6-1-2010

RecoNode: Towards an Autonomous Multi-Robot Team Agent for USAR

Kang Li
University of Denver

RecoNode: Towards an Autonomous Multi-Robot Team Agent for USAR

A Thesis

Presented to

The Faculty of Engineering and Computer Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Kang Li

June, 2010

Advisor: Dr. Richard M. Voyles

Author: Kang Li
Title: RecoNode: Towards Autonomous Multi-Robot Team Agent for USAR
Advisor: Richard Voyles, Ph.D.
Degree Date: June, 2010

Abstract

Urban search and rescue (USAR) robots can benefit from small size as it facilitates movement in cramped quarters. Yet, small size limits actuator power, sensor payloads, computational capacity and battery life. We are alleviating these issues by developing the hardware and software infrastructure for high performance, heterogeneous, dynamically-reconfigurable miniature USAR robots, as well as a host of other relevant applications. In this thesis, a generic modular embedded system architecture based on the RecoNode multiprocessor is proposed, which consists of a set of hardware and software modules that can be configured to construct various types of robot systems for dynamic and unforeseen changes in the USAR environment. The benefit of this Reconfigurable Node is that, at run-time, the system can react to unexpected changes in configuration, such as nodes exhausting their batteries or the failure of sensors. These modules include a high performance microprocessor supporting complete on board processing for autonomous control, a reconfigurable hardware component, and diverse sensor and actuator interfaces. The design of all the modules in the electrical subsystem allows for the replacement of the motion control and serial communication capabilities within a dedicated FPGA logic module, which helps gain system performance by releasing the CPU from these tasks. The selection of module components and real-time scheduler and operating system (OS) are described. The portable power supply solution is also designed and tested.

ACKNOWLEDGEMENTS

First of all, I would like to express my appreciation to my adviser Professor Richard Voyles for providing me with such an interesting project work on and his valuable guidance to an interesting direction. Also, I would like to thank Steve Elzinga for his help in setting up the design system and for his expert advice on FPGA. I would like to thank current and former members of our research group, particularly, Xiaoting, Robert, Mustafa, Salah and Sam for our numerous discussions on various research topics.

I am also grateful to Professor Matt Rutherford and Professor Mohammad Mahoor for serving as the committee members in my final oral examination. Their comments and suggestions helped me to improve the quality of my thesis. Finally, I would like to give special thanks to my family, my friends and my host family for their love, encouragement, and constant support to continue my studies.

Partial support for my work was provided by the NSF Safety, Security and Rescue Research Center and by the NSF through grants 0923518, 0841483 and 0719306. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

List of Figures

CHAPTER 1 Introduction

In the past, catastrophic disasters such as earthquakes, mine disasters such as Chinese miners, the explosion in a mine in West Virginia and the terrorist attacks on the World Trade Centers have frequently appeared on the news. These events clearly demonstrate the need for special-purpose resources to respond to incidents of partial or complete structural collapse caused by these types of major disasters. Urban search and rescue (USAR) is defined to be the emergency response function which deals with the collapse of man-made structures, has a different emphasis than traditional wilderness rescue or underwater recovery efforts, and can be even more demanding on robot hardware and software design than military applications. [1]

1.1 Background and Motivation: urban search and rescue robot

The USAR robot's main role is a source of collecting and feeding back information to help responders identify victim in an ambient environment that generally implies collapsed structures. USAR robots need to be mobile to achieve a higher degree of coverage and connectivity. In the presence of obstacles at the USAR site, mobile robots can plan ahead to avoid debris and move appropriately to obstructed regions to increase rescue target exposure. During the above process, there are mainly two challenges that the USAR environment presents for a USAR robot: limited time and limited space. From the time perspective, the search and rescue task is most effective in the 48-72 hours "Golden Window" immediately after the disaster.

If survivors are not found within this window, profound difficulties for survival worsen as time passes. From the space perspective, survivors are usually entombed in voids with minimal access and entry ways are often blocked or tortuous. Therefore, intended solutions for USAR require a small-scale, rugged, and lightweight robot to access small spaces /voids and cross over tough terrain to inspect victims. Currently, no robot system design serves the above search and rescue needs, and these challenges motivate us to create and design a next generation USAR robot system.

1.2 Problem statement

1.2.1 Resource constraints

Miniature robots for USAR are decided to meet the need to access small spaces in a collapsed structure. Small size inherently benefits USAR robots as it facilitates movement in cramped quarters and permits easy navigation through small, tight spaces and voids deep in a rubble pile that humans and dogs cannot safely enter (as well as larger robots). Yet meanwhile, small size consequently brings a new problem: limiting capability such as sensing, actuation (torque), computation capacity (precision) and power (battery life), etc. Therefore, creating a suitable actuation mechanism, making maximum use of sensors, finding balance between computation and power are all our concern.

a. Actuation

Most popular mobile robotics platform consist of wheels or tracks. Wheeled robots are simple to construct but they require continuously traversable terrain. Big wheeled robots can surmount large obstacles while smaller robots will get stuck if

they encounter obstacles larger than the robot's wheels themselves. Tracked robots are much more common in the arena of search and rescue because their design is only slightly more complex than wheeled robots, yet remain simple to control. Most of tracked vehicles are typically heavy and big with similar problem as wheeled robots. Thus, we need to come up with a small, light and highly mobile robot may be able to more easily and rapidly explore voids deep in a rubble pile unreachable by humans, dogs, or currently available commercial robots.

b. Sensors

Sensors are needed for collecting vision features, heat, and sound information for two main tasks: survivor identification, and navigation of the robot. Most robots' relationship to their environments is limited by sensors; normally, we use cameras for detecting survivors Sensory System for USAR environment may also include CO2/CO/O2 /pulse detectors, GPS locators, infrared thermal camera FIR, audio transmitters, and laser beam scanner .etc. Apparently, we cannot expect the USAR robot carry all the sensors due to size or power limit. However, we would like robots to prepare for unexpected situation with as much sensors as possible. For example, if survivor detecting is being done with the camera, but it suddenly becomes very foggy so the camera cannot be used. Then we hope the hardware previously set aside for the camera can be reconfigured for thermal sensor to compensate for the loss of sight. So we need come up a sensors configuration scheme to adapt complex, changing and unknown environment.

c. Power

For a multi-robot team working in large scale USAR area, using tether to provide unlimited power is not applicable anymore, because cables are easy to tangle with each other or obstacle commonly in USAR condition. Then battery package is naturally considered as power source for mobile robotics systems. But for miniature robot, the addition battery to the robot is greatly limited by robot confined spaces. Therefore, the power consumption issue has become a fundamental concern. Especially we need to find a balance between limited robots power supply and dense computing demand applications like processing and communication. Our approach is to rededicate limited resources to current task and provide low-power consumption and power aware design.

1.2.2 Time constraints

To meet the golden window time frame requirement, responders need to quickly deploy a team of robots to maintain adequate sensing coverage and network connectivity. In this way, USAR might help responders reduce search time to identify victims; besides, autonomous and flexible robotics system architecture proposed might help responder reduce operation time of robots.

a. Reconfiguration and modular design

A USAR robot designed for a single purpose can perform that specific task well however poorly on some other tasks, especially in a different environment. Specifically, a fixed-architecture robot is acceptable if the environment is structured, but for tasks in a complex urban search and rescue (USAR) environment, which

normally contains unstructured components, tough terrain and uncertain/ unknown factors; a robot with highly adaptive capabilities is more suitable. Because it's impossible for a robot to carry all kinds of sensors and operate them at the same time which will kill battery fast, we want to assign and configure potential usable sensors before the robots deployment and then decide specific ones we need to use after deployment. Modular reconfigurable robots show the promise of great versatility in this scenario: it allows off-line swapping the sensor and actuator peripheral hardware and then reprogramming the new added hardware on- the-fly to serve a specific task. In contrast most robots today are built monolithically; the characteristic of modular design allows robots to be constructed in a fast and easy manner at large quantity and could be used and deployed for various complex applications. Moreover, a modular design allows a wide range of heterogeneous robots to be assembled from a basic set of modules like CPU and peripheral board etc. Besides maintains, modular unit change within system structure, and specifically focuses on the reconfiguration of internally developed. We could remove devices if their functions are not in use or if modules break they can easily be replaced by spare modules. Finally, the reconfiguration/repair time and cost of the entire system can be reduced because we do not have to repair or replace the whole robot system and individual modules can be mass-produced. Our goal is to introduce of modular robotics concepts into our design and thereby take the advantages of it.

b. Autonomous multi-robot team

Urban search and rescue robots working together based on the cooperation in a team could dramatically reduce rescue time. There are several reasons: first of all, if the given targets will be assigned separately, we could send out multi-robot team and every agent communicate with the others and perform distributed activities. Secondly, for victim identification and environmental monitoring in disaster areas, manual deployment of multi-Robot team might not be possible. Mobile multi-robot team can move to areas of events after deployment to provide the required coverage. Finally, we also want to increase multi-robot team agent's autonomy with lower demands on humans after deployment so they can do collaborative computation, sensing, action and make decisions to perform higher level task on target. For each agent, we would like it to have high and fast computation capability to support for autonomous mobility for complete real-time on-board processing for autonomous control including motion control application such as hardware timers, PWM outputs for controlling motors and other actuators, vision application such image data acquisition and processing and wireless communication. In summary, our vision for ultimate robotic design and functionality is to create more versatile robots by combining basic modules with general reconfiguration capabilities to achieve greater system efficiency and flexibility for complex tasks.

1.3 Heterogeneous multi-robot team

In collapsed areas where intervention by humans or dogs may not be possible, a large team of miniature robots that are densely deployed has distinct advantages over

single robots with respect to sensing various information either inside or close to the observation objective. Team members may share sensor information and help each other to scale obstacles and detect victims. Multi-robots can move to areas of events after deployment to provide the required coverage and interact with their environment by sensing or controlling objectives and collaborate to fulfill their tasks since a single robot is incapable of doing so on its own. As for sensing, a team of robots can perceive its environment from multiple disparate viewpoints. Besides sensing, Multi-robot team has actuation advantage as well: For example, when multi-robot team manipulate or carry large objects, a given load can be distributed over several robots so that each robot can be built much smaller, lighter, and less expensive[2], which meet our criteria of miniature scale USAR robots. A multi-robot team system is usually classified as heterogeneous if one or more agents are different from the others based upon physical attributes such as sensors and actuators [3]. And heterogeneous multi-robot team could refer to multiple robots with different HW/SW configuration, sensor/actuator components and other external entities communicating and cooperating such as motion coordination to achieve a common goal. Heterogeneous multi-robots could provide powerful situation awareness capability due to different locomotion capabilities and sensor information. However, it would be expensive and time consuming to develop specific embedded system for different type of robots. So we need come up with a new node platform, where the node itself could be reconfigurable to a different HW/SW system for various USAR applications with multiple sensors and actuators. This feature for USAR robots is also called adaptive

capability. So each agent is a multi-purpose robot and should be reconfigurable adaptive to deal with complex USAR environment. As a result, it will avoid unnecessary redistribution of multi-robot to save much energy and rescue time. More over, the robotics platform will take completely modular architecture that allows individual components to easily be added, replaced, or modified and many different miniature mobile robot systems could be constructed.

1.4. Innovations and contributions

A summary of the primary innovations and contributions includes:

1. Test out FPGA baseboard and PROM.
2. Finish the schematics and PCB design, assembly, and test process of motor board and power board.
3. Physical external Morphing Bus realization.
4. Build embedded processor system platform for DU100 under Xilinx EDK XPS.
5. New IP Development: Come up with PWM and encoder, UART VHDL IP core for motion control and communication purpose.
6. Port software infrastructure Scheduler, UART and PID software.

1.5. Thesis Outline

Chapter 1 offers a brief outline of the project, its motivations and its background.

Chapter 2 goes through existing networked mobile robotics hardware architecture.

Chapter 3 presents some concepts and technologies in the areas of high performance reconfigurable computing systems where we could bring to my design.

Chapter 4 describes the RecoNode architecture including HW and SW infrastructure

Chapter 5 evaluates the success of the design and its fulfillment of the design criteria, details the performance obtained from the system

Chapter 6 summarizes the project and discusses future work on the RecoNode

CHAPTER 2 Literature Review

We are interest in reconfigurable hardware infrastructure architectures for an agent of a group of distributed robots systems working cooperatively on a common task for urban search and rescue. In this application, if robots operate in a well-coordinated manner, the overall mission can be accomplished much more effectively in shorter time. Let us go through current existing hardware platforms to see whether they support our applications.

2.1 Mobile WSN node: Cotsbots/ MAS-net/ Robomote

Several research groups have developed different kinds of mobile node platforms for wireless sensor networks. MAS-net [4], Robomote [5] and CotsBots [6] are examples of such systems, based on MICA boards that, together with sensing and motor control stacks, constitute a wireless sensing node. Cotsbots is a combination of Mica Mote, Motor Board on a RC Car Platform; Cotsbots use commercial off-the-shelf (COTS) components to build distributed robotics, so it is inexpensive and modular. Open-source hardware and software platform for distributed robotics. USU MAS-net is very similar to CotsBots, especially the mobility platform is also based on MICA serial mote. Employing Atmel ATmega128L 7.3827 MHz 8-bit CPU and Chipcon CC1000 radio the difference is the mobility mechanism. Unlike CotsBots, which implements an Ackerman steered wheel robot, MASmote is a two-wheel differentially steered mobile robot. It is structurally much simpler, yet still

manipulation flexible. MASmote has digital IR reflector as its wheel encoders which can be used for odometry measurement and localization. The USC Robomote is a single board with dimensions 3.81cm x 2.23cm based on an Atmel AT9OS8535L 8 bit micro controller. This board connects to a Renemote, the "Mote" component, making the complete Robomote.

2.2 Kephera and Tyndall Mote

Compared with most of these WSN nodes focus on low-power, simple microprocessor and sensors based platform hardware infrastructure all duplicated across the network, we are more interested in distributed robotics system utilizes high-end FPGA devices to achieve more powerful onboard computing capability as well as hard/soft reconfigurability and implements the DSRP mechanism for providing more flexible and reliable applications such as video and communication. Some research groups use FPGAs as coprocessor in MCU- FPGA embedded systems module architecture such as Kephera [7] and Tyndall Mote [8]. The FPGA module act as a coprocessor dedicated for running real-time or parallel tasks such as the image processing (image data acquisition and computation) and Bluetooth communication task due to advantages in the parallelism performance and the flexibilities. The Kephera's base module possesses the processing platform from a Motorola 68331 microcontroller, running with 25MHz clock frequency, 512 KB RAM and 512 KB Flash. The FPGA is reconfigured to change its tasks during run-time and also supports partial reconfiguration. Tyndall Institute developed a WSN platform for environment monitoring application, it's stackable and modular. The platform runs an adapted

version of TinyOS. On the other hand, the platform includes an FPGA module, for fast DSP processing. They have some sensor layers for different applications and the power supply layer can include batteries or energy harvesting elements. The system incorporates a 2.4 GHz transceiver with a special protocol to minimize power consumption.

2.3 TerminatorBot

Besides the computing platform, we also need to consider the actuation mechanism which is adaptable to explore rough, uneven and unknown terrains especially common in collapsed structures in USAR environment. Apparently, we need to keep current TerminatorBot's [9] actuation mechanism and fit improved reconfigurable computing platform into the enclosure. TerminatorBot is a small size, resource-constrained, search-and-rescue robot and it has a small form factor for core-bored search and rescue operation and its dual use mechanism for both locomotion and manipulation with its two 3 DOF articulated limbs for surveillance, search and rescue purpose [10]. Limbs are not used only for locomotion but also for dexterous manipulation (e.g. Terminator lifts a small wheeled robot like the Scout into a place that the Scout cannot otherwise access). Thus, T-bot is a good research platform for robotics mobile manipulation. The applications for mobile manipulators have been growing because of the increased ability of robots to interact dynamically with their environment in a precise manner. It combines a wide variety of research areas, ranging from force control to mechanism design and sensor design.

Figure1: TerminatorBot

CHAPTER 3 Reconfigurable computing system

Robotics platform usually take general-purpose microprocessor such as MCUs and DSPs as CPU. These devices are designed with fixed hardware, leaving software as the only method for designers to update designs; also limiting the development of application-specific functions. In previous design, FPGAs logic is mostly used as hardware acceleration, which means we need separate FPGAs and microprocessor boards. In comparison, current fully reconfiguration computing platform with FPGAs logic driven by an embedded hard/soft core processor could be completed in one chip instead of separate boards. Specifically, FPGAs based SoC solution can integrate processor, bus standards, custom modular logic, peripheral interfaces, and DSP functions in one consolidated device. SoC give us freedom to create custom functions completely adapted to specific USAR application requirements by enabling both hardware and software customization. The following concepts show how we can design the next generation of USAR robot with the required functions fully integrated on a single SoC chip.

3.1 Reconfigurable computing

Reconfigurable computing is a computer architecture combining some of the flexibility of software with the flexible and high performance of hardware fabric like FPGAs. Reconfigurable computing is intended to fill the gap between hardware and software, achieving potentially much higher performance than software alone, while

maintaining a higher level of flexibility than hardware alone. [11] Mostly, reconfigurable computing systems use FPGAs or other programmable hardware, and what we specially use here is not only FPGAs logic but also refer to a System-on-a-chip (SoC) architecture containing hard core microprocessor. Here is a scenario to demonstrate how it could benefit USAR robot application: During the USAR Robot real-time operation, it may also become necessary to perform an update to take advantage of new hardware and algorithms. How to switch hardware and software resources is a challenge for these applications. For example, when a USAR robot is deployed, some application may need a module to perform in a mode that demands high power consumption for only 25% of the time. After the high power task finished, we do not need to keep it on in the rest of time. In this situation, if the system is developed using FPGAs with the capability to perform partial reconfiguration and morphing bus, the system can take advantage of operating in a low power consumption mode at other 75% time, reducing the whole power supply needs of the system.

