FPGA Design for Monitoring CANbus Traffic in a Prosthetic Limb Sensor Network

A. Bochem*, J. Deschenes*, J. Williams*, K.B. Kent*, Y. Losier+
Faculty of Computer Science* Institute of Biomedical Engineering+
University of New Brunswick
Fredericton, Canada
{alexander.bochem, justin.deschenes, jeremy.williams, ken, ylosier}@unb.ca

***Abstract*—This paper presents a successful implementation of a Field Programmable Gate Array (FPGA) CANbus Monitor for embedded use in a prosthesis device, the University of New Brunswick (UNB) hand. The monitor collects serial communications from two separate Controller Area Networks (CAN) within the prosthetic limb's embedded system. The information collected can be used by researchers to optimize performance and monitor patient use. The data monitor is designed with an understanding of the constraints inherent in both the prosthesis industry and embedded systems technologies. The design uses a number of verilog logic cores which compartmentalize individual logic areas allowing for more successful validation and verification through both simulations and practical experiments.**

keywords: FPGA; CANbus; Data Monitor; Prosthesis; Verilog

I. Introduction

In the prosthetics field, various research institutions and commercial vendors are currently developing new microprocessor-based prosthetic limb components which use a serial communication bus. Although some groups' efforts have been of a proprietary nature, many have expressed interest in the development of an open bus communication standard. The goal is to simplify the interconnection of these components within a prosthetic limb system and to allow the interchangeability of devices from different manufacturers. This initiative is still in development and will undoubtedly face some obstacles during its development and implementation as there are currently no embedded devices available to reliably monitor the bus activity for the newly developed protocol.

The open bus communication standard uses the CAN bus protocol as its underlying hardware communication platform. Higher levels of the protocol define the initialization, inter-module communication, and data streaming capabilities. Commercially available off-the-shelf CANbus logic analyzers, although capable of decoding the primary CAN fields, are unable to interpret the protocol messages in order to provide detailed information of the system behavior. The design of an FPGA-based prosthetic limb data monitor will allow embedded system engineers to monitor the new protocol's communication activity occurring in the system. This provides an effective developmental tool to not only help develop new prosthetic limb components but also advance the open bus standard initiative. This monitor will help develop new prosthetic components as well as provide a means to assess their rehabilitation effectiveness. The design of the system will be flexible enough to meet future needs and follow current standards, such as simplifying the work required for end users to utilize the system. Furthermore, the monitor's data logging capabilities will allow the prosthetic fitting rehabilitation team to analyze the amputee's daily use of the system in order to assess its rehabilitation effectiveness. The evaluation of the data monitor's capabilities will be performed in conjunction with UNB researchers who are leading members of the Standardized Communication Interface for Prosthetics forum [1].

Section 2 of the report outlines the field of biomedical engineering and presenting an overview of related work. Section 3 gives an introduction into the CANbus standard. Section 4 covers the system design and its implementation. Section 5

"978-1-4577-0660-8/11/$26.00

shows how the functionality of the system has been tested and evaluated. Finally, Section 6 concludes the paper.

II. Embedded Systems

Within this section we will look at traditional embedded system and biomedical engineering projects. It will highlight the characteristics of some approaches with the applied technology and projects that implement CANbus communication in particular. Initially, robots were controlled by large and expensive computers requiring a physical connection to link the control unit to the robot. Today the shrinking size and cost of embedded systems and the advances in communication, specifically wireless methods, have allowed smaller, cheaper mobile robots. Robots operate and interact with the physical world and thus require solutions to hard real-time problems. These solutions must be robust and take into account imperfections in the world. These autonomous systems usually consists of sensors and actuators; the sensors collect information coming into the system and the actuators are the outputs and can be used to interact with the outside environment. The signal and control data is sent to a central processing unit which runs the main operating system. For reducing power consumption and complexity, these sensor networks use communication buses to exchange data.

