A DIGITAL CONTROLLER FOR A UNITY POWER FACTOR CONVERTER

by

AHMED MITWALLI

S.B., Electrical Engineering and Computer Science
Massachusetts Institute of Technology
(June 1991)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

Massachusetts Institute of Technology

January 1993

Signature of Author [illegible]
Department of Electrical Engineering and Computer Science
January 15, 1993

Certified by [illegible]
George C. Verghese
Professor of Electrical Engineering
Thesis Supervisor

Certified by [illegible]
Steven B. Leeb
Assistant Professor of Electrical Engineering
Thesis Supervisor

Accepted by [illegible]
Campbell L. Searle
Chairman, Department Committee on Graduate Students

A Digital Controller for a Unity Power Factor Converter

by

Ahmed Mitwalli

Submitted to the Department of Electrical Engineering and Computer Science
on January 15, 1993, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

Digital control is rare in the area of power converters due mainly to cost considerations. As operating requirements of power supplies become more stringent, however, their control poses difficult problems better dealt with using digital rather than analog control. With the falling prices of microcontrollers and the increase in their processing power, microprocessor-based controllers for power converters should soon become practical. In this thesis, a digital controller for a Unity Power Factor AC-DC converter is designed, based on a linear large-signal model of the power supply. A hardware implementation of the design is presented and analyzed, along with simulations of the closed-loop system. Issues in digital control of power systems, such as quantization effects and fixed-point representation of system parameters, are examined in the context of this system. An adaptive controller is then designed and implemented using the same system. The experimental results are then compared with the simulations and used to evaluate the implementation.

Thesis Supervisor: George C. Verghese
Title: Professor of Electrical Engineering

Thesis Supervisor: Steven B. Leeb
Title: Assistant Professor of Electrical Engineering

To my wonderful parents, whose love and support have turned out to be my greatest assets. I owe you more than I can ever repay.

To my beautiful sister, Dalia, who has always stood by my side. I have never met anyone who is more unselfish.

To my loving brother, Mohammed. May you have a bright and successful future.

God bless them all.

Acknowledgements

I had an incredibly stimulating and rewarding experience working on my Master's thesis, and I have many people to thank for it. Mainly, I had the opportunity to work with two exceptional supervisors.

Professor Verghese has to be one of the most brilliant human beings I have ever met. The knowledge and insights I have gained from interacting with him are invaluable. His excellence as a person also extends far beyond his academic achievements and intellectual abilities, and he has provided me with a role model in many aspects of life. Thank you George.

Professor Steve Leeb has the rare combination of a deep understanding of theoretical issues and knowledge of practical matters. He worked closely with me on this project, and his contributions are in every part of it. He was there on the long nights of work, always ready to help, and never out of answers. I had the great opportunity to work with him as a fellow student and friend and then as a supervising faculty member. Thank you Steve.

Professor Aleksandar Stankovic, now at Northeastern University, was yet another exceptional person I had the privilege of working with. It is hard to imagine my last year and a half without his help and encouragement (on and off the basketball court). His intelligence and good nature are of a rare magnitude.

Everyone in LEES deserves some thanks. The friendliness of the place made my experience the best it could have been. Special thanks to Derek, Kamakshi, Haatchitaba, John, Dave, Brett, Jeff, Mary, and of course, Vivian.

Thanks to family and friends. Without my parents, my sister Dalia, and my brother Mohammed, and their constant support from so far away, none of this would have been possible. I also thank my roommates and friends with all my heart.

I would like to acknowledge and thank Digital Equipment Corporation for supporting the project, and in particular Dr. Joseph Thottuvelil, who was a great help and a pleasure to work with.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Objectives

The main goal of this thesis is to explore an implementation of a digital controller for a power electronic circuit. Although digital control techniques are widely used in the area of motor drives, they are rare in other areas of power electronics, mainly because of cost considerations. As operating requirements on power converters become more stringent, however, controlling them becomes a complex problem, better solved with digital rather than analog control. There has been some effort in researching digital control for power supplies [17] [5], but very little has been done in terms of actual implementation of such controllers. This experiment should provide insight into some of the difficulties that arise in realizing such systems as well as answers to these problems.

A Unity Power Factor (UPF) converter was chosen for the implementation. The increasing popularity of this power supply makes it a natural target for research. Moreover, as a high end product in the power supply industry, it is a prime candidate for experimenting with advanced control techniques that are conceivable with a digital controller.

In implementing this controller, several issues are examined. A large-signal sampled data model is used for control design and simulation. This model, as well as the simulation techniques used, are evaluated based on the experimental results. The discrete-time analysis methods, simulation environment, and the development system should provide a taste of what capabilities are available for achieving digital control in power electronics.

Demonstrating the availability of the necessary development tools and the realizability of digital controllers for power supplies is not enough to motivate choosing them in practice, however. Additional features that are not easily implemented with analog circuits, such as parameter estimation, are therefore included in the design. Potential reductions in the size and cost of such controllers as well as possible improvement in performance are explored. Overall, an evaluation of the prototype and a look at product feasibilty are used to determine whether digital control of power supplies may be the direction in which we should be heading.

1.2 The Power Factor

The power factor, k_p, of a two-terminal network is defined in [3] as the ratio of the average power measured at the terminals to the product of the rms values of the terminal voltage and current.

$$k_p = \frac{< p(t) >}{V_{rms} Irms} = \frac{< p(t) >}{S} \tag{1.1}$$

$< p(t) >$ is known as real power and is measured in watts, and S is known as apparent power and is measured in volt-amperes. The optimal value for this ratio is unity, indicating that the source is delivering all of its power to the load.

In an AC system, the voltage and current are typically periodic waveforms in time, and the measured power is the product of these waveforms over a cycle [18]. In order for the product of the rms values of the voltage and current to equal the measured power and yield a unity power factor, the two waveforms need to coincide in shape and phase. To understand this better, we consider as an example an ac system with a single harmonic of voltage and a distorted current. The standard utility voltage supply is nominally a sinusoidal waveform:

$$v(t) = V_s \sin \omega t \tag{1.2}$$

Assuming a port with periodic current $i(t)$:

$$i(t) = \sum_{n=0}^{\infty} I_n \sin (n\omega t + \theta_n) = I_1 \sin (\omega t + \theta_1) + \sum_{|n| \neq 1} (I_n \sin n\omega t + \theta_n) \tag{1.3}$$

The average power delivered by this system is:

$$< P >= \frac{1}{T} \int_0^T vidt = V_{srms} I_{1rms} \cos \theta_1 = S k_d k_\theta \tag{1.4}$$

where $k_d = \frac{I_{1rms}}{I_{rms}}$ and $k_\theta = \cos\theta$; k_d is the *distortion factor* and k_θ is the *displacement factor*. The power factor is then defined as:

$$k_p = k_d k_\theta \tag{1.5}$$

If the shapes of the voltage and current waveforms coincide, k_d is unity, and if they are in phase, k_θ is unity. Hence, for a unity power factor, we need two waveforms of the same harmonic content and phase [3].

In the past, low power factors were mainly due to phase displacement, or a non-unity k_θ. Most of the loads that an ac power system had to provide power to were linear, drawing a sinusoidal current at the same frequency as the input voltage sine wave. These loads were usually inductive, causing the current to lag the voltage. The displacement factor was then less than unity, resulting in poor power factors. Power factor correction in such a situation can be simply achieved by adding the appropriate capacitor in shunt with the load. More recently, however, the power factor problem has taken on a different form. Today, electronic devices and power supplies present a highly non-linear load to the AC source, therefore drawing a distorted, non-sinusoidal current from the utility. An example of such a load is the basic configuration of the AC-DC power converter used to supply most computers and other electronic equipment with power, namely a simple rectifier circuit with capacitive filtering. Figure 1-1 shows a diagram of such a circuit and its waveforms.

The line rectifier adds a dc component to the input voltage, and the capacitor input filter extracts it and supplies it to the load. The capacitor is replenished by the rectifier at each half cycle with brief bursts of current at each voltage peak. The resulting current is distorted as shown. The poor power factor here is due to k_d, the distortion factor. The contribution to load power is made by a harmonic component of the current which is in phase with the input voltage, but heavily attenuated due to distortion. Power factors in such setups may be as low as 50% . Power factor correction here involves reshaping the input current waveform, and is a much more difficult problem than adding phase to correct for displacement effects.

1.3 High Power Factor AC-DC Converters

Most AC-DC power converters used today are based on a design similar to the one presented above, with a rectifier and an input filter. The resulting power factor is usually around .6, and the current waveform is distorted with higher harmonics. Most computers require only a few hundred watts and this poor power factor multiplied by

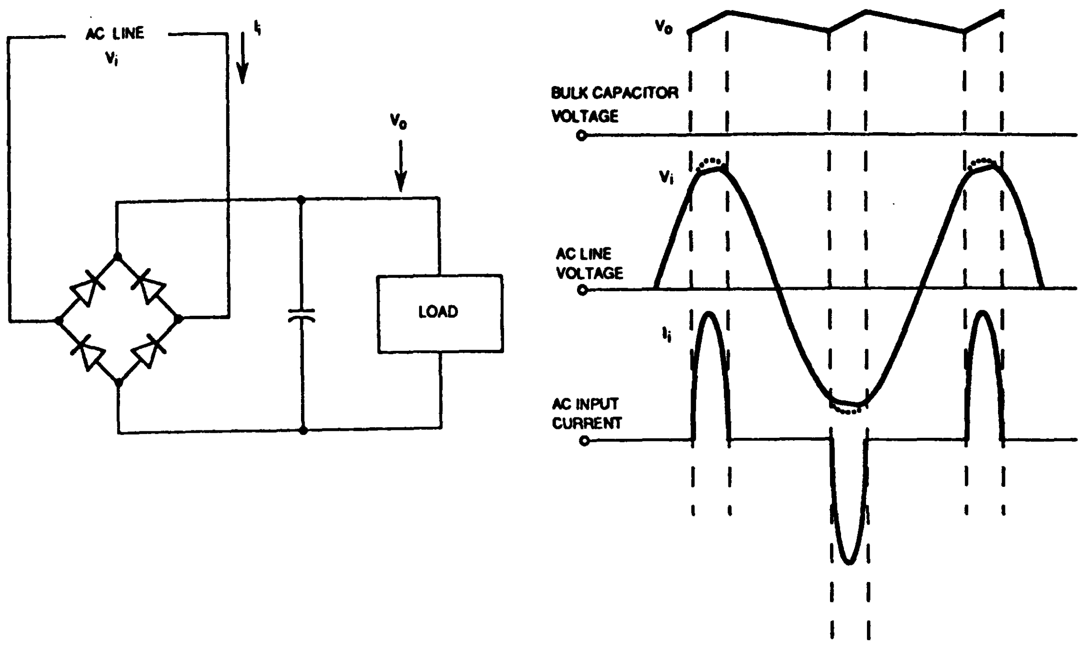


Figure 1-1: Voltage and current in conventional capacitor-filtered rectifiers.[15]

the 1400 or so watts available from a standard outlet provides the necessary power. However, recent trends in computers and electronic systems have resulted in an increasing demand for power factor corrected power converters. The overall number of computers (and other electronic devices that use switch-mode power supplies) has increased dramatically as a result of cost reductions, increasing level of computer literacy, and user-friendly interfaces. This results in more power wasted due to poor power factor, which translates directly into utility losses, since the utility usually charges customers for real power consumed. Moreover, the increasing demand for computing power in workstations has led to an increase in workstation components and therefore in their electrical power requirements. As the power limits of standard electrical outlets are approached, one way to meet these new requirements is to rewire offices with nonstandard outlets. Besides incurring extra cost on the customer, this solution requires rewiring of the outlets every time the workstation is moved [1]. A more practical solution would be to increase the power factor of the power supply.

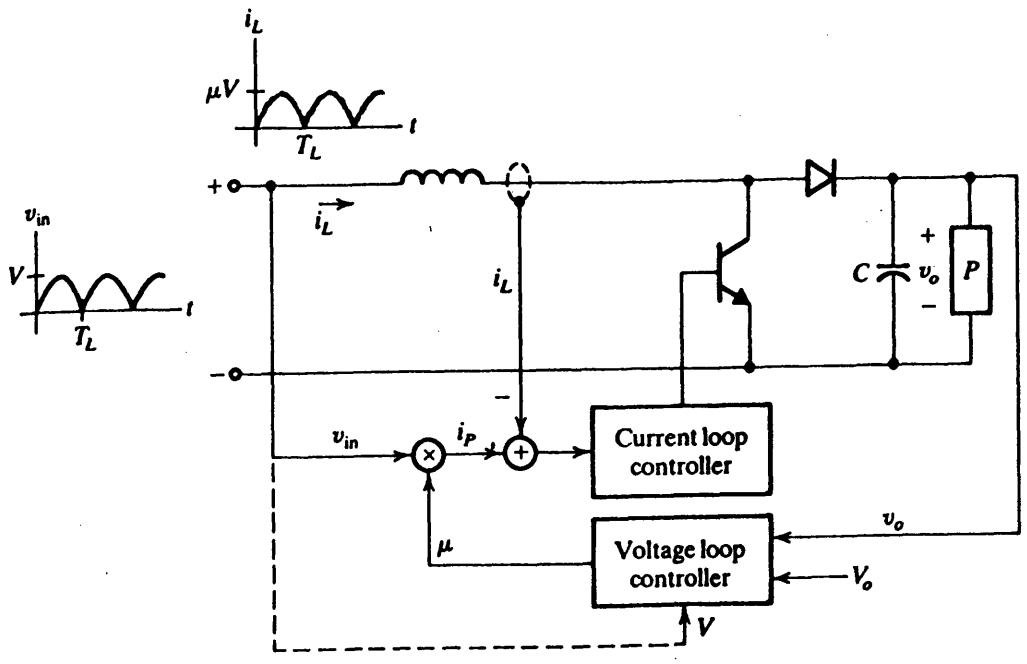
As mentioned earlier, a poor power factor in switched power converters is usually accompanied by distorted current waveforms. The higher harmonics in these waveforms may interfere with nearby instruments, often making it necessary to add extra filtering components. Higher harmonics may also resonate and interfere with the utility's circuit protection devices, causing them to malfunction. Power factor correction through current shaping would essentially eliminate many or all of these higher harmonics. In Europe, more stringent limits on the harmonic content of power supply waveforms are already being legislated. According to the tougher standards recommended by the International Electrotechnical Commission (IEC) 555-2, power supplies rated higher than 300 watts need to incorporate some form of power factor correction [18].

Responding to this increasing need for higher power factors, several systems with power factor correction have been developed. The multiple output Modular Power System with high power utilization [2] provides an 80% power factor to deliver 1200 watts to a workstation from a standard outlet. It does not, however, solve the problem with input current harmonics. More recently, power supplies with almost unity power factors (UPF) have been introduced, providing levels of power near the maximum that can be drawn from the standard outlet. A UPF uses a certain power supply topology with some control scheme that draws a current with the same frequency and shape as the input voltage.

One increasingly popular scheme for achieving this sinusoidal input current uses the circuit in Figure 1-2. The boost converter in the circuit receives as its input the rectified AC waveform, $V|sin\omega t|$. The inner loop controls the source current to the shape and phase of the input voltage by providing a switching sequence for the transistor that forces the inductor current $i_L(t)$ towards a desired current $i_p(t)$, which is in turn made proportional to the input voltage. The outer voltage loop regulates the output voltage v_o to the reference voltage V_o by adjusting the proportionality constant used to generate $i_p(t)$ every line cycle [4]. This allows the system to correct for deviations from the nominal point in the output voltage due to disturbances in the constant load P.

1.4 Modeling of UPF Power Supplies

There are different approaches to modeling UPF power supplies and designing their control loops. We focus here on the boost converter topology presented above. In [1], Williams investigates five different control circuits. They are all based on a small-signal transfer function between $i_p(t)$ and $v_o(t)$. Henze and Mohan [5] use a hysteretic current controller for the inner loop and a digital PI controller for the outer voltage loop. One model presented by [4] and [17] is derived by linearizing a power balance description of the boost converter above with a load resistor in parallel with the constant power load and is shown in Figure 1-3. The linearization is done around the nominal operating point for the output voltage, V_o.



Without the load resistor, a power balance description averaged over half a line cycle yields:

$$\frac{dv_o^2(t)}{dt} = \frac{1}{C}(V^2k(t) - 2P) \tag{1.6}$$

Averaging techniques used to arrive at this and other models will be discussed in more detail in Chapter 2. Linearizing this about the nominal operating point, and using tildes to denote deviations from the nominal, we get the following transform

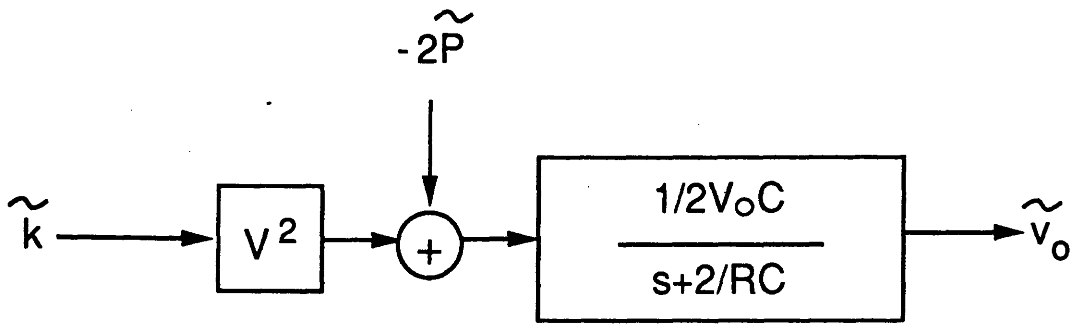


Figure 1-3: Block diagram for a linearized model of a UPF.

domain description (which is the result in Figure 1-3 for the case $R = 0$).

$$\tilde{v}_o = \frac{1}{2sV_oC}(V^2\tilde{k} - 2\tilde{P}) \tag{1.7}$$

The models mentioned above and the one developed here are all derived through some form of linearization. Linearized models are valid only in the vicinity of the nominal operating point and are therefore small-signal models. In Chapter 2 we develop a large signal model that we use for the design of the digital controller. A large signal model is more accurate, and controllers based on it are potentially more robust and reliable than those based on linearized models.

1.5 Digital Control

In the last decade or so there has been a rapid increase in the use of digital controllers in dynamic systems. Digital controllers have become an integral part of the operation of many industrial systems [6]. Originally, they were used as components only in large-scale control systems. However, the evolution of microprocessors and microcomputers that can rapidly perform a variety of control functions has brought digital control into small-scale systems as well.

The ability to "make decisions" (i.e. implement complex, nonlinear, time varying control laws) is one of the main strengths of digital control. Applications that utilize this capability include "intelligent" industrial robots and fuel economy optimization in automobiles. This and other advantages of digital computing over analog computing combine with the decreasing cost and size of microprocessors and microcomputers to produce the current trend toward digital controllers rather than analog controllers [6]. Some of the advantages of digital control are:

- Straightforward data processing allows for more complex calculations and the implementation of modern control schemes.
- A simple change of a program allows for a change in controller characteristics. This flexibility allows the designer to modify the controller at will, as well as implement adaptive control schemes.
- Digital controllers are superior in terms of internal noise, aging, and drift effects.

Some of the disadvantages of digital control are:

- System performance is likely to degrade as sampling and quantizing result in more errors.

- A digital controller that compensates for such degradation involves a more complex design than its analog counterpart.
- A digital controller can be slower and more expensive than an analog controller [6].

Generally, digital control of a continuous time system involves sampling of the control signals, A-D conversion, processing of the digital data using a digital computer, D-A conversion of the digital outputs, and interpolation of the resulting signal. Figure 1-4 illustrates a typical digital control system.

In power electronics, digital control of motors and motor drives has become very popular, [7], [8]. However, the use of digital control in other converter systems is not widespread. Power converter systems place high demands on digital control systems, and digital controllers that meet these high demands are often considered too costly. The systems must be able to handle fast closed-loop control processes with sampling times of 1 ms and below. They must also meet specific time requirements in generating control signals and reacting to process interrupts. A good user interface is required so that changes to the controller may be made easily during planning and testing [9]. In [9], Siebert reviews a new multicomputer control system for all types of converters. The design uses 3 80196 Intel microprocessors and provides a fairly high-level language interface. Very little has been done in terms of actual implementation of digital controllers for power converters, however. A low cost digital controller for DC - AC inverters and DC - DC PWM converters is discussed in [14].

1.6 Motivation

With the increasing need for high power factor converters, UPF power supplies have rapidly gained popularity. It is essential that power supplies for workstations and other computers be robust, reliable, and quick to respond to perturbations in circuit parameters. The linear large signal model introduced in the next chapter leads to a controller that promises quick and reliable correction of changes in the output voltage that result from perturbations in load power or input voltage. Simulations show transients settling down after eight line cycles, but even faster response is possible [4]. Power factors of .97 in steady state are also simulated. Large perturbations are handled well due to the linearity of the model that underlies the controller design. Prior to this project, however, only an analog controller based on the large signal model had been implemented.

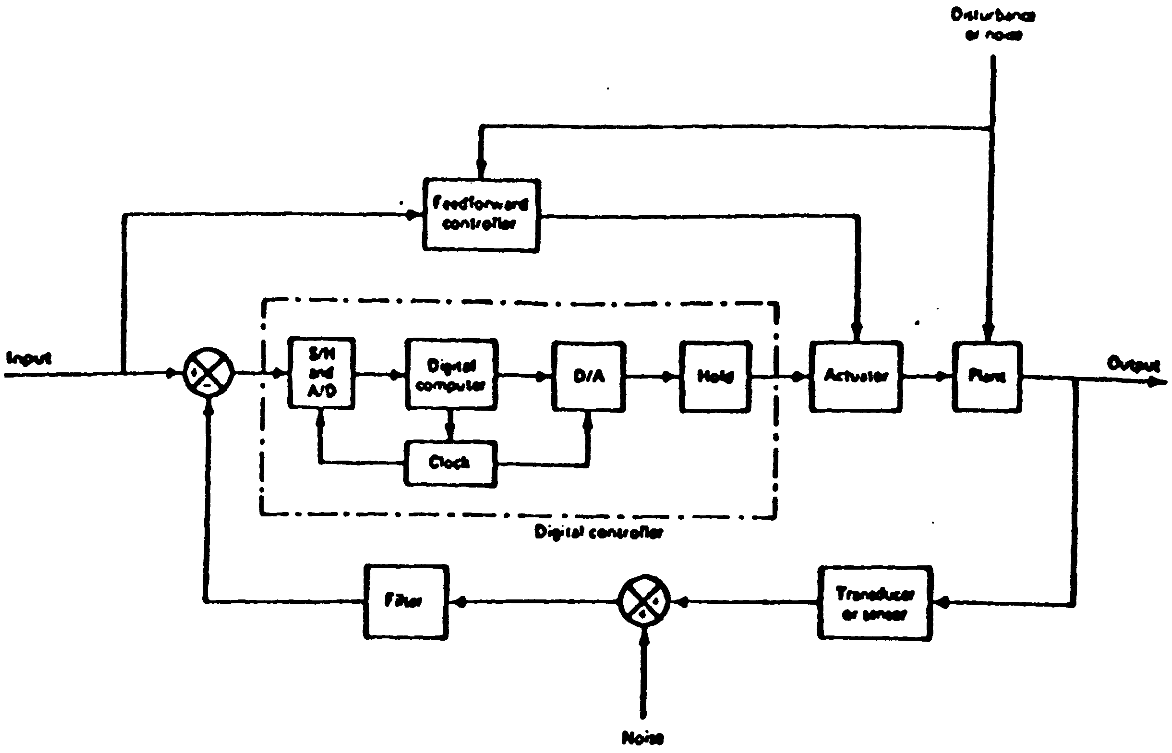


Figure 1-4: A typical digital controller [6].

The cyclic nature of the UPF converter, due to its sinusoidal inputs, makes it easily and accurately described via a sampled data model. Once such a model is available, the design of discrete-time controllers using the model can be straightforward. A discrete time controller lends itself naturally to implementation in a digital setting. A digital implementation would add to the flexibility and robustness of the system. Moreover, it would allow for the on-line real time estimation of various circuit parameters that are likely to change with age or for other reasons, thereby facilitating adaptive control. One such parameter is the bus capacitance, which can be estimated from the ripple on the output voltage. The microprocessor may also be programmed to shut down under emergency conditions and at the request of the user. Considering that the cost of microprocessors is decreasing rapidly and that we could conceivably use part of the processing capability already available on a system, the cost of a digital implementation may not be significantly more than an analog one, if at all. Moreover, a digital controller need not be slower than an analog one; closed-loop bandwidths can be comparable, and digital controllers can even result in faster settling than analog controllers. This will become clear in the case of UPF converter control, as the sampled data model and controller are developed in later chapters.

Digital control theory is well-developed [10], and microprocessors are becoming less and less expensive and more powerful. Nevertheless, the cost of digital hardware is still too high for many small-scale applications. This combines with programming complexity and time to hinder the development of digital control applications. The digital controller explored in this thesis is developed with cost as well as design time in mind. An inexpensive microprocessor is chosen, and a high-level language (C) is used for algorithmic implementation. Entry of digital controllers into a field of small-scale inexpensive products such as power electronics would be highly unlikely without attention to cost and design time. Our goal is to demonstrate that digital controllers can provide equal or better performance than their analog counterparts in controlling inexpensive power electronic circuits with little extra cost in design time or money.

1.7 Thesis Organization

This thesis document is organized into seven chapters. Chapter 2 develops a large-signal model for the boost converter, averaged over the time scales of both the switching period and half the line period. It then presents a sampled data model based on the averaged models, which is then used for the design of a discrete-time controller. Simulations of the resulting discrete-time system are then described.

Chapter 3 describes the implementation of the digital controller. It begins by discussing common issues in digital control, such as quantization error and sampling. Problems more specific to the system at hand are then introduced. A description of the microprocessor development system is provided, and the initial implementation is discussed. Software, digital hardware, and analog hardware used for interfacing to the power converter are described. Simulations used in the development process are also shown.

Design and implementation of adaptive control are discussed in Chapter 4. Other additional features and potential improvements are also explored. While some are simple enough to implement, others do not promise significant improvement in this system but are worth exploring for future reference.

Chapter 5 presents the results of experiments. These results are compared with simulations for verification of successful operation. They are also compared with an analog implementation of a related control scheme (to evaluate the digital implementation), and with an implementation using a commercially available circuit that utilizes a different control scheme (to evaluate our choice of controller).

Chapter 6 investigates more compact implementations of this system. The first one uses the same microprocessor as in the development of the prototype, and interfaces it directly to off-chip memory and other hardware essential to this application, eliminating the need for the development board and all the extra hardware and firmware. A second uses a VLSI implementation, which would be cheaper, faster, and more compact.

Chapter 7 summarizes the pertinent conclusions of this research work, and identifies problems for future research. Matlab simulation code, C programs for testing and running the microcontroller, and hardware schematics are included in the appendices.

Chapter 2

Modeling and Control Design

Systems that are inherently cyclic, such as the one at hand, can be accurately described using a sampled data model (SDM). Such a model is a discrete time description useful for designing discrete time controllers. Sampled data models and discrete time control for the UPF are developed below. Simulations of the closed-loop system under various control designs are included, but omitting details such as quantization.

2.1 Dynamic Models of the UPF

The circuit in Figure 1-2 can be described by a power balance equation, as mentioned earlier. This power balance relationship is used to develop continuous- and discrete-time models of the circuit, averaged over both the switching period and the period of the rectified line voltage. These models are used to develop controllers and to obtain simulations in MATLAB ^{TM} for the closed-loop system with different controller designs.

2.1.1 The T_s-Averaged Model

In [4] Mahabir develops models for the UPF using a resistive load in parallel with a constant power load. The models below are developed in the same manner, ignoring the effect of the load resistor in parallel with the constant load. Equating the power input to the system in Figure 1-2 with the power dissipated in the different components, we find:

$$\frac{1}{2}C\frac{dv_o^2}{dt} = v_{in}i_L - \frac{1}{2}L\frac{di_L^2}{dt} - P \tag{2.1}$$

Typically, the inner-loop, or the current loop controller, is operating at a much higher bandwidth than the outer loop. If we choose that to be the case, we can make the approximation that the inner loop is working ideally, and that the inductor current is following the current commanded by the controller:

$$i_L(t) = k(t)v_{in}(t) \tag{2.2}$$

(Note that $k(t)$ is referred to in Figures 1-2 and 2-2 as $\mu(t)$.) Substituting for the inductor current above:

$$\frac{1}{2}C\frac{dv_o^2(t)}{dt} = k(t)v_{in}^2(t) - \frac{1}{2}L\frac{d[k^2(t)v_{in}^2(t)]}{dt} - P \tag{2.3}$$

In developing this model, the switching frequency ripple in the output voltage was ignored. This result is essentially the same as that arrived at by averaging a switch model over the switching period. Hence it will be referred to as the continuous-time T_s-averaged model. It is clear that if v_o is taken as the state variable, 2.3 is a nonlinear description of the power converter. However, if v_o^2 is used, the model is effectively linear [4].

2.1.2 The T_L-Averaged Model

The T_s-averaged model may still be simplified further by averaging over a longer period. Let T_L be the period of the rectified input. Averaging 2.3 over T_L yields an averaged model on the time scale of the input period. The running average defined by

$$\bar{w}(t) = \frac{1}{T_L}\int_{t-T_L}^{t} w(\tau)d\tau \tag{2.4}$$

is used. Denote $v_o^2(t)$ by x. If the ripple in $v_o(t)$ is small, then $x \approx \bar{v}_o^2$. Assuming that $k(t)$ varies slowly enough to be considered constant over any interval of length T_L further simplifies our analysis, as

$$\int_{t-T_L}^{t} k(t)v_{in}^2(t) \approx k(t)\frac{V^2}{2} \tag{2.5}$$

and

$$\int_{t-T_L}^{t} \frac{d[k^2(t)v_{in}^2(t)]}{dt} \approx 0 \tag{2.6}$$

with our assumption that $k(t)$ is slowly varying. The resulting T_L -averaged model is given by the *linear time-invariant (LTI) first-order* description

$$\frac{dx(t)}{dt} = \frac{1}{C}(V^2k(t) - 2P) \tag{2.7}$$

The T_L-averaged model is not as accurate as the T_s-averaged one, as it ignores the ripple on the output voltage. This ripple is very small, however, as the load capacitor is usually very large. The T_L-averaged model provides a much simpler description, which facilitates the straightforward design of controllers. Moreover, the assumption that $k(t)$ is essentially constant over a rectified line period allows for the implementation of "slow" controllers that operate on that parameter. This is especially helpful in digital implementations, as will become clear shortly. The T_s-averaged model is required for simulations and in formulating an expression for the output voltage ripple, which is used in Chapter 5 to estimate the value of the output capacitor. However, controllers based on the T_L -averaged model should yield satisfactory results in regulating the output voltage in our application. Controllers based on the T_s -averaged model would need to react with the frequency of the switch. An analog implementation of such a "fast" controller is currently being investigated [15], but a digital implementation at that speed with the currently available technology would be too expensive to be practical.

2.1.3 Sampled Data Models

To achieve a unity power factor, the input current has to coincide in shape and phase with the input voltage. This dictates that $k(t)$ is kept constant over each cycle. Under this condition, it is natural to investigate sampled data models (SDM) and controllers. A sampled data model on the time scale of the input period T_L is developed below and eventually used for designing a discrete-time controller for the digital controller implementation. With $T_L = 8.3ms$ as the sampling period, there is ample processing time for most present day micro-controllers to execute significant control algorithms. A sampled data model on the time scale of the switching period T_s is also developed. It is used later in simulations and in developing adaptive control schemes, for which the "T_L -SDM" proves insufficient [4].

T_L-SDM

Assume that $k(t)$ is essentially constant over the input period, the value of $k(t)$ in the nth cycle being denoted by $k[n]$. Let the value of v_o^2 at the beginning of the nth cycle equal $x[n]$. Integrating either the T_s-averaged model in 2.3 or the T_L-averaged model in 2.7 over T_L yields the following equation [3]:

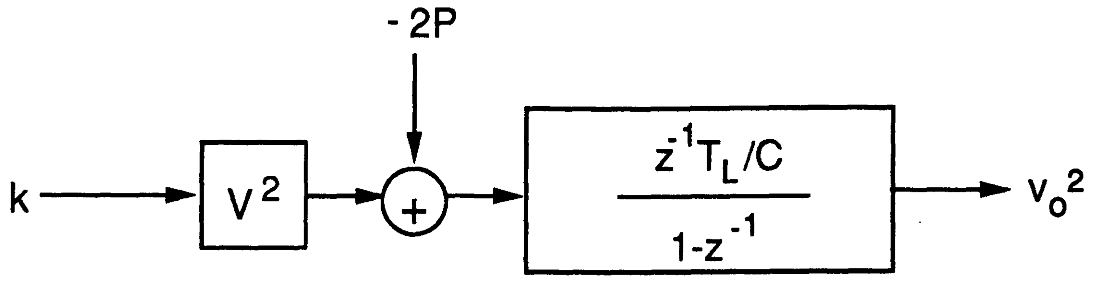


Figure 2-1: The T_L-based sampled-data-model.

$$\frac{1}{2}C(x[n+1]-x[n]) = T_L(k[n]\frac{V^2}{2} - P) \tag{2.8}$$

or

$$x[n+1] = x[n] + \frac{T_L V^2}{C}k[n] - \frac{2T_L}{C}P \tag{2.9}$$

This T_L-SDM is a state-space representation of a first-order LTI discrete-time system, with control input $k[n]$ and disturbance input P, as shown in Figure 2-1 (which actually represents the z-transformed version of 2.9).

T_s-SDM

The development of a sampled data model at the time scale of the switch period is similar to that of the T_L-SDM. Assume that $k(t)$ is constant over T_s. The time index η is used to denote the switching period. The sampled data model is then acquired by integrating the T_s-averaged model of 2.3 over T_s. As presented in [4], the T_s -SDM is

$$x[\eta+1] = x[\eta] + b_1[\eta]k[\eta] + b_2[\eta]k^2[\eta] - \frac{2PT_s}{C} \tag{2.10}$$

where $b_1[\eta]$ and $b_2[\eta]$ are time varying input gains given by:

$$b_1[\eta] = \frac{V^2}{C}T_s - \frac{V^2}{C}\{\frac{T_L}{2\pi}[\sin(2\pi(\eta+1)T_s/T_L) - \sin(2\pi\eta T_s/T_L)]\} \tag{2.11}$$

$$b_2[\eta] = \frac{V^2 L}{C}\{\sin^2(\pi(\eta+1)T_s/T_L) - \sin^2(\pi\eta T_s/T_L)\} \tag{2.12}$$

whereas the T_L -SDM ignores the ripple in the output voltage, and therefore reaches a steady state where $x[n+1] = x[n]$, The T_s -SDM does not ignore this ripple. It therefore reaches a cyclic steady state with a period equal to that of the output voltage ripple.

2.2 Controller Design

The models developed above are verified through simulation and comparison with results from a test circuit in [4]. These models are useful for designing controllers for the UPF. Controllers may be designed based on the linearized model presented earlier, or on any of the linear models of the last section. Both the T_s-Averaged and T_L-Averaged models have been used to design analog controllers for the UPF [15], [4]. To take advantage of the T_s-SDM in a digital controller would require processing speed that cannot be met today with a commercially viable digital implementation. The T_L-SDM is therefore chosen as the basis for designing a discrete-time controller to be implemented digitally.

2.2.1 PI Control

A natural way to regulate the output voltage to a desired value V_o is to use a discrete time version of proportional integral, or PI, control [3]. This provides for zero steady-state error despite uncertainties in system parameters (including load power) as shown below. Proportional control by itself could not avoid load regulation in this case. To exploit the linear model of the power supply, the state variable to be controlled needs to be the *square* of the output voltage instead of the voltage. This variable is x, and is regulated to a desired value $X = V_o^2$. Note that the value of v_o at the beginning of each input cycle is what is being regulated to the reference, and not the average value of v_o . The ripple on v_o is assumed to be small enough that this is not a major concern.

To assure that in steady-state the error in the output voltage goes to zero, the error in $v_o^2 = x$ is fed into an accumulator, i.e. a discrete-time version of an integrator. For the accumulator, a new state variable σ is defined:

$$\sigma[n+1] = \sigma[n] + (x[n] - X) \tag{2.13}$$

The input to the accumulator is the error term $x[n] - X = v_{err}[n]$. If a steady state is attained, $x[n+1] = x[n] = X$ and $v_{err}[n] = 0$ (Otherwise the accumulator would

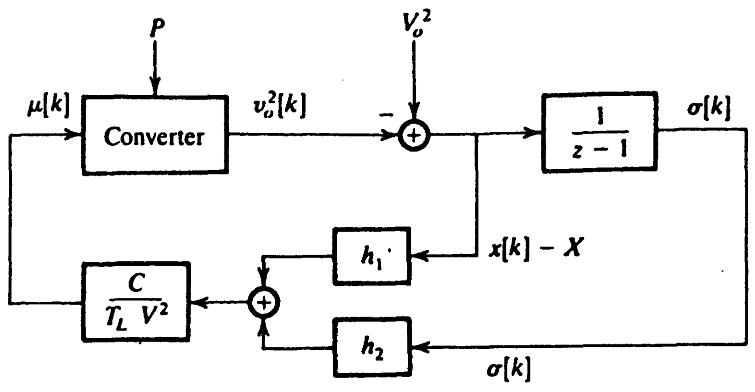


Figure 2-2: A discrete-time PI controller [3].

not be in steady state). To implement PI control then, the value of the accumulator is multiplied by an accumulator gain G_I and added to the product of a proportional gain G_P and v_{err} . The result is fed into the open-loop system through its control input $k[n]$:

$$k[n] = G_P v_{err}[n] + G_I \sigma[n] \tag{2.14}$$

Defining the normalized gains

$$h_1 = \frac{T_L V^2}{C} G_P \tag{2.15}$$

$$h_2 = \frac{T_L V^2}{C} G_I \tag{2.16}$$

yields the following:

$$k[n] = \frac{T_L V^2}{C}(h_1(x[n] - X) + h_2 \sigma[n]) \tag{2.17}$$

The resulting closed-loop system is shown in Figure 2-2.

The accumulator in the figure is shown as $\frac{1}{z-1}$. This is the z -transform transfer function of the accumulator described above in 2.13 from its input v_{err} to its output σ.

