An Overview of Embedded Systems and Prosthesis Technology

by

A. Bochem, J. Deschenes, J. Williams
and K. B. Kent

TR 10-205, June 15, 2010

Faculty of Computer Science
University of New Brunswick
Fredericton, NB, E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
Email: fcs@unb.ca
http://www.cs.unb.ca

Contents

List of Figures

1 Introduction

This report communicates the technologies, methodologies and considerations necessary to understand when working on embedded system's, specifically within the biomedical engineering field. Biomedical engineering is a multi-disciplinary field which requires an understanding of mechanical engineering, electrical engineering, biology, math, system design and many others. Within this field there are numerous opportunities for computer scientists to apply their skillset. One such area is in the analysis, design and implementation of an embedded system and its features. This report presents an overview of embedded system projects in extreme situations such as the fields of biomedical engineering and space exploration. Then a synopsis of the tools and methods used to design and analyze an embedded system is produced. Finally a summary of the various components that make up an embedded system including hardware and communication technologies is outlined.

This report was written to provide a single document which encapsulates the components and thought processes required to design within the domain of embedded systems, with a biomedical engineering slant. The overview of the various fields presents an unbiased look at technologies allowing individuals new to embedded system projects to learn about concerns and issues they will have to deal with. This document provides enough information not only to help readers make an informed choice of technologies, but also practical information on what is available in an embedded system design suite and how it can be used effectively. Additionally examples of embedded system projects are highlighted, giving readers practical insight and exposure to real-world projects.

The first section of the report outlines the field of biomedical engineering, presenting an overview of ongoing and completed embedded system projects emphasizing the University of New Brunswick's advanced prosthetic hand project. The next section discusses methodologies and techniques that allow for adequate analysis, design and testing of an embedded system. This includes an overview of Altera's software and design suite, Quartus II. The rest of the report is focused on properties and components of embedded systems. This includes first an overview of the main hardware components used in embedded systems, programmable logic devices, application specific integrated circuits and microprocessors. The second technology overview sections focuses on communication. This includes communication techniques, a high-level presentation of properties that make up communication protocols and an overview of three communication protocols.

2 Embedded Systems Projects

Embedded systems have many possible applications, they are used in numerous industries and allow for solutions to practical problems. Within this section we will look at traditional embedded system and biomedical engineering projects. Highlighting approaches, techniques and projects which apply the use of embedded technologies. The traditional areas of embedded systems focus naturally on automotive and robotic systems, the biomedical projects are often centered around advanced prosthetics. Through an overview of projects within both sections the hope is a pattern of useful information on embedded systems will emerge.

2.1 Traditional Projects

With embedded systems being prevalent in a number of areas, the idea of a traditional embedded system is hard to define. In this section, the focus is on areas which have adopted embedded system technologies relatively early in their life cycle. The robotics field is an obvious early adopter with a natural mapping of robotic subsystems to embedded systems, communication protocols would then be used to allow a system wide sharing of information. Another fields which relies heavily on embedded systems is the automotive industry. The automotive industry has increased its use of embedded systems to control vehicle systems. The average vehicle in 2007 had 80 embedded systems[21], a strong indicator of the maturity of embedded systems in this field.

2.1.1 Robotics

Robotics has come a long way, technological improvements in numerous aspects of computing has allowed for smaller, more reliable and more useful robots. 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 has allowed smaller, cheaper mobile robots. Robots operate and interact with the physical world requiring solutions to hard, real-time problems, requiring a solution that is robust and takes into account the imperfections it will encounter. The usual method in which robots are designed is the use of more powerful microprocessors to analyze the results communicated to it by numerous embedded system devices. The embedded devices represent subsystems used to handle one aspect of the overall design. The embedded devices usually consists of sensors and actuators, the sensors represent information coming into the system and the actuators are the outputs and can be used to interact with the outside environment. The embedded devices would analyze the signals and pass along important information to the processing units, the processing units then can act on the information, communicating commands to the

devices. The devices would then process the command and attempt to carry out the commands through specific actuator actions. Robotic systems require real-time processing, concurrent and reliable communication and agile solutions. Additional requirements stem from the environment and conditions that surround the system.

One environment that contains an interesting set of problems is space. The use of "space arms" developed by Canada and other countries have a number of additional concerns. Some of the concerns include:

- Redundancy - Repairing aspects of the devices is extremely expensive or impossible once deployed. The system must be designed with communication and component redundancies. This enhances the reliability of the system and allows for multiple component failure.

- Reliability - Internal communication must be reliable and fast, subpar speed or reliability could irreparably damage the device. Situations that could potentially damage the device must be handled quickly regardless of other situations occurring.

- Flexibility - The system must be flexible in its execution of objectives and in its ability to determine the potential for errors. The system must be designed to understand what it can do, what will be harmful and how it should recover from bad situations.

- Reporting - A strategy for collecting necessary information must be decided upon. Procedures for use of the information, the amount of information required to collect and the method for sending information is important. Distances between the command center and the system are vast, if an error occurs that harms the system, the information required to troubleshoot the issue is necessary.

- Testing - Space is a low gravity environment with huge fluctuation in temperature and random electromagnetic and physical interference. Testing to handle these properties requires innovative thinking.

While other constraints exist, this list gives a good point in how design decisions should help to minimize constraint issues.

In the paper "Redundant Design of A CAN BUS Testing and Communication System for Space Robot Arm" by Yhang et al[28] 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 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 CAN-bus which were desirable

including, error detection mechanisms, error handling by priority, adaptability and a high cost-performance ratio. In order to further reliability they implemented a "Hot Double Redundancy" technology. This technology was implemented as the communication system which consisted of an ARM microprocessor, the CAN-bus controller circuit, data storage, system memory and a complex programmable logic device (CPLD) used to implement the redundant strategy. The CPLD interfaced with two redundant CAN controller circuits each connected to its 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, 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.

The Canadian space agency has other projects which attempt to improve several of the concerns. The Canadarm2, the robotic arm attached to the international space station uses an embedded intel386 device to run seven-joint servo mechanisms on the arm, sending the commands to the motor-control embedded processors of each. The designers made the choice to use the Ada language as "Ada inherently encourages developers to spend more time notating their code in writing, communicating information to future readers of that code"[10]. Ada coupled with the 386 processor allowed for better tool support, providing flexibility and an improved reporting format. The reporting format benefitted from Ada's ability to communicate bugs and other faults local to certain areas of the system. Additional bonuses of Ada in this project include portability, code size, reusability and verifiability[10].

2.2 Biomedical Projects

Biomedical engineering is an industry which requires a multi-disciplinary skillset in which engineering principles are applied to medicine and the life sciences. Major breakthroughs in biomedical engineering have advanced quality of life for most societies. Technological advances in the biomedical engineering industry include: x-ray technology, electro-cardio graph machines, artery graphs, pacemakers and various imaging technologies. In recent years interest from the american military has promoted the advancement of artificial limb technology. In 2005 Defense Advanced Research Projects Agency (DARPA) launched a prosthetic revolution program. The goal to deliver a fully

functional upper extremity to an amputee, replacing much of the three-hundred year old technology that still exists. 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. Furthermore, the technology must include sensors for feedback, 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.