In Zhang et al. [2] the researchers started with a typical robotic arm setup which included a number of controllers and command systems communicating through a communication bus to one another. The researchers, who consider the communication system the most important aspect of the space arm design set out to improve it. Their system utilized the Controller Area Network (CAN) communication bus and enhanced reliability through the implementation of a redundant strategy. The researchers cited features of the CANbus that are desirable, which include: error detection mechanisms, error handling by priority, adaptability and a high cost-performance ratio. In order to increase reliability, they implemented a "Hot Double Redundancy" technology. This technology was implemented as the communication system which consisted of an ARM microprocessor, the CANbus controller circuit, data storage, system memory and a complex programmable logic device (CPLD) used to implement a redundant strategy. The CPLD interfaced with two redundant CAN controller circuits, which were each connected to their own set of system devices, while taking commands from the main microprocessor. The logic to handle the "Hot" aspect of the technology, the ability to switch from one system to the other without any down time, was implemented in the hardware definition language VHDL and put onto the CPLD. This increases the redundancy, but also significantly improves the reliability by handling major system faults without down time and increases the flexibility of the design by having hardware components which can be updated and changed without physical contact. The process was tested through the Quartus II software tool from Altera, which simulated normal and extreme activity for a period of sixteen to thirty-two hours all the while alternating between the two redundant systems. The researchers reported the tests to be successful, with a 100% transmission rate and no error frames or losses due to the redundancy switch time.

Biomedical engineering is an industry which requires a multi-disciplinary skillset in which engineering principles are applied to medicine and the life sciences. In recent years renewed interest from the American military has promoted the advancement of artificial limb technology. In 2005, the Defense Advanced Research Projects Agency (DARPA) launched two prosthetic revolution program, revolutionizing prosthetics 2007 (RP2007) and revolutionizing prosthetics 2009 (RP2009) [3]. The goal of RP2007 was to deliver a fully functional upper extremity prosthesis to an amputee, utilizing the best possible technologies. The arm is to have the same capacity as a native arm, including fully dexterous fingers, wrist, elbow and shoulder with the ability to lift weight, reach above one's head and around one's back. The prosthetic will include a practical power source, a natural look and a weight equal to that of the average arm. Another, more difficult goal includes control of the prosthesis through use of the patient's central nervous system. In RP2009 the prosthesis technology will be extended to include sensors for proprioception feedback, a 24 hour power source, the ability to tolerate various environmental issues and increased durability. Although the prostheses developed during the course of both projects proved to be technological achievements [4], it is still unclear whether the cost of these systems, aimed at being around $100,000[5], will prohibit their use when they become commercially available.

The University of New Brunswick's (UNB) hand project [6] seeks to design a low-cost three axis, six

basic grip anthropomorphic hand, with control of the hand using subconscious grasping to determine movement.The UNB hand team built intelligent Electromyography (EMG) sensors [7] that could amplify and process signal information, passing required information to the main microprocessor through the use of serial communication bus. This allows for a reduction in wiring, which reduces the weight and simplifies the component architecture. The serial bus chosen, the Controller Area Network bus (CANbus), allowed a power strategy to be implemented, reducing overall power consumption. The CANbus is also noted as having a good compromise between speed and data protection, a necessity for prosthetics [8]. The hand project creators have begun creating a communication standard [9] to improve interchangeability and interconnection between limb components. If adopted, major increases in flexibility would be gained which would be beneficial to all people involved in the prosthesis industry.