The overall system is a combination of the converter and the controller, both of which are first-order subsystems. The closed-loop model is therefore second-order, and is described by the following state-space model:

$$\begin{pmatrix} \sigma[k+1] \\ x[k+1] \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ h_2 & 1+h_1 \end{pmatrix} \begin{pmatrix} \sigma[k] \\ x[k] \end{pmatrix} - \begin{pmatrix} 1 \\ h_1 \end{pmatrix} X - \begin{pmatrix} 0 \\ \frac{2T_L}{C} \end{pmatrix} P \tag{2.18}$$

If

$$A = \begin{pmatrix} 1 & 1 \\ h_2 & 1+h_1 \end{pmatrix} \tag{2.19}$$

The characteristic polynomial for this system, obtained by finding the determinant of $zI - A$, is

$$z^2 - (h_1 + 2)z + 1 + h_1 - h_2 \tag{2.20}$$

The roots of the characteristic polynomial are the poles of the system:

$$z_1, z_2 = \frac{(h_1 + 2) \pm \sqrt{h_1^2 + 4h_2}}{2} \tag{2.21}$$

The transient response of any variable in the system is of the form $c_1 z_1^k + c_2 z_2^k$ (for $z_1 \neq z_2$), where c_1 and c_2 are constants that depend on the initial conditions (and on which variable is being considered). If G_I and G_P can be chosen such that z_1 and z_2 have magnitude less than 1, the system will be stable, and will reach a steady state in which $v_{err} = 0$.

Without the accumulator part, the state variable x would be governed by the first-order equation

$$x[n+1] = x[n] + h_1(x[n] - X) - \frac{2T_L P}{C} \tag{2.22}$$

which in the steady state settles down to

$$x[n+1] = x[n] = X + \frac{2T_L P}{Ch_1}. \tag{2.23}$$

This system exhibits load regulation and a nonzero steady-state error.

Figure 2-3 shows the response of the closed-loop system with proportional control to a doubling in the load and Figure 2-4 shows the response to the same perturbation with PI control. In these simulations, as well as in all simulations hereafter, plots of k will be used to illustrate the envelope of the input current, since from Figure 1-2, $i_P = kv_{in}$.

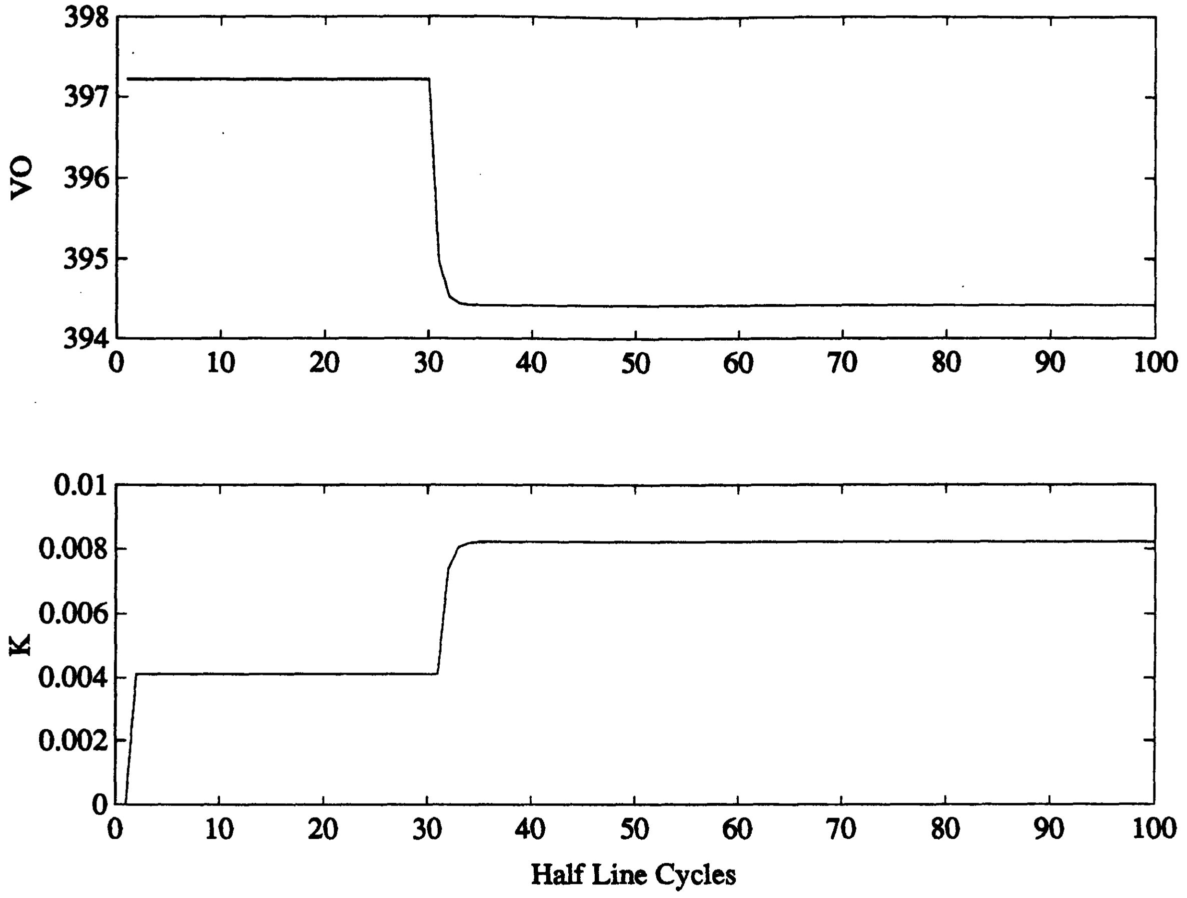


Figure 2-3: Response of the closed-loop system with proportional control.

The scheme above assumes that no knowledge of the load is available, and proceeds to compute and store in the accumulator a value that would provide zero error for any load. In most cases, however, a nominal value for it is known; if measurements of the load are available, they can be taken advantage of as well. Based on knowledge of the actual or nominal load, the control variable $k[n]$ may be found as the sum of a nominal $K[n]$ and a $\tilde{k}[n]$ to be computed by a PI controller. This scheme has its advantages. In particular, the controller does not need to work as hard if the estimate for $K[n]$ is accurate. This can translate into less computation time in a digital implementation. Moreover, outputting $\tilde{k}[n]$ instead of $K[n]$ could mean more resolution per unit of control variable, if, as in the case of a digital controller, the output resolution is limited. The controller developed above is more straightforward to implement however, and is better equipped to deal with the most general cases in which the load and other parameters in the circuit are not well known, and is therefore chosen for the implementation. The issues introduced here regarding the difference between the two designs are discussed further as implementation issues.

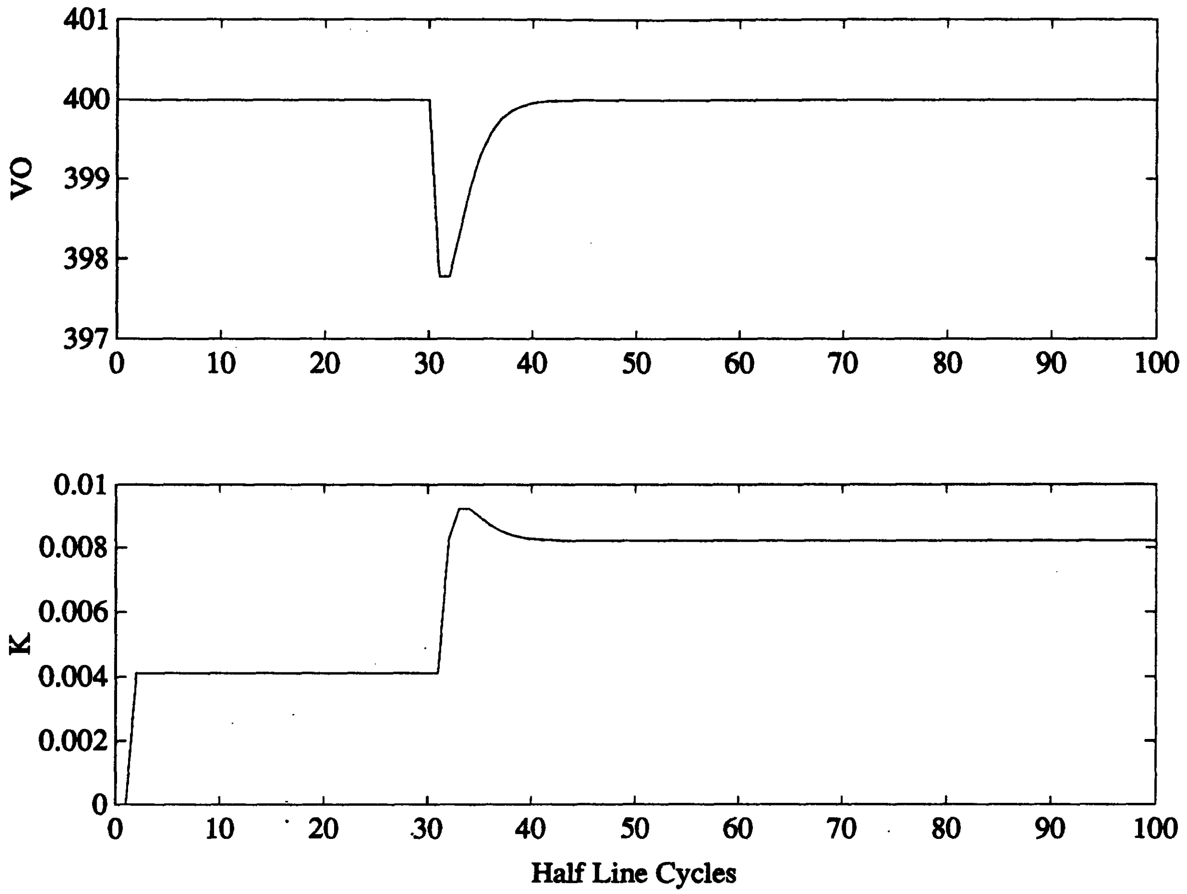


Figure 2-4: Response of the closed loop system with PI control — system poles at .5.

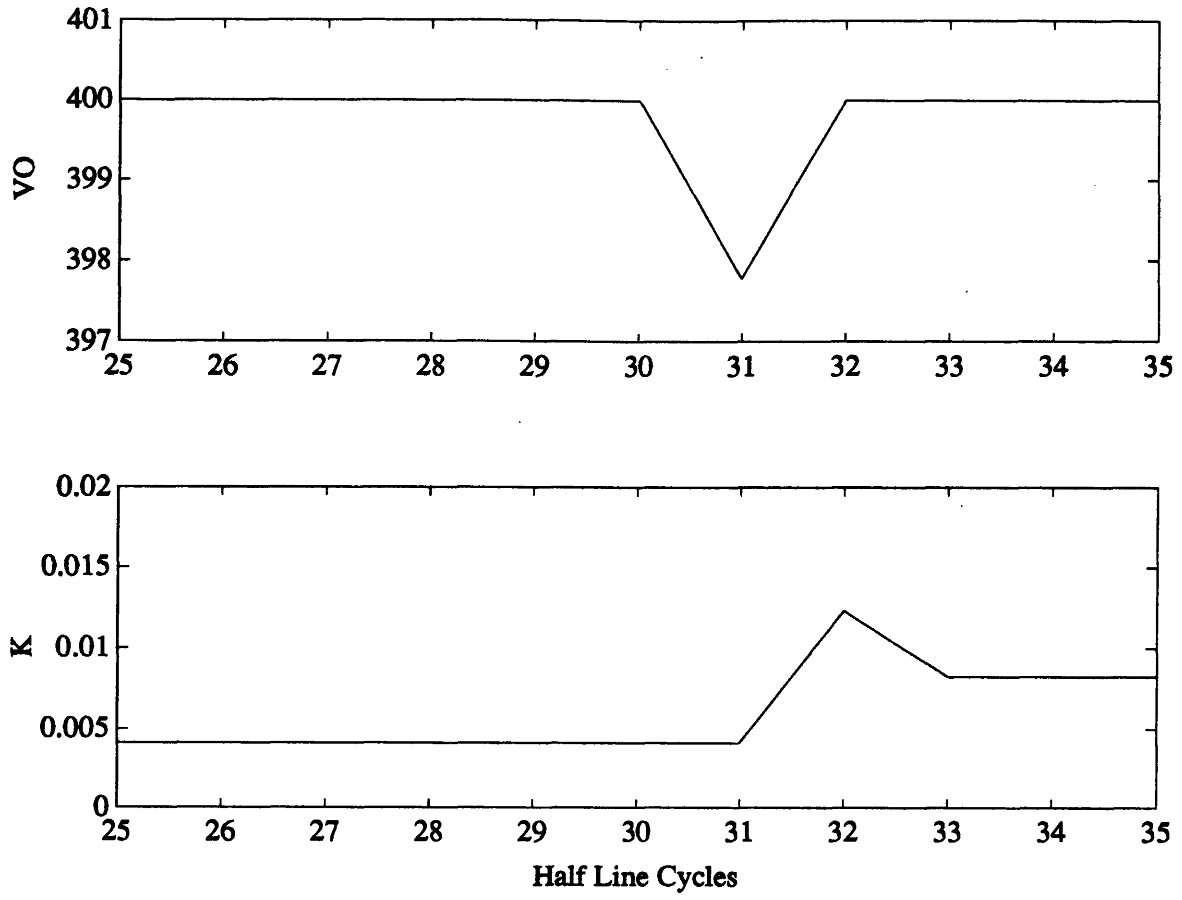


Figure 2-5: Deadbeat response to load disturbance.

2.2.2 Deadbeat Control

A stable LTI continuous-time system cannot settle down to its steady-state in finite time after a disturbance occurs, since its transients decay exponentially. In discrete-time systems, however, the case is different. Transients evolve as $\boldsymbol{A}^n$, where n is the number of time steps. By choosing $\boldsymbol{A}$ to be a nilpotent matrix, with eigenvalues at 0, this quantity will go to zero in finite time. Specifically, if $\boldsymbol{A}$ is a nilpotent matrix, $\boldsymbol{A}^n = 0$ will definitely be true for $n = m$, where m is the dimension of $\boldsymbol{A}$, or for $n < m$ under certain conditions. The result is a so-called *deadbeat* system, whose transients last only a finite number of steps.

For the UPF closed-loop system with PI control, setting $z_1, z_2 = 0$ in 2.21 yields the values $h_1 = -2$ and $h_2 = -1$ for the normalized controller gains. The simulations in Figure 2-5 show the response of the deadbeat system to a doubling in the load . Steady state is recovered in 2 cycles, the order of the closed-loop system.

Clearly, deadbeat response requires very high gains and high levels of control command that may not be available to the designer. As will be shown in Chapter 3, high gains and bandwidth also result in an increase in quantization error in the digital system. . In the case of the UPF, a tremendous increase in input current (to correct for the error in the output voltage induced by the step in power) is required to achieve deadbeat behavior. Physical limitations in the actual implementations deny the system such input current levels. These limitations along with the complications due to high bandwidth in the quantized system prevent the implementation of deadbeat control.

2.2.3 Additional Design Features

The PI controller introduced above yields satisfactory results in analysis and simulation. However, power supply component limitations, system specifications, and other implementation difficulties require the utilization of other control techniques in conjunction with the basic controller. A digital implementation facilitates these extra features with relative ease. Most of these control techniques are developed in more detail in the implementation section, as they resulted from some implementation difficulty and many of their parameters are implementation dependent. However, their general features were part of the initial design process and are introduced below.

Adaptive Control

Uncertainties in modeling and circuit parameters motivate the implementation of adaptive control. An adaptive controller has the ability to "redesign" or adjust its control characteristics based on measured or estimated system parameters. Regulators that employ such methods to automatically adapt to variations in system parameters are also known as self-tuning regulators (STR). Figure 2-6 shows a block diagram of a self-tuning regulator. An outer loop in the regulator adjusts the parameters of the regulator through estimation and design calculations. The process and a feedback regulator, the PI controller in our application, make up an inner loop which is "adapted" to parameter variations using inputs from the outer loop.

The system at hand provides a good test-bed for parameter estimation and adaptive control techniques. A microprocessor based system has the ability to carry out the complex computations necessary, and uncertainty in the circuit parameters of the power supply provides a problem to tackle.

The gain block with gain value $\frac{C}{T_L V^2}$ in Figure 2-2 provides several circuit parameters of which our knowledge is not necessarily perfect. The value of T_L depends on whether the input frequency is $50Hz$ or $60Hz$. Adapting to variations in input frequency requires detection of the zero-crossings of the input waveform and is not

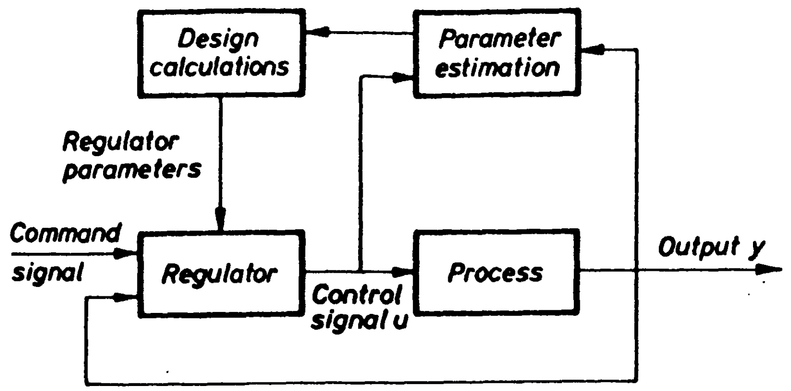


Figure 2-6: A self-tuning regulator [10].

implemented here. V, the peak input voltage, depends on whether the input is supposed to be $110 voltsrms$ or $220 voltsrms$. It may also change during operation. To correct for perturbations due to disturbances in the input voltage, a DC filter is used to estimate the peak input voltage and use the updated value in place of the original estimate. This is the path shown as a dashed line in Figure 1-2. A more difficult problem is responding to changes in the value of the load capacitance. The design and implementation of a self-tuning controller which responds to variations in output capacitance are presented in Chapter 5.

Bang-Bang Control

Components in the power supply are rated for operation below certain peak values of voltage and current. For this reason, it may be necessary to apply an automatic shutoff of the system if the PI controller allows the output voltage or other variables to reach these limits for certain unlikely but possible disturbances. There may also be a specification on the lower bound the voltage is permitted to reach. A slow but robust controller may behave optimally in dealing with common disturbances but would allow excursions outside this allowed band for some extreme operating conditions.

One way to deal with this problem is to have different controllers for the different situations. A linear controller, such as the PI controller discussed earlier, may be designed and optimized for one range of voltages around the nominal operating point. A second controller detects if the circuit variables are near their specified limits and applies more extreme commands to bring them back within the safe range of operation. A controller that switches back and forth between two control commands in this manner is known as a bang-bang or an on-off controller. [19] discusses a similar a similar idea.

In a digital implementation of a controller such as the one at hand, quantization error may be a problem in the steady state, and is usually a function of the gains of the feedback loop. While reducing these gains will reduce the effects of quantization, it may result in a regulator that fails to meet the specifications of the system. An on-off controller could then be used to insure that these specifications ar met. A more detailed analysis requires more knowledge of the implementation and the system limitations, and is presented later.

Band Control

Another application of the division of the regions of operation among different controllers for optimal results should be helpful in dealing with quantization error in the digital implementation. Increasing the resolution on the input to the controller and reducing its gains can bring the control quantization error to within tolerable levels for normal operation. With the inclusion of adaptive control, in this case capacitance estimation and self-tuning based on the estimated capacitance value, measurement of the output voltage ripple is required. This ripple may be very low in amplitude, close to the size of the error due to quantization of the control. For such a measurement, therefore, it will be necessary to eliminate quantization noise, at least for the time of the measurement.

To achieve cancellation of quantization noise, the controller command cannot be allowed to fluctuate. This fluctuation is inherent in the digital implementation of the controller, however, and may only be eliminated by disabling the PI controller for some time while the output voltage is in (or near) the steady state. The general algorithm for this scheme involves averaging the control command within a certain distance from the steady-state and using the average value while measurement of the output ripple is being carried out. The converter isrunning open-loop in this regime, and care has to be tâken to close the loop if significant deviation from the steady state is detected. The result is that within the band of $\pm\epsilon$, the voltage is allowed to settle to any value. If ϵ is small enough, the error is insignificant, and a reasonable resolution on the AD converter and the output port should make such a value feasible. Of course, the transient response is no longer the same for certain load disturbances. The details of this algorithm and results are also discussed later.

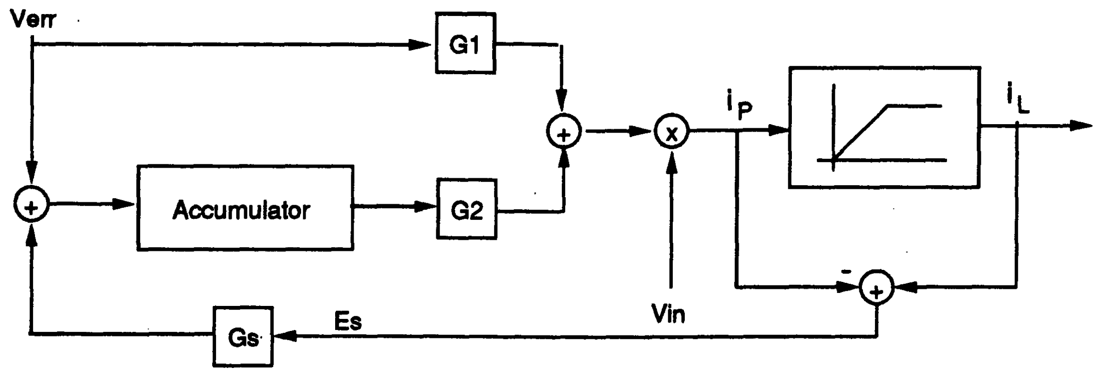


Figure 2-7: Regulator with antiwindup

2.2.4 Integrator Windup and Startup Behavior

In the actual UPF system, there are upper and lower limits on the input current that the feedback controller can command. For certain load disturbances and controller gains, these limits may be reached. When that happens, the input current will saturate at its limit, regardless of how far beyond the limit the controller commands it to be. The feedback path is effectively broken. In a regulator that utilizes integral action, this results in integrator windup [10].

When the feedback path is broken, the integrator is left unconnected to the rest of the system except through its input, which is the error in this case. Being an unstable system, it may integrate to a very large value. When the error is finally reduced to the desired level, the integral may be so large that it takes considerable time until it assumes a normal value again, so the commanded current will remain stuck at its limit far longer than it should for proper recovery. One way to avoid integrator windup is through an additional feedback path that sends the error between the actual and desirable commanded currents back to the integrator through an appropriate gain, as shown in Figure 2-7. The error between the commanded and the actual currents is zero when the current is not saturated.

A more easily implemented solution is to completely isolate the integrator when the current is saturated. The integral action is disabled, and the value in the integrator is held until the current is no longer saturated.

Significant windup effects are witnessed in the UPF system simulations only during the startup transient. The load transients did not produce enough of a voltage disturbance to cause the integrator to wind up. Another problem with the startup

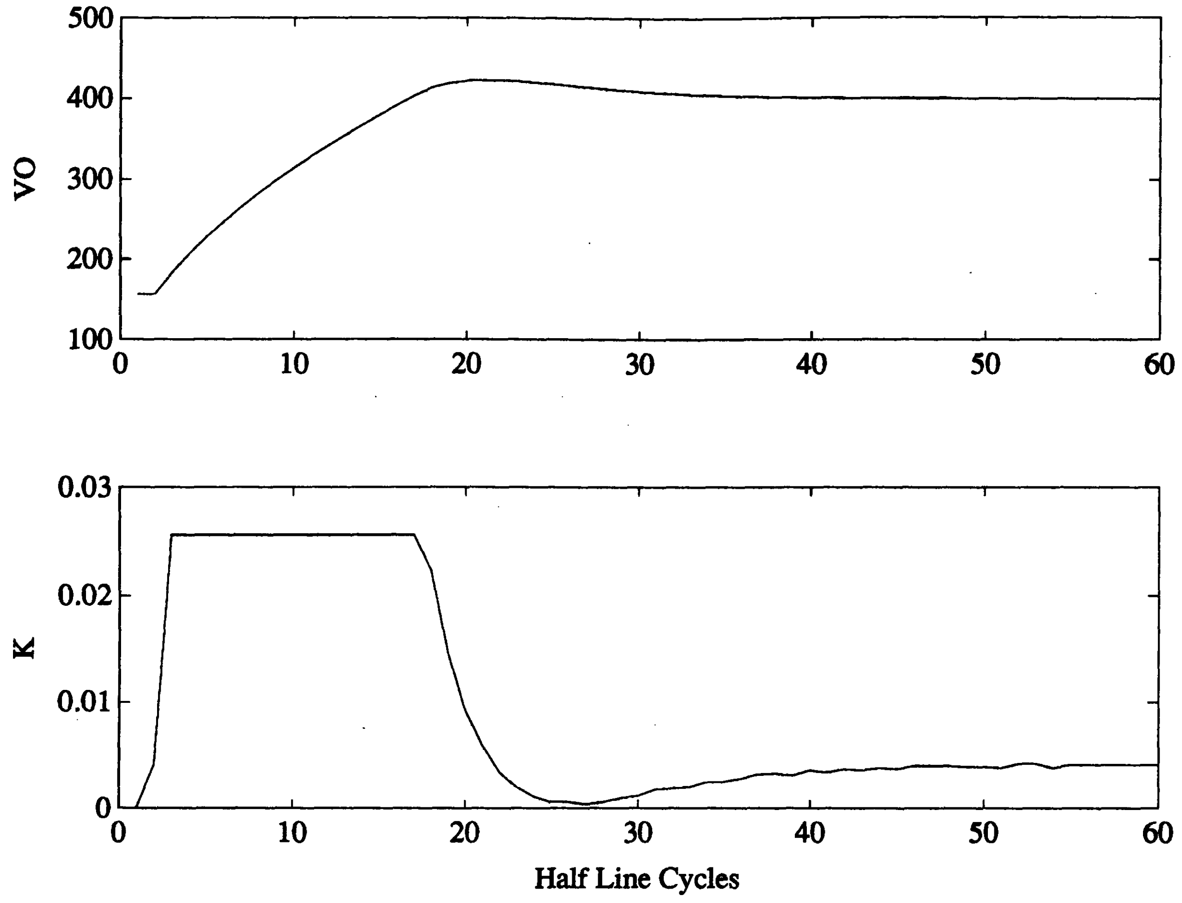


Figure 2-8: Startup with integrator windup and no soft-start.

transient is the shape of the current, which shoots to the saturation level due to the large error detected. This problem is dealt with using a soft-startup routine. The reference voltage is initialized to a lower voltage than the desired one. It is then stepped up to the final value over a few cycles, during which the current level rises slowly as it raises the output voltage level to the reference. Figure 2-8 demonstrates the effect of integrator windup and the sharp current transient during startup. The implementation of a soft-startup routine and an anti-windup scheme are discussed in Chapter 3.

2.2.5 Overall Design

Combining the PI controller, the adaptive controller for estimating and adjusting to the capacitance, the bang-bang controller, the "band" controller, and the anti-windup mechanism, the system in Figure 2-9 is obtained.

The combination of several features and control loops complicates the design be-

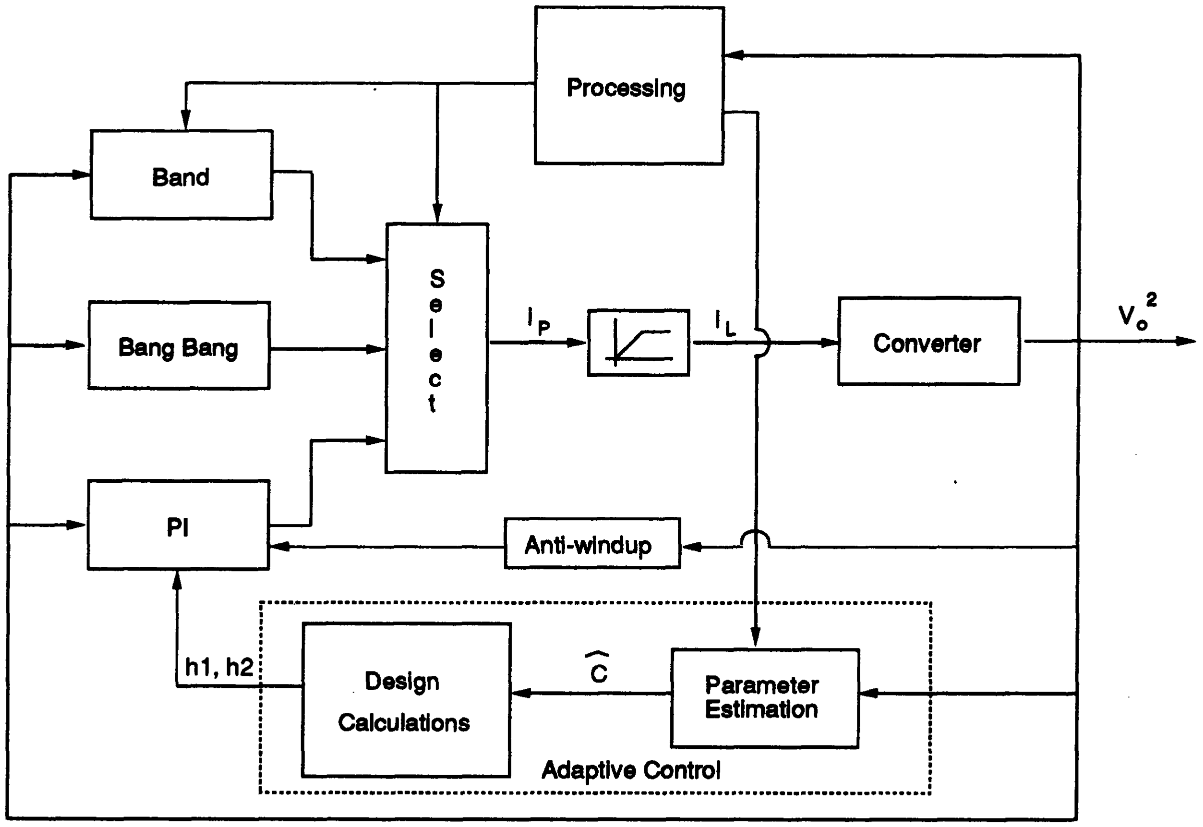


Figure 2-9: Overall controller design.

yond the basic PI controller. However, with a fairly cheap digital implementation, the inclusion of even more features is possible, as will be demonstrated in the next chapter.

Chapter 3

Implementation

A digital implementation of the controller can add to the robustness of the system by allowing a more complex and flexible design, and a highly repeatable implementation. The disadvantages, however, are higher cost, and performance degradation due to sampling and quantization [6]. In the following sections, the various hardware and software components used to implement different sections of the layout in Figures 1-2 and 2-9 are presented. The implementation discussed here excludes the adaptive control section. A general description of the development system is given, with reference to the system manuals for more details. Use of the relevant components in this specific application is then illustrated. Implementation circuits are mentioned by function and illustrated by block diagrams. The algorithms used are presented and the methods used to arrive at the design parameters are discussed. Calculations with actual values and circuit diagrams are reserved for the appendices, however.

3.1 Hardware Implementation

Hardware is needed to replace every component in Figure 1-2. Digital hardware is used for implementing the voltage-loop controller. The current-loop controller is implemented in analog hardware and interfaced to the boost converter. Interfacing between the digital hardware and the rest of the circuit requires some extra miscellaneous hardware as well. The implementation of these circuits is discussed below, and their construction method briefly outlined.

3.1.1 Digital Voltage-Loop Controller

The voltage-loop controller in 1-2 is to be implemented digitally. From the system model and control design described in Chapter 2, the output of this loop only changes once every T_L seconds. This low-speed loop naturally lends itself to a digital implementation that is not too expensive. To implement the current-loop controller on a microprocessor would require more expensive technology. Moreover, the digital implementation of the voltage-loop controller provides most of the flexibilty and robustness that one could hope to achieve in this system. The extra cost required to include the current loop controller on the microprocessor is not warranted presently.

The ROMless 16-bit 80C196KB was chosen as the microprocessor for the application. Much less expensive than 32-bit microprocessors and DSP chips, this embedded microcontroller provides ample processing power for the system at hand. A straightforward implementation of the PI controller discussed above actually should not require even as sophisticated or expensive a microprocessor as the 80C196KB. However, the difficulties and complications in design due to quantization may cause a simpler microprocessor to end up requiring a considerably more complex implementation. Moreover, to achieve high levels of performance and to explore the various control schemes and features made possible by operating in the digital domain, a reasonably powerful microprocessor is required.

The EV80C196KB software evaluation tool for the 80C196KB microcontroller allows control and monitoring of the processor through an Embedded Controller Monitor (ECM) that supports basic debug facilities in the target system [12]. The ECM is broken into two programs, one executing in the EV80C196KB and the other in an IBM PC compatible. They communicate via an asynchronous serial channel to allow the downloading of microcontroller programs, execution of these programs, and monitoring the various processor states as the programs execute. The 80C196KB has a 16-bit CPU and 232 bytes of on-chip RAM. It is a register-to-register machine, so no accumulator is needed, and most operations can be directly performed from or to any of the registers. In addition, there are many peripherals that are directly controlled through register operations. Figure 3-1 illustrates the features of the evaluation board and the microprocessor necessary in the application. Details of this system and other features of the 80C196KB and the EV80C196KB are found in [13] and [12].

Interrupt Handling

In a digital control system, some tasks such as sampling and output generation need to be synchronized to specific points in time. Others are event generated, such as shutdown or other precautionary measures taken when a certain condition is detected. Still other tasks, such as system operation services or computations carried out during time gaps free of other processing tasks, have no time restriction. For such a system, an interrupt oriented software structure is the best choice. In the UPF controller setup, the sampling and control output need to be carried out at specific points in time (8.33 ms apart). System shut-off, whether user generated or due to failure, is

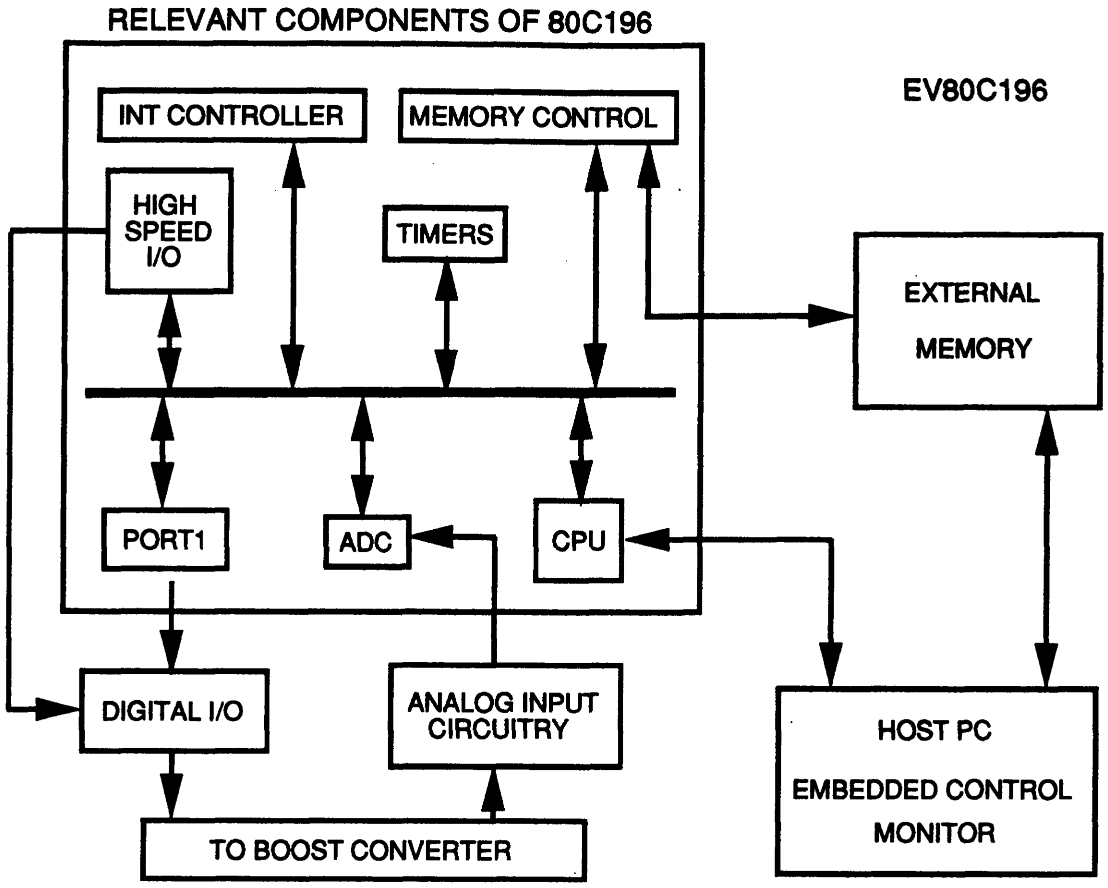


Figure 3-1: The digital development system.

event-generated, and fine tuning of the system over several control cycles may be accomplished through estimation of circuit parameters to be computed over some time, whenever the processor is free. The event-generated tasks are carried out when an event occurs, and the time specific tasks are executed when timers on the microcontroller expire or reach a specified value. These events and timers generate interrupts, with the event-generated interrupts in this case having the higher priority. When an interrupt is generated, the CPU transfers control to the interrupt service routine (ISR) associated with the detected interrupt, as instructed by an interrupt vector. Tasks that have no timing constraints are not executed within an ISR and therefore have the lowest priority. On the 80C196KB, special function registers (SFR) and other devices handle interrupt generation, as shown in Figure 3-2.

