f_{MAX}} A_k^2 f_k \; / \sum_{k=-f_{MAX}}^{k=f_{MAX}} A_k^2$$

and (iii) an intensity weighted average (IWA) waveform evaluated by

$$IWA = \sum_{k=-f_{MAX}}^{k=f_{MAX}} |A_k| f_k \; / \sum_{k=-f_{MAX}}^{k=f_{MAX}} |A_k|$$

The processor has sufficient computational capacity to permit additional threads to calculate band-pass variants of all three measures to allow the tracking of blood flow signals with more than one key component, e.g. automatic removal of the signals from veins that may accidently be included in the ultrasound sensor's field of view;

(D) Another thread determines signal input power or power in specific frequency ranges. This calculation is used to avoid frequency indices equal in intensity to the maximum Doppler back scattered frequency. These can be introduced into the blood flow measurements whenever the blood flow changes direction which causes the calculation of the blood flow indices to become unreliable and noisy.

(E) Another thread encodes the blood flow indices as 8 amplitude modulated 32 kHz audio signals which are then decoded by a diode rectifier bank before being fed to the ADInstruments PowerLab multi-channel A/D converter. This novel analog approach was necessary as PowerLab does not support the transmission of blood flow indices over the digital communication channels available from the SHARC processor (SPI, etc).

(F) A final thread, using the SHARC UART, provided a user interface capable of (i) allowing adjustment of experimental parameters used during audio capture, audio generation and data analysis, (ii) activation of velocity calibration signals and (iii) the replay of known Doppler signals to provide standards against which signal-to-noise ratios and other important parameters could be monitored.

IV Discussion of Issues found in XPI - Development Stage 2 (MATLAB Analysis)

Automated unit testing frameworks to support the validation of MATLAB code are available (see [11, 12] for details). However in our opinion, these tend to remain in the realm of "concept good" with the tools only being demonstrated by the development of trivial practical applications. For example, one of the frameworks examined only allowed validation of floating point results via bit-wise comparison. This approach does not take into account the practical differences in the least significant bits when the expected values are compared with actual values calculated using a different software approach to the same algorithm.

A key mentoring issue that we struggle to overcome in our attempts to demonstrate how these frameworks can usefully support the application of TDD in a research environment is the following:

> *How does the developer avoid a pointless validation exercise where the code under development exactly mirrors the test code producing the "expected" test vectors?*

Validation of code using tests from a number of independent developers, e.g. student and supervisor, provides a possible solution to overcome this issue, albeit a time consuming one.

In our mind, despite these limitations, there remains a three-fold advantage of generating tests at this particular medical device lifecycle stage: (A) to formalize (record) the discussions between the "domain unfamiliar" graduate student developer and a supervisor / manager with the experience to recognize that "the current result does not make sense in this context", (B) provide the test vectors for Stage 3 and Stage 4 development, and (C) provide an executable design document to support the work of the "next" graduate student developer.

We consider optimizing the MATLAB code as inappropriate [11, 12]. However we recommend that a preliminary analysis of the expected system time requirements be made as soon as the code concepts are in place; even before the first early prototype is demonstrated to the customer. This contrasts with the standard XP approach to treat non-functional testing as a separate process to be undertaken upon finishing system production [18]. It is fairly straight to identify which part of the MATLAB

code is likely to consume the most cycles in a practical application. Digital signal processors permit dual access to multiple data banks (data stored in program and data memory) combined with simultaneous multiplications and additions with instructions executing in 0.5 to 2 processor cycles. This simplicity permits order of magnitude calculations to be performed that quickly determine which processor resource is likely to cause a bottleneck. Poor algorithms, unlikely to permit real-time operations, can then be identified before unnecessary development time is wasted.

V Discussion of Issues found in XPI Development Stages 3 and 4 (MATLAB to C conversion and Product Release)

Translating MATLAB's floating point calculations to run on low-cost integer processors is a general issue that requires developers to have practical knowledge on issues such as signal scaling. Given that such concepts will not form part of the core courses taken by most BME students, we have recently adopted a pragmatic process to solving this issue. We rely on floating point emulation, and anticipate meeting real time constraints given the high speed of current processors which have both superscalar and SIMD capability. These provide multiple ALU operations per instruction cycle, which can be combined with special (high speed) floating point number representations that meet the practical accuracy requirements associated with 16-bit or 24-bit sensors.

