MULTI-MODAL RECONFIGURABLE ROBOTIC PLATFORM WITH 3D-IMMERSIVE TELEPRESENCE SYSTEM

By

Mohammad Mehrabi

Bachelor of Engineering, Ryerson University, 2011

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions of individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopy or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

MULTI-MODAL RECONFIGURABLE ROBOTIC PLATFORM WITH 3D-IMMERSIVE TELEPRESENCE SYSTEM

Master of Applied Science 2014

Mohammad Mehrabi

Electrical and Computer Engineering

Ryerson University

Abstract

The concept of reconfigurability and its applications in robotics have become prominent in the past few years as they provide versatility, adaptability and scalability to the systems. The reconfigurable robots can perform tasks in outer space, under the sea and in hazardous environments by rearranging their physical configurations to alter the system's behavior and geometry. However, the concept of reconfigurable robots is not just constrained by the mechanical reconfiguration of the components, for the system should also demonstrate a modular reconfigurable behavior to newly imposed conditions.

The objective of this work was to design and implement a multi-modal reconfigurable platform based on the concept of "form follows function" to be integrated with 3D-Immersive telepresence systems. The developed system was analyzed to verify the feasibility and functionality of the proposed architecture, and suggestions were made for future improvements.

Acknowledgement

The completion of this work would have never been possible without the consistent help and support of my family, professors, friends and colleagues.

I would like to express my sincere gratitude to my supervisor, Dr. Lev Kirischian, for the support, patient guidance, enthusiasm, and immense knowledge he has provided me throughout my time as his student.

I would also like to thank my friends and colleagues in the lab: Victor Dumitriu, David Diaz and Rares Raducu for their advice and sharing of their knowledge throughout my studies. Special thanks go to Rares Raducu and Liviu Raducu for all the good memories and years of true friendship.

I also would like to thank the Department of Electrical and Computer Engineering at Ryerson University for providing me with the great facilities and equipment needed for my research, as well as the review committee, Dr. Yuan, Dr. Geurkov and Dr. Raahemifar for their participation and helpful feedback. My personal gratitude goes out to Dr. Kamran Raahemifar, for his constructive guidance and suggestions during writing this paper.

Lastly and most importantly, I would express my deep sense of gratitude to my parents who have always believed in me, stood by me like a pillar in times of need and who gave up their own dreams to help me achieve mine.

Dedication

This work is dedicated to my parents, brother and sisters, who have always loved and supported me unconditionally.

Contents

List of Tables

List of Figures

List of Abbreviations

3D-Image Int.	3D Image Interpolation
3D-P	3D- Panoramic
ADC	Analog-to-Digital Converter
Addr	Address
ASICS	Application Specific Integrated Circuits
BitGen	Bitstream Generator
BRAM	Block RAM
C. C.	Clock Cycle
CEBOT	Cellular Robotic System
CLK	Clock
CMD	Command
COFDM	Coded Orthogonal Frequency Division Multiplexing
CPU	Central Processing Unit
CS	Chip Select
Cut_dir	Current Direction
DAC	Digital-to-Analog Converter
DCM	Digital Clock Manager
FF	Flip-Flops
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
FW	Firmware
GPIO	General Purpose Input/Output
HW	Hardware
I/O	Input/Output
I2C	Inter-Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
JPEG	Joint Photographic Experts Group
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LeftCntr	Left Counter
Lin_Act	Linear Actuator
LMB	Local Memory Bus
LUT	Look-Up Table
M1-M6	Motor 1-Motor 6
MARS	Multi-stream Adaptive Reconfigurable System
MCC	Mobile Configuration Change
MISO	Master-In-Slave-Out
ModRED	Modular Self-Reconfigurable Robot for Exploration and Discovery
MOSI	Master-Out-Slave-In
MOT CTRLR	Motor Controller
M-TRAN	Modular self-reconfigurable robots
MULT	Multiplier

MUX	Multiplexer
NavigFlag	Navigation Flag
NiCd	Nickel Cadmium
NiMH	Nickel-Metal Hybrid
PLB	Processor Local Bus
PR	Partial Reconfiguration
PROM	Programmable Read-Only Memory
PWM	Pulse Width Modulation
RF	Radio Frequency
RightCntr	Right Counter
RISC	Reduced Instruction Set Computing
RST	Reset
SCK	SPI Clock
SDI	Slave Device In
SDO	Slave Device Out
SEN_A	Sensor A
SMORES	Self-assembling Modular Robot for Extreme Shape shifting
SoC	System on Chip
SPI	Serial Peripheral Interface
SR	Self-Reconfigurable
SW	Software
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
VAPP	Video Acquisition and Pre-Processing
VHC	Virtual Hardware Component
VHDL	Very-high-Speed-integrated-circuit hardware description language
VSC	Virtual Software Component
WBL	Whole Body Locomotion
WheelRot	Wheel Rotation
XPA	XPower Analyzer
XPS	Xilinx Platform Studio
YaMoR	Yet another Modular Robot

1. Thesis Introduction

1.1 Motivation

Generally speaking, the evolution of the robotic systems can be categorized into three generations: i) robots capable of performing repetitive tasks as used in production lines of factories, ii) robots equipped with sensory devices capable of performing multiple tasks by switching from one repetitive motion to another and iii) robots with some degree of intelligence to make decision and adapt their configurations based on circumstances they encounter.

The adaptation to the environmental conditions and constraints not only requires physical reconfiguration of the robotic system but also the behavioral conformation to the confronted conditions. Therefore, the physical configuration involves changing the shape, topology and position of the physical parts of any kind of robotic system. However, the main difference between the physical reconfiguration of the traditional and reconfigurable robotic system is due to the multi-modality or multi-functionality of the system. For instance, a traditional industrial robotic system changes its topology while performing the same operation, such as assembling, manipulating, etc., but a reconfigurable robotic system changes the configuration of the components when switching from one mode of operation to another. In other words, the reconfiguration of the physical form is the means of adaptation to the new mode of operation along with the behavioral accommodation to the new functionality of the system. In this way, we employ the concept of “form follows function”.

The emergence of reconfigurable logic blocks, such as Field Programmable Gate Arrays (FPGA) has opened new horizons for the design and implementation of multi-modal reconfigurable robots that are much more efficient in contrast to the expensive and inflexible traditional robotic applications implemented by microcontrollers and ASICs. The dynamic partial reconfiguration feature of today’s FPGAs leads to more cost-effective designs, as it allows effective adaptation of the multi-modal robots to have their functions modified during run-time without entirely interrupting their normal operations.

However, the most effective utilization of intelligent robotic systems still assumes the involvement of an operator for strategic level control, while the robotic system provides the tactical level control of current operation due to the existing intelligence gap between humans and machines. This allows augmenting the intelligence of both humans and machines for the most effective performance in areas where humans are physically incapable and robots are intellectually incapable of performing the assigned tasks.

One of the important applications of this utilized intelligence augmentation is in telepresence systems. In such systems, autonomous and semiautonomous robots perform tasks where the physical presence and/or operation are too dangerous or impossible for human beings [1]. The visual and control aspects of the contemporary telepresence systems that benefit from multi-modality and multi-video-stream processing systems, such as [1, 2], has made it possible to remotely monitor and control mobile systems working in harsh environments, such as space, nuclear power plants, mine fields, the bottom of the sea, underground pipelines and many other educational and health care related areas without endangering people's lives.

Therefore, the main goal of this work is to design, implement and verify a mobile multi-functional reconfigurable robotic platform based on the concept of "form follows function" that unlike the traditional application-specific robotic systems, could be used as an instrument to be integrated with existing telepresence systems.

1.2 Objectives

The main objective of this work is to develop a universal multi-functional mobile and run-time mode adaptive robotic platform to integrate with systems, in which a mobile robotic subsystem is required to eliminate the physical presence of humans in hazardous or inaccessible places. The following stages were considered to achieve the above goal:

1. Research and analyze the existing available approaches in implementation of reconfigurable robotic systems;

2. Propose an alternative approach based on the concept of hybrid reconfigurability of mobile applications;

3. Develop the architecture for a reconfigurable mobile robotic platform to integrate with the existing telepresence system;

4. Design and implement the reconfigurable mobile robotic platform to integrate with the existing telepresence system;

5. Analyze the experimental results associated with the performance parameters;

6. Verify the proposed approach and identify potential limitations during the test and verification process.

1.3 Contributions

The following contributions were made during the course of the work to meet the previously stated objectives:

1. Extended literature research in the area of telepresence, telerobotics and reconfigurable robotic system.

2. Developed the framework of the reconfigurable multi-modal system based on the required specifications of the mobile application associated with the implemented telepresence system.

3. Designed the system's hardware components and implemented the architecture of the system for the first prototype based on the concepts of reconfigurability and multi-modality. The mobile robotic system was presented at the annual conference SVAR-2013: Space Vision and Advanced Robotics held at MDA Space Missions, Brampton, Ontario and won the first place for the best presentation and demonstration.

4. Performed in-depth analysis on the experimental results and observation and alternative approaches were suggested and discussed for future expansion and implementation of the system.

1.4 Thesis Organization

The remaining organization of this thesis is as follows:

- Chapter 2 is associated with the analysis of the current approaches and tools in the field to address the necessity of redefining the concept of reconfigurability in robotics systems.

- Chapter 3 presents the proposed architecture of the reconfigurable multi-modal robotic platform by providing an in-depth analysis of the system's modes of operation according to the anticipated operational tasks for an existing telepresence system. Based on the assigned modes of operation, the required resources involved in each mode and their operational functions are discussed. Furthermore, the architecture organization of the system is realized after addressing the shared and common resources in each operational mode and determining the type of implementation for the components associated with each mode of operation.

- Chapter 4 presents the process undertaken to select components of the system according to the specifications of each mode of operation discussed in Chapter 3 and covers the detailed implementation of the system, including the hardware, firmware and software development.

- Chapter 5 presents the observations and experimental results based on the described experimental setup to test and verify the proposed system. Additionally, this chapter discusses solutions for further improvement and expansion of the system.

2. Related Works

2.1 Introduction

Over the past few years, the concept of reconfigurability and its applications in computer architecture and robotics have become prominent as they provide flexibility, adaptability and scalability to systems that require rapid changes and adaptation due to the environmental, social, economic and technological conditions.

Reconfigurable robots have been used in demining, undersea experiments and planetary exploration missions, such as NASA's Mars rover Curiosity [3], to reduce the risk and eliminate the physical presence of humans from hazardous environments. However, absolute replacement of humans with robots is not yet attainable due to the limited level of intelligence of existing robotic systems. Hence, in most cases human supervision and decision making skills are required to strategically monitor and control the robots.

The main barriers to designing and developing such systems to monitor and control robot operations are latency, data volume and bandwidth. Therefore, it is necessary to create a universal reconfigurable semi-autonomous robotic platform that can be integrated with existing telepresence systems to perform different tactical tasks without human assistance.

In the following sections, we first discuss some of the existing telepresence systems to address the necessity of employing automated tactical adaptation by mobile reconfigurable systems. Then, we examine the concept of reconfigurability in robotic systems to differentiate the two aspects of reconfigurations, namely, physical and behavioral adaptations by studying some of the related works in this area.

2.2 Telepresence System

The concept of telepresence refers to a set of technologies that provide a person with a sense of physical presence remote from his/her current location. The rapidly growing applications of telepresence systems can be found in business, government, education and health care sectors, where the need for transportation seeks to be eliminated due to economic and environmental reasons [1].

The applications of the system are not only limited to teleconferencing; they can be extended by introducing the concept of telerobotics to the system. In such a system human sensory elements of vision, sound and manipulation are triggered by incorporating much more advanced video conferencing equipment onto mobile robot devices that can be steered from remote locations [4]. These telepresence systems can be divided into the three categories below:

i) Systems controlled directly by the remote operator, such as telerobotics surgery, that enable surgeons to operate on patients from remote locations using robotic instruments that mimic the movements of a surgeon‘s hands and provide him with 3D imaging system to perform complex operations [5].

In this type of system, the remote operator becomes the key factor in making decisions that are context dependent in real time. Thus a reliable real-time communication between the remote operator and the robot is required that supports both visual and audio streaming to provide the operator with a realistic experience via haptic feedback and to assist him in making meaningful and crucial decisions [6].

ii) Surveillance systems for monitoring and control, such as industrial and commercial security systems, where multiple vision units may be integrated in order to expand the surveyed area equipped with alarm systems. Upon occurrence of an alarm condition, the remote operator can have visual access to the remote environment in which the alarm was activated to take further necessary actions [7]. However, these systems require heavy integration with sensing elements to provide best results since constant human supervision is not possible. The mobile laptop robot by [8] is a very simple commercial example of these systems, where a mobile adjustable frame can hold a variety of small laptops to act as a surveillance system which uses the laptop’s webcam as the vision system and allows users to control the carrier over the internet.

iii) Immersive telepresence system that creates an equivalent experience to a human presence in an actual remote environment. For instance, [9] has developed a 4D real-time augmented virtual reality

(AVR) browser to provide an interactive, integrated, mixed virtual reality remote view of the area-of-interest (AOI) to the operator. Therefore the operator has a remote presence in the real world and can interact with personnel and sensor assets in the AOI in order to analyze and share the captured information from multiple sources in real-time.

However, this immersive technology vastly relies on the data volume and bandwidth. For this reason, [10] has categorized the requirements of the immersive telepresence system based on the applications and design goals, as shown in Table 2.1.

Table 2. 1: Immersive Telepresence Requirements Comparison for Design Goals courtesy of [10]

Requirement	Science	Operational	Public
Data samples	High values	Medium	Low
Completeness	Low priority	Medium	High
Fidelity of data	High	Varies	Low
Non-verified data	Low priority	Varies	High
Latency demands	Required loop	Virtually Ok	Virtual
Interactivity	Real	Simulated	Virtual
Real time bandwidth for autonomy	High priority for complex interactions	Varies	Low

As can be seen in the table above, the level of system complexity depends on its application, as the scientific applications associated with the science exploration, discovery and surface interaction such as planetary rovers generally do not require complete and high-precision data sets. In contrast, public engagement concerns do receive a large value benefit upon completion due to the existence of an incentive to obtain a threshold of observations that will allow the derivation of an acceptable product.

Furthermore, [11] has also addressed the traditional issues in the application of telepresence in mining operations, such as latency, poor vision subsystems, compression schemes, communication protocols, etc., and proposed the design of a mining robot tele-operation system that uses a combination of several techniques, such as intelligent data-rate adaptive video compression, computer vision, obstacle avoidance using range sensing, physics-based motion prediction, way-point guidance and augmented virtuality-based rendering in order to permit realistic immersive telepresence for the operator of the mining robot.

Similar to mining tele-operation, applications for planetary exploration missions have also been developed using semi-autonomous robots, such as Robonaut, the first dexterous humanoid robot to enhance and expand the ability of astronauts to safely and accurately construct and discover [12] without suffering from the speed of light latency by operating from Earth.

However, the main downfalls with these existing immersive telepresence systems are that they do not make use of electromechanical (moving) components, have low video quality performance, including frame resolution and/or frame rate, and consume a relatively high amount of power in case of CPU-based implementation [1]. The designed adaptive 3D-P telepresence system for mobile applications by [2] as shown in Figure 2.1, resolved the issues associated with the quality and transmission of the telepresence system.

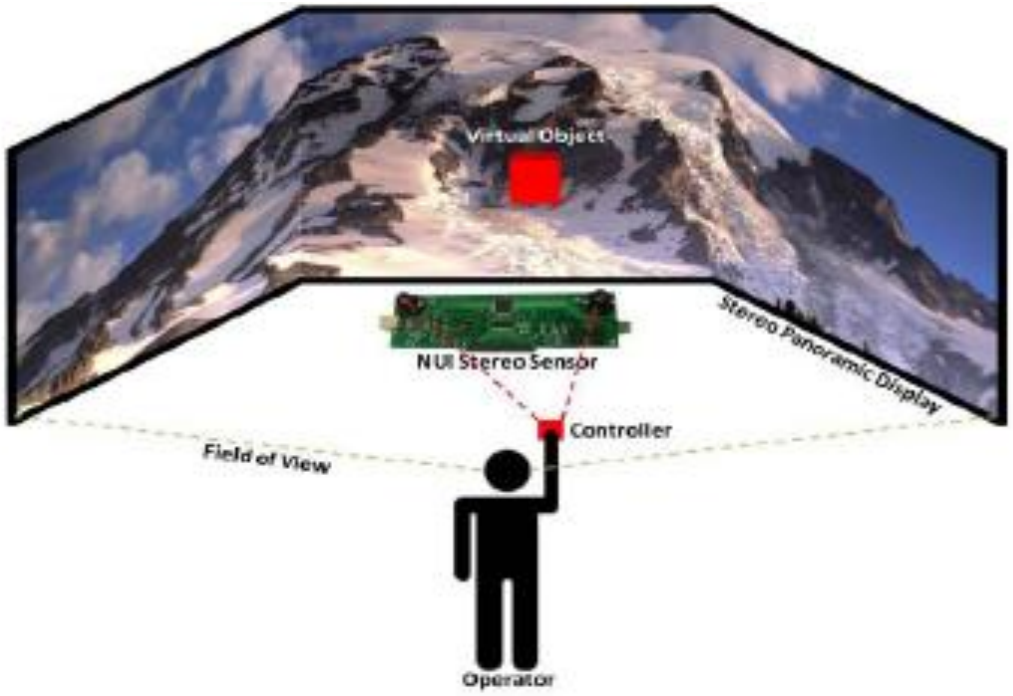

Figure 2. 1: Operation Using the Proposed Telepresence System Courtesy of [2]

The focus of this work will be on the design and development of a multi-functional reconfigurable robotic platform that can be integrated with existing telepresence systems to utilize the electromechanical components of the system and enhance the tactical reliability and adaptability of the system. In the following sections, the existing approaches in designing reconfigurable robotic systems will be discussed.

2.3 Robotics and Reconfigurability

A simple robotic platform consists of mechanical parts (chassis, housing, wheels, etc.), electromechanical parts (motors, buttons, switches, LEDs, etc.), and sensors (odometer, infra-red, accelerometer, etc.). Many of the parts related to the physical configuration of the robots are solely monolithic such that introducing a minor upgrade to the system requires scrapping the old design and reconstructing the system based on the new requirements.

In contrast with the traditional fixed-structure robots, modular robotic systems capable of reconfiguring their morphology by rearranging the connectivity of their parts have revolutionized the design architecture of robotic systems [13]. In other words, the initial configuration of the robot is designed for a specific task and, upon completion of the task, the modules in the structure can be disconnected and reassembled to create a new configuration to support multiple modalities of locomotion, configuration and perception [14]. This feature of modular robots is considered to be the foundation of the reconfigurable robotic systems, including self-assembling, self-reconfiguring and self-organizing systems that can perform tasks in space, under the sea and human-inaccessible environments due to their adaptability and flexibility to serve as different instruments in various environments.

It is worth mentioning that the concept of reconfigurability is not just bounded by the physical reconfiguration of the systems; rather, behavioral aspects as well as utilization of hardware components by reusing the components for various purposes need to be considered as well. Therefore, the reconfigurable system for the physical world can be divided into the following:

i) Physical adaptation associated with the reconfiguration of the physical and mechanical components of the system to ensure the tactical functionality of the system upon confronting environmental interferences.

ii) Behavioral adaptation to utilize the physical components used in the design for a function or application that can be further subdivided into :
   a. Reconfiguration by changing only the procedure of the architecture.
   b. Reconfiguration by changing the system structure, including components, links between components and procedures.

2.3.1Physical Adaptation

The physical adaptation of reconfigurable robotic systems is associated with the physical and mechanical reconfiguration of the system by changing the orientation and assembly of the physical components to expand robotic system capabilities beyond the traditional single-task robotic designs. One of the first robotic infrastructures to employ this concept was proposed by [15] to be used for planetary surface operations, as shown in Figure 2.2.

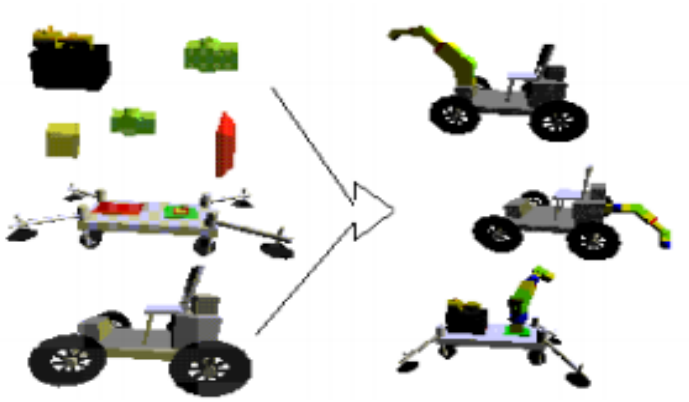

Figure 2. 2: Modular Robot Concept Proposed by [15]

The proposed design consisted of the following modules:

- Based module: the core of the robot that comes in different shapes based on the application and that interconnects with various modules.
- Power modules: the power supply of the robotic platform.
- Actuation modules: the modules that produce rotational motion that come in various sizes based on the required tasks to be done.
- Kinematic Modules: also known as links, which are used to alter the dimension of the robot by changing the distance between its joints.
- End-Effecter Modules: they include manipulators to perform assembly and sample collection.

However, the proposed design fails to establish an actual prototype design, and most results presented occur only in simulation. Moreover, even though the reconfiguration of the infrastructure enhances the functionality of the robot, the proposed approach seems to be simply introducing multiple robotic designs for different operations, as the base module is the only one that remains unchanged.

In contrast with the above approach, where human interaction was required to set up the physical configuration of the robot based on the required tasks, a more advanced type of modular reconfigurable robotic systems, also known as Self-Reconfigurable (SR) robots, can be employed that are capable of adapting their physical configuration for environmental variations without any human assistance.

A Self-Reconfigurable (SR) robot is a robust, multi-functional, scalable and self-repairable modular based robotic system, capable of metamorphosing its shape and changing physical connections and functions based on the conditions of the surroundings without any assistance from the outside world [16]. Configurations with arbitrary shapes and forms can be achieved to perform useful functions, such as navigation or handling objects, due to the existence of identical modules, which have no specific functionality on their own.

The roots of creating SR robots were born back in the 1970s when robots with several modules were invented. They were able to automatically switch between modules due to the required functions at a specific time. This approach serves the concept of SR robot design as each module can be replaced by another similar module, while the replaced module can also function as the initial module to help the system reach its objectives. The first SR robot to employ the concept of modular reconfigurability is the CEBOT (1988), which was composed of several modules, such as transportation and rotational joints and telescopic arms that enabled CEBOT to perform multiple tasks [17].

Furthermore, the modular design of SR robots offers multi-functionality, robustness, flexibility and self-restoration. The self-restoration ability of the SR robots refers to both internal and external environmental changes imposed on the system, such as mitigation of hardware faults associated with manufacturing defects, module failures, radiation and electromagnetic interferences, etc., and efficient and quick adaptation to unexpected changed circumstances.

Over the past two decades, many advanced and sophisticated SR robot designs in terms of hardware architecture, planning and control algorithms, efficient simulation and system integration have been developed. However, the general architecture of the SR robots can be classified into two major groups [13]: Mobile Configuration Change (MCC) and Whole Body Locomotion (WBL) according to the nature of mobility patterns and the reconfigurable properties of the robot, as shown in Figure 2.4. Each of these two classes is also branched out to sub-classes based on the geometric arrangements of the hardware components of the robots.

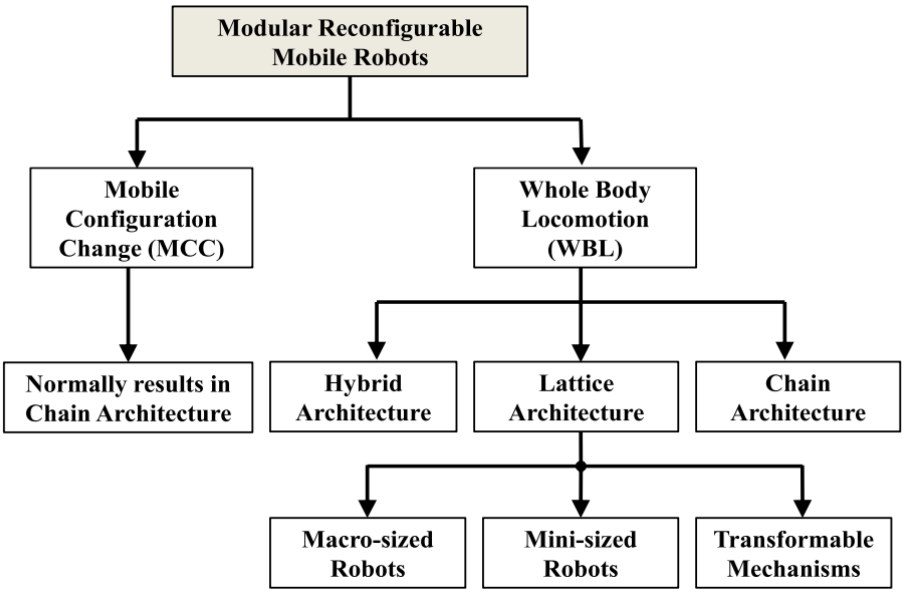

Figure 2. 3: Classification of the Reconfigurable Robots Courtesy of [13]

The MCC configuration refers to modular robots, where the modules are self-contained and interact independently with the environment. However, the WBL refers to the modular robots whose morphology provides different types of locomotion, such as walking, crawling and rolling [18]. The sub-categories associated with these two classes of modular robots are as follows:

- Lattice architecture, which consists of units that are arranged and connected in some regular 3D pattern, such as a cubic or hexagonal grid. The kinematic features of lattice robots can be categorized by their crystallographic displacement patterns [19], which allow specific motion patterns that result in simpler physical reconfiguration. However, many actuators and connectors are required to transfer the motions among the modules.

- Chain/Tree architecture, which consists of units that are connected together in a string or tree topology in such a way that they are able to reach any point in the space. However, many units are required to construct a specific configuration that increases the complexity and overhead of the systems.

- Hybrid architecture, which takes advantages of both chain/tree and lattice architectures by maintaining the control and mechanism of lattice type and exploiting the versatility of chain/tree architecture.

The advantages and disadvantages of the above architectures with respect to their level of complexity, scalability and feasibility are listed in Table 2.2, according to [17, 20, 21].

Table 2. 2: SR Hardware Architecture Comparison

Architecture	Advantages	Disadvantage	Examples
Chain/Tree	versatile	Less symmetrical compared to Lattice and requires a chain of many units to perform a task	ModRED, Polybot
Lattice	easy to self-reconfigure and suitable for forming various static configuration	Difficulty in generating motion. Complex mechanical design due to many connectors and actuators	ATRON, M-Cubes
Hybrid	The cross-over between the chain and lattice types	Anisotropic symmetry that makes it hard to self-reconfigure	M-TRAN, SMORES

However, there are currently no standard benchmarks for the performance parameter analysis of the modular robotic systems since it is difficult to directly compare the mechanical hardware architectures with one another. Therefore, most performance analyses such as [22], [23] and [24] are subjective and do not clarify if the reported values are based on actual measurements or engineering estimations.