2.2.1 Advanced Prosthetics

The University of New Brunswick's (UNB) hand project [27] seeks to extend the crude, one degree of freedom, one grip type old hand prosthetics, into a three axis, six basic grip anthropomorphic hand, with control of the hand using subconscious grasping to determine movement. The use of the subconscious grasping element requires embedded microprocessors, the use of which puts strain on existing power technology, increases overall expenses and reduces component interoperability. With this in mind the UNB hand project outlines the use of a number of technologies designed to reduce power usage, lower production costs and increased flexibility minimizing the impact of the new technologies. The hand projects method to power reduction uses a multiple technique approach, the creators used smart design principles and innovative use of technology to reduce overall consumption in numerous areas. The UNB hand team built intelligent Electromyography (EMG) sensors that could amplify and process signal information, passing only required information to the main microprocessor. This allowed the different sensors to process and reduce excess noise, removing the need for a single high-speed processor to handle all processing, significantly reducing overall power consumption. Another technique used in the hand project was connecting the sensors to the central processor through a serial bus. This allows for a reduction in wiring, reducing overall weight and simplifying component architecture. The serial bus they had chosen, a modified Controller Area Network bus, also allowed for a reduction in power consumption, mainly through the transceiver only requiring power when transmitting. The CAN bus is also noted as having a good compromise between speed and data protection, a necessity for prosthetics[3]. An additional method in which the UNB hand team used to reduce energy consumption was in the form of an aggressive power saving technique. This technique consists of the main processor putting specific sensors into a sleep state when activity is below a pre-defined threshold and awakens them when that threshold is exceeded. This method allows for considerable energy savings. An additional concern of lowering production costs was approached through the parallel design and the use of the serial bus. The parallel design makes it possible to use a less powerful and therefore less expensive central processor, the serial bus reduces wiring,

these methods reduce overall costs. The hand project creators have begun creating a communication standard [24] 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 "A CAN-based distributed Control System for Upper Limb Myoelectric Prosthesis" by Banzi and Mainardi [3] 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 CAN bus to the UNB hand project, adding evidence of successful device integration in the CAN buses traditional area, automotives and how this could parallel the prosthetic industry. The paper also outlines reasons behind not choosing other communication protocols such as technologies which seemed to be good choices: I^2C and Serial Peripheral Interface (SPI), personal computing standards such as: Firewire, USB, ethernet and industrial control systems such as: Profibus, FIP, P-NET, etc. The two initial choices, I^2C and SPI, 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 CAN bus offered everything the researchers required, a good quantity 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.

3 Design and Analysis

The area of analysis and design in embedded system projects requires special consideration beyond normal software design strategies. Embedded systems are generally more intricate as they often have both a hardware design and a software design component. These two components add an additional complexity overhead as the relationship between the two is often highly coupled. Physical constraints and conflicting design criteria require early design considerations, these considerations will shape the design space affecting the overall direction of a project. A change or missed design requirement could invalidate an entire project.

3.1 Analysis and Design

Designing an embedded system requires a thorough analysis of the objectives and constraints. Often the constraints and objectives conflict with one another, causing the necessity of a trade-off between competing requirements. One of the major design challenges is balancing the requirements to optimize the system to best fit the overall design goals. The optimization problem is incredibly complex, with a huge number of design possibilities and many design goals. The paper "Conflicting Criteria in Embedded System Design" by Eisenring, Thiele and Zitzler [12] has outlined a methodology for such a design. The authors suggest the following components can help manage both the optimization criteria and increased complexity:

- Model design - Modelling system design with tools that provide flexibility and modularity, allowing the solution to be broken into sub-problems.
- Use of optimization algorithms - Computer aided tools can use optimization algorithms to solve combinatorially hard multi-objective optimizations and explore possible design avenues.
- Object-based representation - Object representation of hardware components allows fast design exploration and design simplification through abstraction.

Model design is helpful in capturing and managing complexity. Numerous models of computation exist, these models can be used to define a number of attributes and properties such as: components, procedures, processes, functions and states. Additional models can be used to define interaction, communication protocols and component domain or knowledge. The use of models allows for specification completeness, verification of design and can promote a straight forward strategy to quickly form an implementation. Computer aided tools exist which help the process, specification tools take user-defined models and constraints and formulate a refined specification,

this is then used by architecture, communication and hardware synthesis tools. The synthesis tools contain detailed information on their specific areas such as available computing resources, timing issues, implementation costs, circuitry required. From this the tools then synthesize a detailed model, each model is then fed into the next synthesis tool. The architecture synthesis tool compiles all the information required on the targeted architecture to allow the communication tool to start building communication components and software drivers. These components are then passed into the hardware synthesis tool which determines how to physically represent each gate and circuit the communication synthesis tool requires. At each level, the tool can use optimization algorithms or manual inputs in an attempt to optimize overall design. Due to the large and complex search space, evolutionary algorithms are often used to find near-optimal solutions. The algorithms operates under a time constraint as the solutions are intractable [12], the goal of the algorithm is to find the best fit, or most optimal solution in a given time frame. After a number of synthesis runs, the embedded system designer should have a better understanding of the costs vs. benefits of the proposed system, specifications and hardware components. This understanding would then allow the designer to revise the overall design, change assumptions and run more simulations or choose a possible design and begin working on realizing the synthesized components.

3.2 Intellectual Property Cores

Intellectual property (IP) cores are a fundamental block in hardware design, they are analogous to libraries in computer software. IP cores are designed to be reusable blocks of logic which have defined interfaces consisting of inputs and outputs. IP cores can be black boxes, much like functions of a software library where the internals workings may not be known to the user providing the designer with a way to encapsulate their logic.

IP cores can be distributed as "soft" cores or "hard" cores. Soft cores are distributed in a hardware description language providing the user with the most flexibility, allowing the user to modify the core with ease by editing the high level design and then resynthesizing. Hard cores are defined with a low level physical description. This will limit the flexibility of the design, but provides the user with more concrete timing and performance information, forgoing the need to synthesize the design and thus any timing disparities as a result.

3.3 Tool Suite

Keeping with the idea of using tools that break the solution into subproblems, allow for optimization algorithms and produce object-based representations of the architecture. For the development and design with Altera products a number of different tools are provided. These tools

and mechanisms will be looked at in this section.

3.3.1 Quartus II - Environment

For implementation and design on Altera FPGA technology, Altera provides the development environment Quartus II (Figure 1). The software supports all Altera products, and offers several different methods to design hardware.

Various HDLs are supported, such as AHDL, VHDL, Verilog and SystemVerilog. In addition the Quartus II environment allows the design of modules to be in state machines or block design diagrams. A project can contain combinations of different design files. This option allows the developer to choose the best design technology to match the project's goals.