The A/D Converter

The 80C196KB has an analog to digital (A/D) converter system on board. The system has an 8-channel multiplexer to allow the sampling of 8 different analog signals at

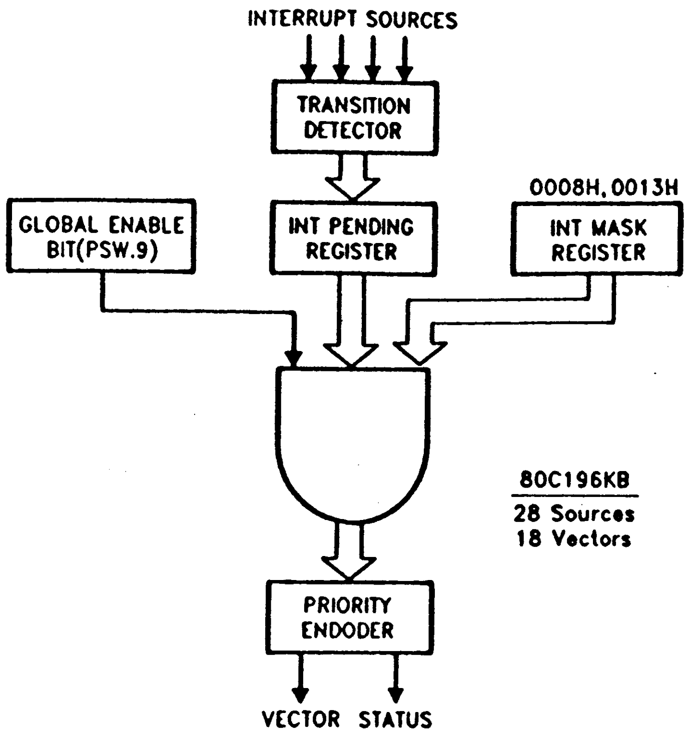


Figure 3-2: The 80C196 interrupt structure [13].

different times. The output of the multiplexer is passed through a sample and hold and then a 10-bit successive-approximation A/D converter. On the $12MHz$ version of the microprocessor, a conversion is completed in approximately $26\mu s$. As is the case with other peripherals, the A/D converter is SFR controlled. In this application, an A/D sample is acquired by writing a value to an SFR that specifies the channel to be sampled and initiates the acquistion. When the conversion is complete, an interrupt is generated. The results of the conversion (the value sampled and the number of the source channel) are found in other SFR's. This process is outlined below:

1. Conversions are started by loading the AD_COMMAND register with the channel number and setting the GO bit to one to initiate conversion immediately.

2. The AD_RESULT_HI register contains the 8 most significant bits of the conversion result.

3. The 2 least significant bits are acquired from the AD_RESULT_LO rgister. This SFR is also contains information on which channel has just completed a conversion. This information is necessary as the UPF application requires the use of more than one channel.

The A/D converter generates an interrupt when conversion is completed. This method of detection is chosen over the possible polling of a status bit in AD_RESULT_LO, in accordance with the interrupt driven software structure of the whole system.

Timers

There are two timers on the 80C196KB. Timer1 is a 16-bit free-running timer incremented every 16 clock cycles, giving it a timing resolution of $1.33\mu s$. Timer2 is clocked externally through an input pin. Four "software timers" may be implemented using the high speed output unit (HSO) in conjunction with one of these timers. This is accomplished by programming the HSO to generate interrupts at preset times. As each programmed time is reached by the specified timer, a Software Timer Interrupt is optionally generated. The timing of the digital controller implemented here is accomplished by writing to SFR's specifying Timer1 as the source of timing and enabling the Software Timer Interrupt. When the interrupt is generated, time-specific tasks are carried out. The source of the interrupt is identified from the contents of another register. This information is necessary when using more than one software timer (e.g. to implement a digital filter during testing, or for adaptive control which uses much faster sampling to estimate circuit parameters). The steps below describe this implementation:

1. The HSO_COMMAND register is loaded with the number of the software timer to be initiated. Timer1 is specified and the Software Timer Interrupt is enabled also as part of the byte written to the HSO_COMMAND register.

2. The HSO_TIME register is loaded with the sum of the current value of Timer1 and another number. Timer1 will then increment once every 16 clock cycles until it reaches the number loaded into the HSO_TIME register, at which time it will generate a software interrupt.

3. The I/O status register 1 (IOS1) contains information on which software timer expired. In our application this information is necessary when using more than one software timer (e.g.. to implement a digital filter during testing and for adaptive control, which uses much faster sampling to estimate circuit parameters).

Output Ports

Ports 3 and 4 on the 80C196KB are output ports. On the EV80C196, however, these ports are used for memory access, and may not be used for output without off-board latches and decoding [12]. Port 1 and two pins on Port 2 are "quasi-bidirectional", and may be used as output pins [13]. In our implementation, the 8 lines on Port 1 are used for the output, as it turns out that 8 bits provide sufficient resolution. However, it should be noted that in an application not constrained by the evaluation board's use of resources, higher output resolution may be achieved, and control of more than one system on the same microcontroller is feasible. The output mechanism is discussed further in the next section.

3.1.2 D/A Conversion

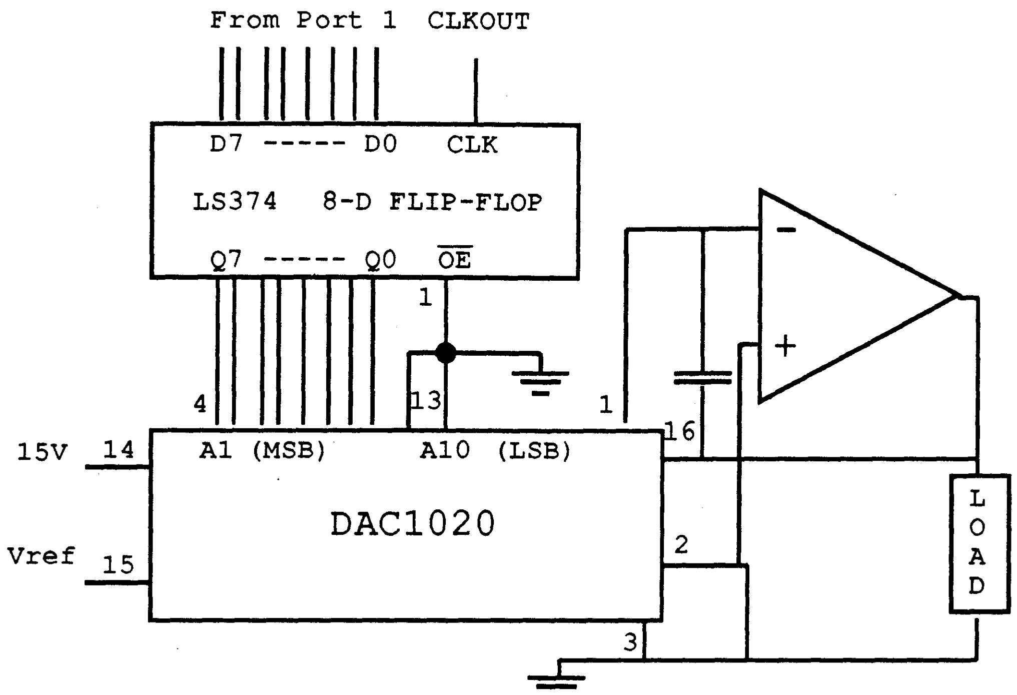
The output of the digital voltage-loop controller needs to be fed to the current-loop controller as an analog signal. A natural implementation of the system computes the commanded current i_P in Figure 1-2 inside the microprocessor and then converts it to an analog signal to send to the current loop controller. This requires, in addition to computing $k[n]$, that the input waveform be sampled and reconstructed to provide the correct sinusoidal shape on the controller output. This extra sampling and processing is demanding in terms of microcontroller power and time. A better method utilizes a multiplying digital-to-analog converter (multiplying DAC). A multiplying DAC outputs a certain function of a digital and an analog input, usually a product. It is used to replace the multiplier in Figure 1-2 with inputs $k[n]$ and $v_{in}(t)$. Figure 3-3 illustrates the multiplying DAC in the implemented circuit configuration.

The latch at the input to the multiplying DAC is used to synchronize the signals to its pins. When used as outputs, the quasi-bidirectional pins will change state shortly after the system clock (CLKOUT) falls. The LS374 Flip-Flop shown clocks in the values of Port 1 pins to the multiplying DAC on the rising edge of CLKOUT. Figure 3-4 illustrates the latch timing. The inputs to the multiplying DAC are guaranteed to all change at the same time, on the rising edge of CLKOUT.

3.1.3 Analog Current-Loop Controller

The current-loop controller is implemented using the Unitrode UC3854. The UC3854 was designed to be used in a fully analog setting to implement both the voltage and current control loops, [16]. Three external control inputs to the chip, A, B, and C, are fed to a function block that outputs the function $\frac{AB}{C}$. This output is used as the current reference to which the input current is controlled through a current-mode controller on the chip, as shown in Figure 3-5. For a purely analog controller, one of the inputs is made to be the input voltage waveform $v_{in}(t)$, another is the scale factor $k(t)$ derived from the output voltage error, and the third (the one in the denominator) is proportional to the square of the input voltage rms value. The system in Figure 1-2 is thereby achieved [16].

For our digital implementation, inputs A and C are fixed in value and B is chosen to be the output of the multiplying DAC from the previous section. This setup is



```
Vout = -Vref(A1/2 + A2/4 + ... + A10/1024)
An = 1 if nth digital input is high.
An = 0 if nth digital input is low.
```

Figure 3-3: Output port 1 and multiplying DAC configuration.

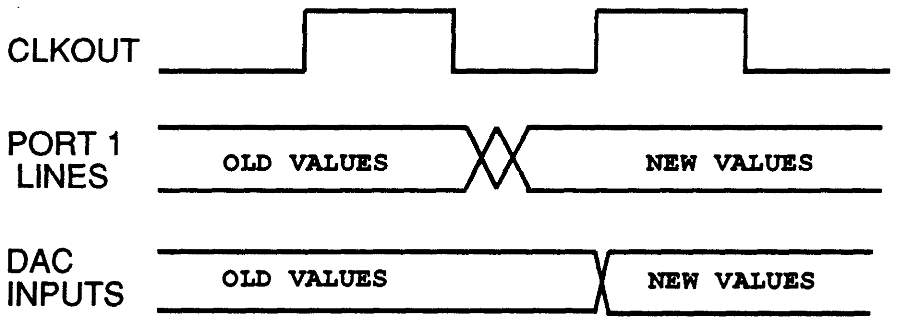


Figure 3-4: Output synchronization.

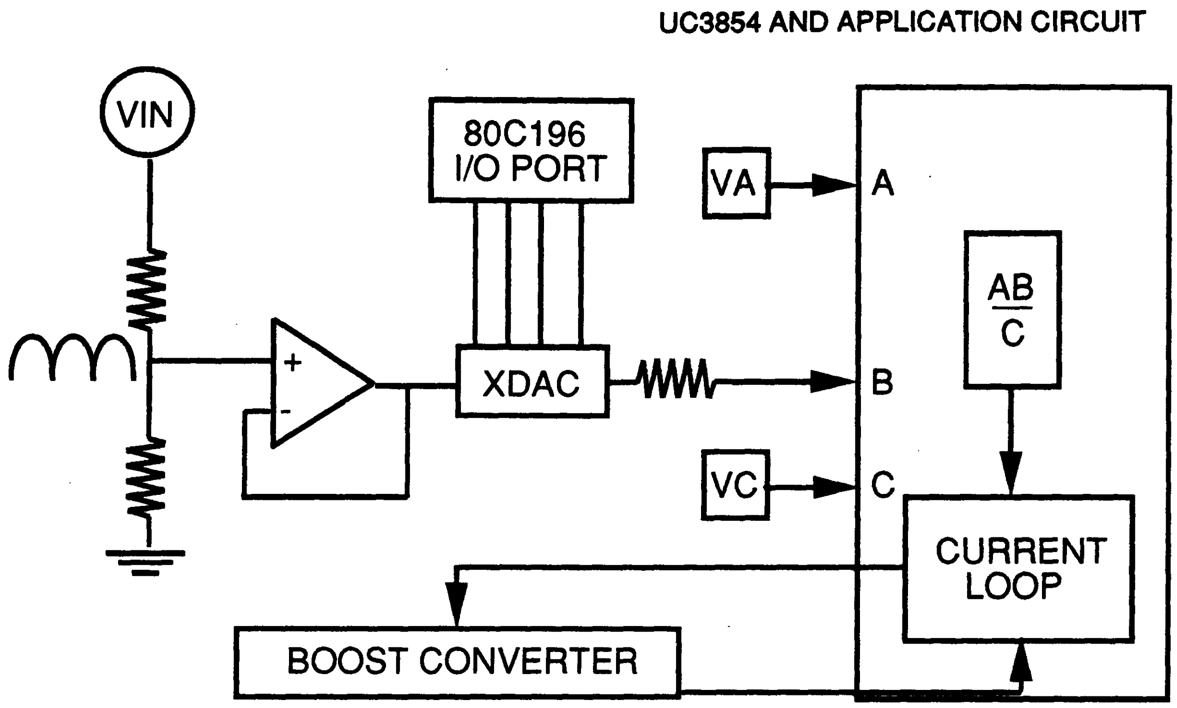


Figure 3-5: Implementation of the current loop.

particularly attractive for testing and development since it allows the flexible limiting of commanded power to safe values defined by A, C, and the upper limit on B. This upper bound and the voltages and circuits used to set it are shown in the appendices.

3.1.4 A/D Resolution Enhancement

The A/D converter on the 80C196KB has a 10-bit resolution. However, these bits are not all found in the same register, as noted above. It therefore requires an extra read operation to get the full 10 bits. Moreover, the microprocessor can be instructed to perform an 8-bit conversion and yield a faster conversion time. However, the extra time incurred is insignificant, and the quantization effects are much less if 10 bits are read. Figure 3-6 shows the simulation of a PI-controlled UPF converter transient and steady-state response with 8-bit input quantization and Figure 3-7 shows it with 10-bit input quantization. (The output quantization specified for the simulation is 8 bits. It turns out that this is the optimal output resolution for the system at hand, as will be argued later under software implementation.) The poles of both systems are at .5.

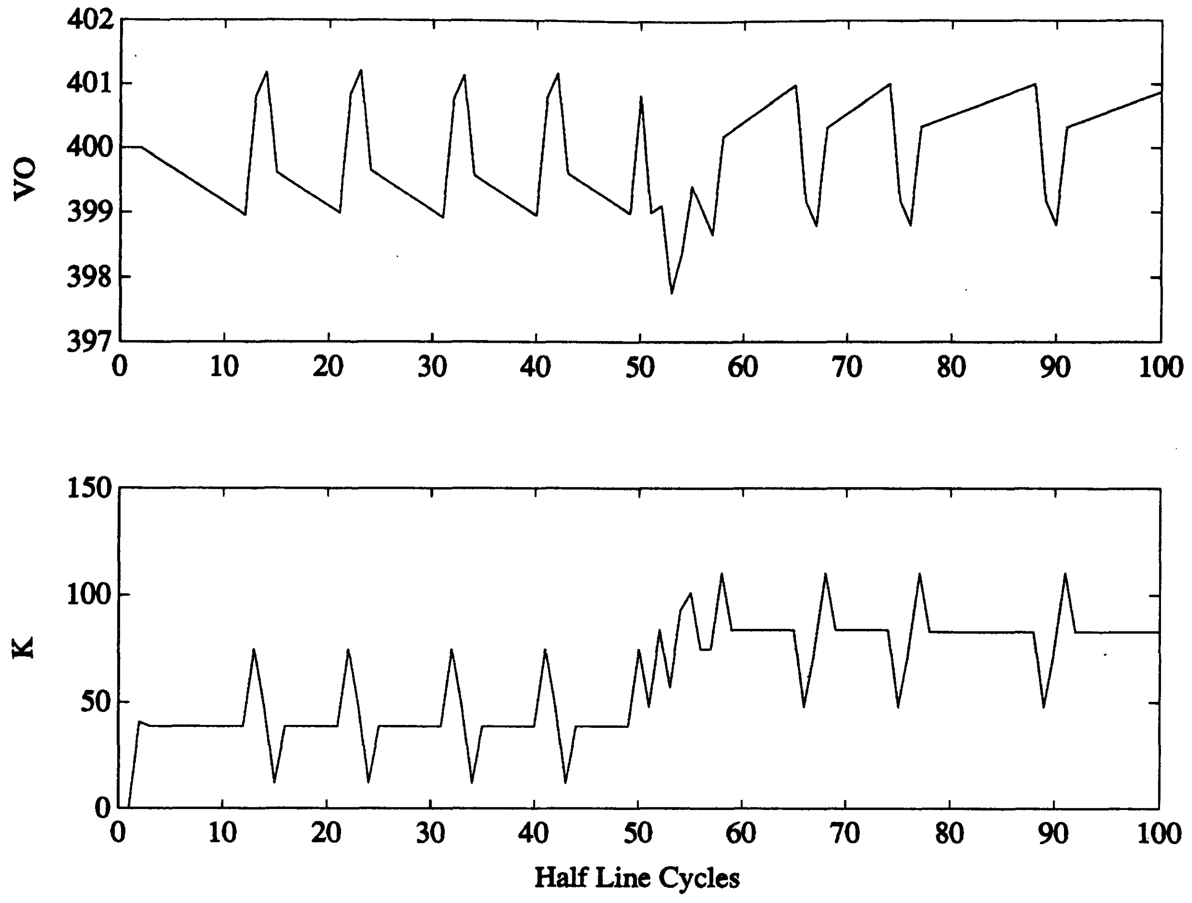


Figure 3-6: Closed-loop system with 8-bit input quantization.

As it turns out, the 10-bit resolution on the A/D does not yield satisfactory results without an additional step. Since only a small portion of the range of voltages (a section around steady state) needs to be resolved for successful PI control within the limits of the control command, it is possible to increase this resolution with the circuit in Figure 3-8. A small range of voltages is mapped onto the 0-5 volt range of the A/D converter, effectively increasing its resolution by a number of bits equal to $log_2(m)$, where m is the slope of the ascending section of the curve. This fix is used in conjunction with the 10-bit A/D converter.

3.1.5 A/D Anti-Aliasing Filter

Although the assumption that the output voltage is essentially flat is good, there still remains some $120Hz$ ripple on the load capacitor. In addition, there are other harmonics in the system due to the use of noisy components in the construction of the prototype. To ensure that upon sampling these frequencies do not get aliased and

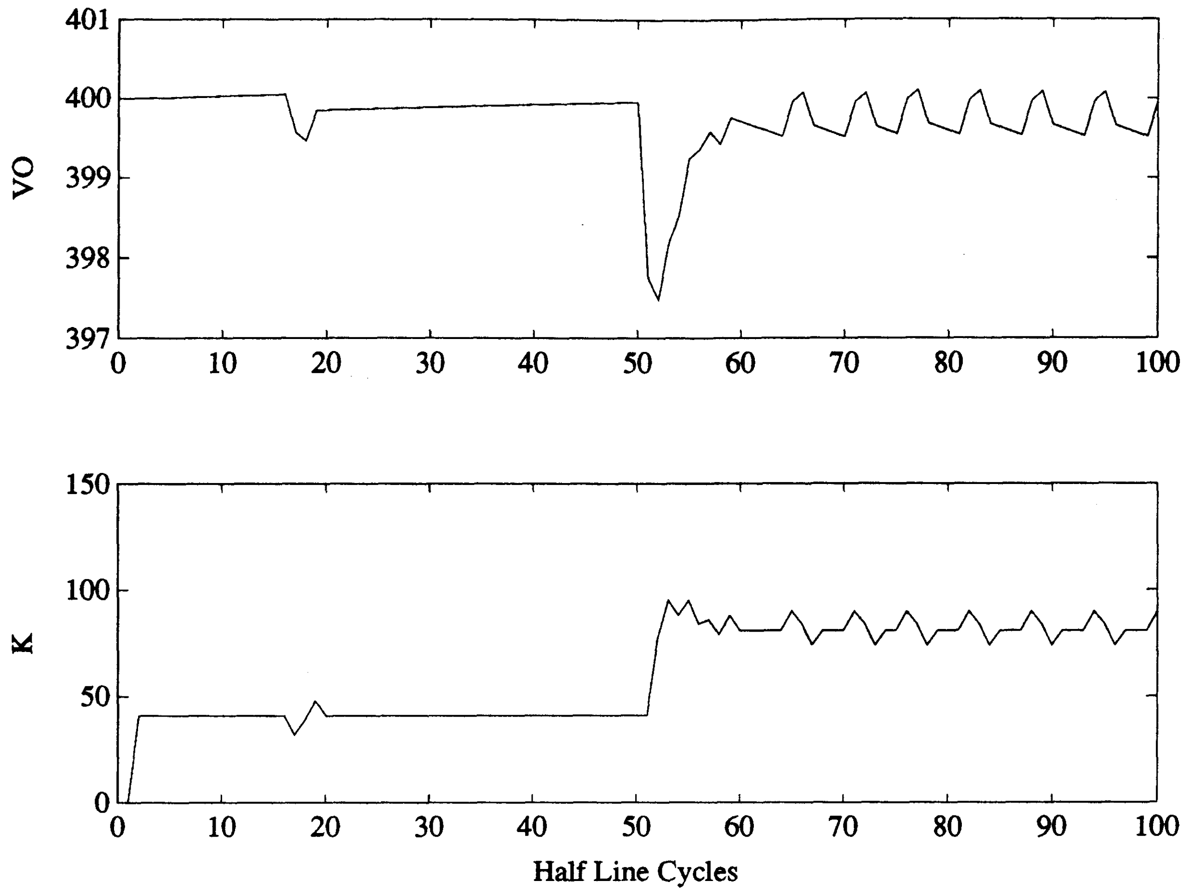


Figure 3-7: Closed-loop system with 10-bit input quantization.

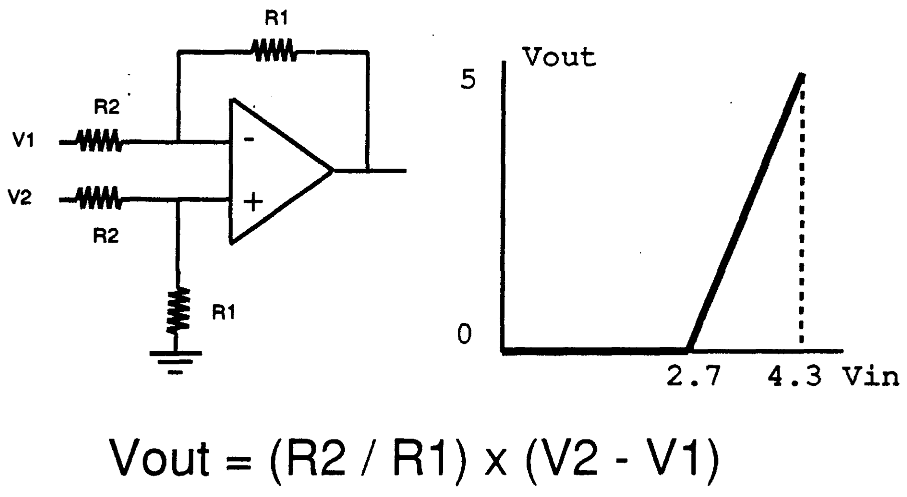


Figure 3-8: Increasing Input Resolution.

corrupt the measurement, an anti-aliasing filter is built to reject frequencies outside the $60Hz$ bandwidth. It is also important to insure that the filter does not cut into the bandwidth of the closed-loop system and change its response to load transients. Design and implementation of the anti-aliasing filter are discussed in the appendices.

3.1.6 The Complete System

Additional circuitry is used to provide buffering, scaling, reference voltages to limit the current commanded by the current-loop controller, and variables required by the adaptive controller. In addition, isolation amplifiers are used to isolate the digital system from the analog components. Detailed schematics of the complete system are shown in the appendices. Figure 3-9 shows a diagram of the overall system. The boost converter uses a load capacitor $C = 470\mu f$ and inductor $L = 1mH$. The capacitor is rated for 450 volts. The load is composed of 3 light-bulbs in series, combining for a total of approximately 3200 ohms. The load may be effectively doubled by dropping an equal resistor in parallel with the light-bulbs. The DC filter is used to provide a measurement of the peak input voltage needed to implement the control as discussed in Section 2.2.3 under Adaptive Control. The inputs to the UC3854 in Figure 3-5 and the analog input to the multiplying DAC are chosen so that the maximum current command at 110 input voltage rms outputs 50 watts of power, which at the lower load level of 3200 ohms produces 400 volts across the load capacitor, sufficiently below its rated maximum.

3.2 Implementation of Control Algorithm in Software

The software implementation of the controller uses fixed-point arithmetic to avoid the extra cost in terms of processing time and memory incurred by including a floating-point library. Care also needs to be taken to prevent extraneous gains from changing the characteristics of the closed-loop system. These gains are introduced by the hardware interfaces or by scaling to enable the use of fixed-point representation. Timing constraints are also of extreme importance when working with sampled-data models. Timing as well as parameter quantization and scaling are at the core of the software design problem.

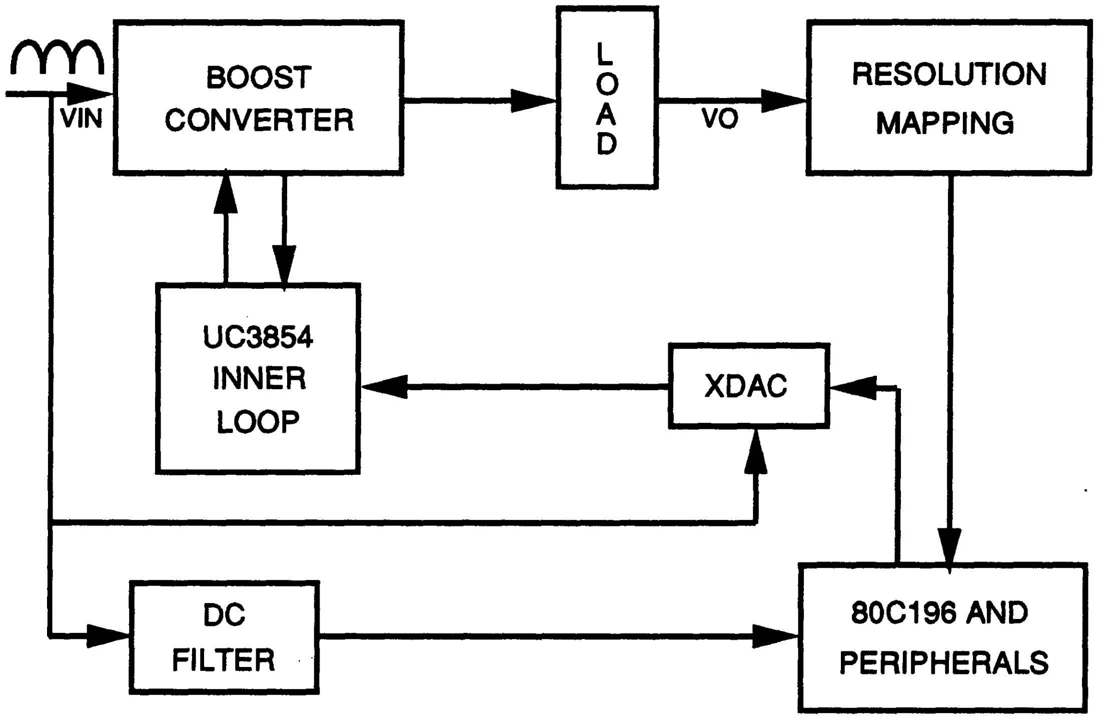


Figure 3-9: Overall structure of hardware system.

3.2.1 Timing

Timing for the controller is implemented using the HSO unit and Timer1 in the manner described earlier. To implement the discrete-time PI controller, the sampling and control output frequencies need to be $120Hz$. To start the system up, Software Timer 0 is set to expire after $8.33ms$. It is extremely important to maintain this frequency as accurately as possible, so the first instruction in the ISR for the software timer interrupt is to reload the timing register with the same value for the next cycle. In this manner, an $8.33ms$ period is insured every cycle, to within the resolution of the timer. The time-specific tasks of this controller, i.e. control output and sampling, are carried out next, before any further processing is executed. In doing so, uncertainty in the time periods between samples or outputs due to varying processing time from cycle to cycle (caused by conditionals in the code, for instance) is avoided. Once all of the necessary variables are sampled, the PI algorithm may be carried out, and the only time restriction on this task is that it be completed within one sampling cycle. It is therefore also included within the software timer ISR. Note that the $8.33ms$ time period is ample time for the controller implemented here. A simple PI computation requires less than $1ms$ to complete. The extra time is then left for estimation and adaptive (or more complex) control, or to be used by the operating system. Shutdown generates an interrupt of higher priority that transfers control to its own ISR when detected. Figure 3-10 illustrates the timing of the implementation.

An important issue in an implementation which uses a digital timer such as the one on the 80C196KB is timer resolution. Timer1 on the microprocessor is updated once every 8 state times of the processor, or 16 clock cycles, with a timing resolution of 1.33 μs. Our sampling and output period can be accurate only to within the limits of this resolution. As a result, the sampled points are not exactly one ripple period apart and our output is not being updated every exactly T_L seconds.

Two schemes were considered for dealing with this problem. The first is to use two different sampling periods, one the smallest time that Timer1 can count that is larger than T_L and the other the largest time that Timer1 can count less than T_L. This would prevent drifting along the ripple on the output voltage and at times controlling the peak to V_o while at others controlling the trough to that level. The second scheme would involve hardware circuitry to to detect the zeros of the input voltage and use them to trigger the sampling and control output operations. These two systems were relatively easy to implement but were found to be unnecessary. The large output capacitor results in an effectively DC output voltage, and the timer resolution of $1.33\mu s$ is high enough for us to ignore the timer quantization issue. However, in a different system this may not be the case and one must be aware of this problem.

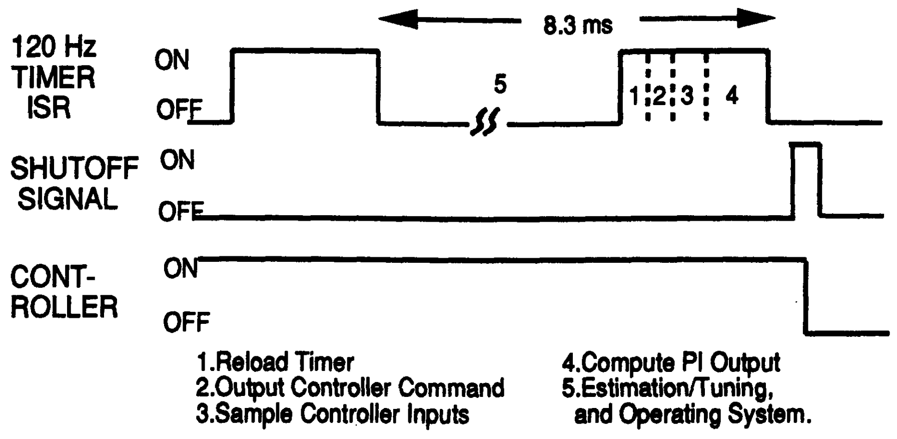


Figure 3-10: System timing.

3.2.2 Scaling and Digital Parameters

In order to achieve the correct implementation of the PI controller, the end result of our computation should be, as in Figure 2-2:

$$k[n] = \frac{C}{T_L V^2}(h_1(v_o^2 - V_o^2) - h_2\sigma[n]) \tag{3.1}$$

Several problems arise in the digital implementation of this equation. Quantization of inputs to the controller occurs as a result of A/D conversion, and output quantization is a function of the number of bits that are feeding the input to the multiplying DAC, as was demonstrated in Figures 3-6 and 3-7. The equality above is therefore never met, and the desired zero-error steady state is not achieved. Instead, the output voltage and input current exhibit cyclic behavior around the "infinite-precision" or (analog) steady-state. Quantization error is also a function of the speed of the system, as is further discussed below.

The use of fixed-point arithmetic also dictates that many of the parameters of our system be scaled to be represented by an integer (as many of them are smaller than 1). This scaling, as well as other extraneous gains due to hardware, A/D conversion, or D/A conversion enter the loop in the transfer relation between the inputs to the controller and its outputs. The software must be designed to undo any such undesirable gains and ensure the relationship in 3.1 as the controller transfer relation. Define the following parameters:

- Div_o is the fraction of the output voltage that is fed into the resolution mapping circuit. The output voltage is scaled down by a voltage divider to a level where it can be used in the control circuit. See Figure 3-8.

- Div_i is the fraction of the input voltage peak value V that is measured by the A/D converter to use in the control equations.
- m is the slope of the linear region in the resolution mapping circuit for v_o. See Figure 3-8.
- x_{int} is the x-intercept in the resolution mapping function. See Figure 3-8.
- h_{1d}, h_{2d} are the digital gains specified in the microcontroller software.
- V_{od} is the reference voltage as specified in the microcontroller software.
- Acc is the accumulator value in the microcontroller software.
- Acc_{max} is the maximum accumulator register value.
- G_{dig} is a gain used to shift the digital gains to retain more of their significant digits.
- k_{max} is the maximum command the voltage loop can issue as defined by the current limit in the setup and the peak input voltage.
- σ_{max} is the maximum integrator value from the discrete time controller described in Equation 3.1.
- F_{AD} is the A/D transfer gain, which for a 10-bit A/D with a 5 volt input range is 1023/5 (the result of the A/D conversion is quantized of course).

Controller Inputs

The inputs to the controller, namely v_o and V, are transformed through peripheral circuitry and A/D conversion. The output voltage, v_o, is divided down to a level that a digital circuit can handle. Only a fraction, Div_o, of v_o is fed into the control circuitry. This fraction is the input to the resolution mapping circuit of Figure 3-8, with slope m and x-intercept x_{int} as listed above. The output of this circuit is then fed into the A/D converter on the 80C196KB, with a quantized transfer gain F_{AD}. In order to recover a multiple of v_o for computing the error value, the negative offset acquired in this process is added back in software. The result is then squared, as required by our linear model, which dictates that the square of the output voltage is the state variable.

Figure 3-11 shows the propagation of the output voltage through the system until it is compared with the reference to detect the error. For each block in the figure, x denotes its input and y its output. The boxed functions are the result of the hardware system and the encircled ones are implemented in the software. The result of the overall transformation is:

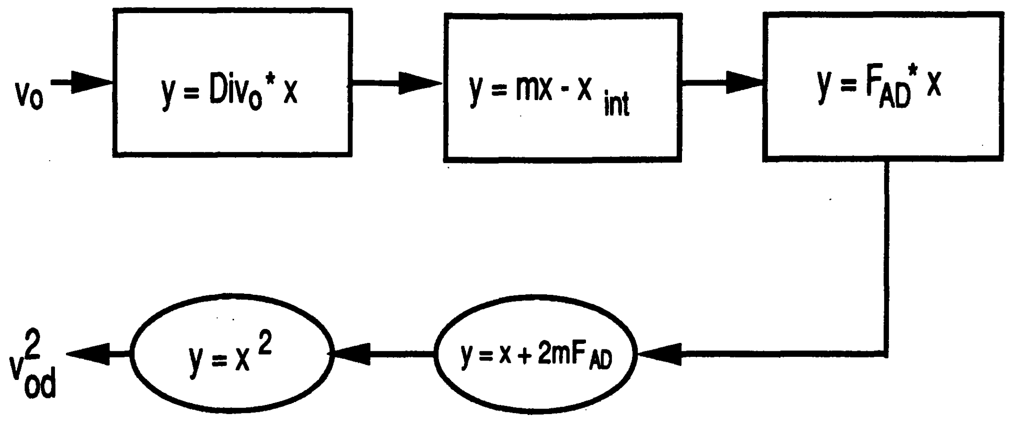


Figure 3-11: Transformation of v_o to the digital value of v_o^2.

$$v_{od}^2 = \{[((v_o \times Div_o) - x_{int}) \times m \times F_{AD}] + 2mF_{AD}\}^2 = m^2 Div_o^2 F_{AD}^2 \times v_o^2 \qquad (3.2)$$

and the reference, V_{od} is defined with the same transfer function.

$$V_{od} = m^2 Div_o^2 F_{AD}^2 \times V_o^2 \qquad (3.3)$$

The input voltage peak V is divided through a voltage divider which sends a fraction Div_i to the A/D converter. On entry into the microprocessor, this value is multiplied by a digital gain, G_{dig}, that serves to cancel a gain in the digital parameter that is used to improve the accuracy of the fixed point computation (see Controller Gains and Parameter Limits below). A squaring operation follows in accordance with the controller developed earlier. Figure 3-12 shows this flow of transformations. The resulting digital value is:

$$V_d = G_{dig}(Div_i F_{AD} V)^2 = G_{dig} Div_i^2 F_{AD}^2 \times V^2 \qquad (3.4)$$

Controller Gains and Parameter Limits

Given the analog gains, the digital gains are chosen as follows:

$$h_{id} = round\{\frac{C}{T_L} \times G_{dig} \times \frac{Div_i^2}{Div_o^2 m^2} \times \frac{255}{k_{max}}\} \qquad (3.5)$$

Multiplying by $\frac{C}{T_L}$ before processing as shown above saves a multiply operation.

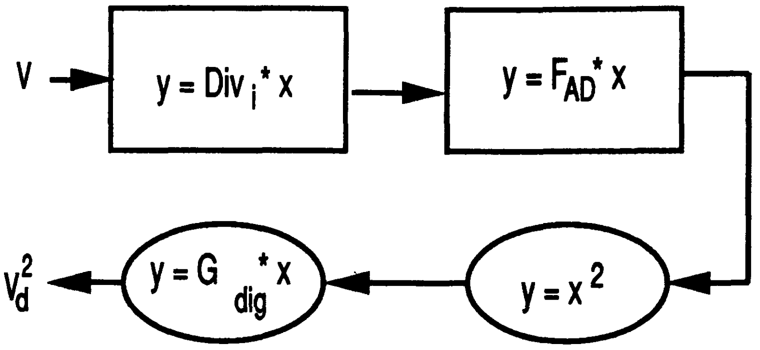


Figure 3-12: Transformation of V to its digital value.

The gain of G_{dig} serves to provide a more accurate representation of the quantized parameter. The fourth gain stage serves to map the values of k, the control command, to the range from 0 to 255 on the 8-bit output port. The 8-bit output resolution was found to be sufficient for the implemented system and higher resolution rquiring extra digital hardware was not warranted, as shown in the appendices.

This digital output is then fed into the multiplying DAC which commands maximum power for a digital number of 255. The DAC effectively scales the output of Port 1 by $k_{max}/255$. Combining these results, the digital controller then performs the operation:

$$k[n] = \frac{k_{max}}{255V^2}[h_{1d}(v_{od}^2[n] - V_{dd}^2) + h_{2d}Acc[n]] \tag{3.6}$$

$$k[n] = \frac{k_{max}}{255V^2}[h_{1d}(v_{od}^2[n] - V_{dd}^2) + h_{2d}\sum_{i=0}^{n-1} v_{od}^2[i] - V_{dd}^2] \tag{3.7}$$

which reduces to the desired control algorithm in Equation 3.1.