Even though advanced mechanical design of today's self-reconfigurable robots, such as SMORES [25] and M-TRAN [26], which are capable of rearranging their modules in all three reconfiguration architectures (as shown in the table above) demonstrate a giant leap for the field of reconfigurable robotic systems, the majority of related works in this field rely mostly on the physical configuration of the robot and use of redundant and complex hardware to implement a modular reconfigurable design [27]. Therefore, in the following section, the behavioral aspect of reconfigurable robotic system will be discussed.

2.3.2 Behavioral Adaptation

Behavioral adaptation is associated with the system's accommodation to new functionality imposed by environmental or application variations. As mentioned earlier, the behavioral adaptation of the reconfigurable system can be achieved by either changing the procedure element or the entire system structure, including the components, the links between the components and the procedure of the system architecture.

The main difference between these two forms of adaptation is the level of involvement of resources and the ability to recover in case of hard faults, such as radiation, manufacturing defects and EMI, etc. For instance, a mobile robot navigating on an angular terrain uses its sensing elements to avoid rolling-over; however, if the components associated with these sensing elements were defective, then the system based on the first method could not pursue its objectives. On the other hand, the second method would offer other forms of behavioral adaptation such as decreasing the speed or lowering the height of the platform for stabilization.

Moreover, the first method of adaptation requires switching procedures to alter the behavior of the system; this switch can be implemented using traditional high-performance sequential processors. This procedure could be as simple as multiplexing among different functions or as complex as loading configuration data via custom boot-loaders capable of loading the appropriate firmware upon start up [28]. However, consider the system in [26], which uses four microcontrollers for each module. The reliability, high power consumption and cost of some of these elaborate systems may not be able to keep up be able to keep up with the performance parameter requirements of most embedded systems including reconfigurable robotic systems.

On the other hand, the second method offers a completely versatile robotic system, in which, not only the system can reconfigure its behaviors based on the required task, but it can also recover from hard faults by partially functioning. For instance, the Spirit Mars rover became stuck in soft oil on Mars, but it continued to work as a stationary science platform for a few more months [29]. This method requires a high degree of complexity as not only the procedure part of the architecture is required to be modified but also the components and their links need to be changed to realize a versatile system capable of mitigating to hardware disturbances and adapting to unexpected environmental changes. Therefore, these systems can be best implemented using reconfigurable logic blocks.

The reconfigurable logics have been used in industry mainly for development and rapid prototyping purposes. However, in robotics application wherein hardware and software could constantly change

due to different circumstances, designing a flexible, reconfigurable and adaptable system has many advantages. The life cycle and performance of the robot would be enhanced, while power consumption, total cost and risks would be reduced [30].

There have been some works in design and development of reconfigurable robotic systems using reconfigurable logic blocks of FPGAs such as [31], [32], [33] and [34], which will each be addressed in the following paragraphs.

The robot designed by [31] used custom cores described in HDL along with an embedded soft-core processor with a traditional architecture to increase the versatility of a modular worm-like robot by dynamic hardware modification and hardware/software co-design and remote hardware reconfiguration. However, the focus of this work was to extend the previous work done by [35] by reducing the number hardware components. It does not examine the potentials of using dynamic hardware modification to define a set of modes of operation to reconfigure the system's behavior as well as its physical arrangement.

The dependability of the existing reconfigurable robots to their physical forms was addressed by [32] and [33]. They offered a flexible framework for adaptive locomotion control by taking advantage of dynamic partial reconfigurability of the FPGAs (see figure 2.4). However, the proposed system does not support the physical aspect of reconfiguration since human assistance is required to connect the modules.

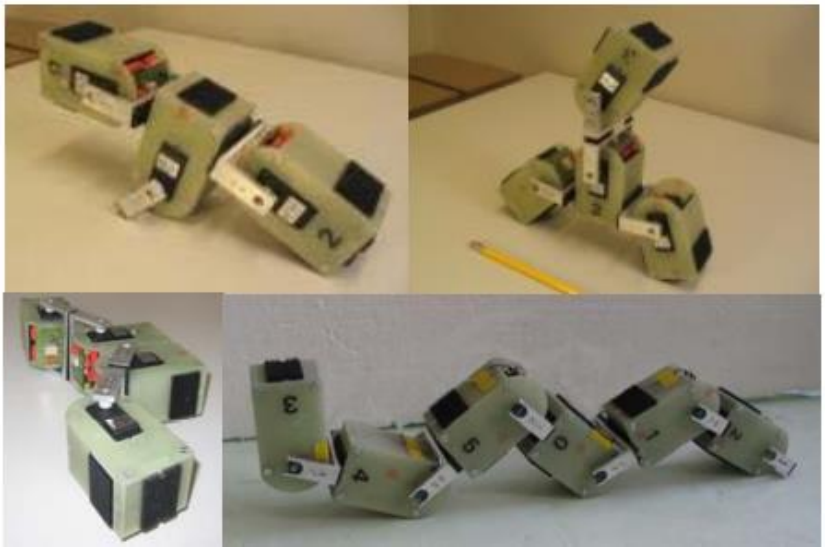

Figure 2. 4: Different Configuration of YaMoR Courtesy of [33]

The dynamic reconfigurable robotic system developed by [34] employed the dynamic partial reconfiguration feature of Xilinx Virtex-2FPGA to implement a dynamic run-time behavior reconfiguration in their system which includes two mobile robots. The system performs a certain task that can involve following a wall, avoiding obstacles, normal driving and leading/following during the run-time by downloading the partial configuration bit stream of required behavior to implement the corresponding interface circuit. Even though, the proposed system properly aimed for the modularity concept [36], where the system's behavior is constructed from multiple control and communication modules to create different modes of operation, the main issue with [34] is the incapability of the system to utilize its resources, as many components are idly present during different modes of operation.

The proposed system by [37] took a similar approach and incorporated the hardware task scheduling into the traditional scheduling process using conventional processors. The system partitioned a complex software task into five independent subtasks, which are stored in a memory device. A hardware task scheduler controlled the execution order of the subtasks by downloading different partial configurations bitstream on to the FPGA. However, the proposed approach was examined only through simulation due to the nonexistence of any benchmark packages for a reconfigurable system.

Based on what has been discussed so far, many works have been done in the area of reconfigurable robotic systems. However, the main focus of these works is on the physical configuration of the system, with a disregard for the behavioral aspects in most cases. On the other hand, the recent works focusing on the behavioral aspects are often application-specific and fail to exploit the full potentials of the behavioral adaptation aspect of a reconfigurable robotic system. Therefore, it is essential that we redefine the concept of reconfigurable robotic system to design a multi-functional universal robotic platform capable of functioning in various environments.

2.4 Summary

In summary, we addressed the necessity of using mobile reconfigurable robotic systems in a telepresence system, whereby human beings can realistically and remotely operate mobile robots in hazardous and inaccessible environments, from safe locations. We discussed the recent and existing approaches in design and development of reconfigurable robotic systems and clarified the conceptual difference between the physical and behavioral aspects of reconfigurable robotic systems. We outlined the limitations of each approach to come up with a universal reconfigurable robotic platform that can be integrated with existing systems, including telepresence systems.

3. Architecture Development of Multi-Modal Reconfigurable Robotic Platform

3.1 Introduction

The main idea behind implementing reconfigurable robotic systems is the ability to adapt to environmental conditions and constraints by altering the physical reconfiguration of the robotic system and still maintain the functionality and behavioral conformation to the applied physical changes. In other words, the configuration of the robotic system including its physical, computing architecture control and communication configuration, etc. should correspond to the workload and environmental conditions imposed upon the system.

Many works have explored the area of reconfigurable robotic systems in terms of changing the physical configuration, as any type of robotic systems that can change its shape, topology and position of its physical parts can be considered reconfigurable. However, the main difference between the conventional robotic systems and an actual reconfigurable robotic system is the fact that the functionality of the first system does not change as the topology of the system alters, such as assembling equipment, moving or manipulating other objects, etc. On the other hand, a true reconfigurable robotic system is a multi-modal system, in which the configurations (topological and behavioral) are optimized for the modes of operation. In this case the physical reconfiguration becomes a process for adaptation to the new mode of operation even as the behavioral accommodation leads to the new functionality of the system.

Moreover, as it was discussed in Chapter 2, most of the works done in the area of reconfigurable robotic systems targeted the physical aspect of reconfigurability, while the behavioral aspect and utilization of resources are often disregarded. This chapter contains a detailed description of our proposed system by analyzing the various modes of operation to determine the architecture of a multi-modal autonomous mobile reconfigurable robotic system.

3.2 Concept and Theory Analysis

The concept of reconfigurability has become a popular subject in the realm of computer architecture. Reconfigurability is referred to as the ability to change the hardware or parts of the hardware either on a problem-by-problem basis or during the lifetime of an algorithm solving one problem instance [38].

Reconfigurability has a tight relationship with the concept of survivability, which can be simply defined as the ability to continue to exist by avoiding regression and external disturbances and thus maintain overall stability. In engineering, survivability is the overall ability of a system and subsystem's process to continue functioning despite the occurrence of natural or man-made disturbances, such as electromagnetic wave interferences, hardware malfunctions, etc.

A deeper look into nature and wildlife verifies the existence of reconfigurability, wherein the form or the physical descriptions of animals determine their characteristics, habitats and activities of species. Depending on the type of the species habitat, animals need to adjust themselves to change along with the environment. The polar bear is a good example of adaptation of mammals in extremely cold Arctic weather. The thick fur and layer of stored fat under the skin help the bears to trap air and insulate their body in order to survive [39].

Furthermore, survivability dictates the configuration and lifestyle of the species as it tends to achieve sustainable growth and avoid regression and degradation. The need to survive has been the foundation of species' evolution to guarantee their existence in a particular ecosystem. For example, the difficulty in finding food resources during the winter forces certain mammals to conserve energy by reducing their normal body processes to almost a stand-still (hibernation) in order to survive in harsh weather conditions.

The same analysis can be applied when designing reconfigurable systems, where a reconfigurable system can be redefined as a system that requires multi-functionality and the ability to change its configuration including physical and behavioral aspects to mitigate the environmental harshness and adapt to the surrounding variations. Having said this, the general organization of our proposed reconfigurable robotic platform can be discussed.

The general organization of our proposed multi-modal robotic system is based on the concept of "form follows function" and is expected to be used as a universal multi-functional robotic platform for integration with existing systems, such as telepresence systems. The proposed concept simply implies that the physical form and configuration of the system should comply with the task in

process. For instance, a mobile robotic subsystem of a telepresence system should be able to perform multiple tasks such as navigation, recording and transmission of image data to the control center. However, the number of physical components of the system can be reduced by utilizing the hardware resources during the performance of each task. For instance, the tower holding the camera modules of the robotic system can be placed down during navigation to enhance stability and avoid roll-overs and therefore eliminate the need to have sensing elements, such as accelerometers to constantly monitor and adjust the tower's level. Additionally, the number of camera modules can be reduced based on the application's need to see far objects; the two front-view cameras can be adequate to provide a telescopic view rather than using four cameras to provide a panoramic view to a remote operator.

Based on what has been discussed so far, it is necessary to define the reconfigurable system by its operations and to categorize its functions into a set of modes of operation that also incorporates resources arrangement and optimization. In this case the physical adaptation aspect of the reconfiguration becomes a process for adaptation to the new mode of operation with the behavioral accommodation to the new functionality of the system. Therefore, the multi-functional characteristics of our reconfigurable moving platform imply that unlike the uni-functional systems, traditional ASICs, there should be a set of operation modes, which may vary the physical configuration of the system and switch during various stages of task executions on the basis of request or environmental conditions.

To put our proposed robotic platform to the test, we took advantage of the existing 3D-P telepresence system, designed and developed earlier by this laboratory, and defined a set of operational modes for the platform to determine the general system architectural organization.

3.3 Modes of Operation Analysis

In order to distinguish the primary modes of operation for the moving platform, the required tasks of the system need to be defined. A typical scenario will be discussed to shed some light on the behavior of the system and the actions of which it is capable:

- The system is out on the field and initially at rest. The system conserves energy by eliminating and reducing unnecessary activities. However, the system is able to communicate with the teleoperator, who is located at the master side.

- The system enters an observation stage upon receiving a request from the master side, in which the vision system performs stereo-panoramic acquisition of the scene and then pre-processes and transmits the captured data back to the master site for further actions. The hardware organization of the system needs to be formed in such a way to enhance and maintain the quality of the captured images and perform further adaptation mechanisms, such as changing video frame resolution, frame rate, compression levels, disparity, noise reduction, etc.

- The transmitted data is received by the teleoperator, who now has a clear vision and perception of what is happening on the slave side and can act accordingly by remotely controlling the moving platform via the provided manipulators.

One of the most important and interesting features of the system comes into effect when the system is required to relocate itself to another position due to a request by the master side. In this case the system will not capture any more images due to noise, vibrations and other factors that can affect the data and the operator's vision and perceptual capabilities during the movement. However, the system autonomously drives itself to the requested position by exploiting its navigation sub-system that controls the speed, distance, balance and obstacle avoidance features.

It should be noted that, even though no image is captured and transmitted to the master side, the teleoperator will not notice the interruption and continues to observe stereo-panoramic video, which is modified and updated by the control unit at the master side, based on the video generated earlier by the moving platform.

Upon arriving at the destination, the moving platform will notify the control center and then start to transmit new stereo-panoramic video, which will be picked up and demonstrated to the teleoperator for further decision making.

By knowing the possible tasks of the moving platform, the behavior of the FPGA- based control and communication system for the mobile autonomous re-configurable robotic platform can be grouped and implemented as shown in Figure 3.1, to accommodate various tasks requirements.

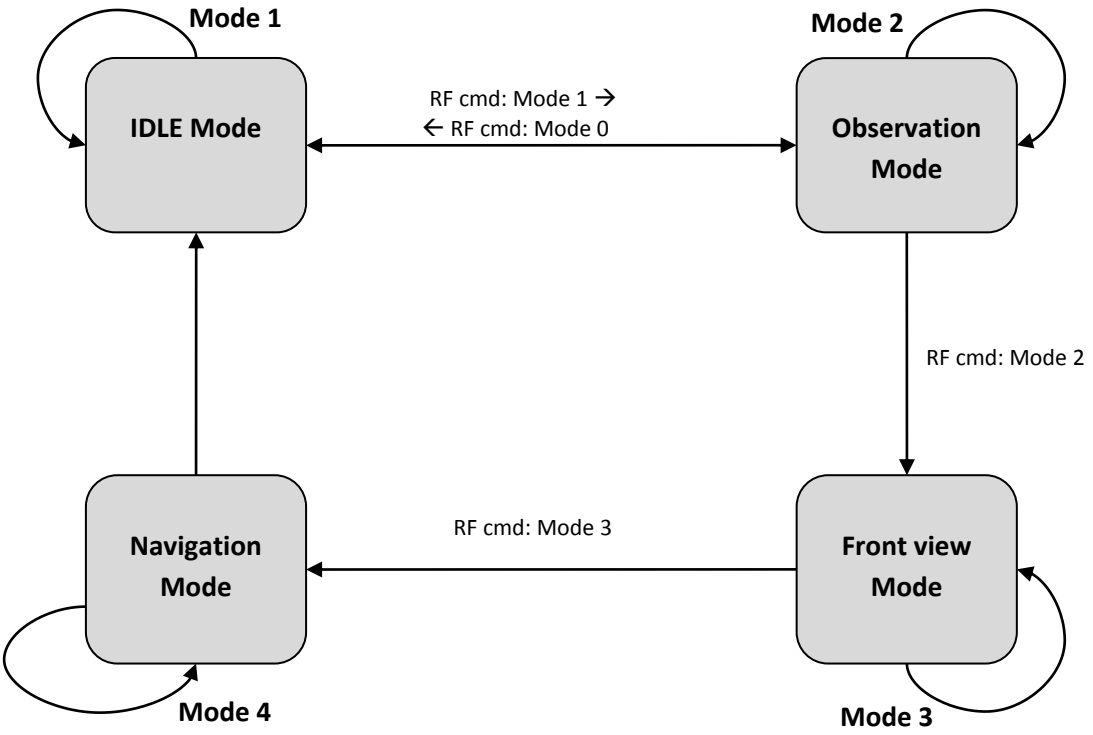

Figure 3. 1: Robotic Platform's Modes of Operation

One of the advantages of using a multi-modal design is the scalability or the ability to expand the system's tasks by simply adding individual modes without interfering with other modes of operation. A multi-modal system also enhances flexibility of the design in the sense that modes of operation can be modified or even removed without sabotaging the overall performance of the system. For instance, UAVs, Quadra-copters, equipped with cameras enhance the flexibility and scalability of the system by increasing the range of motion, are able to fly and gather data from places otherwise difficult or impossible for a heavy surface-drive platform to reach. These features along with versatility, robustness and low cost become handy when the system is unable to perform its regular tasks due to an external interference or disturbance from the surrounding environment. In this case the system can carry on by operating in other modes.

In the following subsections, we will describe the required tasks, physical configuration and resources of each mode to come up with the architectural organization of the system.

3.3.1 Idle Mode

Function: In the idle mode, the robotic platform is in a stationary state and conserves energy by shutting down all the unnecessary components. The RF link is engaged to receive further instructions from the control center.

Physical configuration: The platform is stationary, the Head and Tower are in low position facing the horizon, as shown in Figure 3.2.

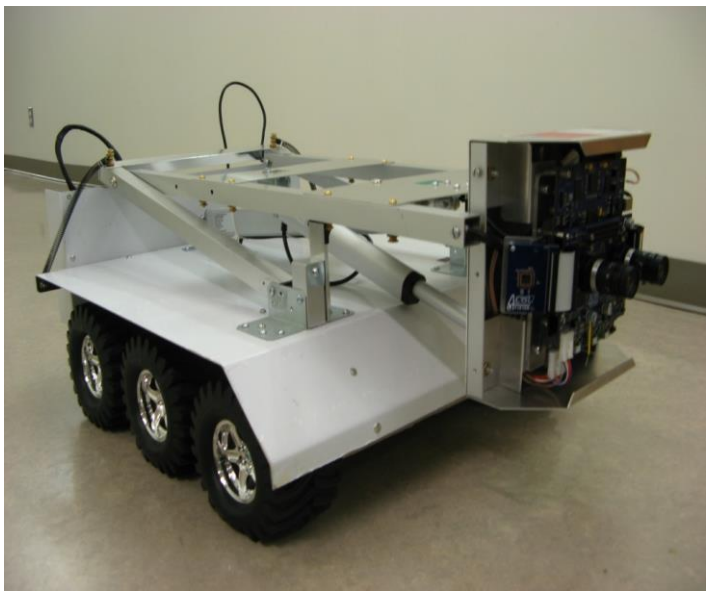

Figure 3. 2: Description of the Idle Mode

Resources: To realize the system in this mode, the system requires the following subsystems as shown in Figure 3.3:

- Power-saving

The power saving module is responsible for conserving energy by turning off all unnecessary devices and placing them in disabled mode. Such components as the camera module, motors and transceivers may be turned off to save power.

- RF-Link

The RF-Link module is responsible for sending and receiving commands from the master's side and direct the mode transitions based on the received command.

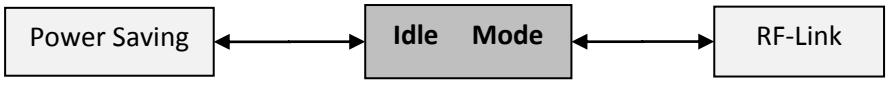

Figure 3. 3: Initial Block-Diagram of Idle Mode

3.3.2 Observation Mode

Function: In the observation mode, the system captures, processes, compresses and transmits 3D-Semi-Panoramic images of the scene by means of the 4 video sensors located at the top of the Head compartment. An RF-Link is also used to obtain further instructions, such as Head angles and future destinations.

Physical Configuration: The platform is in stationary state, the Head, which contains the video sensors is facing the horizon, parallel to the body of the platform. The Tower, which connects to the base of the platform to its "Head," is in high position, as shown in Figure 3.4.

Figure 3. 4: Description of the Observation Mode

Resources: to realize the system in the Observation mode, the system requires the following subsystems, which will be described below:

- RF-Link

The RF-Link module, discussed in Section 3.3.1 can be applied in the observation mode as well.

- Video-Acquisition and Pre-Processing Unit

The video-acquisition and pre-processing module is responsible for capturing 3D-Semi-Panoramic images of the scene by means of 2 frontal and 2 peripheral video sensors located at the top of the Head. The 3D-SP video packages are compressed by the JPEG2000 compressors and then transmitted to the control center via Coded Orthogonal Frequency Division Multiplexing (COFDM) Modulator based Satellite transmitter as discussed in [1]. This module has been implemented as shown in Figure 3.5 by [1].

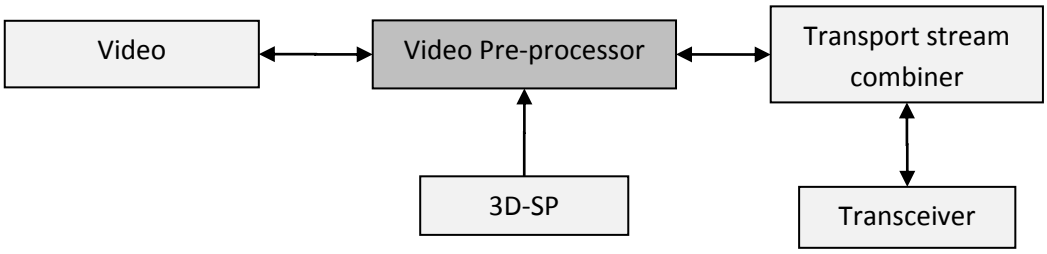

Figure 3. 5: Video-Acquisition and Pre-Processing Subsystem Block-Diagram Courtesy of [1]

- Linear Actuators

To move the Head, which contains the video sensors, up or down, a linear actuator sets the level of observation. Another linear actuator can be used to enhance the degree and focus of observation by lifting the Head vertically.

- Accelerometer

The angle of observation can be precisely set by using an accelerometer placed in the Head compartment mode since the platform is stationary and the orientation of the robotic platform relative to any surface can be simply identified.

The overall required components of this mode can be categorized as illustrated in Figure 3.6:

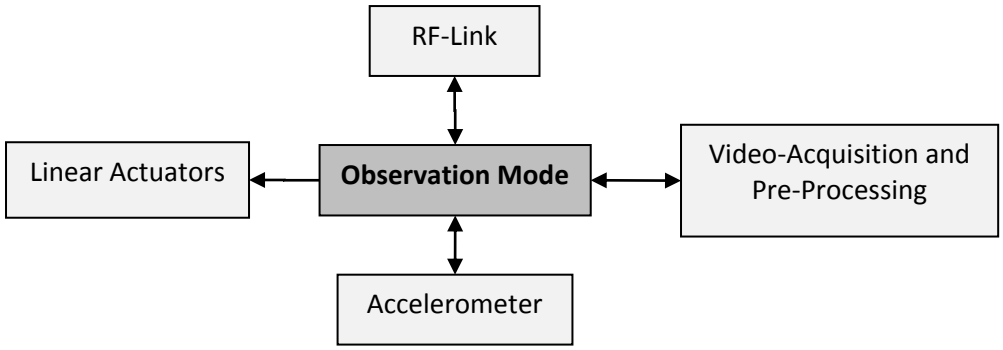

Figure 3. 6: Initial Block-Diagram of Observation Mode

3.3.3 Front-View mode

Function: Two video-sensors are capturing 3D images at each of N-angular positions of the Head. Disparity mapping analyzer is "ON" to create the map of obstacles and their allocation on frontal distance of X-steps of "Body" movement. COFDM RF-transmitter is actively transferring the number of compressed 3D images to the mission control centre, and an RF-Link is present to obtain further instructions.

Physical configuration: The platform is in the stationary state similar to idle mode. However, the Head is moving from the horizontal position to the vertical as shown in Figure 3.7.

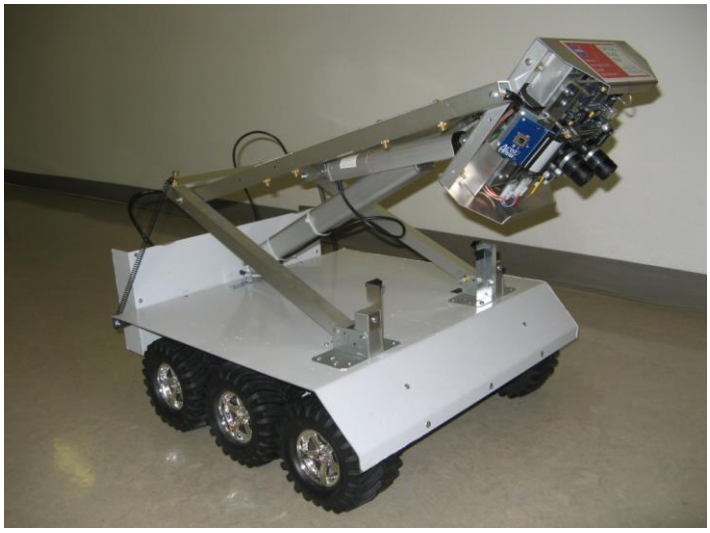

Figure 3. 7: Description of the Front-View Mode

Resources: The Front-View mode requires the following resources:

- 3D- Image Interpolation Unit

This subsystem is similar to the VAPP, in terms of capturing, compressing and transmitting the image to the control center. However, only 2 frontal cameras are used to create a map of obstacles and their distance from the platform. The generated map is then sent to the control center, which will use it to predict the robotic platform during navigation, whenever no video data is being transmitted to the control center.

- Linear Actuator

A linear actuator is also used to set the level of observation along the Y-axis to help provide a frontal view in 3D.

- RF-Link

The RF-Link module, discussed in Section 3.3.1 can be applied in the observation mode as well.

3.3.4Navigation Mode

Function: In the navigation mode, the system drives the robot's motors, monitors displacement, avoids hitting the obstacles and communicates with the control center for verification and receiving further instructions.

Physical configuration: The Head is in low position, facing the horizon to avoid hitting obstacles. The Tower is also in low position similar to the configuration of idle mode to enhance stability and avoid roll-overs while the platform is moving toward the requested location.

Resources: The navigation mode consists of the following individual component, which are illustrated in Figure 3.8:

- RF-Link

The RF-Link module, discussed in Section 3.3.1 can be applied in the observation mode as well.