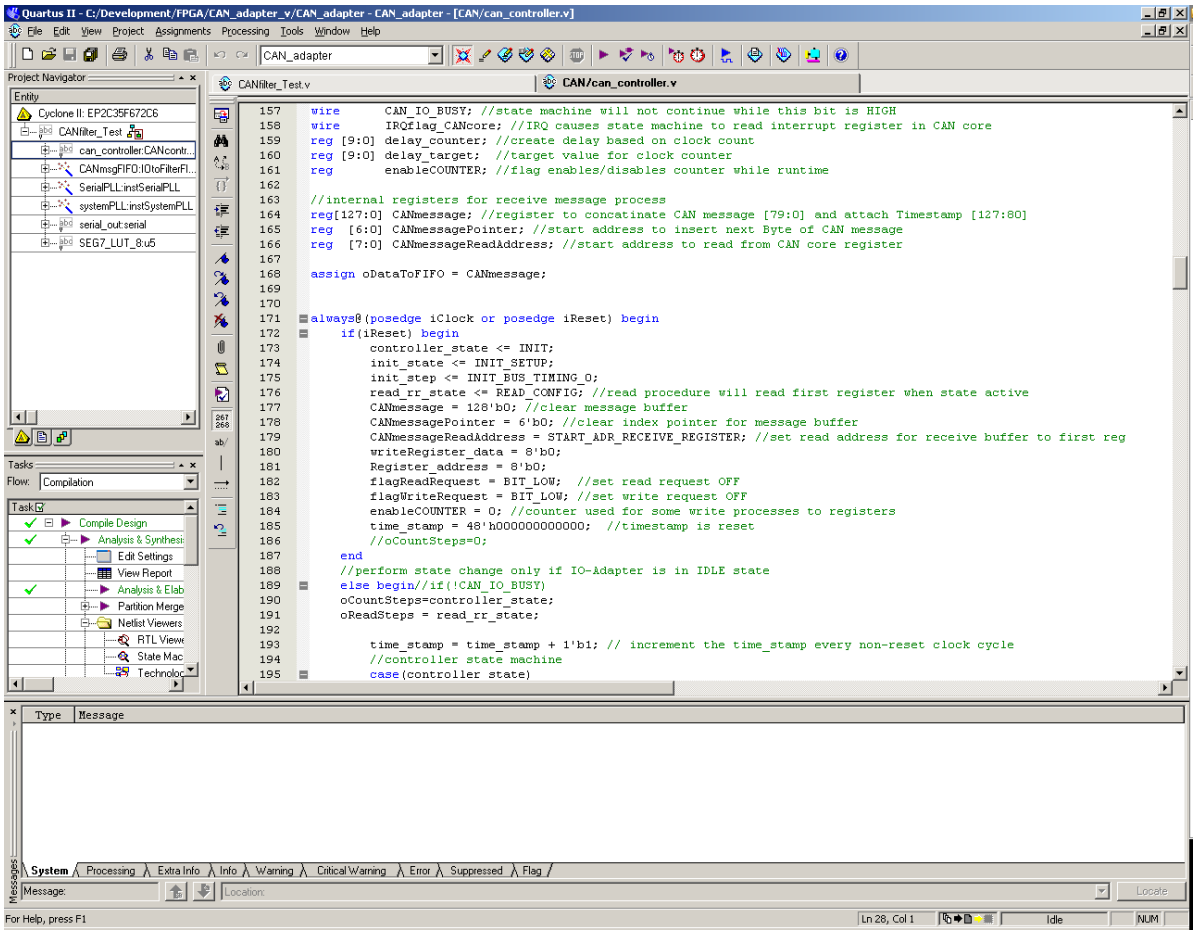

Figure 1. Quartus II Development Environment from Altera.

Quartus II performs the necessary steps required to create a working hardware design. The first step is the analysis and synthesis. Based on the design files in the project, the synthesizer begins minimizing the logic elements and maps the logic operations in the design onto the resources

of the targeted device architecture. In the next step the Quartus II Fitter performs the place and route for the identified device resources. The logic functions are assigned to logic cells with specific locations in the FPGA and the interconnection paths and pin assignments for input and output are assigned. The Fitter tool uses additional information about timing requirements from the analysis and synthesis step, when selecting the logic cells for the logic functions.

The Quartus II environment uses one of the two Timing Analysis Tools. The Classic Timing Analyzer and the TimeQuest Timing Analyzer. The first Classic Timing Analyzer is a proprietary solution by Altera which checks for specific parameters, like maximum clock frequency or signal set-up and hold times. The TimeQuest Analyzer supports the Synopsis and Design Contraints format, which is an industry standard for design constraints and timing assignments. Both of these tools analyze the design and validate its performance. The tools estimate the signal run times between the created interconnection paths for the specific target device. This helps the developer to identify critical paths the synthesized design has created. The results can be used to define performance constraints and assignments for the Fitter tool, improving the timing behavior of the system.

3.3.2 Altera IP Cores

For the design of hardware systems with more complex functionality, Altera provides a library of modules that can be integrated into the architecture. These design blocks are known as intellectual property (IP) cores (Figure 2). These cores are useful in that they fully implement a standard area of a systems functionality. They often come with timing guarantees, technology features and well defined documentation. A downside is that these cores may require additional licensing fees, depending on various aspects of the design and its intended use.

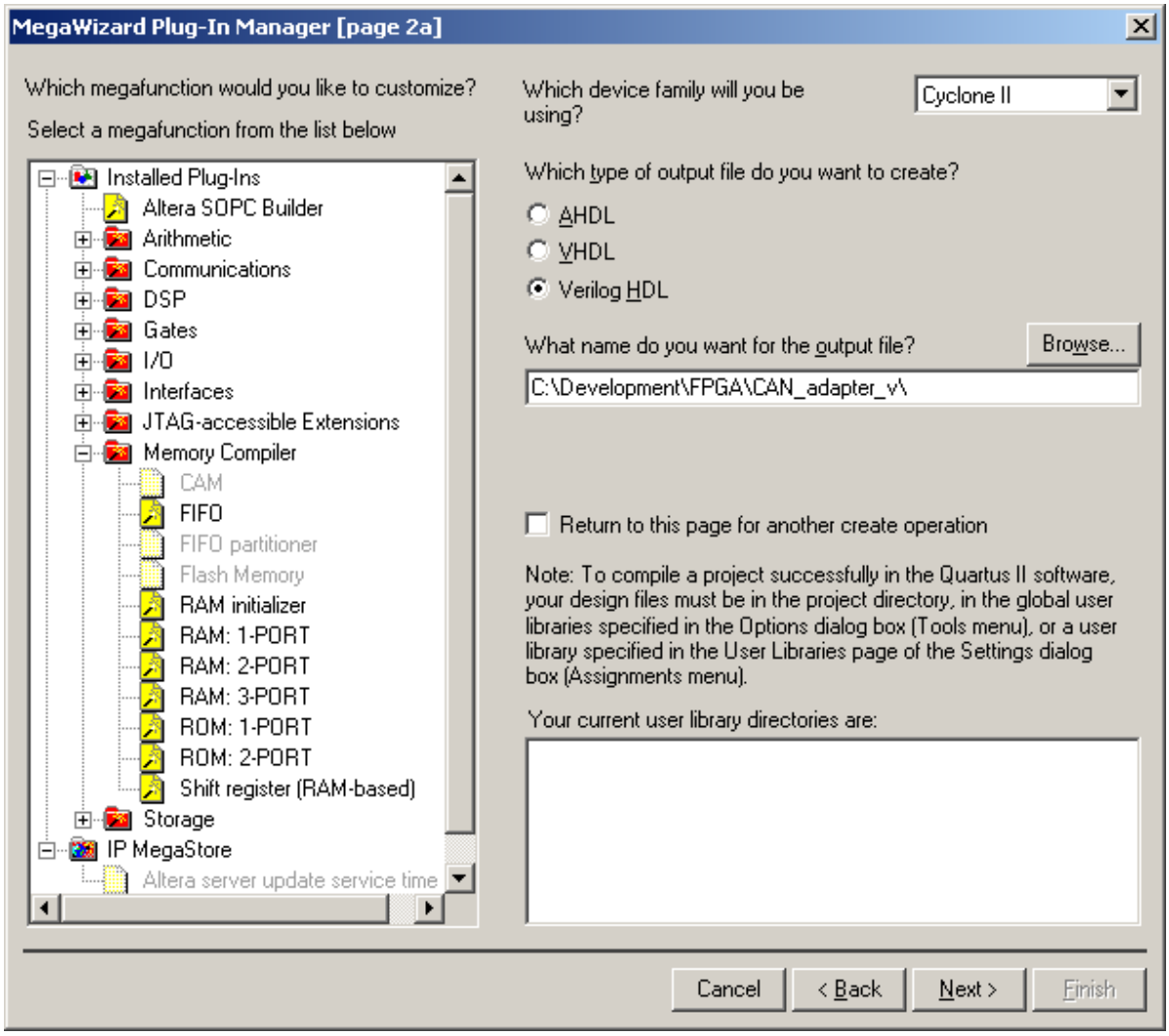

Figure 2. MegaWizard PlugIn Manager for IP core selection.

3.3.3 Quartus II - Waveform Simulator

For simulation and verification of the hardware design, the Quartus II tool has an integrated simulator (Figure 3). It provides functional simulation, to identify logic design errors and timing simulation to locate signal runtime problems. The tool uses waveform files as input and generates waveform files as output. The design of a waveform input file is very time consuming, especially for timing simulations. For timing simulation the signal set-up and hold times have to be according to the timing requirements of the hardware design.