The paper by Banzi Mainardi and Davalli [8] extends the idea of a distributed control system and Controller Area Network serial bus to an entire arm prosthesis. In this project, along with the parallel distribution cited in the UNB hand project, the device had the additional task of handling external communication through either a Bluetooth or RS232 serial connection. The paper also cites similar reasons in using the CANbus to the UNB hand project, adding evidence of successful device integration in the CANbus' traditional area, automotives and how this could parallel the prosthetic industry. The paper also outlines reasons behind not choosing other communication protocols or technologies. The two initial choices, Inter-Integrated Circuit (I^2C)[10] and Serial Peripheral Interface (SPI) Bus[11], were rejected because they failed to have adequate data control and data protection systems and were unable to handle faults acceptably. The SPI system required additional hardware overhead which would allow for device addressing. The personal computing standards and industry standards were unable to adapt to the space and weight constraints required by the profile of the prosthesis. The CANbus allowed for a reasonable number of sensors, flexibility in expansion and interfacing, microcontrollers with integrated bus controllers and efficient, robust, deterministic data transmission with a reduction of cables required and near optimal voltage levels.

III. CANBUS

The CANbus is a serial communication standard used to handle secure and realtime communication between electrical and microcontroller devices and primarily defines the physical and data link layers of the open systems interconnection model (OSI model). The CANbus was originally created to support the automotive industry and its increased reliance on electronics; however because of its reliability, high speed and low-cost wiring, it has been used in many additional areas. The CANbus supports bit rates of up to 1Mbps and has been engineered to handle the constraints of a security conscious real-time system[12]. The physical medium is shielded copper wiring, which utilizes a non-return-to-zero line coding. The CAN message frame is either an 11 bit identifier base frame (CAN 2.0A) or the 29 bit identifier extended frame (CAN 2.0B). There are a number of custom bit identifiers of which most are used to synchronize messages, perform error handling or signal various values. The CAN standard operates on a network bus therefore all devices have access to each message, addressing is handled by identifiers in each message frame. The CANbus standard outlines the arbitration method as CSMA/BA where the BA stands for bit arbitration. The bit arbitration method allows any device to transmit its message onto the bus. If there is a collision the transmitter with the greatest priority, identified by the most successive recessive bits wins bus arbitration. The lesser priority devices then standoff for a predefined period of time and re-transmit their message. This allows the highest priority messages to get handled the fastest. Other important properties defined in this standard are:

- Message prioritization - Critical devices or messages have priority on the network. This is done through the media arbitration protocol.
- Guaranteed latency - Realtime messaging latency utilizes a scheduling algorithm which has a proven worse case and therefore can be reliable in all situations[13].
- Configuration flexibility - The standard is robust in its handling of additional nodes, nodes can be added and removed without requiring a change in the hardware or software of any device on the system.
- Concurrent multicasting - Through the addressing and message filtering protocols in place, the CAN-bus can have multicasting in which it is guaranteed that all nodes will

accept the message at the same time. Allowing every node to act upon the same message.
- Global data consistency - Messages contain a consistency flag which every node must check to determine if the message is consistent.
- Emphasis on error detection - Transmissions are checked for errors at many points throughout messaging. This includes monitoring at global and local transmission points, cyclic redundancy checks, bit stuffing and message frame checking[12].
- Automatic Retransmission - Corrupted messages are retransmitted when the bus becomes idle again, according to prioritization.
- Error distinction - Through a combination of line coding, transmission detection, hardware and software logic the CAN-bus is able to differentiate between temporary disturbances, total failures and the switching off of nodes.
- Reduced power consumption - Nodes can be set to sleep mode during periods of inactivity. Activity on the bus or an internal condition will awaken the nodes.

IV. System Design

The open bus standard-based data monitor system for Prosthetic Limbs captures and collects the serial information from two separate Controller Area Network (CAN) buses, the sensor bus and the actuator bus. With the collected information it then would transform the data into the correct CAN message format. A timestamp would be added and the messages would be passed through a user controlled filter to dictate which messages should be logged. After the filter the two buses' messages are merged and sorted according to their timestamp. Once sorted, the information is then sent to an output device for further processing. The current design allows to choose between three different output interfaces. First, the RS232 serial interface that sends the CAN message data encoded in ASCII format to the serial port using a RS232 chip on the DE2 board. Second, the direct connection of the RS232 serial interface module with the pin interface of the DE2 board. This allows the transmission with higher bandwidth, using an external RS232 chip with better performance. And the third communication module uses the USB interface of the DE2 board to transmit the CAN message data [14]. This project will allow the engineers to observe the communication on the buses and search for sources of errors. An overview of the system design is given in Figure 1.

