A SERIES-PARALLEL LOAD-RESONANT CONVERTER FOR A CONTROLLED-CURRENT ARC-WELDING POWER SUPPLY

Helen Pollock

Thesis submitted for the examination of degree of Doctor of Philosophy

Department of Engineering
University of Warwick
Coventry CV4 7AL

March 1996

To Jenny, Anne and Richard

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

APPENDIX A

APPENDIX D

GLOSSARY OF SYMBOLS

A	current gain
a_1, a_2, a_3	coefficients of polynomial in real part of impedance equation
b_1, b_2, b_3, b_4	coefficients of polynomial in imaginary part of impedance equation
c_1, c_2, c_3	coefficients of polynomial in denominator of impedance equation
C_{iss}	input capacitance
C_L	capacitance in load-leg of circuit
C_P	capacitance in parallel-leg of circuit
C_S	capacitance in series-leg of circuit
D	duty cycle
D_2	fraction of time when current flows in inductor
d_0, d_1	expressions governing critical value of C_S
E	input voltage
I_{LOAD}	current in load-leg of circuit
I_{max}	maximum inductor current
I_{min}	minimum inductor current
I_S	current in series-leg of circuit
i_L	instantaneous inductor current
k_a, k_b, k_c	normalised polynomial coefficients
L	inductance
$L_{boundary}$	inductance value when converter operation is only just continuous
L_P	inductance in parallel-leg of circuit
L_L	inductance in load-leg of circuit
L_S	inductance in series-leg of circuit
P_{av}	average power
P_{HIGH}	high power
P_{LOW}	low power
R	apparent load resistance
R_{ds} on	on-state resistance of power device
R_L	resistance of load referred to primary-side of isolation transformer
R_{LOAD}	resistance of load

R_{TOT}	resistance of circuit
S, S_1, S_2	switches
t	time
t_{on}	on-time
t_{off}	off-time
T_S	repetition period
$V_{ce}(sat)$	Collector-emitter voltage when power device is saturated
V_L	inductor voltage
V_O	output voltage
V_{Srms}	r.m.s. value of the fundamental component of a square-wave voltage
X_L	reactance of load-leg
X_P	reactance of parallel-leg
X_S	reactance of series-leg
Z_{TOT}	impedance of circuit
ω	angular frequency
ω_0	angular primary resonant frequency
ω_1	angular second resonant frequency
ω_2	angular third resonant frequency

ACKNOWLEDGEMENTS

Thanks are due to Telcon Ltd., Semikron Ltd., Industrial Capacitors (Wrexham) Ltd., Unitrode Ltd. and B.H.C. Aerovox Ltd. for the supply of sample components, to Professor John Flower for his supervision and support, to my colleagues in the Power Electronics and Drives Laboratory for their help in times of need, and most of all to my husband and family for their continuous support and encouragement.

ABSTRACT

A power supply incorporating a series-parallel load-resonant converter, capable of very efficient operation over a wide range of output power is presented. The series-parallel load-resonant converter is shown to have three pairs of resonant frequencies. Operation of the circuit at each of these resonant frequencies maintains zero current switching and high frequency operation. Design mathematics is developed which allow series-parallel load-resonant converters to be designed with specific resonant frequencies and circuit resistances. A new method of power control for series-parallel load-resonant converters is presented; the power delivered to the circuit and hence the load is shown to var substantially depending on which resonant frequency the circuit is excited at. Two series-parallel load-resonant converters are designed simulated, constructed and tested. There is good agreement between the simulation and experimental results. One of the circuits produces an output current of 200 A while the second demonstrates the new power control technique pulsing between 55 A and 145 A while running at frequencies of 63 kHz and 100 kHz. The new power supply is particularly suited to arc-welding. It contains an active rectifier and draws near unity power factor.

CHAPTER 1 INTRODUCTION

This thesis will describe the analysis and design of series-parallel load-resonant converters for a controlled current arc-welding power supply with a new method of power control.

1.1 The need for welding

Welding is the most efficient way to permanently join two or more pieces of metal. It is the only way to join two pieces of metal so that they act as one piece. Welding occurs by heating the metals to a temperature at which they coalesce with or without the application of pressure. The weld is often completed by the application of a filler metal under these conditions. The filler metal has properties which match the properties of the base metal being welded.

Welding is the lowest cost joining method and makes better use of materials than other joining methods such as bolting or riveting. It can join all commercial metals, can be used almost anywhere and allows flexibility in design.

1.2 Historical Development of welding

The process of welding can be traced back to the Bronze Age when small gold boxes were made, through to the Iron Age when iron pieces were joined together by welding. During the Middle Ages, the art of blacksmithing was developed and many articles made from iron were welded by hammering. Sometime during the period 1877-1903 Sir Humphrey Davy produced an arc between two carbon electrodes using a battery. This was the beginning of arc-welding and resistance welding. Further developments of these methods produced practical joining processes. In 1890 the first American patent for the arc-welding process using a metal electrode was awarded. Work then followed on improving the electrode composition. In 1885 the first set of patents for resistance welding were granted. The First World War produced a tremendous demand for welded products and many other techniques, specific to

particular applications, arose.

Welding today is now the universally accepted method of permanently joining metals. It is a growing industry on a world-wide basis. Conventional electric arc-welding equipment and filler metals make up two thirds of the welding industry sales. It is projected that the arc-welding process will continue to grow. Trends in the industry will be to increase automation and improve the quality of welds. The main restriction affecting these trends are the welding power supplies on which the electric arc-welding processes rely.

The voltage supplied by the utility for industrial purposes is too high to use directly in arc-welding. A power supply is therefore required to reduce the utility voltage to a suitable range. Either a transformer, solid-state inverter or a motor-generator can be used to achieve this. It is the use of solid-state inverters, in arc-welding power supplies, that has produced a dramatic change in the industry. Their controllability and portability has enabled automation leading to a further leap in the quality of welds produced. It is to this area that this particular project is aimed. The work described in this thesis advances the use of solid-state inverters in arc-welding power supplies by the use of load-resonant converter topologies.

The size of the magnetic components present in inverter supplies is essentially determined by the frequency of operation of the power supply. The higher the frequency of operation the smaller these components. A welding power supply with a high frequency of operation is more compact and will respond more quickly to current fluctuations owing to the smaller values of inductance present in the circuit. Load-resonant converters can be used to implement such power supplies.

1.3 The proposed welding power supply

The work described in this thesis is the analysis, design and construction of a series-parallel load-resonant converter in a pulsed arc-welding power supply. Chapter 2 describes the arc welding process and the design imperatives of a welding power

supply. Chapter 3 discusses conventional hard switched power conversion and the use of semiconductor devices with lower switching losses to achieve high frequency operation. Chapter 4 introduces soft-switching load-resonant converters and known methods of power control in these converters. A simple series load-resonant converter is constructed and the dead-time control method for producing pulsed current evaluated. Chapter 5 presents a frequency domain analysis of the series-parallel load-resonant converter with six energy storage elements. This identifies the multiplicity of resonant frequencies of the converter. A design procedure for the resonant converter components is given allowing the location of the resonant frequencies of the circuit to be specified. Two series-parallel load-resonant converters are designed and simulated in Chapter 6 and a novel control method to produce pulsed current is described. Chapter 7 presents the results of these converters in operation and compares the results with the circuit simulations. Chapter 8 concludes the thesis by highlighting the major advantages of the proposed pulsed arc-welding power supply over more conventional power supply technology, and goes on to suggest further possible work in this area.

1.4 Author's Principal Contributions

The principal contributions of the author, in this thesis, include the testing of the Static Induction Transistor involving very advanced circuit design techniques, particularly in the gate drive circuit and power component layout. The switching test results obtained from this device demonstrate its potential and still remain as some of the fastest power switching waveforms to be published.

The work on load-resonant converters includes a detailed analysis of the series-parallel load-resonant converter in the frequency domain with six energy storage elements and identifies some features of these circuits for the first time. The unique design procedure allows the component values to be calculated by specifying the position of the resonant frequencies of the circuit. The analysis is verified by simulation and testing at current levels up to 200 A. Two different series-parallel load-resonant converters are simulated and tested with good agreement in the results. The

continuous operation of this circuit topology at full rated voltage under open-circuit and short-circuit conditions is verified. The resonant nature of the circuit makes it compatible with E.M.C regulations.

A completely new control method for pulsing between substantially different power levels is developed. This will have significant impact on applications demanding this facility, since operation at very different power levels can be achieved while maintaining zero-current switching in the converter. The success of this technique relies on the design mathematics for a six energy storage element resonant circuit. This allows series-parallel load-resonant converters to be designed with specific resonant frequencies and circuit resistances.

Various parts of the above work have been described at an IEE Colloquium [29] and an IEE Conference [54] and two papers to the IEE and IEEE have undergone final acceptance [55],[56].

CHAPTER 2 WELDING PROCESSES AND WELDING POWER SUPPLIES

2.1 Introduction

There are approximately 50 distinct welding processes. These are sub-divided into seven groups; arc-welding, brazing, solid-state welding, soldering, resistance welding, oxyfuel-gas welding and other welding techniques such as laser welding. The details of the welding processes in each of these groups are summarised in Appendix A.

Arc-welding and resistance welding processes both require electrical power sources. In this project the main welding processes under consideration were the arc-welding processes. Good innovative power supply design for these processes was likely to have the greatest potential benefit on welding performance. To evaluate existing arc-welding power supply technology effectively a knowledge of the arc-welding process was first established.

2.2 Arc-welding processes

In the arc-welding process the arc is a sustained high current, low voltage electrical discharge through a high conducting plasma that produces sufficient thermal energy to join metals by fusion. Arc-welding processes can be classified according to whether a consumable or non-consumable electrode is used in the welding process.

2.2.1 Non-consumable arc-welding processes

The non-consumable arc-welding process uses an electrode that does not melt in the arc. Tungsten is usually used because it is a good emitter of electrons and it has a high melting point. The voltage across the arc is proportional to the length of the arc gap [1]. The arc is made out of two concentric zones, an inner core which is a plasma and an outer flame. The plasma carries most of the current and has the highest temperature. The outer zone of the arc is much cooler and tends to keep the plasma in the centre. Figure 2.1 shows how the arc is formed under two different sets of conditions. In Figure 2.1(a), the small tungsten electrode acts as the cathode and the workpiece as the anode. When the arc is struck the electrode becomes hot and

emits electrons. The emitted electrons are attracted to the positive workpiece and while travelling through the arc gap raise the temperature of the shielding gas atoms by colliding with them. The positively charged gaseous atoms are attracted to the negative tungsten electrode where their kinetic energy is dissipated as heat. This keeps the tungsten electrode hot enough for electron emission by the process of thermionic emission. Although positive ions travel from the workpiece to the electrode the current flow is dominated by electrons. The bombardment of the workpiece by electrons of high mobility produces a deep and narrow weld.

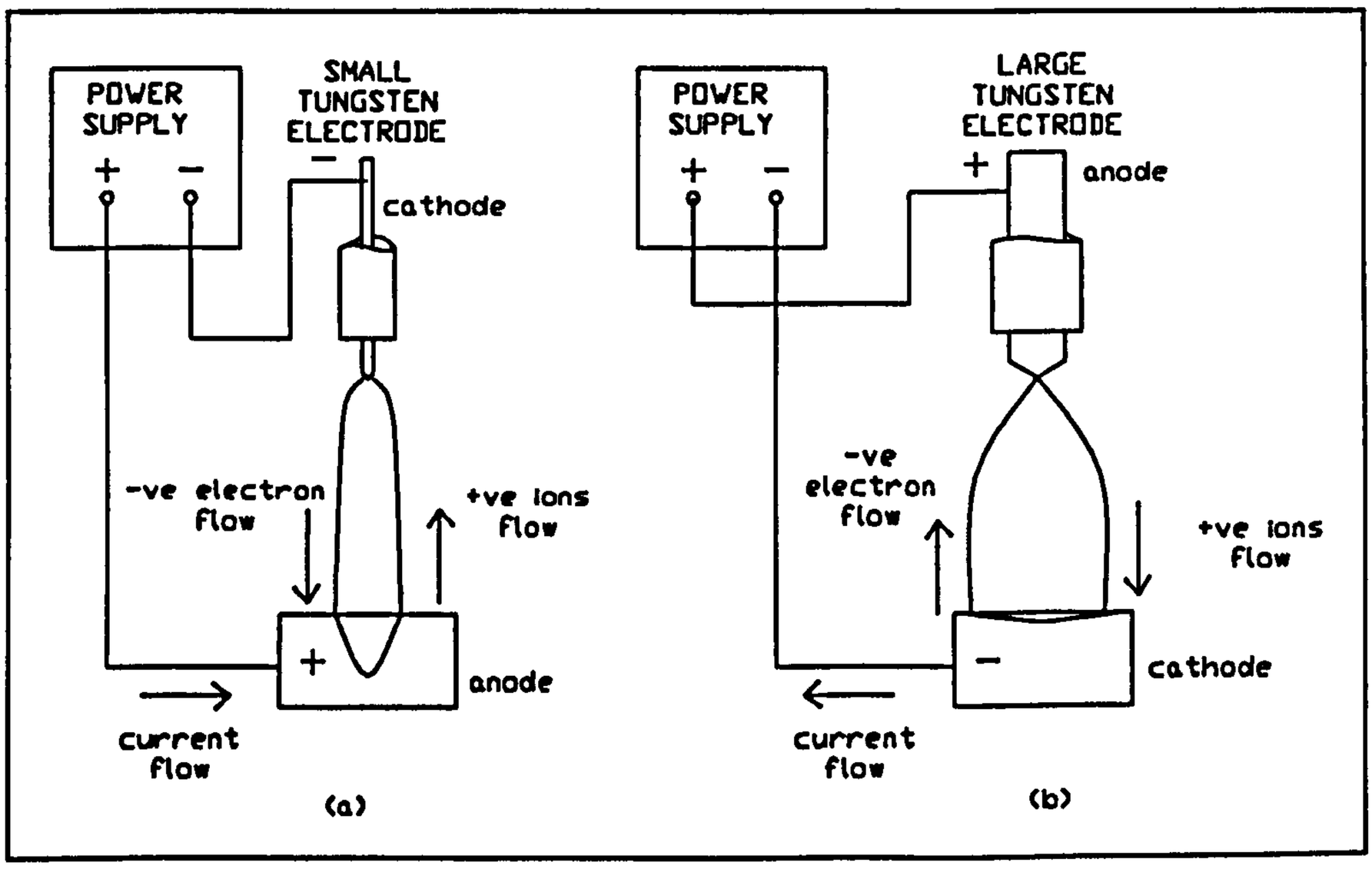


Figure 2.1 Welding with (a) a negative electrode, (b) a positive electrode

In Figure 2.1(b) the tungsten electrode is positive and the workpiece is negative. The electrons flow from the workpiece to the electrode where they create intense heat. A large electrode is required for this process to avoid overheating and the current must be limited. Positive ion bombardment occurs at the workpiece producing a shallower and wider weld. This also cleans the workpiece. Figure 2.2 shows the relationship between the voltage and the different regions of the arc. A voltage drop known as the cathode drop occurs at the connection between the arc column and the negative electrode. There is also a large temperature drop at this point. The anode drop is the electrical connection between the anode and the arc column. In the central region, between the electrode and the workpiece, a circular magnetic field surrounds the arc. This field, produced by the current flow, tends to constrict the plasma. The constriction causes high pressures in the arc plasma and extremely high velocities resulting in a plasma jet at sonic speed.

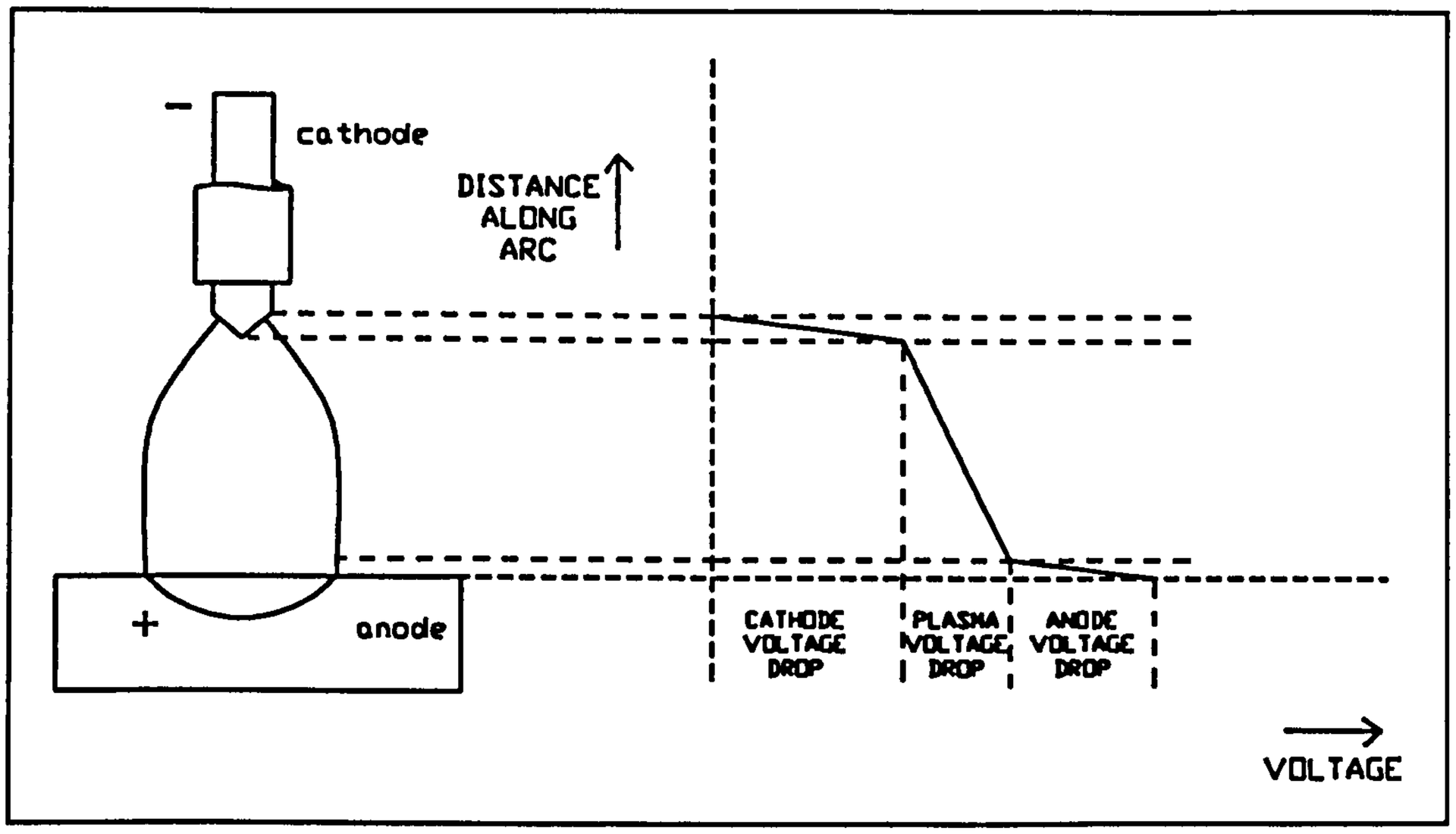


Figure 2.2 Voltage across the welding arc with a non-consumable electrode

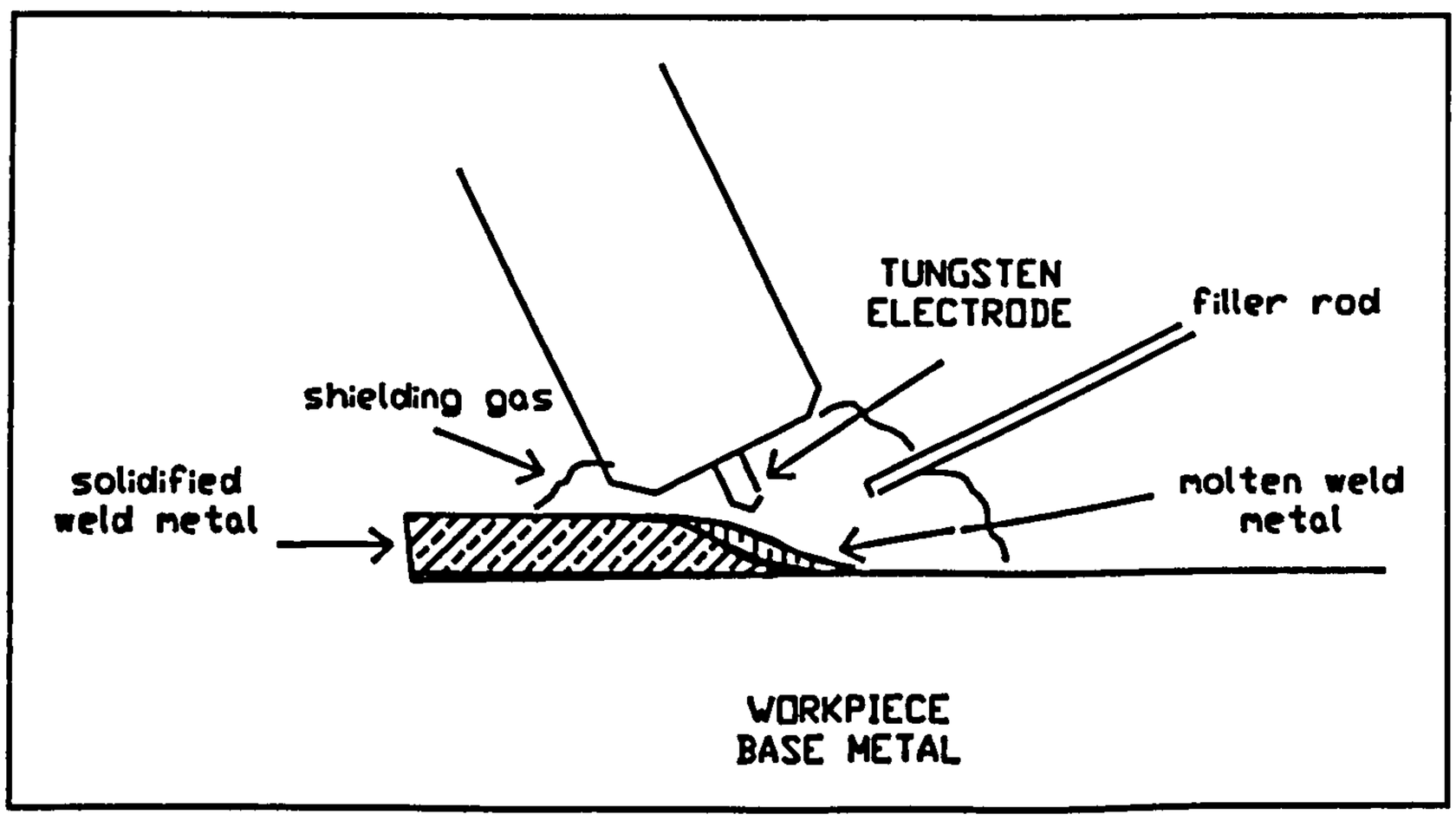


Figure 2.3 Process of TIG welding

The non-consumable welding arc processes were listed in Appendix A. The most common are gas tungsten arc-welding (GTAW or TIG welding) and plasma arc-welding. The process of TIG welding is shown in Figure 2.3. A shielding gas of argon or helium is used and this prevents oxygen and nitrogen in the air coming into contact with the molten metal or the hot tungsten electrode. This process makes high quality welds in almost all metals and alloys.

