THE CONTROL OF A STATIC VAR COMPENSATOR AND ACTIVE POWER FILTER

MOHD FADZIL MOHD SIAM

B.Eng. (Hons), AMIEE

Thesis Submitted for the Degree of Doctor of Philosophy

Heriot-Watt University

Department of Computing and

Electrical Engineering

September 1998

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to both of his supervisors, Professor B. W. Williams and Dr. S. J. Finney for their professional guidance and support throughout the research reported in this thesis.

Recognition is given to various members of the academic and techical staff at Heriot-Watt University. Particular appreciation is given to Mr. A. Houstin, Dr. J. E. Fletcher and Mr. N. Mc.Neill for their assistance. Thanks are also due to past and present members of the Power Electronics Group for their help and friendship.

During the the course of this study, the author was supported by Tenaga Nasional Berhad, Malaysia.

Finally, a heartful of thanks to both of my parents for their love and support thoughout the years, for without them I would never have gone this far.

ABSTRACT

In an AC supply system, good management of reactive power plays an important role in ensuring a good quality of supply. A solid-state static VAr compensator enables precise and continuous reactive power control to be achieved. A leading and lagging VAr can be compensated to give a system with unity power factor. This thesis describes a solid-state static VAr compensator which utilises a deadband PWM switching pattern. By using simulation as well as experimental results, a comparison is made with conventional sinusoidal PWM. The use of deadband PWM enables higher modulation indices to be achieved hence facilitating a smaller size of reactive component on the DC side. Deadbanding reduces the effective switching frequency thus minimising the switching losses and resulting stresses.

Power quality is also affected by harmonic distortion which originates from the non-linear characteristics of electrical devices and loads. The use of an active power filter to provide harmonic compensation as well as power factor correction is described in this thesis. Modified delta modulation is proposed to control the switching of an active power filter. Compensation is achieved with a reduction in switching losses. The application of a variable structure control system is considered. Sliding mode switching control is used to ensure good tracking of the reference current, thus providing the required compensation.

LIST OF PRINCIPAL SYMBOLS AND ACRONYMS

SYMBOLS

ϕ	Phase shift angle for SVC control
θ	Displacement angle between d-q frame and stationary α-β frame
e_{do}	Inverter output voltage
f_c	Triangular carrier waveform
f_o	Reference modulation waveform
m	Modulation index in PWM
I_{CE}	IGBT continous collector current
i_{Ca}, i_{Cb}, i_{Cc}	Compensating current references
i_α	Current component in α co-ordinate
i_β	Current component in β co-ordinate
p, q	General symbol for active and reactive power.
p*, q*	Active and reactive power to be compensated
p_{av}	Real power to maintain a constant DC capacitor voltage
$\overline{p}_L$	Average active component
$\tilde{p}_L$	Alternating active component
$\overline{q}_L$	Average reactive component
$\tilde{q}_L$	Alternating reactive component
u_d, u_q	Control vector in sliding mode
v_α	Voltage component in α coordinate
v_β	Voltage component in β coordinate
Vdc	DC voltage across the capacitor
Vi	Inverter output voltage
$\Delta\delta$	Change in phase shift angle

ABBREVIATIONS AND ACRONYMS

AC	Alternating current
ADC	Analogue to digital converter
APF	Active power filter
BJT	Bipolar junction transistor
CAE	Computer aided engineering
CLB	Configurable logic block
CSI	Current source inverter
DAC	Digital to analogue converter
DSP	Digital signal processor
EEPROM	Electrical erasable programmable read only memory
EPROM	Electrical programmable read only memory
ESR	Equivalent series resistance
FPGA	Field programmable gate array
FIR	Finite impulse response
IIR	Infinite impulse response
IOB	Input output block
IRPT	Instantaneous reactive power theory
IGBT	Insulated gate bipolar transistor
MOSFET	Metal oxide semiconductor field effect transistor
PC	Personal computer
PCM	Pulse code modulation
PWM	Pulsewidth modulation
RAM	Random access memory
ROM	Read only memory

rms	Root mean squared
SPWM	Sinusoidal pulse-width-modulation
SPICE	Simulation Program with Integrated Circuit Emphasis
SR	Saturated reactor
SVC	Static VAr compensator
THD	Total harmonic distortions
TSC	Thyristor switched capacitor
TCR	Thyristor controlled reactor
VSI	Voltage source inverter
VAr	Volt-amp reactive
VCO	Voltage control oscillator
VSC	Variable structure control
XACT	Xilinx Automated CAE Tools

PREFACE

SCOPE OF THESIS

This thesis considers the design and control of a solid-state static VAr compensator and active power filter. The system employs a shunt connected voltage source inverter. The control processes are implemented digitally using a floating-point digital signal processor and a field programmable gate array. Simulation is performed using PSpice and MATLAB (with Simulink) to aid the design process and provide evaluation of system performance.

The first part of the thesis describes a static VAr compensator. The system provides leading and lagging VAr for reactive power compensation. The application of deadbanding to PWM operation is explained, and it improves system performance compared with a similar system using sinusoidal PWM. The second part of the thesis presents an active power filter which is used to provide power factor correction as well as harmonic compensation. The deadbanding concept is applied to discrete delta modulation in order to provide switching control. Sliding mode switching control is considered, which tracks the reference current thus providing the required compensation.

STRUCTURE OF THESIS

The thesis is presented in seven chapters. Chapter 1 gives a brief justification of the need for the investigation and provides an introduction to harmonic and reactive power

compensation. Chapter 2 presents the background theory for the operation and analysis of the SVC and APF. A literature review is given in the area relevant to the study.

In Chapter 3, the hardware and software requirements to design, build and operate the compensation system are described. The design procedure for the power circuit is given together with its switching arrangement. The chapter explains a modulation technique which gives a 15.5% increase in gain over conventional sinusoidal PWM and minimises the switching loss. The modelling techniques are explained, and use PSpice and MATLAB software packages.

Chapter 4 investigates the operation and performance of the static VAr compensator with deadband PWM. Detailed explanation is given on the design implementation of the Xilinx FPGA and Motorola DSP96002 programming. The deadbanding concept improves the system performance and reduces switching losses. Simulation and experimental results are presented to study the effectiveness of the proposed technique.

Chapter 5 presents an active power filter system which is able to compensate both harmonics and reactive power. The ability to detect the instantaneous active and reactive components is an important criteria. The implementation of the detection technique is explained in this chapter. Discrete delta modulation is used to provide the switching control. The deadbanding concept is used to obtain a reduction in the switching losses.

In Chapter 6, sliding mode switching control is used to give the required compensation current. A brief introduction on sliding mode control theory is given. A model is defined in the *d*-q reference frame to derive equations for determining the switching control action.

The concluding chapter provides a summary of the conducted work. The author's contributions are highlighted and areas of further research are identified.

CHAPTER 1

INTRODUCTION

The term VAr (volt-amp reactive) compensation is widely used to define the management of reactive power in order to improve the quality of supply in alternating current (AC) power systems. This is done by either storing or supplying an appropriately variable quantity of reactive power. In practice this effectively means compensating a leading or lagging power factor for unity power factor where the voltage and current are in phase.

An electric power system consists of three principal parts, namely generating stations, transmission lines and distribution systems [1.1]. Transmission lines are the connecting links between the generating stations and the distribution systems and lead to other power systems through interconnections. A distribution system connects all the individual loads to the transmission lines at substations which perform voltage transformation and switching functions. An inherent characteristic of electric energy transmission and distribution by alternating current is that any real power consumption is generally associated with reactive power. The following components contribute towards the generation and storage of reactive power [1.2].

i. ***Synchronous generators :*** These can be used to generate or absorb reactive power. The ability to supply reactive power is determined by the short circuit ratio (1/synchronous reactance). In modern machines the value of this ratio is made low for economic reasons, hence the inherent ability to operate at leading

power factors is not large. An over-excited machine generates reactive power whilst an under-excited machine stores it.

ii. ***Overhead lines and transformers :*** AC transmission and distribution lines are predominantly reactive networks. When fully loaded, lines store reactive power. On light loads the shunt capacitances of longer lines may become predominant and the lines become VAr generators. On the other hand transformers always store reactive power.

iii. ***Customers' load :*** Most of the electrical loads used on the customer side, for example in factories and homes, are inductive.

1.1 IMPORTANCE OF VAR COMPENSATION

Reactive power has been recognised as a significant factor in the design and operation of alternating current electric power systems for a long time. A lot of work has been done to study its effect and ways to control it. The following points summarise why it is necessary to have a VAr compensator [1.3] :

i. since transmission and distribution lines are predominantly reactive, any changes in load and load power factor will alter the voltage profile along the transmission lines.

ii. variations in supply voltage will degrade the performance of loads, possibly leading to equipment failure.

iii. reactive power increases the transmission losses.

iv. it has been increasingly difficult to construct both generation facilities and new transmission lines due to energy, environment, right-of-way and cost problems. Therefore it is important to have better utilisation of existing systems. The use

of VAr compensators can increase stability limits thereby increasing the transmittable power.

In recent years VAr compensators have been successfully used to solve a variety of problems on large interconnected transmission and distribution systems [1.4]. These include :

i. to achieve voltage control

ii. to increase active power transfer capacity

iii. to increase transient stability margins

iv. to increase damping of power oscillations

v. to reduce temporary overvoltages

vi. to balance loading of individual phases

vii. to damp sub-synchronous resonances

viii. to provide reactive power to AC/DC converters

1.2 TYPES OF VAR COMPENSATORS

Various types of VAr compensators have been proposed and implemented as shown in Figure 1.1. Traditionally the synchronous condenser has been used to provide reactive power compensation. A condenser is basically a synchronous machine running without load. The leading or lagging VAr can be provided by controlling the field excitation current. Another obvious method that has been used for a long time is the connection of fixed capacitors or mechanically switched capacitor banks. A static VAr compensator (SVC) is a static device which controls the flow of reactive power in an electrical system. Unlike synchronous condensers, SVCs do not have inertia or any major moving

or rotating components. One example is a saturated reactor (SR) compensator. Figure 1.2 shows the basic circuit and operating characteristic of a saturated reactor compensator. The SR compensator is unique in the sense that it does not employ any solid state switches or active control. The SR is a self regulating device that responds only to changes in its terminal voltage. Compared to other types of SVCs, it has the disadvantage of incurring higher losses. However the control system is simple because of its inherent ability to vary reactive current with voltage.

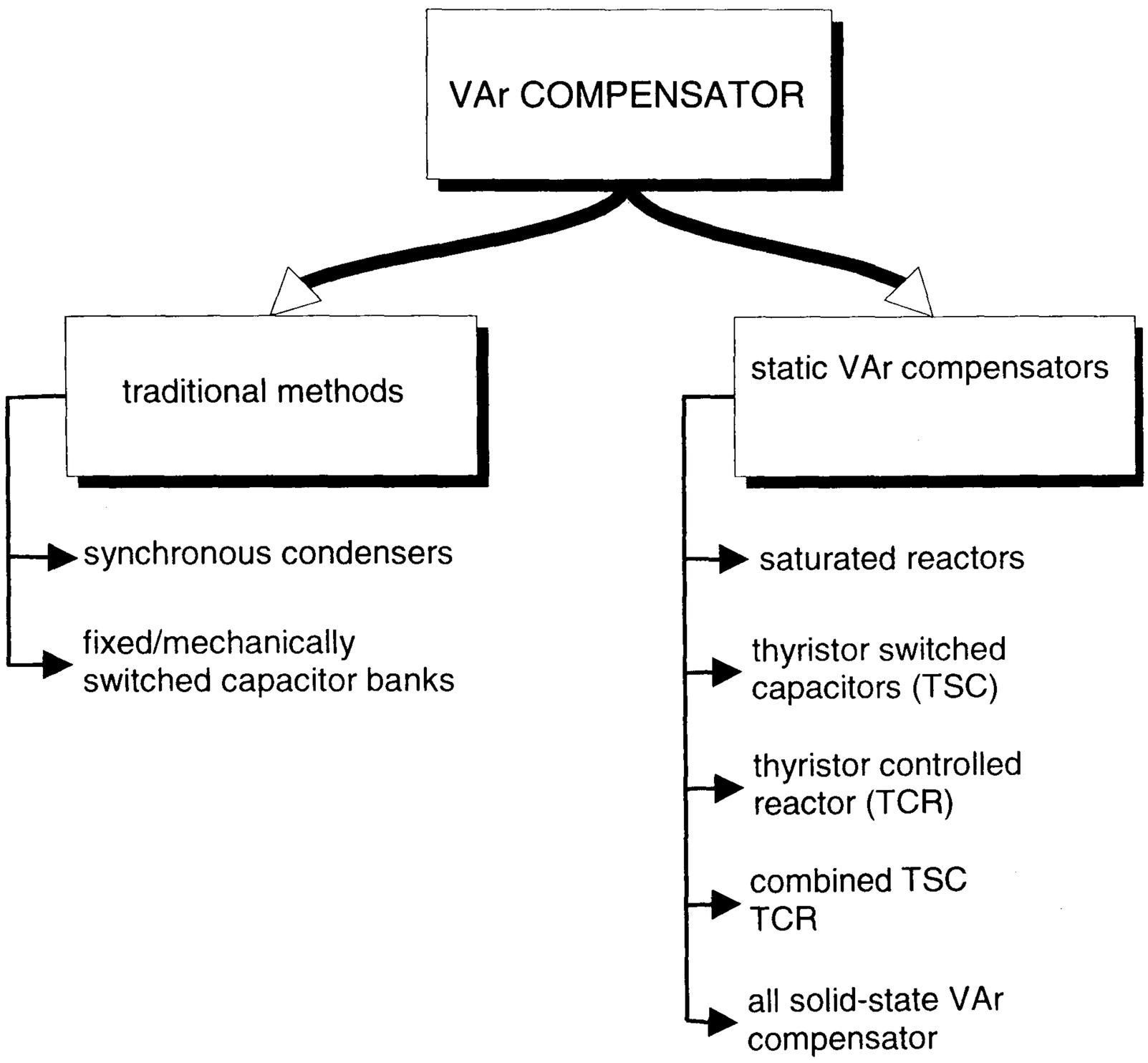
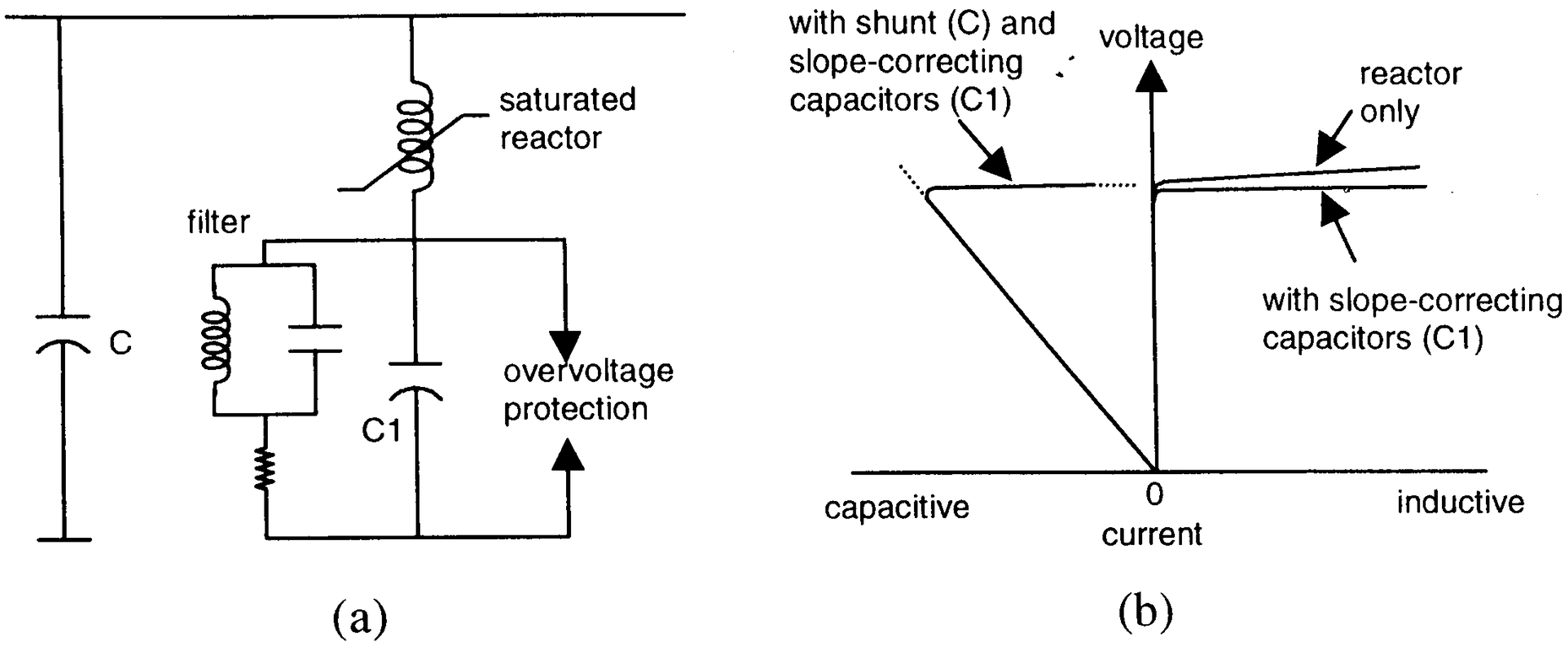


Figure 1.1 Different types of VAr compensators

Figure 1.3 (a) shows the basic circuit for a thyristor controlled reactor (TCR). A TCR is a shunt connected device which consists of a fixed reactor of inductance L and a bidirectional thyristor (SW). The variable reactive (lagging) current can be produced by

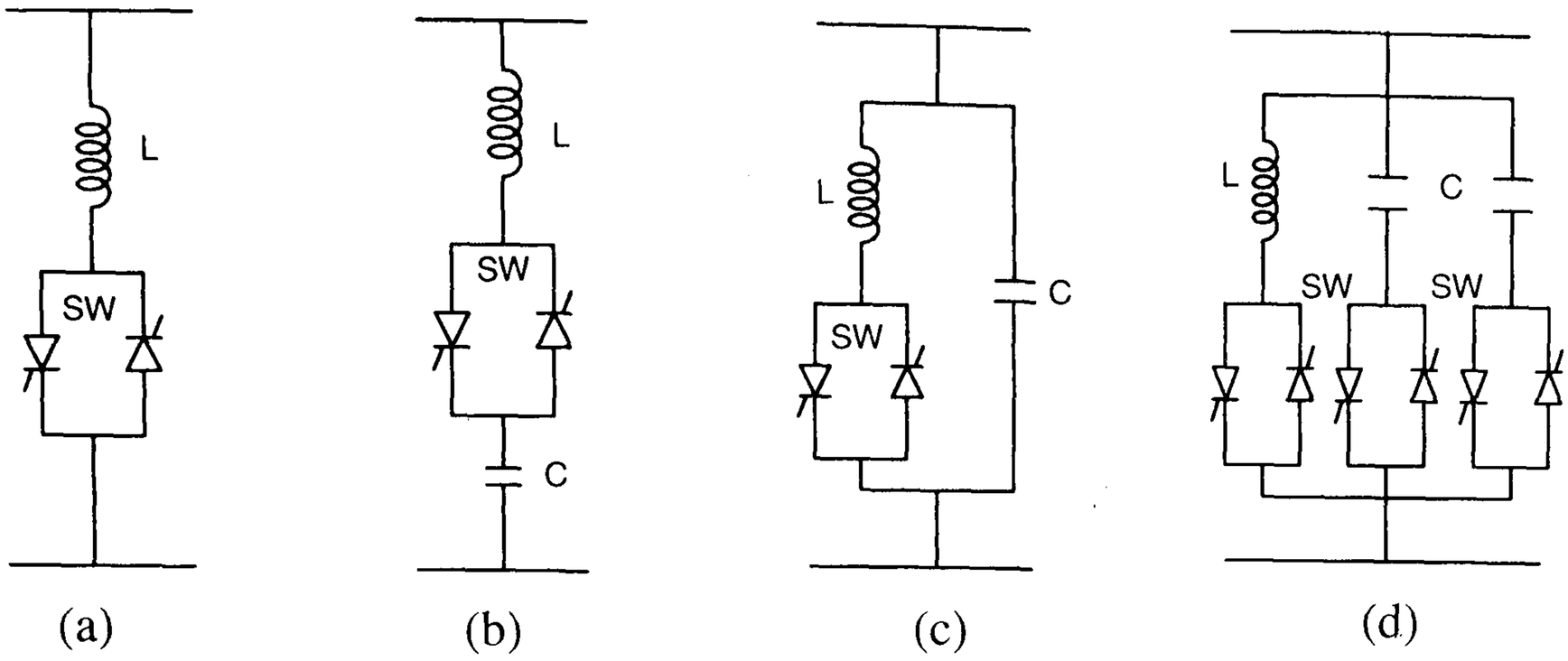
controlling the firing angle of the thyristors. Figure 1.3 (c) shows an arrangement where the compensator also includes a fixed capacitor. The fixed capacitor in practice is usually substituted by a filter network which will generate the reactive power required as well as absorbing the dominant harmonics produced by the TCR.



(a) (b)

Figure 1.2 Saturated reactor compensator (a) basic circuit (b) operating characteristic

A basic thyristor switched capacitor (TSC) circuit is shown in Figure 1.3 (b). It has a series damping reactor connected to limit the rate of rise of current through the thyristors and also to prevent resonance with the system network. A capacitor bank is split up into appropriately small steps which will be individually switched in or out resulting in a stepwise control of reactive power. In a practical compensator, dynamic compensation of both capacitive and inductive reactive power is required. This can be done by combining different types of compensators, for example by combining TSC and TCR as shown in Figure 1.3 (d). For a given reactive power to be compensated, the required number of TSC branches are switched in (with a positive surplus). TCR switching is then controlled to cancel the surplus capacitive current.



(a) (b) (c) (d)

Figure 1.3 Basic circuit configuration for (a) TCR (b) TSC (c) fixed capacitor with TCR (d) combination of TSC and TCR

In the different SVCs mentioned earlier, thyristors are used as a control element, to vary the reactive power generated by the reactor and capacitor banks. Each component of a SVC (reactor, capacitor, thyristor) is subjected to a similar VA rating hence contributing significantly to the size and cost of the equipment. Another option to obtain VAr generation and control without requiring large banks of reactors and capacitors is by using a solid-state compensator. A number of possible approaches have been described in [1.5]. Part of this thesis will focus on this type of compensator. In the following discussions the term SVC will refer to a solid-state type of VAr compensator.

1.3 POWER QUALITY AND ACTIVE POWER FILTER

Power quality can be defined as the relative absence of utility related voltage variations particularly the absence of outages, sags, surges and harmonics, as measured at the point of services [1.6]. A lot of attention has been given to this issue, mainly because it has economic impacts on the utilities, the customers and also the suppliers of load

equipment. Much development and research has been done in the area of electrical power quality as reviewed in reference [1.7].

One of the most significant power quality problems is harmonic distortion. Harmonics are sinusoidal voltages or currents having frequencies that are integer multiples of the frequency at which the supply system is designed to operate (fundamental frequency). Distorted waveforms can be decomposed into a sum of the fundamental frequency and the harmonics. Harmonic distortion originates from the nonlinear characteristics of devices and loads on the power system. Power converters, specifically, are responsible for a disproportionate amount of the harmonics trouble with power systems today. Arc furnaces are another significant source of harmonics. Some of the undesirable effects of harmonic distortion on electrical equipment include [1.8] :

i. heating effects in transformers, capacitors and electrical machines

ii. failures and disturbances in sensitive electronic equipment

iii. telephone interference

iv. performance deterioration in circuit breakers and protection circuits

v. torque pulsations and vibration in electrical machines

IEEE Standard 519-1992 [1.9] provides guidelines for harmonic current and voltage distortion levels on distribution and transmission circuits.

When problems occur, the basic corrective measures for controlling harmonics are :

i. locate the source and reduce the harmonic currents in the load

ii. use filters or power conditioning equipment to absorb the harmonic and buffer the sensitive load from the disturbances.

There are two general types of filters, viz., passive filters and active power filters. Passive filters are made of inductors, capacitors and resistors. They are relatively inexpensive compared to other means of eliminating harmonic distortion. Passive filters are designed to shunt the harmonic currents from the line or to block their flow between parts of the system by tuning the elements to resonate at a selected harmonic frequency. However passive filters may cause resonance problems if they are not designed carefully.

Active power filters, the subject of the second part of this thesis, have the distinct advantage that they do not resonate with the system. They have the ability to control the output current according to the harmonic current variation. Active power filters can effectively absorb harmonic currents with variable frequencies. They can also be used for power factor correction as well as harmonic compensation.

1.4 REFERENCES

[1.1] W. D. Stevenson, "Elements of Power Systems Analysis", Fourth Edition, McGraw Hill 1982.

[1.2] B. M. Weedy, "Electric Power Systems", Third Edition Revised, Wiley 1992.

[1.3] L. Gyugyi, "Power electronics in electric utilities: static var compensators", Proc. of the IEEE, Vol. 76, No. 4, April 1988, pp. 483-493.

[1.4] S. K. Lowe, "Static var compensators and their applications in Australia", Power Engineering Journal, September 1989, pp. 247-256.

[1.5] L. Gyugyi, "Reactive power generation and control by thyristor circuits", IEEE Trans. on Industry Applications, Vol. IA-15, No. 5, Sep/Oct. 1979, pp. 521-531.

[1.6] J. J. Burke, D. C. Griffith and D. J. Ward, "Power quality - two different perspectives", IEEE Trans. on Power Delivery, Vol. 5 No. 3, July 1990, pp. 1501-1513.

[1.7] J. D. Van Wyk, "Power quality, power electronics and control", EPE 93 Brighton, pp. 17-32.

[1.8] IEEE Task Force on the Effects of Harmonics on Equipments, "Effects of Harmonics on Equipment", IEEE Trans. on Power Delivery, Vol. 8, No.2, April 1993, pp. 672-680.

[1.9] ANSI/IEEE Standard 519-1992, IEEE Recommended Practices and Requirements for Harmonic Control in Electrical Power Systems.

CHAPTER 2

BACKGROUND THEORY AND LITERATURE REVIEW

2.1 INTRODUCTION

This chapter will explain the background theory needed to understand the operation and analysis of the static VAr (SVC) compensator and active power filter (APF). A brief literature review on the development of the SVC and APF is incorporated into this chapter. As pointed out in chapter one the term SVC will henceforth be used to refer to the solid state type of VAr compensator.

2.2 BASIC SVC AND APF CONCEPT

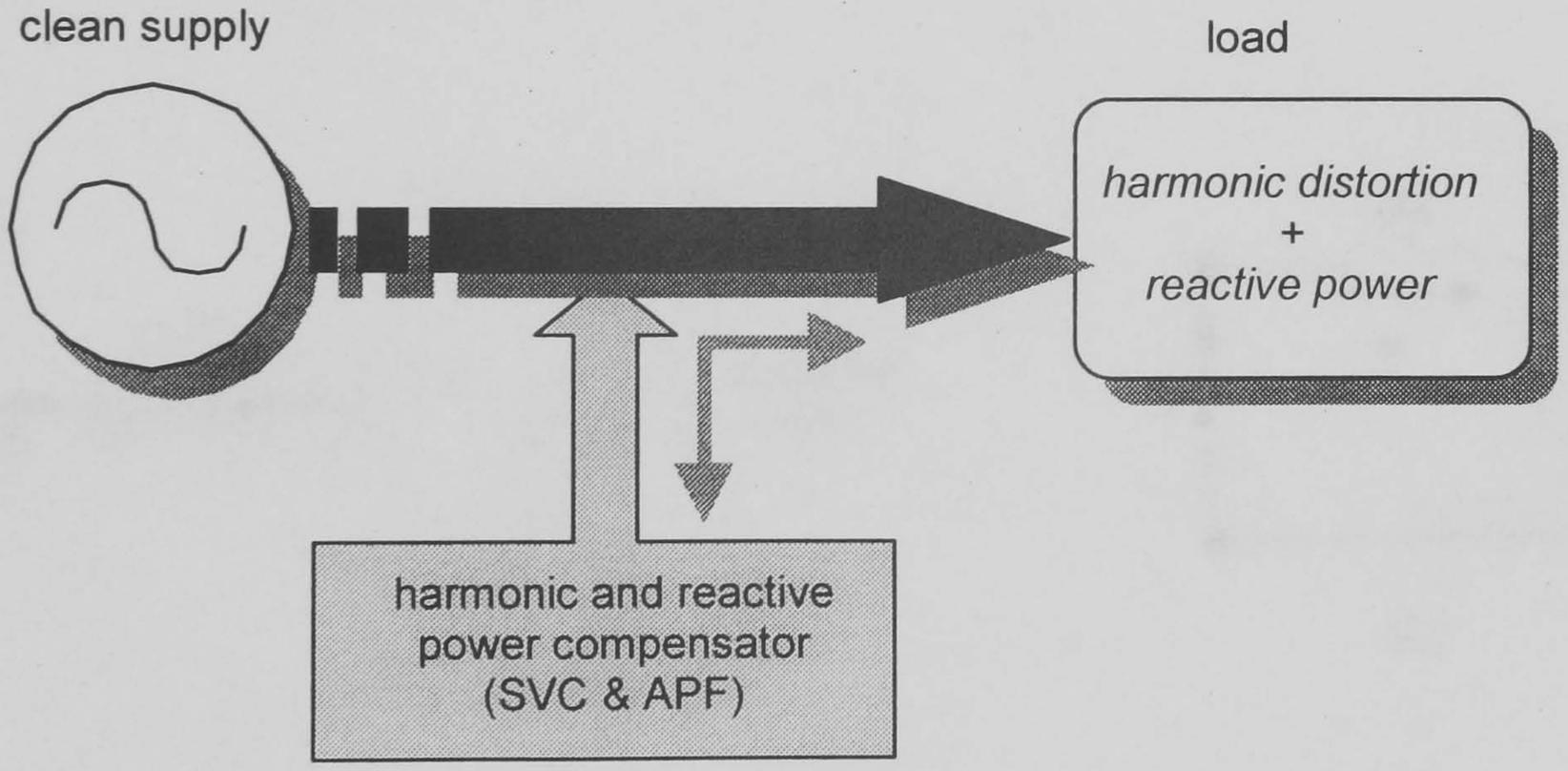


Figure 2.1 Basic function of SVC and APF

The diagram in Figure 2.1 outlines SVC and APF functions. The compensators will maintain a good clean sinusoidal voltage and current waveform on the supply side. Any harmonic distortion will be compensated and prevented from affecting other sensitive loads connected to the same supply system. Reactive power will be supplied by the SVC

whether it is leading or lagging thus improving the power factor. By maintaining a unity power factor, costs and losses will be reduced or minimised.

The basic operation concept of a SVC is similar to a synchronous condenser [2.1]. Figure 2.2 shows the basic diagram of a synchronous condenser. With a synchronous condenser, the reactive power is controlled by controlling the excitation. When the condenser is overexcited, this will increase the magnitude of E above V, causing a leading VAr to be drawn from the supply. If the condenser is underexcited, E will decrease to a value less than V hence producing a lagging VAr. The phasor diagram for both modes of operation (i.e. overexcitation and underexcitation) are shown in Figure 2.2 (c) and (d). For simplicity the stator resistance is assumed negligible. However in reality, a small amount of real power will flow from the AC supply to the machine due to mechanical and electrical losses.

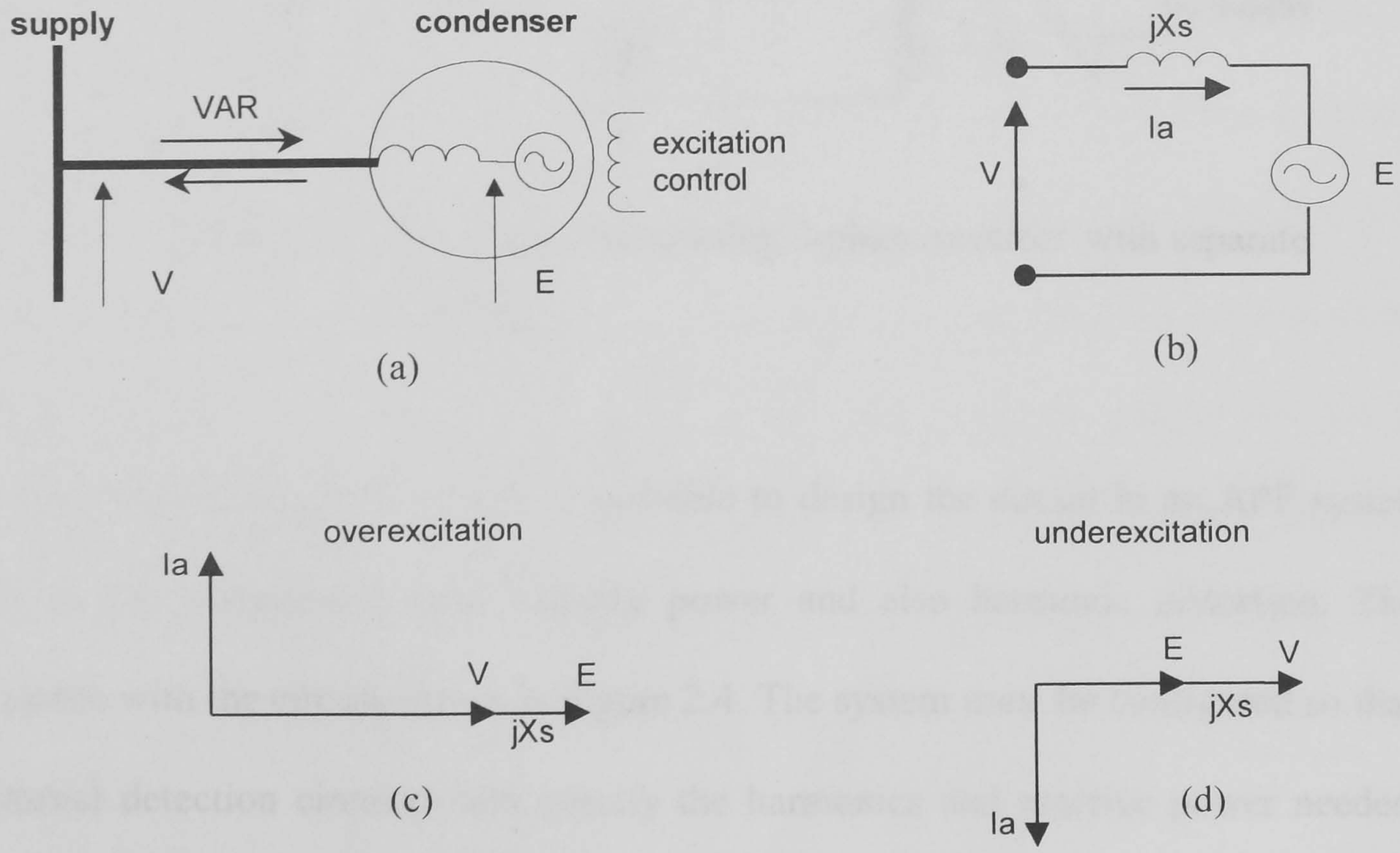


Figure 2.2 Using synchronous condenser as a VAr compensator. (a) basic circuit diagram, (b) simple equivalent circuit, (c) phasor diagram for leading VAr compensation, and (d) phasor diagram for lagging VAr compensation

Similarly, using the same concept, the VAr compensation performed by the condenser can be achieved with a power electronic inverter. The main requirement is for the inverter to have a variable output voltage to produce leading or lagging VArs. This can be achieved by controlling the switching (hence modulation depth) of the inverter. Figure 2.3 shows the possible setup of such a system. A constant DC supply is connected to the three phase inverter with the output connected to the supply system through inductors. A control circuit is designed to maintain the appropriate inverter output voltage magnitude hence giving the required reactive power compensation. In an ideal compensator, no real power will flow from the AC supply, therefore a separate DC supply is needed. However such a system can be improved by using a suitable DC reservoir capacitor as shown in Figure 2.4. By controlling the inverter switching real power flow is used to charge the capacitor to the DC voltage level required.

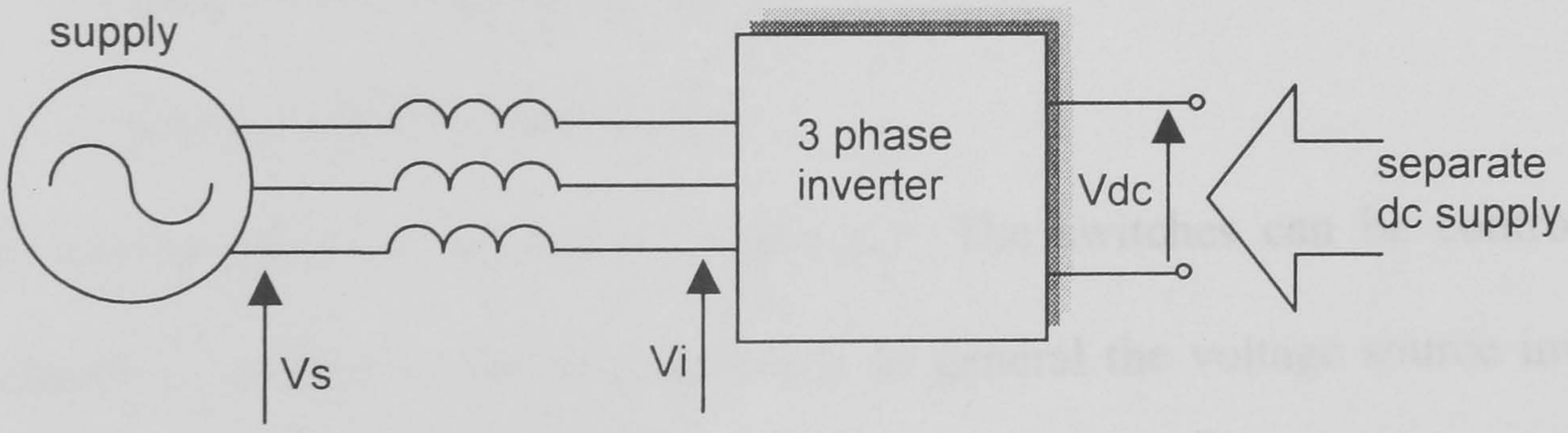


Figure 2.3 VAr compensation using 3-phase inverter with separate DC supply

Using a similar configuration, it is possible to design the circuit in an APF system so that it can compensate both reactive power and also harmonic distortion. This is possible with the circuit shown in Figure 2.4. The system must be configured so that the external detection circuitry will specify the harmonics and reactive power needed for compensation. Combined with the appropriate feedback control, the necessary switching control action for the inverter can be determined so as to perform the required compensation.

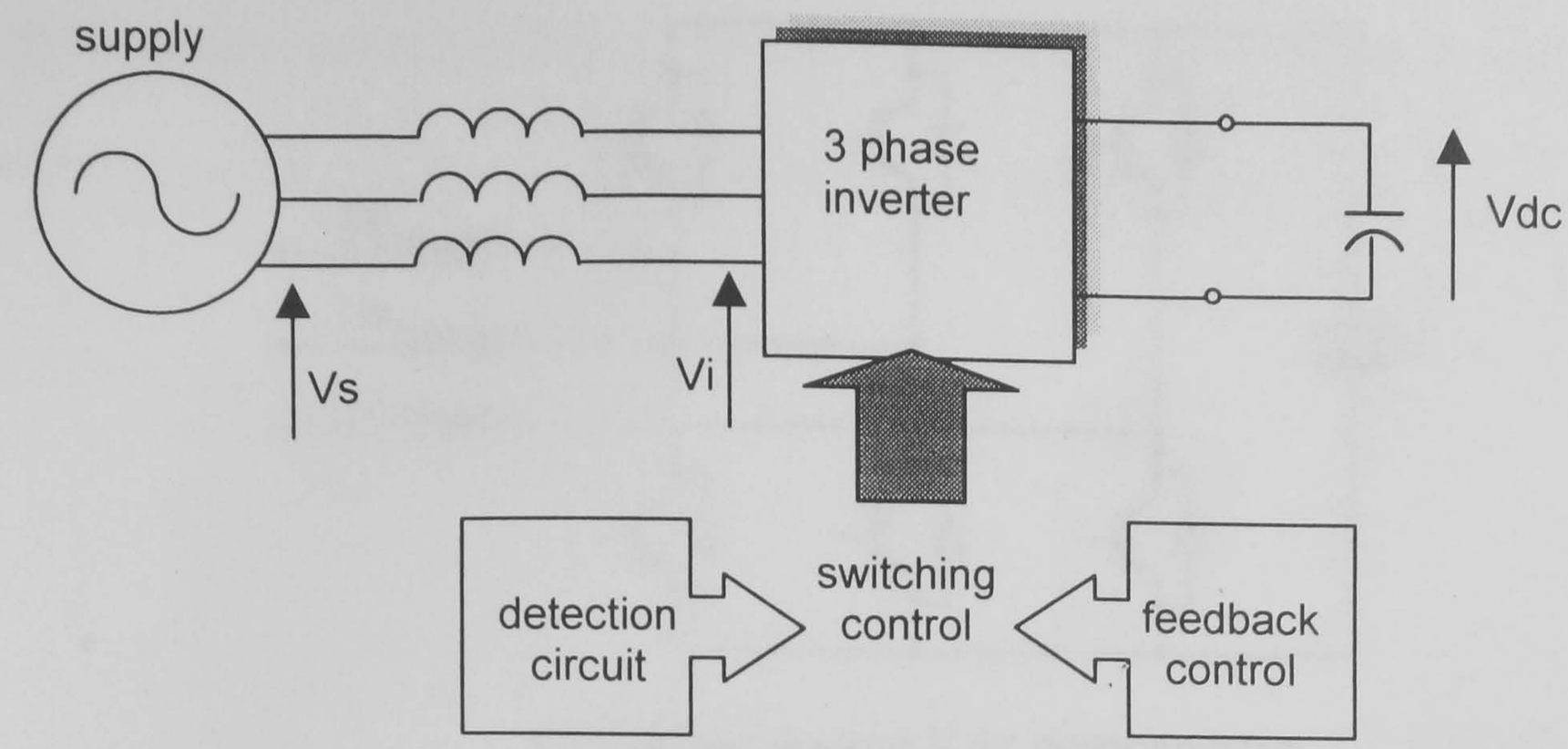


Figure 2.4 Basic diagram for SVC and APF

2.3 CIRCUIT CONFIGURATIONS

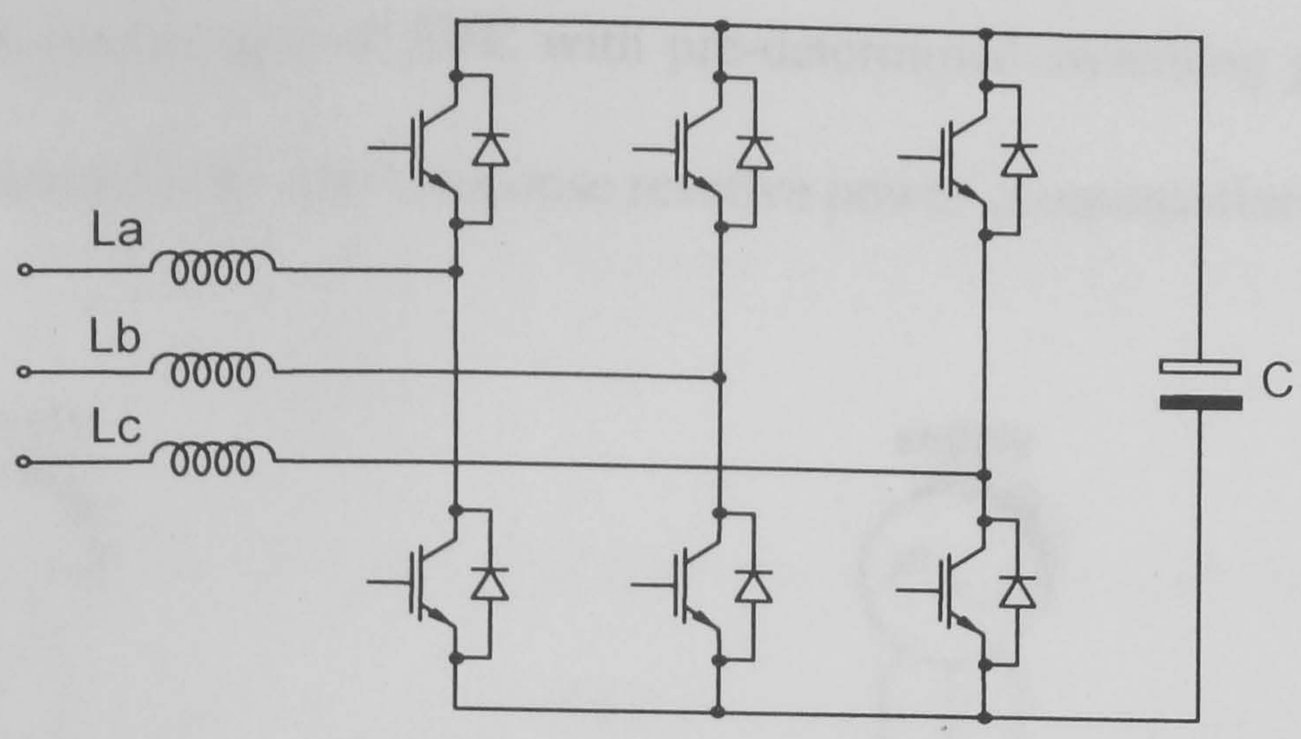
The circuits for SVC and APF are similar but they differ in terms of control action and detection techniques. There are two main types of inverter circuits that can be used:

i) voltage source inverter (VSI)

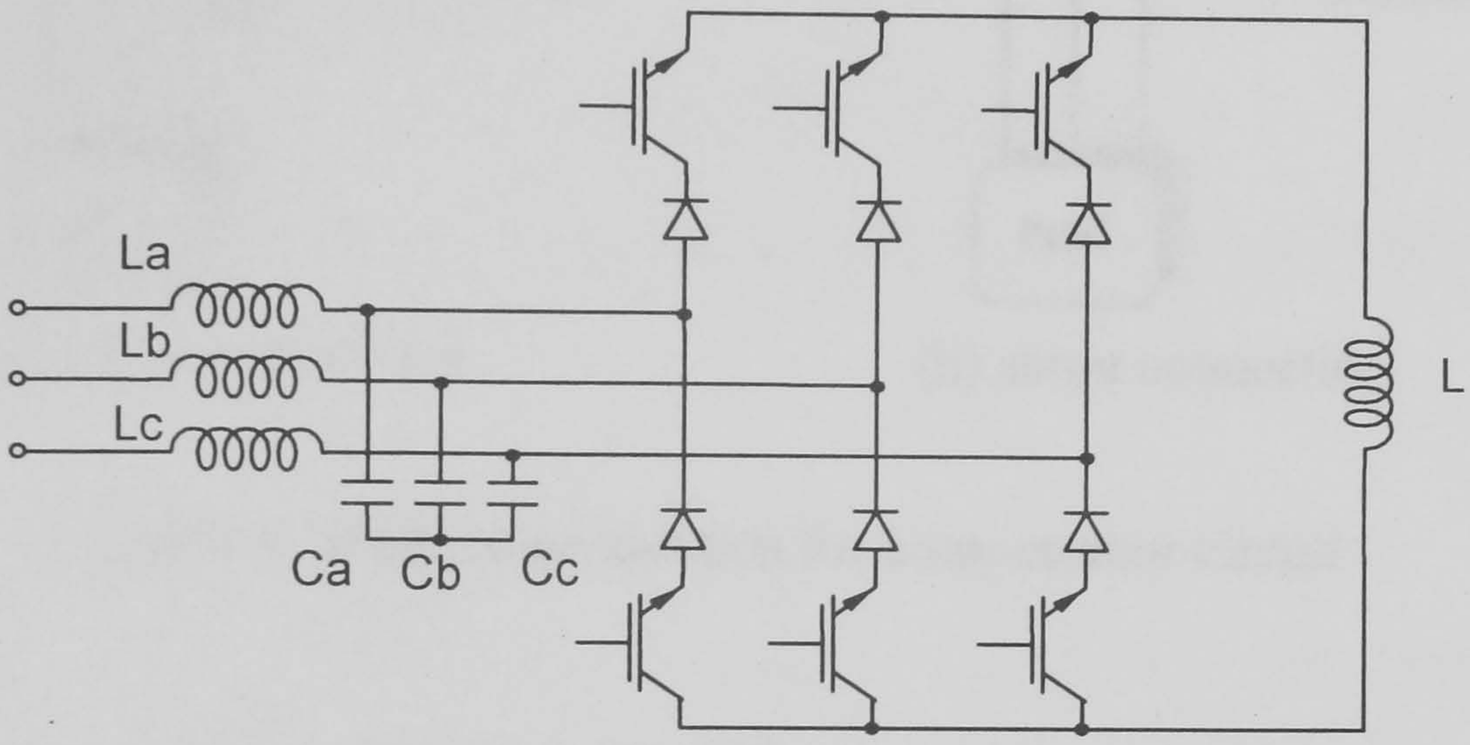
ii) current source inverter (CSI)

Both circuit types are shown in Figure 2.5. The switches can be controlled using pulsewidth modulation (PWM) techniques. In general the voltage source inverter type has advantages over the current source inverter. Inductors are placed in series with the output, giving an output current which is inherently continuous and of low switching content [2.2]. In contrast, the current source inverter draws pulsewidth modulated currents. The AC side filter can cause resonance problems if it is not designed carefully [2.3, 2.4]. The voltage source PWM inverter is higher in efficiency and lower in initial cost, compare to the current source PWM inverter [2.5].

The majority of SVCs and APFs are implemented using a voltage source inverter [2.6-26]. Several compensators employing a current source inverter have been reported in the



(a) voltage source VAr compensator

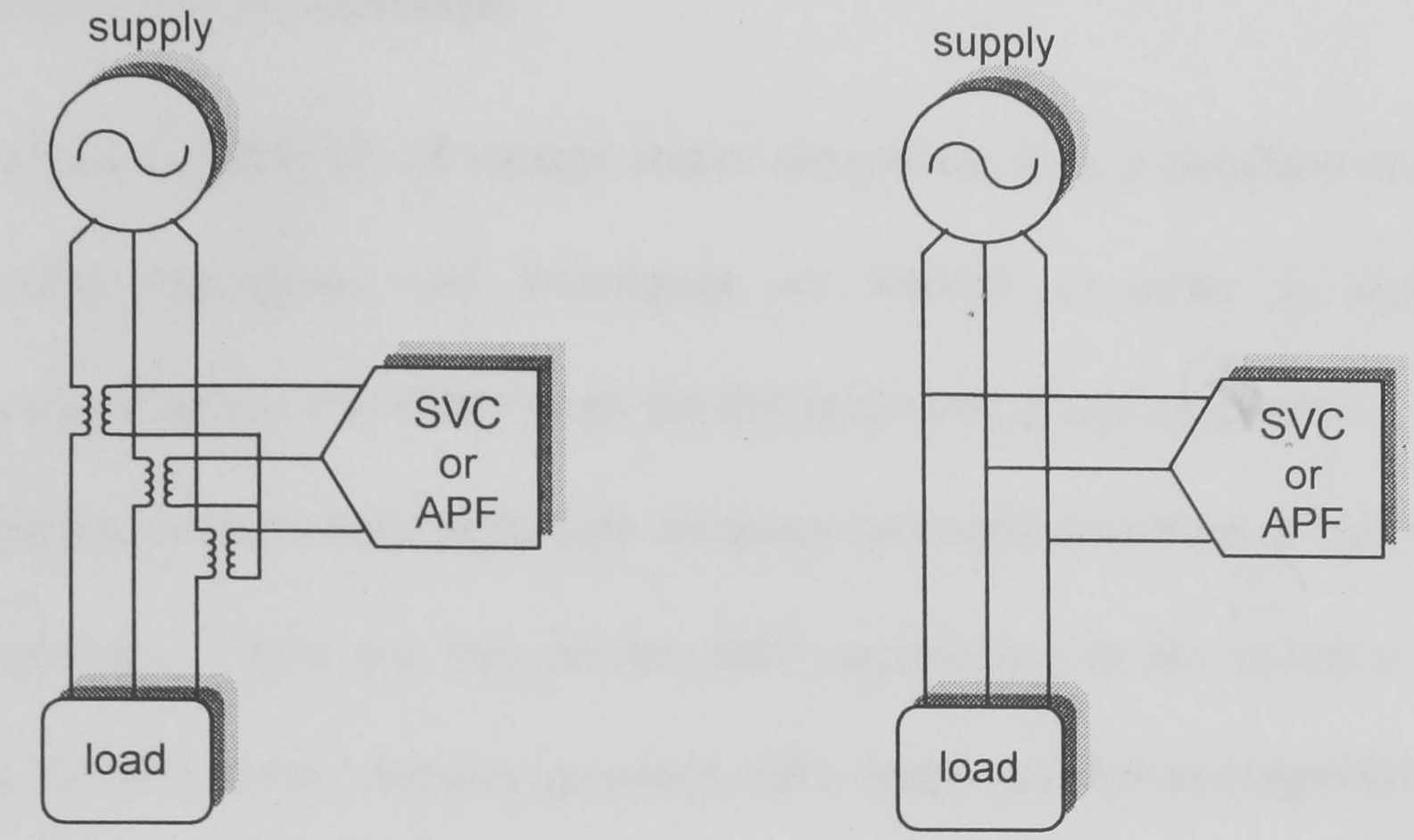


(b) current source VAr compensator

Figure 2.5 Different circuit configuration for VAr compensator

literature [2.26-32]. It has been suggested that the selection of the type of inverter circuit depends on the application. For high power utility applications, to achieve low switching losses and low harmonic distortion, it has been reported that the CSI is more suitable than the VSI [2.30]. High power GTO switches are used and switched at low frequency. Further improvement by increasing the switching frequency is not possible due to device limitations. The VSI is not suitable for switching at low frequency as it will generate unacceptable harmonic components. Both VSI and CSI are considered for the proposed compensator in [2.30]. The VSI circuit is used to provide harmonic compensation while the CSI circuit handles larger reactive power compensation. In

[2.29] a current source type of SVC with pre-determined switching patterns stored in EPROM is used to provide slow response reactive power compensation.



(a) series connection (b) shunt connection

Figure 2.6 Mode of connection for compensator circuit

The inverter circuits can be connected to the electrical system either in a shunt or series connection. These two connection modes are shown in Figure 2.6. The use of a series connected SVC has been reported in the literature [2.17, 2.31, 2.33]. The compensator is connected to the system through a three phase transformer which gives a practical advantage that the inverter can have a lower voltage rating. A series APF is normally used in combination with a shunt passive filter. Its function is not to directly compensate for the harmonics of the load, but rather to improve the filtering characteristics of the shunt passive filters. It acts not as a harmonic compensator but as a harmonic isolator between the supply system and the load [2.34-36]. A multi-level type of inverter has also been tested. The advantages of this type of configuration are :

i. to further suppress harmonics without increasing the switching frequency [2.7].

ii. to reduce the size of reactive components for a given power rating [2.37].

2.4 DETECTION METHOD

In order for a compensator system to function as required, the following fundamental questions must first be addressed:

i. which components of voltage and/or current are to be sensed and detected

ii. what procedures and techniques are needed in order to determine the compensating waveform from the detected voltage and/or current.

The measuring systems must yield high accuracy and rapid detection of harmonic and reactive currents. There are two fundamental approaches to the solution of these questions, namely, a time domain approach and a frequency domain approach.

2.4.1 Time Domain Approach

Correction in the time-domain is based on the principle of maintaining the instantaneous voltage or current within a reasonable tolerance of a sine wave. An instantaneous error function is computed on-line. This can be done simply by finding the difference between actual and reference waveforms or by using a more elaborate function such as given by instantaneous reactive power theory (IRPT) [2.38]. This theory is explained further in section 2.5. Another technique which evolved around the same theory has been suggested by Furuhashi et al. [2.39]. This technique is said to give a better physical meaning and contains both the instantaneous reactive power as well as the zero-phase component. Luo and Huo [2.40] proposed an adaptive closed loop detecting method based on adaptive interference cancelling theory. The system is maintained in the best operating state by continuously self-studying and self-adjusting. This adaptive technique provides the system with operating characteristics which are almost independent of parameter variations.

The main advantage of the time domain approach is its fast response to power system changes. It is also easy to implement and has little computational burden compare to the frequency domain approach.

2.4.2 Frequency Domain Approach

A compensation technique using the frequency domain approach is based on the principle of Fourier analysis and periodicity of the distorted voltage or current. The error signal from a measured waveform is extracted using a filter circuit. A Fourier transform is applied to the error signal. The compensating switching function is then constructed by solving a set of non-linear equations to determine the precise switching times and the magnitude of the correcting signal. The nonlinear equations are usually linearized about an operating point [2.41-2.42].

The frequency domain approach depends on the periodic characteristics of the distortion. For example, voltage or current distortion produced by adjustable speed motor drives tends to be periodic. The highest harmonic to be eliminated can be selected beforehand using theoretical limits that are a function of device switching frequency. The main disadvantage of the frequency domain approach lies in its high computational requirements.

2.5 INSTANTANEOUS REACTIVE POWER THEORY

The main requirement for a compensator working in the time domain is the ability to calculate instantaneous reactive power. A widely used technique has been developed by Akagi et al. [2.38] where the instantaneous power uses the instantaneous value concept

for arbitrary voltage and current waveforms, including transient states. The three phase voltages and currents are transformed into orthogonal α-β co-ordinates according to :

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{2.2}$$

Figure 2.7 shows schematically the instantaneous space vectors. From (1.1) and (1.2), the instantaneous real power p_L and the instantaneous reactive power q_L flowing into the load side are expressed by :

$$\begin{bmatrix} p_L \\ q_L \end{bmatrix} = \begin{bmatrix} v_\alpha & v_\beta \\ -v_\beta & v_\alpha \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{2.3}$$

The instantaneous real and imaginary powers can each be decomposed into an average component and an alternating component.

$$p_L = \bar{p}_L + \tilde{p}_L \tag{2.4}$$

$$q_L = \bar{q}_L + \tilde{q}_L \tag{2.5}$$

The average components, $\bar{p}_L$ and $\bar{q}_L$, correspond to the conventional active and reactive components. The alternating components, $\tilde{p}_L$ and $\tilde{q}_L$, are caused by load unbalance and harmonics. Compensation can be done by providing $\tilde{p}_L$, $\bar{q}_L$ and $\tilde{q}_L$ locally while the average power component, $\bar{p}_L$, comes from the source.

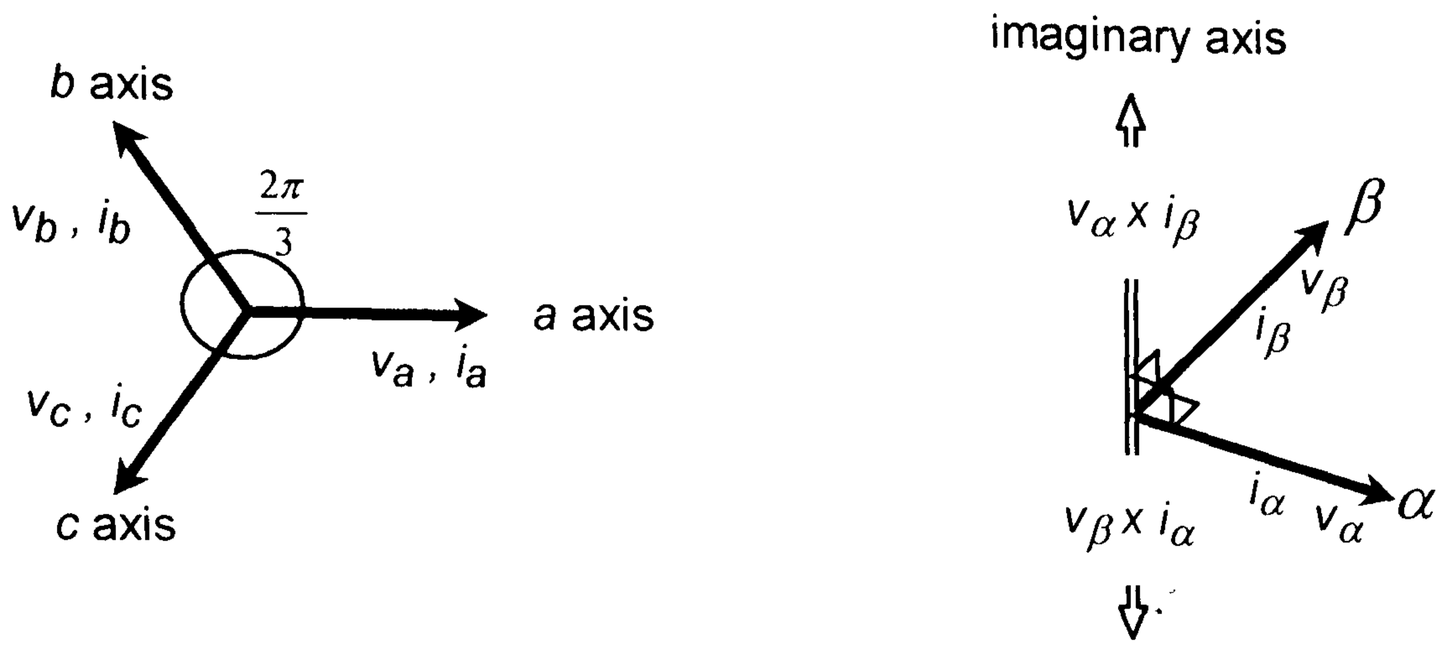


Figure 2.7 Instantaneous space vectors

2.6 CONTROL METHODS

As mentioned earlier, SVC and APF circuit configurations differ in their control technique. The control strategies for each compensator will be explained separately.

2.6.1 Control Strategy for Solid-state Static VAr Compensator

The control circuit for a SVC must perform two basic functions :

i) maintain the capacitor voltage at the required level

ii) provide the required leading or lagging VAr.

Moran et al. suggested a phase shift control system in which the modulation index of the PWM switching pattern is controlled directly by the VAr calculator [2.10]. The scheme allows the use of optimised PWM patterns for harmonic rejection and is suitable for slowly varying VAr demand applications. The output from the VAr calculator is connected to a voltage control oscillator (VCO) using a PI controller. The gating signal synchronism is simplified by the use of the VCO. An alternative topology that requires no VAr sensing is proposed in [2.12]. The modulation index is fixed and the capacitor voltage is controlled so that the inverter output voltage matches the magnitude of the

line voltage. This ensures unity power factor without the need for VAr sensing. Capacitor voltage control is achieved through phase-shifting of the inverter voltage with respect to the line voltage. This type of load angle control scheme does not allow a fast response to be achieved, particularly at high power ratings. The same author proposed a current control system which gives a fast response at a constant switching frequency [2.15]. The required reactive power compensation is achieved by controlling the amplitude and the sign of the inverter current reference waveform. This reference current is compared with the generated current. The error signal is then used to provide the necessary switching action.

In a simple PI control method, phase angle is controlled to keep a constant DC voltage across the capacitor and the modulation index is controlled to change the output reactive power. The interaction between these two loops prevents the PI controller from having a quick response. For this reason, a H infinity controller has been proposed to decouple the two control loops thus giving faster responses and superior robustness over PI type controllers [2.43].

2.6.2 Control Strategy for Active Power Filter

Once the reference waveform has been produced by the detection circuit, a control scheme must be designed to generate the switching action for the inverter. There are two basic schemes for current control [2.7]. The first is to determine the PWM switching sequence by means of comparing the current error signal with a triangular carrier signal. The switching frequency is equal to the frequency of the triangular carrier signal. The second scheme uses hysteresis control. A hysteresis band is imposed around the reference current. Whenever the actual current tries to leave the

band, the appropriate device is switched on or off, forcing the current to remain within the band. Another approach that gives similar quick control is to compare directly the reference current with the actual current using discrete delta modulation technique [2.47]. The signal is sampled and compared at regular intervals.

A simpler control technique has been proposed where the reactive current is not sensed and calculated directly [2.30]. Separate voltage control is required to keep the DC voltage across the capacitor constant. The distorted load current is sensed and filtered to extract the fundamental component with 180 degree phase difference. By adding the extracted fundamental component to the load current, the reference current waveform required to compensate harmonic distortion is obtained.

A current control technique using a vector control approach is proposed in [2.24]. The current reference signals are obtained using instantaneous reactive power theory. The switching region is divided into six sub regions. A different switching mode is assigned to each region. By applying certain equations, the region where the current vector error is located can be identified to give the required switching signals. A predictive current control technique has also been proposed to improve control performance [2.25]. The method requires considerable calculation hence a digital signal processor (DSP) is used. However there is a possibility of error due to the sampling time delay. A method to compensate this error is proposed in the paper. Active power filter control based on the sliding mode control strategy has also been proposed [2.44-46]. This will be examined in Chapter 6 in detail.

2.7 REFERENCES

[2.1] L. Gyugyi, "Reactive power generation and control by thyristor circuits", IEEE Trans. on Industry Applications, Vol. IA-15, No. 5, Sep/Oct. 1979, pp. 521-531.

[2.2] T. C. Green, "The impact of EMC regulations on mains-connected power converters", Power Engineering Journal, February 1994, pp. 35-43.

[2.3] F. Jenni, "The high frequency PWM current-source inverter: Its inherent problems and their remedies", Conf. Proc. Power Electronics and Variable Drives (PEVD) 1990, pp. 162-167.

[2.4] X. Wang and B. T. Ooi, "Real-time multi-dsp control of three phase current-source unity power factor pwm rectifier", Conf. Proc PESC '92 pp. 1376-1383.

[2.5] H. Akagi, "Trends in active power line conditioners", IEEE Trans. on Power Electronics, Vol. 9, No. 3, May 1994, pp. 263-268.

[2.6] Y. Sumi, Y. Harumoto, T. Hasegawa, M. Yano, K. Ikeda and T. Matsuura, "New static var control using force-commutated inverters", IEEE Trans. On Power Apparatus and Systems, Vol. PAS-100, No. 9 September 1981, pp. 4216-4224.

[2.7] H. Akagi, A. Nabae and S. Atoh, "Control strategy of active power filters using multiple voltage-source PWM converters", IEEE Trans. on Industry Applications, Vol IA-22, No. 3, May/June 1986, pp. 460-465.

[2.8] M. Takeda, K. Ikeda, A. Teramoto and T. Aritsuka, "Harmonic current and reactive power compensation with an active filter", Conf. Proc. Power Electronics Specialist Conference (PESC) 1988, pp. 1174-1179.

[2.9] L. T. Moran, P. D. Ziogas and G. Joos, "Analysis and design of a novel 3 phase solid-state power factor compensator and harmonic suppressor system", IEEE Trans. on Industry Applications, Vol. 25 No. 4, July/August 1989, pp.609-619.

[2.10] L. T. Moran, P. D. Ziogas and G. Joos, "Analysis and design of a three-phase synchronous solid-state VAR compensator", IEEE Trans. on Industry Applications, Vol. 25, No. 4, July/August 1989, pp. 598-608.

[2.11] F. Peng, H. Akagi and A. Nabae, "A study of active power filters using quad-series voltage-source PWM converters for harmonic compensation", IEEE Trans. on Power Electronics, Vol. 5, No. 1, January 1990, pp. 9-15.

[2.12] G. Joos, L. Moran and P. Ziogas, "Performance analysis of a PWM inverter VAR compensator", IEEE Trans. on Power Electronics, Vol. 6, No. 3, July 1991, pp. 380-390.

[2.13] H. A. Kojori, S. B. Dewan and J. D. Lavers, "A large -scale PWM solid-state synchronous condenser", IEEE Trans. Industry Applications, Vol. 28, No.1 Jan./Feb. 92, pp. 41-49.

[2.14] S. Mori, K. Matsuno, M. Takeda and M. Seto, "Development of a large static var generator using self-commutated inverters for improving power system stability", IEEE Trans. on Power Systems, Vol. 8, No. 1, February 1993, pp. 371-377.

[2.15] L. Moran, P. D. Ziogas and G. Joos, "A solid-state high performance reactive power compensator", IEEE Trans. on Industry Applications, Vol. 29, No. 5, September/October 1993, pp. 969-977.

[2.16] D. R. Trainer, S. B. Tennakon and R. E. Morrison, "Analysis of GTO-based static var compensators", IEE Proc. Electr. Power Appl., Vol. 141, No. 6, November 1994, pp. 293-302.

[2.17] A. Campos, G. Joos, P. D. Ziogas and J. F. Lindsay, "Analysis and design of a series voltage unbalance compensator based on a three-phase VSI operating with unbalanced switching functions", IEEE Trans. on Power Electronics, Vol. 9, No. 3 May 1994, pp. 269-274.

[2.18] J. B. Ekanayake, N. Jenkins and C. B. Cooper, "Experimental investigation of an advanced static var compensator", IEE Proc. Gener. Trnsm. Distrib., Vol. 142, No. 2, March 1995, pp. 202-210.

[2.19] M. Aredes and E. H. Watanabe, "New control algorithms for series and shunt three phase four-wire active power filters", IEEE Trans. on Power Delivery, Vol. 10, No. 3, July 1995, pp. 1649-1656.