During Stage 3 of the XP-inspired life-cycle, the development environment's statistical profiler provides a second approach to determining whether the proposed algorithms has the potential of being implemented in real time after identifying practical optimizations (Stage 4). If this initial analysis indicates that real-time constraints will not be met, then the prior adoption of a TDD process (in particular test availability) provides many reliable routes for changing ("refactoring for speed") the algorithm implementation. For example, current processor development environments commonly include digital processing "C" and "C++" language extensions which make it (reasonably) straightforward to rework the floating point tests and code to use the native 'DSP processor' signed-signed-fractional (SSF) integer representations.

However, given the changing capabilities and lowering cost of processors and evaluation boards suitable for research applications, combined with the increasing similarity of embedded system development environments, we prefer a simpler approach. With the processor families from Analog Devices and Texas Instruments, we have successfully taken the developed code, co-operative scheduler, tests and testing framework and recompiled on a faster processor. Often this approach requires no more than changing of processor specific header files [12, 13, 19] and code movement is greatly simplified by the availability of a common GUI interface across a manufacturer's family of integer and floating point processors.

VI Discussion of Development Issues found in Stage 5 (Accelerator Development)

The initial prototype of our XP-inspired development lifecycle was conceived around 2003. At that time, we assumed that the fourth and final lifecycle stage would involve optimal matching of the medical device's software architecture to the processor's architecture. We anticipated using software methods to activate custom features in the optimizing compiler and assembler [11, 12]. However, with processor technology changing so rapidly, this assumption must be re-examined.

In the early years of the last decade, a developer would have expected to spend considerable time "rewriting critical code components in low-level, highly customized machine code". However, it is now becoming more common for the manufacturer of 'medium-level entry' processors to provide on-chip DSP accelerators to off-load intensive calculations from the main processor core. For the Analog Devices SHARC 21XXX family of processors used in the ultra-sound post-analysis tool (SUSAN) discussed in this paper, these accelerators are capable of supporting FIR, IIR and FFT calculations driven by chained DMA data activities with no core intervention. We have performed an initial examination of how we might extend our XP-inspired lifecycle to include interactions between with existing software produced from XPI Stage 4 and the accelerators. We conclude that this stage would have many similarities with, and associated problems from, introducing new code and tests into a legacy software system [20].

However the Stage 5 questions that require answering would be very different if the SUSAN application was moved onto a system where processors and DSP accelerators are entirely based on a customizable FPGA chip, e.g. Microblaze [21]. We are currently investigating a related, but less complicated, situation of designing an accelerator to support the reliable merging of one software thread onto a series of existing threads supported by a pre-emptive scheduler found on many real-time systems. Unlike the co-operative scheduler used in controlling SUSAN, merging threads on the more common pre-emptive schedulers can lead to data races and other resource conflict issues between the new and existing threads [16]. We are in the early stages of developing an accelerator to improve our current capability of 'hardware assisted' data watch activities [22] used to identify possible data race threats with lower overhead than software instrumentation approaches. We are investigating whether there would be practical advantages of using a TDD approach that allows the accelerator to be developed incrementally. We are in the early stages of developing an automated testing framework capable of supporting a working accel-

erator whose functionality is an ever changing software / hardware combination.

VII Conclusion

Recent industrial surveys indicate that medical devices are often developed by personnel with considerable domain knowledge but that the reliability of the device is compromised as the developer has no formal training in testing. Two Agile methodologies, test driven development (TDD) and eXtreme-Programming (XP) were identified as providing appropriate paths to the development of more reliable embedded systems. In this paper, we have provided a case study involving the construction of a real-time analyser for signals from an Echo Doppler Imaging system to demonstrate how to overcome the practical issues of using TDD and XP concepts with embedded systems.

VIII Acknowledgements

Financial and in-kind support provided through an industrial collaborative research and development grant from Analog Devices, CDL Systems and Natural Sciences and Engineering Research Council (NSERC) of Canada: CRD-365295. Additional support provided by the University of Calgary, Canada. Thanks to the 2011 class of ENCM515 students at the University of Calgary for useful discussions associated with the development of optimized DSP algorithms for embedded systems. M. Smith is the 2011 Analog Devices University Ambassador.