3.2. Self-adaptive

The RecoNode system should have the self-adaptive ability to adapt to system/ component failure or complex USAR environment changes; self-adaptive allows RecoNode to change its configuration and functionality in order to improve the operation of the system in a changing environment. The adaptation of a system can occur from hardware to software. Here, we study self-adaptive at the hardware level and especially with reconfigurable architectures. Reconfigurable architectures such as

FPGAs are a key technology for implementing self-adaptive and flexible systems, since the possibility for dynamic and partial hardware reconfiguration described later offers a higher degree of freedom in the resource allocation. Reconfiguration here not only means reprogramming but also means the change of all the hardware and software system. FPGAs provide the capability to implement functions in hardware, accelerating performance through accelerate algorithm execution by mapping compute-intensive calculations to the reconfigurable substrate and simplifying the software porting effort. This freedom will enhance overall system performance and flexibility.

3.3. Dynamic Partial Reconfiguration (DPR)

The RecoNode robot supports dynamic reconfiguration of its hardware during runtime. At startup of the RecoNode, the FPGA is programmed by the processor with the contents of the on-board memory. Specifically, processor loads partial bit streams from a bit stream repository via wireless communication, and stores them on PROM memory; this allows loading different hardware configurations for the FPGA at power-on of the robot. The FPGA is capable of dynamical reconfiguration, so that parts of the hardware design on the FPGA can be exchanged on demand by new modules, kept in PROM or received wirelessly by Bluetooth or WiFi via a network link. RecoNode could realize various hardware configurations or algorithm depending on the task.

Dynamic partial reconfiguration [12] is considered as a way to make RecoNode accomplish reconfiguration as fast as possible. The whole process is controlled either

by the user or by the system itself. DPR sets up a communication between the source of configuration and the configurable unit and permits change to parts of the device while the rest of an FPGA is still running; In other words, the dynamic reconfiguration can swap the logic contents of a FPGA during system operation without affecting the remaining parts operation. It will always be much faster to change a small piece of the logic than the entire FPGA contents. Not only does DPR give RecoNode system flexibility, but the use of DPR is one efficient way to reduce static power consumption by reduction of chip area for the adaptive system.

Partial reconfiguration is supported by Xilinx Virtex II, Virtex II Pro, and Virtex 4 FPGA. A special software flow in ISE with emphasis on modular design is used dring configuration; typically the design modules are built along well defined boundaries inside the FPGA that require the design to be specially mapped to the internal hardware. However, to our point of view, the dynamic reconfiguration is an important specificity of FPGAs and is still largely underexploited. This would probably change in the future. The most feasible way to implement DPR is using HWICAP interface and open the routed design with FPGA editor, modify any LUT function or memory content and generate a partial bit stream with bitgen.

3.4. PowerPC 405 Core and Bus Architecture

RecoNode use Virtex-4 FPGAs as computing platform. as the IBM PowerPC is widely used in embedded system design, The PowerPC405 is also available in Virtex-4 family of devices. Here we present a brief overview of the PowerPCs general capabilities and internal organization. The PowerPC 405 CPU Core is a 32-bit

Reduced Instruction Set Computer (RISC) PowerPC Embedded Processor, which possesses all of the qualities necessary to make system-on-a-chip designs a reality. This core consumes minimal power, and provides a high performance 100% PowerPC architecture compatible platform capable of taming the most demanding embedded applications. The PowerPC 405 is a low-power consumption processor with operating frequency at approximately 450 MHz in the Vitex-4 FX, which will resolve our low power but high computation requirement. The PowerPC uses two different buses to connect memory and peripherals, namely the Processor Local Bus (Processor Local Bus (PLB) and the On-Chip Memory (OCM) bus. The PowerPC also supports the IBM CoreConnect bus system [13]. The core connect system introduces two new main buses, the On-chip Peripheral Bus (OPB) and the Device Control Register (DCR) bus. The OPB is used to connect relatively slow peripherals and typically runs on a frequency lower than the PLB. The DCR bus is mainly used to transfer device configuration data and settings, the DCR is not used to transfer data or instructions. Figure 2 depicts the IBM CoreConnect bus system as used in the Xilinx implementation of the PowerPC core. The OCM buses are not part of the IBM CoreConnect system, but are depicted to give a complete overview. The OCM bus and the PLB will be discussed in more detail.

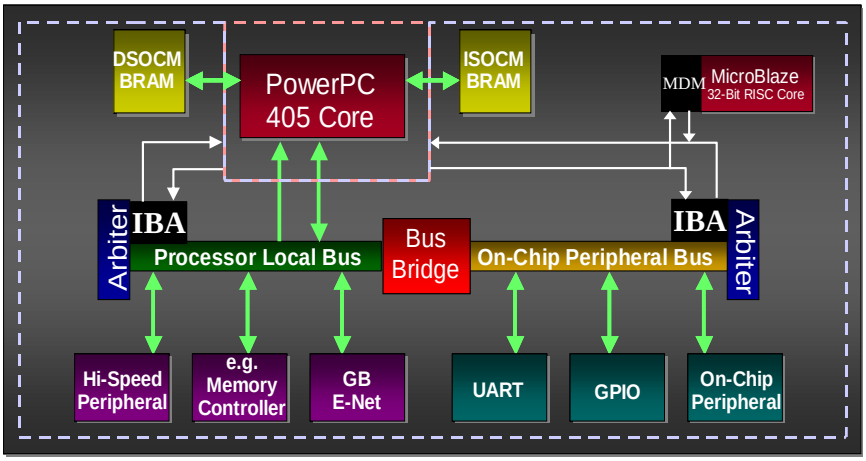


Figure 2: Xilinx PowerPC CoreConnect bus system

3.5. Morphing bus

The morphing bus [14] is a feature that allows for different peripherals to be attached to the main CPU base board without having custom hardware interfaces for each peripheral device. This is valuable because small robots like the TerminatorBot don't often have space for large motherboards. The morphing bus design allows for stacked peripheral boards that can be attached only when they are needed. Due to the applications of small search and rescue robots, many times their operators are not trained robot technicians. This means that any software changes needed when attaching a new peripheral module must be as transparent to the operator as possible. The way the morphing bus works is to send the signals directly to the sensor. The conversion logic is incorporated into the bus interface when it is statically reconfigured for the sensor. This combination of reconfigurable hardware and easy-to-use software make for a robot that is easy and fast to customize and deploy. The morphing bus uses a small section of the FPGA already used by other features of the DU100 to statically reconfigure the inputs and outputs of the main computational system. This means that any peripherals can be attached to any physical ports on the

FPGA. The physical connector takes advantage of this arbitrary pin-out, by allowing each peripheral to take the signals it needs and allowing the rest to pass through to the next connector. For example, if one wanted to connect a three wire video capture peripheral and two wire motor controllers, all one would have to do is plug the video capture board into the DU100 and the motor controllers into it. The video capture board would take its three wires and shift the rest of the bus over by three wires as seen in figure 3. This architecture assures that all wires are usable while allowing any peripherals to be connected in any order.

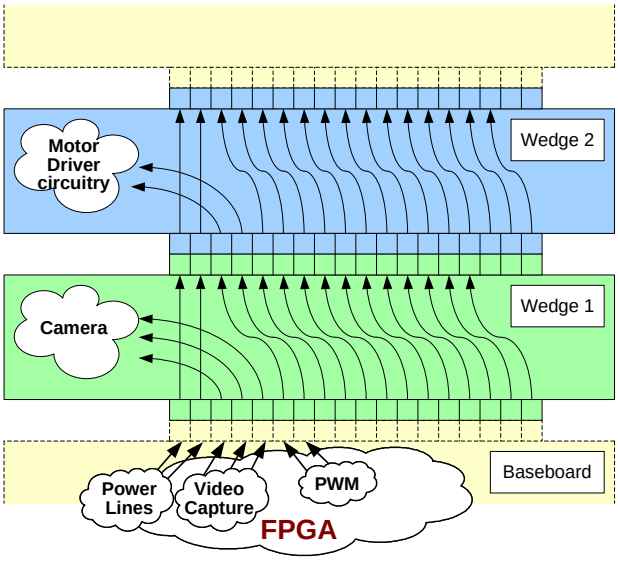


Figure 3: Morphing bus signal routing [14]

The morphing bus is made up of circuit boards each of which is dedicated to only one or more sensors or actuators. The main emphasis is that the boards should be of low complexity and thus small size. Each board has electrical connectors at both ends. All the boards provide the same interface to the preceding and succeeding stages. Thus their position in the bus can be swapped. Each board uses as many bits of the bus as required to support the logic on that wedge and the remaining are fed to the

next connector of the next stage, which in turn does the same and so on. The input lines to a wedge are used as follows: few initial lines are dedicated to power and ground. These are common to all circuit boards and run through all of them. Starting from the next connection the wedge circuitry uses as many I/O pins as it requires. The remaining lines are shifted to the output connector such that the unused lines are now immediately after the power lines.

CHAPTER 4 RecoNode architecture

We develop a novel small computational platform named RecoNode based on reconfiguration technology for urban search and rescue application features only, instead of a comprehensive one. We adopt reconfigurable hardware architectures for increasing demand of complex computationally intensive applications due to the possibility for parallel processing but also to the high flexibility of such architectures.

4.1 Architecture Overview

Currently the RecoNode hardware is composed of four main components: the base board DU100 with Xilinx Virtex 4 chip [15], the image sensor processing board, the motion control board and wireless communication board, Block diagram (Figure 4) of FPGA-based RecoNode architecture shows the basic hardware architecture of RecoNode. All of parts mentioned above are connected via an innovative Bus System-- Morphing Bus and we will present the detail later. Besides designing the various peripheral boards, all kinds of user IP (Intellectual Property) to realize full-fledged robotic functions (e.g., motion control, wireless communication, and on-board real time vision processing) were designed to take advantage of the hardware/software re-configurable nature of the Virtex 4 FPGA. The RecoNode is based around Virtex-4 FX20 FPGA device which combines dual 32-bit RISC embedded microprocessor hard cores (on-chip 400 MHz IBM PowerPC 405 (PPC405) processor cores [16]) and FPGA logic in one silicon chip, design is greatly simplified.

The guiding principle of this SoC approach is to combine large amounts of reconfigurable logic with embedded RISC processors, in order to enable highly flexible and tailorable combinations of hardware and software processing to be applied to a design problem. In addition the Virtex-4 on the baseboard there is CMOS SDRAM (128 Mb x8 bits format MT48LC16M8A2 from Micron) for memory intensive applications, like image processing. Benefiting from the powerful computation ability of dual core mechanism, the processors could run Real Time Operating System (RTOS) / TinyOS (specifically for Wireless Sensor Network application) for some time sensitive applications and also run many application programs at the same time such as various vision algorithms. The FPGA fabric is used for custom computational logic and interfaces. Examples of the computational logic include PWM and Quadrature Encoder Module, as well as an image capture module. Besides these user IP cores, Xilinx provides us IP Cores which enable us to use pre-verified, pre-optimized design blocks to implement commonly used functions such as memory management and UART interfaces. To let the peripherals get communication with PowerPC, we use one of the PPC internal bus structures like PLB (Processor Local Bus), a general-purpose OPB (On-chip Peripheral Bus), a bus bridge, and two arbiters [13]. We used the Xilinx Embedded Development Kit (EDK) in conjunction with the Xilinx ISE tools to develop our embedded hardware and software system co-design, thus facilitate our whole design procedure. The whole design methodology is HW/SW co-design flow using Xilinx XPS (part of Xilinx EDK) includes hardware configuration and embedded software developments

software on the hardware side, the order is generate libraries and BSPs (broad support package), create custom peripherals, populate and connect design, build hardware and generate HW bit stream. On the software side, the basic system library is generated and system will configure and generate SW Platform, then we could develop & debug SW apps in SDK like normal software development environment.

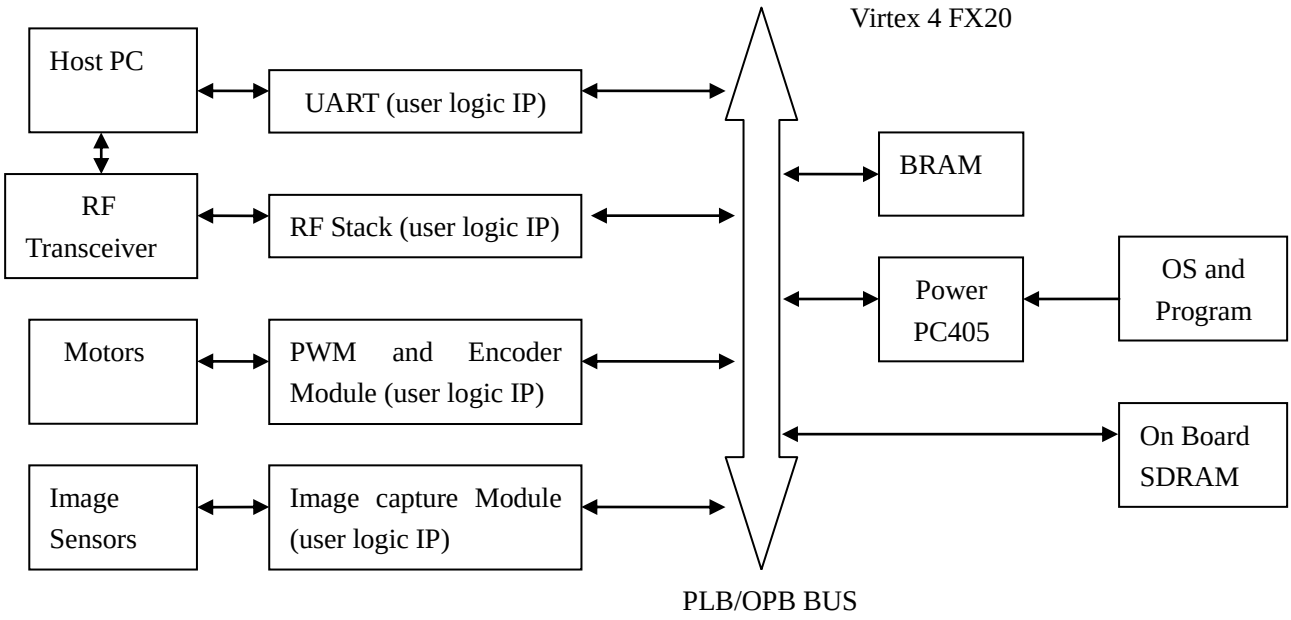


Figure 4: Block diagram of FPGA-based RecoNode architecture

Terminatorbot's unique dual locomotion and actuation mechanism is a big advantage for Core-Bored SAR application, so RecoNode will adapt its mechanical design and fit in DU100 into previous can shaped enclosure. So DU100 will still drive 6 DC motors without losing degree of freedom. Also we could see the figure below: we still have a lot room for adding extra sensors and actuators peripheral.

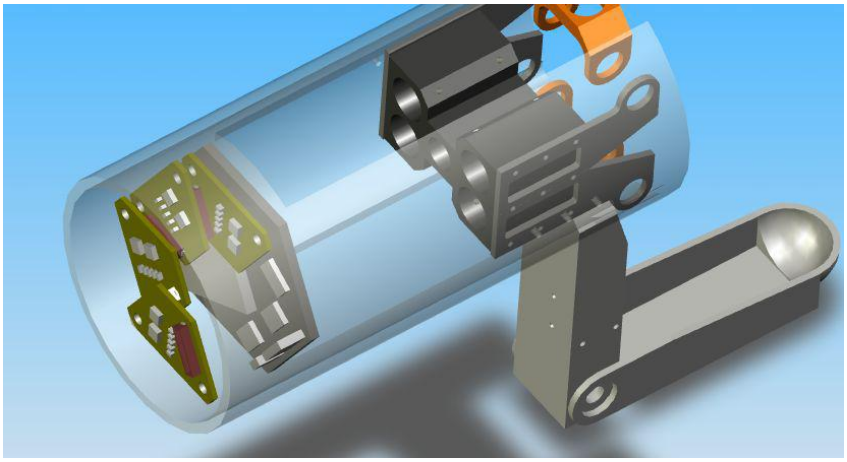
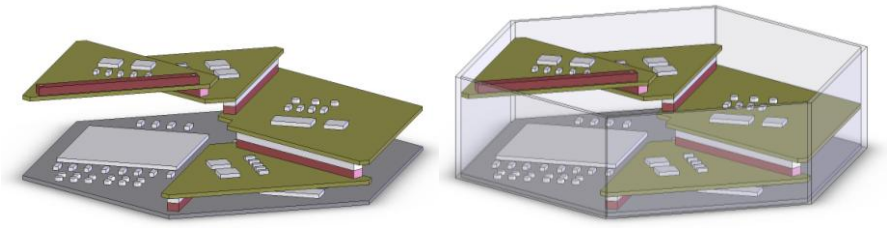


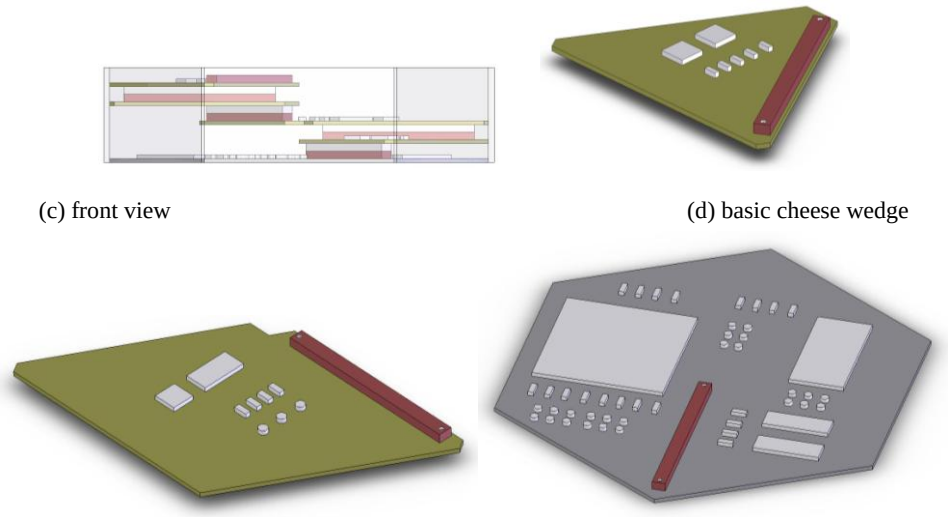
Figure5: A proposed version of the DU100 installed in the TerminatorBot enclosure

4.2 Hardware infrastructure

The hardware is following a modular design paradigm and basic modules are to be as small and simple as possible in terms of physical size, because the smaller the module, the greater the range of shapes that can be built from it. The modules should also be able to function independently of one another. The circuit board design (Figure 6) for the wedges required a hole in the middle of the RecoNode for wires to pass through, so part of the "tip" of the wedge needed to be cut off to allow for this space. It was also required to leave a small gap between the circuit boards and the exterior wall of the RecoNode for extra airflow. For hardware part, I finished all the PCB design including schematic entry design and PCB layout by following OrCAD design flow with detail as following.