[2.20] L. A. Moran, J. W. Dixon and R. R. Wallace, "A three-phase active power filter operating with fixed switching frequency for reactive power and current harmonic compensation", IEEE Trans. on Industrial Electronics, Vol. 42, No. 4, August 1995, pp. 402-408.

[2.21] L. Moran, E. Mora, R. Wallace and J. Dixon, "Line conditioning system with simple control strategy and fast dynamic response", IEE Proc. Gen. Transm. Distrib., Vol. 142, No.2 March 1995, pp. 128-134.

[2.22] J. W. Dixon, J. J. Garcia and L. Moran, "Control system for three phase active power filter which simultaneously compensates power factor and unbalanced loads", IEEE Trans. on Industrial Electronics, Vol. 42, No. 6. December 1995, pp. 636-641.

[2.23] V. B. Bhavaraju and P. N. Enjeti, "An active line confitioner to balance voltages in a three-phase system", IEEE Trans. on Industry Applications, Vol. 32, No. 2, March/April 1996, pp. 287-291.

[2.24] L. A. Moran, L. Fernandez, J.W. Dixon and R. Wallace, "A simple and low-cost control strategy for active power filters connected in cascade", IEEE Trans. on Industrial Electronics, Vol. 44, No. 5, October 1997, pp. 621-629.

[2.25] S. G. Jeong and M. H. Woo, "DSP-based active power filter with predictive current control", IEEE Trans. on Industrial Electronics, Vol. 44, No. 3, June 1997, pp. 329-336.

[2.26] P. Verdelho and G. D. Marques, "An active power filter and unbalanced current compensator", IEEE Trans. on Industrial Electronics, Vol. 44, No. 3, June 1997, pp. 321-328.

[2.27] L. H. Walker, "Force-commutated reactive-power compensator", IEEE Trans. on Industry Applications Vol. IA-22, No. 6, November/December 1986, pp. 1091-1104.

[2.28] L. T. Moran, P. D. Ziogas and G. Joos, "Analysis and design of a three-phase current source solid-state var compensator", IEEE Trans. on Industry Applications, Vol. 25, No.2 , Narch/April 1989, pp. 356-365.

[2.29] Y. Tang and L. Xu, "A new converter topology for advanced static var compensation in high power applications", Conf. Proc. Industrial Applications Society (IAS) 1993, pp. 947-953.

[2.30] S. Fukuda and T. Endoh, "Control method and characteristics of active power filters", Conf. Proc. European Power Electronics Association Conference (EPE) 1993, pp. 139-144.

[2.31] G. Joos and J. Espinoza, "Three phase series var compensation based on a voltage controlled current source inverter with supplemental modulation index control", Conf. Proc. IEEE PESC 1994, pp. 1437-1442.

[2.32] Y. Sato, T. Sugita and Teruo Kataoka, "A new control method for current source active power filters", Conf. Proc. Industrial Application Society Conf. (IAS) 1997, Vol. 2, pp. 1463-1470.

[2.33] B. T. Ooi and S. Z. Dai, "Series type solid-state static var compensator", IEEE Trans. on Power Electronics, Vol. 8, No. 2, pp. 164-169, April 1993.

[2.34] H. Fujita and H. Akagi, "A practical approach to harmonic compensation in power systems - series connection of passive and active filters", IEEE Trans. on Industry Applications Vo. 27, No. 6, Nov/Dec 1991, pp. 1020-1025.

[2.35] J. W. Dixon, G. Venegas and L. A. Moran, "A series active power filter based on a sinusoidal current-controlled voltage-source inverter", IEEE Trans. on Industrial Electronics, Vol. 44, No. 5, October 1997, pp. 612-620.

[2.36] H. Akagi and H. Fujita, "A new power line conditioner for harmonic compensation in power systems", IEEE Trans. on Power Delivery, Vol. 10 No. 3, July 1995, pp.1570-1575.

[2.37] D. Wuest, H. Stemmler and G. Scheuer, "A comparison of different circuit configurations for an advanced static var compensator (ASVC)", Conf. Proc. PESC 92, pp. 521-529.

[2.38] H. Akagi, Y. Kanazawa and A. Nabae, "Instantaneous reactive power compensators comprising switching devices without energy storage componnets", IEEE Trans. on Industry Applications, Vol. IA-20, No. 3, May/June 1984, pp. 625-630.

[2.39] T. Furuhashi, S. Okuma and Y. Uchikawa, "Study on instantaneous reactive power", IEEE Trans. on Industrial Electronics, Vol. 37, No. 1, February 1990, pp.86-90.

[2.40] S. Luo and Z. Huo, "An adaptive detecting method for harmonic and reactive currents", IEEE Trans. on Industrial Electronics , Vol. 42, No. 1, February 1995, pp. 85-89.

[2.41] H. S. Patel and R. C. Hoft, "Generalized techniques of harmonic elimination and voltage control in thyristor inverters: Part 1 - harmonic elimination", IEEE Trans. on Industry Applications, Vol. IA 9, No. 3 May/June 1974 pp. 310-317.

[2.42] H. S. Patel and R. C. Hoft, "Generalized techniques of harmonic elimination and voltage control in thyristor inverters: Part 11 - voltage control techniques", IEEE Trans. on Industry Applications, Vol. IA 10, No. 5 Sept/October 1974, pp. 666-673.

[2.43] M. Yano and H. Ikeda, "H infinity controller for the static VAR compensator", Conf. Proc. Power Electronics Specialist Conference (PESC) 1994, pp. 1088-1094.

[2.44] Z. Radulovic and A. Sabanovic, "Active Filter control using a sliding mode approach", Conf. Proc. PESC 94, pp. 177-182.

[2.45] S. Saetieo, R. Devaraj and D. A. Torrey, "The design and implementation of a three-phase active power filter based on sliding mode control", IEEE Trans. on Industry Applications, Vol. 31 No. 5, Spet/Oct 1995, pp. 993-1000.

[2.46] B. Singh, K. Al-Haddad and A. Chandra, "Active power filter with sliding mode control", IEE Proc. Generation and Distribution, Vo. 144 No. 6, Nov. 1997, pp. 564-568.

[2.47] T.G. Habetler and D.M. Divan, "Performance characterization of a new discrete pulse modulated current regulator", Conf. Proceedings, Industry Application Society Conference 1988, pp. 395-405.

CHAPTER 3

SYSTEM DESIGN

3.1 INTRODUCTION

This chapter describes the hardware and software involved in the design, building and operation of the SVC and APF system. The system design can be broadly divided into four main categories :

i. power circuit design

ii. PWM switching strategies

iii. modelling techniques

iv. hardware and control software requirements

Careful selection of modulation strategies ensures that the circuit operates at optimum efficiency. To aid the design process and understanding of circuit performance and behaviour, the circuit is first simulated using two different software packages. The approach used to determine the circuit model is also explained in this chapter. Finally, at the end of this chapter the hardware used to implement the system is described together with the necessary control software.

3.2 POWER CIRCUIT DESIGN

The SVC is implemented using a PWM controlled shunt connected voltage source inverter (VSI). The circuit is as shown in Figure 3.1. The AC terminals of the inverter are connected to the AC mains through a synchronous link which also serves as a first order low-pass filter. The DC side of the PWM VSI is connected to a DC capacitor

which carries the input ripple current of the inverter and is the main reactive storage element.

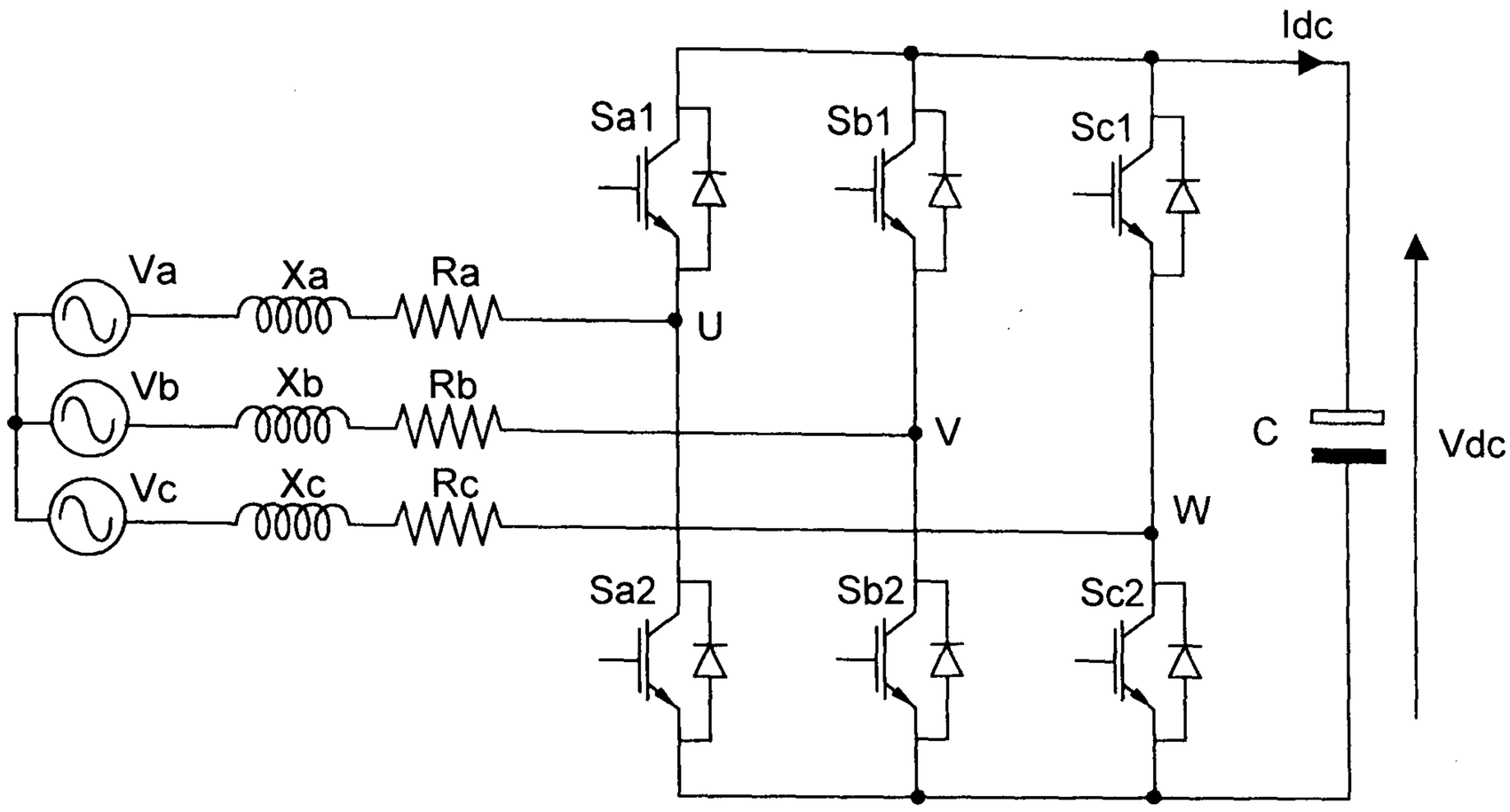


Figure 3.1 SVC with PWM controlled voltage source inverter

The three phase VSI comprises three legs, one for each phase. Each leg consists of two switches with free-wheeling diodes connected in anti-parallel. The two switches in each leg are switched in such a way that when one of them is in its off state, the other switch is on. Therefore the switches are never off (or on) simultaneously. However in practice, a period of underlap must be provided where both switches are kept off. This is due to the finite turn-off and turn-on times associated with any type of switch. When a switch is turned off, the turn on of the other switch in the same leg is delayed by a time which is conservatively chosen to avoid shoot-through or cross-conduction through the leg.

3.2.1 Switching Arrangement

Each leg of a VSI can be used in 3 possible states (i.e. 01, 10 and 00 - for the case of underlap), hence the inverter has 27 possible states. Only the 8 states that do not involve under-lap are useful for controlling the phase voltages. During the 19 transitional states

one or more of the phase voltages depend on which free-wheel diode conducts. The 8 useful states are shown in Table 3.1. Only during these states are the current paths independent of the phase current polarities.

State	Switching status						Idc
	Sc1	Sb1	Sa1	Sc2	Sb2	Sa2	
0	off	off	off	on	on	on	0
1	off	off	on	on	on	off	Ia
2	off	on	off	on	off	on	Ib
3	off	on	on	on	off	off	-Ic
4	on	off	off	off	on	on	Ic
5	on	off	on	off	on	off	-Ib
6	on	on	off	off	off	on	-Ia
7	on	on	on	off	off	off	0

Table 3.1 Switching states for VSI

Figures 3.2 parts (a) to (h) show in detail the current flow for each state. When the switch is defined as '1', this means that upper switch is 'on' and 'off' for the lower switch. The opposite is true in the case of '-1'.

3.2.2 Switching Device

The inverter is constructed using three half bridge IGBT (insulated gate bipolar transistor) modules BSM50GB120DN2 manufactured by Siemens. The maximum rating for continuous collector current, I_{CE} is 50 A and 1200 V for the collector-emitter voltage, V_{CE} [3.1].

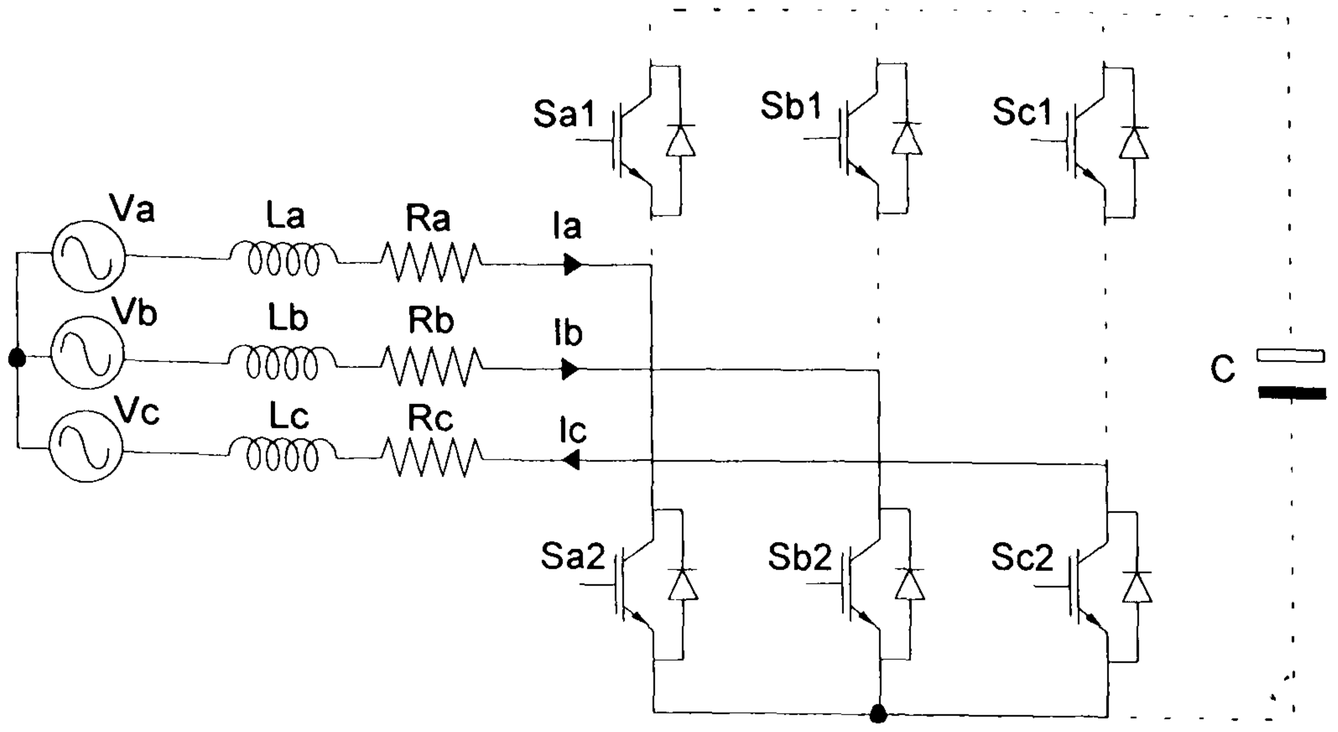


Figure 3.2(a) State 0

Sa	Sb	Sc
-1	-1	-1

Idc ≡ 0

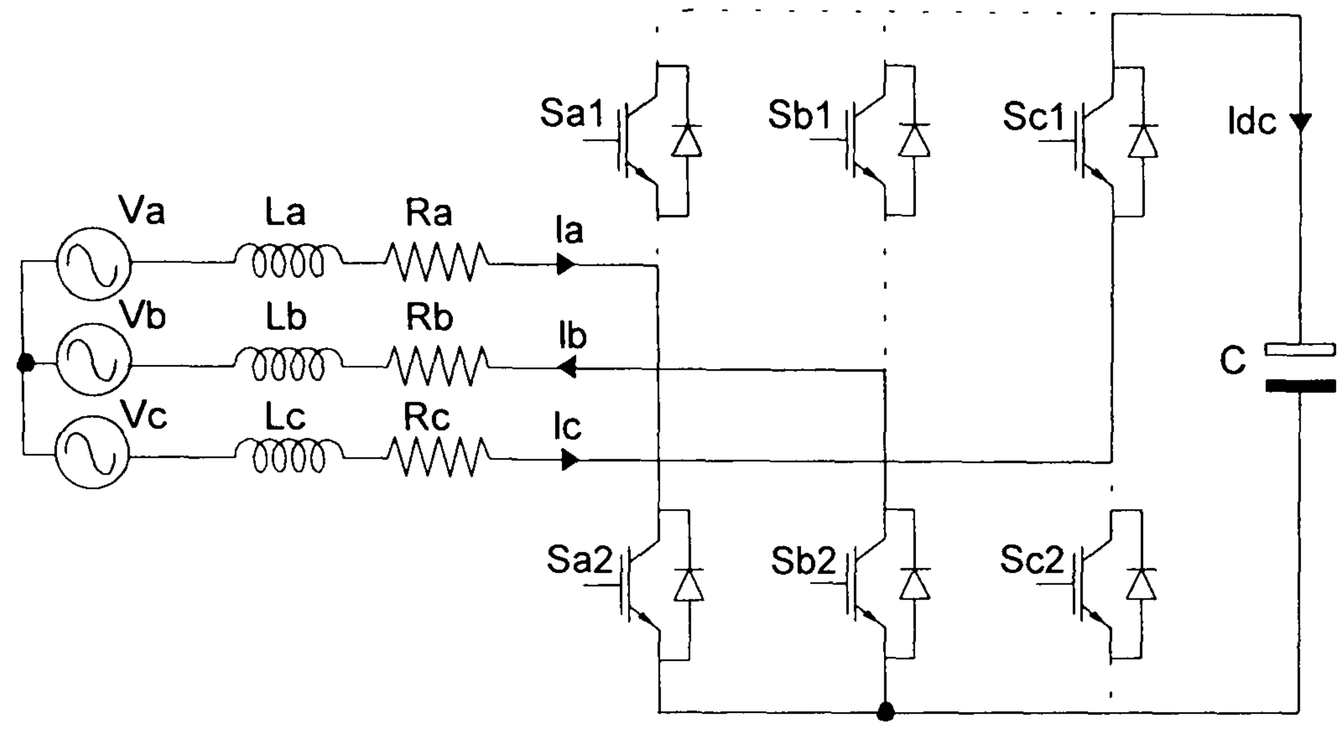


Figure 3.2(b) State 4

Sa	Sb	Sc
-1	-1	1

Idc ≡ Ic

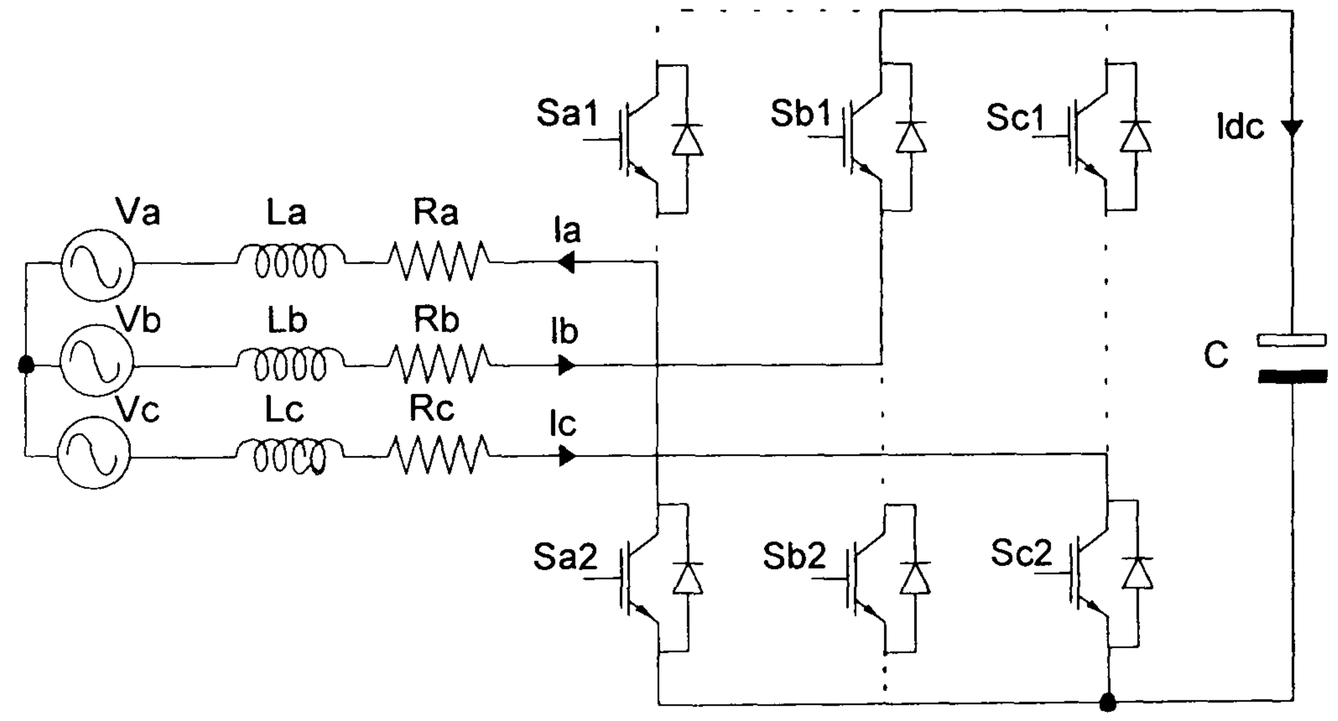


Figure 3.2(c) State 2

Sa	Sb	Sc
-1	1	-1

Idc ≡ Ib

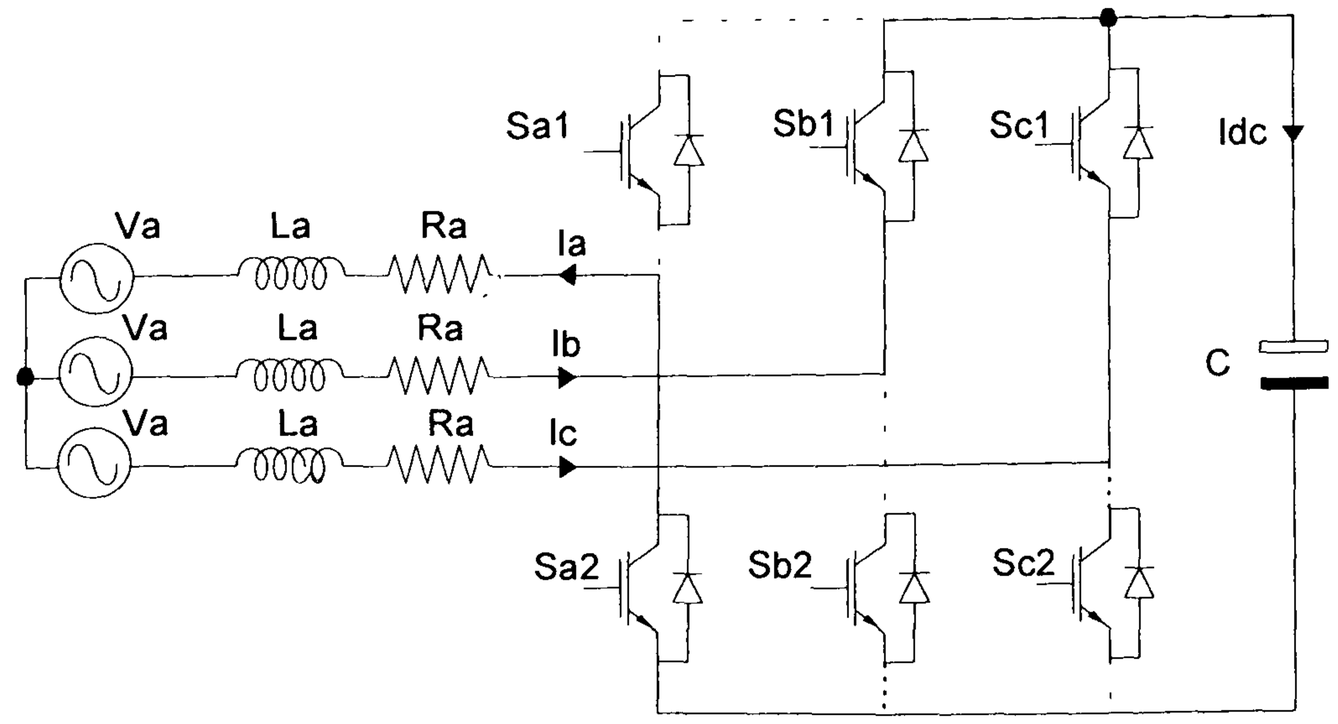


Figure 3.2(d) State 6

Sa	Sb	Sc
-1	1	1

Idc ≡ -Ia

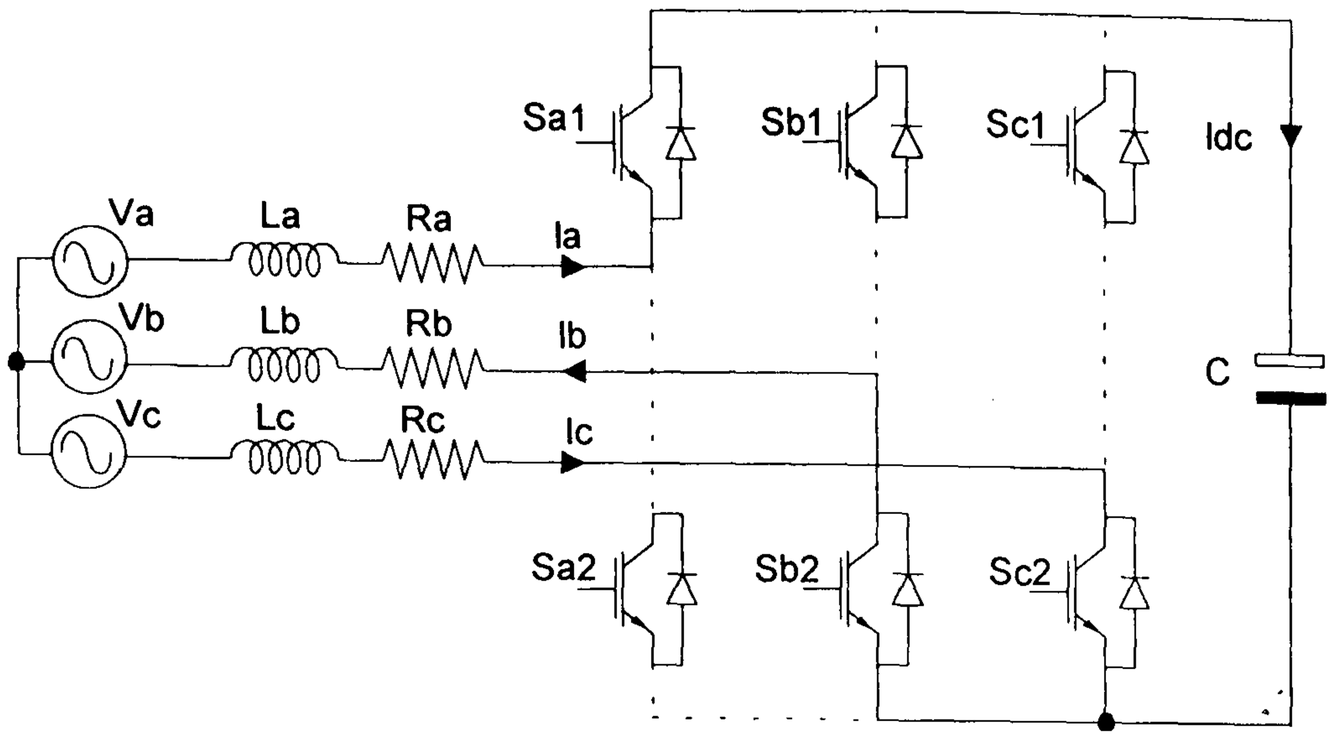


Figure 3.2(e) State 1

Sa	Sb	Sc
1	-1	-1

Idc ≡ Ia

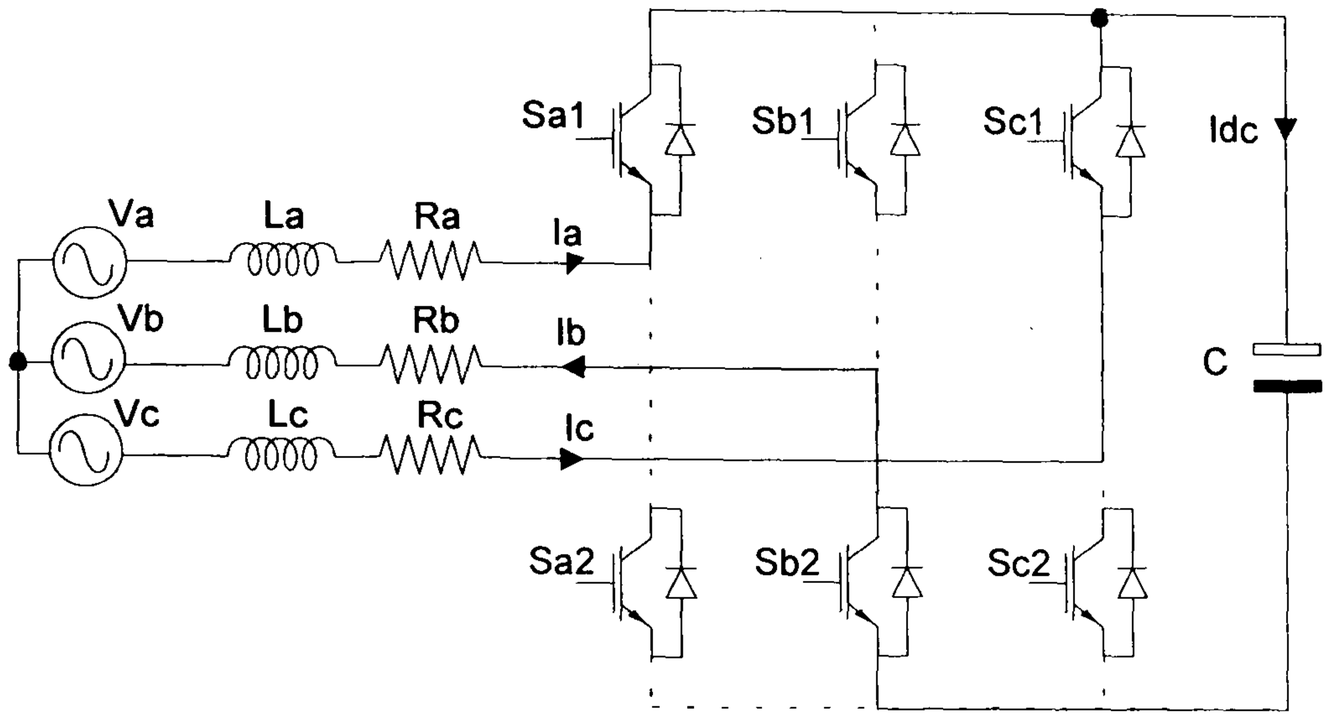


Figure 3.2(f) State 5

Sa	Sb	Sc
1	-1	1

Idc ≡ -Ib

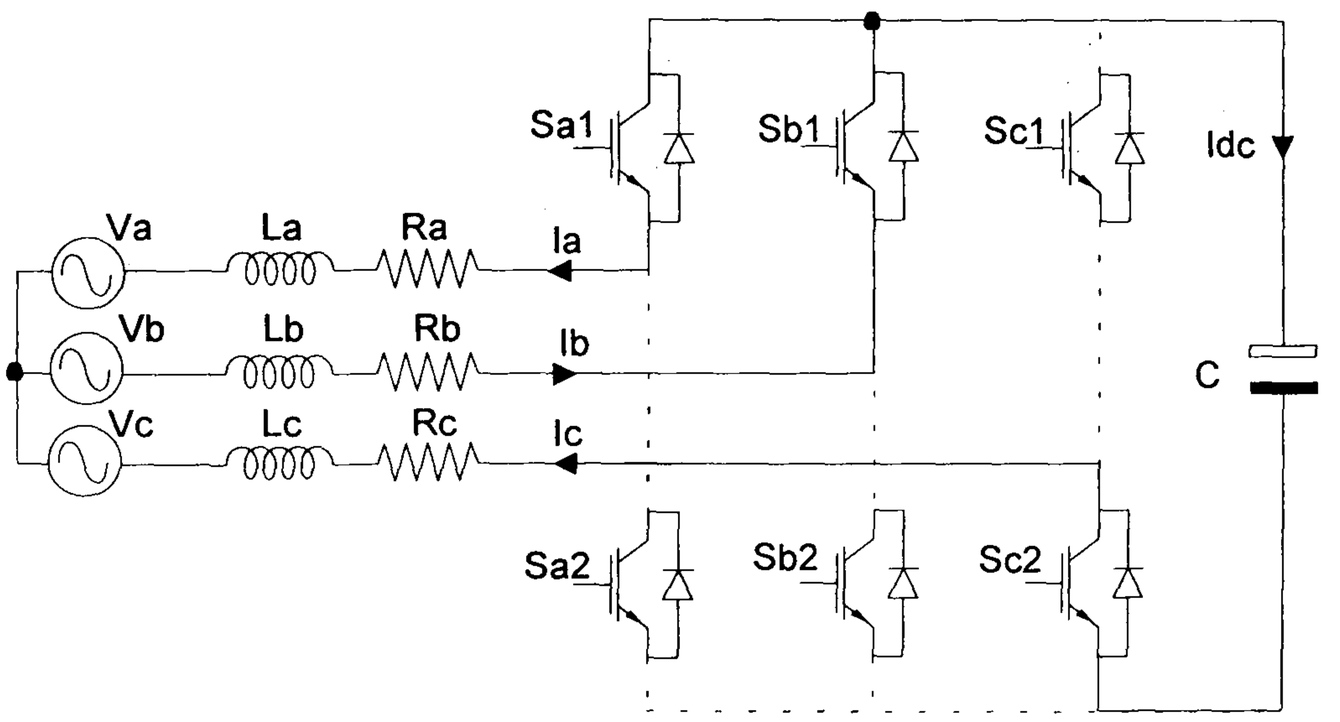


Figure 3.2(g) State 3

Sa	Sb	Sc
1	1	-1

Idc ≡ -Ic

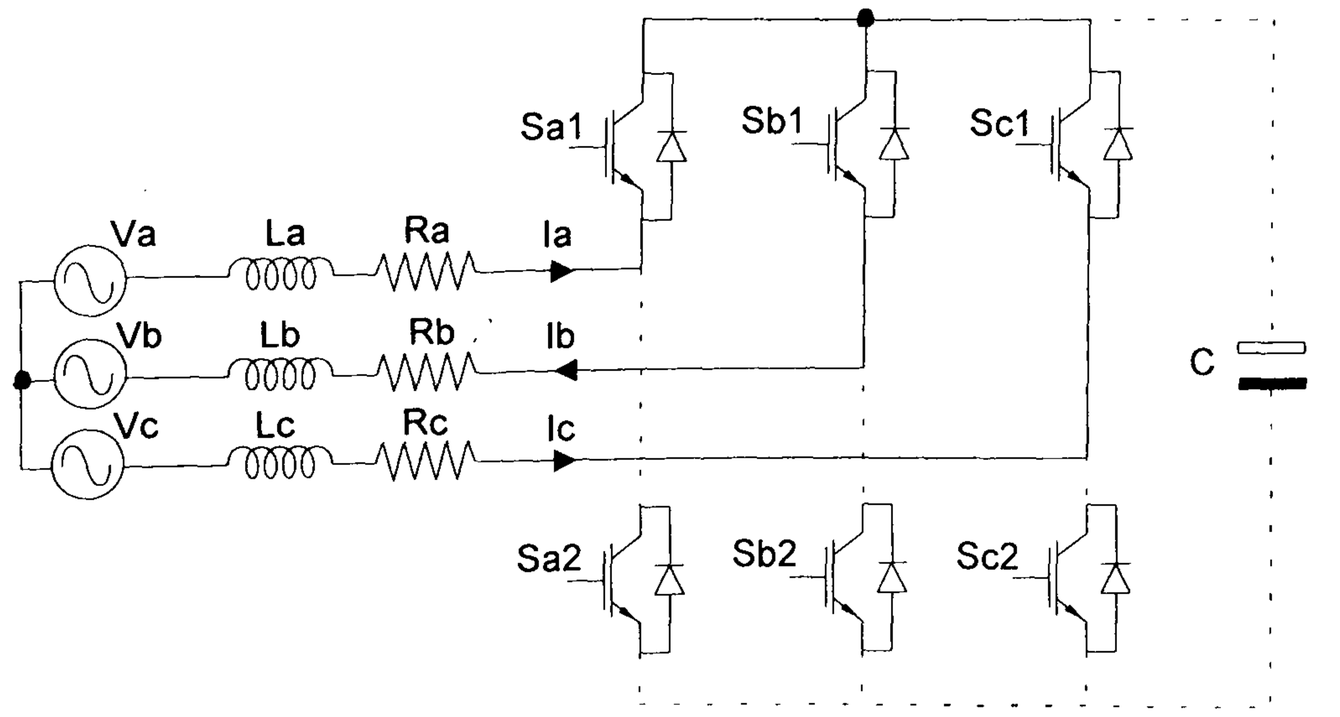


Figure 3.2(h) State 7

Sa	Sb	Sc
1	1	1

Idc ≡ 0

The IGBT has been developed in order to combine the qualities of MOSFET (metal oxide semiconductor field effect transistor) and BJT (bipolar junction transistor) devices. For this reason the device is suitable for medium power applications which require a fast switching speed. However the device has one undesirable characteristic at switch off as shown in Figure 3.3.

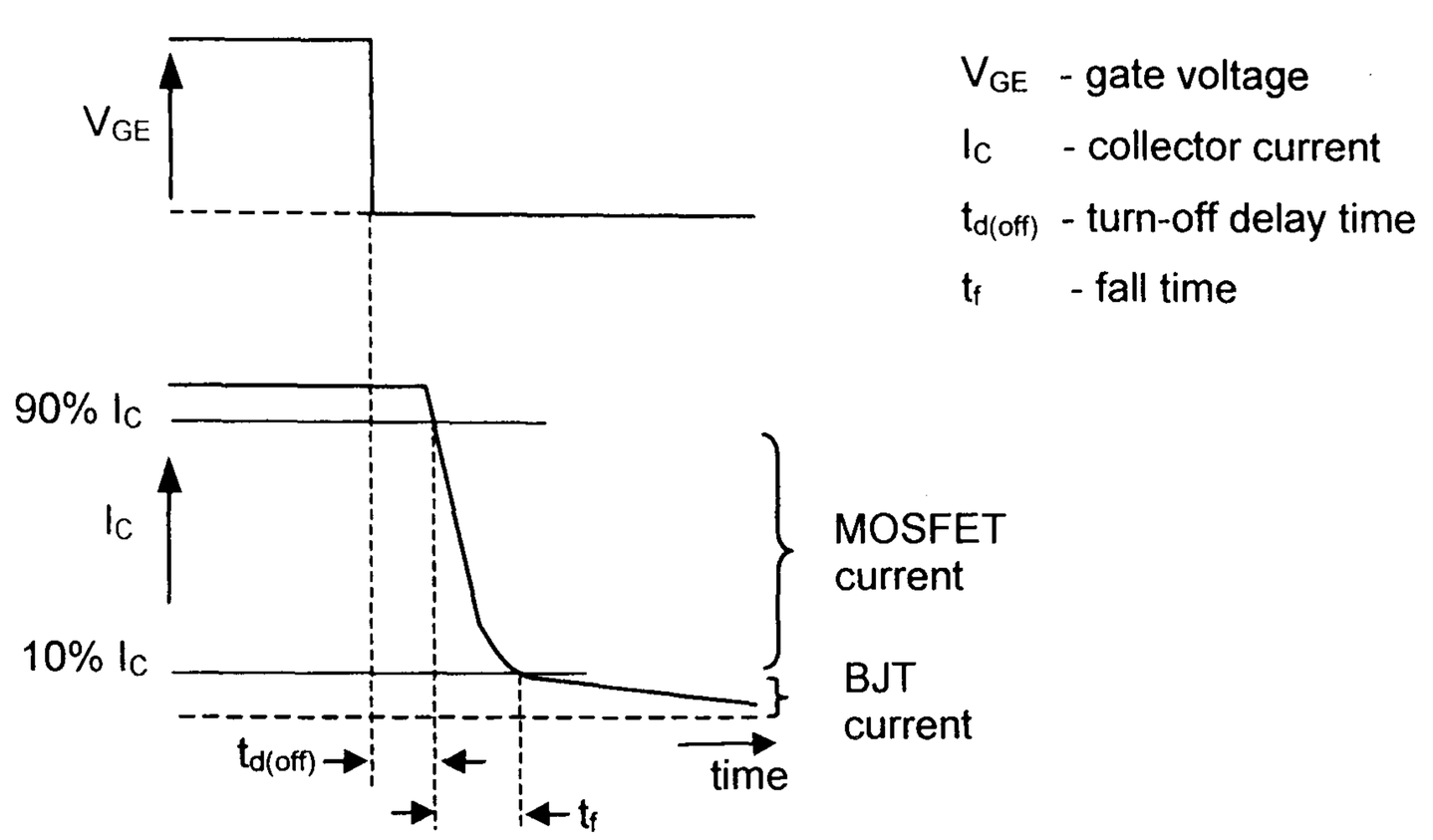


Figure 3.3 Turn-off characteristic for IGBT

The diagram shows the turn-off voltage and current waveforms for the IGBT. Two distinct time intervals can be observed in the current waveform. The initial time intervals t_d and t_f are governed by the MOSFET portion of the IGBT. The tailing of the collector current during the following interval is due to the stored charge in the n^- drift region of the IGBT. Since the MOSFET section is off and there is no reverse voltage applied to the IGBT terminals that could generate a negative collector current, there is no possibility for removing the stored charge by sweeping-out carriers. The only way that the excess carriers can be removed is by recombination within the IGBT, which takes a considerable time thus causing the current tailing problem. The set underlap time period must take this into account in addition to t_d and t_f. The underlap time is set to 1.5 μs.

3.2.3 AC Link Inductor

The inductor is designed on the basis of the peak-to-peak ripple current. The maximum peak-to-peak ripple, I_{ppmax} is set at 8% of the peak line current (40 A). The value of the inductor can then be found using the formula [3.2] :

$$L = \frac{V_{dc}}{6 f_s I_{pp\max}} \tag{3.1}$$

where V_{dc} - DC voltage across the capacitor

f_s - switching frequency

The value of V_{dc} must be at least twice the value of V_s(supply voltage). This is set at 600 V with a constant switching frequency of 5 kHz. Hence from (3.1) :

$$\begin{aligned} L &= \frac{600}{6 \text{ x } 5 \text{ x } 10^3 \text{ x } 0.08 \text{ x } 40} \\ &= 6.25\ mH \end{aligned} \tag{3.2}$$

The inductor is designed using the procedures described in [3.3]. The procedures outlined in the literature give a good estimate in finding the size and effective volume of inductor core. Once the core size has been determined, the required number of turns and air gap can then be calculated. The air gap is required to prevent saturation since 50 Hz and higher frequency components must be transmitted. The designed inductor has the following parameters (at 100 Hz) :

L = 6.45 mH, R=1.1 Ω

3.2.3 DC Link Capacitor

The capacitor is used to maintain a constant DC voltage with a small ripple. This is achieved using two 375 V 2200 μF electrolytic capacitors connected in series. The value of C obtained is based on the following equation:

$$C = \frac{1}{\Delta V} \int_{t2}^{t1} i_c(t)\, dt \qquad \textbf{(3.3)}$$

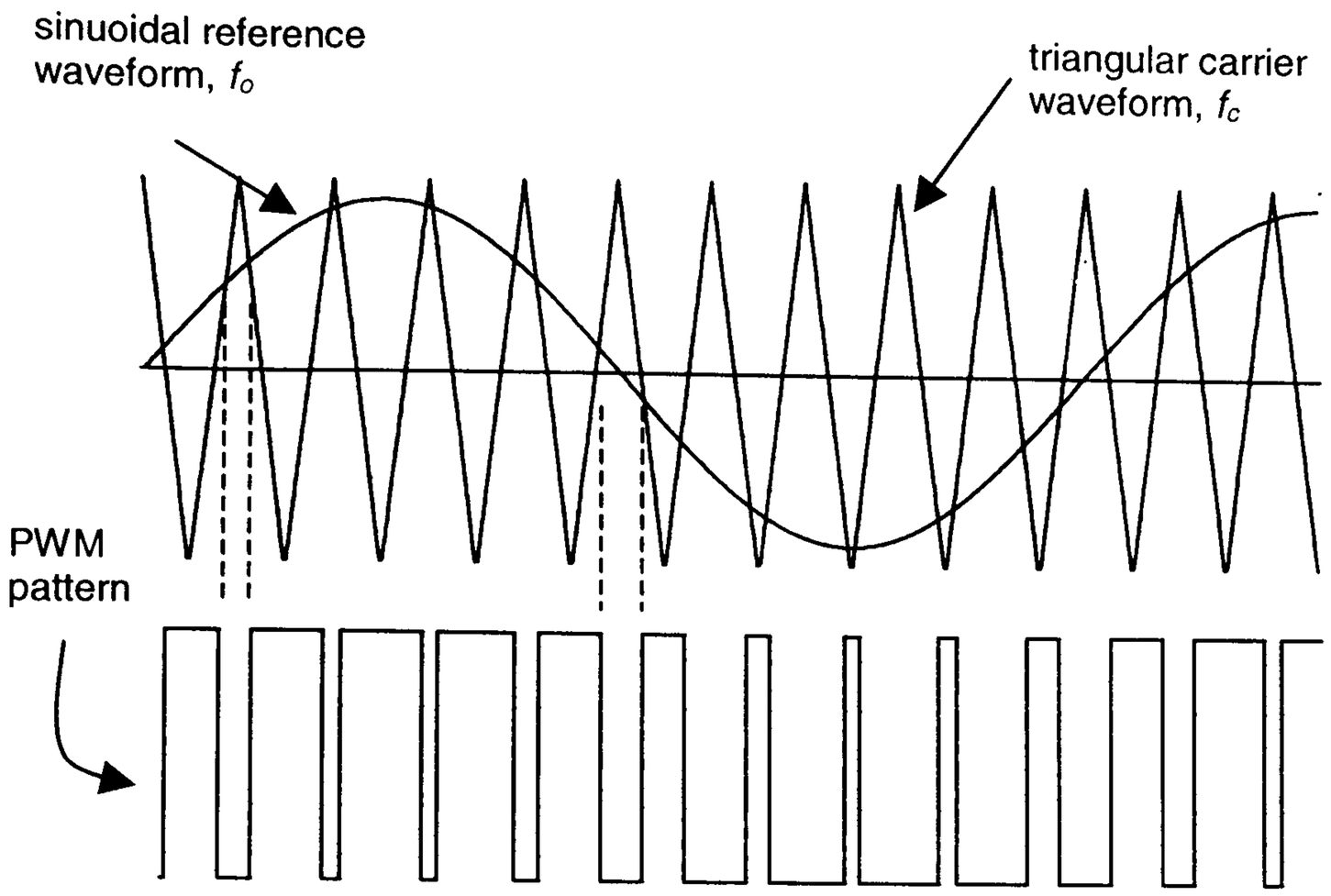
where $i_c(t)$ is the instantaneous current flowing through the DC capacitor and ΔV is the voltage fluctuation of the DC bus. The ripple voltage across the capacitor is an issue for two reasons. Firstly, the controllable switches must support the peak capacitor voltage and secondly, if the capacitor voltage drops below the peak of the supply voltage, then the compensator will lose its ability to force the line current to follow the intended shape. A voltage sharing resistor is connected across each capacitor to balance the voltages and also to provide a discharge path for stored energy in the capacitors at power down. The capacitor must be capable of operating with the ripple current produced by the ripple voltage. This ripple current produces heating.

3.3 MODULATION TECHNIQUES

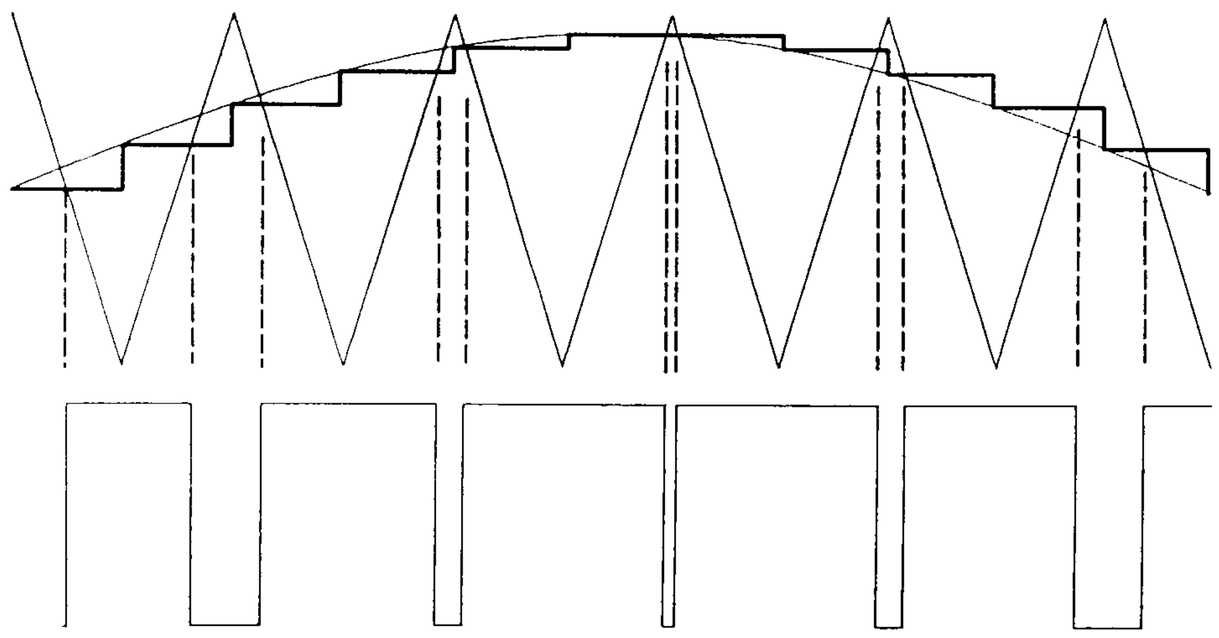
The power circuit is switched using a PWM technique. In power electronics, various types of PWM techniques are widely employed to control the output voltage of power converters because they can provide voltage and current waveshaping customised to the specific needs of the application under consideration. The simplest type of carrier PWM is sinusoidal PWM (SPWM) which is based on the principle of comparing a triangular carrier signal with a sinusoidal reference waveform. Figure 3.3 (a) shows natural sampling SPWM in which the switching points are determined by the intersection of the triangular carrier waveform, f_c and the reference modulation waveform, f_o. The output voltage is varied by controlling the modulation index, m, defined as :

$$m = \frac{\hat{v}_{ref}}{\hat{v}_{carrier}} \qquad \textbf{(3.4)}$$

where $\hat{v}_{carrier}$ is the amplitude of the triangular carrier waveform (usually fixed) and $\hat{v}_{ref}$ is the amplitude of the sinusoidal reference waveform.



(a) natural sampling



(b) regular sampling, asymmetrical modulation

Figure 3.4 Sinusoidal pulse-width-modulation (SPWM)

In digital PWM using microprocessors or digital signal processors, natural sampling is not possible due to processing time delays. A more practical approach is to use a stepped representation of the waveform as shown in Figure 3.3 (b). The modulating waveform is sampled regularly at the top and bottom peak of the carrier waveform. By

doing so, the waveform intersects the triangular carrier waveform nonequidistant about the carrier peak, hence the name asymmetrical modulation.

Although the implementation of sinusoidal PWM is relatively simple, it has a drawback. With reference to Figure 3.1, consider the case where each phase is pulse-width modulated at terminals U, V and W by the modulating functions :

$$u = \frac{1}{2}[m\sin(\omega t)+1] \qquad \textbf{(3.5a)}$$

$$v = \frac{1}{2}\left[m\sin(\omega t-\frac{2\pi}{3}) + 1\right] \qquad \textbf{(3.5b)}$$

$$w = \frac{1}{2}\left[m\sin(\omega t+\frac{2\pi}{3}) + 1\right] \qquad \textbf{(3.5c)}$$

where m is the modulation index that can vary from 0 to a maximum of 1. When the DC voltage is E, the demodulated line-to-line voltage will be :

$$V_{UV} = \left(\frac{\sqrt{3}}{2}\right) mE\sin(\omega t + \frac{\pi}{6}) \qquad \textbf{(3.6a)}$$

$$V_{VW} = \left(\frac{\sqrt{3}}{2}\right) mE\sin(\omega t + \frac{\pi}{6} - \frac{2\pi}{3}) \qquad \textbf{(3.6b)}$$

$$V_{WU} = \left(\frac{\sqrt{3}}{2}\right) mE\sin(\omega t + \frac{\pi}{6} + \frac{2\pi}{3}) \qquad \textbf{(3.6c)}$$

From (3.5), the ratio of the fundamental component of the maximum A.C line voltage to the D.C bus voltage is limited to $\sqrt{3}/2$, viz., 0.866.

It is possible to increase the ratio by adding 1/6 of the third harmonic component into the sinusoidal modulating waveform [3.6]. This enables a 15.5 % increase in gain over conventional sinusoidal PWM. In a three wire system with a balanced load, the third

harmonic components will be eliminated from the line waveform. The modulating waveform is described by :

$$u = \frac{1}{\sqrt{3}} m\left[\sin(\omega t) + \left(\frac{1}{6}\sin 3\omega t\right)\right] + \frac{1}{2} \quad \textbf{(3.7a)}$$

$$v = \frac{1}{\sqrt{3}} m\left[\sin\left(\omega t + \frac{2\pi}{3}\right) + \left(\frac{1}{6}\sin 3\omega t\right)\right] + \frac{1}{2} \quad \textbf{(3.7b)}$$

$$w = \frac{1}{\sqrt{3}} m\left[\sin\left(\omega t - \frac{2\pi}{3}\right) + \left(\frac{1}{6}\sin 3\omega t\right)\right] + \frac{1}{2} \quad \textbf{(3.7c)}$$

This will give a resultant line voltage of :

$$V_{UV} = mE\sin(\omega t + \frac{\pi}{6}) \quad \textbf{(3.8)}$$

This proves that the maximum output line voltage obtainable is increased by a factor of 15.5%, i.e. it is possible to have 100% controllability of output voltage as compare to 86.6% for conventional sinusoidal PWM. Different other techniques have also been suggested to overcome the limitation of sinusoidal PWM [3.7]. Triplen harmonics can also be injected to give a flat-topped modulating waveform [3.8]. In this thesis, a modified discontinuous switching waveform will be used in the implementation of the SVC and APF system. This technique, which is known as deadband PWM, yields better performance than other counter-parts, e.g. conventional sinusoidal PWM [3.9].

In this technique, one leg of the inverter is clamped in a 01 or 10 state for a certain period of time in each cycle, hence resulting in a deadband region in which no switching of the leg will occur. In a three phase circuit, keeping one leg of the inverter inactive will not result in loss of controllability since the remaining two legs are used for control purposes, but the effective switching frequency is reduced. Different waveforms can be used to satisfy this condition. Clamping for one third of each cycle (1/6 of the period for

both the upper and lower leg switches) is the optimal quantity for each phase voltage. If all of the phases are clamped for more than 1/3 of a cycle, then two or more phases would be clamped simultaneously, producing distorted line currents.

This modulating waveform is shown in Figure 3.5. The waveform over the first quarter cycle can be described by the following function [3.10]:

$$Va = \begin{cases} 2M\sin(wt+30°)-1 & 0° \le wt \le 60° \\ 1 & 60° \le wt \le 90° \end{cases} \quad \textbf{(3.9)}$$

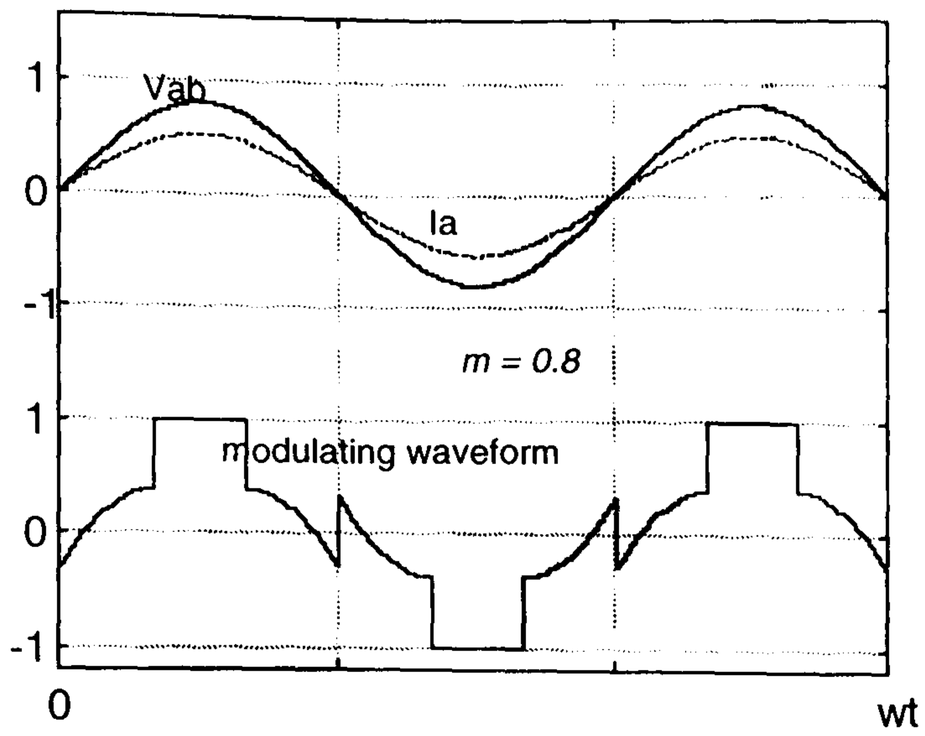
The only harmonics are odd-triplens which will be eliminated in a balanced 3 phase system. A further improvement to this technique is proposed to maximise switching loss reduction by linking the deadband regions of the discontinuous modulating waveforms about the peaks of the associated phase leg currents [3.11]. Figure 3.5 illustrates different modulating waveforms for deadband pwm for the case of a lagging power factor. For power factors between 0.866 to -0.866, the deadband coincides with the peaks of the line currents as shown in Figure 3.5(a) (for unity power factor) and Figure 3.5(b) (for 0.9 lagging power factor). For operation between 0.866 to 0 and 0 to -0.866, the clamping function occurs on the phase leg carrying the second highest line current as shown in Figure 3.5(c) (0 lagging power factor) and Figure 3.5(d) (0.6 lagging power factor).

The algorithm for deadband pwm can be explained using the truth table in Table 3.2. Sa1 .. Sc2 correspond to the switch arrangement shown in Figure 3.1. Table 3.2 shows the switch that will be clamped on for a particular interval. As an example, if Va>Vc>Vb and |Ic|>|Ia|>|Ib| (Va, Vb, Vc - phase voltages and Ia, Ib, Ic - line currents), then the switching arrangement will be as follows:

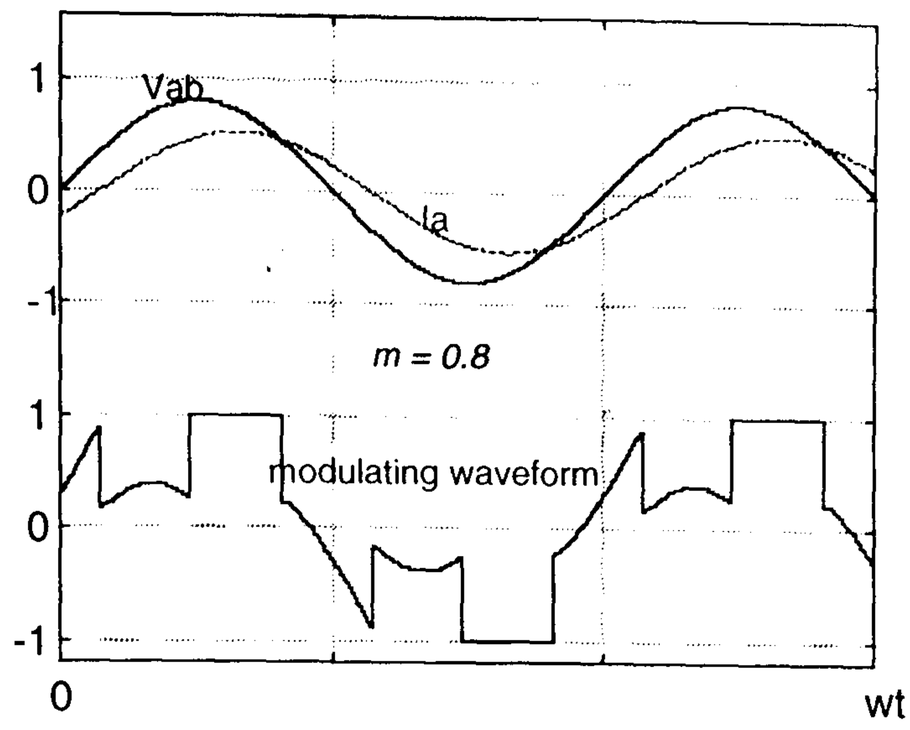
Sa1 : 1 Sa2 : -1

Sb1 : 1 - (Va - Vb) Sb2 : -1 + (Va - Vb)

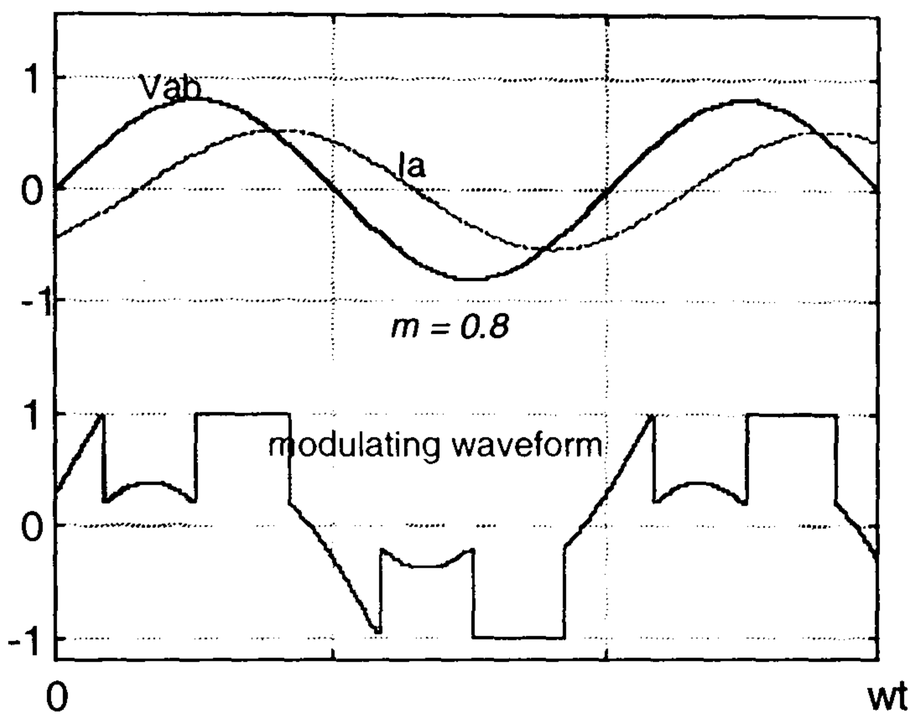
Sc1 : 1 - (Va - Vc) Sc2 : -1 + (Va - Vc)



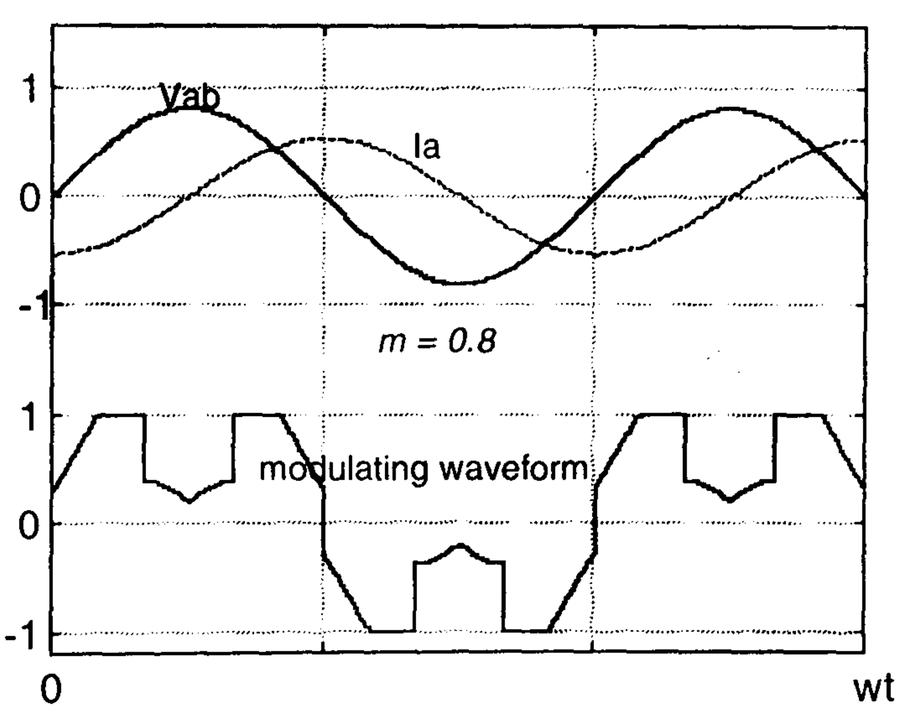
(a) Deadband at unity p.f



(b) Deadband at 0.9 lagging p.f



(c) Deadband at 0 lagging p.f



(d) Deadband at 0.6 lagging p.f

Figure 3.5 Modulation waveform variation for deadband PWM at different power factor (p.f)

With one leg clamped, the other two must be controlled to maintain sinusoidal line voltages. Using this approach the system performance will be improved in the following ways :-

i. increased overall transfer ratio of the power converter by 15.5%

ii. reduced effective switching frequency which minimises the switching losses and resulting stresses

iii. reduced size of the DC side reactive component whether voltage or current controlled

The experimental results will be presented and explained in the next chapter. It is possible to apply 120° deadbanding, however switch losses will not be evenly distributed between all six inverter switches.

voltage status	current status (1)	(2)	(3)	(4)	(5)	(6)
(a)	Sa1	Sa1	Sa1	Sc2	Sc2	Sc2
(b)	Sa1	Sa1	Sb2	Sb2	Sa1	Sb2
(c)	Sb1	Sc2	Sb1	Sb1	Sc2	Sc2
(d)	Sb2	Sa2	Sb1	Sb1	Sa2	Sb1
(e)	Sb2	Sc1	Sb2	Sb2	Sc1	Sc1
(f)	Sa2	Sa2	Sa2	Sc1	Sc1	Sc1

Table 3.2. Clamping function for different voltages and currents

current status	condition
(1)	|Ia|>|Ib|>Ic|
(2)	|Ia|>|Ic|>|Ib|
(3)	|Ib|>|Ia|>|Ic|
(4)	|Ib|>|Ic|>|Ia|
(5)	|Ic|>|Ia|>|Ib|
(6)	|Ic|>|Ib|>|Ia|

voltage status	condition
(a)	Va>Vb>Vc
(b)	Va>Vc>Vb
(c)	Vb>Va>Vc
(d)	Vb>Vc>Va
(e)	Vc>Va>Vb
(f)	Vc>Vb>Va

Table 3.3 Current and voltage conditions for different switching status

3.4 MODELLING TECHNIQUES

In order to study the controllability and performance of the SVC and APF system, various simulation packages can be used. In general there are two types of simulator that can be used namely, circuit-oriented simulators and equation solvers. Both types are used in this work to simulate different circuit types and operating conditions. A model must first be developed so that its dynamic behaviour can be studied i.e. the consequence of errors, disturbances, uncertainties, noise and circuit variations which cause circuit operation to deviate from normal operating conditions.