- DC brushed motors

6 DC-brushed motors are required to enable the heavy robotic platform to move on any terrains and carry the subsystems placed in the Head compartment.

- Motor controller

A motor controller is required to control the speed and direction of the motors.

- Odometer

An odometer is used to measure the distance travelled by counting the number of times it takes for one wheel of the robot to make a full wheel rotation.

- Obstacle sensors

Obstacle sensors are required to avoid colliding with any objects that may be in the way. The Obstacle sensors consist of optical devices placed around the Head compartment to protect it against hitting unwanted objects.

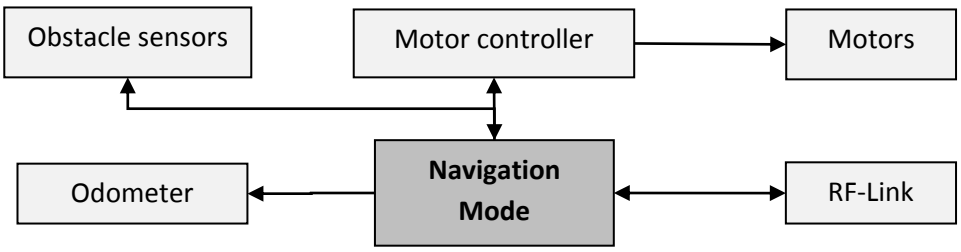

Figure 3. 8: Initial Block-Diagram of Navigation Mode

3.4 System Architecture Organization

The modes of operation analysis, discussed in the previous section made it possible to define a set of functional specification required for the general architecture development of the system. As can be seen in the previous sections, common resources exist in different modes of operations. However, the respective configurations and functions may vary depending on the active mode. For instance, even though the Tower is involved in both observation and navigation modes, switching to observation mode from the navigation mode require stopping the robotic platform and elevating the Tower to capture video data. Therefore, the main goal of this section is to determine the actual required resources, their functions and interactions in different modes of operation. Table 3.1 demonstrates the status of resources in the four modes of operation.

Table 3. 1: Resource Interactions among the Modes of Operation

Mode/Resource	Idle	Observation	Front-View	Navigation
Power-saving	**Enabled**	Disabled	Disabled	Disabled
VAPP	Disabled	**Enabled**	Disabled	Disabled
Tower Actuator	Low	High/Low	High	Low
3D-Image Int.	Disabled	Disabled	**Enabled**	Disabled
Head Actuator	Low	Low	High/Low	Low
Motors & Controller	Disabled	Disabled	Disabled	**Enabled**
Odometer	Disabled	Disabled	Disabled	**Enabled**
Obstacle Sensors	Disabled	Disabled	Disabled	**Enabled**
RF-Link	**Enabled**	**Enabled**	**Enabled**	**Enabled**

As can be seen in Table 3.2, the RF-Link is present in every single mode. However, the motors, odometer and obstacle Sensors are active only in navigation mode. Moreover, the Video-Acquisition and Pre-Processing, VAPP, is only enabled when the system is in observation mode. The power saving is present only during the idle mode; last but not the least, the Head is active only during the front-view mode.

Furthermore, based on the proposed mode of operation analysis in Figure 1, the level of Tower does not change during the front-view mode as the Tower is already in high position when transitioning from observation to front-view mode. Hence, resource optimization can be achieved on to dedicate appropriate resources to each mode based on the defined and default functions described in Section

3.3. This analysis will be discussed shortly after defining the required processing unit to implement the described modes of operation.

Due to what has been discussed in chapter 2 of this thesis, FPGAs provide higher performance and lower overall costs compared to conventional microprocessors. The ability of FPGA to provide soft and hard core processing options, such as PicoBlaze (obsolete), MicroBlaze and PowerPC, helps the system designer to exploit the advantages of using 32-bit microcontrollers as well as designing parts or an entire system in the form of Virtual Hardware Components.

In the subsequent sections, the organization of each individual operational mode of the proposed system architecture is described.

3.4.1Idle Mode Architecture

Due to the required resources and tasks mentioned in Sections 3.3 and 3.4, the idle mode consists of two major components, namely, the power-saving and RF-link components. Power-saving exists to conserve energy, hence disabling inactive components of other modes of operation and eliminating or reducing unnecessary activities to help us to reach this goal.

On the other side, an RF-transceiver is required to communicate with the control center, located at the master's side. The implementation of the RF-Link interface will be determined based on what kind of transceiver is used.

To achieve the goal of minimizing energy use, we need to define the subsystem's constraints in order to choose the best Radio Frequency solution for the design. The most important system features include the range of operation, data rate, overall performance and development and actual costs. Therefore, some of the wireless standards authored by the Institute of Electrical and Electronics Engineers (IEEE), including IEEE 802.11, 802.15.1 and 802.15.4 that can be used in mobile applications are examined and summarized in Table 3.2.

Table 3. 2: Comparison of RF Technologies

Standard	Bluetooth	ZigBee	WLAN/WiFi
Freq. band	2.4 GHz	868/915 MHz, 2.4GHz	2.4 GHz,5 GHz
Max signal rate	1 Mb/s	40kb/s,250 Kb/s	54 Mb/s
Nominal range	10 - 100m (industrial)	**10 – 100+ m**	**100 m**
Channel bandwidth	1 MHz	0.3/ 0.6MHz; 2MHz	22 MHz
Modulation type	GFSK	BPSK(+ASK), O-QPSK	COFDM, CCK,M-QAM
No. of nodes	**7**	**64000**	32
Security	**Low**	**High**	**High**
Complexity	**complex**	**Simple**	**Very complex**
Cost	**Low**	**Low**	**High**
Power consumption	**Low**	**Low**	**High**

Since the amount of data to be transmitted from/to the control center to/from the mobile platform is limited to a few instructions, the minimum bandwidth does meet our requirements. Hence, based on the results shown in the table above, the ZigBee option makes a better candidate than Bluetooth and WLAN/WiFi for transmiting the commands between the master and slave sides.

The next step is to choose a processing platform to interface with the selected RF standard. Generally speaking, the amount of information transmitted to and from the system is limited, hence speed and parallelism does not play a crucial role in our system design. For this reason, a slower processor, such as instruction-based soft-core processor, can be used. Using interrupt-based RF-Link, which helps maintain the system without waiting for the arrival of further instructions, can help overall performance, design implementation and time required to develop the system all due to its simplicity. The overall component description of this mode is shown in Figure 3.9.

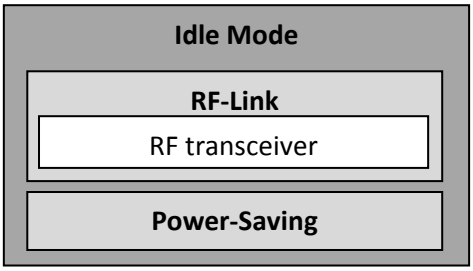

Figure 3. 9: Idle Mode Block-Diagram

3.4.2 Observation Mode Architecture

As mentioned earlier, the observation mode shares the RF-Link with the idle mode and uses the previously implemented VAPP subsystem by [1, 2] to capture, process and transmit 3D-SP images to the control center. However, the RF-Link can be omitted from the module to simplify and optimize the design. Therefore, the system gives control to the idle mode upon finishing the required tasks of observation mode. The use of accelerometer and the linear actuator help to accurately adjust the level and focus of observation. Hence the following hardware elements can be supposed for the architecture design of this mode.

- Tower:
    - Accelerometer
    - Linear Actuator
- VAPP:
    - 4 Image sensors
    - Video Compressor/Decompressor
    - COFDM Transceiver

Generally, controlling of linear actuators can be as simple as applying voltage to their inputs. Hence, performance and speed are not subjects of concern. However, linear actuators change directions by having the voltage in reverse polarity. Therefore, a hardware mechanism needs to be set in place to alter the polarity of the input voltage of the actuator.

Furthermore, depending on the Accelerometer, various communication protocols can be implemented. However, the level of actuators needs to be adjusted while the actuator is running. Hence, implementing a parallel procedure would be a better option than using a sequential processor. The overall component description of this mode is shown in Figure 3.10.

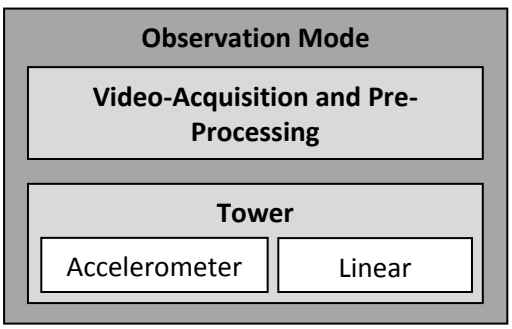

Figure 3. 10: Observation Mode Block-Diagram

3.4.3 Front-View Mode Architecture

Similar to the observation mode, the front-view mode can exclude the RF-Link since the control of the system makes the transition back to the idle mode, by which the system can communicate with the control center. Moreover, a linear actuator is used to set the Head compartment at defined angular positions to help the 3D-image interpolation unit capture a map of obstacles before entering the navigation mode, when no video data is transmitted to the control center due to the emergence of noise and vibrations during the movement. This feature helps the operator in the master's side to guide the robot through the best possible route and have a clear vision of the scene without any interruption during the navigation mode. Based on what has been discussed, the following hardware elements can be presumed for the architecture design of this mode.

- Head:
    - Linear Actuator
- 3D- Image Interpolation:
    - 2 out of 4 Image sensors of VAPP
    - Disparity mapping analyzer
    - Video Compressor/Decompressor
    - COFDM Transceiver

The architecture organization of this mode is very similar to the observation mode because RF-link is shared among both modes and can be omitted as the control of the system is makes the transition to the idle mode upon task completion. Moreover, the same analysis performed for the linear actuator of the Tower can be applied to the one associated with Head. Moreover, the focus of this thesis is on the control section of this project. Hence the 3D-image interpolation subsystem will not be discussed here. The overall component description of this mode is shown in Figure 3.11.

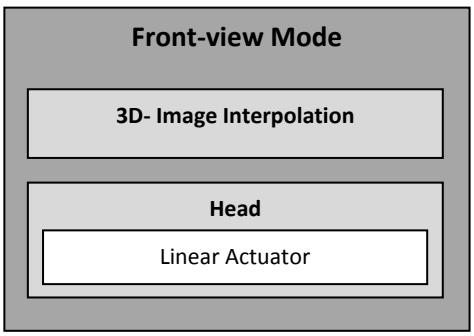

Figure 3. 11: Front-View Mode Block-Diagram

3.4.4Navigation Mode Architecture

The navigation mode is responsible for driving the platform to the destination specified by the control center based on the map obtained via the Front-View mode. No video will be transmitted to the control center since a moving platform can generate a great deal of noise and vibration that will affect the quality of the images, performance of the system and lead to an unpleasant experience by the operator. Hence, by sending the current location of the platform to the control center, the video processing subsystem [1], located at the master's side can interpolate the frame based on the current speed and location of the robot and present a real-time video of the scene to the operator. As mentioned in Section 3.4, the navigation mode consists of the following hardware elements:

- Navigator:
    - Motors
    - Motor Controller
    - Odometer
    - Obstacle Sensors

The overall component description of this mode is grouped and shown in Figure 3.12.

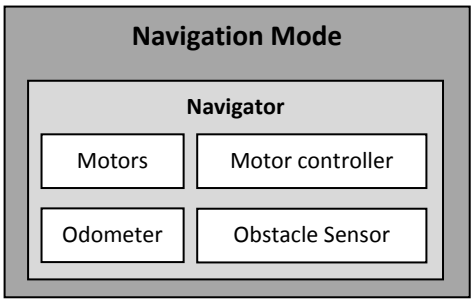

Figure 3. 12: Navigation Mode Block-Diagram

The motor controller is responsible for setting the speed and direction of the motors. Depending on the type of motor controller, which can be designed or purchased off-the-shelf, a specific communication interface is required. This interface could be as simple as toggling a few pins or as complicated as implementing a UART interface.

Furthermore, the navigator component of the navigation mode also consists of an odometer unit placed across the shafts of the middle wheels and obstacle sensors which are placed around the Head of the platform, as the Head is located at the very front of the platform during the movement. The overall description of the navigator module is shown in Figure 3.13.

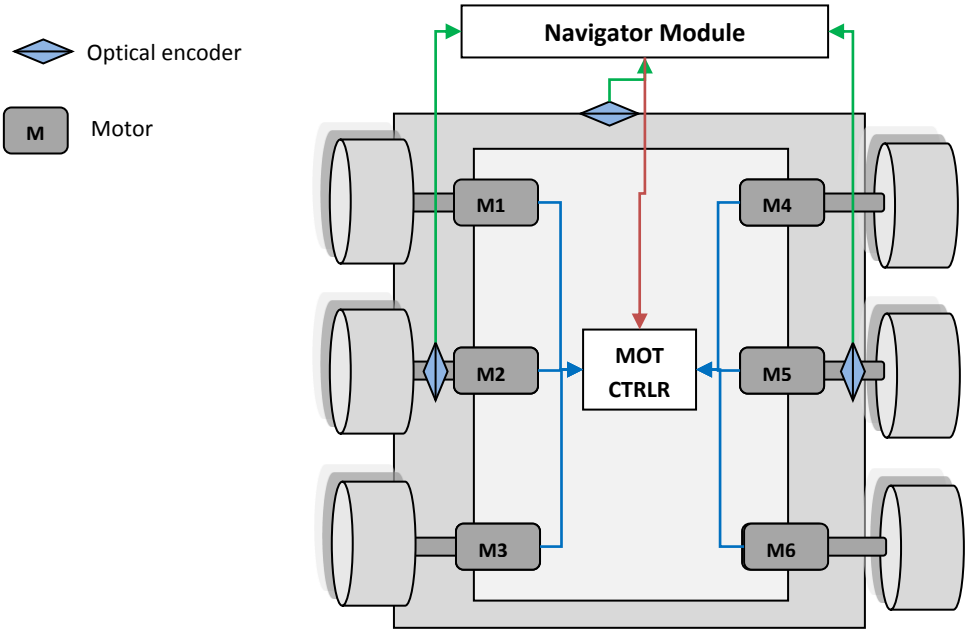

Figure 3. 13: Description of the Navigator Module

Even though the functions described in Section 3.3.4 suggests that the Navigator may be implemented in a sequential form, parallelism offers more benefits, such as the ability to avoid and respond to obstacles and interferences in real time and to navigate with higher precision. Hence, hardware implementation of the navigator component will lead to higher performance, speed and less logic resources and area. In order to match the performance of the hardware implementation, the sequential processor needs to run at a faster clock rate, which is not possible in the case of embedded soft-core processor; in the case of using microcontroller, additional algorithms to increase the frequency leads to higher power consumption, which would require extra specialized cooling system to eliminate the excessive and potentially damaging heat from the device.

3.5 Summary

In this chapter, we discussed the specification of a true reconfigurable robotic system and proposed the idea of “form follows function” to emphasize that a reconfigurable system is not only defined by reconfiguration of its physical/mechanical components. Rather, the physical configuration of the system may alter based on the operational task of the system to accommodate its new functionality. Therefore, we proposed a robotic platform to be integrated with the previous work done regarding the telepresence system to demonstrate our proposed concept. To realize such a system, a set of tasks were specified for the system, which were then divided into multiple modes of operation. The required resources and components for each mode of operation were discussed to arrive at the actual architecture organization of the design.

4. Implementation of the Multi-Modal Reconfigurable Robotic System

4.1 Introduction

The objective of this chapter is to discuss the specific aspects of implementation for the proposed reconfigurable robotic platform. The analysis and architectural organization of the system, presented in the previous chapter, makes it possible to select the most appropriate tools and components to create a functioning prototype, which can allow us to test and verify all the aspects of the system performance. The implementation of the proposed system is divided into these two major parts: 1) the electro-mechanical part and 2) control and communication part.

The electro-mechanical part of the system consists of all the elements constructing the frame and physical base of the platform, which are beyond the scope of this paper. The control and communication part of the platform is the part involving the component selection, system architecture implementation as well as the test and verification of the design. Therefore, in this chapter we will look at the process of selecting components and associated hardware, firmware and software components for each mode of operation.

4.2 System design Implementation

As mentioned in Section 3.2, the proposed robotic platform is to be integrated with a more complicated system known as 3D-P telepresence system that consists of two major subsystems: Video Acquisition and Pre-Processing Subsystem and Video-Processing Subsystem, which are interconnected via a communication channel consisting of data and control busses (see Figure 4.1).

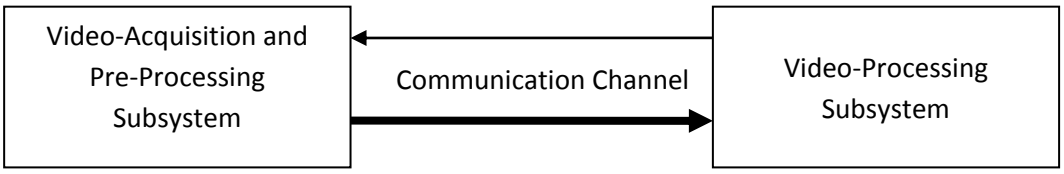

Figure 4. 1: Telepresence System Architecture Courtesy of [1]

The Video-Processing Subsystem is located on the master side and performs the reception of the compressed video stream, distribution of video data to a storage memory, execution of various image processing algorithms and output of the results to the display system. The analysis of this subsystem is beyond the scope of this document. The focus of this thesis is on the slave side of the operation and the extension of previous work done in this area based on the system architecture discussed in the previous chapter. The implementation of the design is divided into the following components:

- The hardware, HW, component of the design refers to any part of the system that consists of electronic circuits or as a set of electronically interconnected components located on a common area.
- The firmware, FW, component of the system refers to any type of circuits described at logic level using HDL and represented in a form of configuration bit stream.
- The software, SW, component of the system is referred to as any type of instruction based program described in High-level or Low-level programming language and executed on a sequential general-purpose processor, such as microprocessor of a microcontroller or soft/hard-core processor of an FPGA.

In the following subsections, the organization of each individual operational mode of the proposed system architecture is described, and the logic behind implementing each part in HW, FW and SW is discussed.

4.3 Modes of Operation

4.3.1 Idle mode

As mentioned in Chapter 3.4.2, an RF-link is required to provide a wireless communication between the slave and master sides. In the earlier work, an RF transceiver interface component using serial differential link [1] was implemented to communicate between the two subsystems described in Section 3.4. However, the main purpose of implementation of such an RF interface was to transmit the captured video data to the control center. Therefore, to prove the functionality of our proposed system and test the platform, another RF transceiver is required to transmit data to/from the robotic platform from/to the control center.

4.3.1.1 Idle Mode Hardware Architecture

An XBee module, which is the embedded solution to provide wireless connectivity using ZigBee, IEEE 802.15.4 protocol, was selected as the backbone of the RF-Link module. From the wide variety of XBee modules, XBee-Pro 802.15.4 Extended Range module was selected with the following features [40]:

- Power output: 63mW(+18dBm)
- Indoor/Urban range : up to 90m
- Outdoor: up to 1.6 Km RF Line-of-sight
- Receiver sensitivity : -10dBm
- RF Data rate: 250kbps
- Interface data rate: up to 115.2 kbps
- Operating frequency: 2.4 GHz
- Supply voltage: 2.88-3.4 VDC
- Current: 215mA (Transmit), 55mA(Receive) @ 3.3V
- Antenna option: Chip antenna or wired whip antenna
- Interface option: UART

The features of the selected XBee module indicate that we need to implement the universal asynchronous receiver/transmitter, UART, serial interface to communicate with the XBee module. The required UART settings for the XBee module are as follows:

- Baud rate (BR): 9600 bps
- Data: bits : 8 bits - Parity: none - Stop bit: 1 bit

4.3.1.2 Idle Mode Firmware Component

To develop and implement the functionalities of the UART to interface the processor with the XBee module, we have the option of implementing the design in the form of Virtual Hardware Components, VHC, or Virtual Software Components, VSC, on an FPGA's soft-core processor. The following subsections describe the features of each implementation approach and the required FW associated with the RF-Link module.

4.3.1.2.1 Implementation of UART in VHC

To simplify the design, the UART protocol interface is divided in two subsystems, namely, Receiving and Transmitting subsystems.

- **Receiving Subsystem**

Due to the asynchronous nature of UART, in which no clock information is sent from the transmitted signal, the oversampling procure is commonly used to estimate the middle points of transmitted bits and correct reception of the bits [41]. The conceptual block diagram of a UART receiving subsystem consists of three components, as shown in Figure 4.2:

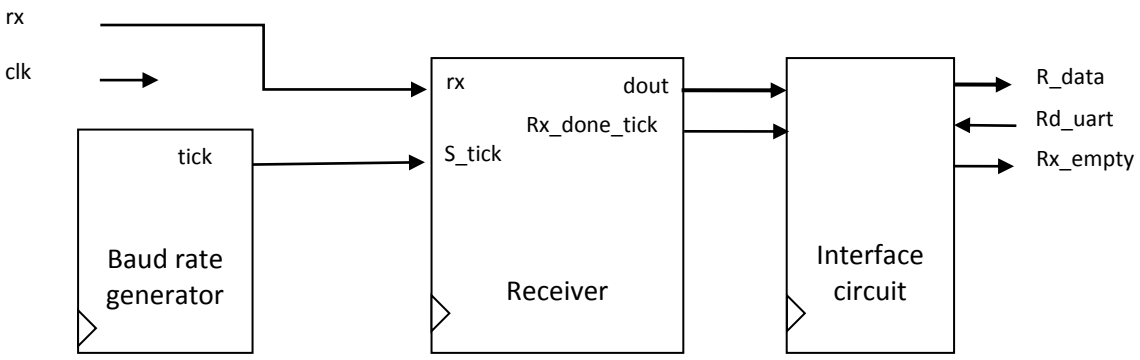

Figure 4. 2: Block-Diagram of a UART Receiving Subsystem Courtesy of [41]

- UART receiver: the circuit to obtain data word via oversampling, in which a 3-state state machine is used to process the start bit, data bits buts and stop bit upon receiving a notification signal from the baud rate generator.
- Interface circuit: the circuit to provide buffer between the UART receiver and the main system to prevent receiving a word data multiple times.
- Baud rate generator: the circuit to generate sampling signal with a frequency of 16 times UART designated baud rate. Hence, the following equation is applied to estimate required clock cycle for a one-clock-cycle tick.

$$Counter_{baud\ rate\ generator} = \frac{System\ clock}{Baud\ rate\ \times 16}$$

- **Transmitting Subsystem**

The organization of transmitting subsystem is similar to the receiving one. It is essentially a special shift register that loads data in parallel and shifts in data bit by bit and then reassembles them. The transmitting subsystem consists of a UART transmitter, baud rate generator and interface circuit. The interface circuit is similar to the receiving subsystem with the exception that the UART transmitter clears a flag or reads a buffer, while the main system sets the flag or writes into the buffer to indicate whether any received data word is available [41]. The lock diagram of a complete UART system as proposed by [41] is illustrated in Figure 4.3.

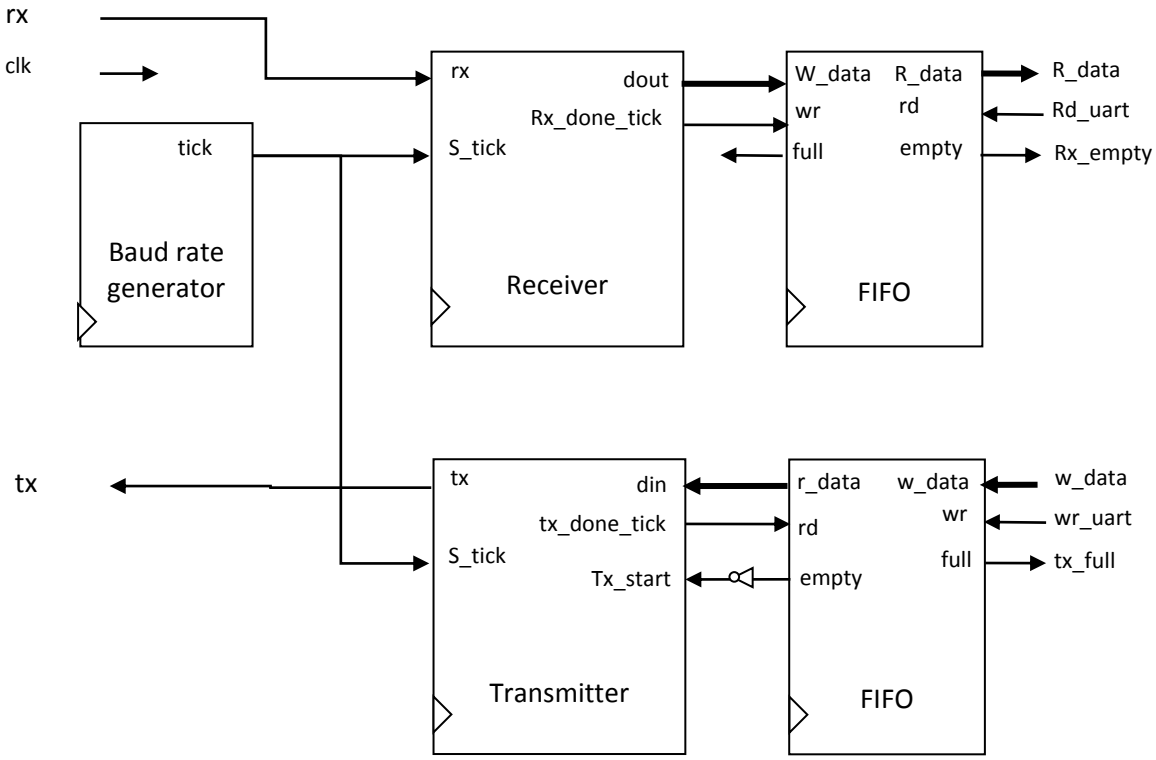

Figure 4. 3: Clock-Diagram of a Complete UART Courtesy of [41]

4.3.1.2.2 Implementation of UART in VSC

The implementation of the UART interface in MicroBlaze seems easier using Xilinx Platform Studio, XPS. The XPS implements the hardware and software functionality of the UART Lite that performs parallel to serial conversion on characters received from CPU through Peripheral Local Bus, PLB, and

serial to parallel conversion on characters received from a serial peripheral [42]. The XPS UART Lite is capable of transmitting and receiving 8, 7, 6 or 5 bit characters, with 1 stop bit and odd, even or non-parity. The detailed block diagram of the XPS Lite UART is shown in Figure 4.4, where the top level modules are:

- **PLB Interface Module**: It provides bi-directional interface between the UART module and the PLB and implements the PLB protocol logic.
- **UART Lite Register Module**: It includes all memory mapped registers and interface to the PLB through the PLB interface module. It consists of an 8-bit status register, an 8-bit control register and a pair of 8-bit Transmit/Receive FIFOs.
- **UART Control Module**: It consists of an RX module, a TX module, a parameterized baud rate generator (BRG) and a control unit. It incorporates the sate machine for a) initialization and b) start and stop bit control logic.