A variation of TIG welding is pulsed-current TIG welding. In this process the

magnitude of the welding current continuously changes between two levels. During the periods of high pulsed current, heating and fusion take place at the work piece and during the low current periods cooling and solidification occurs. Pulsed TIG welding has a lower heat input and as a result reduces distortion and warpage when welding thinner materials.

2.2.2 Arc-welding with a consumable electrode

In the arc-welding process, using a consumable electrode, the electrode melts and the resultant molten metal is carried across the arc gap. A uniform arc length is maintained between the melting end of the electrode and the weld pool by feeding the electrode into the arc as fast as it melts. The arc is formed of a plasma of electrically and thermally excited gas atoms. The potential gradient along the axis of the arc is not uniform, see Figure 2.4.

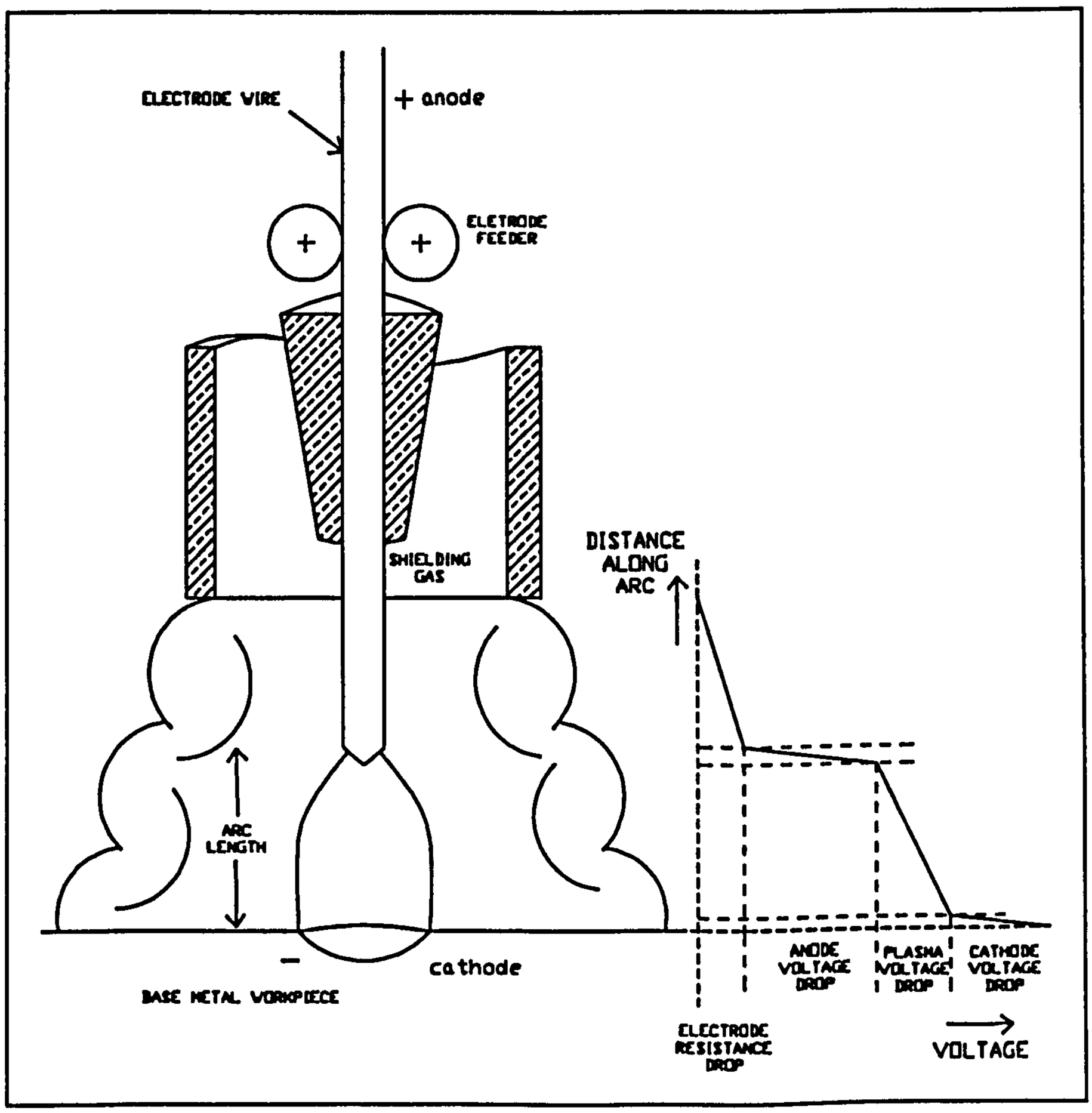


Figure 2.4 Potential gradient along the axis of an arc during consumable electrode welding

There are four different potential gradients, the largest being across the anode and cathode. The resistance drop is caused by the resistance of the exposed electrode to the current and is dependent on the material and dimensions of the electrode. The remaining voltage drop is directly proportional to the arc length but varies little with current.

Like the non-consumable arc-welding process there are two possible circuit arrangements that can be employed i.e. one with the workpiece acting as a cathode, and the other where the workpiece acts as the anode. Again this causes two different temperature distributions in the workpiece. Metal from the electrode transferring across the arc is subjected to surface tension, gravity, electromagnetic forces and the plasma jet. The electromagnetic force is proportional to the square of the current.

The mode of metal transfer across the arc is determined by the welding process, the metal being welded, the shielding gas, the electrode, the characteristics of the power source, the welding current, the current density and the heat input. There are four main types of transfer.

(i) Spray transfer - this is a very smooth mode of transfer which occurs at the fastest wire speed. The droplets crossing the arc are smaller in diameter than the electrode. As the current increases the drop size decreases and their formation frequency increases. Electromagnetic forces are dominant in this type of transfer due to the high current density. This limits the size of the molten metal droplet. At high currents the line of drops begins to rotate rapidly about the axis of the electrode. Further increase in current causes the diameter of rotation to increase and spatter occurs.

(ii) Globular transfer - this occurs at lower wire speeds and currents (below 200 A in carbon steel) and it is characterised by a drop size greater than that of the electrode. The globule grows on the tip of the electrode and it is then transferred by gravity and electromagnetic forces across the arc. It follows an irregular path and then splashes into the weld-pool normally producing a lot more spatter than spray transfer.

(iii) Short-circuiting transfer - This is a low-energy mode of metal transfer. The electrode is fed at a high rate such that the molten tip periodically comes into contact

with the molten weld pool. This creates a short circuit extinguishing the arc. Molten metal from the electrode is transferred to the weld-pool by surface tension. The electrode then separates from the weld-pool and the arc is re-established.

(iv) Pulsed-spray metal transfer - This transfers droplets of molten metal across the arc at a fixed frequency set by the power source. The pulsed-spray transfer method produces droplets approximately equal to the diameter of the electrode. One drop of metal is carried across the arc for each pulse of current. Pulsed-spray metal transfer achieves spray transfer at a lower average current. This means that the average heat input is lower than normal spray transfer and this results in a smaller weld-pool and reduced spatter occurring.

Appendix A lists the main types of consumable electrode welding. In gas metal arc-welding (MIG welding), any of the above modes of metal transfer can occur. They are summarised in Table 2.1.

Metal transfer	**(i)Spray**	**(ii)Globular**	**(iii)Short Circuiting**	**(iv)Pulsed spray**
Shielding gas	Argon + oxygen	CO_2	CO_2 or CO_2 + argon	Argon + oxygen
Metals to be welded	Low carbon and medium carbon steels	Low carbon and medium carbon steels	Low carbon and medium carbon steels	Al, nickel, steels, nickel alloys
Metal thickness	0.25" - unlimited thickness	0.14"- 0.5"	0.038"- 0.25"	Thin to unlimited
Welding positions	All positions with small electrode wire	Flat and horizontal	All positions	All positions
Major advantages	Smooth surface; deep penetration; high travel speed	Low cost gas; high speed travel; deep penetration; high deposition	Thin material will bridge gaps and minimum cleanup	Uses larger electrode
Limitations	Position and size of minimum thickness	Spatter removal sometimes required, high heat	Uneconomical in heavy thickness	Special power source
Appearance of weld	Smooth surface-minimum spatter	Smooth, some spatter	Smooth surface-minor spatter	Smooth surface minimum spatter
Travel speeds	Up to 150"/min	Up to 250"/min	Up to 50"/min	Up to 100"/min

Table 2.1 Methods of metal transfer in MIG welding

2.2.3 Pulsed-arc-welding

Both consumable and non-consumable arc-welding processes can be carried out in pulsed mode. Pulsed welding is classified as follows

(i) Low-frequency pulse welding, 0.5-5 Hz,

(ii) Intermediate-frequency pulse welding, 10-500 Hz,

(iii) High-frequency pulse welding, 1-20 kHz.

Table 2.2 below summarises the characteristics of the two main types and compares them to constant d.c.excitation.

Type of arc		Arc condition	Bead Appearance	Advantages
DC		Soft arc	Smooth bead	Thin and thick plates with stable arc
Low-frequency pulse (0.5 - 5Hz)		Large arc volume	Uniform bead	All position welding Welding different plate thicknesses Reverse polarity welding
Intermediate-pulse (50 - 500 Hz)		Arc concentrate Noisy	Fine grained uniform bead	High speed welding of thin plate

Table 2.2 Characteristics of pulsed welding

Pulsed welding uses a droplet of filler metal per current pulse. The peak current level and duration of current peak time are set to allow one droplet to be detached from the continuous wire electrode and to be transferred across the arc into the molten pool. The current background level is set just high enough to hold a stable arc but low enough not to transfer any further material. The repeat frequency of current pulses is adjusted to give the correct electrode burn-off rate in relation to wire-feed rate, to obtain constant arc-length.

Spatter from the pulsed welding process is dependent on the short-circuit current rise-time and the fall-time of current to a background level. To achieve low spatter at high speed, a square-wave current pulse is used. Most spatter occurs from short-circuits occurring during the pulse peak and transition periods. Therefore using a square current pulse instead of a trapezoidal wave reduces spatter.

2.3 Requirements of an arc-welding power supply

Figure 2.5 shows a block diagram of a welding power supply. The power supply is supplied from the a.c. mains. The power supply contains an isolation transformer, a rectifier, a smoothing inductor and the welding arc [2,3]. An isolation transformer is present because the workpiece forms part of the electrical circuit and therefore must be isolated from the power supply for safety reasons. A smoothing inductance is present to smooth any current ripple on the supply current to the welding arc. The size of both of these magnetic components depends on the frequency of operation of the circuit. Operation at frequencies of several kHz significantly reduces the size of the transformer (over its 50 Hz counterparts) and furthermore increases the speed at which the circuit can respond to current and voltage fluctuations.

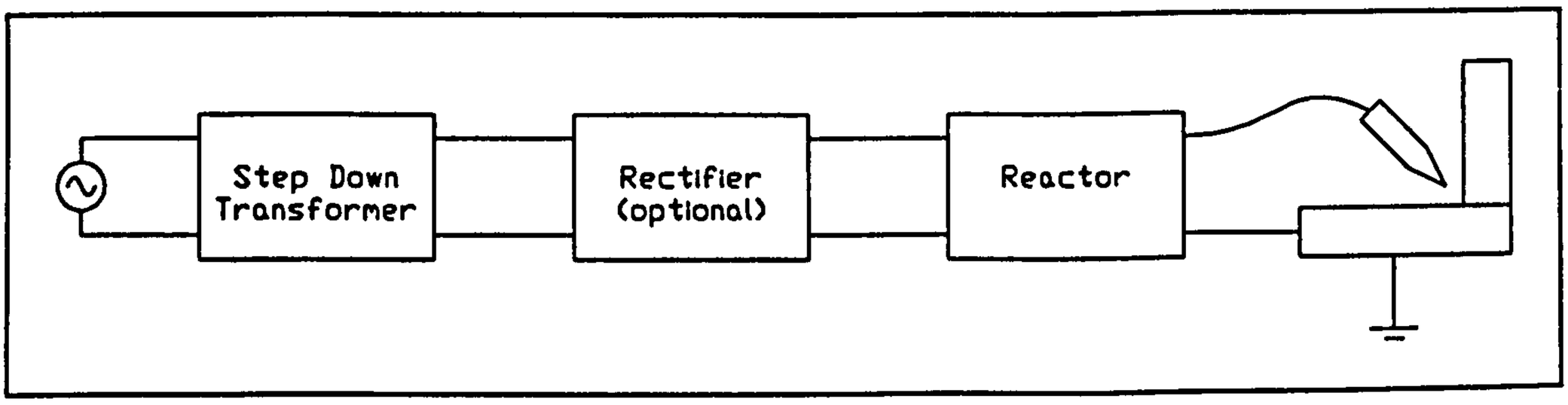


Figure 2.5 Block diagram of welding power supply

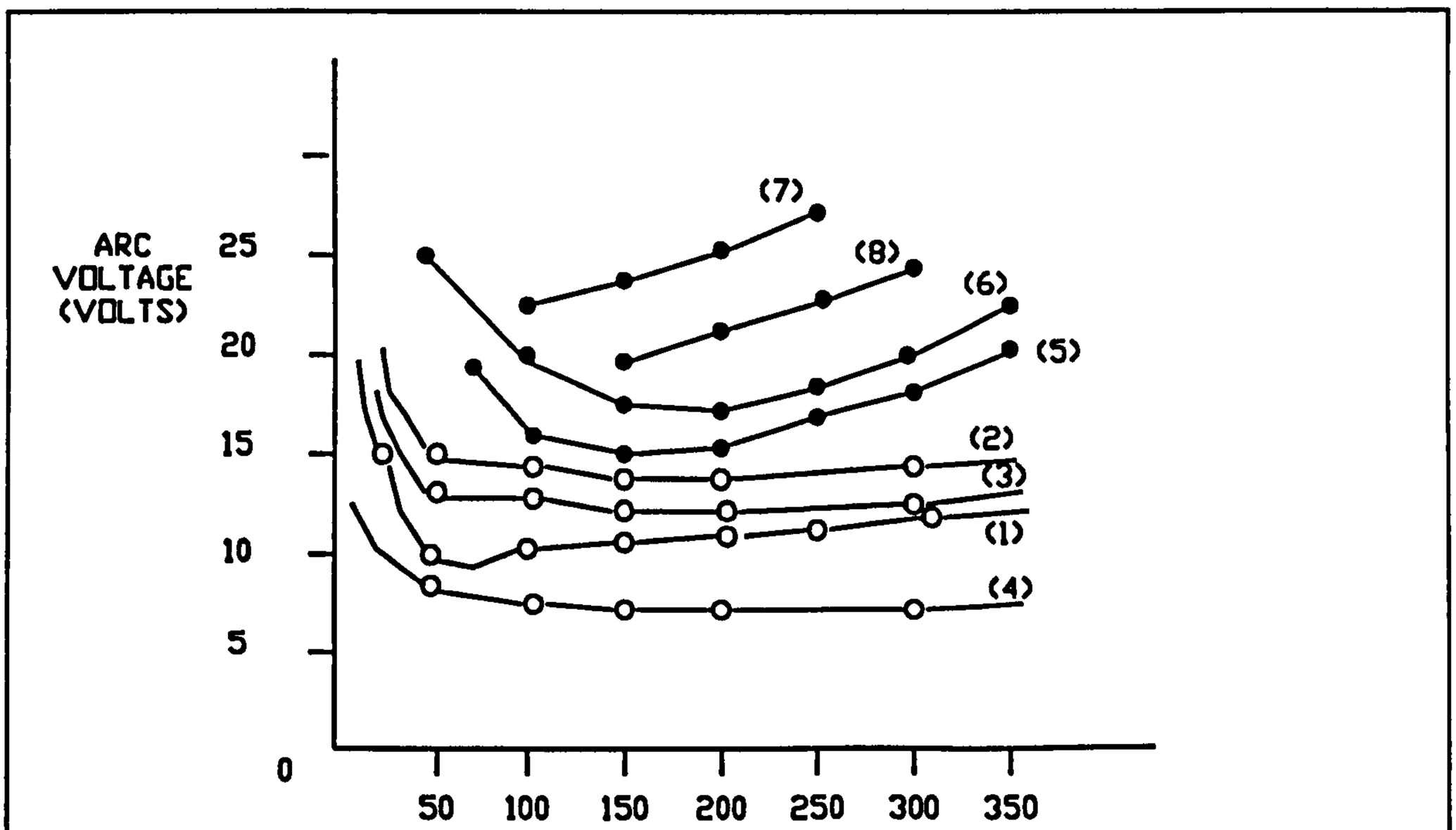
There are several different conditions which arise during the welding process.

2.3.1 Starting characteristics

Ideal arc starting involves a rapid initial current rise to establish the arc quickly [4] and a quick response to the current rise, once the arc has been established, to limit the current levels to those required for welding. The rapid initial rise of the current is most commonly achieved by bringing the welding electrode into contact with the workpiece. If the current rise, once the arc is established, is not controlled burn-back occurs, as the arc creeps up the tip of the electrode, and excess welding wire is deposited. A fast arc ignition is therefore important. This is predominantly determined by the time constant of the power supply circuit and the accuracy to which the arc current can be monitored and controlled.

2.3.2 Normal operating conditions

Some representative current and voltage characteristics for TIG and MIG welding under normal conditions are shown in Figure 2.6. These curves were obtained from experimental evaluation by Cook and Merrick [9]. From this data a model of the welding arc was created which represented the arc as a voltage source and series-resistance. This model of the welding arc has been used in this thesis for all the analysis and development of the new power supply.



Curve	Welding Type	Electrode Type	Shielding Gas	Polarity of Electrode	Arc Length
(1)	TIG	Al		+ve	.16"
(2)	TIG	Cu	Ar	+ve	.4"
(3)	TIG	Cu	Ar	+ve	.3"
(4)	TIG	Cu	Ar	+ve	.025"
(5)	TIG	Al	He	+ve	.08"
(6)	TIG	Al	He	+ve	.16"
(7)	MIG	Al	Ar		8.3" per minute
(8)	MIG	Al	Ar		8.3" per minute

Figure 2.6 Current and voltage curves for consumable and non-consumable welding

2.3.3 Short-circuit characteristics

Under normal welding conditions the electrode can come into contact with the

workpiece and the arc becomes a short-circuit. The power supply must limit the load current under these conditions. The smaller the short circuit load current the less spatter occurs [5,6,7].

2.3.4 Open-circuit characteristics

If the arc is extinguished during the welding process an open-circuit occurs across the power supply. The performance of an arc-welding power supply under these conditions is specified in the British Safety Standard 638 [8]. The no-load voltage must not exceed a d.c.value of 113 V or an a.c. value of 68 V peak (48 V r.m.s.). A voltage reducing device must automatically reduce the output voltage to these levels when the resistance of the external circuit exceeds 200 Ω. The voltage must be reduced within 0.3 second.

2.4 Arc-welding power supplies

2.4.1 Conventional arc-welding power sources with line frequency transformers

Conventional power sources used in arc-welding are traditionally described in terms of their static volt-ampere characteristics. They are either "constant-current" or "constant-voltage"[10]. Table A.2 in Appendix A summarises the type of conventional welding power sources used for the various arc-welding processes. Both the static and dynamic performance of power supplies have great influence on the properties of the arc-welding process.

For non-consumable arc-welding processes, such as TIG, a constant-current (either d.c. or a.c.) power supply is used. For consumable-electrode welding processes, such as MIG, constant-voltage power supplies are generally used. Constant-current power supplies can be used for pulsed MIG welding although they require special control.

The static output curves of "constant-current" and "constant-voltage" are shown in Figure 2.7. The characteristics are not truly constant. It is the skill of the welder in using different techniques that achieves the weld characteristics required.

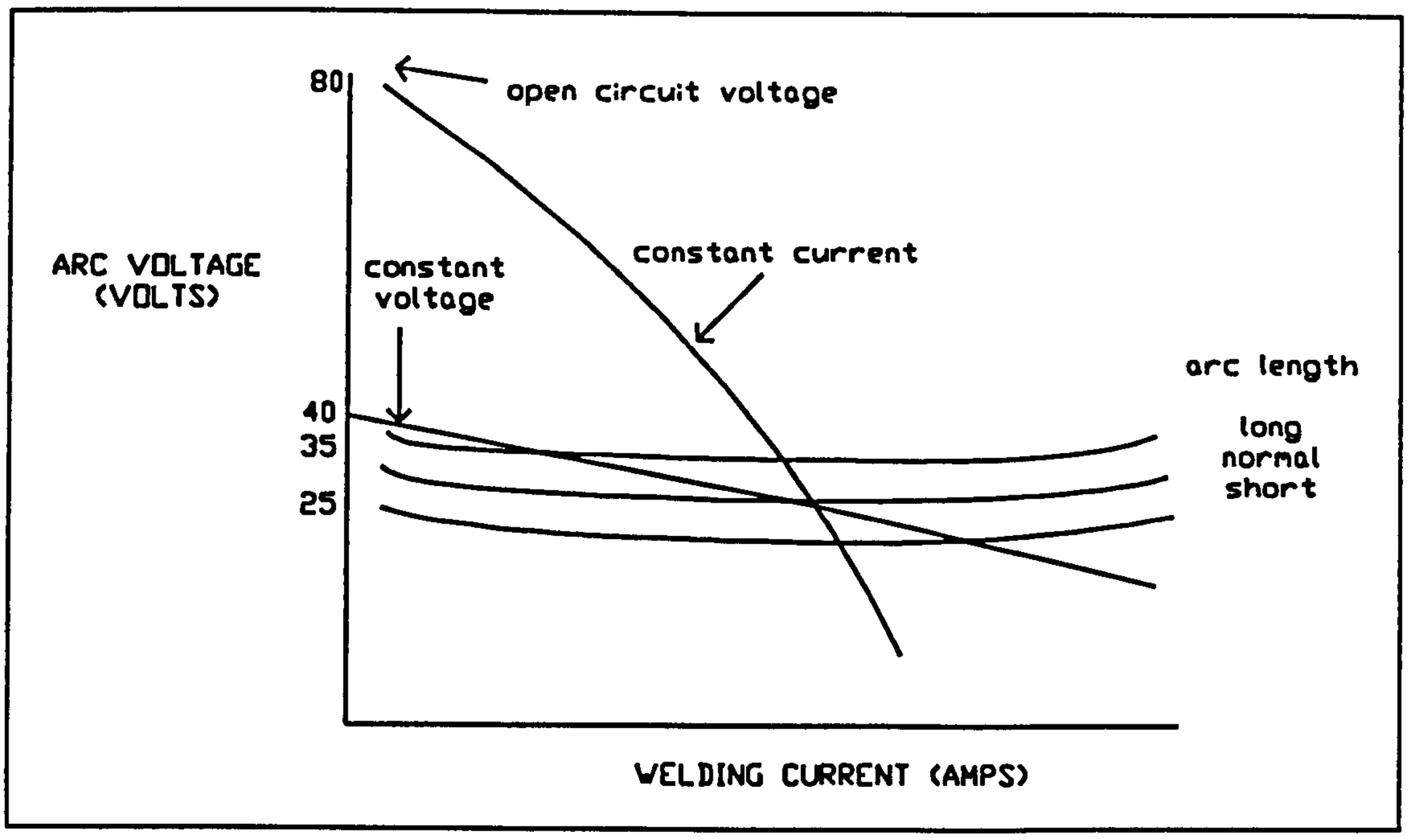


Figure 2.7 Static output curves of traditional welding power supplies

When using the "constant-current" power supplies, the welder varies the output of the power supply by changing the length of the welding arc. The preset value of the welding current determines the rate of change of voltage and current. The most common d.c. welding power supply, (the 50 Hz step-down transformer feeding a controlled rectifier), uses this method of current control. A typical TIG constant-current welding supply operates with a current range of 3 to 200 A, or 5 to 300 A, with a range of 10 to 35 V. The power supplies are usually rated to weld for 60% of the on-time of the supply as it is assumed there is some down-time between jobs.