With a circuit-oriented simulator, a model has been developed to study the open loop response of the system using PSpice [3.12] which incorporates a program written in C. PSpice is a commercial version of SPICE (Simulation Program with Integrated Circuit Emphasis) running on a personal computer. Using this software package, the simulator will internally generate and solve the circuit equations from the supplied circuit topology and component values given. Theoretically, the package can also be used to study the closed loop system operation, but this tends to be more complicated. Such a simulation will take hours and is subject to convergence problems.

An alternative method to study closed loop performance is to use MATLAB with its graphical user interface, SIMULINK [3.13]. MATLAB is an integrated technical computing environment that combines numeric computation, advanced graphics and a high level programming language. In Simulink a model is defined using block diagrams. The model can be analysed by choosing options in Simulink menus or by entering commands in the Matlab command window. One useful feature is that it is possible to include a user defined function (S-function) written in Matlab language.

3.4.1 Solution Techniques for System Modelling

For power electronic circuits, one method to study dynamic characteristics is by applying a linearisation technique. The technique yields a linear model that approximately describes small deviations or perturbations from nominal operating conditions. For example in a static VAr compensator, applying a d-q transformation and the linearization technique, the equations for the model around the operating point can be described by :

$$v_d - e_d = Ri_d + L\frac{di_d}{dt} - \omega_o Li_q - Li_q\frac{d\Delta\delta}{dt}$$
$$v_q - e_q = Ri_q + L\frac{di_q}{dt} + \omega_o Li_d + Li_d\frac{d\Delta\delta}{dt} \qquad \textbf{(3.10)}$$

and the steady state equations are :

$$v_{do} - e_{do} = Ri_{do} + L\frac{di_{do}}{dt} - \omega_o Li_{qo}$$
$$v_{qo} - e_{qo} = Ri_{qo} + L\frac{di_{qo}}{dt} + \omega_o Li_{do} \qquad \textbf{(3.11)}$$

$$e_{do} = mV_{dc0}$$

where :

e_{do} - inverter output voltage

$\Delta\delta$ - change in phase shift angle

The switch averaging technique can also be applied to find the dynamic model of a system. This technique can be applied when the average values of voltages and currents are of concern rather than their instantaneous values.

However the method just mentioned is not suitable for the work done in this thesis. One of the aims of this study is to compare the use of deadband PWM and sinusoidal PWM. It is therefore necessary to observe the instantaneous values. With linearisation or circuit

averaging techniques it is difficult to see the effect of using different types of PWM. As shown in equations (3.10) and (3.11), in the model found, it is only necessary to control the modulation index and phase angle without the need of giving switching signals. Hence it is not possible to test the model with different types of PWM. A model needs to be developed which includes nonlinearities and at the same time reduce the complexities of the model and enables short simulation times. One solution is to use a state space approach. This will be described further in section 3.4.3.

3.4.2 Circuit Model in PSpice

PSpice analysis requires that all nodes in the circuit be numbered or given alphabetic designation. The input file to model the circuit (Figure 3.1) with sinusoidal PWM is given in Appendix A.1. The power inverter circuit is modelled using ideal voltage switches and the built-in diode model. The PWM signals are implemented by using voltage controlled voltage sources which behave as comparators. Appendix A.2 shows another PSpice model which incorporates triplen harmonic (flat top) PWM.

In these two models the simulation parameters must be carefully defined to ensure convergence. This problem is mainly caused by the PWM circuits which are included in the model. Depending on the switching frequency, a significant amount of simulation time is required to generate the PWM signals hence increasing the overall simulation time. To minimise convergence problems and to reduce simulation times, an alternative method is used which generates the PWM switching signals outside the PSpice model. This can be implemented using a C program.

PSpice has a function .INC which can be used to include an external file in PSpice format. This file consists of the switching signals generated by the C program in a piecewise-linear source (PWL) form. The listing of the program file is included in Appendix B. As a comparison both the model with sinusoidal PWM in PSpice and combination of C and PSpice have been simulated and the results are as shown in Table 3.4. The combined use of PSpice and C gives a circuit model which reduces the simulation time by 83 %. The simulation times for six cycles are with a Pentium 90 MHz PC with 24 MByte of RAM.

System model in	Simulation time for 6 cycles
PSpice only	1088 s
Combination of PSpice and C	178 s

Table 3.4 Simulation times for the different circuit models

3.4.3 Circuit Model in MATLAB

In section 3.2.1, eight switching states that do not involve underlap have been identified by which it is possible to control the phase voltages. From these eight switching states, state space equations can be derived using basic circuit laws. For example, consider the case of State 2 (-1 1 -1). The circuit is redrawn as shown in Figure 3.6. The following assumptions are made to simplify the analysis :

- Initial direction of line currents is +ve i.e. from left to right (once the equation is solved, the sign will give the correct direction);
- Ideal switches and capacitor (no ESR) are used;
- Under-lap cases are not considered;
- Inductors are linear (no saturation).

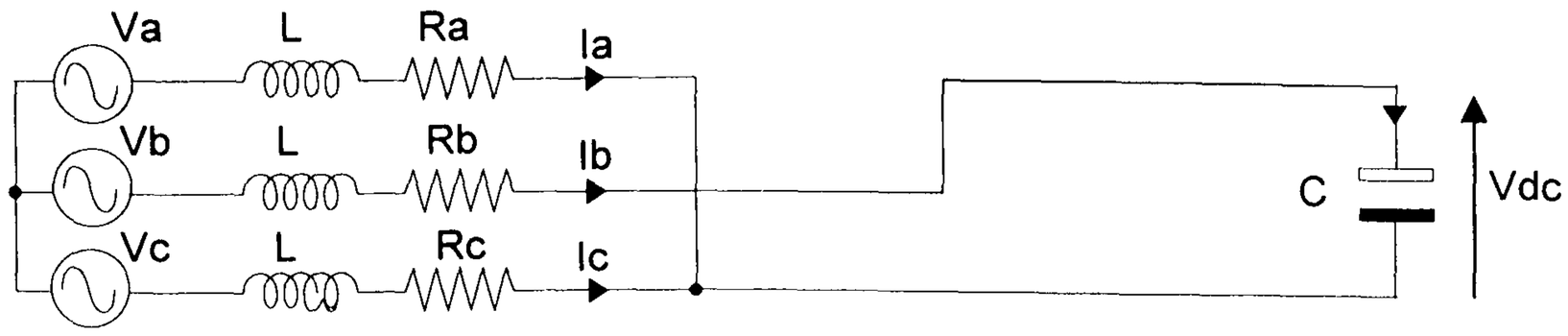


Figure 3.6 Simplified circuit for switching state 2 (-1 1 -1)

Applying Kirchoff's voltage and current law:

$$L\frac{dI_b}{dt} = V_b - V_{dc} - V_c - I_b R_b + I_c R_c + L\frac{dI_c}{dt} \tag{3.12}$$

$$3L\frac{dI_c}{dt} = 2V_c - V_a - V_b - I_b R_a - I_c R_a + I_b R_b - 2I_c R_c \tag{3.13}$$

$$I_a = -I_b - I_c \tag{3.14}$$

$$\frac{dV_{dc}}{dt} = \frac{I_b}{C} \tag{3.15}$$

Now, these equations can be re-arranged to obtain the state space representation of the model for State 2, which is :

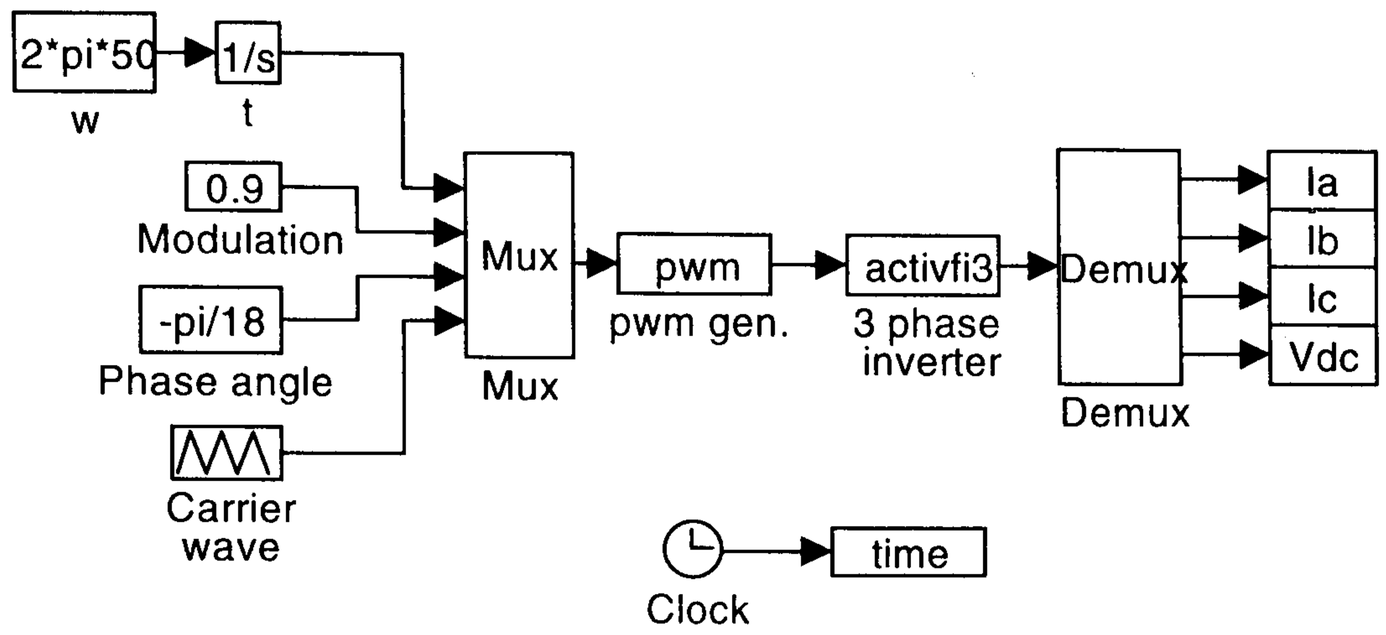
$$\dot{x} = Ax + Bu$$

$$p\begin{bmatrix} I_b \\ I_c \\ V_{dc} \end{bmatrix} = \begin{bmatrix} \frac{-R_a - 2R_b}{3L} & \frac{-R_a + R_c}{3L} & -\frac{2}{3L} \\ \frac{-R_a + R_b}{3L} & \frac{-R_a - 2R_c}{3L} & \frac{1}{3L} \\ \frac{1}{C} & 0 & 0 \end{bmatrix} \begin{bmatrix} I_b \\ I_c \\ V_{dc} \end{bmatrix} + \begin{bmatrix} -\frac{1}{3L} & \frac{2}{3L} & -\frac{1}{3L} \\ -\frac{1}{3L} & -\frac{1}{3L} & \frac{2}{3L} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} \tag{3.16}$$

$$y = Cx$$

$$\begin{bmatrix} I_a \\ I_b \\ I_c \\ V_{dc} \end{bmatrix} = \begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_b \\ I_c \\ V_{dc} \end{bmatrix} \tag{3.17}$$

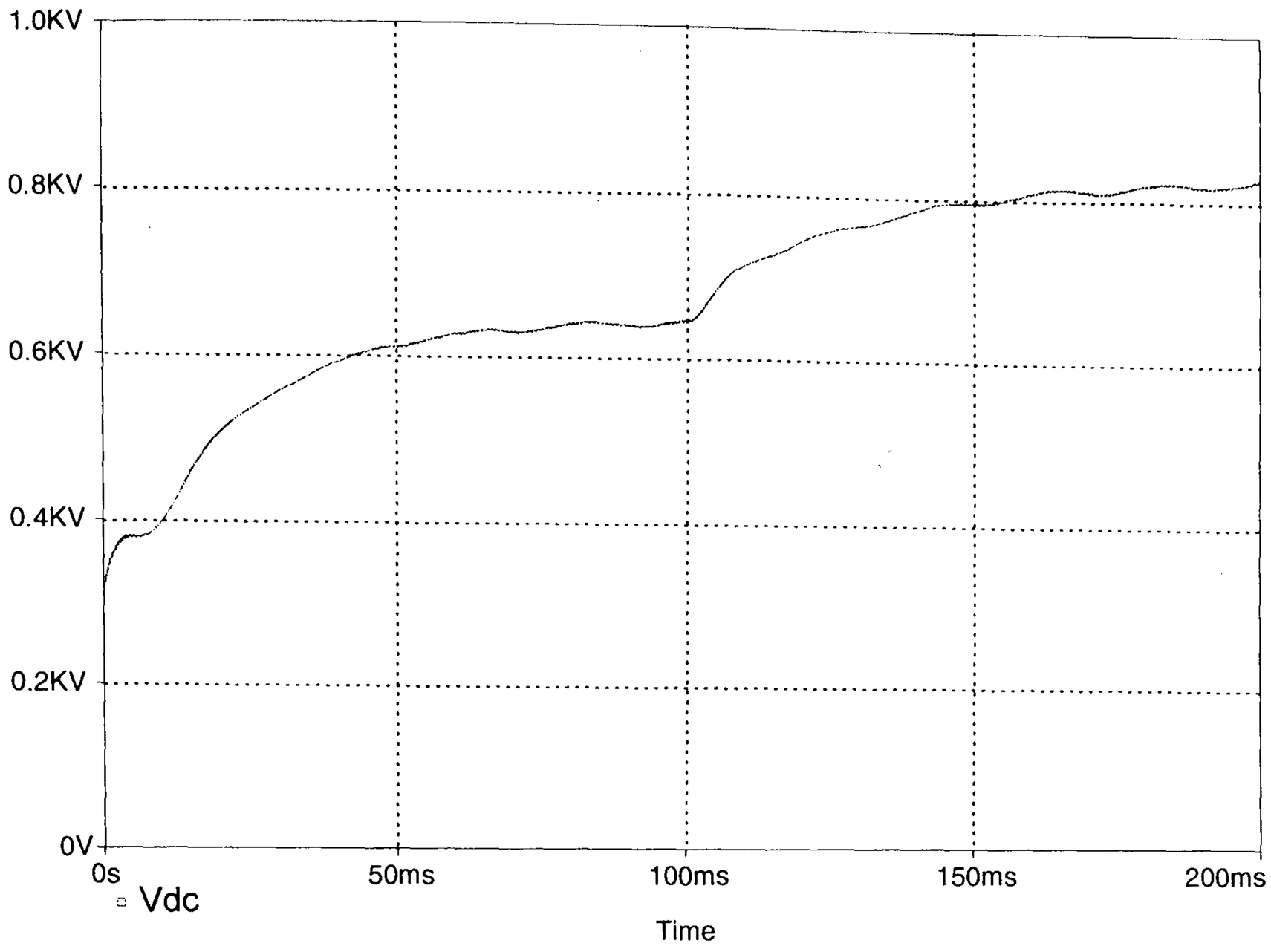
Using the same approach, the state space representation for the other 7 states can be found. A program can then be written in MATLAB to model the inverter using the derived equations. This program is shown in Appendix C.1. Figure 3.7 show how the model is represented in Simulink.



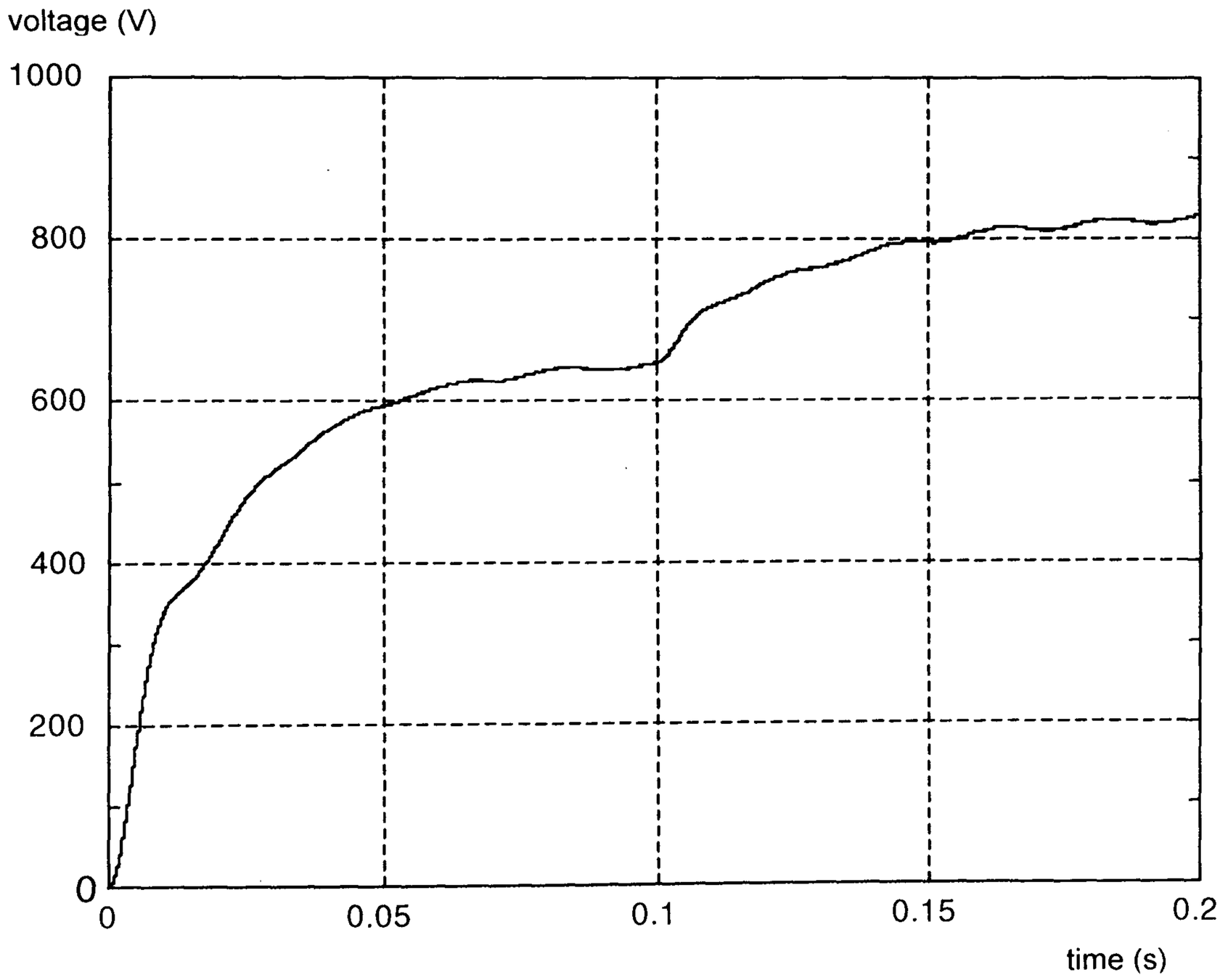
pwm : S-function to implement the PWM control action
activfi3 : S-function to model 3-phase inverter

Figure 3.7 Basic model representation for the SVC and APF

To validate the derived model, the simulation results are compared with PSpice simulations. This can be done by comparing the waveforms for the DC voltage and line current as shown in Figure 3.8 and Figure 3.9 respectively, for an open loop test. Both results are similar although initially the magnitude differs (for $t < 30ms$) during which the circuit has not reached steady state. The simulation packages use different approaches to simulate the circuit thus setting up different initial conditions. To observe how the models behave when there is a sudden change in a circuit control variable, a step change in modulation index is set at $t = 100ms$. Looking at both simulation results and assuming the PSpice simulations are correct (later verified experimentally), it may be concluded that the MATLAB model is valid and will give the correct results when used for closed loop analysis. Better results can be obtained if the same intial conditions are used in both simulation techniques.



(a) PSpice simulation results

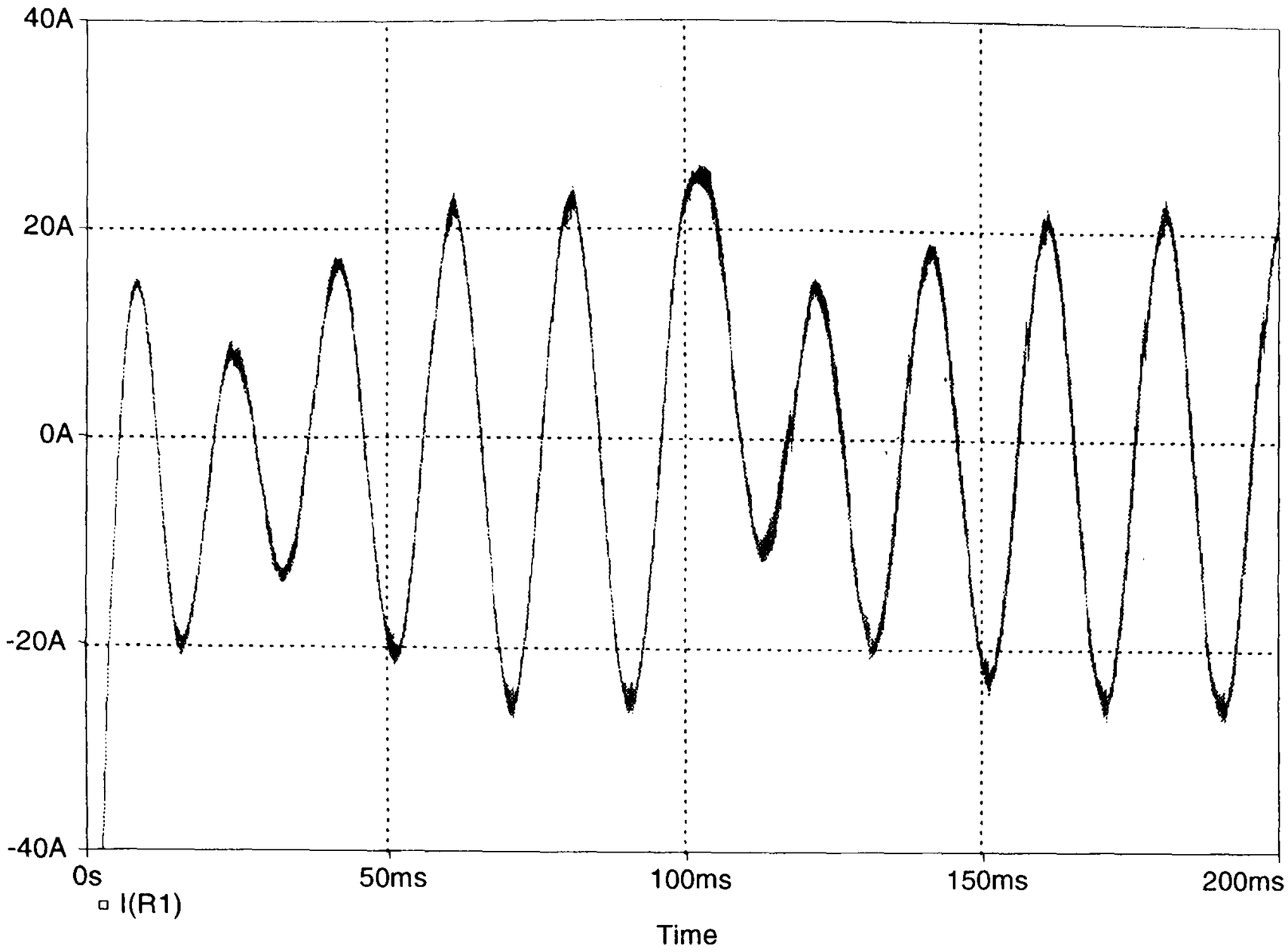


(b) MATLAB simulation results

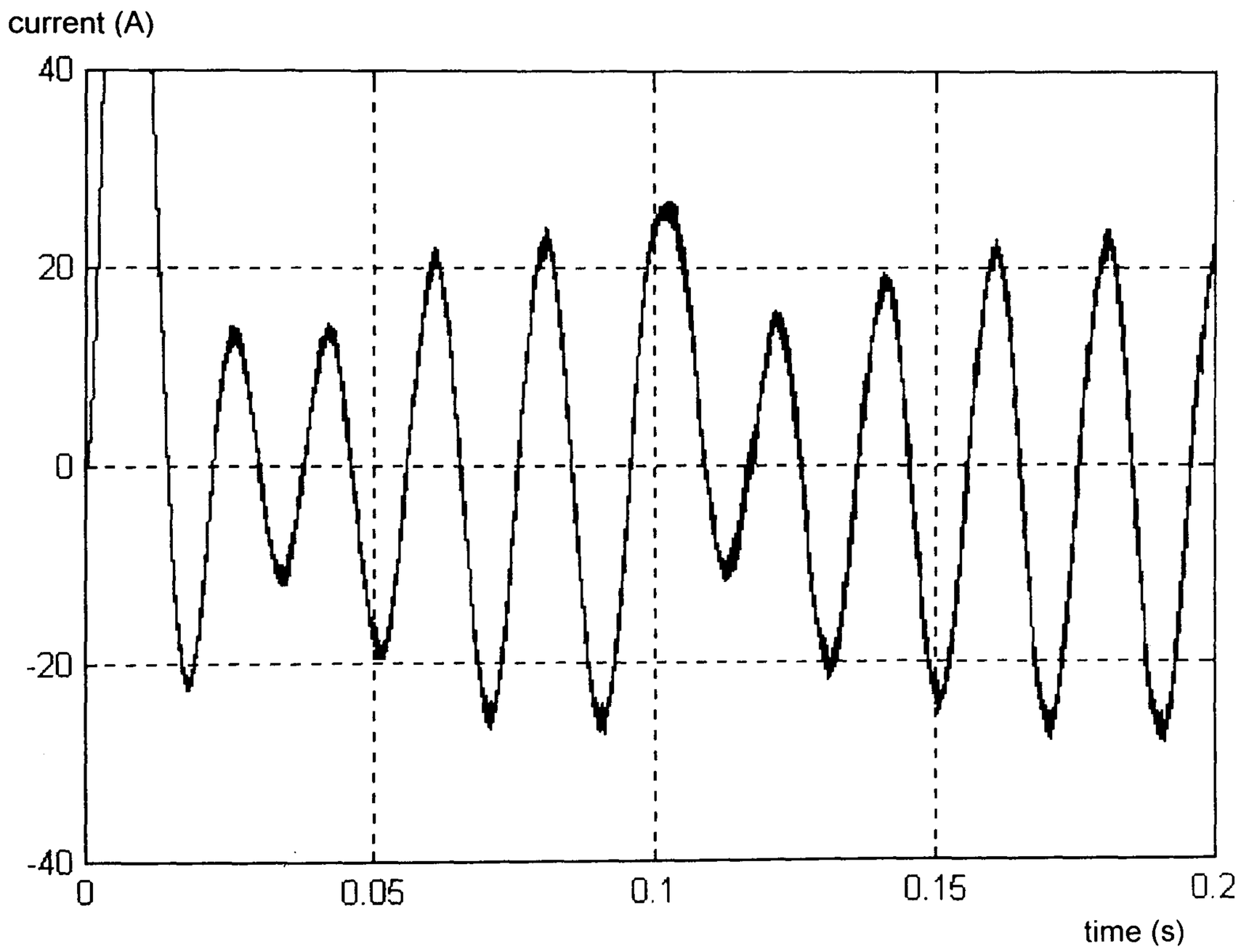
Figure 3.8 DC voltage across the capacitor with

$m = 0.9$: $0 < t < 100ms$

$m = 0.7$: $100ms < t < 200ms$



(a) PSpice simulation results



(b) MATLAB simulation results

Figure 3.9 Line current waveform with
m = 0.9 : 0 < t < 100ms
m = 0.7 : 100ms < t < 200ms

3.5 HARDWARE AND SOFTWARE IMPLEMENTATIONS

In this section a brief explanation is given on the implementation of the hardware and software involved in designing and building the SVC and APF system. In general the following basic requirements must be met:

- ability to sample and store analogue signals at fast speed
- have the processing power to analyse the data and detect the harmonic and reactive power
- ability to implement a high performance control system
- provide PWM switching signals to the inverter circuit

These requirements can be implemented either with analogue, digital or a combination of both technologies. An analogue system will provide a continuous process and high measured resolution although it suffers from component aging and temperature drift. It requires hard-wired solutions which make circuit modifications or upgrades difficult. With a digital system, analogue data is sampled at discrete time intervals thus limiting the system. The accuracy of the signal is limited by the resolution of the analogue to digital converter (ADC) being used. However with a digital system, it is not affected by component aging or temperature drift and can provide stable performance. The system is programmable and thus easier to upgrade and change the initial design. The complex process of determining the active/reactive power and generating the control and switching signals is easier to implement in a digital system. A digital approach has been used to meet these requirements. This approach involves the use of a digital signal processor (DSP) and a field programmable gate array (FPGA).

3.5.1 General Overview

Figure 3.10 shows a general system overview for the SVC and APF. The digital signal processor (DSP) is used to process and calculate the reactive power and harmonics that need to be compensated as well as performing the necessary control action in real time. Using the algorithm suggested in the previous chapter, the modulating waveform is calculated and passed to the field programmable gate array (FPGA) which implements the gating logic. A state machine is designed in the FPGA for DSP interfacing and synchronising the control algorithm. An interrupt signal is sent to DSP when new data is available.

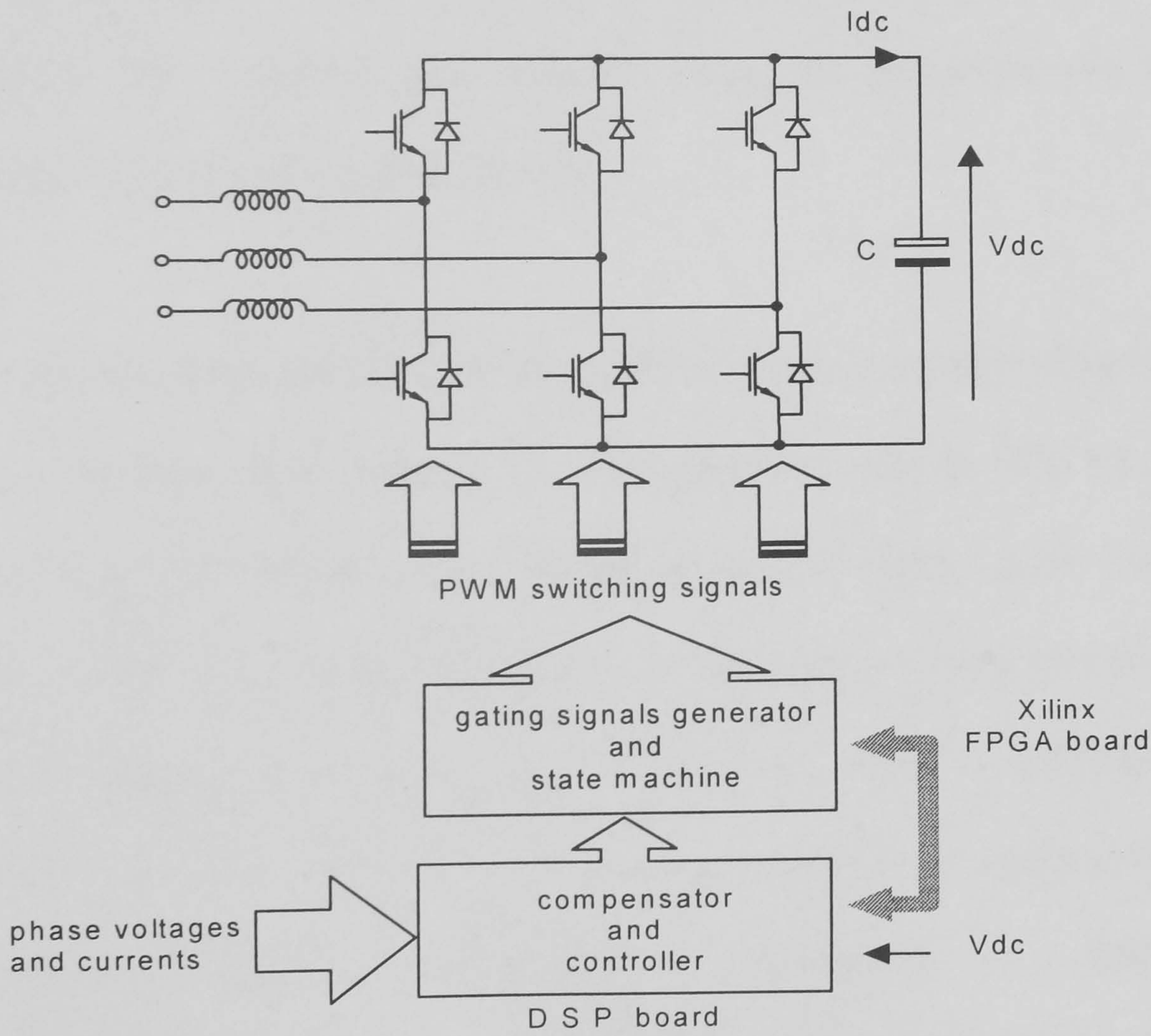


Figure 3.10 General system overview

The system is implemented using a fast, floating point Motorola DSP96000 operating at a clock frequency of 40 MHz [3.14-3.15]. It uses dual port architecture (two independent data and address buses) which allows 32-bit floating point operations to be

carried out. It has the capability to access two physical memories simultaneously and perform both floating point addition/subtraction and multiplication processes in one instruction. There are two forms of hardware interrupts available, fast and slow interrupts. When the fast interrupt is activated, it will insert two instructions into the pipeline without flushing the pipeline and without incurring any overheads such as status or program counter saving. This allows a very fast and efficient response to a peripheral interrupt request. The DSP program is downloaded via a supervisor board through the on-chip emulation (ONCE^{TM}) port using the Application Development System (ADS) supplied by Motorola. The board is controlled via a PC which is also used to write and download the program. The DSP program is written using assembly language code which is then compiled using the assembler and linker programs provided with the development system.

The FPGA used is a XILINX XC4003-5PC84 which is capable of operating at 40MHz. Using the same clock signal as the main processor enables both the DSP and FPGA board to operate synchronously. FPGA devices are made up of combinatorial logic block functions, input/output blocks and programmable interconnections. The design file is created schematically using Active-CAD which is provided in the Xilinx Foundation Series. The design file is then translated to a configuration file using the Xilinx Design Manager before it can be implemented in a Xilinx device. This configuration file can be stored in an EEPROM (electrical erasable programmable read only memory) which is then downloaded into the Xilinx device at power up.

3.6 REFERENCES

[3.1] IGBT Modules, Data Book 08/95, Siemens Semiconductor Group

[3.2] N. Mohan, T. M. Underland and W. P. Robbins, "Power Electronics - Converters, Applications and Design", Second edition, 1995, John Wiley & Sons.

[3.3] W. Y. Manka, "Design power inductors step by step", Electronic Design 26, December 20, 1977, pp. 9-15.

[3.4] B. W. Williams, "Power electronics - Devices, Drivers, Applications and Passive components", Second edition, 1993, Macmillan.

[3.5] T. Kenjo, "Power Electronics for the Microprocessor Age", 1990, Oxford University Press.

[3.6] J. A. Houldsworth and D. A. Grant, "The use of harmonic distortion to increase the output voltage of a three-phase PWM inverter", IEEE Trans. on Industry Applications, Vol IA-20, No. 5, September/October 1994, pp. 1224-1227.

[3.7] M.A. Boost, P.D. Ziogas, "State of the art carrier PWM techniques : A critical evaluation", IEEE Trans. on Industry Applications, Vol. IA-24, March/Apr. 1988, pp. 271-280.

[3.8] F. G. King, "A three phase transistor class-B inverter with sinewave output and high efficiency," IEE Conf. Publications 123, Power Electronics, Power Semiconductors and Their Applications, 1974, pp. 204-209.

[3.9] V.G. Agelidis, P.D. Ziogas and G. Joos, "Dead-band PWM switching patterns", Conf. Proc. of the IEEE-PESC 1992, pp. 427-434.

[3.10] D.R. Alexander and S.M. Williams, "An optimal PWM algorithm implementation in a high performance 125 kVA inverter", Conf. Rec. APEC '93.

[3.11] J. E. Fletcher, "Design, analysis and control of a synchronous reluctance machine", Ph.D. Thesis, Heriot-watt University, Edinburgh, 1995.

[3.12] PSpice, Microsim Corporation, 20 Fairbanks, Irvine, CA 92718, USA.

[3.13] MATLAB, The Math Works Inc., 24 Prime Park Way, Natick, MA 01760.

[3.14] Motorola Semiconductors Inc., "96-bit General Purpose IEEE Floating-point Dual-port Processor", Motorola Technical Data - advance information, 1990.

[3.15] Motorola Semiconductors Inc., "DSP96002 IEEE Floating-point Dual-port Processor User's Manual, Motorola Technical Data, 1989.

CHAPTER 4

STATIC VAR COMPENSATOR WITH DEADBAND PWM

4.1 INTRODUCTION

This chapter investigates the operation and performance of a static VAr compensator (SVC) with deadband PWM. As discussed in the previous chapter, this PWM technique enables higher modulation indices to be achieved hence facilitating a smaller size of reactive component on the DC side. The deadbanding concept reduces the effective switching frequency thus minimising the switching losses and resulting device stresses. At the end of this chapter the effectiveness of the proposed design will be studied using simulation and experimental results. Detailed explanation will also be given on the design implementation of the Xilinx FPGA and Motorola DSP960002 programming.

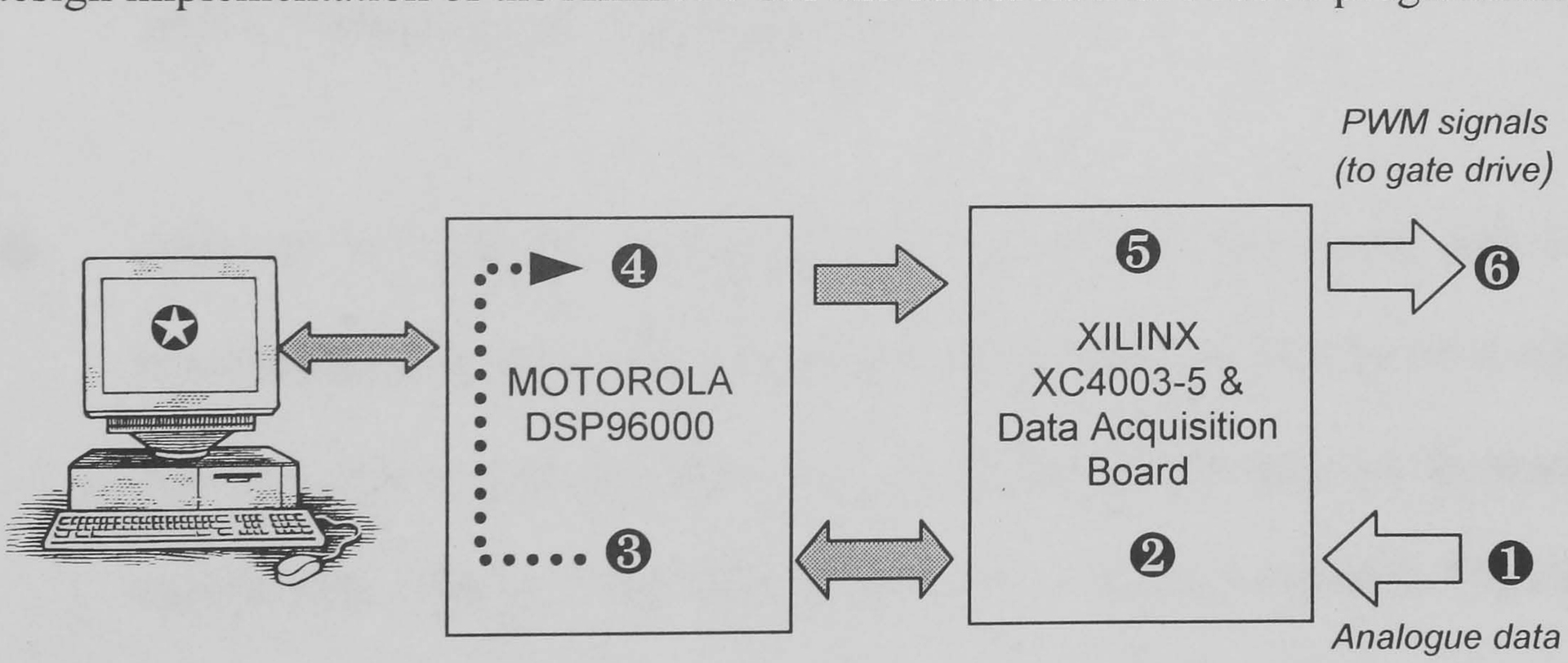


Figure 4.1 Control stages

4.2 OVERVIEW OF DATA FLOW IN XILINX FPGA AND DSP

Figure 4.1 shows the functional block diagram of the Xilinx and DSP based SVC system. These different blocks are described below:

✪ The PC is used to compile and download the DSP program via the supervisory board. During the development of the DSP program, simulation is used to check the output data. The contents of various registers can also be accessed during program debugging. When the final program is executed, the DSP operates independently.

❶ The analogue signals are obtained using LEM voltage and current transducers. These transducers employs Hall effect technology to obtain high bandwidth measurements of instantaneous voltage and current waveforms.

❷ Sampled analogue data is transferred from the sensor board to the data acquisition system board which has 8 input channels. The analogue data is then converted to its digital value using a 12 bit analogue to digital converter (ADC) AD7572 which has a 3 μs conversion time.

❸ Once all the analogue signals have been converted, the digital data is then transferred to the DSP. An appropriate scaling factor is introduced to obtain a real value for bipolar operation. Using instantaneous reactive power theory (as explained in Chapter 2) the reactive power value can be calculated. The value of m (modulation index) and ϕ (phase shift angle) is then determined from equations (3.10) and (3.11) to:

i) maintain a constant DC voltage across the capacitor

ii) provide the required VAr compensation

❹ From the information obtained earlier, the DSP will decide the modulating waveforms that will be transferred back to the Xilinx block. To generate the waveform, the DSP program utilised the full cycle sine and cosine look-up tables which are available in the on-chip ROM of the DSP.

❺ The 3 phase modulating waveforms (each 10 bits) are transferred to the Xilinx FPGA in order to generate the required PWM signals. The PWM signals are obtained by comparing the modulating waveforms with a triangular waveform. A digital circuit then inserts the underlap time in order to prevent shoot through conduction in the power inverter circuit.

❻ The gating signals are transferred to the gate drive circuits via optocouplers. The gate drive circuits provide isolation and ensure that the correct drive voltage is provided to turn the power switches on or off.

4.3 DESIGN FOR THE XILINX FPGA

The FPGA device must perform the two main functions below as specified in the previous section.

i) state machine to control data acquisition and flow of data processed

ii) PWM signals generation

4.3.1 State Machine Design

As described in the previous sections, the data flow is processed in different stages. This sequence must be maintained in order to prevent any software and hardware conflict or loss of data. Different hardware and software control routines can be synchronised by

PRESENT STATE					INPUT										NEXT STATE				OUTPUT					
	D	C	B	A	BUSY	A2	A1	A0	DC3	DC2	DC1	DC0	DSPRD	PWMSYNCH	D	C	B	A	RD	RD/WR	INCA	DCLR	S/H	IRQA
S1	0	0	0	0	X	X	X	X	X	X	X	X	1	X	0	0	0	0	1	1	0	1	1	0
S1	0	0	0	0	X	X	X	X	X	X	X	X	0	0	0	0	0	0	1	1	0	1	1	0
S1	0	0	0	0	X	X	X	X	X	X	X	X	0	1	0	0	1	1	1	1	0	1	1	0
S2	0	0	1	1	X	X	X	X	0	X	X	X	X	X	0	0	1	1	1	1	0	0	0	0
S2	0	0	1	1	X	X	X	X	1	0	X	X	X	X	0	0	1	1	1	1	0	0	0	0
S2	0	0	1	1	X	X	X	X	1	1	0	X	X	X	0	0	1	1	1	1	0	0	0	0
S2	0	0	1	1	X	X	X	X	1	1	1	1	X	X	0	0	1	1	1	1	0	0	0	0
S2	0	0	1	1	X	X	X	X	1	1	1	0	X	X	1	0	1	1	1	1	0	0	0	0
S3	1	0	1	1	X	X	X	X	1	X	X	X	X	X	1	0	1	1	1	1	0	0	1	0
S3	1	0	1	1	X	X	X	X	0	1	X	X	X	X	1	0	1	1	1	1	0	0	1	0
S3	1	0	1	1	X	X	X	X	0	0	0	X	X	X	1	0	1	1	1	1	0	0	1	0
S3	1	0	1	1	X	X	X	X	0	0	1	1	X	X	1	0	1	1	1	1	0	0	1	0
S3	1	0	1	1	X	X	X	X	0	0	1	0	X	X	1	1	1	1	1	1	0	0	1	0
S4	1	1	1	1	X	X	X	X	X	X	X	X	X	X	0	1	1	1	0	1	0	0	1	0
S5	0	1	1	1	X	X	X	X	X	X	X	X	X	X	0	1	1	0	0	1	0	0	1	0
S6	0	1	1	0	0	X	X	X	X	X	X	X	X	X	0	1	1	0	0	1	0	0	1	0
S6	0	1	1	0	1	X	X	X	X	X	X	X	X	X	0	1	0	0	0	1	0	0	1	0

	PRESENT STATE					INPUT											NEXT STATE					OUTPUT					
	D	C	B	A		BUSY	A2	A1	A0	DC3	DC2	DC1	DC0	DSPRD	PWMSYNCH		D	C	B	A		RD	RD/WR	INCA	DCLR	S/H	IRQA
S7	0	1	0	0		X	X	X	X	X	X	X	X	X	X		1	1	0	0		0	0	0	0	1	1
S8	1	1	0	0		X	0	X	X	X	X	X	X	X	X		1	1	1	0		1	1	1	1	1	0
S8	1	1	0	0		X	X	0	X	X	X	X	X	X	X		1	1	1	0		1	1	1	1	1	0
S8	1	1	0	0		X	X	X	0	X	X	X	X	X	X		1	1	1	0		1	1	1	1	1	0
S8	1	1	0	0		X	1	1	1	X	X	X	X	X	X		1	0	0	0		1	1	1	1	1	0
S9	1	0	0	0		X	X	X	X	X	X	X	X	X	X		0	0	0	0		1	1	0	0	1	1
S10	1	1	1	0		X	X	X	X	X	X	0	X	X	X		1	1	1	0		1	1	0	0	1	0
S10	1	1	1	0		X	X	X	X	X	X	1	0	X	X		1	1	1	1		1	1	0	0	1	0

X : don't care

Table 4.1 State table to implement the desired control sequence in Xilinx FPGA.

Symbol	Function
A, B, C, D	designation letter for decoder circuits
BUSY	A/D is busy (active low)
A0, A1, A2	address counter for RAM
DC0, DC1, DC2, DC3	delay counter to ensure time constraint is met. The counter run continously
DSPRD	signal from the DSP to inform that it is ready for new signal
PWMSYNCH	command signal to ensure that data is sampled for a period of 5 μs (acquisition time) before carrier waveform reaches its peak
RD	signal sent to A/D to activate start of data conversion
RD/WR	signal to inform whether to read (RD - active high) or write (WR - active low) data from/to RAM
INCA	increase address counter
DCLR	reset delay counter
S/H	sample (active low) and hold (active high) signal
IRQA	interrupt request signal to DSP (active low)

Table 4.2 Different function for symbols used in the state table.

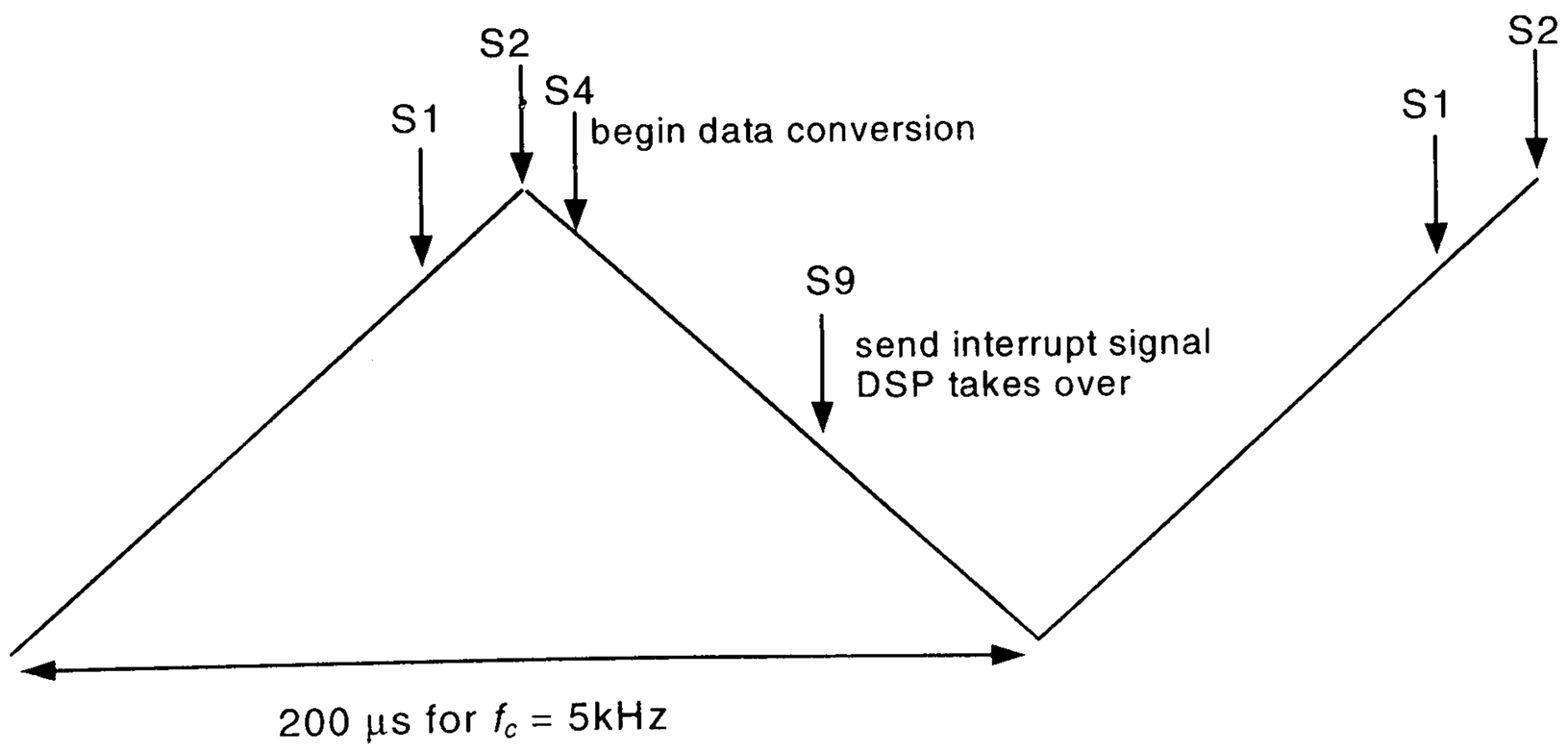


Figure 4.2 Timing diagram for state assignments

State	Description
S1	Idle state. Reset delay counter. Wait for DSPRD and PWMSYNCH signal before begin sampling
S2	Begin analogue data sampling. Set S/H - low. Activate delay counter. Move to the next state after 5 μs
S3	Hold data. Set S/H - high. Move to the next state after 1.5 μs
S4	Begin A/D data conversion. Set RD - low.
S5	Ensure A/D has enough time to start conversion. Set RD - low.
S6	Set RD - low. Wait until conversion process has finished. When BUSY signal is high, move to the next state
S7	Transfer digital data to RAM. Set RD/WR - low.
S8	Reset delay counter. Increase address counter. If less than 8 (111), move to S10 to begin another conversion process. If counter is 8 (111) move to S9 to initiate data process in the DSP
S9	Send IRQA (active low) signal to the DSP. Move back to S1.
S10	Set enough time (700 ns) to acquire new signal before commencing another data conversion

Table 4.3 Brief description for different state as defined in the state table

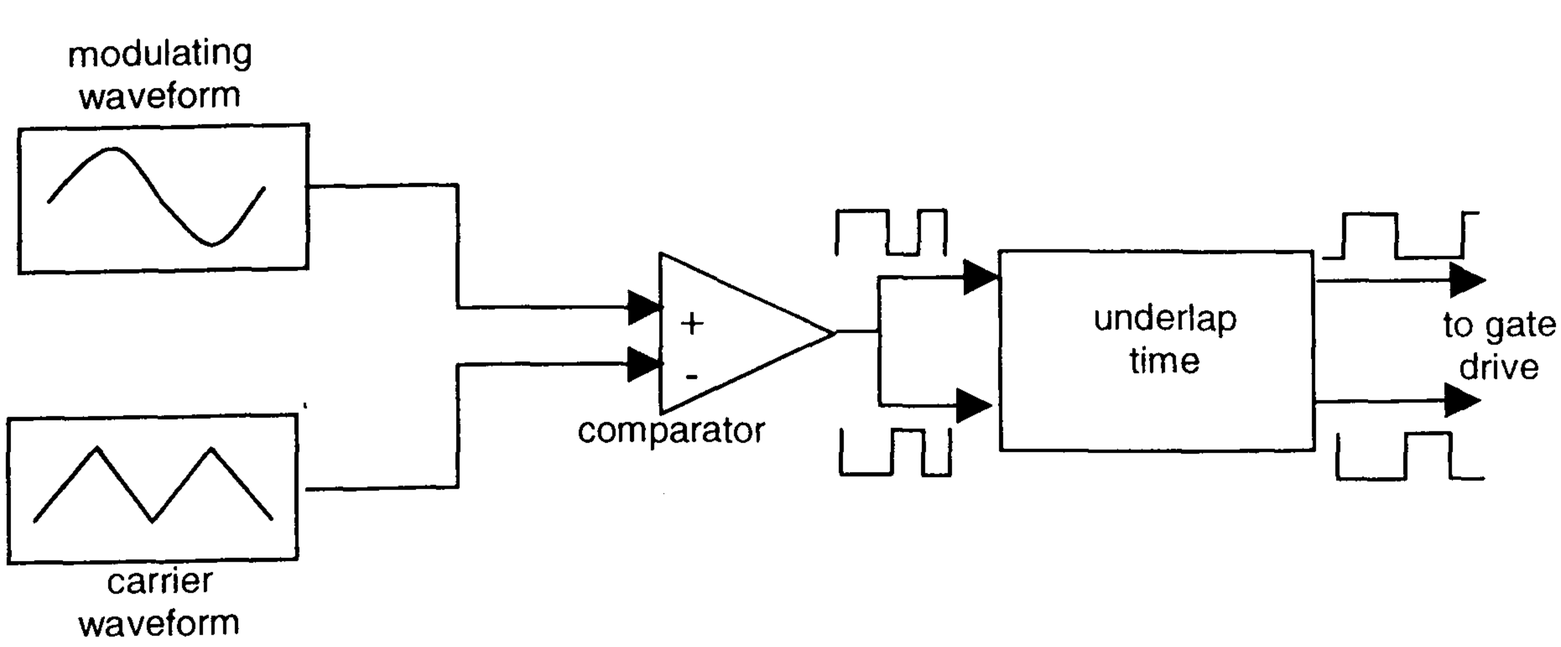


Figure 4.3 Flow diagram for PWM signals generation

sending interrupt and control signals to and from the DSP. These control signals are monitored and generated by a state machine configured within the Xilinx FPGA.

One technique to implement the desired control sequence is by using combinations of D type flip-flops and decoder circuits [4.1]. The input and output functions can be obtained directly from the state table without the need of an excitation table. The state table consists of lists of present states and inputs and their corresponding next states and outputs as shown in Table 4.1. Symbol definitions are given in Table 4.2. Table 4.3 gives a brief description of the ten different states used in the control sequence. The PWM system is designed to operate at a switching frequency of 5 kHz. In Figure 4.2, the timing diagram shows the position of these different states within one cycle of the carrier waveform.

4.3.2 PWM Signals Generation

Once the three phase modulating waveform has been determined by the DSP program, the waveform will be transferred to the Xilinx device one phase at a time. Each is compared with a common triangular waveform to produce gating signals for all six switches in the three phase inverter. The triangular waveform is designed using an up-down counter. The same clocking signal (40 MHz) is used for both the DSP and Xilinx board. The carrier frequency, f_c, is defined by the following equation :

$$f_c = \frac{f_{clk}}{2^{n+1}} \tag{4.1}$$

where f_{clk} is the clock frequency and n is the number of bits.

For 10 bit PWM operation, this gives a switching frequency of 4.88 kHz. Figure 4.3 shows the processes involved in creating the PWM signals.

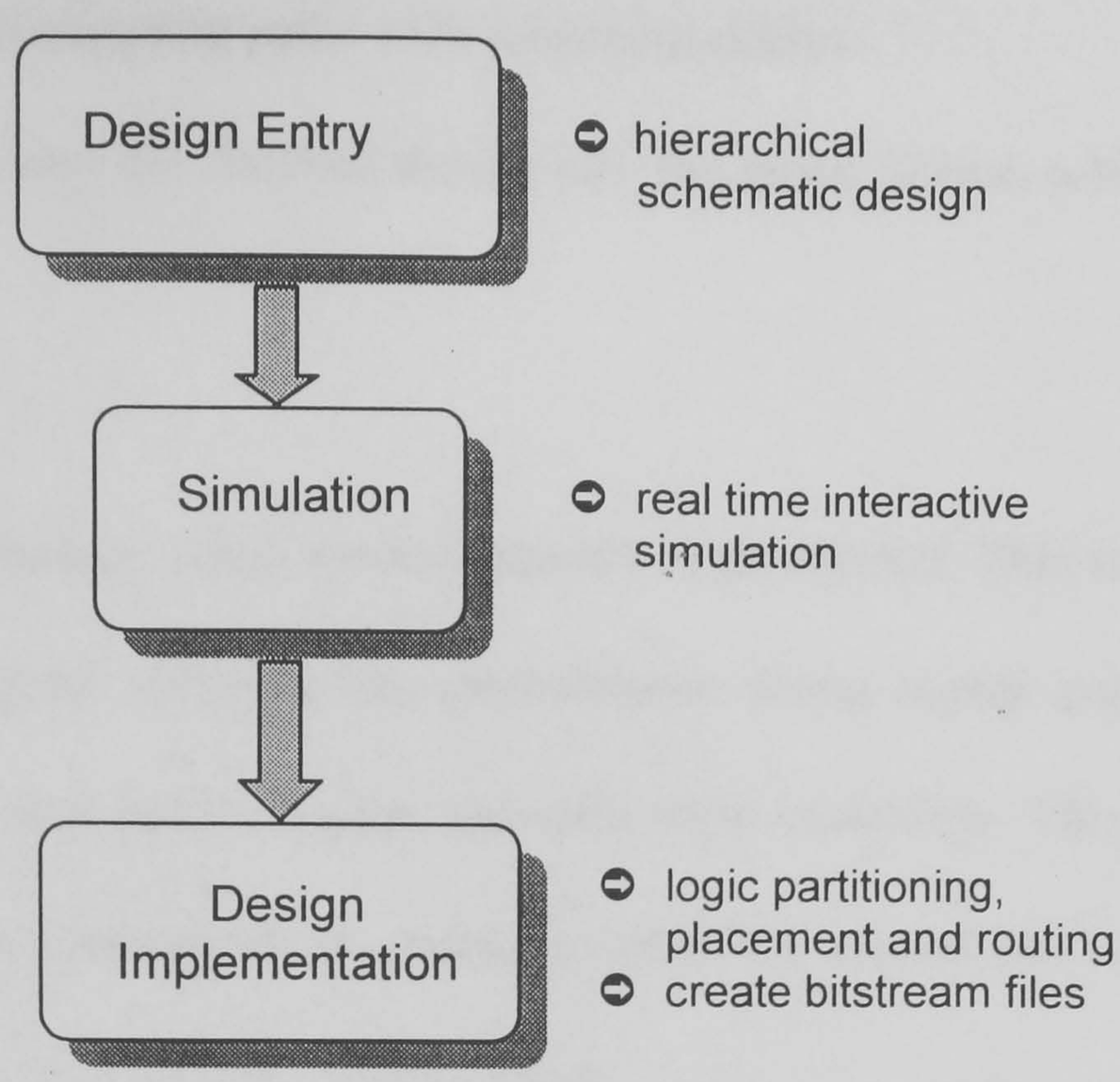


Figure 4.4 Design flow for Xilinx FPGA

4.3.3 Design Implementation

Figure 4.4 shows the design procedure used to implement the desired control sequence in the Xilinx FPGA. The design is first created using the Active-CAD Schematic Editor. To cater for the large design used, a hierarchical design structure is applied by dividing the design into smaller sections. In order to verify the designed schematics, a real-time interactive analysis tool is used to obtain simulation results. Various stimulus signals can be applied to ensure that the design is able to operate as intended. The design can be implemented in the Xilinx FPGA device by using the XACT (Xilinx Automated CAE Tools) Step Design Manager. This involves the following processes [4.2, 4.3]:

i. the translated design is optimised and mapped into the configurable logic blocks (CLB) and input output blocks (IOB).

ii. the mapped CLBs and IOBs are then placed inside the target device so that a minimum amount of routing is needed to connect them.

iii. the design is then routed automatically using a routing algorithm which determines interconnecting paths with minimum delays.

iv. in order to download the finished design into the target device, a bitstream file is generated.

To verify the completed design, static timing analysis is performed. This examines the design's logic and timing to calculate the performance along signal paths, identify possible race conditions and detects set-up and hold time violations. This is done by using the back annotation function which creates a file of the placed and routed design that includes timing data for logic and routing delays.

The design is implemented using Xilinx XC4003-5 PC84. The device utilisation is shown below :

```
64% utilization of I/O pins.                    ( 39  of  61)
82% utilization of CLB FG function generators. (164  of 200)
45% utilization of CLB H function generators.  ( 45  of 100)
58% utilization of CLB flip-flops.             (116  of 200)
```

The schematic design to implement the state machine in the Xilinx device is as shown in Appendix D.1. Appendix D.2 shows the main top level schematic design for the state machine and PWM signals generation.

4.4 DSP PROGRAM

As described in Section 4.3 (steps ❸ and ❹), the DSP96002 must be programmed to transfer sampled data from external memory and generate modulating waveforms from the information obtained. The listing for the assembly language program is included in

Appendix E.2. This program is used to implement open loop tests to compare the performance of the SVC with sinusoidal PWM and deadband PWM.

4.5 SIMULATION RESULTS

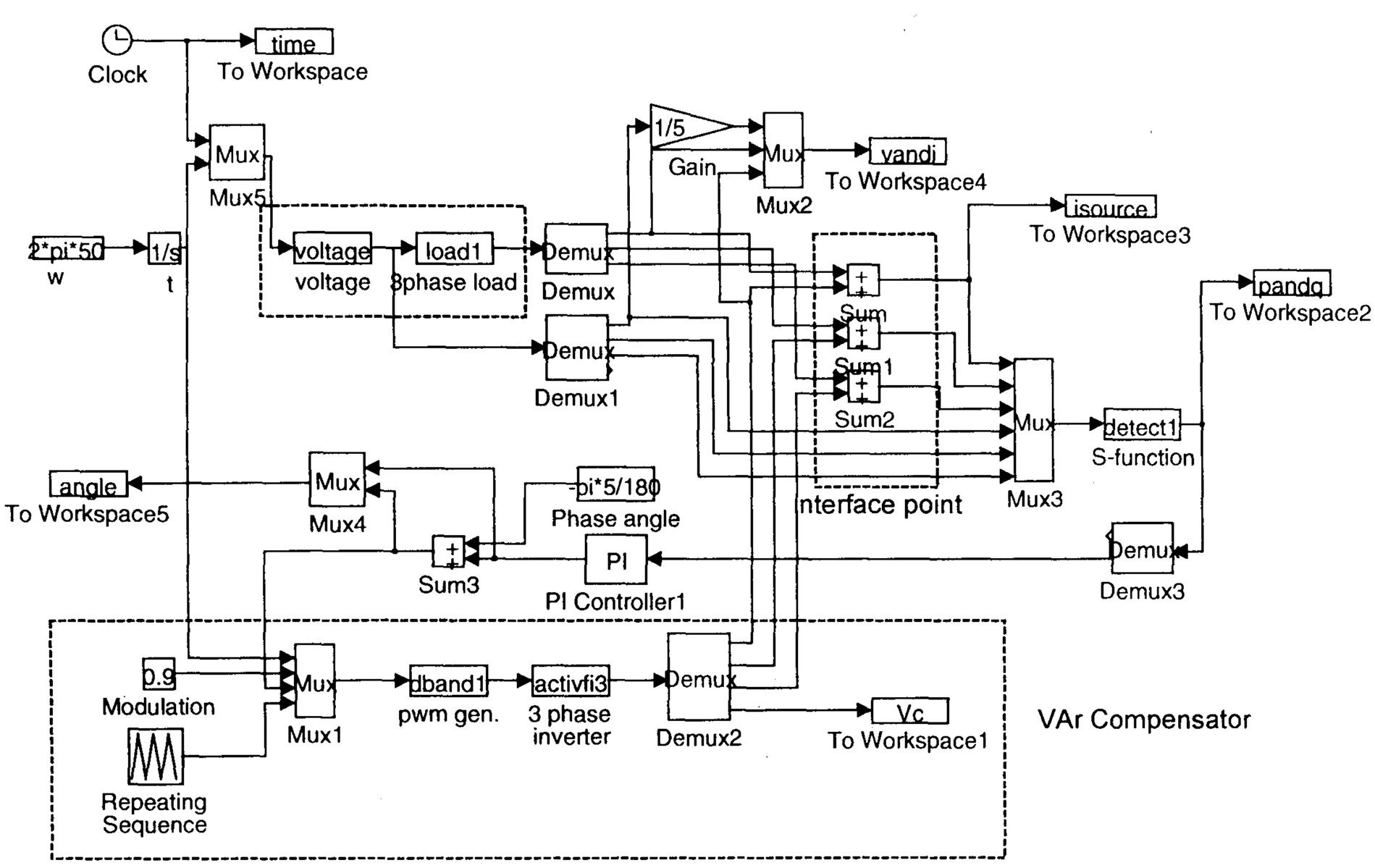
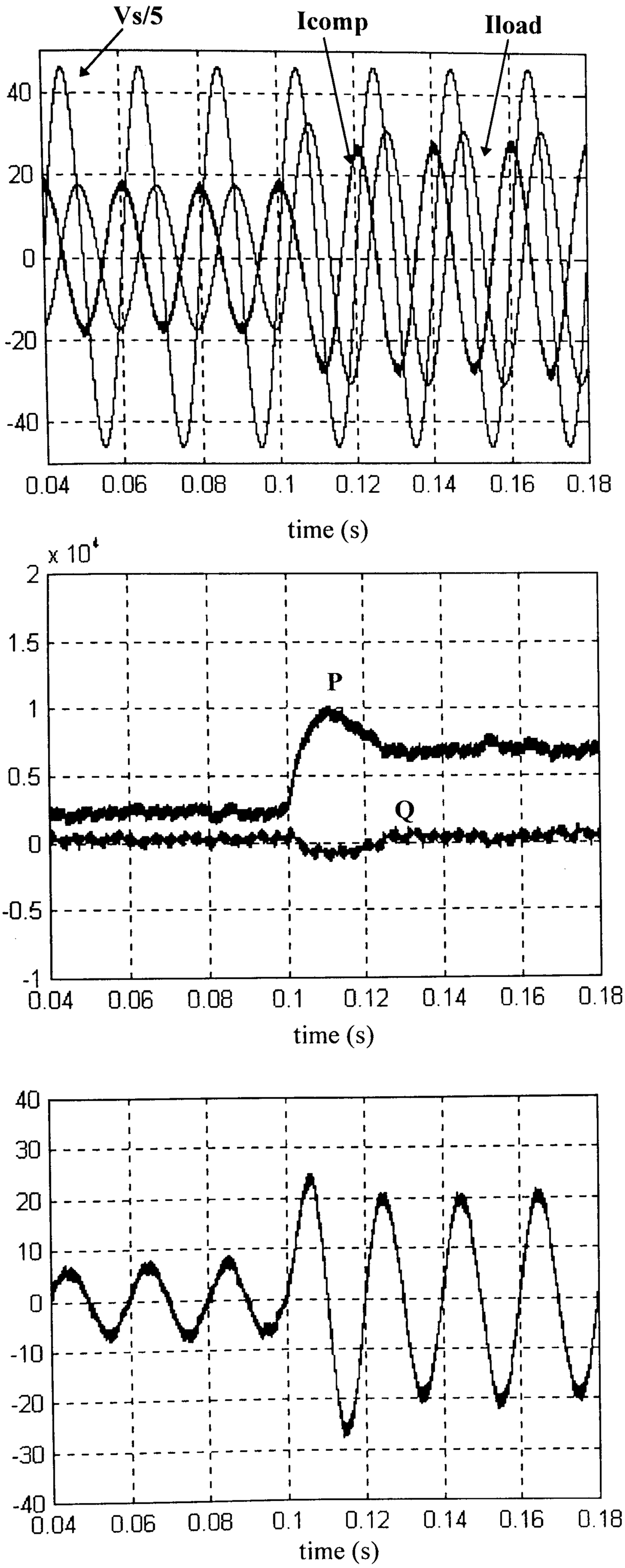


Figure 4.5 Simulink model for VAr compensation

Figure 4.5 shows the Simulink model used to simulate the SVC system with a three phase inductive load. In an ideal compensator the supply system will only provide active power, P, and any reactive power, Q, will be stored or supplied by the compensator. In the model, an 'S-function' is written to determine the reactive power required to perform the necessary compensation. A simple PI controller is used to control the phase shift angle to provide either leading or lagging VAr compensation. The results are as shown in Figure 4.6. To compensate a system with an inductive load, the SVC will supply leading VAr (4.6a), thus ensuring that only active power, P is present in the system



(a) Supply voltage (Vs), load current (Iload) and compensator current (Icomp)

(b) P (active power) and Q (reactive power)

(c) Supply current, Isupply

Figure 4.6 SVC simulation results with lagging VAr compensation

(4.6b). Figure 4.6(c) shows that the supply current is maintained in phase with the supply voltage, i.e. unity power factor.