(a) Spiral structure without wrapping around (b) Spiral structure with wrapping around



(c) front view (d) basic cheese wedge

(e) double cheese wedge (f) FPGA mainboard

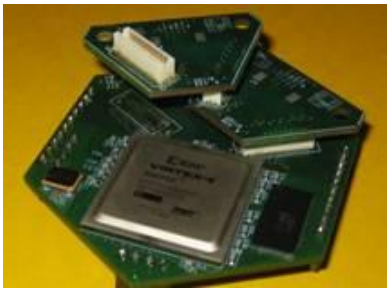
Figure 6: Circuit Board design for the RecoNode with Spiraling diagram [14]

4.2.1. External Morphing Bus implementation

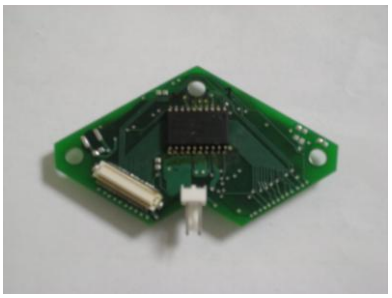
The morphing bus is designed for use in the RecoNode and its structure is shown in Figure7. Because of the shapes of the wedges, when they are stacked up they take the form of spiraling staircase. To provide support to this structure mechanical reinforcements are provided. Air is blown from the base upward, which follows the path along the spiral, cooling the ICs on every wedge. The number of devices that can be connected in the morphing bus architecture is limited by the number of available pins routed from the FPGA through the wedges, since each board has a dedicated connection to an FPGA pin. This is ultimately determined by size of the connector that can fit on each circuit board which in our case is limited by the size of the robot this system is being used in. Also a large portion of the wedge is taken up by the pass through routing of the unused lines, which again restricts us. However this is acceptable, since although this places an upper limit on the number of devices, it has

the great advantage of being able to swap peripherals without interface and arbitration hardware on the devices plugged in. Thus they can be very small, ideal for deployed field robots.

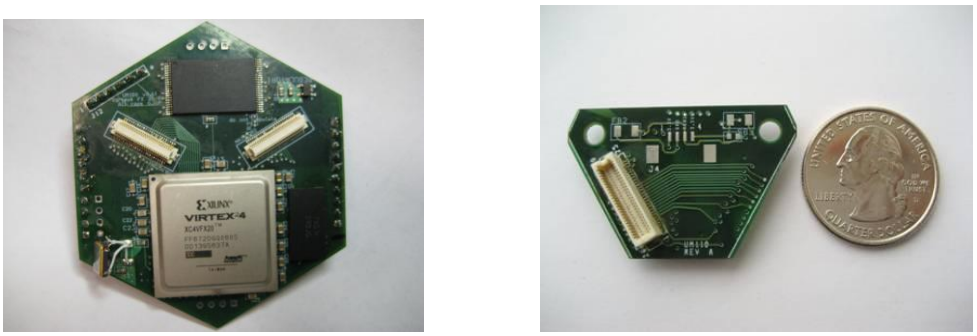
The connectors chosen for the morphing bus have a 0.3A per pin current rating. This is another important factor that needs to be considered when creating the individual wedges. They need to be designed such that the current draw is not enough that it will exceed the limit of the connectors. For typical wedges, this current limit should not be an issue as the current draw for each wedge should not exceed about 50mA. Assuming each wedge takes about 7 or 8 signals, there would be enough room to add 6 wedges onto the morphing bus. At 50mA each, that would just reach the current limit of the connector, but typically it is not expected that the wedges will require 50mA each (the camera, for instance, uses a significant amount of power, but the maximum current under normal operation is 30mA). If more current is needed for an individual wedge, the wedge will need to have its own power supplies; the control circuitry for the motors are one such wedge where part of it will need to be driven by an external power source due to the current requirements.



a



b.



c. d.

Figure7: Morphing bus spiraling structure. (a) One wedge is connected to the base board, starting off a chain where every wedge is connected to the previous. (b) Double wedge (c) FPGA base board. (d) A single cheese wedge.

4.2.2 Base Board

The base board DU100 is composed of four main components; the base board DU100 with Xilinx Virtex 4 FPGA, Platform ROM, JTAG download as well as external memory device SDRAM (as shown in Fig 8). The Xilinx Virtex-4 strongly supports reconfigurable computing, because it contains two 32-bit RISC PowerPC405 embedded processor cores which provide high computation performance: 450 DMIPS@ 300MHz, compared with Atmege128 CPU (16 MIPS at 16MHz) in Terminatorbot. Additionally, multi-core with soft-core is a possible option beside the PPC405 hardcore processor, inherent parallel I/O interfacing manner and dedicated DSP slices are extremely helpful for processing signal from various sensors in USAR. Moreover, a large amount of FPGA logic gates and cells (CLB) are useable for hardware acceleration to help us build our digital circuit at we need. In summary, this System-on-Chip hardware solution will give RecoNode system powerful on-board real-time processing ability for computing intense multitask applications and enable robot system with hardware reconfigurability.

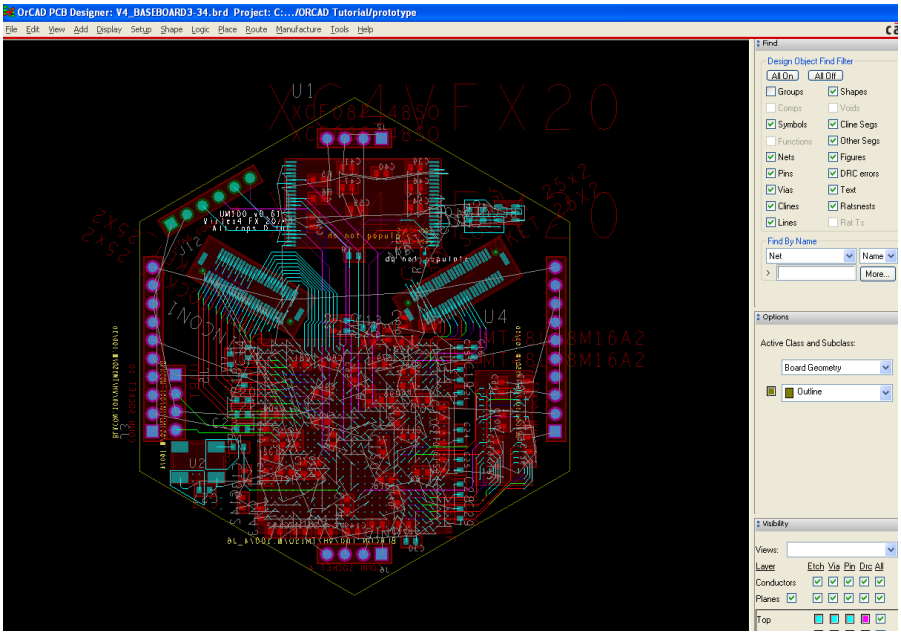


Figure8: Base board PCB and Schematics

4.2.3 Motion control board

We need to get the baseboard DU100 with Motor board DU120 so the PowerPC can control the motors, thus motor control board was developed first. The TermionatorBot has 6 Maxon DC motors [17]. So does RecoNode. The nominal voltage of the motors is 6 volts and the output power is 1.41 Watts. The efficiency of the motors is 71% with a no-load speed of 16,300 rpm and a no-load current of 30 mA. The motors are controlled using an L6205 H-bridge, made from discrete IC components, and utilizes Pulse Width Modulation (PWM) for its operation. The four signals which control the motors are PWM1, PWM2, Direction1 and Direction2. By changing the direction bits, the direction of the motors can be reversed. Motion control is triggered by velocity and distance commands from the upper layer. These commands are converted to the corresponding PWM and tick values. The gear ratio is 25:1. The RecoNode relies on precise odometry for movement from one location to

another. The feedback uses built in optical encoder mechanism for sensing the number of ticks on the motors. This is then fed back to the counters of the RecoNode's decoder logic via 50 pins morphing bus connector and the two 6 pins motor connectors.

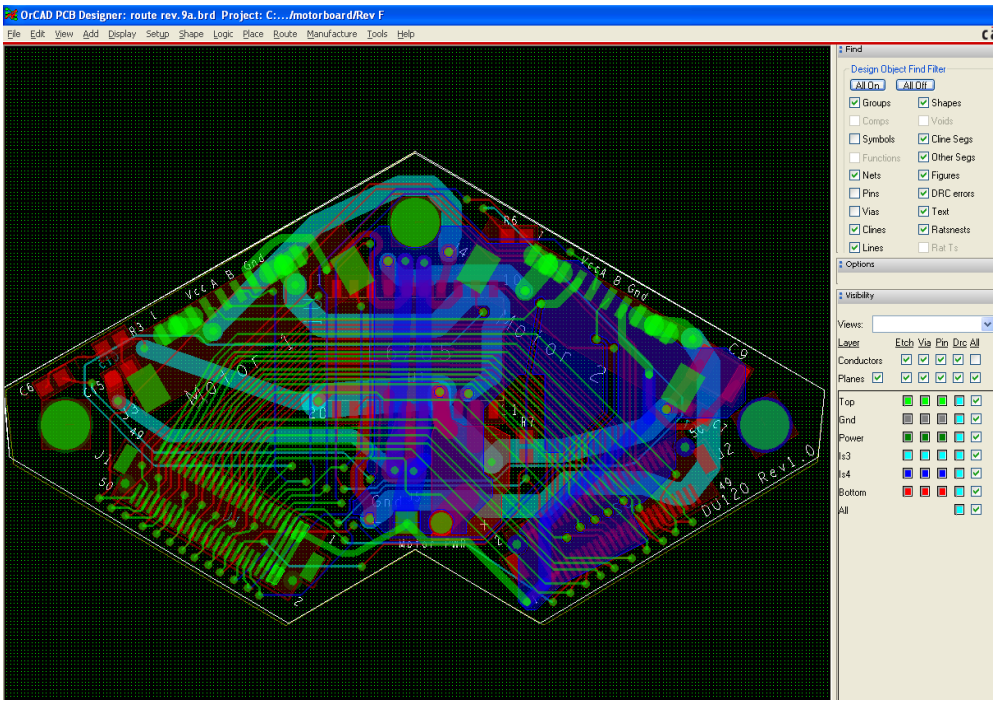


Figure9: Four layer motion control schematics &PCB layout

Prototype of motion control board is shown as figure 10; during the design process we have PCB Layout considerations as following: many PCB layout techniques have been taken into consideration while designing the PCB layout. Special Emphasis has been laid on reducing EMI; signal routing, trace width, footprint designs and board-size constraints.

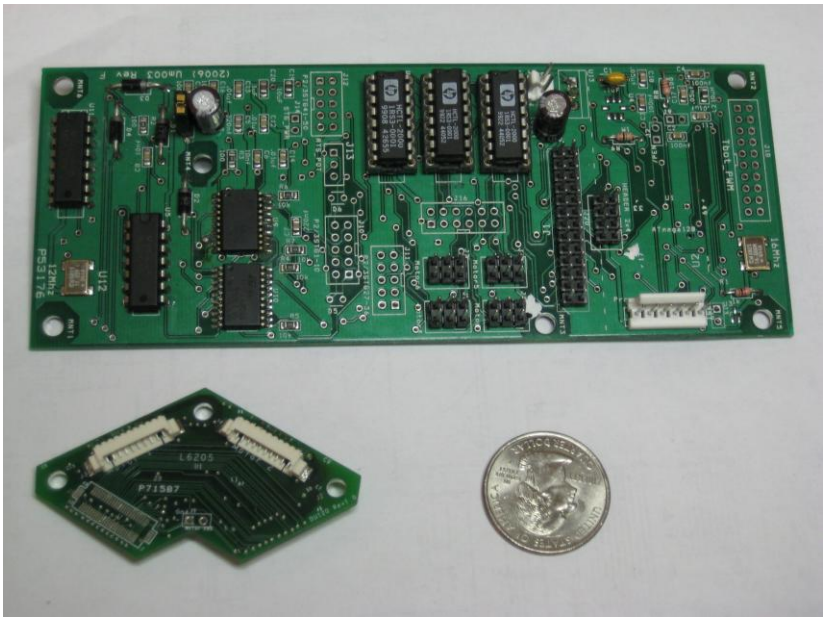


Figure 10: Prototype of motion control board

Standard PCB layout procedures such as grounding unused general purpose I/O pins and routing wires at 45 degree turns to reduce transmission reflection have also been implemented. Board-size was one of the major constraints for us while designing the layout. Since we have 50 pins morphing bus connecting rout will occupy most of the square and double wedge shape reduce even more usable space, space constrain is the biggest concern in design. Regarding trace widths and signal routing, we used an online PCB trace width calculator [18] to find appropriate trace widths for our layout. Standard copper thickness on PCBs is 1oz.; 12A traces in 1 oz copper should be 150 – 200 mils wide. At 3 Amps, a 100 mil wide trace will heat up 10 deg C; a 70 mil trace will heat up 20 deg C. The 3-A motor traces should be at least 70 mil wide and short as possible or preferably 100 mils wide. The power coming in to the L6502 chip is up to 6 A. These traces should be 200 mils wide and also short. For all the other signals, we used a width of 12 mils. It was impossible for us to route all the signals on two layers. Therefore, we used four layers for routing all the signals. We used Vias to

switch between the top layer and the bottom layer.

4.2.4 Power solution

The power supply section is to supply electrical power to the whole RecoNode system including FPGAs base board, sensor/actuator board such as plug-in camera and wireless communication peripherals; we name the power supply board as DU120. Given the FPGAs and other components on-board, numerous voltages are required. The main power supply must provide different voltage levels for the FPGA. There are 3 required voltage levels VCCINT (1.2V), VCCAUX (2.5V), VCCO (3.3V) for Xilinx Virtex4 FPGAs need. Besides the power supply input is from 3.7V Lithium battery. TI provides highly integrated TPS75003 [19] three supplies in one package to power Virtex-4 chips and input voltage range from 1.2V-6.5V which meet our specification. Each supply has soft-start circuits to eliminate inrush current spikes and slow down the ramp time. TPS75003 has two high current buck controllers with about 95% efficiency and a lower power LDO for the 2.5V auxiliary supply. Two integrated buck controllers allow efficient voltage conversion 1.2V to 6.5V for both low and high current supplies such as core and I/O. so that operation is guaranteed even with deeply discharged batteries. A 300mA with Vout range from 1.0v to 6.5V LDO is integrated to provide an auxiliary rail such as VCCAUX on the FPGA. All three supply voltages are offered in user-programmable options for maximum flexibility.

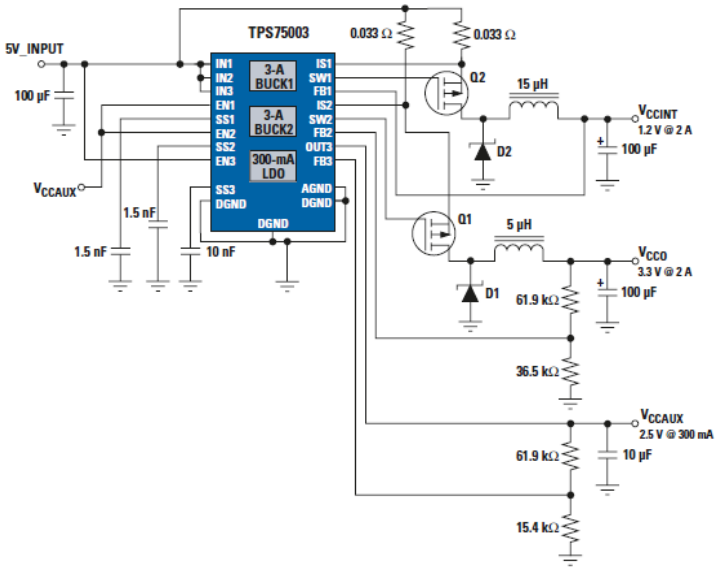


Figure 11: Highly Integrated TPS75003 Triple Supply Powering Virtex-4

Power-on current is not an issue for Virtex-4 because current requirement is equivalent to the quiescent current for unconfigured FPGA, the recommended sequencing for Virtex-4 is VCCINT--VCCAUX—VCCO. The load current and the voltage difference between the input supply and Morphing Bus power determine which dc/dc converter to use. Specifically, Morphing Bus need 5V input to drive sensors and actuators with 3.7 V lithium batteries input, so DC/DC boost up converter is needed. There are two high current output step-up converters we could choose: Linear LTC3426 [20] and TI TPS61032 [21]. TPS61032 is used at the beginning, because TI TPS61032 will probably meet our 1.2A load current at 5V output voltage need. Specifically, after I went through all typical characteristics, TPS61032 is the better to fit our requirement. Like EFFICIENCY vs OUTPUT CURRENT/INPUT VOLTAGE, MAXIMUM OUTPUT CURRENT vs INPUT VOLTAGE [21].