References

[1] Baxter International, "Baxter's February 2005 Corrective Action for COLLEAGUE Infusion Pump Receives FDA Class I Designation." vol. 2009, 2005.

[2] FDA, "Class 1 Recall: Medtronic SynchroMed EL Programmable Infusion Pumps." vol. 2009, 2008.

[3] R.L.Feldmann, F.Shull, C.Denger, M.Host, C.Lindholm, "A Survey of Software Engineering Techniques in Medical Device Development," *Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability, HCMDSS-MDPnP.*, pp. 46-54. 2007.

[4] J.Qiao, "Application of Test-Driven Development to Biomedical Algorithms", M. Sc. thesis, Electrical and Computer Engineering, University of Calgary, Calgary, Alberta, Canada, 2010.

[5] J.Moore, G.Zouridakis, "Biomedical Technology and Devices Handbook" CRC, 2003.

[6] J.D.Bronzino, "The Biomedical Engineering Handbook", CRC/Taylor & Francis, 2006.

[7] P.H.King, "Book review for The Biomedical Engineering Handbook," *IEEE Engineering in Medicine and Biology,* 1996.

[8] J.P.O'Leary, "Medical Product Design," in *Standard handbook of biomedical engineering & design*, M. Kutz, Ed. New York: McGraw-Hill, 2002, pp. 19.3-19.19.

[9] J.Miller, M.R.Smith, "A TDD Approach to Introducing Students to Embedded Programming", *Proceedings of the 12th Annual Conference in Innovation and Technology in Computer Science Education*, pp. 33 – 37, ITiCSE 2007, Dundee, Scotland, U.K, June 25-27, 2007.

[10] R.Mugridge, "Test driven development and the scientific method," *Proceedings of the Agile Development Conference, 2003*, 2003, pp. 47-52.

[11] M.R.Smith, J.Miller, F.Huang, A.Tran, "A case for taking a more 'Agile' approach in the development of embedded systems", *IEEE Software Magazine (Special issue on Embedded Software),* pp 50 – 57, May 2009.

[12] M.R.Smith, J.Miller, S.Daeninck, "An Embedded Test oriented production methodology", *Journal of Signal Processing Systems*, Vol. #56:1, pp 69 – 89, 2009, Appeared on-line 2008.

[13] W.Cunningham, "Framework History," *FIT: Framework for Integrated Tests*, 2002, fit.c2.com/wiki.cgi?FrameworkHistory.

[14] K.Beck, M.Beedle, A.v.Bennekum, A.Cockburn, W.Cunningham, M.F.J.Grenning, J.Highsmith, A.Hunt, R.Jeffries, et al., "Manifesto for Agile Software Development." vol. 2008, 2001.

[15] Analog Devices SHARC processors *www.analog.com/en/embedded-processing-dsp/sharc/products/index.htmlpages.* Accessed 5th April, 2011.

[16] M.J.Pont, "Patterns for time-triggered embedded systems: Building reliable applications with the 8051 family of microprocessors", Addison Wesley, 2005.

[17] M.R.Smith, "ENCM511 and ENCM515 embedded system development courses", www.enel.ucalgary.ca/People/Smith, accessed 5th April 2011.

[18] K.Beck, *Test-Driven Development—By Example*, Addison-Wesley, 2002.

[19] M.R.Smith, "Demonstration of Embedded UnitTest++ running on a variety of processors". www.enel.ucalgary.ca/People/Smith/2010webs/encm511_10/10Labs/10AdditionalMaterial.htm, accessed 5th April, 2011.

[20] M.C.Feathers, "Working Effectively with Legacy Code", Prentice Hall Professional Technical reference, Upper Saddle River, 2005.

[21] Xilinix FPGA Microblaze processor, direct.xilinx.com/tools/feature/csi/microblaze.htm, accessed 4 April, 2011.

[22] F.Huang, M.R.Smith, A.Tran, J.Miller, "E-RACE: A Hardware-Assisted Approach to Lock-set-Based Data Race Detection on Embedded Products", *19th International Symposium on Software Reliability Engineering*, Seattle, USA, Nov. 2008.