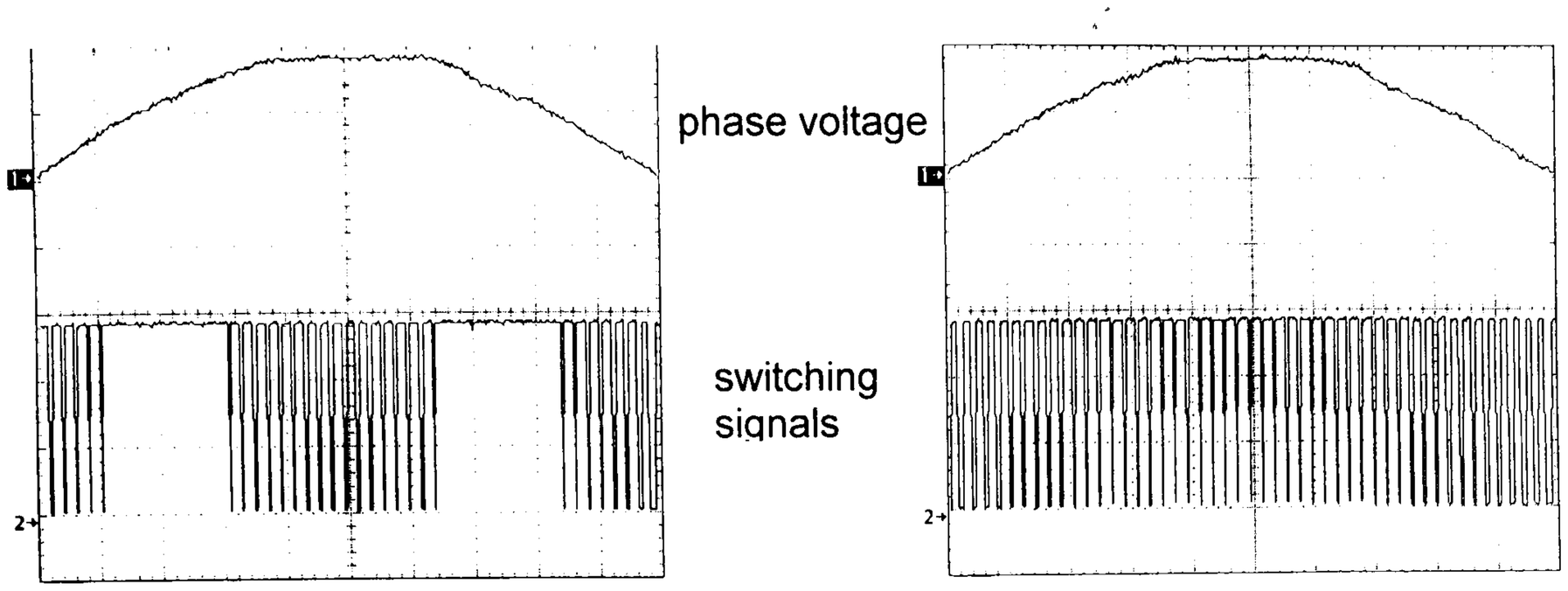
4.6 EXPERIMENTAL RESULTS

The SVC system was tested using both sinusoidal PWM and deadband PWM. For VAr compensation, the line currents will lead or lag the voltages by 90°. As described in Chapter 3, for such operation the clamping function in deadband PWM occurs on the phase leg carrying the second highest line. Figure 4.7 shows the switching pattern (for half of a cycle) with m = 0.8. It can be seen that with deadband PWM the switch remains 'on' for 1/3 a cycle (the modulating waveform is as shown in Figure 3.5(d)).

To demonstrate the validity of the predicted advantages for the deadband PWM technique, open loop tests were performed on the system. As a benchmark, the results obtained using deadband PWM are compared with conventional sinusoidal PWM. Figures 4.8 and 4.9 show the line current spectrum and demonstrate the ability of the system to provide leading and lagging VAr compensation. The figures show the typical harmonic spectrum for PWM where harmonics exist as sidebands, centred around f_c and its multiples.

In order to study system performance, different graphs are plotted (Figures 4.10 and 4.11) to observe the following parameters: dc bus voltage, line current, supply voltage and heat sink temperature rise (which represents switching plus on-state losses). The temperatures shown in Figures 4.10(c) and 4.11(c) are the values above ambient temperature (24 °C). To compare the performance of both PWM techniques, consider

the case for a line current of 20 A. The values for other parameters (obtained from the graph) are shown below in Table 4.4. The results (from the graphs and table) show that for a given line current, the deadband PWM system requires a lower dc voltage and generates a smaller heatsink temperature rise hence has lower switching losses.

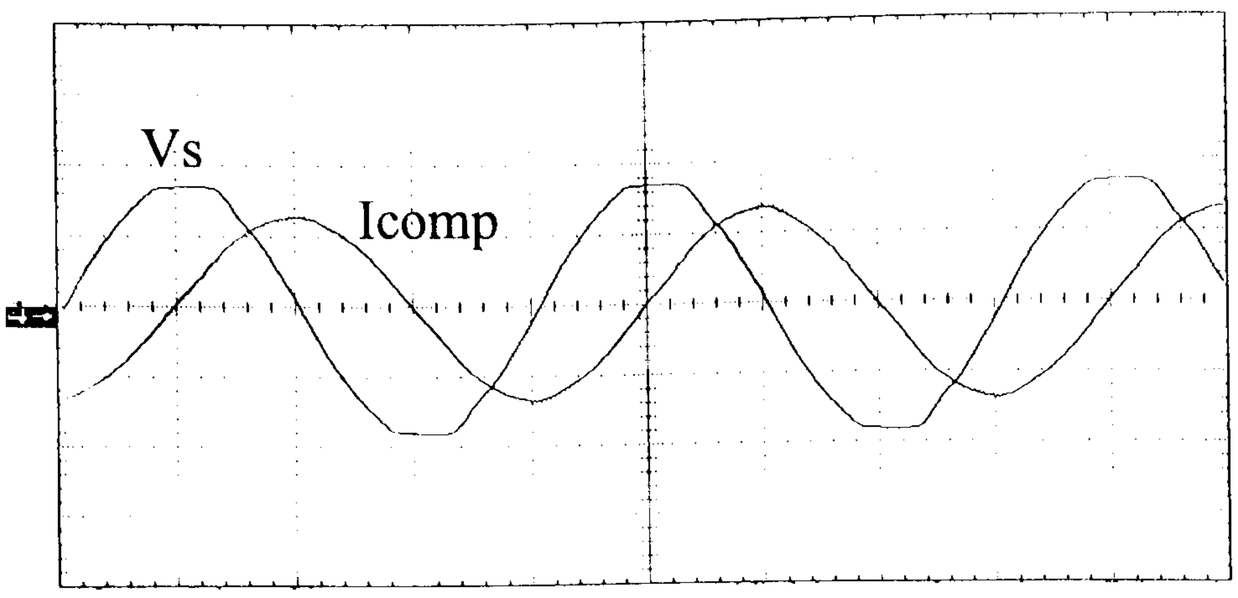


(a) deadband PWM (a) sinusoidal PWM

Figure 4.7 Switching pattern for VAr compensation

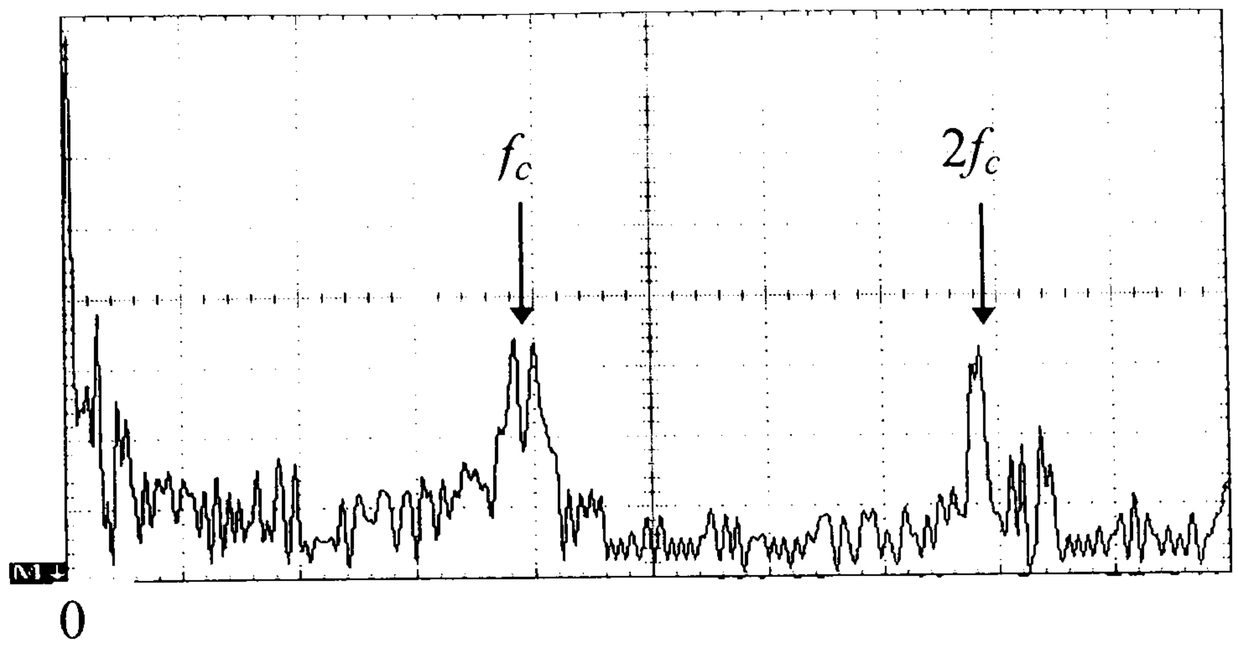
		sinusoidal PWM ①	deadband PWM ②	①/②
leading VAr	line current (A)	20	20	1
	Δ temperature (°C)	13.5	9.5	1.42
	dc voltage (V)	413.8	310.7	1.33
lagging VAr	line current (A)	20	20	1
	Δ temperature (°C)	13.6	9.8	1.38
	dc voltage (V)	256.8	182.5	1.40

Table 4.4. Comparisons between sinusoidal PWM and deadband PWM



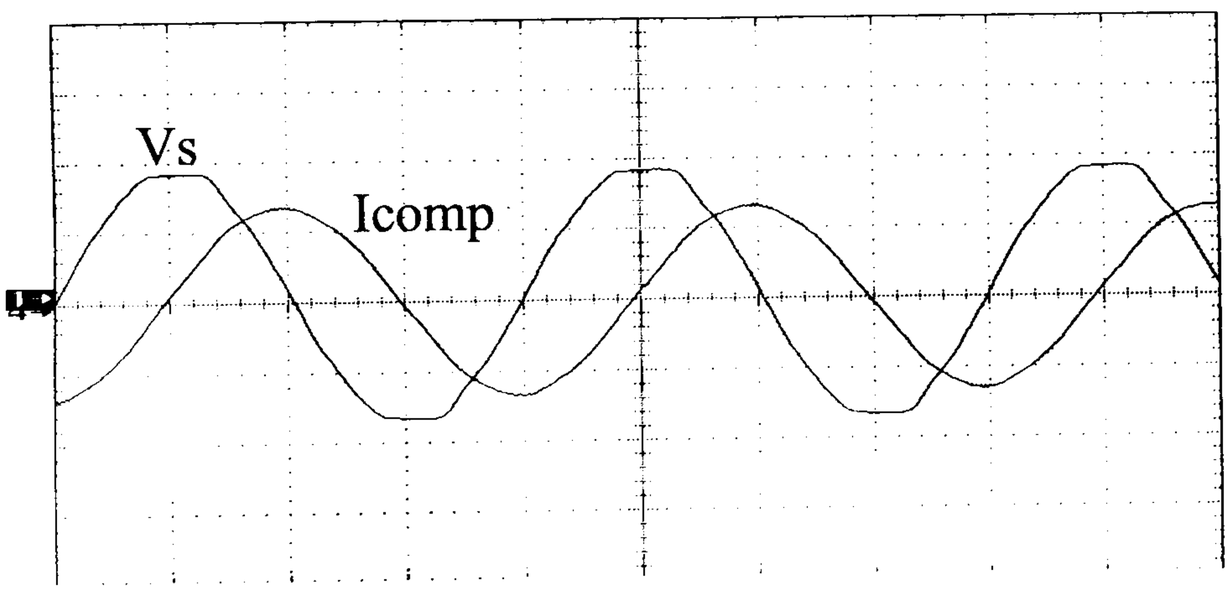
y-axis : 20 A/div
y-axis : 120 V/div

x-axis : 5 ms/div



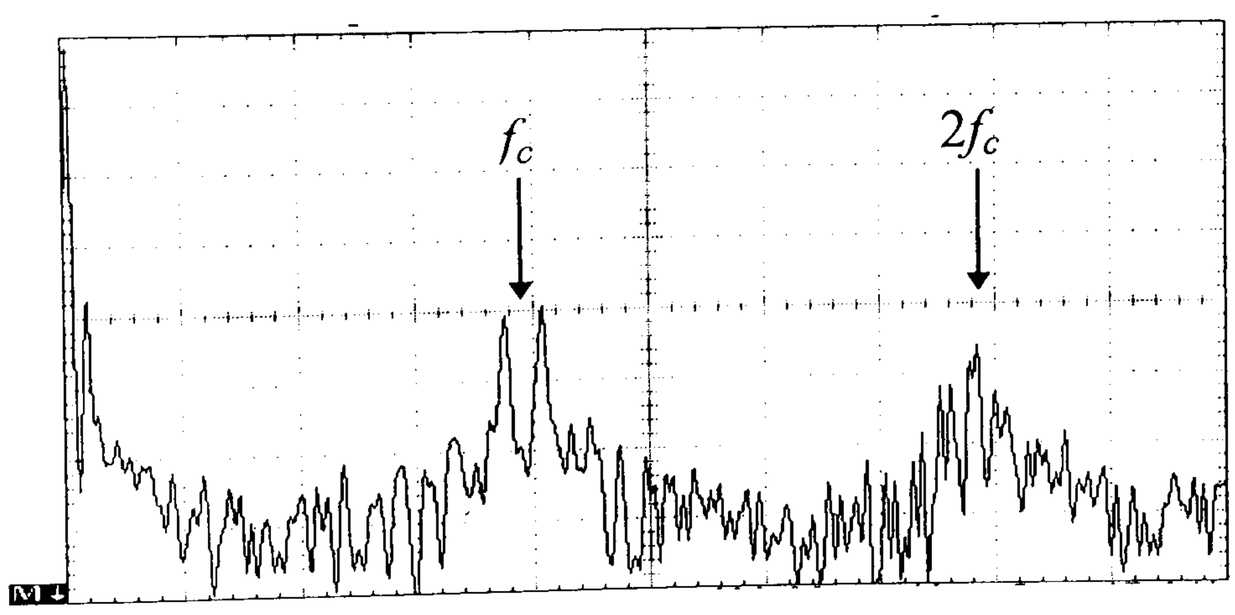
(a) sinusoidal PWM

y-axis : 10 dB/div
x-axis : 1.25 kHz/div



y-axis : 20 A/div
y-axis : 120 V/div

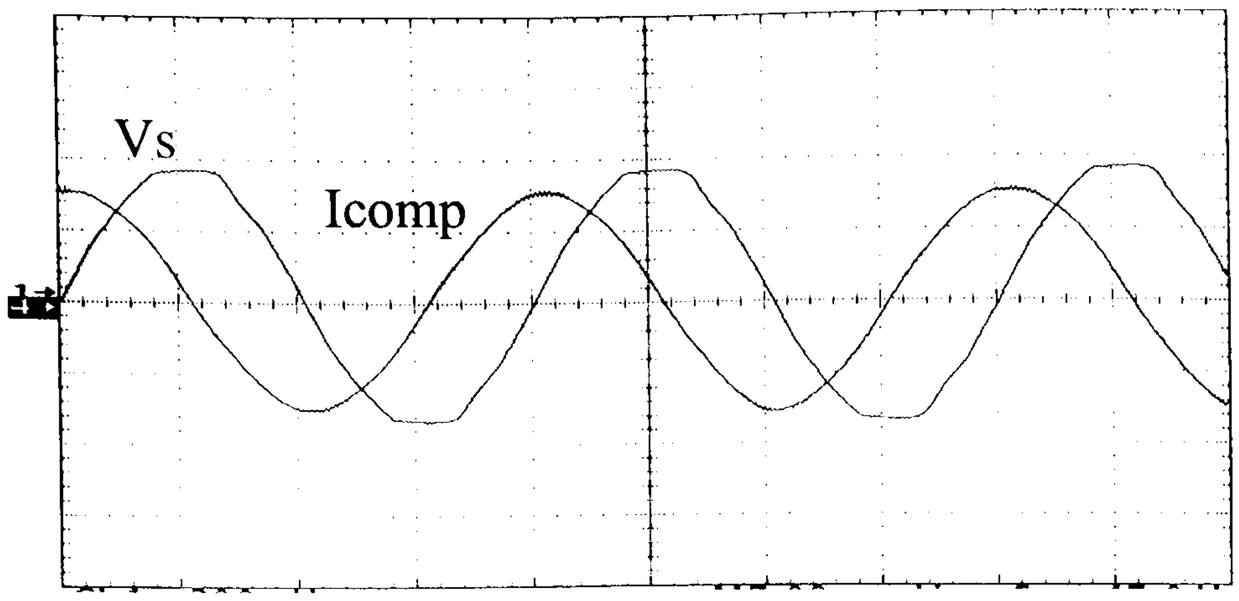
x-axis : 5 ms/div



(a) deadband PWM

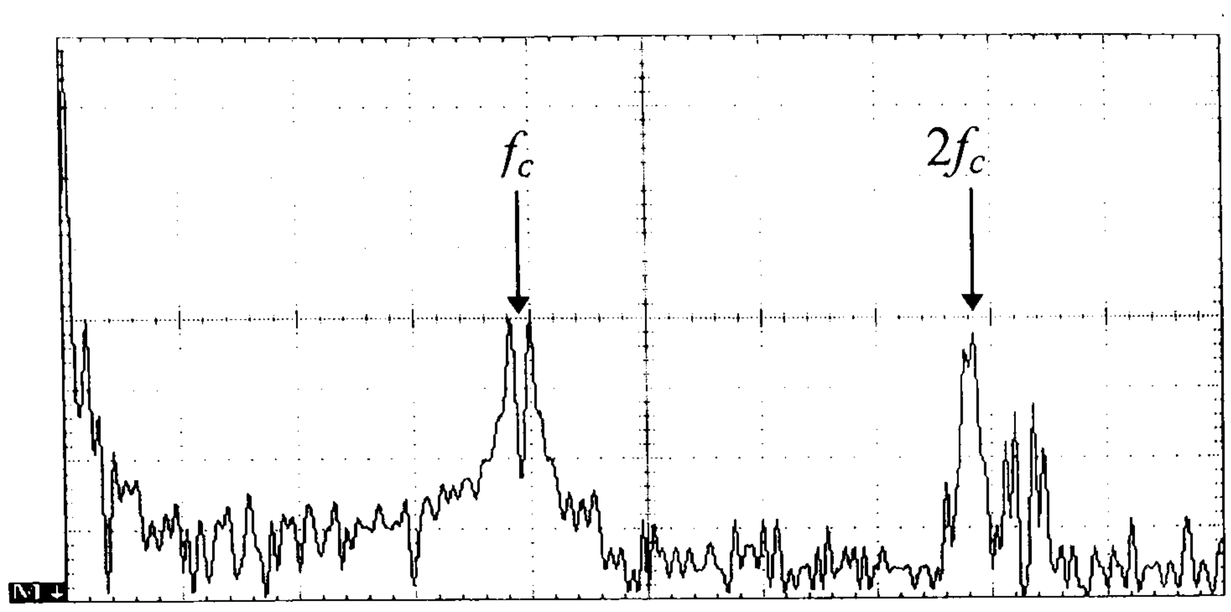
y-axis : 10 dB/div
x-axis : 1.25 kHz/div

Figure 4.8 Lagging VAr operation, showing supply voltage (Vs), compensator current (Icomp) and current spectrum

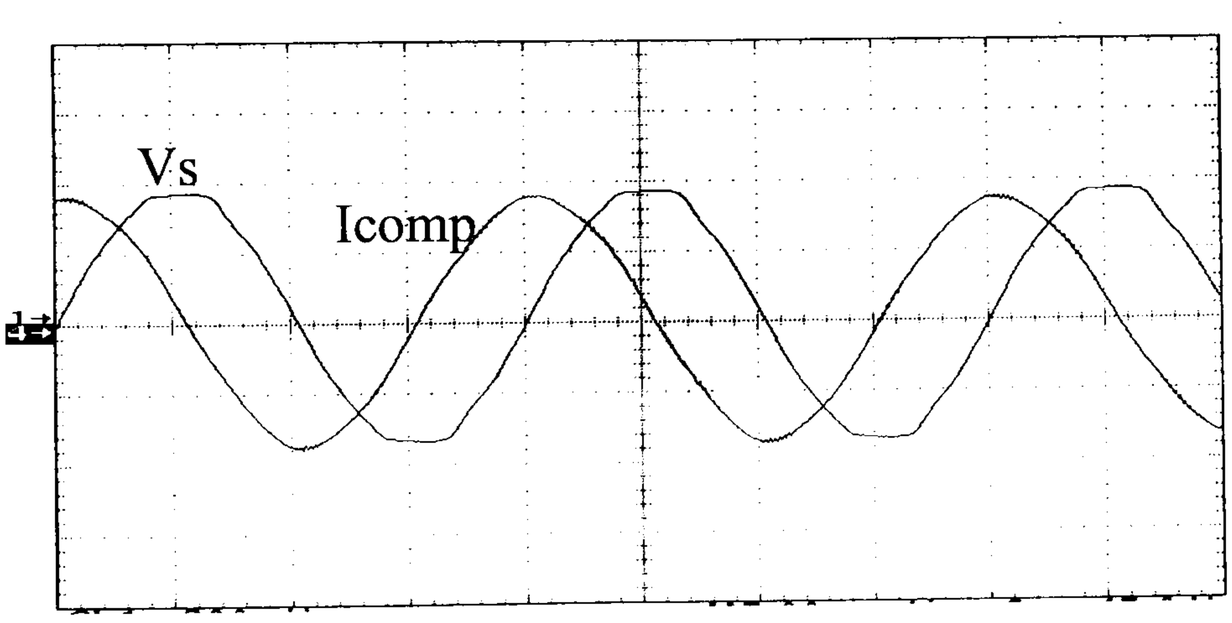


y-axis : 20 A/div
y-axis : 120 V/div

x-axis : 5 ms/div

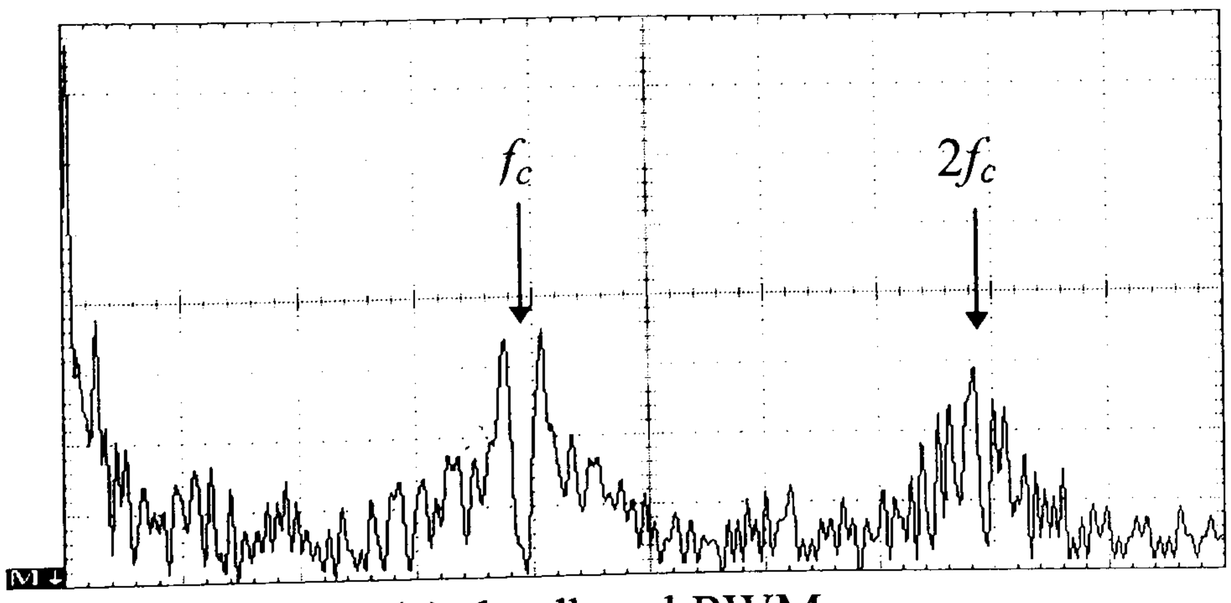


(a) sinusoidal PWM



y-axis : 20 A/div
y-axis : 120 V/div

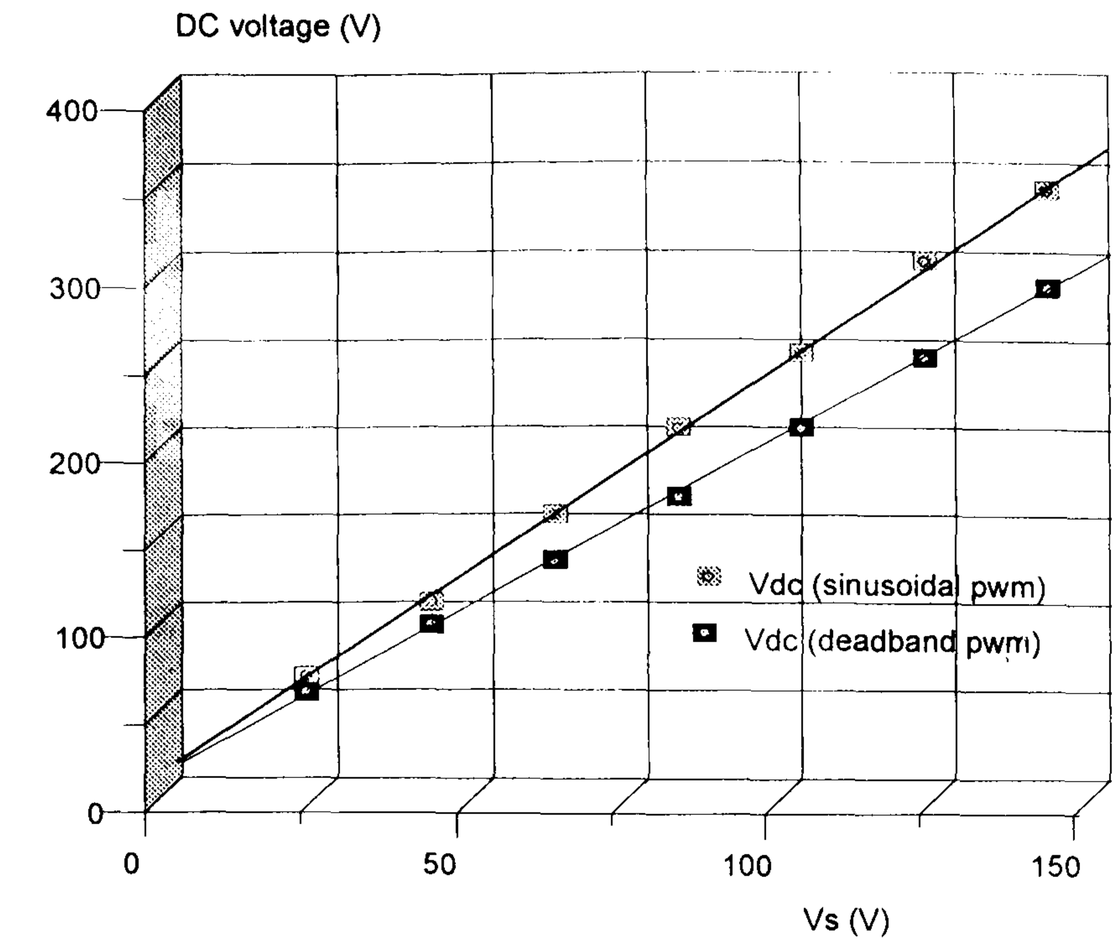
x-axis : 5 ms/div



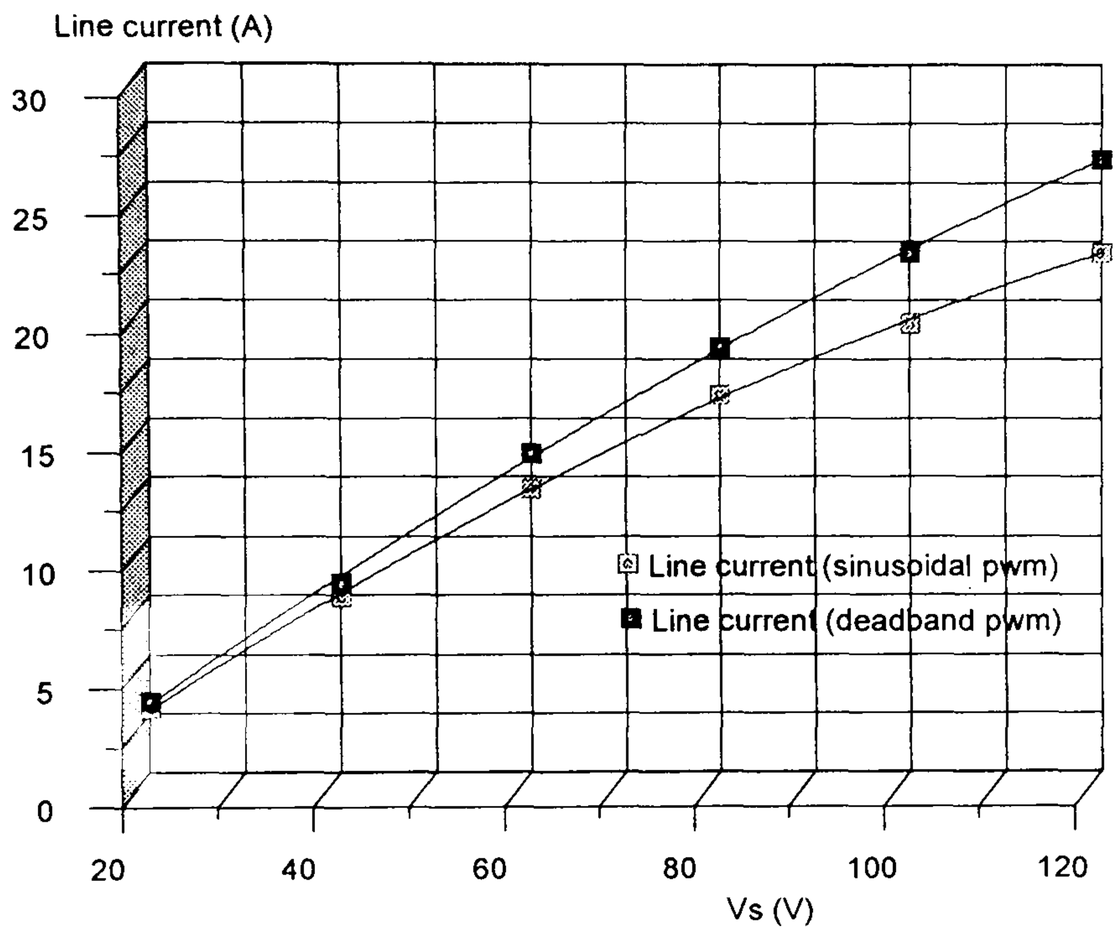
(a) deadband PWM

y-axis : 10 dB/div
x-axis : 1.25 kHz/div

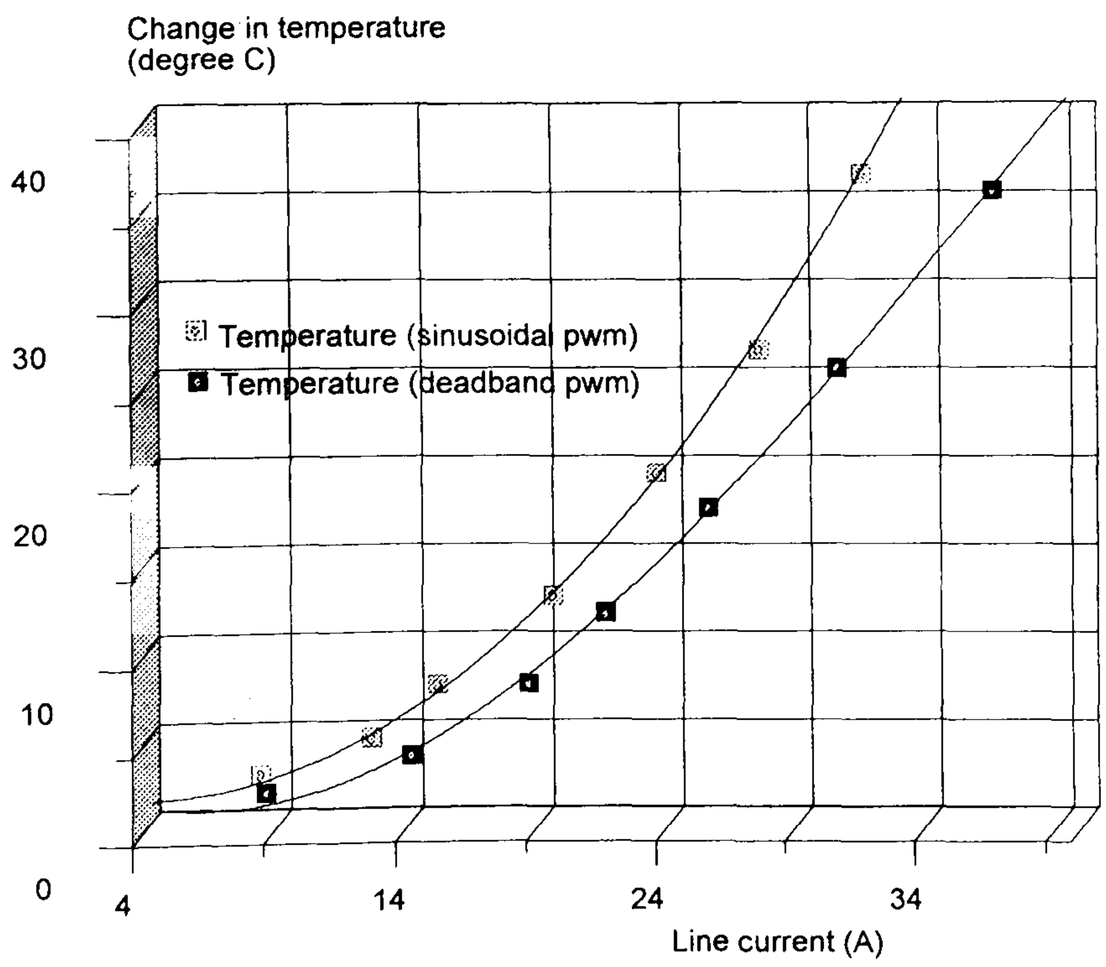
Figure 4.9 Leading VAr operation, showing voltage/current waveforms and line current spectrum



(a) DC voltage

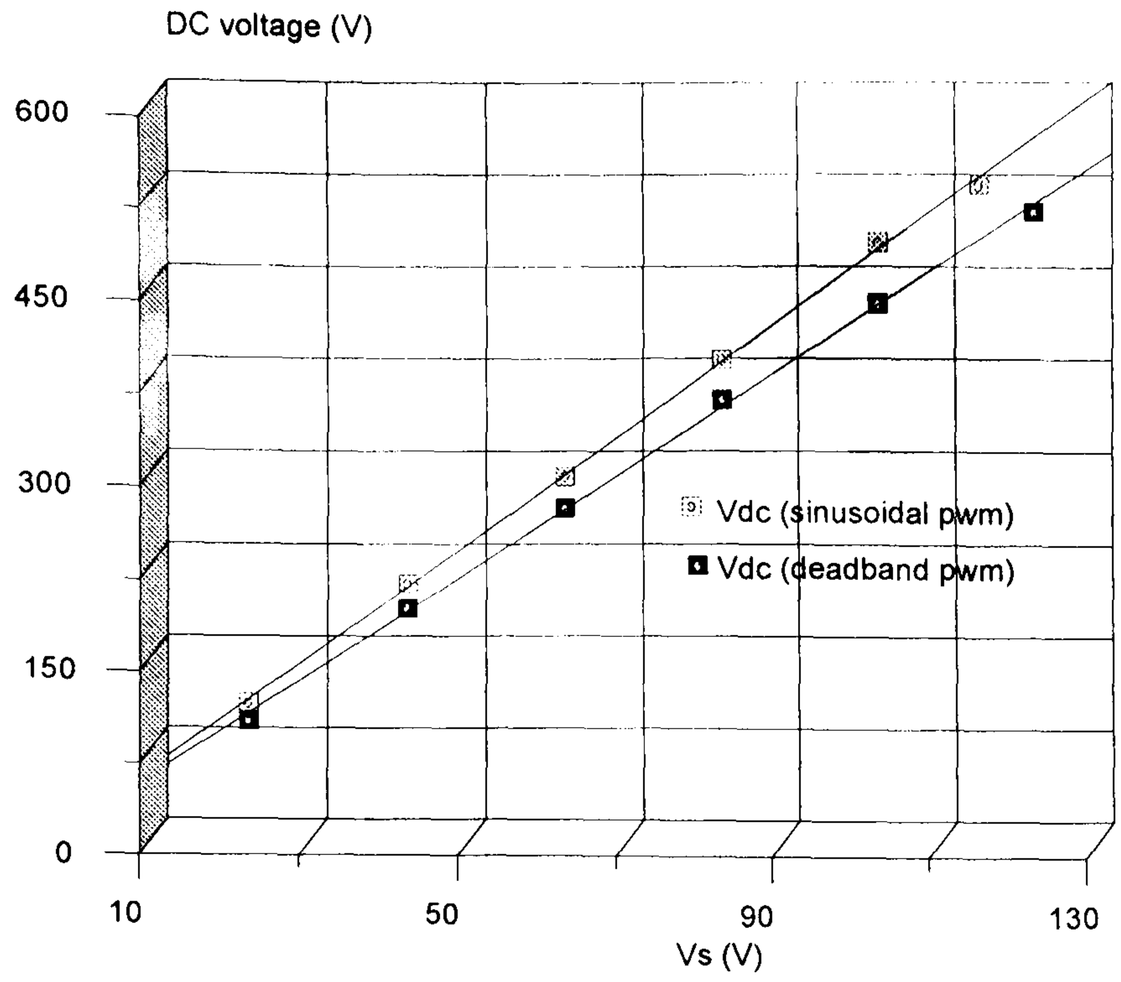


(b) Line current

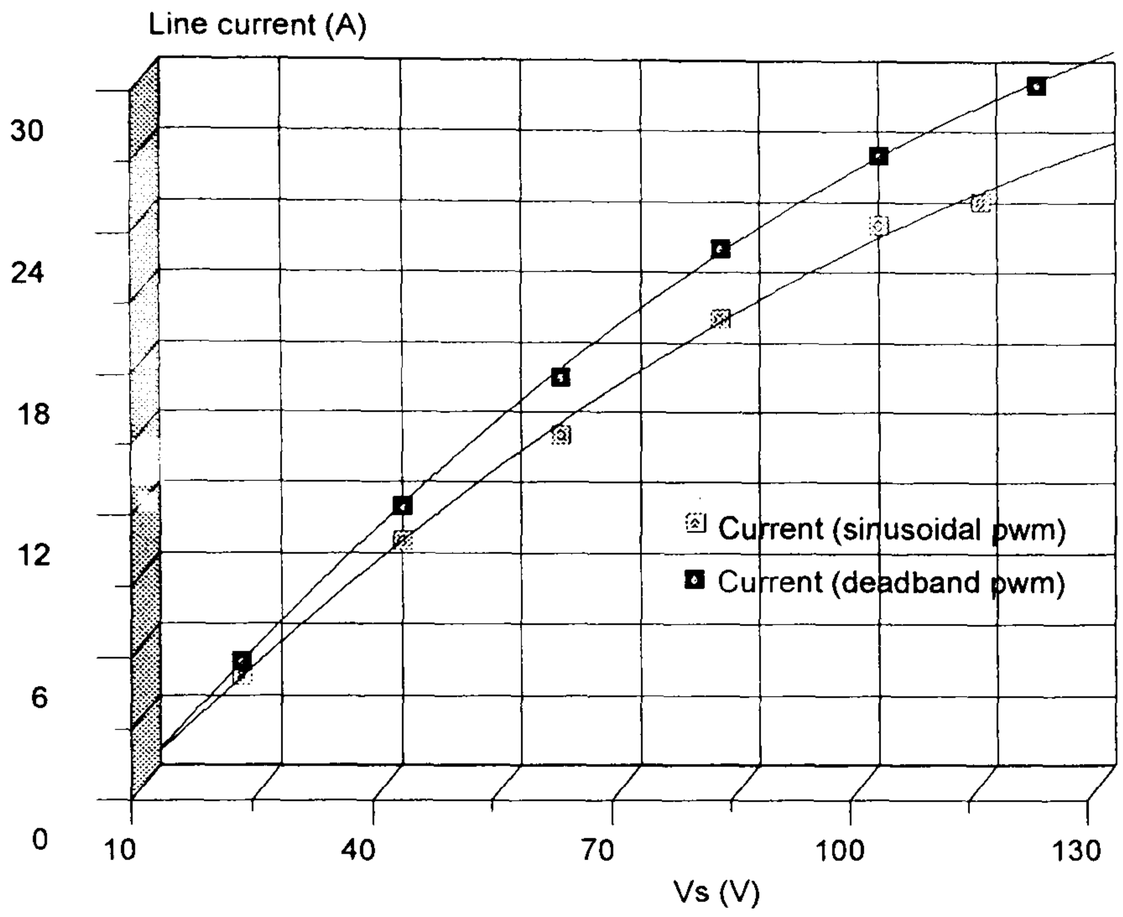


(c) change in heatsink temperature

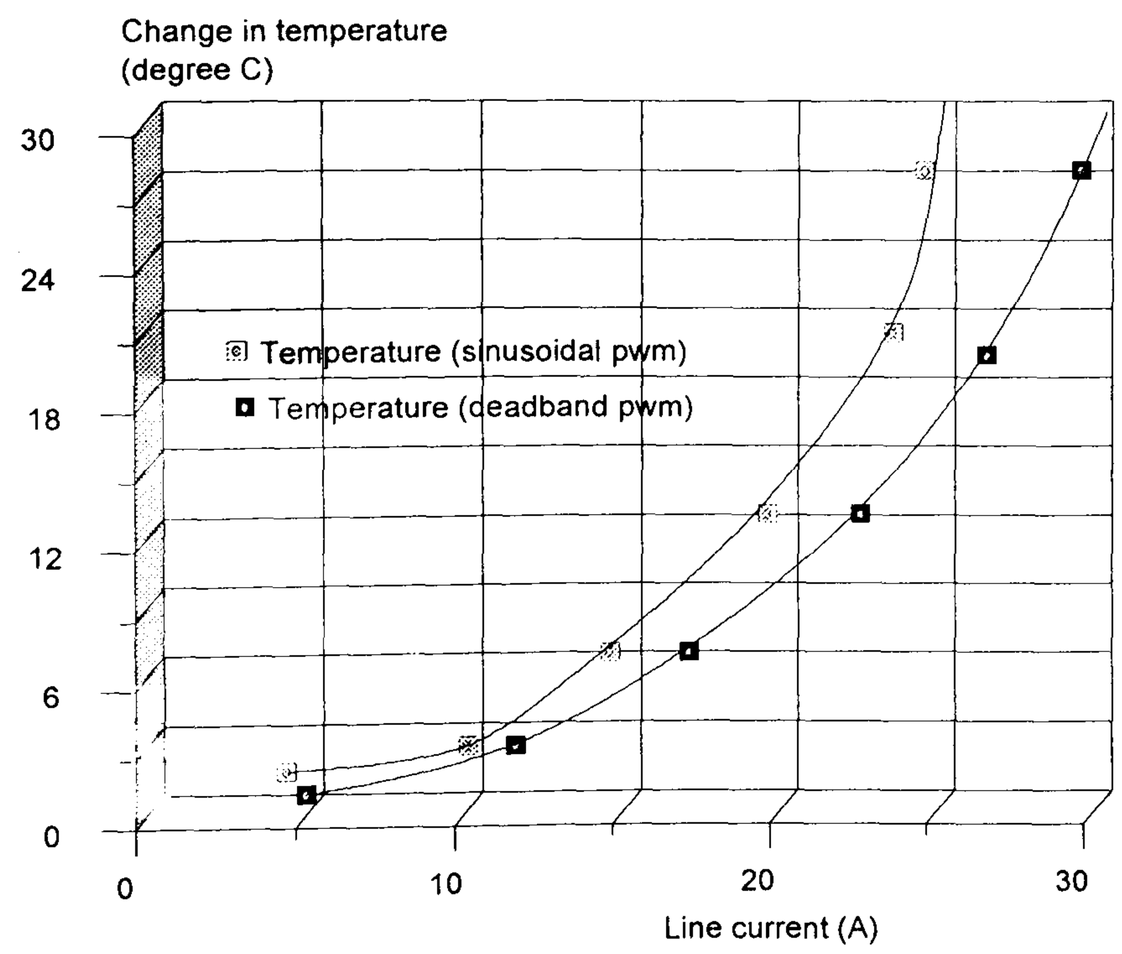
Figure 4.10. Graphs showing system performance for lagging VAr operation



(a) DC voltage



(b) Line current



(c) change in heatsink temperature

Figure 4.11. Graphs showing system performance for leading VAr operation

4.7 SUMMARY

This chapter presented the operation of the SVC system, demonstrating its ability to provide leading and lagging VAr compensation. The requirement of providing precise and continuous reactive power control is realised by adjusting the modulation index, m and the phase shift angle, ϕ, of the modulating waveform. The complex process of determining the active/reactive power and generating the control and switching signals is implemented using a Motorola DSP96002 and Xilinx FPGA.

The work is supported by simulation and experimental results. Comparison between results obtained with deadband PWM and sinusoidal PWM proved that the system is capable of providing leading and lagging reactive power compensation with a significant increase in the fundamental output voltage. This increased fundamental output voltage means that for a given SVC requirement, the DC capacitor voltage rating can be decreased thus allowing its physical size to be reduced. The use of deadband modulating waveforms reduced the switching losses and resulting stresses hence improves the efficiency of the SVC system. The same techniques can be applied to current controlled VAr compensation where a similar reduction in current source inductor stored energy is possible, with switch deadbanding.

4.8 REFERENCES

[4.1] M.M Mano, "Digital Design", Prentice Hall Inc. 1984.

[4.2] Developement System User Guide, XILINX, 1993.

[4.3] Schematic Editor User Guide, Active-CAD, ALDEC, 1996.

CHAPTER 5

INVESTIGATION OF ACTIVE POWER FILTER

5.1 INTRODUCTION

The development of modern solid state electronic devices has enabled efficient and safe control of electric power for example in AC/DC converters and adjustable speed motor drives. However these non-linear loads can also create undesirable effects by causing harmonic interference which might affect other electric components and cause additional power losses. These problems are partially solved by the use of fixed passive filters. However, these filters cannot cope with load variations and can also cause system resonance with the source impedance. These problems can be solved by using shunt active power filters which have been widely investigated and were considered in the literature survey in Chapter 2. These filters work as current sources which inject equal but opposite current waveforms to provide the required compensation.

The aim of this chapter is to present an active power filter (APF) system which is able to compensate both harmonics and reactive power caused by the non-linear loads. This ensures that the supply system will only need to provide sinusoidal current with unity power factor. The design of the APF system using a Motorola DSP and Xilinx FPGA is presented. A modified switching control strategy based on the deadbanding concept is proposed, which ensures that the switching losses are minimised. Simulation and experimental results are presented at the end of this chapter to prove the effectiveness of the APF system.

5.2 P AND Q DETECTION CIRCUIT

One important feature of the APF system is its ability to detect the instantaneous active and reactive power. This is to ensure that it can provide the required harmonic and reactive power compensation. Figure 5.1 shows the general overview of the processes involved in determining the required reference current to provide compensation.

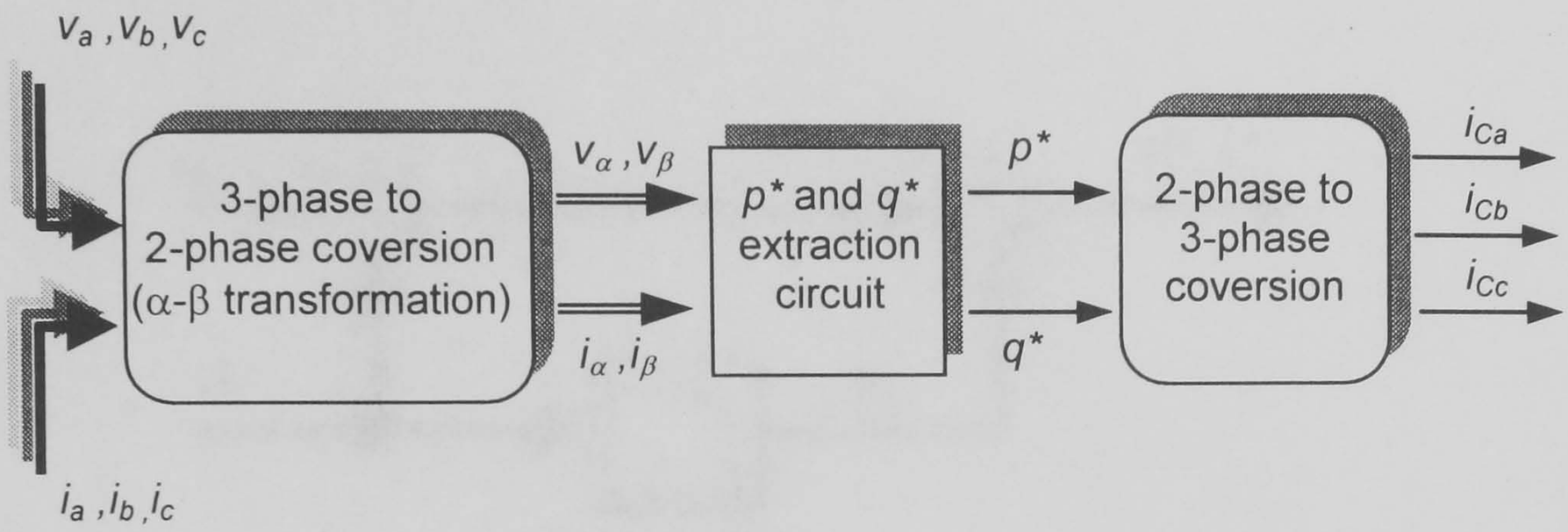


Figure 5.1 Determination of compensating current

As described in Section 2.5, the method developed by Akagi et al. is used [5.1]. The theory starts with equations (2.4) and (2.5) :

$$p_L = \bar{p}_L + \tilde{p}_L$$

$$q_L = \bar{q}_L + \tilde{q}_L$$

The compensating current that must be injected into the system is found by inverting equations (2.2) and (2.3) to give :

$$\begin{bmatrix} i_{Ca} \\ i_{Cb} \\ i_{Cc} \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_\alpha & v_\beta \\ -v_\beta & v_\alpha \end{bmatrix} \begin{bmatrix} p^* + p_{av} \\ q^* \end{bmatrix} \quad \textbf{(5.1)}$$

where i_{Ca}, i_{Cb} and i_{Cc} are the compensating current references, p_{av} is the real power necessary to maintain a constant DC capacitor voltage and p^* and q^* are the instantaneous active and reactive power that must be compensated to obtain a sinusoidal supply current with unity power factor. This is defined by the following equations :

$$\begin{aligned} p^* &= \tilde{p}_L \\ q^* &= \bar{q}_L + \tilde{q}_L \end{aligned} \qquad \textbf{(5.2)}$$

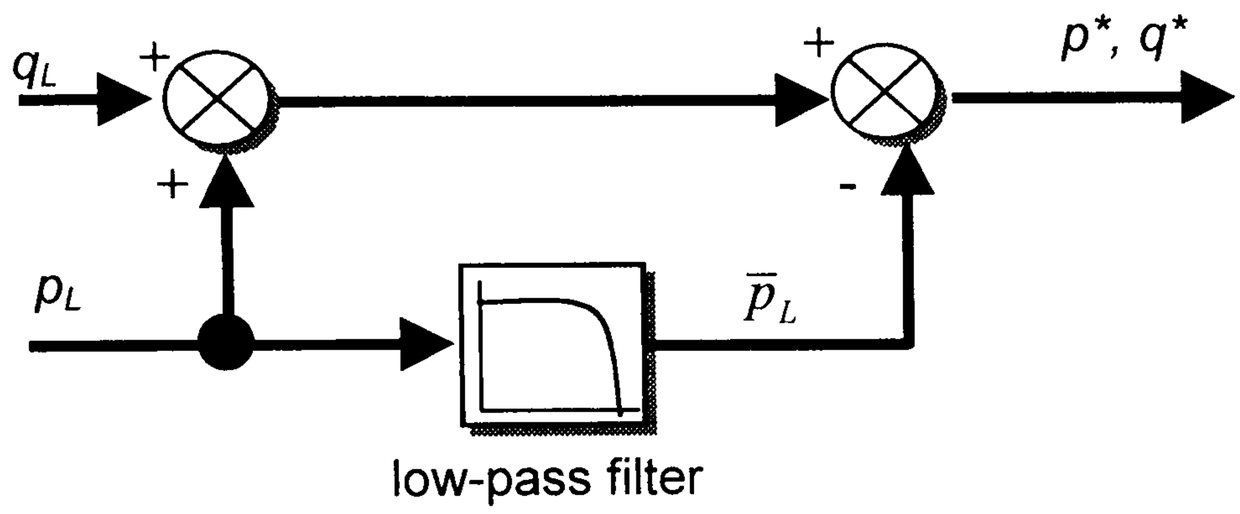


Figure 5.2 Method to determine p^* and q^*

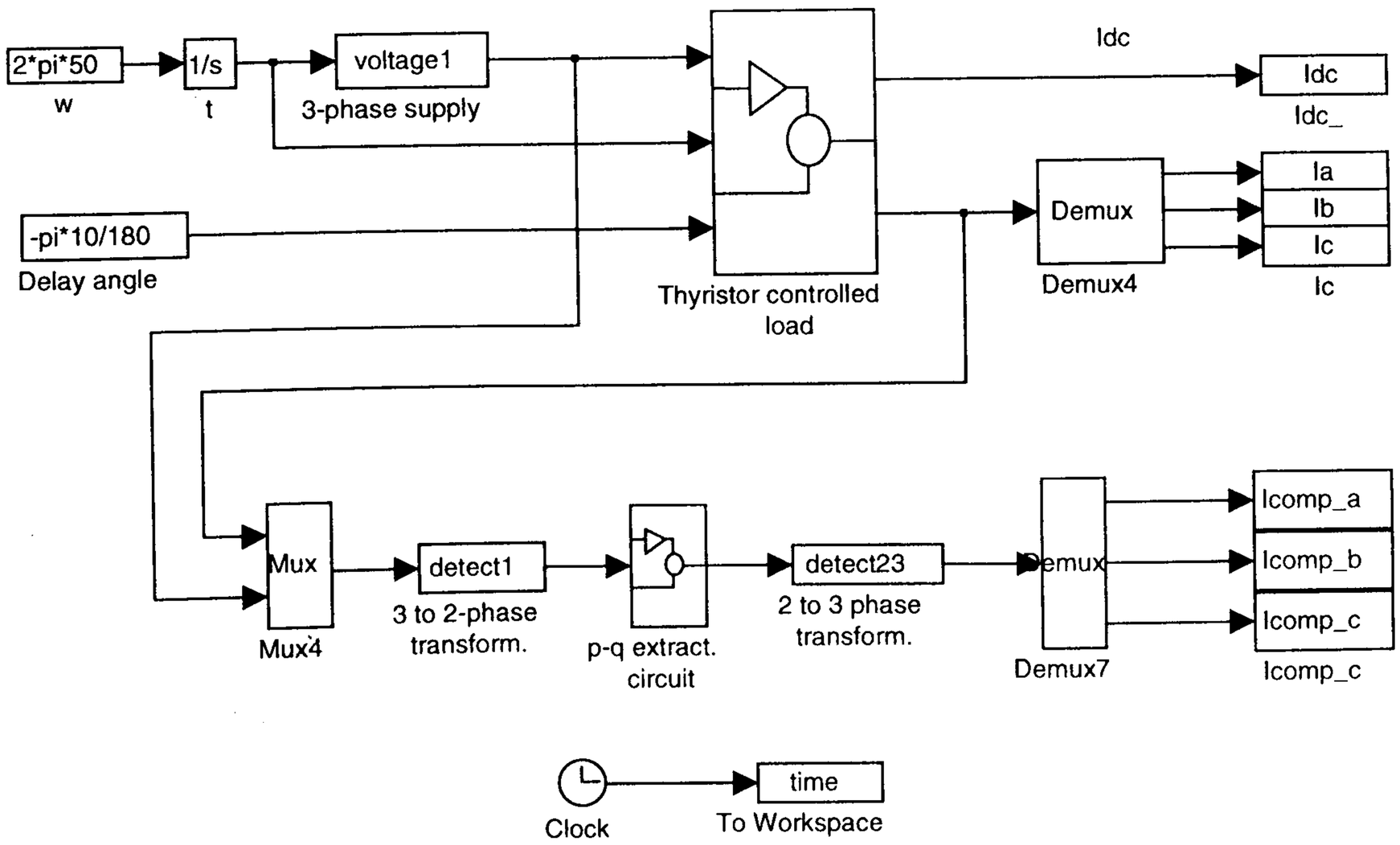
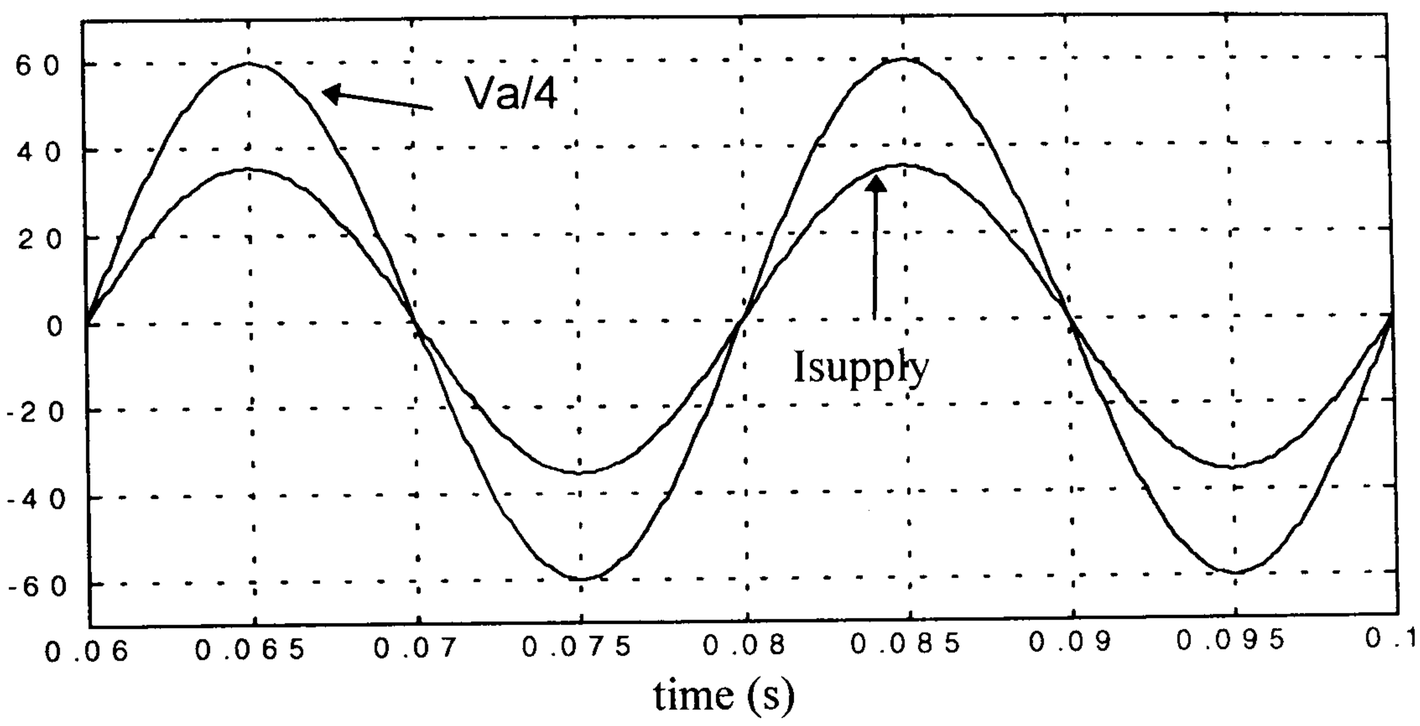
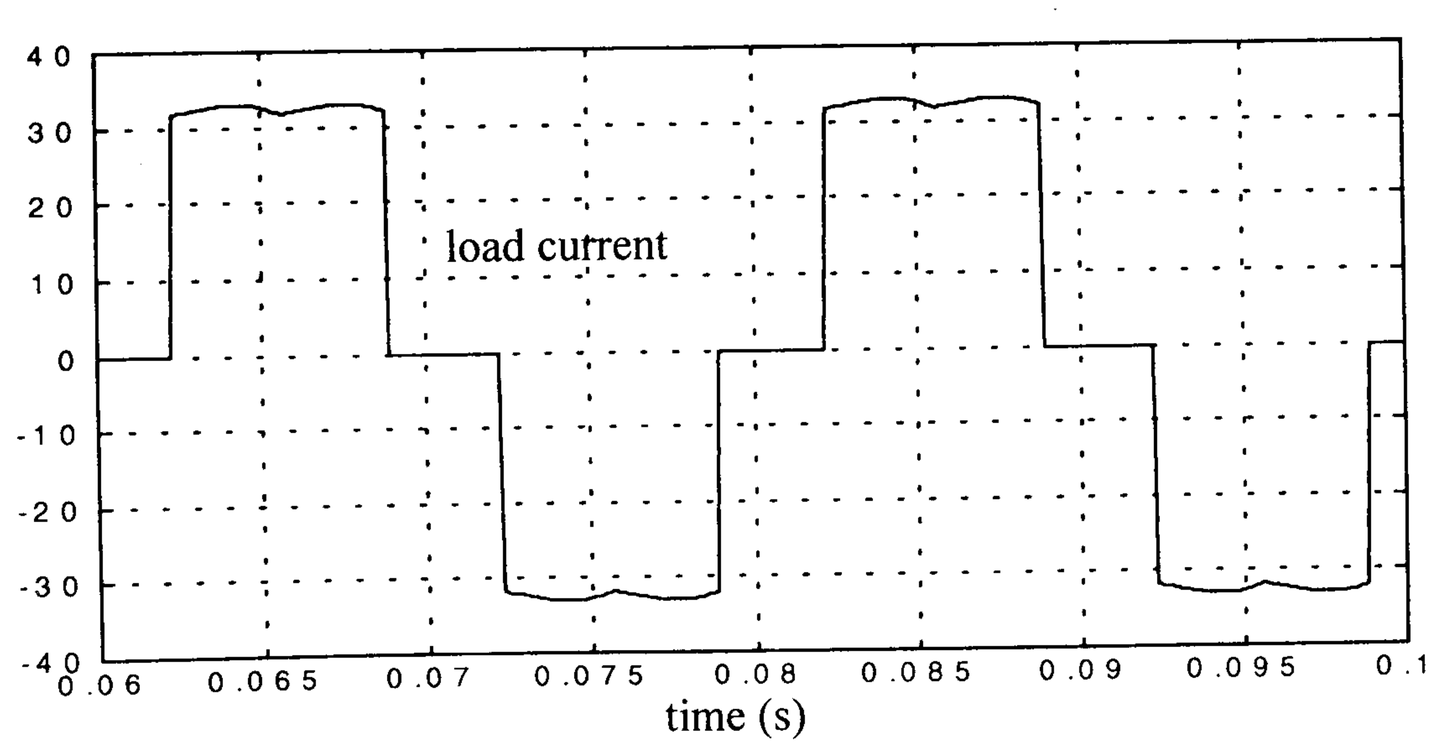


Figure 5.3 Simulink model for p^* and q^* detection circuit

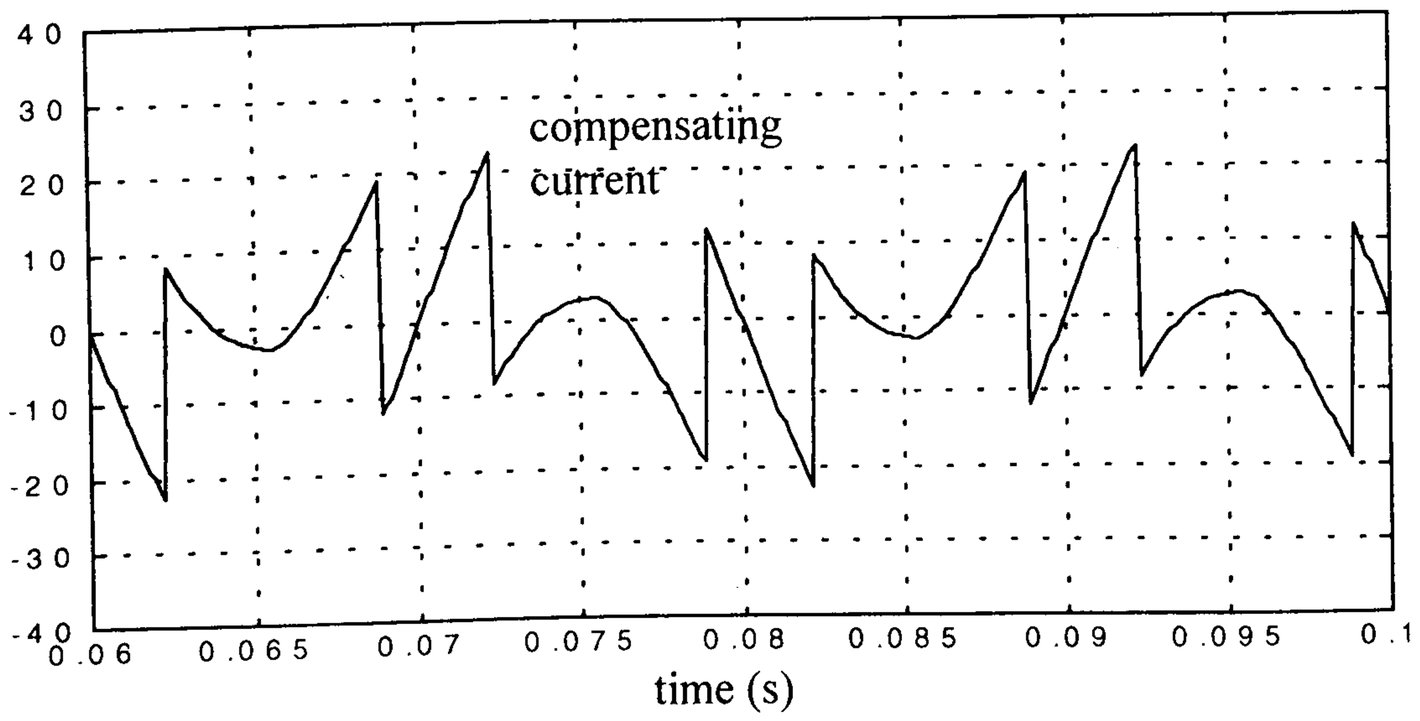
The required components of p^* and q^* can be obtained by using a low pass filter as shown in Figure 5.2. To eliminate the most dominant harmonic currents, namely the 5th, 7th and 11th harmonics, the filter is designed to have a cut off frequency at 150 Hz. The low pass filter is used to extract the $\overline{p}_L$ component from p_L. This is then subtracted from the main signal to give p^* and q^*.



(a)



(b)



(c)

Figure 5.4 Simulation results for p^* and q^* detection

In order to verify the effectiveness of the detection technique, a simulation was performed using MATLAB with Simulink. The Simulink model is as shown in Figure 5.3 with a three-phase controlled bridge used as a load. The results in Figure 5.4 prove that the detection technique described earlier is able to detect the required p^* and q^* thus yielding a reference current that will maintain a sinusoidal supply current in phase with the supply voltage.