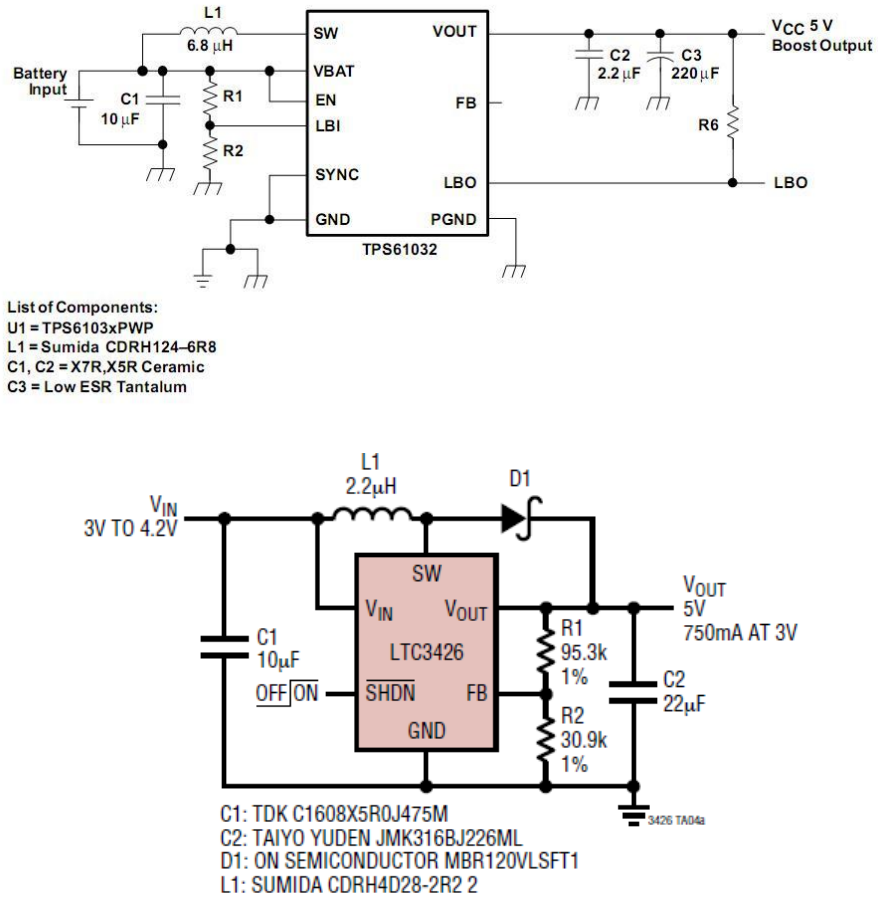


Figure12: 5V DC/DC convertors TPS61032 and LTC3426

The LTC3426 step-up switching regulator generates an output voltage of up to 5.5V from an input voltage as low as 1.6V. Ideal for applications where space is limited, it switches at 1.2MHz, allowing the use of tiny, low cost and low profile external components. And the output 5v at efficiency is estimated 89% when input is 3.7 V, LTC 3426 takes SOT-23 Package where there is a possibility to adopt to old PCB layout. The LTC3426 demands less careful attention to board layout. So we finalize it as out solution.

Another concern is to provide sufficient current to 6 motors; there are 2 Motor power configurations now, serial and parallel. We just use some pads for hard wiring the different configurations. There is little need to change this and changing the

configuration will require a change in the maincon3 and maincon4 and J4 connectors. For serial, parallel, and separate, we only need 5 pads to meet our requirement. Parallel & Serial Mode switch: 2 batteries in parallel, 3.7 V, 3.6-6 A and 2 batteries in series, 7.4 V, 1.8-3 A

PCB Design Considerations follows: As for power supply design, the layout is an important step in the design, especially at high peak currents and high switching frequencies. If the layout is not carefully done, the regulator could show stability problems as well as EMI problems. Therefore, wide and short traces for the main current path and for the power ground tracks are chosen, also large area of copper pour for ground is used in the design. Copper pour is isolated from all tracks, even if they belong to the same net as the copper pour. We want to use the copper pour to create an EMI shield. Normally, copper pour flows over tracks and vias belonging to the same net as the copper pour. I placed the input capacitor, output capacitor, and the inductor as close as possible to the IC. The trace width depends on three design considerations. The first consideration is the capabilities of board manufacturer. The traces need to be wider than their minimum fabrication capability. The second consideration is the required current handling capability, and the third is the impedance. Use a common ground node for power ground and a different one for control ground to minimize the effects of ground noise. Connect these ground nodes at any place close to one of the ground pins of the IC. The feedback divider should be placed as close as possible to the control ground pin of the IC and trace area for FB and VC pins are kept small. Lead length to battery should be kept short and use short

traces to lay out the control ground as well, separated from the power ground traces. This avoids ground shift problems. Plus, Traces carrying high current are direct. The PCB with TPS75003 uses PowerPAD packages which features have included in the design to create an efficient thermal path to remove the heat from the package. As a minimum, there must be an area of solderable copper underneath the PowerPAD package. This area is called the thermal land. In addition, this thermal land may or may not contain thermal vias depending on PCB construction.

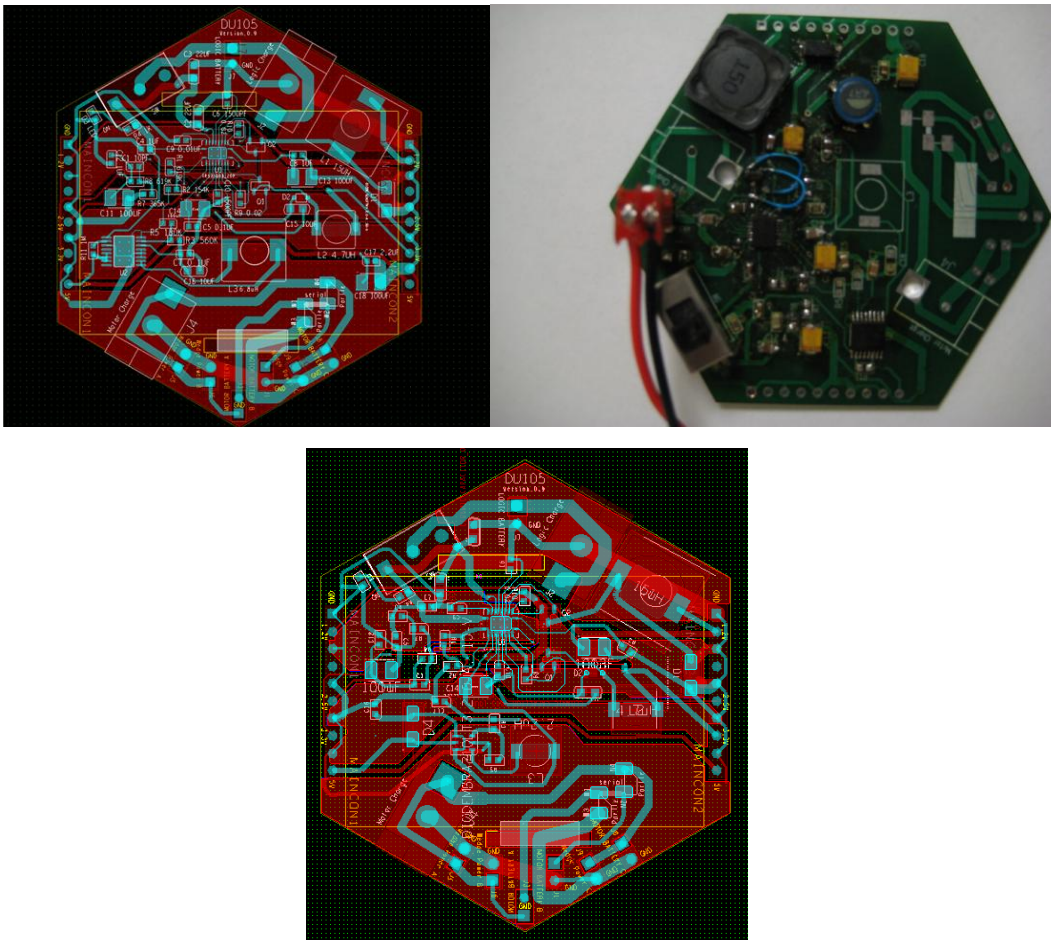


Figure13: power board PCB layout and prototype

4.3 Software infrastructure

RecoNode Robotics system's real-time embedded processor and high-performance FPGA enable us not only to create an application with normal

software implementation of various system functions (e.g. motion control) on the 32 -bit hardcore PowerPC405 microprocessor, but also allow us to write VHDL IP Cores to realize Hardware acceleration to significantly increase the performance of the embedded system built on programmable logic. We would like Real-Time scheduler and driver are reusable which means we want to keep the previous function names.

4.3.1 Porting Scheduler or OS

Since the hardware design is finalized and fabricated, we will mainly focus on developing the software on the RecoNode systems. Portability/Reusability of current software infrastructure is another key concern for the RecoNode system; because we would like to reuse T-bot's basic module function (such as HMI menu, two 3 DOF motion control, and UART communication) across different microprocessors (from Atmega128 to PowerPC 405) and toolsets with minimized development effort.

First all, we need have a real time operating system or scheduler to have multiple applications running on it. In RecoNode, a real time operation system (RTOS) is needed to support for CPU, memory, network, as well as, sensor and actuator. For this work, we have first modified scheduler (Sched2) for PowerPc 405 in Xilinx Virtex-4, and then aim to run wireless sensor networks concentrated RTOS Nano-RK [22] onto PowerPC architecture. Sched2 is a library of routines that provide task scheduling and task dispatch functions on the ATmega128, ARM7, PSX embedded platforms. The sched2 library is modeled after the Chimera Port-Based Object interface [23] for subsystems servers (SBS). It is a coarse object-oriented model of a task in which each task consists of several standard "methods" that govern its real-time operation.

Different from the task scheduling micro-kernel scheduler (Sched2), Nano-RK is thread or task control Nano-kernel scheduler and it is a fully preemptive reservation-based real-time operating system, it includes a light-weight embedded resource kernel (RK) with rich functionality and timing support(shown in the Fig 14) and supports fixed-priority preemptive multitasking for ensuring that task deadlines are met. Another Nano-RK's advantage over other RTOS is network management: Nano-RK has an architecture that supports easy installation of various wireless networking link layer protocols including RT-Link [24] for collision free Real-Time communication and b-mac for low-power contention based communication. Besides Nano-RK takes the approach of socket-like abstraction, which is convenient for software developers. Currently, Nano-RK runs on the FireFly [25] Sensor Networking Platform as well as the MicaZ motes, both of them are ATmega128 based.

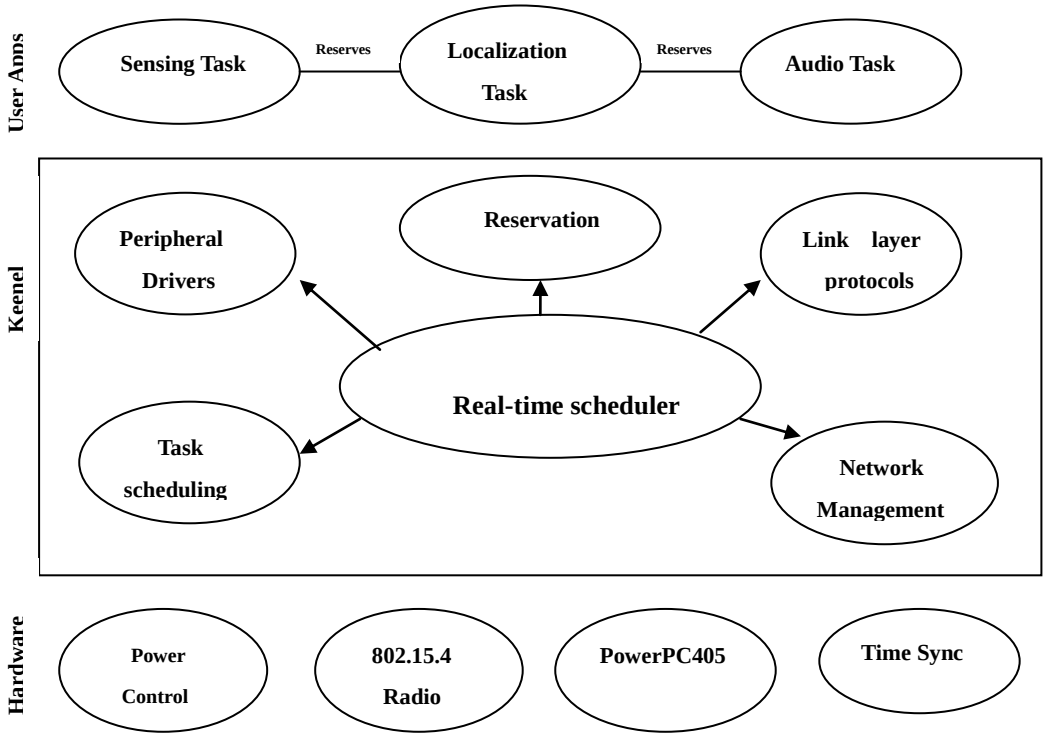


Figure14: Nano-RK architecture figure showing user applications and RT-Link.

For a scheduler, we need using timer to provide periodic tick interrupts. There are three timer-event interrupts options that I could use: XPS TIMER IP, Fixed Interval timer (FIT) or Programmable-interval timer (PIT) to do that [26]. PIT timer is suitable for our case. The programmable interval timer (PIT) is a 32-bit register that is decremented at the time-base increment frequency. The PIT register is loaded with a delay value. When the PIT count reaches 0, a PIT interrupt occurs. Optionally, the PIT can be programmed to automatically reload the last delay value and begin decrementing again. In order to configure the PIT, two special registers are going to be used. The first one is the Timer Control Register (TCR) and the second one is the Timer Status Register (TSR). The following code shows the functions we used.

```
  /* Initialize exception handling */
  XExc_Init();

  /* Register PIT interrupt handler */
  XExc_RegisterHandler(XEXC_ID_PIT_INT, (XExceptionHandler)pit_timer_int_handler, (void
*)0);

  /* Initialise and enable the PIT timer */
  XTime_PITSetInterval( 0xffffff00 );
  XTime_PITEnableAutoReload();

  /* Enable pit interrupt */
  XTime_PITEnableInterrupt() ;
```

4.3.2 Drivers

Besides porting the scheduler, I also duplicated T-bot application codes including UART and PID control from the Atmega128 to PowerPC 405, considering they are all hardware depend code, For software compatibility, it is important to keep the API functions names and calling conventions, but change the hardware-specific I/O access

code to work with the PowerPC platform. The library generator--Libgen in Xilinx EDK-- automatically generates Xilinx libraries such as xparameters.h, which contains important system parameters used by the application drivers and the Board Support Package (BSP). The input file of Libgen is MSS which defines the drivers associated with peripherals, standard input/output devices; interrupt handler routines, and other related software features. The Xilinx ISE employs the GNU GCC (powerpc-eabi-gcc) compiler for the PowerPC processor.

a. UART Software Module

To implement serial communications, I designed some compatible function codes for the UARTLite hardware IP core. I use UARTLite function library directly under EDK XPS. One big difference between the UARTLite and the UART for the Atmega128 is the UARTLite has a fixed baud rate that is set in the HDL code. The baud rate is set through the UARTLite PCORE under the XPS project interface to the FPGA.

b. Menu Software Module

Menu acts as an interface between PC and RecoNode via the UART serial port and print results out on HyperTerminal. So we could get updates information like motor speed or trajectory needed to predict the unknowns in each stage. Currently, we successfully transfer initialization function menu_init(uint8_t index), toggle display of measured position (Q_MezG), toggle display of measured velocity (Qd_MezG)

c. PD controller Software Module

By customizing FPGA logic, we can accurately control torque, velocity and

position with feedback showing on HIM, and implement classic control algorithm such as PD/PID. We can implement functions such as supervisory control and trajectory generation for multi-axis coordination like RecoNode six degree of freedom (DOF) and accurate velocity/acceleration profiles for smooth movements. Sample of PD control module is shown as below:

```
char pd_cycle(uint8_t index)
{
  int32_ttorque[6], tmp;

  read6Encs();    /* get current robot pos and vel */

  /* computer joint torques */
  torque[0] = KPosG[0]*(Q_RefG[0] - Q_MezG[0]) - KVelG[0]*(Qd_MezG[0]);

  /* re-compute coupled motor torques */
  tmp = (torque[0] + torque[1])/2;
  torque[1] = (torque[1] - torque[0])/2;
   torque[0] = tmp;

  /* Write the PWM values */
  /* Channel 1 on first wedge */
  if (torque[0] < 0){
    if (torque[0] < -PWM_MAX)
      torque[0] = -PWM_MAX;
    MOTOR1_PWM_VALUE = -torque[0];
    /* set direction for reverse */
  } else {
    if (torque[0] > PWM_MAX)
      torque[0] = PWM_MAX;
    MOTOR1_PWM_VALUE = torque[0];
    /* set direction for forward */
  }
```

4.3.3 PWM Generator IP

We implemented Pulse Width Modulation (PWM) generation module in hardware by VHDL. And have it working with the software – brake, forward and

backward .The two key parameters of PWM are the duty cycle and period of each pulse can be set in software. Fully digital accumulator based pulse-width modulation (PWM) permit to autonomously control frequency parameters independently without the need for additional processor resource.

a. PWM generator

The PWM code is imported as a pcore to be used in an EDK project for the DU100. I created a PWM module (attached) that has two parameters:

1. pwm_period : A 16 bit number that is based off of the system clock. This value determines how frequently the pulse happens

2. pwm_duty_cycle: A number that will subtract from pwm_period above. Pwm_duty_cycle's resolution increases as the pwm_period number gets larger.

For example, if the pwm_period is the same as the system clock (a value of 1), then the duty cycle is either 0% or 100%. There will be two separate commands to write the PWM period and duty cycle in software. In the user_logic.vhd, we define PWM pulse duration register 0x<BASE_ADDRESS> + offset for PWM with read and write ability and the register is indexed as [0:31], all data is MSB: LSB. Correspondently, in the application code PDCTRL.C we define MOTOR1_PWM_VALUE with the same location, so we could set PWM pulse duration in hardware by software. DC motors controlled by PWM will have a PWM period that is much longer (1 ms) than the clock of the embedded system (10 ns) so the above two values should work out for now. Clk is the system clock. Clk_period is the testbench clock and pwm1 is the PWM output.

