Project Lab 1 – Mini Motor Project

Dirk Thieme

R11636727

Texas Tech University

February 2022

Abstract

This paper is meant to describe how Group 3 fulfilled the requirements for the Motor Mini Project while meeting the requirements of the Written Lab Requirements for the Electrical and Computer Engineering department. Group 3 consists of Dirk Thieme, Erik Manis, and Mohammed Ansari.

Table of Contents

Table of Figures

List of Tables

1. Introduction

The Mini Motor Project was assigned in order to introduce the Rover 5 Tracked Robot Chassis and the Basys 3 board for use in the final project of the semester. The Robot Chassis is a simple platform that utilizes two brushed DC motors to run caterpillar treads that move the device at a maximum speed of 25 cm/s [1]. This speed is achieved with the help of the two 86.8:1 gearboxes [1] that give the motors enough torque to move the contraption's weight along with anything attached.

The brain of the device is the Basys 3 Board provided by Digilent, which is where most of the complexity of this project lies. The board is powered by the Artix-7 FPGA, which features 33,280 logic cells in 5,200 slices, 1,800 Kbits of RAM, five clock management tiles, 90 DSP slices, a 100MHz clock, and an XADC which was very important to this implementation of the project [2]. The Board also includes a multitude of input and output options, most notably user-programmable switches which acted as inputs for the speed of the motor, a built-in seven segment display which was used to display the direction of the Rover and the current threshold, and the Pmod ports which were used to output signals to turn the motors on or off based on the current and switch position. The Board was able to interact with the Rover using an L298N H Bridge, a chip which can take in the relatively low voltage signals of the Pmod ports and output the correct forward or reverse signals to the motors. The Basys 3 Board is configured with the use of Verilog, a hardware description language which is used to model and emulate electrical circuits and systems [3]. The Verilog modules are synthesized, implemented, and uploaded to the board using the Xilinx Vivado IDE, which is also where code may be written and checked for errors. Another important feature of the Vivado software is the use of Xilinx's IP catalogue, in which the XADC Wizard is available to be configured for use [5], which played a major role in the success of the project's goals. The main goals were to use the switches to drive the motor at multiple speeds, record the current threshold and direction to the seven-segment display, and to implement some form of current protection, as the motors can take a maximum of 1A each before risking damage. To power the board, a jumper wire was connected to the output leg of the L7805C chip located on the H-Bridge circuit, which takes the 9-11V of the battery and converts it down to 5V, which is the nominal input voltage of the Board. A 3D printed enclosure was used to mount and cover the various parts of the rover, including the H-Bridge, the Basys 3 Board, and the battery. The final assembly is pictured below:

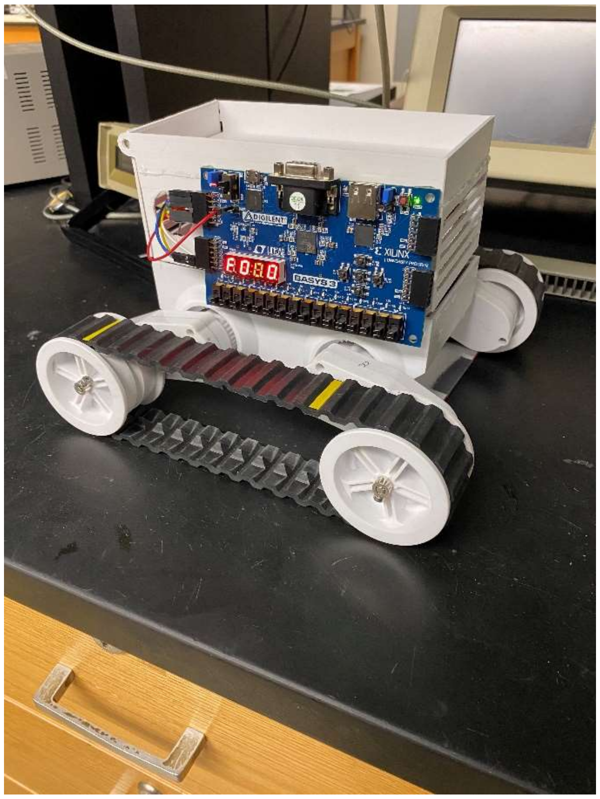


Fig. I – Fully Assembled Rover

2. Body

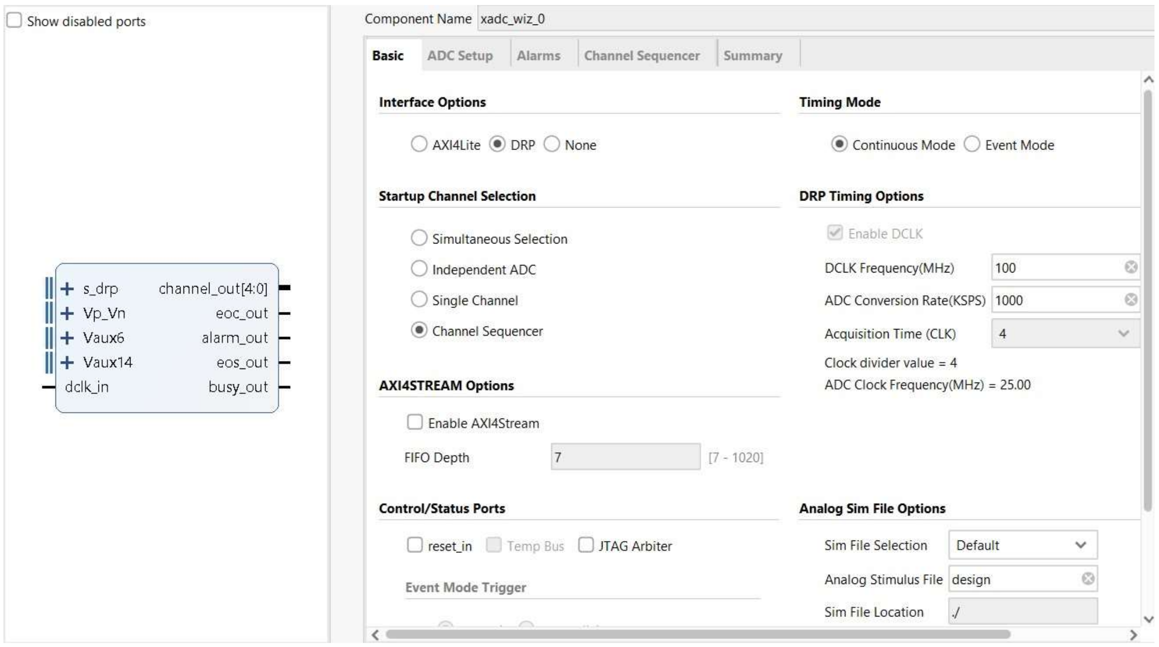
One of the most important distinctions to make before continuing with the discussions about how the Rover was created is the difference between a microcontroller, for example an Arduino or XBee3, and an FPGA, which is what powers the Basys 3 Board. While they seem similar to control, the methods of programming and interfacing with them are very different. A microcontroller, which in this case is an XBee3, has a built in CPU, RAM, and other peripherals to interact with, usually programmed with a standard language such as Python. The CPU and instructions are predefined and not user editable, so while they are still powerful, they cannot be precisely tuned for use like an FPGA. A field programmable gate array, or FPGA, is a collection of ICs which can be defined and configured by the user programming them. This presents an advantage over a standard microcontroller in speed and size, as a user can prune and optimize their design to exactly what they need, with nothing more. The biggest downside with using the FPGA is the much higher learning curve to actually configure it for use, as the user has to actually create a circuit using a hardware description language such as Verilog or VHDL, which has a much steeper learning curve. Now that the differences are clearer, the design of the Rover can be discussed with less ambiguity. The Verilog that runs the Rover consists of a top module that functions as a motherboard of sorts and four sub-modules: the Current Sensing module, the Seven-Segment Display Controller, the Comparator, and finally the Motor Controller.

2a. Top Module

The top module simply consists of the wires that connect the four modules together. It receives inputs from the eight switches and the four pins of the ADC, a positive and negative connection for current sense pins A and B. After processing all the signals through their respective modules, it accesses the Pmod ports to output signals to enable or disable the movement of the motor depending on switches pulled and current threshold. It also assigns the LEDs above the switches to indicate which one is pulled at which time.