5.3 CURRENT CONTROL TECHNIQUE

Once the reference waveform has been determined by the detection circuit, the APF system must incorporate a control scheme which will give a quick response and accurately follows the current references. Several possible techniques have been described in Chapter 2 (section 2.6). The technique employed here is based on delta modulated switching strategies [5.2-5.6]. Compared to other techniques, it is more robust to noise and operates at fixed switching frequency. However the technique is more complex to implement compared to hysteresis current control.

Delta modulation is a variation of pulse code modulation (PCM) used in communication systems [5.7]. The simplest form of delta modulation is shown in Figure 5.5. The analogue reference current is encoded into digital pulses by the delta modulator. These pulses are decoded back into an analogue waveform by an integrator in the feedback loop. This is substracted from the reference waveform to give an error signal which is quantized to one of two possible levels depending on the error polarity. One drawback of such a simple system is that it provides no control over the switching frequency.

Better control of the switching frequency can be obtained by using a discrete current regulated delta modulation technique [5.8] which is an instantaneous discrete time switching regulator i.e. zero hysteresis. The analogue signal is sampled and compared at regular intervals. Figure 5.6 shows the functional block diagram of such a system. The switching frequency is thus fixed by the sampling frequency. The switching signals control the inverter output voltage. This in turn will produce the desired line current, given by :

$$\Delta i_k = \frac{E_{ik} - Vs}{L} \Delta t \qquad \textbf{(5.3)}$$

where L is the AC link inductance and E_{ik} is the inverter output voltage corresponding to the switching state k (from eight different switching configurations).

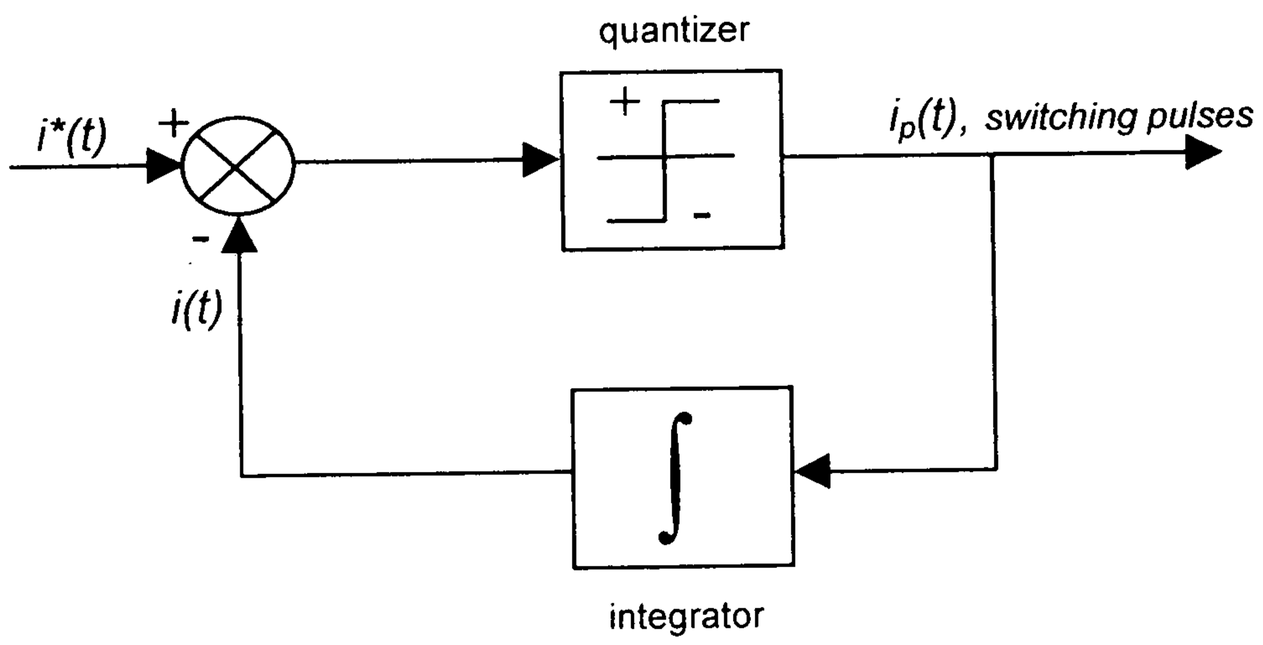


Figure 5.5 Block diagram for simple delta modulation

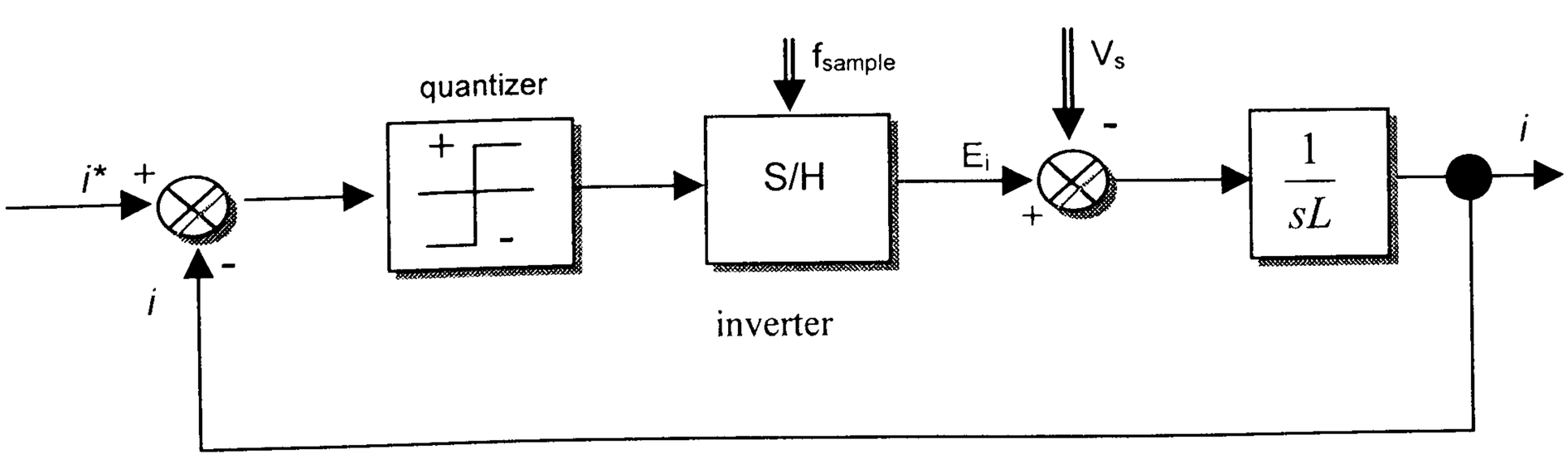


Figure 5.6 Discrete delta modulation

The deadband concept mentioned in the previous chapters can also be applied to discrete delta modulation. Once the switching signals have been produced by the delta modulator, the deadband rule is applied to control the on and off region so that switching losses are minimised. The results discussed in the following sections will show how the use of the deadband concept will reduce the switching losses thus improving system efficiency.

5.4 XILINX FPGA AND DSP DESIGN IMPLEMENTATION

A similar setup to that described in the previous chapter is used. A delta modulated switching strategy is simpler to implement in the available hardware, compared to alternative PWM techniques. The actual current can be compared directly with the reference waveform in the DSP to produce switching signals for all six IGBTs in the three phase inverter. These are then transferred to the Xilinx FPGA where underlap time is inserted. Appendix D.3 shows the modified top level schematic design for the state machine and switching signals generation.

As explained in Section 5.2, a filter is used in the detection circuit. The filter type is critical to ensure good p^* and q^* detection. A 10th order Chebyshev Type II IIR low pass digital filter is used. Although a Chebyshev Type II filter permits a certain amount of ripple in the stop band, it is chosen because it has a steeper roll-off than other filter types. The gain response of a Chebyshev filter is given by [5.9]:

$$\left| \frac{V_{out}}{V_{in}} \right| = \frac{1}{\sqrt{1 + \varepsilon^2 Cn^2 \left(\frac{\omega_{stop}/\omega_{pass}}{\omega_{stop}/\omega} \right)}} \tag{5.4}$$

where *Cn* is a special polynomial which is a function of '*n*' (the order of the filter) and ε is a constant which determines the amount of ripple in the pass band. The ripple is always of equal magnitude throughout the stop band and the ripple number increases with filter order. An IIR (infinite impulse response) filter consists of a series of feedback loops and with such a design, under certain conditions it may become unstable. However, IIR filters have the advantage that for the same roll-off rate they require fewer taps than FIR (finite impulse response) filters. With careful design, a stable filter can be achieved which requires lower processor resources. The filter order and coefficients are determined with the aid of a MATLAB program. Figure 5.7 shows the frequency response of the designed filter. Figure 5.8 shows the frequency response and phase response in the pass band region. Phase is important since a signal entirely within the pass band of a filter will emerge distorted if the time (group) delay of different frequencies through the filter is not constant.

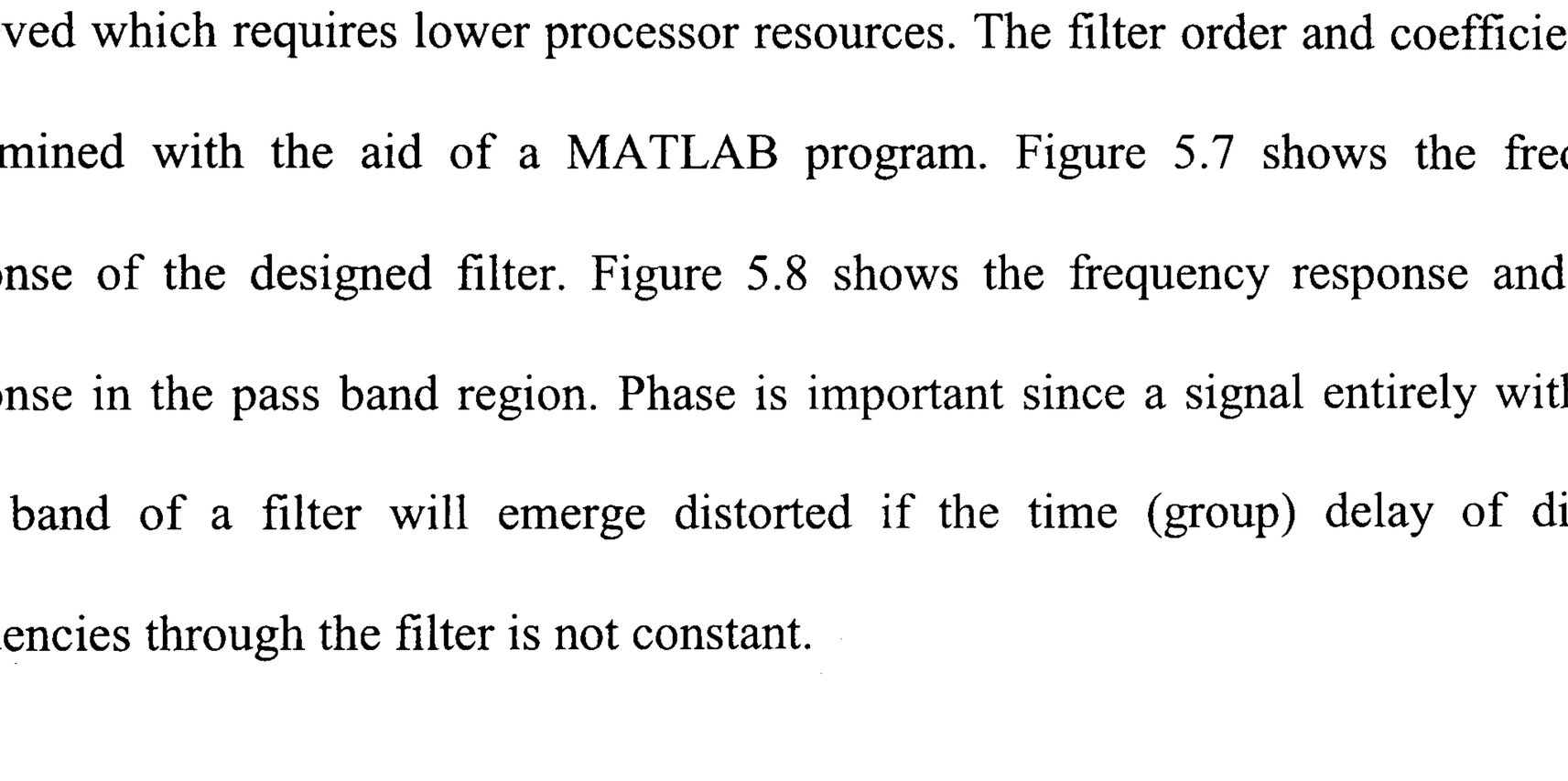


Figure 5.7 Frequency response for Chebyshev filter

The discussion on the detection circuit based on instantaneous reactive power theory is made on the assumption that the supply voltage is sinusoidal. However in reality this is

not the case. The supply voltage waveform obtained in the laboratory has a flat-topped characteristic as shown in Figure 5.9. This slight distortion is common in an installation where there are many single phase power supplies being used, for example as in personal computers.

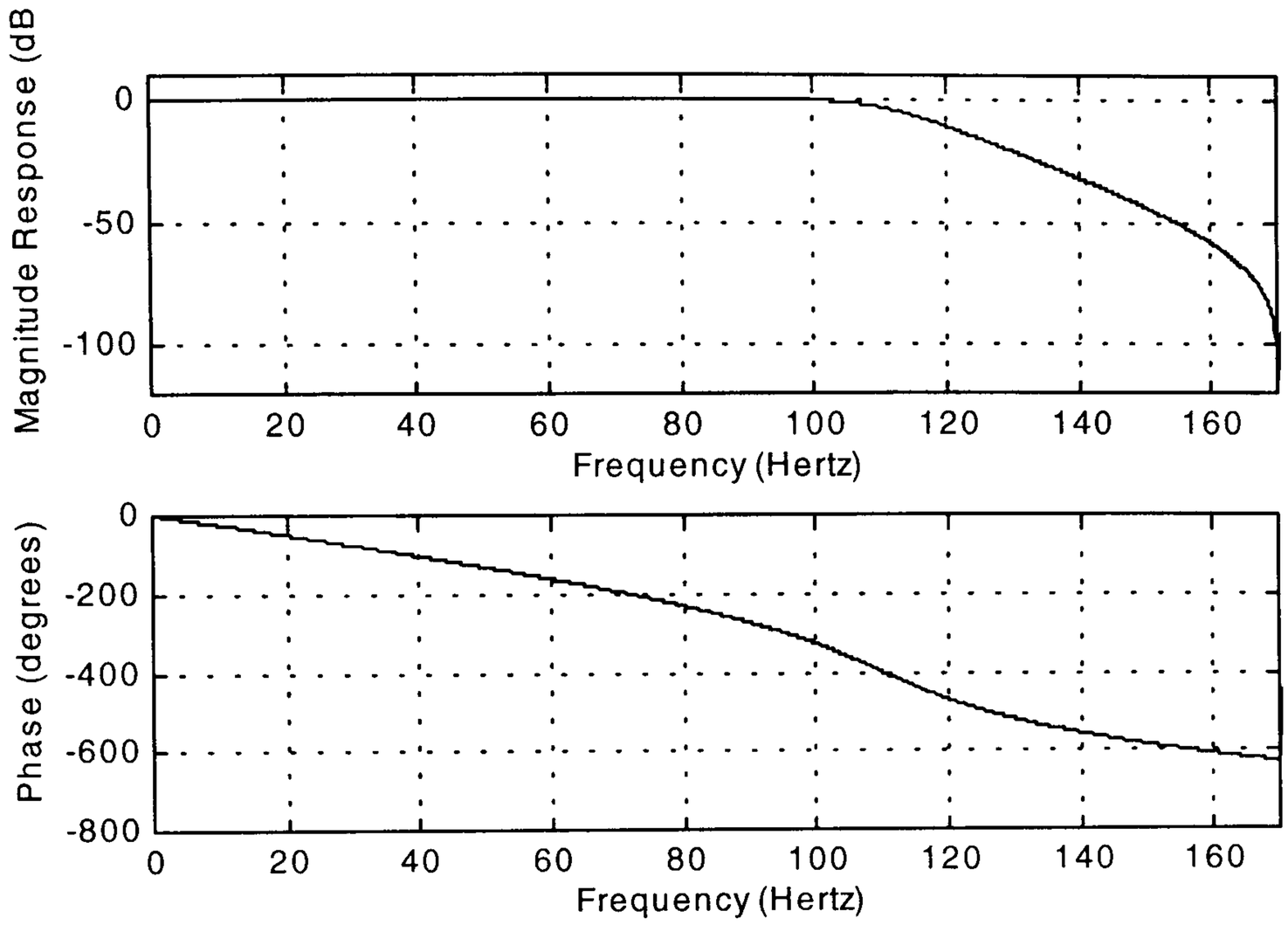


Figure 5.8 Frequency and phase response in the pass band region for Chebyshev Type II filter

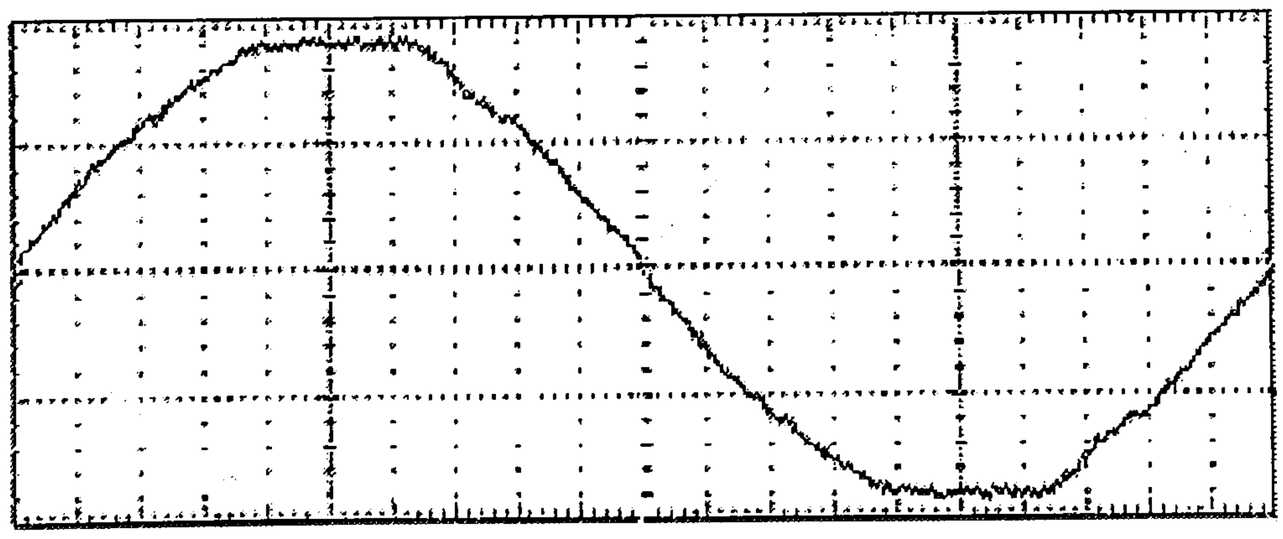


Figure 5.9. Flat-topped supply voltage waveform

With such a voltage waveform, error will occur in the calculation of the reference current. The APF system works by injecting current to compensate the harmonics and reactive power to give a sinusoidal supply current with unity power factor. If the voltage

waveform in Figure 5.9 is used, the reference current generated by the detection circuit will try to compensate not only the load current, but also the supply voltage. The problem is simulated in Simulink and the waveforms are given in Figure 5.10. As shown in Figure 5.9 (a), the resultant supply current is not sinusoidal, particularly around the peaks.

In order to solve this problem, the sampled voltage waveforms are filtered in the DSP to give a sinusoidal supply voltage. This however introduced additional time delay which can cause errors. The delay can be compensated when the three phase components are transformed to their two phase equivalents. Instead of using equation (2.1), the two phase voltage components are determined using the following equation:

$$\begin{bmatrix} v_a \\ v_b \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos\theta & \cos(120+\theta) & \cos(240+\theta) \\ \cos(90-\theta) & \cos(30+\theta) & \cos(150+\theta) \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \quad \textbf{[5.5]}$$

where θ is the delay angle caused by the use of a digital low pass filter.

The listing of the full DSP program, which does the following tasks, is included in Appendix E.3 together with memory location information.

i. transfer data from ADC memory to DSP
ii. apply low pass filter to voltage supply signal
iii. apply 3-phase to 2-phase transformation
iv. p^* and q^* detection
v. determine reference current and generate switching signals
vi. send synchronisation signal to set underlap time
vii. transfer signals to the Xilinx FPGA and DAC

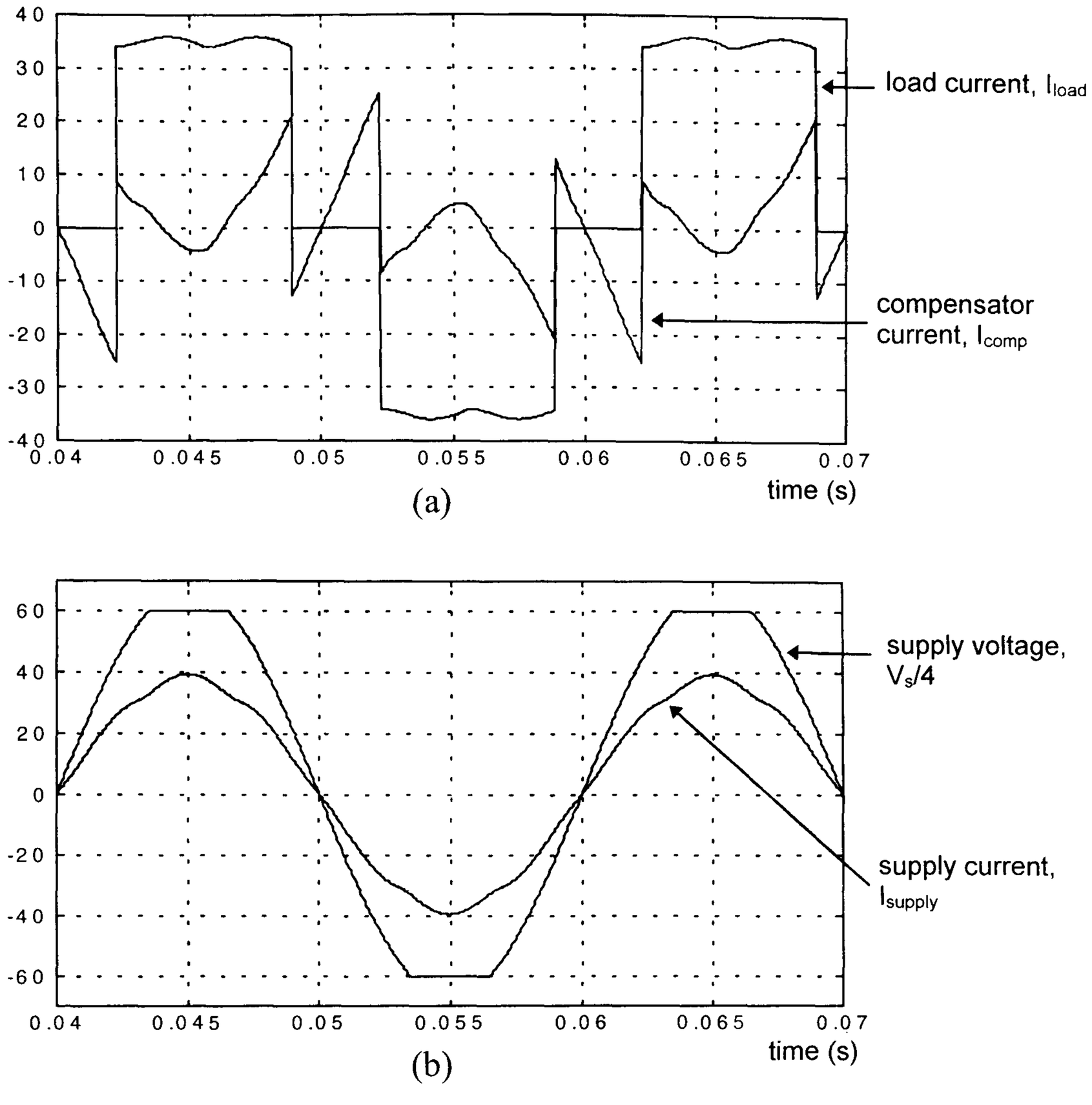


Figure 5.10 Effect of flat-topped supply voltage

5.5 SIMULATION RESULTS

MATLAB with Simulink was used to simulate the complete APF system. The Simulink model used is shown in Figure 5.11 with a three phase controlled bridge as a load. Results for basic delta modulation and modified delta modulation with deadbanding are presented to evaluate performance. These results are shown in Figure 5.12(a) and (b). Both sets of results confirmed that the APF is able to compensate the load current to give a sinusoidal supply current with unity power factor.

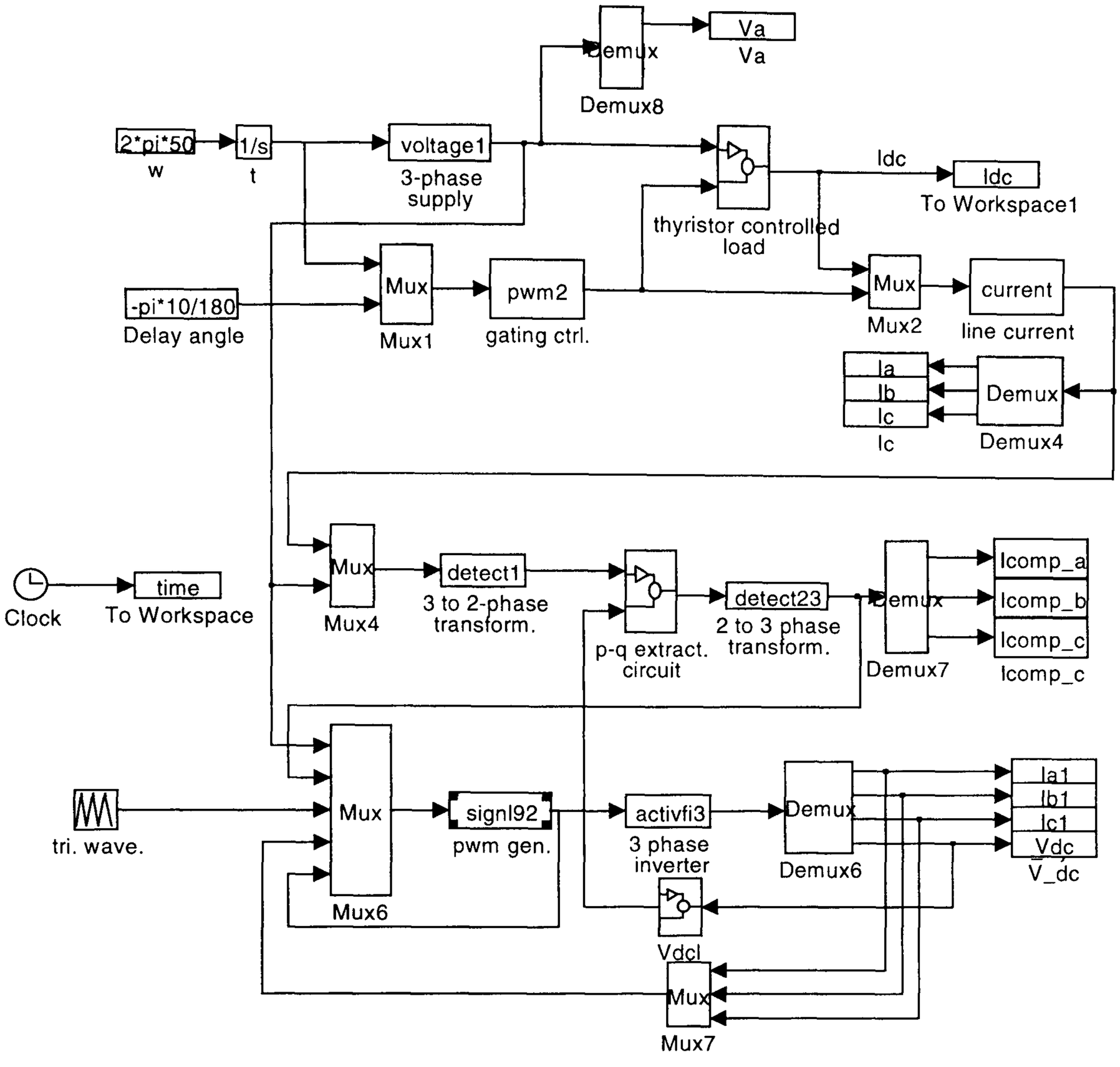
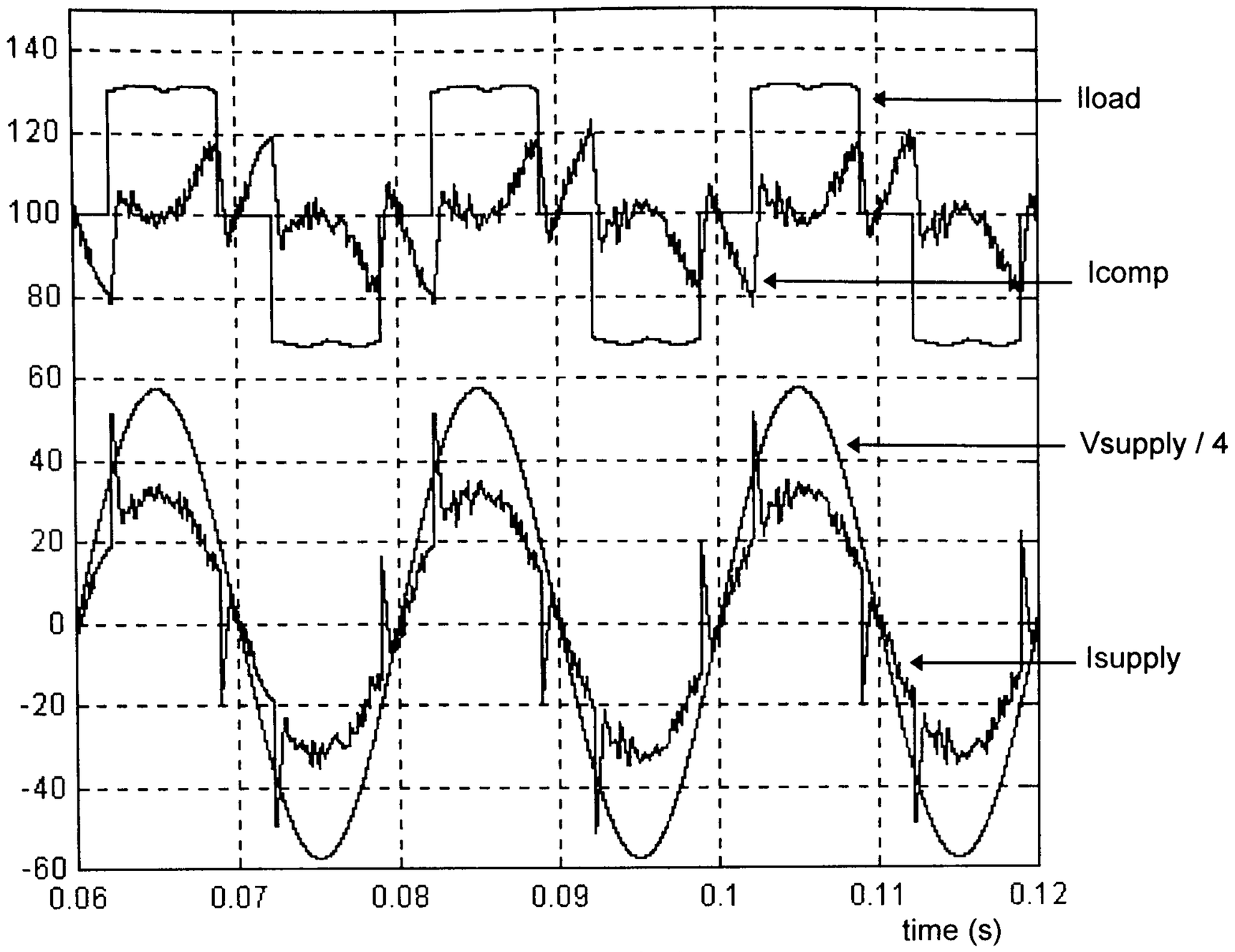
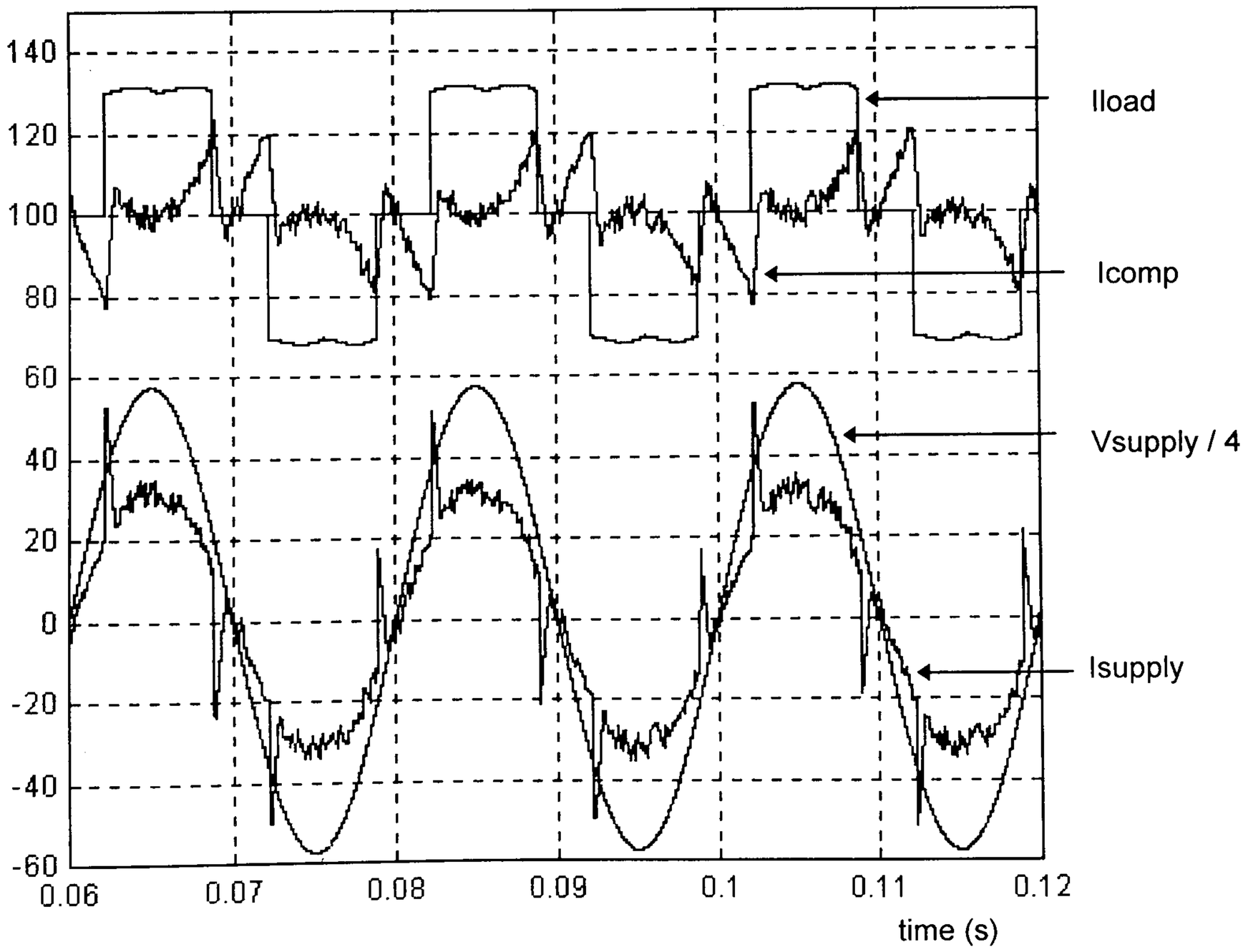


Figure 5.11 Simulink model for active power filter system

However it can be seen that the supply current also consists of current spikes which correspond to the phase change of load current. This is due to the way the load is modelled in Simulink, which gives a sudden jump in load current from zero to its peak. In reality, the effect of AC side inductance will give a more gradual change (commutation) in the load current. To give more realistic results, the switching signals generated in the Simulink model were used in a PSpice simulation. The listing of the PSpice program is included in Appendix A.3. The results are shown in Figure 5.13 for the modified delta modulation case.



(a) with delta modulation



(b) modified delta modulation with deadband concept

Figure 5.12 Matlab simulation results for active power filter system

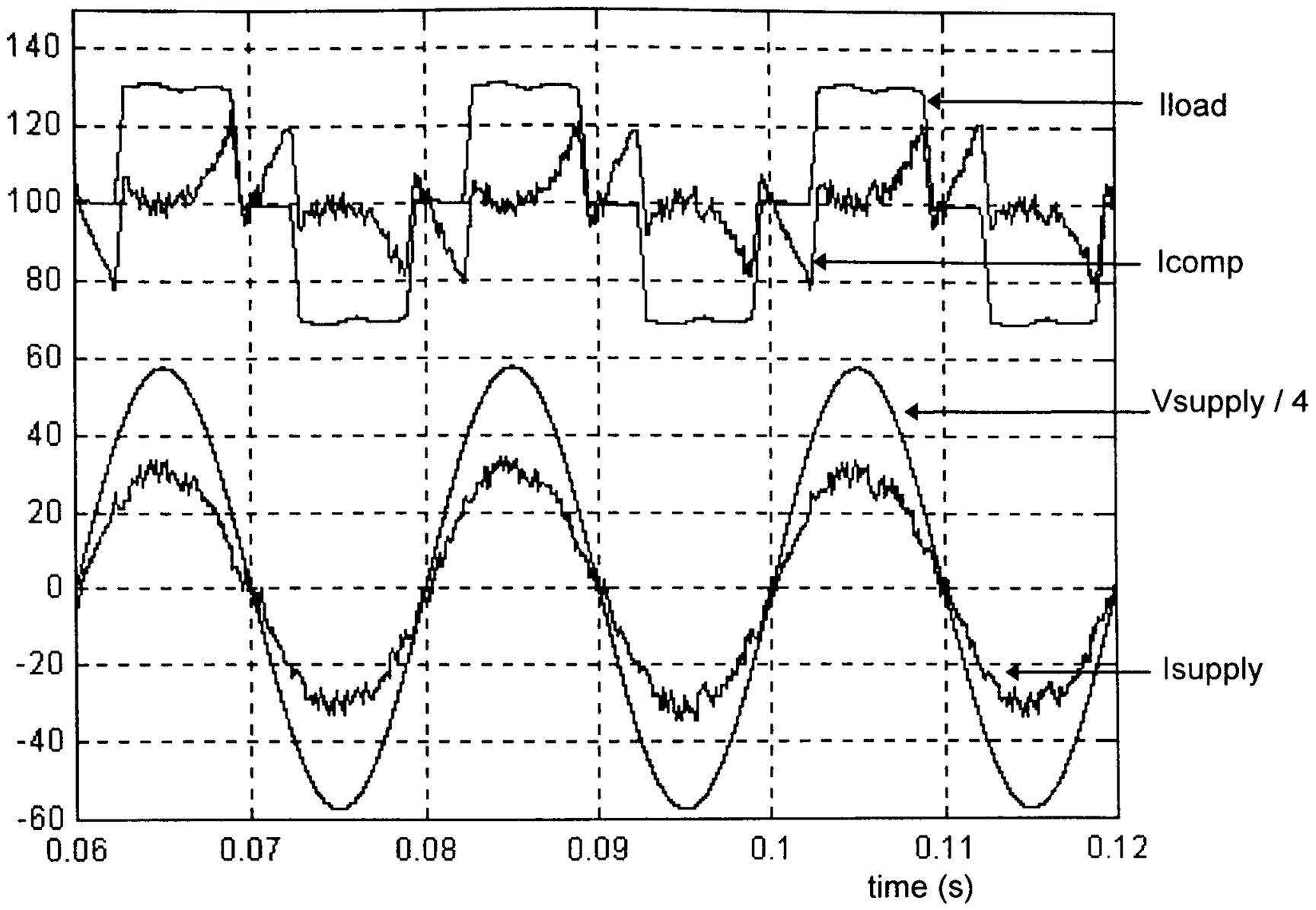


Figure 5.13 PSpice simulation results for the active power filter with modified delta modulation

System performance is evaluated by examining the current harmonics which are shown in Figure 5.14. In Figure 5.14 (a), the load current harmonics are evident at $(6k \pm 1)$ where k is the harmonic order. The APF system with modified delta modulation gave similar performance to the one using a basic delta modulated switching strategy. Better results are obtained in the PSpice simulation where the effect of AC side inductance is included. The total harmonic distortion (THD) of the supply current is determined from the Simulink simulations using the equation:

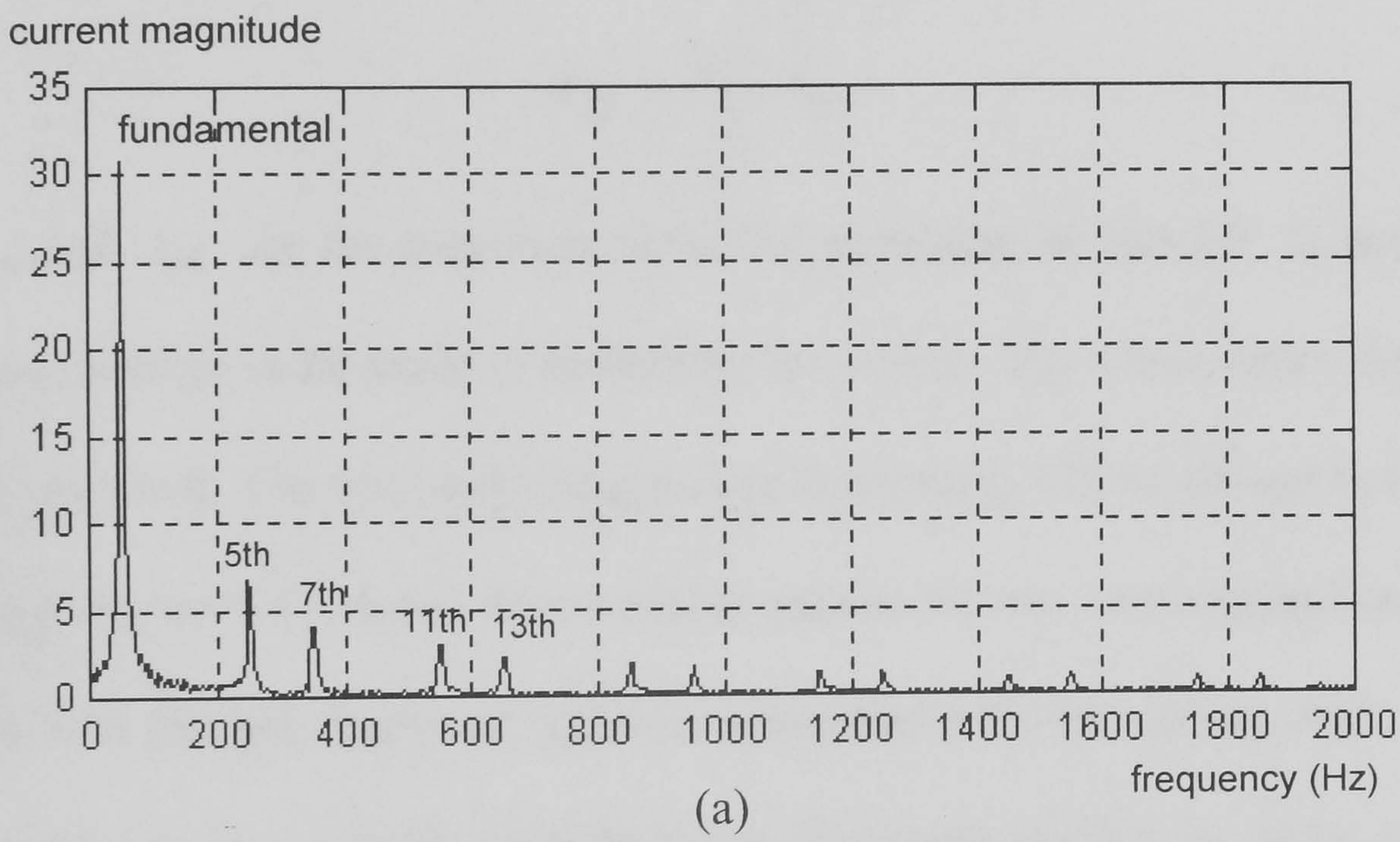
$$THD\ (\%) \ = \ 100\ x \sqrt{\sum_{h \neq 1} \left(\frac{I_h}{I_1} \right)^2} \qquad \textbf{(5.6)}$$

where h is the harmonic order. The results are tabulated in Table 5.1 (up to the 40th harmonic). A high value is obtained due to the presence of current spikes (as explained earlier). This is acceptable for the purpose of comparison between the two modulation strategies since the spikes do not dominate the harmonic distortion results. The results

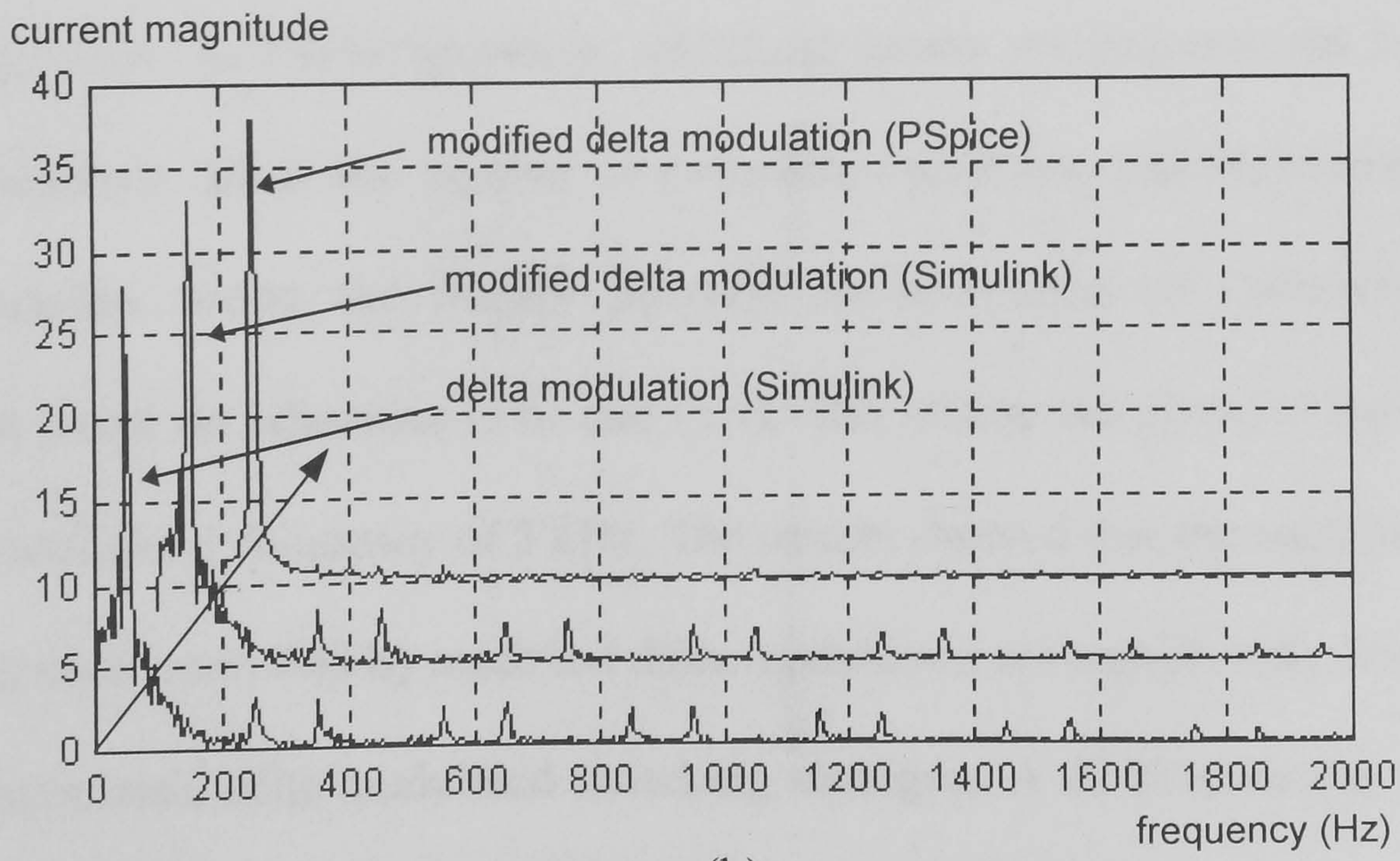
show that the use of modified delta modulation strategy increased the THD contents by 1.1 %.

	Basic delta modulation	Modified delta modulation
THD (%)	28.04	29.16

Table 5.1 Comparison of THD values



(a)



(b)

Figure 5.14 Harmonic currents for (a) load current and (b) supply current

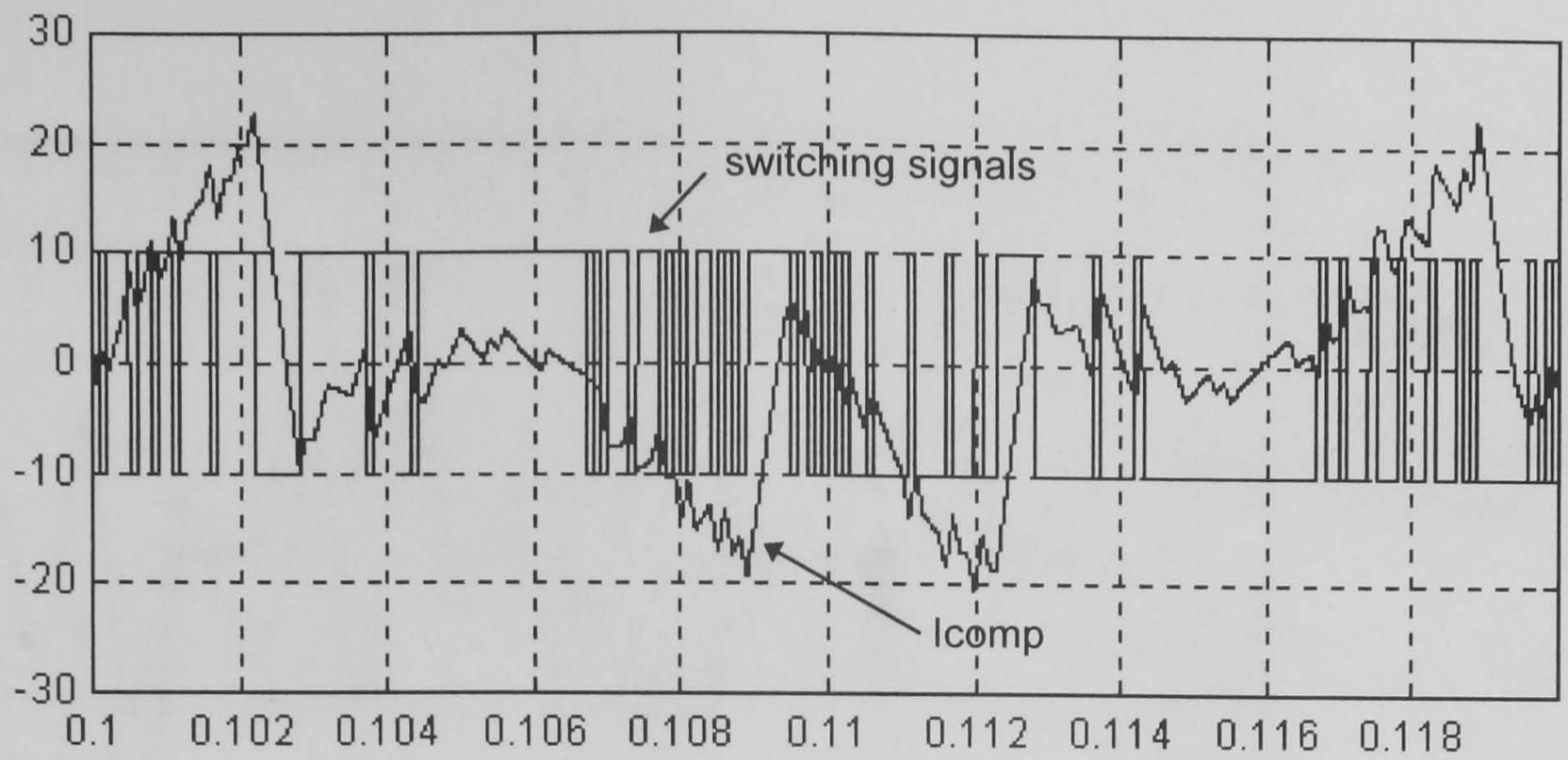
Switching device losses associated with a hard switched system will comprise two components, viz., on-state conduction losses and the switching losses. Here the conduction losses are assumed the same for each modulation system, hence may be neglected when comparing performance. The advantages of using modified delta modulation are significant when analysing device switching losses.

For an inductive load the energy loss per cycle can be approximated by [5.10]:

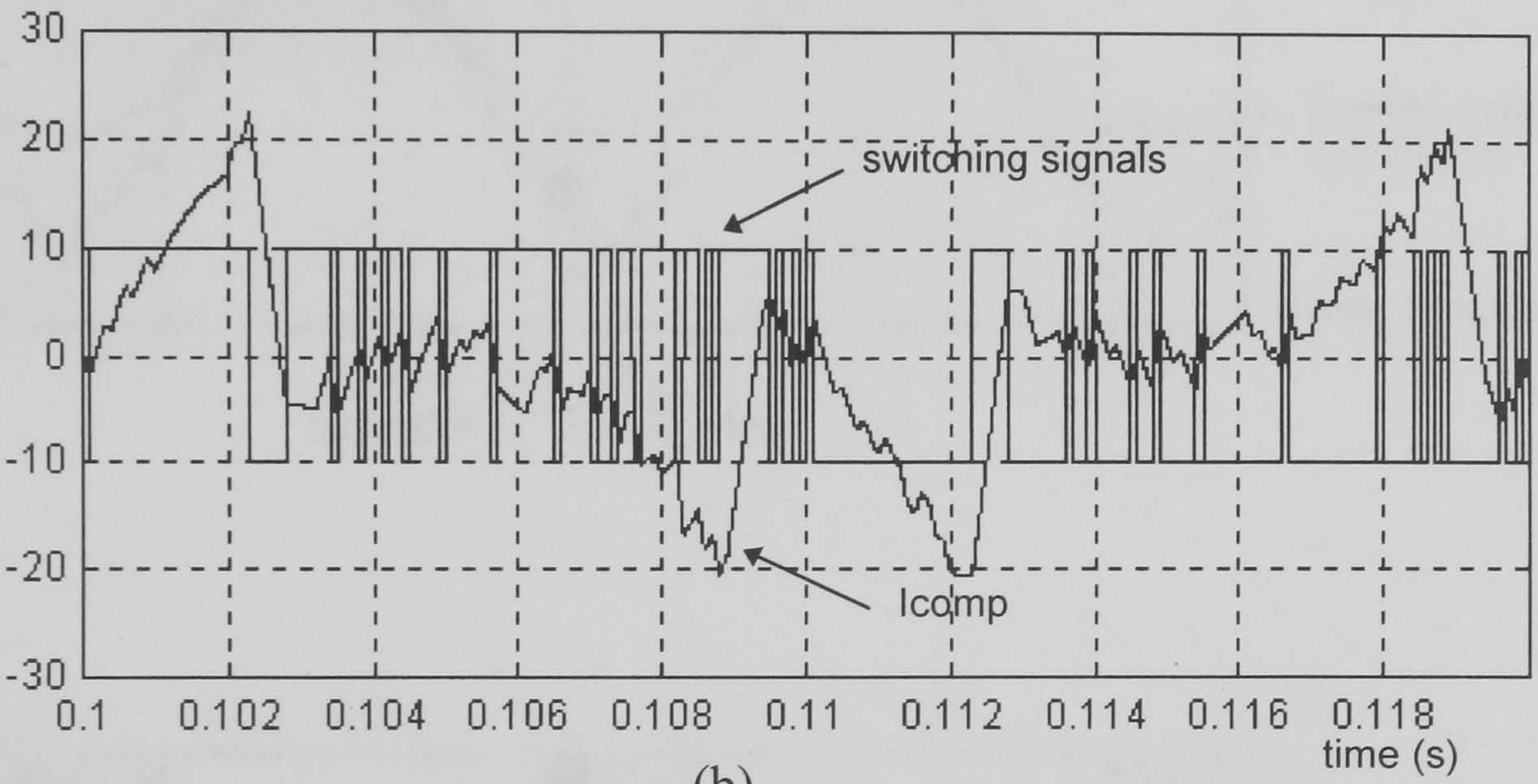
$$W_{on} = \frac{i_m v_s}{2} t_{on} \qquad \textbf{[5.6]}$$

$$W_{off} = \frac{i_m v_s}{2} t_{off} \qquad \textbf{[5.7]}$$

where t_{on} and t_{off} are the transition times for switching on and off, i_m and v_s are the current and voltage to be made or broken by the switch. The value varies depending on the point-on-wave. The total switching power dissipation will be the sum of operations per second. Figure 5.15 shows the switching signals for one compensator current cycle. In Figure 5.15 (b), the deadband region is associated with the current peak. The region also depends on the voltage magnitude (as discussed earlier) in order to minimise switching losses. In PWM operation, switching losses are proportional to the PWM carrier frequency. Here, the number of switching operations can be counted to give a loss indication. Using the Matlab program an estimation of switching losses is calculated based on equations (5.6) and (5.7). The results are given in Table 5.2 for a sampling/switching frequency of 5 kHz. The results showed that the switching losses of an APF system controlled by modified delta modulation are significantly lower than in a system using basic delta modulated switching strategies. A 22 % reduction in switching losses is achieved at the expense of a 1.1 % increment in the THD value.



(a)

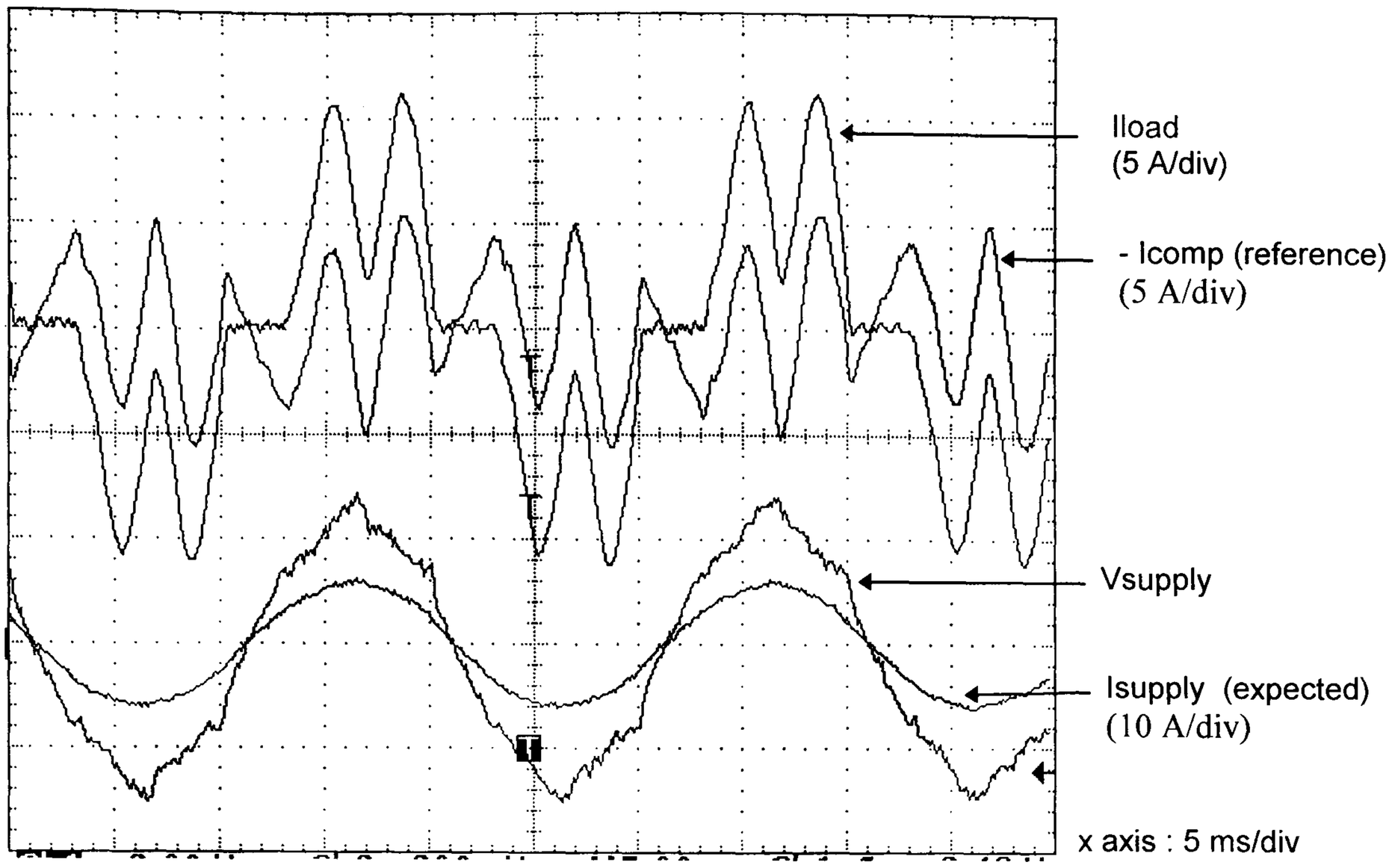


(b)

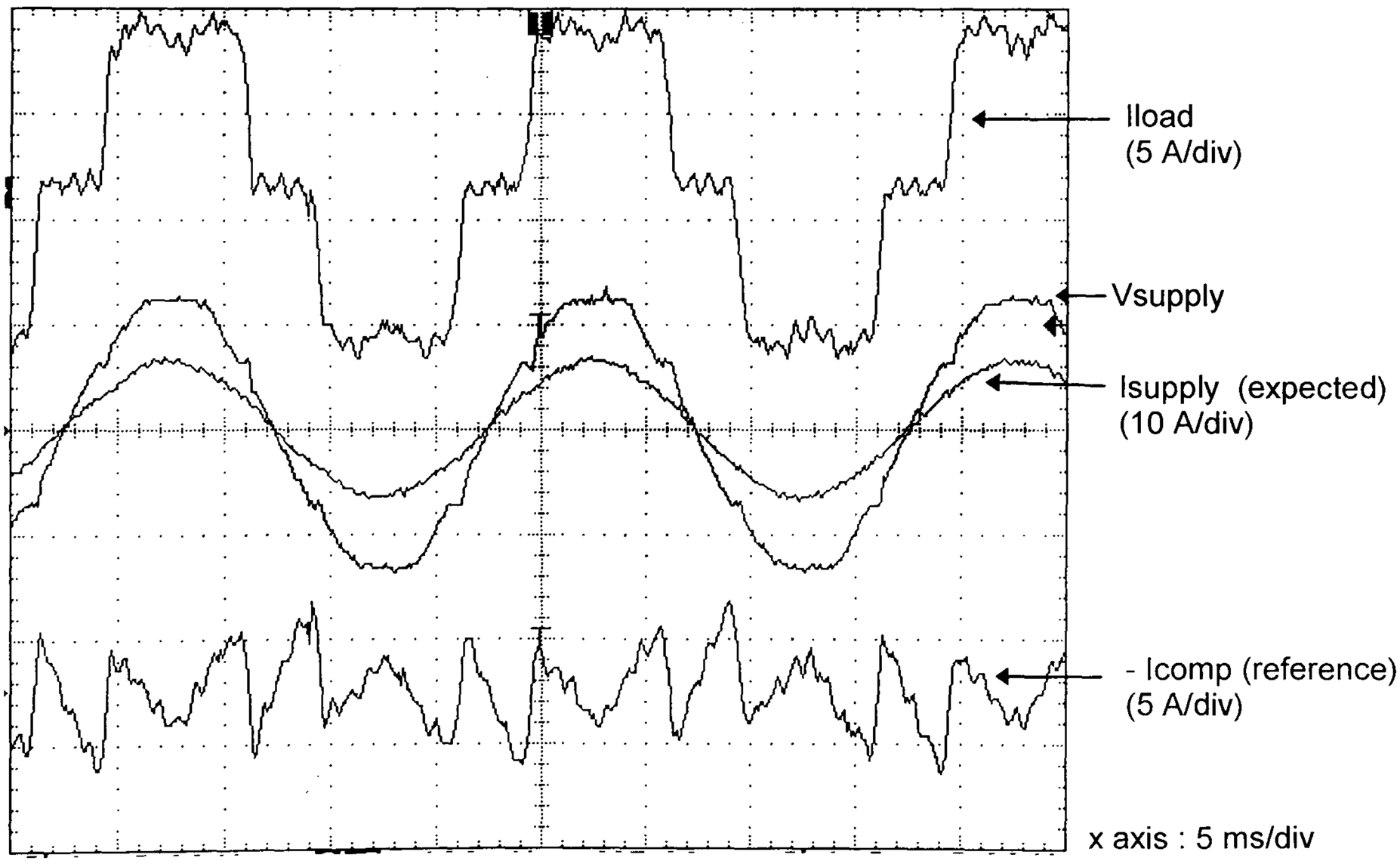
Figure 5.15 Switching signals with (a) delta modulation and (b) modified delta modulation with deadbanding

	Modified delta modulation	Basic delta modulation	Improvement
Turn-on losses (W)	0.995	1.295	23.2 %
Turn-off losses (W)	4.615	5.895	21.7 %
Total losses (W)	5.610	7.190	22.0 %
No. of switching operations per cycle	185	225	17.7 %

Table 5.2 Switching loss comparison between delta modulation and modified delta modulation

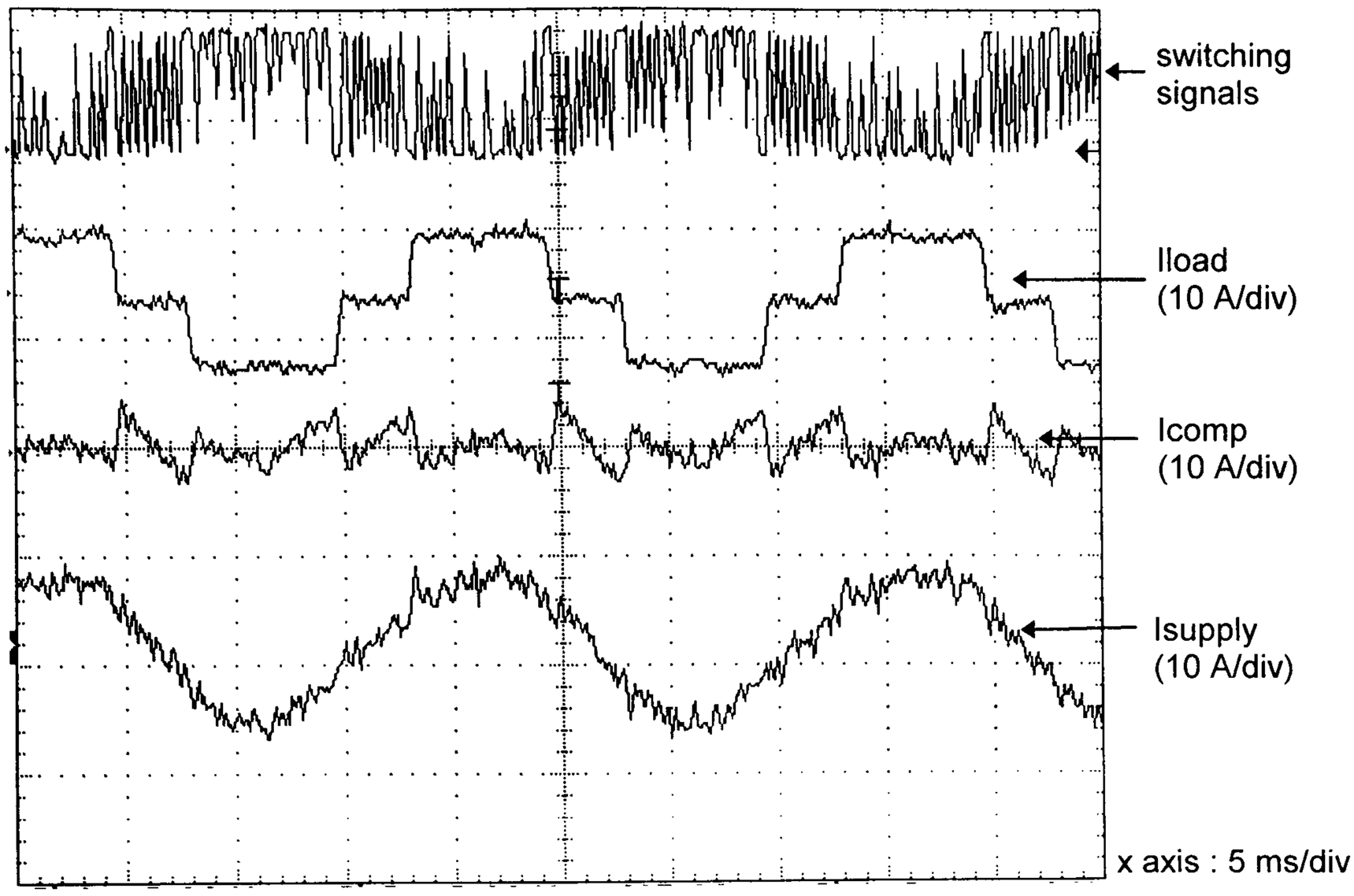


(a) with L - C - R load

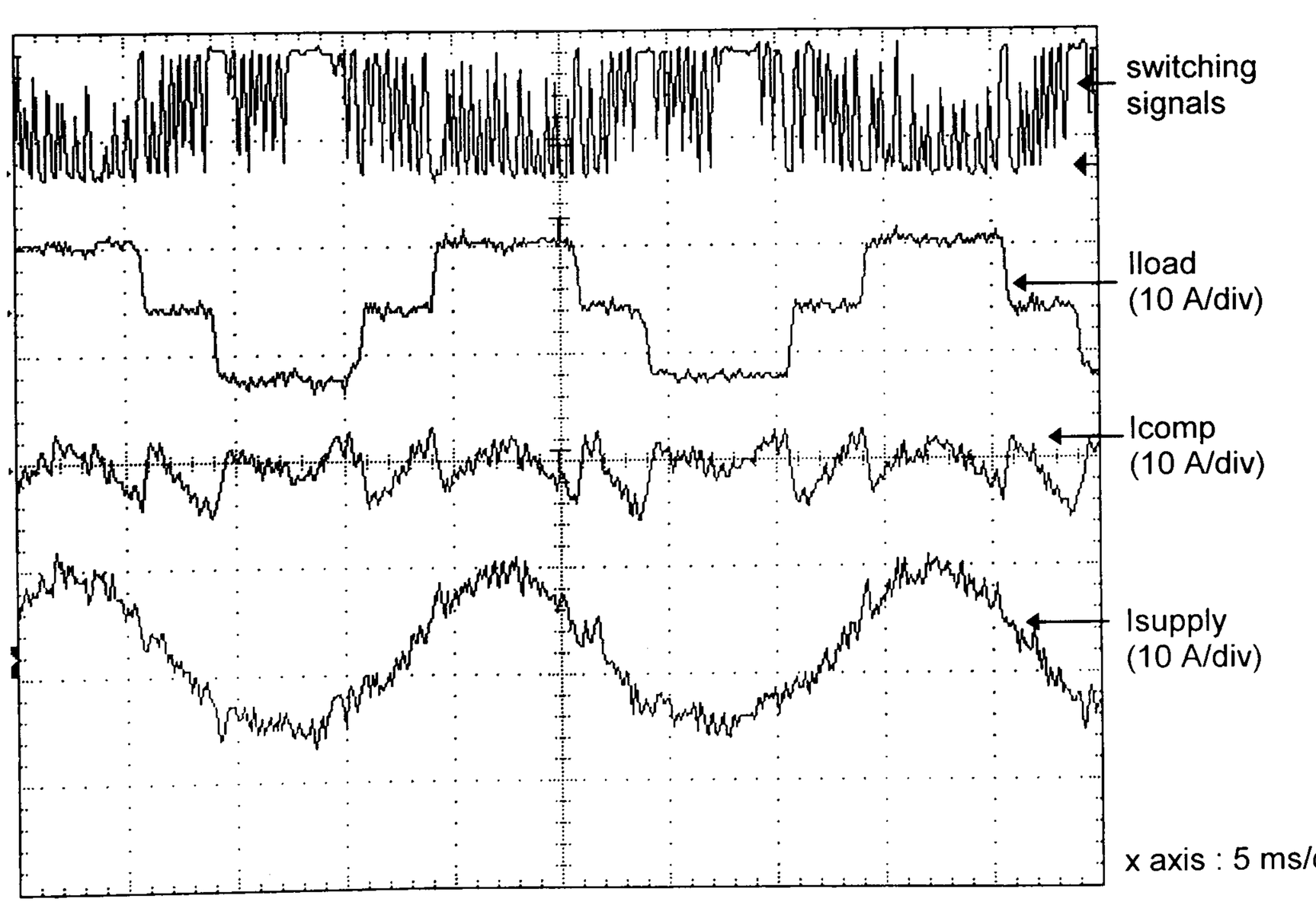


(a) with L - R load

Figure 5.16 Experimental results showing the effectiveness of the detection circuit



(a) with delta modulation



(b) modified delta modulation with deadband concept

Figure 5.17 Experimental results for active power filter with different switching techniques

5.6 EXPERIMENTAL RESULTS

The active power filter system was tested in the laboratory using the power circuit described in Chapter 3. The load is provided by a three-phase full-bridge diode rectifier, SKD 60/12. The detection circuit was tested with two different types of load; an L - R load and an L-C-R load. Experimental results are shown in Figure 5.16. With both load conditions, the system can detect the required harmonic and reactive power to provide compensation. The system produces a reference compensator current, I_{comp} which gives a sinusoidal supply current with unity power factor, $I_{supply\ (expected)}$.