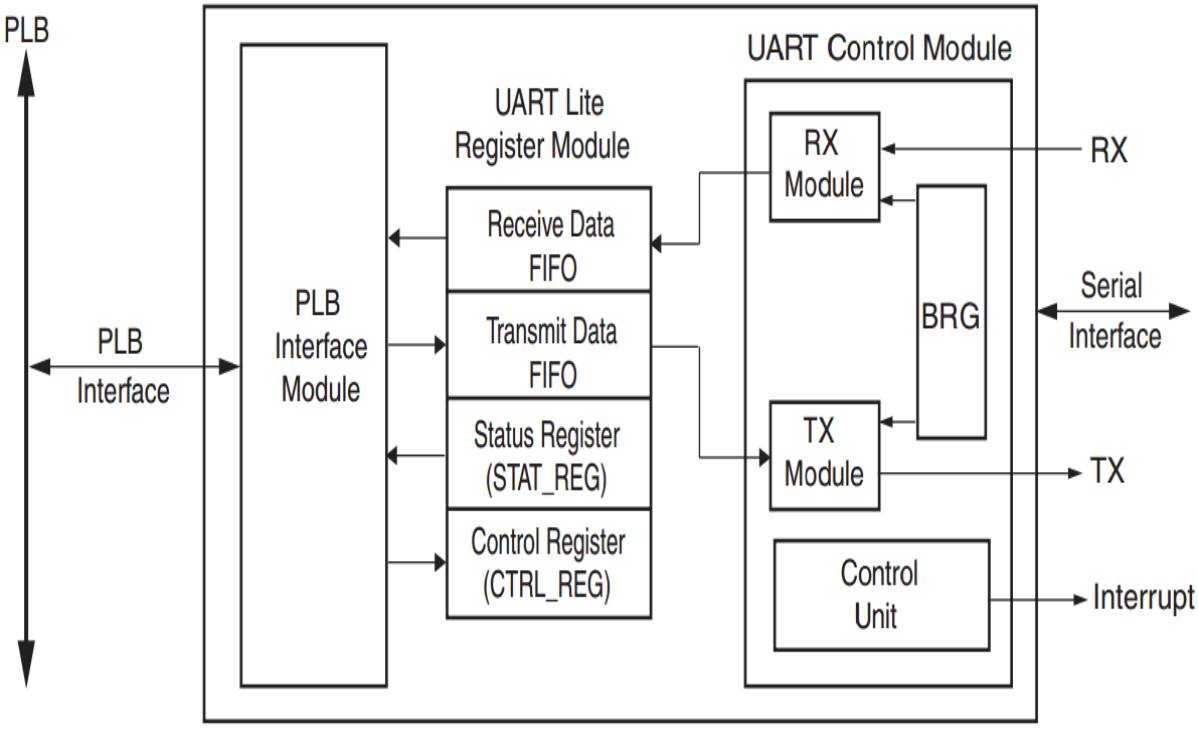

Figure 4. 4: Block-Diagram of XPS UART Lite Courtesy of [42]

Based on the analysis made above and due to the fact that the customized UART module, supplied by Xilinx can be considered as a gate-level description that utilizes Xilinx-specific components, the VSC option is a better choice in terms of overall efficiency, development time, cost and sufficient performance, as suggested in Chapter 3.

Furthermore, since the architectural organization of design consists of hardware and software components, as mentioned is Section 3.4, it is necessary to implement a storage device to retain the crucial information from each mode of operation, such as the distance to be travelled by the robotic platform. Therefore, an 8KB-dual block RAM memory was implemented to store the information to be shared among the modes of operation. Therefore, the component symbol of the soft-core processor part of the design consisting of the required memory blocks and their controllers, the UART, RS232, interface component, clock and reset components and a debug module are all depicted in Figure 4.5. The more detailed MicroBlaze structure is found in appendix A.

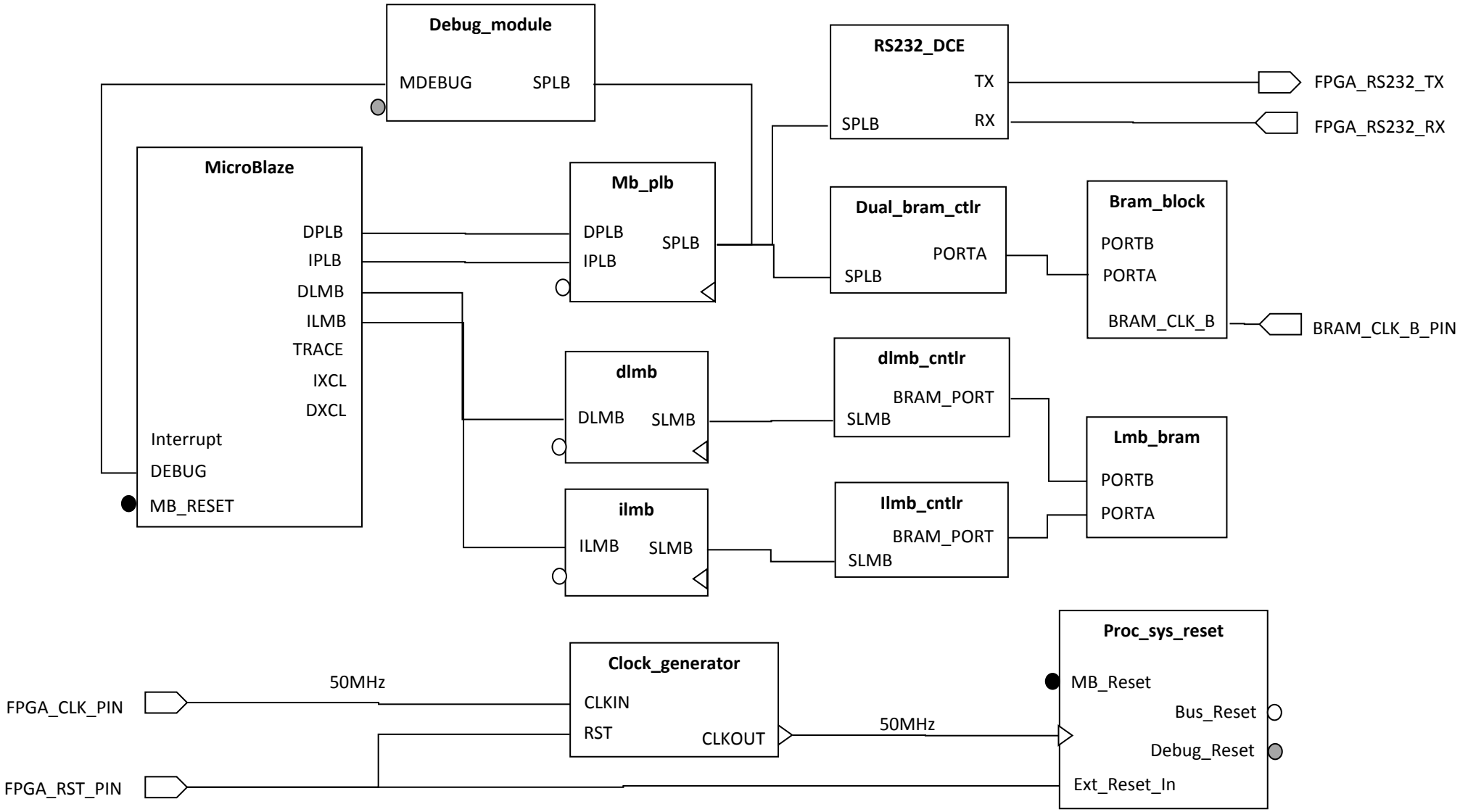

Figure 4. 5: MicroBlaze Internal Organization

❖ **Component Symbol**

Based on what has been discussed, the component symbol of the idle mode consists of input and output ports for the RS232 interface, a clock and a reset input ports to feed the component with a clock rate of 50 MHz and half the activities of the component, and finally the required dual BRAM ports to interface the VSC and VHC of the design (see Figure 4.6).

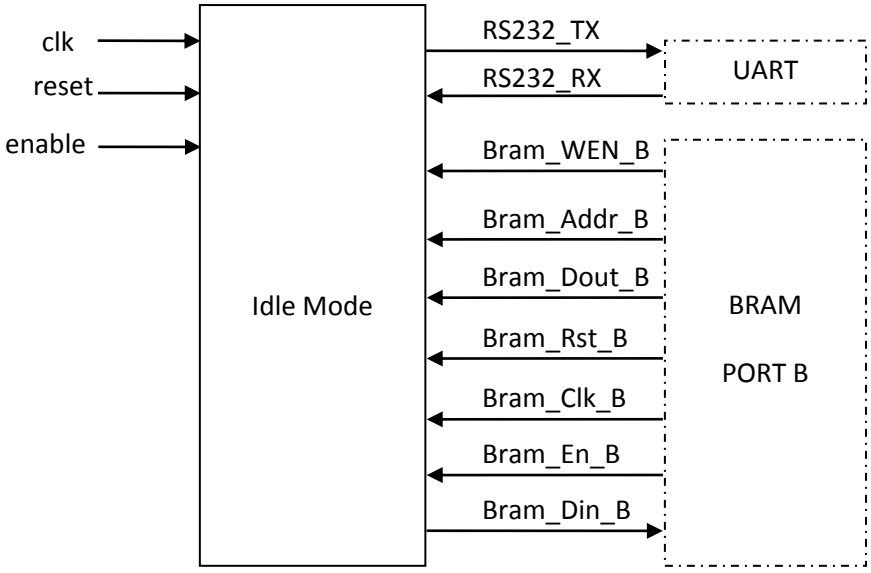

Figure 4. 6: Idle Mode Component Symbol

As shown in the figure above, Port B of the dual block Ram is inputted to the VHC part of the design, while the port A is accessible in the VSC part of the design. Therefore, data written via either port of the BRAM can be accessed and manipulated by the other port.

4.3.1.3 idle Mode Software Component

The software component of the idle mode is responsible for receiving and transmitting commands and acknowledgements to the control center via the RS232 interface with XBee module. The received instructions are placed in the shared memory via port A of the BRAM.

4.3.2 Observation Mode

The observation mode of the system consists of a VAPP module, a linear actuator and an accelerometer to meet the functional specifications. The VAPP subsystem has been implemented by [1] and will be used with no modifications. Therefore, the remaining hardware components to be examined are the linear actuator and the accelerometer placed in the Tower module.

In the following subsections the hardware and firmware components of this mode are addressed, and the observation mode component symbol of the system to be implemented is described.

4.3.2.1 Observation Mode Hardware Architecture

In this section the required hardware components of this mode and the steps in selecting those components are discussed.

4.3.2.1.1 Linear actuator

The linear actuator acts as the robot's manipulator and is strategically placed to level the camera module up and down. Some things need to be considered when selecting a linear actuator, such as the force required to move the load, the distance the load needs to be moved and the time required to move the load. Based on the mechanical hardware design of the robotic platform, the following specifications were considered:

- Maximum weight of the camera module, including accessories : W= less than 50lb ≈ 23Kg
- Maximum angle of inclination : $\theta = 75°$
- Maximum distance: 25 cm
- Maximum speed: 0.02 m/S^2
- Peak Acceleration: 0.04 in/S^2

In order to lift the VAPP module, a 12" stroke linear actuator capable of lifting a full load of 40 lb; features below were selected from Firgelli Technologies [43].

- Maximum speed: 4.45 cm/sec at no load and 2.54 cm/sec with load
- Stroke size : 12"
- thrust/pull force: 44 lb
- holding force: 100 lb
- Require voltage : 12 VDC
- Maximum current: 5 A
- Power consumption: 60 Watts

To control the direction of the linear actuator, the voltage level needs to be set because positive voltage will cause an outward movement, while negative voltage will cause an inward movement. Therefore, an external hardware circuit is required to alter the polarity of the input voltage based on the required direction of movement. Relays which are electrically operated switches using electromagnets to control a circuit by a low-power signal, can be used to achieve the above goal. Therefore, a pair of relays from ZETTLER can be used along with a Darlington transistor array chip from Texas Instruments to form the circuit shown in Figure 4.8.

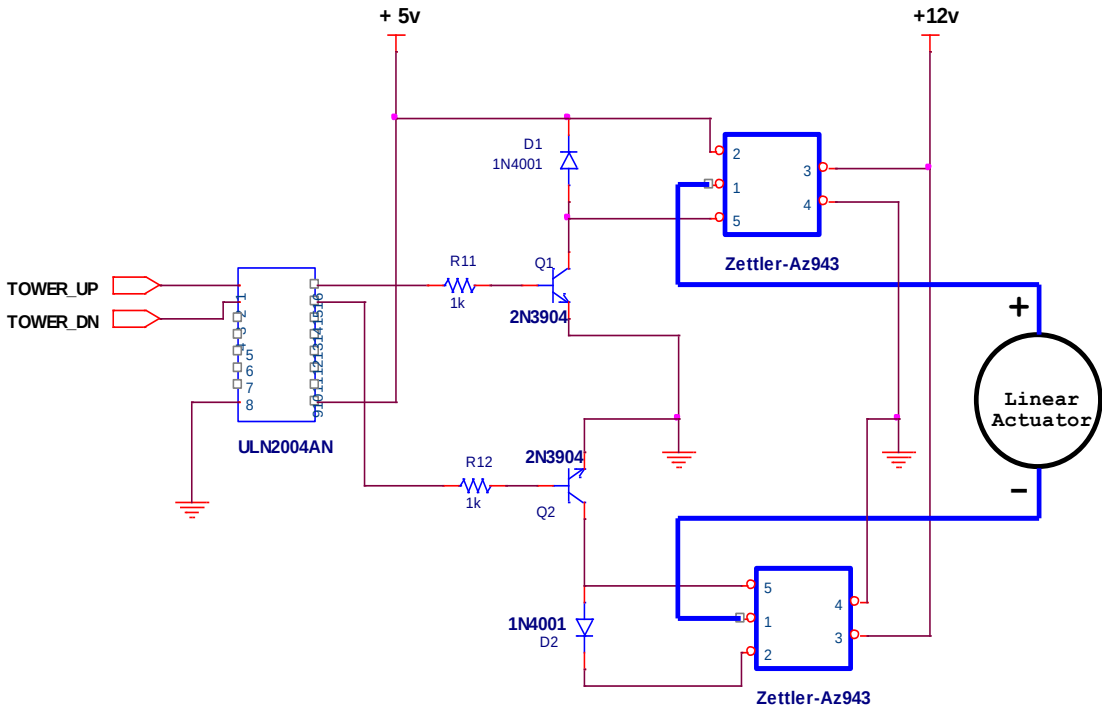

Figure 4. 7: Linear Actuator Controller Circuit

As can be seen in the above schematic, the linear actuator is initially connected to ground on both poles due to the default position of the relays. By setting the right pin, (Tower_UP or Tower_DN), the associated relay becomes active, and the appropriate voltage is fed to the actuator. It should be noted that only one relay should be set at each specific time to avoid applying +12v to both poles of the actuator.

4.3.2.1.2 Accelerometer

An accelerometer can be used to give accurate tilt angle measurement in three dimensions. Even though Gyroscopes seem to be a better solution, since they maintain their level of effectiveness by measuring the rate of rotation around a particular axis, the accelerometers can fulfill the requirement of the observation mode as the platform is in stationary state and therefore, the orientation of the robotic platform relative to any surface can be simply identified.

A triple axis accelerometer kit from Sparkfun Electronics, containing BMA180 of Bosch, was selected to meet the requirement of this mode that presents the following features [44]:

- Three-axis accelerometer with integrated temperature sensor in the range of -40°C to +85°C
- Ultra high performance g-sensor with 12-bit and 14-bit ADC operation
- Digital Interfaces: 4-wire SPI, I2C, interrupt pin
- Wide variety of measurement ranges (±1g, 1.5g, 2g, 3g, 4g, 8g and 16g)
- Programmable integrated digital filters
- Various modes of operations: Low-noise, Low-power, sleep mode, wake-up mode and self-test

4.3.2.2 Principals of Observation Mode Operation

As discussed earlier, the accelerometer is placed in the Head to measure the elevation of the Head subcomponent by means of the Tower's linear actuator. Upon entering the observation mode, system puts a Tower up at a specific height set by the operator in the control center. The VAPP module becomes active and starts capturing, processing and transmitting the video-data to the control center. The implementation of the VAPP module is beyond the scope of this thesis, as the focus is more on the control and communication of the robotic platform to extend the previous work by adding mobility to the system. The main processor for implementing the functions of this mode of operation is an FPGA device, which was addressed in Chapter 3. In the following section, the implementation of functions associated with each module of the observation mode will be discussed.

4.3.2.3 Observation Mode Firmware Component

The observation mode of the system is designed to interface the selected linear actuator and accelerometer with the processor. In the following sections the required firmware to interface with the hardware components are discussed and the component symbol of the observation mode will be illustrated.

4.3.2.3.1 Linear Actuator

As can be seen in the schematic shown in Figure 3, by simply toggling two general purpose output pins of an FPGA board, we could control the direction of the actuator. Therefore, implementation of the function of these pins is best done in VHC due to its design simplicity and its minimal requirements for area and resources to implement the circuitry.

4.3.2.3.2 Accelerometer

The required communication protocol to control the accelerometer chip is Inter-Integrated Circuit, I2C, or Serial Peripheral Interface, SPI. Mode 3 of the 4-wire SPI protocol was opted to interface with the accelerometer due to its setup, as shown in Figure 4.8.

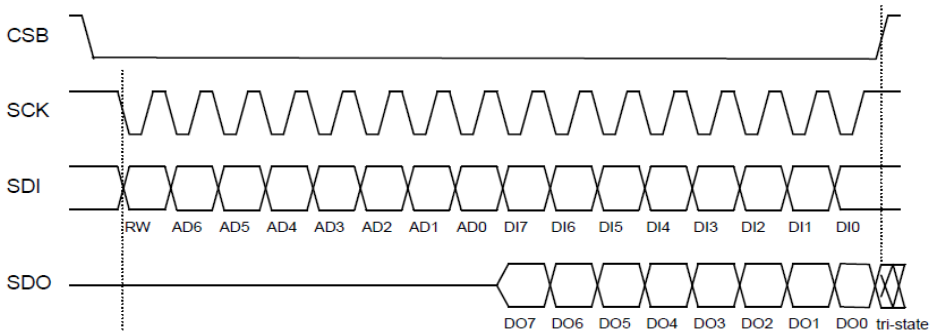

Figure 4. 8: Accelerometer SPI protocol and Timing Diagram Courtesy of [44]

Table 4.1 is used to demonstrate the required SPI protocol to read and write to the selected accelerometer chip in VHC.

Table 4. 1: Timing Analysis of the Accelerometer on FPGA

Time slots	SDO	SDI	SCK	CS	C.C.	time (ns)
T0			1	1	1	20
T1				0	5	100
T2		R/W	0		5	100
T3			1		5	100
T4		AD6	0		5	100
T5-T13		AD5-AD2	1		45	900
T14		AD1	0		5	100
T15			1		5	100
T16		AD0	0		5	100
T17			1		5	100
T18		DI7	0		5	100
T19	DO7		1		5	100
T20		DI6	0		5	100
T21	DO6		1		5	100
T22-29	DO5-DO2	DI5-DI2	0		40	800
T30		DI1	0		5	100
T31	DO1		1		5	100
T32		DI0	0		5	100
T33	DO0		1		2	40
T34				1	1	20
Total					**164**	**3280**

The total time to read/write one register of the accelerometer chip is 3280 ns or 164 clock cycles when the fed clock frequency to the component runs at 50MHz.

❖ Component Symbol

The observation mode component consists of control and data lines for the SPI protocol interface with the accelerometer and two general purpose output ports to control the actuator controller circuit. The top-level symbol of the observation mode component is shown in Figure 4.9.

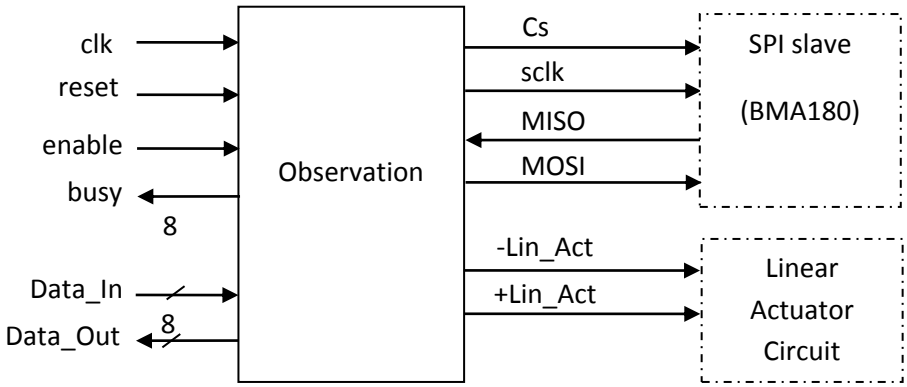

Figure 4. 9: Observation Mode Component Symbol

The component synthesizes and outputs the received clock, clk, to required clock rate by the accelerometer via "SCLK". The component receives data (memory address or byte) via "Data_in" and transmits the data to the BMA180, via SPI protocol. The obtained information from the accelerometer is received by the component through the "Data_out". The "–Lin_Act" and "+Lin_Act" ports each control a specific relay switch, which is used to set the polarity voltage input to the linear actuator.

4.3.3 Front-View Mode

Similar to the observation mode, the front-view mode uses the RF-link and an actuator to provide 3D images of N-angular position of the Head. The front view images are used by the 3D-image interpolation unit to subsequently help create a map of obstacles and their distance from the platform. The RF-link subcomponents are shared among other modes of operation and are used as described in section 4.3.1. The Head subcomponent of the front-view mode contains the linear actuator that is used to move the Head up and down, as shown in Figure 4.10.

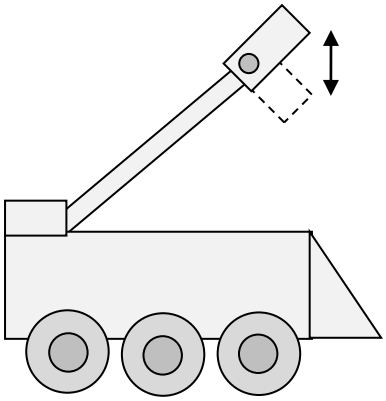

Figure 4. 10: Description of the Front-View Mode

4.3.3.1 Front-View Mode Hardware Architecture

The linear actuator used in the observation mode is also applicable in this mode. Therefore, based on the factors such as the Head weight, required angle of movement and speed a 4” stroke linear actuator capable of lifting full load of 40 lb, from Firgelli Technologies, was selected to meet the requirement [43]:

- Maximum speed: 4.45 cm/sec at no load and 2.54 cm/sec with load
- Stroke size : 4”
- thrust/pull force: 15lb
- holding force: 45lb
- Require voltage : 12v DC
- Maximum current: 5A
- Power consumption: 60 Watts

The linear actuator controller circuit discussed in Section 4.3.2.1 is also used to control the new linear actuator.

4.3.3.2 Principals of Front-View Mode operation

The front-view mode adjusts the Head subcomponent to the desired angle by means of a strategically placed linear actuator that connects the Tower and Head of the robotic platform. The 3D map of the scene is captured and process by the 3D-Image interpolation module and the result is sent to the control center, where the operator can choose a suitable path prior to sending the robot to a new location. The 3D-image interpolation module is currently being developed; the current prototype will be used as a plug-in module for the front-view mode of the system. The processor to achieve the above goal, as addressed in Chapter 3 of this paper is an FPGA. In the following section the implementation of functions associated with each module of the front-view mode will be discussed.

4.3.3.3 Front-View mode Firmware component

The firmware component of the front-view mode is designed to enable/disable the 3D image interpolation module and to control the direction of the linear actuator by toggling two GPIO pins to engage the appropriate relays.

- **Component Symbol**

The top-level symbol of the front-view mode component is shown in Figure 4.11.

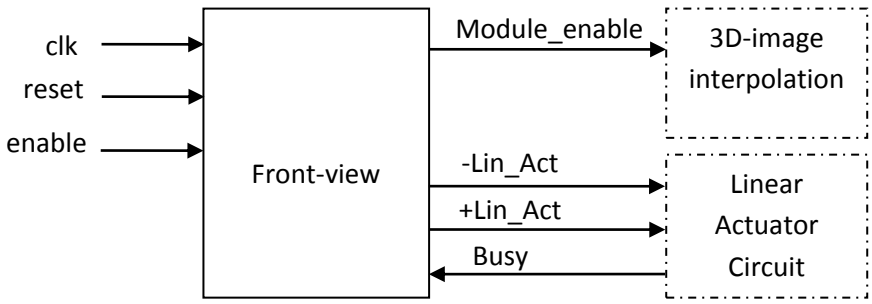

Figure 4. 11: Front-View Mode Component Symbol

As can be seen in the above figure, the “clk”, “reset” and “enable” input ports are fed to the component to drive them at the required clock rate and start and/or stop the operation of the mode. The output ports “-Lin_Act” and “+Lin_Act” control the direction of the actuator by setting the relays to the appropriate positions. Lastly, the “Module_enable” output port enables the 3D-image interpolation module and the “Busy” input port indicates if the previous operation on the linear actuator is done.

4.3.4Navigation Mode

The Navigator subcomponent of the Navigation mode consists of motors, motor controller, odometer and obstacle sensors to drive the robotic platform to the specified location without hitting potential obstacles, based on the 3D-map of the scene captured and processed in the front-view mode.

In the following subsections the steps taken to select the hardware components of the navigator subcomponent as well as the firmware and software components associated with each module is discussed to arrive at the navigation mode component of the system.

4.3.4.1 Navigation Mode Hardware Architecture

For simplicity of design implementation, the motors, motor controller and the required power supply of the motors are grouped as a single module, called “motor-driver” and the odometer and obstacle sensors modules are treated as two individual modules. The hardware architecture and associated components of each module are discussed in the following subsections.

4.3.4.1.1 Motor-driver

The motor driver module is composed of six brushed DC motors to provide mobility, a motor controller to control the speed and direction of the motors and a battery pack to supply them with power, as shown in Figure 4.12.

