DC Motor Control System

Austin Dial, Sean Morris, Darius Gray

Contact Information

Austin Healy Dial
Electrical and Computer Engineering Department
Caterpillar College of Engineering and Technology
Bradley University

Peoria Harper Hall #414
1314 West Main Street
Peoria, IL, 61606, USA

Phone: 1+ (858) 414-4347
E-Mail: adial@mail.bradley.edu

Table of Contents

Project Confirmation Letter

1. Project Proposal
    1. Specifications
    2. Block Diagram
    3. Division of Labor
    4. Timeline Milestones
2. Project Report
    1. Introduction
    2. Background
    3. Design Procedure
    4. Project Results and Demonstration
    5. Summary and Conclusion
    6. Works Cited

Project Confirmation Letter

Our team is fit for the DC Motor design project because we have the desire to expand our horizons and face dynamic challenges with innovative solutions. We have a desire to begin working directly with control systems early in our careers and the DC Motor control system offers us the opportunity to do so. Furthermore Sean Morris and Austin Dial offer the programming experience and technique required to manipulate the VHDL programming language in the manners necessary to complete the project with expertise and flair. Additionally Darius Gray offers the valuable skills in mathematics necessary to manipulate the frequencies involved in the pulse-width modulation signal creation. Together, we will strive to not only satisfy the base requirement of this project, the creation of the PWM signal, but to include additional achievements in our project to enhance the value of our resulting control system.

By the end of this project we will have gained insight into the workings of the FPGA development process using the sequential logic design topics. Accordingly, we will have gained experience working with the FPGA clock system, effectively bringing us into the world of clocked circuitry and sequential logic design.

1 Project Proposal

1.1 Specifications

- Design a pulse-width modulator signal generator.
- Vary duty-cycle percentage by tens (e.g. 10%, 20%, 30%, etc.)
- Send signals through an external accessory header pin to control a direct current motor.
- Control duty cycle percentage via the rotary shaft encoder.

1.2 Block Diagram

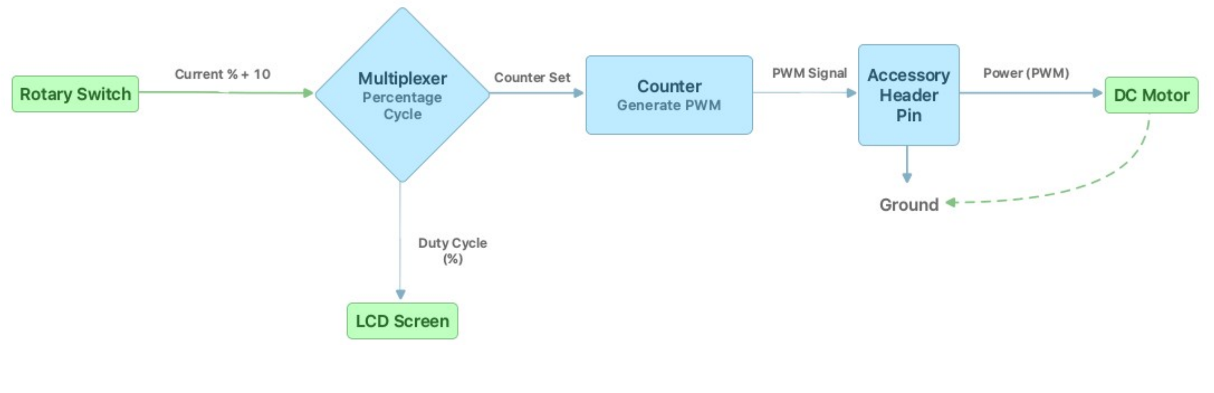

Figure 1: System Block Diagram

As illustrated in the diagram above, the system receives input from the user via the rotary switch and sends the input to a variable representing the current percentage duty cycle, for every tick the current duty cycle increases by ten percent until reaching one hundred percent such that the duty cycle cannot increase any further. The current state is sent into a multiplexer which sends the duty cycle to be displayed on an LCD screen for the user's convenience. It also sets the PWM signal settings for the counter which then creates the PWM signal and sends it through an accessory pin to control the motor, which is ground relative to the board via the accessory pin system's ground function.

1.3 Division of Labor

Austin Dial will be responsible for the VHDL design components of the project. This entails conducting research on VHDL techniques relevant to the creation of the PWM signal and it's auxiliary inputs and outputs such as the rotary . Sean Morris will be responsible for the management of the motor and for understanding the manner in which the motor will interact with the PWM signal and the main VHDL program. Darius Gray has the duty of researching PWM and motor control theory along with collaborating with Austin Dial and Sean Morris in the creation and optimization of the PWM generator.