The APF system was tested to assess the ability of the proposed switching technique to track the reference current. In order to reduce current error the sampling/switching frequency was doubled to 9.77 kHz. The results are shown in Figure 5.17. Figure 5.17 (a) shows the APF system with a delta modulated switching strategy while Figure 5.17 (b) was obtained with modified delta modulation. Both results proved that the system is able to compensate the load current to give sinusoidal supply current with unity power factor. The switching signals in Figure 5.16 (b) show the deadband region introduced into the system which results in lower switching losses as discussed in the previous section.

5.7 SUMMARY

The design and operation of an active power filter system has been presented and discussed in this chapter. To provide compensation, the system must first be able to detect harmonic and reactive power. The detection circuit is designed using a Chebyshev type II IIR filter and is based on instantaneous reactive power theory. Two

different switching control strategies were described. Discrete current regulated delta modulation enables the injected current to follow the reference current with good accuracy at a switching frequency associated with the system sampling frequency. Another technique is to use a modified delta modulated switching strategy which enables compensation to be achieved but with a reduction in device switching losses.

5.8 REFERENCES

[5.1] H. Akagi, Y. Kanazawa and A. Nabae, "Instantaneous reactive power compensators comprising switching devices without energy storage components", IEEE Trans. on Industry Applications, Vol. IA-20 No. 3, May/June 1984, pp. 625-630.

[5.2] P.D. Ziogas, "The delta modulation technique in static PWM inverters", IEEE Trans. on Industry Applications, Vol. IA-17 No.2, March/April 1981, pp. 199-204.

[5.3] M.H. Kheraluwala and D.M. Divan, "Delta modulation strategies for resonant link inverter", IEEE. Trans. on Power Electronics, Vol. 5 No.2 April 1990, pp. 220-228.

[5.4] M.A. Rahman, J.E. Quaicoe and M.A. Choudhury, "Performance analysis of delta modulated PWM inverters", IEEE Trans. on Power Electronics, Vol. 2 No. 3, July 1987, pp. 227-233.

[5.5] N.A. Rahim and J.E. Quaicoe, "A single phase delta-modulated inverter for UPS applications", IEEE Trans. on Industrial Electronics, Vol. 40 No.3, June 1993, pp. 347-354.

[5.6] G. Joos and P.D Ziogas, "On maximizing gain and minimizing switching frequency of delta modulated inverters", IEEE Trans. on Industrial Electronics, Vol. 40 No. 4, August 1993, pp. 436-444.

[5.7] R.F.W. Coates, "Modern Communication Systems", 2nd Edition 1983, MacMillan.

[5.8] T.G. Habetler and D.M. Divan, "Performance characterization of a new discrete pulse modulated current regulator", Conf. Proceedings, Industry Application Society Conference 1988, pp. 395-405.

[5.9] R.W. Hamming, "Digital filters", Second Edition 1983, Prentice Hall.

[5.10] B.W. Williams, "Power electronics : devices, drivers, applications and passive components", Second edition 1992, Mc Millan.

CHAPTER 6

SLIDING MODE CONTROL OF AN ACTIVE POWER FILTER

6.1 INTRODUCTION

This chapter explains and investigates the application of sliding mode control to an active power filter system. Initially a brief introduction is given on the theory of sliding mode control. Equations will be derived using control theory to determine the control action which can be used to provide the switching signals for the active power filter. The work is supported with simulation results performed in MATLAB with Simulink, which is presented at the end of this chapter. Experimental tests are included to verify the results.

6.2 VARIABLE STRUCTURE CONTROL (VSC) WITH SLIDING MODE

Variable structure control (VSC) with sliding mode was first introduced and discussed in the early 1950's by researchers in the Soviet Union [6.1, 6.2]. The most distinguished feature of VSC is its ability to produce very robust control systems. Essentially, the sliding-mode control system utilises a switching control law which drives the state of the concerned system to a predesigned curve (or sliding curve) in the phase plane. Once the sliding condition has been reached, the control system is insensitive to parametric uncertainty and external disturbances. Extensive research and development has enabled sliding mode control to be applied in wide variety of engineering systems, for examples robot manipulators, underwater vehicles, electric machines and power systems [6.2].

6.2.1 Basic Background

The basic introduction of VSC with sliding mode can be illustrated by considering a second order system [6.2] described by the following equations:

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= 2y - x + u \\ u &= -\psi x \end{aligned} \tag{6.1}$$

where

$$\psi = \begin{cases} 4 & , \; s(x,y) > 0 \\ -4 & , \; s(x,y) < 0 \end{cases} \tag{6.2}$$

and

$$\begin{aligned} s(x,y) &= x\sigma \\ \sigma &= 0.5x + y \end{aligned} \tag{6.3}$$

This is a product of two functions described by:

$$\begin{aligned} x &= 0 \\ \sigma &= 0.5x + y = 0 \end{aligned} \tag{6.4}$$

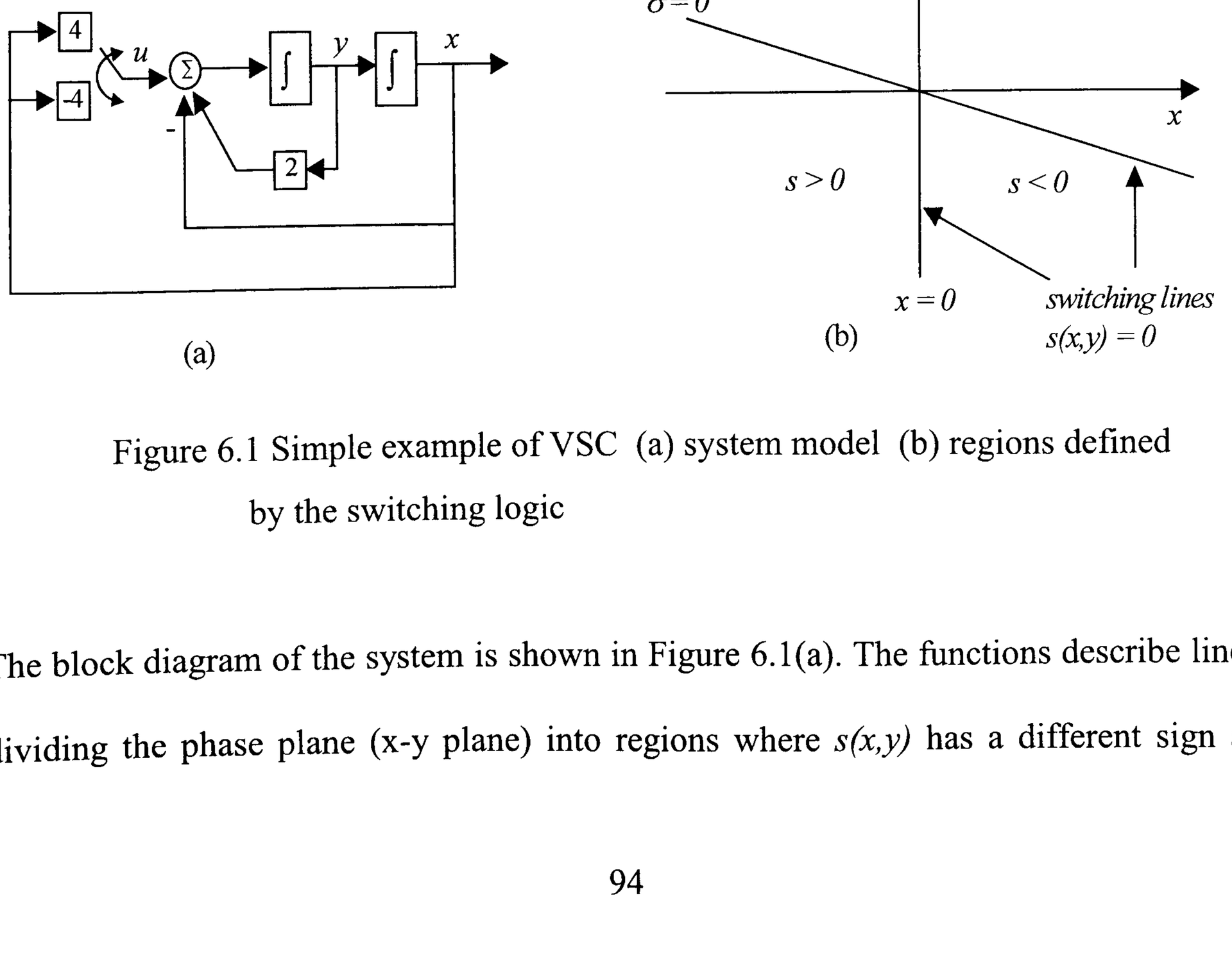


Figure 6.1 Simple example of VSC (a) system model (b) regions defined by the switching logic

The block diagram of the system is shown in Figure 6.1(a). The functions describe lines dividing the phase plane (x-y plane) into regions where *s(x,y)* has a different sign as

shown in Figure 6.1(b). These lines are often called switching lines and $s(x,y)$ is called a switching function. The lines also define the set of points in the phase plane where $s(x,y) = 0$. This set of points is known as the switching surface. The feedback gain ψ is switched according to the sign of $s(x,y)$. Therefore the system is analytically defined in two regions of the phase plane by two different mathematical models. In region I where $s(x,y)=x\mathbf{s} > 0$, the model is

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= 2y - x - 4x \\ &= 2y - 5x \end{aligned} \tag{6.5}$$

In region II where $s(x,y)=x\mathbf{s} < 0$, the model is

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= 2y - x + 4x \\ &= 2y + 3x \end{aligned} \tag{6.6}$$

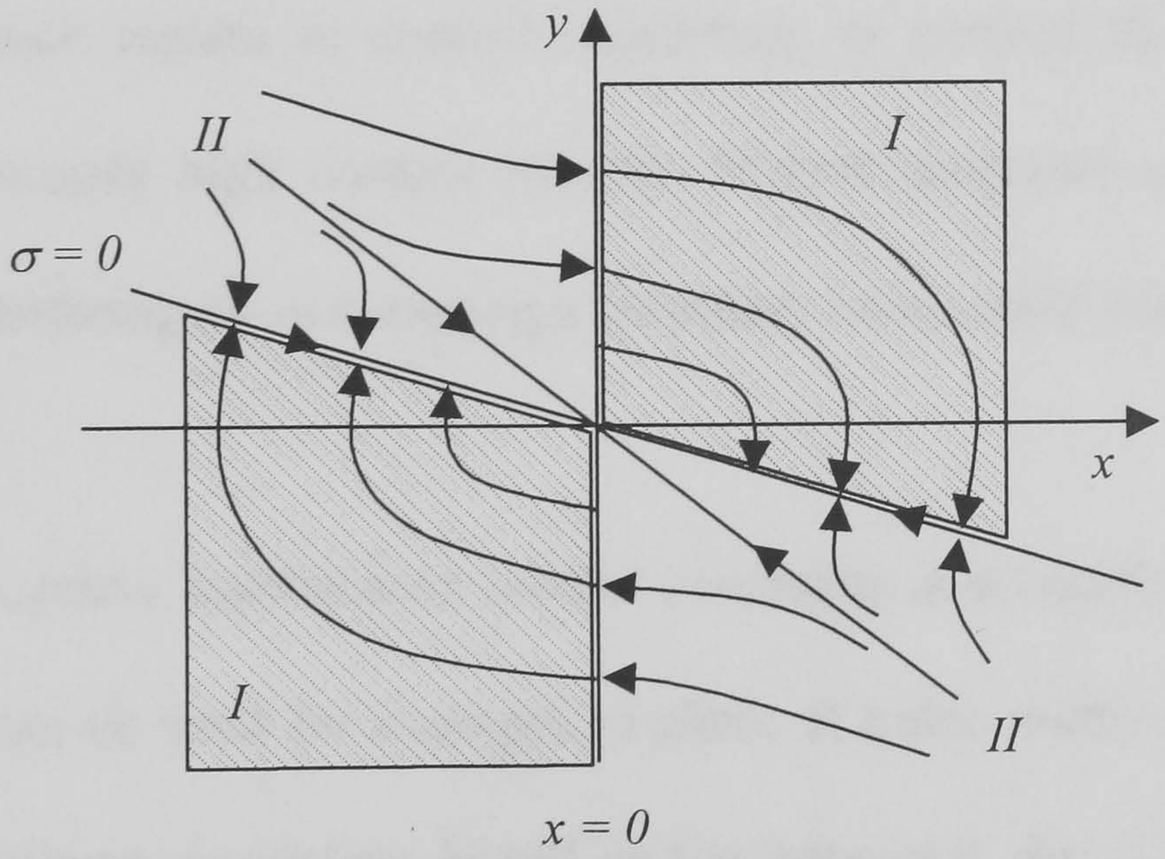


Figure 6.2 Phase-plane plot showing both regions I and II

The phase-plane trajectories described by equations (6.5) and (6.6) can be drawn together to give a phase-plane plot as shown in Figure 6.2. The complete phase-plane plot of the system shows that there are no unusual motion characteristics on the line

$x = 0$ other than possible discontinuities in motion direction. However, the line $\sigma = 0$ contains only endpoints of those trajectories coming from both sides of the line. These points constitute a special trajectory along the $\sigma = 0$ line, representing motion called a sliding mode. Thus, a phase trajectory of this system generally consists of two parts, representing two modes of the system. The first part is the reaching mode in which the trajectory starting from anywhere on the phase plane moves toward a switching line and reaches the line in finite time. The second part is the sliding mode in which the trajectory asymptotically tends to the origin of the phase plane.

During the control process, the control structure varies from one structure to another thus the name variable structure control (VSC). The described control system is also known as sliding mode control to emphasise the important role of the sliding mode in this type of VSC. In practice, the implementation of the associated control switching is imperfect which results in control chattering. In general this is undesirable since it involves extremely high control activity. Slotine proposed an approach [6.8, 6.9] to reduce the chattering by introducing a boundary layer either side of the switching line.

However, in certain applications control chattering is acceptable. The derived switching control law can be used for example in place of pulse-width modulation to control the switching of power converters. Based on this approach, direct implementation of sliding mode control in the active power filter will be discussed in this chapter.

6.3 CONTROL IMPLEMENTATION IN ACTIVE POWER FILTER

The voltage source inverter used in the implementation of the active power filter is a natural variable structure system because its topology changes with switching action. Using the equivalent control concept of VSC with sliding mode, a control law can be derived to ensure that the circuit remains on the sliding surface. The implementation of sliding mode control in an active power filter has been described by Torrey et al. [6.10, 6.11]. Active filtering has been achieved without the need to calculate real or reactive power from the load current.

In this chapter, a different approach will be used to implement sliding mode control. A similar hardware and software setup described in previous chapter will be used. Instantaneous active and reactive power will be analysed to produce a reference waveform to provide the necessary compensation. Sliding mode control is then applied to the active power filter to ensure good tracking of the reference current. The work discussed here is based on an initial study reported in [6.12]. In order to determine its switching control law, a mathematical model of the active power filter is produced in the *d-q* reference frame.

6.3.1 The Active Filter's Mathematical Model

The simplified schematic model of the active power filter system is shown in Figure 6.3.

The line voltages can be described by the following equations:

$$
\begin{aligned}
v_a &= \sqrt{2}E\cos(\omega t) \\
v_b &= \sqrt{2}E\cos(\omega t - \frac{2\pi}{3}) \\
v_c &= \sqrt{2}E\cos(\omega t + \frac{2\pi}{3})
\end{aligned}
\qquad \textbf{(6.7)}
$$

where E is the rms value of the phase voltage. The values in equation (6.7) are presented in an a-b-c frame of reference. To analyse the system, these will be transformed into the rotating *d-q* frame using the transformation matrix in equation (6.8).

$$T = \sqrt{\frac{2}{3}} \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3/2} & -\sqrt{3/2} \end{bmatrix} \tag{6.8}$$

Equation (6.7) can then be represented by:

$$\begin{aligned} v_d &= \sqrt{3}E \\ v_q &= 0 \end{aligned} \tag{6.9}$$

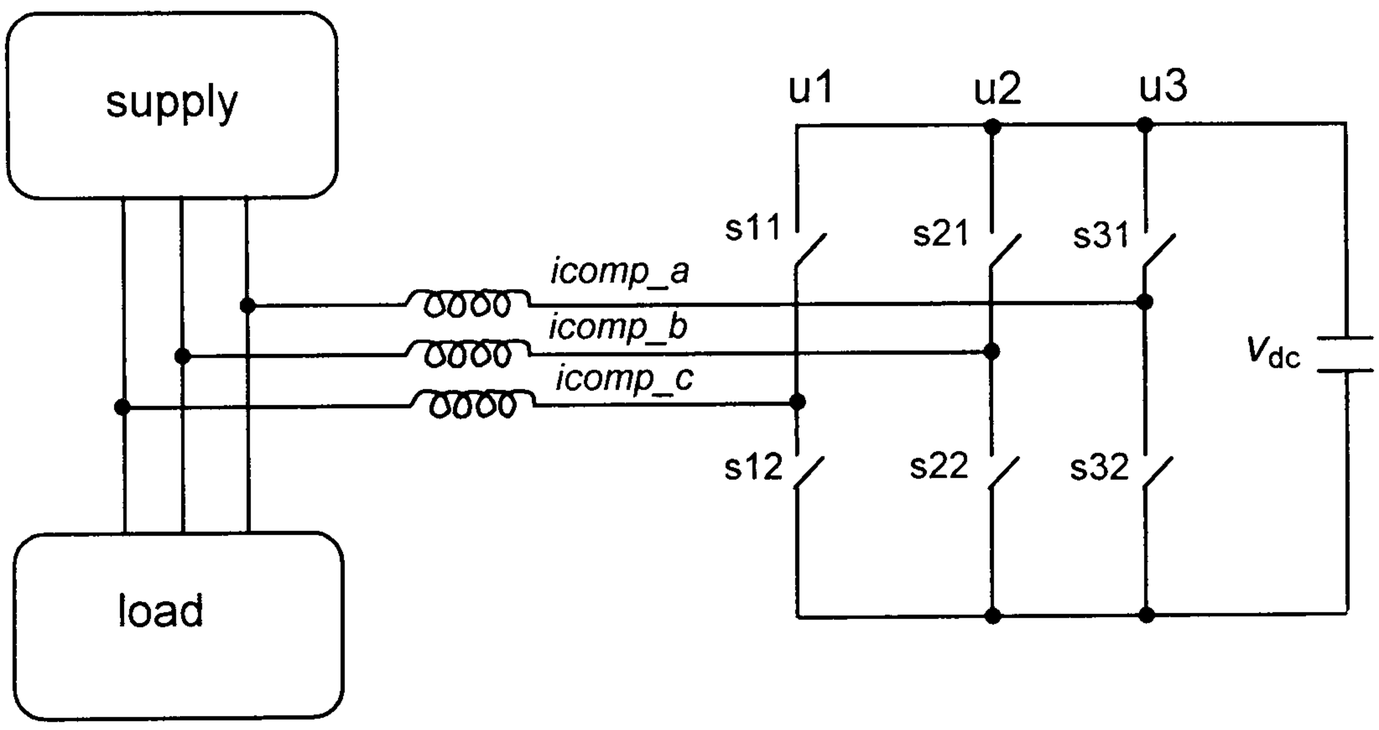


Figure 6.3 Simplified model of active power filter

The active power filter system in Figure 6.3 can be described by the following equations:

$$\begin{aligned} L\frac{di_{comp\_a}}{dt} &= v_{dc}u_1 - v_a \\ L\frac{di_{comp\_b}}{dt} &= v_{dc}u_2 - v_b \\ L\frac{di_{comp\_c}}{dt} &= v_{dc}u_3 - v_c \end{aligned} \tag{6.10}$$

where $\{i_{comp}\}$ are the compensation currents, v_{dc} represents the DC voltage across the capacitor and u_k (u_1, u_2, u_3) is a vector whose coordinates are the modulation functions of the corresponding switches. The vector u_k can take the values 1 or -1 :

$$u_k = \begin{cases} 1 & \text{if the switch } S_{k1} \text{ is ON (upper switch)} \\ -1 & \text{if the switch } S_{k2} \text{ is ON (lower switch)} \end{cases} \quad \textbf{(6.11)}$$

The current paths and DC voltage level depend on the switches in each leg of the inverter. Figure 6.4(a) shows a method of representing different switching modes using three axis coordinates. If the view is taken with state G and H on the same point, the diagram can be redrawn as shown in Figure 6.4(b). In the diagram, the stationary α-β axes and rotating *d-q* axes are also added.

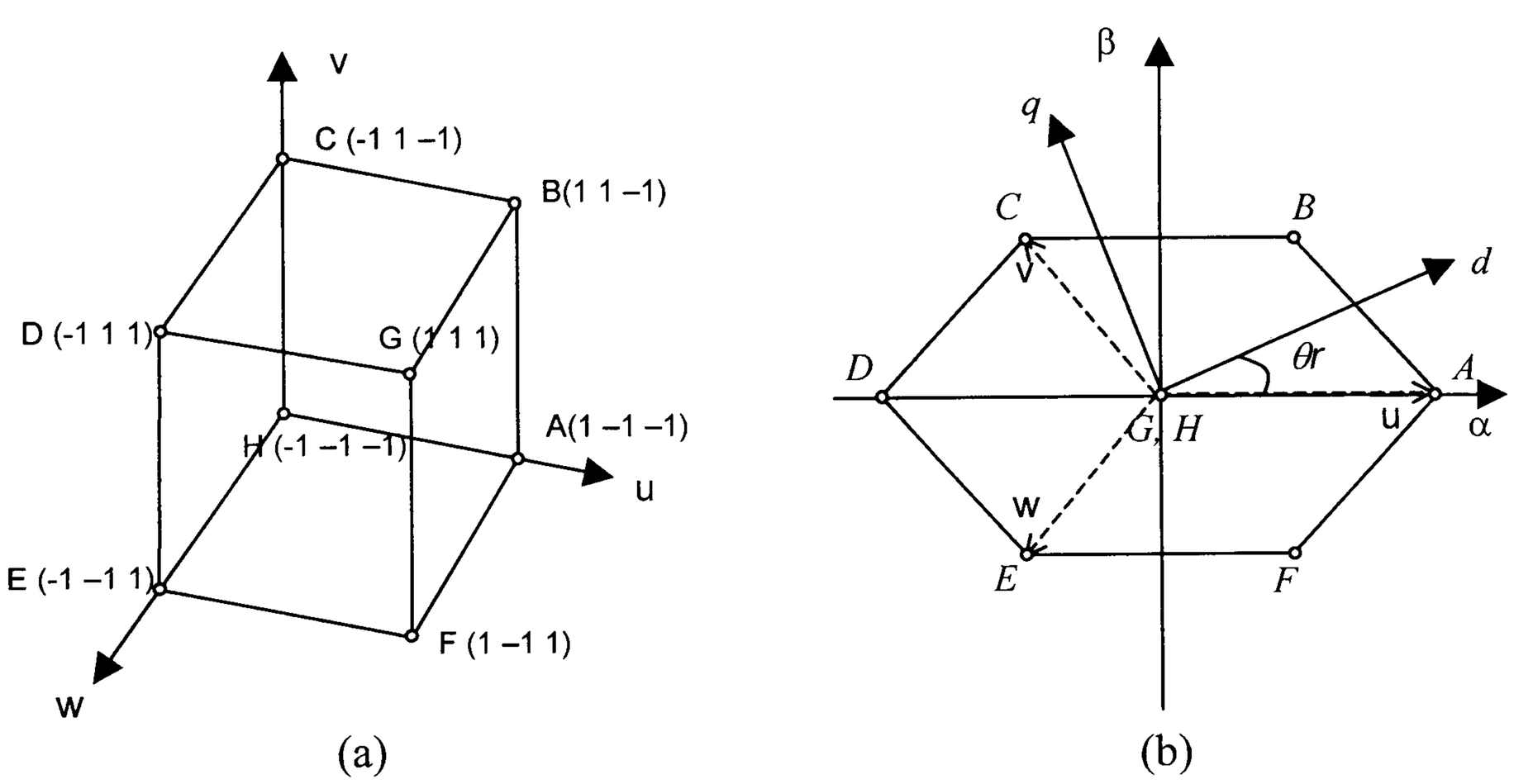


Figure 6.4 Switching states representation

Using this information, a mathematical model in the *d-q* frame of reference can then be determined to give the following equation:

$$\begin{bmatrix} \frac{di_{comp\_d}}{dt} \\ \frac{di_{comp\_q}}{dt} \\ \frac{div_{dc}}{dt} \end{bmatrix} = \begin{bmatrix} 0 & w & 0 \\ -w & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_{comp\_d} \\ i_{comp\_q} \\ v_{dc} \end{bmatrix} + \begin{bmatrix} \frac{v_{dc}}{L} & 0 \\ 0 & \frac{v_{dc}}{L} \\ \frac{comp\_d}{L} & \frac{comp\_d}{L} \end{bmatrix} \begin{bmatrix} u_d \\ u_q \end{bmatrix} - \begin{bmatrix} v_d \\ 0 \\ 0 \end{bmatrix} \quad \textbf{(6.12)}$$

6.3.2 Sliding Mode Control Design

To apply the sliding mode control theory to the active power filter, the sliding surfaces or the trajectories must first be defined. The aim of the design is to ensure that accurate tracking of the reference current is achieved. The reference current is determined using the IRPT concept described in earlier chapters. Thus, written in standard form for the sliding surfaces:

$$s = i_{ref} - i_{comp} \quad \textbf{(6.13)}$$

The conditions for the existence of the sliding mode can be expressed in the form

$$\begin{aligned} \lim_{s \to 0^-} \quad \dot{s} > 0 \\ \lim_{s \to 0^+} \quad \dot{s} < 0 \end{aligned} \quad \textbf{(6.14)}$$

or equivalently

$$s\dot{s} < 0 \quad \textbf{(6.15)}$$

From equation (6.12)

$$\begin{aligned} \frac{di_{ref\_d}}{dt} &= wi_{ref\_q} + u_d \frac{v_{dc}}{L} - v_d \\ \frac{di_{ref\_q}}{dt} &= -wi_{ref\_d} + u_q \frac{v_{dc}}{L} \end{aligned} \quad \textbf{(6.16)}$$

To achieve the sliding mode, the equation is set to zero to find the required control action, which gives:

$$u_d = \frac{1}{v_{dc}}\left[L\frac{di_{ref\_d}}{dt} - wLi_{ref\_q} + Lv_d \right]$$
$$u_q = \frac{1}{v_{dc}}\left[L\frac{di_{ref\_q}}{dt} + wLi_{ref\_q} \right] \quad \textbf{(6.17)}$$

The sign of the control components, u_d and u_q, which lead to the sliding mode are decided by the sign of control errors.

$$sgn(u_d) = sgn(e_d)$$
$$sgn(u_q) = sgn(e_q) \quad \textbf{(6.18)}$$

where

$$e_d = i_{ref\_d} - i_{comp\_d}$$
$$e_q = i_{ref\_q} - i_{comp\_q} \quad \textbf{(6.19)}$$

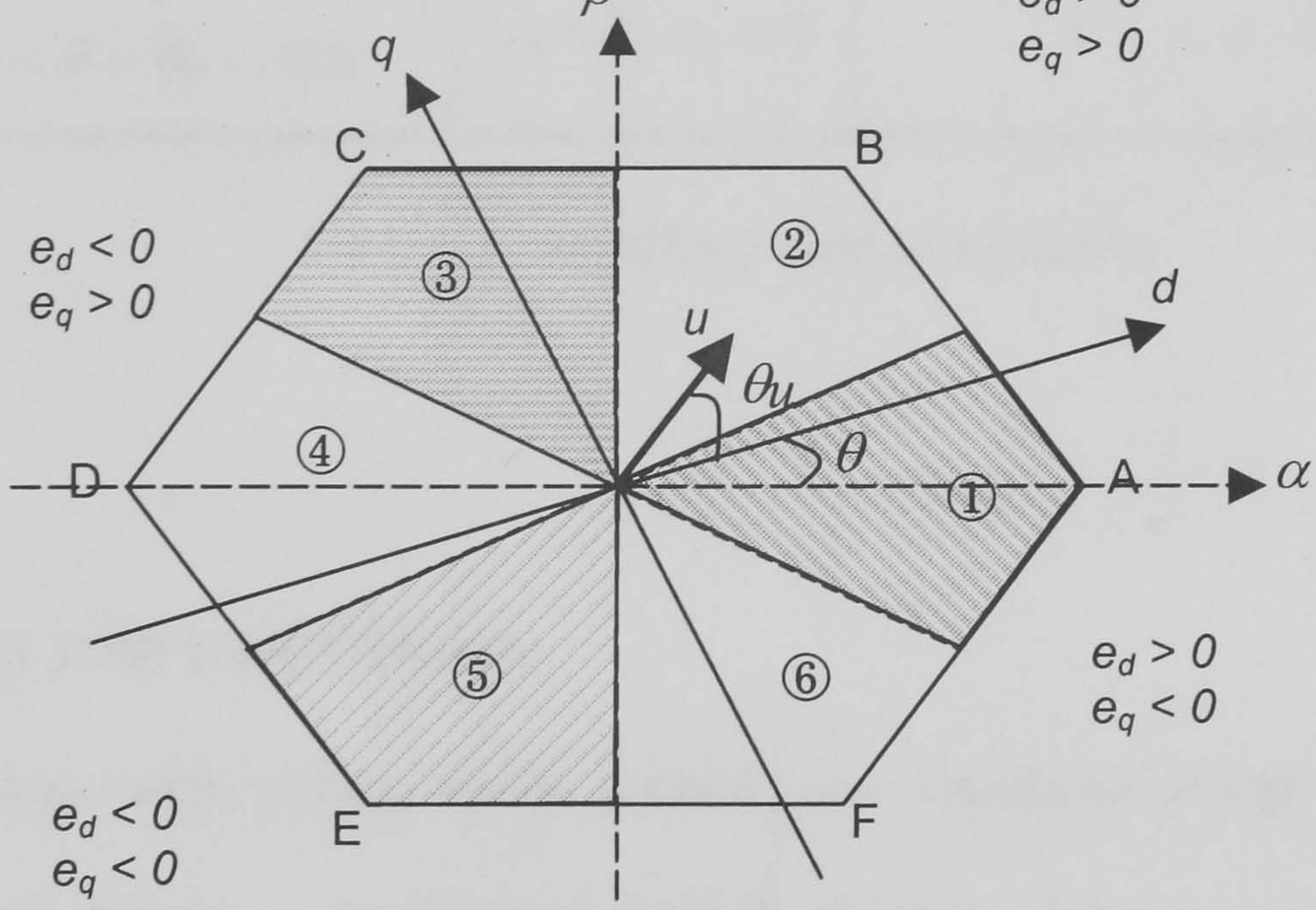


Figure 6.5 Different switching control regions

Using this information and knowledge of the relative displacement between the d-q frame and the stationary α-β frame, a switching strategy can then be designed to provide the required active power filter operation. Figure 6.5 shows how this can be achieved.

Regions ① to ⑥ correspond to 6 possible switching configurations. As an example consider when the position of the control vector u is as shown in Figure 6.5 with $e_d > 0$ and $e_q > 0$. It is located in region ②, hence the switching configuration B (1 1 -1) is chosen. Table 6.1 shows different switching modes for different values of displacement angle, θ. A similar approach is used to select the switching configuration for other possible combinations of u_d, u_q, e_d, e_q and θ.

range of $\theta + \theta_u$	e_d e_q	switching configurations
$330 < \theta + \theta_u < 30$	$e_d > 0$, $e_q > 0$	A (1 1 -1)
$30 < \theta + \theta_u < 90$	$e_d > 0$, $e_q > 0$	B (-1 1 -1)
$90 < \theta + \theta_u < 150$	$e_d > 0$, $e_q > 0$	C (-1 1 1)
$150 < \theta + \theta_u < 210$	$e_d > 0$, $e_q > 0$	D (-1 -1 1)
$210 < \theta + \theta_u < 270$	$e_d > 0$, $e_q > 0$	E (1 -1 1)
$270 < \theta + \theta_u < 330$	$e_d > 0$, $e_q > 0$	F (1 -1 -1)

Table 6.1 Switching control selection

6.4 RESULTS AND DISCUSSION

The APF system with sliding mode control was simulated using MATLAB with Simulink. Figure 6.6 shows the functional block diagram of the process involved. Once the reference current has been determined (as explained in previous chapters), the sliding mode control strategy is applied to obtain the control components which are then used to provide the gating signals for the power inverter circuit. Figure 6.7 shows the Simulink model used in the simulation.

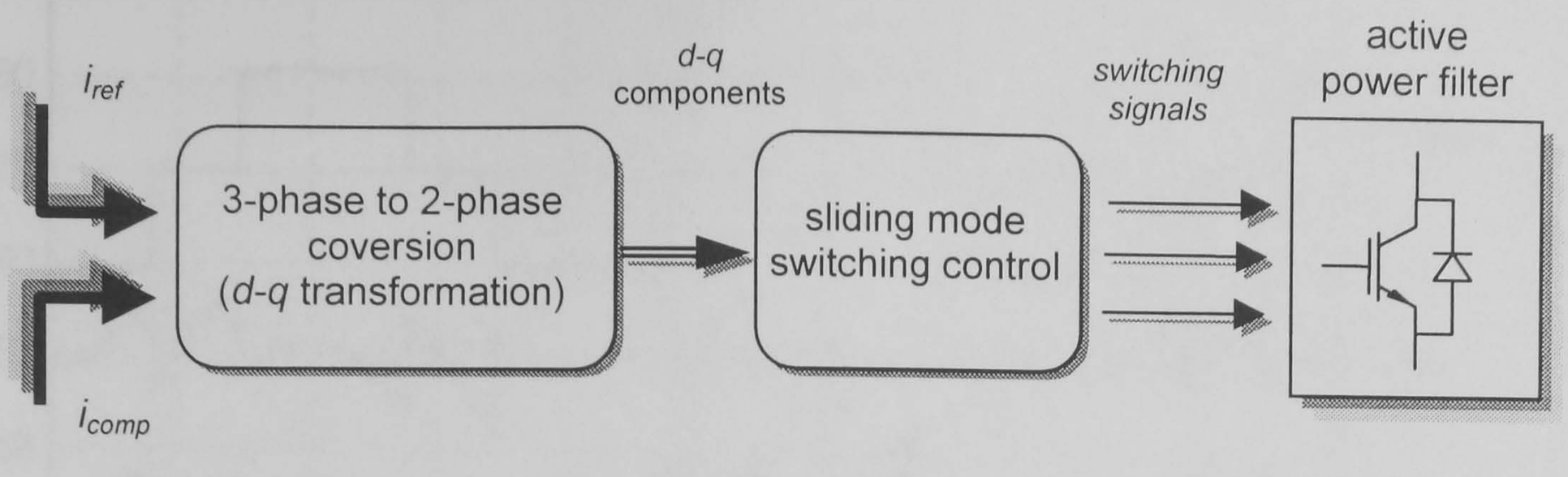


Figure 6.6 Implementation of sliding mode switching control

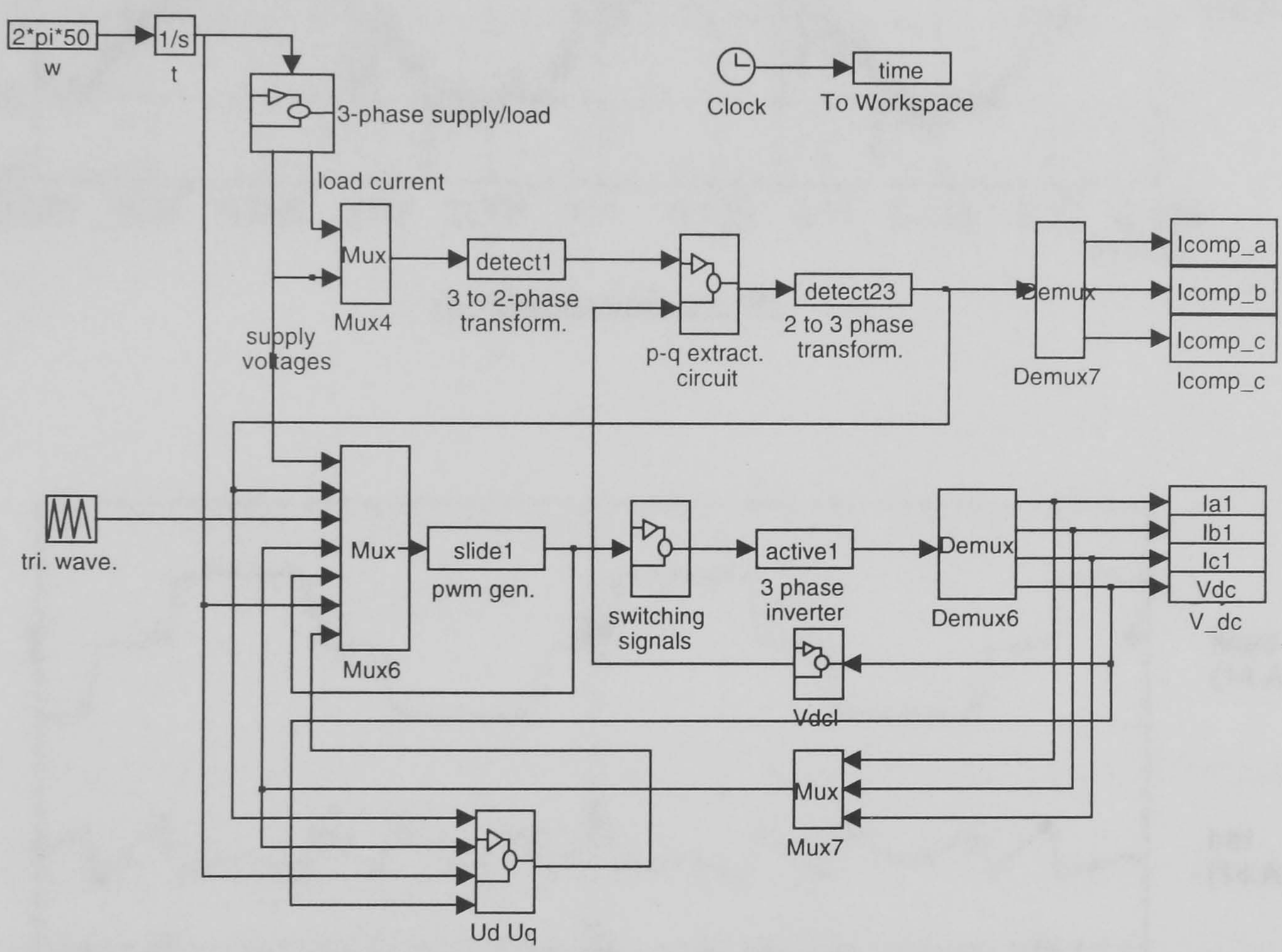
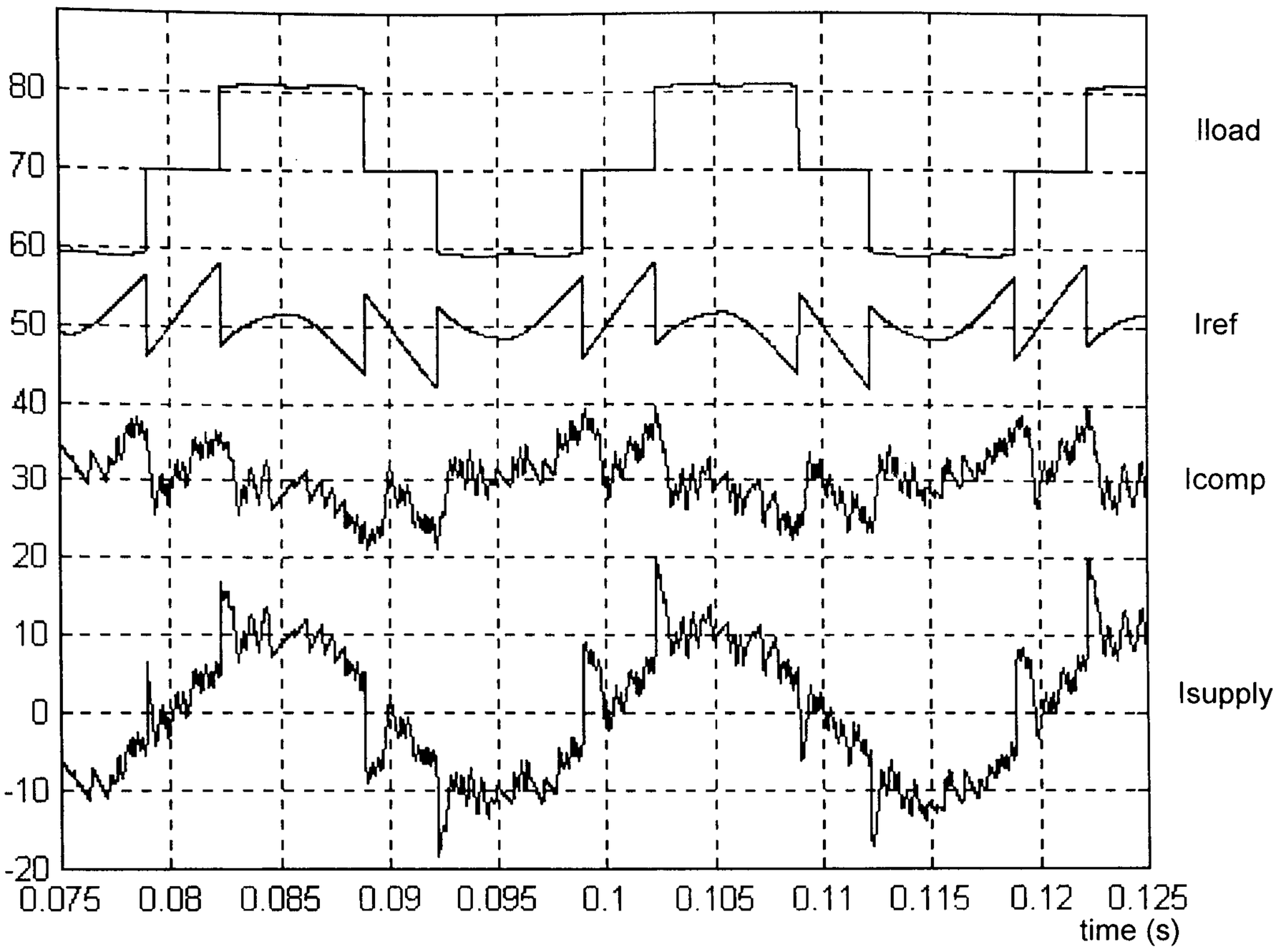
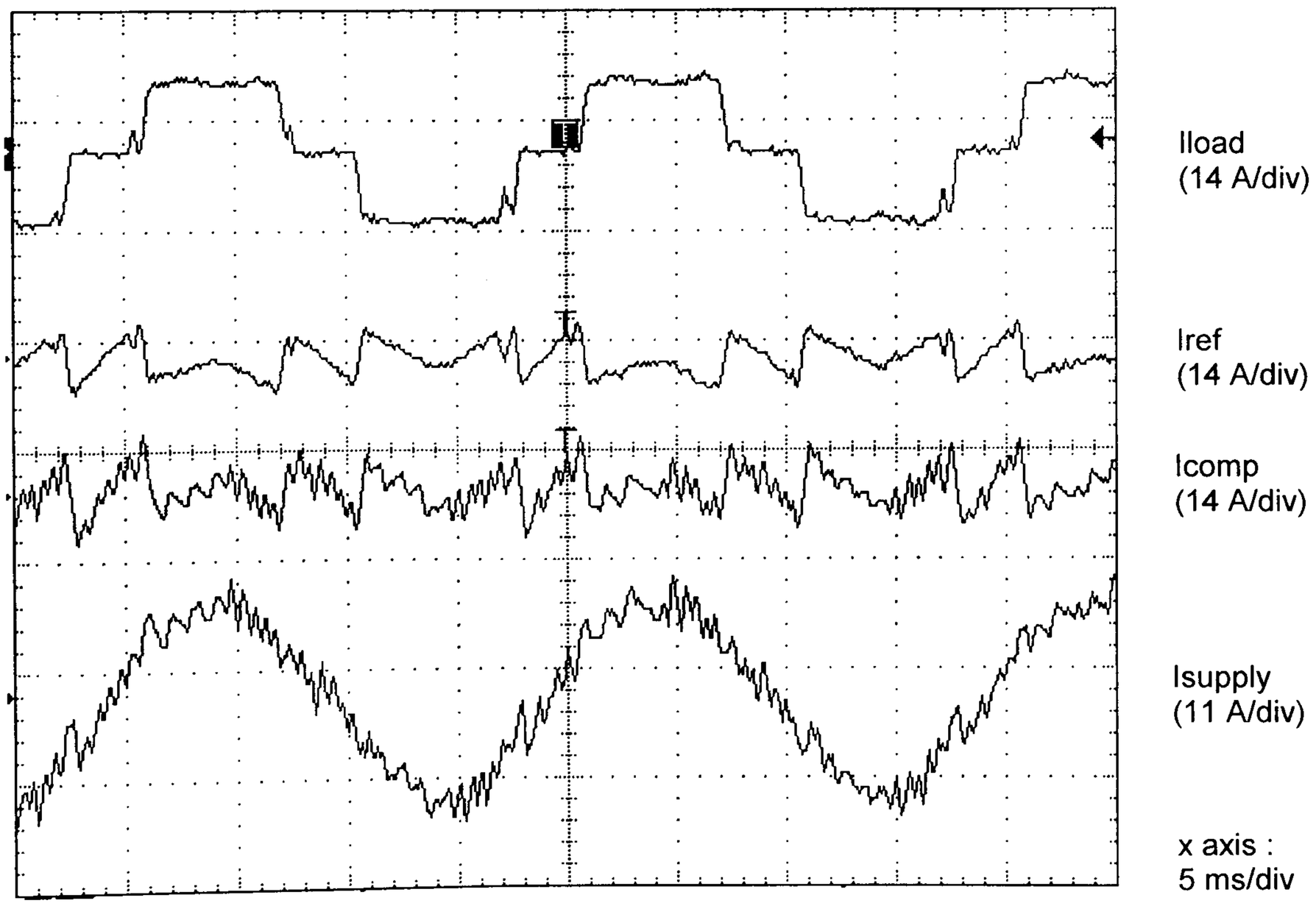


Figure 6.7 Simulink model for sliding mode control simulation

The 'slide1' s-function is used to determine the switching control for the inverter circuit. The calculation of e_d, e_q, u_d and u_q are performed in the 'Ud Uq' sub-system block. Sliding mode switching control has also been tested experimentally. A program was written implementing the control action in the DSP96000. The listing of the MATLAB m-files and the DSP assembly language program are included in the appendix.



(a) simulation results



(a) experimental results

Figure 6.8 Simulation and experimental results for sliding mode control of the active power filter

Figure 6.8 shows both simulation and experimental results. The results show that by using the sliding mode control concept, the active power filter tracks the reference current and provides the required compensation current to maintain a sinusoidal supply current with unity power factor. Figure 6.9 shows the resultant sinusoidal supply current which is in phase with the supply voltage.

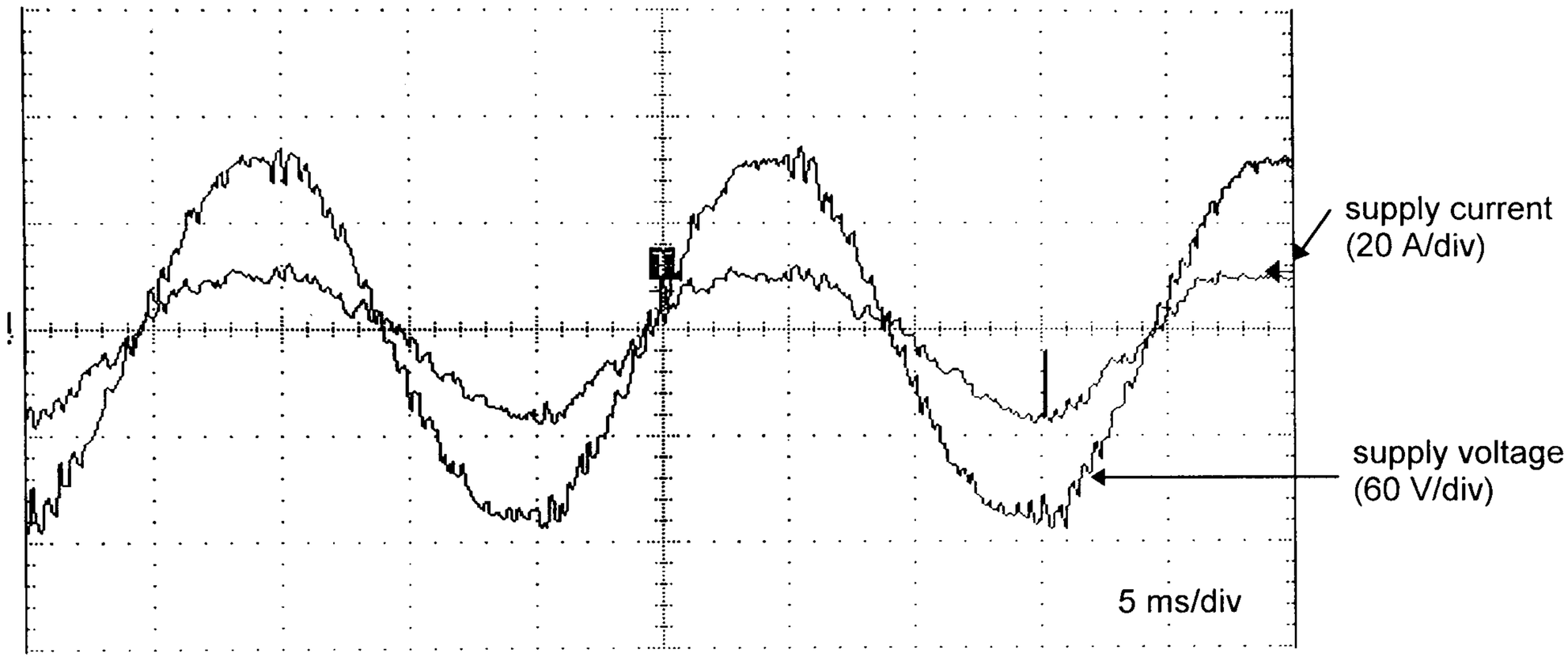


Figure 6.9 Experimental results - supply current is in phase with the supply voltage, i.e. unity p.f.

In comparison with the results obtained in Chapter 5 (with discrete delta modulation), the robust switching control strategy used has a drawback. The supply current shown in Figure 6.8 has a higher ripple contents. These results are obtained with a sampling frequency of 10 kHz. Further improvement can be made if the sampling frequency is increased. However, due to processor limitations, a higher sampling frequency was not possible.

6.5 SUMMARY

This chapter discussed the implementation of sliding mode switching control in an active power filter. Using sliding mode control theory, sets of control equations are derived to force the system to stay on the sliding surface. This ensures that the active power filter will track the reference current which was calculated using the IRPT method. The information obtained from u_d, u_q, e_d, e_q and θ is used to decide which set of switching configurations is to be selected. Simulation and experimental results proved that compensation can be achieved using the proposed sliding mode switching control.

6.6 REFERENCES

[6.1] V.I. Utkin, "Variable structure systems with sliding modes", IEEE Trans. on Automatic Control, Vol. AC-22 No. 2, April 1977, pp. 212-222.

[6.2] J.Y. Hung, W. Gao and J.C. Hung, "Variable structure control : A survey", IEEE Trans. on Industrial Electronics, Vol. 40, No.1, February 1993, pp. 2-22.

[6.3] V.I. Utkin, "Sliding mode control design principles and applications to electric drives", IEEE Trans. on Industrial Electronics, Vol. 40 No.1, February 1993, pp. 23-36.

[6.4] N. Sabanovic, A. Sabanovic and K. Ohnishi, "Sliding modes control of three phase switching power converters", Conference Proceedings, IECON 1992, pp. 319-324.

[6.5] M. Carpita and M. Marchesoni, "Experimental study of a power conditioning system using sliding mode control", IEEE Trans. on Power Electronics, Vol. 11 No. 5, September 1996, pp. 731-741.

[6.6] S.L. Jung and Y.Y Tzou, "Discrete sliding mode control of a PWM inverter for sinusoidal output waveform synthesis with optimal sliding curve", IEEE Trans. on Power Electronics, Vol. 11 No. 4, July 1996, pp. 567-577.

[6.7] L. Rosetto, G. Spiazzi, P. Tenti, B. Fabiano and C. Licitra, "Fast response high-quality rectifier with sliding mode control", IEEE Trans. on Power Electronics, Vol. 9 No. 2, March 1994, pp. 146-151.

[6.8] J.J.E. Slotine, "Sliding controller design for non-linear systems", Int. Journal Control, Vol. 40 No. 2, 1984, pp. 421-434.

[6.9] J.J.E Slotine and L. Weiping, "Applied Nonlinear Control", Prentice-Hall 1991.

[6.10] S. Saetieo, R. Devaraj and D.A. Torrey, "The design and implementation of a three-phase active power filter based on sliding mode control", IEEE Trans. on Industry Applications, Vol. 31 No. 5, September/October 1995, pp. 993-1000.

[6.11] P.F. Wojciak and D.A. Torrey, "The design and implementation of active power filters based on variable structure system concepts", Conf. Proceedings, IAS 1992, pp. 850-857.

[6.12] Z. Radulovic and A. Sabanovic, "Active filter control using a sliding mode approach", Conf. Proceedings, Power Electronics Specialist Conference 1994, pp. 177-182.

CHAPTER 7

CONCLUSIONS

7.1 CONCLUDING REMARKS

In this thesis, the design and operation of a static VAr compensator and active power filter has been presented. In particular, different switching control strategies have been considered. Design procedures have been described to implement the SVC and APF system using a shunt connected voltage source inverter. System simulations were performed using PSpice and MATLAB with Simulink which aided the design process and helped in the understanding and analysis of control aspects.

Different modulating techniques have been simulated and tested for the SVC system. Leading and lagging VArs are generated to compensate the reactive power thus maintaining a supply system with unity power factor. The use of deadbanding has improved system performance and minimised switching losses.

The operation of the APF system using modified discrete delta modulation has been described. Calculation of the reference compensation is performed in the time domain using the instantaneous reactive power theory concept. This gives precise reference waveforms for harmonic and reactive power compensation. The processing power requirements for complicated mathematical operations and system control have been achieved by using a DSP and a Xilinx FPGA. The inverter circuit used in the APF is a natural variable structure system because its topology changes with switching action. Using the equivalent control concept of a variable structure system with sliding mode, a

control law has been derived to ensure the reference current is followed. The scheme has been verified by simulation and experimental results. The APF system with discerete delta modulation gave better results in comparison with the system with sliding mode switching control. Both techniques enabled a supply system with unity power factor to be achieved but with higher ripple currents in the case of sliding mode technique.

7.2 AUTHOR'S CONTRIBUTION

The SVC and APF systems have been simulated using PSpice and MATLAB (with Simulink). System models were tested in both simulation packages to verify their accuracy. Significant improvement has been achieved in PSpice simulation times by generating the switching signals externally. A state-space approach was used to develop the model in MATLAB. The model enabled the effects of various modulation and switching strategies to be studied for harmonic and reactive power compensation.

A performance improvement was obtained in VAr compensation, with the application of deadbanding in the modulation waveforms. A comparison was made with sinusoidal modulating waveshapes. The use of deadband PWM enabled higher modulation indices to be achieved hence facilitating a smaller size of reactive component on the DC side. The deadband region is linked to the current peaks. Deadbanding also reduces the effective switching frequency. The combination of these two factors minimised the switching losses and resulting stresses.

The deadbanding concept has also been applied to a discrete delta modulator. It is used to provide switching control for the APF system. In the proposed scheme, a reduction in

switching losses is achieved with a slight increase in the total harmonic content. The detection circuit has been implemented in a DSP and is based on instantaneous reactive power theory. The use of variable structure control in the APF system has been explained. A model is described in the *d-q* synchronous rotating frame to derive the equations governing the switching control action. Sliding mode control has been used to track the reference current to provide the required compensation.

7.3 AREAS OF FURTHER WORK

Several possibilities for future research will be outlined. Essentially, two control loops are required in the operation of a compensator system. This thesis focused on the internal control loop which ensures the required VAr to be compensated or the reference current waveform to be followed. Detail study can be made of the outer control loop to maintain a constant DC voltage thus ensuring decoupled operation. There is also a possibility to set different DC voltage levels for different load conditions.

The detection circuit used for the research presented in this thesis required load currents and supply voltages to be sensed. A simpler system can be achieved if the same performance is obtained without the complex process of transformation, multiplications and divisions. The system will only require the measurement of the inverter output voltage. This in combination with sliding mode switching control, would result in a simple and robust system. Less mathematical processing will allow a higher sampling frequency thus reducing the control errors.

Various switching control strategies have been studied while operating under steady-state conditions. Although this is sufficient to verify the effectiveness of the proposed control method, control behaviour analysis under transient operating conditions may be an interesting study. Also the sliding mode control of the inverter switches can be readily extended to incorporate the deadbanding concept. Further investigation can be made to study the robustness of the sliding mode switching control under different load conditions. The control law presented in this thesis can be further improved to obtain better results.

At the distribution end of a supply system, unbalanced single phase loads and non-linear loads cause unequal voltage drops in the transformer and line impedances. The resultant unbalanced supply voltage can have undesirable effects on electric machinery. The compensator system can be used to provide active correction. It is possible to inject the required balancing current at the load side. Alternatively the system can be connected in series to inject a correcting phase voltage.

The application of an active power filter in an uninterruptible power supply (UPS) system can also be considered. It can be used to suppress harmonic disturbances as presented by loads connected to the UPS as well as providing transient back-up due to dips in supply.