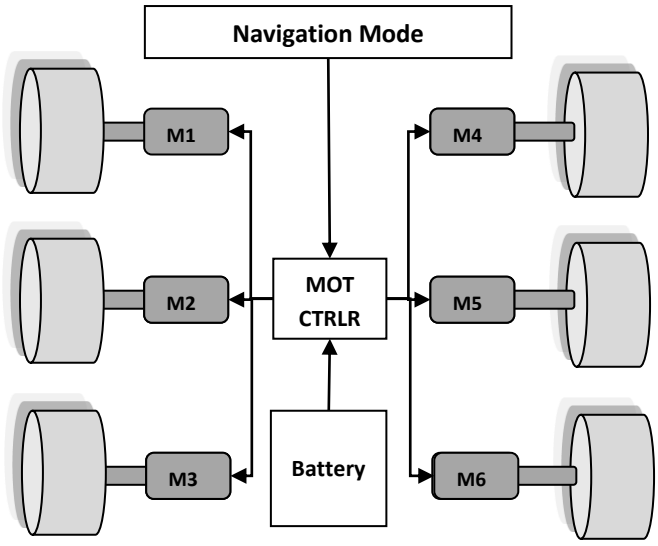

Figure 4. 12: Motor-driver Hardware Organization

As can be seen in the above figure the Motor-driver module consists of the following hardware elements and communication interface:

- Elements:
  - Motors
  - Motor controller
  - Battery pack
- Interface:
  - “Motor controller – Motors” interface
  - “Motor controller – battery” interface
  - “Navigation mode processor – motor controller” interface

The interface of the motors to the motor controller and motor controller to the battery pack occurs via traditional wire connections. However, the interface between the motor controller and the processor will be discussed in the following sections.

❖ Selection of elements

Selection of elements for the motor-driver module requires several considerations, including power consumption, performance and actual and development cost. Therefore, in the following subsections, the required elements for the motor-driver module are addressed.

➢ Motors

The 6-wheel robotic platform required six brushed DC motors to enable the heavy robot platform to move on any terrains and carry the camera module and possible quadra-copter modules. Choosing the right motor type to meet the above criteria requires further analysis of many factors, such as overall platform weight, terrain, torque, output power, efficiency of the motor, etc.; therefore, the following motor type with features below was selected [45]:

- Reduction Rate: 1:24
- Rated Speed: 5900 rpm
- No Load Speed: 7000 rpm
- Rated output: 34.7 W
- Motor Rated: < 2.3 A
- No Load Current: < 0.650 A
- Operating voltage: 24 v
- Efficiency: 70%

- **Motor Controller**

Choosing an interface to control the motors is the next step of the motor-driver hardware architecture of the system. This step could be done by designing a DC-Motor driver circuit or using an off-the-shelf motor controller. Selection of a motor controller requires consideration of motor specifications, such as the motor's nominal voltage, maximum current rating, control method and number of motors to control.

Based on the factors mentioned above and the system compatibility and cost-performance, Sabertooth Dual 25A Regenerative Motor Driver with the following features [46] was chosen:

- Dual motor drivers for two DC brushed motors with up to 25 A each
- Suitable for high-powered robots up to 300 lb
- Over-current and thermal protection circuit
- Multiple motor control protocols: Analog, radio control, serial and packetized serial.
- Various operating mode and application-specific mode

To control the speed and direction of the motors, the motor controller can interface with the motors and a processor using various types of communication protocols, including UART, SPI, PWM, analogue voltage, etc., which will be addressed shortly.

- **Batteries**

The Sabertooth controller requires a 24v battery supply to power the 2 pairs of the motors. Therefore, the last element of the motor-driver module is to select a power source. Since there are 6 motors to be controlled, the following calculations were made:

$$\boldsymbol{Power = 34.7 * 70\% = 26.40\ watts}$$

$$\boldsymbol{Current\ Draw = \frac{26.40}{24} = 1.10\ A}$$

Having six motors leads to 6.6 A of current consumption for the robot in active mode with minimal opposite force interferences. To run a system continuously over an hour we need a battery with over 6 Ah. NiCd batteries offer the shortest charge time, lowest overall cost and highest lifetime [47]; however, due to its toxic behavior and demanding maintenance requirement, it is not a good candidate for our system. Hence, a 10Ah-24-volt rechargeable NIMH battery pack with low maintenance and cost efficient features was chosen.

4.3.4.1.2 Odometer

The second module associated with the navigator subcomponent is the odometer or displacement measurement module, by which the distance is measured by counting how many times one wheel of the robotic platform has made a full turn. The odometer module (see Figure 4.13) consists of the following elements:

- Elements:
  - 2 Optical encoders
- Interface:
  - "Navigation Mode processor - optical encoders" interface

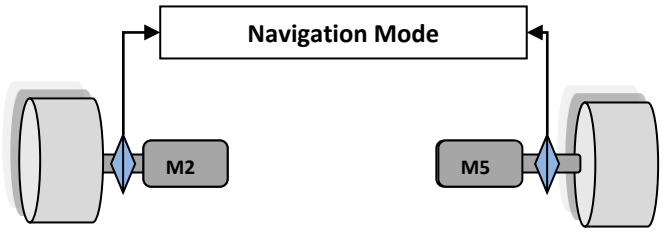

Figure 4. 13: Odometer Hardware Organization

❖ Selection of elements

To reach the above goal, which is counting the number of wheel rotations to find the travelled distance, an optical encoder circuit was designed by using TCRT5000L reflective optical sensor from Vishay [48]. The schematic of designed circuit is illustrated in Figure 4.14.

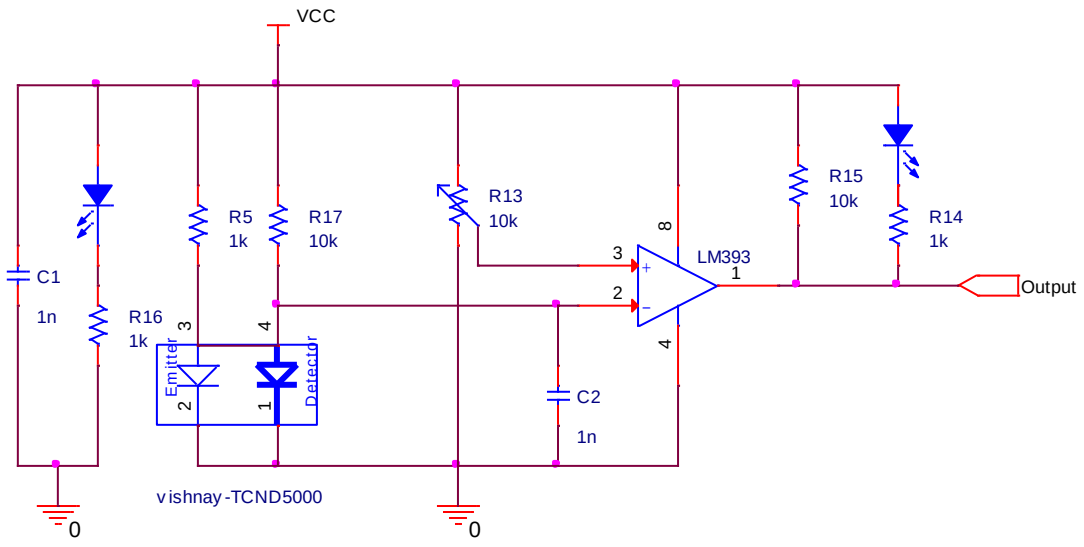

Figure 4. 14: Odometer Circuit Diagram

4.3.4.1.3 Obstacle Sensors

The third module associated with the Navigator subcomponent is the obstacle sensor system, by which the robotic platform avoids colliding with any objects that may be in the way. Similar to the odometer module, the obstacle sensors module (see Figure 4.15) consists of the following elements:

- Elements:
  - 3 Optical encoders
- Interface:
  - "Navigation Mode processor- optical encoders" interface

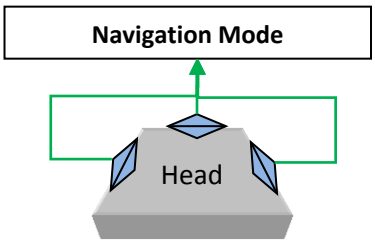

Figure 4. 15: Obstacle Sensors Hardware Organization

4.3.4.2 Principals of Navigation mode operation

Based on what has been discussed in the previous sections, the overall hardware organization of the navigation mode, including its major components, is presented in Figure 4.16.

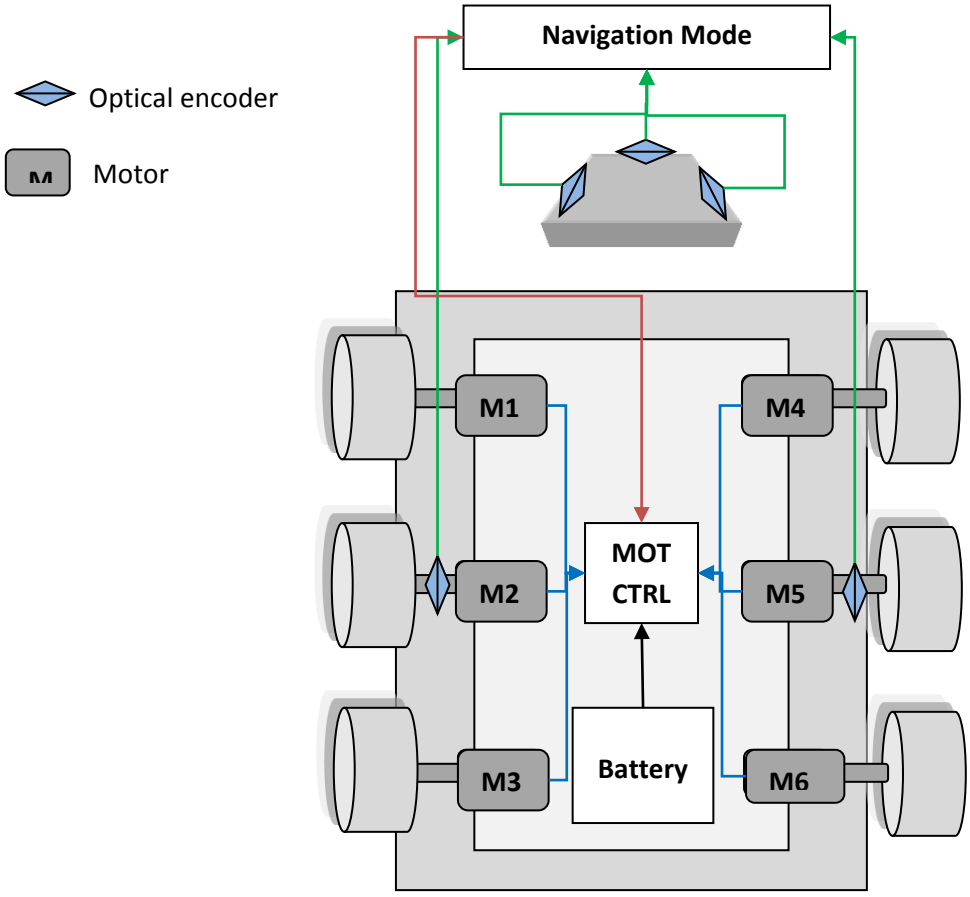

Figure 4. 16: Description of the Navigation Mode

As was mentioned earlier, the six motors attached to the wheels of the robots are divided into two pairs and are controlled by a Sabertooth motor controller. The selected motor controller accepts various communication controls, including UART, Analog, SPI, PWM, etc, by setting the switches to the appropriate positions. However, to keep the design as simple as possible, the normal mode of the motor controller, Analog mode, is selected by setting the switch to 4, 5 and 6 [46]. The analog mode of the motor controller takes two analog inputs in the range of 0-5v and uses them to set the speed and direction of the motors. Signals below and above 2.5v are reserved for reverse and forward motion, respectively, while 2.5 will stop the motion. To provide the motor controller with the above Analog signals, a Digital-to-Analog Converter, DAC, circuit is required.

Furthermore, the odometer module of the navigator subcomponent is responsible for measuring the distance travelled by the robot by counting the number of wheel rotations. The designed circuit exploits the process of triangulation, by which a beam of infrared light is emitted on a surface via the emitter and then the emitted light is reflected and received by the receiver sensor. The amount of reflected beam is used to measure the distance, surface type, obstacle avoidance, velocity, etc. In other words, for every pulse sent out by the encoder, as it is placed across the shaft of the motor while fully isolated from the ambient light, the wheel has travelled a certain angle. Hence, by knowing the wheel diameter and the encoder resolution, we could count the number of wheel rotations and eventually calculate the travelled distance.

A bi-color tape, five black lines on a white surface, is placed across the shaft at a few centimeters of the encoder to meet the TCRT5000L chip requirement of maximum operating distance. The output is fed to the processor in order to count the black and white lines. Counting four black lines is equivalent to a full turn of the wheel or 50 cm on the surface. The designed circuit of this module requires a voltage source of 3.3v to activate, and it will output a certain voltage level based on the encoder position. Hence, to translate the output of the circuit, which is in analog form, an Analog-to-digital Converter, ADC, is required.

Last but not the least, the obstacle sensor module also exploits the optical encoders used in odometer module. The optical circuit is designed in such a way as to transmit a digital output, while the range of detection can be adjusted by the provided potentiometer. Hence, a simple General Purpose Input (GPI) port can detect an obstacle within the adjusted range by simply reading the input voltage.

To implement the requirements of the navigation mode, further hardware components are required. Based on what was discussed in Chapter 3, the implementation of the design on FPGA was found to be a better option compared to the traditional microprocessors, due to the existence of reconfigurable logic block. Therefore, choosing an FPGA device that can accommodate the design architecture and the required components of each mode of operation is the most optimal and logical choice. As a result, a simple Spartan 3E starter board from Xilinx with following features was chosen [49]:

➢ SPARTAN-3E Starter Kit Board's key features and components

- **Xilinx devices on board**
  - Spartan-3E FPGA (XC3S500E-4FG320C)
    - Up to 232 user-I/O pins
    - 320-pin FBGA package
    - Over 10,000 logic cells
  - CoolRunner™-II CPLD (XC2C64A-5VQ44C)
  - Platform Flash (XCF04S-VO20C)
- **Clocks:**
  - 50 MHz crystal clock oscillator
- **Memory:**
  - 128 Mbit Parallel Flash,
  - 16 Mbit SPI Flash
  - 64 MByte DDR SDRAM
- **Connectors and Interfaces:**
  - JTAG USB downloader
  - Two 9-pin RS-232 serial port
  - Four slide switches
  - Eight individual LED outputs
  - 100-Pin expansion connection ports
  - Three 6-pin expansion connectors
  - Two-input, SPI-based Analog-to-Digital Converter (ADC) with programmable-gain pre-amplifier
  - Four-output, SPI-based Digital-to-Analog Converter (DAC)

4.3.4.3 Navigation Mode Firmware Component

The navigation mode component of the system is designed to interface the three individual modules addressed above. In the following sections the required firmware component of each module to form the navigation mode component is described.

4.3.4.3.1 Motor-driver

The Spartan-3E board includes an SPI-compatible, four-channel, serial DAC, LTC2624 from Linear Technology with 12-bit unsigned resolution [50]. The SPI bus used for the DAC chip is a full-duplex, synchronous, character-oriented channel employing a simple four-wire interface [49]. The FPGA drives the required bus clock, SPI_SCK, and transmits serial data, SPI_MOSI, to the LTC 2624 DAC. The DAC chip responds back to the FPGA via the SPI_MISO.

The LTC2624 DAC chip can run at the high frequency of 50 MHz [50] and can support both 24-bit and 32-bit protocol that consists of a command, an address and data as shown in Figure 4.17. To control the two pairs of motors, we would need to use two channels of DAC.

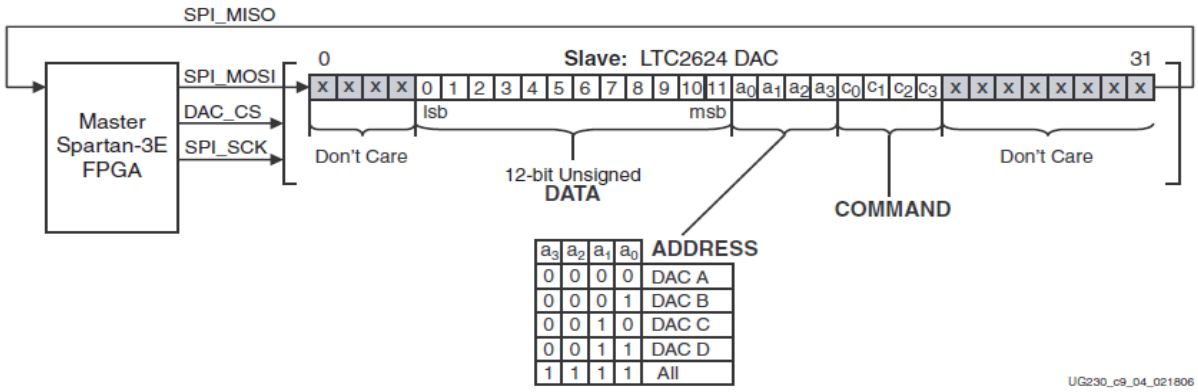

Figure 4. 17: The SPI Protocol of DAC courtesy of [49]

Each DAC output level is analog equivalent of a 12-bit unsigned digital value, Data [11:0], written to the DAC chip via SPI interface. To setup the DAC output voltage the equation below shall be used:

$$\boldsymbol{Vout} = \frac{\boldsymbol{Data}[\mathbf{11:0}]}{\mathbf{4096}} \times \boldsymbol{Vref}$$

$$\boldsymbol{Vref} = \mathbf{3.3}\,\boldsymbol{V} \pm \mathbf{5\%} \qquad for \qquad CHA\ and\ CHB$$

$$\boldsymbol{Vref} = \mathbf{2.5}\,\boldsymbol{V} \pm \mathbf{5\%} \qquad for \qquad CHC\ and\ CHD$$

Therefore, the next logical step is to implement the required SPI communication to interface with the controller and eventually control the speed and direction of the motors. This interface may be implemented in the form of hardware logical blocks of the FPGA in VHDL or as a Virtual soft-core system.

In case of implementing the SPI interface as logic blocks, the latency to output the DAC values on the two channels is 136 clock cycles. However, the consequent DAC value can be processed in 132 clock cycles, as illustrated in Table 4.2.

Table 4. 2: Timing Analysis of the Design on FPGA

Time slots	State	MISO	MOSI	SCK	CS	i	C.C.	Ch #
T0	Init		0	0	1	0	4	A
T1	Begin		DataA(0)	0	0		1	
T2	Go			1		1	1	
T3	Begin		DataA(1)	0			1	
T4	Go			1		2	1	
T5	Begin		DataA(2)	0			1	
T6	Go			1		3	1	
T7	Begin		DataA(3)	0			1	
T8	Go			1		4	1	
T9	Begin		DataA(4)	0			1	
T10	Go			1		5	1	
T11	Begin		DataA(5)	0			1	
T12	Go			1		6	1	
T13 - T56	Begin/Go			0/1		7-28	44	
T57	Begin		DataA(28)	0		28	1	
T58	Go			1		29	1	
T59	Begin		DataA(29)	0			1	
T60				1		30	1	
T61			DataA(30)	0			1	
T62				1		31	1	
T63			DataA(31)	0			1	
T64	Go			1		0	1	
T65	Final			0	1		1	
T66	Done						1	B
T67	Begin		DataB(0)	0	0		1	
T68	Go			1		1	1	
T69	Begin		DataB(1)	0			1	
T72 –T128	Begin/Go		Datab(i)	0/1		2-30	58	
T129	Begin		DataB(30)	0		30	1	
T130	Go			1		31	1	
T131	Begin		DataB(31)	0			1	
T132	Go			1		0	1	
T133	Final			0	1		1	
Total Time = 136 c.c.								

Performing the same the DAC operation on a 32-bit RISC processers based on the flowchart shown in Figure 4.18 leads to the results recorded in Table 4.3.

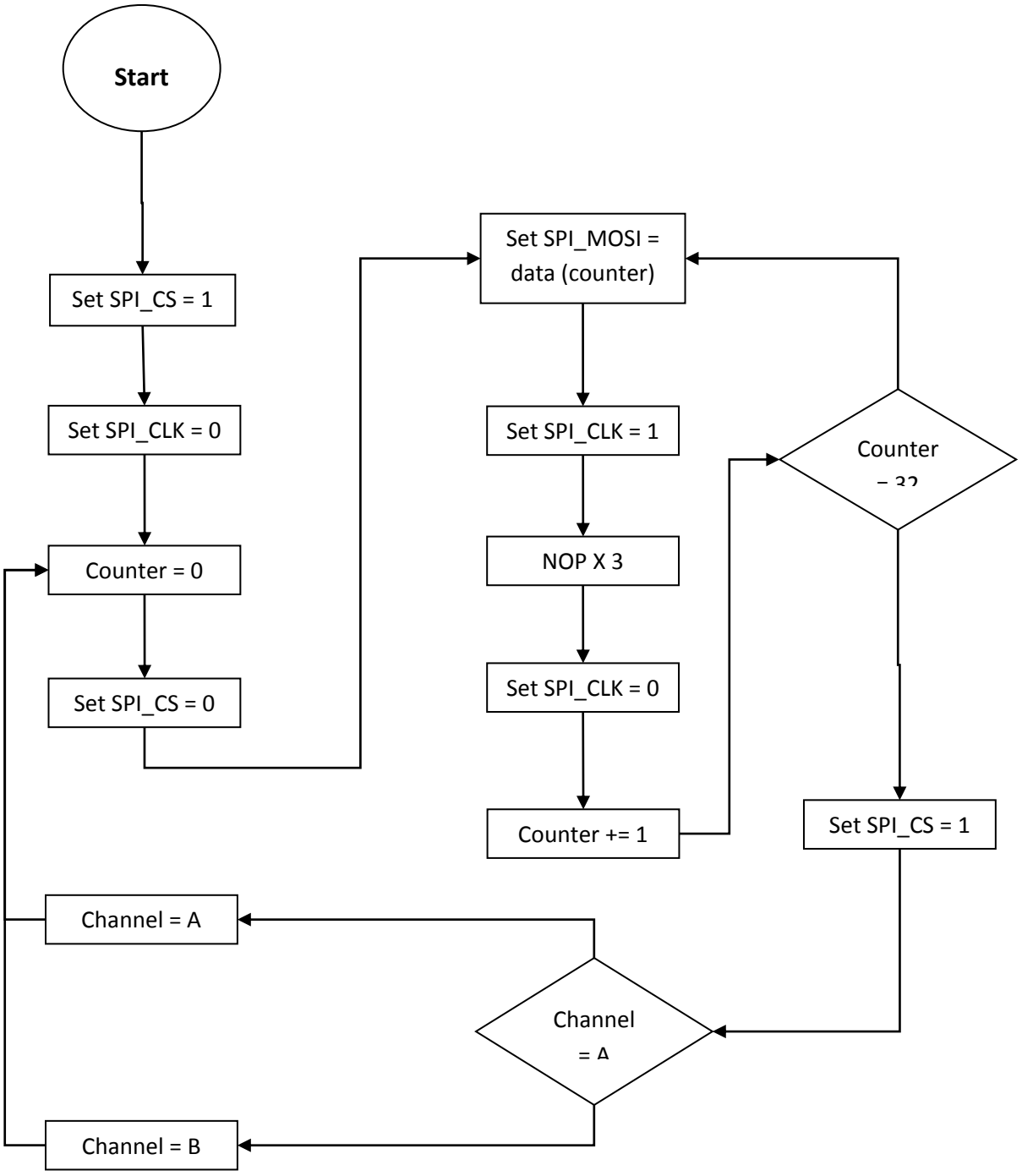

Figure 4. 18: RISC Processor Instruction Execution Process

Table 4. 3: Timing Analysis of the Design on a RISC Processor

Time Slots	Type	MISO	Cs	MOSI	SCK	I	C.C.	Pipelined	Ch #
T0	Load/store		1	0	0	0	12	6	
T1	Load/store		0				3	1	
T2	Load/store			DataA(0)			3	1	
T3	Load/store				1		3	1	
T4	Load/store				0		3	1	
T5	Add					1	3	1	
T6 –T155	Instruction			DataA(i)	1/0	1-31	450	151	**A**
T156	Load/store			DataA(31)			3	1	
T157	Load/store				1		3	1	
T158	Load/store				0		3	1	
T159	Load/store					0	3	1	
T160	Branch						3	1	
T161	Load/store		1				3	1	
T162	Load/store		0				3	1	
T163	Load/store			DataB(0)			3	1	
T164	Load/store				1		3	1	
T165	Load/store				0		3	1	
T166	Add					1	3	1	
T167-T316	Instruction			DataB(i)	1/0	4-30	450	151	**B**
T317	Load/store			DataB(31)			3	1	
T318	Load/store				1		3	1	
T319	Load/store				0		3	1	
T320	Load/store					0	3	1	
T321	Branch						3	1	
T322	Load/store		1				3	3	
Total Time							978	332	

As can be seen from Table 4.3 in an ideal condition, in which no instruction or data hazards are introduced, the total number of required clock cycles to output a DAC value, also known as the latency, is 978. However, due to the parallel pipelining of the embedded soft-core processors such as MicroBlaze, three instructions can be executed simultaneously, as shown in Figure 4.19. Therefore, the cycle time for the subsequent channel is 332 cc, which is almost 2.5 times greater than VHC implementation design.

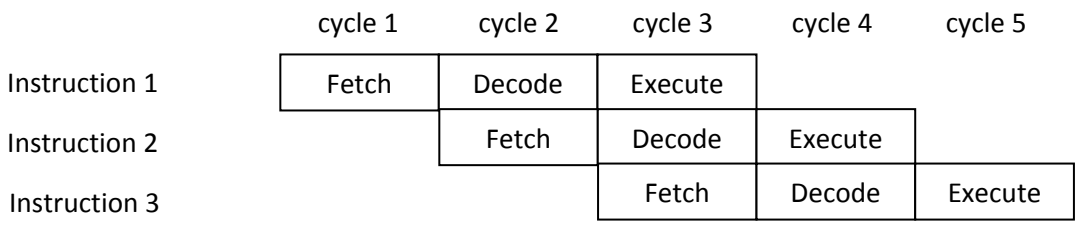

Figure 4. 19: MicroBlaze Execution Process

Hence, in case of implementing the module in VHC, the total execution time is:

$$\boldsymbol{Texec_{FPGA} = [L + (n - 1) * Ctime] * Tcycle = [136 + (10 - 1) * 132] * 20 = 3.9\ \mu s}$$

Where L is the number of cycles required for obtaining the first result, n is the total number of the required cycles (we assume 10 DAC values for simplicity of calculation), Ctime is a cycle time and Tcycle is a clock period at 50MHz. Additionally, the total execution time required for processing of a single DAC channel on MicroBlaze based design is:

$$\boldsymbol{Texec_{FPGA\_MicroBlaze} = [332 + 9 * 330] * 20 = 66.04\ \mu s}$$

The performance speedup of the two solutions can be tested by:

$$\boldsymbol{Speedup = \frac{TexeMicroBlaze}{TexeFPGA} = \frac{66.04\ \mu s}{3.9\ \mu s} \approx 17\ times}$$

In order to match the performance of the hardware implementation, the soft-core processor needs to run at a faster clock rate of:

$$\boldsymbol{F_{MicroBlaze} = \frac{1}{\frac{Texec_{FPGA}}{[L + (n - 1) * Ctime]}} = \frac{1}{\frac{3.9\ usec}{[3302\ ]}} \approx 846\ MHz}$$

This speed is not possible as MicroBlaze is running at its highest frequency of 50 MHz. Moreover, in case of using microcontroller, additional algorithms for increasing the frequency lead to greater power consumption, which would require extra specialized cooling system to eliminate the excessive and potentially damaging heat from the device.