In "constant-voltage" power supplies a uniform arc length is theoretically maintained by feeding the electrode into the arc as fast as it melts. Fluctuations in the speed at which the electrode is fed in, (this occur if the welder moves toward or away from the workpiece), are compensated for by a change in supply current.

Both these types of power supply require large smoothing inductors to remove the 50 Hz ripple from the load current. These large inductors prevent the circuit from responding quickly to current, or voltage, fluctuations under all conditions described in Section 2.3.

2.4.2 Welding power supplies with inverters

Electronic constant-current power supplies have more recently been developed that are inverters operating at around 20 kHz using pulse-width modulation techniques [11-15]. In the inverter the a.c. mains is converted to d.c.. Pulse-width modulation of the inverter produces sine-wave current in the transformer at about 1 kHz. The current is rectified on the secondary of the transformer, and a small inductor is used to smooth the 1 kHz ripple on the output current. The inverter is significantly smaller and lighter than an equivalent 50 Hz transformer power supply. The characteristics of the inverter power supply can be controlled to be either constant current or constant voltage.

Steady state conditions are attained almost instantaneously when an inverter controlled welding arc is struck. The current ripple, using inverter equipment, is extremely small and stable conditions are easily maintained. In reverse-polarity welding, inverter control produces little dispersion and reverse bead-width is uniform and wide.

2.4.3 Pulsed-arc-welding power supplies

Inverter power supplies [16-20] have allowed the development of pulsed welding described in Section 2.2.3. Some pulsed-welding power supplies have the property of synergetic, one knob, control. The operator varies the wire speed rate and the power electronics modify the other parameters according to a program stored in the controller's memory. The voltage between the workpiece and the welding electrode is continuously monitored. If the voltage drops below a predetermined value it is assumed that a short-circuit has occurred and a short pulse of high current is applied to clear the short-circuit and restore the arc. If however, the voltage fluctuates without falling to the short-circuit value it is assumed that this is indicative of arc length variations. This may be corrected by either increasing the wire-feed speed or reducing the melting rate of the filler material. The latter is reduced by decreasing the pulse-frequency, and hence mean current. The time between droplet detachment from the electrode increases, less metal is transferred, and the arc-length is restored.

2.5 Conclusion

This chapter has identified the requirements of an arc welding power supply. These are:

(i) The load must be isolated.

(ii) The load current must be able to rise rapidly at start-up and the power supply must respond rapidly to load conditions.

(iii) The power supply must have continuous short circuit capability.

(iv) The power supply must have a continuous and safe open-circuit capability.

(v) The power supply must operate into a load of very low resistance and inductance. Load currents in the range of 100 A to 350 A are required.

(vi) Possibility of both a.c. and d.c. load current.

(vii) The power supply must be controlled so that there is either constant load voltage or constant load current.

(viii) Possibility of a power supply with pulsed load current operation.

CHAPTER 3 SEMICONDUCTOR DEVICES FOR 'HARD SWITCHED' POWER CONVERSION

3.1 Introduction

Chapter 2 discussed the design imperatives for welding power supplies. Raising the frequency of operation in an arc-welding power supply gave advantages in reducing the size and improving the performance of the power supply. As the load forms part of the welding power supply circuit, and is earthed for safety reasons, the arc-welding power supply must contain an isolation transformer. A higher frequency of operation, of the converter, allows a physically smaller isolation transformer for a given power and reduced ripple on the rectified load current. The required size of any necessary smoothing inductor is consequently smaller. The reduction in size of the magnetic components of the circuit also allow the current to change more rapidly under conditions requiring pulsed current. The circuit can also respond more rapidly to short circuit and open circuit conditions. This latter attribute is explored in Chapter 7.

There are two ways to increase the operating frequency of a power converter. These are,

(i) with 'hard-switched' power circuits by using devices with low switching losses

(ii) with 'soft-switched' power circuits using conventional power devices. In these the power circuit has a topology in which the current or voltage across the devices naturally falls to zero so that the devices can be switched with almost zero switching losses.

In this chapter conventional 'hard-switched' power circuits are presented and a comparison of the performance of existing and state-of-the-art semiconductor devices in this type of configuration is made. The next Chapter will deal with 'soft-switched' power converter circuits.

3.2 Conventional 'hard-switched' power converter circuits

Consider the single switch step-down d.c./d.c. converter shown in Figure 3.1(a). In d.c./d.c. converters the output voltage is controlled by repetitive switching of the power device. The switching frequency is usually constant (limits dictated by the particular semiconductor technology).

If the repetition period is $T_s = t_{on} + t_{off}$ = on-time + off-time, then the duty cycle, D, is defined as,

$$D = \frac{t_{on}}{T_S} \quad . \tag{3.1}$$

Varying the ratio t_{on}/T_s is known as pulse-width modulation (PWM). In Figure 3.1, when the switch is closed, current flows through the switch, through the inductor and charges the capacitor. When the switch is opened, current continues to flow in the inductor and the diode comes into conduction.

If the current in the inductor is continuous (never drops to zero) it can be shown [30] that

$$V_0 = DE \quad , \tag{3.2}$$

where V_o is the output voltage and E is the input voltage. Thus, the ratio of the output voltage to input voltage depends on the fraction of time that the switch is closed during each switching cycle, provided that the inductor current is continuous.

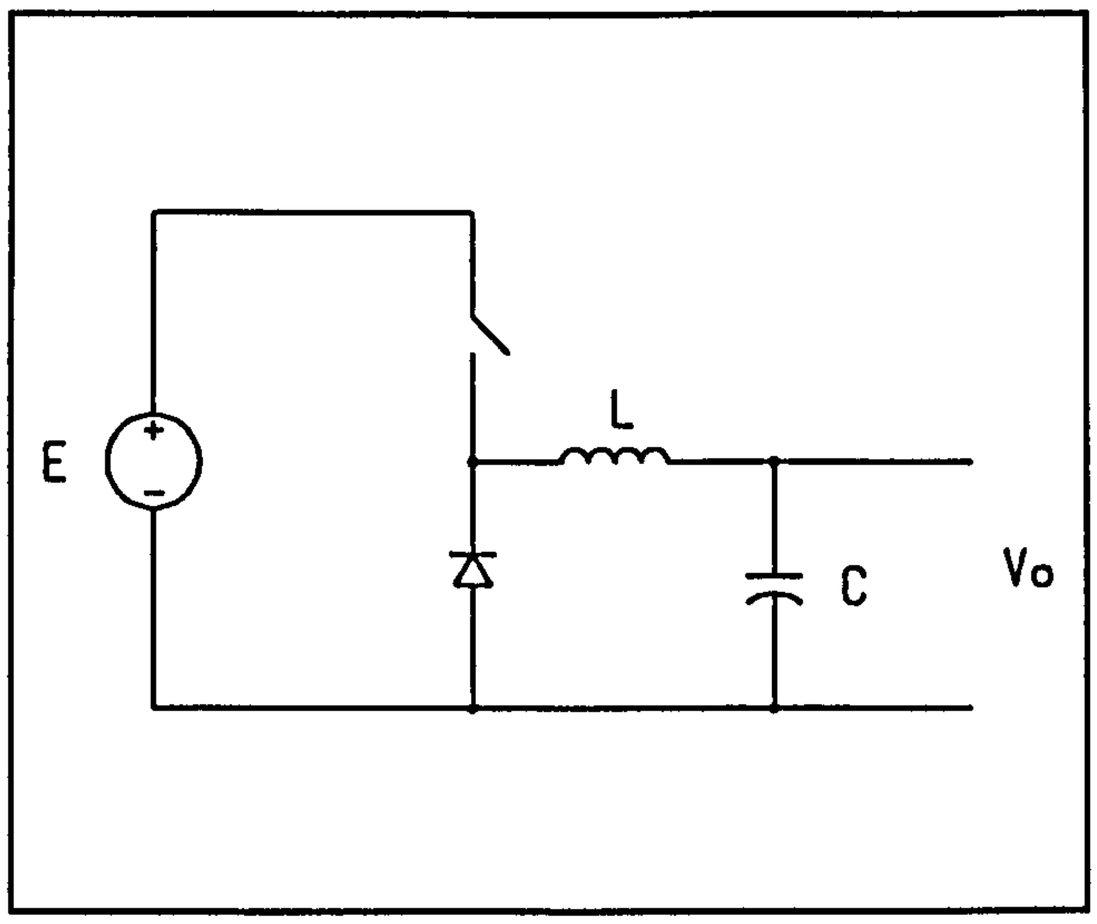


Figure 3.1 (a) Step-down d.c-d.c. converter.

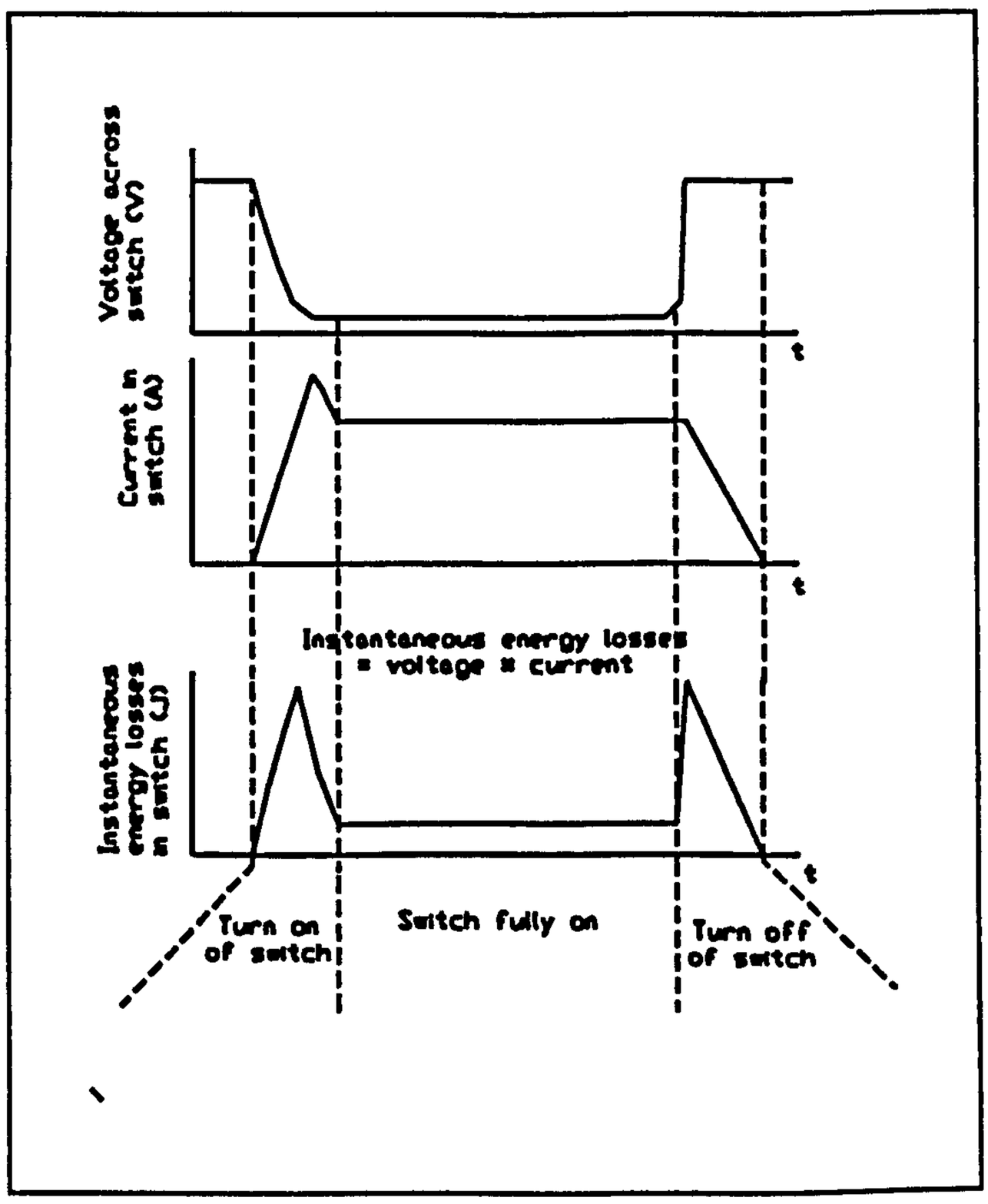


Figure 3.1(b) Energy losses in hard-switched devices

The circuit shown in Figure 3.1 (a) can also be referred to as a 'hard-switched' circuit. Figure 3.1 (b) shows the current and voltage across the device as the device is turned

on and off and the instantaneous energy losses in the device. When the device is off there are minimal losses in the device. The inductor current is flowing in the diode; the voltage across the switch is equal to the input voltage but no current flows through it. During the transition between the 'on' state to the 'off' state the energy losses in the device are greatest as the switch carries current and has non-zero voltage across it simultaneously. When the power switch is fully on the losses are determined by the magnitude of the switch current and the on-state resistance or voltage drop of the switch.

The total energy losses per switching cycle of the power switch is the sum of the on-state energy losses and switching losses.

The average power losses of the device is the product of the losses per switching cycle and the frequency of operation of the device. Increasing the frequency of operation of the circuit significantly from 20kHz to 100kHz can only be achieved if the losses of the switching device are sufficiently small so that the thermal limits of the device are not exceeded at the upper frequency.

3.3 Comparison between common power semiconductor devices

There are four commonly available types of power switching devices, the bipolar transistor, the thyristor, the MOSFET (metal-oxide semiconductor field effect transistor), and the IGBT (insulated gate bipolar transistor). Each device has a different construction and therefore different switching transition times (switching speeds) and different values of on-state losses. The devices based on thyristor technology have not been considered for this work as they have very long switching times and consequently high switching losses. To fully explore the possibility of high-frequency converters a further device was investigated for this project. The device was the SIT (Static Induction Transistor). This has the fastest known switching times of all presently available power switching devices. The construction of the power devices are first compared.

3.3.1 Bipolar transistors

A transistor is a three layer n-p-n or p-n-p semiconductor device effectively forming two p-n junctions. The transistor operates by the injection and collection of minority carriers. In power electronic applications the bipolar transistor is used as a switch either fully-on or fully-off. As a result the bipolar power transistor has a different structure to its logic-level counterpart. This is because it needs to sustain a high blocking voltage when 'off' and to have a high current carrying capacity when 'on'. In power switching applications, operation of the transistor in the linear region is kept to a minimum. This is because the instantaneous power losses, the product of the collector current and the collector-emitter voltage, are very high. The switching transition time of a power bipolar transistor is of the order of 1 μs, limiting their maximum operating frequency to about 10 kHz, unless they are operated below their maximum ratings.

3.3.2 Metal-oxide semiconductor field effect transistors (MOSFET).

The MOSFET is essentially a voltage controlled device (compared to the bipolar transistor which is current controlled). When a positive voltage is applied to the gate, with respect to the source, it converts the layer of p-type silicon, beneath the gate, into an n-type layer, i.e. an n-channel, thus connecting the drain to the source via totally n-type material, and allowing current to flow freely between the two. The gate is isolated from the p-type body region, by a layer of silicon dioxide which is a very good insulator and therefore no minority carriers can be injected into the body region via the gate. The MOSFET is a unipolar device as the current is made up of only one type of carrier. MOSFETs are intrinsically faster than bipolar devices because they have no excess minority carriers that must be moved into, or out of, the device as it turns 'on' or 'off'. The switching speed of power MOSFETs are hundreds of nanoseconds compared with a couple of microseconds for the bipolar transistor. Power MOSFETs can therefore be used without snubbers, but these may still be desirable at very high powers. MOSFETs do have higher on-state losses, for a given

drain-source voltage, than their equivalent bipolar transistor counterparts.

3.3.3 Insulated Gate Bipolar Transistor (IGBT)

The IGBT is a MOSFET with an additional p^+ layer added to the structure. The resulting device is effectively equivalent to a p-n-p bipolar transistor with an insulated MOSFET gate. The on-state characteristics of the IGBT are similar to a bipolar transistor with low on-state voltage. The turn-on switching characteristics and speeds, however, are very similar to those of the power MOSFET. There is a plateau in the gate-source voltage due to the Miller effect. The turn-off speeds are slightly longer than the power MOSFET. At turn-off the gate-source voltage does not exhibit the Miller effect. However, the drain current exhibits two distinct time intervals. A rapid initial drop occurs as the MOSFET part of the device turns 'off'. A subsequent current tail is due to the stored charge in the n^- drift region. IGBTs are divided into two categories. The first are low on-state voltage devices in which the switching speed will be slower and the second are devices which have been optimised for switching speed but they have higher on-state losses.

The gate drive requirements of the IGBT and MOSFET are very similar, and also much simpler than the bipolar transistor base drive circuits.

3.3.4 Static Induction Transistor (SIT)

The SIT has a vertical multichannel structure as shown in Figure 3.2 [21,22]. The structure is similar to that of the FET except that the length of the conduction channel is very short. Paired gates are used to increase the current capability of the device. The current can then flow in multiple channels through the n-type silicon. This reduces the power losses, the channel resistance and the gate charge required to switch the device.

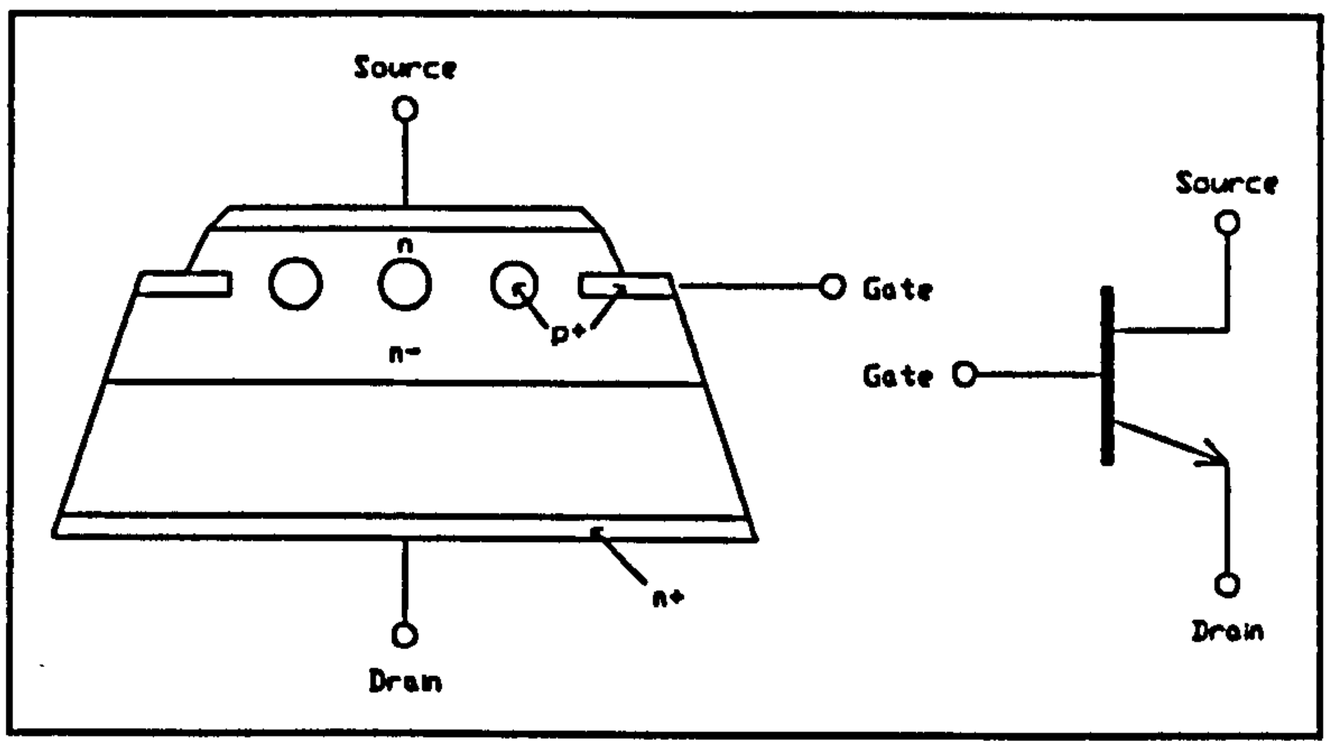


Figure 3.2 Structure of SIT

The SIT is a normally 'on' device. The channel between the drain and the source conducts when the gate source voltage is at zero volts. The SIT is turned 'off' when the gate is negatively charged. The SIT has a negative temperature coefficient of resistance and it can operate at high currents under thermally stable conditions [23]. The overall I-V characteristics of the SIT are essentially those of the triode valve [24]. The short channel in the SIT reduces the gate capacitance as well as the series gate resistance and, hence, the time constant of the gate. Gate capacitances of the power SIT are a few nanofarads. The average time constant for the SIT is 10^{-10} s [25, 26]. The SIT can have breakdown voltages of 1500 V and a drain current rating of 180A. The SIT has been used effectively in high frequency pulse-width modulation converters [27,28].