APPENDIX A : PSPICE PROGRAM LISTINGS

A.1 SVC WITH SINUSOIDAL PWM

Voltage Source Inverter with Sinusoidal PWM

```
.OPTIONS ABSTOL 1UA ITL5 0 RELTOL 0.01 itl4=20
.PARAM G=100K

**********************************************************************
*                                                                    *
*                          3 phase inverter                          *
*                                                                    *
**********************************************************************

Cdc1    50  40  1100uF IC 640
Rdc1     1   50  0.05
Rdc11     1   40  44k

L1       7    2  6.44mH
L2       8    3  6.44mH
L3       9    4  6.44mH
R1      41    7  1.1
R2      42    8  1.1
R3      43    9  1.1
V1      41    0  SIN(0 240 50 0 0    0)
V2      42    0  SIN(0 240 50 0 0 -120)
V3      43    0  SIN(0 240 50 0 0  120)

D11      2    1  DIODE
D12     40    2  DIODE
S11      1 2 14 0 SWITCH
S12      2 40 15 0 SWITCH

D21      3    1  DIODE
D22     40    3  DIODE
S21      1 3 24 0 SWITCH
S22      3 40 25 0 SWITCH

D31      4    1  DIODE
D32     40    4  DIODE
S31      1 4 34 0 SWITCH
S32      4 40 35 0 SWITCH


**********************************************************************
*                                                                    *
*                          sinusoidal pwm                            *
*                                                                    *
**********************************************************************

*          ****************  1st phase  ********************

V10     10   0  SIN(0 6 50 0 0 10)
R10     10   0  10
V11     11   0  SIN(0 6 50 0 0 10)
R11     11   0  10

VTri11   12   0  PULSE(-9.999 9.999 0 250U 250U 100N 500.1U)
RTri11   12   0  10
```

```
Ecomp11  14   0  TABLE {G*(V(10,0)-V(12,0))}=(-1,-10) (1,10)
Rout11   14   0  10
Ecomp12  15   0  TABLE {G*(V(12,0)-V(11,0))}=(-1,-10) (1,10)
Rout12   15   0  10


*              ****************    2nd phase  ********************

V20      20   0  SIN(0 6 50 0 0 -110)
R20      20   0  10
V21      21   0  SIN(0 6 50 0 0 -110)
R21      21   0  10


Ecomp21  24   0  TABLE {G*(V(20,0)-V(12,0))}=(-1,-10) (1,10)
Rout21   24   0  10
Ecomp22  25   0  TABLE {G*(V(12,0)-V(21,0))}=(-1,-10) (1,10)
Rout22   25   0  10


*              ****************    3rd phase    *****************

V30      30   0  SIN(0 6 50 0 0 130)
R30      30   0  10
V31      31   0  SIN(0 6 50 0 0 130)
R31      31   0  10


Ecomp31  34   0  TABLE {G*(V(30,0)-V(12,0))}=(-1,-10) (1,10)
Rout31   34   0  10
Ecomp32  35   0  TABLE {G*(V(12,0)-V(31,0))}=(-1,-10) (1,10)
Rout32   35   0  10

.TRAN 100us 120mS  UIC
.MODEL SWITCH VSWITCH
.MODEL DIODE D(RS=1 CJO=.1N)
.PROBE V(41,0) I(R1) V(2,40) V(3,40) v(7,2)
+      I(Rdc1) V(1,40) V(40,0) V(2,0)

.END
```

A.2 SVC WITH TRIPLEN INJECTION PWM

Voltage Source Inverter with Triplen Injection PWM

```
.OPTIONS ABSTOL 1NA ITL4 20 ITL5 0 RELTOL .01
.PARAM G=100K DEL=356 T={10m/3}                ;del - phase shift
+      SEC={356m/18}                           ;delay time due to phase shift

* delay set at 356 to prevent convergence problem
* if at -4 ; no waveforms at all for the first 4 degrees

*****************************************************************
*                          3 phase inverter                     *
*****************************************************************
```

```
Cdc1    150  140  1100uF IC 545
Rdc     1    150  0.05
Rdc11   1    140  44k

L1      7    2  6.44mH
L2      8    3  6.44mH
L3      9    4  6.44mH
R1      141   7  1.1
R2      142   8  1.1
R3      143   9  1.1
V1      141   0  SIN(0 240 50 0 0    0)
V2      142   0  SIN(0 240 50 0 0 -120)
V3      143   0  SIN(0 240 50 0 0  120)

D11     2    1  DIODE
D12     140    2  DIODE
S11     1 2 80 0 SWITCH
S12     2 140 81 0 SWITCH

D21     3    1  DIODE
D22     140    3  DIODE
S21     1 3 82 0 SWITCH
S22     3 140 83 0 SWITCH

D31     4    1  DIODE
D32     140    4  DIODE
S31     1 4 84 0 SWITCH
S32     4 140 85 0 SWITCH

**********************************************************************
*                          reference waveforms                       *
**********************************************************************

V10     10   0  SIN(0 1 50 0 0 {30+DEL})
R10     10   0  10
V11     11   0  SIN(0 1 50 0 0 {-30+DEL})
R11     11   0  10
V12     12   0  SIN(0 1 50 0 0 {-150+DEL})
R12     12   0  10
V13     13   0  SIN(0 1 50 0 0 {-210+DEL})
R13     13   0  10
V14     14   0  SIN(0 1 50 0 0 {-90+DEL})
R14     14   0  10
V15     15   0  SIN(0 1 50 0 0 {-270+DEL})
R15     15   0  10
V17     17   0  SIN(0 1 50 0 0 {150+DEL})
R17     17   0  10
V18     18   0  SIN(0 1 50 0 0 {90+DEL})
R18     18   0  10

********************* carrier frequency ************************

VTRI    19   0  PULSE(-9.999 9.999 0 250U 250U 100n 500.1U)
RTRI    19   0  10

**********************************************************************
*                          triplen  injection pwm                    *
**********************************************************************
```

```
*              ***************  1st phase  ********************

VP1     20   0  PULSE(0 1 {-SEC} 1e-19 1e-19 {T} 0.02)
RP1     20   0  10
VP2     21   0  PULSE(0 1 {(2*T)-SEC} 1e-19 1e-19 {T} 0.02)
RP2     21   0  10
VP3     22   0  PULSE(0 -1 {10m-SEC} 1e-19 1e-19 {T} 0.02)
RP3     22   0  10
VP4     23   0  PULSE(0 -1 {10m+(2*T)-SEC} 1e-19 1e-19 {T} 0.02)
RP4     23   0  10

ES1     24   0  VALUE = {(2*V(10) - 1)*V(20)}
ES2     25  24  VALUE = {(2*V(11) - 1)*V(21)}
ES3     26  25  VALUE = {(2*V(12) - 1)*V(22)}
ES4     27  26  VALUE = {(2*V(13) - 1)*V(23)}
VP5     28  27  PULSE(0 1 {T-SEC} 1e-19 1e-19 {T} 0.02)
VP6     29  28  PULSE(0 -1 {10m+T-SEC} 1e-19 1e-19 {T} 0.02)
R29     29   0  10
E30     30   0  VALUE = {V(29)*10}
R30     30   0  10

************************  2nd phase  ********************************

VP11    40   0  PULSE(0 -1 {T-SEC} 1e-19 1e-19 {T} 0.02)
RP11    40   0  10
VP12    41   0  PULSE(0 1  {(2*T)-SEC} 1e-19 1e-19 {T} 0.02)
RP12    41   0  10
VP13    42   0  PULSE(0 1 {10m+T-SEC} 1e-19 1e-19 {T} 0.02)
RP13    42   0  10
VP14    43   0  PULSE(0 -1 {10m+(2*T)-SEC} 1e-19 1e-19 {T} 0.02)
RP14    43   0  10

ES11    44   0  VALUE = {(2*V(10) - 1)*V(40)}
ES12    45  44  VALUE = {(2*V(14) - 1)*V(41)}
ES13    46  45  VALUE = {(2*V(12) - 1)*V(42)}
ES14    47  46  VALUE = {(2*V(15) - 1)*V(43)}
VP15    48  47  PULSE(0 1 {10m-SEC} 1e-20 1e-19 {T} 0.02)
VP16    49  48  PULSE(0 -1 {-SEC} 1e-19 1e-19 {T} 0.02)
R49     49   0  10
E50     50   0  VALUE = {V(49)*10}
R50     50   0  10

******************************** 3rd phase  **************************

VP21    60   0  PULSE(0 1 {-SEC} 1e-19 1e-19 {T} 0.02)
RP21    60   0  10
VP22    61   0  PULSE(0 -1 {T-SEC} 1e-19 1e-19 {T} 0.02)
RP22    61   0  10
VP23    62   0  PULSE(0 -1 {10m-SEC} 1e-19 1e-19 {T} 0.02)
RP23    62   0  10
VP24    63   0  PULSE(0 1 {10m+T-SEC} 1e-19 1e-19 {T} 0.02)
RP24    63   0  10

ES21    64   0  VALUE = {(2*V(18) - 1)*V(60)}
ES22    65  64  VALUE = {(2*V(11) - 1)*V(61)}
ES23    66  65  VALUE = {(2*V(14) - 1)*V(62)}
ES24    67  66  VALUE = {(2*V(17) - 1)*V(63)}
VP25    68  67  PULSE(0 -1 {(2*T)-SEC} 1e-19 1e-19 {T} 0.02)
VP26    69  68  PULSE(0 1 {10m+(2*T)-SEC} 1e-19 1e-19 {T} 0.02)
```

```
R69     69  0  10
E70     70  0  VALUE = {V(69)*10}
R70     70  0  10

**********************************************************************
*                          pwm for triplen injection                 *
**********************************************************************

ECOMP11  80  0  TABLE {G*(V(30)-V(19))}=(-1,0) (1,10)
ROUT11   80  0  10
ECOMP21  81  0  TABLE {G*(V(19)-V(30))}=(-1,0) (1,10)
ROUT21   81  0  10

ECOMP12  82  0  TABLE {G*(V(50)-V(19))}=(-1,0) (1,10)
ROUT12   82  0  10
ECOMP22  83  0  TABLE {G*(V(19)-V(50))}=(-1,0) (1,10)
ROUT22   83  0  10

ECOMP13  84  0  TABLE {G*(V(70)-V(19))}=(-1,0) (1,10)
ROUT13   84  0  10
ECOMP23  85  0  TABLE {G*(V(19)-V(70))}=(-1,0) (1,10)
ROUT23   85  0  10


.TRAN 100US 120mS UIC

.MODEL SWITCH VSWITCH(von=8 voff=1)
.MODEL DIODE D(RS=1 CJO=.1N)

.PROBE  V(141,0) V(1,140) V(2) V(2,3) I(R1) I(R2) I(Rdc) V(2,40)

.END
```

A.3 PSPICE MODEL WITH EXTERNALLY GENERATED SWITCHING SIGNALS

Active Power Filter With Modified Delta Modulation

```
.PARAM PERIOD={1/50}, DEG120={1/(3*50)}, DELAY={ 10}, G=100K
.PARAM ALFA={((30+DELAY)/180)*10m}, PULSE_WIDTH=0.1m

.OPTIONS ABSTOL 1uA ITL4 40 ITL5 0 RELTOL .05

**********************************************************************
*                          thyristor controlled load                 *
**********************************************************************

RL1    441    4117    0.01
Ll1    4117   4114    0.9m
RL2    442    4118    0.01
Ll2    4118   4115    0.9m
RL3    443    4119    0.01
Ll3    4119   4116    0.9m

LD     4122   4121     25mH   IC=45A
RD     4120   4122     12
```

```
XTHY1   4114    4120     SCR    PARAMS:TDLY={0+ALFA}                 ICGATE=2V
vsense1   41114   4120     0
XTHY3   4115    4120     SCR    PARAMS:TDLY={DEG120+ALFA}            ICGATE=0V
vsense3   41115   4120     0
XTHY5   4116    4120     SCR    PARAMS:TDLY={(2*DEG120)+ALFA}        ICGATE=0V
vsense5   41116   4120     0
XTHY2   4121    4116     SCR    PARAMS:TDLY={(DEG120/2)+ALFA}        ICGATE=0V
vsense2   41117   4116     0
XTHY4   4121    4114     SCR    PARAMS:TDLY={(3*DEG120/2)+ALFA}      ICGATE=0V
vsense4   41118   4114     0
XTHY6   4121    4115     SCR    PARAMS:TDLY={(5*DEG120/2)+ALFA}      ICGATE=2V
vsense6   41119   4115     0

V441     441    0  SIN(0 230 50 0 0    0)
V442     442    0  SIN(0 230 50 0 0 -120)
V443     443    0  SIN(0 230 50 0 0  120)


.SUBCKT SCR 4101 4103 PARAMS: TDLY=1ms ICGATE=0V
SW         4101    4102    453     0      switch
VSENSE  4102    4103    0V
RSNUB   4101    4104    200
CSNUB   4104    4103    2uF

VGATE   451     0       PULSE(0 1V {TDLY} 0 0 {PULSE_WIDTH} {PERIOD})
RGATE   451     0       1MEG
EGATE   452     0       TABLE {I(VSENSE)+V(451)} = (0.0,0.0) (0.1,1.0) (1.0,1.0)
RSER    452     453     1
CSER    453     0       1uF     IC={ICGATE}

.MODEL  SWITCH  VSWITCH(RON=0.01)
.ENDS
*****************************************************************
*                        3 phase voltage source  inverter                        *
*****************************************************************

Cdc1     50  40  1100uF
Rdc1      1   50  0.01

L1       7    2  6.45mH
L2       8    3  6.45mH
L3       9    4  6.45mH

R1      41    7  1.1
R2      42    8  1.1
R3      43    9  1.1
V1      41    0  SIN(0 230 50 0 0    0)
V2      42    0  SIN(0 230 50 0 0 -120)
V3      43    0  SIN(0 230 50 0 0  120)

S11      1 2 10 0 SWITCH
S12      2 40 11 0 SWITCH

S21      1 3 20 0 SWITCH
S22      3 40 21 0 SWITCH

S31      1 4 30 0 SWITCH
S32      4 40 31 0 SWITCH

.INC " sign921.dat"
R11 10 0 1K
```

```
E12 11 0 VALUE = {-1*V(10)+10}
R12 11 0 1K

.INC " sign922.dat"
R20 20 0 1k
E22  21 0 VALUE = {-1*V(20)+10}
R22 21 0 1K

.INC " sign923.dat"
R30 30 0 1K
E32 31 0 VALUE = {-1*V(30)+10}
R32 31 0 1K

.MODEL SWITCH VSWITCH(ron=0.01 von=8 voff=2)
.MODEL DIODE D(RS=0.05 CJO=.1N)

.TRAN 1u 100mS 0 50u uic
.PROBE  V(41,0) V(1,40) V(1,2) V(2) V(2,3) I(R1) I(R2) I(R3) I(Rdc1)
+      V(10) V(20) v(11) v(30) v(42,0) v(43,0) I(S11) I(S21) I(S31)
+      I(Ll1) I(Ll2) I(Ll3) V(3,40)
.END
```

APPENDIX B : C PROGRAM LISTINGS

```
/*************************************************************************
*                          File      : svcpwm.c                          *
*                          Function  : main program                      *
*                          Modified  : Mohd Fadzil Mohd Siam             *
*                                      Nov 1995                          *
*************************************************************************/

#include<math.h>
#include<stdio.h>


#define REF_PERIOD  (double)1/50        /* time for 1 cycle,50 Hz          */
#define CARRIER_PER (double)1/5000      /* sampling frequency,5 kHz        */
#define NUM_CYCLES   (int)6             /* no. of cycles                   */
#define MOD_INDEX   (float)0.8          /* modulation index                */
#define PHASE_ANG   (float)5            /* phase angle between V & I       */

double t[3500];
double p[3500];

void main()

{
void create_pwl();                      /* convert switching times to PWL for PSPICE   */
int pwm_sig();                          /* compute switching times i.e on/off          */
double deadband();                      /* create ref. waveforms with deadband         */
int pwm_time;                           /* compute pwm times and return its duration   */
void plot_sig();

int cycles=NUM_CYCLES;
double pi = (double)4*atan(1.0);
double theta=(double)PHASE_ANG*pi/180;
double T_ref=REF_PERIOD;
double T_c=CARRIER_PER;
float mod_index=MOD_INDEX;
float  phase=0;


/*****************  for 1st phase ,  angle=0  ***************************/
pwm_time = pwm_sig(cycles, T_ref, T_c, mod_index, phase,theta);
create_pwl("PWLFILE0.OUT",pwm_time, t, phase);
/*printf(" %.8f \n ", peak[1]);
getchar();*/
plot_sig("SIGNAL0.OUT",pwm_time, T_c, p, phase);          /*
printf(" phase  =  %g \n", phase);
printf("press any key to continue");
getchar();                                 */
phase++;

/*****************  for 2nd phase ,  angle= 120 ***************************/
pwm_time = pwm_sig(cycles, T_ref, T_c, mod_index, phase,theta);
create_pwl("PWLFILE2.OUT",pwm_time, t, phase);
plot_sig("SIGNAL2.OUT", pwm_time, T_c, p, phase);                  /*
printf(" phase  = %g \n",phase);
printf("press any key to continue");
getchar();                                  */
phase++;
```

```
/****************** for 3rd phase ,  angle=-120 ************************/
pwm_time = pwm_sig(cycles, T_ref, T_c, mod_index, phase,theta);
create_pwl("PWLFILE1.OUT", pwm_time, t, phase);
plot_sig("SIGNAL1.OUT", pwm_time, T_c, p, phase);                    /*
getchar();                                                           */

}


/*****************************************************************************
*                          File        : sinusoid.c                             *
*                          Function    : produce sinusoidal reference waveform  *
*                          Modified :  Mohd Fadzil Mohd Siam                    *
*                                      Nov 1995                                 *
*****************************************************************************/

#include<stdio.h>
#include<math.h>

double deadband(P, phase, mod_index, T_c, T_ref, theta)
double T_c, T_ref, theta;
float mod_index, phase, P;


  {
        double ang_freq, pi=(double)4*atan(1.0);
        double Va, Vb, Vc, Ia, Ib, Ic;
        double Vac, Vbc, Vcc;
        double d120, Vref, del;
        double fabs(), sin();

        ang_freq = 2*pi/T_ref;
        d120 = 2*pi/3;
        del = 70*pi/180;

        Va = mod_index*sin(P*ang_freq*T_c  + del);
        Vb = mod_index*sin(P*ang_freq*T_c  + del - d120);
        Vc = mod_index*sin(P*ang_freq*T_c  + del + d120);


        if(phase==0) Vref = Va;
        if(phase==1) Vref = Vc;
        if(phase==2) Vref = Vb;
        /*
        printf("phase = %g \n", phase);
        printf("Vref =  %g \n", Vref); */

        return(Vref);

  }


/*****************************************************************************
*                          File        : pwm_sig.c                              *
*                          Function    : calculate switching times              *
*                          Modified :  Mohd Fadzil Mohd Siam                    *
*                                      Nov 1995                                 *
*****************************************************************************/
```

```
#include<stdio.h>
#include<math.h>

extern double t[3500], p[3500];

int pwm_sig(cycles, T_ref, T_c, mod_index, phase,theta)
int cycles;
double T_ref, T_c, theta;
float phase, mod_index;
        {
        float N;
        float  P, on, off;
        double deadband();


        for(N=0;(N/2*T_c)<(cycles*T_ref);N=N+2)
                {
                P=(float) N/2;

                on=(float) T_c/4*(1.0 - deadband(P,phase,mod_index,T_c, T_ref, theta));
                t[N+1]=(double) P*T_c + on;
                p[N+1]=deadband(P,phase,mod_index,T_c,T_ref,theta);

                P=P+0.5;
                off=(float) T_c/4*(1.0 - deadband(P,phase, mod_index,T_c, T_ref, theta));
                t[N+2]=(double) (P+0.5)*T_c - off;
                p[N+2]=deadband(P,phase,mod_index,T_c,T_ref,theta);
                }
                /*
                printf(" off= %g \n",p[1]);
                getchar();
                        */

        return(P);
        }


/*****************************************************************************
*                       File        : plot_sig.c                              *
*                       Function  : store reference waveform for PSPICE       *
*                       Modified :  Mohd Fadzil Mohd Siam                     *
*                                   Nov 1995                                  *
*****************************************************************************/

#include <stdio.h>
#include <math.h>


void plot_sig( f_name, length, T_c, max, phase)
char f_name[];
int length;
double max[], T_c;
float phase;

{
FILE *fopen();
int fclose(), a;
FILE *stream;
double t1;
```

```
stream=fopen(f_name,"w");
if (phase==0) fprintf (stream, "V100 10 0 PWL ( \n");   /*   create   */
if (phase==1) fprintf (stream, "V300 20 0 PWL ( \n");   /*    file    */
if (phase==2) fprintf (stream, "V200 30 0 PWL ( \n");   /*   headers  */


for (a=1;a<(2*length);a=a+1)
            {

            t1=a*T_c/2;
            fprintf (stream," + %.8f %.8f \n ", t1, max[a]);

                                                        /*
            printf(" value = %d \n",a);
            printf(" max = %f \n",t2);
            getchar();                          */
            }
fprintf (stream," )");
fclose (stream);
return;
}



/*****************************************************************************
*                                                                            *
*                       File        : deadband.c                             *
*                       Function    : produce deadband reference waveform    *
*                       Modified :  Mohd Fadzil Mohd Siam                    *
*                                   Nov 1995                                 *
*                                                                            *
*                                                                            *
*****************************************************************************/

#include<stdio.h>
#include<math.h>

double deadband(P, phase, mod_index, T_c, T_ref, theta)
double T_c, T_ref, theta;
float mod_index, phase, P;


  {
        double ang_freq, pi=(double)4*atan(1.0);
        double Va, Vb, Vc, Ia, Ib, Ic;
        double Vac, Vbc, Vcc;
        double d120, Vref, del;
        double fabs(), sin(), d60;
        double V1, V2, IL1, IL2, IL3;

        ang_freq = 2*pi/T_ref;
        d120 = 2*pi/3;
        d60 = pi/3;
        del = -85*pi/180;

        Ia = sin(P*ang_freq*T_c + del + theta);
        Ib = sin(P*ang_freq*T_c + del + theta - d120);
        Ic = sin(P*ang_freq*T_c + del + theta + d120);

        Va = mod_index*sin(P*ang_freq*T_c  + del);
        Vb = mod_index*sin(P*ang_freq*T_c  + del - d120);
```

```
Vc = mod_index*sin(P*ang_freq*T_c  + del + d120);

V1 = sin(P*ang_freq*T_c);
V2 = sin(P*ang_freq*T_c - d60);

if( (0<V1<0.866)&&(0<V2<0.866) )
        {
        IL1 = 0;
        IL2 = -sin(P*ang_freq*T_c + d60);
        IL3 = -IL2;
        }

if( (V1>0.866)&&(0<V2<0.866) )
        {
        IL1 = sin(P*ang_freq*T_c);
        IL2 = -IL1;
        IL3 = 0;
        }

if( (0<V1<0.866)&&(V2>0.866) )
        {
        IL1 = sin(P*ang_freq*T_c - d60);
        IL2 = 0;
        IL3 = -IL1;
        }

if( (-0.866<V1<0)&&(0<V2<0.866) )
        {
        IL1 = 0;
        IL2 = -sin(P*ang_freq*T_c + d60);
        IL3 = -IL2;
        }

if( (V1<-0.866)&&(-0.866<V2<0) )
        {
        IL1 = sin(P*ang_freq*T_c);
        IL2 = IL1;
        IL3 = 0;
        }

if( (-0.866<V1<0)&&(V2<-0.866) )
        {
        IL1 = sin(P*ang_freq*T_c - d60);
        IL2 = 0;
        IL3 = -IL1;
        }


if( (Va>Vb)&&(Vb>Vc) )
        {
        if( fabs(Ia)>fabs(Ic) )
                {
                Vac = 1.0;
                Vbc = 1.0-(Va-Vb);
                Vcc = 1.0-(Va-Vc);
                }
        else
                {
                Vcc = -1.0;
                Vac = -1.0+(Va-Vc);
                Vbc = -1.0+(Vb-Vc);
```

```
                    }
            }

if( (Va>Vc)&&(Vc>Vb) )
            {
            if( fabs(Ia)>fabs(Ib) )
                    {
                    Vac = 1.0;
                    Vbc = 1.0-(Va-Vb);
                    Vcc = 1.0-(Va-Vc);
                    }
            else
                    {
                    Vbc = -1.0;
                    Vac = -1.0+(Va-Vb);
                    Vcc = -1.0+(Vc-Vb);
                    }
            }

if( (Vb>Va)&&(Va>Vc) )
            {
            if( fabs(Ib)>fabs(Ic) )
                    {
                    Vbc = 1.0;
                    Vac = 1.0-(Vb-Va);
                    Vcc = 1.0-(Vb-Vc);
                    }
            else
                    {
                    Vcc = -1.0;
                    Vac = -1.0+(Va-Vc);
                    Vbc = -1.0+(Vb-Vc);
                    }
            }

if( (Vb>Vc)&&(Vc>Va) )
            {
            if( fabs(Ib)>fabs(Ia) )
                    {
                    Vbc = 1.0;
                    Vac = 1.0-(Vb-Va);
                    Vcc = 1.0-(Vb-Vc);
                    }
            else
                    {
                    Vac = -1.0;
                    Vbc = -1.0+(Vb-Va);
                    Vcc = -1.0+(Vc-Va);
                    }
            }

if( (Vc>Va)&&(Va>Vb) )
            {
            if( fabs(Ic)>fabs(Ib) )
                            {
                            Vcc = 1.0;
                            Vac = 1.0-(Vc-Va);
                            Vbc = 1.0-(Vc-Vb);
                            }
            else
                            {
```

```
                                        Vbc = -1.0;
                                        Vac = -1.0+(Va-Vb);
                                        Vcc = -1.0+(Vc-Vb);
                                        }
                }

if( (Vc>Vb)&&(Vb>Va) )
                {
                if( fabs(Ic)>fabs(Ia) )
                                {
                                Vcc = 1.0;
                                Vac = 1.0-(Vc-Va);
                                Vbc = 1.0-(Vc-Vb);
                                }
                else
                                {
                                Vac = -1.0;
                                Vcc = -1.0+(Vc-Va);
                                Vbc = -1.0+(Vb-Va);
                                }
                }

if(phase==0) Vref = IL1;
if(phase==1) Vref = IL3;
if(phase==2) Vref = IL2;
/*
printf("phase = %g \n", phase);
printf("Vref =  %g \n", Vref); */

return(Vref);

}
```

```
/*****************************************************************************
*                       File      : crt_pwl.c                                 *
*                       Function  : convert switching times to PWL for PSPICE *
*                       Modified  : Mohd Fadzil Mohd Siam                     *
*                                   Nov 1995                                  *
*****************************************************************************/

#include <stdio.h>
#include <math.h>

void create_pwl( f_name, length, c,  phase)
char f_name[];
int length;
double c[];
float phase;

{
FILE *fopen();
int fclose(), n, P1, P2;
FILE *stream;
double t1, t2, Tr, Tmin;

Tr = 100e-9;            /* rise time for PWL */
Tmin = 2e-6;            /*    min. on time   */
P1 = 0;                 /*    switching      */
```

```
P2 = 10;                        /*      signals         */

stream=fopen(f_name,"w");
if (phase==0) fprintf (stream, "V10 10 0 PWL ( \n");   /*    create     */
if (phase==1) fprintf (stream, "V30 30 0 PWL ( \n");   /*     file      */
if (phase==2) fprintf (stream, "V20 20 0 PWL ( \n");   /*    headers    */

c[0]=-5*Tr;
for (n=1;n<=2*length;n=n+1)
            {
/*          printf("value of Tr %g \n",Tr);
            getchar();                                  */
            if ( ((c[n+1]-c[n])>Tmin)&&((c[n]-c[n-1])>Tmin))
                        {
                        t1 = c[n];
                        t2 = c[n]+Tr;
                        fprintf (stream," + %.8f %d \n ", t1, P1);
                        fprintf (stream," + %.8f %d \n ", t2, P2);
/*                      printf(" t2 = %.8e \n",t2);
                        getchar();                      */
                        }
            n++;
/*          printf("value of n %d \n",n);           */
            if ( ((c[n]-c[n-1])>Tmin)&&((c[n+1]-c[n])>Tmin))
                        {
                        t1 = c[n];
                        t2 = c[n]+Tr;
                        fprintf (stream," + %.8f %d \n", t1, P2);
                        fprintf (stream," + %.8f %d \n", t2, P1);
                        }
             }
fprintf (stream," )");
fclose (stream);
return;
}
```

APPENDIX C : MATLAB/SIMULINK M-FILES

C.1 M-FILES FOR VSI MODEL

```
function [sys,x0]=activfil(t,x,u,flag)

%*************************************************************%
%                  using state space approach to model        %
%                        3 phase inverter                     %
%                     filename : activfi3.m                   %
%*************************************************************%

if nargin==0
        sys=[3,0,4,6,0,1];
end

% Define variables
R1 = 1.1;
L = 6.45E-3;
R2 = 1.1;
C = 1100E-6;
R3 = 1.1;

% u(4),u(5),u(6) are Sa, Sb, Sc respectively.
%  1 - upper switch is on
% -1 - lower switch is on

if abs(flag)==1
  if u(4)==1
        if u(5)==1
                if u(6)==1
                        A=[((-2*R2)-R1)/(3*L)   (-R1+R3)/(3*L)      0;
                          (R2-R1)/(3*L)  (-(2*R3)-R1)/(3*L)  0;
                          0              0                   0];

                        B=[-1/(3*L)   2/(3*L)  -1/(3*L)   0  0  0;
                          -1/(3*L)  -1/(3*L)   2/(3*L)   0  0  0;
                           0        0         0         0  0  0];

                else
                        A=[((-2*R2)-R1)/(3*L)   (R3-R1)/(3*L)      -1/(3*L);
                          (-R1+R2)/(3*L)((-2*R3)-R1)/(3*L)   2/(3*L);
                          0                -1/C                0     ];

                        B=[-1/(3*L)  2/(3*L)   -1/(3*L)  0  0  0;
                          -1/(3*L) -1/(3*L)    2/(3*L)  0  0  0;
                           0       0          0        0  0  0 ];

        end

elseif u(5)==-1
        if u(6)==1
                A=[((-2*R2)-R1)/(3*L)   (-R1+R3)/(3*L)      2/(3*L);
                  (-R1+R2)/(3*L)((-2*R3)-R1)/(3*L)  -1/(3*L);
                   -1/C                0                   0     ];

                B=[-1/(3*L)   2/(3*L)  -1/(3*L)  0  0  0;
```

```
                    -1/(3*L)  -1/(3*L)   2/(3*L)  0  0  0;
                     0        0         0        0  0  0 ];

            else
                    A=[((-2*R2)-R1)/(3*L)   (R3-R1)/(3*L)       1/(3*L);
                      (R2-R1)/(3*L)              ((-2*R3)-R1)/(3*L)  1/(3*L);
                      -1/C              -1/C               0      ];

                    B=[-1/(3*L)   2/(3*L)   -1/(3*L)  0  0  0;
                      -1/(3*L)  -1/(3*L)    2/(3*L)  0  0  0;
                      0         0         0        0  0  0];

            end
        end
end

if u(4)==-1
        if u(5)==1
            if u(6)==1
                    A=[((-2*R2)-R1)/(3*L)   (-R1+R3)/(3*L)      -1/(3*L);
                      (R2-R1)/(3*L)  ((-2*R3)-R1)/(3*L)   -1/(3*L);
                      1/C                1/C                0      ];

                    B=[-1/(3*L)   2/(3*L)  -1/(3*L)  0  0  0;
                      -1/(3*L)  -1/(3*L)   2/(3*L)  0  0  0;
                      0         0         0        0  0  0];


            else
                    A=[((-2*R2)-R1)/(3*L)   (R3-R1)/(3*L)       -2/(3*L);
                      (-R1+R2)/(3*L)             ((-2*R3)-R1)/(3*L)   1/(3*L);
                      1/C                0                  0      ];

                    B=[-1/(3*L)   2/(3*L)   -1/(3*L)  0  0  0;
                      -1/(3*L)  -1/(3*L)    2/(3*L)  0  0  0;
                      0         0         0        0  0  0];

 end
elseif u(5)==-1
        if u(6)==1
                    A=[((-2*R2)-R1)/(3*L)   (-R1+R3)/(3*L)      1/(3*L);
                      (R2-R1)/(3*L)  ((-2*R3)-R1)/(3*L)  -2/(3*L);
                      0                  1/C                0      ];

                    B=[-1/(3*L)    2/(3*L)  -1/(3*L)  0  0  0;
                      -1/(3*L)   -1/(3*L)   2/(3*L)  0  0  0;
                      0         0         0        0  0  0];


            else
                    A=[((-2*R2)-R1)/(3*L)   (R3-R1)/(3*L)       0;
                      (R2-R1)/(3*L)  ((-2*R3)-R1)/(3*L)  0;
                      0                  0                  0 ];

                    B=[-1/(3*L)   2/(3*L)   -1/(3*L)  0  0  0;
                      -1/(3*L)  -1/(3*L)    2/(3*L)  0  0  0;
                      0         0         0        0  0  0];


            end
        end
end
```

```
        sys=A*x + B*u;

elseif flag==3

  C=[-1  -1  0;
     1   0  0;
     0   1  0;
     0   0  1];

  sys=C*x;

elseif flag==0
        sys=[3,0,4,6,0,1];
else
        sys=[];
end
```

C.2 M-FILES FOR DEADBAND PWM

```
function[sys,x0]=dband1(t,x,u,flag)

%*************************************************************%
%                  program to implement deadband pwm          %
%                          dband1.m                           %
%*************************************************************%

if flag == 3

% Return system outputs:
% u(1) - wt
% u(2) - modulation depth
% u(3) - phase angle control
% u(4) - triangular carrier waveform

    if (u(3)>0)
        theta=(-pi/2)+u(3);
    else
         theta=(pi/2)+u(3);
    end

        d120=2*pi/3;

        V1=240*sin(u(1));
        V2=240*sin(u(1)-d120);
        V3=240*sin(u(1)+d120);

        Ia=sin(u(1)+u(3)+theta);
        Ib=sin(u(1)+u(3)+theta-d120);
        Ic=sin(u(1)+u(3)+theta+d120);

        if t<0.1
          m=u(2);
        else
          m=0.7;
        end

        Va=m*sin(u(1)+u(3));
        Vb=m*sin(u(1)+u(3)-d120);
```

```
Vc=m*sin(u(1)+u(3)+d120);

if (Va>Vb) & (Vb>Vc)
        if abs(Ia)>abs(Ic)
                Vac=1;
                Vbc=1-(Va-Vb);
                Vcc=1-(Va-Vc);
        else
                Vcc=-1;
                Vac=-1+(Va-Vc);
                Vbc=-1+(Vb-Vc);
        end

elseif (Va>Vc) & (Vc>Vb)
        if abs(Ia)>abs(Ib)
                Vac=1;
                Vbc=1-(Va-Vb);
                Vcc=1-(Va-Vc);
        else
                Vbc=-1;
                Vac=-1+(Va-Vb);
                Vcc=-1+(Vc-Vb);
        end

elseif (Vb>Va) & (Va>Vc)
        if abs(Ib)>abs(Ic)
                Vbc=1;
                Vac=1-(Vb-Va);
                Vcc=1-(Vb-Vc);
        else
                Vcc=-1;
                Vac=-1+(Va-Vc);
                Vbc=-1+(Vb-Vc);
        end

elseif (Vb>Vc) & (Vc>Va)
        if abs(Ib)>abs(Ia)
                Vbc=1;
                Vac=1-(Vb-Va);
                Vcc=1-(Vb-Vc);
        else
                Vac=-1;
                Vbc=-1+(Vb-Va);
                Vcc=-1+(Vc-Va);
        end

elseif (Vc>Va) & (Va>Vb)
        if abs(Ic)>abs(Ib)
                Vcc=1;
                Vac=1-(Vc-Va);
                Vbc=1-(Vc-Vb);
        else
                Vbc=-1;
                Vac=-1+(Va-Vb);
                Vcc=-1+(Vc-Vb);
        end

else
        if abs(Ic)>abs(Ia)
                Vcc=1;
                Vac=1-(Vc-Va);
```

```
                Vbc=1-(Vc-Vb);
        else
                Vac=-1;
                Vcc=-1+(Vc-Va);
                Vbc=-1+(Vb-Va);
        end
end


 if Vac>u(4)
   S1=1;
 else
   S1=-1;
end

        if Vbc>u(4)
          S2=1;
        else
          S2=-1;
        end

        if Vcc>u(4)
          S3=1;
        else
          S3=-1;
        end

        sys=[V1,V2,V3,S1,S2,S3];
elseif flag==0
        sys=[0,0,6,4,0,1];

end
end
```

C.3 M-FILES FOR P AND Q DETECTION

```
function [sys,x0]=detect(t,x,u,flag)
% file name = detect1.m
% M-file to transform,detect and compensate reactive power
% Inputs are 3 phase voltage and current waveforms
% Output is calculated P and Q

if flag ==3
  % return system output

T = [1    -1/2        -1/2;
    0  sqrt(3)/2  -sqrt(3)/2];
V = [u(4);u(5);u(6)];
e = sqrt(2/3)*T*V;
e1=e(1);
e2=e(2);

A = [u(1);u(2);u(3)];
i = sqrt(2/3)*T*A;
i1=i(1);
i2=i(2);
```

```
  power=[e1 e2;-e2 e1];
  power=power*i;
  q = power(2);
  p = power(1);

  sys=[p,q,u(1),u(2),u(3),u(4),u(5),u(6)];

elseif flag==0
  sys=[0,0,8,6,0,1];
end
```

```
**************************************************************************
function [sys,x0]=detect(t,x,u,flag)
% file name = detect23.m
% M-file to transform,detect and compensate reactive power
% Inputs are 3 phase current and voltage waveforms
% and filtered p and q
% Output is compensated source current

if flag ==3
  % return system output

  T = [1    -1/2       -1/2;
       0  sqrt(3)/2  -sqrt(3)/2];
  V = [u(6);u(7);u(8)];
  e = sqrt(2/3)*T*V;
  e1=e(1);
  e2=e(2);

  A = [u(3);u(4);u(5)];
  i = sqrt(2/3)*T*A;
  i1=i(1);
  i2=i(2);

  p = u(1);
  q = u(2);
  T = T';
  T1= [ e1   e2;
       -e2  e1];
  T1= inv(T1);

  icomp=sqrt(2/3)*T*T1*[p;q];
  is= A-icomp;

  sys= [icomp(1),icomp(2),icomp(3)];

elseif flag==0
  sys=[0,0,3,8,0,1];
end
```

C.4 M-FILES FOR APF SLIDING MODE CONTROL

```
function[sys,x0]=slide(t,x,u,flag)

%*************************************************************%
%          program to implement sliding mode switching control          %
```

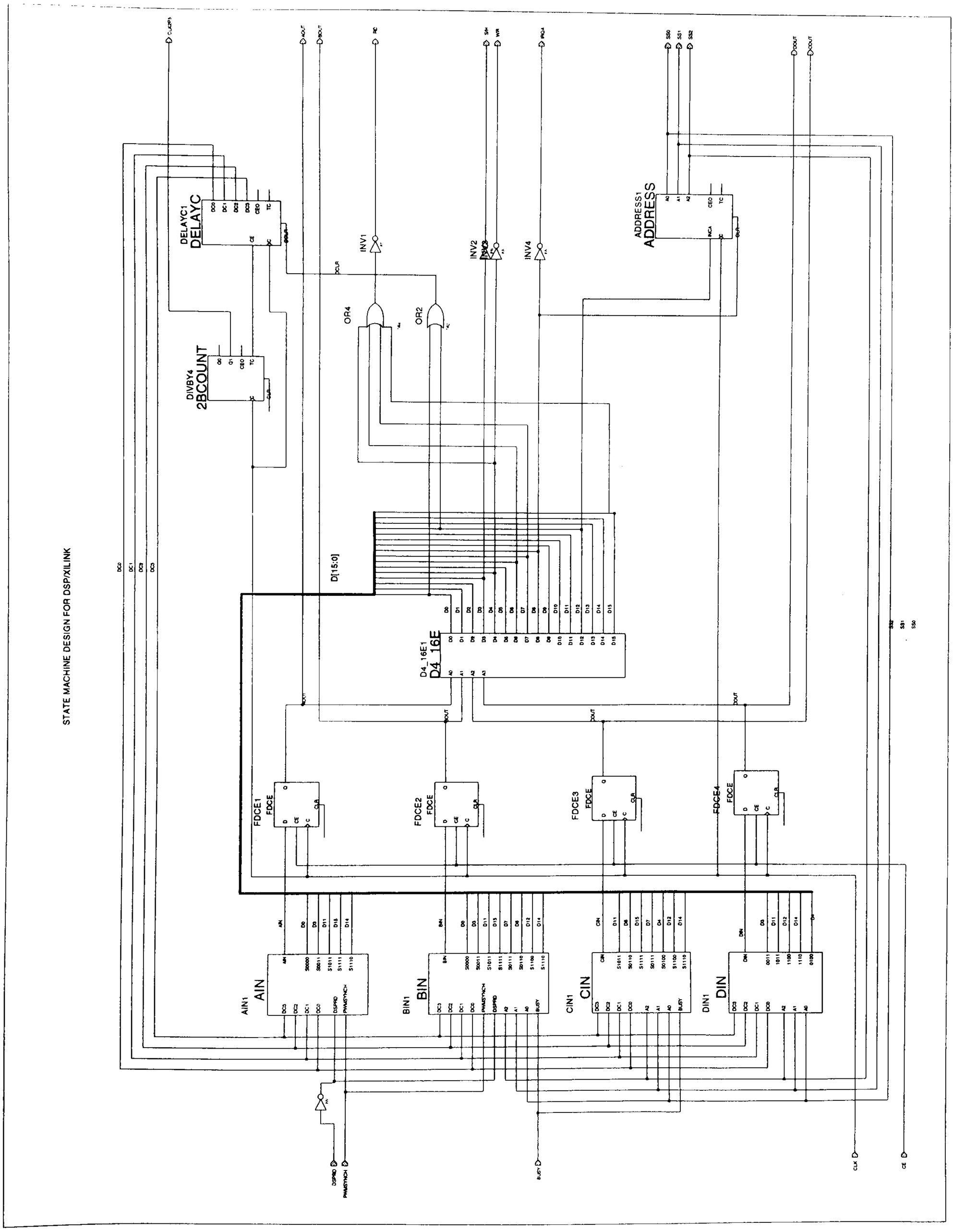
```
%                          slide1.m                                    %
%                  determine switching from info                       %
%                  Ud, Uq, Ed, Eq and angle(deg)                       %
%*******************************************************************%

if flag == 3

% find angle
        theta=u(20);
    angle=(sin(theta));
        angle1=(cos(theta));

if angle>0
  if angle1>0
    deg=asin(angle);
    deg=deg*180/pi;
  else
    deg=acos(angle1);
    deg=(deg*180/pi);
  end
else
  if angle1>0
    deg=acos(angle1);
    deg=360-(deg*180/pi);
  else
    deg=asin(angle);
    deg=180-(deg*180/pi);
  end
end

deg1=atan(abs(u(22)/u(21)))*180/pi;
if u(21)>0
  if u(22)>0
    deg2=deg1;
  else
    deg2=360-deg1;
  end
else
  if u(22)>0
    deg2=180-deg1;
  else
    deg2=180+deg1;
  end
end

extra=0;
% find true displacement angle from Ud, Uq and wt
deg=deg+deg1;
if deg>=360
  deg=deg-360;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Return system outputs
% First, find component values:

   if u(7)>0.9 | u(7)<-0.9

        T = [1    -1/2       -1/2;
       0  sqrt(3)/2  -sqrt(3)/2];
```

```
T1= [ cos(theta)  sin(theta);
-sin(theta)  cos(theta)];
Iinject = [u(4);u(5);u(6)];
Iref    = sqrt(2/3)*T1*T*Iinject;
Iactual = [u(8);u(9);u(10)];

Iact    = sqrt(2/3)*T1*T*Iactual;
E       = Iref - Iact;
Ed      = -E(1);
Eq      = -E(2);

if Ed>0
  if Eq>0
        if (deg>(30+extra) & deg<(90+extra))
          S1=1; S2=1; S3=-1;
        elseif (deg>(90+extra) & deg<(150+extra))
          S1=-1; S2=1; S3=-1;
        elseif (deg>(150+extra) & deg<(210+extra))
           S1=-1; S2=1; S3=1;
        elseif (deg>(210+extra) & deg<(270+extra))
           S1=-1; S2=-1; S3=1;
        elseif (deg>(270+extra) & deg<(330+extra))
           S1=1; S2=-1; S3=1;
        else
           S1=1; S2=-1; S3=-1;
   end
else
        if (deg>(30+extra) & deg<(90+extra))
         S1=1; S2=-1; S3=-1;
        elseif (deg>(90+extra) & deg<(150+extra))
          S1=1; S2=1; S3=-1;
         elseif (deg>(150+extra) & deg<(210+extra))
          S1=-1; S2=1; S3=-1;
         elseif (deg>(210+extra) & deg<(270+extra))
          S1=-1; S2=1; S3=1;
        elseif (deg>(270+extra) & deg<(330+extra))
          S1=-1; S2=-1; S3=1;
        else
          S1=1; S2=-1; S3=1;
        end
  end
else
  if Eq>0
        if (deg>(30+extra) & deg<(90+extra))
          S1=-1; S2=1; S3=-1;
        elseif (deg>(90+extra) & deg<(150+extra))
          S1=-1; S2=1; S3=1;
        elseif (deg>(150+extra) & deg<(210+extra))
          S1=-1; S2=-1; S3=1;
        elseif (deg>(210+extra) & deg<(270+extra))
          S1=1; S2=-1; S3=1;
        elseif (deg>(270+extra) & deg<(330+extra))
           S1=1; S2=-1; S3=-1;
        else
           S1=1; S2=1; S3=-1;
        end
else
        if (deg>(30+extra) & deg<(90+extra))
          S1=-1; S2=-1; S3=1;
        elseif (deg>(90+extra) & deg<(150+extra))
          S1=1; S2=-1; S3=1;
```

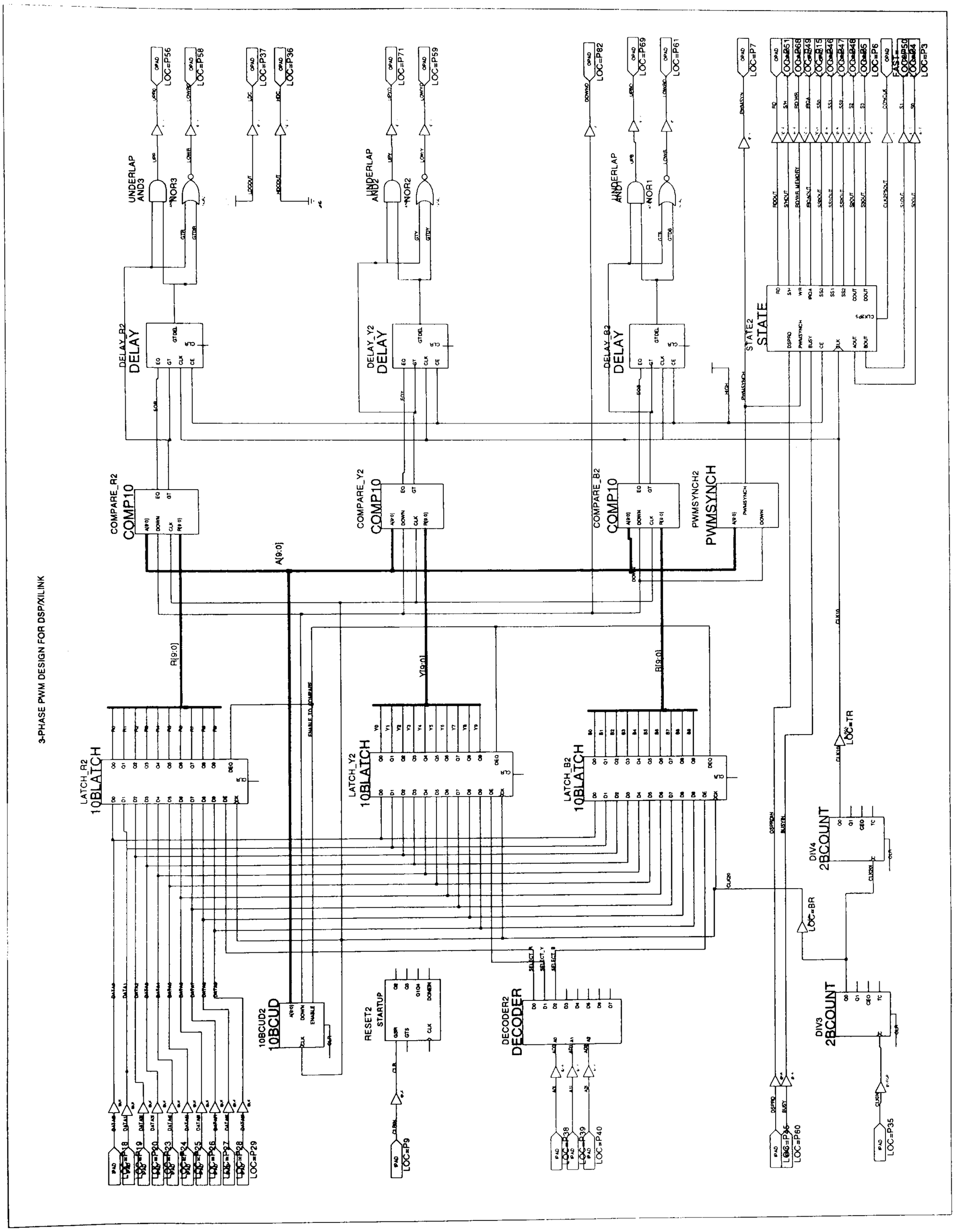
```
            elseif (deg>(150+extra) & deg<(210+extra))
              S1=1; S2=-1; S3=-1;
            elseif (deg>(210+extra) & deg<(270+extra))
              S1=1; S2=1; S3=-1;
            elseif (deg>(270+extra) & deg<(330+extra))
              S1=-1; S2=1; S3=-1;
            else
              S1=-1; S2=1; S3=1;
            end
          end
  end

else
        S1=u(14);
        S2=u(15);
        S3=u(16);
        Ed=u(17);
        Eq=u(18);
        deg=u(19);
    end

        sys=[u(1),u(2),u(3),S1,S2,S3,Ed,Eq,deg2];
elseif flag==0
        sys=[0,0,9,22,0,1];

end
```

APPENDIX D : SCHEMATIC DIAGRAMS FOR XILINX DESIGN

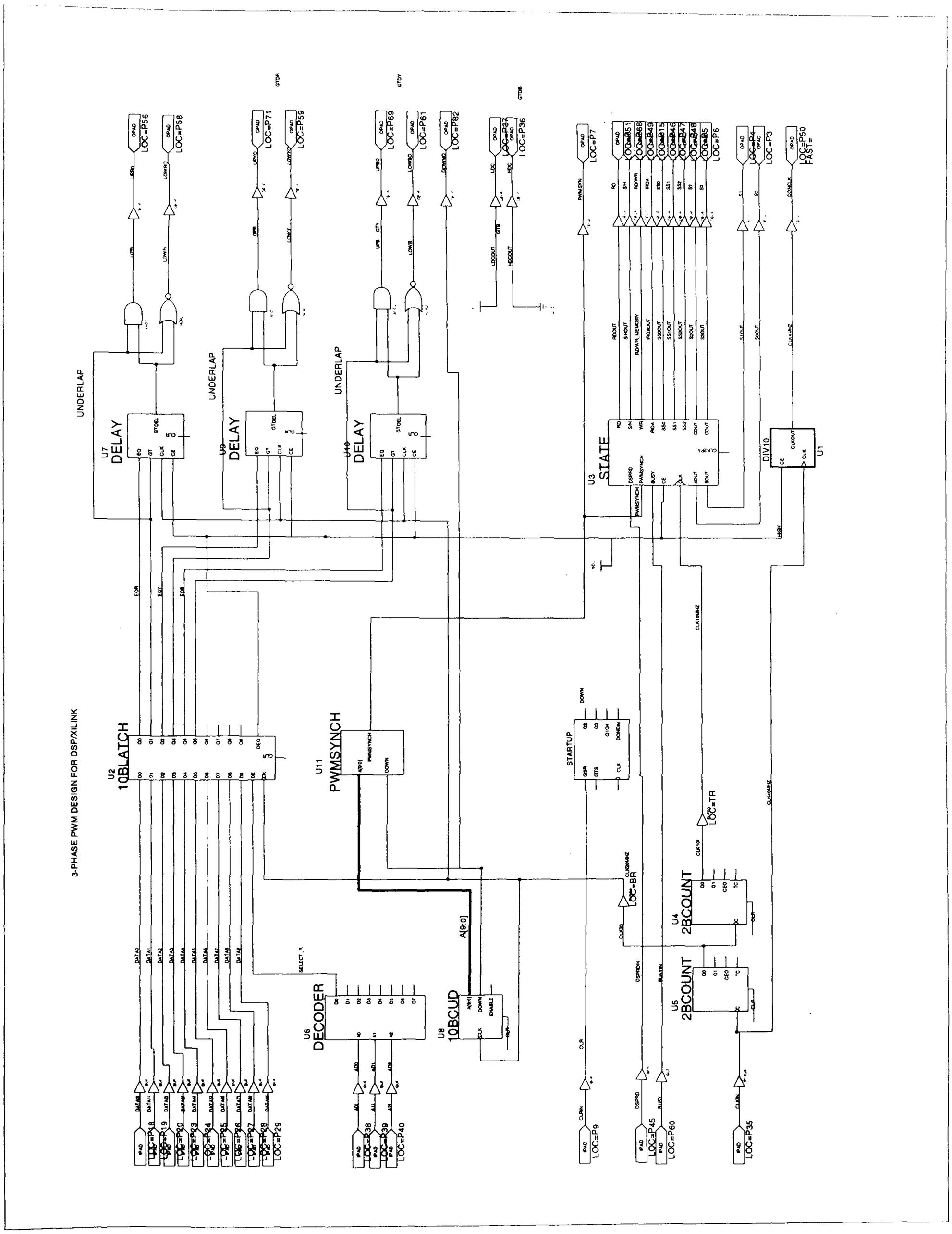
D.1 STATE MACHINE FOR DSP/XILINX CONTROL



D.2 XILINX FPGA DESIGN FOR SVC



D.3 XILINX FPGA DESIGN FOR APF



APPENDIX E : DSP PROGRAM LISTINGS

E.1 MEMORY MAPPING AND DATA STORAGE

```
**********************************************
*       Memory Mapping and Sytem Variables   *
*            for MOTOROLA DSP 96002          *
**********************************************
```

P program memory

0-3FF	internal program RAM
400-1FF	not mapped
2000-3FFF	external RAM on port A
4000-FFFFFFFE	external byte-wide eprom on port A

X data memory

0-1FF	internal data RAM used for system variables
200-3FF	non-existent internal memory
400-7FF	internal cosine ROM (full cycle)
800-1FFF	not mapped
2000-3FFF	external RAM on port B (shared with Y memory)
FFFFFF80-FFFFFFFF	not mapped

Y data memory

0-1FF	internal data RAM used for stack (R7)
200-3FF	non-existent internal memory
400-7FF	internal sine ROM (full cycle)
800-1FFF	not mapped
2000-3FFF	external RAM on port B (shared with X memory)
4000-FFFFFF7F	not mapped
FFFFFF80-FFFFFFFF	external i/o

```
**********************************************
*                storage locations           *
**********************************************
```

x:$10	Vr	x:$13	Irload	x:$51	Vrfiltered
x:$11	Vy	x:$14	Iyload	x:$52	Vyfiltered
x:$12	Vb	x:$15	Ibload	x:$53	Vbfiltered
x:$16	Ealfa	x:$1A	P	x:$1D	Aalfa
x:$17	Ebeta	x:$1B	Q	x:$1E	Balfa
x:$18	Ialfa	x:$1C	P-Pfiltered	x:$54	Irfiltered
x:$19	Ibeta				

x:$20..x:$38	coeff. for 8th/10th order IIR filter
y:$2000..200e	location for d(i,n-2),d(i,n-1) .. etc
x:$3EFF..3FFE	location for arctan lookup table
x:$3FFF	true value of angle (Uq/Ud)

x:$40	Iacomp	x:$43	Iainject	x:$46	Vdc
x:$41	Ibcomp	x:$44	Ibinjetc	x:$47	Vinmax (A)
x:$42	Iccomp	x:$45	Icinject	x:$48	Vinmax (B)- peak value

(previous sw)		(current sw)			
x:$55	switch1	x:$58	temp_sw1	x:$61	Varesistor
x:$56	switch2	x:$59	temp_sw2		
x:$57	switch3	x:$60	temp_sw3		

```
x:$63   Isupply1                x:$66   swdac1          x:$69   sw1_eq
x:$64   Isupply2                x:$67   swdac2          x:$6A   sw2_eq
x:$65   Isupply3                x:$68   swdac3          x:$6B   sw3_eq

x:$6C   pwm_only
x:$6D   pwm_eq

x:$70   Iacomp-Iainject         x:$73   Iaload-Iainject
x:$71   Ibcomp-Ibinject         x:$74   Ibload-Ibinject
x:$72   Iccomp-Icinject         x:$75   Icload-Icinject

**************** the following locations for sliding mode control **********************

x:$77   prev value of Vr        x:$7A   current sine
x:$78   counter                 x:$7B   pointer for sine/cosine table
x:$79   max value of counter

x:$80   Iref - alpha            x:$84   Iinj - alpha        x:$88   Ed
x:$81   Iref - beta             x:$85   Iinj - beta         x:$89   Eq
x:$82   Iref_d                  x:$86   Iinj_d
x:$83   Iref_q                  x:$87   Iinj_q

x:$8A   d_Irefd                 x:$8C   previous Irefd      x:$93   previoud d_Irefd
x:$8B   d_Irefq                 x:$8D   previous Irefq      x:$94   previous d_Irefq

x:$8E   Ud                      x:$90   Uq/Ud
x:$8F   Uq                      x:$91   angle (arctan Uq/Ud)
                                x:$92   current pointer for true angle
```

E.2 SVC CONTROL - SINUSOIDAL / DEADBAND PWM

```
;*************************************************************************
;*      Assembly language program for DSP96002 to:                       *
;*              1. set up the initialization routine                     *
;*              2. transfer data from memory to dsp                      *
;*              3. generate PWM reference waveform                       *
;*************************************************************************

ipr             equ     $FFFFFFFF       ; address of Interrupt Priority Register
psr             equ     $FFFFFFFC       ; address of Port Select Register
bcra            equ     $FFFFFFFE       ; address of Port A Bus Control Register
bcrb            equ     $FFFFFFFD       ; address of Port B Bus Control Register

inport1         equ     $FFFFFF90       ; 1st input port
inport2         equ     $FFFFFF91       ; 2nd input port
outport1        equ     $FFFFFF92       ; 1st ouput port
outport2        equ     $FFFFFF93       ; 2nd output port

adcmask         equ     $FFF
ad2047          equ     $7ff            ;(825)2047+41 (due to filter offset)
ip_data         equ     $10             ;for storage
ph_angle        equ     8.0
mod_index       equ     0.7

out_clr         equ     $1c00           ;
```

```
out_r           equ     $0000                   ;decoder signal
out_y           equ     $0400                   ;for Xilinx
out_b           equ     $0800                   ;

store_r         equ     $17                     ;reference for 1st phase
store_y         equ     $18                     ;reference for 2nd phase
store_b         equ     $19                     ;reference for 3rd phase


;****************************** main ************************************

        org p:$0                                ; jump from reset to start
        jmp $1FF                                ; of initialization
        nop

        org p:$8                                ; IRQA
        jsr $2000                               ; jump to service routine
        nop

        org x:$0
setup_regs  ds    6


        org x:setup_regs
        dc      $18100                          ; PSR
        dc      $200A0000                       ; BCRA
        dc      $200A0000                       ; BCRB
        dc      $66                             ; IPR
        dc      $00000000                       ; SR (interrupt level priority)

        org p:$1FF                              ; starting address for initialisation


        move #ph_angle,d5.s                     ;phase angle for leading/lagging VAr
        move #360.0,d2.s
        fseedd d2,d2                            ;approx of 1/360 (8 bits accuracy)
        fmpy.s d5,d2,d5  #1024.0,d2.s           ;angle/360
        fmpy.s d5,d2,d5                         ;angle/360 * 1024 convert degree to step

        move #$500,d0.l                         ;set the ref. at cos 90
        float.s d0                              ;convert to floating point
        fadd.s d5,d0                            ;add with delay
        move d0.s,d5.s
        int d5
        move d5.l,x:$1fe                        ;ref in integer (for r)
        move d0.s,x:$1ff                        ;ref in floating pt (for d)

        move #0,d3.l
        add d3,d5                               ;
        move d5.l,x:$1fc                        ; another reference
        move #0.0,d3.s                          ;  pointer for sine/cosine
        fadd.s d3,d0                            ; lookup table
        move d0.s,x:$1fd

        move x:$1fe,r2                          ;ref in integer (for r)
        move x:$1ff,d0.s                        ;ref in floating pt (for d)
        move x:$1fc,r3                          ;ref in integer (for r)
        move x:$1fd,d5.s                        ;ref in floating pt (for d)

        move #$2000,r4                          ;
        move #$2800,r5                          ;storage for Vref
        move #$3600,r6                          ;
```

```
        move #setup_regs,r0
        move #$8,omr                    ; set operating mode register
        movep x:(r0)+,x:psr             ;
        movep x:(r0)+,x:bcrb            ; set up  relevant
        movep x:(r0)+,x:bcra            ; control registers
        movep x:(r0)+,x:ipr             ; IRQA ipl=2 -ve edge
        move x:(r0),sr                  ;
        move #ip_data,r1                ;starting add. for storage
        movep #0,y:outport2             ; clear o/p port 2
        movep #$1000,y:outport2         ; set DSPRD enabling line conversion


loop
        jmp loop

;  ************    interrupt service routine for IRQA      ***************

        org p:$2000
        clr d1.l
        lsl #9,d1.l
        move d1.l,y:outport2            ;ready to transfer data
        move y:inport1,d2.l             ;
        move #adcmask,d3.l              ;mask wanted bits only (12 lsb)
        and d3.l,d2.l
        move #ad2047,d3.l               ;shift down for bipolar
        sub d3,d2                       ;operation
        float.s d2
        move d2.s,x:(r1)                ;store data of V from transducer

        fbmi minus                      ;jump if -ve to minus
        move x:(r2),d4.s                ;store ref. from cos ROM
        fneg.s d4
        move #mod_index,d6.s            ;multiply by
        fmpy.s d6,d4,d4                 ;modulation index
        move d4.s,x:$28
         move #ph_angle,d3.s            ;determine the required
         move #1.0,d4.s                 ;phase angle control
         fmpy.s d3,d4,d3
         fbmi lead_R1
lag_R1   move y:(r2),d4.s               ; 1st phase lagging VAr
         fneg.s d4                      ; compensation
         move d4.s,x:$32
         jmp next_1
lead_R1  move y:(r2),d4.s               ; 1st phase leading VAr
         move d4.s,x:$32                ; compensation


next_1
        move #171.0,d4.s                ;yellow -120
        fadd.s d0,d4                    ;correct position at cos ROM
        int d4                          ;change to integer
        move d4.l,r0                    ;sin -120 = cos 150
        nop                             ;prevent pipelining error
        move x:(r0),d4.s                ;retrieve cos 150
        move #mod_index,d6.s            ;multiply by
        fmpy.s d6,d4,d4                 ;modulation index
        move d4.s,x:$29
        move #ph_angle,d3.s             ;determine the required
        move #1.0,d4.s                  ;phase angle control
```

```
            fmpy.s d3,d4,d3
            fbmi lead_Y1
lag_Y1      move y:(r0),d4.s                ; 2nd phase lagging VAr
            move d4.s,x:$33                 ; compensation
            jmp next_2
lead_Y1     move y:(r0),d4.s                ; 2nd phase leading VAr
            fneg.s d4                       ; compensation
            move d4.s,x:$33


next_2
            move #85.0,d4.s                         ;blue +120
            fadd.s d0,d4                    ;sin 120
            int d4
            move d4.l,r0
            nop                             ;prevent pipelining error
            move y:(r0),d4.s                ;retrieve sin 120
            move #mod_index,d6.s            ;multiply by
            fmpy.s d6,d4,d4                 ;modulation index
            move d4.s,x:$30
            move #ph_angle,d3.s             ;determine the required
            move #1.0,d4.s                  ;phase angle control
            fmpy.s d3,d4,d3
            fbmi lead_B1
lag_B1      move x:(r0),d4.s                ; 2nd phase lagging VAr
            fneg.s d4                       ; compensation
            move d4.s,x:$34
            jmp next_3
lead_B1     move x:(r0),d4.s                ; 2nd phase leading VAr
            move d4.s,x:$34                 ; compensation

next_3
move #5.23266,d3.s                          ;orig 5.24288 (5.23266)
            fadd.s d3,d0                    ;
            move d0.s,d3.s                  ;point to next location
            int d3                          ;
            move d3.l,r2                    ;

            move x:$1fc,r3                  ;
            move x:$1fd,d5.s                ;reset the 'minus' condition
            move r2,d3.l
            jmp meet_1

minus       move x:(r3),d4.s                ;store ref. for Vr
            move #mod_index,d6.s            ;multiply by
            fmpy.s d6,d4,d4                 ;modulation index
            move d4.s,x:$28
            move #ph_angle,d3.s
            move #1.0,d4.s
            fmpy.s d3,d4,d3
            fbmi lead_R2
lag_R2      move y:(r3),d4.s
            move d4.s,x:$32
            jmp next_4
lead_R2     move y:(r3),d4.s
            fneg.s d4
            move d4.s,x:$32

next_4
            move #171.0,d4.s                ;same as above but negate
            fadd.s d5,d4
```

```
                int d4
                move d4.l,r0
                nop                             ;prevent pipelining error
                move x:(r0),d4.s
                fneg.s d4
                move #mod_index,d6.s            ;multiply by
                fmpy.s d6,d4,d4                 ;modulation index
                move d4.s,x:$29
                move #ph_angle,d3.s
                move #1.0,d4.s
                fmpy.s d3,d4,d3
                fbmi lead_Y2
lag_Y2          move y:(r0),d4.s
                fneg.s d4
                move d4.s,x:$33
                jmp next_5
lead_Y2         move y:(r0),d4.s
                move d4.s,x:$33

next_5
                move #85.0,d4.s                 ;
                fadd.s d5,d4                    ;same as above but negate
                int d4
                move d4.l,r0                    ;
                nop                             ;prevent pipelining error
                move y:(r0),d4.s                ;take ref from sin. rom
                fneg.s d4                       ;negate value
                move #mod_index,d6.s            ;multiply by
                fmpy.s d6,d4,d4                 ;modulation index
                move d4.s,x:$30
                move #ph_angle,d3.s
                move #1.0,d4.s
                fmpy.s d3,d4,d3
                fbmi lead_B2
lag_B2          move x:(r0),d4.s
                move d4.s,x:$34
                jmp next_6
lead_B2         move x:(r0),d4.s
                fneg.s d4
                move d4.s,x:$34

next_6
                move #5.23266,d3.s              ;orig 5.24288 (5.23266)
                fadd.s d3,d5                    ;
                move d5.s,d3.s                  ;point to next location
                int d3                          ;
                move d3.l,r3                    ;
                move x:$1fe,r2                  ;
                move x:$1ff,d0.s                ;reset  the 'plus' condition
                move r3,d3.l

meet_1

;************************** deadband PWM ************************
; The position that points to sin. look-up table is stored in
; R0. R1, R2  so that this can be easily incremented.
; Remember that ref. sin must be stored in d2,d3 and d4
; The results will be stored in x:$21 - x:$26
;   a-b  a-c  b-a  b-c  c-a  c-b

                move #1.0,d3.s
```

```
move x:$28,d4.s
move x:$29,d6.s
fsub.s d6,d4                    ;Va-Vb store in d4
fsub.s d4,d3                    ;1-(Va-Vb) store in d3
move d3.s,x:$21                 ;store result
move #1.0,d3.s
move x:$28,d4.s
move x:$30,d6.s
fsub.s d6,d4                    ;Va-Vc store in d4
fsub.s d4,d3                    ;1-(Va-Vc) store in d3
move d3.s,x:$22                 ;store result in x:$22

move #1.0,d3.s                  ;1
move x:$28,d4.s
move x:$29,d6.s
fsub.s d4,d6                    ;Vb-Va store in d6
fsub.s d6,d3                    ;1-(Vb-Va) store in d3
move d3.s,x:$23                 ;store in x:$23

move #1.0,d3.s
move x:$29,d4.s
move x:$30,d6.s
fsub.s d6,d4                    ;Vb-Vc
fsub.s d4,d3                    ;1-(Vb-Vc)
move d3.s,x:$24                 ;store in x:$24

move #1.0,d3.s
move x:$28,d4.s
move x:$30,d6.s
fsub.s d4,d6                    ;Vc-Va
fsub.s d6,d3                    ;1-(Vc-Va)
move d3.s,x:$25                 ;store in x:$25
move #1.0,d3.s
move x:$29,d4.s
move x:$30,d6.s
fsub.s d4,d6                    ;(Vc-Vb)
fsub.s d6,d3                    ;1-(Vc-Vb)
move d3.s,x:$26                 ;store in x:$26


; Above Va etc is from the sin lookup table
; The measured V's and I's must be stored at
; #512-d0  Va-d1   Vb-d2   Vc-d3  Ia-d4  Ib-d5  Ic-d6
; the pwm o/p at Vr=x:$17  Vy=x:$18  Vb=x:$19
; remember that this time the result must be scaled i.e
;      -1 : 0     0 : 511     1 : 1023
;   using the eq. : (x * 511) + 511

start
        move x:$28,d3.s
        move x:$29,d4.s
        move x:$30,d6.s
        fcmp d4,d3              ;Va-vb
        fblt loop_1             ;if Va<Vb jmp to loop_1
        fcmp d6,d4              ;Vb-Vc
        fblt loop_11            ;if Vb<Vc jmp to loop_11
        move x:$32,d3.s
        move x:$34,d4.s
        fcmpm d4,d3             ;Ia-Ic compare magnitude
        fblt loop_111           ;if |Ia|<|Ic| jmp to loop_111
```

```
                move #1023,x:$17            ;Vr=1
                move x:$21,d4.s             ;1-(Va-Vb)
                jsr pwm_10bit
                move d4.l,x:$18             ;Vy
                move x:$22,d4.s             ;1-(Va-Vc)
                jsr pwm_10bit
                move d4.l,x:$19             ;Vb
                jmp end_1

loop_111
                move #0,x:$19               ;Vb = -1
                move x:$22,d4.s             ;1-(Va-Vc)
                fneg.s d4                   ;-1+(Va-Vc)
                jsr pwm_10bit
                move d4.l,x:$17             ;Vr
                move x:$24,d4.s             ;1-(Vb-Vc)
                fneg.s d4                   ;-1+(Vb-Vc)
                jsr pwm_10bit
                move d4.l,x:$18             ;store Vy
                jmp end_1

loop_11
                move x:$28,d3.s             ;Va
                move x:$30,d4.s             ;Vc
                move x:$33,d6.s             ;Ib
                move x:$34,d7.s             ;Ic
                fcmp d3,d4                  ;Vc-Va
                fblt loop_1                 ;if Vc<Va jmp to loop_1
                fcmpm d6,d7                 ;|Ic| - |Ib|
                fblt loop_112               ;if |Ic|<|Ib| jmp to loop_112

                move #1023,x:$19            ;Vblue = 1
                move x:$25,d4.s             ;1-(Vc-Va)
                jsr pwm_10bit
                move d4.l,x:$17             ;Vr
                move x:$26,d4.s             ;1-(Vc-Vb)
                jsr pwm_10bit
                move d4.l,x:$18             ;Vyellow
                jmp end_1

loop_112
                move #0,x:$18               ;Vy = -1
                move x:$21,d4.s             ;1-(Va-Vb)
                fneg.s d4                   ;-1+(Va-Vb)
                jsr pwm_10bit
                move d4.l,x:$17             ;Vr
                move x:$26,d4.s             ;1-(Vc-Vb)
                fneg.s d4                    ;-1+(Vc-Vb)
                jsr pwm_10bit
                move d4.l,x:$19             ;Vblue
                jmp end_1

loop_1
                 move x:$28,d3.s            ;Va
                move x:$29,d4.s             ;Vb
                move x:$30,d6.s             ;Vc
                move x:$34,d7.s             ;Ic
                fcmp d3,d4                  ;Vb-Va
                fblt loop_2                 ;if Vb<Va jmp to loop_2
                fcmp d6,d3                  ;Va-Vc
                fblt loop_22                ;if Va<Vc jmp to loop_22
```

```
        move x:$33,d6.s                 ;Ib
        fcmpm d7,d6                     ;|Ib| - |Ic|
        fblt loop_221                   ;if |Ib|<|Ic| jmp to loop_221

        move #1023,x:$18                ;Vy=1
        move x:$23,d4.s                 ;1-(Vb-Va)
        jsr pwm_10bit
        move d4.l,x:$17                 ;Vr
        move x:$24,d4.s                 ;1-(Vb-Vc)
        jsr pwm_10bit
        move d4.l,x:$19                 ;Vblue
        jmp end_1

loop_221
        move #0,x:$19                   ;Vblue=-1
        move x:$22,d4.s                 ;1-(Va-Vc)
        fneg.s d4                       ;-1+(Va-Vc)
        jsr pwm_10bit
        move d4.l,x:$17                 ;Vr
        move x:$24,d4.s                 ;1-(Vb-Vc)
        fneg.s d4                       ;-1+(Vb-Vc)
        jsr pwm_10bit
        move d4.l,x:$18                 ;Vy
        jmp end_1

loop_22
        move x:$29,d3.s                 ;Vb
        move x:$30,d4.s                 ;Vc
        move x:$32,d6.s                 ;Ia
        move x:$34,d7.s                 ;Ic
        fcmp d3,d4                      ;Vc-Vb
        fblt loop_2                     ;if Vc<Vb jmp to loop_2
        fcmpm d6,d7                     ;|Ic|-|Ia|
        fblt loop_222                   ;if |Ic| < |Ia| jmp to loop_222

        move #1023,x:$19                ;Vblue=1
        move x:$25,d4.s                 ;1-(Vc-Va)
        jsr pwm_10bit
        move d4.l,x:$17                 ;Vred
        move x:$26,d4.s                 ;1-(Vc-Vb)
        jsr pwm_10bit
        move d4.l,x:$18                 ;Vyellow
        jmp end_1

loop_222
        move #0,x:$17                   ;Vr=-1
        move x:$25,d4.s                 ;1-(Vc-Va)
        fneg.s d4                       ;-1+(Vc-Va)
        jsr pwm_10bit
        move d4.l,x:$19                 ;Vblue
        move x:$23,d4.s                 ;1-(Vb-Va)
        fneg.s d4                       ;-1+(Vb-Va)
        jsr pwm_10bit
        move d4.l,x:$18                 ;Vyellow
        jmp end_1

loop_2
        move x:$28,d3.s                 ;Va
        move x:$29,d4.s                 ;Vb
        move x:$30,d6.s                 ;Vc
        fcmp d6,d4                      ;Vb-Vc
```

```
        fblt loop_3                     ;if Vb<Vc jmp to loop_3
        fcmp d3,d6                      ;Vc-Va
        fblt loop_3                     ;if Vc<Va jmp to loop_3
        move x:$32,d3.s                 ;Ia
        move x:$33,d4.s                 ;Ib
        fcmpm d3,d4                      ;|Ib| - |Ia|
        fblt loop_33                    ;if |Ib|<|Ia| jmp to loop_33

        move #1023,x:$18                ;Vyellow =1
        move x:$23,d4.s                 ;1-(Vb-Va)
        jsr pwm_10bit
        move d4.l,x:$17                 ;Vred
        move  x:$24,d4.s                ;1-(Vb-Vc)
        jsr pwm_10bit
        move d4.l,x:$19                 ;Vblue
        jmp end_1

loop_33
        move #0,x:$17                   ;Vred = -1
        move x:$23,d4.s                  ;1-(Vb-Va)
        fneg.s d4                       ;-1+(Vb-Va)
        jsr pwm_10bit
        move d4.l,x:$18                 ;Vyellow
        move x:$25,d4.s                 ;1-(Vc-Va)
        fneg.s d4                       ;-1+(Vc-Va)
        jsr pwm_10bit
        move d4.l,x:$19                 ;Vblue
        jmp end_1


loop_3
        move x:$28,d3.s                 ;Va
        move x:$30,d4.s                 ;Vc
        move x:$32,d6.s                 ;Ia
        move x:$33,d7.s                 ;Ib
        fcmp d4,d3                      ;Va-Vc
        fblt start                      ;if Va<Vc jmp to start
        fcmpm d7,d6                     ;|Ia| - |Ib|
        fblt loop_4                     ;if |Ia|<|Ib| jmp to loop_4

        move #1023,x:$17                ;Vred = 1
        move x:$21,d4.s                 ;1-(Va-Vb)
        jsr pwm_10bit
        move d4.l,x:$18                 ;Vyellow
        move x:$22,d4.s                 ;1-(Va-Vc)
        jsr pwm_10bit
        move d4.l,x:$19                 ;Vblue
        jmp end_1


loop_4
        move #0,x:$18                   ;Vyellow = -1
        move x:$21,d4.s                 ;1-(Va-Vb)
        fneg.s d4                       ;-1+(Va-Vb)
        jsr pwm_10bit
        move d4.l,x:$17                 ;Vred
        move x:$26,d4.s                 ;1-(Vc-Vb)
        fneg.s d4                       ;-1+(Vc-Vb)
        jsr pwm_10bit
        move d4.l,x:$19

end_1
```

```
                move x:$17,d4.l                 ;
                move d4.l,x:(r4)+               ;
                move x:$18,d4.l                 ; store results in
                move d4.l,x:(r5)+               ; x:$2000, $2800
                move x:$19,d4.l                 ; and x:$3600
                move d4.l,x:(r6)+               ;

out_pwm
                move #out_clr,d3.l
                move x:store_r,d6.l
                move #out_r,d7.l
                add d7.l,d6.l                   ; send 1st reference wavefrom
                movep d6.l,y:outport1           ; to Xilinx
                move #out_y,d7.l
                move x:store_y,d6.l
                add d7.l,d6.l                   ; send 2nd reference wavefrom
                movep d6.l,y:outport1           ; to Xilinx
                move #out_b,d7.l
                move x:store_b,d6.l
                add d7.l,d6.l                   ; send 3rd reference wavefrom
                movep d6.l,y:outport1           ; to Xilinx
                movep d3.l,y:outport1           ;  clear output port

                movep #0,y:outport2             ; clear output port 2
                movep #$1000,y:outport2         ; set DSPRD enabling line conversion

                rti
                nop

pwm_10bit
                move #511.5,d6.s                ;convert data
                fmpy.s d6,d4,d4                 ;for 10 bit
                fadd.s d6,d4                    ;PWM operation
                int d4                          ;
                rts                             ;return back
```