Based on the above analysis, even though implementation of the SPI interface in hardware and soft-core processor does not significantly affect the overall design performance, design simplicity and the area analysis shows that the implementation in the form of VHC is a better option, as suggested in Chapter 3.

❖ Component Symbol

The component symbol of the motor-driver module of the navigation mode consists of the necessary signal ports to provide the SPI interface signals such as: SPI clock (“SCK”), SPI chip select (“CS”), SPI MISO (“MISO”), SPI MOSI (“MOS”I) with the DAC chip. The clock input port (“clk”) is used to drive the components at 50 MHz, which will be synthesized and divided to generate the required clock rate by the DAC chip. The reset input port (reset) is used to reset the operation of the module. The digital

data input ports (“Data_A” and “Data_B”) provide the required digital data to be converted to the analog form to be fed to the motor controller. The “DAC_CLR” and “DAC_Enable” ports are used to control the DAC chip, while the busy input port indicates if a DAC conversion is in process. The motor-driver component is shown in Figure 4.20.

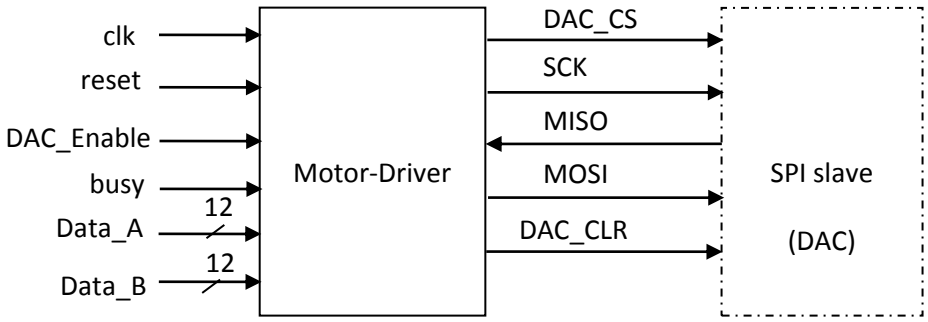

Figure 4. 20: Motor-driver Component Symbol

4.3.4.3.2 Odometer

The Spartan-3E board comes with a two-channel analog capture circuit, consisting of a programmable scaling pre-amplifier, LTC6912-1, and ADC, LTC1407A-1, which can be controlled serially by the FPGA [49]. The analog capture circuit converts the voltage on VINA or VINB and converts it to a 14-bit digital representation, Data [13:0] as follows:

$$\boldsymbol{Data[13:0] = Gain} \times \frac{\boldsymbol{Vin - 1.65}}{\boldsymbol{1.25v}} \times \boldsymbol{8192}$$

Where the gain, which controls the allowable voltage range for each ADC channels, VINA, VINB, is the setting loaded into the programmable pre-amplifier. Similar to the motor-driver module, the SPI communication is required to interface signals between the FPGA and the amplifier and ADC. Hence the following pins of the FPGA are used for the SPI. Therefore, following the same analysis performed in motor-driver, it is clear that implementing the odometer module as a hardware component would give better performance and area. On the other hand, the timing constraint of the amplifier and ADC channel makes MicroBlaze processor a good candidate. Nevertheless, due to implementation of the SPI controller in the motor-driver which can be shared by the odometer, implementing the design in hardware would be a better choice.

❖ Component Symbol

The component symbol of the odometer module of the navigation mode consists of the necessary signal ports to provide the SPI interface signals, such as: SPI clock (“SCK”), SPI chip select (“CS”), SPI MISO (“MISO”), SPI MOSI (“MOSI”) with the ADC chip. The clock input port (“clk”) is used to drive the components at 50 MHz, which will be synthesized and divided to generate the required sampling rate of the ADC chip, which is 1.5 MHz. The reset input port (“reset”) is used to reset the operation of the module [51]. The digital data output ports (“Data_1” and “Data_2”) present the captured digital data. The “ADC_Enable” Input port is used to control the ADC chip, while the busy input port indicates if a DAC conversion is in process. The motor-driver component is shown in the figure below. Since, the SPI communication is shared among multiple components on the FPGA board, the output and input ports such as: “AD_CONV”, “AMP_SHDN”, “ADMP_DOUT”, “SF_CF0”, “FPGA_INIT_B”, “SPI_SS_B”, “AMP_CS”, “DAC_CS” need to be set up to control the ADC chip [49]. The odometer component is shown in Figure 4.21.

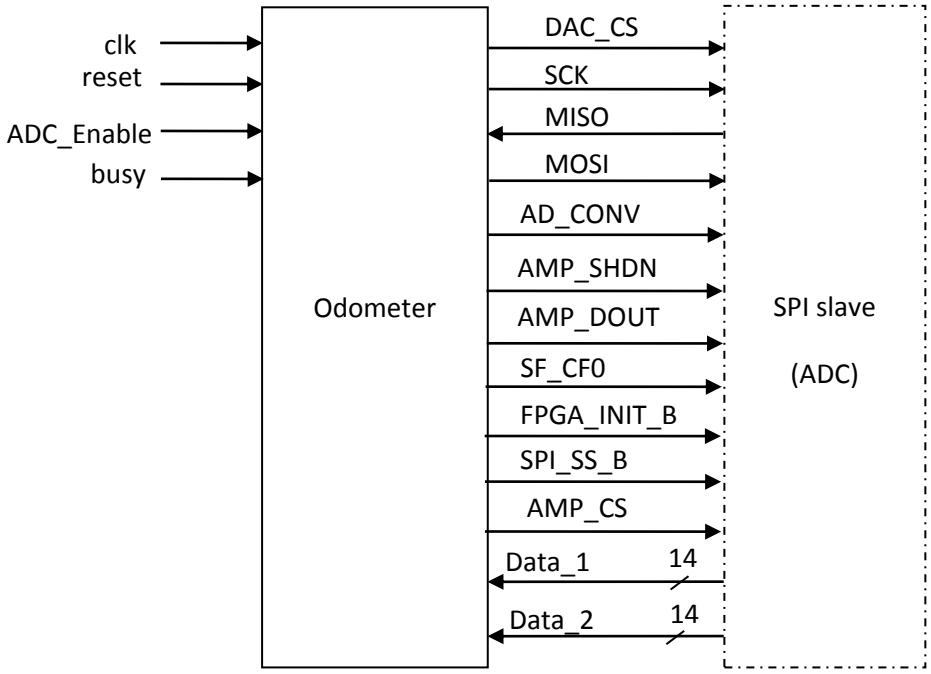

Figure 4. 21: Odometer Component Symbol

4.3.4.3.3 Obstacle Sensors

The last module in the navigator subcomponent of the navigation mode is the obstacle sensors module. The optical encoder circuit addressed in section 4.3.4.1.2 can also be applied here. The circuit is designed in such a way as to be able to output digital data when an object is placed in front of the sensor. The range of detection can be adjusted by varying the provided potentiometer. Therefore, a simple General Purpose Input port is able to detect possible objects.

- **Component Symbol**

The component symbol of the obstacle sensors module of the Navigation mode consists of three general input ports to detect any high voltage caused by approaching an object. A clock, reset and enable input ports to provide a sampling clock and to stop or start the module operation. The component symbol of this module is shown in Figure 4.22.

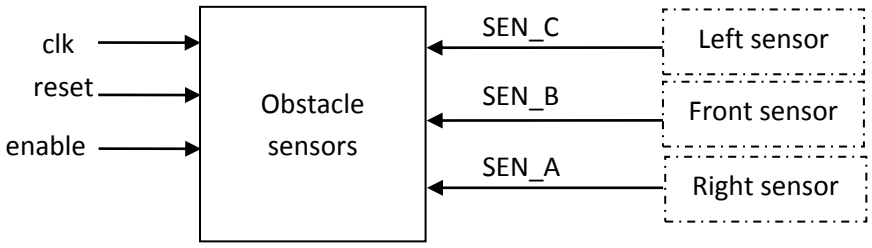

Figure 4. 22: Obstacle Sensors Component Symbol

The component symbol of the navigator mode is constructed based on the subcomponents discussed in the previous sections as shown in Figure 4.23. The subsequent components synchronously receive the clock signals on their input in accordance to their required clock rates. The clock source of the navigation mode is also sourced from the main component that accommodates the components of all the other modes of operation.

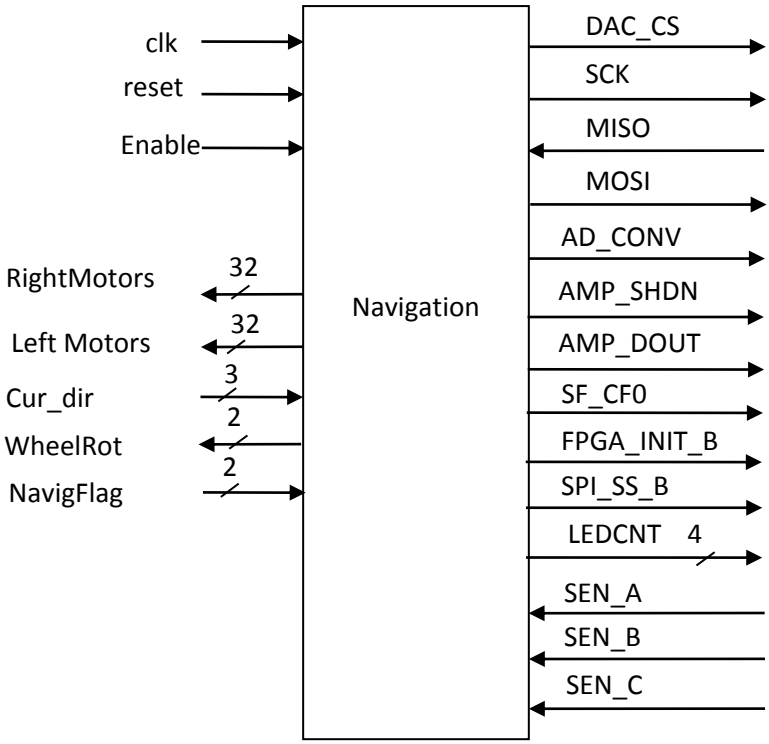

Figure 4. 23: Navigation Mode Component Symbol

As can be seen in the above figure, the “clk”, “reset” and “enable” input ports are to drive the 50MHz clock to the component and start/stop the navigation mode component. The SPI protocol interface ports along with other shared input/output ports addressed in the previous section are to be used with the DAC and ADC chips. The three input ports, “SEN_A”, “SEN_B” and “SEN_C” are the object detectors placed around the Head of the robot. The direction of the motors (“Cur_dir”), the distance each pair of wheels needs to travel (“RightMotors”, “LeftMotors”) to get to the destination is also addressed in the above figure. The “NavigFlag” also presents the “enable” input port for the navigator subcomponent, while the “WheelRot” indicates if any pair of the motors has finished the required number of turns. Finally, the “LEDCNT” ports are used for testing and debugging purposes.

4.4 Complete System Implementation

Discussing the hardware, firmware and software components of each mode of operations made it possible to address the need to have a primary component that can accommodate all the other major components of the modes of operation. The new component needs to be able to act as a dispatcher and direct the present operations to the respective mode. The main system component symbol consists of 2-bit output port to enable one of the modes of operation in a multiplexing scheme, as shown in Figure 4.24. The Required SPI interface ports to interfaces with the motor controller, accelerometer and optical encoder are also present in the component symbol. The LED ports and input port ("Debug_SW") are used for testing and debugging purposes.

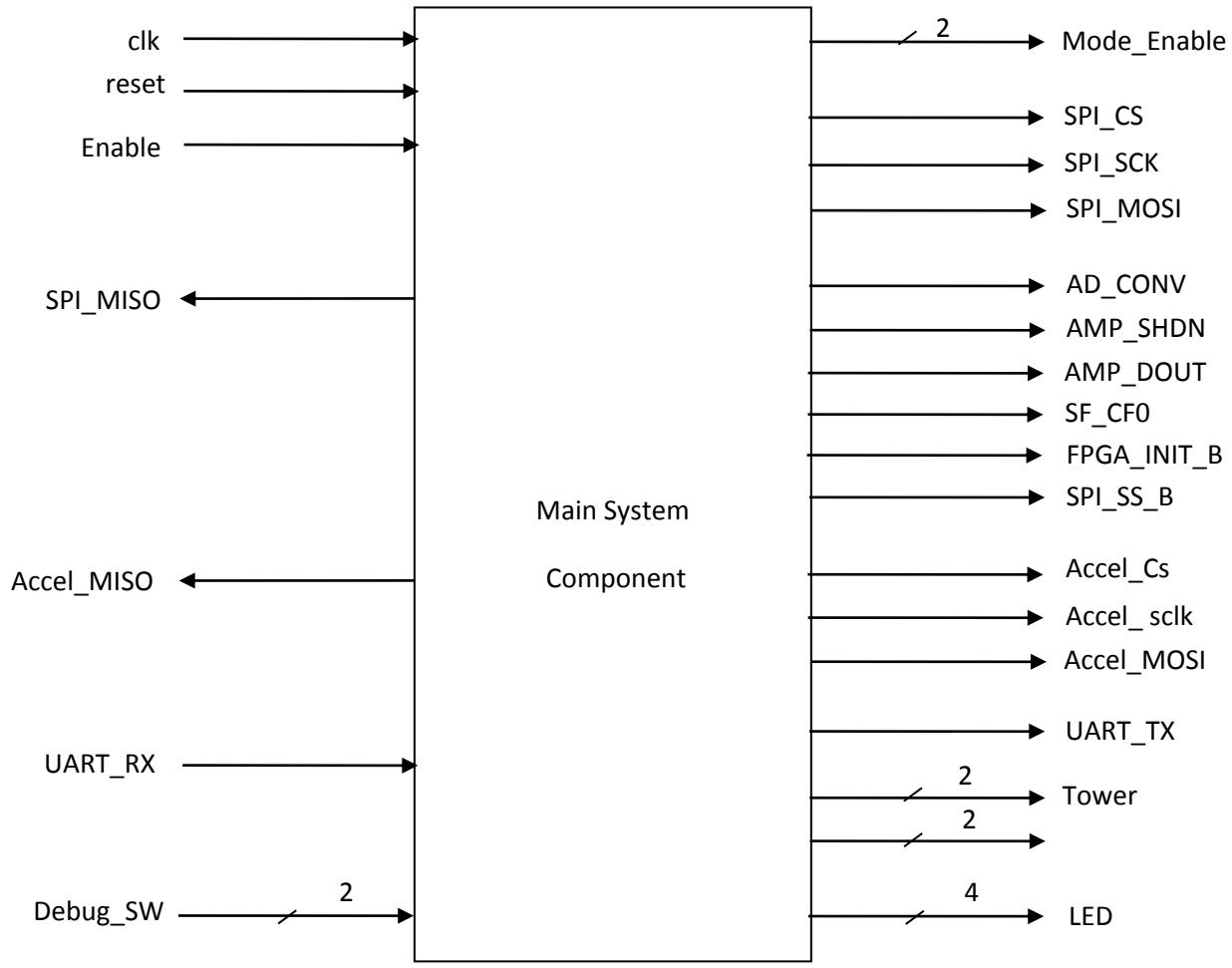

Figure 4. 24: Main System Component Symbol

4.5 Summary

In this chapter, the implementation specifics of the proposed reconfigurable robotic platform was discussed, and the process of selecting elements for each mode of operation based on the required resources discussed in the previous chapter was presented. The implementation of the proposed system was divided into two major parts: 1) the electro-mechanical part and 2) control and communication part and the associated hardware. Firmware and software components of each mode of operation were designed and developed to arrive at the complete system implementation of the proposed robotic platform.

5. Experimental Results Analysis and Discussion

5.1 Introduction

This chapter is dedicated to the analysis of the experimental results of the implemented design, discussed in chapter 4. The main objective of the conducted experiments was to collect and analyze the performance parameter data of the system, such as timing characteristics, power consumption, occupied area, and used resources to investigate the behavior of the proposed design.

This chapter is concluded by introducing an alternative approach in the design implementation that conforms to the concept of reconfigurability discussed in chapter 3, while extending the scalability of the system despite the potential limitation of memory and resources of any FPGAs.

5.2 Experimental Setup

In order to prepare a test and verification environment, the proposed reconfigurable robotic system was incorporated with the MARS, 3D-P camera and 4-Vision subsystems as designed and implemented by [1] to collect data and investigate the performance and reliability of the system within the actual telepresence system. However, only the control and communication aspects of the robotic subsystem of the telepresence system are discussed in this paper. Figure 5.1 illustrates the major hardware components of the system used during test and verification process.

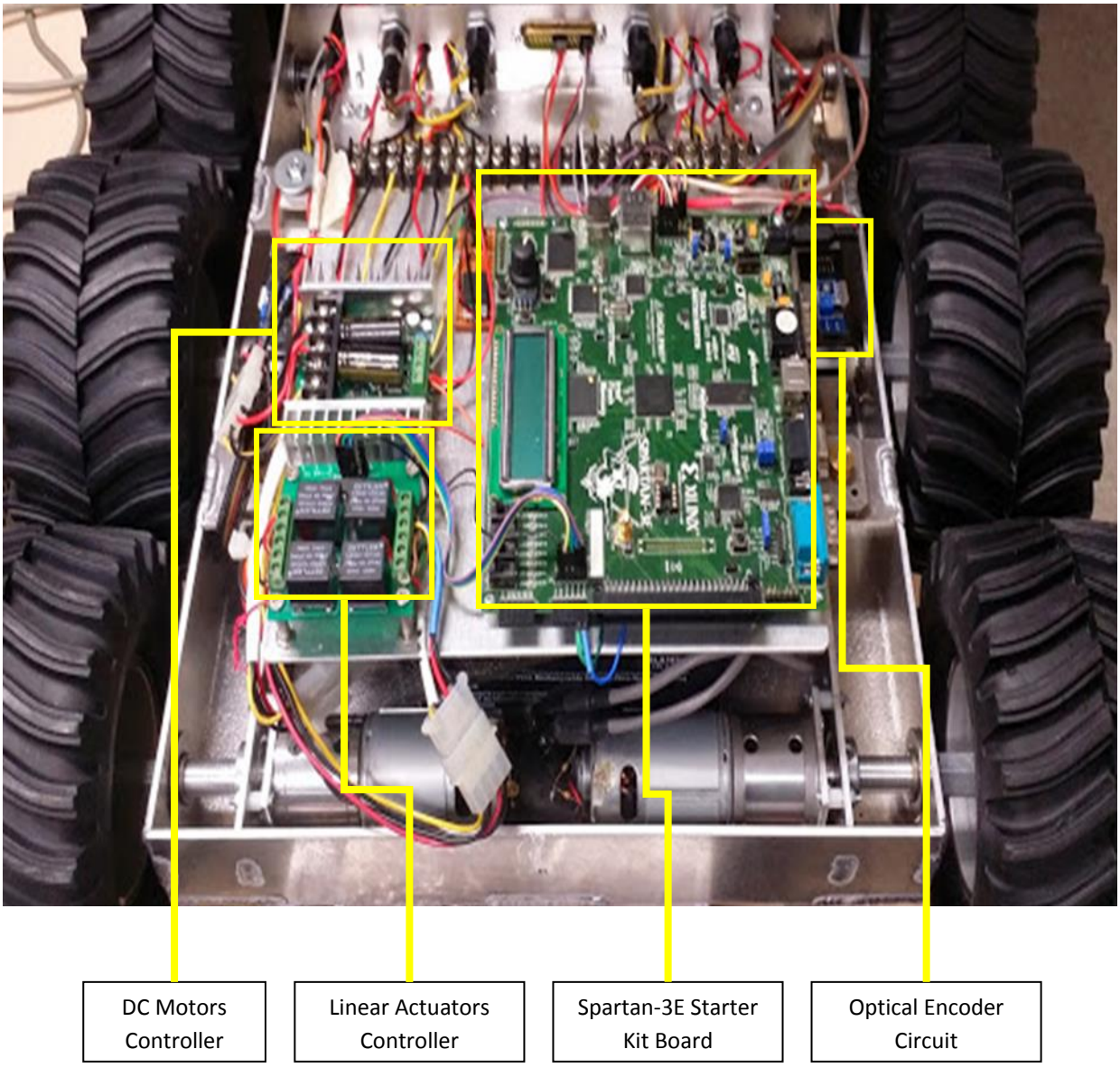

Figure 5. 1: Robotic Platform inside View

The operational hardware components shown in the figure above are isolated from the electro-mechanical hardware components such as motors, which along with the batteries, are placed at the bottom of the platform to reduce noise and potential interferences.

The control and communication process of the system incorporates the following hardware component as described in Section 4.3-4.5.

1. A Spartan-3E Starter Kit board that consists of Xilinx XC3S500E Spartan-3E FPGA [49]. This element is the main component of the system as it starts, maintains and manages its entire communication and control process. The board includes the required controller chip for ADC and DAC for the odometer and DC-Motor controller components; the RS232 and SPI interface for the XBee module and accelerometer components; general I/O port interfaces for the linear actuators and obstacle sensors interfaces. The board also exploits the USB interface based on FTDI's FT2232HL USB controller to store, load and program the FPGA and on-board Flash PROMs that will be discussed shortly. The maximum clock frequency of the board is 50MHz, which is distributed within the system according to the required clock rate of involved components.

2. A DC-Motor controller to control the speed and direction of the six DC-brushed motors, as described in Section 4.3 of the navigation mode of operation.

3. An XBee module to communicate with the control center through RF-link. The module is located at the rear end of the robotic platform to avoid any potential interference by other hardware components of the system.

4. Two designed optical encoder circuits across the motor shafts of the robot's middle wheels to measure the distance travelled by counting the number of wheel rotations.

5. Three optical sensors placed at the three outer corners of the Head as part of the obstacle sensors component of the navigation mode, as described in Section 4.3.

6. An Accelerometer placed in the Head compartment to measure the level and angle of elevation during the observation mode.

7. A Linear Actuators controller to manage the direction of the two linear actuators of the Head and Tower compartments of the system.

By knowing the hardware components involved in the test and verification process, it is time to list the system configuration/verification tools used during the experimental setup process. Therefore, we took advantage of the following elements to pursue our analysis on the system's performance parameters.

Programmers:

- Xilinx Platform USB cable
- XBee USB interface board

Software:

- Xilinx ISE 14.2 [52]
- Xilinx Platform Studio 14.2 (XPS) [53]
- Xilinx Software Development Kit 14.2 (SDK) [54]
- Xilinx ChipScope Pro 14.2 [55]
- Comm. Port Navigation Application

Equipment:

- HP 54620C Logic Analyzer
- BK Precision Digital Multi Meter and Power supply

The "Xilinx Platform USB programmer" listed above was used along with the corresponding software to configure and verify the implemented design on the Spartan-3E board. However, once the design is final, the FPGA is configured via the on-board serial flash PROM. Therefore, the system does not require configuration on the subsequent power ON/OFF cycles via the programmer.

The configuration of the selected FPGA device was performed using the programmer listed, attached to the USB port of a PC running the iMPACT application. The verification of this hardware component of the device was done through Xilinx ChipScope 14.2, and the software component was tested and debugged via the Xilinx XPS and SDK.

The "Comm. Port Navigation Application" was developed to transmit commands to the robotic platform via the RF-Link. The application is able to send multiple encoded commands and receive acknowledgements from the system via XBee USB interface board, which connects the XBee module with a USB port of a PC.

In the following sections the analysis of the system's performance parameters including timing characteristics, power consumption, occupied area and used resources are presented.

5.3 Timing Analysis

The first step before investigating the system's timing characteristics is to determine the required configuration time of the FPGA on power up or the estimated startup time of the system. The startup time is directly related to the method of loading the design on the selected FPGA. The FPGA can be configured from four available options provided by the Spartan-3E board including:

- Direct configuration of the FPGA with the design via JTAG, using the on-board USB interface.
- Programming the on-board 4Mbit Xilinx XCF04S serial Platform Flash PROM and configuring the FPGA from the image stored in Platform Flash PROM using master serial mode [49].
- Programming the on-board 16Mbit ST Microelectronics SPI serial Flash PROM and configuring the FPGA from the image stored in SPI serial Flash PROM using SPI mode [49].
- Programming the on-board 128Mbit Intel StrataFlash parallel NOR Flash PROM and configuring the stored image using BPI Up or BPI Down configuration modes [49]. This configuration option can be used for applications that require dynamic reconfiguration as two configuration bit streams can be loaded on the Flash PROM.

Based on the options described above and the size of the implemented design, option one was chosen as the most convenient choice for configuring the FPGA during the test and verification of the design, and option 2 was opted to configure the FPGA with the final design. [56] suggests that the required time to load the configuration bit stream, $T_{BITLOAD}$ is a function of device family, density, clock frequency and configuration data port width as shown in equation below:

$$T_{BITLOAD} = \frac{Bitstream\ length\ (bits)}{Clock\ freq.\ (MHz) * Configuration\ Port\ Width\ (bits)}$$

Therefore, with an approximate configuration bit stream file of 278KB, including the hardware and software components of the design and a default 1.5MHz configuration clock, CCK, to be used during loading from an external PROM, the required configuration time is:

$$T_{BITLOAD}\ \frac{2.3Mbits}{1.5MHz*1\ bit} = 1.61\ s$$

However, configuring the FPGA through the Xilinx XCF04S Platform Flash allows us to increase the CCLK frequency up to 25 MHz and hence, reduce the configuration time as shown below [57]:

$$T_{BITLOAD}\ \frac{2.3Mbits}{25MHz*1\ bit} = 96.4\ ms$$

Therefore, the system can start up approximately 95% faster.

To analyze the actual timing characteristics of the design a set of experiments was performed on the proposed system to determine the timing associated with each mode of operation and the required time to switch from one mode to another using the existing multiplexing approach. The timing characteristics were obtained and analyzed by using HP 54620C Logic Analyzer and ChipScope Logic Analyzer from Xilinx Software Kit.

The first test was performed on the transition of the Idle-to-Navigation mode of operation and vice-versa and the actual timing recorded by Xilinx ChipScope was measured as shown in Figure 5.2.