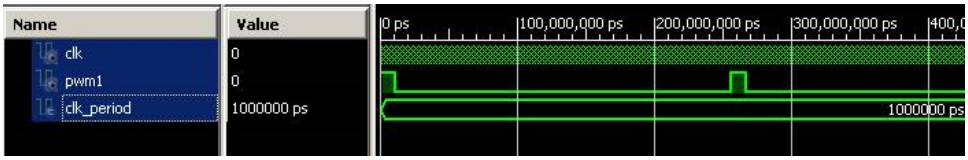


Figure15: Behavior simulation

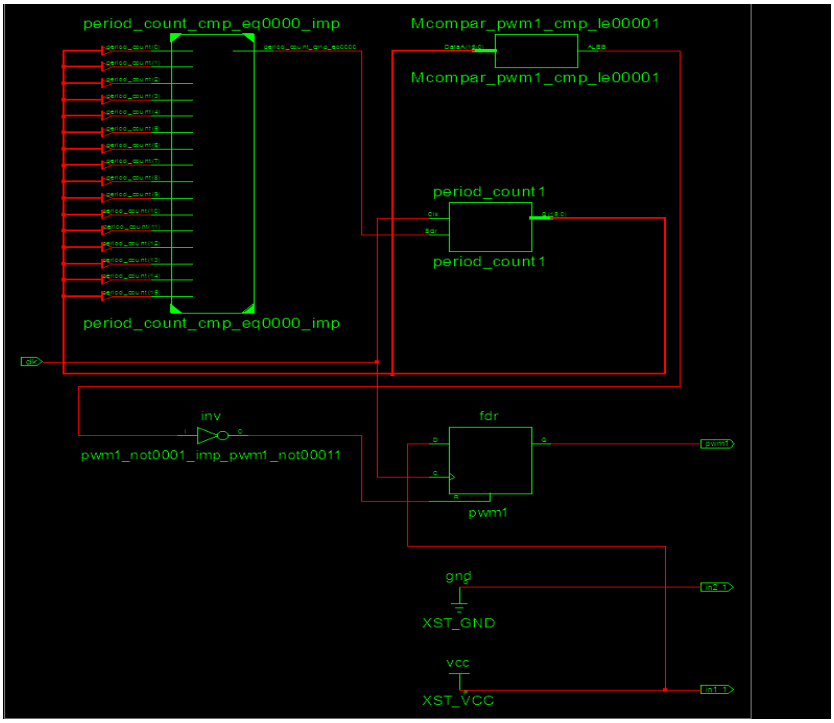


Figure16: screen shot of a simulation and RTL view

It should be real easy to add in as many PWM channels as necessary. The important parts of my project are:

1. Add the PWM core (drag and drop the core into project, connect the core to the PLB bus) - Generate addresses so that the core has its own address (addresses tab in XPS, then push the Generate Addresses button)

2. Select the Ports tab, expand the PWM core and notice the only port that is not connected. Make this port external

3. Open the MHS file to notice the newly added port. Grab the port name from the MHS and add it to the UCF placing a LOC constraint on F14, then rerun the design through implementation and download the BIT file

a. Interfacing the FPGA with the L6205 drive IC for Motor Control

The FPGA will process the PWM program and the output will be given to enable pin of H-bridge chip L6205 which activate the L6205 chip (L6205 Datasheet) and controls the speed of the motor. Fig17 shows the truth table we use to get the L6205 to perform the different movement operations such as "Forward", backward" and Brake etc.

TRUTH TABLE

INPUTS			OUTPUTS	
EN	IN1	IN2	OUT1	OUT2
L	X	X	High Z	High Z
H	L	L	GND	GND
H	H	L	Vs	GND
H	L	H	GND	Vs
H	H	H	Vs	Vs

X = Don't care

High Z = High Impedance Output

Figure17: the truth table of the L6205

We notice that when the outputs are in a high impedance state that the motor will freely run. And when the outputs are both tied to Vs or are both tied to GND, and then the motors will brake. So we modify the motor_wedge pcore to handle this extra functionality.

Our method to accomplish this is to set the direction of the bridge with IN1, IN2 and to generate PWM by pulsing the ENABLE line. Because EN Signal enables the L6205 out1 and out2 signals on motor 1, a value of 1 for the setting register turns on the brake. Here is a portion of the PWM VHDL code added brake function:

```
INA1 <= pulse_duration_msb_1 AND (NOT brake1);

INB1 <= pulse_duration_msb_1 NOR brake1;
```

b. **Interface IP core with software on PowerPC405**

We can do a memory write command to PWM's address will modify the full 32 bits under XMD shell tool or SDK environment.

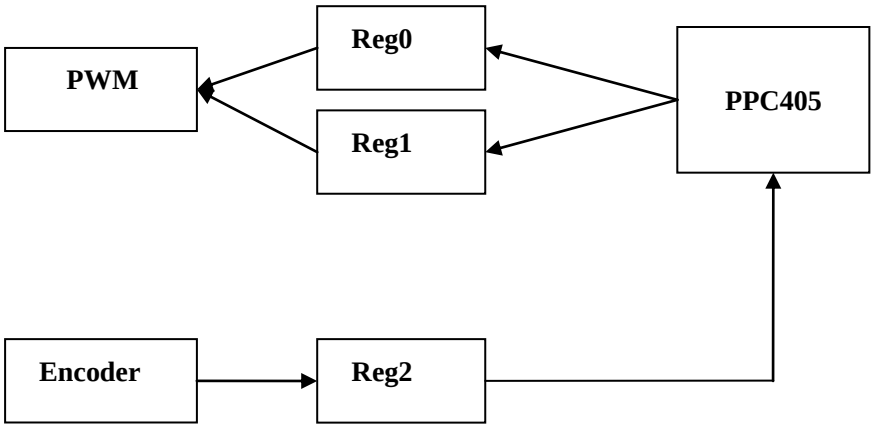


Figure18: software accessible registers schematic

Where Reg0, Reg1 and Reg2 are 32 bits software register which are accessible by Power PC405, Reg2 is the register will show the motors' current values of the encoders and this is actually another register for presetting the motors' encoder values. Reg0 register containing the "frequency" (pwm_period) of the PWM pulses. The number represents an absolute upper value before starting the count over again. Reg1 "pwm_duty_cycle" value cannot be greater than the value in Reg0. The PWM pulse happens at every pwm_period - pwm_duty_cycle.

c. Generating custom IP and Adding OWN IP to PLB/OPB bus

Different from using Uartlite IP core directly from existing library, we need to customize our own IP core for PWM generator and encoder. There is a Create/Import IP wizard helps us create our own peripheral, set bus system interface and then import

custom IP into XPS design, the next screen is the IP Interface Services screen. The IP Interface (IPIF) provides a variety of services for easily connecting custom peripheral to a processor bus. A detailed description of each service can be found in the IPIF Features document2. For our application, we will use the User logic software register services. The peripheral will appear under the Peripheral or Project Repository folder in the IP Catalog after all the steps finished. After completing this module, we will be able to have functional IP attached to PLB bus which enables PowerPc talk to custom peripheral logic interface for sensors and actuators. This is the essence of RecoNode hardware structure.

4.3.4 Quadrature decoder IP

a. Encoder logic

In this part we have designed a quadrature decoder/counter interface IC like logic that performs the decoding, counting, and latching in digital motor control systems, employing a Virtex 4 FPGAs device.

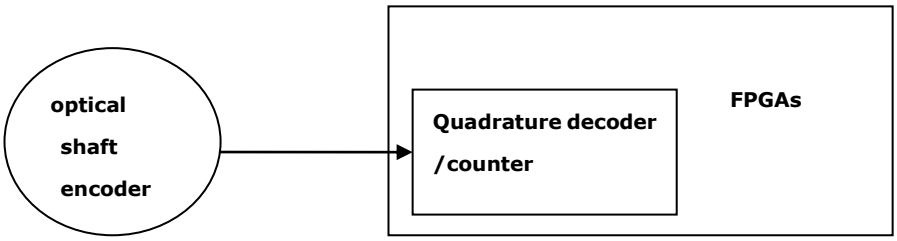


Figure19: Quadrature decoder /counter logic inside Virtex-4 FPGAs

When DC motor shaft is moving, build in encode will convert rotary mechanical motion into a digital output .Then we have two channels from a rotary encoder: CHA

and CHB the direction of rotation is indicated by a +/-90 ºphase (quadrature phase) difference between the two channels. For Terminatorbot, HCTL 2000 [27]Quadrature decoder /counter are used to interface an optical shaft encoder to a microprocessor's system, it contains 16 bit counter, 16 bit Latch & Multiplexer.

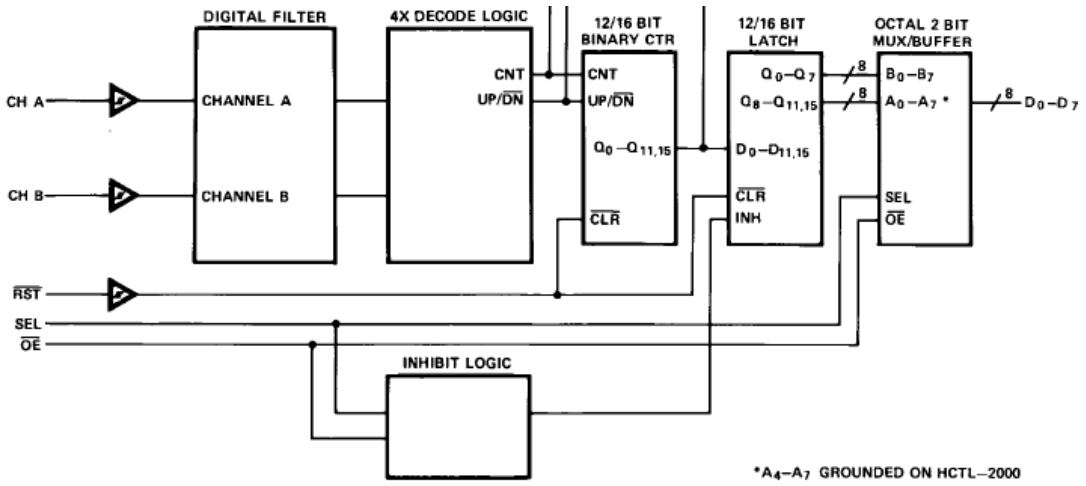


Figure20: A Simplified Logic block diagram of the HCTL 2000[27]

When it comes to Quadrature Decider for RecoNode, we adopt similar features of Htcl2000 in VHDL: Input noise filter, 4X decoder, 32bits counter, and counter register to store the encoder count value.

b. Shift Registered Digital noise Filter

Clearly the objective of the filter is to eliminate input signal's short duration noise spike and switch chatter completely. This is a achieved by detecting only the first change of the signal and ignoring all subsequent activity on the same signal until the other switch also changes state[28]. In other words, the filter's output can change only after its input had the same value at three consecutive triggering clock edges. The filtered signals are then passed to a four-bit delay filter (Fig.21). Therefore, the filtered output waveform is four-bit shift registered and can change only if the input

has the same value for three consecutive rising clock edges

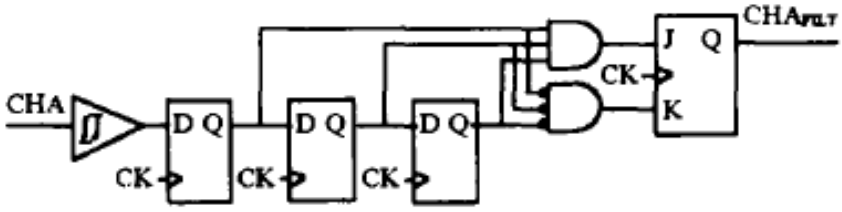


Figure21: Digital noise filters architecture [28]

Each input channel(A and B) is filtered by a separate copy of the digital noise filter, a VHDL code description of the functionality of the shift register and J-K flip-flop with gated inputs in Figure 21 is as following: the filtered output is computed from the right most 3 bits of the shift register:

```
 if enca_q_1(2 downto 0) = "111" then
  enca_filtered_1 <= '1';
elsif enca_q_1(2 downto 0) = "000" then
  enca_filtered_1 <= '0';
else
  null;
end if;

if encb_q_1(2 downto 0) = "111" then    ---filter
  encb_filtered_1 <= '1';
elsif encb_q_1(2 downto 0) = "000" then
  encb_filtered_1 <= '0';
else
  null;
end if;
enca_q_1 := ENCA1 & enca_q_1(3 downto 1); ---right shift
encb_q_1 := ENCB1 & encb_q_1(3 downto 1);
enca_q_2 := ENCA2 & enca_q_2(3 downto 1);
encb_q_2 := ENCB2 & encb_q_2(3 downto 1);
```

c. Decoders

Examination of the CHA, CHB waveforms as in Fig 22 shows: CHB lagging CHA: CHB = ‘1’ at CHA rising edge and CHB leading CHA: CHB = ‘0’ at CHA

rising edge.So CHB can be used to indicate direction, if we can hold the value on CHB at CHA rising edge

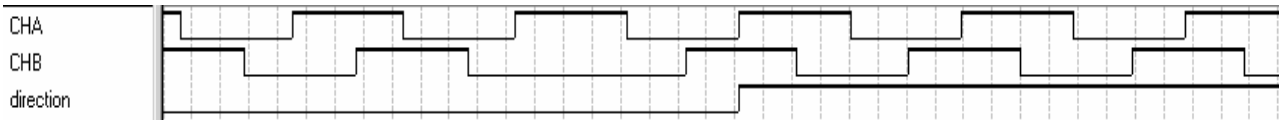


Figure22: two 90 degree phase different encoder signals

But the problem with the simple detector is that the direction indication is synchronized to the rising edge of CHA. We adopt decoder logic of TRC 040 for PUMA560 as following, where we double the pulse rate from the XOR function by 2 to give 4 times the pulse rate

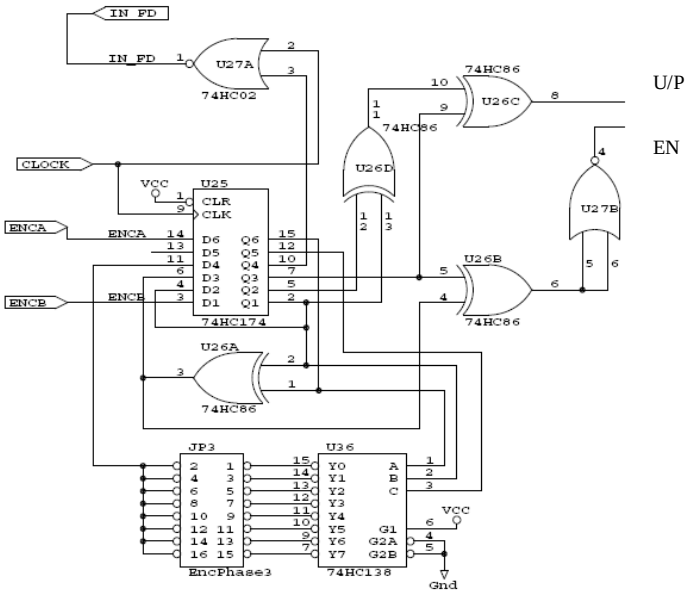


Figure23: TRC040 decoder logic

```
-- Process to generate the clock enable (ce) and the up / down (ud)
-- signals for the encoder counter.
enca_d1_1 <= enca_filtered_1;
enca_d2_1 <= enca_d1_1;
encb_d1_1 <= encb_filtered_1;
encb_d2_1 <= encb_d1_1;
enca_d3_1 <= enca_d1_1 xor encb_d1_1;
encb_d2_1 <= encb_d1_1;
```

```
ce_1 <= enca_d3_1 xor (enca_d1_1 xor encb_d1_1);
ud_1 <= enca_d3_1 xor (encb_d1_1 xor encb_d2_1);
```

Also, we can double the pulse rate with Four times decoder which will increases the effective resolution of the encoder by a factor of four, compared to simply counting positive edges of A (or B) directly. To accomplish this we need to build an edge-detect circuit that detects both positive-going and negative-going edges – a double-edged detector. Specifically, the 4X decoder logic uses the system clock to decode the incoming filtered signals into count information. The decoder samples the combination of outputs change of the Channel A & B samples. Based on the past binary state of the two signals and the present state, it asserts a count enable signal (CE) and direction signal (UD) to the position counter until the next triggering clock edge. Channel A leading Channel B results in counting up. Channel B leading channel a results in counting down.

d. 32 bits Counter

The counter module requires a simple 32bits up/down counter with an enable input. Such a counter was described in the following VHDL code segment.

```
-- The counting of the encoder values happens here
        when others =>
          cnt_value_1 := slv_reg1;
          cnt_value_2 := slv_reg3;
          if (((encb_d1_1 xor encb_d2_1) = '1') or ((enca_d1_1 xor enca_d2_1) = '1')) then
            cnt_digit_1 := '1';
          else
            cnt_digit_1 := '0';
          end if;
          if ce_1 = '1' then
            if ud_1 = '1' then
              cnt_value_1 := cnt_value_1 + cnt_digit_1;
            else
              cnt_value_1 := cnt_value_1 - cnt_digit_1;
```

```
    end if;
  end if;
```

Notice that the HCTL2000 includes an output multiplexer and double buffer inhibit logic. The multiplexer is for the 12 bits output of the double buffer register to the 8-bit data bus output, while the double buffer inhibit logic is designed for reading a counter's output while it is counting (read on the fly). If the counter's output were read directly and the read occurred while the counter was in the process of changing value, the value read might be incorrect. For our case, there is no need to consider this issue because we use 32 bits register and counter. The simulation is done as following in ISE's simulator called Isim. The testbench was created using ISE's automatic testbench creator.