The extra gain factor, G_{dig}, is limited by the register sizes and possible overflow after multiplication by the error or the accumulator value. The maximum value of the integrator, σ_{max}, is chosen as the value for which the maximum command is generated in the steady state, with an output voltage error of zero.

$$\sigma_{max} = \frac{k_{max}T_L V^2}{h_2 C} \tag{3.8}$$

The maximum value of the accumulator, Acc_{max}, is then chosen accordingly by multiplying Acc by the ratio of digital error to analog error (the resulting transfer function from Figure 3-11.

$$Acc_{max} = m^2 Div_o FAD^2 \times \sigma_{max} \tag{3.9}$$

This value when substituted into Equation 3.6, must not cause the result of the numerator computation to overflow the register used to hold its value. To insure that this never occurs, the worst case scenario, in which $v_o = 0$ was checked for the implemented parameters with a 32-bit signed integer register type.

3.2.3 Other Controller Characteristics

Features that were introduced in the design section include Bang-Bang control, Band control, Integrator-anti-windup, and Soft-startup. Dead-beat control, as mentioned earlier, is not feasible in this application because it requires very large current to achieve a two-cycle transient decay. Safety features that shutoff the controller when it is above the safe caopacitor voltage level and apply the maximum power level when the voltage is below a minimum are used, and are what is referred to in the design section as Bang-Bang control.

A digital filter was also implemented during the testing stages. The simplest form of a filter, which used a second, faster, software timer to collect more samples of the output voltage and average their value, was used. The digital filter was found to improve performance slightly in terms of quantization. However, the averaging technique would have slowed down the system response, and a more complicated filter would conflict with the goal of achieving a simple digital implementation. However, digital filtering is an option that may be used depending on the application, and the code used for its implementation is included in the appendices.

The Band controller, Integrator-anti-windup, and soft-startup are discussed below, along with the speed of the system, which was compromised due to quantization effects.

Integrator-anti-windup and Soft-startup

As mentioned earlier, integrator windup is a problem that arises in PI controllers when the controller output is saturated. When saturation occurs, the integrator is effectively isolated from the closed loop system, and is therefore unstable [10]. Its value may increase excessively during that period, and it may take some time to "unwind" it from this state, which may result in undesirable behavior of the system (in this case

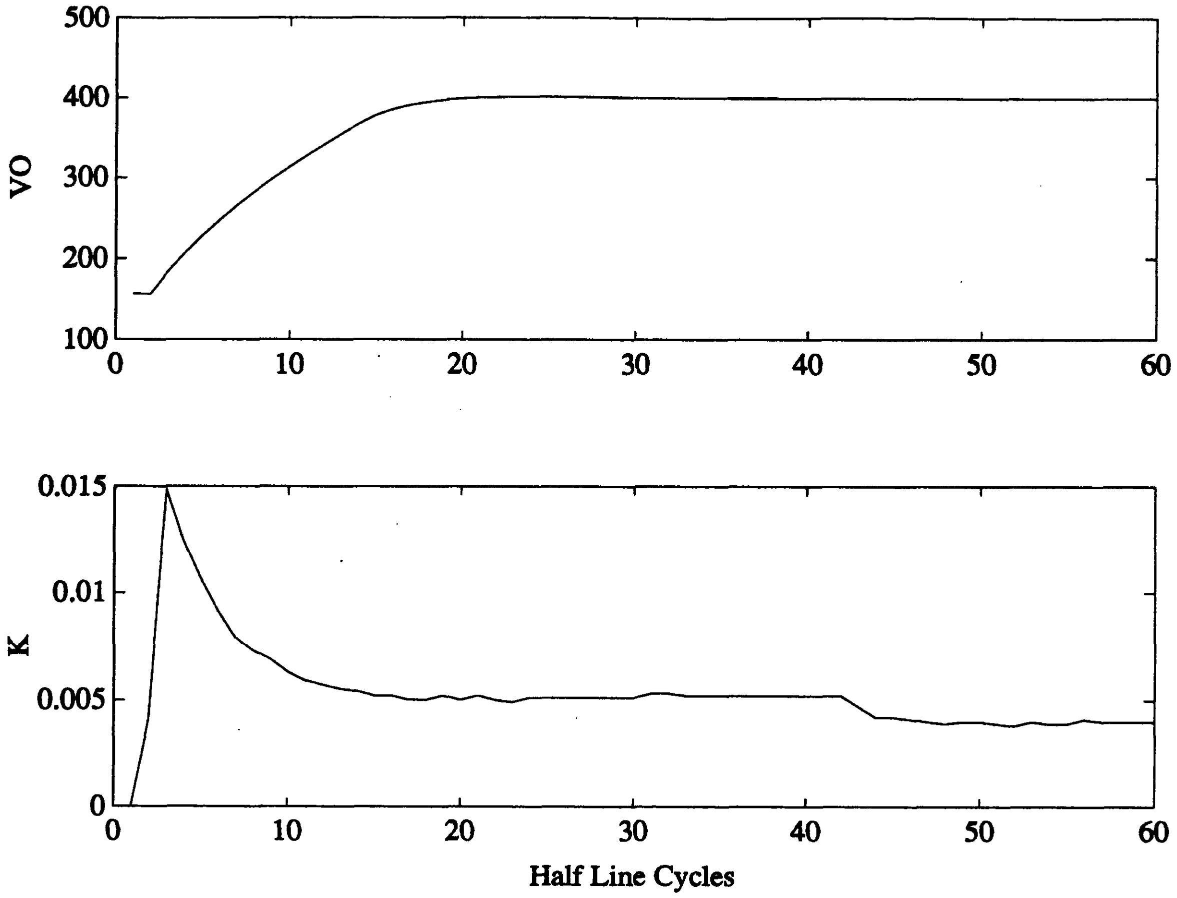


Figure 3-13: Startup transient with anti-windup mechanism.

voltage overshoot). As an anti-windup mechanism, the accumulator is disabled when the output of the digital controller is saturated. Simulations of the startup transient of the system with the anti-windup scheme implemented are shownin Figure 3-13. The voltage does not overshoot as it did in Figure 2-8. However, the current transient is still very sharp on startup.

The soft-startup mechanism used simply steps up the reference voltage to its final value in a few cycles. The result is that the controller does not detect as large an error as it would with the true reference, and therefore does not command too much current instantaneously. Figure 3-14 shows the simulation of the startup transient with this soft-startup mechanism employed in addition to the anti-windup scheme. Although the current transition is much smoother, the voltage overshoots. The simple anti-windup scheme used fails when the reference is stepped up gradually during soft-startup. Since the output does not saturate, the integrator is not disabled for the period of the startup. It therefore accumulates error throughout this period, and is wound up past the point of the steady-state integrator value when the steady-state is

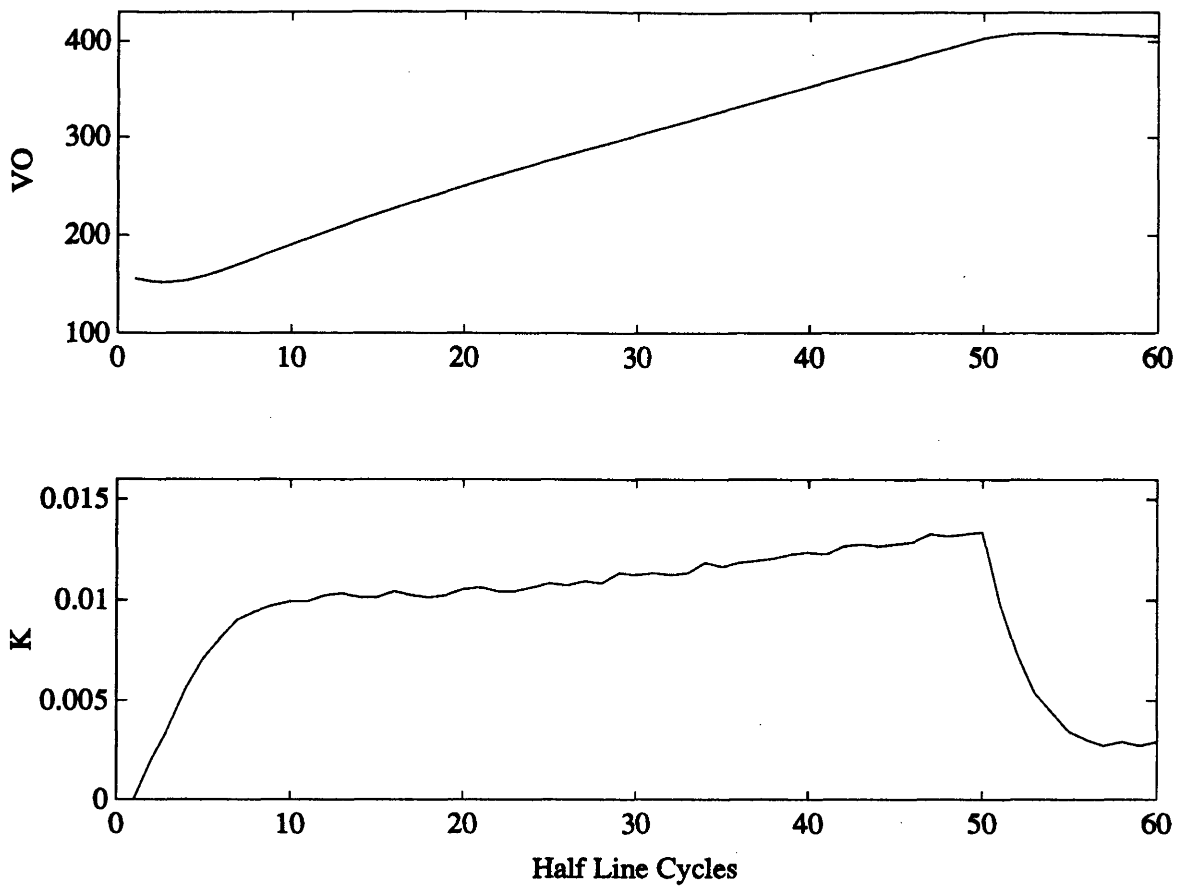


Figure 3-14: Startup behavior with soft-startup and initial anti-windup scheme.

reached, causing a voltage overshoot. To solve this problem, the integrator is disabled for the duration of the startup. A simulation of the result of this design is shown in Figure ??. The current transition is soft and the output voltage does not overshoot. The figures here are for 10-bit quantization, without the hardware resolution mapping circuit.

Band Control

The Band control algorithm, as mentioned earlier, is used to eliminate quantization-induced oscillations by allowing the steady-state value to settle to a value that is close to the reference voltage but not equal to it. If that level is tolerable by the system, this oscillation-free state can be used to observe the ripple on the output voltage to be used in the capacitance estimation discussed later. A general algorithm is presented below:

Define the following variables and constants

- ϵ is the distance in volts away from V_o within which the system is allowed

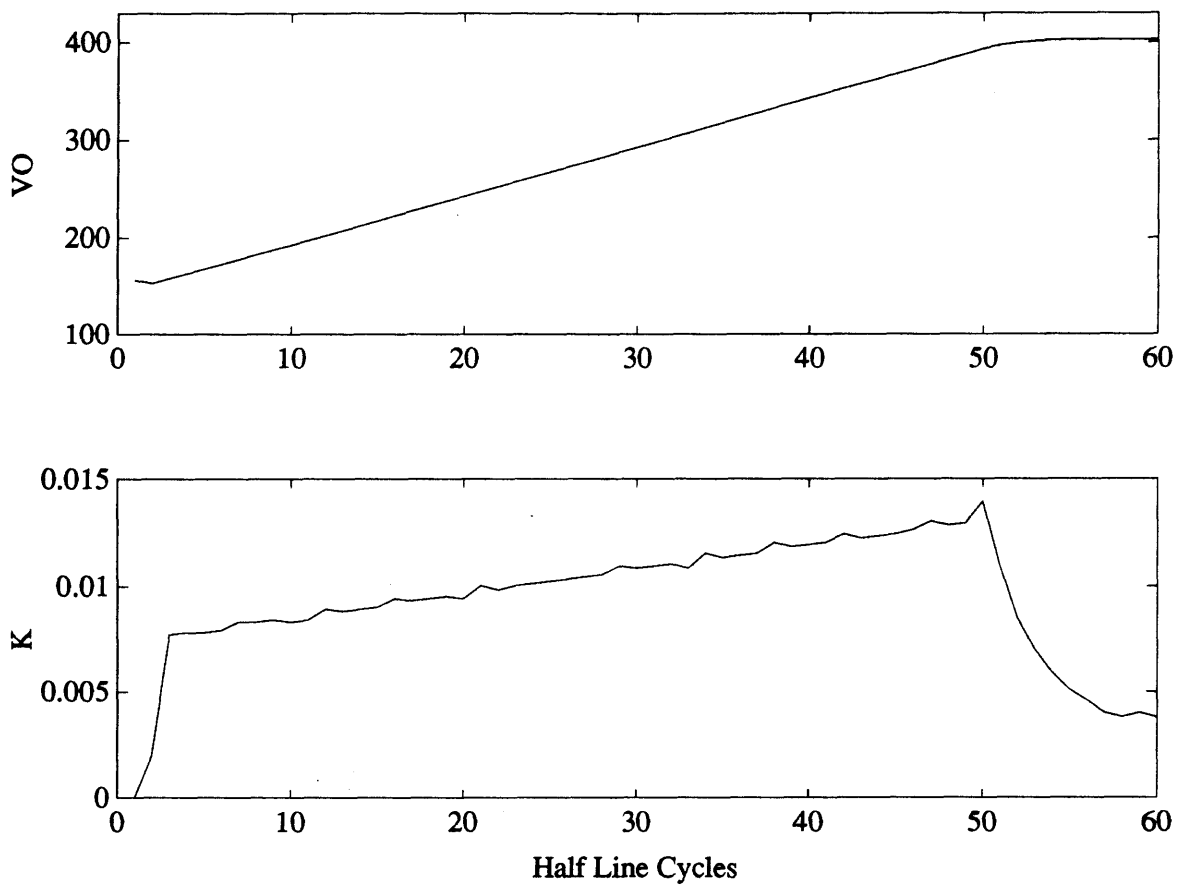


Figure 3-15: Startup behavior with soft-startup and modified anti-windup scheme.

to settle down. In other words, all the points within an ϵ away from V_o are considered an acceptable steady-state. ϵ is of course a multiple of the system quantum of input.

- n is the number of consecutive cycles that the system has spent in steady-state, with $V_o - \epsilon < v_o < V_o + \epsilon$.
- S_{cmd} is the sum of controller commands accumulated over consecutive "steady-state" cycles.
- n_{max} is the number of cycles sufficient for computing an estimate for the steady-state command.
- K_{nom} is the "nominal" command, used in the steady-state operation of the circuit.
- k is the PI controller command on the current cycle.

The sequence of control commands to be included with the PI controller and Bang-Bang controller is:

1. Set the values of ϵ and n_{max}.
2. Initialize the variables: $S_{cmd} = 0, n = 0, K_{nom} = 0$.
3. Sample v_o.
4. If v_o is NOT within an ϵ of V_o then apply normal PI command k and goto 2.
5. If v_o is within an ϵ of V_o and $n < n_{max}$:

$$S_{cmd} = S_{cmd} + k \qquad\qquad n = n + 1 \tag{3.10}$$

   Then apply PI command k and goto 3.
6. If $n = n_{max}$ set $K_{nom} = S_{cmd}/n_{max}$ and use K_{nom} for the controller command. When K_{nom} is used, the integrator should be disabled.
7. Goto 3.

The algorithm in its roughest form above was used in the implementation, with details and actual values used shown in the appendices and the microprocessor code. Slight variations in the algorithm may provide optimality under certain conditions. The value of ϵ may be chosen differently depending on the noise level and the error tolerance of the system, and similarly for the value of n_{max}. Situations where two values for ϵ, one used to determine when the system has entered the "steady-state" regime and another, looser one to decide when a disturbance has taken place, may be used. The tight value may be required to accurately estimate the steady-state command. However, the system may be capable of tolerating larger disturbances than dictated by this narrow band. The use of two distinct values in such a manner results in a "hysteresis" type function, and would not be difficult to implement as an additional feature.

Other obvious variations on this algorithm would update the estimate for the steady state more frequently, or would switch out of the steady-state regime periodically as a safety measure. More complicated adaptive control may slow down the system while it is making its measurements in order to reduce the quantization in the control command (Which is a function of the feedback gains as is shown below). Also, there is no need to reassign $K_{nom} = S_{cmd}/n_{max}$ on every cycle, but it was done so here to illustrate the algorithm in the simplest manner.

This technique works effectively only with levels of current quantization significantly better than those of the 10-bit A/D system with poles at .5 or even .8. The algorithm works successfully in the implemented circuit, which uses the analog resolution mapping circuit and has system poles around .9 or so. The results of this implementation are presented with the experimental results and in the appendices.

Trading off Speed for Less Quantization Error

It has been shown that the A/D resolution is a factor in determining the level of quantization error in the output voltage, and therefore in the steady-state current. The speed of the system is another variable that effects different quantization error levels, especially in the input current. The gains of the feedback controller determine how much current is commanded from the input in response to a an error in the voltage. Whether this error is due to a disturbance in the load or to quantization is transparent to the controller. It will therefore ask for a change of current in response to quantization error, and this change will be bigger if the gains are larger. An obvious tradeoff is therefore the speed of the system vs. the level of quantization in the system variables. The final implementation of the system had to be slowed down somewhat from the initial design in order to bring down the levels of quantization to tolerable levels. An example of how the speed of the system can affect quantization levels is shown in Figure 3-16. The input quantization level is 10 bits, and the poles have been set at .8. In comparison with Figure 3-7 which has the same A/D resolution but has poles at .5, this system exhibits much less quantization in the steady-state current, and has fewer harmonics in the output voltage as well.

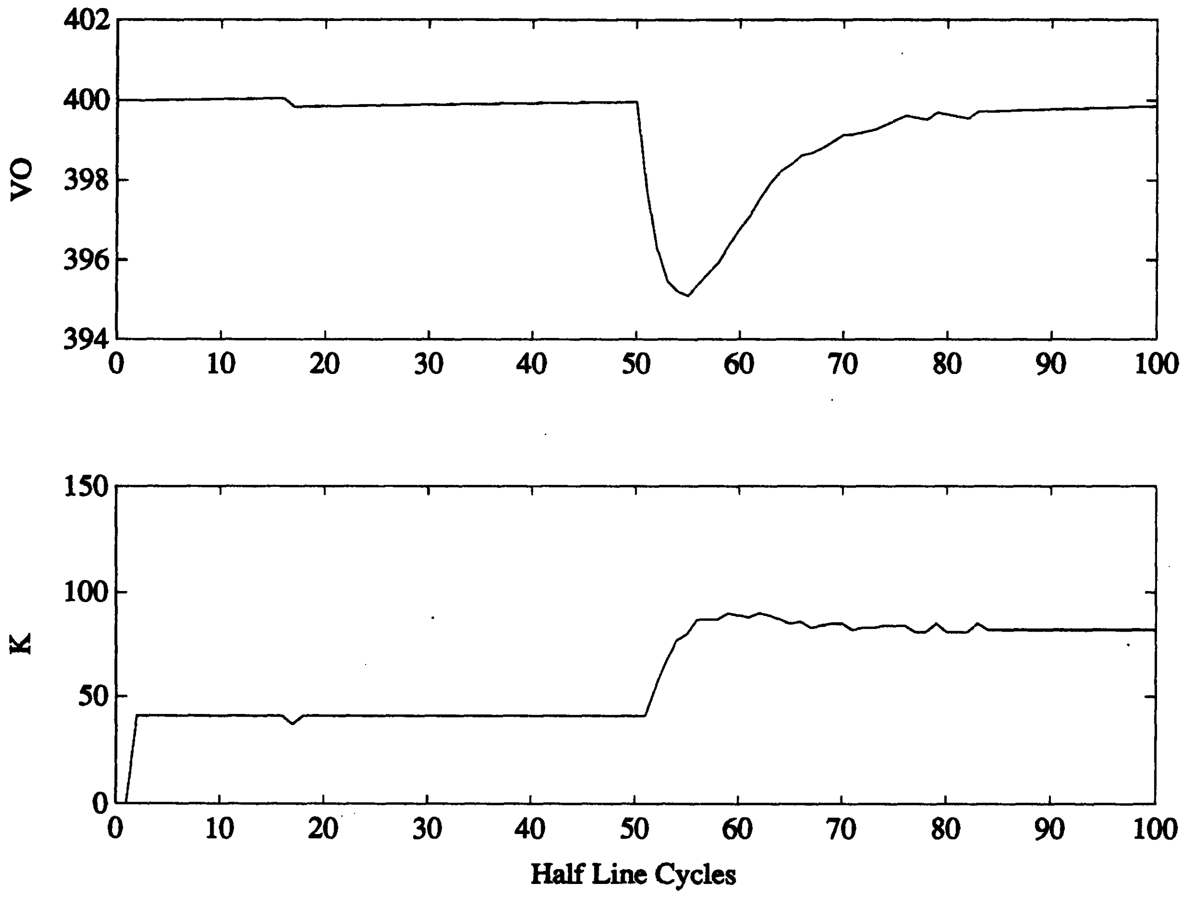


Figure 3-16: 10-bit quantized system with poles at .8.

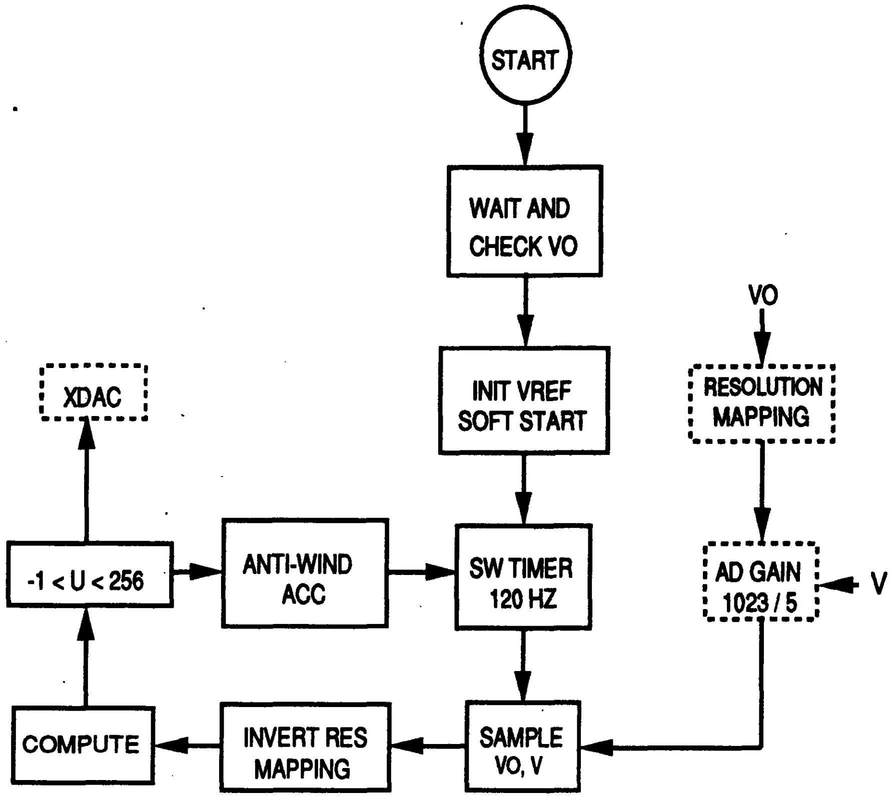


Figure 3-17: Software structure.

3.2.4 Overall Software Implementation

A simplified overall view of the software implementation is shown in Figure 3-17. The timing is accomplished as explained earlier, and the software is interrupt driven. The diagram shows the general operation of the code as well as an example of how extraneous gains produced in the hardware are offset in the software. The initial check on the output voltage is a precuationary check in case of a malfunction in the power supply. If the level of the boost converter output voltage does not rise to slightly below that of the peak input voltage without closing the controllable switch, then the switch is not closed. The computation in the diagram includes deciding the region of operation, which control scheme to apply, and the resulting output. Many of the details have been left out here, but are described in the appendices in detailed flow charts and documented code.

Chapter 4

Design and Implementation of Adaptive Control

As discussed in Chapter 2, the system at hand provides a good test-bed for parameter estimation and adaptive control techniques. A digital system has the ability to execute the complex computations necessary, and as shown below, uncertainty in the circuit parameters of the power supply provides a problem to tackle.

4.1 Motivation

The need for adaptive control in the system at hand arises as certain circuit parameters may vary over time or under specific conditions. In particular, the load capacitance may be imprecisely known or may change due to variations in the connected loads. As is clear from Figures 1-2 and 2-2, the capacitance value is an important parameter in the controller design. Errors in knowledge of this value will degrade the performance of the controller, as the poles of the closed-loop system move off their desired locations.

In the controller shown in Figure 2-2, the capacitance value multiplies the sum of the proportional and integral parts. If the assumption of perfect knowledge of the capacitance is removed, the value of C in the controller is replaced by $\hat{C}$, the best available estimate of the capacitance. This new system is equivalent to the old one, but with new gains, g_i, defined as:

$$g_1 = \frac{\hat{C}}{C} h_1 \qquad g_2 = \frac{\hat{C}}{C} h_2 \tag{4.1}$$

The roots of the characteristic polynomial as defined in Equation 2.21 are then:

$$z = \frac{(g_1 + 2) \pm \sqrt{g_1^2 + 4g_2}}{2} \tag{4.2}$$

This expression simplifies to:

$$z = \frac{1}{2}\{(\frac{\hat{C}}{C}h_1 + 2) \pm \frac{\hat{C}}{C}\sqrt{h_1^2 + 4\frac{C}{\hat{C}}h_2}\} \tag{4.3}$$

Clearly, given that the gains are negative for this system, a decrease in the load capacitance moves the poles apart on the real axis and may move one of them outside the unit circle or onto the negative real axis, while an increase in the capacitance increases the negative term under the square root sign, moving the poles off the real axis. Figure 4-1 demonstrates the transient response obtained (in a simulation) in the first case, with the poles at $\pm\sqrt{2}/2$. Compare this simulation with that in Figure 2-4. As shown in Chapter 5, an error in the estimate of the capacitance may result in excessively high gains which, in addition to moving the poles of the system, increase the quantization error in the digital system.

4.2 Design and Implementation

4.2.1 Estimation

To tune the controller to changes in the load capacitor, its value must be estimated. This information is available in the ripple on the output voltage. Reference [11] defines an approximate relationship between the output voltage ripple amplitude ϵ and the capacitance:

$$\epsilon \approx \frac{P}{\omega_2 V_o C} \tag{4.4}$$

where ω_2 is twice the line frequency. Figure 4-2 demonstrates these dependencies of the ripple on the load and the capacitance. The simulation uses the T_s-sampled model and simulates the quantization effects in the digital system.

The value of the ripple is very small compared to the DC level of the output, which causes problems in resolving the ripple at the input to the A/D converter on the microprocessor. To get around this difficulty, the ripple frequency is filtered out and sampled through a different A/D channel, as is shown in the Appendices C and E. The rest of the hardware necessary is a current sensor to allow the measurement of the output load power, which is part of the relationship above (also shown in the Appendix E). As it turns out, a smaller load capacitance was needed to get a mea-

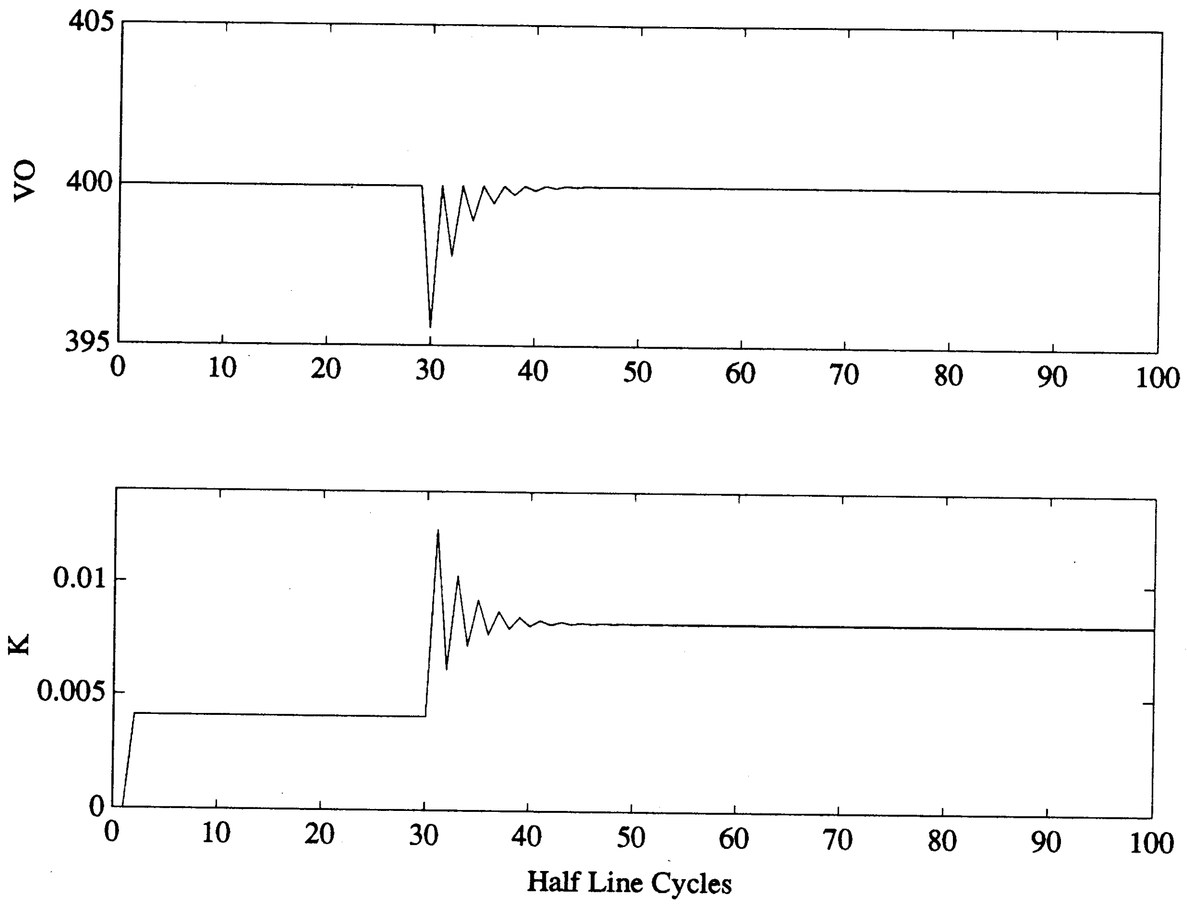


Figure 4-1: Load transient response with $\hat{C} > C$.

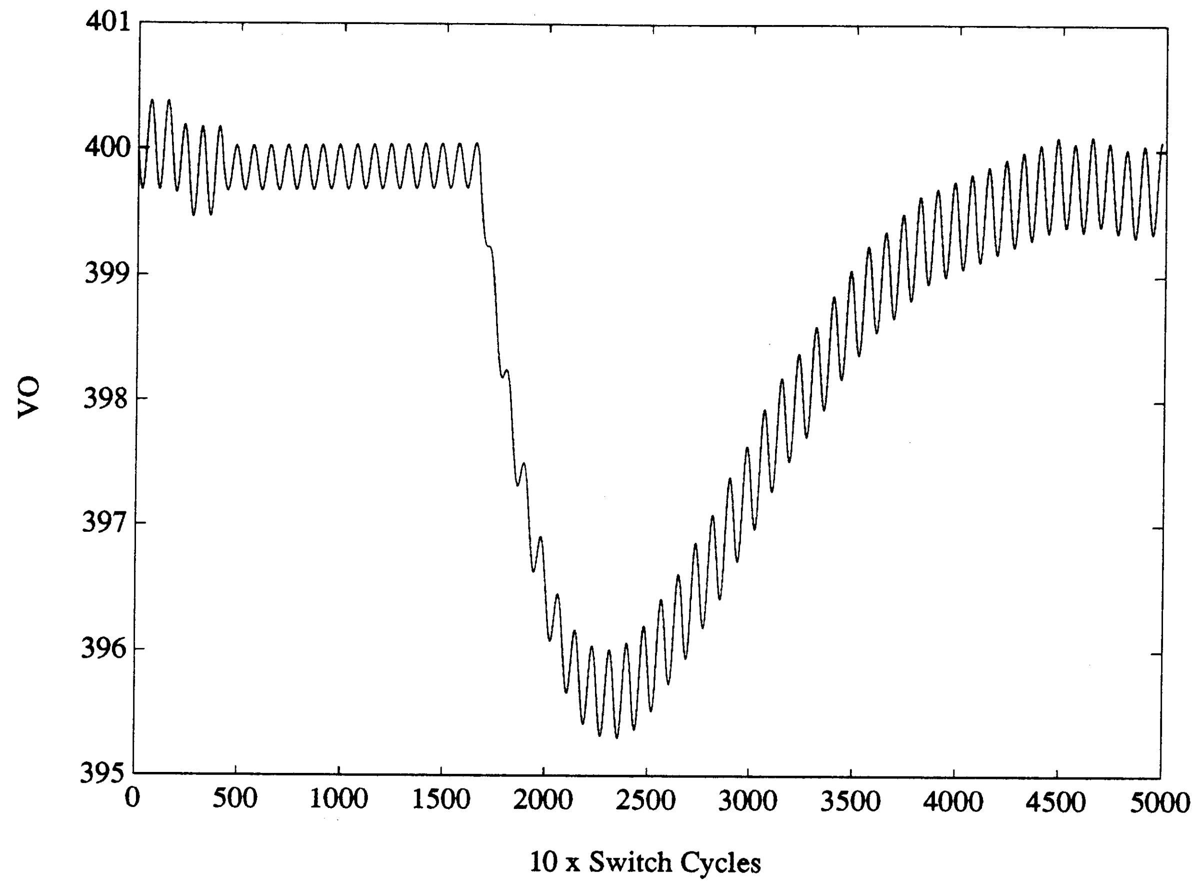


Figure 4-2: Ripple-response to capacitance doubling followed by load doubling.

surable signal for our experiments.

Detecting the Ripple Amplitude

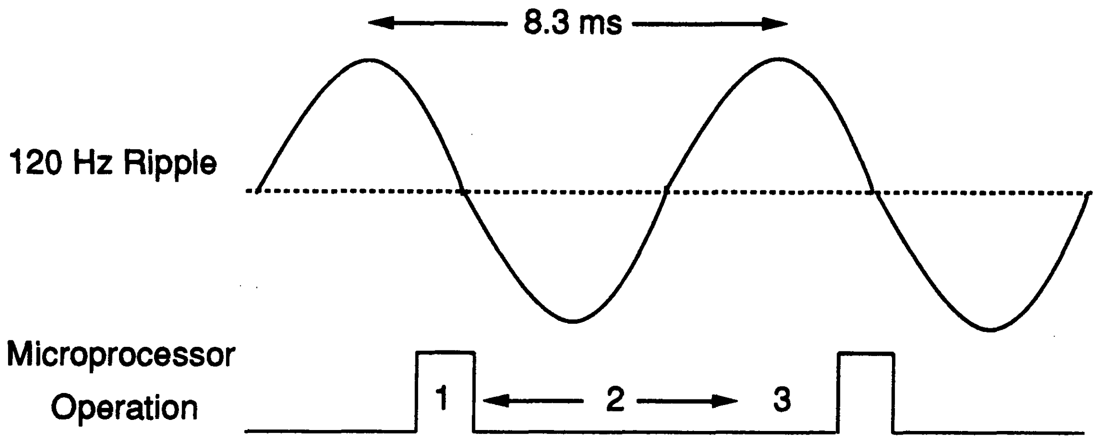
The first step in estimating the capacitance value is detecting the other quantities in the relationship in 4.4. The current is sensed as a voltage off a small resistor in series with the load. The ripple is seen by the A/D converter through a band-pass filter. One way to estimate the ripple amplitude is through interpolation based on a few samples of the waveform. However, that would necessitate the use of both the math and floating-point libraries with the microprocessor, which would slow down the operation and result in higher cost in terms of memory. Moreover, the math library is not yet availabe for the IC96 compiler. Instead, a simple sampling scheme that sweeps the waveform in search of the peak was used. This algorithm is described below and illustrated in Figure 4-3.

1. When the 8.33 *ms* timer (which is not synchronized with the ripple or the input current) expires, output the control command computed from the previous cycle and sample the output voltage. If the state of the controller and the output voltage indicate steady state (assuming operation in the band around steady state, where the command is locked and chosen as an average of steady state commands, see Sections 2.2.3 and 3.2.3), then goto 2. If not steady state, then sample input voltage, compute PI command, reinitialize algorithm variables, and repeat 1.

2. Sample the (filtered) ripple on the A/D channel around 100 times, with approximately 50 μs between samples, keeping the largest and smallest values. This sweeping of more than half a period guarantees either the peak or the trough will be detected. Store the larger in absolute value between the maximum and minimum points detected. After the last ripple sample, sample the load current A/D channel. If this sampling has been carried out for 20 consecutive cycles, goto 3, otherwise return to 1.

3. Average the ripple samples from the 20 cycles of measurement to filter out voltage spikes or other noise. (The number of cycles used in the averaging may be varied depending on the nature of the noise in the system and the desired frequency of estimation.) The load current measurement may be averaged as well, although in the implemented system this measurement is fairly clean and only the last measurement was kept.

4. Compute the capacitance as explained below, then restart the estimation cycle. Note that this estimation cycle is reinitialized if a disturbance takes the system out of the steady-state.



1. **Short Controller Cycle - Output command, sample vo, detect steady-state, then halt and hand over to estimator.**
2. **Sample ripple. Note that as long as this spans more than half of the 8.3 ms period, the amplitude will be detected.**
3. **Sample load current.**

Figure 4-3: Ripple Detection.