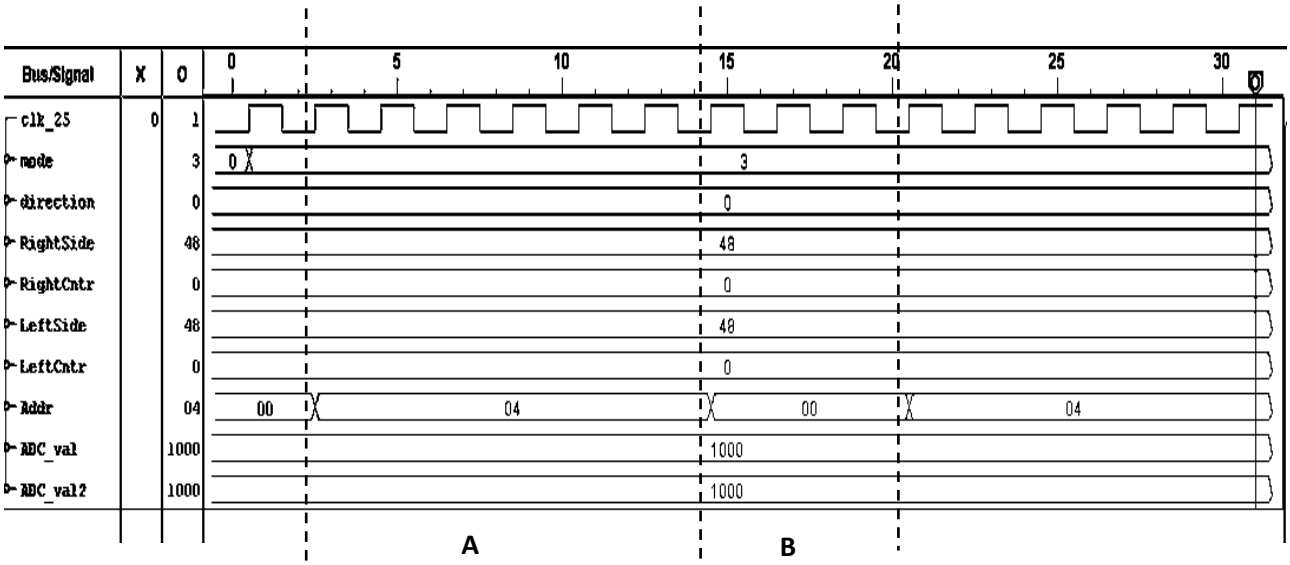

Figure 5. 2: Idle-Navigation Mode Transition

As mentioned before, the shared dual-BRAM is used by the component symbol to trigger and enable the appropriate mode of operation. Therefore, as can be seen in Figure 5.2, the “Addr” indicates which mode of operation is active at the moment. Furthermore, the transition from idle to Navigation mode of operation represented as “mode” signal takes 6 clock cycles or 100ns as “clk_25” signal represents half of the actual clock rate of the system as shown at point “B”. It should be noted that, since the “direction” signal is set to “stop” the mode of operation will transfer back to the idle mode, which takes 12 clock cycles as shown at point “A”. The required time can be mainly related to the reading and/or writing form and/to shared storage memory.

Further analysis is also made on the amount of time a typical task takes in navigation mode of operation. To achieve the above goal, an experiment was setup to investigate the required time to move the robotic platform forward for 6.5 meters, the wheels which have a diameter of 18cm, need to turn 12 times, assuming no-load and no-friction conditions. The results shown in Figures 5.3 and 5.4 were observed.

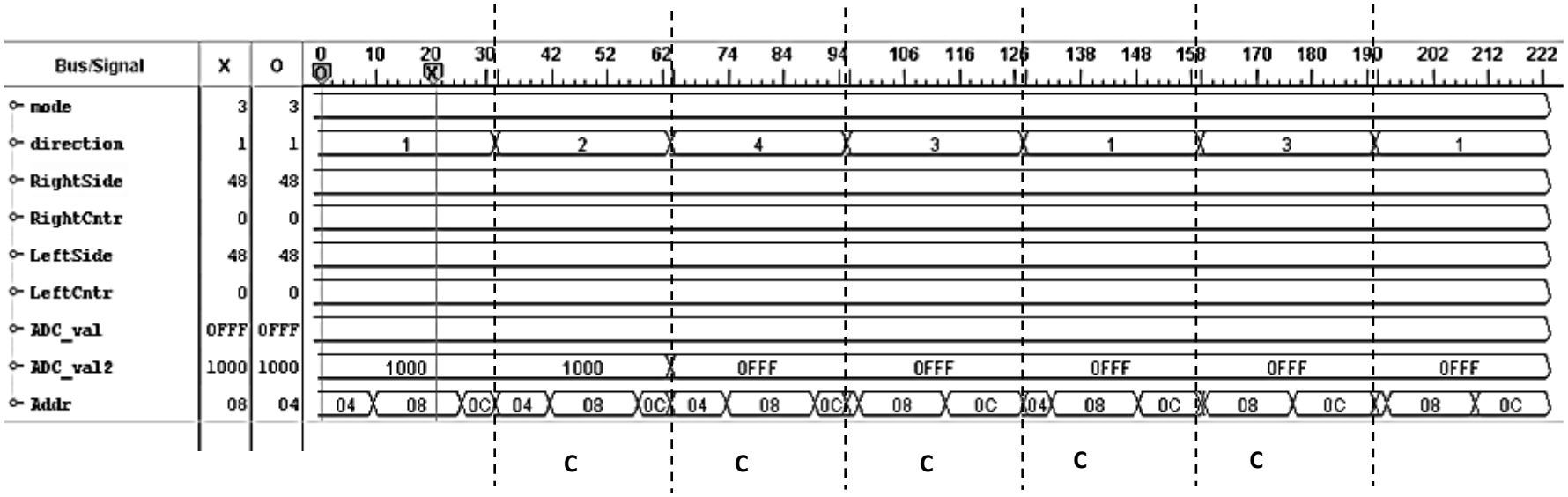

Figure 5. 4: Various Directions in Navigation Mode

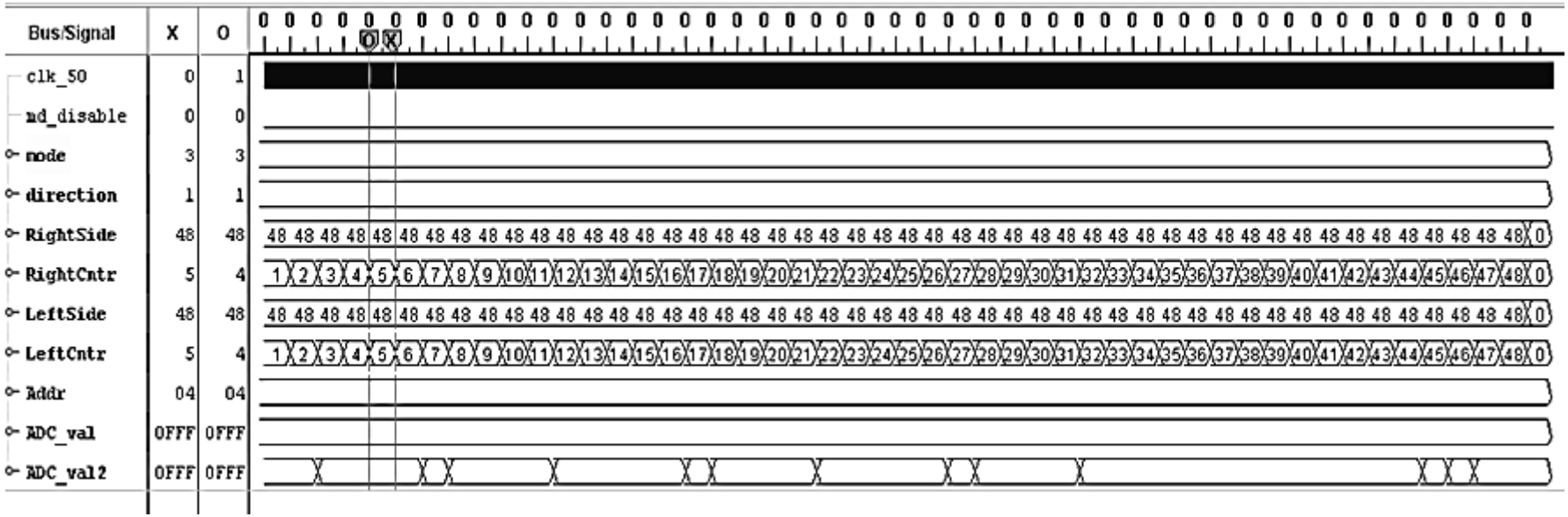

Figure 5. 3: Navigation Turn for 12 Wheel Turns

The required time to switch from one direction to another in the navigation mode of operation is 40cc, assuming no movement has been made, since three memory locations, “Addr”, in the BRAM need to be read as shown in Figure 2.

As discussed in chapter 4, the encoder counts the number of black and white lines placed on the shaft of the motors. Therefore, to have the wheels to turn 13 times, 48 lines need to be counted by the encoder as shown in “RightCntr” and “LeftCntr” of Figure 3. The “ADC_val” and “ADC_val2” represent the digital values obtained from the ADC operation of the encoders to determine the detected black and white lines. It is clear that, the time spent during the navigation mode depends on many factors including the type and condition of the terrain in which the robot is moving on, battery condition, selected route by the operator, possible obstacles, etc.

A similar experiment was performed on the idle to observation mode transition of the system, and It was observed that the time required to shift from the idle to the observation mode of operation, “B”, is 12 c.c. , and the time to switch from the observation to the idle mode, “A”, is 18 c.c. assuming as shown in Figure 5.5.

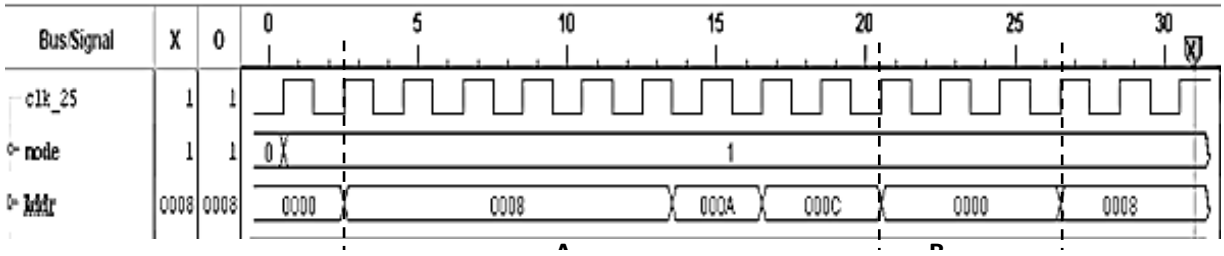

Figure 5. 5: Idle-Observation Mode Transition

5.4 Power Consumption Analysis

The power consumption analysis of the design was first estimated by means of Xilinx XPower Analyzer (XPA) 14.2 [58]. Therefore, the parameters of the implemented design were passed to the software and estimated power consumption of the design in static and dynamic was obtained and recorded in Table 5.1.

Table 5. 1: Estimated Power Summary Courtesy of [58]

Static (W)	Dynamic (W)	Total (W)
0.097	0.000	0.097

The static power results mainly from transistor leakage current in the device and the dynamic power is associated with the design activity and switching events in the core or I/O of the device [58].

The next step in pursuing our analysis is to investigate the actual power consumption of the robotic platform of the telepresence system. Therefore, a set of tests involving measurement of the power consumption for each mode of operation was performed. It is worth mentioning that power supply for the control and communication part of the system is isolated from the mechanical components, such as motors, actuators and their controllers to reduce any noise or unwanted interferences that may occur. Furthermore, the input voltage based on the selected FPGA board was set to 5.5 VDC; hence, the current-I was measured for all modes of operation during their peak activity to calculate the total power consumption, as recorded in Table 5.2.

Table 5. 2: Power Consumption of the Control and Communication Part

Mode	Current (mA)	Voltage (V)	Power Consumption(watts)
Default	185	5.5	1.01
Main	250		1.37
Observation	235		1.29
Navigation	250		1.37
Front-View	220		1.21
Idle	210		1.15

As can be seen in the table above, the default power consumption of the board is 1.01 watts, which is the consumption of the board when no configuration bit stream is loaded on the device and it is relatively close to the estimated power consumption determined by the Xilinx XPA. The difference between the estimated and measured power consumption value can be due to the power supply, the incomplete user-defined constraints and specifications, providing power from the board to other components and the heat generated by other components of the system, affecting the power dissipation from FPGA to the environment.

Moreover, the power consumption of each mode of operation is relatively low but close to the total power consumption of the main component since the implemented multiplexing algorithm allows only a certain mode to be active at any specific time.

The above experiments help to determine appropriate power supply for the board. Therefore a 6VDC-10Ah rechargeable battery (as discussed in Section 4.3), which is scaled down to 5.5 VDC through a DC-DC converter, will allow a non-stop operation of the system for over 34 hours. Lastly, the low power dissipation makes it possible to avoid using any fans or heat sinks in the design for cooling purposes.

5.5 Area and Resources Analysis

To investigate the occupied area and used resources in the implemented design, Xilinx Software Kit was used. The design goal of the system was set to a balanced optimization of performance vs. run time during the test. Verification state, the floor planning and I/O planning of the components made by the Xilinx software kit was recorded and analyzed to determine the occupied area and used resources. Table 5.3 demonstrates the estimated number of resources used in the major components of the system.

Table 5. 3: Resource Organization of the System as Obtained via XPA

Resources	Idle	Observation	Front-View	Navigation
Multipliers (MULTs)	3	0	0	0
Lookup Tables (LUTs)	1789	127	2	390
Shift Register LUTs	130	0	0	0
Block Memory (BRAMs)	40	0	0	0
Distributed RAM	256	0	0	0
Clock Manager(DCMs)	1	0	0	0
Flip-Flops (FFs)	1233	133	2	219

Furthermore, the estimated resource utilization of the implemented system on the selected FPGA device is recorded in Figure 5.6.

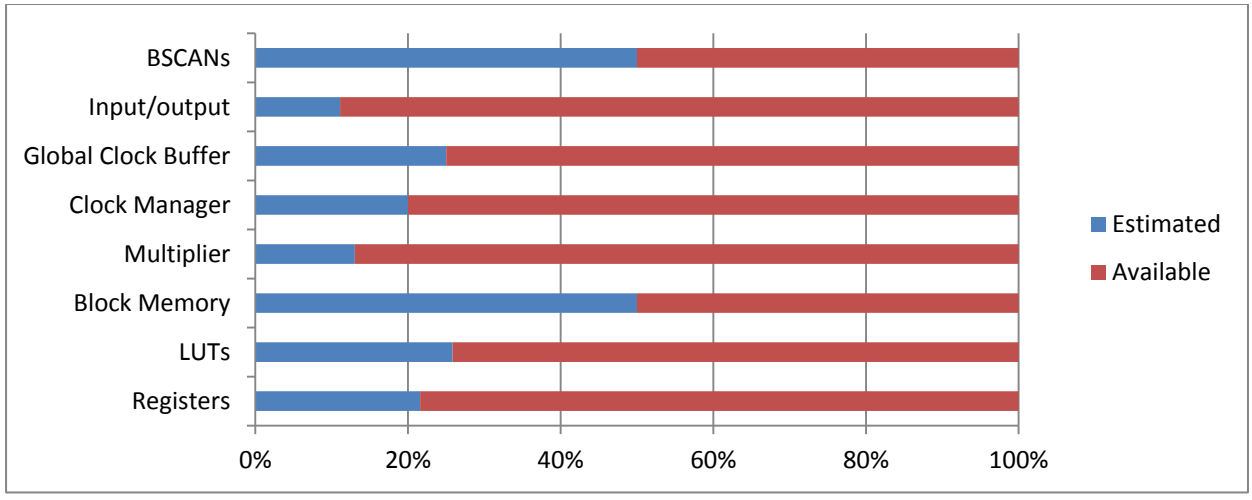

Figure 5. 6: Logic Utilization of the System

The total number of LUTs includes the LUTs used as logic, route-through, shift registers and for dual port RAMs. The number of occupied slices only contains the related logical blocks [58]. The BSCANs block is used to enable an extension of the JTAG interface to internal user defined scan chains and can be used by ChipScope and MicroBlaze loader for testing and debugging purposes. The Global Clock Buffer is the global clock multiplexer buffer of the FPGA to select between two input clocks [59]. The Multiplier is a 36-bit output, 18x18-bit input signed multiplier to perform asynchronous and synchronous multiplication operations [60]. The Clock Manager is a digital clock manager that provides advanced clocking ability to the applications implemented on the selected FPGA by optionally multiplying or dividing the incoming clock frequency to synthesize a new clock frequency [61].

The data recorded in the tables above indicate that the implemented system occupies less than half of the FPGA device. Hence, more modes of operation may be added to the system to enhance the functionality of the system, if required. Moreover, the data presented in Figure 5.6, suggests that the MicroBlaze component, which is shared among other modes of operation, consumes the most resources. However, some of the subcomponents used in MicroBlaze, such as the debugging module and relatively large memory block, are used for testing and debugging purposes and can be omitted and modified to reduce the number of resources and, therefore, the area used by the MicroBlaze.

It can also be concluded that even though the multiplexing approach seems to help us meet the requirement of this project, expanding the project in future may be affected by the limited number of resources within the selected device.

5.6 Discussion and Conclusion

The reconfigurability concept of the system was discussed in Chapter 3; it was proved that in terms of mechanical components of the design a single hardware component could perform various tasks in different modes of operation. Even though the assigned functions were quite simple, the theory behind the implemented design can be expanded to more complicated systems with numerous hardware elements capable of mimicking various forms to perform different tasks. The flexibility and scalability of the designed platform are the main features of the system that differentiates the robotic platform from the traditional robotic systems, for it can be used as an instrument for new and existing systems, in which the physical and behavioral components of the system may undergo minor or major changes.

Furthermore, the performance parameters of the proposed reconfigurable system discussed in the previous sections verifies the feasibility of the design using the multiplexing approach. However, in this section we will examine the optimization of the proposed design through using partial reconfiguration based on the observed performance parameters in the previous sections.

Partial reconfiguration is defined as the process of updating/changing some portions of the hardware circuitry while the other parts remain unchanged. Hence, even electronic hardware can be designed in modular/block form by creating sub components, in which the functionality of its hardware can be enhanced by altering the organization of these sub components [62, 63]. Figure 5.7 illustrated the concept of partial reconfiguration in FPGAs.

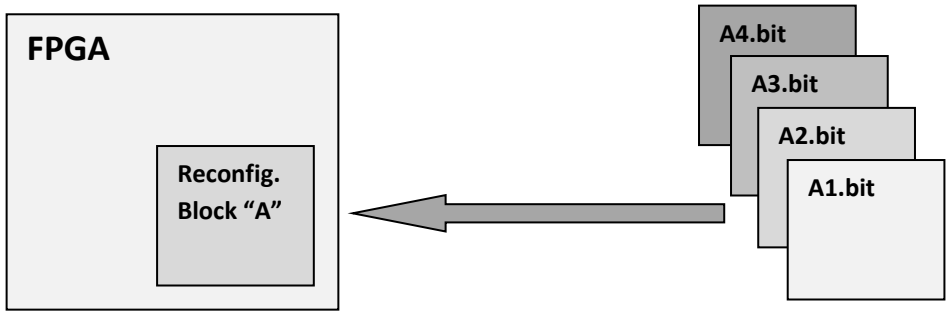

Figure 5. 7: Basic Premise of Partial Reconfiguration courtesy of [62]

As can be seen in Figure 5.7, the logic in the FPGA design is divided into static, grey part, and in the block portion, reconfigurable logics. The functionality of reconfigurable block A can be modified by downloading one of several partial bit files A1.bit- A4.bit without affecting what is stored in the static

part. These partial data files may contain the required hardware organization of each mode of operations and may be stored in an external memory storage device, such as a flash memory, EEPROM, etc.

The timing analysis performed in Section 5.3 confirmed that around 100 ms is required to configure the FPGA from the external flash memory. Moreover, based on the specified modes of operation and the architectural organization of the system, the system shifts to idle mode after completion of each operation. Hence, to examine the applicability of partial reconfiguration in our existing system in terms of optimization and higher performance of the system, we can study the behaviour of systems in the following cases:

Case 1: Idle-Observation Mode

Upon receiving a command to enter the observation mode from the control unit, the system immediately enters the observation mode by setting the Tower up and turning on the camera module to start capturing and transmitting video data. After the Tower is in high position, which takes almost four seconds, the system can enter back the idle mode to receive further instructions. The existing multiplexing approach takes only 12 c.c. (240 ns) as described in Section 5.3, which is far less and faster than the 100 ms required time to reconfigure the FPGA with a new bitstream. In other words, loading the dedicated circuitry associated with the new mode of operation to the specified area of the FPGA will take longer that the existing approach. However, if the functions associated with some of the operational modes change and result in occupying larger area and using more resources, the partial reconfiguration option will be a better choice, since it allows various modes of operation to be placed in the same blocks of FPGA. The same analysis is valid when entering the idle mode from the front-view and navigation modes of operation.

Case 2: Observation-Front-view Mode

Entering the front-view mode from the observation mode requires very minimal physical configurations, since the tower is already in the high position. Therefore, the system receives the associated command while it is in the idle mode and can switch to the front-view mode after disabling the observation mode. The hardware circuitry associated with the head movement of the platform does not need to wait for the tower to go up due to its initial physical configuration which speeds up the adaptation process. The system starts to prepare a 3D map of the scene to be used for the navigation mode of operation which requires N-angular positions of the Head. The system enters the idle mode,

upon finishing the task. The existing multiplexing approach takes 12 c.c. for the specified mode transitions, which is less than the required time to reconfigure the FPGA. Therefore, the applied approach meets the timing requirements of this mode transition faster than its counterpart.

Case 3: Observation-Navigation Mode

To enter the navigation mode from the observation mode, the system needs to enter the idle mode and then navigation mode. However, the system needs to physically adapt itself to the new mode of operation. For instance, the Tower which is set high in the observation mode needs to be placed down, before entering the navigation mode to meet the physical specification of this mode of operation.

The existing multiplexing scheme takes 12 c.c. to switch from the observation to the idle mode and remains in this mode depending on how fast the navigation command becomes available. Upon receiving the navigation command it enters the navigation mode after the transition to observation mode and setting the Tower to down position. The whole process takes less than 100 c.c. (20 μs), if the reception of the navigation command is almost immediate.

The estimated time to have the Tower fully down is around three seconds, which provides plenty of time to reconfigure the FPGA with the bitstream of the new mode while the physical configuration is still in the process of adaptation. In other words, if the partial reconfiguration was to be used, the hardware circuitry associated with the idle mode of operation would be the static part of the FPGA and the other operational modes would be downloaded on to the FPGA via module-based partial reconfiguration to change the system's behaviour by reallocating only the related hardware resources without completely reconfiguring the entire FPGA. Similar analysis can be performed on the front-view to navigation and observation to front-view operational modes transitions.

Based on the above analysis, the existing multiplexing approach in the transition of specified operational modes is suitable for the system, while partial reconfiguration of the system with the defined modes offers no significant advantages for the current robotic platform. However, in terms of area and resource utilization of the design, as described in Sections 5.5, partial reconfiguration makes a good candidate for further improvements of the system since adding more functionality in the form of modes of operation does not require the switch to a larger FPGA device.