2b. Current Sensor Module

The current sensor module, while nearly the most important module on the Board, provided by far the most difficult challenge to complete. The current sensor functions by inputting two differential signals into the JXADC Pmod port of the Board, a positive and negative, and subtracts them to get the actual voltage. The inputs were provided on the H-Bridge as SENSA, SENSB, and GND, which were one Ohm current sense resistors where the current is equal to the output voltage. These analog voltage signals were pass through to the XADC itself, which was instantiated into the module using the XADC Wizard IP. The Wizard allows the user to customize the function of the ADC to their use case, which for this Rover was configured to Simultaneous Continuous Sampling at 25MHz, which was repeatedly switched between the pins for SENSA and SENSB throughout simulation. This allows the ADC to read the current output from both motors in the event one is approaching 1A while the other isn't. The repeated switching also prevents any errors due to averaging the two values, which is another option to configure in the Wizard. Pictured below is the XADC Wizard with the settings previously configured:



Show disabled ports

Component Name xadc_wiz_0

Interface Options

AXI4Lite [✓ DRP] None

Timing Mode

[✓ Continuous Mode] Event Mode

Startup Channel Selection

Simultaneous Selection
Independent ADC
Single Channel
[✓ Channel Sequencer]

DRP Timing Options

[✓ Enable DCLK]
DCLK Frequency(MHz) 100
ADC Conversion Rate(KSPS) 1000
Acquisition Time (CLK) 4
Clock divider value = 4
ADC Clock Frequency(MHz) = 25.00

AXI4STREAM Options

Enable AXI4Stream
FIFO Depth 7 [7 - 1020]

Control/Status Ports

reset_in Temp Bus JTAG Arbiter

Event Mode Trigger

Analog Sim File Options

Sim File Selection Default
Analog Stimulus File design
Sim File Location ./

Figure II – XADC Wizard Configuration

The XADC outputs a 16-bit binary number between 0 and 4095 to represent the input voltage between 0 and 1V using this equation:

$$output = \left(\left(\frac{n}{1.0}\right) * 0xFFF\right) \qquad (1)$$

where *n* corresponds to the input voltage in volts. The 12 MSBs of that number contain the relevant data while the four LSBs contain unimportant noise [4]. The remainder of this module takes that number and converts it to a decimal number using multiple state machines which determine which bits of the number correspond to the final decimal number. The new decimal number which came from the conversions is then output to both the Comparator Module and the Seven-Segment Display Controller Module.

2c. Comparator Module

The comparator is a simple module that reads in the number that was output from the Current Sensor Module and outputs a digital high reset signal if it meets certain criteria. As stated earlier, the Rover can safely operate only below 1A, so the threshold to output a high reset signal was anytime the current was greater than 900mA to account for any delays in the system. If none of those criteria are met, the reset output remains low. The reset signal is then output to the Motor Controller module, where it is decoded and handled appropriately.

2d. Motor Controller Module

This module is what actually creates the movement for the Rover. It begins by receiving the inputs from the eight switches built-in to the Basys 3 Board and the reset signal provided from the Comparator. To be able to control the speed the motors function at, a PWM signal is what must be output to the motors, as otherwise it would be running at either max power or off without any finer control. To generate that PWM signal, a counter of 17 bits was utilized, with the signal corresponding to a percentage of that number. For example, to generate a 50 percent pulse, the output to the motors remains one for 65536 ticks, and then goes to zero for the remaining ticks. Using a 17-bit number means the Rover functions at ~762Hz, which is fast enough to have a noticeable affect between speeds, but slow enough to respond to the lower PWM signals, as higher frequencies did not leave enough time for smaller PWM values to register. The Rover had seven speeds which were determined by switches 0 – 6, the logic for deciding which switches meant which speed is defined in Table 1:

Table I – PWM Output Based on Switch Configuration

Switch Position	**Output PWM Signal**
8’b00000000	0%
8’b00000001	30%
8’b00000010	40%
8’b00000100	50%
8’b00001000	60%
8’b00010000	70%
8’b00100000	80%
8’b01000000	90%
8’b10000000	0%
8’b10000001	30%
8’b10000010	40%
8’b10000100	50%
8’b10001000	60%
8’b10010000	70%
8’b10100000	80%
8’b11000000	90%

The switches are read as an 8-bit number and the table above tells the motor controller how much to count to based on which switch is pulled. If more than one switch is pulled besides switch 7, the output is set to zero to prevent issues. The motor controller then uses switch 7 to determine which two of the four IN pins to output two, IN1 and IN4 if pulled, which is reverse, or IN2 and IN3 for forward if not toggled. The controller is also set to output zero to the four pins either after their PWM cycle has completed or if the reset signal is detected, which turns the motors off before any damage is caused.