3.4 Comparison of switching characteristics of MOSFET, SIT and IGBT.

Switching times of power devices are quoted in the data books for MOSFETs, SITs and IGBTs. These are conventionally given for resistive load conditions. Few power circuits encompass a purely resistive load therefore switching times need to be confirmed in a practical arrangement to provide useful information as to a device performance.

A practical comparison was therefore made of a MOSFET, IGBT and SIT to evaluate suitability for high-frequency operation in a converter. The devices were chosen so

that they had a voltage rating of at least 600V and a current rating of at least 24A. A comparison of the data sheet values for each device is given in Table 3.1.

Data Sheet Characteristics (Maximum values)	MOSFET BSM181F	IGBT IRGBC30U	STATIC INDUCTION TRANSISTOR THF51
Drain-source voltage	800V	600V	600V
Drain current	34A	24A	30A
Gate-source voltage	±20V	±20V	-50V
Type of device	Normally off	Normally off	Normally on
Gate drive requirements	+15V -5V	+15V -15V	+5V -50V
R_{ds}on (V_{ce}sat)	0.18Ω	(3.4V @ 23A)	0.7Ω
C_{iss}	22000pF	660pF	5000pF
Power dissipation @ 25 C	700 W	100W	400 W
COST OF DEVICE	£76-00	£4-00	£200

Table 3.1 Component data sheet values

The devices were driven as a bottom switch with an inductive load in parallel with a freewheel diode. The d.c. link voltage was 400 V. An RC snubber was placed around the freewheel diode consisting of a resistor of 17Ω and a capacitor of 0.0047μF. These values were chosen so that the voltage overshoot as the switch turned off was within the voltage rating of the device. For ease of comparison all the devices were driven by the same gate drive circuit. The voltage rails of the gate drive were modified for each device (-40V, +5V for the SIT, +15V, -5V for the MOSFET, +15V, -15V for the IGBT). The switching times were measured from the waveforms obtained. The turn-on time was defined as the sum of the delay-time and rise-time of the current. The turn-off time was defined as the sum of the turn-off delay-time and the fall-time of the current. Further details on switching time measurement are given in [29]. The experimental turn-off waveform for the MOSFET is shown in Figure 3.3(a), for the IGBT is shown in figure 3.3(b) and for the SIT is shown in Figure 3.3(c). Table 3.2 summarises the turn-off times measured from the experimental switching waveforms.

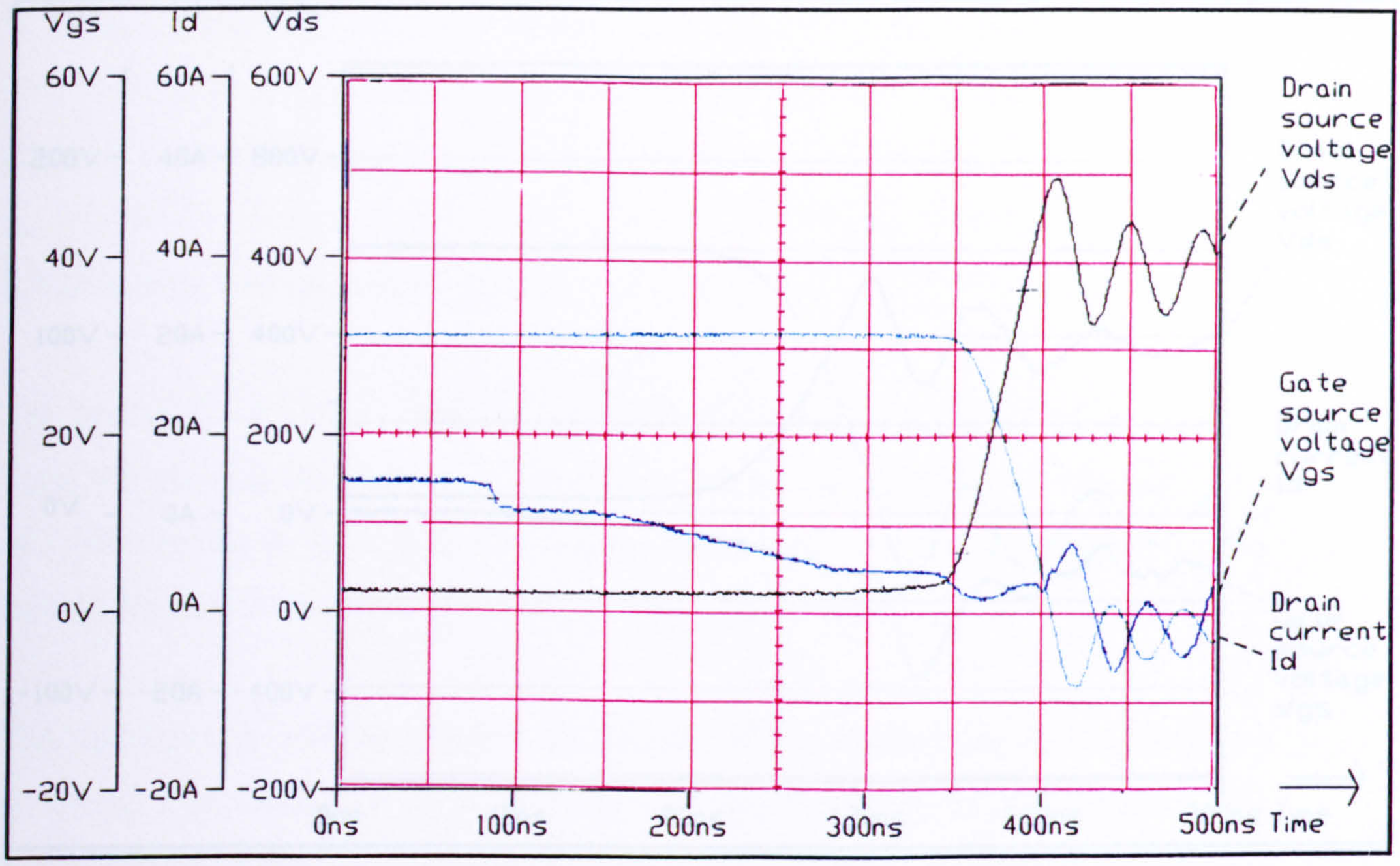


Figure 3.3(a) Experimental turn-off waveform of MOSFET showing drain-source voltage, gate-source voltage and drain current against time

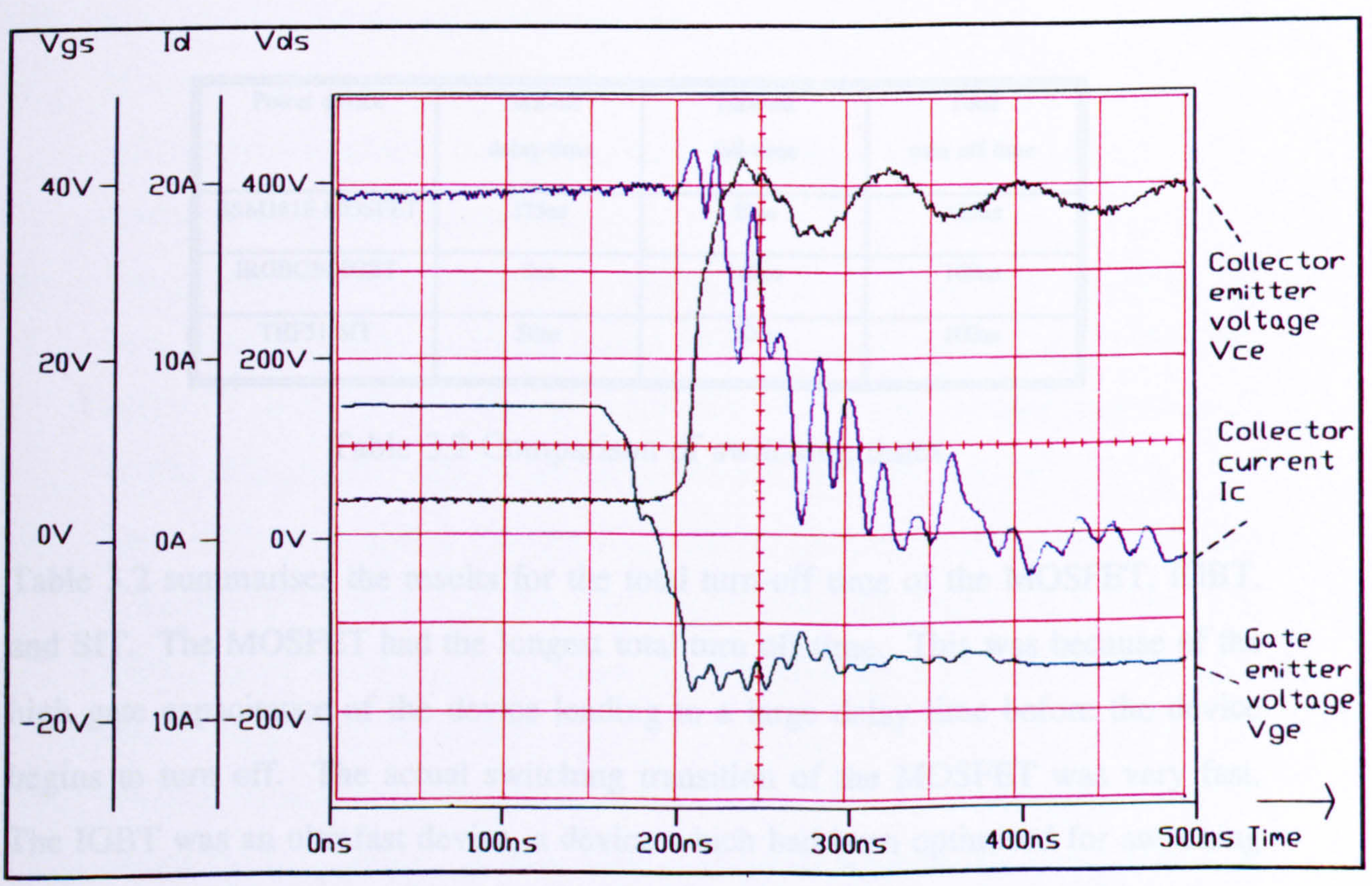


Figure 3.3(b) Experimental turn-off waveform of IGBT showing collector-emitter voltage, gate-emitter voltage and collector current against time

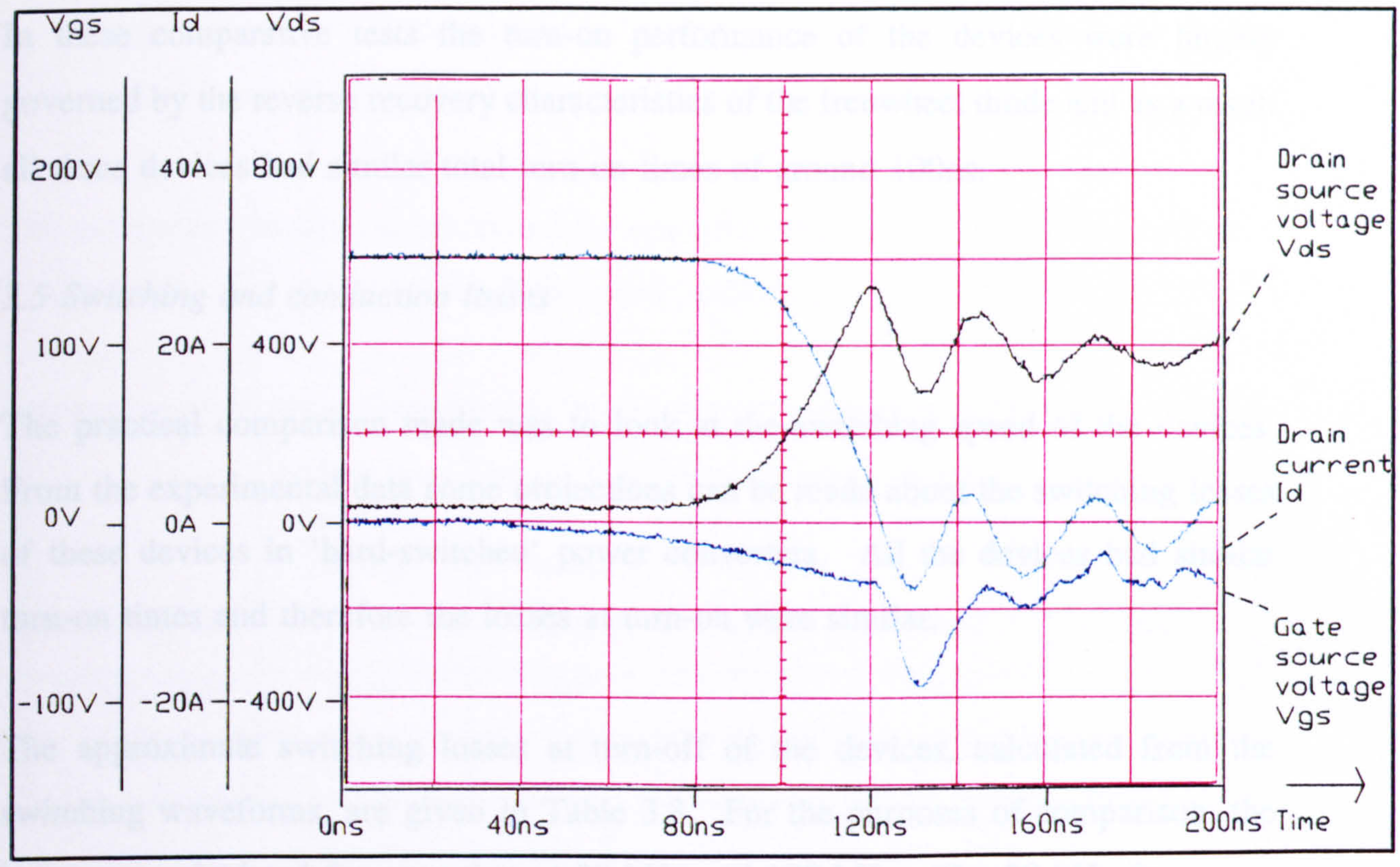


Figure 3.3(c) Experimental turn-off waveform of SIT showing drain-source voltage, gate-source voltage and drain current against time

Power device	Turn-off delay-time	Turn-off fall-time	Total turn-off time
BSM181F MOSFET	375ns	50ns	425ns
IRGBC30 IGBT	0ns	160ns	160ns
THF51 SIT	50ns	50ns	100ns

Table 3.2 Comparison of switching times

Table 3.2 summarises the results for the total turn-off time of the MOSFET, IGBT, and SIT. The MOSFET had the longest total turn off time. This was because of the high gate capacitance of the device leading to a large delay time before the device begins to turn off. The actual switching transition of the MOSFET was very fast. The IGBT was an ultrafast device, a device which had been optimised for switching speed. It exhibited very little delay in turning off once the gate signal had been applied but the fall time of the current was, as expected, slower than the MOSFET. The SIT exhibited the fastest total turn-off switching time.

In these comparative tests the turn-on performance of the devices were largely governed by the reverse recovery characteristics of the freewheel diode and as a result all three devices had similar total turn-on times of around 100ns.

3.5 Switching and conduction losses

The practical comparison made was to look at the switching speed of the devices. From the experimental data some projections can be made about the switching losses of these devices in 'hard-switched' power converters. All the devices had similar turn-on times and therefore the losses at turn-on were similar.

The approximate switching losses at turn-off of the devices, calculated from the switching waveforms, are given in Table 3.3. For the purposes of comparison, the losses are calculated for operation at 20 kHz and 100 kHz. The 20 kHz frequency was chosen because this is commonly chosen as specified operating frequency, being just above the audible range. Making the comparison of the switching losses at 100 kHz can be justified for the following reasons.

(i) It was one of the objectives of this work to produce a power supply which was substantially smaller than existing units. To achieve this a substantial jump in operating frequency is required;

(ii) The power devices suitable for a power supply of several kW are likely to have relatively large gate capacitance (the gate capacitance of the BSM181F MOSFET is 22,000 pF. Charging and discharging that capacitance to ±15 V at 100 kHz places a significant burden on the gate drive circuit.

(iii) The switching transition time for these devices (see table) will start to become a significant proportion of the total period when the frequency reaches 100 kHz. Higher switching frequencies will dramatically reduce the period when the switch is fully on or fully off.

(iv) In some inverter topologies (see Chapter 4) two devices are connected directly across the d.c. supply. A dead time is required between the turn-off of one switch and the turn-on of the other switch of about 1 μs to prevent a shoot-through current. At 100 kHz with two switching transitions per cycle this

represents 20 % of the total period and as a result raising the frequency further would start to significantly reduce the inverter output voltage.

The conduction losses are calculated for operation with a 50% duty cycle and 30 A load current. The losses for the IGBT were scaled.

Power device	Turn-off switching losses operating at 20 kHz	Turn-off switching losses operating at 100 kHz	Conduction losses (50% duty cycle, 30A load current)	Total conduction and turn-off losses at 20 kHz	Total conduction and turn-off losses at 100 kHz
BSM181F MOSFET	5.1W	25.5W	81W	86.1W	106.5W
IRGBC30U IGBT	15.2W	76W	39.1W	54.3W	115.1W
THF51 SIT	3.5W	17.5W	315W	318.5W	332.5W

Table 3.3 Turn-off switching losses and on-state losses

Table 3.3 shows that the characteristic of the SIT produces the lowest turn-off switching losses of the three devices compared. The SIT turn-off losses were 70% of the losses of the MOSFET and only 23% of the losses of the IGBT. The conduction losses of the SIT, which are independent of frequency, are however far greater than the switching losses and considerably larger than those of the MOSFET and IGBT. The conduction losses of the SIT exceed the power dissipation rating of the device. These can only be reduced by decreasing the drain current.

It can be concluded therefore that the current SIT is not an appropriate device for high power applications, despite having faster turn-off times than both the MOSFET and IGBT. The SIT is presently very expensive and its normally-on characteristic leads to a more complex gate drive with reduced fault tolerance. There is the possibility of device failure on power up of the converter or following the loss of the gate signal. These characteristics made the SIT unsuitable as the choice of a power device in an efficient and easy to control high frequency power converter.

3.6 Choice of power device for efficient high-frequency power supply

At 20 kHz the total losses in the IGBT are lower than the MOSFET. As the switching frequency is increased, the total losses in the IGBT increase more rapidly than in the MOSFET and at a frequency of 100 kHz this particular MOSFET has lower losses (still neglecting turn-on losses which are assumed to be equivalent) than the chosen IGBT. Both devices would operate within their power dissipation ratings at 20 kHz but the rating of the IGBT would be exceeded by operation at 100 kHz. It should be noted that the IGBT tested was a device with a lower current and voltage rating than the MOSFET. An IGBT with an equivalent current and voltage rating would have a larger gate capacitance and slower switching speeds increasing the turn-off switching losses. Data for a larger IGBT module is shown in Table 3.4.

Data Sheet Characteristics (Maximum values)	IGBT Module SKM50 GB 101 D
Collector-emitter voltage	1000V
Collector-emitter current	50A
Gate-emitter voltage	±20V
Type of device	Normally off
Gate drive requirements	+15V -15V
V_{ce}sat	3V @ 50A
C_{ies}	6nF
Power dissipation @ 25 C	400W
COST OF MODULE (containing 2 IGBTs and two diodes)	£50-00

Table 3.4 Component data sheet values

Power device	Theoretical turn-off switching losses operating at 20 kHz	Theoretical turn-off switching losses operating at 100 kHz	Conduction losses (50% duty cycle 30A current)	Total conduction and turn-off losses at 20 kHz	Total conduction and turn-off losses at 20 kHz
SKM 50 GB 101 D	40W	200W	90W	130W	290W

Table 3.5 Theoretical switching losses of SKM 50 GB 101 D device

The losses in the device calculated from data sheet values are shown in Table 3.5. These are significantly higher than for the smaller IGBT. The switching losses quoted in Table 3.5 are for a resistive load and therefore are the optimum which could be achieved.

The MOSFET would therefore appear to be the optimum choice for an efficient high frequency power converter operating in hard-switched mode. However, the measurement of turn-off switching losses was carried out with an R-C snubber in the circuit to prevent the voltage rating of the device being exceeded. Energy was dissipated in this snubber both at turn-on and turn-off of the device. The power dissipation in the snubber at 100 kHz would be 75 W which would be difficult to manage. The efficiency of the converter would be reduced unless energy recovery snubbers were employed.

To achieve significant reduction in power losses in the proposed power supply operating at high frequency, soft switching techniques for power converters were investigated. These would significantly reduce the switching losses of the power devices and leave only the on-state losses of the devices. If the choice of the most suitable power component was determined solely on the basis of reducing the power losses in the converter the power MOSFET would be selected. However in practice the cost of two BSM181 MOSFETs is almost three times the cost of the power module SKM50GB101D containing two IGBT devices. This IGBT module was therefore selected for subsequent development of 'soft switching' converters.

CHAPTER 4 'SOFT-SWITCHING' LOAD-RESONANT CONVERTERS

4.1 Introduction

Chapter 3 has shown that it is only possible to achieve higher operating frequencies in 'hard-switched' converters by employing snubber circuits to reduce the switching losses. In higher power applications this is not always cost effective. Resonant, 'soft-switching', power conversion techniques provide an alternative. In these converters either the current or the voltage, at the moment power switching, naturally commutates to zero. The power switch is turned 'on', or 'off', at this instant, and so the switching losses are very low. Resonant techniques can also produce low electromagnetic interference, and low reverse-recovery losses in the power diodes.

This Chapter will introduce the three basic types of load-resonant topologies. The inverter circuits used to drive these resonant circuits are described and the known methods by which power can be controlled are discussed. As an initial examination of load resonant converters and to confirm their potential in a high frequency power converter, a series load-resonant converter was designed and constructed. This chapter presents the results of this converter operating using dead-time control to achieve variation in output power. The construction of the converter allowed the practical problems in constructing and controlling a high frequency 3kW resonant converter to be identified and overcome. This experience was invaluable in the work that followed.

All resonant converters contain both capacitive and inductive components. The circuits essentially oscillate in simple harmonic modes. The electrical circuit is excited and the energy of the system is exchanged between the magnetic-field energy stored in an inductor and the electric-field energy stored in a capacitor. The resulting current in the circuit is sinusoidal as energy is transferred between the two energy storage components. Any resistance in the circuit dampens the current oscillation.

There are three main types of resonant converters, namely quasi-resonant, link-

resonant and load-resonant. Quasi-resonant converters place the resonant components around the power switches. In these types of converters the on-time, or off-time, of the switch is determined by the period of the resonant cycle [30, 31], unless an additional commutation switch is added. D.C.-link resonant converters contain the resonant components in the d.c. supply to the converter and the d.c.-link voltage therefore commutates to zero [32]. This type of resonant circuit requires complex control. Load-resonant converters use the load as part of the resonant circuit. There are three types of load-resonant converter the series load-resonant, the parallel load-resonant and the series-parallel load-resonant converter [33,34].

