5-2014

Implementation of prediction algorithms to reduce latency in control of switching converters

Brent Bell
University of Arkansas, Fayetteville

This thesis is approved.

Thesis Advisor:

[illegible]

Thesis Committee:

____

____

Implementation of Prediction Algorithms to Reduce Latency in Control of Switching Converters

An Undergraduate Honors College Thesis

in the

Department of Electrical Engineering

College of Engineering

University of Arkansas

Fayetteville, AR

by

Brent Austin Bell

ABSTRACT

Using pulse-width modulation (PWM) to control switching converters such as a dual active bridge introduces latency into a system [1]. Latency in a switching converter decreases the effective bandwidth that the converter can be switched at and for the case of a closed loop control system decreases the effective closed loop system stability. This delay can be attributed to sampling of the control signal and calculation time for the PWM. If a prediction scheme is implemented in this calculation, the next switching time could be predicted to remove any latency. There are effectively two steps in order to achieve this prediction: 1.) analog to digital conversion of the control signal; 2.) prediction of next point of interest in the control signal [1]. In this paper, several different methods of prediction are compared in order to determine the relative effectiveness of different prediction techniques. Prediction of future data points of the control signal can reduce latency in PWM creation and thereby improve the stability and/or control bandwidth of the system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

A. SWITCHING CONVERTERS

Switching converters such as a dual active bridge (DAB) convert one voltage or current to another voltage or current. In the case of a dual active bridge, this voltage can be stepped up or down in order to deliver power in either direction [2]. A DAB can be implemented in situations where energy is needed to flow in either direction such as in batteries or other storage devices that both take in and deliver power. On either side of the DAB is an H-bridge. The side where the power is flowing from is known as the active side and uses the H-bridge to create a square wave that induces a voltage on the secondary winding of the transformer. This voltage is then rectified on the other side of the DAB resulting in an average voltage at the output. In a DAB, this can occur in either direction making this switching converter topology excellent for many applications including energy storage. The phase shift between the switching of the two H-bridges determines the magnitude and direction of the power flow in the converter. Figure 1 shows a typical schematic for a Dual-Active Bridge.

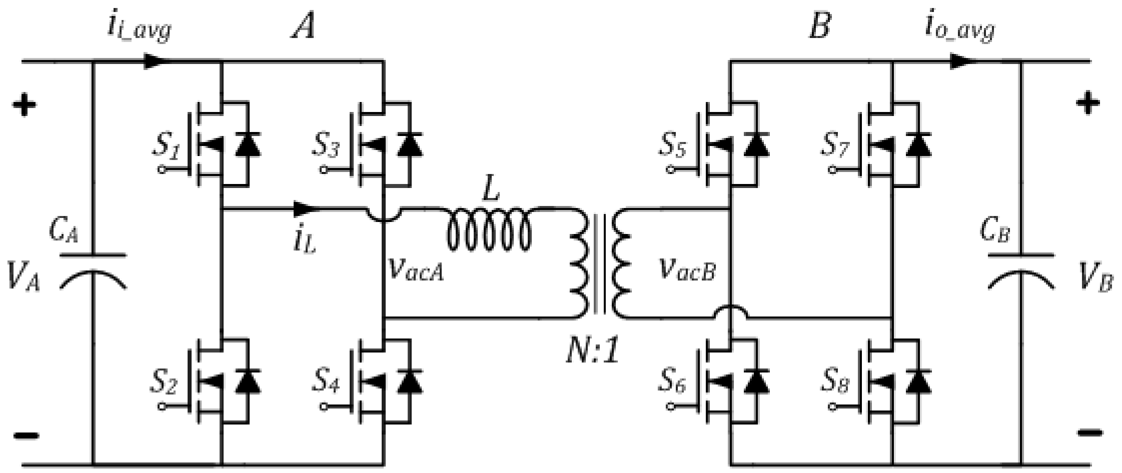

Figure 1: Schematic of Typical Dual Active Bridge [3].

With all switching converters, there are both static and dynamic power losses. Static losses occur when the converter is not switching through leakage current in the switches as well as any other voltage drops in the circuit. Dynamic losses in a DAB occur during switching and have three major contributors: turn-on losses, turn-off losses, and shoot-through current. Transistors do not turn on or off instantly and during the turn-on and turn-off stages power is lost. In the case of MOSFETs, these losses can be attributed mostly to the charging and discharging of the gate capacitance [2]. The other main cause of power losses in a DAB as well as other converter topologies is shoot-through current. When both the upper and lower transistors in an H-bridge being used to control a DAB are on, current is allowed to pass straight from Vdd to Gnd. This is known as shoot-though current and is usually avoided by adding a "dead time" where both transistors are off in order to prevent the case of both transistors being on at the same time.

Dynamic power dissipation increases linearly with switching frequency; therefore, as the demand for higher frequency switching converters goes up, so does the demand for more efficient switching techniques. One technique that reduces dynamic power losses in DABs as well as many other switching converter topologies is "soft switching." One example of soft switching and perhaps the most widely used is zero-voltage switching (ZVS). ZVS can be used in order to mitigate power losses in the switch on time for the high side switch in a switching converter [2]. This is done by switching the high side transistor when the voltage across it is at or near zero. In order to perform ZVS, very accurate driver circuitry must be designed.

B. PULSE WIDTH MODULATION

Control of switching converters is most often achieved through the use of pulse-width modulation (PWM). There are two main types of PWM: fixed-frequency and variable-frequency PWM. As the name suggests, fixed-frequency PWM is achieved by varying the duty cycle of the square wave while maintaining a constant frequency. This is the most common modulation technique due to the predictability of electromagnetic interference (EMI) and its ease of implementation. The most common method for creating this PWM scheme is comparing the signal to be modulated (often a sine wave) with a higher frequency triangular or sawtooth waveform acting as a carrier signal.