2e. Seven-Segment Controller Module

The final part to complete the Rover is a module which displays the direction and current on the built-in display on the Basys 3 Board. This module inputs the current from the Current Sensor and switch 7, which determines the direction of the motor. A 20-bit counter is used to drive the four displays at ~95Hz, fast enough to prevent strobing but slow enough to allow the numbers to be clearly read. The display on the Board works by first selecting which anode to output to, then decoding that digit into segments [2]. The Basys 3 Board uses a transistor, which means the desired anode and segments are driven low to be activated, while unwanted anodes and segments are driven high. For example, to display a three into the first digit, the 4-bit number which is used to drive the anodes is set to 4'b0111, and the 7-bit number that is used to drive the segments is set to 7'b0110000. To denote direction, if switch seven is low, an 'F' is displayed in the first digit for forward, and if it is pulled high an 'r' is displayed for reverse. The module takes the 2 MSBs of the 20-bit counter and uses them as a multiplexer to rapidly update the anodes, while simple math is done to separate each digit of the three-digit decimal number, which is then displayed on the appropriate place.

2f. Physical Design

The physical design of the project was relatively simple compared to the coding, as the Rover and Board have already been designed for use, with few actual connections required to get things moving. The main challenge was creating the H-Bridge to power the motors, as it needed to be soldered piece by piece. Group 3 was successfully able to make three working H-Bridges, in the event of accidental damage to one there are backups to rely on. Another hurdle which was overcome quickly was the actual wiring of the Pmod ports to the input ports of the H-Bridge, as little instruction was given as to which lines were for what. Luckily, early on in the design process a wiring diagram was created for which wires were to plug in to which inputs, shown below:

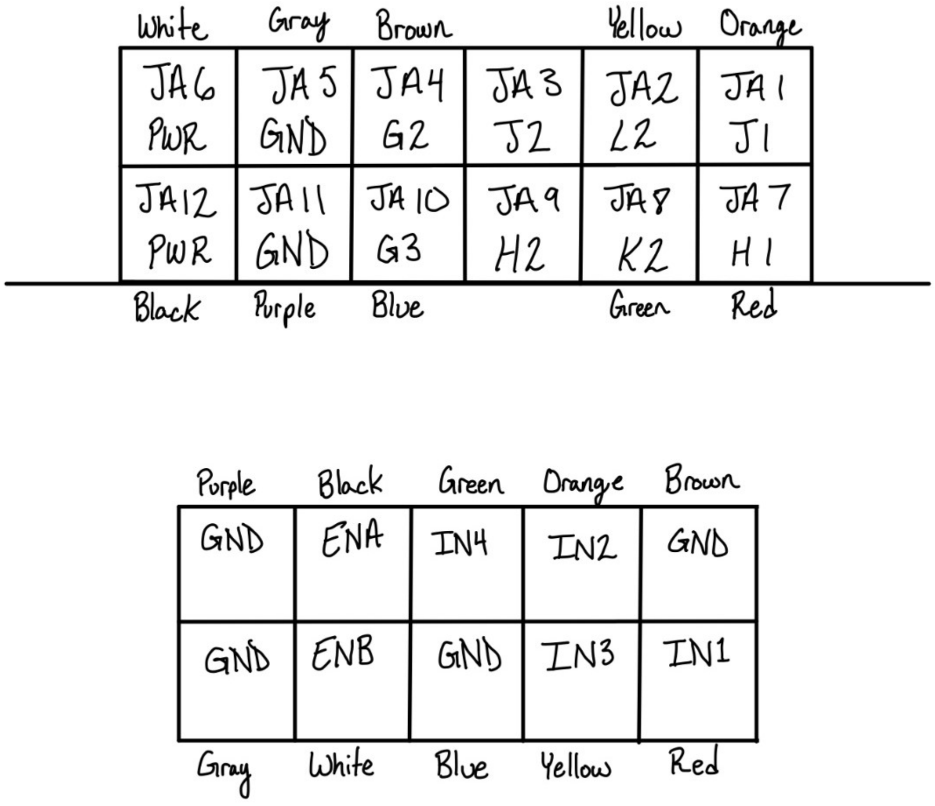


Figure III – Physical Connections between Pmod port JA (above) and the H-Bridge (below)