4.2 Basic types of load-resonant converters

4.2.1 Series load-resonant converter

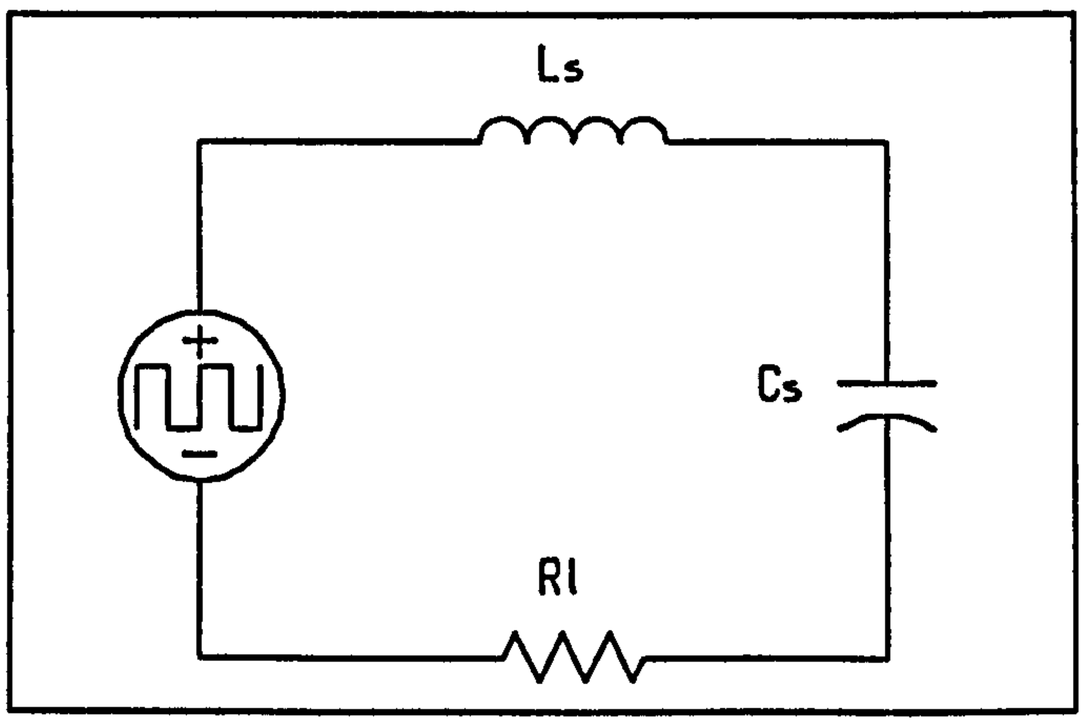


Figure 4.1 Series load-resonant converter

Figure 4.1 shows the basic configuration of a series load-resonant converter. It consists of an L-C filter placed in series with the load R_L. At the resonant frequency of the circuit the reactances of the inductor and capacitor exactly cancel and the effective impedance of the circuit is purely resistive. The resonant frequency of the circuit is given by

$$\omega_0 = \frac{1}{\sqrt{L_s C_s}} \qquad (4.1)$$

If a square-wave of voltage of frequency equal to the resonant frequency is applied to the circuit, the circuit attenuates all the harmonics of the applied voltage leaving

a current through the circuit which is essentially sinusoidal and in phase with the voltage. At frequencies below resonance the circuit appears capacitive with the current leading the voltage whereas at frequencies above resonance the current lags the applied voltage and the circuit appears inductive.

4.2.2 Parallel load-resonant converters

Figure 4.2 shows the basic configuration of a parallel-resonant converter. The resonant components are placed in parallel and the network is fed from a square wave current source (instead of the square-wave voltage source used for the series load-resonant circuit).

The resonant frequency of the circuit is also as given in equation (4.1), but this time the impedance of the circuit is a maximum at the resonant frequency. This means that, at resonance, the voltage developed across the load resistor is essentially sinusoidal and in phase with the square-wave current source feeding the circuit.

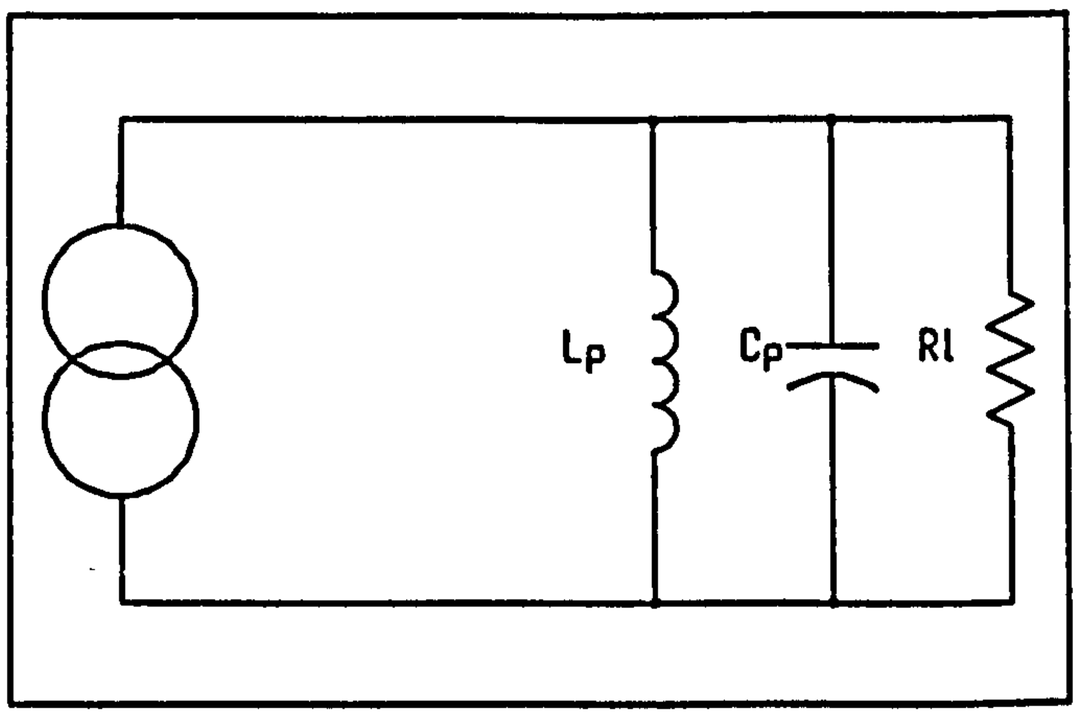


Figure 4.2 Parallel load-resonant converter

4.2.3 Series-parallel load-resonant converters

In the series-parallel load-resonant converter there is a combination of both series and parallel current paths. This circuit arrangement is fed from a square-wave voltage source in a similar manner to the series load-resonant converter.

The analysis of this circuit is more complex and it will be described in more detail in Chapter 5.

4.3 Power control in resonant converters

Power control in resonant converters is difficult to achieve while maintaining zero-current switching. There are three existing methods of achieving power control in load-resonant converters: fixed-frequency PWM, variable-frequency control and dead-time control. Figure 4.3 (a,b) shows a full-bridge and half-bridge resonant converter respectively which will be used to illustrate these methods.

4.3.1 Fixed-frequency PWM power control

Fixed-frequency PWM power control of a load-resonant converter is shown in Figure 4.5. Bhat [35] gives an analysis of this technique. The technique uses the full-bridge inverter shown in Figure 4.3(a).

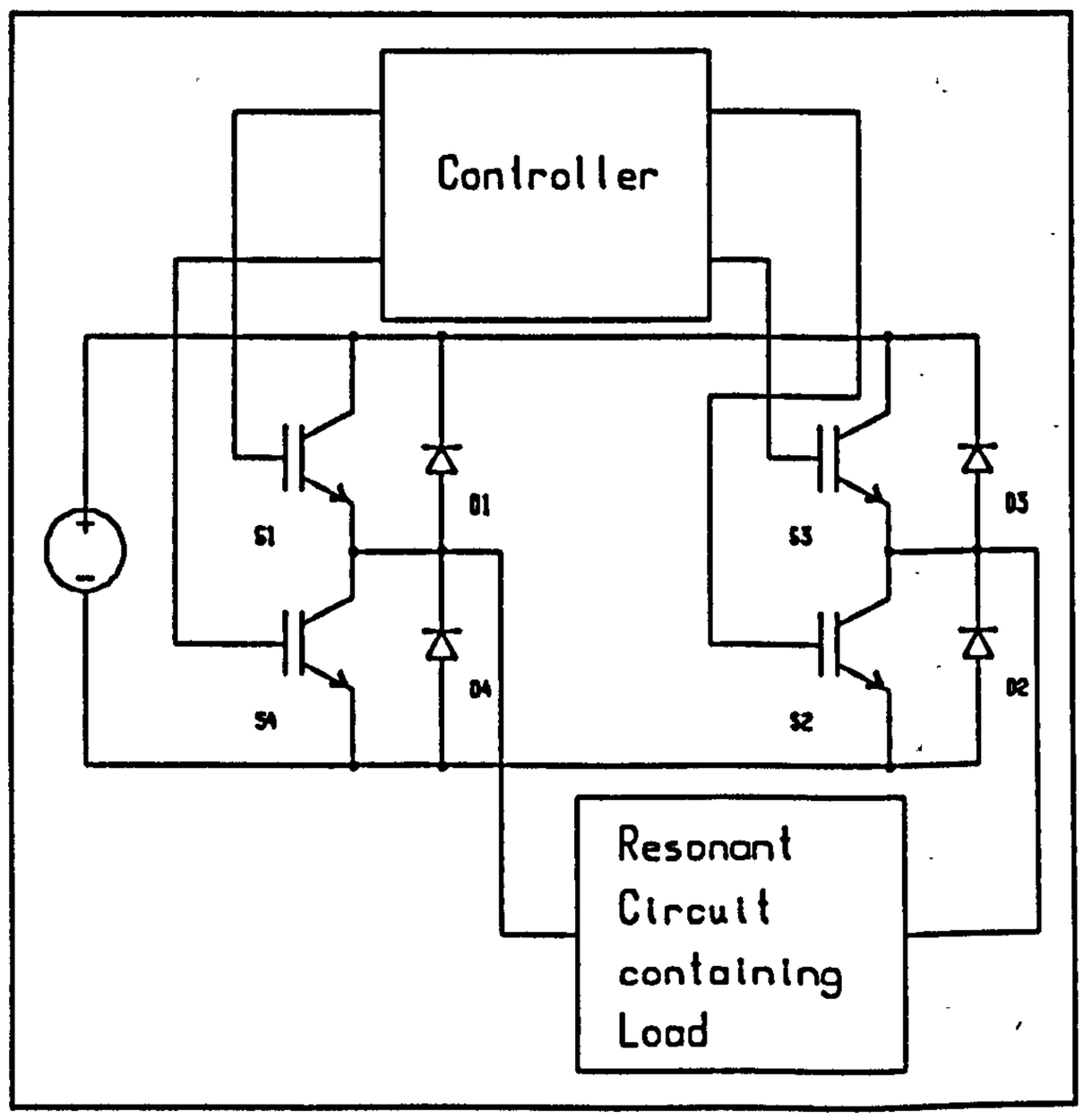


Figure 4.3 (a) Full-bridge inverter

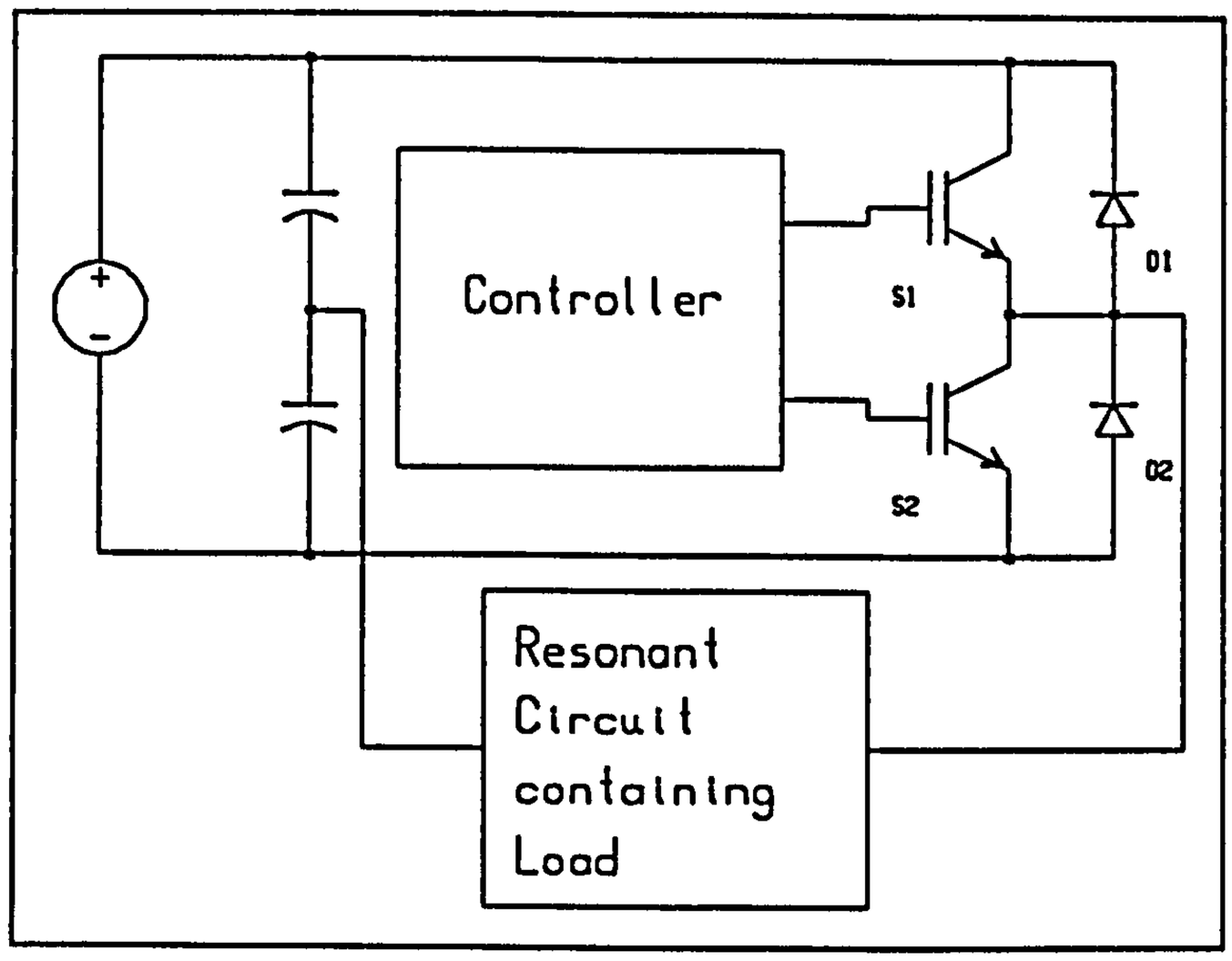


Figure 4.3 (b) Half-bridge inverter

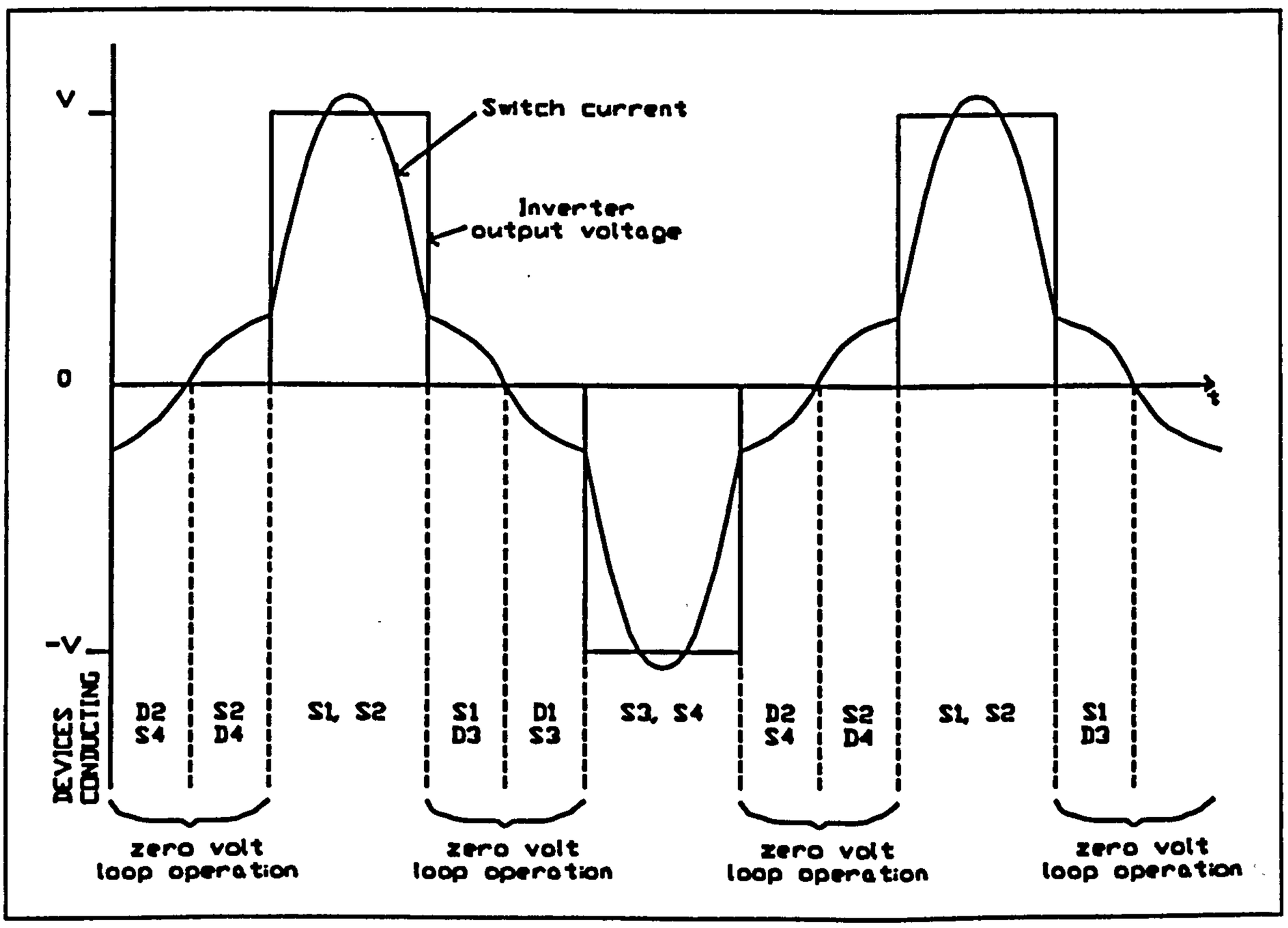


Figure 4.4 Fixed-frequency PWM power control

Power control is obtained by introducing a period of time around each zero-crossing of the switch current during which only one of the four switches is in operation (zero-

volt loop operation). This has the effect of decreasing the time within each resonant cycle that the full supply voltage excites the resonant circuit and hence reduces the output power of the power supply.

Figure 4.4 gives the switching sequence used in this technique. Power control is achieved by varying the duration of the zero-volt loop period in the converter. Zero-current switching is not maintained as the duration of the zero-volt loop increases. Both turn-on and turn-off losses are introduced but this is compensated for by the fact that as the fundamental applied voltage is reduced the peak current in the circuit drops, thus reducing conduction losses. Switching losses do however occur in both the power switches and the freewheel diodes and therefore limit the maximum frequency at which the converter can operate and hence the main advantage of using resonant converter topologies is reduced. Lossless capacitive snubbers which have been proposed for resonant converters running above resonance [34] cannot be used with fixed frequency power control because there are cases when an on-coming switch would have to charge the snubber capacitor with a large pulse of current (see section 4.3.2 for details). Jain [36] designs a parallel-leg of the circuit to act as a high-frequency filter in the circuit and places snubber capacitors around the switches thus minimising switching losses. The snubber capacitors must, however, be very carefully positioned if they are not to introduce stray oscillations.

4.3.2 Variable-frequency control (phase-control)

In variable frequency control [37] (also referred to as phase-control) of load-resonant circuits, power control is achieved by moving the operation of the resonant circuit away from resonance (Figure 4.5). This method of power control means that the voltage is no longer in phase with the current. The power in the circuit can be reduced to zero as the voltage and current eventually become 90° out of phase. The circuit is then essentially either inductive or capacitive depending on whether it is operating above or below resonance respectively. The power devices no longer have zero-current switching using this method and the frequency of operation of the power converter is again limited by the power device losses. It is most common in this

mode of control to operate the circuit above resonance. The circuit is then inductive. In this mode the power switch is turned off before the current reaches zero in each half cycle. The resonant current transfers into the freewheel diode around the opposing switch. The opposing switch can be turned on before the current reaches zero and begin conduction as the resonant current reverses thus giving zero voltage and zero current turn-on with no losses in the diodes. However, the disadvantage of operating the circuit above resonance is that the power switches must switch off current and therefore turn-off losses occur.