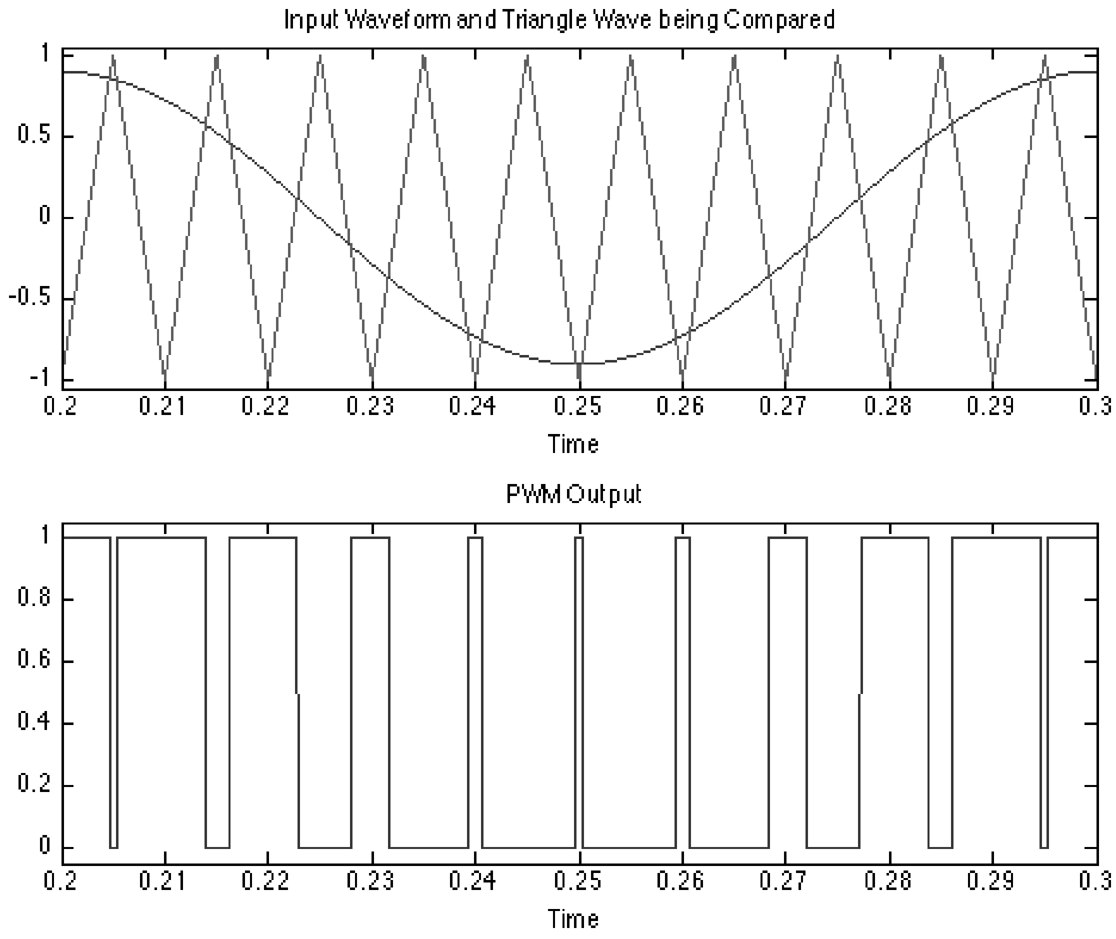

Figure 2: Creation of PWM Signal from Sinusoidal Waveform [4].

Figure 2 shows how a signal can be compared with a triangular signal to create a PWM waveform. When the input waveform is higher than the sawtooth waveform, the PWM signal is high; whereas, when the input waveform is lower than the sawtooth waveform, the PWM signal is low. Increasing the frequency of the sawtooth or triangular signal increases the resolution and accuracy of the PWM signals encoding of the original signal; however, this also increases the switching frequency leading to higher dynamic power dissipation.

The first component in digital PWM creation is an analog-to-digital converter. This takes the input signal which is analog and converts it to a digital number that can be used in calculations. Next, the digital values of the input can be compared with a lookup table for a sawtooth or triangular waveform in order to generate the PWM signal. PWM can also be achieved through the use of analog components, but that is not explored in this paper.

The process of creating the PWM control signal for a switching converter introduces latency into the system. The analog-to-digital converter samples and holds the value of the input signal and then converts it into a digital value. This introduces a delay between the input signal and the PWM signal. The calculation time for the PWM signal also introduces delay. This delay between the input and the PWM signal creates an unwanted phase shift between the input and the output of the system. Also, when accurate control is needed for the case of ZVS these timing errors, can add to the power consumption significantly.

C. PREDICTION

In order to compensate for these delays, future values of the analog input can be predicted and used for the calculation of the PWM signal. Before that is done, the total average delay time between the input signal and the PWM signal needs to be calculated. The delay time can be used to determine how early a point should be predicted in order to minimize the delay. This will improve the achievable control bandwidth of the system or the closed loop system stability [1].

One prediction method is to create a linear model of the input signal based on previous inputs. This is known as linear prediction, a mathematical operation where future values of a discrete time signal can be estimated as a linear function of previous input samples [5]. This can be done in many ways ranging from simple slope calculations to linear predictive coding (LPC) which is discussed later. Linear calculations can usually be performed very quickly. This can significantly decrease the delay time of the system if implemented correctly.