The implementation consists of various "working" cores, that can be individually tested each containing one piece of the functionality which is required for the overall system. These cores interface with one another through a standard FIFO, alleviating timing issues. The FIFO modules are dual-clocked memory buffers that work on the first-in/first-out concept. With the dual-clock ability, those cores can be used to exchange data between two different modules in a hardware design, even if those modules run with different clock speeds. The FIFO cores belong to the Intellectual Property (IP) cores library, that is available within the development environment Quartus II from Altera. The usage of those cores is usually free for academic use, but requires a license fee for industrial or end-consumer development purposes.

The module "CAN Reader" is handling the observation of the connected CANbus. It receives all messages that are transmitted without causing a collision on the bus and forwards it to a FIFO buffer, from where the "Filter" module gets its input data. One pair of the "CAN Reader " and the "Filter" are connected to the control bus while the other pair is listening to the sensor bus. The modular design would allow to connect more or fewer CANbus' to the monitoring system. In the current implementation of the "Message Writer" module, only the data transmission on the RS232 interface is implemented. The data compression of the messages and the application of different interface technologies has been evaluated during the project. Their implementations have been postponed as future work. RS232 was implemented as it required the least overhead to establish a communication channel [15].

The design of the "CAN Reader" module is based upon the "CAN Protocol Controller" project by Igor Mohor from opencores.org. This design implements the specification for the "SJA100 Standalone CAN controller" from Philips Semiconductors [16]. To allow the integration of the core design, an I/O module that handles the Wishbone interface had to be implemented. The configuration of the CAN controller is working register based, as defined in the specification of the SJA1000. The created I/O module is used by the "CAN Reader" that is configuring the "CAN controller". The "CAN controller" receives the data from the CANbus, while the "CAN Reader" collects the messages from the "CAN controller" and forwards

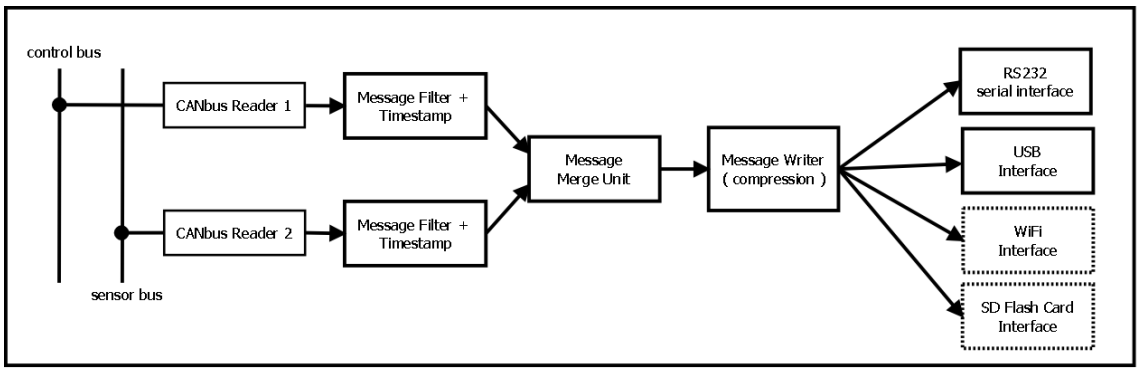

Fig. 1: System design of CANbus monitor.

them to the timestamp and filter procedures. For a manageable processing flow of the modules their internal control has been designed as state machines. This design concept allowed the localization of problems caused by signal runtimes and race conditions. Figure 2 gives an idea of the state machine that describes the functionality of the "CAN Reader module".