1.4 Time-line and Milestones

Date	Goal
Nov 26	Produce code ready for compiling
Nov 27	Compile code
Nov 30	Successfully generate multiple PWM signals
Dec 6	Control Duty-cycle using the rotary shaft encoder
Dec 7	Prepare for design presentation
Dec 8	Submit project report

Table 1: Project Time-line

Milestones for the project include configuring the clock divider to generate multiple frequencies and duty cycles, configuring the signals to control the DC Motor, linking the duty-cycle settings to the rotary switch, and . By working hard and following the schedule, it is entirely within the realm of possibility that these deliverables may be met.

However, preparing for unseen errors and complications is always advantageous for projects and given our late start relative to our aspirations it is also within the realm of possibility that our project fall short of our goals.

2 Project Report

2.1 Introduction

The driving force for pulse-width modulation (PWM) is quite simple; with the availability of higher-level processing, the ability to design advanced control systems aimed at regulating motor and servo outputs at high resolution with minimized power waste became a reality. Seeing as PWM regulates the power perceived by an electrical system, its applications transcend that of motor control to touch on such fields as telecommunications, audio amplification, and even the regulation of Field-Effect transistors. By exploring PWM creation, we open the door to understanding more about many of today's electrical systems.

For example, the brightness of an LED may be controlled using PWM signals. This is due to the theory governing PWMs which states that if one oscillates an independent voltage source at a high frequency such that the signal spends a disproportionate amount of time on or off then the voltage perceived by the load can be measured by the amount of time spent on a state of on (duty-cycle) multiplied by the voltage source, shown below:

$$Voltage\ perceived = Duty\ cycle \cdot Voltage\ source$$

This means that with a constant source of ten volts and a duty-cycle of fifty percent the load will only perceive a five volt source. The beauty of this is that the duty cycle may be altered in real time to change the voltage perceived by the load, this is the fundamental theory behind pulse-width modulation and is the reason why we can alter the brightness of an LED. If one powers an LED using a six volt source and provides a fifty percent duty-cycle then the LED will be essentially powered by three volts.

2.2 Background

Systems of motor control independent of that of PWM have long been available to engineers. By adjusting the resistance between power source and a motor one can adjust the voltage perceived by said motor (see figure 2). This technique works quite effectively when the percentage of power available from the source that's delivered to the motor is high, in other words the amount of heat disbursed from the resistor is low. Unfortunately this efficiency deteriorates as the amount of power required by the motor decreases as it puts an increased amount of work load on the resistor and thus dramatically increases the heat produced by the system, this is problematic as the heat changes the system's dynamics. Additionally, the amount of power wasted by the system through heat increases, ergo the system is fundamentally flawed as it cannot control the motor without wasting energy.

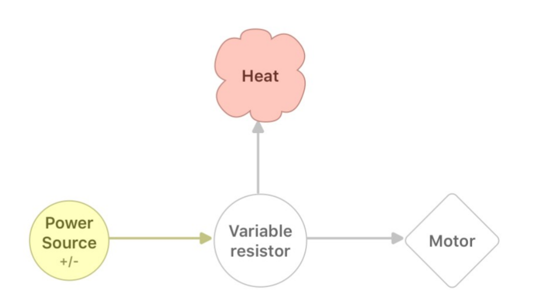

Figure 2: Control w/ Resistor

This drives the motivation behind the creation of a new control system which can alter the voltage perceived by the desired component without creating such a waste of energy. The strategy which is born therefrom is to alter the amount of time spent on an active high and an active low signal rapidly such that the desired component only perceives voltage directly proportional to the percentage of time spent on active high in a given period (duty-cycle) multiplied by the voltage source. Ergo, if a PWM controlling a motor were to operate at a twenty-five-percent duty-cycle at twelve volts then the motor would perceive a supply of three volts seeing as point twenty-five multiplied by twelve volts equals three volts. This is the fundamental mechanism behind pulse-width modulation (see figure 3).