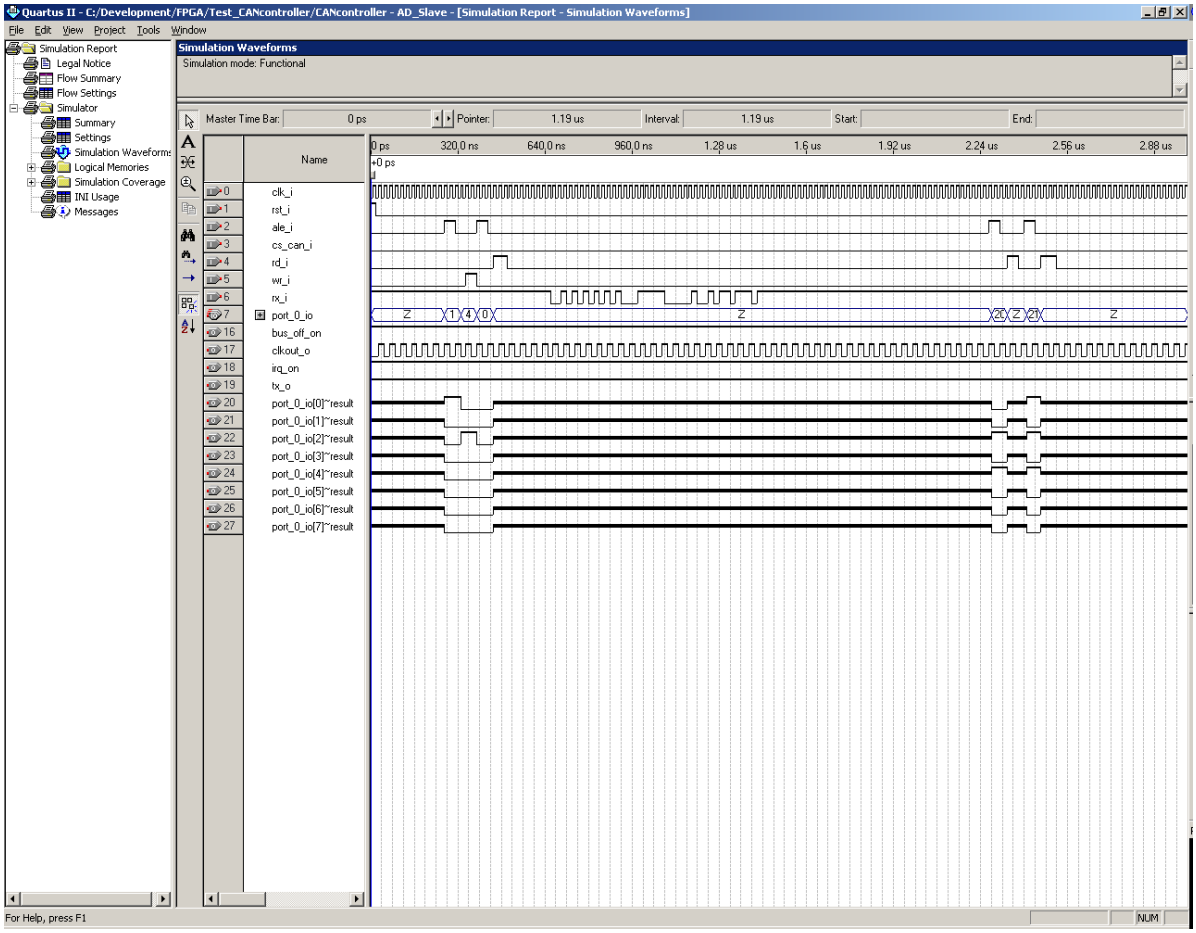

Figure 3. Quartus II waveform simulation.

3.3.4 Model Sim - Simulator

The company Mentor Graphics offers another tool for simulation and debugging of hardware designs. The ModelSim ALTERA STARTER EDITION (Figure 4) uses the HDL design files and vendor libraries to create a simulation of the hardware design. In its current version the tool can only simulate projects containing one kind of HDL. The tool allows to simulate the functional behaviour of the system. Global signals and input values are defined in a testbench file, that connect to the inputs and outputs of the system design. The simulator allows the source code to be executed in a step-wise fashion.

This simulator makes the hardware debugging similar to software debugging, allowing the average programmer a degree of comfort in identifying logic errors. The downside to the simulator is that it does not allow performance timing simulation. Without performance timing it is difficult to identify problems with signal hold and set-up times.

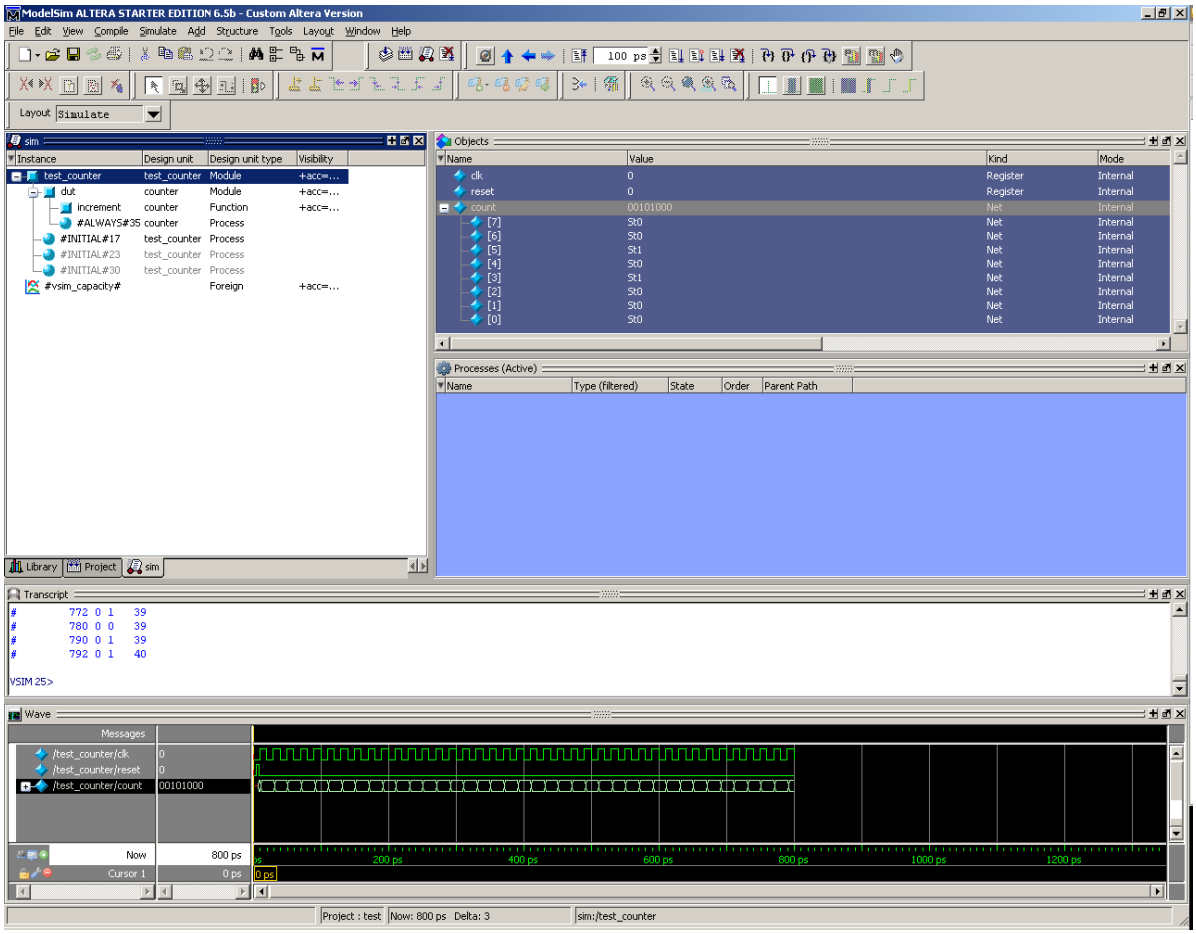

Figure 4. ModelSim ALTERA STARTER EDITION.