To give an idea of the design complexity, the state "Read_Receive_Buffer" is using the I/O module to communicate with the "CAN controller". This leads to a design with state machines encapsulated in state machines. Such designs should be avoided in hardware if possible, since they tend to obscure runtime race conditions. In the current design the occurence of race conditions is probhibited by event driven mutexes.

V. Validation & Verification

The validation of the system design has been done by simulation and runtime experiments on the target platform. For verification of the functionality the single modules have been simulated with Altera's ModelSim tool. This allowed a step-wise execution of the Verilog code to identify logic errors in the implementation. Afterwards the modules have been tested on the target platform, with testbench modules providing the required input data. The output of the tested modules has been transmitted on the RS232 interface to a connected computer and verified by hand. The timing verification of the hardware design has been done by an experimental test setup, with two sensor nodes on one connected CANbus (Figure 3).

The two sensor nodes were configured to send messages on the bus continuously at 250Hz frequency. To allow the verification that all messages can be received and no message is lost, each message contained a set of four consecutive numbers. The first node was configured to send numbers in the range from 0 to 999. The second node was sending numbers in the range from 1000 to 1999. Due to the CANbus standard, each node would resend its current message until it receives an acknowledge response on the bus from the other node. This would cause the node to increment the numbers for the next message. If a collision on the bus has occurred, the CANbus protocol standard ensures that the transmitting node is informed by a global collision signal. This would be sent by the first node on the CANbus, which detects the collision.

The analysis of the system lead to the conclusion that the hardware design is working correctly. All messages that have been sent from both nodes on the CANbus could be received successful and with correct data by the "CAN Reader" module in the FPGA design. This could be verified with the logged data from the output of the serial module in the hardware design. Since the test setup only had one CANbus the input pin has been used twice to evaluate the system with two CANbuses connected. This allowed to verify the proper ordering of the messages by the "Message Merge Unit". It turned out that the only flaw in the current design is the RS232 interface. The available chip on the DE2 board has a maximum bandwidth of 120kbit/s. That becomes the bottleneck if the connected CANbus runs at maximum bandwidth of 1Mbit/s and even more for two connected CANbuses. The design for an USB module as communication interface has been created but could not be sufficiently evaluated for further usage.