Computing the Capacitance

The value read in as the voltage off the current sense resistor in series with the load is:

$$v_l = R_{sense} \times i_l \tag{4.5}$$

This is then sampled through the A/D converter with gain F_{AD}, as discussed in Chapter 3. The output ripple is also sensed through three second-order filters implementing a band-pass filter, as discussed in the Appendix C. A gain stage is included to boost the ripple to an easily measurable value. The combination of the transfer functions of the filters and the gain stage for 120 Hz frequency are lumped into a single gain term, G_{rip}. This measurement is also made through the same 10-bit A/D converter. The resulting digital parameters are then:

$$i_{dig} = F_{AD} R_{sense} \times i_l \tag{4.6}$$

$$\epsilon_{dig} = F_{AD} G_{rip} \times \epsilon \tag{4.7}$$

The following expression, derived from 4.4, is then used to compute the capacitance:

$$\hat{C}_{dig} \approx \frac{G_C i_{dig}}{\omega_2 \epsilon_{dig}} = \frac{G_C R_{sense} C_{est}}{G_{rip}} \tag{4.8}$$

where G_C is a digital gain introduced to yield a fixed point representation for the value of the capacitance. The same expression should be used to obtain the initial digital estimate, $\hat{C}_{o,dig}$.

4.2.2 Gain Tuning and Testing

Gain Tuning

The digital gains were computed as the product of the load capacitance and other parameters, as discussed in Chapter 3. They are therefore factored as follows:

$$h_{dig} = h_{digc} \times \hat{C}_o \tag{4.9}$$

where $\hat{C}_o$ is the initial estimate for the load capacitance, and $\hat{C}_{o,dig}$ is its digital value. An obvious tuning procedure corrects the gains as follows:

1. Compute the digital estimate of the capacitance based on the measured load current and ripple amplitude, $\hat{C}_{dig}$.

2. Multiply the existing gains by $\hat{C}_{dig}$ and divide by $\hat{C}_{o,dig}$. Note that tuning is done every time an estimate is acquired, ie. only when the system is in steady state.

3. Update the capacitance value by setting $\hat{C}_{o,dig} = \hat{C}_{dig}$

Testing

A simple test was used to verify the operation of the estimator and gain tuner above. In order to avoid hardware modifications emulating capacitance variations, the controller was started with gains based on a capacitance value that was wrong by a factor of 3. A transient response with the adaptive controller was then monitored and compared to that with a controller using the same erroneous controller gains, but without adaptive control. The controller code was optimized for dealing with the case of the actual capacitance value less than the initial estimate (gain scaling for highest accuracy). Results of of this experiment are illustrated in Chapter 5, and indicate the sorts of improvements that may be obtained via adaptive control. Finer tuning of small errors and other situations are straightforward extensions of this case. System noise may need to be eliminated further, along with further code optimization.

Chapter 5

Experimental Results

As mentioned throughout, there are few digital implementations of power converter controllers outside of the area of machine drives. Cost and feasibility issues, as well as doubt about the usefulness of digital control in the domain of power supplies, have for the most part limited this area to analog control. The outcome sought here is thus a functional controller that demonstrates potentially low-cost implementations of a product that may have demands better met with digital control rather than simple analog control.

5.1 Experimental Setup

A layout of the hardware system was presented in Section 3.1.6. The construction of the system was carried out in different stages. The AC line was fed through a VARIAC (VARiable AC supply) to allow for adjustment of the input voltage. The boost converter components were soldered onto a copper-coated circuit proto-typing board and laid out horizontally. The input to the power stage was connected to the output of the VARIAC. The power stage output was connected to 3 lightbulbs in series, each 15 watts in strength. A resistor of approximately 3 $k\Omega$ in value was placed in parallel with the light bulbs, to be switched on when a load increase is needed. The resistor switch along with the VARIAC ON switch, a "fuse-blown" indicator, and voltmeters for the boost converter DC side and the controller power are housed in a panel on the front side of a platform which contains the power supply and loads. 24 Volt, ± 15 Volt, and 5 Volt power lines are available from a switching power supply adjacent to the plastic platform [1].

The isolation amplifiers, Unitrode chip and surrounding discretes, the multiplying DAC, and the output latch are soldered onto a prototyping board. The connections to the boost converter are made through the edge of the prototyping board, which plugs vertically into an edge-connector soldered onto the horizontal power supply board. The connections to the EV80C196KB are made by wirewrapped connections to a ribbon cable. The EV80C196KB is mounted onto a solid copper board on top of switching power supply. The power supply provides the ±12V and 5V levels required by the board. The board communicates with the PC host through an RS-232 cable and connector. Analog filters and the resolution mapping circuit are constructed on a breadboard mounted next to the EV80C196KB and connected to it via wirewrap connections. This setup is noisy but easy to use and modify.

To start up the system, the main power is turned on. The VARIAC is then switched on. This sends the DC side of the boost converter to the peak value of the AC side waveform. In a product, this stage would be produced by introducing a delay in starting up the controller, to allow the converter to reach this initial state. The control code is downloaded onto the evaluation board and its execution is started. A 3-second delay is inserted at the beginning to allow the user to turn on the power to the Unitrode chip. (This is necessary, as the default state of the Port 1 lines is high. If the Unitrode chip is powered up while these lines are high, the maximum current command will be requested.) A load transient is simulated by dropping the 3 kΩ resistor in parallel with the light bulbs. To power off the system, the Unitrode chip power is turned off first, then the VARIAC and then the main power.

5.2 System Simulation and Results

5.2.1 Simulation

Using the large-signal model and PI controller developed above, MATLAB simulations [2] are constructed for the closed-loop system. With the general parameters of the development system and techniques presented above, simulations of the digital system and the experimental setup are also defined. Detailed simulations of the system, including such features as the resistive load and audio-susceptibility effects were also

[1]The boost converter, load setup, housing structure, and front panel were constructed by Steven Leeb

[2]MATLAB is a trade mark of The Mathworks Inc.

carried out, but showed no significant differences in system response from simulations without the details. Simulations were used in all stages of the development. The digital controller was initially tested open-loop, by feeding it with a constant voltage and monitoring the controller command. The increasing command (in response to the constant 'error') with saturation was compared to a simulation of the open-loop controller and verified as the desired response. Closed-loop system comparisons with simulations were also carried out. More simulations were used to examine the effect of changes in the output capacitance on the transient behavior. Documented simulations based on the T_L-SDM are included in Appendix D. Simulations of the implemented system are also shown below.

Simulations were also carried out using the T_s-SDM. This provided more detailed examination of the output voltage waveform and ripple. The ripple behavior is essential in implementing adaptive control. Simulations shown in Chapter 4 were used to confirm the relationship between the ripple and the load used in estimating the capacitance. Documented simulations based on the T_s-SDM are also included in Appendix E.

Figure 5-1 shows a simulation of the startup response of the implemented system. A straightforward ramping of the reference voltage is implemented for soft-startup. More complicated schemes may be used, especially to deal with the effects of the nonlinearity introduced by the resolution mapping circuit. The system transitions from a nonlinear region of "minimum" voltage to a linear region around the steady state as it ramps up, and the effects of this transition have not been examined. Figure 5-2 shows a simulation of the load transient response in the working digital system after slowing it down by moving the poles closer to the unit circle (around .91) and improving the input resolution using the resolution mapping circuit discussed earlier. The effects of quantization are negligible in this design, which was used for the final implementation.

5.2.2 Digital Controller Without Adaptive Control

The experimental setup described above exhibited a high susceptibility to noise and interference between signals. This was the result of the temporary nature of the structure (part of which was built on a breadboard), the parasitic inductances on the evaluation board, and the large number of active analog components in the final system. Isolation amplifiers, the Unitrode chip, and many operational amplifiers were used. Some of the operational amplifiers were used as buffers around the isolation amplifiers, which could only source up to 2 mA of current. Others were used in providing the appropriate inputs to the Unitrode chip. Although passive filters were

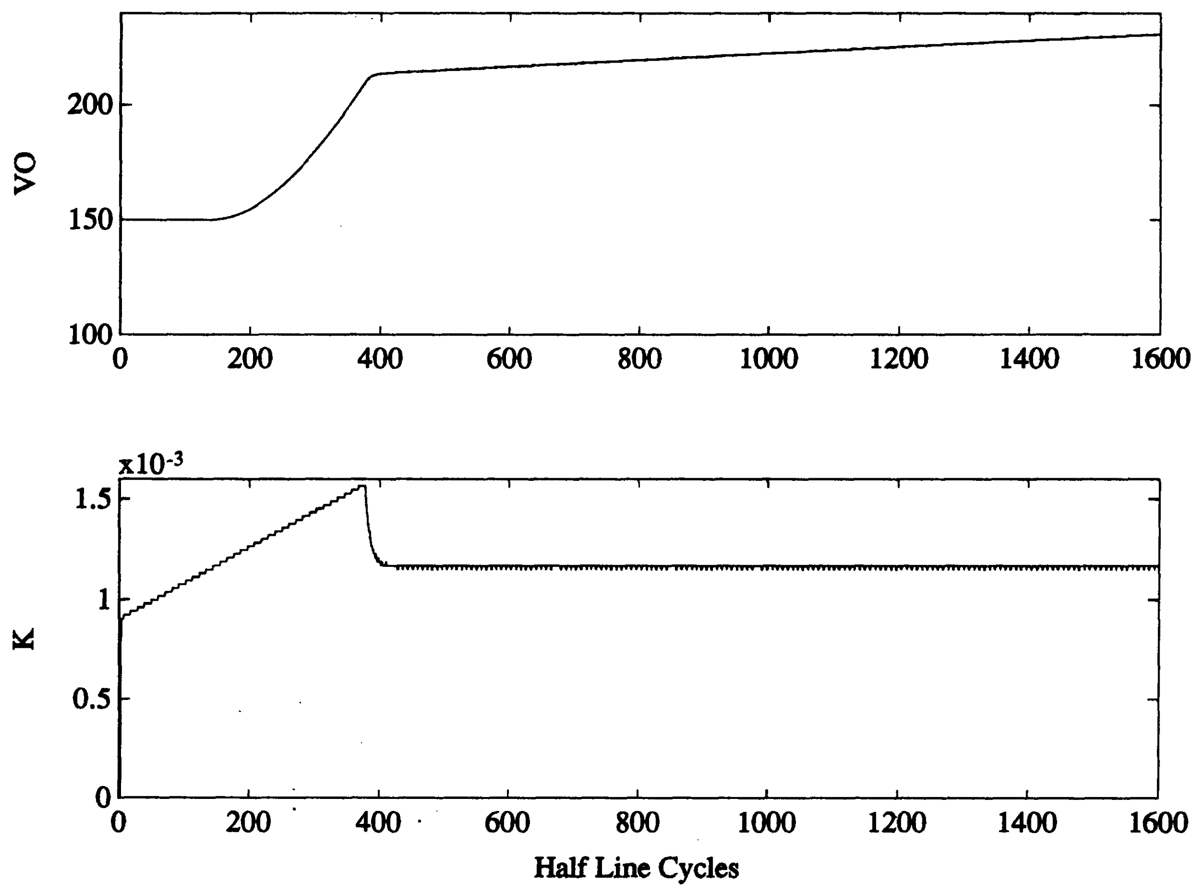


Figure 5-1: Simulation of Startup in Implemented System.

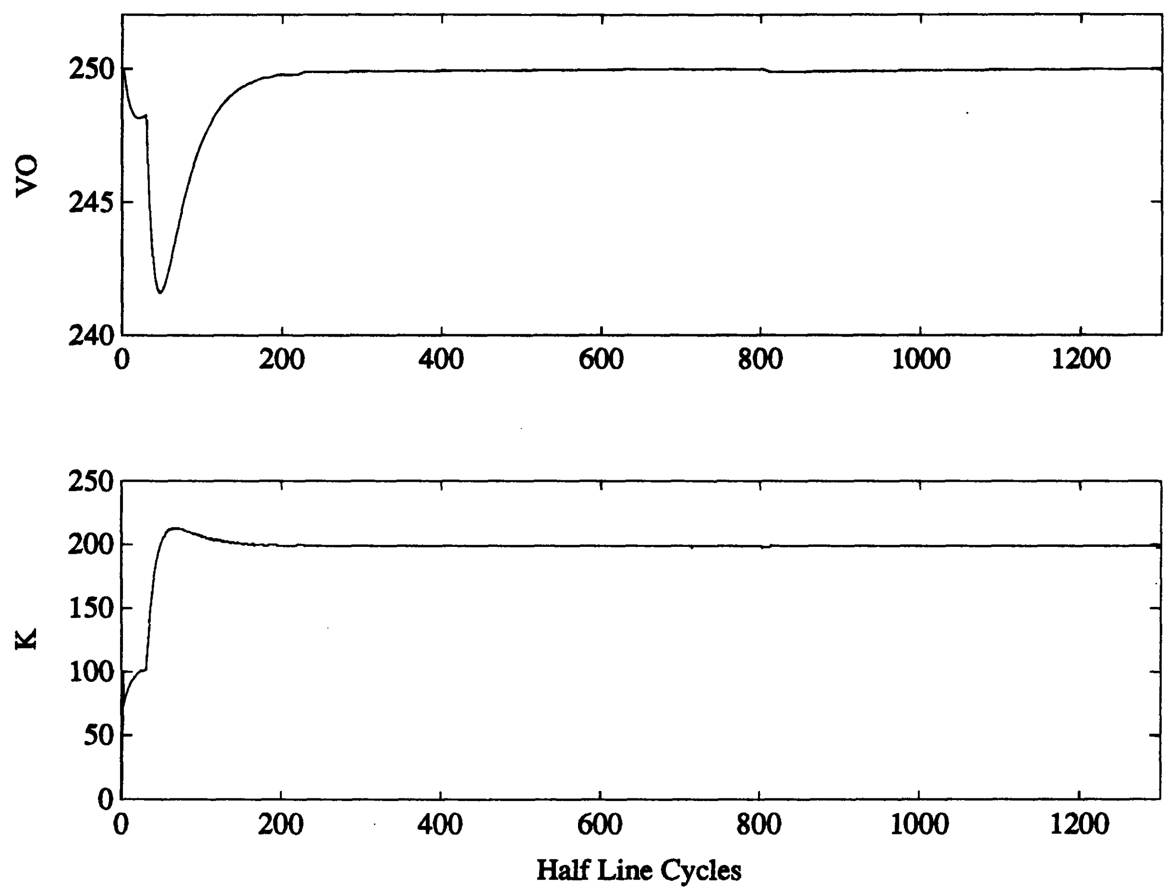


Figure 5-2: Simulation of Load Transient Response in Implemented Digital System.

used, they were constructed as cascades of second order filters, connected to each other through an operational amplifier configured as a follower. In a final product, most of these active components may be eliminated.

However, in our experiment, the high level of noise proved very limiting. Filters with low bandwidth had to be built to remove the noise from the output voltage at the A/D channel. This was especially important as the quantization of the system served to accentuate the error due to noise. The system bandwidth had to be made lower to achieve acceptable steady state behavior (smooth input current and output voltage). The bandwidth of the controller can be made higher in a less noisy system.

A digital filter was implemented during the testing stages as well in an attempt to reduce the effects of the noise. Although it was not used in the final implementation to leave as much room as possible in the processing cycle for adaptive control, it demonstrated the ability of the digital microprocessor to replace a variety of analog components.

Experimental results for the startup, load transient response, and steady state operation of the physical system are shown below. The poles of the system have been placed around .9, as in the simulation above. The results agree with the simulations and demonstrate satisfactory performance for the UPF converter. Quantization error can be seen in the current waveform, but not so much in the voltage waveform, due to scale. The sinusoidal current yields a near unity power factor. Note that the startup behavior of the UPF is not uniform in the slope of the output voltage or the input current. Nonlinearities are introduced as the sampled output voltage goes from the region in the resolution mapping circuit where it is flat to where it is ascending. A different A/D channel, sampling the output voltage directly without passing it through the resolution mapping circuit, may be used for startup if needed. More involved soft-startup mechanisms, where the gains as well as the reference change for instance, may also be used.

5.2.3 Adaptive Control

The design and implementation of an example of adaptive control in the context of this probelm were also explored. Specifically, as discussed in Chapter 4, the load capacitance was estimated from measurements of the output voltage ripple and load power. This capacitance was used to tune the PI controller gains. One simple example of the initial capacitance estimate being much larger than the actual value was tested. The controller managed to estimate the capacitance correctly and to tune the

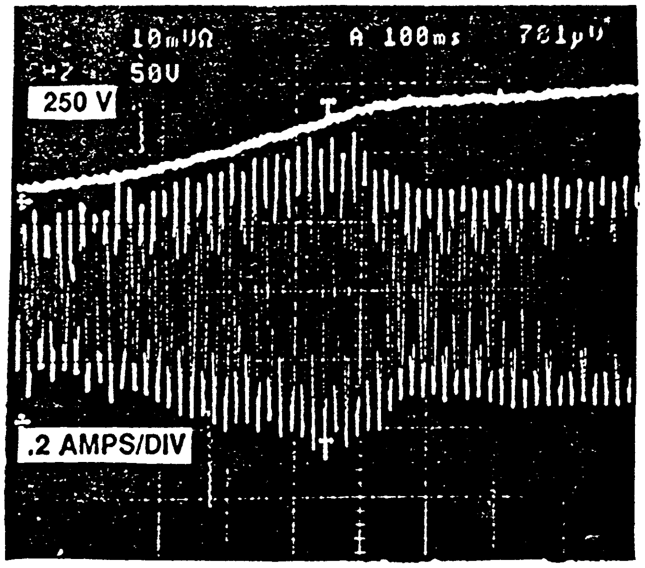


Figure 5-3: Startup in Implemented System

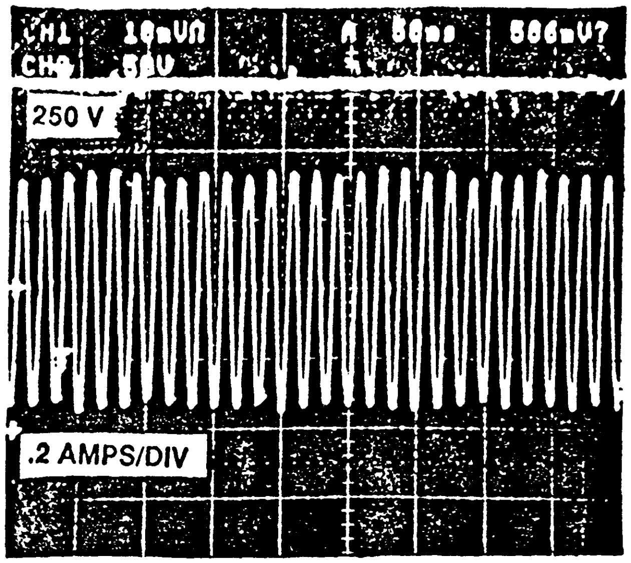


Figure 5-4: Steady-State in Implemented System

gains to their desired value for the given capacitance. These values were monitored through the EV80C196KB monitor software, and were verified to be correctly tuned. The transient response of the controller with the same erroneous initial PI gains, but without adaptive control, is compared in Figures 5-6 and 5-7 to that of the system with the estimation and gain tuning implemented.

A smaller capacitor was used for implementing the adaptive controller. With the original load capacitor, the ripple on the output voltage was too small to be accurately measured. The smaller capacitor has a shorter holdup time, and the response to a transient is worse than before (the voltage dips lower when the load is stepped up for instance).

5.2.4 Overall Results

The experimental results match the simulations well, verifying that the large-signal model is fairly accurate. Moreover, no load regulation and no steady state error are

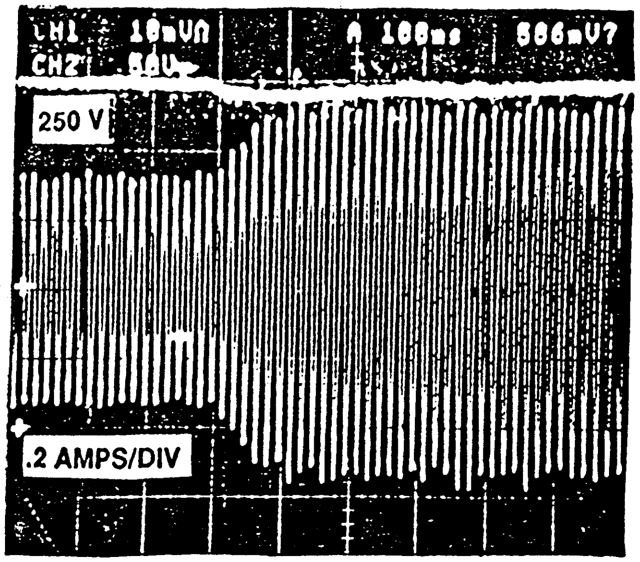


Figure 5-5: Response of Implemented System to Load Transient

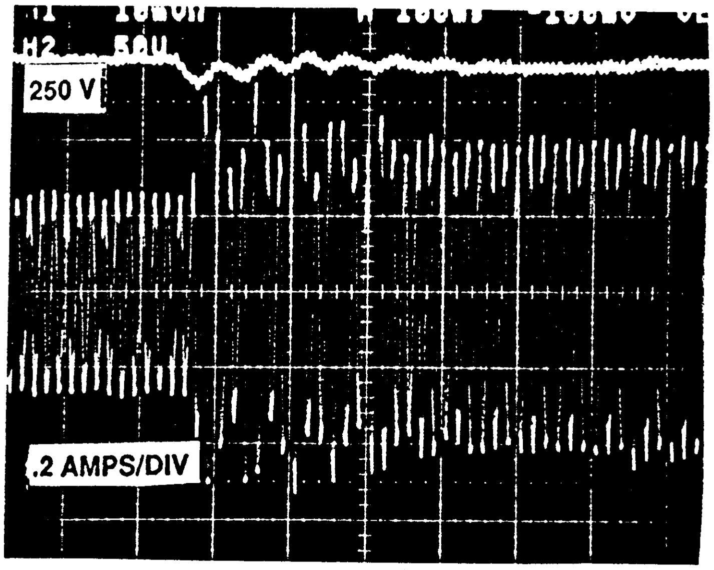


Figure 5-6: Response of System with Incorrect Load Capacitor Value without Adaptive Control

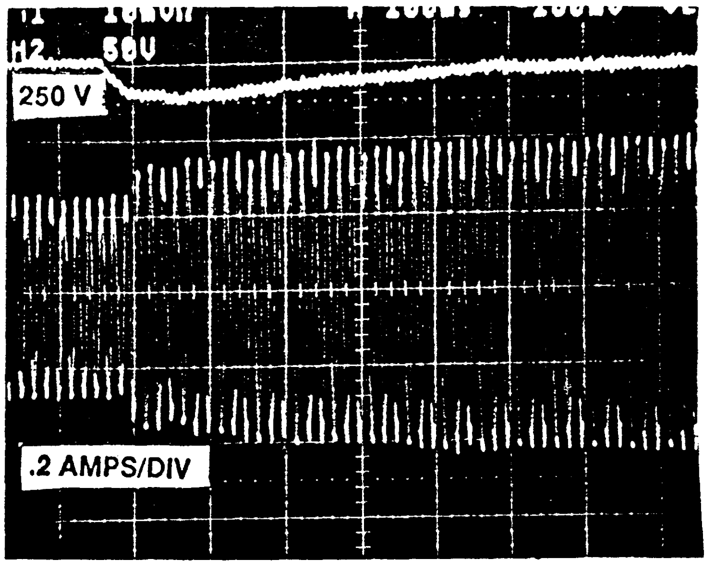


Figure 5-7: Response of System with Incorrect Load Capacitor Value with Adaptive Control

evident. A Unitrode application circuit built around the UC3854 to implement total control for the UPF exhibits load regulation. Although an analog implementation of a PI controller should in theory have zero steady-state error, in reality this is not the case. True PI control, with infinite gain at DC, is not possible due to parasitic resistances. The digital controller suffers from no such parasitics.

The basic closed-loop controller was successfully implemented, with a few additional features that were included with relative ease, and that illustrates the potential for rather complex control. Adaptive control was also successfully implemented. The microprocessor used is relatively inexpensive and quite powerful, leaving room to implement more complex algorithms or to decrease the cost of the implementation. With the decreasing cost of semiconductors and their increasing functionality, the possibilities are even greater. Adaptive control allows the system to achieve higher level performance and self-tuning. As power requirements become more stringent, this higher performance becomes more and more important. Digitally controlled power converters are better suited to meet these needs.

Chapter 6

More Compact Implementations

Implementing the digital controller using the EV80C196KB provided a good test-bed for the system and facilitated a proof-of-concept experiment. The flexibility of the evaluation board and its user-friendly interface with a host PC made it possible to test various control schemes and to easily modify and optimize the controller. A product design cannot make use of the evaluation board, however. Cost and size considerations require that only the essential interface circuitry to the microprocessor is used and the evaluation board removed. Since one of the main arguments commonly presented against digital control in power supplies is the cost of implementation, more compact and less expensive implementations are explored below. Only a brief description of these implementations is presented, as they were carried out as separate projects and documented in more detail separately.

6.1 Microprocessor Implementation Without the Evaluation Board

For implementing the digital controller, interface circuitry to the A/D converter and output port is required. Memory is necessary for storing the controller code and needs to be interfaced to the microprocessor. In the implementation using the EV80C196KB, off-chip memory and memory and I/O interfacing are provided by the evaluation board. However, these components represent only a small fraction of the total board size and functionality, and hence of the cost. The result is an expensive implementation.

In order to achieve a cost-effective implementation, the EV80C196KB has to be replaced with only the essential circuitry for our application (namely the memory and interfacing mentioned above). Additional hardware, such as the multiplying DAC and the LS374 used to latch the outputs from Port1 and into the multiplying DAC, is also necessary. This hardware, however, is also built separately for the evaluation board implementation, and has already been described. The remaining interfaces (those that are on the EV80C196KB on the evaluation board implementation) are discussed below [1].

6.1.1 Memory Interfacing

As on the evaluation board, Port3 and Port4 are used for memory interfacing. The lines on these ports are used for both Data and Address communication. In this implementation, two EPROM's are used for external memory. An external memory access cycle occurs over two CLKOUT cycles. On the first clock cycle, the microcontroller presents the address to be accessed. In order to preserve this information for the second cycle of retrieving the data from the EPROMs, this address must be latched. Two LS373 transparent latches are used to hold the address values for an extra cycle. These latches are controlled by the ALE signal from the 80C196KB. The ALE signal indicates whether the information on the lines is Address or Data. A block diagram of this design is shown in Figure 6-1. The data lines are unidirectional in this case since only memory reads are necessary. Detailed schematics and timing diagrams are shown in Appendix G.

The 80C196KB has 256 bytes of on-chip RAM. The controller code does not require more than this amount in accessible memory, and therefore no additional SRAM is required. Had that not been the case, an SRAM chip would need to be interfaced to the microcontroller as well. That may be done through the same I/O port lines, but with data flowing both ways. Another point to be made is that not all of the memory used in this implementation is necessary. In a final version of the controller in which the code has been optimized for space, some of the lines on ports 3 and 4 may be free for improving the output resolution or perhaps controlling another power supply.

6.1.2 A/D Interfacing

The A/D interfacing circuitry is also shown in Figure 6-1. This circuitry is suggested by and discussed in detail in [13]. At the input to every A/D converter channel is sense capacitor that gets charged by the input to the channel during a 1 μs conversion window. There is also about 3 μA of leakage current. For inputs with high output impedance, the converter accuracy may degrade due to the leakage current causing

[1]Safroadu Yaboah-Amankwah is responsible for this work as part of an undergraduate lab supervised by the author. The schematics in Appendix G are taken from his report on the project.

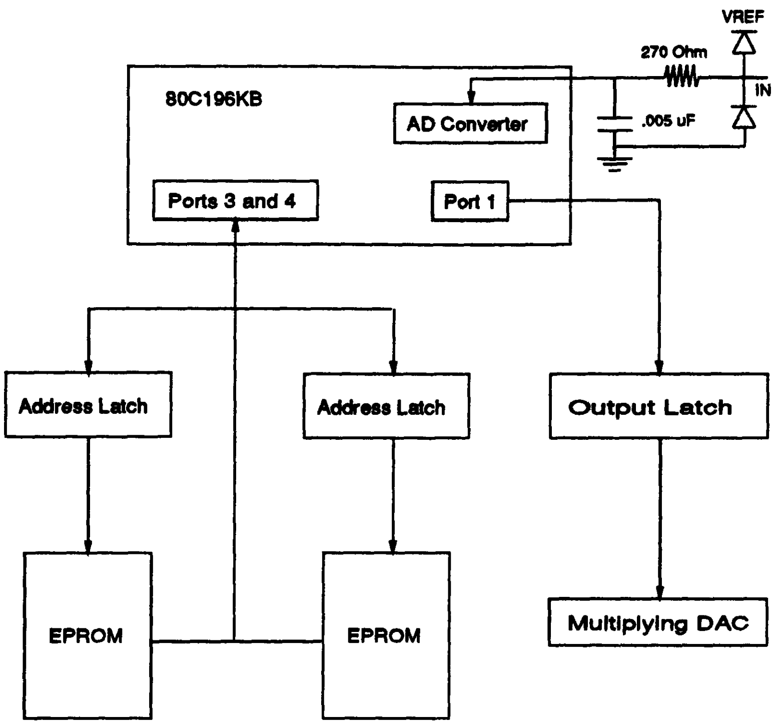


Figure 6-1: Block diagram of compact microprocessor implementation.

a significant drop in the signal voltage. Moreover, the time constant of the RC connection may be long enough to prevent the sense capacitor from charging fully during the 1 μs sampling window. Adding an external capacitor larger than the sense capacitor compensates for this and ensures the charging of the 2 pF sense capacitor. Combining it with a small resistor as shown in the figure also reduces sensitivity to noise with the resulting low pass filter. The resistor also limits the input current. The diodes in the interface circuitry are used to protect against overvoltage conditions on the inputs.

6.1.3 Experimental Results and Conclusions

The system described above was built and tested open-loop. The controller code as well as other test code was burnt into the EPROMs and the microcontroller was reset. Inputs were generated by power supplies and fed into the analog channels. The digital outputs were monitored and verified to match their expected values. No closed-loop verification was carried out as it was felt that these tests were enough. The system implemented was shown to perform the functions of the EV80C196KB

Table 6.1: Estimated Cost for Components

Components	Estimated Cost
Board	\$ 12.50
Manufacturing	\$ 9.25
Controller	\$ 16.70
EPROMs and Latches	\$ 2.50
Discretes	\$ 2.00
Total	\$ 42.95

that are necessary for the UPF digital controller.

The table below shows the approximate cost of the different components and the total cost as based on a PC-board implementation of the circuit. The provided figures are conservative. The cost is lower for bulk quantities of microprocessors, and the EPROMs may be removed completely by hard-coding the control algorithm into the microprocessor chip. Also, a cheaper microcontroller may be used, as mentioned earlier. The area of this controller board, especially if the EPROMs are removed, would be basically that of the microprocessor, the latch, the multiplying DAC, and a few discrete components. This is negligible compared to the size of the power supply components.

The remaining cost and area are those of the analog interfacing circuitry and the power supply. Note that a simple current-loop will replace the Unitrode UC3854 and the isolation amplifiers will be removed in a product version of this system. Further studies are needed to obtain an accurate estimate for the likely cost of this system, and to compare it to existing analog controlled systems.

6.2 VLSI Implementation

To take the compaction of the digital controller one step further, a VLSI implementation of the control algorithm was explored. This work was carried out as a class project, so the limited time and chip area constrained the implementation. The project was meant to demonstrate the feasibility of a VLSI implementation rather than to produce an actual working controller. The area available did not permit the implementation of wide enough data paths to produce high enough resolution for practical use. Although a working integrated circuit has not been fabricated, tremendous insight into the problem has been gained as a result of this experiment. Specifically, the implemented controller uses multiplication and addition, important components of many controllers (and of all linear controllers). The components of this integrated circuit should be therefore very similar to those of the generic linear controller.

The following sections discuss the overall solution to the design problem and introduce the functional blocks that implement the different operations. Although much of the flexibility of the microprocessor is lost, some is retained and more can be incorporated with a more complex design. The advantages are reductions in cost and area, and an increase in speed.

6.2.1 Overall System Layout

The controller in Figure 2-2 comprises addition and multiplication operations. One implementation would lay out a multiplier and an adder circuit for each operation. Although this would have the fastest propagation time from input to output, the area consumed by such an implementation is excessively large. A better circuit would pipe all the multiply operations through one multiplier and the additions through one adder. The control for such a circuit is more difficult, but it greatly reduces the area requirements.

In addition to the addition and multiplication modules, multiplexors are needed to select the numbers to be added or multiplied, registers to hold the intermediate values which are to be used later in the computation, and a PLA (Programable Logic Array) to implement a finite state machine controlling the data flow through the circuit. Additional circuitry is needed to deal with the negative numbers that are encountered in the computation. Figure 6-2 shows a block diagram of the different modules and the data flow.

6.2.2 Control and Data Path

Two-phase non-overlapping clocking is used to pipeline the circuit. The clock can be as fast as the worst delay between two registers. One addition or multiplication is carried out every clock cycle. The delay through the circuit is therefore the number of operations multiplied by the clock period. The shift registers are loaded after a reset is detected by the gain values. This feature was included to allow for the modification of the controller parameters. The final multiply operation is incorporated into the gains, except for the capacitance value. Although the capacitance estimator is not implemented on the chip, the capacitance is provided as an immediate input. This allows an estimator running off-chip to modify the controller's estimate of the capacitance in real time.

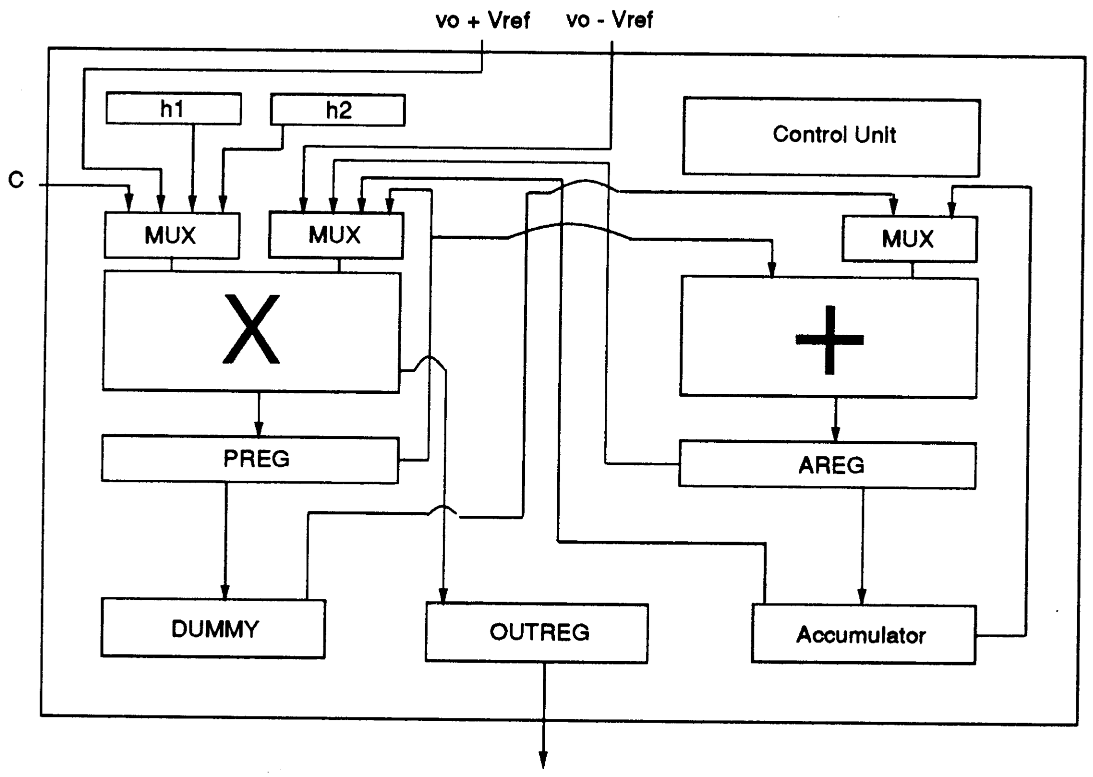


Figure 6-2: Block diagram of VLSI implementation.

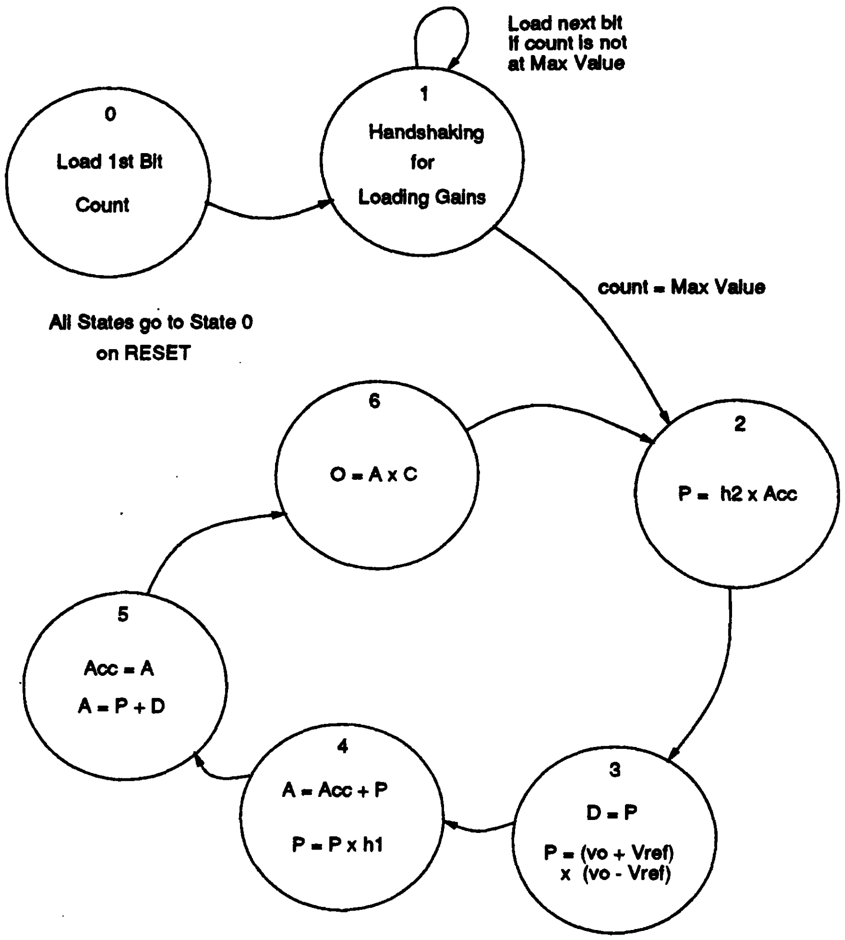


Figure 6-3: Finite state machine for data path control.