Appendices

A. MicroBlaze Component Internal Organization

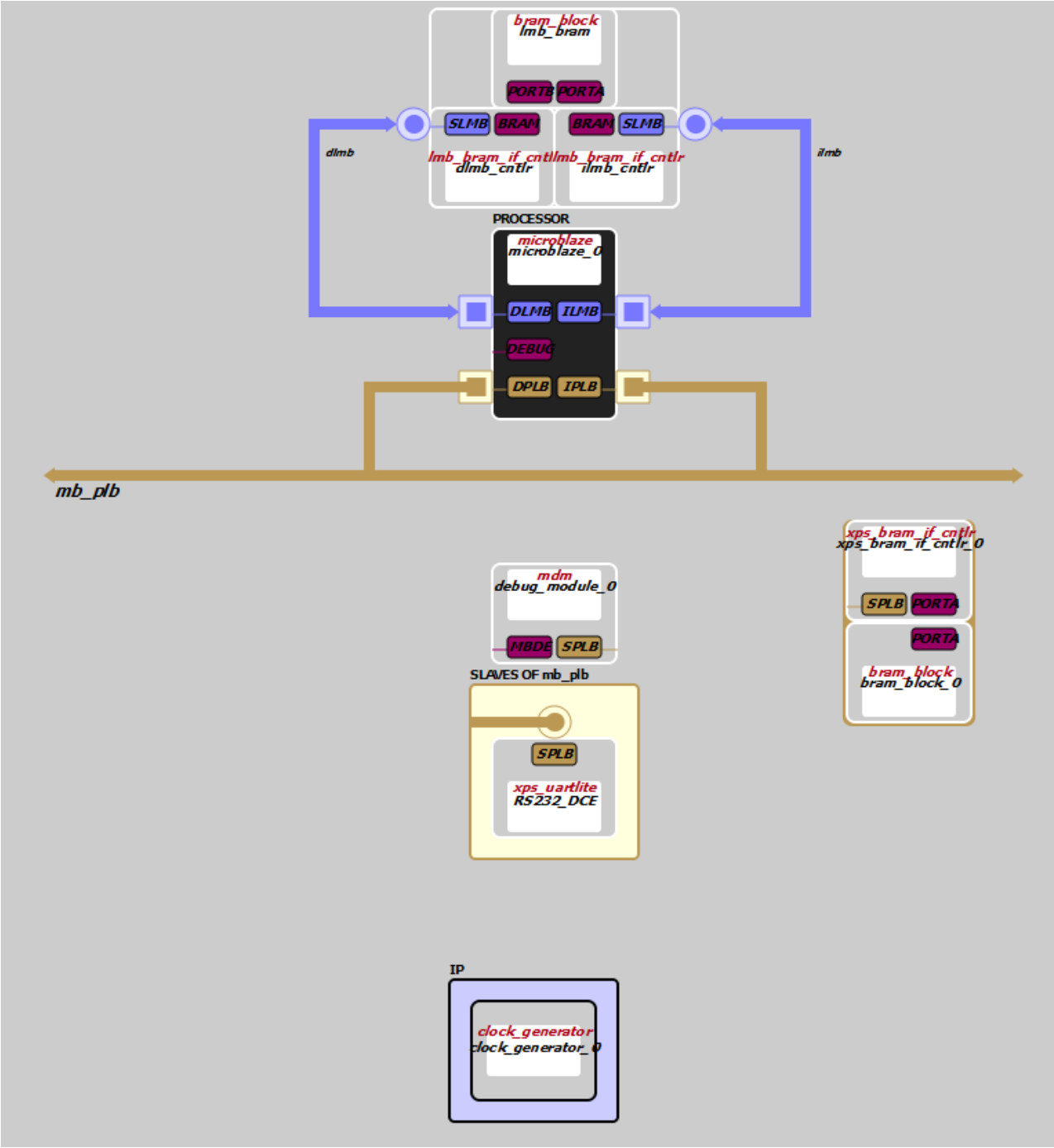

Figure A. 1: MicroBlaze Internal Component Organization

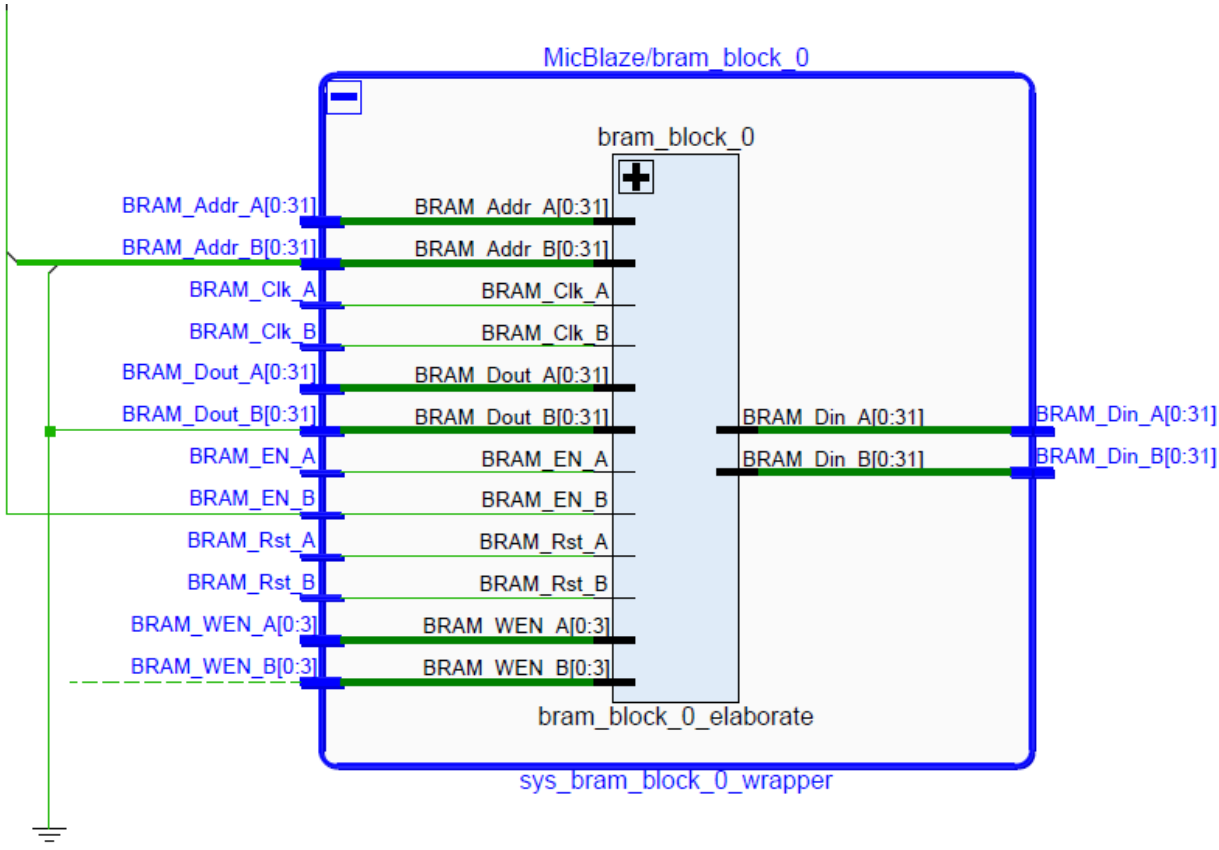

Figure A. 2: MicroBlaze-Dual BRAM Component Organization

B. System Component Symbols

The following figures demonstrate the actual component symbols of the system for each modes of operation.

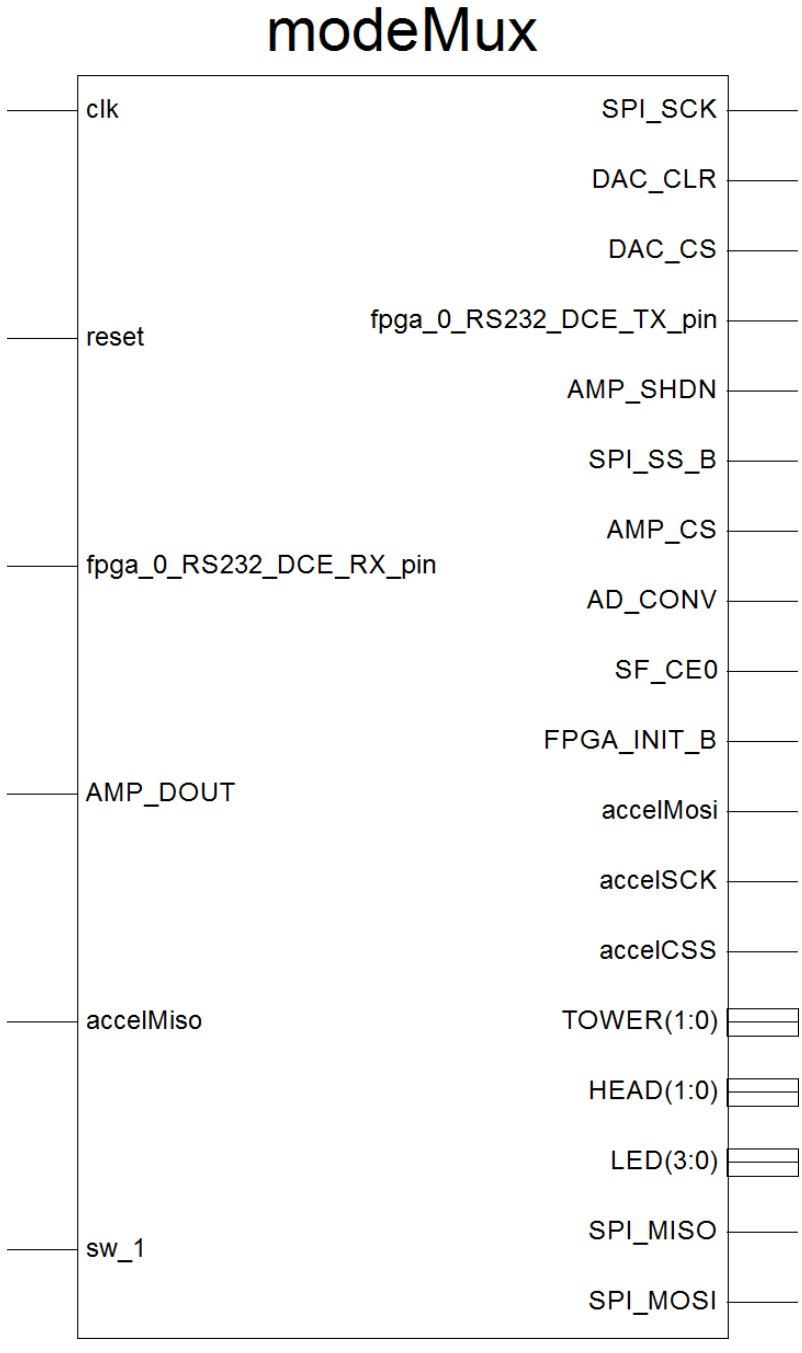

Figure B. 1: Main Component Symbol for Multiplexing Approach

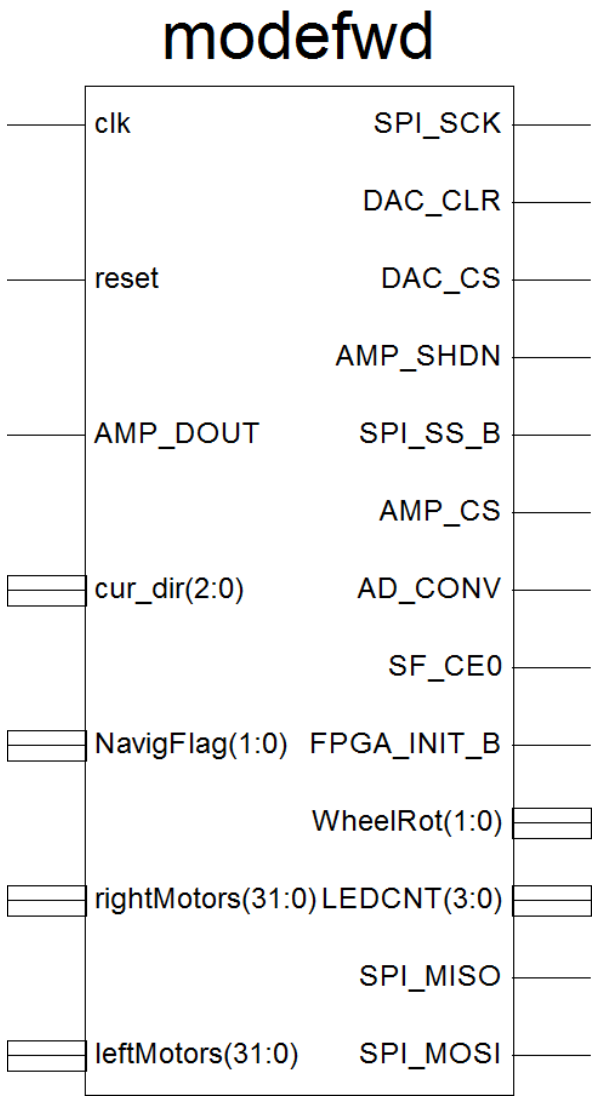

Figure B. 2: Navigation Mode Component Symbol

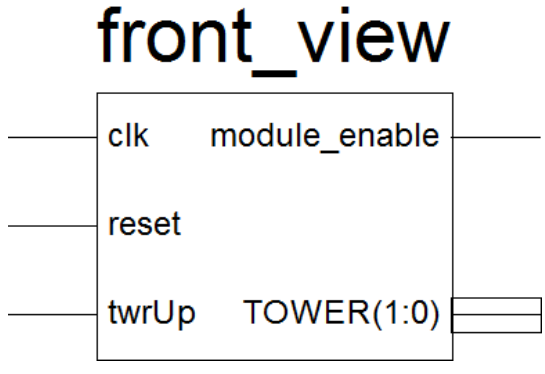

Figure B. 3: Front-View Mode Component Symbol

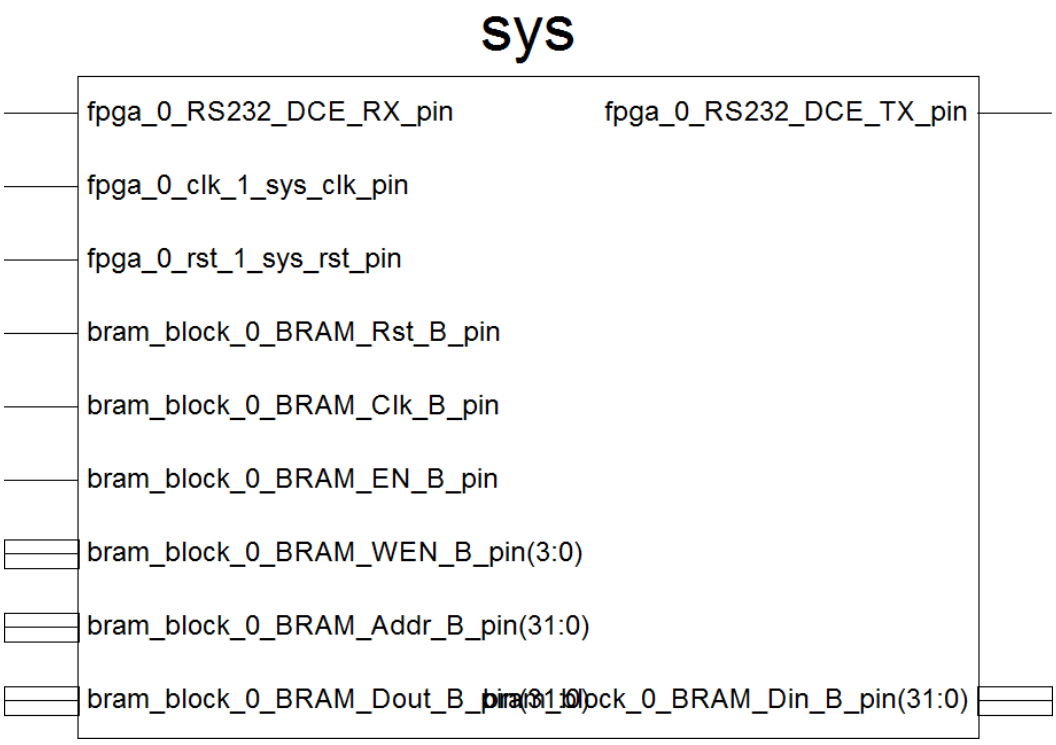

Figure B. 4: MicroBlaze Component Symbol for Idle Mode

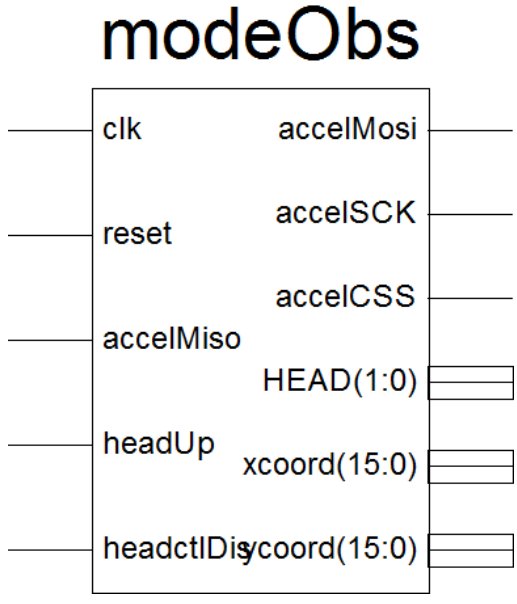

Figure B. 5: Observation Mode Component Symbol

Bibliography

[1] A. Saakov, “Reconfigurable platform for 3D-panoramic telepresence system for mobile applications” in *Theses and dissertations*, Paper 758, 2011, pp. 6-48.

[2] A. Saakov, D. Marcantonio, V. Dumitriu , and L. Kirischian, “Embedded Reconfigurable System for 3D-Panoramic Telepresence Application with Natural User Interface”, *Presentation at workshop SVAR-2010: Space Vision and Advanced Robotics,* MDA Space Missions, Brampton, Canada*,* 2010.

[3] J. Denhart, T. Gemmer, L. Edwin, S. Ferguson, and A. Mazzoleni, "Assessing Reconfigurable Design for a Chaotic Objective in a Mars Exploration Rover." in *14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012, pp. 1-4.

[4] A. Kristoffersson, S. Coradeschi, and A. Loutfi, "A review of mobile robotic telepresence." *Advances in Human-Computer Interaction,* 2013, pp. 1-2.

[5] G.H Ballantyne,. "Robotic surgery, telerobotic surgery, telepresence, and telementoring." *Surgical Endoscopy and Other Interventional Techniques* 16, no. 10, 2002, pp. 1389-1402.

[6] J. Kim, N. Prabakar, and C. Tope, "Efficient concurrent operations of telepresence avatars." in *Robotics (ISR), 2013 44th International Symposium on*, IEEE, 2013, pp. 1-5.

[7] P. Peixoto, J. Gonçalves, H. Antunes, J. Batista, and H. Araujo, "A surveillance system integrating visual telepresence." in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 4, IEEE, 2000, pp. 98-101.

[8] Oculus, “Oculus Surveillance and Telepresence Robot”, Oculus [Online], March 2013, Available: http://www.xaxxon.com/

[9] R. E Balfour, and B.P. Donnelly, "The what, why and how of achieving urban telepresence." In *Systems, Applications and Technology Conference (LISAT), 2013 IEEE Long Island*, IEEE, 2013, pp. 1-6.

[10] R. Terrile, and J. Noraky, "Immersive telepresence as an alternative for human exploration." in *Aerospace Conference, 2012 IEEE*, IEEE, 2012, pp. 1-11.

[11] K.M. Varadarajan, and M. Vincze, "Augmented virtuality based immersive telepresence for control of mining robots." in *Computational Intelligence and Intelligent Informatics (ISCIII), 2011 5th International Symposium*, IEEE, 2011, pp. 133-138.

[12] M.A. Diftler, J. S. Mehling, M.E. Abdallah, N.A. Radford, L.B. Bridgwater, A.M. Sanders, R. Scott R.S. Askew et al, "Robonaut 2-the first humanoid robot in space." in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 2178-2183.

[13] P. Moubarak, and P. Ben-Tzvi, "Modular and reconfigurable mobile robotics." in *Robotics and Autonomous Systems* 60, no. 12, 2012, pp. 1648-1663.

[14] K. Gilpin, and D. Rus, "Modular robot systems.", in *Robotics & Automation Magazine, IEEE* 17, no. 3 2010, pp. 38-55.

[15] S. Farritor, and J. Zhang, "A reconfigurable Robotic Infrastructure to Support Planetary Surface Operations.", Department of Mechanical Engineering, University of Nebraska, Lincoln, USA, 2001.

[16] S. Murata, and H. Kurokawa, "Self-reconfigurable robots." in *Robotics & Automation Magazine, IEEE* 14, no. 1, 2007, pp. 71-78.

[17] M. Yim, P. White, M. Park, and J. Sastra, "Modular self-reconfigurable robots." in *Encyclopedia of complexity and systems science*, Springer New York, 2009, pp. 5618-5631.

[18] S. Jin, D. Kim, X. Dai Pham, and J. Wook Jeon, "FPGA-based image processing system for remote robot control." in *Robotics and Biomimetic, 2008. ROBIO 2008. IEEE International Conference on*, 2009, pp. 120-124.

[19] N. Brener, F.B. Amar, and P. Bidaud, "Characterization of Lattice Modular Robots by Discrete Displacement Groups", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS),* Tapei, Taiwan, 2010, p. 1.

[20] P. Dasgupta, et al, "Mechanical design and computational aspects for locomotion and reconfiguration of the ModRED modular robot." in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1359-1360.

[21] Y. Fei, and C. Wang, "Self-Repairing Algorithm of Lattice-Type Self-Reconfigurable Modular Robots." in *Journal of Intelligent & Robotic Systems,* 2013, pp. 1-11.

[22] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, and S. Kokaji, “Hardware design of modular robotic system,” in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2000, pp. 2210–2217.

[23] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, “Modular self-reconfigurable robot systems [Grand Challenges of Robotics],” *IEEE Robot. Autom. Mag*., vol. 14, no. 1, 2007, pp. 43-52.

[24] M.D.M. Kutzer, M.S. Moses, C.Y. Brown, D.H. Scheidt, G.S. Chirikjian, and M. Armand, "Design of a new independently-mobile reconfigurable modular robot.", in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 2758-2764

[25] J. Davey, N.Kwok, and M. Yim, "Emulating self-reconfigurable robots-design of the SMORES system." in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 4464-4469.

[26] M-TRAN, “M-TRAN Modular Transformer”, [Online], November 2010, Available: https://unit.aist.go.jp/is/frrg/dsysd/mtran3/

[27] K. Stoy, and H.Kurokawa, "Current topics in classic self-reconfigurable robot research." in *Proceedings of the IROS Workshop on Reconfigurable Modular Robotics: Challenges of Mechatronic and Bio-Chemo-Hybrid Systems*, 2011, pp. 1-4.

[28] O. Anton, B. Gelineau, and J. Sauget, “Firmware and bootlader”, [Online], March 2012, Available: http://rose.eu.org/2012/wp-content/uploads/2012/03/Firmwares-and-bootloaders.pdf

[29] N. Wolchover, “Nasa Gives Up On Stuck Mars Rover Spirit”, [Online], May 2011, Available: http://www.space.com/11773-nasa-mars-rover-spirit-mission-ends.html

[30] P. Leong, H. Wai, and K. Hung Tsoi, "Field Programmable Gate Array technology for robotics applications." in *Robotics and Biomimetic (ROBIO), 2005 IEEE International Conference on*, 2005, pp. 295-298.

[31] J. González-Gómez, E. Aguayo, and E. Boemo, "Locomotion of a Modular Worm-like Robot using a FPGA-based embedded MicroBlaze Soft-processor.", in *Climbing and Walking Robots*, Springer Berlin Heidelberg, 2005, pp. 869-878.

[32] A. Upegui, R. Moeckel, E. Dittrich, A. Ijspeert, and E. Sanchez, "An FPGA dynamically reconfigurable framework for modular robotics." in *Workshop Proceedings of the 18th International Conference on Architecture of Computing Systems 2005 (ARCS" 05)*, no. BIOROB-CONF-2005-001, VDE Verlag, Berlin, 2005.

[33] R. Möckel, C. Jaquier, K. Drapel, E. Dittrich, A. Upegui, and A. Ijspeert, "YaMoR and Bluemove—an autonomous modular robot with Bluetooth interfaces for exploring adaptive locomotion." in *Climbing and Walking Robots,* Springer Berlin Heidelberg, 2006, pp. 685-692.

[34] V. Tadigotla, L. Sliger, and S. Commuri, "FPGA implementation of dynamic run-time behavior reconfiguration in robots." in *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, 2006, pp. 1220-1225.

[35] M. Yim, D. Duff, and K.D. Roufas, "PolyBot: a modular reconfigurable robot," in *Robotics and Automation, 2000, Proceedings.* ICRA'00. IEEE International Conference on, vol. 1, EEE, 2000, pp. 514-520.

[36] A. Kongmunvattana, and P. Chongstivatana, "A FPGA-based behavioral control system for a mobile robot.", in *Circuits and Systems*, 1998. IEEE APCCAS 1998. The 1998 IEEE Asia-Pacific Conference on, IEEE, 1998, 759-762.

[37] T. Kwok, and Y. Kwok, "Hardware Task Scheduling for an FPGA-Based Mobile Robot in Wireless Sensor Networks." in *Handbook on Mobile and Ubiquitous Computing: Status and Perspective*, 2012, p. 441.

[38] O. Berthold, "Self-reconfiguring System-on-Chip using Linux on a Virtex-5 FPGA", in *Master's thesis*, Humboldt-University of Berlin, Berlin, Germany, 2012, pp. 2-24.

[39] V. Adnitt, "Wildlife in Winter- Adaptation for Survival", in *Young People's Trust for the Environment,* 2010, [Online], Available: http://www.ypte.org.uk/environmental/wildlife-in-winter-adaptations-for-survival/112

[40] Digi International Inc., “XBee/XBee-PRO RF Modules”, *Product Manual for 802.15.4 Protocol*, September 2009, [Online], Available: https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

[41] P. P.Chu, *FPGA Prototyping by VHDL Examples: Xilinx Spartan-3*, John Wiley & Sons, 2008. [E-book], Available: Google Books.

[42] Xilinx, “XPS UART Lite (v1.01a)”, Xilinx [Online], December 2009, Available: http://www.xilinx.com/support/documentation/ip_documentation/xps_uartlite.pdf

[43] Firgelli, Application note, [Online], Available: http://www.firgelliauto.com

[44] Bosch, “BMA180 Digital, triaxial acceleration sensor”, *BMA180 datasheet*, December 2010, [Online], Available: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Accelerometers/BST-BMA180-DS000-07_2.pdf

[45] Smart Motor Devices, Application note, [Online], Available: http://www.stepmotor.biz/

[46] Dimension Engineering, “Sabertooth 2x25 V2 User’s Guide”, April 2012, [Online], Available: http://www.dimensionengineering.com/datasheets/Sabertooth2x25v2.pdf

[47] Battery University, “Comparison Table of Secondary Batteries”, 2011, [Online], Available: http://batteryuniversity.com/learn/article/secondary_batteries

[48] Vishay, “Reflective Optical Sensor with PIN Photodiode Output”, TCND5000 datasheet, July 2009 [Online], Available: http://www.vishay.com/docs/83795/tcnd5000.pdf

[49] Xilinx, “Spartan-3E Starter Kit Board User Guide (v1.2)”, *Xilinx User Guide*, [Online], January 2012, Available: http://www.xilinx.com/products/boards-and-kits/HW-SPAR3E-SK-US-G.htm

[50] Linear Technology, “Quad 16-bit Rail-to-Rail DACs in 16-Lead SSOP”, *LTC2624 datasheet*, [online], Available: http://cds.linear.com/docs/en/datasheet/6912fa.pdf

[51] Linear Technology, “Dual Programmable Gain Amplifiers with Serial Digital Interface”, *LTC6912 datasheet*, [online], Available: http://cds.linear.com/docs/en/datasheet/6912fa.pdf

[52] Xilinx, “ISE In-Depth Tutorial”, *Xilinx User Guide*, April 2012, [online], Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/ise_tutorial_ug695.pdf

[53] Xilinx, “EDK Concepts, Tools, and Techniques: A Hands-On Guide to Effective Embedded System Design”, *Xilinx User Guide*, July 2012, [online], Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/edk_ctt.pdf

[54] Xilinx, “Xilinx Software Development Kit Help Contents”, *Xilinx User Guide*, [Online], Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_2/SDK_Doc/index.html

[55] Xilinx, “ISE tutorial: Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA Applications”, *Xilinx User Guide*, March 2013, [online], Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/ChipScope_pro_sw_cores_ug029.pdf

[56] E. Crabill, “Powering and Configuring Spartan-3 Generation FPGAs in Compliant PCI Applications”, *Xilinx User Guide*, June 2007, [online], Available: http://www.xilinx.com/support/documentation/application_notes/xapp457.pdf

[57] A. Khu, F. Shokouhi, J. Hussein, A. Patel, “Using Xilinx XCF02S/XCF04S JTAG PROMs for Data Storage Applications”, *Xilinx User Guide*, January 2011, [online], Available: http://www.xilinx.com/support/documentation/application_notes/xapp544.pdf

[58] Xilinx, “Xilinx Power Estimator User Guide”, *Xilinx User Guide*, June 2013, [online], Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/ug440.pdf

[59] Xilinx, “Spartan-3 Generation FPGA User Guide: Extended Spartan-3A, Spartan-3E, and Spartan-3 FPGA Families”, *Xilinx User Guide*, June 2011, [Online], Available: http://www.xilinx.com/support/documentation/user_guides/ug331.pdf

[60] Xilinx, “Using Embedded Multipliers in Spartan-3E FPGAs”, Xilinx User Guide, May 2003, [online], Available: http://www.xilinx.com/support/documentation/application_notes/xapp467.pdf

[61] Xilinx, “Using Digital Clock Managers (DCMs) in Spartan-3E FPGAs”, *Xilinx User Guide*, January 2006, [online], Available: http://www.xilinx.com/support/documentation/application_notes/xapp462.pdf

[62] Xilinx, “Partial Reconfiguration User Guide”, *Xilinx User Guide*, May 2010, [online], Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/ug702.pdf