II. RELATED WORK

A. PREDICTION VS RESONANT CIRCUITRY

Previous work has been done in the area of prediction algorithms, and control of switching converters including dual active bridges; however, more research needs to be done in the area of prediction in order to reduce latency in the control of switching converters. Many techniques of prediction that have been researched in the past require a comprehensive model of the system in order to reduce latency. In the past, zero-voltage switching has primarily been done through the use of resonant networks. These networks have several disadvantages; they add to the complexity of analysis, have poor efficiency at light load, and they can optimize performance at one operating point only, but not with a wide range of input voltages or load variations [6]. In order to create a PWM signal that can perform ZVS, it is very important that there is no latency between the input signal and the PWM signal. Any latency will affect the power consumption of the system drastically, especially for high switching frequencies.

B. CURRENT-MODE PREDICTION

Some work has been done in the area of predictive current control [7]. This prediction technique was achieved by ensuring the inductor current follows the reference input current. While there are several advantages to current-mode control, one major disadvantage is that it becomes unstable for duty cycles greater than 50% [2]. For this reason, the designer must ensure that this case never occurs. This predictive control scheme also has specific equations that cannot be applied directly to other converter topologies because it requires a model of the switching converter. It can be adjusted for other simple switching converters, but as the converters become more advanced such as the case of a DAB, the equations for predictive current control become more advanced. The proposed methods do not require a model of the switching converter in order to decrease the latency.

III. IMPLEMENTATION

A. SINGLE SLOPE PREDICTION

The most simple prediction scheme for predicting the future value of the input signal is calculating the slope of the previous two points and using that slope to solve a linear function for the next point. The equation of the proposed single slope prediction with a constant sampling frequency is

$$x(n) = 2x(n-1) - x(n-2)$$

This is a very quick and inexpensive; however the higher the frequency of the input signal, the more error that is introduced in this model. For example, if the input signal is a sinusoid, the slope is not constant. If there is very little delay in the system compared to the frequency of the input signal, a high sampling frequency can be used and this prediction method could be feasible.

In order to quickly simulate and test different prediction techniques for control of switching converters, MATLAB/Simulink can be used for simulation. Simulink also has the capability of exporting hardware description language (HDL) through its HDL Coder. This allows a quick transition from simulations using Simulink to testing on a field programmable gate array (FPGA). For generation of HDL from Simulink models, certain HDL compatible blocks must be used. This restricts the available Simulink library to blocks that are easier to implement on the FPGA; however, it also removes some of the powerful functions that are available in Simulink.

This prediction technique only requires a few blocks to calculate the predicted value. Delays are added in order to take time to sample two data points. Then the slope of the two data points is calculated, multiplied by the time interval, and added to the most recent point. This results in the next point being predicted before it is calculated by the ADC. This technique can be improved, however, by taking in more data points to create a better linear model of the system. The sampling frequency of the ADC would need to be determined by the delay created by PWM creation. So, the point is predicted so that the PWM calculations can be completed in time to completely eliminate the delays in PWM creation. Figure 3 shows the Simulink model for the 2 and 3 point prediction algorithms that were simulated in order to determine the accuracy of signal prediction.

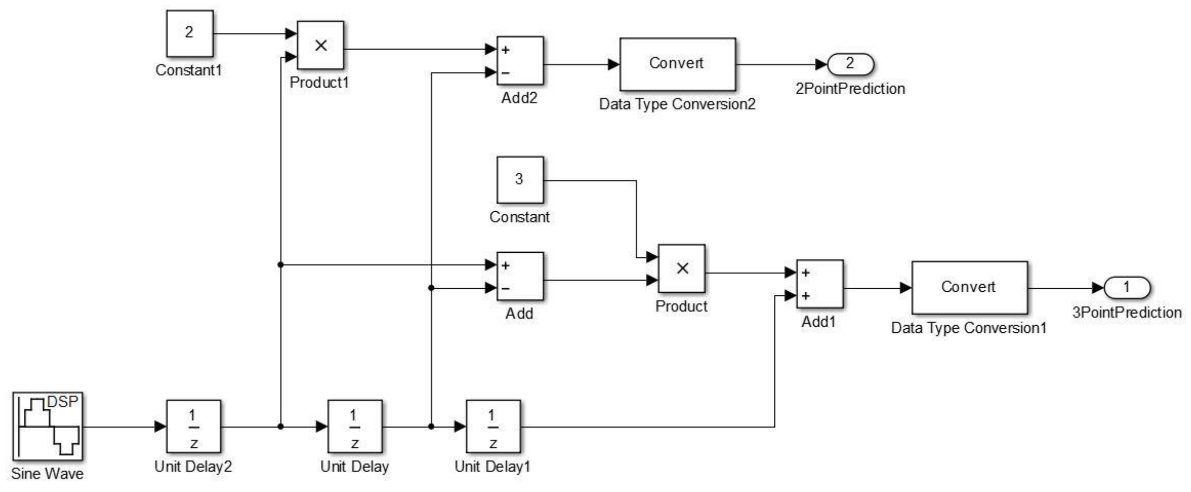

Figure 3. Simulink Model for 2-point and 3-point Prediction.

B. TWO SLOPE PREDICTION

If a prediction is made off of the previous three samples as opposed to the previous two samples, two slopes can be calculated in order to predict more accurately what the slope will be between the last measured point and the point to be predicted. In this model, the first slope calculated is subtracted from the most recent slope in order to calculate the slope of interest. For a constant sampling rate, this can be represented by the equation

$$x(n) = 3\big(x(n-1) - x(n-2)\big) + x(n-3).$$

This model does not assume a constant slope like the previous model but instead assumes a constant rate of change of slope. Depending on the frequency of the input signal this can reduce the error of the prediction from the single slope model by more than half. If the frequency of the input is high enough relative to the sampling frequency, this method can generate more error than the prediction with just a single slope in some situations. However, as long as the sampling frequency is significantly greater than the frequency of the analog input signal, the error between the predicted value and the actual value will be less using this method.