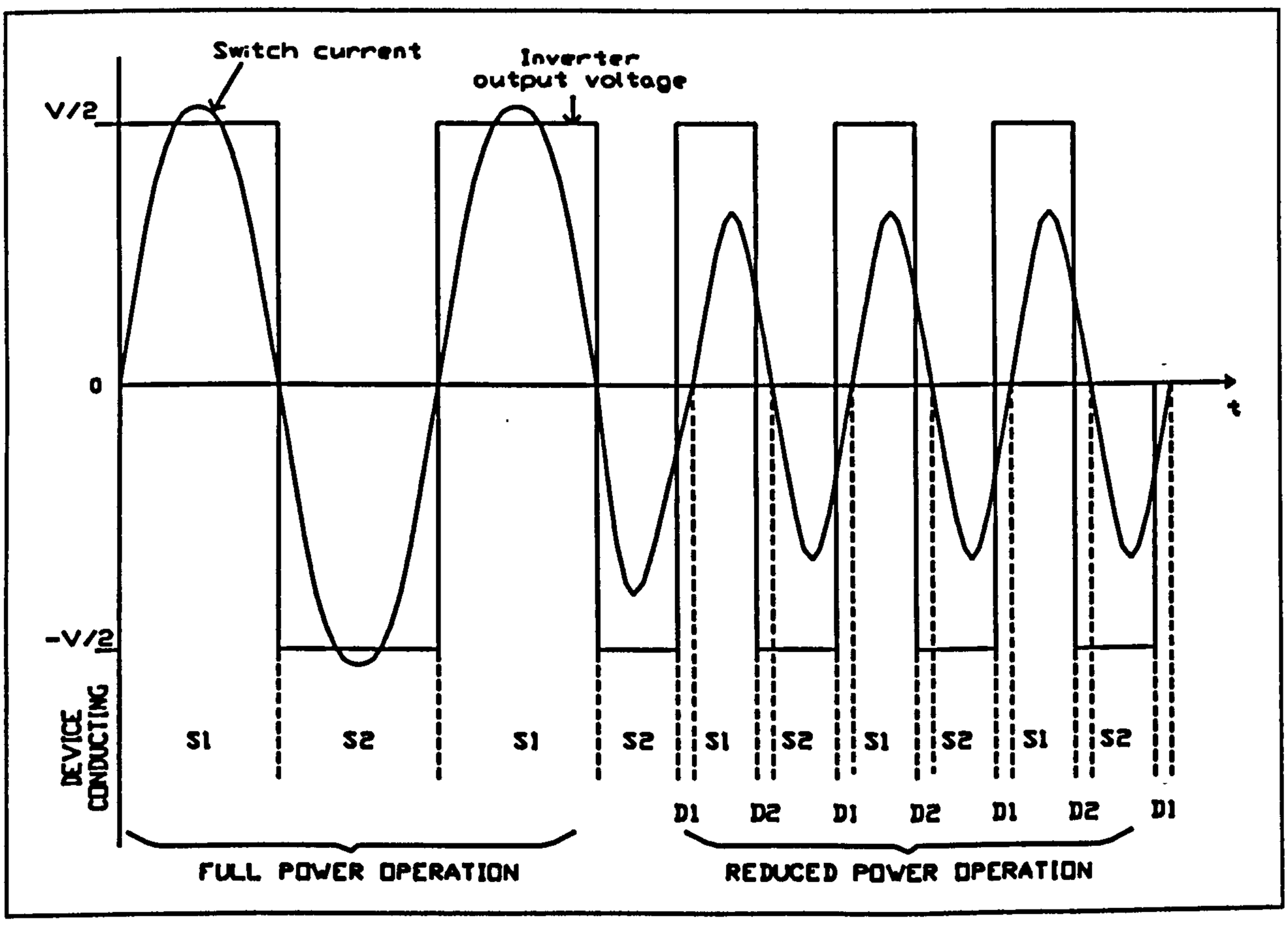


Figure 4.5 Power control by variable-frequency (phase-control)

Lossless snubbers have been proposed [34, 38] which use a capacitor directly across the switches to give zero-voltage turn-off. Alternatively, a single capacitor can be placed directly across the resonant circuit terminals. Providing the circuit always runs above resonance and the on-coming switch is always in a conducting state before the current reverses, the capacitor(s) are charged and discharged by the resonant current.

This lossless snubber circuit cannot operate when the circuit is running at full power near to the resonant frequency as it relies on the load current to charge the capacitor. The switch coming into conduction will therefore have to charge the snubber capacitor resulting in a large current spike at turn on. Furthermore, if a switch is late coming on (after current reversal) the capacitor will start charging in the wrong sense and a large pulse of current will be seen when the switch does turn on in order to reset the capacitor. In either of these two situations, a large current when a switch turns on will set up a substantial undamped oscillation between the snubber capacitor(s) and the d.c. link capacitor(s). The result of the presence of the snubber capacitor in the circuit is that although losses are reduced as the circuit operates away from resonance substantial problems can occur when the circuit is operating at full power near to its resonant frequency.

It should also be noted that although the output power is reduced when the circuit is driven away from resonance the conduction losses in the semiconductor devices remain high since there is a large reactive current flowing in the circuit. Therefore with this mode of power control the conduction losses at light loads are high and the efficiency of the load-resonant converter is reduced.

4.3.3 Dead-time control

Dead-time control [39] of a half-bridge load-resonant converter is shown in Figure 4.6. In full power operation each power switch is alternately turned 'on' for half the period of a resonant cycle of the circuit and the circuit is operated at the resonant frequency. The switch current is therefore maximum and almost sinusoidal. To reduce the power, each power switch is still turned 'on' for half the period of a resonant cycle, maintaining zero-current switching, but a delay is introduced between each switch turning 'on'. The switch current therefore transfers to the corresponding freewheel diode in the circuit before returning to zero. The current remains at zero until the next switch turns 'on'. This method of current control maintains zero-current switching at all power levels. The current ripple, when in a reduced power mode, is however quite significant and increases as the power is reduced. In some cases the load current may

be discontinuous. This would be unacceptable in a power supply for a welding arc as the welding arc would be extinguished.

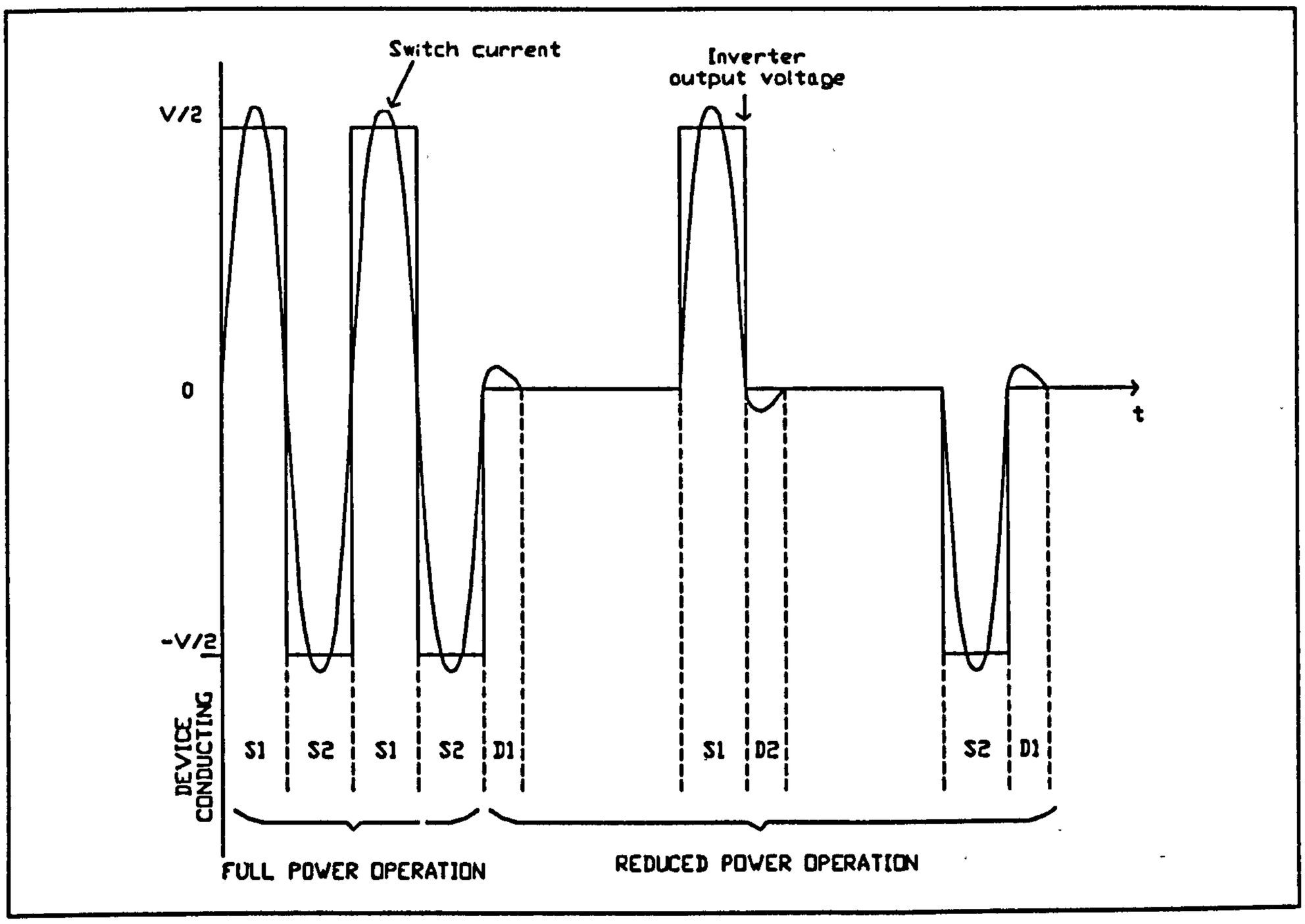


Figure 4.6 Dead-time power control

4.3.4 Summary of existing method of power control in load resonant converters

None of these methods of power control of load-resonant converters allow for a wide range in modulation of output power without reducing the efficiency of the converter switching losses or introducing output current ripple. A new method of power control which exploits the particular frequency characteristic of the series-parallel load-resonant converter will be developed later in this thesis so that load-resonant converters can deliver a wide range of output power levels while maintaining zero-current switching, maximum efficiency and minimum output-current ripple. Before this however, it is necessary to investigate the control and construction of an experimental load resonant power supply. A series-load resonant converter was constructed because this was the simplest resonant circuit to design, yet it allowed all

of the power electronic circuits required for more complex circuits to be constructed and tested (e.g. gate drives, mechanical assembly and testing of power switches at rated voltage). Control algorithms were developed so that power control was achieved using dead-time control.

4.4 Construction of a pulsed series-resonant converter using dead-time control

4.4.1 Choice of the power electronic components

Figure 4.1 showed the basic configuration of the series load-resonant converter. A half-bridge series resonant converter with feedback was constructed and is shown in Figure 4.7.

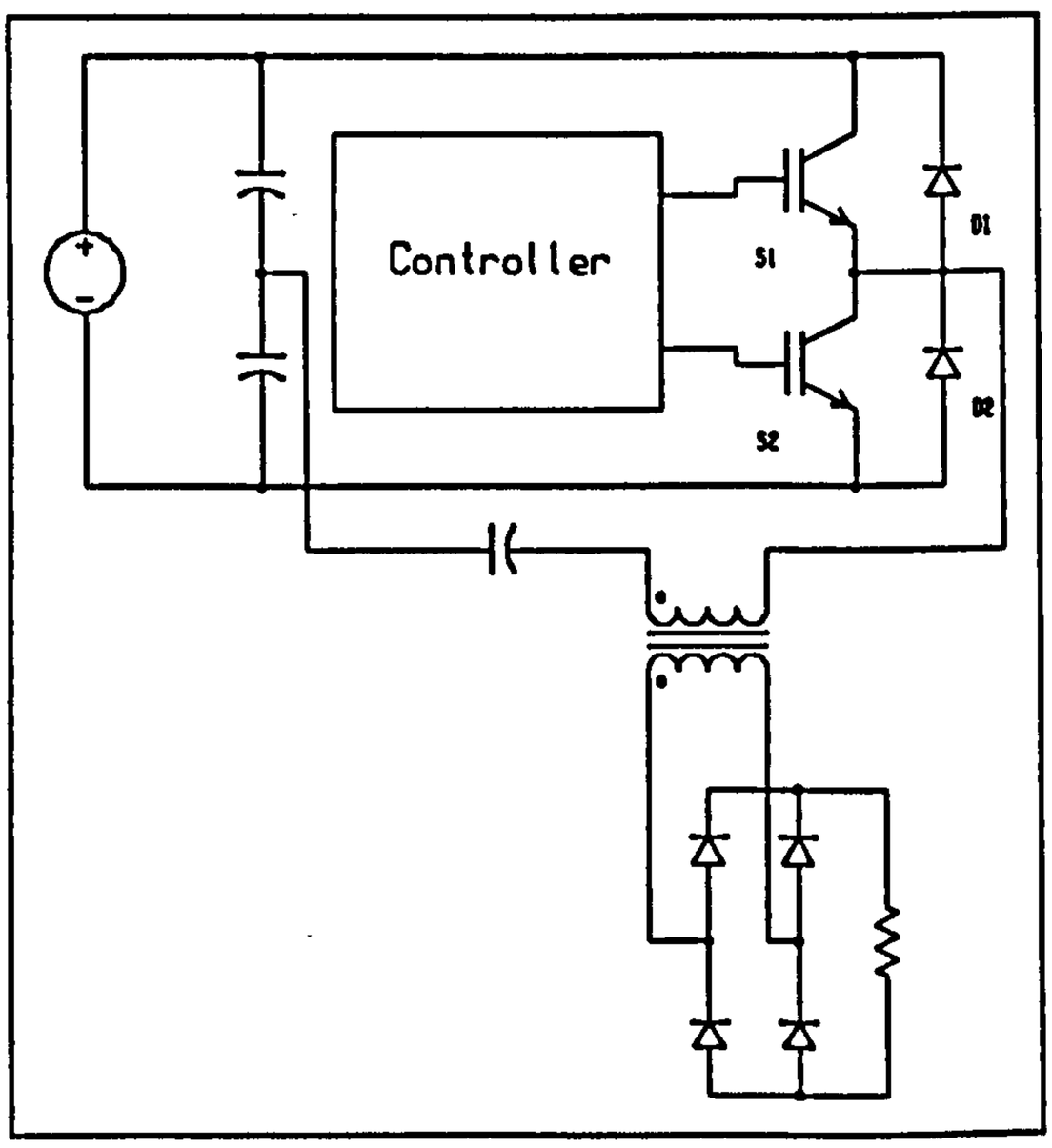


Figure 4.7 Circuit diagram of series load-resonant converter

The power switches used in the series resonant converter were IGBTs. These were chosen because of their low on-state losses and their availability in a module but most importantly, as was shown in Chapter 3, because they offered a more cost effective alternative to power MOSFETs. The resonant components in the circuit were a polypropylene capacitor (58 nF) and the leakage inductance of the isolation transformer (54 μH). The value of the capacitor was chosen after construction of the

transformer so that the resonant frequency of the circuit (as given by equation (4.1)) was 90 kHz. This was shown in Chapter 3 to be well beyond the switching frequency of a 'hard-switched' converter and it would therefore demonstrate the clear benefits of the load-resonant converter. A polypropylene capacitor was used owing to its relatively constant value of capacitance over a wide frequency range. The isolation transformer had a turns ratio of 14:4 and was a U and I core combination of 3C85 material. The number of turns was chosen so that the magnetic flux density in the core did not exceed ±300 mT.

The secondary of the isolation transformer was connected to a diode-bridge rectifier. Power diodes BYT30PI800 were used in this bridge. The bridge supplied a resistive and inductive load with unidirectional current representing a load requiring pulsed current.

A low profile Hall-effect current transducer was used to monitor the current in the resonant circuit to determine the optimum time for switching the power switches. This had a bandwidth of 150 kHz. Accurate information about the current level was only required as the switch current passed through zero. The zero-crossing was detected by a comparator and a 5 V logic signal was generated. This signal was the basis for operating the control circuit.

4.4.2 Control of the experimental series-resonant converter

To achieve pulsed current operation, using the series resonant converter, the converter was operated using dead-time control maintaining zero-current switching while giving a high-power and low-power mode. In the high-power mode, each power switch was alternately turned 'on' for half the period of a resonant cycle. The delay between each alternate switch conducting was limited to the switching times of the power devices. The period of operation in the high-current mode determined the duration of the high-current pulse in the load. In the low-power mode, each power switch was turned 'on' for half the period of a resonant cycle, to maintain soft switching, but a longer delay was introduced between each switch turning 'on'. The current in the high-frequency

transformer therefore freewheeled, after each switch had been turned 'on', returning to zero for a period of time before the next switch turned 'on'. The load current, however, was still continuous.

A novel digital controller was designed for the series load-resonant converter. It used two state machines to implement the pulsed-control algorithm for the reliable control of the power switches. State machines were used to avoid the problems caused by noise in asynchronous logic implementation. Figure 4.8 shows a diagram of the complete digital controller. The inputs to the system were an on/off signal, the comparator signal, generated by the current-monitoring circuit, and a signal from a signal generator which controlled the frequency and duration of the high-current pulses in the load of the converter.

The first state machine, 'mode', used the comparator signal to determine which power switch needed to be turned 'on', and at what time, depending on the mode of operation of the converter. Figure 4.9 shows the principle of operation of the first state machine. The machine had three control inputs. The first was the comparator signal, CP, which was generated by the current monitoring circuit. The second was a comparator blocking signal, OS, which went high 3.2 μs after a power switch had turned 'on' and the third was a delay signal, FS, which went high either 200 ns, 1 μs or 50 μs after a power switch had been turned 'off'. The delay in this signal, FS, depended on the mode of operation of the converter and which of the two power switches had last been 'on'.

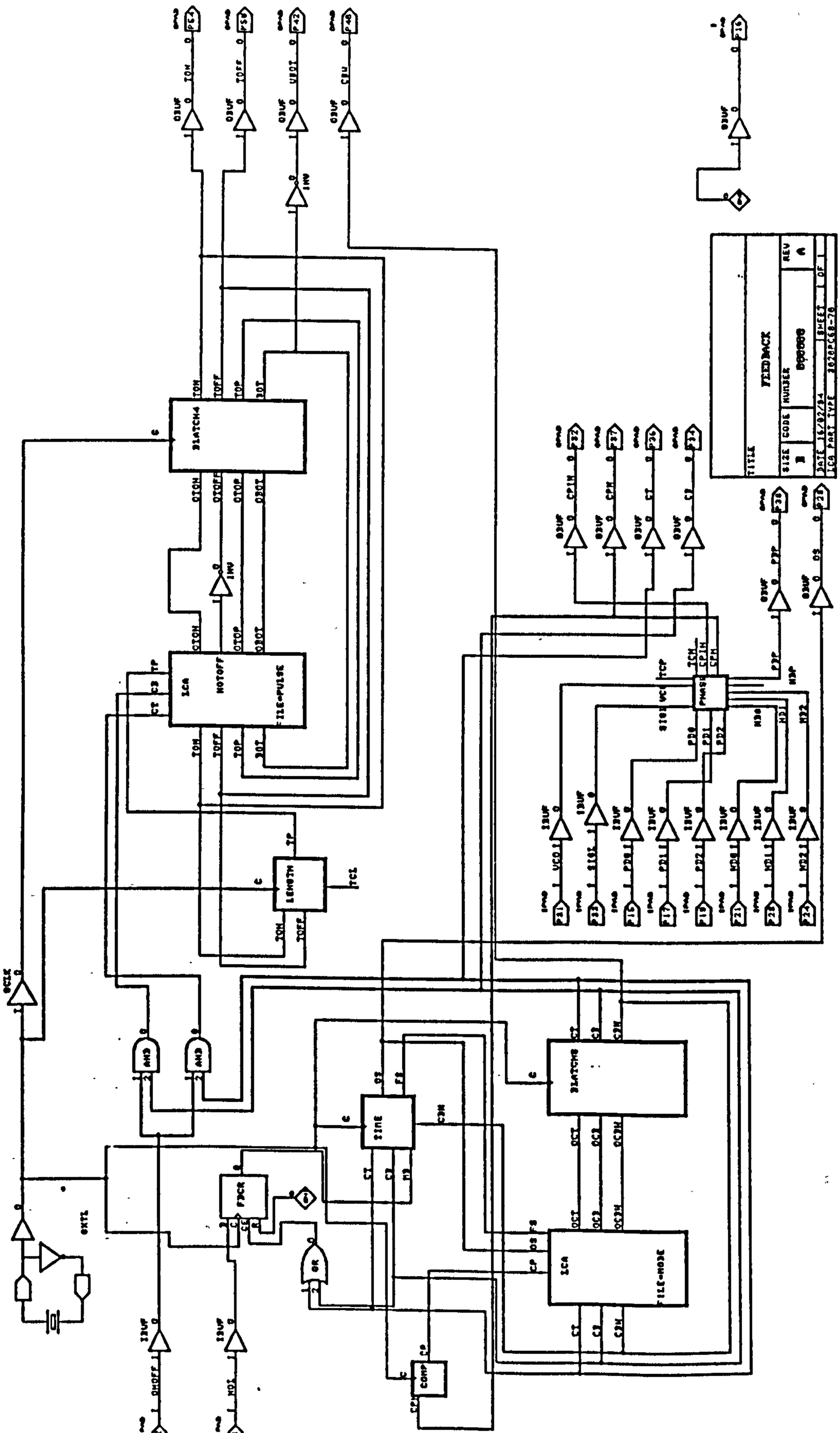


Figure 4.8 Digital controller implemented in Xilinx logic cell array

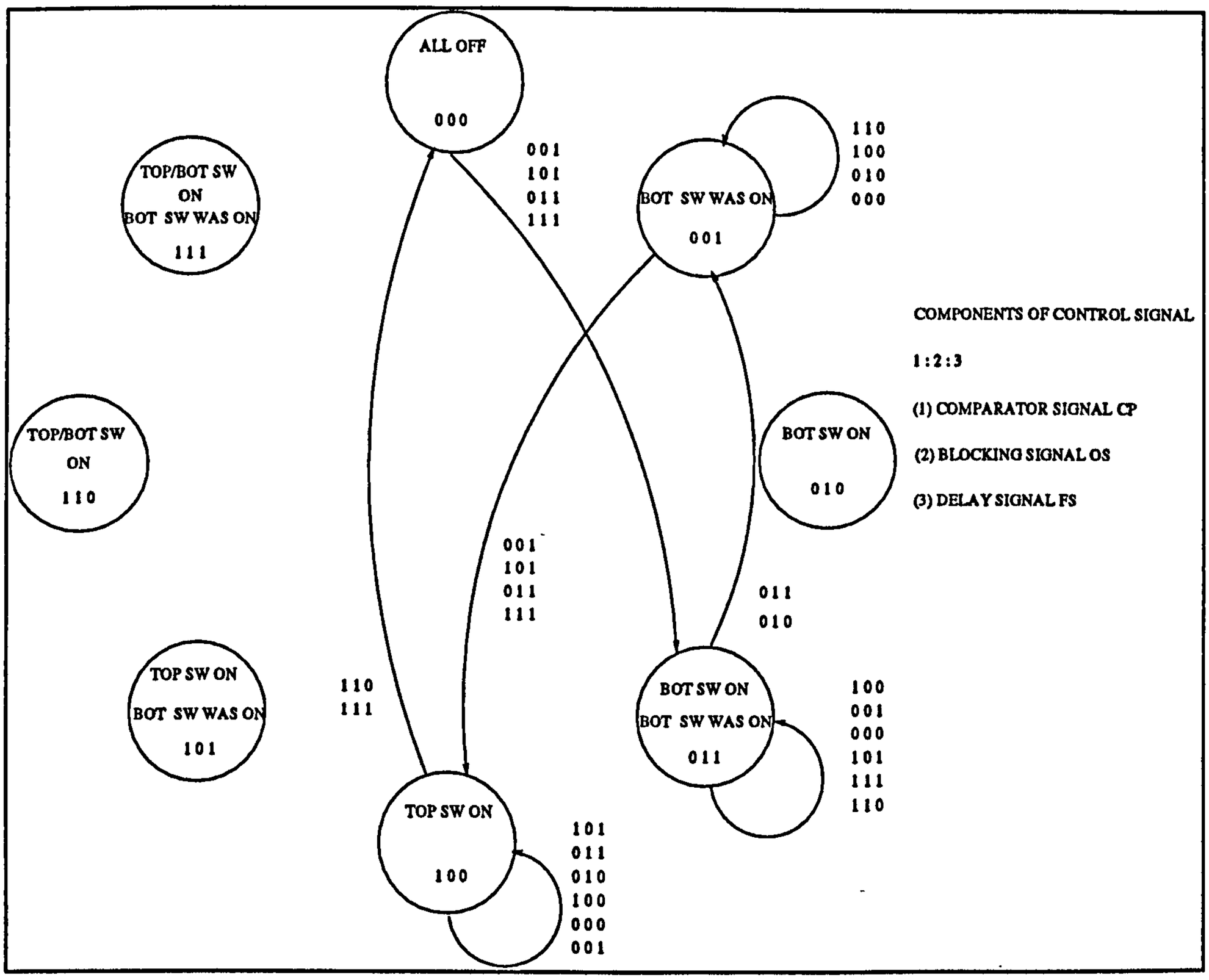


Figure 4.9 State machine diagram for the series resonant pulsed-power controller, 'mode'.