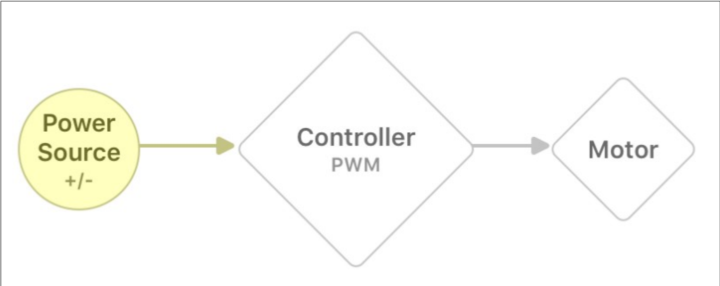

Figure 3: Control w/ PWM

The real beauty of this process is the energy efficiency. By using a constant voltage source and simply oscillating between on and off states the PWM system ultimately wastes very little energy relative to the variable resistor method. The added benefit is that the PWM system doesn't dramatically increase the heat in the system as it's predecessor so does. This becomes important when the energy in the system must be kept low dues to the presence of micro-components or other forms of sensitive equipment. Such is true for a system using an FPGA.

Seeing as our design project uses the rotary shaft encoder to alter the duty-cycle assigned to the PWM, it becomes necessary to explain the purpose, mechanics, and programming behind operating this physical component. The rotary shaft encoder operates by turning on one axis unbounded. This means that one could theoretically continue turning the rotary shaft encoder forever without pause. This contrasts with the rotary switch which has a set number of potential states. This distinction is important as the later component has a fundamentally different system of operation. For example, if one turns the rotary switch from one position to another the component simply returns it's current position to the control system, eliminating the necessity for measuring the direction of turn. The rotary shaft encoder, however, can continue spinning forever and thus perform spins greater than 360. Ergo each resting state may be considered unique and relative to its previous states; this necessitates a system to measure the direction the encoder is turned relative to its previous position.

From these necessities the mechanics of the rotary shaft encoder was designed. Quite beautifully, the component was designed with two channels, A and B, which are set to active high by default. When the rotary shaft encoder is turned both A and B are momentarily grounded; however, one channel is grounded before the other such that one can measure who grounded first. This helps determine which direction the encoder was turned and use this information to adjust a setting. When the encoder is turned clock-wise, the encoder grounds A just before it grounds

B and thus one can measure its having turned clock-wise. The opposite is true for counter clock-wise turns, in which B is grounded just before A. By measuring the difference between the two, one can determine the direction.

Applied in the context of this problem, the rotary shaft encoder governs the duty-cycle and has the potential to increase or decrease the duty-cycle by ten percent. As one can see it becomes crucial that the main program can distinguish between a turn clock-wise and a turn counter clock-wise as if it simply detects rotation independent of direction then it cannot accurately change the value of the duty-cycle and thus surrenders its value to the project. This step was actually a source of much frustration

The reason behind the ten percent resolution is the use of an if-then conditional structure to assign values to the counting variables. The greater the resolution of the duty-cycle the more if-then conditionals are needed, as one conditional is needed for each possible state of the duty-cycle variable. We decided that setting the duty cycle by tens provided a high enough resolution to be useful for controlling the motor but small enough to make coding easier. Additionally, the motor will be responsive as the difference between a twenty percent and an eighty percent duty-cycle is simply six clicks of the encoder, whereas with a one percent resolution that difference would be many revolutions of the encoder and may distract from the usefulness that this system has to offer the user.

As mentioned below in the design procedure, our attempts at designing the PWM creator using the rotary shaft encoder were ultimately quite complicated and distracted form the main objective, to control pulse width. After many attempts at solving the component, we decided to set the clear switch to the push buttons and set the duty-cycle control to the switches.

2.3 Design Procedure

Initially our team began by isolating the goals we wished to achieve through our assigned project. These goals included the creation of multiple PWM signal duty-cycles controlled using the rotary-shaft encoder and displayed on the on-board LCD screen. From here we developed input/output diagrams to clarify our plans throughout the team and to agree as to how to approach the problems. We isolated each problem and began to research how to best represent these problems in VHDL and ultimately load onto the FPGA. Once our research was compiled we began to develop our code.

The design procedure ultimately employed by our team was to modularize our code into separate encapsulated files written in the VHDL language for testing and then to concatenate these modules individually until a fully functioning main programming file was achieved. We then optimized our code for efficiency as well as comprehension and began testing the code's ability to control the DC motor.

Ultimately PWM signal creation may be described as a particular type of counter in VHDL. The formula behind creating the signal must distinguish between time spent on a state of active-high (1) and active-low (0) and count the two of them proportionally such that in each period the counter sets the PWM state to the exact specifications demanded by the duty cycle, which once again is the percentage of time spend on a state of active high within one period. We can represent the number to which the counter must meet by using two variables: ONN and OFF, hereby referred to as the “counting-variables”. These represent the the amount of time the signal spends in each binary state and is represented in code as a standard-logic-vector.