C. LINEAR PREDICTIVE CODING

Linear predictive coding is a more advanced prediction technique that is often used in speech modeling, but also has applications in other areas where values are to be predicted. This form of linear prediction is represented by

$$\hat{x}(n) = \sum_{i=1}^{p} a_i x(n-i)$$

where $\hat{x}(n)$ is the predicted value, $x(n-i)$ are the previously observed values and a_i is the predictor coefficients [5].

The most common way of calculating the predictor coefficients is known as the autocorrelation method. This method assumes that the error created in prediction is to be minimized over the infinite duration $-\infty < n < \infty$. In this method, the autocorrelation matrix of the input signal is calculated in order to find the predictor coefficients. The coefficients are found by solving the matrix equation $Ra = -r$, where R is the autocorrelation matrix, a is the coefficient vector and r is the autocorrelation vector $r_j = R(j)$, $1 < j < p$.

For example:

$$\begin{bmatrix} R_{xx}(0) & R_{xx}(1) & R_{xx}(2) \\ R_{xx}(1) & R_{xx}(0) & R_{xx}(1) \\ R_{xx}(2) & R_{xx}(1) & R_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = -\begin{bmatrix} R_{xx}(1) \\ R_{xx}(2) \\ R_{xx}(3) \end{bmatrix}$$

This model is very computationally expensive to implement; however, there are a few algorithms that can increase the efficiency of solving for the parameters. For example, because the autocorrelation matrix is both symmetric and a Toeplitz matrix special calculation methods can be used other than the typical Gauss algorithm. One method of solving this is known as the Levinson-Durbin Recursion. This increases the coefficient calculation speed from $\Theta(n^3)$ to $\Theta(n^2)$. This is extremely important for systems with large numbers of samples [5].

D. TESTING

The first two algorithms were implemented in Simulink in order to simulate the prediction schemes and confirm that they work as intended. This also allows use of the HDL Coder for a quick transition to hardware for actual testing of the prediction algorithm. This prediction algorithm would be part of a larger control system that calculates the control signals based on the predicted values of the input waveform. The sampling frequency of the analog-to-digital converter would need to be changed along with the sampling frequency of the Simulink code in order to match the delay caused in generation of the PWM signal. The more complicated linear prediction algorithm was not implemented in Simulink because of its complexity. It was however modeled in Matlab in order to get a broader understanding of how LPC works.

The linear predictive coding algorithm was also implemented on an Arduino prototyping microcontroller in order to test its capability of controlling a DAB or other switching converter. If the Arduino could be used for testing control of switching converters there would be several advantages. Due to the fact that Arduinos are easy to program and have a built in ADC. Creating the control signals and making changes to them would be very simple.

IV. RESULTS

A. SLOPE BASED PREDICTION SCHEMES

Simulink provides an easy transition from simulation of a circuit to actual implementation on an FPGA through the use of the HDL coder. Table 1 shows the results and percent error for the simulation of the 2-point prediction scheme and the 3-point prediction scheme. These simulations were done using Simulink blocks that are HDL compatible, so that for future testing the diagram can be used to generate HDL using the software for a fast transition between simulation and testing.

Table 1. Simulink Simulations for 2-point and 3-point Prediction.

Point	Actual Value	2 Point Predicted	% Error	3 Point Predicted	% Error
X_4	0.482	0.488	1.21	0.484	0.391
X_5	0.588	0.595	1.29	0.590	0.305
X_6	0.685	0.694	1.35	0.686	0.244
X_7	0.771	0.781	1.40	0.772	0.198
X_8	0.844	0.856	1.44	0.846	0.161
X_9	0.905	0.918	1.47	0.906	0.129
X_{10}	0.951	0.965	1.50	0.952	0.100

From this data it is clear that the 3-point prediction scheme provides dramatic decrease in the error of the predicted signal. Figures 4 and 6 show the actual vs predicted values for the signal. Both are fairly accurate; however the percent error of the 3-point prediction values is much lower than the 2-point prediction values. Figure 5 and Figure 7 show a graph of the error for the 2-point prediction and 3-point prediction respectively. While the values for the 2-point prediction scheme are very close to the expected values, they are not as accurate as the values obtained for the 3-point prediction scheme.

It is obvious that the second prediction scheme produces better results than the first. This is as expected because a more comprehensive model of the signal can be created with the addition of each data point. The error of the second prediction algorithm is as low as 0.1% for the simulation shown in Table 1.