To best describe the operation of the circuit and the data flow, a diagram of the finite state machine (FSM) is shown in Figure 6-3 and a description of every state is presented below. Every state in the state machine represents an operation during a clock cycle. The output is sampled by an external clock running at 120 Hz.

The states of the FSM are described below:

0 This is the RESET state. The RESET pin is an input to the chip, and all states transition to this state when this line goes high. A counter and the Accumulator are set to zero when the RESET pin goes high as well. Once in the RESET state, the first bit of the numbers representing the gains h_1 and h_2 is loaded and the counter is incremented. State 1 is entered next.

1 In State 1, the gains are loaded bit-serially. This is accomplished by handshaking that is not discussed here, and the counter keeps track of when this

operation is done. Once this operation is terminated, State 2 is entered.

2 In this state, the Accumulator value is multiplied by the gain h_2 and loaded into the PREG. State 3 is entered automatically on the next clock cycle.

3 Load the value in PREG into register DUMMY, and then multiply the input values of $v_o + V_{ref}$ and $v_o - V_{ref}$ (In order to simplify the computation, it is assumed that the input comes from analog adders and subtractors that provide these quantities, resulting in one required multiplication for the digital system). Store the value in PREG. Transition to State 4.

4 Multiply PREG by h_1 and store the value in PREG. On the same cycle, add the values in the accumulator and the value from PREG (which before the end of this cycle is still the value of the error). Store the addition result in AREG. Transition to State 5.

5 Load the Accumulator register from the value in the AREG. Add the values in DUMMY and PREG. Result is stored in AREG. Go to State 6.

6 Multiply the the values in AREG and the input C (capacitance value). Store the result in OUTREG. Transition to State 2.

A simplified timing diagram is shown in Figure 6-4. The multiplexor control signals are changed every cycle to specify which inputs are to be used. They are outputs of the state machine but are not shown in the timing diagram for simplicity. The serial loading of the controller gains after RESET is also not shown in detail.

Many details of the design have been left out. A rough estimate of the delay of the whole control computation for a maximum register size of 10 bits yields a figure on the order of microseconds or 10's of microseconds. This figure increases with the sizes of the registers and adder and multiplier blocks. However, for the most part, the whole 8.33 *ms* is left for the off-line capacitance estimation or for any additional features in the control. With enough area, any width of data path may be implemented. Details of the project are not included here, but a sample transistor level schematic of one cell used in the design, and the configuration in which the cells are used to implement a functional block, are shown in Appendix H. In general, the unit cells operating on single-bit inputs are used in arrays to implement multi-bit operations. The same design may be easily modified to accommodate wider words. This implementation would also prove much cheaper and smaller in size.

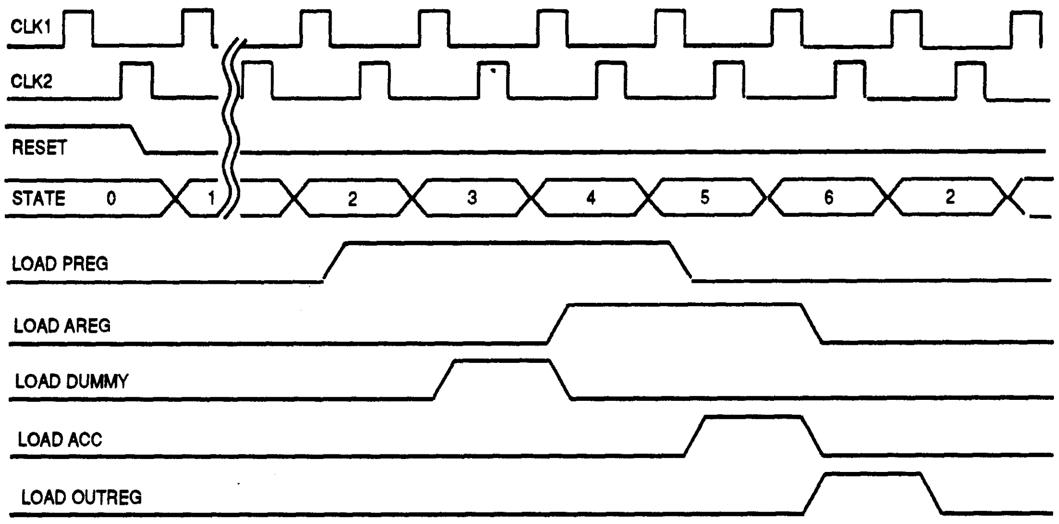


Figure 6-4: Timing of Control Circuitry.

Chapter 7

Conclusions and Future Work

The goal of this research was to explore the issues surrounding a digital implementation of a controller for a UPF converter. The design and implementation of such a controller was carried out, with feasibility, performance, cost, and size in mind. As operating requirements on power converters become more stringent, power supplies with high power factors and complex control for high levels of performance become more valuable. Digitally-controlled UPF's appear attractive from several points of view. However, cost and lack of experience with such systems are significant hurdles in the way of their development. Nevertheless, with the falling prices of semiconductor products and the increasing processing power of digital microcontrollers, it appears that it may be time to consider this alternative. This thesis explored this possibility.

In Chapter 1, high power factor converters were reviewed along with general background in digital control. The research was motivated and its goals outlined.

Modeling of UPF converters was discussed in Chapter 2. A large-signal model for the boost converter topology of the UPF was developed in detail. Averaging over half the line period provided a first-order LTI model from which a sampled data model was derived. Continuous-time and discrete-time models at the time scales of the switching frequency were similarly obtained. The "slow" sampled data model was used to develop a discrete-time PI controller for the UPF. More complex nonlinear control was also designed for the system. The availability of a microprocessor allowed much freedom in the control design.

In Chapter 3 the implementation of the controller was presented. The implementation details as well as the difficulties encountered were discussed. This controller was taken one step further in Chapter 4, with the design and implementation of adaptive control. This was a critical step since it clearly demonstrated the advantage of digital control over analog control.

Experimental results were reviewed in Chapter 5. Overall, the results were in favor of digital control of such a system. Although some problems still need to be ironed out, in general the digital controller exhibited more flexibility and better performance than the analog counterparts. With more research and product development, a commercial digitally controlled UPF seems very feasible, even in cost.

As cost is an important parameter in this experiment, cheaper implementations of the controller were explored in Chapter 6. A microprocessor implementation without the evaluation board and a VLSI implementation were described. Again the results were promising. The cost of a digital controller should eventually not be much higher than that of an analog controller.

Much remains to be done, however, towards producing a commercially feasible digitally controlled UPF. The implemented prototype is built in a temporary and noisy experimental setup. A final product would need to be constructed as a permanent, less noisy structure, perhaps as a PCB. Reductions in cost are also required. Toward that end the compact implementations discussed in Chapter 6 should prove to be helpful. An option not explored in this thesis is the use of a less powerful and less expensive microprocessor.

Future work should also attempt to analyze more deeply the phenomena encountered here that will be part of every digital controller implementation. Specifically, mathematical models for quantization error need to be developed and analyzed. Input/output resolution enhancement methods should also be explored further. For example, one idea is to design a controller that divides the command signal into a constant nominal component based on knowledge of the load and a variable component to be set by the feedback. By doing so, the same number of bits used to specify the full command earlier may be used for a smaller portion of it, hence increasing the output resolution of the digital controller. Other controllers, such as those based on different models, should be explored as well. The flexible experimental setup used here will allow the examination of different control algorithms in a physical system and not only in simulation.

The results of this thesis, the questions raised and investigated, and the experimental setup that was constructed indicate that digital control of power supplies is a viable choice. The thesis provides a knowledge base and a test-bed that should be helpful in carrying out this further work.

Appendix A

Derivation of Closed-Loop Transfer Function in the z-Domain

The sampled-data-model for the open-loop system presented in Chapter 2 is shown again below in Equation A.1:

$$x[n+1] = x[n] + \frac{T_L V^2}{C} k[n] - \frac{2T_L}{C} P \tag{A.1}$$

Taking the z-transform of this discrete-time description of the system yields:

$$zX(z) = X(z) + \frac{T_L}{C}(V^2 K(z) - 2P(z)) \tag{A.2}$$

$$X(z) = \frac{1}{z-1}(T_L/C(V^2 K(z) - 2P(z))) \tag{A.3}$$

This open-loop system is shown in Figure 2-1.

The discrete-time PI controller shown in Figure 2-2 is summarized by the following transfer function:

$$K(z) = \frac{C}{T_L V^2}(h_1 + h_2(\frac{1}{z-1})) \times (X(z) - X_{ref}(z))) \tag{A.4}$$

Closing the loop with the PI controller yields the system in Figure A-1.

The resulting transfer functions between the output $X(z)$ and the two inputs, reference $X_{ref}(z)$ and disturbance $P(z)$, are:

$$\frac{X(z)}{X_{ref}(z)} = \frac{-(h_1 z + h_2 - h_1)}{z^2 - (2+h_1)z + 1 + h_1 - h_2} \tag{A.5}$$

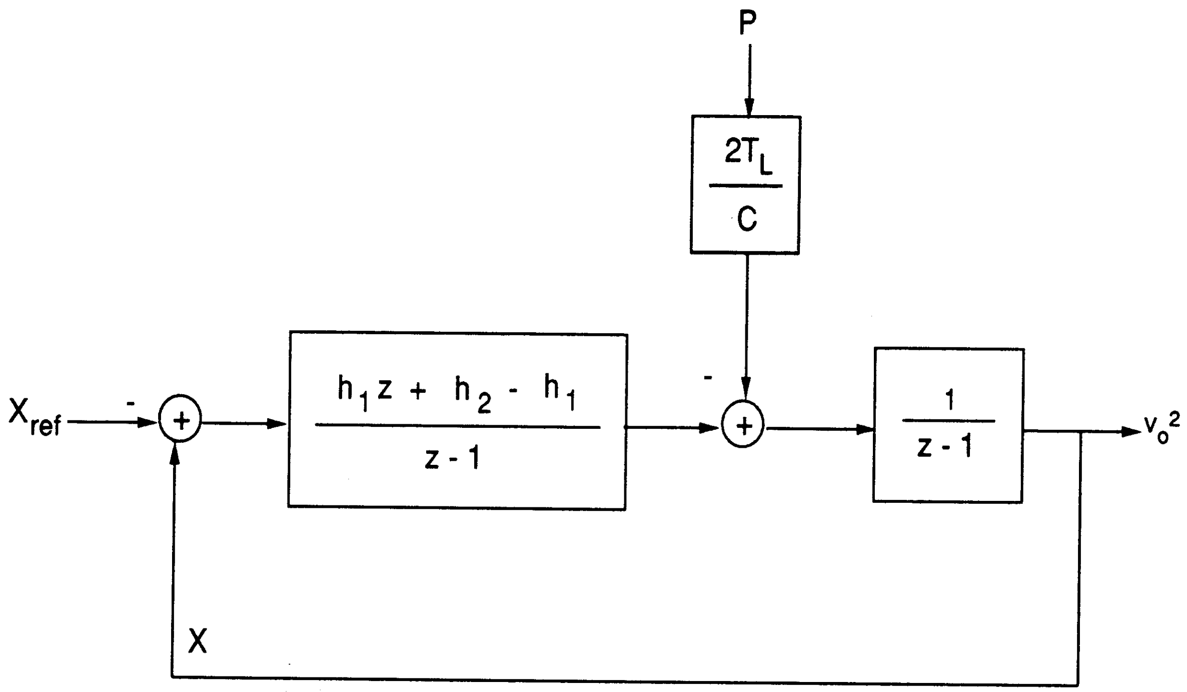


Figure A-1: The Closed-Loop System.

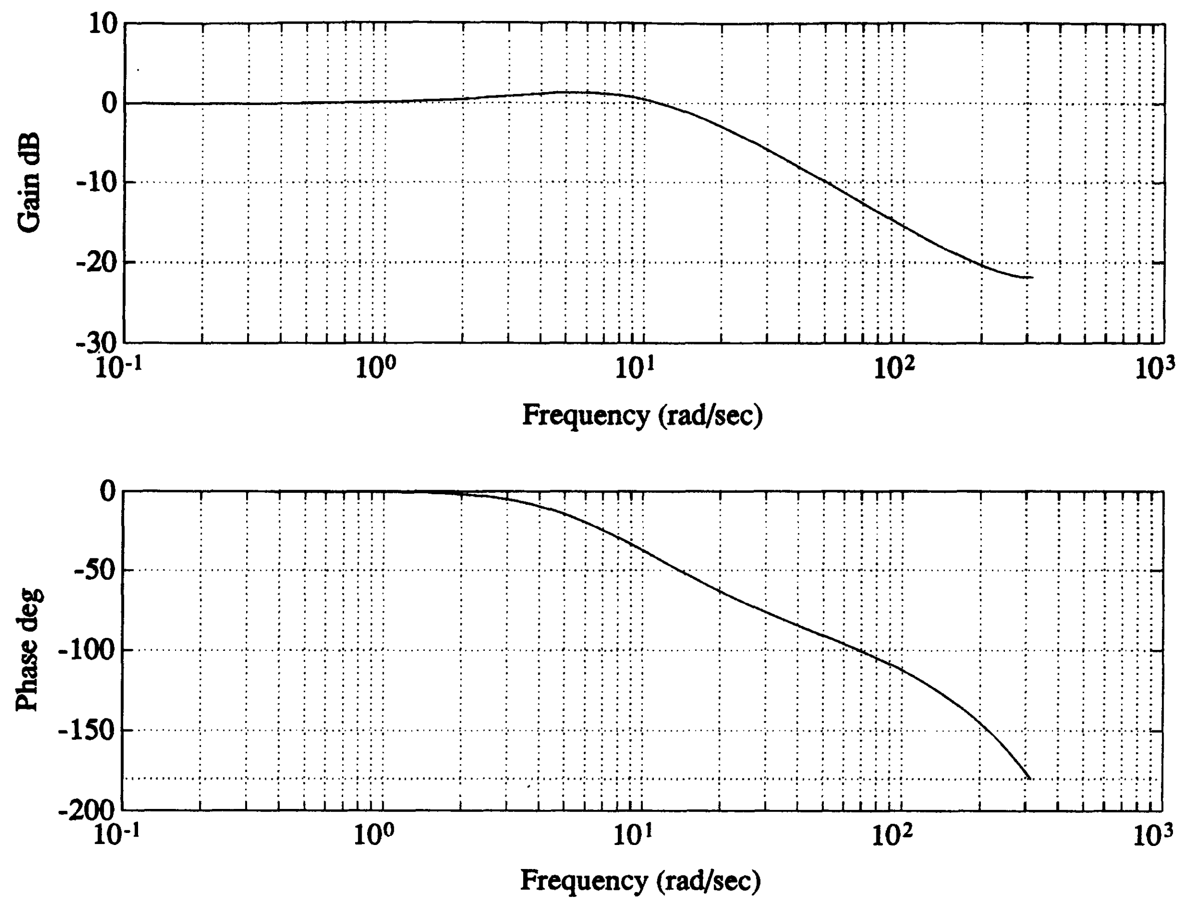


Figure A-2: Bode Plot for Transfer Function Between Reference Voltage and Output Voltage.

$$\frac{X(z)}{P(z)} = \frac{-(z-1)2T_L/C}{z^2-(2+h_1)z+1+h_1-h_2} \tag{A.6}$$

The bandwidth of the closed-loop system is an important parameter in the design of the anti-aliasing filter, as discussed in Appendix B. The figures below demonstrate the system transfer functions, with the system gains used in the final implementation, which correspond to:

$$h_1 = .1527 \qquad h_2 = .0056 \tag{A.7}$$

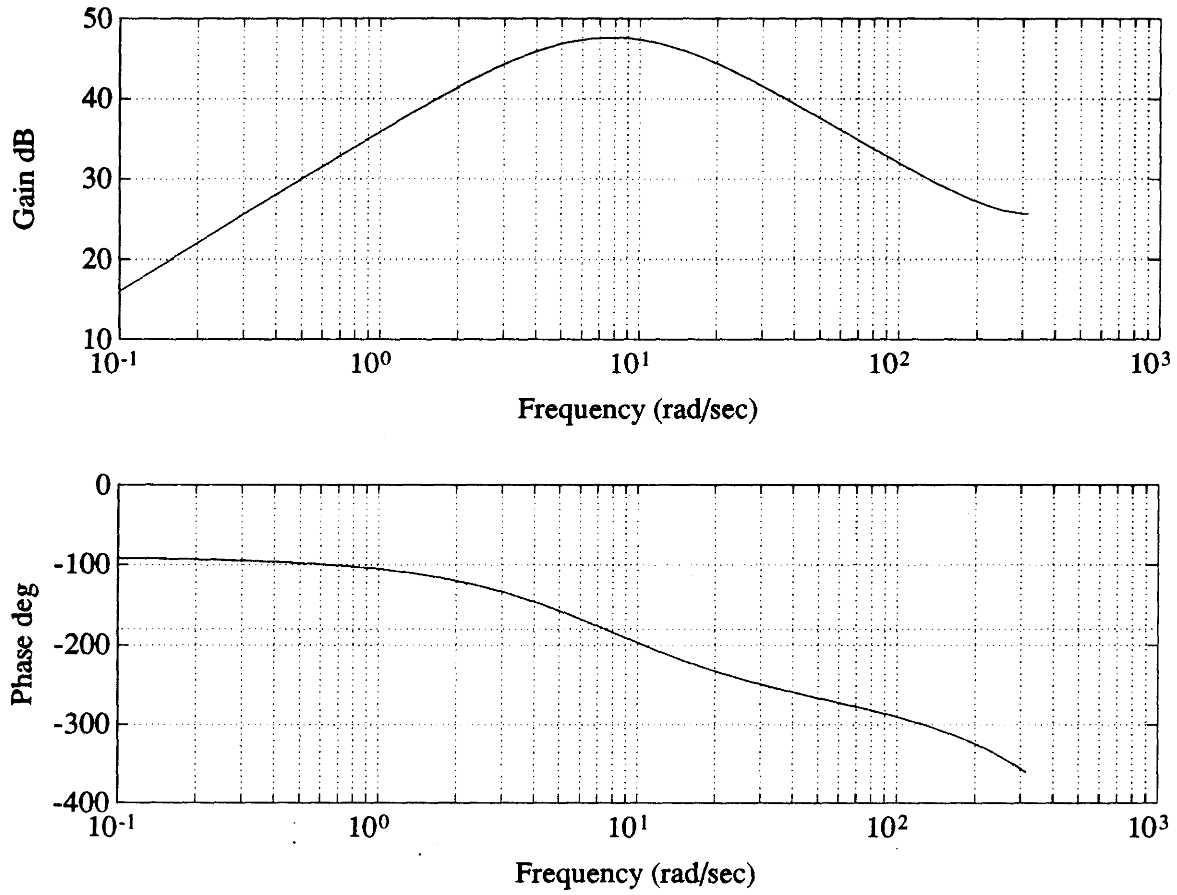


Figure A-3: Bode Plot for Transfer Function Between Reference Voltage and Output Voltage.

Appendix B

Comparison of Large- and Small-Signal Models

Simulations were carried out to compare controllers designed on the basis of large-signal and small-signal models. It was found that in the simple first-order system at hand, the large-signal model does not have a great advantage over the linearized one. However, with sufficiently large disturbances, some differences are witnessed. In a more complex system or one that has tighter specifications, these differences may be crucial. The two controllers were used in simulation to control a large-signal model of the UPF, as that is the more accurate of the two models.

The controller based on the large-signal model was designed in Chapter 2. The time-domain representation of the linearized model in Equation 1.7 is:

$$\frac{d\tilde{v}_o}{dt} = \frac{1}{2V_dC}(V^2\tilde{k} - 2\tilde{P}) \tag{B.1}$$

Following the same steps used in Chapter 2 to design the PI controller based on the large-signal model, a PI controller is derived using this model of the converter. As it turns out, given a controller bsed on the large signal model with gains G_I and G_P and poles z_1 and z_2, the corresponding controller based on the linearized model with the same poles will have the same gains but a different error term. In particular, whereas in the first case the error is:

$$V_{err} = v_o^2 - V_d^2 = 2V_d\tilde{v}_o + \tilde{v}_o^2 \tag{B.2}$$

in the second case it is:

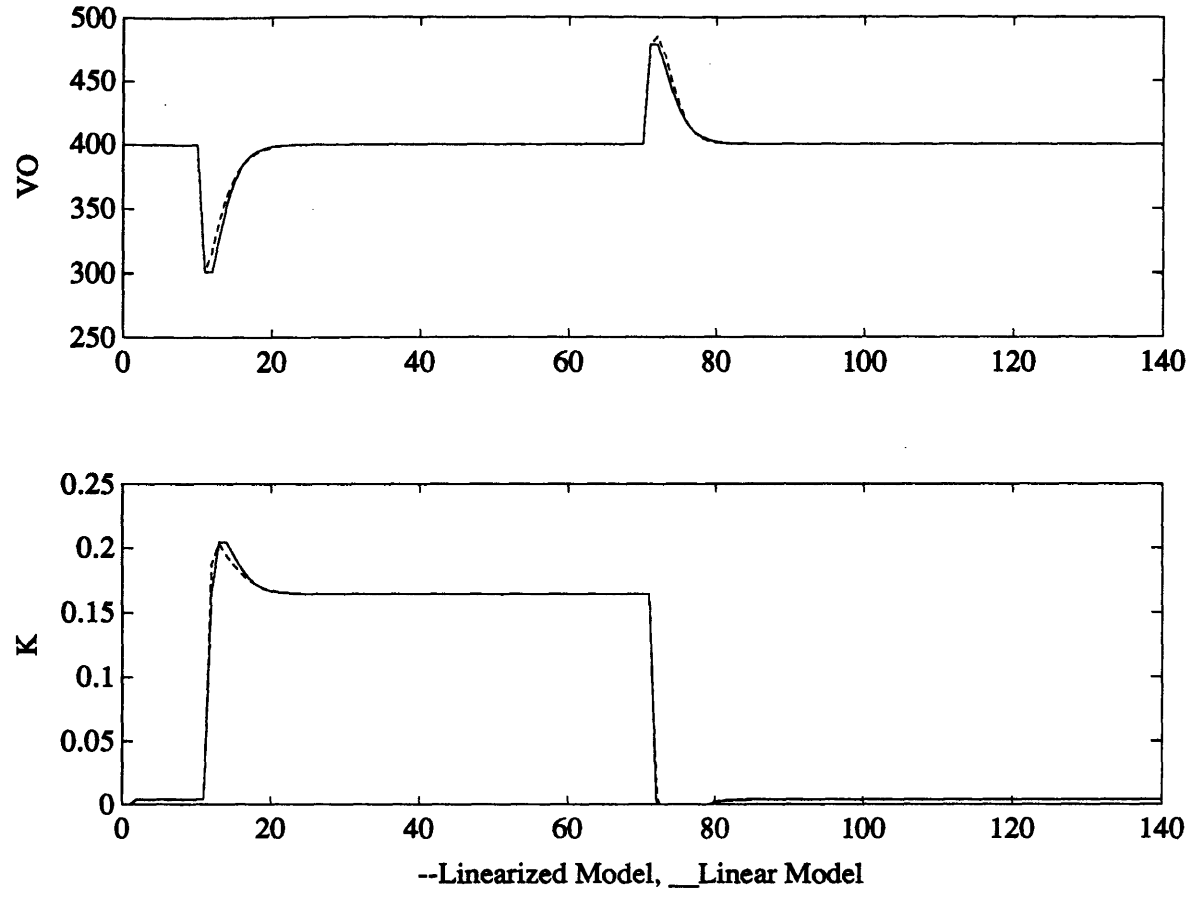


Figure B-1: Response of large-signal and small-signal controllers to load transients from 50 Watts to 2000 Watts and back. With the large capacitor used, there is hardly any difference.

$$V_{err} = 2V_d\tilde{v}_o \tag{B.3}$$

The difference in the reaction of the two controllers to disturbances is proportional to the square of the deviation from the nominal, which for small deviations and large reference voltages, is negligible compared to the other error term, the product of the reference voltage and the voltage deviation. In the UPF system, the reference voltage is large and the load transients are not significant enough to produce large error signals. In some of the simulations that follow, the load capacitor is changed or a large voltage transient is forced at the output in order to witness the differences between the two models. The simulations also verify the relationship stated above between the two error expressions. The linearized model controller reacts more strongly than the linear one for negative deviations from the nominal, and more weakly for positive deviations.

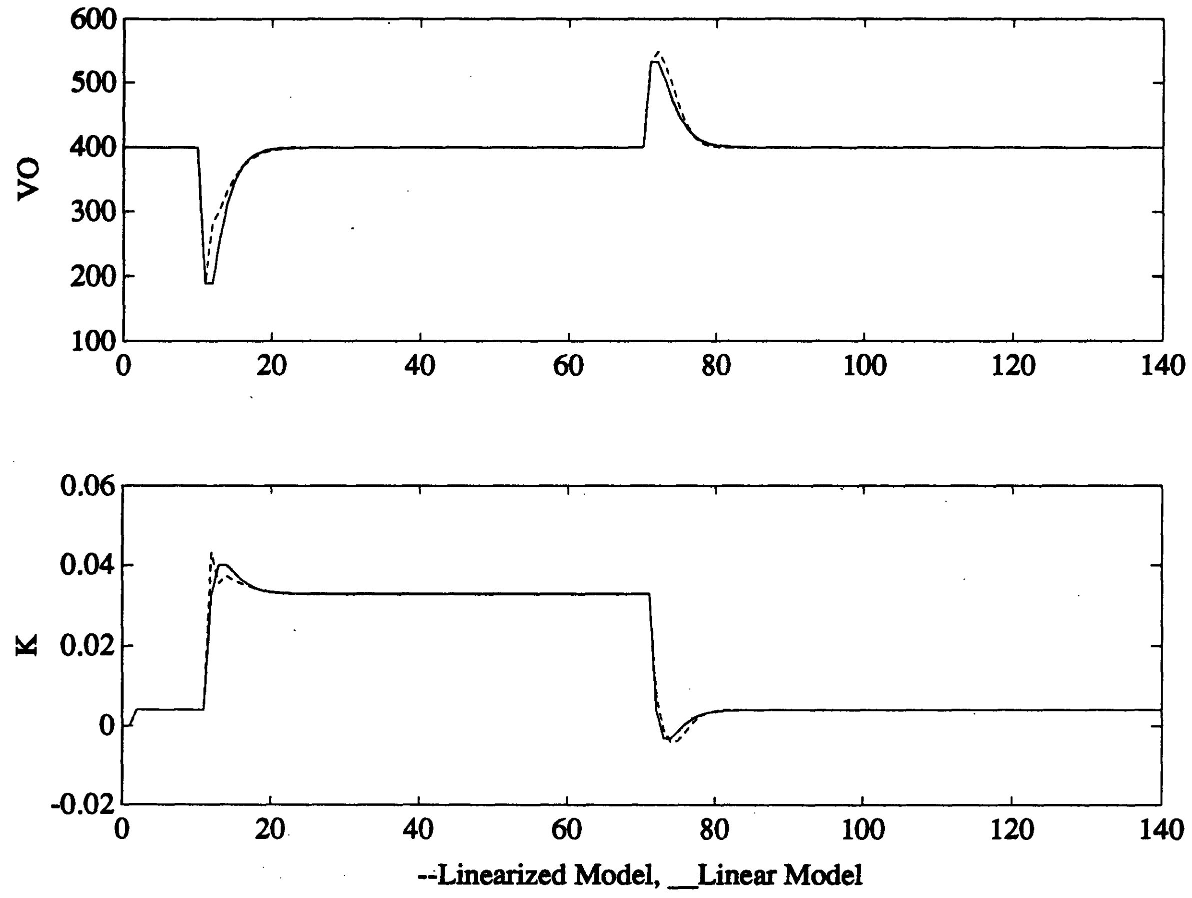


Figure B-2: Response of large-signal and small-signal controllers to load transients from 50 Watts to 400 Watts and back. The capacitor used is a tenth of the size of the capacitor used earlier.

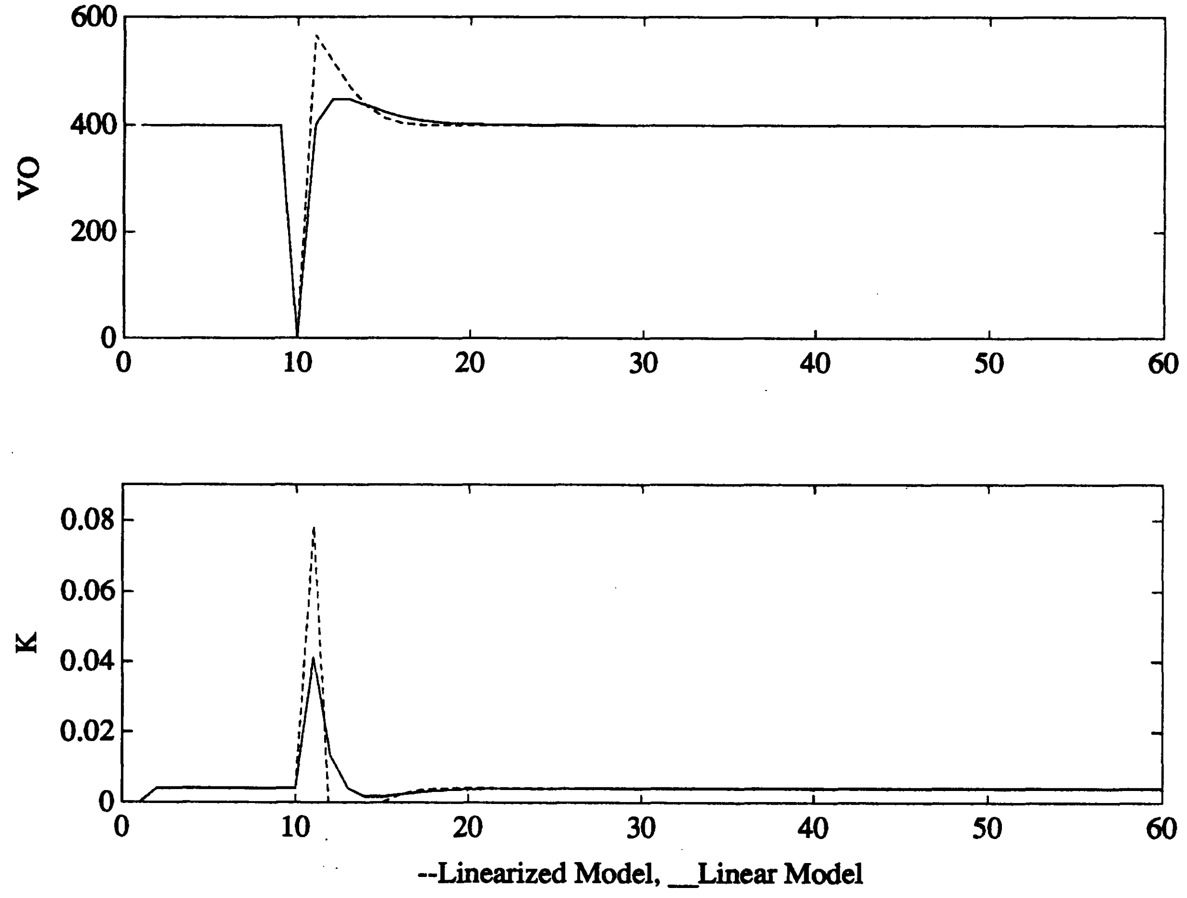


Figure B-3: Response of large-signal and small-signal controllers to large voltage transients.

Appendix C

Passive Filters for Anti-Aliasing and Ripple Frequency Selection

C.1 Anti-Aliasing Filter

The output voltage signal is composed mainly of a DC component and a $120Hz$ ripple component. The microprocessor A/D samples the output voltage once every $8.33ms$, or at a rate of $120Hz$. This situation results in aliasing of the ripple harmonic and other noise components that may exist in the system. An anti-aliasing filter is constructed using passive elements to prevent this. The circuit in Figure C-1 illustrates the basic low-pass configuration used.

The transfer function for the circuit above is:

$$H(s) = \frac{1}{R_1R_2C_1C2s^2 + (R_1C_1 + R_1C_2 + R_2C_2)s + 1} \tag{C.1}$$

MATLAB^{TM} programs were written to take a set of available resistor and capacitor values and to return all the filters with poles within a specified range that may be implemented with the available resources. These programs are included at the end of this appendix.

The implemented filter uses a cascade of three second-order filters to produce a sixth-order one. The break point of the individual filters had to be well outside the bandwidth of the closed-loop system so as to not interfere with the controller's operation. This break point is especially conservative since, when such filters are cascaded, the breakpoint moves closer to zero. Also note that the sum of the phases of these three cascaded filters and the phase in the pass band of the closed-loop system needs to be a safe phase margin below 180 degrees. The presented filter provided good

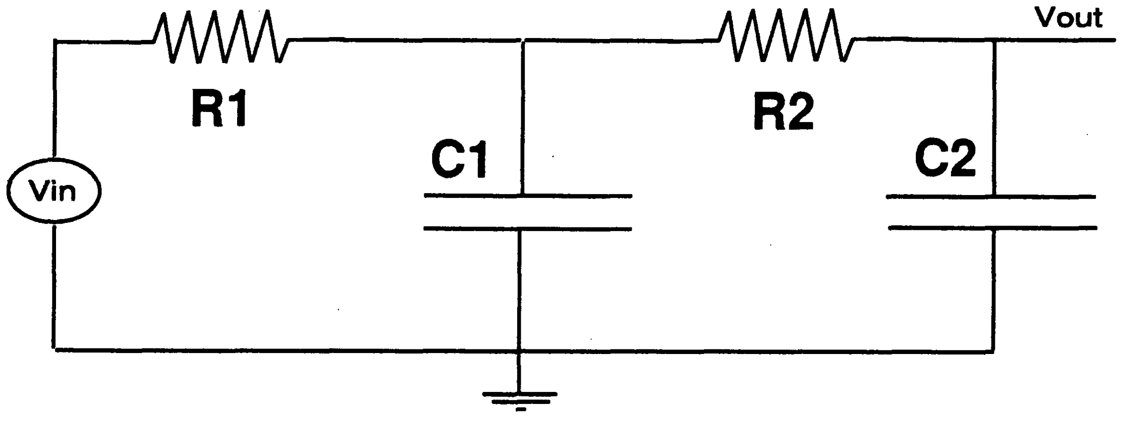


Figure C-1: A Second-Order Low-Pass Filter.

experimental results for the setup used.

The parameters of the implemented filter are:

$$R_1 = 280 Ohms \qquad R_2 = 2250 Ohms \qquad C_1 = 1\mu F \qquad C_2 = 3.3\mu F \tag{C.2}$$

The Bode plot of each individual second-order filter is shown below.

C.2 Bandpass Filter for the 120Hz Ripple

In order to select the $120Hz$ component of the output voltage to estimate the load capacitance, a bandpass filter is required. Again, only passive components were used to avoid introducing additional noise into the system. The filter was implemented using a cascade of second-order lowpass and highpass filters. The highpass configuration is shown below in Figure C-3.

The transfer function for this filter is:

$$H(s) = \frac{R_1R_2C_1C_2s^2}{R_1R_2C_1C_2s^2 + (R_1C_1 + R_1C_2 + R_2C_2)s + 1} \tag{C.3}$$

The parameters of the implemented high-pass filter are:

$$R_1 = 200 Ohms \qquad R_2 = 3400 Ohms \qquad C_1 = 10\mu F \qquad C_2 = .47\mu F \tag{C.4}$$

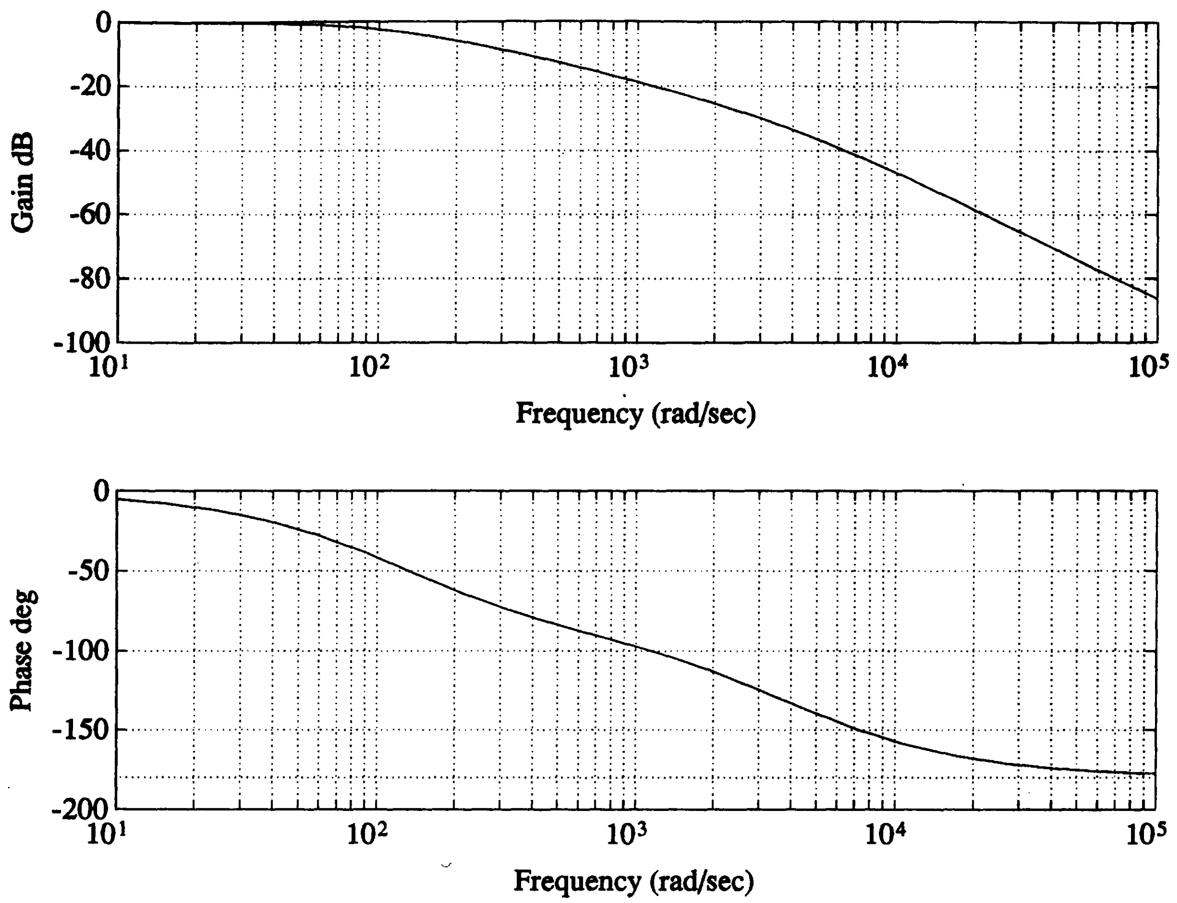


Figure C-2: Bode plot for implemented anti-aliasing filter.