At this point we encounter some simple arithmetic in terms of computing the values for the counting-variables. If the frequency of the main clock powering the counter is fifty MHz, then representing ONN and OFF as percentages of fifty million would set the period to one second, at a frequency of one Hz. This is impractical as the resolution is far too low to practically power the motor. Ergo, we set the values of the counting-variables to percentages of a higher resolution. For example, if we wanted a resolution of roughly ten kHz then we would divide the main clock by the desired frequency to get the total number of counts per period, this is illustrated below:

Total Count Value = Main Clock / Desired Frequency

This demonstrates that with a desired signal frequency of ten kHz we would have the counter count to a total of five-thousand within a given period. Similarly, we could increase the frequency to any given resolution be decreasing the total number of counts per period. This is because the faster the PWM signal oscillates between high and low within one second the higher the resolution, this indicates that having smaller values for the counting-variables is advantageous. However, setting the oscillations too high makes the PWM signal less measurable using an oscilloscope as is may reach frequencies of, say, 100 kHz. We settled on a resolution of ten-kilo-Hertz as it's effective and measurable using an oscilloscope.

From here we simply must relate the duty-cycle to the values set to the counting-variables. This was achieved in our code through the use of if-then conditionals which measured the integer value of Duty_Cycle (range 0 to 100) and set the values of OFF and ONN accordingly. For example, if the Duty-Cycle is set to a value of fifty then the value of ONN would be set to .5 times 2500 and OFF set to (1.0 - .5) times 2500. This process is continued for each possible combination of Duty_Cycle by tens such that the variable will never account for values which are not percentages of ten.

Once the values of ONN and OFF have been assigned they are sent to the counter, which is responsible for controlling the width of the PWM signal. The counter works by setting the signal to active-high and then counting to the value of ONN. It then inverts the signal to an active-low and counts to the value of OFF. This has the effect of controlling the time spent on a state of active-high (ONN) and active-low (OFF). Because the counter works off of the counting variables, the PWM may be adjusted in real time to the sensitivity of the duty-cycle variable. This is actually quite robust given that many PWMs work off of an integer input and not a representative variable.

Now that we've walked through the process of creating a PWM signal from the duty-cycle integer we can control said integer to control PWM in real time. Using the rotary shaft encoder we attempted to increase or decrease the duty-cycle with a resolution of+/- 10% per click. Seeing as the rotary shaft encoder's turn direction may be detected by comparing the channels A and B, it became quite difficult to determine spin direction after debouncing the signal. For example, if we fed the data from the channels through a delay which waited a certain amount of time and then measure the channels after said time had elapsed, then the precedence between the two channels could not be distinguished and the signal was effectively useless. We then tried to use flip flops to pass the values from the channels and this too failed as the D-flip flops were computed so quickly that the program failed to overcome the debouncing. We then tried passing the variables into each other such that they may delay the reading enough so as not to bounce and yet keep the channel precedence. VHDL treated these transfers just as it had treated the D-flip flops and didn't prevent the bouncing of the encoder. In another attempt we forced the program to take only the first inputs and to reject anything else for a sizable half-second, this too failed for unknown reasons. We requested assistance from the TA, professor, and finally from other groups. Strangely their code was logically identical to ours and the solution to our problems evaded even the most experienced.

Finally we made the judgment call to make use of the pushbutton switches for controlling our clear signal and then dedicated the slide switches to controlling the duty-cycle via a 4-bit multiplexer. To prevent the pushbuttons from bouncing we assigned one of them to setting the clear signal to active-low and another button to setting the clear signal to active-low. This means that even if the signal bounced upon being pressed, it would simply declare clear as active-high or active-low multiple times, having the same effect as having not bounced. This work-around allowed us to bypass the debouncing problem that had so plagued us and present our project immediately.