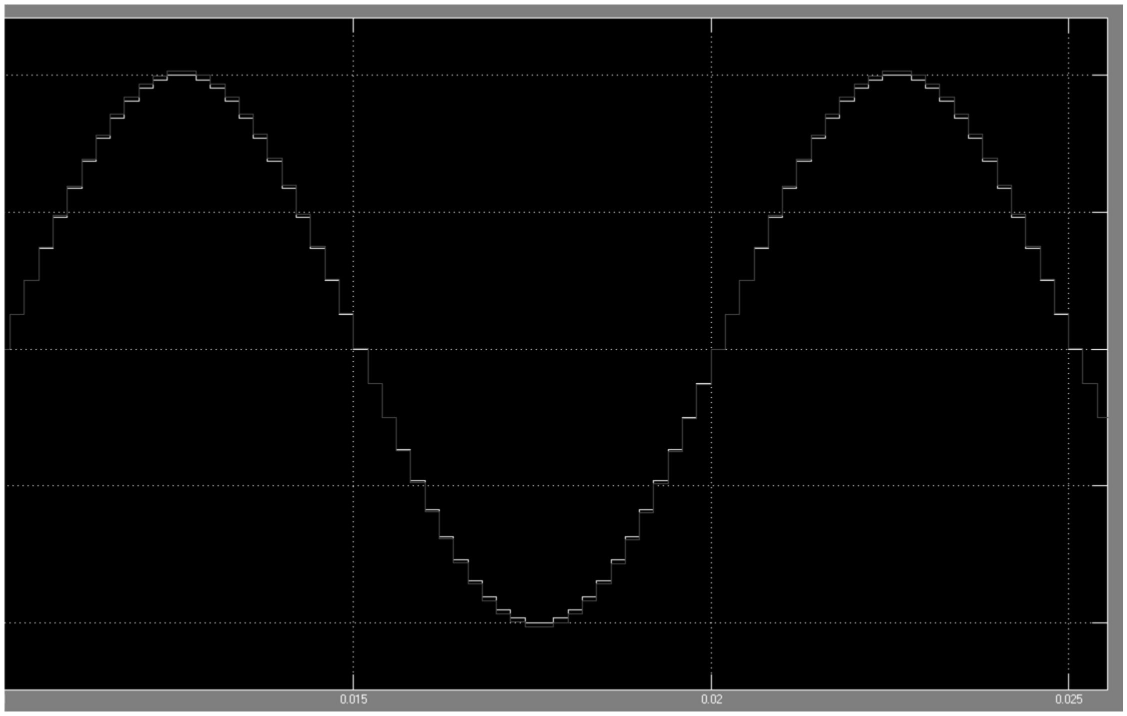

Figure 4. Actual and Predicted values for 2-point prediction.

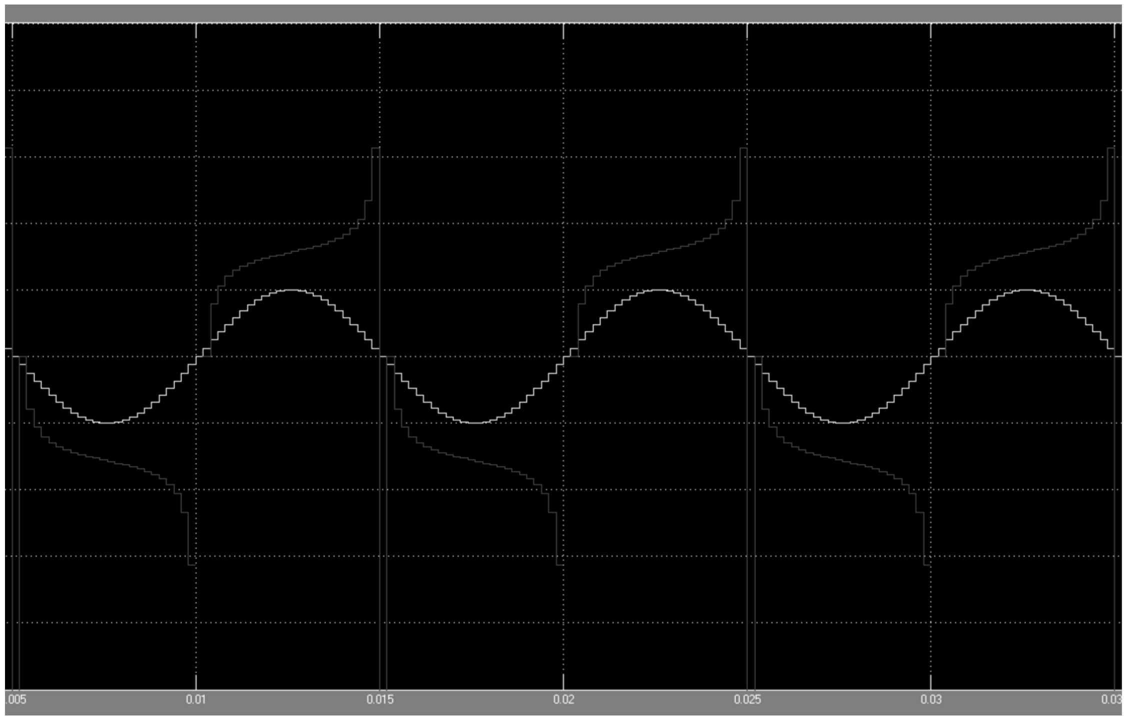

Figure 5. Input sine wave and percent error for 2-point prediction.