E.3 APF CONTROL - DELTA / MODIFIED DELTA MODULATION

```
;*****************************************************************************
;*      DSP Program to/for :                                                 *
;*              1. set up the initialization routine                         *
;*              2. transfer data from memory to dsp                          *
;*              3. 10th order IIR filter, direct transpose II                *
;*              4. 3-2-3 phase conversion                                    *
;*              5. active/reactive/harmonic calculation                      *
;*              6. harmonic & reactive compensation                          *
;*****************************************************************************

opt     rp                              ;gen NOP to accom. pipeline delay

ipr     equ     $FFFFFFFF               ; address of Interrupt Priority Register
psr     equ     $FFFFFFFC               ; address of Port Select Register
bcra    equ     $FFFFFFFE               ; address of Port A Bus Control Register
bcrb    equ     $FFFFFFFD               ; address of Port B Bus Control Register
```

```
inport1   equ     $FFFFFF90          ; 1st input port
inport2   equ     $FFFFFF91          ; 2nd input port
outport1  equ     $FFFFFF92          ; 1st ouput port
outport2  equ     $FFFFFF93          ; 2nd output port

adcmask     equ     $FFF
ad2047      equ     $7FB             ;(825)2047+41 (due to filter offset)
current_co  equ     0.04374996       ;multiplier for ip current
voltage_co  equ     2.646168         ;multiplier for ip voltage
voltdc_co   equ     5.292336         ;multiplier for ip voltage
o_currco    equ     5.7142909        ;(/2)reciprocal of current multiplier
o_voltco    equ     0.3037357        ;(/2)reciprocal of voltage multiplier
sqrt2_3     equ     (@sqt(2))/(@sqt(3))
sqrt3_2     equ     (@sqt(3))/2

coef        equ     32               ;starting add coef. for IIR filter
data        equ     $2000            ;starting add d(n) for IIR filter
outdata     equ     $2100

band        equ     0.0

out_clr     equ     $1c00
out_r       equ     $0000
out_y       equ     $0400
out_b       equ     $0800

store_r     equ     $55
store_y     equ     $56
store_b     equ     $57


;******************************** main ************************************
;Chebyshev Type II Filter coefficient
            org x:coef
            dc  -0.9402
            dc   0.0130
            dc   1.9393
            dc  -0.0250
            dc   0.0130
            dc  -0.9482
            dc   0.1121
            dc   1.9472
            dc  -0.2232
            dc   0.1121
            dc  -0.9701
            dc   0.2250
            dc   1.9692
            dc  -0.4491
            dc   0.2250
            dc  -0.9912
            dc   0.2894
            dc   1.9904
            dc  -0.5780
            dc   0.2894

org         p:$0                     ; jump from reset to start
jmp         $1FF                     ; of initialization
nop

org         p:$8                     ; IRQA
```

```
jsr     $2000                   ; jump to service routine
nop

org     x:$0
setup_regs  ds   6


org x:setup_regs
dc      $18100                  ; PSR
dc      $200A0000               ; BCRA
dc      $200A0000               ; BCRB
dc      $66                     ; IPR
dc      $00000000               ; SR (interrupt level priority)

        org     p:$1FF          ; starting address for init
        move    #setup_regs,r0
        move    #$8,omr         ; set operating mode register
        movep   x:(r0)+,x:psr   ;
        movep   x:(r0)+,x:bcrb  ; set up  relevant
        movep   x:(r0)+,x:bcra  ; control registers
        movep   x:(r0)+,x:ipr   ; IRQA ipl=2 -ve edge
        move    x:(r0),sr

        move    #outdata,r3
        move    #$2800,r6
        move    #$3600,r7

;initialise switches
        move    #0,x:$55        ;switch1 - OFF
        move    #0.1,x:$58
        move    #1023,x:$56     ;switch2 - ON
        move    #100.2,x:$59
        move    #0,x:$57        ;switch3 - OFF
        move    #0.1,x:$60

        move    #data,r4        ;starting add for data(IIR filter)
        do      #$50,loop2      ;
        move    #0.01,d0.s
        move    d0.s,y:(r4)+    ;clear initial d(n)
loop2
        move    #0.0,d2.s       ;clear initial
        move    d2.s,x:$47      ;Vinmax (A)
        move    #0.0,d2.s       ;clear initial
        move    d2.s,x:$48      ;Vinmax (B)

        movep   #0,y:outport2   ; clear o/p port 2
        movep   #$1000,y:outport2 ; set DSPRD enabling line conversion

loop
        jmp loop

;*************************************************************************
;*                      interrupt service routine for IRQA               *
;*************************************************************************

;*************************************************************************
;*                      1. Transfer data from a/d                        *
;*************************************************************************
        org     p:$2000
        clr     d1.l
        lsl     #9,d1.l
```

```
move    d1.l,y:outport2
move    y:inport1,d2.l
move    #adcmask,d3.l                   ;mask wanted bits only (12 lsb)
and     d3.l,d2.l
move    #$7FB,d3.l                      ;shift down for bipolar  (7FB)
sub     d3,d2                           ;operation
float.s d2
move    #voltage_co,d1.s                ;
fmpy.s  d1,d2,d2                        ;multiply by voltage constant
move    d2.s,x:$10                      ;store data of Vr from transducer

move    #1,d1.l
lsl     #9,d1.l
move    d1.l,y:outport2
move    y:inport1,d2.l
move    #adcmask,d3.l                   ;mask wanted bits only (12 lsb)
and     d3.l,d2.l
move    #$7FA,d3.l                      ;shift down for bipolar
sub     d3,d2                           ;operation
float.s d2
move    #voltage_co,d1.s                ;
fmpy.s  d1,d2,d2                        ;multiply by voltage constant
move    d2.s,x:$11                      ;store data of Vy from transducer

move    x:$10,d3.s                      ;move Vr to d3
fneg.s  d3                              ;-Vr
fsub.s  d2,d3                           ;Vb=-Vr-Vy
move    d3.s,x:$12                      ;store Vb in x:$12

move    #2,d1.l
lsl     #9,d1.l
move    d1.l,y:outport2
move    y:inport1,d2.l
move    #adcmask,d3.l                   ;mask wanted bits only (12 lsb)
and     d3.l,d2.l
move    #$7FF,d3.l                      ;shift down for bipolar(7FF)
sub     d3,d2                           ;operation
float.s d2
move    #current_co,d1.s                ;
fmpy.s  d1,d2,d2                        ;multiply by current constant
fneg.s  d2
move    d2.s,x:$13                      ;store data of Ir from transducer

move    #3,d1.l
lsl     #9,d1.l
move    d1.l,y:outport2
move    y:inport1,d2.l
move    #adcmask,d3.l                   ;mask wanted bits only (12 lsb)
and     d3.l,d2.l
move    #$7FF,d3.l                      ;shift down for bipolar (7F9)
sub     d3,d2                           ;operation
float.s d2
move    #current_co,d1.s                ;
fmpy.s  d1,d2,d2                        ;multiply by current constant
fneg.s  d2
move    d2.s,x:$14                      ;store data of Iy from transducer

move    x:$13,d3.s                      ;move Ir to d3
fneg.s  d3                              ;-Ir
fsub.s  d2,d3                           ;Ib=-Ir-Iy
move    d3.s,x:$15                      ;store Ib in x:$15
```

```
move    #4,d1.l
lsl     #9,d1.l
move    d1.l,y:outport2
move    y:inport1,d2.l
move    #adcmask,d3.l           ;mask wanted bits only (12 lsb)
and     d3.l,d2.l
move    #$7FC,d3.l              ;shift down for bipolar
sub     d3,d2                   ;operation
float.s d2
move    #current_co,d1.s        ;
fmpy.s  d1,d2,d2                ;multiply by current constant
move    d2.s,x:$43              ;store data of Irinject from transducer

move    #7,d1.l
lsl     #9,d1.l
move    d1.l,y:outport2
move    y:inport1,d2.l
move    #adcmask,d3.l           ;mask wanted bits only (12 lsb)
and     d3.l,d2.l
move    #$7FC,d3.l              ;shift down for bipolar
sub     d3,d2                   ;operation
float.s d2
move    #current_co,d1.s        ;
fmpy.s  d1,d2,d2                ;multiply by current constant
move    d2.s,x:$44              ;store data of Iyinject from transducer

move    x:$43,d3.s              ;move Irinject to d3
fneg.s  d3                      ;-Irinject
fsub.s  d2,d3                   ;Ibinject=-Irinject-Iyinject
move    d3.s,x:$45              ;store Ibinject in x:$45

move    #5,d1.l
lsl     #9,d1.l
move    d1.l,y:outport2
move    y:inport1,d2.l
move    #adcmask,d3.l           ;mask wanted bits only (12 lsb)
and     d3.l,d2.l
move    #ad2047,d3.l            ;shift down for bipolar
sub     d3,d2                   ;operation
float.s d2
move    #voltdc_co,d1.s         ;
fmpy.s  d1,d2,d2                ;multiply by voltage constant
move    d2.s,x:$46              ;store data of Vdc from transducer

move    #6,d1.l
lsl     #9,d1.l
move    d1.l,y:outport2
move    y:inport1,d2.l
move    #adcmask,d3.l           ;mask wanted bits only (12 lsb)
and     d3.l,d2.l
move    #ad2047,d3.l            ;shift down for bipolar
sub     d3,d2                   ;operation
float.s d2
fmpy.s  d1,d2,d2                ;multiply by voltage constant
move    d2.s,x:$61              ;store data of Vdc from transducer

movep   #0,y:outport2           ; clear o/p port 2
movep   #$1000,y:outport2       ; set DSPRD enabling line conversion
```

```
;*****************************************************************************
;*                    Low pass (50 Hz) filter for Vsupply                    *
;*****************************************************************************

        move #19,m0                             ;modulo add for coef
        move #7,m4                              ;modulo add for data
        move m4,m5                              ;ditto
        move #coef,r0                           ;starting add. for coef
        move #$10,r1
        move #$51,r2
        move #data,x:$50

    do #3,end_filter2
        move x:$50,d1.l
        move #8,d2.l
        add d1,d2
        move d2.l,x:$50                         ;pointer storage for data
        move d2.l,r4                            ;starting add for data
        move r4,r5                              ;ditto
        move x:(r1)+,d0.s                       ;move Vr into d0.s

    do #4,end_filter1
        fclr d1 x:(r0)+,d4.s  y:(r4)+,d6.s      ;a(2,i)
                                                ;d(i,n-2)
        fmpy.s d4,d6,d1                         ;a(2,i)d(i,n-2)
        fadd.s d1,d0  x:(r0)+,d4.s              ;a(2,i)d(i,n-2) + y(i-1,n)
                                                ;b(2,i)
        fmpy.s d4,d6,d2 x:(r0)+,d4.s  y:(r4)+,d6.s
                                                ;b(2,i)d(i,n-2)
                                                ;a(1,i)
                                                ;d(i,n-1)
        fmpy.s d4,d6,d1                         ;a(1,i)d(i,n-1)
        fadd.s d1,d0  x:(r0)+,d4.s  d6.s,y:(r5)+ ;a(2,i)d(i,n-2) + a(1,i)d(i,n-1) + y(i-1,n)
                                                ;this is d(i,n)
                                                ;b(1,i)
                                                ;d(i,n-1) becomes d(i,n-2)
        fmpy.s d4,d6,d1                         ;b(1,i)d(i,n-1)

        fadd.s d1,d2  x:(r0)+,d4.s  d0.s,y:(r5)+ ;b(2,i)d(i,n-2) + b(1,i)d(i,n-1)
                                                ;b(0,i)
                                                ;d(i,n) becomes d(i,n-1)

        fmpy.s d4,d0,d0                         ;b(0,i)d(i,n)
        fadd.s d2,d0                            ;b(2,i)d(i,n-2) + b(1,i)d(i,n-1) + b(0,i)d(i,n)
                                                ;y(i,n)
end_filter1
        move d0.s,x:(r2)+
end_filter2

;*****************************************************************************
;*                          voltage peak detector                            *
;*****************************************************************************

        move x:$51,d2.s                         ;Vrfiltered
        move #0.0,d3.s
        fsub.s d3,d2
        fbmi minus
        move x:$47,d1.s                         ;Vinmax
        fcmp d1,d2                              ;Vr-Vinmax
        fblt less                               ;jmp to less if -ve
        move d2.s,x:$47                         ;store Vinmax in x:$47
        jmp less1
```

```
less
        move x:$47,d1.s
        move d1.s,x:$48                         ;store peak value in x:$48
        jmp less1
minus
        move #0.0,d1.s                          ;reset Vinmax
        move d1.s,x:$47                         ;when data is -ve
less1

;*************************************************************************
;*                          3ph to 2ph transformation                    *
;*************************************************************************

        move    #$51,r1                         ;load starting address for Efiltered

;****************************** for V's *********************************

        move    x:(r1)+,d2.s                    ;load Er into d2
        move    x:(r1)+,d3.s                    ;load Ey into d3
        move    x:(r1)+,d4.s                    ;load Eb into d4

        move    d3.s,d5.s                       ;save to be
        move    d4.s,d6.s                       ;restored
        move    d2.s,d7.s
        move    #0.083678,d1.s
        fmpy.s  d1,d3,d3                        ;-0.996493*Ey
        move    #0.821149,d1.s
        fmpy.s  d1,d4,d4                        ;0.570714*Eb
        fadd.s  d3,d4                           ;-0.996493*Ey + 0.570714*Eb
        move    #-0.904827,d1.s
        fmpy.s  d1,d2,d2                        ;0.42578*Er
        fadd.s  d4,d2                           ;0.42578*Er -0.996493*Ey + 0.570714*Eb
        move    #sqrt2_3,d0.s
        fmpy.s  d0,d2,d2                        ;sqrt2_3(0.42578*Er -0.996492*Ey +0.570714*Eb)
        move    d2.s,x:$16                      ;store E - alpha

        move    d5.s,d3.s                       ;restore original contents
        move    d6.s,d4.s                       ;for E's to find E-beta
        move    d7.s,d2.s

        move    #-0.996493,d1.s
        fmpy.s  d1,d3,d3                        ;-0.083678*Ey
        move    #0.570714,d1.s
        fmpy.s  d1,d4,d4                        ;-0.821149*Eb
        fadd.s  d3,d4                           ;-0.083678*Ey - 0.821149*Eb
        move    #0.425779,d1.s
        fmpy.s  d1,d2,d2                        ;0.904827*Er
        fadd.s  d4,d2                           ;0.904827*Er - 0.083678*Ey - 0.821149*Eb
        move    #sqrt2_3,d0.s
        fmpy.s  d0,d2,d2                        ;sqrt2_3(0.904827*Er - 0.083678*Ey-0.821149*Eb)
        move    d2.s,x:$17                      ;store E - beta

;****************************** for I's *********************************

        move    #$13,r1                         ;load starting address for I
        nop                                     ;to prevent pipelining error

        move    x:(r1)+,d2.s                    ;load Ir into d2
        move    x:(r1)+,d3.s                    ;load Iy into d3
```

```
move    x:(r1)+,d4.s                ;load Ib into d4

move    d3.s,d5.s                   ;save to be
move    d4.s,d6.s                   ;restored
move    #0.5,d1.s
fmpy.s  d1,d3,d3                    ;0.5*Iy
fmpy.s  d1,d4,d4                    ;0.5*Ib
fadd.s  d3,d4                       ;0.5Iy + 0.5Ib
fsub.s  d4,d2                       ;Ir - 0.5(Iy + Ib)
move    #sqrt2_3,d0.s
fmpy.s  d0,d2,d2                    ;sqrt2_3(Ir - 0.5(Iy+Ib))
move    d2.s,x:$18                  ;store I - alpha

move    d5.s,d3.s                   ;restore original contents
move    d6.s,d4.s                   ;for I's

move    #sqrt3_2,d1.s
fmpy.s  d1,d3,d3                    ;sqrt3_2 * Iy
fmpy.s  d1,d4,d4                    ;sqrt3_2 * Ib
fsub.s  d4,d3                       ;(sqrt3_2 * Iy) - (sqrt3_2 * Ib)
fmpy.s  d0,d3,d3                    ;sqrt2_3 times above
move    d3.s,x:$19                  ;store I - beta

move    x:$16,d2.s                  ;move Ealfa to d2
move    x:$18,d3.s                  ;move Ialfa to d3
fmpy.s  d2,d3,d4  x:$19,d3.s        ;Ealfa x Ialfa, move Ibeta to d3
move    x:$17,d2.s                  ;move Ebeta to d2
fmpy.s  d2,d3,d5  x:$16,d2.s        ;P & move Ealfa to d2
fadd.s  d5,d4
move    d4.s,x:$1A                  ; move P to x:$1A
fmpy.s  d2,d3,d4 x:$17,d2.s         ; Ealfa x Ibeta, move Ebeta to d2
move    x:$18,d3.s                  ;move Ialfa to d3
fmpy.s  d2,d3,d5
fsub.s  d5,d4                       ; Q
move    d4.s,x:$1B                  ;move Q to x:$1B

;*****************************************************************************
;*              implement 8th order IIR filter with                          *
;*              4 cascaded biquad                                            *
;*              coef store at x:$20..$38     d(n) store at y:$2000..$200E    *
;*              input p at x:$1A                                             *
;*****************************************************************************

        move    #19,m0                  ;modulo add for coef
        move    #7,m4                   ;modulo add for data
        move    m4,m5                   ;ditto
        move    #coef,r0                ;starting add. for coef
        move    #data,r4                ;starting add for data
        move    r4,r5                   ;ditto

        move    x:$1A,d0.s              ;move P into d0.s
        do      #4,end_filter

        fclr d1 x:(r0)+,d4.s  y:(r4)+,d6.s      ;a(2,i)
                                                ;d(i,n-2)
        fmpy.s  d4,d6,d1                        ;a(2,i)d(i,n-2)
        fadd.s  d1,d0  x:(r0)+,d4.s             ;a(2,i)d(i,n-2) + y(i-1,n)
                                                ;b(2,i)

        fmpy.s  d4,d6,d2 x:(r0)+,d4.s  y:(r4)+,d6.s
                                                ;b(2,i)d(i,n-2)
```

```
                                                ;a(1,i)
                                                ;d(i,n-1)
fmpy.s  d4,d6,d1                                ;a(1,i)d(i,n-1)

fadd.s  d1,d0  x:(r0)+,d4.s  d6.s,y:(r5)+
                                                ;a(2,i)d(i,n-2) + a(1,i)d(i,n-1) + y(i-1,n)
                                                ;this is d(i,n)
                                                ;b(1,i)
                                                ;d(i,n-1) becomes d(i,n-2)
fmpy.s  d4,d6,d1                                ;b(1,i)d(i,n-1)

fadd.s  d1,d2  x:(r0)+,d4.s  d0.s,y:(r5)+
                                                ;b(2,i)d(i,n-2) + b(1,i)d(i,n-1)
                                                ;b(0,i)
                                                ;d(i,n) becomes d(i,n-1)

        fmpy.s  d4,d0,d0                        ;b(0,i)d(i,n)
        fadd.s  d2,d0                           ;b(2,i)d(i,n-2) + b(1,i)d(i,n-1) + b(0,i)d(i,n)
                                                ;y(i,n)
end_filter
        move    x:$1A,d1.s                      ;move original P to d1
        fsub.s  d0,d1                           ;Poriginal - Pfiltered
        move    d1.s,x:$1C                      ;store in x:$1C

;*****************************************************************
;*                          control P                            *
;*****************************************************************
        move    x:$48,d1.s                      ;Vin (Vr)
        move    #0.004347826,d2.s               ;1/230
        fmpy.s  d1,d2,d1  #400.0,d2.s           ;Vin/230
        fmpy.s  d1,d2,d2  x:$46,d3.s            ;(Vin/230)*400, Vdcref
                                                ;move Vdcsample
        fsub.s  d3,d2                           ;Vdcref - Vdcsample (Vdcerror)
        move    x:$61,d4.s                      ;input from variable resistor
        move    #0.04584,d0.s                   ;200/4363 (to give +-200V)
        fmpy.s  d0,d4,d0                        ;sample*200/4363 (Vdcoffset)
        fadd.s  d0,d2                           ;Vdcerror + Vdcoffset
        move    #30.0,d3.s                      ;
        fmpy.s  d1,d3,d1                        ;(Vin/230)*50, scaled gain
        fmpy.s  d1,d2,d3   x:$1C,d1.s           ;gain*(Vdcerror+Vdcoffset)
                                                ;move req. P into d1
        fadd.s  d3,d1                           ;P=P+ gain*(Vdcerror+Vdcoffset)
        move    d1.s,x:$1C                      ;store again in x:$1C

;*****************************************************************
;*                                                               *
;*                  2ph to 3ph transformation                    *
;*                  d1 - P      d0 - Q                           *
;*****************************************************************

        move    x:$16,d2.s                      ;move Ealfa to d2
        fmpy.s  d2,d2,d4                        ;Elfa * Ealfa
        move    x:$17,d3.s                      ;move Ebeta to d3
        fmpy.s  d3,d3,d5                        ;Ebeta * Ebeta
        fadd.s  d5,d4                           ;Ealfa*Ealfa + Ebeta*Ebeta (det.)
        fseedd  d4,d4                           ;1/det

        move    x:$1C,d5.s                      ;move Poriginal-Pfiltered into d5
        fmpy.s  d2,d5,d0  x:$1B,d5.s            ;Ealfa*(Poriginal-Pfiltered)
                                                ;move Q into d5
        fmpy.s  d3,d5,d1                        ;Ebeta*Q
```

```
fsub.s   d1,d0                    ;Ealfa*(Poriginal-Pfiltered) - Ebeta*Q
fmpy.s   d4,d0,d0  x:$1C,d5.s     ;(1/det)*(Ealfa*Ialfa - Ebeta*Ialfa)
                                  ;move Poriginal-Pfiltered into d5
move     d0.s,x:$1D               ;store Aalfa into x:$1D

fmpy.s   d3,d5,d0  x:$1B,d5.s     ;Ebeta*(Poriginal-Pfiltered)
                                  ;move Q into d5
fmpy.s   d2,d5,d1                 ;Ealfa*Q
fadd.s   d1,d0                    ;Ebeta*(Poriginal-Pfiltered) + Ealfa*Q
fmpy.s   d4,d0,d0                 ;(1/det)*(Ebeta*Ialfa + Ealfa*Ialfa)
move     d0.s,x:$1E               ;store Balfa into x:$1E

move     x:$1D,d0.s               ;Aalfa
move     #sqrt2_3,d1.s            ;sqrt(2/3)
fmpy.s   d1,d0,d2  #-0.5,d4.s     ;Iacomp, move -0.5 into d2
fneg.s   d2
fmpy.s   d4,d0,d3  d2.s,x:$40     ;-0.5*Aalfa, store Iacomp in x:$40
move     x:$1E,d2.s               ;move Balfa into d2
move     #sqrt3_2,d4.s            ;(sqrt(3))/2 - 0.866
fmpy.s   d2,d4,d4                 ;0.866*Balfa
move     d4.s,d0.s
fadd.s   d3,d0                    ;-0.5*Aalfa + 0.866*Balfa
fmpy.s   d1,d0,d5                 ;sqrt(2/3)*(-0.5*Aalfa + 0.866*Balfa)
fneg.s   d5                       ;Ibcomp
fsub.s   d4,d3  d5.s,x:$41        ;-0.5*Aalfa + 0.866*Balfa
                                  ;store Ibcomp in x:$41
fmpy.s   d1,d3,d5                 ;sqrt(2/3)*(-0.5*Aalfa - 0.866*Balfa)
fneg.s   d5                       ;Iccomp
move     d5.s,x:$42               ;store Iccomp in x:$42

;*****************************************************************
;*                        switching signals                      *
;*****************************************************************
        move    x:$40,d6.s        ;Iacomp
        move    x:$43,d7.s        ;Iainject
        fcmp    d6,d7             ;Iainject - (Iacomp+2)
        fblt    end_1             ;if Iainject is lt, jmp to end1
        move    #0,x:$58          ;switch1 - OFF
        move    #0.1,x:$66
        jmp     end_11
end_1
        move    x:$40,d6.s        ;Iacomp
        move    x:$43,d7.s        ;Iainject
        fcmp    d6,d7             ;Iainject - (Iacomp-2)
        fbgt    end_11            ;if Iainject is gt, jmp to end_11
        move    #2,x:$58          ;switch1 - ON
        move    #100.2,x:$66
end_11

        move    x:$41,d1.s        ;Ibcomp
        move    x:$44,d2.s        ;Ibinject
        fcmp    d6,d7             ;Ibinject - (Ibcomp+2)
        fblt    end_2             ;if Ibinject is lt, jmp to end_2
        move    #0,x:$59          ;switch2 - OFF
        move     #0.1,x:$67
        jmp     end_21
end_2
        move    x:$41,d6.s        ;Ibcomp
        move    x:$44,d7.s        ;Ibinject
        fcmp    d6,d7             ;Ibinject - (Ibcomp-2)
        fbgt    end_21            ;if Ibinject is gt, jmp to end_21
```

```
        move    #8,x:$59                        ;switch2 - ON
        move    #100.2,x:$67
end_21
        move    x:$42,d6.s              ;Iccomp
        move    x:$45,d7.s              ;Icinject
        fcmp    d6,d7                   ;Icinject - (Iccomp+2)
        fblt    end_3                   ;if Icinject is lt, jmp to end_3
        move    #0,x:$60                ;switch3 - OFF
        move    #0.1,x:$68
        jmp     end_31
end_3
        move    x:$42,d6.s              ;Iccomp
        move    x:$45,d7.s              ;Icinject
        fcmp    d6,d7                   ;Icinject - (Iccomp-2)
        fbgt    end_31                  ;if Icinject is gt, jmp to end_31
        move    #32,x:$60               ;switch3 - ON
        move    #100.2,x:$68
end_31

;*****************************************************************
;*          include the following sections for modified delta modulation    *
;*                          with deadband                                   *
;*****************************************************************
        move    x:$10,d0.s                      ;Va
        move    x:$11,d1.s                      ;Vb
        move    x:$12,d2.s                      ;Vc
        move    x:$40,d3.s                      ;Iacomp
        move    x:$41,d4.s                      ;Ibcomp
        move    x:$42,d5.s                      ;Iccomp

        fcmp    d1,d0                           ;Va-Vb
        fblt    dband_1                         ;if Va<Vb jmp to dband_1
        fcmp    d2,d1                           ;Vb-Vc
        fblt    dband_2                         ;if Vb<Vc jmp to dband_2
        fcmpm   d5,d3                           ;Ia-Ic
        fblt    amp_1                           ;if abs(Ia)<abs(Ic) jmp amp_1
        move    #2,x:$58                        ;switch1 - ON
        move    #100.2,x:$66                    ;(Va>Vb)&(Vb>Vc) abs(Ia)>abs(Ic)
        jmp     end_dband

amp_1   move    #32,x:$60                       ;switch3 - ON
        move    #100.2,x:$68                    ;(Va>Vb)&(Vb>Vc) abs(Ia)<abs(Ic)
        jmp     end_dband                       ;

dband_1 fcmp    d2,d0                           ;Va-Vc
        fblt    dband_3                         ;if Va<Vc jmp to dband_3
        fcmpm   d5,d4                           ;abs(Ib)-abs(Ic)
        fblt    amp_2                           ;jmp if abs(Ib)<abs(Ic)
        move    #8,x:$59                        ;switch2 - ON
        move    #100.2,x:$67                    ;(Vb>Va)&(Va>Vc) abs(Ib)>abs(Ic)
        jmp     end_dband

amp_2   move    #0,x:$60                        ;switch3 - OFF
        move    #0.1,x:$68                      ;(Vb>Va)&(Va>Vc) abs(Ib)<abs(Ic)
        jmp     end_dband                       ;

dband_2 fcmp    d0,d2                           ;Vc-Va
        fblt    dband_3                         ;if Vc<Va jmp to dband_3
        fcmpm   d4,d5                           ;abs(Ic)-abs(Ib)
        fblt    amp_3                           ;jmp if abs(Ic)<abs(Ib)
        move    #32,x:$60                       ;switch3 - ON
```

```
        move    #100.2,x:$68            ;(Vc>Va)&(Va>Vb) abs(Ic)>abs(Ib)
        jmp     end_dband

amp_3   move    #0,x:$59                ;switch2 - OFF
        move    #0.1,x:$67              ;(Vc>Va)&(Va>Vb) abs(Ic)<abs(Ib)
        jmp     end_dband               ;

dband_3 fcmp    d2,d0                   ;Va-Vc
        fblt    dband_4                 ;if Va<Vc jmp to dband_4
        fcmp    d1,d2                   ;Vc-Vb
        fblt    dband_5                 ;if Vc<Vb jmp to dband_5
        fcmpm   d4,d3                   ;abs(Ia)-abs(Ib)
        fblt    amp_4                   ;jmp if abs(Ia)<abs(Ib)
        move    #2,x:$58                ;switch1 - ON
        move    #100.2,x:$66            ;(Va>Vc)&(Vc>Vb) abs(Ia)>abs(Ib)
        jmp     end_dband

amp_4   move    #0,x:$59                ;switch2 - OFF
        move    #0.1,x:$67              ;(Va>Vc)&(Vc>Vb) abs(Ia)<abs(Ib)
        jmp     end_dband               ;

dband_4 fcmp    d2,d1                   ;Vb-Vc
        fblt    dband_5                  ;if Vb<Vc jmp to dband_5
        fcmpm   d3,d4                   ;abs(Ib)-abs(Ia)
        fblt    amp_5                   ;jmp if abs(Ib)<abs(Ia)
        move    #8,x:$59                ;switch2 - ON
        move    #100.2,x:$67            ;(Vb>Vc)&(Vc>Va) abs(Ib)>abs(Ia)
        jmp     end_dband

amp_5   move    #0,x:$58                ;switch1 - OFF
        move    #0.1,x:$66              ;(Vb>Vc)&(Vc>Va) abs(Ib)<abs(Ia)
        jmp     end_dband               ;

dband_5 fcmpm   d3,d5                   ;abs(Ic)-abs(Ia)
        fblt    amp_6                   ;jmp if abs(Ic)<abs(Ia)
        move    #32,x:$60               ;switch3 - ON
        move    #100.2,x:$68            ;for the rest & abs(Ic)>abs(Ia)
        jmp     end_dband

amp_6   move    #0,x:$58                ;switch1 - OFF
        move    #0.1,x:$66              ;for the rest & abs(Ic)<abs(Ia)

end_dband

;************ for equal signal
        move    x:$55,d1.l              ;previous state of switch1
        move    x:$58,d2.l              ;current state of switch1
        fcmp    d2,d1                   ;compare both
        fbne    eq_1                    ;if not equal jump
        move    #0,x:$69                ;equal signal is zero
eq_1
        move    #1,x:$69                ;equal signal is 1
        move    d2.l,x:$55              ;move current to previous

        move    x:$56,d1.l              ;previous state of switch2
        move    x:$59,d2.l              ;current state of switch2
        fcmp    d2,d1                   ;compare both
        fbne    eq_2                    ;if not equal jump
        move    #0,x:$6A                ;equal signal is zero
eq_2
        move    #4,x:$6A                ;equal signal is 1
```

```
        move    d2.l,x:$56              ;move current to previous

        move    x:$57,d1.l              ;previous state of switch3
        move    x:$60,d2.l              ;current state of switch3
        fcmp    d2,d1                   ;compare both
        fbne    eq_3                    ;if not equal jump
        move    #0,x:$6B                ;equal signal is zero
eq_3
        move    #16,x:$6B               ;equal signal is 1
        move    d2.l,x:$57              ;move current to previous

        move    x:$56,d1.l              ;switch1
        move    x:$57,d2.l              ;switch2
        add     d2,d1
        move    x:$58,d2.l              ;switch3
        add     d2,d1
        move    d1.l,x:$6C              ;pwm_only
        move    x:$69,d2.l              ;equal1
        add     d2,d1
        move    x:$6A,d2.l              ;equal2
        add     d2,d1
        move    x:$6B,d2.l              ;equal3
        add     d2,d1
        move    d1.l,x:$6D              ;pwm and equal

;*****************************************************************
;*              create pwm and equal signals for underlap        *
;*****************************************************************
        move    #out_clr,d3.l
        move    #out_r,d7.l
        move    x:$6D,d6.l
        add     d7.l,d6.l
        movep   d6.l,y:outport1         ;o/p pwm and equal

        do      #4,end_delay            ;no of count determine
        move    x:$6C,d6.l              ;duration for underlap
        add     d7.l,d6.l               ;(this is delay for
        nop                             ; equal signal )
end_delay
        movep   d6.l,y:outport1         ;o/p pwm only

        movep   d3.l,y:outport1         ;clear

;*****************************************************************
;*                      output results to DAC                    *
;*****************************************************************
        move    #$2C00,d1.l             ;first DAC output (use Gray code)
        move    #o_currco,d3.s          ;reciprocal of current multiplier
        move    x:$66,d2.s              ;restore Iacomp
        fmpy.s  d2,d3,d2                ;value for DAC
        int     d2                      ;integerized
        move    #128,d5.l               ;scale for bipolar operation
        add     d5,d2
        move    #$1000,d5.l             ;set dsprd all the time
        add     d5,d2
        movep   d1.l,y:outport1         ;select DAC output
        movep   d2.l,y:outport2         ;output Iacomp

        move    #$6C00,d1.l             ;2nd DAC output (use Gray code)
        move    x:$13,d2.s              ;restore Ibcomp
        move    #o_currco,d3.s          ;reciprocal of current multiplier
```

```
fmpy.s  d2,d3,d2                ;value for DAC
int     d2                      ;integerized
move    #128,d5.l               ;scale for bipolar operation
add     d5,d2
move    #$1000,d5.l             ;set dsprd all the time
add     d5,d2
movep   d1.l,y:outport1         ;select DAC output
movep   d2.l,y:outport2         ;output Ibcomp

move    #$4C00,d1.l             ;3rd DAC output (use Gray code)
move    x:$40,d2.s              ;restore Iccomp
fmpy.s  d2,d3,d2                ;value for DAC
int     d2                      ;integerized
move    #128,d5.l               ;scale for bipolar operation
add     d5,d2
move    #$1000,d5.l             ;set dsprd all the time
add      d5,d2
movep   d1.l,y:outport1         ;select DAC output
movep   d2.l,y:outport2         ;output Iccomp

move    #$CC00,d1.l             ;4th DAC output (use Gray code)
move    x:$18,d2.s              ;restore Iccomp
fmpy.s  d2,d3,d2                ;value for DAC
int d2                          ;integerized
move    #128,d5.l               ;scale for bipolar operation
add     d5,d2
move    #$1000,d5.l             ;set dsprd all the time
add     d5,d2
movep   d1.l,y:outport1         ;select DAC output
movep   d2.l,y:outport2         ;output Iccomp

move    #$8C00,d1.l             ;5th DAC output (use Gray code)
move    x:$43,d2.s              ;restore Ibcomp
fmpy.s  d2,d3,d2                ;value for DAC
int     d2                      ;integerized
move    #128,d5.l               ;scale for bipolar operation
add     d5,d2
move    #$1000,d5.l             ;set dsprd all the time
add     d5,d2
movep   d1.l,y:outport1         ;select DAC output
movep   d2.l,y:outport2         ;output Ibcomp

movep   #$0C00,y:outport1       ;lacth DAc selector
rti
```

E.4 APF CONTROL - SLIDING MODE

```
;*****************************************************************************
;*                        For initialisation, see section B.3                *
;*****************************************************************************
        org     p:$1FF                  ; starting address for init
        move    #setup_regs,r0
        move    #$8,omr                 ; set operating mode register
        movep   x:(r0)+,x:psr           ;
        movep   x:(r0)+,x:bcrb          ; set up  relevant
        movep   x:(r0)+,x:bcra          ; control registers
        movep   x:(r0)+,x:ipr           ; IRQA ipl=2 -ve edge
```

```
        move    x:(r0),sr

        move    #outdata,r3
        move    #$2800,r6
        move    #$3600,r7

;initialise switches
        move    #0,x:$55                ;switch1 - OFF
        move    #0.1,x:$58
        move    #1023,x:$56             ;switch2 - ON
        move    #100.2,x:$59
        move    #0,x:$57                ;switch3 - OFF
        move    #0.1,x:$60

        move    #data,r4                ;starting add for data(IIR filter)
        do      #$50,loop2              ;
        move    #0.01,d0.s
        move    d0.s,y:(r4)+            ;clear initial d(n)
loop2
        move    #0.0,d2.s               ;clear initial
        move    d2.s,x:$47              ;Vinmax (A)
        move    #0.0,d2.s               ;clear initial
        move    d2.s,x:$48              ;Vinmax (B)

;initialise counter
        move    d2.s,x:$77              ;clear previous Vs
        move    d2.s,x:$78              ;clear counter
        move    d2.s,x:$7A              ;clear for lookup table
        move    #$400,d2.l
        move    d2.l,x:$7B              ;set pointer at $400

;*************************************************************************
;*                lookup table for arctan (0 - 90)                       *
;*************************************************************************

        move    #$3EFF,r0               ;starting add for arctan
        move    #$400,r1                ;starting add for sin/cos

        do      #256,arctan
        move    y:(r1),d0.s             ;sine
        move    x:(r1)+,d1.s            ;cosine
        fseedd  d1,d1                   ;1/cosine
        fmpy.s  d0,d1,d0                ;sine/cosine (tan)
        move    d0.s,x:(r0)+            ;store tan
arctan

;*************************************************************************


        movep   #0,y:outport2           ; clear o/p port 2
        movep   #$1000,y:outport2       ; set DSPRD enabling line conversion

loop
        jmp     loop

;*************************************************************************
;*                interrupt service routine for IRQA                     *
;*************************************************************************

;*************************************************************************
;*                1. Transfer data from a/d                              *
;*************************************************************************
```

```
;*                                    (see B.3)
;*****************************************************************************

;*****************************************************************************
;*                        counter  to determine angle                        *
;*****************************************************************************
            move    x:$78,d3.s              ;restore counter
            move    x:$10,d1.s              ;restore Vr
            move    #1.0,d0.s               ;
            fcmp    d0,d1                   ;d1 - d0
            fbmi    inc                     ;if Vr<0 then jump
            move    x:$77,d2.s              ;prev value of Vr
            move    #0.0,d0.s
            fcmp    d0,d2                   ;d2 - d0
            fbgt    inc                     ;if Vrprev>0 then jump

            move    #300.0,d0.s             ;do not reset if
            fcmp    d0,d3                   ;count < 300
            fblt    inc                     ;(360 deg = 389)

            move    d3.s,x:$79              ;max value of counter
            move    #0.0,d0.s
            move    d0.s,x:$78              ;reset counter
            move    y:$400,d6.s             ;reset to 0 deg
            move    d6.s,x:$7A
            move    #$400,d6.l
            move    y:(r6),d5.s
            fneg.s  d5
            move    d5.s,x:$7A

            float.s d6
            move    d6.s,x:$7B              ;store location of sine
            jmp     end_count
inc
            move    #1.0,d4.s
            fadd.s  d4,d3
            move    d3.s,x:$78              ;inc counter

            move    #2.618926,d4.s          ;1024/391
            move    x:$7B,d6.s              ;previous sin pointer
            fadd.s  d4,d6                   ;
            move    d6.s,x:$7B              ;store current pointer
            int     d6                      ;
            move    d6.l,r6
            move    y:(r6),d6.s
            fneg.s  d6
            move    d6.s,x:$7A


end_count
            move    x:$10,d1.s              ;save current Vr as
            move    d1.s,x:$77              ;previous Vr

;*****************************************************************************
;*                                                                           *
;*                  See B.2 for :                                            *
;*                          i.  Low pass (50 Hz) filter for Vsupply          *
;*                          ii. Voltage peak detector                        *
;*                          iii. IIR Chebyshev filter                        *
;*                          iv. P control                                    *
;*                          v.  2ph to 3ph transformation                    *
;*****************************************************************************
```

```
;*****************************************************************
;*                    3 phase to d-q transformation              *
;*****************************************************************
        move    #$40,r1                 ;load starting address for I
        move    #$80,r2                 ;storage for I alfa-beta, d-q
        nop                             ;to prevent pipelining error

        do      #2,end_dq
        move      x:(r1)+,d2.s          ;load Ir into d2
        move      x:(r1)+,d3.s          ;load Iy into d3
        move      x:(r1)+,d4.s          ;load Ib into d4

        move      d3.s,d5.s             ;save to be
        move      d4.s,d6.s             ;restored
        move      #0.5,d1.s
        fmpy.s    d1,d3,d3              ;0.5*Iy
        fmpy.s    d1,d4,d4              ;0.5*Ib
        fadd.s    d3,d4                 ;0.5Iy + 0.5Ib
        fsub.s    d4,d2                 ;Ir - 0.5(Iy + Ib)
        move      #sqrt2_3,d0.s
        fmpy.s    d0,d2,d2              ;sqrt2_3(Ir - 0.5(Iy+Ib))
        move      d2.s,x:(r2)+          ;store Iref - alpha

        move      d5.s,d3.s             ;restore original contents
        move      d6.s,d4.s             ;for I's

        move      #sqrt3_2,d1.s
        fmpy.s    d1,d3,d3              ;sqrt3_2 * Iy
        fmpy.s    d1,d4,d4              ;sqrt3_2 * Ib
        fsub.s    d4,d3                 ;(sqrt3_2 * Iy) - (sqrt3_2 * Ib)
        fmpy.s    d0,d3,d3              ;sqrt2_3 times above
        move      d3.s,x:(r2)+          ;store I - beta

        move      x:$7B,d0.s            ;pointer for sin/cos table
        int       d0                    ;integerized
        move      d0.l,r0               ;restore address for sin/cos
        nop                             ;prevent pipelining error
        move      x:(r0),d0.s           ;current cosine value
        fneg.s    d0
        move      y:(r0),d1.s           ;current sine value
        fneg.s    d1
        fmpy.s    d0,d2,d4              ;(I_alpha)*cos(theta)
        fmpy.s    d1,d3,d5              ;(I_beta)*sin(theta)
        fadd.s    d5,d4                 ;I_d
        move      d4.s,x:(r2)+          ;store I_d

        fneg.s    d1                    ;-sine
        fmpy.s    d1,d2,d4              ;-(I_alpha)*sin(theta)
        fmpy.s    d0,d3,d5              ;(I_beta)*cos(theta)
        fadd.s    d5,d4                 ;I_q
        move      d4.s,x:(r2)+          ;store I_q
end_dq

        move    :$82,d1.s               ;Iref_d
        move    x:$86,d2.s              ;Iinj_d
        fsub.s  d2,d1                   ;Iref_d - Iinj_d
        fneg.s  d1
        move    d1.s,x:$88              ;store Ed

        move    x:$83,d1.s              ;Iref_q
```

```
        move    x:$87,d2.s                      ;Iinj_q
        fsub.s  d2,d1                           ;Iref_q - Iinj_q
        fneg.s  d1
        move    d1.s,x:$89                      ;store Eq

;*****************************************************************
;*                          differentiator                       *
;*****************************************************************
        move    #$82,r0                         ;starting add for Iref
        move    #$8A,r1                         ;starting add for d_Iref
        move    #$8C,r2                         ;starting add for previous Iref

        do      #2,end_diff
        move    x:(r0),d0.s                     ;current Ia_comp
        move    x:(r2),d1.s                     ;previous Ia_comp
        fsub.s  d1,d0                           ;I_current - I_previous
        move    #20000.0,d3.s                   ;1/(time diff) 1/50e-6
        fmpy.s  d0,d3,d0                        ;
        move    d0.s,x:(r1)+                    ;d_Iacomp
        move    x:(r0)+,d0.s
        move    d0.s,x:(r2)+                    ;for previous Ia_comp
end_diff

;*****************************************************************
;*                          find Ud/Uq                           *
;*****************************************************************
        move    #314.159265,d0.s                ;Wr
        move    x:$87,d1.s                      ;Iq_inject
        fmpy.s  d0,d1,d0                        ;Wr * Iq_inject
        move    x:$48,d1.s                      ;E (peak value)
        move    #1.2247449,d2.s                 ;sqrt(3/2)
        fmpy.s  d1,d2,d1                        ;sqrt(3/2) * E
        fsub.s  d0,d1                           ;sqrt(3/2)*E - Wr*Iq_inject
        move    x:$8A,d2.s                      ;dIref_d
        fadd.s  d2,d1                           ;dIref_d+sqrt(3/2)*E - Wr*Iq_inject
        move    x:$46,d0.s                      ;Vdc
        fseedd  d0,d0                           ;1/Vdc
        move    #0.00645,d2.s                   ;L
        fmpy.s  d2,d0,d2                        ;L/Vdc;
        fmpy.s  d2,d1,d2                        ;L/Vdc * (sum above)
        move    d2.s,x:$8E                      ;Ud

        move    #314.159265,d0.s                ;Wr
        move    x:$86,d1.s                      ;Id_inject
        fmpy.s  d0,d1,d1                        ;Wr * Id_inject
        move    x:$8B,d2.s                      ;dIref_q
        fadd.s  d2,d1                           ;dIref_q + Wr*Id_inject
        move    x:$46,d0.s                      ;Vdc
        fseedd  d0,d0                           ;1/Vdc
        move    #0.00645,d2.s                   ;L
        fmpy.s  d2,d0,d2                        ;L/Vdc;
        fmpy.s  d2,d1,d2                        ;L/Vdc * (sum above)
        move    d2.s,x:$8F                      ;Uq

        move    x:$8E,d1.s                      ;Ud
        fseedd  d1,d1                           ;1/Ud
        fmpy.s  d1,d2,d1                        ;Uq/Ud
        fabs.s  d1
        move    d1.s,x:$90                      ;save Uq/Ud

;*****************************************************************
```

```
;*                          search for the true angle                                *
;*                          approx. valid up to 45 deg                               *
;*****************************************************************************
          move    x:$90,d0.s                    ;value of Uq/Ud
          move    #142.0,d1.s                   ;(30/0.6)*(256/90)
          fmpy.s  d0,d1,d0                      ;angle in deg
          move    #128.0,d2.s                   ;maximum  at 45 deg
          fcmp    d2,d0
          fbgt    end1_angle                    ;if >45, limit at
          move    d2.s,x:$91                    ;45 only
          jmp     end_angle
end1_angle
          move    d1.s,x:$91
end_angle


;*****************************************************************************
;*                          switching signals                                *
;*****************************************************************************
          move    x:$7B,d3.s                    ;pointer for sin/cosine table
          move    x:$91,d0.s                    ;angle
          fadd.s  d0,d3                         ;add together,new angle
          move    d3.s,x:$92                    ;store current pointer

          move    x:$88,d0.s                    ;Ed
          move    x:$89,d1.s                    ;Eq
          move    #0.0,d2.s
          fsub.s  d2,d0                         ;Ed - 0
          fblt    sw_Ed                         ;jump if Ed<0
          fsub.s  d2,d1                         ;Eq - 0
          fblt    sw_Eq1                        ;jump if Eq<0
sw_11     move    #1109.3333,d4.s               ;angle = 30 deg
          fsub.s  d3,d4                         ;30 - deg
          fbgt    sw_16                         ;if deg<30, jump to sw_11
          move    #1280.0,d4.s                  ;angle = 90 deg
          fsub.s  d3,d4                         ;90 - deg
          fblt    sw_12                         ;if deg>90,jump to sw_12
          move    #2,x:$58                      ;switch1 - ON
          move    #100.2,x:$66                  ;switch1 for DAC
          move    #8,x:$59                      ;switch2 - ON
          move    #100.2,x:$67                  ;switch2 for DAC
          move    #0,x:$60                      ;switch3 - OFF
          move    #0.1,x:$68                    ;switch3 for DAC
          jmp     end_sw
sw_12     move    #1450.6667,d4.s               ;angle = 150 deg
          fsub.s  d3,d4                         ;150 - deg
          fblt    sw_13                         ;if deg>150.jump to sw_13
          move    #0,x:$58                      ;switch1 - OFF
          move    #0.2,x:$66                    ;switch1 for DAC
          move    #8,x:$59                      ;switch2 - ON
          move    #100.2,x:$67                  ;switch2 for DAC
          move    #0,x:$60                      ;switch3 - OFF
          move    #0.1,x:$68                    ;switch3 for DAC
          jmp     end_sw
sw_13     move    #1621.3333,d4.s               ;angle = 210 deg
          fsub.s  d3,d4                         ;210 - deg
          fblt    sw_14                         ;if deg>210.jump to sw_14
          move    #0,x:$58                      ;switch1 - OFF
          move    #0.2,x:$66                    ;switch1 for DAC
          move    #8,x:$59                      ;switch2 - ON
          move    #100.2,x:$67                  ;switch2 for DAC
```

```
        move    #32,x:$60                ;switch3 - ON
        move    #100.2,x:$68             ;switch3 for DAC
        jmp     end_sw
sw_14   move    #1792.0,d4.s             ;angle = 270 deg
        fsub.s  d3,d4                    ;270 - deg
        fblt    sw_15                    ;if deg>270.jump to sw_15
        move    #0,x:$58                 ;switch1 - OFF
        move    #0.2,x:$66               ;switch1 for DAC
        move    #0,x:$59                 ;switch2 - OFF
        move    #0.2,x:$67               ;switch2 for DAC
        move    #32,x:$60                ;switch3 - ON
        move    #100.2,x:$68             ;switch3 for DAC
        jmp     end_sw
sw_15   move    #1962.6667,d4.s          ;angle = 330 deg
        fsub.s  d3,d4                    ;330 - deg
        fblt    sw_16                    ;if deg>330.jump to sw_16
        move    #2,x:$58                 ;switch1 - ON
        move    #100.2,x:$66             ;switch1 for DAC
        move    #0,x:$59                 ;switch2 - OFF
        move    #0.2,x:$67               ;switch2 for DAC
        move    #32,x:$60                ;switch3 - ON
        move    #100.2,x:$68             ;switch3 for DAC
        jmp     end_sw
sw_16   move    #2,x:$58                 ;switch1 - ON
        move    #100.2,x:$66             ;switch1 for DAC
        move    #0,x:$59                 ;switch2 - OFF
        move    #0.2,x:$67               ;switch2 for DAC
        move    #0,x:$60                 ;switch3 - OFF
        move    #0.2,x:$68               ;switch3 for DAC
        jmp     end_sw

    sw_Eq1
    sw_21   move    #1109.3333,d4.s      ;angle = 30 deg
            fsub.s  d3,d4                ;30 - deg
            fbgt    sw_26                ;if deg<30, jump to sw_26
            move    #1280.0,d4.s         ;angle = 90 deg
            fsub.s  d3,d4                ;90 - deg
            fblt    sw_22                ;if deg>90,jump to sw_22
            move    #2,x:$58             ;switch1 - ON
            move    #100.2,x:$66         ;switch1 for DAC
            move    #0,x:$59             ;switch2 - OFF
            move    #0.2,x:$67           ;switch2 for DAC
            move    #0,x:$60             ;switch3 - OFF
            move    #0.1,x:$68           ;switch3 for DAC
            jmp     end_sw
    sw_22   move    #1450.6667,d4.s      ;angle = 150 deg
            fsub.s  d3,d4                ;150 - deg
            fblt    sw_23                ;if deg>150.jump to sw_23
            move    #2,x:$58             ;switch1 - ON
            move    #100.2,x:$66         ;switch1 for DAC
            move    #8,x:$59             ;switch2 - ON
            move    #100.2,x:$67         ;switch2 for DAC
            move    #0,x:$60             ;switch3 - OFF
            move    #0.2,x:$68           ;switch3 for DAC
            jmp     end_sw
    sw_23   move    #1621.3333,d4.s      ;angle = 210 deg
            fsub.s  d3,d4                ;210 - deg
            fblt    sw_24                ;if deg>210.jump to sw_24
            move    #0,x:$58             ;switch1 - OFF
            move    #0.2,x:$66           ;switch1 for DAC
            move    #8,x:$59             ;switch2 - ON
```

```
        move    #100.2,x:$67            ;switch2 for DAC
        move    #0,x:$60                ;switch3 - OFF
        move    #0.2,x:$68              ;switch3 for DAC
        jmp     end_sw
sw_24   move    #1792.0,d4.s            ;angle = 270 deg
        fsub.s  d3,d4                   ;270 - deg
        fblt    sw_25                   ;if deg>270.jump to sw_25
        move    #0,x:$58                ;switch1 - OFF
        move    #0.2,x:$66              ;switch1 for DAC
        move    #8,x:$59                ;switch2 - ON
        move    #100.2,x:$67            ;switch2 for DAC
        move    #32,x:$60               ;switch3 - ON
        move    #100.2,x:$68            ;switch3 for DAC
        jmp     end_sw
sw_25   move    #1962.6667,d4.s         ;angle = 330 deg
        fsub.s  d3,d4                   ;330 - deg
        fblt    sw_26                   ;if deg>330.jump to sw_26
        move    #0,x:$58                ;switch1 - OFF
        move    #0.2,x:$66              ;switch1 for DAC
        move    #0,x:$59                ;switch2 - OFF
        move    #0.2,x:$67              ;switch2 for DAC
        move    #32,x:$60               ;switch3 - ON
        move    #100.2,x:$68            ;switch3 for DAC
        jmp     end_sw
sw_26   move    #2,x:$58                ;switch1 - ON
        move    #100.2,x:$66            ;switch1 for DAC
        move    #0,x:$59                ;switch2 - OFF
        move    #0.2,x:$67              ;switch2 for DAC
        move    #32,x:$60               ;switch3 - ON
        move    #100.2,x:$68            ;switch3 for DAC
        jmp     end_sw

sw_Ed
        move    x:$89,d1.s              ;Eq
        move    #0.0,d2.s
        fsub.s  d2,d1                   ;Eq - 0
        fblt    sw_Eq2                  ;jump if Eq<0
        move    x:$7B,d3.s              ;pointer for sin/cosine table
sw_31   move    #1109.3333,d4.s         ;angle = 30 deg
        fsub.s  d3,d4                   ;30 - deg
        fbgt    sw_36                   ;if deg<30, jump to sw_31
        move    #1280.0,d4.s            ;angle = 90 deg
        fsub.s  d3,d4                   ;90 - deg
        fblt    sw_32                   ;if deg>90,jump to sw_32
        move    #0,x:$58                ;switch1 - OFF
        move    #0.2,x:$66              ;switch1 for DAC
        move    #8,x:$59                ;switch2 - ON
        move    #100.2,x:$67            ;switch2 for DAC
        move    #0,x:$60                ;switch3 - OFF
        move    #0.1,x:$68              ;switch3 for DAC
        jmp     end_sw
sw_32   move    #1450.6667,d4.s         ;angle = 150 deg
        fsub.s  d3,d4                   ;150 - deg
        fblt    sw_33                   ;if deg>150.jump to sw_33
        move    #0,x:$58                ;switch1 - OFF
        move    #0.2,x:$66              ;switch1 for DAC
        move    #8,x:$59                ;switch2 - ON
        move    #100.2,x:$67            ;switch2 for DAC
        move    #32,x:$60               ;switch3 - ON
        move    #100.1,x:$68            ;switch3 for DAC
        jmp     end_sw
```

```
sw_33   move    #1621.3333,d4.s         ;angle = 210 deg
        fsub.s  d3,d4                   ;210 - deg
        fblt    sw_34                   ;if deg>210.jump to sw_34
        move    #0,x:$58                ;switch1 - OFF
        move    #0.2,x:$66              ;switch1 for DAC
        move    #0,x:$59                ;switch2 - OFF
        move    #0.2,x:$67              ;switch2 for DAC
        move    #32,x:$60               ;switch3 - ON
        move    #100.2,x:$68            ;switch3 for DAC
        jmp     end_sw
sw_34   move    #1792.0,d4.s            ;angle = 270 deg
        fsub.s  d3,d4                   ;270 - deg
        fblt    sw_35                   ;if deg>270.jump to sw_35
        move    #2,x:$58                ;switch1 - ON
        move    #100.2,x:$66            ;switch1 for DAC
        move    #0,x:$59                ;switch2 - OFF
        move    #0.2,x:$67              ;switch2 for DAC
        move    #32,x:$60               ;switch3 - ON
        move    #100.2,x:$68            ;switch3 for DAC
        jmp     end_sw
sw_35   move    #1962.6667,d4.s         ;angle = 330 deg
        fsub.s  d3,d4                   ;330 - deg
        fblt    sw_36                   ;if deg>330.jump to sw_36
        move    #2,x:$58                ;switch1 - ON
        move    #100.2,x:$66            ;switch1 for DAC
        move    #0,x:$59                ;switch2 - OFF
        move    #0.2,x:$67              ;switch2 for DAC
        move    #0,x:$60                ;switch3 - OFF
        move    #0.2,x:$68              ;switch3 for DAC
        jmp     end_sw
sw_36   move    #2,x:$58                ;switch1 - ON
        move    #100.2,x:$66            ;switch1 for DAC
        move    #8,x:$59                ;switch2 - ON
        move    #100.2,x:$67            ;switch2 for DAC
        move    #0,x:$60                ;switch3 - OFF
        move    #0.2,x:$68              ;switch3 for DAC
        jmp     end_sw

sw_Eq2
sw_41   move    #1109.3333,d4.s         ;angle = 30 deg
        fsub.s  d3,d4                   ;30 - deg
        fbgt    sw_46                   ;if deg<30, jump to sw_46
        move    #1280.0,d4.s            ;angle = 90 deg
        fsub.s  d3,d4                   ;90 - deg
        fblt    sw_42                   ;if deg>90,jump to sw_42
        move    #0,x:$58                ;switch1 - OFF
        move    #0.2,x:$66              ;switch1 for DAC
        move    #0,x:$59                ;switch2 - OFF
        move    #0.2,x:$67              ;switch2 for DAC
        move    #32,x:$60               ;switch3 - ON
        move    #100.1,x:$68            ;switch3 for DAC
        jmp     end_sw
sw_42   move    #1450.6667,d4.s         ;angle = 150 deg
        fsub.s  d3,d4                   ;150 - deg
        fblt    sw_43                   ;if deg>150.jump to sw_43
        move    #2,x:$58                ;switch1 - ON
        move    #100.2,x:$66            ;switch1 for DAC
        move    #0,x:$59                ;switch2 - OFF
        move    #0.2,x:$67              ;switch2 for DAC
        move    #32,x:$60               ;switch3 - ON
        move    #100.2,x:$68            ;switch3 for DAC
```

```
        jmp     end_sw
sw_43   move    #1621.3333,d4.s          ;angle = 210 deg
        fsub.s  d3,d4                    ;210 - deg
        fblt    sw_44                    ;if deg>210.jump to sw_44
        move    #2,x:$58                 ;switch1 - ON
        move    #100.2,x:$66             ;switch1 for DAC
        move    #0,x:$59                 ;switch2 - OFF
        move    #0.2,x:$67               ;switch2 for DAC
        move    #0,x:$60                 ;switch3 - OFF
        move    #0.2,x:$68               ;switch3 for DAC
        jmp     end_sw
sw_44   move    #1792.0,d4.s             ;angle = 270 deg
        fsub.s  d3,d4                    ;270 - deg
        fblt    sw_45                    ;if deg>270.jump to sw_45
        move    #2,x:$58                 ;switch1 - ON
        move    #100.2,x:$66             ;switch1 for DAC
        move    #8,x:$59                 ;switch2 - ON
        move    #100.2,x:$67             ;switch2 for DAC
        move    #0,x:$60                 ;switch3 - OFF
        move    #0.2,x:$68               ;switch3 for DAC
        jmp     end_sw
sw_45   move    #1962.6667,d4.s          ;angle = 330 deg
        fsub.s  d3,d4                    ;330 - deg
        fblt    sw_46                    ;if deg>330.jump to sw_46
        move    #0,x:$58                 ;switch1 - OFF
        move    #0.2,x:$66               ;switch1 for DAC
        move    #8,x:$59                 ;switch2 - ON
        move    #100.2,x:$67             ;switch2 for DAC
        move    #0,x:$60                 ;switch3 - OFF
        move    #0.2,x:$68               ;switch3 for DAC
        jmp     end_sw
sw_46   move    #0,x:$58                 ;switch1 - OFF
        move    #0.2,x:$66               ;switch1 for DAC
        move    #8,x:$59                 ;switch2 - ON
        move    #100.2,x:$67             ;switch2 for DAC
        move    #32,x:$60                ;switch3 - ON
        move    #100.2,x:$68             ;switch3 for DAC

end_sw

;************ for equal signal

        move    x:$55,d1.l               ;previous state of switch1
        move    x:$58,d2.l               ;current state of switch1
        fcmp    d2,d1                    ;compare both
        fbne    eq_1                     ;if not equal jump
        move    #0,x:$69                 ;equal signal is zero
eq_1
        move    #1,x:$69                 ;equal signal is 1
        move    d2.l,x:$55               ;move current to previous

        move    x:$56,d1.l               ;previous state of switch2
        move    x:$59,d2.l               ;current state of switch2
        fcmp    d2,d1                    ;compare both
        fbne    eq_2                     ;if not equal jump
        move    #0,x:$6A                 ;equal signal is zero
eq_2
        move    #4,x:$6A                 ;equal signal is 1
        move    d2.l,x:$56               ;move current to previous

        move    x:$57,d1.l               ;previous state of switch3
```

```
        move    x:$60,d2.l              ;current state of switch3
        fcmp    d2,d1                   ;compare both
        fbne    eq_3                    ;if not equal jump
        move    #0,x:$6B                ;equal signal is zero
eq_3
        move    #16,x:$6B               ;equal signal is 1
        move    d2.l,x:$57              ;move current to previous

        move    x:$56,d1.l              ;switch1
        move    x:$57,d2.l              ;switch2
        add     d2,d1
        move    x:$58,d2.l               ;switch3
        add     d2,d1
        move    d1.l,x:$6C              ;pwm_only
        move    x:$69,d2.l              ;equal1
        add     d2,d1
        move    x:$6A,d2.l              ;equal2
        add     d2,d1
        move    x:$6B,d2.l               ;equal3
        add     d2,d1
        move    d1.l,x:$6D              ;pwm and equal

;*****************************************************************
;*                  create pwm and equal signals for underlap     *
;*****************************************************************
        move    #out_clr,d3.l
        move    #out_r,d7.l
        move    x:$6D,d6.l
        add     d7.l,d6.l
        movep   d6.l,y:outport1         ;o/p pwm and equal

        do      #4,end_delay            ;no of count determine
        move    x:$6C,d6.l              ;duration for underlap
        add     d7.l,d6.l               ;(this is delay for
        nop                             ; equal signal )
end_delay
        movep   d6.l,y:outport1         ;o/p pwm only

        movep   d3.l,y:outport1         ;clear

;*****************************************************************
;*                  output results to DAC                        *
;*****************************************************************

        move    #$2C00,d1.l             ;first DAC output (use Gray code)
        move    #0.05,d3.s              ;reciprocal of current multiplier
        move    x:$92,d2.s              ;restore Iacomp
        fmpy.s  d2,d3,d2                ;value for DAC
        int     d2                      ;integerized
        move    #128,d5.l               ;scale for bipolar operation
        add     d5,d2
        move    #$1000,d5.l             ;set dsprd all the time
        add     d5,d2
        movep   d1.l,y:outport1         ;select DAC output
        movep   d2.l,y:outport2         ;output Iacomp

        move    #$6C00,d1.l             ;2nd DAC output (use Gray code)
        move    x:$91,d2.s              ;restore Ibcomp
        fmpy.s  d2,d3,d2                ;value for DAC
        int     d2                      ;integerized
        move    #128,d5.l               ;scale for bipolar operation
```

```
add     d5,d2
move    #$1000,d5.l                 ;set dsprd all the time
add     d5,d2
movep   d1.l,y:outport1             ;select DAC output
movep   d2.l,y:outport2             ;output Ibcomp

move    #$4C00,d1.l                 ;3rd DAC output (use Gray code)
move    x:$43,d2.s                  ;restore Iccomp
move    #o_currco,d3.s
fmpy.s  d2,d3,d2                    ;value for DAC
int     d2                          ;integerized
move    #128,d5.l                    ;scale for bipolar operation
add     d5,d2
move    #$1000,d5.l                 ;set dsprd all the time
add     d5,d2
movep   d1.l,y:outport1             ;select DAC output
movep   d2.l,y:outport2             ;output Iccomp

move    #$CC00,d1.l                 ;4th DAC output (use Gray code)
move    x:$18,d2.s                  ;restore Iccomp
fmpy.s  d2,d3,d2                    ;value for DAC
int     d2                          ;integerized
move    #128,d5.l                   ;scale for bipolar operation
add     d5,d2
move    #$1000,d5.l                 ;set dsprd all the time
add     d5,d2
movep   d1.l,y:outport1             ;select DAC output
movep   d2.l,y:outport2             ;output Ibcomp

move    #$8C00,d1.l                 ;5th DAC output (use Gray code)
move    #o_currco,d3.s
move    x:$13,d2.s                  ;restore Ibcomp
fmpy.s  d2,d3,d2                    ;value for DAC
int     d2                          ;integerized
move    #128,d5.l                   ;scale for bipolar operation
add     d5,d2
move    #$1000,d5.l                 ;set dsprd all the time
add     d5,d2
movep   d1.l,y:outport1             ;select DAC output
movep   d2.l,y:outport2             ;output Ibcomp


movep   #$0C00,y:outport1           ;lacth DAc selector

rti
```

APPENDIX F : LIST OF FIGURES AND TABLES

F.1 LIST OF FIGURES

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

F.2 LIST OF TABLES