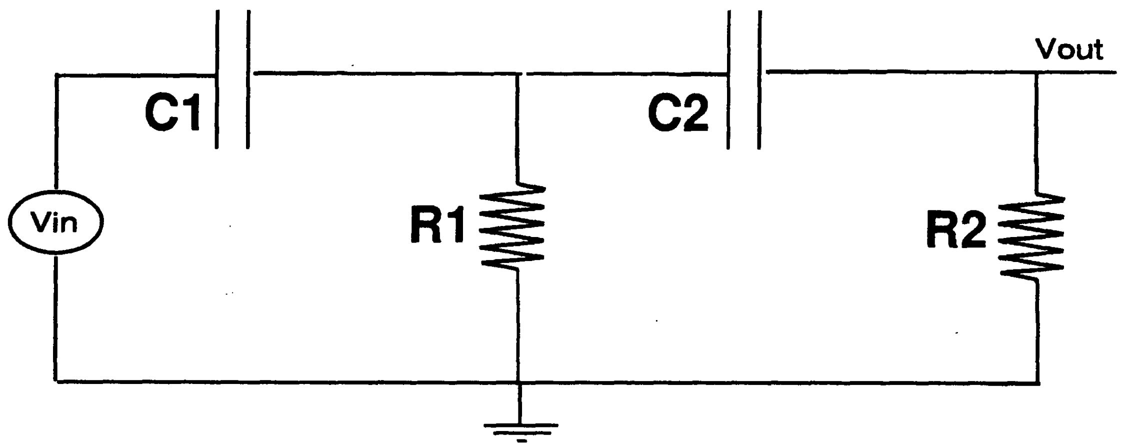


Figure C-3: A Second-Order High-Pass Filter.

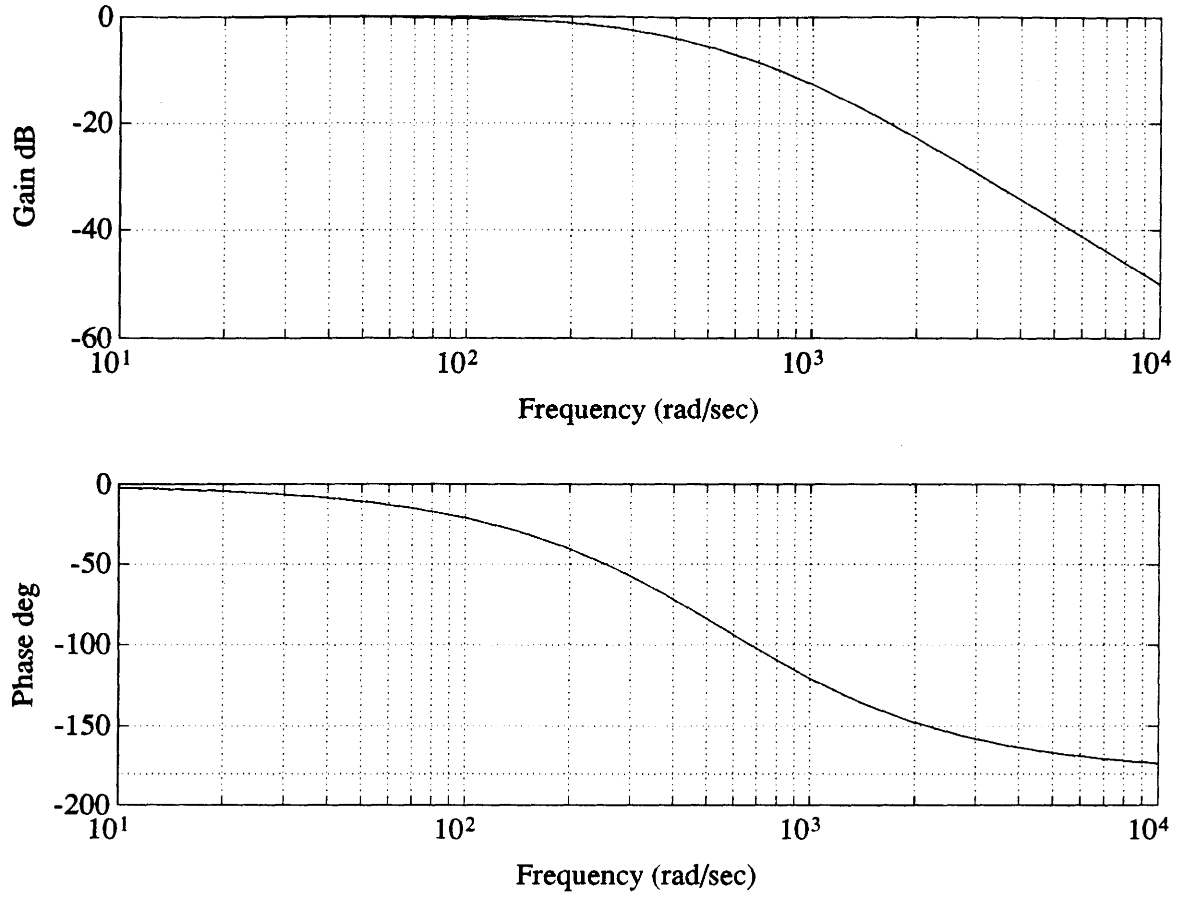


Figure C-4: Bode Plot for Low-Pass Component of Ripple Band-Pass Filter.

The parameters of the implemented low-pass filter are:

$$R_1 = 400 Ohms \qquad R_2 = 2400 Ohms \qquad C_1 = 22\mu F \qquad C_2 = 2.2\mu F \qquad (C.5)$$

The Bode plots for these filters are shown below. The output voltage was passed through a Low-High-Lo configuration of these filters before being sent to the ripple sensing channel on the A/D converter. A gain stage shown in the schematics is used to amplify the ripple signal, which is attenuated due to the less than unity magnitude of the filter responses at the $\omega = 2\pi \times 120 rad/sec$ point. This is discussed under estimation of the load capacitance.

The following pages contain the MATLAB^{TM} code used to design the filters discussed above.

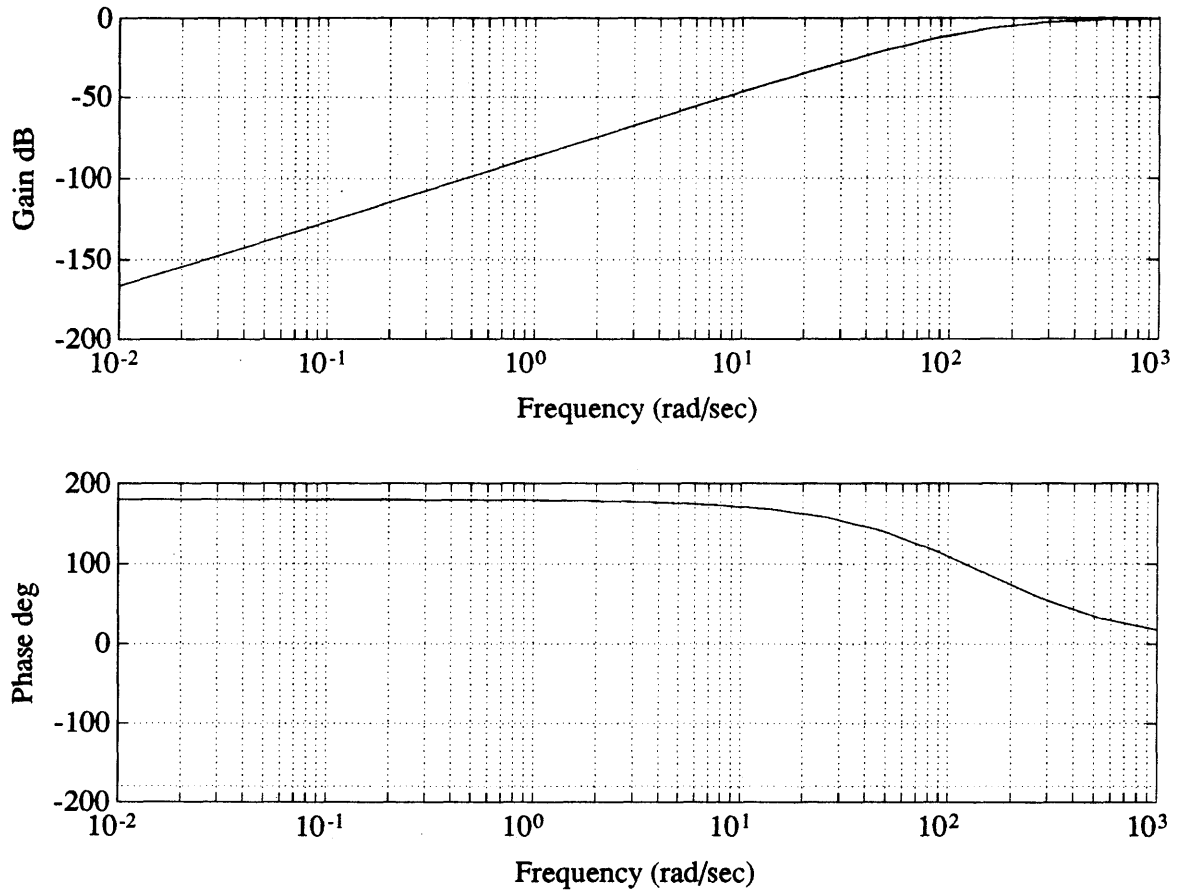


Figure C-5: Bode Plot for High-Pass Component of Ripple Band-Pass Filter.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Searching code to find a list of filters that meet a %%
%% specified range for the poles and zeros of a given   %%
%% filter transfer function.  The code allows the user  %%
%% to specify the limits on the poles and zeros of a    %%
%% transfer function, which is specified also by the    %%
%% coefficients of the numerator and denominator of the %%
%% the transfer function.  The circuits that satisfy    %%
%% the pole-zero specs are returned in the form of a    %%
%% list of the parameters for that circuit and the pole %%
%% and zero locations for those parameters.  The list   %%
%% of parameters to be searched over is specified as a  %%
%% group of arrays in the beginning of the file.        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Assign the array 'a' to different capacitor values to
%% be used.
%% The length of this array may be varied.
a(1) = 1e-6;
a(2) = 3.3e-6;;
a(3) = 10e-6;

%% Since the circuit specified here has two caps that may
%% on the same values, set the search list for the second
%% cap to be that of the first...

b = a;

%% Array 'aa' is a list of resistor values

aa(1) = 200;

aa(2) = 400;
aa(3) = 600;
aa(4) = 1800;

%% The second resistor in this circuit may take on values
%% from the same list as the one before
```

```
bb = aa;

%% Initialize a counter to keep track of how many solns
%% have been found

count = 0;

%% Loop with as many nested loops as you have variable
%% parameters in the circuit -- here 4 times.

for i = 1 : 3

for j = 1 : 3

for k = 1 : 2

for l = 1 : 2

%% Assign a combination of the parameters in the lists
%% to the circuit values. Loop thru all possibilities.

c1 = a(i);
c2 = b(j);
r1 = aa(k);
r2 = bb(l);

%% Specify the transfer function for the circuit
%% to be designed. Here, the coefficients of descending
%% powers of s in the numerator and denominator of the
%% laplace domain transfer function are specified in terms
%% of the variable parameters.

%% For lowpass filter, numerator is 1

n1= 1;

%% Denominator for 2nd order low-pass filter

d1 = r1 * r2 * c1 * c2;
d2 = (r1 * (c1 + c2)) + r2 * c2;
```

```
d3 = 1;

num = [n1];
den = [d1 d2 d3];

%% Specify conditions on the poles and zeros
%% For this example, only poles are necessary,
%% since there are no finite zeros

%% Find roots of denominator (poles) and condition
%% the parameter combinations to be saved by these
%% conditions....

rt = roots(den);
if ((rt(1) > -1000) & (rt(2) > -1000))
if ((rt(1) < -400) & (rt(2) < -400))

%% Increment count of how many solns there are.

count = count + 1;

%% Save the first and second roots (poles)

soln1(count)  = rt(1);
soln2(count) = rt(2);

%% Save the capacitor and resistor values (parameters)

s1(count) = c1;
s2(count) = c2;
s3(count) = r1;
s4(count) = r2;

end
end

%%

end
end
```

```
end
end
```

Appendix D

Simulation Code

Simulations for the system were written in MATLAB^{TM}. Many programs were written and are available from the author. Only the core programs from which the rest of the simulations were derived are included here. At many points, certain features were commented out. Documentation of the code is included to guide the user in changing the parameters of the simulation if necessary. It is important to remember that the code was developed by the author for his own use, and mainly to study the implemented circuit in the easiest manner possible. Hence there is much room for increasing the efficiency of the code and its modularity that was not explored here.

D.1 Simulations of the Discrete-Time System

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This code simulates the closed loop discrete time system  %%
%% without the peripheral resolution mapping hardware or any %%
%% digitizing effects on the inputs, outputs, and controller %%
%% parameters.  It is a simulation of the steady-state       %%
%% behavior and response to transients.  Many variations of  %%
%% this code can and were used, but are not included. Where  %%
%% appropriate, comments have been made as to where changes  %%
%% could be made to accomodate simulations of different      %%
%% situations.  In a more modular simulation, the parameters %%
%% could be defined at the beginning of the file or made     %%
%% arguments of the matlab file for easier modification of   %%
%% the simulation.. This is simply rough code as design aid. %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Initialize the state variable y = square of output voltage
%% Here since simulating steady state, initialize to steady
%% state value.

y(1) = 400 * 400;

%% Period of rectified line cycle, and period of the
%% T1- Sampled Data Model is 8.3 ms.  Assign values to input
%% peak voltage, initial load in watts, initial command
%% (steady state), and the discrete time gains as computed
%% using the discrete time model. verr is 0 in steady state
%% and acc, or integrator, is set to its steady state value.

T1 = 8.33333e-3;
C = .00047;
V = 156;
P = 50;
k(2) = (2 * P) / (V * V);
vo(1) = 400;
h1 = -1;
h2 = -.25;
verr = 0;
acc = 2 * P * T1 / (C * h2);

%% Loop through so many T1 periods, or half line cycles

for i = 2:200

%% Save the accumulator value for displaying
%% and debugging later

ac(i) = acc;
y(i) = y(i - 1) + T1/C * ((V * V) * k(i) - 2 * P);
vo(i) = sqrt(y(i));

%% This implements a load transient

if i == 30
```

```
  P = 100;
end

%% Error computation here assumes reference is 400 V..
%% Other references may be used, or a variable could
%% be placed there and modified at the beginning of the
%% program as desired

verr = y(i) - 400 * 400;

%% g is the output of the controller (input into plant)

g = (C / (T1 * V * V)) * (( h1 * verr) + (h2 * acc));

%% This conditional implements a saturation function on the
%% controller command.  May be removed if the theoretical
%% dynamics of the model are to be tested, or modified based
%% on the setup.

if g > .0255
  k(i + 1) = .0255;
else k(i + 1) = g;
end

%% This part of the code updates the accumulator to implement
%% the integration function.  The conditional implements an
%% upper bound on the accumulator value, specified as the
%% value which will yield the maximum command in steady state
%% with the proportional part contributing nothing.  Again this
%% has a lot of room for improvement in terms of being predefined
%% as a function of h2 and max current.

acc1 = acc + verr;
if acc1 > -44184.5
acc = acc1;
else
acc =  -44184.5;
end
end
```

D.2 Simulation of Digital System and Multiple Features

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This simulation code has as a core the earlier code,  %%
%% but includes a simulation of the different features   %%
%% added in hardware to the system implementation and    %%
%% other features such as integrator anti-windup.  These %%
%% features are documented in the different sections of  %%
%% the code and may be commented out.  Improvements on   %%
%% the existing code in terms of modularity are also     %%
%% mentioned but not implemented, as the purpose was not %%
%% to develop a friendly simulation tool but to use this %%
%% code to study the implemented circuit.  The code was  %%
%% used only by the author, and some work needs to be    %%
%% done before others may find it easy to use........... %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% hack 2 to 2.8 mapped
%% startup with problems (k changes too much)
%% softstartup solutions and antiwindup
%% work2 new simulation with imax = .94 and diff load transient (not doubling)
%% and new gains
%% also has input voltage changing on transient

%%******************************************************%%
%% No startup behavior is shown here.  For simulating   %%
%% startup, I use the same model for the UPF as the one %%
%% used for steady state. To carry out this simulation, %%
%% include the following initial assignments instead of %%
%% the ones in the code below.                          %%
%%******************************************************%%

%% STARTUP ASSIGNMENTS

%% y(1) = 150 * 150;  %% since the output goes to input
      %% peak before boosting is started
%% k1(2) = 0;
%% k(2) = 0;           %% initial command is set to 0 for
      %% soft startup
%% vo(1) = 150;

%% May leave load initial value as is, or since in the
%% actual implementation we use resistive loads, assign
%% P = y(i) / R, where R is a resistance of the load

%% Assign system parameters
%% Period of rectified line cycle, and period of the
%% Tl- Sampled Data Model is 8.3 ms.  Assign values to input
%% peak voltage, initial load in watts, initial command
%% (steady state), and the discrete time gains as computed
%% using the discrete time model. verr is 0 in steady state
%% and acc, or integrator, is set to its steady state value.

y(1) = 250 * 250;
Tl = 8.33333e-3;
C = .00047;
V = 156;

%% As in the expermimental setup the voltage level is dropped
%% the load power is changed in the resistive load.

P = 27.22;

k1(2) = (2 * P) / (V * V);    %% Steady state command

%% Gains are adjusted to move poles of system around.. This
%% system here is slow. poles around .9.

vo(1) = 250;
h1 = -.1527;
h2 = -.0056;
verr = 0;

%% If implementing soft startup need to step reference voltage
%% up initially.  Initialize variable vr to initial reference
%% and then step up over the first few cycles. vr also is
%% based on the transformation of vo thru hardware etc.........


%% vr = 370050    %% uncomment if implementing soft start


%%*******************************************************%%
%% Digital Controller Parameters    %%
%% The digital command is mapped from 0 to 255, with 255  %%
%% corresponding to the max command derived from the max  %%
%% current that can be drawn.  here it is 6.3e-3 amps..   %%
%% A more modular implementation has imax and kmax as     %%
%% as variables.  The peak input is fed through the 10-bit%%
%% AD converter, with a gain of 2^10 / 5, but after being %%
%% divided by 100 or so, the result being the gain here.  %%
%% the round function is used to simulate the quantization%%
%%*******************************************************%%

k(2) = round(k1(2) * 255 / 6.3e-3);
V1 = round(V * 2.046);
h11 = -190;
h21 = -7;
g = 0;

%% Accumulator value determined by transformation of output
%% voltage as shown in chapter 3

acc =  round(2 * P * T1 * 2.046 * 2.046 * 4.6 * 4.6 / (C * h2));

%% Reference voltage is the digital counterpart of 250 volts
%% as determined by transformation in chapter 3

vr = 5017600;

%% Loop with each cycle representing a T1 cycle.

for i = 2:2500

     %% For a resistive load include:
     %% P = vo(i) * vo(i) / R;
```

```
        y(i) = y(i - 1) + (T1/C) * ((V * V) * k1(i) - 2 * P);

     %% Output voltage cannot drop below 150

if y(i) < (150 * 150)
y(i) = 150 * 150;
end
vo(i) = sqrt(y(i));

     %% Load transient.  V is also changed slightly to simulate
     %% audio-susceptibility.

if i == 600
P = 42.78;
V = 153;
V1 = round(V * 2.046);
end


%%**********************************************************%%
%% The mapping of the output voltage as it goes through     %%
%% resolution mapping hardware stage in the circuit. The map %%
%% sets anything below a certain cutoff to zero.. Here it is %%
%% 210 volts.  Anything above the range of interest is as    %%
%% good as maximum.  Here it is 3.7 , but for optimal use of %%
%% full AD range it is set to 5 and the linear region of the %%
%% mapping circuit is used to fill whole range up to 5 volts %%
%% If in the region of interest, the voltage is offset then  %%
%% amplified by the ratio of the resistors................   %%
%%**********************************************************%%


if vo(i) < 210
  d(i) = 0;
else if vo(i) > 296
  d(i) = 3.7;
else d(i) = 4.6 * (vo(i) / 105 - 2);
end
end
```

```
%% Feed through AD converter

a(i) = round(d(i) * 1023 / 5);

%% end of mapping

%% Dummy arrays for monitoring values for debugging

        xx(i) = vr;
x(i) = acc;

%%**********************************************************%%
%% The output voltage is recovered by undoing the offset.    %%
%% The gains are not undone, as part of them is undone in    %%
%% the division by the input voltage squared (this undoes    %%
%% the AD gain), and the rest (gain of resolution mapping    %%
%% circuit) is taken care of in the gains as shown in chap   %%
%% 3.  This saves computation time.......................    %%
%%**********************************************************%%

%% recovery

dd(i) = a(i) + 1882;

s(i) = dd(i) * dd(i);


%% end recovery


verr = round(s(i) - vr);
z(i) = verr;                    %% Save for debugging purposes


%%Uncomment next section if simulating soft startup. The values
%% used (4000, 5017600, and initial value, may all be changed
%% based on preferred behavior... Could also be parameterized.
```

```
%%------------------------------------%%
%% if(vr + 4000) < 5017600
%%   vr = vr + 4000;
%% else vr = 5017600;
%% end
%%------------------------------------%%


%% Rounding simulates the use of fixed point arithmetic.
%% Truncating may be a better simulation, but not much
%% difference witnessed.

g = round((round(h11 * verr) + round(h21 * acc)) / (V1 * V1 * 10));
dum(i) = h11 * verr;

%% Could set g = 0 here if vo above certain level to implement
%% Upper bound shutoff....

%% Command is bound above by 255 and below by 0.

if g > 255
  k(i + 1) = 255;
else k(i + 1) = g;
end
if g < 0
  k(i + 1) = 0;
end

%% Following line simulates action of mult DAC with unitrode setup.
%% 8 bits are used (255 units of resolution) and max current used
%% here is 6.3e-3.

k1(i + 1) = (6.3e-3 / 255) * k(i + 1);


%% Anti-windup - If command in saturation level, then do not
%% accumulate error in accumulator.


if k(i + 1) == 255
```

```
x(i) = x(i);
else
acc1 = acc + verr;

%% If using soft startup and intending to disable integrator during
%% startup, replace line immediately above with following

%%----------------------%%
%% if vr == 5017600
%%   acc1 = acc + verr;
%% else acc1 = acc1; %% or acc1 = 0;
%%----------------------%%


%% Sign of accumulator kept negative..

if acc1 > 0
  acc = 0;
else

%% Upper (negative) bound on accumulator as defined by h2 and
%% maximum command.

if acc1 > -123595950
acc = acc1;
else acc = -123595950;
end

end

end
end
```

D.3 Simulations of T_s-Model for Capacitance Estimation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This code uses the same controller in the feedback loop %%
%% that is used with the Tl based model.. The controller   %%
%% designed based on the slow model.. However, this code   %%
%% models the boost converter according to the Ts model,   %%
%% which is on the time scale of the switching period. The %%
%% main use of this simulation is to show the 120Hz ripple %%
%% which is used for the capacitance estimation.  The      %%
%% dynamics of this ripple are studied with the aid of the %%
%% simulations.  Capacitance and other parameters are      %%
%% perturbed and their effect studied.  Quantization  is   %%
%% also simulated and its effect is not disruptive of the  %%
%% general operation or stability issues.  However, in the %%
%% actual system it distorts the shape of the ripple...... %%
%% This is not demonstrated in these simulations as the    %%
%% command, unlike the real system, is changed only at the %%
%% zero crossings of the ripple, but a slight modification %%
%% in the code could show the same results................ %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Define parameters of system and initial value....
%% This only shows steady state behavior.

y(1) = 400 * 400;
Tl = 8.33333e-3;
C = .00047;
V = 156;
P = 50;

k1(1) = (2 * P) / (V * V);
k1(2) = (2 * P) / (V * V);

%% Digital command with 8 bit output resolution and max
%% analog command equal to .0256

k(1) = round(k1(1) * 255 / .0256);
k(2) = round(k1(2) * 255 / .0256);
vo(1) = 400;
h1 = -.4;
h2 = -.04;
```

```
verr = 0;

%% Parameters not required by slow model
%% are the inductance and switch frequency

L = 1e-3;
Ts = 1e-5;


%% digital controller parameters

V1 = round(V * 2.046);
h11 = -2250;
h21 = -225;
g = 0;
acc = round(2 * P * Tl * 2.046 * 2.046/ (C * h2));


%% Initial Tl Cycle

%% Equations for model are those in chapter 2 under Ts model

%%%%%%%%%%%%%%%%%%
for j = 1:832
cc1 = sin(2 * pi * (j + 1) * Ts / Tl);
cc2 = sin(2 * pi * j * Ts / Tl);
cb1 = (V * V) / C;

cb2 = Tl / (2 * pi);

b1 = cb1 * Ts - cb1 * (cb2 * (cc1 - cc2));
c1 = sin(pi * (j + 1) * Ts / Tl);
c2 = sin(pi * j * Ts / Tl);
b2 = ((V * V * L) / C) * ((c1 * c1) - (c2 * c2));
dum1(j) = b1;
dum2(j) = b2;

y(j + 1) = y(j) + (b1 * k1(1)) + (b2 * k1(1) * k1(1)) - (2 * P * Ts / C);
vo(j) = sqrt(y(j));
```

```
end
%%%%%%%%%%%%%%%%%%

%% Successive cycles.  Each cycle contains 833 switch cycles.
%% An inner loop runs that simulation of switch cycles.  For
%% faster simulation lower switching frequency may be used.
%% The high frequency here is not necessary since this model
%% averages over the switch cycle, thus eliminating these
%% switch frequency harmonics which are the reason for using
%% rapid switching in the physical system.....

for i = 2:60

for j = ((i - 1) * 833) : (i * 833 - 1)
cc1 = sin(2 * pi * (j + 1) * Ts / Tl);
cc2 = sin(2 * pi * j * Ts / Tl);
cb1 = (V * V) / C;

cb2 = Tl / (2 * pi);

b1 = cb1 * Ts - cb1 * (cb2 * (cc1 - cc2));
c1 = sin(pi * (j + 1) * Ts / Tl);
c2 = sin(pi * j * Ts / Tl);
b2 = ((V * V * L) / C) * ((c1 * c1) - (c2 * c2));

dum1(j) = b1; %% For monitoring, debugging
dum2(j) = b2;

y(j + 1) = y(j) + (b1 * k1(i)) + (b2 * k1(i) * k1(i)) - (2 * P * Ts / C);
vo(j) = sqrt(y(j));
end

if i == 5
  P = 100;
end

%% Rest of simulation is straight forward and identical to that
%% of other programs.. The controller is the same....

a(i) = round(2.046 * vo(j));
```

```
s(i) = a(i) * a(i);
x(i) = acc;
verr = round(s(i) - 818 * 818);
g = round((( h11 * verr) + (h21 * acc)) / (V1 * V1 * 10));

if g > 255
  k(i + 1) = 255;
else k(i + 1) = g;
end
if g < 0
  k(i + 1) = 0;
end
k1(i + 1) = (.0256 / 255) * k(i + 1);


acc1 = acc + verr;
if acc1 > -1151300
acc =  acc1;
else acc = -1151300;
end

end
```

Appendix E

Circuit Schematics

This appendix contains schematic diagrams of the various circuits implemented. The microprocessor board was left out and is referred to in the schematics only by its connector pins. Note the following:

- Isolation Amplifiers (AD202) are used to protect the EV80C196KB and isolate it from the analog circuit.
- The output of the isolation amplifier which isolates the output of the multiplying DAC from the Boost converter is flipped in polarity in order to undo the negative sign of the output voltage of the multiplying DAC.
- All the operational amplifiers may be powered with negative and positive rails except the one marked S. Being part of the resolution mapping circuit, it needs to have a ground at its lower rail in order to implement the function described in Chapter 3.
- The clamping at the high rail around 5 volts in the resolution mapping circuit may be achieved using a diode with its anode at the output of the circuit and its cathode at the 5 volt supply from the EV80C196KB. Another way of achieving this is to just use the 5 volt supply as the positive power rail for the operational amplifier in the circuit. If the opamp does not pull close to its high rail, however, the result is loss of resolution.
- The isolation amplifiers can source only about mA of current. This necessitates the addition of followers at the output of these isolation amplifiers as shown in the schematics.
- Tables showing component values are included in this appendix.

Table E.1: Resistor Values

Resistor #	Resistor Value	Resistor #	Resistor Value
R1	374 kΩ	R22	2.4 kΩ
R2	10 kΩ	R23	400 Ω
R3	5.6 kΩ	R24	3.4 kΩ
R4	5 Ω	R25	200 Ω
R5	.25 Ω	R26	2.4 kΩ
R6	20 Ω	R27	400 Ω
R7	15 kΩ	R28	4.4 kΩ
R8	22 kΩ	R29	4.4 kΩ
R9	3.9 kΩ	R30	20.2 kΩ
R10	3.9 kΩ	R31	20.2 kΩ
R11	24 kΩ	R32	400 Ω
R12	1.6 kΩ	R33	600 kΩ
R13	2.7 kΩ	R34	280 Ω
R14	97.3 kΩ	R35	2.25 kΩ
R15	10 kΩ	R36	280Ω
R16	19.8 kΩ	R37	2.25 kΩ
R17	1.6 kΩ	R38	280 Ω
R18	18.4 kΩ	R39	2.25 kΩ
R19	1.4 kΩ	R40	135 kΩ
R20	18.6 kΩ	R41	82 kΩ
R21	8.4 kΩ	R42	62 kΩ

Table E.2: Capacitor Values

Capacitor #	Capacitor Value	Capacitor #	Capacitor Value
CL	470 μF	C11	2.2 μF
C1	.47 μF	C12	22 μF
C2	1 nF	C13	1 μF
C3	62 pF	C14	3.3 μF
C4	620 pF	C15	1 μF
C5	270 pF	C16	3.3 μF
C6	1 μF	C17	1 μF
C7	2.2 μF	C18	3.3 μF
C8	22 μF	C19	.47 μF
C9	.47 μF	C20	.1 μF
C10	10 μF	C21	22 pF

Table E.3: Other Components

Component	Part # or Value	Component	Part # or Value
D1	UHVP806	D2	IN5820
L	1 mH		

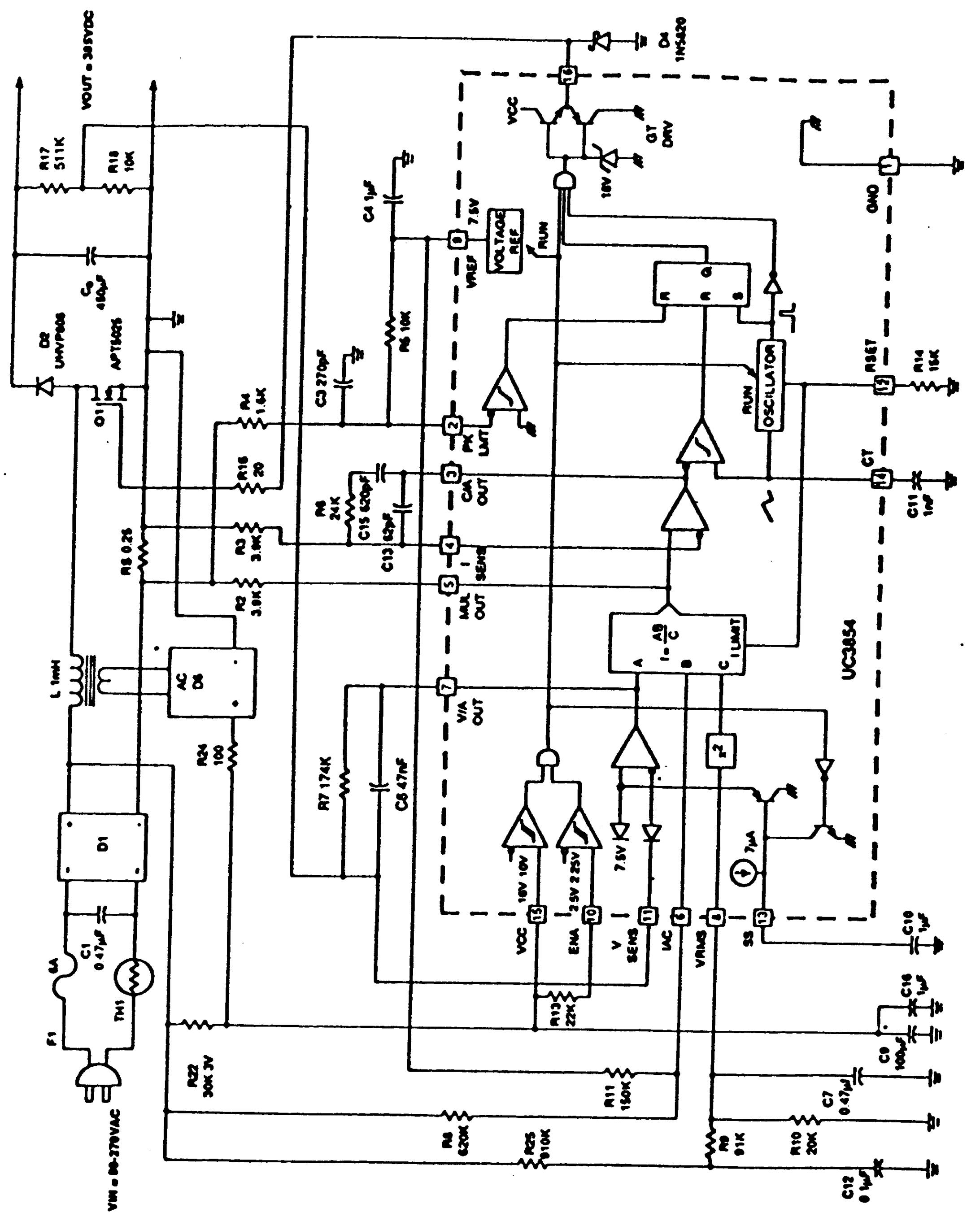


Figure E-1: Unitrode Application Circuit

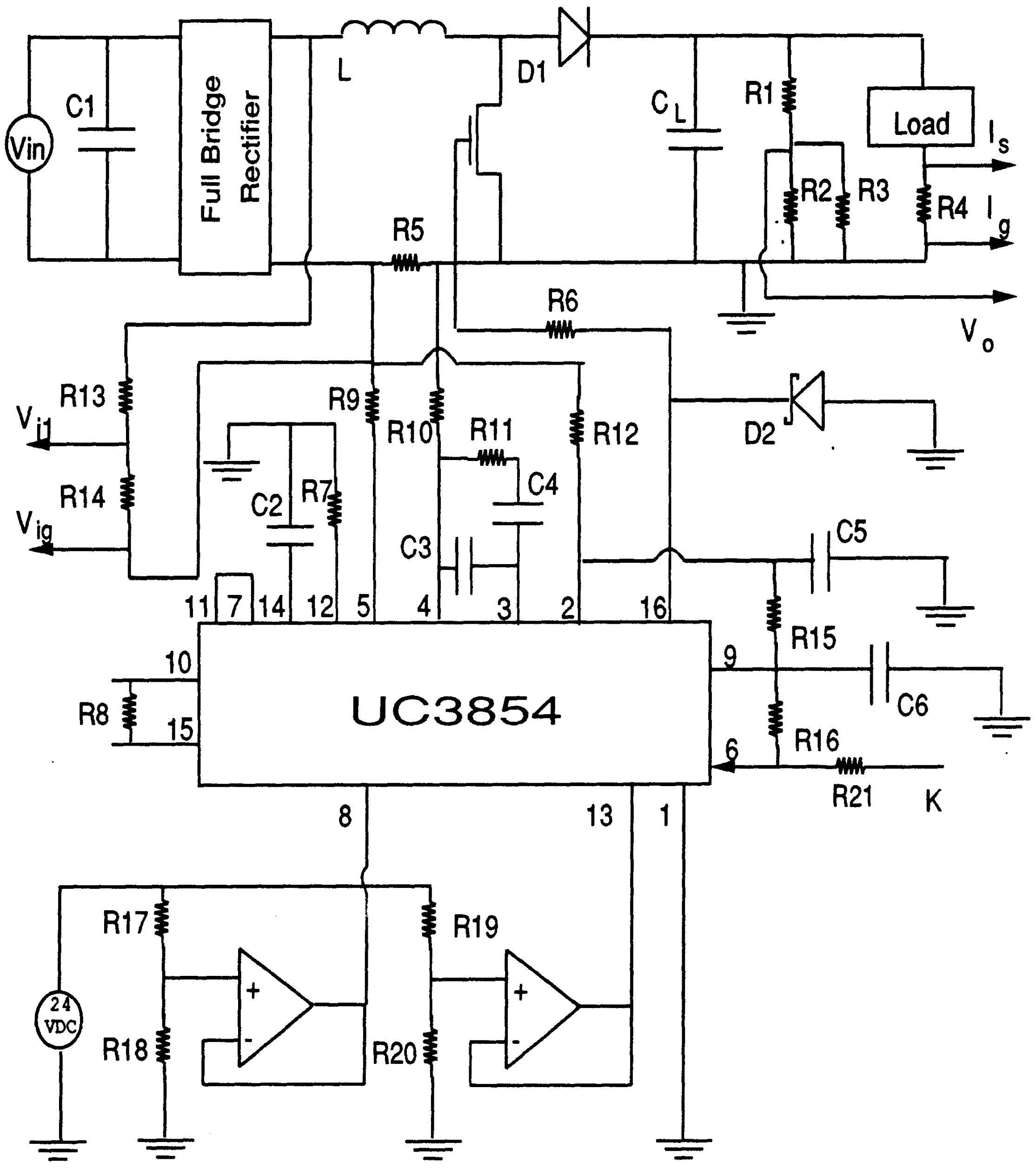


Figure E-2: Boost Converter and Analog Current Loop.