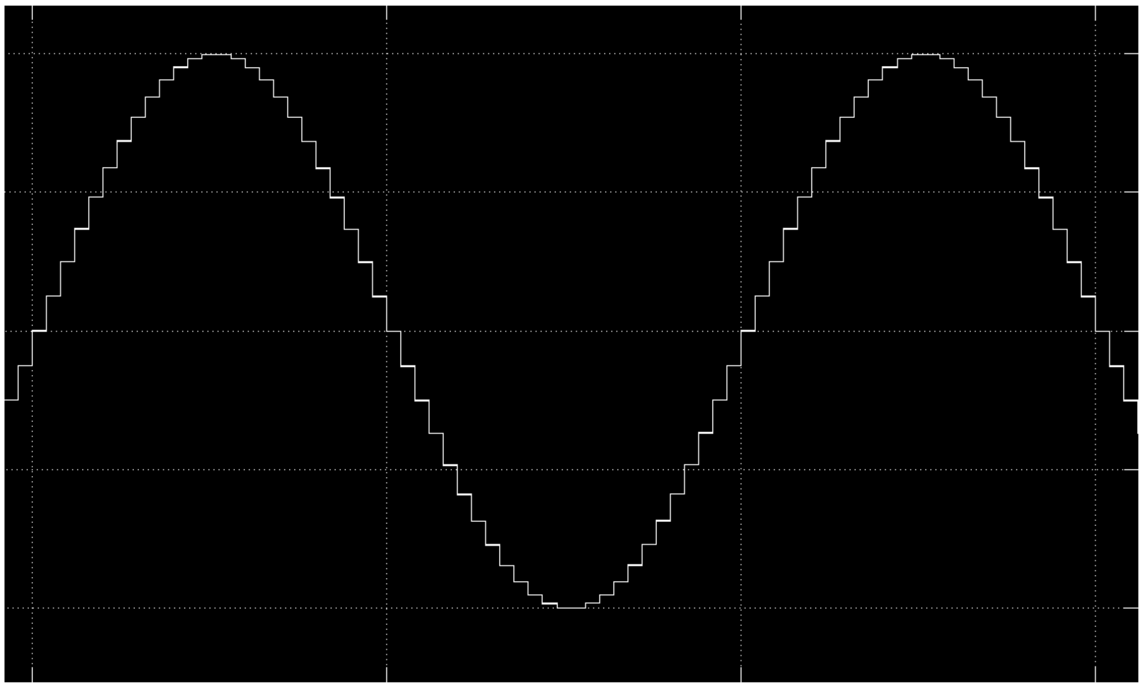

Figure 6. Actual and Predicted values for 3-point prediction.

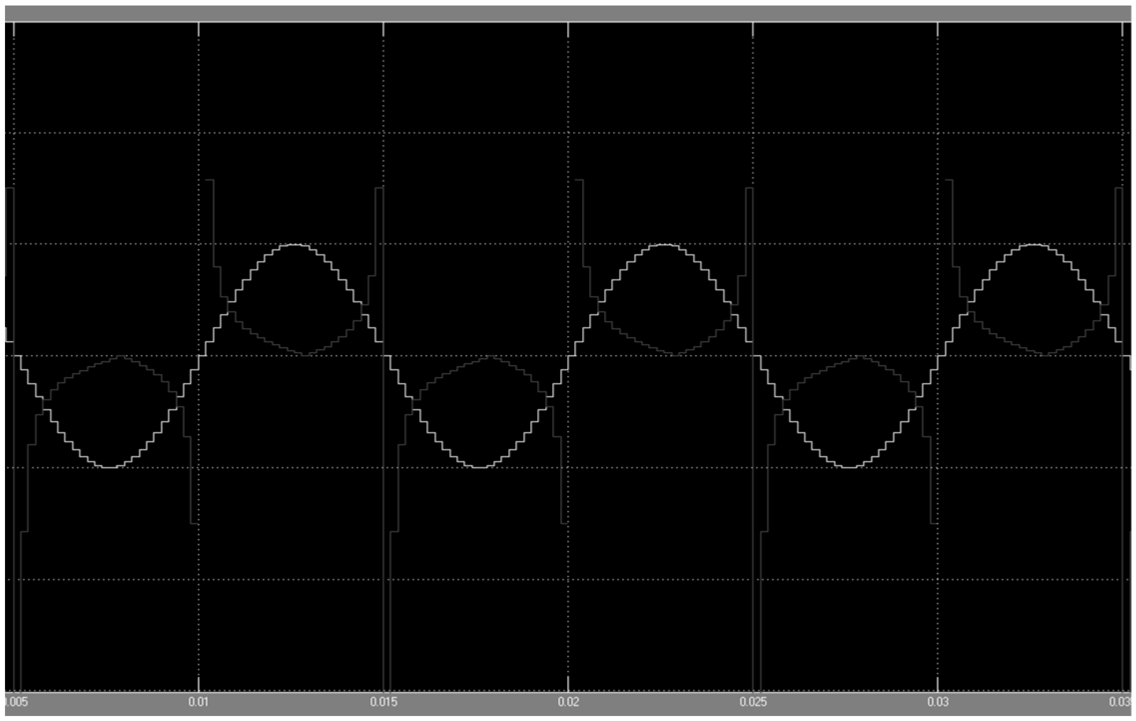

Figure 7. Input sine wave and percent error for 3-point prediction.

B. LINEAR PREDICTION SCHEME

The linear prediction scheme was implemented on the Arduino microcontroller in order to determine the effectiveness of the Arduino in testing control of switching converters. Although the Arduino has a built in analog-to-digital converter, its sampling rate is not high enough for frequencies as high as switching converters require. In addition to this limitation, the calculation time required for even simple LPC analysis was much greater than would be allowed by the switching converter. The calculation of the linear predictor coefficients using Gaussian inversion to solve the systems of equations took about 1 ms to complete. At this speed, the frequency would be limited to significantly less than 1 kHz. This is unacceptable for most applications, but it could be made more efficient by solving the equations beforehand or using the recursive Levinson algorithm.

Matlab also has a built in calculation for linear predictor coefficients that allows for quick simulation of the linear prediction algorithm; however, the corresponding block in Simulink is not HDL compatible and cannot be used for HDL generation. For this reason the code was simulated in Matlab, but the linear prediction algorithm was not implemented in Simulink for HDL generation. Figure 9 shows the accuracy of the prediction when one half of the cycle of the sine wave is taken in order to find the prediction coefficients. As the data points decrease, the predicted values for the signal decrease in accuracy. Linear prediction is good for a large number of data points, but not when only a few samples are available. Figure 10 shows the predicted signal with only one quarter of the sine wave period used in calculation of the prediction coefficients. This reduction to only 5 input points dramatically increases the error of the predicted signal.