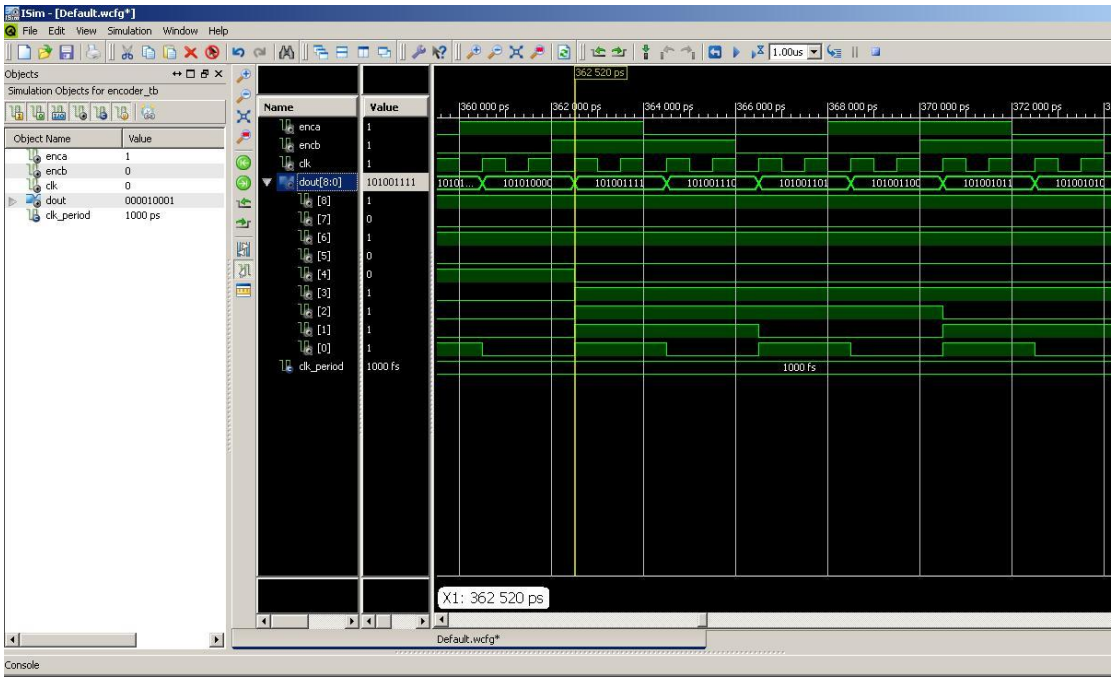


Figure24: Simulation waveform for driving-signals of DC Motors

4.3.5. UART IP

We use RS232 serial cable and serial communication utility (HyperTerminal) to debug our motion control design and wireless communication setting. For UART

serial communication part, RS232 signal is actually connected to the FPGA. Our approach is to add existing peripherals IP called UARTLite into building PLB/OPB bus of RecoNode system. UARTLite is written in VHDL at building XPS library; this will be attached to the OPB bus as follow and we can find the view of IP cores connection with PPC405 under XPS

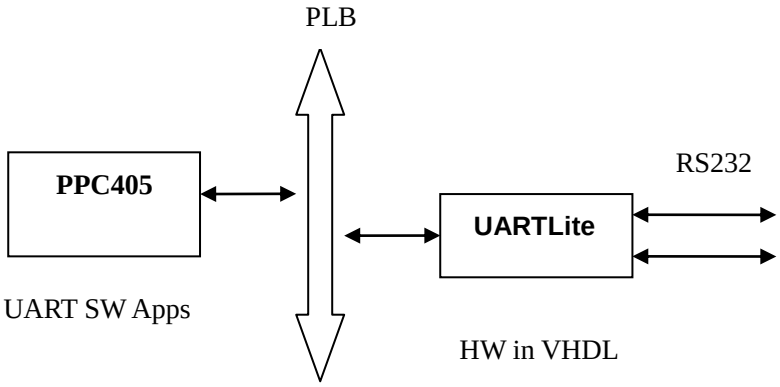


Figure25: UART IP core connecting to PPC

The process is to select and add the opb_uartlite from project menu. UARTLite has a fixed Baud rate that is set in the HDL code, thus, Baud rate will be set through UARTLite PCORE under XPS project interface.

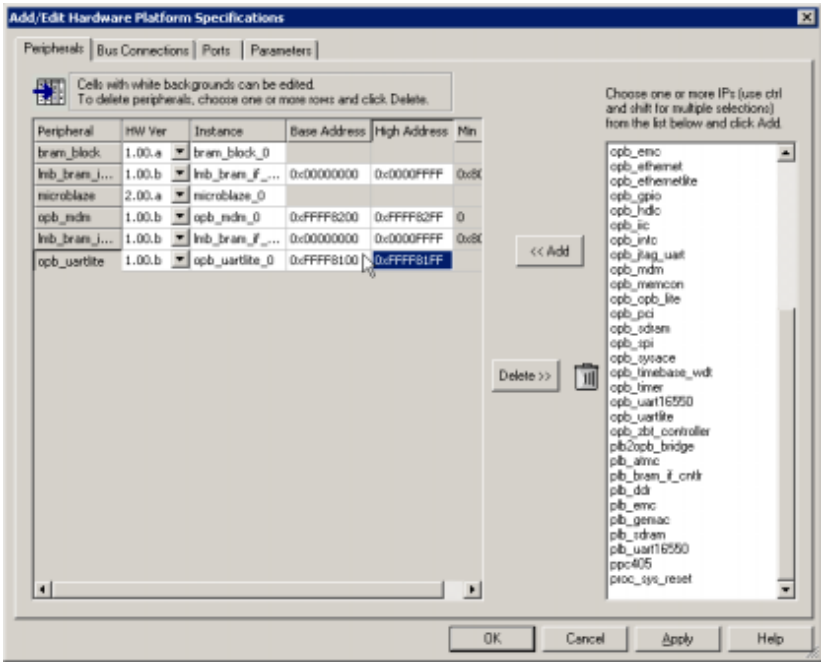


Figure26: UARTLite configuration

The instant UARTLite address is defined in Xparameters to variable in code UARTLITE_BASEADDR = XPAR_RS232_BASEADDR; and there are UARTLite function library we use directly like XUartLite_RecvByte (UARTLITE_BASEADDR) and XUartLite_initial function. For hardware interface, I just set up the Brainstem interface with DU100, so UART hardware configuration for DU100 is ready. Later I have UART Lite to DU100 XPS project embedded design on the board with UCF setting with respect to DU100. After running a simple print command should verify that everything is working: it prints out on screen of HyperTerminal.

```
#include "xparameters.h"

#include "stdio.h"

//====================================================

 int main (void) {

    print("-- Entering main() --\r\n");
```

```
    /*
     * Peripheral SelfTest will not be run for RS232_Uart
     * because it has been selected as the STDOUT device
     */
    print("-- Exiting main() --\r\n");
    return 0;
}
```

Chapter 5 Test and Verification

5.1 Board verification

My main task is to test out FPGAs baseboard functionality by download Ring Oscillator VHDL logic and probed waveform signal on assigned morphing bus pin on DU100. And specifically we use JTAG on the board to download bit stream by IPACT (static configuration) after all are verified as having correct signal, power, and continuity connectivity and we could read device code if connection and chip are all right.

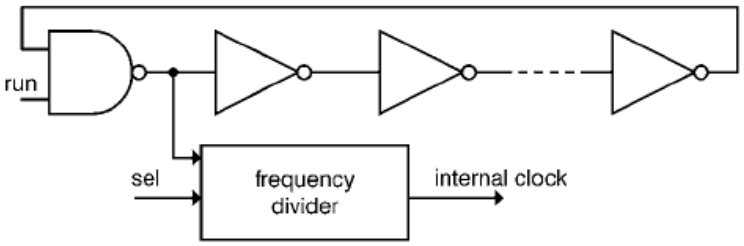


Figure27: Internal free-running clock generator made from ring oscillator

```
entity ERINGOSC is
  generic(len:integer:=1600;   --len even
         invdel:time:=5ps);
  port(
    RUN : in std_logic;
    CLK : out std_logic);
end ERINGOSC;

architecture STRUCTURAL of ERINGOSC is

component EINVS
  generic(len:integer;
         invdel:time);
  port(
       A   : in std_logic_vector(len downto 1);
       Q   : out std_logic_vector(len downto 1)
```

```
      );
end component;

signal Atemp,Qtemp : std_logic_vector(len downto 1);
attribute KEEP : string;
attribute KEEP of Atemp,Qtemp : signal is "true";

begin
  Atemp(1)<=not(RUN) nand Qtemp(len) after 2*invdel;
  Atemp(len downto 2)<=Qtemp(len-1 downto 1);
  DELAY : EINVS
    generic map (len, invdel)
    port map(Atemp,Qtemp);
  CLK <= Atemp(1);
end STRUCTURAL;
```

Also PROM test is our main task will be described as following. Specifically, the oscillator is implemented as a ring oscillator with one inverter replaced by a NAND-gate as shown in Fig. 27. The NAND gate is used to enable power down by shutting down the oscillator. A frequency divider is inserted to give the ability to select between 4 clock rates f, f=2, f=4 and f=8, where f is the output frequency from the ring oscillator.

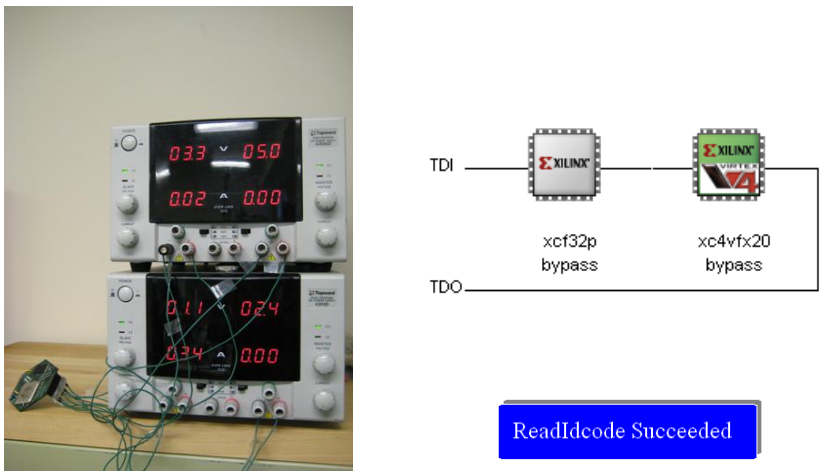


Figure28: Base board test setup

5.2 PROM verification

Another key feature is the RecoNode ability to boot a new system configuration

file at reset. Xilinx FPGAs are volatile because they are based on SRAM technology. That is, the device loses its configuration if the power to the device is turned off. After RecoNode's FPGA logic has been configured, it is often necessary to retrieve new user-defined configuration data that issued by the FPGA during operation. The data needs to be retrieved from an external storage device like PROM without a control circuit is required to interface to the storage device.

Xilinx configuration PROM is generally used to store an FPGA design, which is downloaded to the FPGA upon system power-up. In most cases, this is the PROM's only function, and its capacity is usually not fully used by the FPGA design. Besides PROM is also the component to store bit stream for Dynamic Partial Reconfiguration. The design here describes how bit stream- Dynamic partial reconfiguration data can be stored and retrieved from Xilinx configuration PROMs using existing connections and only one user I/O. This reduces the FPGA pin count, component count, board space, and overall system cost. The user-defined data can be a bit stream revision code and so on. A Jbit or Perl script might be created that automatically modifies existing configuration PROM files with user-defined data with optional bit swapping.

5.2.1 PROM and FPGA Connections

First we come to the mode configuration of PROM; here is a truth table for the mode pins:

```
Configuration Mode   | M2 M1 M0

------------------------------

Master Serial        | 0  0  0
```

```
Slave Serial          | 1  1  1   -- Not needed for us

JTAG                  | 1  0  1
```

The PROM can only program the FPGAs in either Master Serial or Slave Serial. If it is Slave Serial, then the FPGA and the PROM have to get a clock from an external source to drive the configuration. DU100 is truly designed for Master Serial Mode since the FPGA's configuration clock (M14) is connected to the PROM's clock CLK pin and M 0/1/2 are tied to ground which means Master Serial from truth table.

Figure 29 clearly shows the connections necessary to create a suitable interface between the PROM and the FPGA.

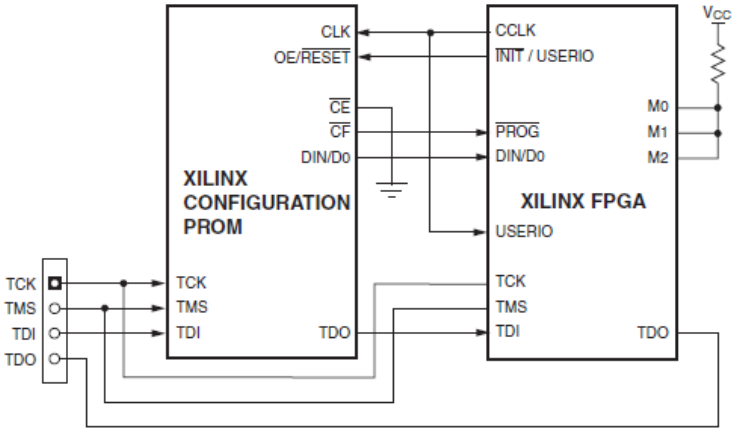


Figure29: PROM and FPGA Connections with Control Signal

Considering I already tested out PROM successfully, I refer Xilinx ML405 design into DU100 PROM verification as following.

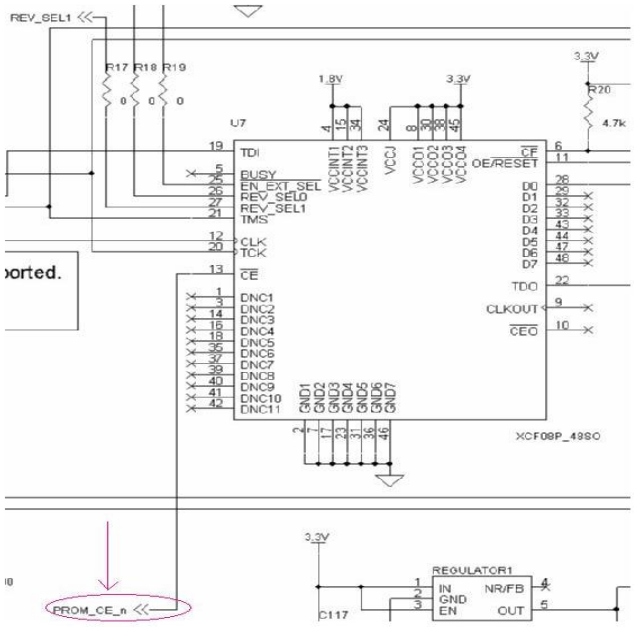
1. The DU100 JTAG config is all right as ML405: specifically FPGA TDI connected to PROM TDO.

2. CE_n (PROM Pin 13) was the one we concern the most, ML405 use Config Switch and pull up resistor to ground it which is similar as I tied CE_n pin on the PROM to

ground via a pull down resistor

3.PROM CLK should be connected to FPGAs CCLK (FPGAs PinM14).

4. Also, I checked CF_n (PROM Pin 6) has to be ground as we expected.



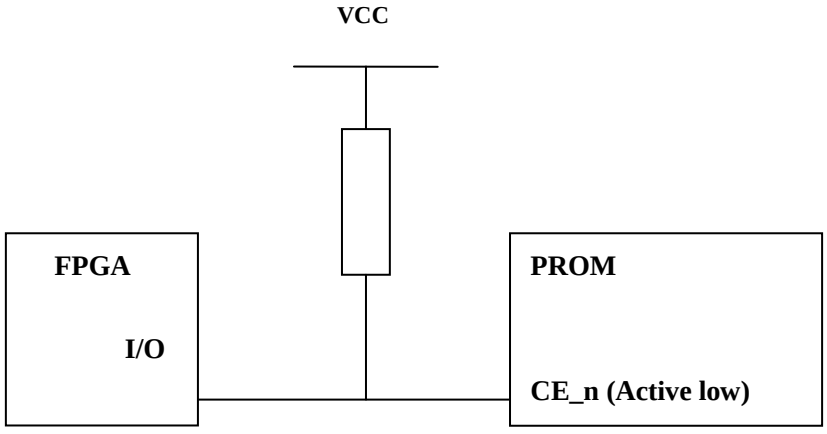


Figure30: PROM connections with improvement on DU100 baseboard

From the above schematic, I did find that the CE_n pin of the PROM goes nowhere. From [29], and there are two options on how to connect the PROM CE pin:

1. Connect the CE pin to GND.

2. Connect the CE pin to a User I/O pin of the FPGA. This option requires an

additional I/O pin to the solution; however, it allows the PROM to be put into standby mode to allow power saving. The FPGA needs to get its bit stream from the PROM. The PROM's clock gets enable by the CE_n pin. All users I/O on the FPGA are tri-stated before the FPGA is configured with its bit stream so the CE_n pin is also tri-stated which means that the FPGA will never get its bit stream. The recommendation is to tie CE_n pin to ground through a pull down resistor and keep it connected to the user I/O as following, this way we can configure the FPGA on power up and have the user I/O configured to be tied high. Once the FPGA is configured, the PROM can still be in power saving mode but we will also be able to reconfigure if we need to.

5.2.2 Programming PROM

After hardware requirements are met, software flows for generating and programming PROM files is described as follow: The iMPACT software tool converts the bitstream targeted to an FPGA family into a PROM file. Figure 31 shows the options available for downloading the PROM file into the PROM device [30]

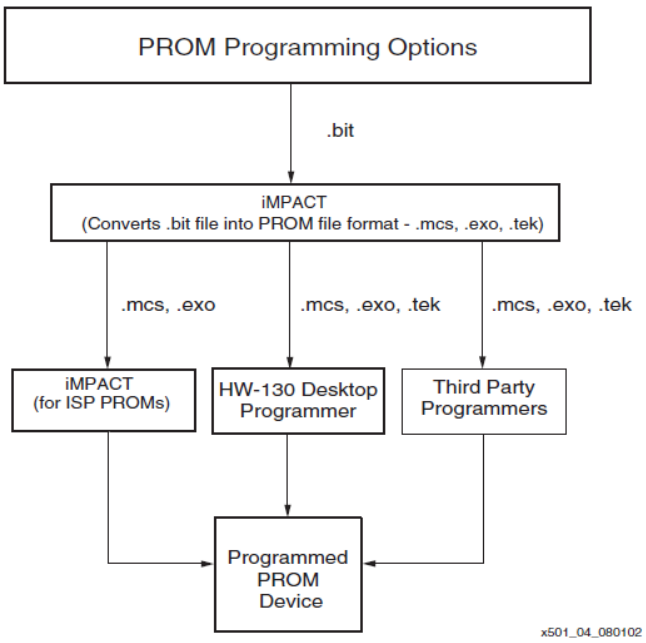


Figure31: PROM Programming Options [30]

Here we use iMPACT to create and download a PROM file. iMPACT accepts any number of bitstreams and creates one or more PROM files containing one or more daisy chain configurations. In iMPACT, a wizard enables us to do the following: 1.Create a PROM file by adding bit streams into PROM File Formatter.2.Generate File in the MCS file format .With the resulting .bit, .mcs and a MSK file generated along with the BIT file, we are ready for programming DU100 using iMPACT.