The analysis of the test results showed some

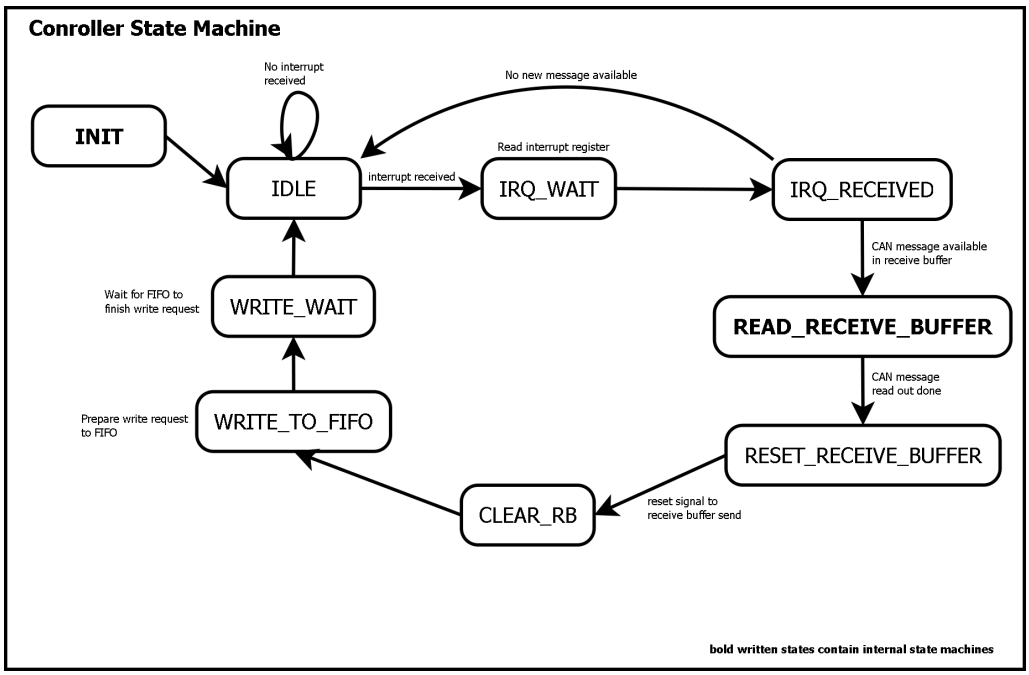

Fig. 2: State machine design for ”CAN Reader” module.

unexpected behavior. While the messages of one node had been received completely, some messages of the other node were missing. By resetting the CANbus system, this effect could switch to the other node or stay the same. For either one of the two nodes some messages were missing, but never for both nodes at the same test execution. The extensive analysis of the system design leads to the conclusion that this error might be caused by the configuration of the sensor nodes. Further evaluation of the system design will be continued, after a new test setup can be provided by the project partner.

VI. Future Work

It might be useful to have the ability running the CANbus monitoring design without a direct connection to a computer. The logged data could be stored on an integrated flash memory module. For this feature it would be helfpul to have a compression module to increase the systems mobility. The compression would need to be fast enough so that it does not become a bottleneck for the system. Ideally the core would take in a stream of data and output a stream of compressed data. This could plug directly into the current system design.

It would be useful to have wireless functionality so that the prosthesis does not need to be tethered to a computer to retrieve the logged data. The biggest hurdle to overcome is to understand how to communicate with the wireless controller on the hardware level. Existing solutions in open source projects could build a starting point here.

At the moment all the values for the CAN controller are hard coded and thus can not be changed after the design has been synthesized. This could be improved in several ways ranging from full configurability at runtime to a bit of code reorganization so that the values can easily be changed for re-synthesis. A compromise could be the ability to configure a small number of values at runtime. The most time effective solution would be to reorganize the code so that the hard coded values are excluded into a configuration file.

The filtering is currently fairly rudimentary and inflexible. It would be advantageous to have more flexibility with how to filter messages. Reconfiguring the filters, and enable or disable filters during runtime would be helpful. This becomes even more crucial as the design moves from lab testing to real world testing where re-synthesizing the design becomes less feasible. The system could potentially be a memory mapped device and masks could be stored in registers which could be written to by a microcontroller. A simpler solution would be to have several kilobytes of persistent memory, to