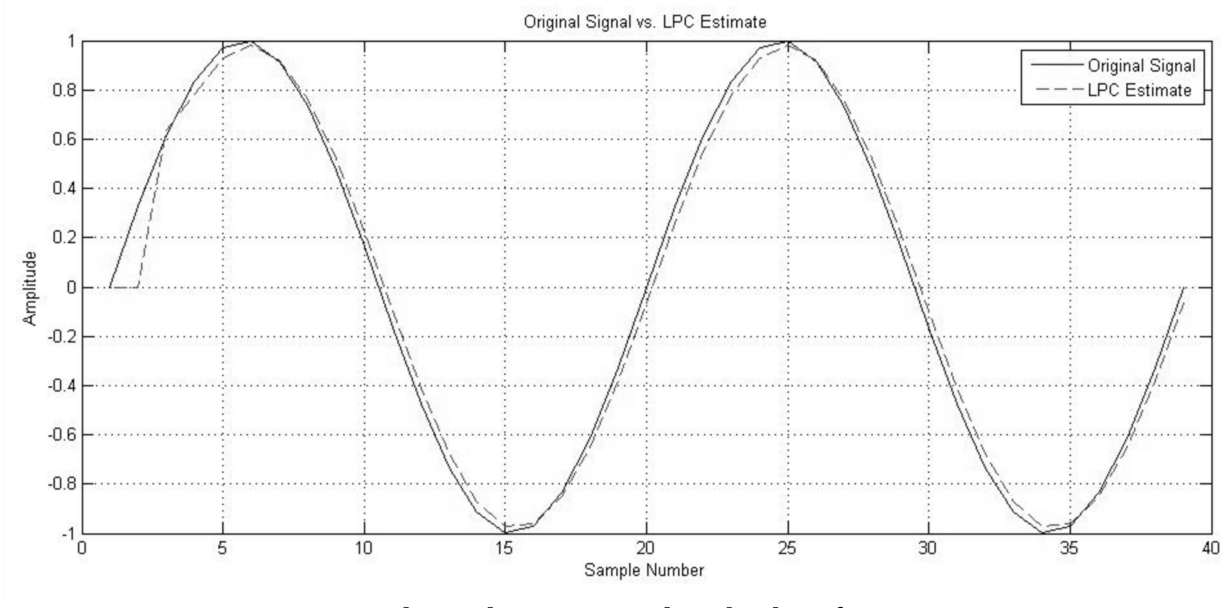

Figure 8. Original signal vs. LPC predicted values for 10 input points.

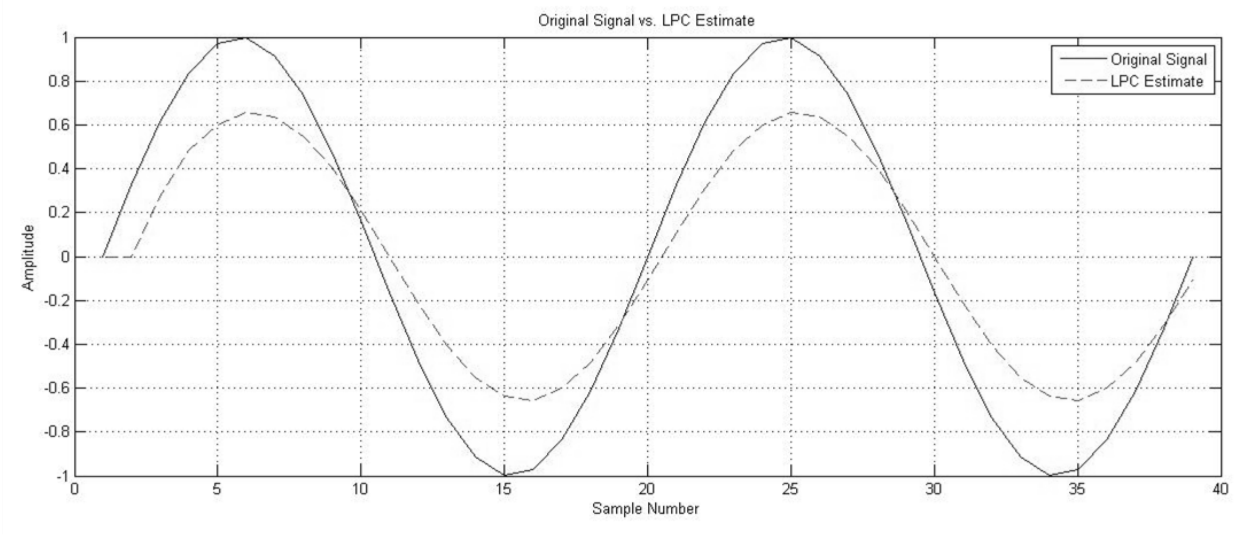

Figure 9. Original Signal vs. LPC predicted values for 5 input points.

V. FURTHER WORK

A. FPGA IMPLEMENTATION

The proposed prediction method worked in simulations to accurately predict future values of the voltage input waveform. Further work could be done in the case of the slope prediction methods by implementing them on an FPGA in tandem with an analog-to-digital converter in order to create a comprehensive control system for a DAB or other switching converter. Because these were done in Simulink, the HDL coder can be used in order to quickly transition to the FPGA development board in order to test the prediction scheme on hardware.

To accomplish this, more Simulink blocks would be required in order to control the ADC that must be connected to the FPGA. Also, the prediction would need to be compounded with a control algorithm to calculate the value of the PWM signal. Delays of different converters could also be explored in order to determine the optimum sampling frequency for any given converter in order to minimize the latency of the system.

C. IMPROVEMENT TO PREDICTION SCHEMES

In the case of the LPC models more work could be done in the testing for different calculation methods of the predictor coefficients. This would be helpful in determining the best tradeoff between number of points used in calculation, number of predictor coefficients, and the accuracy of the predicted waveform. Because the calculations in LPC are so much more complex it was not implemented in Simulink with HDL compatible blocks. This could be an area of future work as well in order to have a Simulink model that could be coded into HDL easily.