4 Hardware

Embedded systems have become increasingly important in modern technologies. As computing components decrease in price and the use of electronics continues to invade all aspects of life, the need to have options that allow flexibility is obvious. Embedded systems are processing systems embedded into larger devices. These systems are designed with a narrow scope of functions which operate in a variety of environments. With the loose definition of embedded systems the range of devices that take that definition includes portable personal electronics, automation systems, large networked controller systems. When an embedded system is adequately specified engineers can optimize the various constraints to provide specific benefits such as cost, size and other properties. With all the elements that make up an embedded system, the three tasks which together determine the success of a system are hardware design, software design and the union of the two, hardware/software codesign.

The task of hardware design has its roots in electric circuit design and electronic logic. This design must consider not only the problem at hand, but also the rules that govern it. The hardware engineer must take into account the laws of physics and electricity while balancing the tradeoffs between the various properties of the system. Hardware designs can quickly grow in complexity, creating the need for the engineer to manage the complexity or risk potential errors. These errors can have adverse effects, in traditional Application Specific Integrated Circuits (ASICs) design if errors are not caught before the design has been put on a chip, millions of dollars can be lost, as there is no way to fix the problem. In programmable logic devices or general computing units, changes to the underlying software can be made that may fix the problem, however fundamentally flawed hardware is an expensive and problematic area to fix.

In programmable logic devices and general computing, software design is a very important part of the system. Software design in embedded systems is often under much tighter constraints then on desktop computers. Memory is often limited, power consumption a concern, traditional areas handled by the operating system may not be available, this leads to the need for the programmer to have a strong grasp of the hardware aspect of the system. Co-design starts becoming an issues as well, changes in the hardware aspect of a system may lead to an obsolete code base, or worse. If careful measures are not put in place to fully test a system after hardware changes, flaws in the co-design could cause serious problems in the system.

4.1 Reconfigurable Computing

Reconfigurable computing is a technology which allows circuits to be defined through the use of high level definitions then quickly loaded onto a circuit. Reconfigurable hardware promotes design flexibility allowing system designers the ability to prototype, test and change their designs

quickly. The advantages of this technology are its reconfigurability and its lower hardware design costs. The ability to change the hardware after the system has shipped decreases the chance of obsolescence and allows for continued customization. The design costs are lowered through the ability to quickly get the system to market. Although modern reconfigurable devices are not as fast as custom designed circuits technological developments have greatly improved overall performance. The other issue with FPGAs is the cost per chip is higher then that of the Application Specific Integrated Circuit, but lower than the microprocessor.

The first reconfigurable chips were called Programmable Logic Devices (PLD) and were introduced in the 1970s. PLDs were the first chips to contain large amounts of configurable logic gates connected by programmable switches. The first available technologies were Programmable Logic Array (PLA) and Programmable Array Logic (PAL). A PLA is based on the sum-of-products (SOM) description of a function. A fixed number of inputs go into a set of buffers and inverters. From there the inputs are sent into a plane of AND gates and finally into a plane of OR gates from where the function results are going out. With the programmable switches on the plane of AND gates and the plane of OR gates, the desired logic function is realized by enabling or disabling the switches. The PAL can be described as a less configurable PLA. Only one of either the AND or OR logic plane is programmable while the other is fixed. This reduces the cost of production but also requires significantly more logic planes, especially where complex functions are being created.

The next generation of programmable devices after PLA and PAL was the complex programmable logic device (CPLD). It contains grids of PLDs where the interconnections between those grids were programmable as well. The PLDs often included additional logic elements such as flip-flops, multiplexers and tri-state buffers. CPLDs allow the implementation of complex circuits which utilize between two and one-hundred of the "extended" PLD logic elements.

While the architecture of a Field Programmable Gate Array (FPGA) is similar to a CPLD, the concept is quite different. An FPGA has I/O-blocks and interconnection wires like a CPLD. Instead of programmable planes of AND-gates and OR-gates, an FPGA contains logic blocks. Those logic blocks are freely programmable, that means they can realize any function that fits the size of the block. For the design of larger functions, the logic blocks on the FPGA are combined through interconnection wires. The architecture of a logic block can be realized in different ways. A logic block is usually realized in the form of a look-up table (LUT). A LUT is capable to realize any small logic function [8].

An advantage of the reconfigurability of FPGAs is hardware/software co-design. The goal of hardware/software co-design is to find properties within the target problem that allows for optimal design parameters to be met through a collaboration of hardware features and software

design. An example of this is in the optimization of a complex functions through the use of hardware supported load balancing and parallel execution. In most cases good to significant improvements in performance are possible. One problem that occurs due to the complexity of hardware/software co-design is that humans are not capable of finding all the possible points of optimizations. However, with the invention of automated design tools, this problem has been partially solved. Specialized design tools using heuristic approaches to perform optimizations and place & route tasks have been successful in improving results beyond human capabilities [25].

Designing a circuit or system in an FPGA is often done through a high level description language, this allows a compiler program to perform the monotonous tasks of logic reduction and gate conversion, reducing the impact of human error. In FPGAs there are two common languages, one is the Very High Speed Integrated Circuit Hardware Description Language (VHDL) and the other is Verilog HDL [2, 22]. Both languages are about the same age but were designed for different purposes. The idea behind VHDL was an alternative way to describe the functionality of hardware circuits. VHDL distinguished between three different design descriptions. Behavioural, Structural, and RTL Dataflow Description, which can also be combined in one design known as Mixed Description.

In a behavioural description the design is based on processes. This process works like an interrupt in a PC, the process is executed every time the status of its input signals are changed. The process flow of the system is fixed by the wiring of the process modules in the design. Another design method is the structural description, this describes the internal and external configuration of entities. The third method is known as Register-Transfer-Level (RTL) Description and characterizes the system's data-flow. Where the behavioural description defines processes which execute on changing conditions like signals or register values. The RTL description defines concurrent blocks of execution, the order in which the blocks must be executed and the connections between the execution blocks.

The design intent of Verilog was to have a practical hardware description language that allowed the simulation of hardware designs. One of the major design goals was to have the description language be similar to the C programming language, a standard in both the software and hardware engineering fields. This would allow many professional programmers to quickly use and become comfortable with Verilog. The first Verilog standard was published in 1995 (IEEE 1364-1995 [15]). With its last extension of the standard in 2001, Verilog is referred to as a verification languages as well [16]. A hardware verification language (HVL) is a programming language which is used to verify electronic circuits. Often HVLs provide simulated real world constraints testing and functional test coverage assistance, allowing verification of complex circuits. Both HDLs have their specialities for hardware design and can be combined in a single system if required [7].

4.2 Application Specific Circuits

Application Specific Integrated Circuits (ASICs) are integrated circuits, that are customized for use in a single application. The ASIC can provide both a lower physical cost and better overall performance than the other technologies in this section. Through the use of economies of scale and the removal or optimization of physical features, overall device cost is lowered. Performance benefits from the inflexibility of its design, all extraneous features that are not required in the application are removed, reducing latency and improving throughput. While the inflexibility of design does increase performance, it is also ASICs biggest drawback. Designed for one purpose, the ASIC is unable to be easily adapted to new functions or changing conditions within the existing application. This increases the time-to-market and makes ASIC unattractive for rapid prototyping purposes and target products with small production volumes. The ASIC design requires a thorough understanding of the application area, a study of optimization possibilities and most importantly a flawless final product [8].