The state machine generated three signals. The three signals represented the state of the top switch, CT, the state of the bottom switch, CB, and the state the bottom switch had last been in, CBW. In each case a high signal represented the switch being 'on'.

The delay-time control signal, FS, which varied depending on whether the converter was operating in low-power or high-power mode, was only allowed to change while one or other of the power switches was 'on'. This ensured that a smooth transition occurred between low- and high-power mode.

The second state machine converted these signals into the form required to drive the gate drive circuits. Figure 4.10 shows the second state machine called 'pulse'. The top switch signal, CT, bottom switch signal, CB, and a signal representing a 3.2 μs

timing pulse, TP, were used as the control inputs to the second state machine. This machine had four output signals: a 3.2 μs 'on' pulse, TON, and a 3.2 μs 'off' pulse, TOFF, which were needed to drive the top switch gate drive, a top switch 'on' signal, TOP, and a bottom switch 'on' signal, BOT. The outputs were used as the state variables for simplicity and so the state machine had sixteen states. Six of these states were useful states.

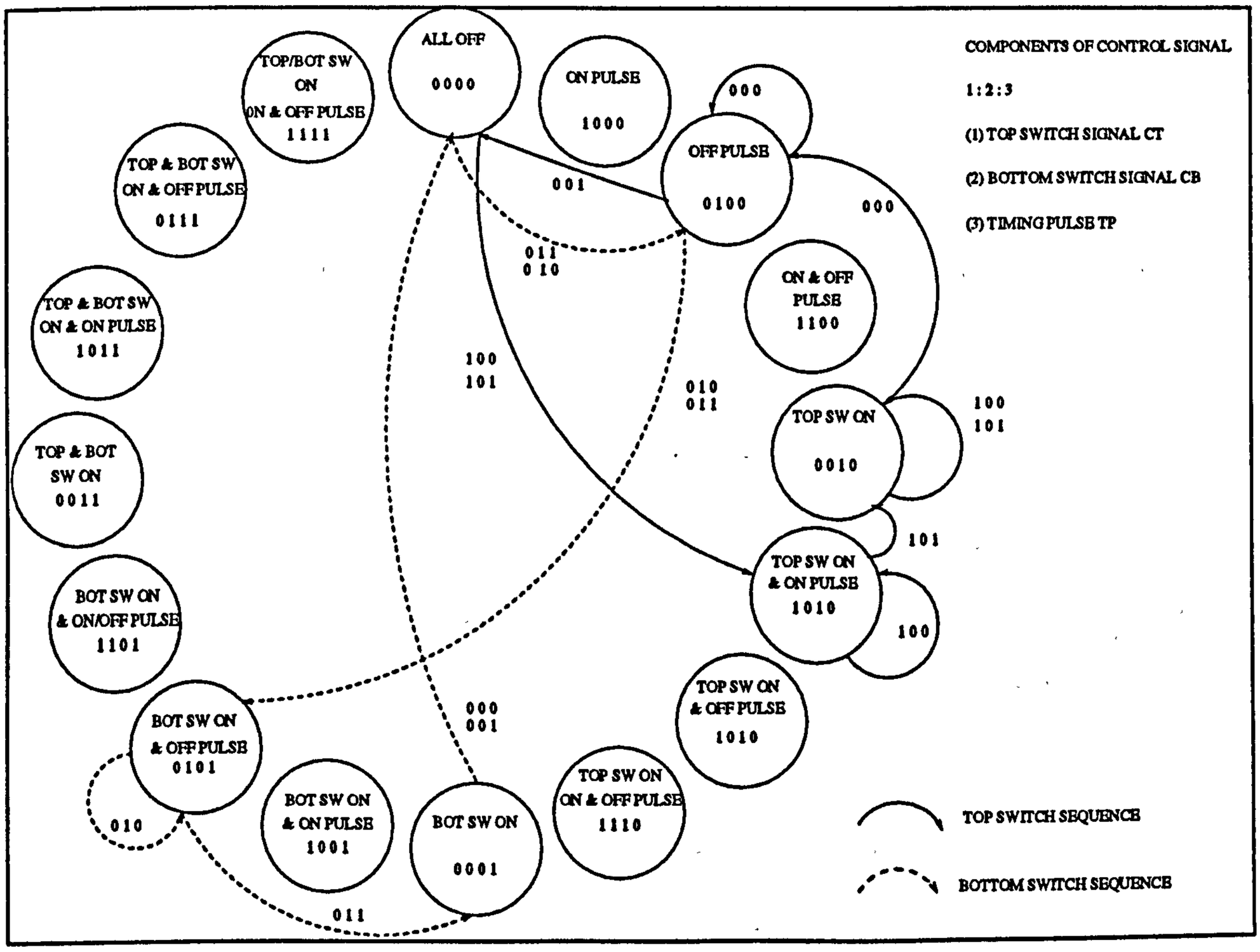


Figure 4.10 State machine diagram for the series resonant gate drive controller, 'pulse'.

The two state machines were written in the PAL logic language and the complete digital controller was implemented in a Xilinx logic-cell array clocked by a 10 MHz crystal oscillator.

4.4 3 Gate drive circuits

Figure 4.11 shows the novel isolated pulse-transformer circuit which was developed

to drive the top switch of the series load-resonant converter. The pulse-transformer provided both the power supply and gate signal required at the gate of the power switch. The circuit consisted of a primary circuit, a pulse-transformer and a secondary circuit. The primary circuit was driven from a low-voltage transistor bridge. This applied a positive voltage pulse to the primary of the pulse-transformer when the gate of the power device needed to be charged and a negative pulse when the gate needed to be discharged. The pulse- transformer was a commercially available component with a turns ratio of 1:1. The secondary circuit combined the positive and negative voltage pulses to produce a + 15 V and - 5 V swing at the gate of the IGBT while preventing discharge during the reset time of the pulse transformer core. A reverse gate voltage of -5 V was maintained across the gate of the IGBT when the IGBT was 'off' to prevent the device spuriously turning 'on'. The bottom switch gate drive circuit was a conventional non-isolated transistor push-pull circuit.

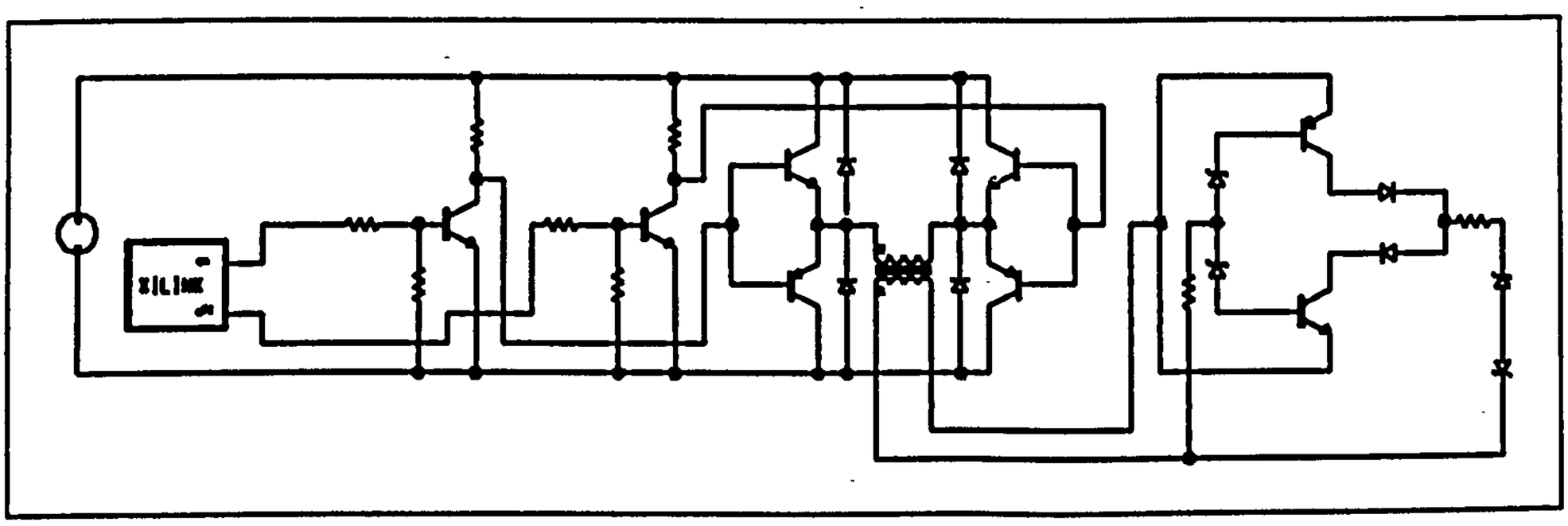


Figure 4.11 Pulse-transformer gate drive circuit

4.5 Test results

The series load-resonant power converter was tested with a 635 V d.c. supply and the load adjusted to produce a peak current of 40 A. The circuit was switched from low-power mode to high-power mode at a frequency of 60 Hz. The high-power mode occurred at a duty cycle of 0.12 producing a high-current pulse in the load for 2 milliseconds.

Figure 4.12 shows the current in the primary and secondary windings of the high-

frequency transformer when the low-power mode was in operation. The primary and secondary currents were related by the turns ratio of the transformer and initially went negative when the top switch was turned 'on'. After half a resonant cycle lasting 5.5 μs the zero-crossing of the current was detected and the top switch was turned 'off'. The current freewheeled thereby resetting the transformer flux to zero. Fifty microseconds after the top switch was turned 'off' the bottom switch was turned 'on' producing a resonant cycle of current in the opposite direction. The load current was the rectified secondary current and contained a high-ripple content due to the small value of the inductance of the load. However, even with the delay-time set to 50 μs the d.c. current in the load was continuous.

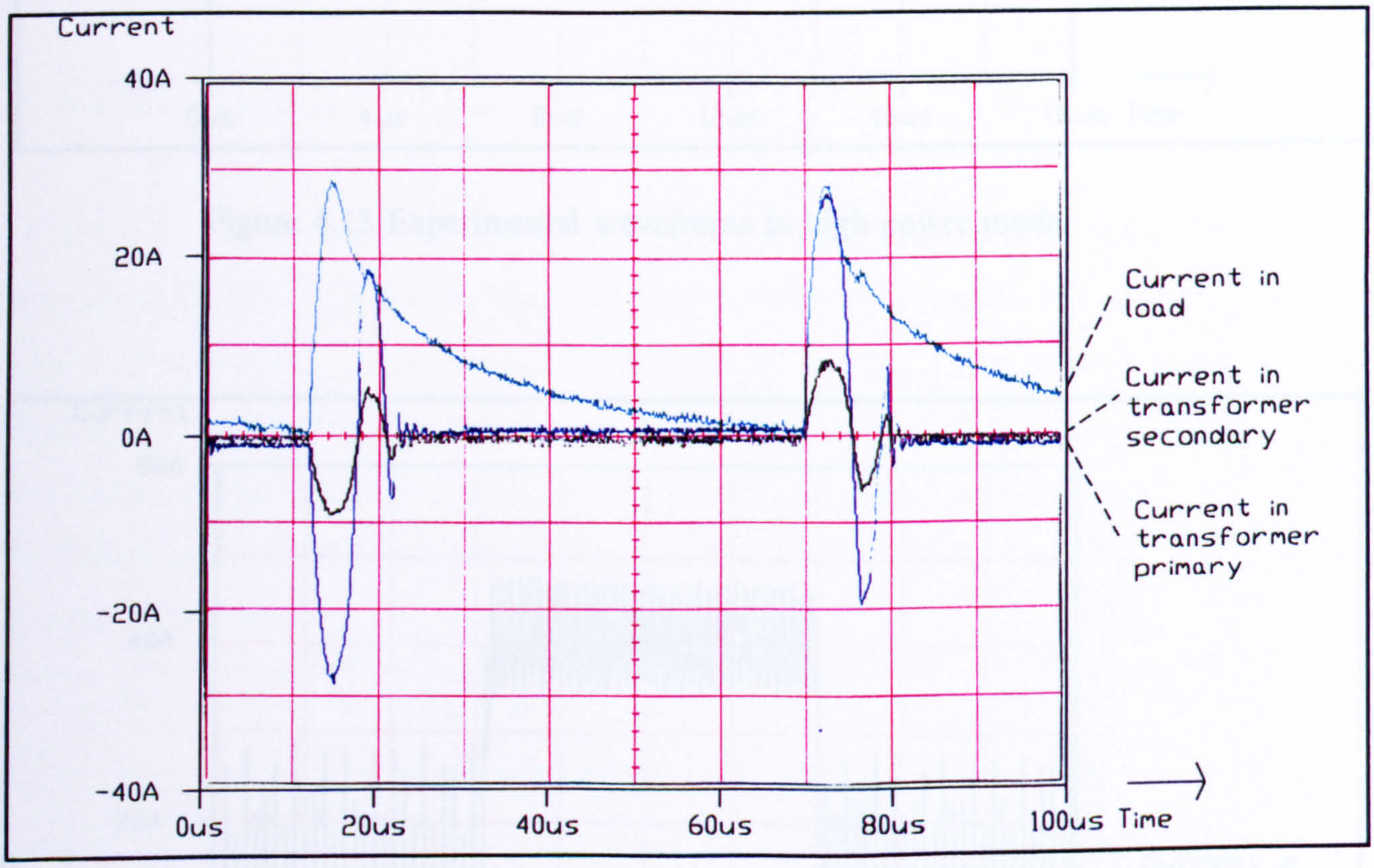


Figure 4.12 Experimental waveforms in low-power mode

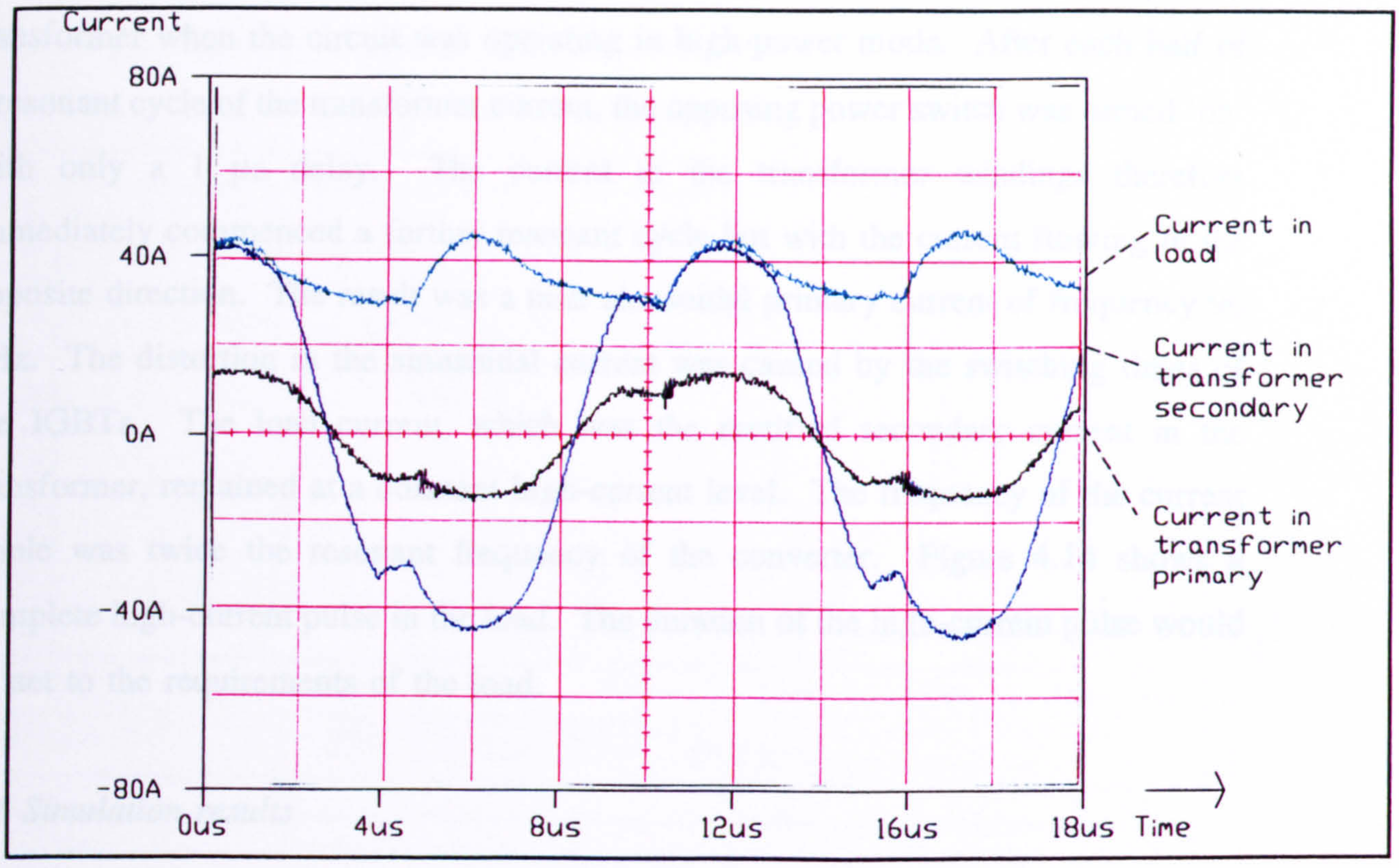


Figure 4.13 Experimental waveforms in high-power mode

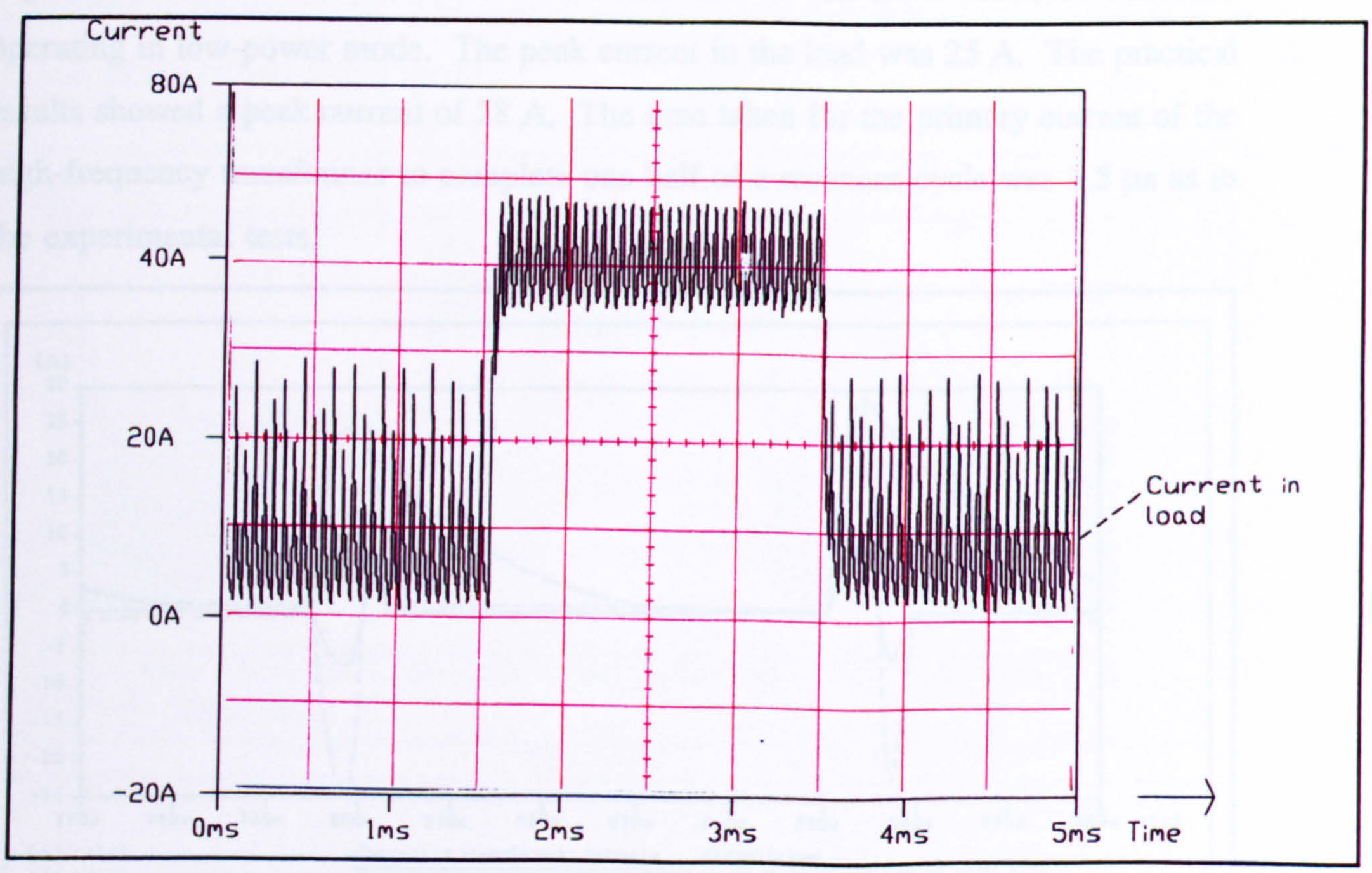


Figure 4.14 Experimental waveforms during a high-power pulse

Figure 4.13 shows the current in the primary and secondary of the high-frequency transformer when the circuit was operating in high-power mode. After each half of a resonant cycle of the transformer current, the opposing power switch was turned 'on' with only a 1 μs delay. The current in the transformer windings therefore immediately commenced a further resonant cycle but with the current flowing in the opposite direction. The result was a near sinusoidal primary current of frequency 90 kHz. The distortion in the sinusoidal current was caused by the switching times of the IGBTs. The load current, which was the rectified secondary current in the transformer, remained at a constant high-current level. The frequency of the current ripple was twice the resonant frequency of the converter. Figure 4.14 shows a complete high-current pulse in the load. The duration of the high-current pulse would be set to the requirements of the load.