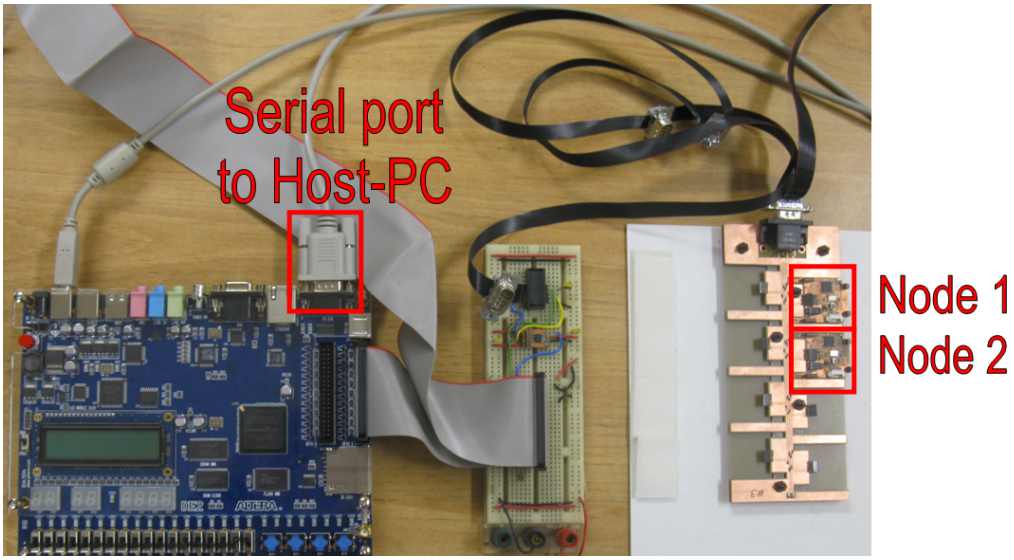

Fig. 3: Experimental test setup for final system verification.

which masks could be written, to be used during the filtering of the messages.

VII. Summary

This paper has shown the successful implementation of a monitoring system for a CANbus sensor network in a hardware design on an FPGA development board. Details of successful projects and related work has been introduced. The information that has been displayed in this paper should offer a good starting point, providing a general understanding of embedded projects, with emphasis on actual biomedical engineering solutions and basics of the CANbus standard. The modular design of the approach allows the application in different projects that have a need for a CANbus monitoring system. All project source code will be made available through opencores.org.

Acknowledgments

This work is supported in part by CMC Microsystems, the Natural Sciences and Engineering Research Council of Canada and Altera Corporation.

References

[1] *Standardised Communication Interface for Prosthetics*. [Online]. Available: http://groups.google.ca/group/scip-forum/

[2] J. Yang, T. Zhang, J. Song, H. Sun, G. Shi, and Y. Chen, "Redundant design of a can bus testing and communication system for space robot arm," in *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, Dec. 2008, pp. 1894–1898.

[3] DARPA, *DARPA: 50 Years of Bridging the Gap*, 1, Ed. Defense Advanced Research Project Agency, 2008.

[4] L. Ward. (2007, October) Breakthrough awards 2007 - darpa-funded proto 2 brings mind control to prosthetics. electronic / periodical. Popular Mechanics.

[5] S. Adee. (2008, February) Ieee spectrum: Dean kamen's "luke arm" prosthesis readies for clinical trials. electronic. IEEE Spectrum.

[6] A. Wilson, Y. Losier, P. Kyberd, P. Parker, and D. Lovely, "Emg sensor and controller design for a multifunction hand prosthesis system - the unb hand," draft document, 2009.

[7] Y. G. P. P. A. L. D. F. Wilson, Adam W. Losier, "A bus-based smart myoelectric electrode/amplifier," in *Medical Measurements and Applications Proceedings (MeMeA), 2010 IEEE International Workshop on*, 2010.

[8] S. Banzi, E. Mainardi, and A. Davalli, "A CAN-based distributed control system for upper limb myoelectric prosthesis," in *Computational Intelligence Methods and Applications, 2005 ICSC Congress on*, Istanbul,.

[9] Y. Losier and A. Wilson, "Moving towards an open standard: The unb prosthetic device communication protocol," 2009.

[10] *The I2C-BUS Specification*, Philips Semiconductors Std. 9398 393 40 011, Rev. 2.1, January 2000.

[11] Motorola, *M68HC11 Microcontrollers Reference Manual*, rev. 6.1 ed., Freescale Semiconductor Inc., Mai 2007.

[12] *CAN Specification Version*, Robert BOSCH GmbH Std., Rev. 2.0, September 1991.

[13] J. Krákora and Z. Hanzálek, "Verifying real-time properties of CAN bus by timed automata," in *FISITA, World Automotive Congress, Barcelona*, May 2004.

[14] *DE2 Development and Education Board - User Manual*, 1st ed., Altera, 101 Innovation Drive, San Jose, CA 95134, 2007.

[15] *MAX232, MAX232I DUAL EIA-232 DRIVERS/RECEIVERS*, Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, March 2004.

[16] *SJA1000 Stand-alone CAN controller*, Philips Semiconductors, 5600 MD EINDHOVEN, The Netherlands, January 2000.