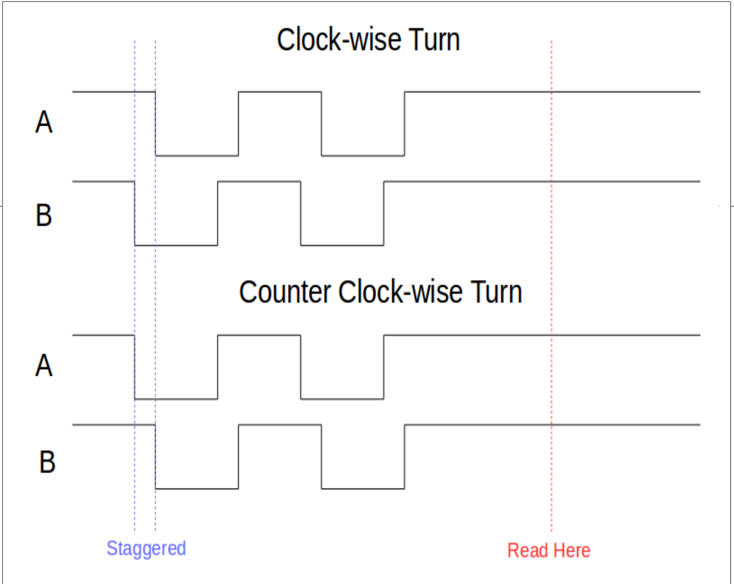

Figure 4: Debouncing Channels

2.4 Project Results and Demonstration

Our project set out to control the PWM using the rotary shaft encoder, send the PWM signal out of the J1 Header pin and control a DC Motor. Seeing as we were forced to choose between continuing to pursue the rotary shaft encoder at the risk of missing our presentation time or setting the switches to control the duty-cycle and making our presentation time with a high grade, as the evening came to a close we were forced to accept defeat relative to the rotary shaft encoder and accept a 100% grade rather than secure the extra credit related to our extra ambitions. It was disappointing to not complete the extra

Duty-Cycle	Integer Values (ONN, OFF)	Binary Values	
		ONN	OFF
100	5000, 0	0001001110001000	0000000000000000
90	4500, 500	0001000110010100	0000000111110100
80	4000, 1000	0000111110100000	0000001111101000
70	3500, 1500	0000110110101100	0000010111011100
60	3000, 2000	0000101110111000	0000011111010000
50	2500, 2500	0000100111000100	0000100111000100
40	2000, 3000	0000011111010000	0000101110111000
30	1500, 3500	0000010111011100	0000110110101100
20	1000, 4000	0000001111101000	0000111110100000
10	500, 4500	0000000111110100	0001000110010100
0	0, 5000	0000000000000000	0001001110001000

Table 2: Duty-Cycle to ONN & OFF conversion

objectives, however given that our project's rotary code worked on other systems, we have reason to believe that we were facing a problem below the surface of our code. Upon using functioning code from Erik's project we discovered an error which directed us to contact tech support, despite manipulating the code this error persisted and we were forced to proceed to another attempt.

By making the judgment call to use the slide-switches and not continue to pursue the rotary shaft encoder helped us present the project in a timely manner. Considering that we were able to control the duty-cycle with ease via the slide-switches, it's not surprising that our project was enjoyable to present. The motor was responsive, the user interface was easy enough to use, and our presentation was a success.

2.5 Summary & Conclusion

In conclusion, our project attempted to tackle several abstract problems in VHDL and motor control and slightly exceeded it's primary objective, PWM control, by allowing the user to alter the duty-cycle with ease. Through our exploration of other interfaces such as the rotary shaft encoder, pushbuttons, and external header pins we expanded upon our existing knowledge of the Spartan 3E starter board to learn new interfacing techniques. Additionally, we learned how to monitor AC signals through the use of the Oscilloscope. Though our project faced several terminal errors in terms of control, we managed to produce tight, functioning VHDL code capable of controlling PWM signals.

In conclusion, our project exceeded it's base-line objectives by allowing the user to alter the PWM duty-cycle with ease. We learned how to solve abstract problems using VHDL, conduct research, and learn new components as well as new equipment. Through this project we expanded our knowledge of digital systems.

2.6 Works Cited

Antiraid. Re: VHDL rotary encoder interface. All about Circuits, 15 February 2011. Web. 26 November 2015. <http://forum.allaboutcircuits.com/threads/vhdl-rotary-encoder-interface.23344/>

Jayramm, Nageswaran. VHDL Reference Guide. University of California Irvine Donald Bren School of Information and Computer Sciences, 2009. Web. 3 December 2015. <https://www.ics.uci.edu/~jmoorkan/vhdlref/>

NTS Press. "NI myRIO: Rotary (quadrature) encoder." On-line video clip. *Youtube*. Youtube, 5 Aug 2013. Web. 5 Dec 2015. <https://www.youtube.com/watch?v=CpwGXZX-5Ug>

Scott Harden. "Rotary Shaft Encoder on AVR for simple menu driven microcontroller menu navigation." On-line video clip. *Youtube*. Youtube, 28 Oct 2013. Web. 5 Dec 2015. <https://www.youtube.com/watch?v=DREGVc00FY8>

Special thanks to Doctor Suruz Miah, Erik Guetz, Karimoon Shaik, Jonathon Jellison, and Morgan Fields for their assistance, insight, and advice throughout our design process.