Another issue that cropped up was how to power the board when unconnected to the computer, as being a Rover means it has to move around on its own. The quick and dirty solution used in this case was by hijacking the L8705C 5V output [6] and wiring it to the 5V input on the Basys 3 Board. In the future, this circuit is used for other purposes on the H-Bridge, meaning that trying to also use it for Board power is not feasible. For this mini project, however, there is more than enough spare power laying around to safely tap into this 5V. To test if this 5V output would hold steady over the life of the battery, a copy of the circuit was designed in LTSpice and was simulated with an input voltage that fluctuated between 9V-11V, to represent the battery discharging over the time. This circuit is shown below:

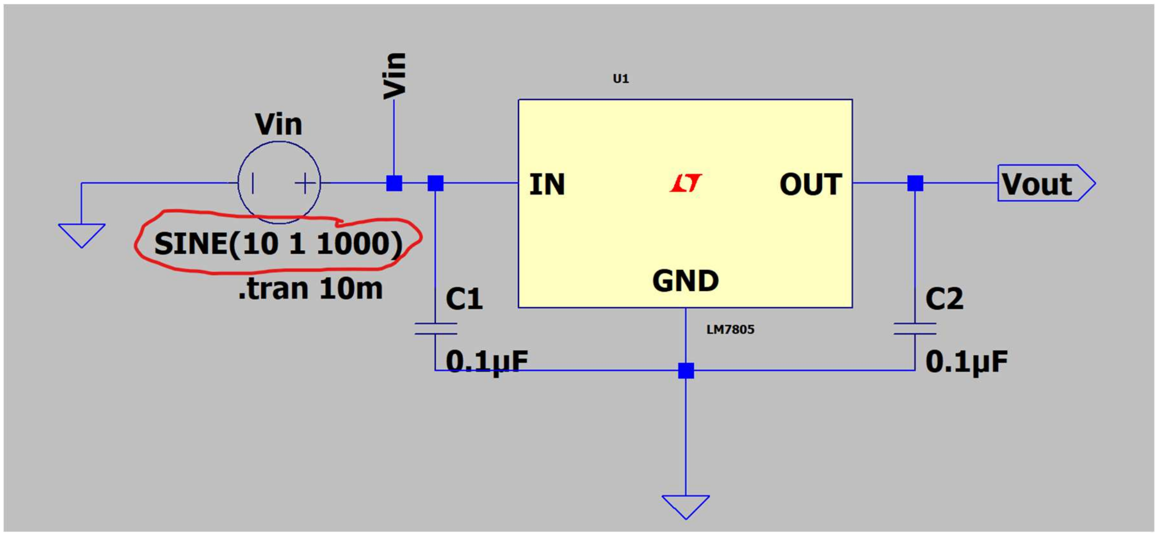


Figure IV – L7805 Circuit

The output was shown to hold steady between the provided voltage in the figure below, where the green wave is the “battery” input voltage, and the blue wave is the output voltage:

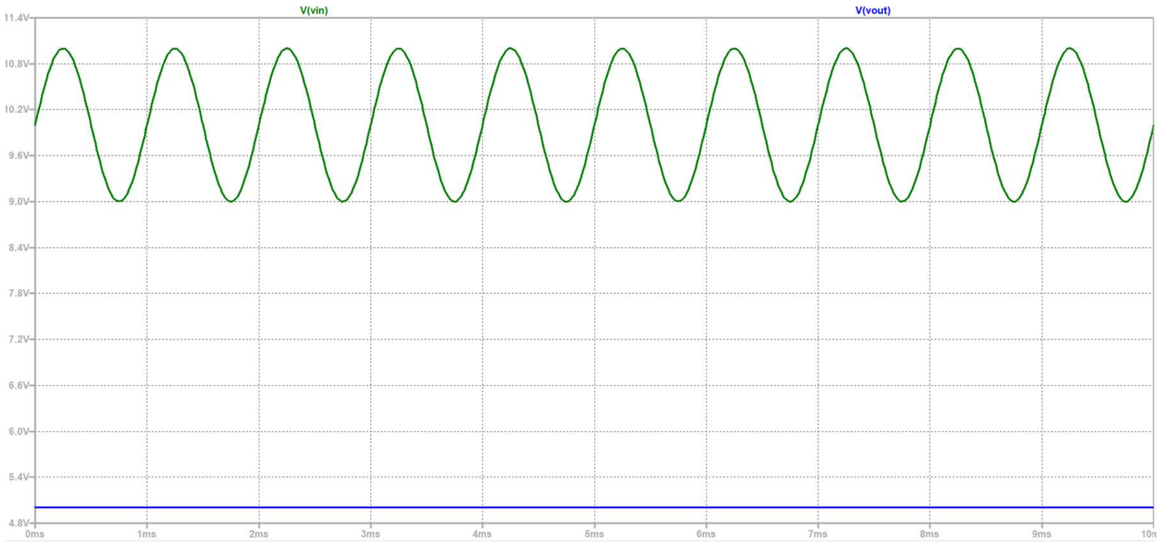


Figure V – Circuit Simulation Results

Finally, the combined H-Bridge, battery, and Basys 3 Board were placed into a 3D-printed enclosure to contain the wires and protect the delicate components while also making the Rover appear more aesthetically pleasing. The enclosure was designed using Sharp3D and printed using the Makerspace printers in the library.

3. Conclusion

This Rover outperformed nearly all the other Rovers presented on Demo Day, as the vast majority had issues with their use of a hardware-based comparator circuit that failed to signal the motors to shut off in the case of reaching 1A of current. The use of a software-based comparator is an upside compared to the standard LM339 as over 16 sources can be read and worked with using just a few more lines of code, rather than having to redesign a circuit each time more than four sources are to be checked. Beyond the fact that this comparator worked flawlessly during presentation, the use of the ADC to display an actual, live current output was only implemented with this design, as all of the other Rovers simply displayed if 1A had been reached or not, rather than a true reading. While learning how to use the XADC Wizard IP and the XADC itself was an immense challenge, in the end it was worth it as this was one of two Rovers to be able to stop upon the set current threshold during the live presentations. Unfortunately, despite the success of certain areas of the Rover, other areas are still not working all the way. One of the biggest issues is a timing issue between what appears to be the XADC and the binary to decimal conversion, which means that only every three to four updates of the XADC are properly displayed upon the seven-segment display, while the others are either zeroes or numbers to small to be true. The silver lining in this is that it seemingly does not affect comparator performance at this time, however much more testing is to be done to sync them correctly. Another issue which was mentioned before is powering the Board without a USB connection, however that is already being worked on, and initial designs for a system of voltage regulators powered by the motor battery are being discussed. In conclusion, while this Rover may not be 100% perfect and ready for every test in the world, the code is robust enough to function even with the errors discussed above, and the physical design is clean and protective to minimize loose connections and shorts from items touching.

References

1. "Pololu - Dagu Rover 5 tracked chassis with encoders," Pololu Robotics Electronics. [Online]. Available: https://www.pololu.com/product/1551. [Accessed: 10-Feb-2022].
2. "Basys 3 Reference Manual," Basys 3 Reference Manual - Digilent Reference. [Online]. Available: https://digilent.com/reference/programmable-logic/basys-3/reference-manual. [Accessed: 10-Feb-2022].
3. "VLSI design - Verilog Introduction," Tutorialspoint. [Online]. Available: https://www.tutorialspoint.com/vlsi_design/vlsi_design_verilog_introduction.htm. [Accessed: 10-Feb-2022].
4. "7 series FPGAs and Zynq-7000 SOC XADC dual 12-bit 1MSPS Analog-to-Digital Converter," 23-Jul-2018. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf. [Accessed: 10-Feb-2022].
5. "XADC Wizard v3," 05-Oct-2016. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/xadc_wiz/v3_3/pg091-xadc-wiz.pdf. [Accessed: 10-Feb-2022].
6. STMicroelectronics, "L7800 Series Positive Voltage Regulators," L7805C datasheet, Nov. 9, 2004.