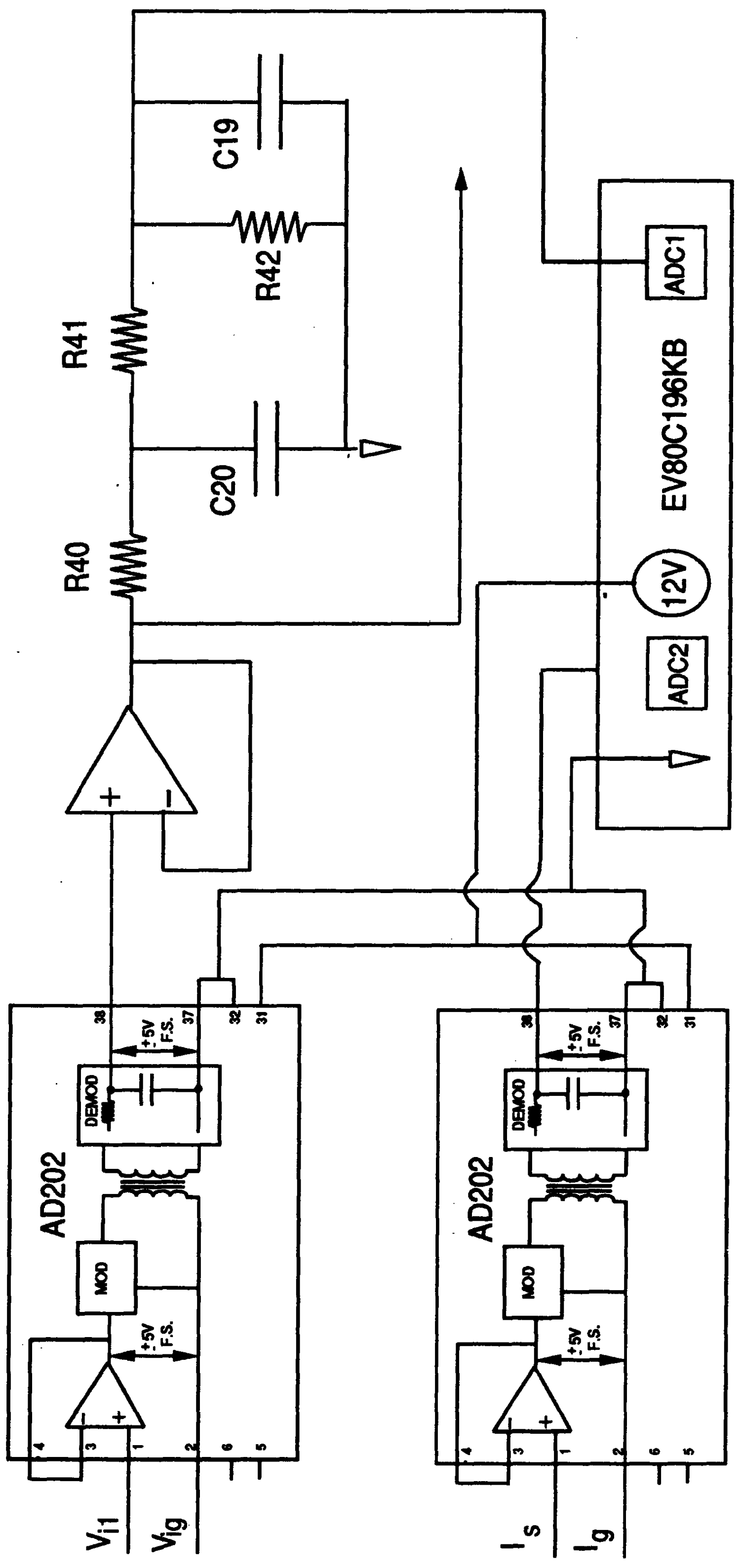


Figure E-3: Input Circuitry for Sampling Input Voltage and Load Current.

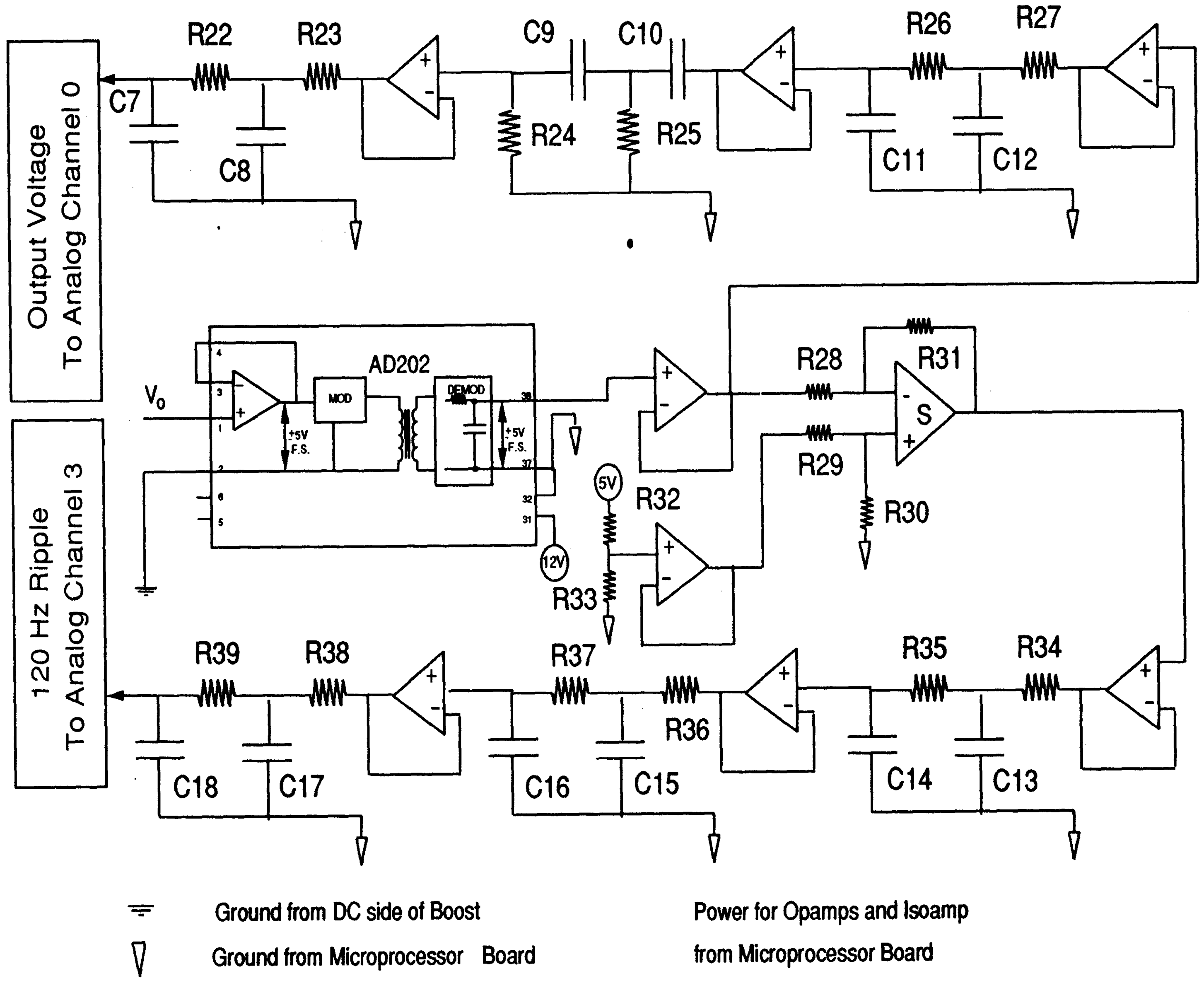
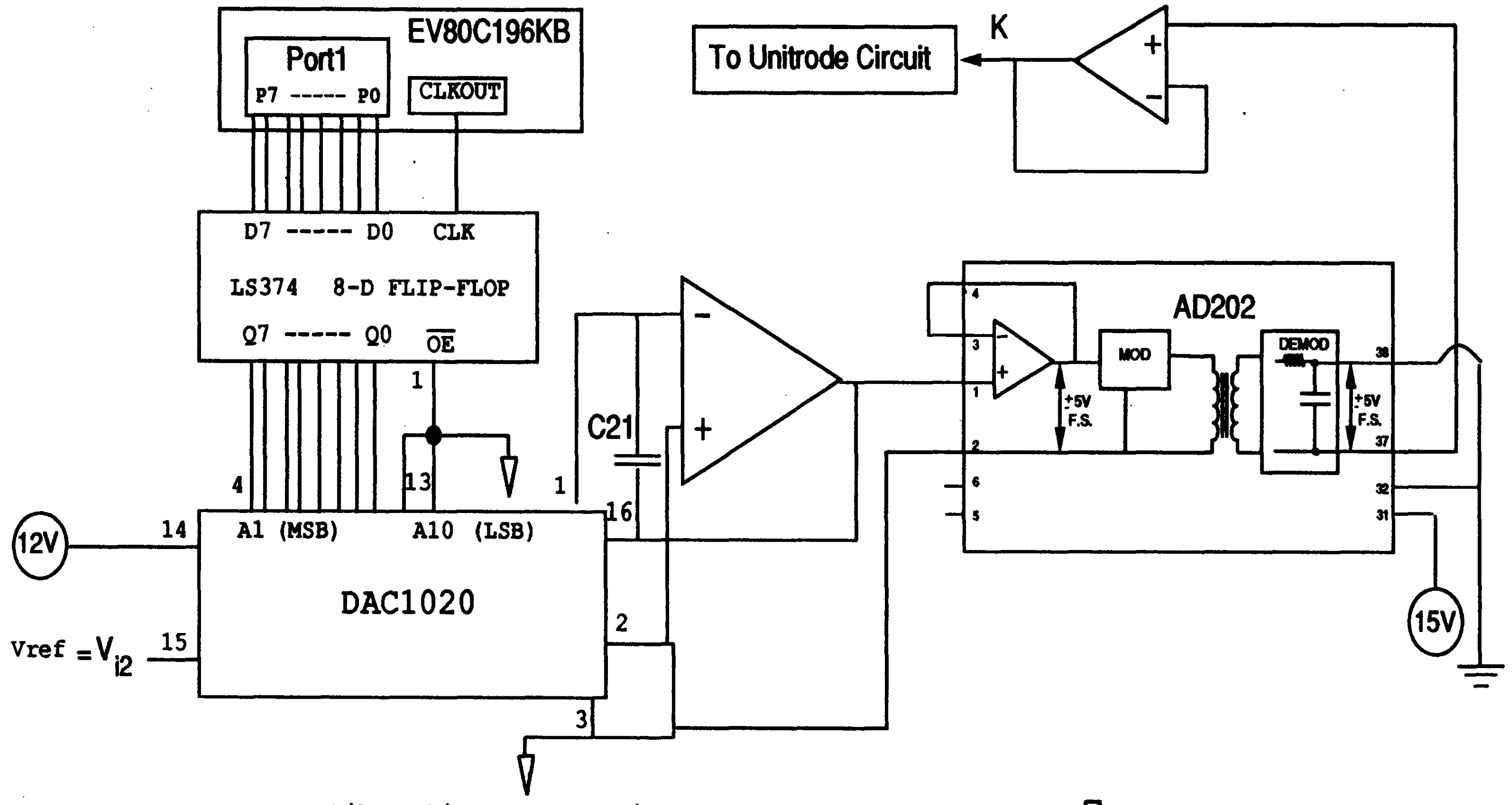


Figure E-4: Input Circuitry for Sampling Output Voltage and 120Hz Ripple.



```
Vout = -Vref(A1/2 + A2/4 + ... + A10/1024)
An = 1 if nth digital input is high.
An = 0 if nth digital input is low.
```

12V and ▽ from EV80C196KB
15V and ⏚ from Unitrode

Figure E-5: Circuitry for Converting Digital Command to Analog Voltage.

Appendix F

Controller C Code

```
/*******************************************/
/*   Micro-Controller C Code          */
/* This code implements the PI controller  */
/* for the UPF with additional features.   */
/* Soft Startup, Antri-windup, Command     */
/* saturation, and fixed command control   */
/* in the steady-state are implemented.    */
/* While this code performs the functions  */
/* required, there is room for improvement */
/* in terms of coding efficiency and time  */
/* and space consumption.  Some effort has */
/* been taken to implement a clever,       */
/* efficient program, but more can be done */
/* in that department.  The software is    */
/* is mainly interrupt driven, and as a    */
/* helpful reference in understanding the  */
/* register control and interrupt methods  */
/* used, see the user's manual for the     */
/*         80C196KB                   */
/*******************************************/

/* The next 2 lines reserve locations 30 to */
/* 38 as they are used by the EV80C196KB    */
/* and should not be modified in code that  */
/* runs on a microcontroller in the board.  */
```

```
register char apple[9];
#pragma locate (apple = 0x30)

/* Specify micro model to be KB */

#pragma model(kb)

/* Specify Interrupt Service Routines for */
/* interrupts expected.   */

#pragma interrupt (software_timer = 5)
#pragma interrupt (analog_conversion_done = 1)

#include<80C196.h>

/* Digital Gains for PI controller */

/*#define h1 190;*/
/*#define h2 7;*/

register long h1;
register long h2;

/* v1 and v2 are used to read in AD registers */
/* and then combined to give 10 bit values for*/
/* vo and vi......       */

register unsigned int v1;
register unsigned int v2;
register unsigned int vo;
register unsigned int vi;

/* result - holds the PI controller command  */
/* check - tells which AD channel was read   */
/* vr - reference voltage squared      */
/* ve - voltage error      */
/* res_temp and res_t - used for computing   */
/*       result, PI command.....             */
/* acc_temp - computes new accumulator value */
/* acc - equals acc_temp unless negative or   */
/*   over saturation limit.....       */
/* count1 and count2 - counters.. see code    */
/* flag, sum, dum1 - used in steady-state     */
/* band control      */

register unsigned char result;
register unsigned char check;
register long vr;
register long ve;
register long res_temp;
register long acc_temp;
register long acc;
register long res_t;
register unsigned int count1;
register unsigned char count2;
register unsigned char flag;
register unsigned int sum;
unsigned int dum1;

/*Adaptive control params*/
/*ripple - holds sampled value of ripple */
/*rmin and rmax hold the maximum and */
/*minimum values for a specific 8.3 ms cycle*/
/*rmin1 and rmax1 accumulate rmin and rmax*/
/*over several cycles for averaging*/
/*rmin2 and rmax2 are the resulting averages*/
/*c and c1 used in computing estimate*/
/*i is sampled current of load*/
/*count3 and count4 used for sampling and */
/* averaging, respectively */
/*eps - ripple measured*/
/*cest = estimated capacitance*/

register unsigned int ripple;
register unsigned char count3;
register unsigned int rmin;
register unsigned int rmax;
register long rmin1;
register long rmax1;
register long c;
register long c1;
register unsigned char count4;
register unsigned int i;
register unsigned int rmax2;
register unsigned int rmin2;
register unsigned int eps;
register long dum2; /*debugging*/
register long cest;

/* Software-timer ISR.. Software timer used */
/* for Sampled-Data-Model timing and Startup*/

void software_timer(void)
{
/* Use count1 and loop 500 times with */
/* time equal to 10000 timer units to */
/* give user time to power up analog  */
/* interface - time is about 3 seconds*/
/* ioport1 writes controller command. */

if (count1 < 500)
{
ioport1 = 0;
count1 = count1 + 1;
hso_command = 0x18;
hso_time = timer1 + 10000;
}
else

/* After startup, if flag = 1, which */
/* means that we are in the steady   */
/* state band in which we need to    */
/* eliminate quantization, disable PI*/
/* and average the last 18 commands  */
/* to find desired command...        */
/* Otherwise, write "result", which  */
/* is output of PI controller...     */

{
```

```
if (flag == 1)
ioport1 = sum / 18;
else
ioport1 = result;

/* Set software timer for 8.3 ms */
/* and sample AD channel 0  */

hso_command = 0x18;
hso_time = timer1 + 6254;
ad_command = 8;
}
/* collect peaks from diff cycles for averaging*/


}

/* AD conversion completed ISR.  Entered when a */
/* is completed, which in this code is either   */
/* from channel 0 or 1... */

void analog_conversion_done(void)
{
/* This implements the soft-startup mechanism */
/* by stepping up the reference voltage.      */

if ((vr + 4000) < 5017600)
vr = vr + 4000;
else vr = 5017600;

/* The channel sampled is in the bottom 3 bits */
/* of ad_result_lo.....        */

check = ad_result_lo;

/* If channel 0 was sampled, then that is vo */

if ((check & 7) == 0)
{
v1 = (unsigned int) ad_result_hi;
```

```
v1 = v1 << 2;
v2 = (unsigned int) ad_result_lo;
v2 = v2 >> 6;
vo = v1 + v2;

/* dum1 used for band control as vo value changes */
/* in undoing resolution mapping etc.. we need    */
/* value of vo at 1 point in computation for later*/
/* on in the controller.....    */

dum1 = vo;

/* Sample Channel 1 */
/* The next few lines implement segmented control around */
/* the steady state.. This means the PI controller is    */
/* disabled and an average of steady state commands is   */
/* used in order to avoid quantization oscillations....  */

/* If not already in "steady-state" for 18 cycles, but   */
/* this cycle is within the band, increment count2..     */

if ((dum1 < 368) && (dum1 > 348) && (vr == 5017600) && (count2 < 18))
 {
 count2 = count2 + 1;
     sum = sum + result;
 }

/* If in "steady-state" for 18 cycles, set flag = 1 to */
/* indicate that officially in steady state and can    */
/* fix output command based on previous 18 commands....*/

if ((count2 == 18) && (dum1 < 368) && (dum1 > 348) && (vr == 5017600))
{
flag = 1;
}
else

/* If outside steady-state band, reset flag = 0, */
/* count2 = 0, sum = 0 */
```

```
if ((dum1 > 380) || (dum1 < 326))
{
flag = 0;
count2 = 0;
sum = 0;
count4 = 0;
rmin1 = 0;
rmax1 = 0;

}
if (flag == 1)
{
ad_command = 11;
count3 = count3 + 1;
}
else
{
ad_command = 9;
}}

/* If channel 1 was sampled, then that is vi */

else if ((check & 7) == 1)
{
v1 = (unsigned int) ad_result_hi;
v1 = v1 << 2;
v2 = (unsigned int) ad_result_lo;
v2 = v2 >> 6;
  vi = v1 + v2;

/* If vo low enough, avoid computations */
/* immediately saturate... */

if (vo < 50) result = 255;
  else
/* Undo Resolution mapping then */
/* carry out PI control.. */

{
vo = vo + 1882;
ve = (long) vo * (long) vo;
ve = ve - vr;
ve = 0 - ve;
res_temp = ve * h1;
res_t = acc * h2;
res_temp = res_temp + res_t;
res_t = (long) vi * (long) vi;
res_t = res_t * 47;
res_temp = res_temp / res_t;
 if (res_temp > 255)
   result = 255;
   else

/* Accumulator control : */
/* Disable during startup and while in  */
/* specified steady-state band (flag = 1) */
/* Also disable if saturation.. This   */
/* prevents windup...  */

{
  if ((vr == 5017600) && (flag == 0))
{
 acc_temp = acc + ve;
 if (acc_temp > 650000000)
 acc = 650000000;
 else if (acc_temp < 0)
 acc = 0;
 else
 acc = acc_temp;
  }
 if (res_temp < 0)
 result = 0;
 else result = (unsigned char) res_temp;
 }
 }}
 else if ((check & 7) == 3)

 /* Succesively sample ripple */

 {
```

```
 v1 = (unsigned int) ad_result_hi;
 v2 = (unsigned int) ad_result_lo;
 v1 = v1 << 2;
 v2 = v2 >> 6;
 ripple = v1 + v2;
 if (ripple < rmin)
 rmin = ripple;
 if (ripple > rmax)
 rmax = ripple;
 if (count3 < 100)
{
count3 = count3 + 1;
ad_command = 11;
}
else
ad_command = 10; /* sample load current */
}
else if
((check & 7) == 2)

/* Read in i, load current */

{
v1 = (unsigned int) ad_result_hi;
v1 = v1 << 2;
v2 = (unsigned int) ad_result_lo;
v2 = v2 >> 6;
i = v1 + v2;
if (count4 < 200)
{
rmin1 = rmin1 + rmin;
rmax1 = rmax1 + rmax;
count4 = count4 + 1;
}
else

/*if done averaging and still in steady state*/
/*then compute capacitance and adjust gains*/
{
if (flag == 1)
{
rmin2 = rmin1 / 200;
rmax2 = rmax1 / 200;
if ((rmax2 - 390) > (390 - rmin2))
eps = rmax2 - 390;
else eps = 390 - rmin2;

c = (long) i * 10000000;
c1 = 750 * (long) eps;
c = c / c1;

/* Round instead of truncate */
/* Make sure gains do not drop */
/* to zero */

h1 = h1 * c;
if ((2 * (h1 % cest)) > cest)
h1 = (h1 / cest) + 1;
else
h1 = h1 / cest;
dum2 = c1;
h2 = h2 * c;
if ((2 * (h2 % cest)) > cest)
h2 = (h2 / cest) + 1;
else
h2 = h2 / cest;

cest = c;
if (h1 == 0) h1 = 1;
if (h2 == 0) h2 = 1;

}
count4 = 0;
rmin1 = 0;
rmax1 = 0;
}
rmin = 1023;
rmax = 0;
count3 = 0;
```

```
}

 }

main()
{
/* main driver for code. Initialize variables, */
/* reset interrupt pending and mask registers, */
/* and send into infinite loop to wait for     */
/* interrupts.....          */

h1 = 760;
h2 = 28;
dum2 = 5 % 3;
cest = 32000;
flag = 0;
count2 = 0;
sum = 0;
count1 = 0;
vr = 3900623;
result = 0;
acc = 0;
ioport1 = 0;
int_mask = 0x22;
int_pending = 0;
hso_command = 0x18;
hso_time = timer1 + 6254;
ad_command = 8;
enable();
while(1);
}
```

Appendix G

Schematic and Timing Diagrams for Compact Microprocessor System

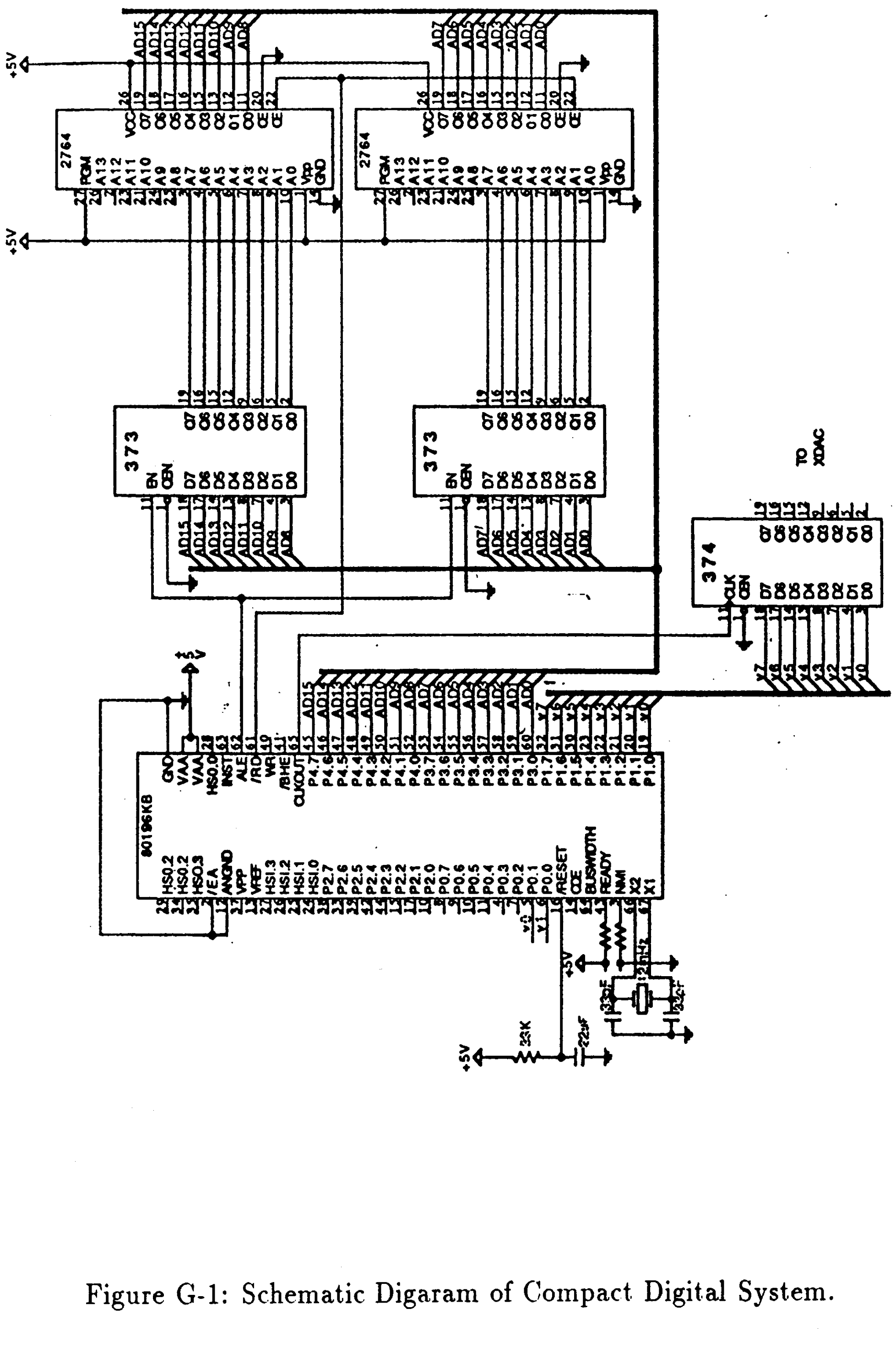
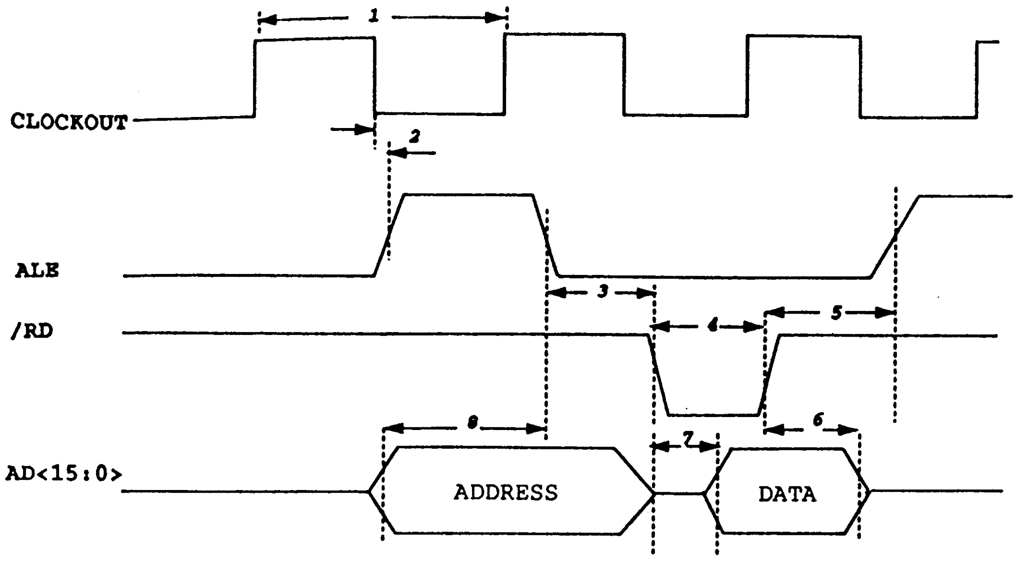


Figure G-1: Schematic Digaram of Compact Digital System.



T	Min(ns)	Max(ns)
1	166	572
2	-5	16
3	.5T1-15	T1-40
4	.5T1-5	.5t1+25
5	.5T1	.5T1+25
6		.5T1-20
7		.5T1-23
8	.5T1-20	

Figure G-2: Timing Diagram for Memory Access Cycle.

Appendix H

Transistor Level Schematics of Multiplier Cell and Block Description of Multiplier Block

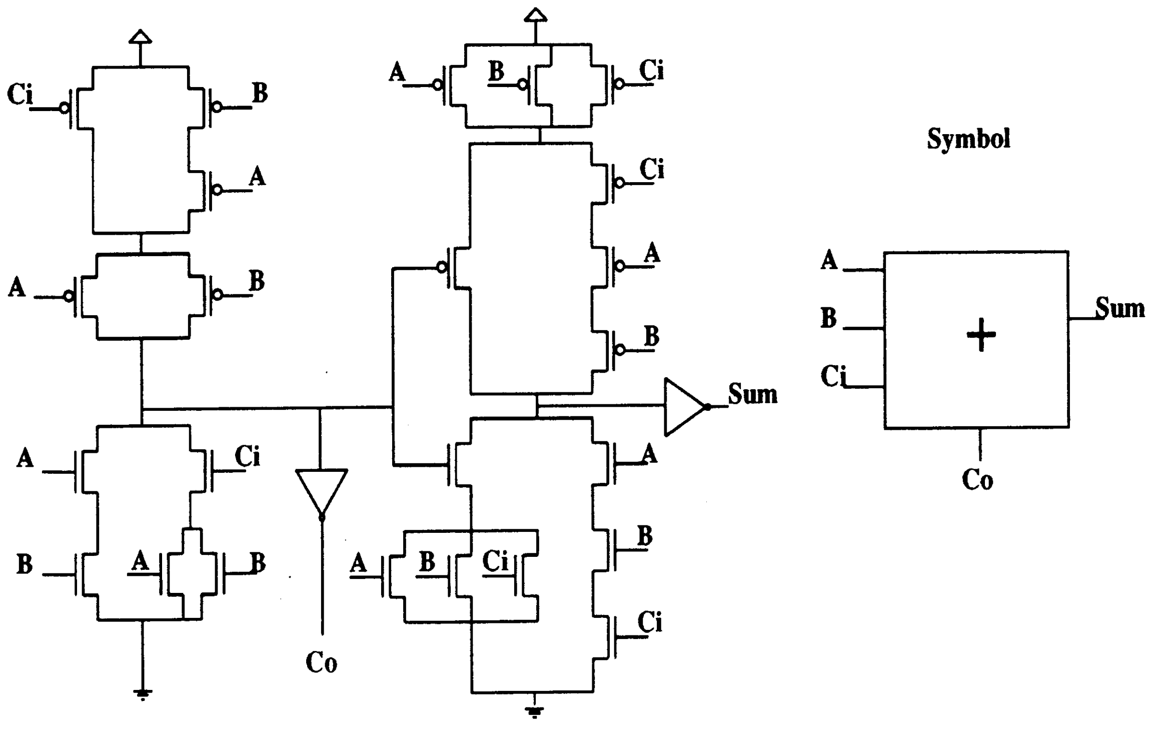


Figure H-1: Circuit Schematic for Adder Cell.

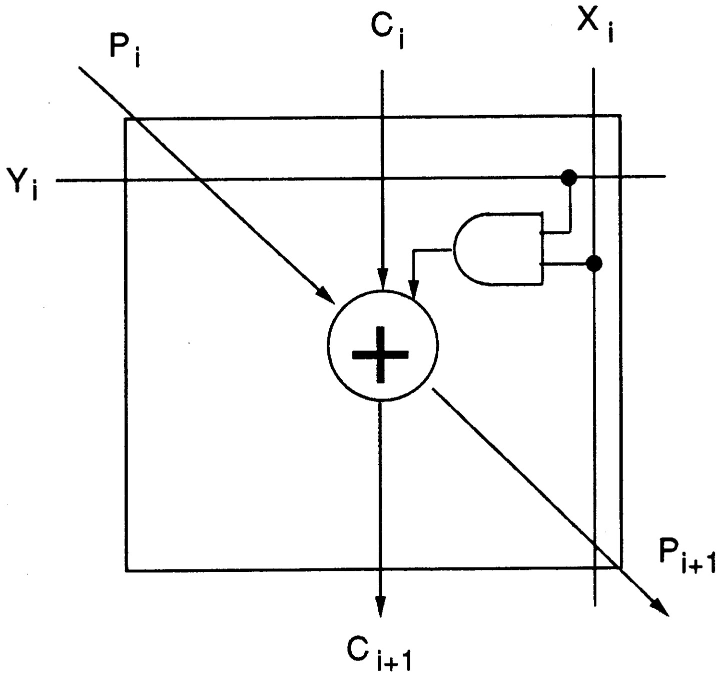


Figure H-2: A Product Cell.

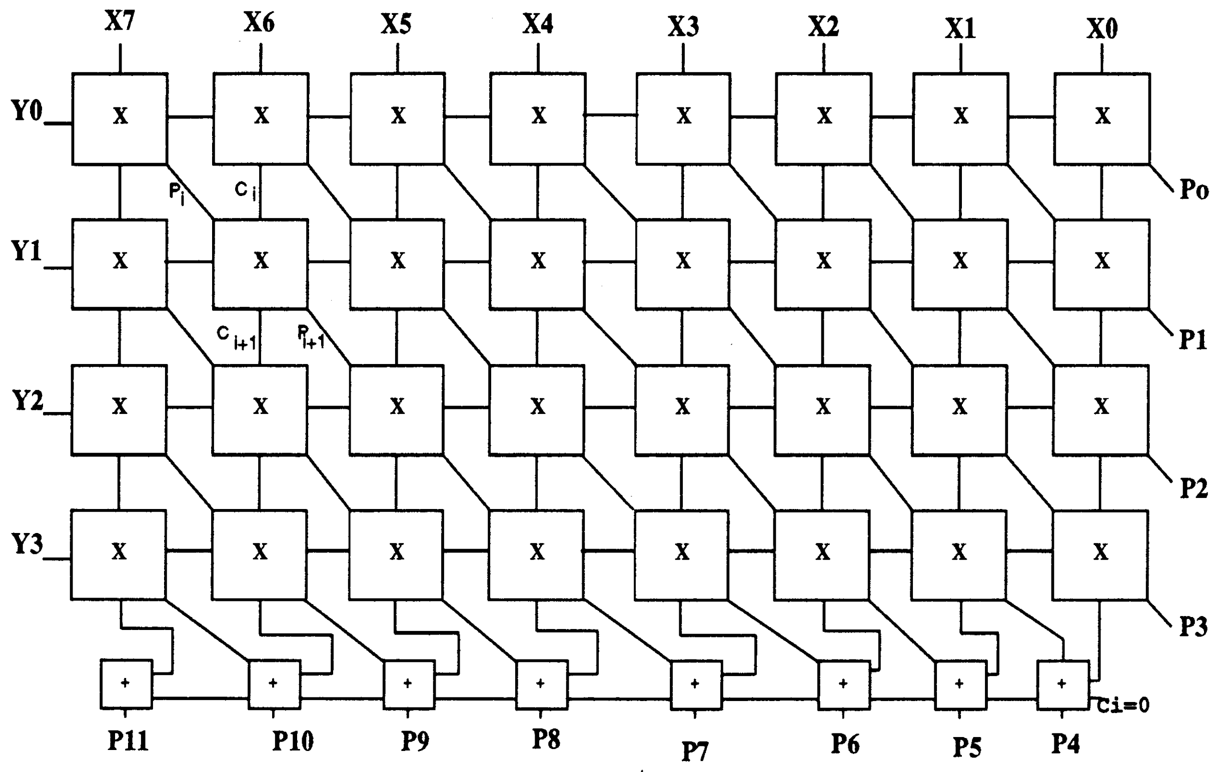


Figure H-3: An 8 by 4 Parallel Multiplier.

Bibliography

[1] J.B. Williams, "Design of Feedback Loop in Unity Power Factor AC to DC Converter", PESC, 1989, pp.959-967.

[2] B. Sharifipour, P. Cacciola, and J. Maddox, "Designing a 1200 Watt Multiple Output Modular Power System with High Power Utilization for the Workstation Environment", APEC, 1989, pp. 439 - 444.

[3] J. G. Kassakian, M. F. Schlect, G. C. Verghese, *Principles of Power Electronics*, Addison-Wesley, 1991, pp. 395-399.

[4] K. Mahabir, G. Verghese, J. Thottuvelil, and A. Heyman, "Linear Averaged And Sampled Data Models for Large Signal Control of High Power Factor AC-DC Converters", PESC, 1990, pp. 291 - 299.

[5] C.P. Henze and N. Mohan, "A Digitally Controlled AC to DC Power Conditioner that Draws Sinusoidal Input Current", PESC, 1986, pp531-540.

[6] K. Ogata, *Discrete-Time Control Systems*, Prentice Hall Inc., 1987, pp. 1 - 36.

[7] B. Orlik, H. Weh, "Microprocessor-Controlled Three-Phase Motors With High Resolution Digital Pulse Width Modulator for High Pulse Frequencies", EPE, 1985, pp. 3.39 - 3.44.

[8] C. Bergman, P. Goureau, "Direct Digital Control of a Self-Controlled Synchronous Motor with Permanent Magnet", EPE, 1985, pp. 3.269 - 3.273.

[9] J. Siebert, "Freely Programmable Digital Open-loop/Closed-loop Control System for Converters and Converter Drives", EPE, 1985, pp.5.7 - 5.10.

[10] K. Astrom, B. Wittenmark, *Computer-Controller Systems, Theory and Design*, Prentice Hall Inc., 1990, pp. 224 - 226.

[11] A. Stankovic, G. Verghese, X. Liu, and J. Thottuvelil, "Fast Controllers for High-Power-Factor AC-DC Converters", European Power Electronics, 1990.

[12] 'EV80C196KB Microcontroller Evaluation Board User's Manual, Release 001', INTEL Corporation, February 20, 1989.

[13] '80C196KB User's Guide', INTEL Corporation, October 1990.

[14] Y. Guijun, L. Norum, "Low Cost Digital Controller for PWM Converter", International Federation for Automatic Control Low Cost Automation, 1990.

[15] Eissa, Mohamed,"A Fast Analog Controller for a Unity Power Factor Converter", Master Thesis, MIT, June 1993.

[16] 'Power Factor Correction with the UC3854 - Application Note', UNITRODE Integrated Circuits,

[17] Mahabir, Krishna N., "Digital Control of Large Signal Behavior in Switched DC-DC Converters", Master Thesis, MIT, May 1989.

[18] Wilkenson, Bruce, "Power Factor Correction and the IEC 555-2", Powertechnics Magazine, Februaury 1991.

[19] Miwa, B. A.,"Interleaved Conversion Techniques for High Density Power Supplies", PhD. Thesis, MIT, June 1992.