Today a popular technique for prototyping an ASIC is through the use of FPGA technology. The design is loaded onto a FPGA chip, this allows the circuit designers the ability to test and verify quickly, without the cost of creating a chip layout. The place and route from the grid structure of an FPGA design to an optimized chip layout is still an open problem. The generated netlist for an FPGA design needs to be transformed into a formal ASIC design specification. The FPGA prototyping method is of course not possible with all technologies, some ASIC designs will be too complex to simulate, making it necessary for formal prototypes to be created. It must be kept in mind that the required design space on a FPGA is several times higher, than the design on the ASIC later on. With the small costs for high production volumes, ASICs are still the most beneficial opportunity for mass production. The suppliers of ASICs can be categorized into two groups, integrated device manufacturer (IDM) and fabless. An IDM supplier provides the whole semiconductor manufacturing itself. A fabless supplier usually outsources many parts of the production line. The IDM can be classified as both, depending on the production method they utilize.

4.3 Microprocessors

A general purpose computer, or microprocessor allows a quick solution for a targeted embedded system application. With the generic approach to the hardware issue, emphasis is placed entirely on the software design aspect. The flexibility of software allows for quick changes of system behaviour allowing error correction, increased functionality and system updates. Another advantage includes a fast time to market. The hardware does not have to be emulated or manufactured as it does with other technologies, this allows rapid analysis, development, testing and implementation.

Disadvantages to this approach includes expensive hardware and slower operation. Microprocessor units are generally more expensive then other solutions, the requirement to be flexible significantly increases the complexity of the chip design and the number of components that make up the chip. This same complexity also creates the lower performance issue. The typical fetch-decode-execute cycle and the generic approach to handling execution found in microprocessing units lowers the performance of this technology. Pure hardware solutions only perform the execute stage, therefore in order to perform at a similar level an increase in the processing speed must occur, this increase is detrimental to many properties that are required in an embedded system.

The first commercially available microprocessor was the Intel 4004, released in 1971 [9]. The processor was a 4-bit Central Processing Unit (CPU), the first of its kind to be placed on a single chip. The CPU had a maximum of 32kb of ROM and 5kb of RAM, 46 instructions a max clock rate of 740 Khz and the ability to execute 92,000 instructions per second [18]. The key to this technological achievement was made possible through the use of P-channel silicon gate Metal Oxide Semiconductor (MOS) technology. MOS is created by growing a layer of silicon dioxide on top of a silicon substrate then amassing a layer of metal on top. The semiconductor is then able to act as an amplifier or switch, modifying the electronic switches as necessary. The silicon gate technology had a number of benefits over previous technology, they allowed for higher speeds with lower power consumption, they were more cost effective and they were order of magnitudes more reliable, a necessary property considering modern processors have as much as a billion transistors. Hundreds of advances have occurred throughout the life of the microprocessor. These advances have increased energy efficiency and performance while reducing physical size and cost. Todays most advanced microprocessors use a greatly modified but still similar method of creation to the Intel 4004. The Intel i7-980x CPU is a 6-core 32nm microarchitecture processor containing 1.17 billion transistors on a 248mm^2 die size. The CPU can handle a maximum of 24GB of RAM, contains a 64-bit instruction set has a max clock rate of 3.6 Ghz and can perform approximately 44 instructions per clock cycle [19]. Testing shows the I7-980x performs at about 147,600 million instructions per second [23].

Embedded systems have been using an increasing number of processors every year, it is estimated that in 2008 98% of the ten billion processors sold were used in an embedded system [4]. Embedded systems have to be efficient and often do not require the verbose instruction sets found on modern personal computers. Instead efforts must be made to find microprocessors that closely map to the instruction set required for the application environment and have an efficient compiler.

5 Communication

With more complex problems often a single hardware device will no longer adequately meet the requirements of the system. It is often necessary to build a communication bus or network that will allow the various aspects of a system to coordinate and communicate with one another. The method in which devices communicate to one another is in generally the bus or network topology. In the bus topology a central controller handles the scheduling and data traffic. The messages contain the address of the needed device, these messages are then broadcasted to all and the device matching the address will then handle the message and do what is requested. As in the bus topology, a controller handles the data, however instead of broadcasting the message for everyone to see, the message is routed to the specific device which will handle the request. The bus topology is much less resource intensive to implement and performs well when the number of node devices on the bus is small. Although a controller is required to perform the communication, it is simple in design. The network topology is more resource intensive, requiring a greater number of interconnects, communication lines and controller logic elements. It is then also able to handle more parallel communications, as it is designed to have no communication collisions even at high speeds. Early communication technology was little more than bundles of wire attaching the various components. The devices communicated to one another through proprietary and custom protocols with their own timings, voltage and instructions. This method fragmented the various vendors with each of them building devices on top of their own formats. Eventually solutions to various problems in communications occurred. Interrupt driven controllers vastly improved time efficiency. Increased throughput allowed for greater speeds and eventually allowed the promotion of serial as opposed to parallel communication methods. With the changes and advances in technology, standards organizations created specifications. This move consolidated much of the computing and electrical engineering fields as a specification governed by a standards organization allowed device makers to operate their hardware on any platform that conforms to the targeted specification. Today there are hundreds of specifications that outline communication technologies and how they address the assorted concerns of their respective industry.

5.1 Communication Methodology

The method in which devices communicate is an easy concept to understand. A device communicates by applying a number of signals to a medium using a predetermined messaging format. This encoded message then travels over the medium and is received by the targeted device. The message is then deciphered and acted on using an agreed upon guide or specification. Through this process there are a number of aspects that must be considered, these decisions affect the various tradeoffs that make up a communication system. In the concept example developed earlier

in the paragraph three specific items were left in an ambiguous state, that of medium selection (what do messages travel on and how does it get on the medium), message encoding (what makes up a message) and communication specification (how do we handle a message). These three items make up a layering of technologies, with each choice beneath the current level both limiting and promoting various properties of the current layer. The Open System Interconnection (OSI) Reference Model is a seven layer description that promotes good layered communications. From a hardware perspective, the bottom two layers are the most applicable the physical and data link layer. For the purpose of describing how hardware interacts with a communication system, three sub-sections of the two layers will be considered. These divisions are the physical media section, the line coding section, and the communication definition section.

5.2 Physical Media

The first consideration is that of physical media section. In this section the primary concern is with medium selection, the physical constraints surrounding each choice and the method in which a bit is represented on the medium. Possible medium selections include copper wires, wireless signals and optical fibers. Each possibility influences how a bit is represented and the inherent limitations of the media. Copper wires utilize an electrical voltage to simulate the bit. Through the process of transmitting bits over a wire, noise from the surrounding electromagnetic environment can degrade the quality of the line voltage. Copper wires are the traditional method for transporting information and as such the equipment required to utilize them is generally inexpensive, however the copper wires themselves can become a factor in total cost. Wireless signals apply energy to an electromagnetic or radio wave, this energy in combination with the wave simulates the bit. Wireless signals operate in a much noisier environment, this causes wireless technologies to often have a slower throughput than wired communications. Wireless signals often have a higher overall power consumption[13] than the other communication methods, this is because the distance a signal can travel is directly related to the energy used to create the signal. With copper and fiber optics the medium can contribute significantly to the overall cost, where as with a wireless technology this is not so. Finally, optical communication uses light as a carrier wave in which bits are transmitted over an optical fiber. This process is resistant to much of the issues inherent in the electrical wiring and wireless communication technologies. it is immune to electromagnetic noise, has significantly higher data throughput and is able to travel further without signal boosting. The main disadvantage is a large medium and communication component costs. Another disadvantage is the lack of commercial devices that directly support fiber optic communication, thus fiber optic communication is not overly relevant in the embedded system community.