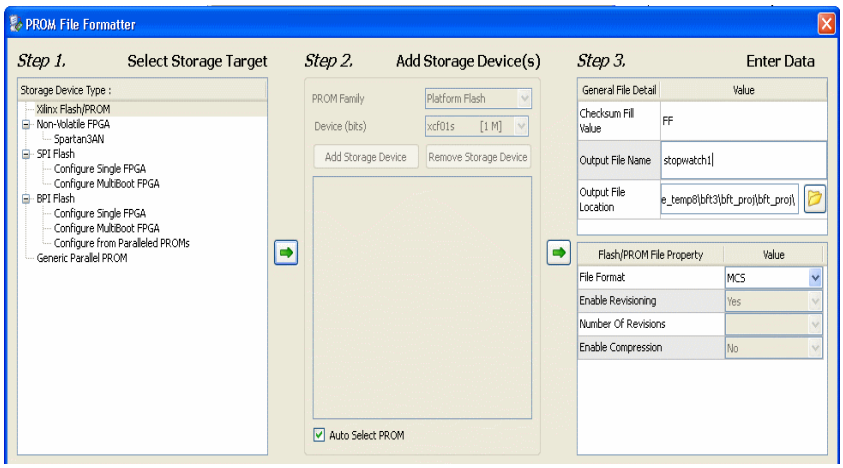


Figure32: PROM File Formatter

5.3 Battery board validation

5.3.1 Battery and charger

Our need for the battery is 7.4V, 5or 6 amp discharge rate to drive DC Motor used in DU100. These batteries aren't rated for the peak discharge rates we need. We may want to put 4 UBP002 batteries in series/parallel rather than 2 UBP001 batteries in series for the motors. This should give us greater peak current capability. So our design seems to accommodate it, considering it is unlikely for six motors working at the same time. When it comes to physical space size to hold batteries between base board and power board, 5 UBP002 batteries are 31mm tall, while 1 UBP002 and 2 UBP001 batteries are 28.2 mm tall. This obviously affects my choice of board-to-board connectors. I choose Semtac board stacker with space could hole 5 batteries at most. There are three height of a board stacker we need to consider tail, post and board space.

On-board charging batteries on RecoNode is other concern for us; here we use The MAX1908 evaluation kit (EV kit) which is an accurate and efficient multichemistry battery charger. It uses analog inputs to control charge voltage and current. The EV kit can charge 2 to 4 series lithium-ion (Li+) cells with a current up to 3A. The EV kit provides outputs that can be used to monitor the input current, the battery-charging current, and the presence of an AC adapter.

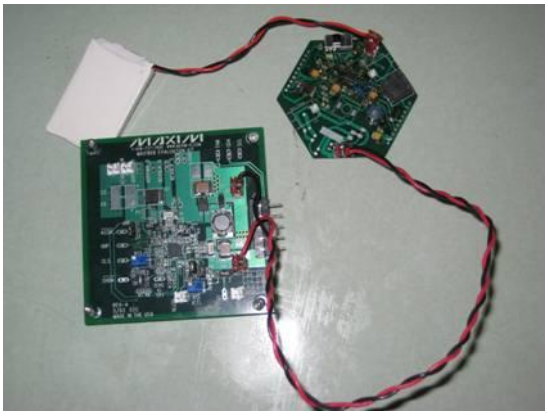


Figure33: DU105 with MAX1908 evaluation kit

5.3.2 Load Test

Our RecoNode electronic systems use a regulated DC power supply to provide DC voltages to its circuitry like FPGAs and other components. The circuitry draws current from the power supply and is said to “load” the supply. The product of the output voltage and current to the circuitry is the power usage or load. I have DU105 tested in lab workbench condition (digital multimeter and oscilloscope).First we test the functionality of the power board DU105; including TI TPS75003 and TPS61032, TPS 75003 output 3 voltages (VCCINT (1.2V), VCCAUX (2.5V), VCCO (3.3V) as we expected. But we face problem from TPS61032 5V after test the ringing and inappropriate layout lead to chip damage, to remove this bug, I added filter capacitor at the input and the Schottky Diode to SW and output with low current limit from the bench supply, but tps61032 was still damaged, high frequency large ring is the reason shock internal PMOS switch. Then we decide to switch to other alternate IC such as LTC 3426 from linear technology, and it is working with current board. Then we will implement function test with load connection to baseboard.

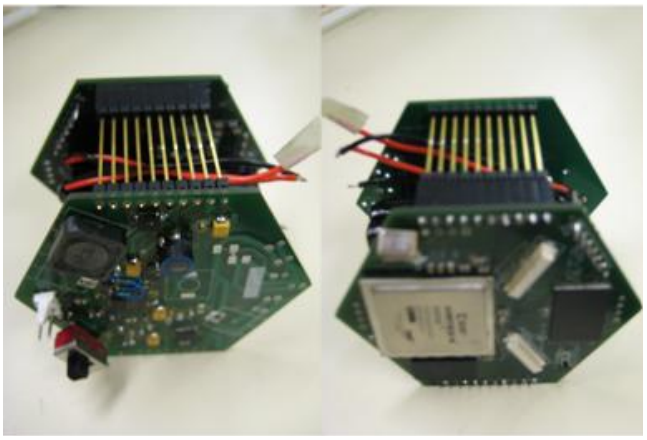


Figure34: Battery board with FPGAs load

5.4 Motion control validation

Here we test RecoNode's motion control part in hardware and software together. RecoNode has a motor control module allowing it to control multiple types of motors. The motor control module contains trajectory generation software running on the PowerPC and a PWM generator, Encoder counter, PID controller hardware logic module inside the FPGA chip. Figure 34 shows how it works as a closed-loop control system. The motor module consists of 5 major components, the trajectory generator in the control area (the PPC), a PD or PID controller, an encoder counter, a PWM generator and the physical motor/power circuitry. The parts of major interest to us are the three modules implemented on the FPGA; the PD controller, the encoder counter and the PWM generator. The PD controller takes in the data from the trajectory generator and the encoder counter to determine where the motor should be and where it is. It uses this information to feed the PWM generator a pulse width. To implement encode counter, we created a VHDL module to replace the HP IC HCTL2000. The PWM module was custom coded to create a modulation at any frequency desired; this

allows the motor module to use many different types of motors. Because their is an FPGA on RecoNode, the PD controller was implemented in it because it allows CPU clock cycles to be saved for other tasks, and the time granularity can be more easily adjusted on the FPGA based controller

PWM-Encoder IP which is built inside the FPGA fabric to drive the motor with the help of power amplifier circuit L6205. The PWM and encoder module is connected to the OPB bus via IPIF (Intellectual Property Interface) logic from Xilinx. The architecture of the PWM module consists of two software accessible registers; PPC405 can access reg1 and reg2 via the OPB. When the motor rotates, the encoder sensor attached to it generates two channels of quadrate encode pulses with 90 degree phase shift, which are fed into the Encoder Module via the two morphing bus I/O pins. The Encoder Module detects and analyzes these two pulses, and determines the direction of the motor (i.e., forward or backward) and then counts up or down accordingly.

The PD/ PID control algorithm, which is implemented in software called PDCTRL.C, is invoked periodically in Real time scheduler. The following major steps of the algorithm will rotate the motor and ensure that the rotation is smooth during its movement from an initial position to a final desired position. 1. The desired position is generated at each control cycle according to a velocity profile. 2. The actual motor position from register is obtained and the error between the desired and the actual positions are calculated generating a control output. 3. The PWM duty cycle value is uploaded into reg1 based on the PID control output to be used by the

PWM module to drive the motor. PWM pcore is now working at 48.8 kHz and Duty cycle is 70% on DU100 with SDK software function support.

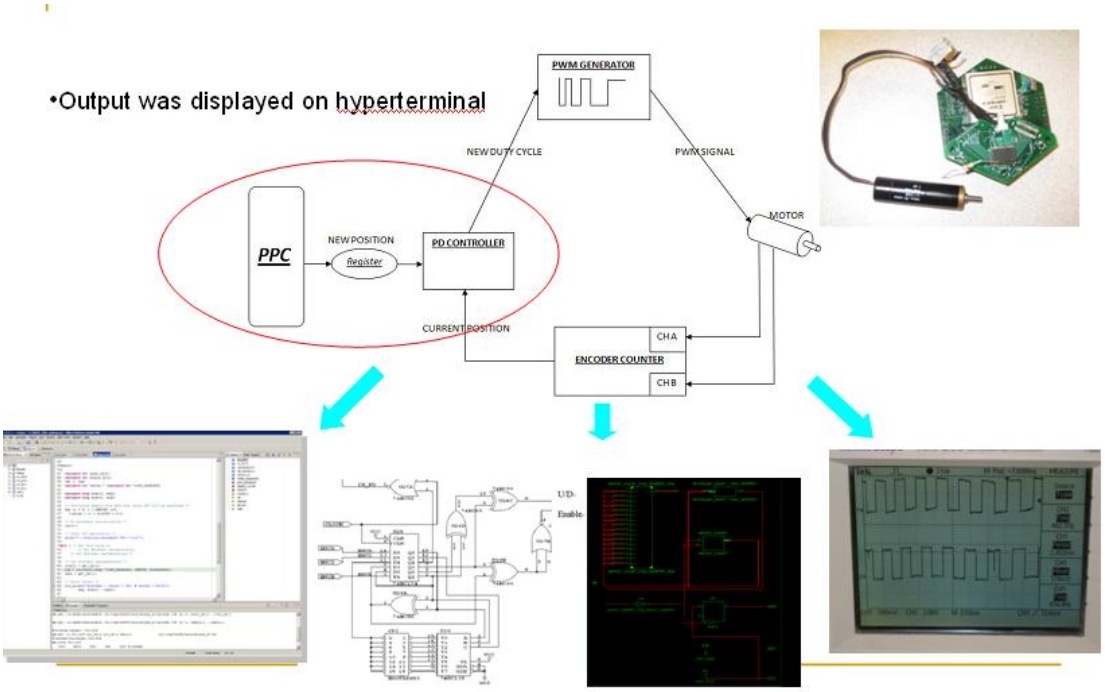


Figure35: Motion control HW/ SW co-test

Because their is an FPGA on RecoNode, the PD controller was implemented in it because it allows CPU clock cycles to be saved for other tasks, and the time granularity can be more easily adjusted on the FPGA based controller. Recently, W. Zhao et al. have implemented the closed-loop PID algorithm on a FPGA (Spartan II) and conducted comparative study of their design options (serial, parallel, and multichannel designs) relative to speed, area, and power consumption [31]. Our main goal is to implement W.Zhaos channel-level parallel (CLP) PID design and this will further reduce the motor modules over all power use. Our experiment results of step response control for all designs are shown in Figure 35. Here no load is placed on the motor for realism. The X-axis is in seconds and the Y-axis is in encoder counts

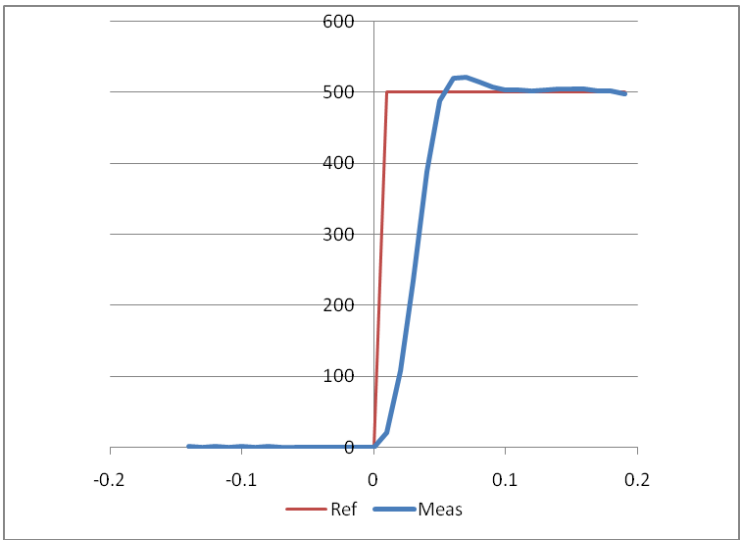


Figure36: The step response of the motor control block(X axis in Second)

Software PD design implementation was developed and tested. A performance evaluation is verified as functionally correct; we implemented and used to perform step response control tests of a DC motor. Additionally, in software, a set of preliminary PD parameters and control periods were determined by experimental method, and then we expected the parameters were tuned to an ideal step response. The parameter tuning experiment yielded the following results: proportional gain Kp = 15, integral coefficient Ki= 0, derivative coefficient Kd = 70, which were used to perform control testing. To test motor control for the PD controller design, the motor was set to toggle from an initial position of 0, and then a desired position command of 500 was issued. From Figure 36, the curve is the real response sampled from the encoder counter at 10Hz sampling period. The results show that all the designs performed correctly and similarly. Response speed is fast, overshoot is small, and static accuracy is high. The average rise time is 40.2 ms and the steady state error is 2.

Chapter 6 Conclusion and Future Work

6.1. Conclusion

Here we are building FPGA-based Real-Time reconfigurable architectures including hardware and software infrastructure for miniature mobile robots, relative concepts like self adaptive, reconfigurable computing; dynamic partial reconfiguration and morphing bus are also presented in this paper. RecoNode is supposed to a small, low power, low-cost, and highly modular platform that have ability to host large amount of sensors and actuators not at the same time to adapt complex USAR environment. VHDL IP cores (such as PWM, Encoder and UART) are developed and simulated and all of them are seamlessly incorporated to the whole embedded system by connecting IP Core with OPB/PLB bus. RecoNode is a ideal research platform for Wireless sensor and actuator/ Wireless control network. The result of RecoNode can not only been apply to comprehensive USAR robotics, but also could transfer to the Infrastructure of industrial process automations, building automation, intelligent traffic control or even future smart grid.

6.2. Future work

6.2.1 Vision module

Because it is one of the largest nodes RecoNode has the most computing power, this allows it to have a vision module. The vision module allows any RecoNode based robot to see the world around it and navigate based on what it sees. A high level

description of RecoNode's vision module architecture will consists of main control logic, I2C interface logic, image grabber control logic and memory module interface logic. The main control logic block is responsible for the high level operation of the digital camera. The image capture module is responsible for reading captured frame data from the image sensor. In order to detect when it needs to capture data, the image grabber module looks for certain embedded codes. These embedded codes are placed within the stream of digital data by the camera. Once it has found an escape sequence; it knows to start or stop capturing data. In addition to data capture, the RecoNode camera module has the ability to control the CCD reader chip to enable dynamic control of settings like contrast. Image processing is probably the most memory intensive operation robots do; because of this the RecoNode has 128MB of onboard RAM, allowing image data to be stored in the control computer or on the attached ram.

6.2.2 Wireless communications

RecoNode definitely should have a data communication wireless interface. , like all the other nodes of the WSAC. Because of its large size, reconfigurability and higher capacity batteries, RecoNode can carry multiple RF data interfaces. RecoNode's RF module contains an RF Stack and a wedge with a RF transmitter chip. The stack will be implemented in PowerPC or VHDL on the FPGA. Having multiple wireless protocols supported will allow RecoNode to communicate not only within the WSAC but also use other protocols to talk to other systems, networks and robots. For wireless sensor network applications in urban search and rescue scenarios, low

power consumption, low cost and being highly robust are the most important characteristics. In our design, we considered two protocol options; ZigBee, and Locally Switchable Protocol (LSP) created by J. Bae and R. Voyles[32]. ZigBee is a new global standard for wireless communication, which provides a short-range cost effective networking capability. ZigBee technology is a low data rate, low power consumption, low cost, wireless networking protocol targeted towards automation and remote control applications. One of the most popular ZigBee chip is CC2520 from Texas Instruments, it is a single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver, it also provides extensive hardware support for packet handling, data buffering, burst transmissions, data encryption, data authentication, clear channel assessment, link quality indication and packet timing information. We will use a ZigBee wedge to interface the RecoNode with other ZigBee enabled nodes.

Bibliography

[1] RR Murphy, Rescue robotics for homeland security - Communications of the ACM, 2004

[2] R Grabowski, LE Navarro-Serment Heterogeneous teams of modular robots for mapping and exploration, Autonomous Robots 8, 293–308, 2000

[3] D Jung, A Zelinsky “Grounded Symbolic Communication between Heterogeneous Cooperating Robots ", Autonomous Robots, 2000 – Springer

[4] Z. Wang, Z. Song, P. Chen, A. Arora, D. Stormout, and Y. Chen, MASmote - a mobility node for MAS-net (mobile actuator sensor networks)," in Proceedings of 2004 IEEE International Conference on Robotics and Biomimetics

[5] Gabriel T. Sibley, Mohammad H. Rahimi and Gaurav S. Sukhatme, "Robomote: A Tiny Mobile Robot Platform for Large-Scale Sensor Networks", Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA2002)

[6] S Bergbreiter, KSJ Piste Cotsbots: An off-the-shelf platform for distributed robotics, 2003 IEEE/RSJ

[7] R. Krohling, Y. Zhou, and A. Tyrrell, “Evolving FPGA-based robot controllers using an evolutionary algorithm,” First International Conference on Artificial Immune Systems, pp. 41-46, Sept, 2002.

[8] J. Barton, G. Hynes, B. O’Flynn, K. Aherne, A. Norman, A. Morrissey, “25mm sensor–actuator layer: A miniature, highly adaptable interface layer”, Sensors and Actuators A 132 (2006), pp.362–369, November 2006

[9] R.M. Voyles, "TerminatorBot: A Robot with Dual-Use Arms for Manipulation and Locomotion," in Proceedings of the IEEE International Conference on Robotics and Automation, 2000.

[10] R.M. Voyles and A.C. Larson, “TerminatorBot: A Novel Robot with Dual-Use Mechanism for Locomotion and Manipulation", in IEEE/ASME Transactions on Mechatronics, 2005.