Another area of future research would be using more efficient calculation techniques to solve the linear system of equations to find the autocorrelation coefficients. A more advanced algorithm that takes advantage of the symmetry of the Toeplitz matrix is the Levinson-Durbin algorithm. This is a recursive algorithm that can solve the matrices as calculations of a second order speed as opposed to a third order speed. As the sample size is increased, the increase in speed would become more and more significant. For linear prediction to be accurate, a large sample size is needed.

By calculating the slopes of the input signal and making predictions off of the calculations a linear model of the signal of interest is being created. This model could be improved with the addition of more points into the system. This data could be used to not only calculate first derivatives of the data, but second derivatives in order to find the rate of change of the slope. This could give a more accurate model of the system in order to reduce the error even further. This in turn could allow the prediction of points further than the next point in the sequence. Increasing the prediction range would allow for lower sampling frequencies with the same amount of latency reduction as long as the prediction algorithm is accurate enough for these calculations. The 2-point prediction algorithm is not accurate enough for prediction of 2 points in the future. The linear prediction algorithm could achieve this; however, the calculation time could be prohibitive if the FPGA is not fast enough to perform the calculations.

VI. SUMMARY

Many prediction methods are available for implementation in control circuitry for switching converters. These methods can reduce the latency in PWM creation and therefore, increase the control bandwidth for open loop systems or improve the closed-loop stability for closed-loop systems. One method that is very simple to implement and works very well for fast sample rates is the prediction proposed that calculates the predicted value based on the previous 2 slopes. This method can successfully predict the next value with error much smaller than 1%. Linear prediction, while more advanced and more powerful in certain situations requires large computational resources and a larger number of data points in order to create an accurate linear model of the system. More work needs to be done in the area of linear prediction to implement faster algorithms in order to speed up calculation time.

In order to use prediction algorithms to reduce the amount of latency in PWM creation the delay caused by both the ADC and the calculation of the PWM signal itself needs to be determined. After the average delay is found, it can be corrected by predicting the point that is of interest, so that the delay can be mitigated by early calculation of the PWM signal.

VII. BIBLIOGRAPHY

[1] Thomas Nussbaumer, "Comparison of Prediction Techniques to Compensate Time Delays Caused by Digital Control of a Three-Phase Buck-Type PWM Rectifier System," in IEEE Transactions on Industrial Electronics, 2008, pp. 791-99.

[2] Simon Ang and Alejandro Oliva, Power-Switching Converters, 3rd ed. Boca Raton: CRC Press, 2011.

[3] (2014, April) TI E2E Community. [Online]. http://e2e.ti.com/group/launchyourdesign/m/c2000microcontrollerprojects/665411.aspx

[4] Patrick Dear, Edward Szoka, and Hanna Lin. (2013) HRTF Cornell. [Online]. http://people.ece.cornell.edu/land/courses/ece5030/FinalProjects/s2013/pmd68_ecs227_hl577/pmd68_ecs227_hl577/

[5] John Makhoul, "Linear Prediction: A Tutorial Review," in Proceedings of the IEEE, 1975, pp. 561-80.

[6] Robert W. Erickson, Fundamentals of Power Electronics, 2nd ed.: Springer, 2001.

[7] Jingquan Chen, Aleksandar Prodić, Robert Erickson, and Dragan Maksimović, "Predictive Digital Current Programmed Control," in IEEE Transactions on Power Electronics, 2003, pp. 411-419.

APPENDIX

A. LPC COEFFICIENTS C CODE

```
void parameters(int data[], float coeff[]){
  //Create and initialize autocorrelation array
  int r[sizeof(data)];
  for (int i = 0; i < sizeof(r); i++){
    r[i] = 0;
  }
  //Calculate Autocorrelation
  for (int i = 0; i < sizeof(r); i++) {
    for (int j = i; j < sizeof(r); j++){
      r[abs(j - i)] += data[i] * data[j];
    }
  }
  //Create Toeplitz matrix
  float matrix[sizeof(r) - 1][sizeof(r) - 1];
  for (int i = 0; i < sizeof(r) - 1; i++){
    for (int j = 0; j < sizeof(r) - 1; j++){
      matrix[i][j] = r[abs(j - i)];
    }
  }
  float tmp[sizeof(r) - 1];
  for (int i = 0; i < sizeof(r) - 1; i++){
    tmp[i] = -r[i + 1];
  }
  //Solve
  Matrix.Invert((float*)matrix, (sizeof(r) -
1));
  Matrix.Multiply((float*)matrix, (float*)tmp,
(sizeof(r) - 1), sizeof(r) - 1, 1, (float*)coeff);
}
```

*The matix functions are from a library called MatrixMath created by Charlie Matlock found on the Arduino website.

B. LPC COEFFICIENTS MATLAB CODE

```
t = [0:2*pi/19:4*pi];

x = sin(t);

a = lpc(x,3)

est_x = filter([0 -a(2:end)],1,x);  % Estimated signal

e = x - est_x;                      % Prediction error

[acs,lags] = xcorr(e,'coeff');      % ACS of prediction error

%   Compare the predicted signal to the original signal

plot(t,x,t,est_x,'--');

title('Original Signal vs. LPC Estimate');

xlabel('Sample Number'); ylabel('Amplitude'); grid;

legend('Original Signal','LPC Estimate')

%   Look at the autocorrelation of the prediction error.

figure; plot(lags,acs);

title('Autocorrelation of the Prediction Error');

xlabel('Lags'); ylabel('Normalized Value'); grid;
```