5.3 Line Coding

Where physical media selection determines the medium, and how the bits access it, the line coding section identifies the methods which maps values and timings on the carrier to both of the binary representations of a bit. This mapping or encoding will allow the digital information to be more resistant to signalling issues during transmission. Line codes are created with respect to the following characteristics[6]:

- Intersymbol interference- Sections of the carrier have poor transfer performance in frequency ranges. The receiver can suppress the unreliable frequencies which will reduce intersymbol interference allowing the system to handle channel variations.
- Timing information extraction- The ability to derive the timing characteristics of the signal from the encoded data itself.
- Transparency- The statistics of the signal should be well-defined and extractable, allowing proper operation.
- Error monitoring- If possible, a line coding should be able to both detect and monitor errors in the signal.
- Word alignment- If bits are encoded in blocks of data, the received sequence must be in the correct order prior to the decoding phase.
- Added signals- A line coding should be able, if required, to handle additional signals added to the carrier. This includes additional signals added such as power or digital and analog information.
- Efficiency- The line code should be optimal, that is it should maximize available data rate and minimize the use of information redundancy.
- Cost- An equilibrium should be met between the value the line coding provides and its inherent costs.

Combine these characteristics, the properties of the physical medium and the various application possibilities and you have the necessity for a large number of unique line codings. The basic categories of line codes include: return-to-zero and non-return-to-zero [6]. Return-to-zero and non-return-to-zero simply specify whether the waveform returns to the base or zero volt level for a portion of time between pulses. Return-to-zero is self-clocking, that is because the line voltage returns to zero an external timing device is not necessary to determine where each value lies[5].

However the time it takes the line to return to zero has a negative effect on overall rate of signal transmission. Where as non-return-to-zero is able to have a faster signal transmission rate then return-to-zero[5]. The tradeoffs with non-return-to-zero problems are synchronicity and larger bandwidth requirements[5]. Many line codes exist with varying degrees of complexity, all provide various trade-offs that must be taken into account when choosing the optimal encoding.

5.4 Communication Definition

The last consideration is the communication definition section. This section often refers to a collection of standards defining how messages are formatted and handled by devices in the system. These standards specify the medium, the line coding and any other information related to transferring data between nodes. The main concerns are delivery protocols, addressing and communication methods, and media arbitration. Delivery protocols include data formatting information, this information is unique to each definition and allows for a number of properties. Common properties included in the data or frame format is:

- A start of frame indicator, allowing the protocol to synchronize the frame sections accurately.
- An address or identifier of the end destination of the message.
- A unique identifier, used to assist in error recovery and retransmission of missing frames.
- A size of payload value, used to indicate the size of the payload contained in the frame.
- The payload section, used to hold the actual communication message.
- An error checking section, often a cyclic redundancy check or other frame check sequence.
- An end of frame indicator, allowing accurate capture of the entire frame.

Numerous other values and sections are custom to the various specifications and allow for innovation, flexibility and reliability in data communication. Along with the actual formatting responsibilities the specification also outlines how the frames are handled. For instance, many specifications outline procedures that occur if a frame fails an error check or certain sections contain predetermined values. In this case a section could identify how the payload data is encoded, which allows the receiver to decode the data correctly. The addressing portion consists of a well defined method in which communication can be targeted or established between two devices on the network or bus. This may include the creation of a communication master which directs the frames to the correct location through an addressing scheme, or a bus method which allows each node to determine which messages it needs to handle. The final concern of these definitions is

media arbitration, the specification must determine how to handle the situation in which more than one device tries to communicate over the medium at the same time. This encompasses both the detection of the frame collision and the method of recovery for such a collision. The most used techniques for detection of collisions and recovery are the Carrier Sense Multiple Access with Collision Avoidance or Collision Detection (CSMA/CA, CSMA/CD) protocols and their variations. CSMA listens to the channel to determine if it is in use, when it is not in use it attempts to send a message. If a collision occurs the protocol dictates that the transmitters wait a random time before attempting to transmit again. This protocol suffers greatly from the effect of propagation delay causing a large number of collisions to occur [26]. The CA modification causes the transmitter to listen on the wire for a given period of time, after this if the channel was clear it would put a small request frame on the carrier. This frame is used to tell other devices not to transmit, then the transmitter would send its full message. This improved the performance over plain CSMA and is often used where CD is not possible (due to the inability to both listen and transmit)[26]. The CD modification uses the ability to sense when another device is interrupting a transmission. Once an interruption is detected the original transmitter then puts a jam signal on the carrier. This signal forces all devices to stop transmitting and a semi-random algorithm determines when the devices should retransmit. The algorithm uses a random number which is multiplied by an exponential collision counter, simply put as a device continually gets caught in a collision, the amount of time it waits to retransmit increases in a random exponential fashion to a pre-defined maximum value. The best aspect of this protocol is the ease in which it is able to be setup[26].

5.5 Example Communication Standards

The following three examples give a good contrast between standards crafted to work in radically different areas. The three standards are: Controller Area Network bus (CAN-bus), RS-232 and 802.15.4.

5.5.1 CAN-bus

The CAN-bus is a serial communication standard used to handle secure and realtime communication between electrical and microcontroller devices. The CAN-bus 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 CAN-bus supports bit rates of up to 1Mbps and has been engineered to handle the constraints of a security conscious realtime system[1]. 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 or the 29 bit identifier extended frame. 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 CAN-bus standard outlines the arbitration method as CSMA/BA where the BA stands for bit arbitration. The bit arbitration method allows any device to use the bus if it is free. If there is a collision the transmitter with the greatest priority, identified by all devices comparing their transmitted message with the received message, if there exists a dominant bit where recessive bit was transmitted the device looses arbitration and then backs off for a predefined period of time. This allows the highest priority messages to get handled 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[20].
- 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[1].
- 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.

5.5.2 RS-232

The RS-232 standard dictates the method in which two binary serial devices communicate using Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE). This standard does not dictate all three sections of communication as the CAN-bus and 802.15.4 standards did. Instead only the most basic physical components are defined and the rest is left up to the device or software designer. This technology has existed since the early 1960's where it was first applied to handle communication between teletypewriters and low-speed modems. The technology has been extended and adapted to new technologies and applications, as a natural process of doing so a number of design limitations have been encountered[14]. The RS-232 specification supports bit rates up to 20,000 bps however speeds in excess of 115,200 bps are common place in devices. The standard specifies a wired medium with a maximum cable capacitance of 2500 pf[11]. It also outlines the circuits required to manage the connections, how the devices operate under synchronous and asynchronous transmission and how voltage levels map to the logical ones and zeroes required for digital communication. In this respect, RS-232 differs from most digital devices in that it uses +3 to +15 volts as a logical 1 and a -3 to -15 as a logical 0[11]. Conventions exist that provide structured communication, this is done through the use of pre-defined signals and hand-shaking.

5.5.3 802.15.4

The 802.15.4 standard deals with low-rate, low-energy, wireless personal area networks. The standard is designed to maximize the desired properties of embedded system components, while allowing for a simple and flexible protocol. 802.15.4 has data rates of 250 kbps, 100 kbps, 40 kbps and 20 kbps, communicates in a peer-to-peer or a star network over a wireless medium. For the arbitration technique 802.15.4 mainly uses CSMA-CA, however it also utilizes Time Division Multiple Access (TDMA) Guaranteed Time Slots. This allows contention free communication between two predetermined devices for a time specified by the network controller, the Personal Area Network (PAN) Coordinator[17]. The standard defines different levels of device functionality, the reduced functionality node and the fully functional node. It also requires a fully functional node to become the PAN coordinator. When used in a star topology, the PAN coordinator controls the communication through-out the network, in this situation all information is passed through the PAN coordinator. In the peer-to-peer topology, all fully functional devices operate in the best interest of the PAN coordinator, they are free to handle requests and route data as required without having to work directly through the PAN coordinator. In both networks the reduced functionality devices transmit and receive data under the direction of the PAN Coordinator, they are not able to provide any network controller assistance.

The line coding used in this standard is a statistical multiplexing technique known as orthogonal code division multiplexing (OCDM). OCDM allows multiple nodes to share the wireless medium through use of added pseudonoise. The carrier wave is modulated with pseudorandomly generated +1's and -1's, giving the carrier the appearance of noise. The receivers who had been synchronized with the transmitters prior to this, are able to recreate the pseudonoise which can then produce the original carrier wave. This method allows all communication devices on a single channel to focus on the signals which contain the known pseudonoise and ignore other white noise, which can in turn be other devices communications[17].