[11] K Compton, S Hauck "An introduction to reconfigurable computing ",IEEE, Apr, 2000

[12] B Blodget, P James-Roxby, E Kelle" A self-reconfiguring platform Logic and Applications", 2003 - Springer

[13] Product Overview PowerPC 405 CPU Core, IBM September 2006

[14]C. D'Souza, B. H. Kim, and R. Voyles, "Morphing Bus: A rapid deployment computing architecture for high performance, resource-constrained robots," in 2007 IEEE International Conference on Robotics and Automation (ICRA 2007) Roma, Italy, April 2007, pp. 311-316.

[15] Virtex-4 FPGA User Guide UG070 (v2.6), Xilinx December 1, 2008

[16]Xilinx PowerPC Processor Reference Guide UG011 (v1.3) January 11, 2010

[17] http://www.maxonmotor.com/

[18]http://www.circuitcalculator.com/wordpress/2006/01/31/pcb-trace-width-calculator/

[19] TPS75003 datasheet Triple-Supply Power Management IC for Powering FPGAs and DSPs REVISED AUGUST 2008

[20] LTC3426 datasheet 1.2MHz Step-Up DC/DC Converter in SOT-23

[21] TPS61030/1/2 - 96% Efficient Synchronous Boost Converter With 4A Switch (Rev. D)

[22] Anand Eswaran, Anthony Rowe, and Raj Rajkumar, "Nano-RK: An Energy-Aware Resource-Centric Operating System for Sensor Networks," IEEE Real-Time Systems Symposium, December 2005.

[23] David B. Stewart, Richard A.Volpe, and Pradeep K. Khosla，"Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects". IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 23, NO. 12, DECEMBER 1997

[24] Anthony Rowe, Rahul Mangharam and Raj Rajkumar, "RT-Link: A Time-Synchronized Link Protocol for Energy Constrained Multi-hop Wireless Networks." IEEE International Conference on Sensors, Mesh and Ad Hoc Communications and Networks Reston, VA, September 2006

[25] Rahul Mangharam, Anthony Rowe and Raj Rajkumar FireFly: A Cross-Layer

Platform for Wireless Sensor Networks, Real-Time Systems Journal, Special Issue on Real-Time Wireless Sensor Networks, 2007

[26] Xilinx XAPP778 (v1.0) Using and Creating Interrupt-Based Systems January 11, 2005

[27] Hewlett Packard, Quadrature Decoder/Counter Interface ICs HCTL-2000, 2016 data sheet.

[28]Abdelkrim K. Oudjidaa, Youssef I. EI-Haffaf,A reconfigurable counter controller for digital motion control applications, Microelectronics Journal Volume 28, Issues 6-7, August-September 1997, Pages 683-690

[29] Xilinx Application Note XAPP482 [1]

[30]Xilinx Application NoteXAPP501 (v1.5) October 2, 2007

[31] W. Zhao, B. H. Kim, A. C. Larson, and R. M. Voyles, “Fpga implementation of closed-loop control system for small-scale robot,”

[32] J. Bae and R. Voyles, “Wireless video sensor networks over Bluetooth for a team of urban search and rescue robots,”

APPENDIX A: DU100 and DU120 Shematics

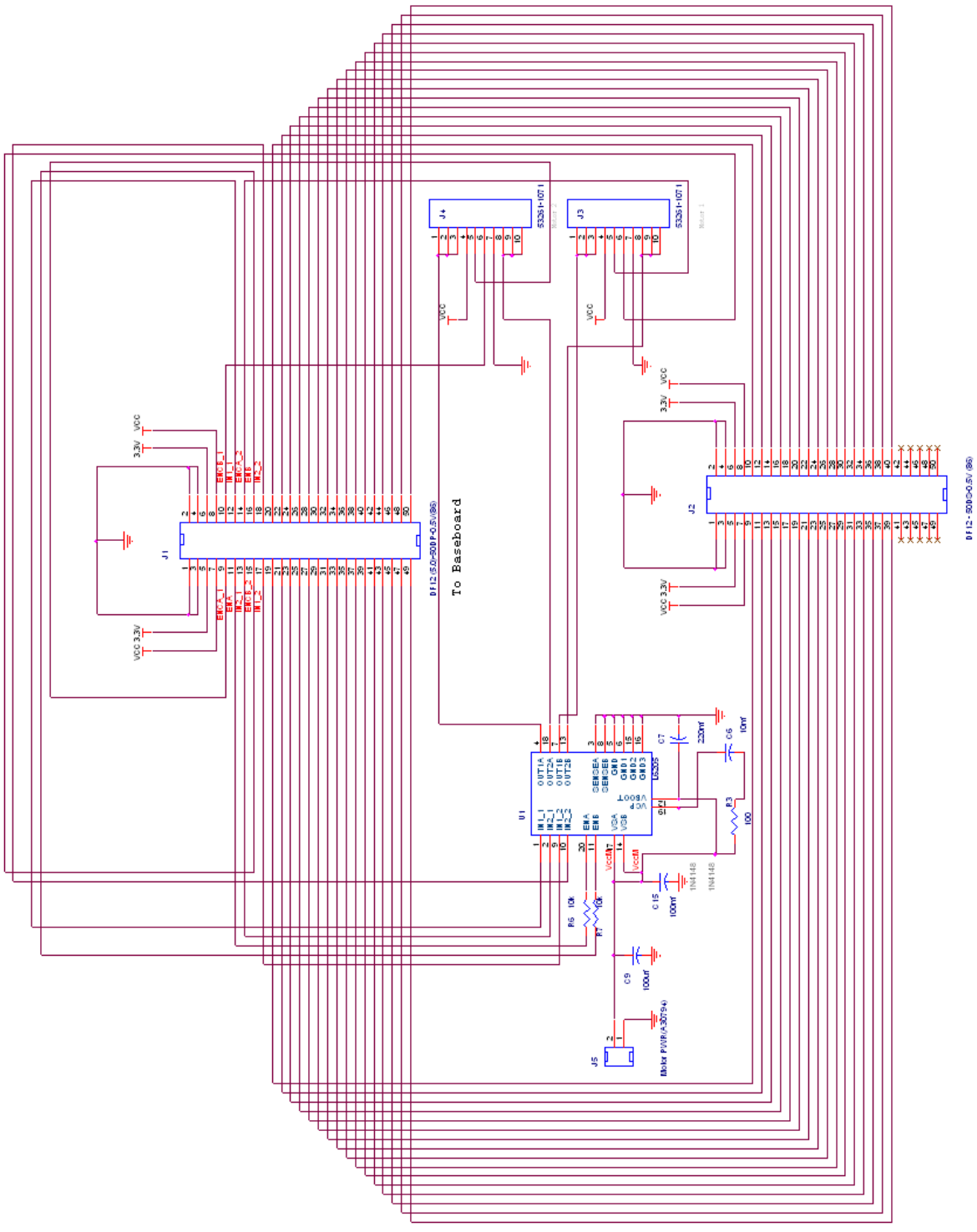


Figure A-1: Motor board DU120

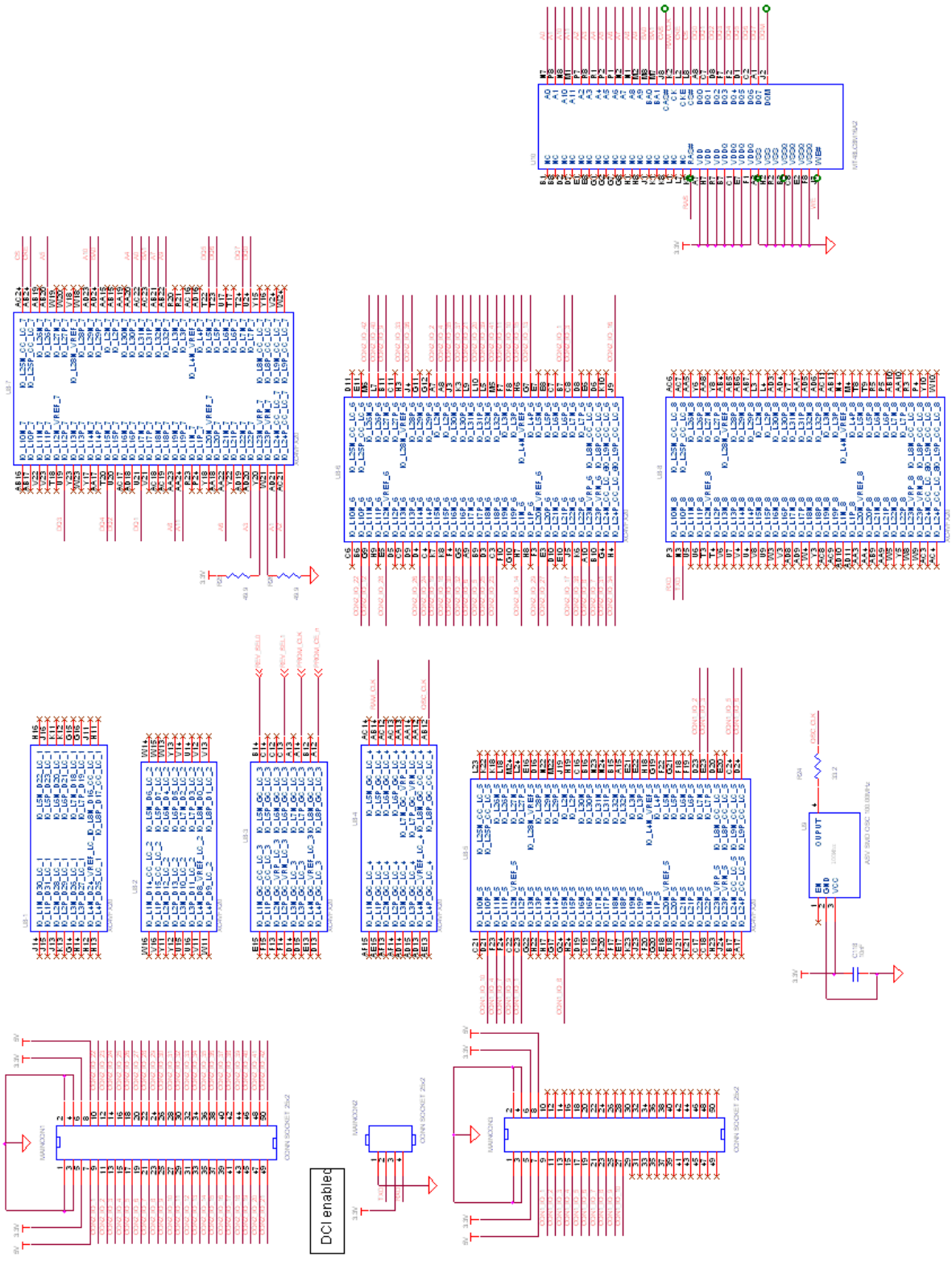


Figure A-2: Baseboard DU100

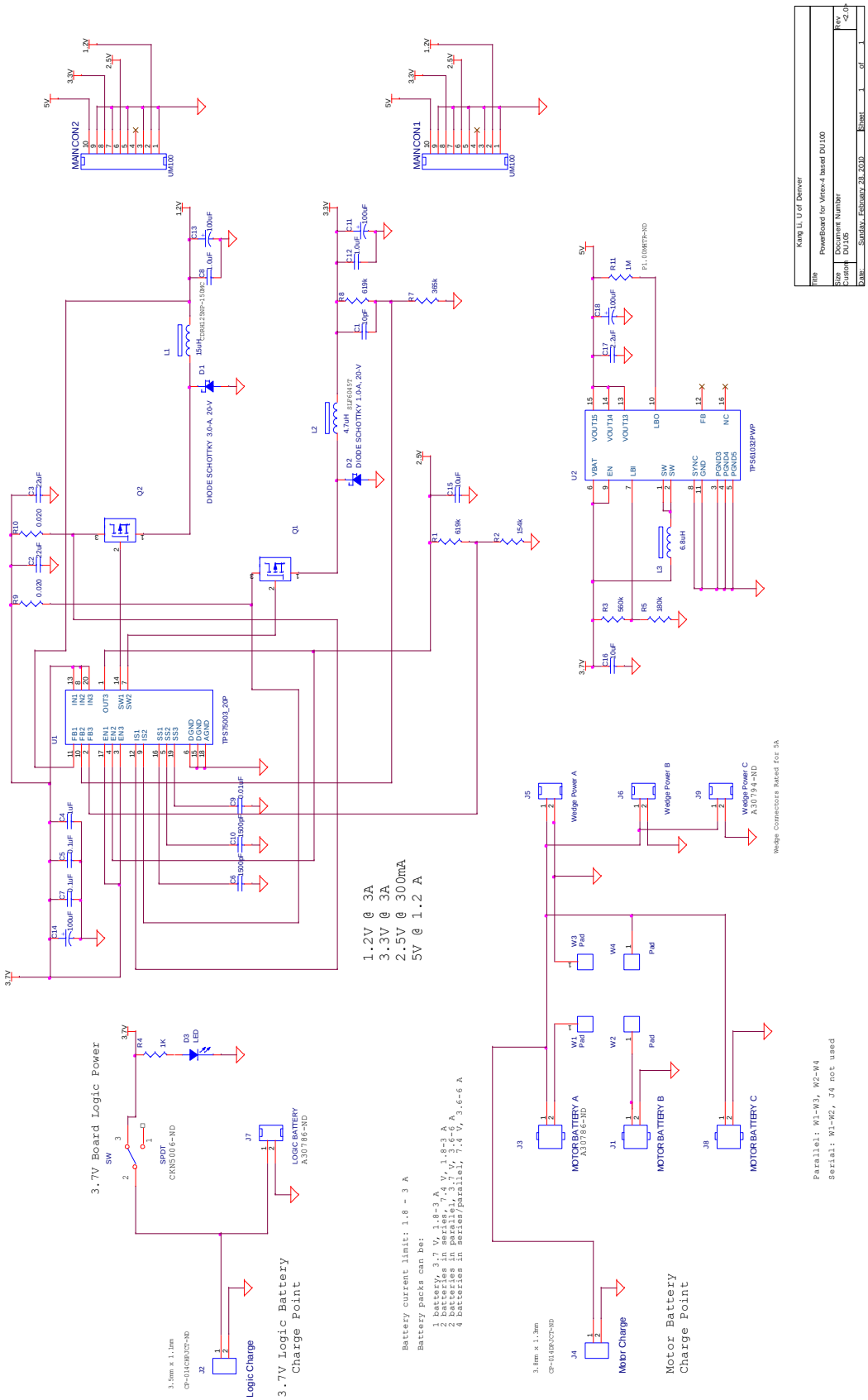
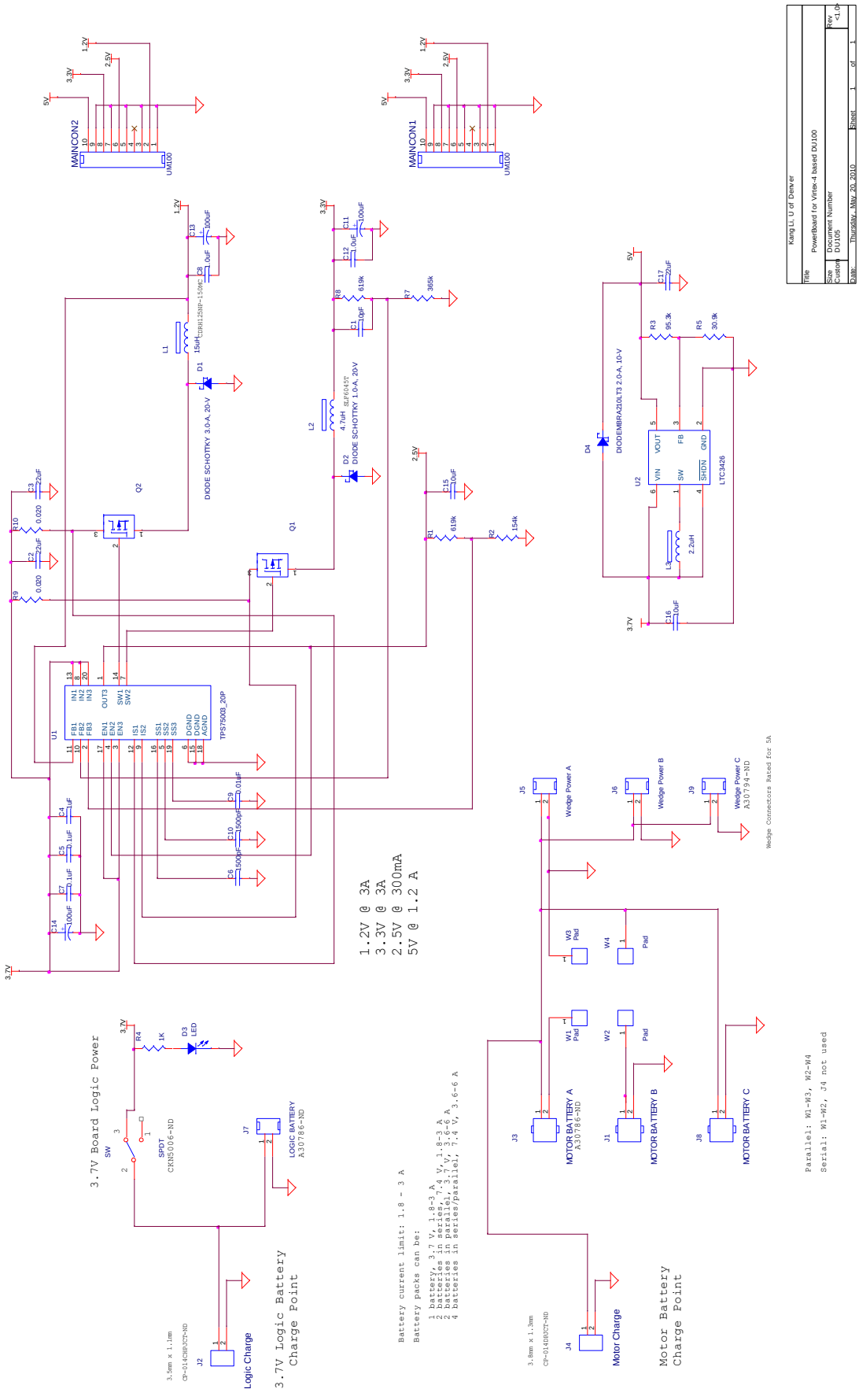


Figure B-1: Version 0.9 with TPS61032



FigureB-2: Version 1.0 with LTC3426