The message frame for 802.15.4 depends upon the mode of operation. The two modes, beacon and non-beacon mode use different communication methods, which use different framing. Both non-beacon and beacon mode use a similar frame (Figure 5) during data contention periods. The frame contains the usual identifying, error correcting, synchronizing and addressing fields.

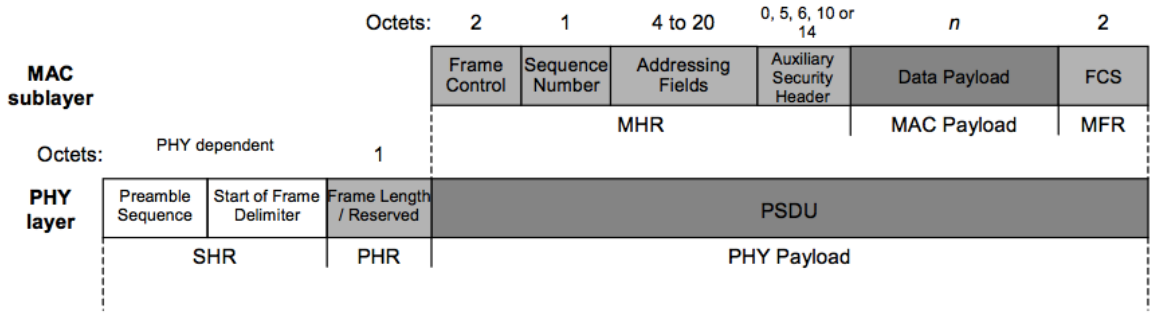

Figure 5. 802.15.4 Data Frame

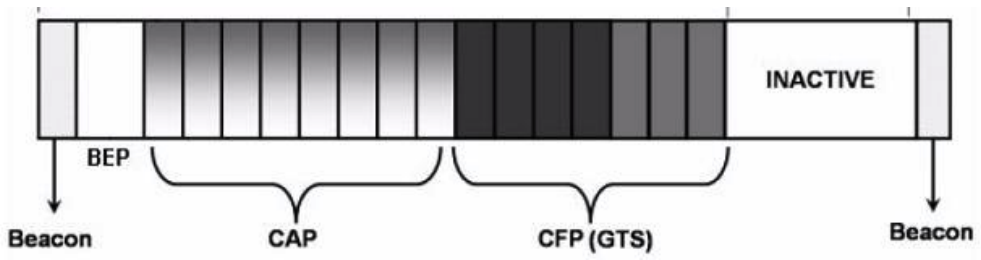

Figure 6. 802.15.4 Super Frame structure

In beacon mode the frames are encapsulated by a superframe (Figure 6). This superframe contains three content sections, the contentious access period (CAP), the contention free period (CFP) which consists of a number of guaranteed time slots (GTS) and the inactive period. The beacon and beacon extension period are administrative sections that help initiate the superframe. The length of time spent in each section and the use of each section is dependent on network settings and the PAN coordinator. During the CAP section devices use the CSMA-CA network

arbitration. During the CFP TDMA communication is guaranteed between predetermined devices, allowing the two devices to have full bandwidth access without the possibility of collisions allowing for full speed data transmission. During the inactive state, devices often enter low-power sleep mode while the PAN coordinators would enter into another superframe in which it would communicate with other areas of the network. The administrative sections are used to synchronize communications and initiate the CAP setup.

6 Summary

This report has shown that the considerations when implementing an embedded system are numerous. Careful evaluation of a projects goals and constraints with an understanding of the tools and technologies is necessary in order to have a successful solution. This report has presented details of successful projects, outlined tools and methods used to build embedded systems and given a synopsis of the hardware and communication ideas and technologies essential to a good project. While each project is different, the information that has been displayed in this paper should provide a good starting point, providing a general understanding of embedded projects, with emphasis on actual biomedical engineering solutions.

The four sections of the report were embedded system projects, analysis and design, hardware components and communication technologies. The embedded systems section communicated the challenges facing system designers. The section also highlighted embedded projects, specifically the University of New Brunswick Hand project, showing examples and techniques used in successful projects. The analysis and design section gave a synopsis of best practices and methodologies used in system design today. The section also showed off the Altera tool suite, useful in the analysis, design and testing phase of an embedded FPGA system. The hardware section provided the history and an overview of the three main embedded technologies, programmable logic devices, application specific integrated circuits and microprocessors. Finally, the communication section outlined methods used to communicate within and outside of embedded systems, properties of the communication systems including physical medium properties, line codings and communication definitions and examined three protocols: Controller Area Network bus, RS-232 serial communication and 802.15.4 wireless communication.

Embedded Systems, as this report has shown are complex technologies. Tools and best practices are continually updated and redefined to reflect new methods in developing successful solutions. A good understanding of the technology behind embedded systems and related fields is necessary in order to manage the trade-offs and risks that occur within all projects.

References

[1] Can specification version 2.0.

[2] J. Aylor, R. Waxman, and C. Scarratt. Vhdl - feature description and analysis. *Design & Test of Computers, IEEE*, 3(2):17–27, April 1986.

[3] 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,.

[4] M. Barr. Real men program in c. Online, April 2010.

[5] T. C. Bartee. *Digital computer fundamentals (6th ed.)*. McGraw-Hill, Inc., New York, NY, USA, 1985.

[6] J. W. Bergmans. *Digital Baseband Transmission and Recording*. Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[7] V. Berman. Standard verilog-vhdl interoperability. In *Verilog HDL Conference, 1994., International*, pages 2–9, Mar 1994.

[8] S. D. Brown and Z. Vranesic. *Fundamentals of Digital Logic with Verilog Design*, volume 1. McGraw-Hill Higher Education, 2003.

[9] I. Corporation. Intel's first microprocessor - the intel 4004. Website, April 2010.

[10] R. Dewar. Case study: Space station robot embeds ada. *COTS journal, the journal of Military Electronics and Computing*, March2002, 2002.

[11] EIA/TIA. Eia standard rs-232-c interface between data terminal equipment and data communication equipment employing serial data interchange, August 1969.

[12] M. Eisenring, L. Thiele, and E. Zitzler. Conflicting criteria in embedded system design. *IEEE Design and Test of Computers*, 17:51–59, 2000.

[13] J. L. Hill and D. E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22:12–24, 2002.

[14] P. Horowitz and W. Hill. *The art of electronics / Paul Horowitz, Winfield Hill*. Cambridge University Press, Cambridge [England] :, 2nd ed. edition, 1989.

[15] IEEE. Ieee standard hardware description language based on the verilog(r) hardware description language, Oct 1996.

[16] IEEE. Ieee standard verilog hardware description language, 2001.

[17] IEEE. Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans). Technical report, 2006.

[18] Intel. *Intel 4004 Single Chip 4-bit P-Channel Microprocessor Data Sheet*. Intel Corporation, 2200 Mission College Blvd. Santa Clara, Ca 95054-1549, USA, 1972.

[19] Intel. *Intel Core i7 Processor Extreme Edition Press Deck*. Intel Corporation, February 2010.

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

[21] P. Leteinturier. Multi-core processors: Driving the evolution of automotive electronics architectures. website, September 2007.

[22] G. Lipovski. Hardware description languages: Voices from the tower of babel*. *Computer*, 10(6):14–17, June 1977.

[23] T. Logan. Intel 980x gulftown synthetic benchmarks. Website, March 2010.

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

[25] G. D. Micheli, R. K. Gupta, Rajesh, and K. Gupta. Hardware/software co-design. *IEEE Micro*, 85:349–365, 1997.

[26] A. Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 2002.

[27] 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.

[28] 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*, pages 1894–1898, Dec. 2008.