4.6 Simulation results

In addition to the experimental work this circuit was also simulated in the Saber simulation software. Full details of the simulation methods are given in Chapter 6. Figure 4.15 shows the results from the simulation of the series resonant converter operating in low-power mode. The peak current in the load was 25 A. The practical results showed a peak current of 28 A. The time taken for the primary current of the high-frequency transformer to complete one half of a resonant cycle was 5.5 μs as in the experimental tests.

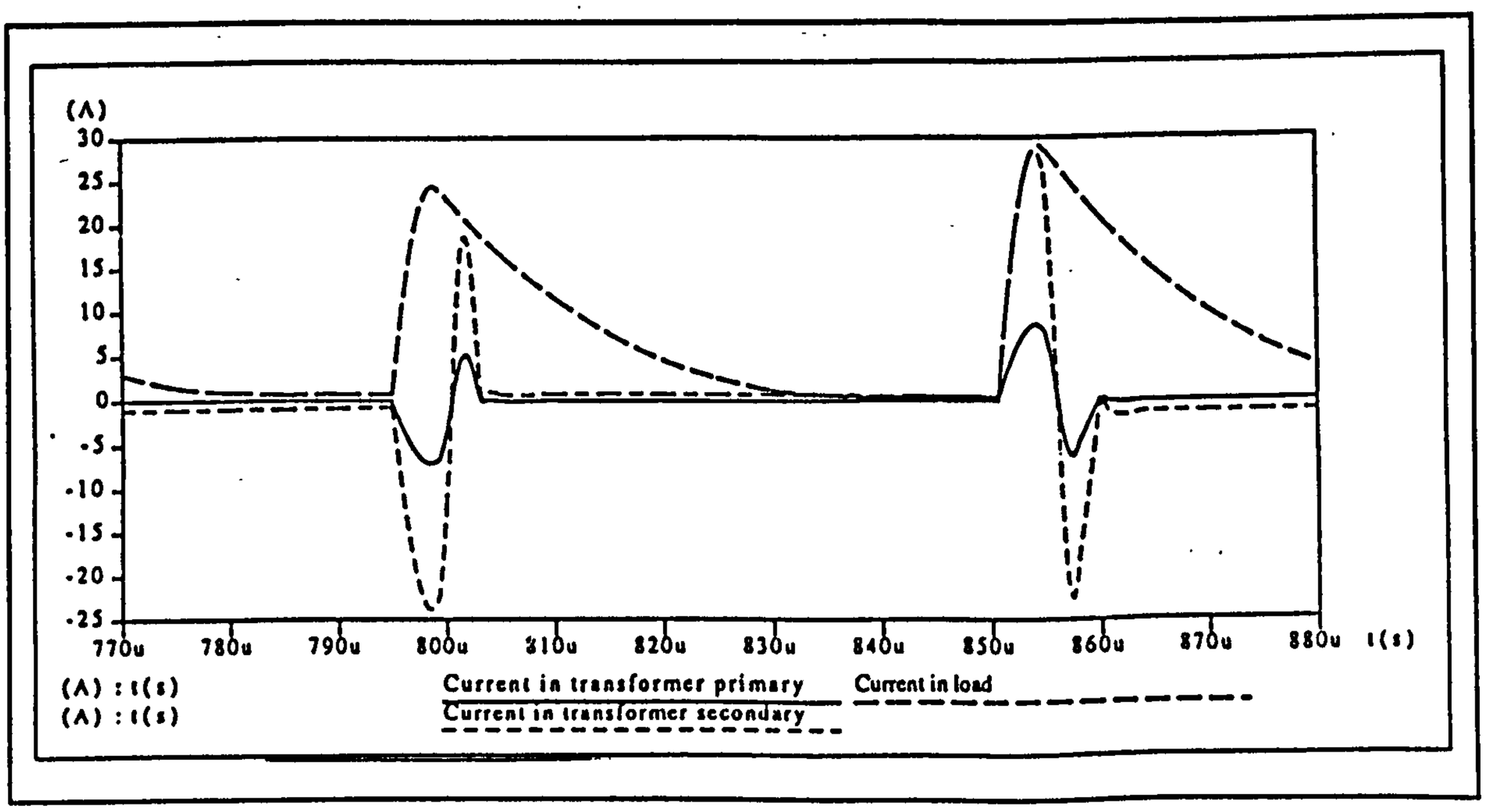


Figure 4.15 Simulated circuit waveforms in low-power mode

Figure 4.16 shows the results from the simulation of the series resonant converter operating in high-power mode. The peak current in the load was 45 A. In practice the value of the peak load current was also measured to be 45 A.

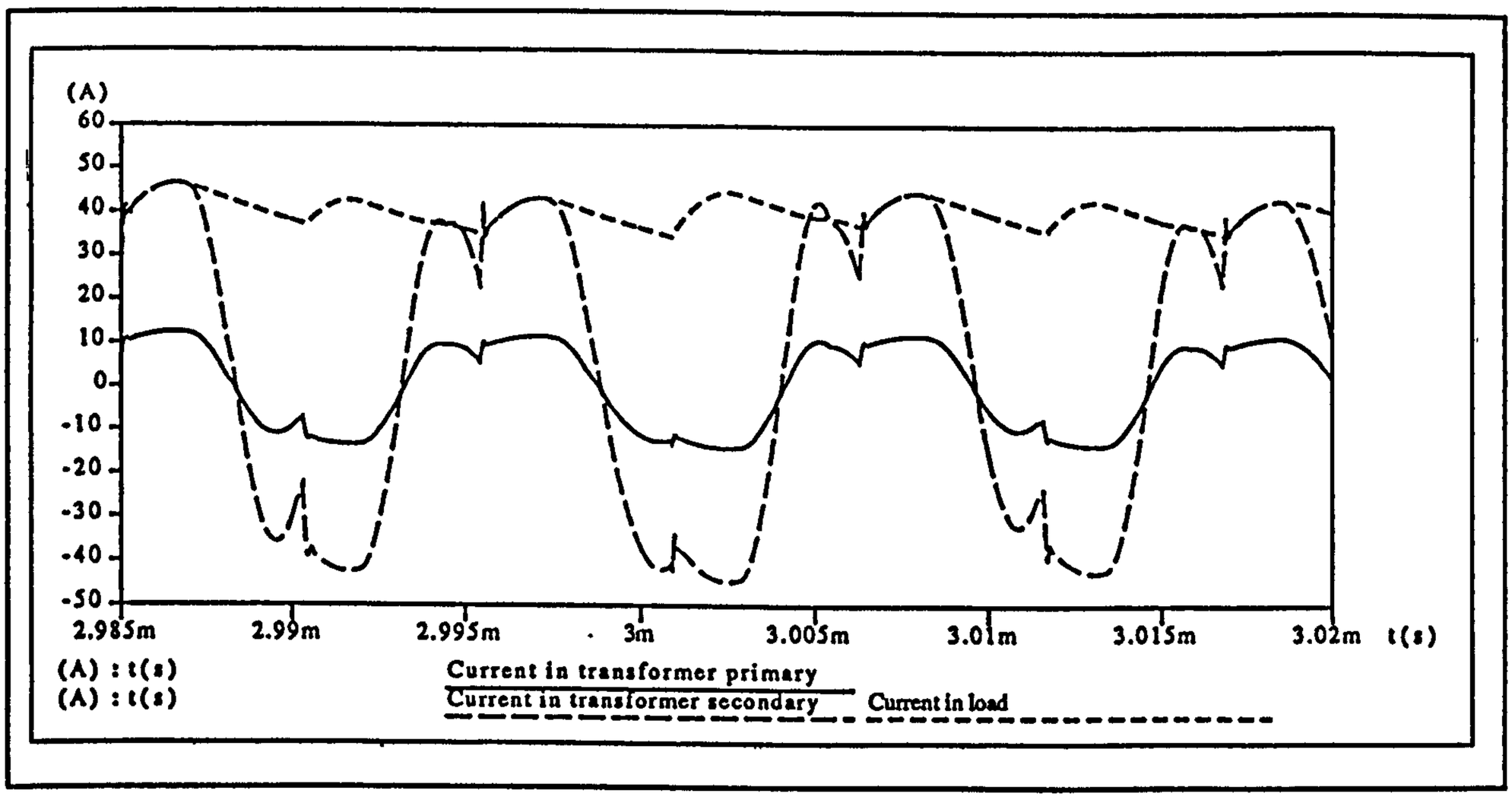


Figure 4.16 Simulated circuit waveforms in high-power mode

Figure 4.17 shows the results from the simulation of the series resonant converter operating in pulsed mode. The average current in the high-power mode was 40 A as it was in the experimental results.

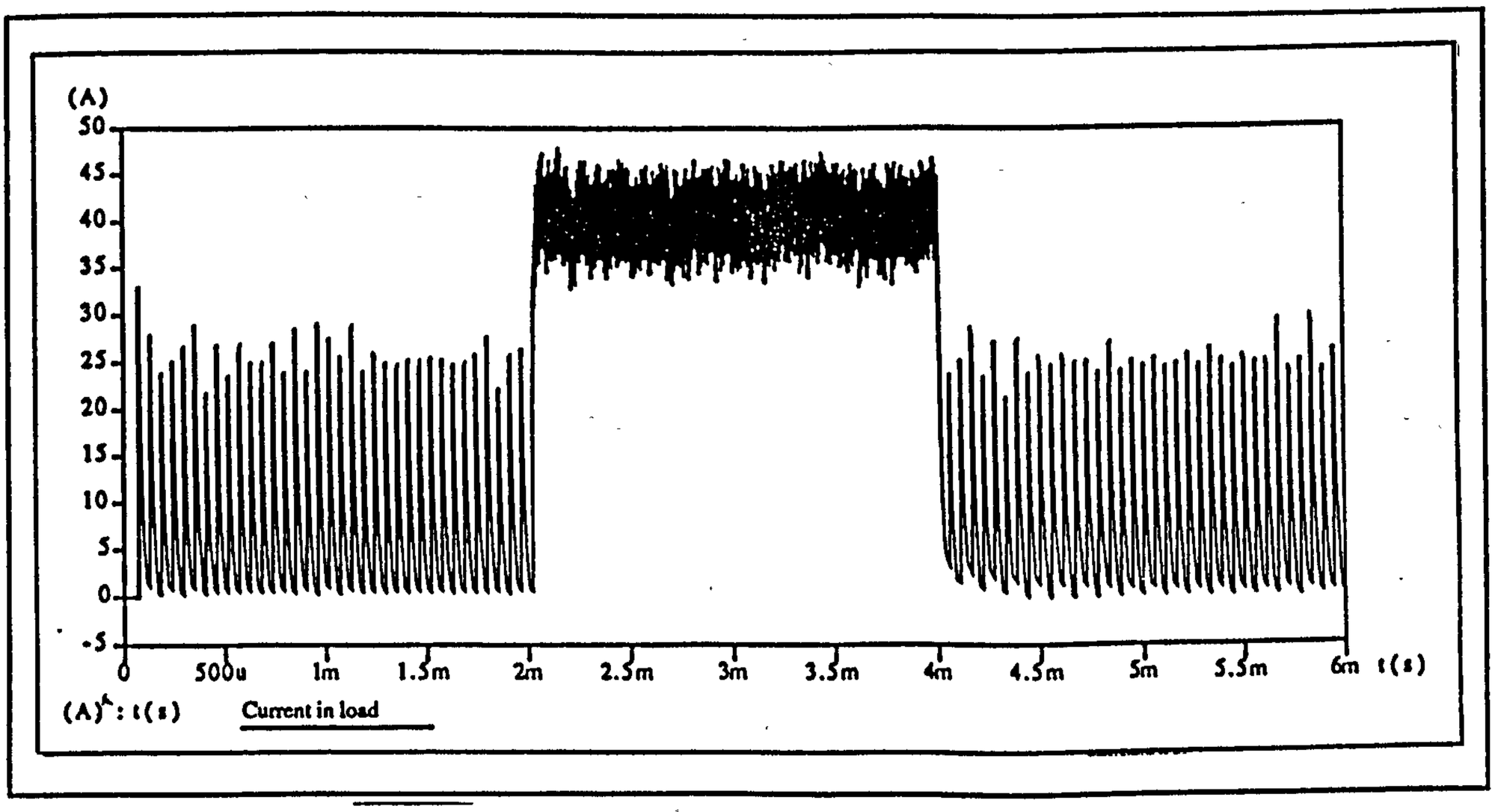


Figure 4.17 Simulated load current during a high-power pulse.

4.7 Conclusions

This chapter has briefly looked at the different configurations of resonant converters and described some of the common methods of power control which are used in load resonant converters. The chapter has presented the results of a series load-resonant converter operating using IGBTs switching at 90 kHz. The circuit was operated in a pulsed mode using dead-time control. The load current changed rapidly under pulsed conditions. Power control in the load-resonant circuit was achieved at constant frequency maintaining the benefits of zero-current switching in the power devices. Peak currents of 45 A were achieved in the load at d.c. voltages of 635 V. The circuit was modelled using the Saber simulator and simulation results produced currents of practically the same magnitude and frequency as had been measured in practice [54].

The development of reliable power circuits for a half-bridge resonant converter has been completed. From the reliability obtained with this circuit it can be concluded that state machines and synchronous logic design are very suitable means of implementing control algorithms for this type of power converter. The reliability of Xilinx programmable logic devices has also been demonstrated as there were no noticeable problems in the circuit caused by noise.. The series resonant converter has shown the effectiveness of resonant topologies in achieving high-frequency operation in a converter. Dead-time control has been shown as effective in producing pulsed current while maintaining resonance but the current ripple in the low-power mode is large.

In Chapter 2 the requirements the welding process were discussed. The series load-resonant converter provides short-circuit protection but no open-circuit protection. This is an important requirement of a welding power supply. The series-parallel resonant converter has both short-circuit and open-circuit protection. Furthermore, it has the possibility of a different and novel method of power control. The rest of this thesis now concentrates on the analysis, design and testing of a series parallel load resonant converter as an innovative pulsed-power supply for arc-welding and similar types of load.

CHAPTER 5 FREQUENCY-DOMAIN ANALYSIS OF SERIES-PARALLEL LOAD-RESONANT CONVERTERS

5.1 Introduction and prior-art analysis

5.1.1 The series-parallel load-resonant converter

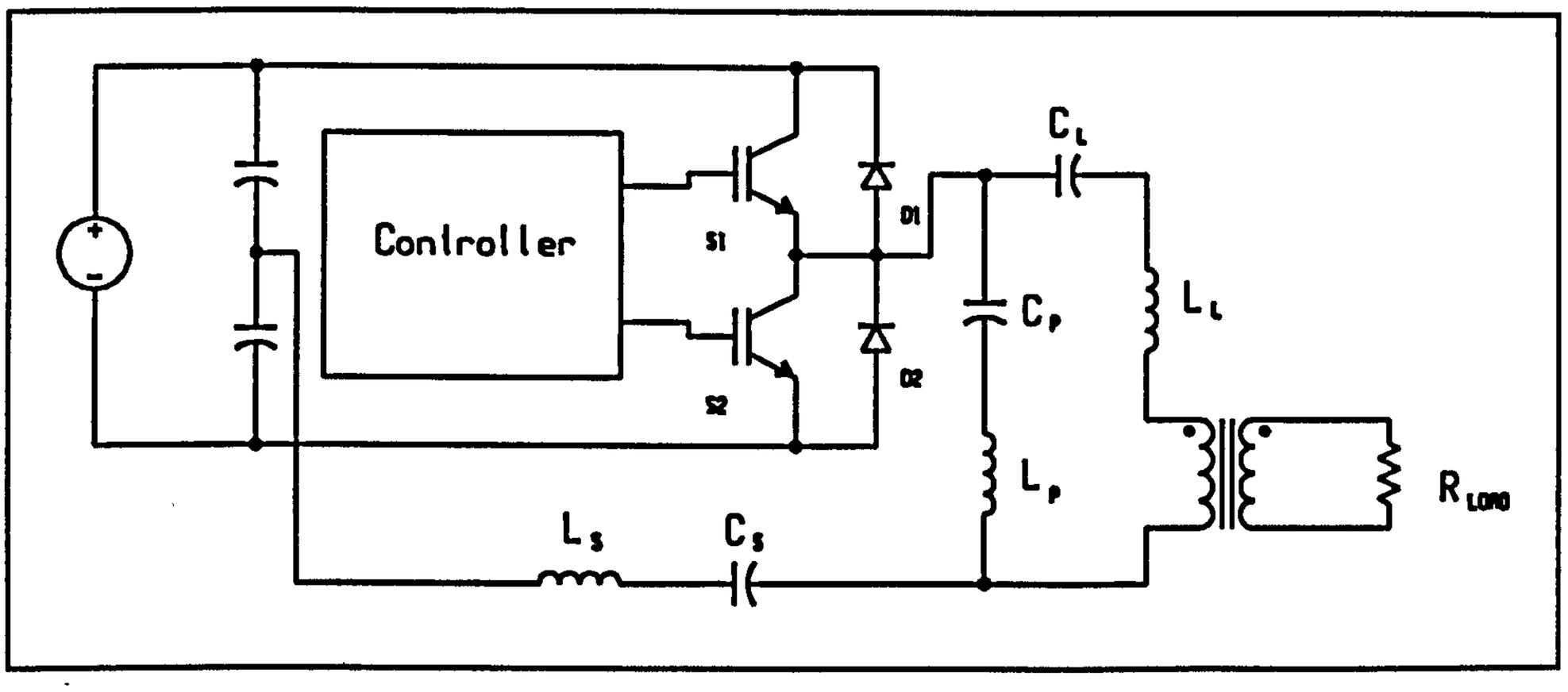


Figure 5.1 Half-bridge series-parallel load-resonant converter

A half-bridge series-parallel load-resonant converter is shown in Figure 5.1. The circuit consists of two power switches and a resonant circuit that includes the load. Each power switch is turned 'on' and 'off' alternately. A square-wave of voltage appears across the resonant components of the circuit. If the switching frequency of the power switches is at a resonant frequency of the resonant circuit, the current, in the series leg of the resonant circuit, passes through zero as each switch is turned 'on' or 'off'. The power switches are switched at zero current incurring, theoretically, no switching losses, thereby allowing the converter to operate at very high frequencies before the power rating of the power switches is exceeded.

5.1.2 Analysis of the circuit in the time-domain

The series-parallel load-resonant converter has been previously analysed in the time-domain using state-space analysis [34,40-44]. Steigerwald [40] proposed the series-parallel resonant configuration and presented the state-space equation of a full-bridge

voltage-input resonant inverter. Bhat [41] presents the state-space equations of a series-parallel resonant circuit known as an LCC type resonant circuit. The LCC circuit is a simplified version of the circuit in Figure 5.1. It comprises the series components L_s, C_s and a single parallel capacitor C_p across the load. The load is modelled as a resistance (no inductance or capacitance). In [41] equations are presented for the current in the series-leg of the circuit, the voltage across the series capacitor and the voltage across the parallel capacitor. Design examples are given of a converter operating above a resonant frequency where component values may be found from design curves.

Steigerwald [34] presents the variation in the ratio of the output voltage to input voltage, of an LCC-type series-parallel circuit, at a resonant frequency of the circuit and above that frequency. This analysis neglects the leakage inductance of the high-frequency isolation transformer.

Bhat [42] gives a generalised approach to the steady-state analysis of load-resonant inverters operating in the continuous-current mode. An equivalent circuit of the converter, with a generalised commutation circuit, is presented. The series and parallel legs of the circuit contain both inductance and capacitance. A further capacitor is in parallel with both the load-leg and parallel-leg of the circuit. Different circuit configurations are analysed by shorting, or opening, capacitive and inductive components, and equations are presented for the voltages and currents across each component in the circuit. The circuit is designed to resonate at 20 kHz and power control in the load is achieved by operating the circuit above resonance.

Bataresh et al [43] analyse an LCC-type parallel load-resonant circuit, as in [41], but simplify the state-plane analysis so that the circuit can be considered as a second-order system. The authors plot a steady-state trajectory of the solution, and then plot design curves of the gain of the circuit, and the normalised d.c. output current, against the ratio of the resonant-to-switching frequencies of the converter. They demonstrate the design of a 150 W converter.

5.1.3 Analysis of the circuit in the frequency-domain

In references [44-50], a simplified version of the series-parallel resonant converter is analysed in the frequency-domain. The basis of this analysis is the LCC type circuit configuration. This circuit is simplified, by all the authors, to a series resonant circuit, with a single resonant frequency. At the operating frequencies of these converters, the leakage inductance of the transformers should not be neglected and consequently the simplification of the circuit to one with only one resonant frequency is an over simplification. Load-resonant converters with up to four energy storage elements have successfully been analysed using a.c. analysis techniques [51,52].

The series-parallel load resonant circuit contains six energy storage elements and has multiple resonant frequencies. It will be shown in this chapter that the component selection determines the location of the resonant frequencies and hence the frequency response of the circuit. Furthermore equations are derived that allow the resonant components to be determined for a specified set of resonant frequencies.

5.2 Analysis of series-parallel load-resonant converter using impedance concepts

5.2.1 A simplified-circuit model

The analysis of the series-parallel load-resonant converter, in the frequency-domain, is undertaken using a simple model to represent the power converter. The power switches, in Figure 5.1, applied a square-wave voltage waveform of a given frequency to the 'resonant' circuit arrangement. In the simple models (see Figures 5.2 and 5.3) the 'resonant' circuit arrangement is assumed to have a sinusoidal voltage of fixed frequency applied to it. The frequency of this voltage is equal to the fundamental frequency of the square-wave voltage, applied in the real converter. The effect of the neglected harmonic components of the square-wave voltage, on the circuit operation, is discussed later. In Figure 5.2 each leg of the circuit is represented by a reactance (and resistance in the load-leg). The series-leg reactance is X_S, the parallel-leg reactance is X_P and the load-leg reactance, representing the isolation transformer and any load reactance, is X_L. The load resistance R_{LOAD} is referred to the primary side

of the isolation transformer and represented by R_L. From Figure 5.3 it is seen that two legs of the circuit contain inductance and capacitance; the series-leg of the circuit comprising a series inductor, L_S, and a series capacitor, C_S; the parallel-leg comprising a parallel inductor L_P and a parallel capacitor, C_P. The high-frequency isolation transformer and load of the actual circuit are replaced by an equivalent inductor, L_L, a capacitor, C_L and a resistor R_L. The high-frequency isolation transformer is assumed to require zero-magnetising current. The inductor, L_L, is used to represent the leakage inductance of the transformer, referred to the primary side, together with any referred load inductance. Additional inductance may also be added. The resistor, R_L, is the resistance of the load referred to the primary side of the transformer. The capacitance, C_L, is added to the load-leg in order to keep the analysis totally general. In practice some loads may be capacitive, or, in other cases, adding capacitance could have a beneficial effect by reducing the effective load inductance.

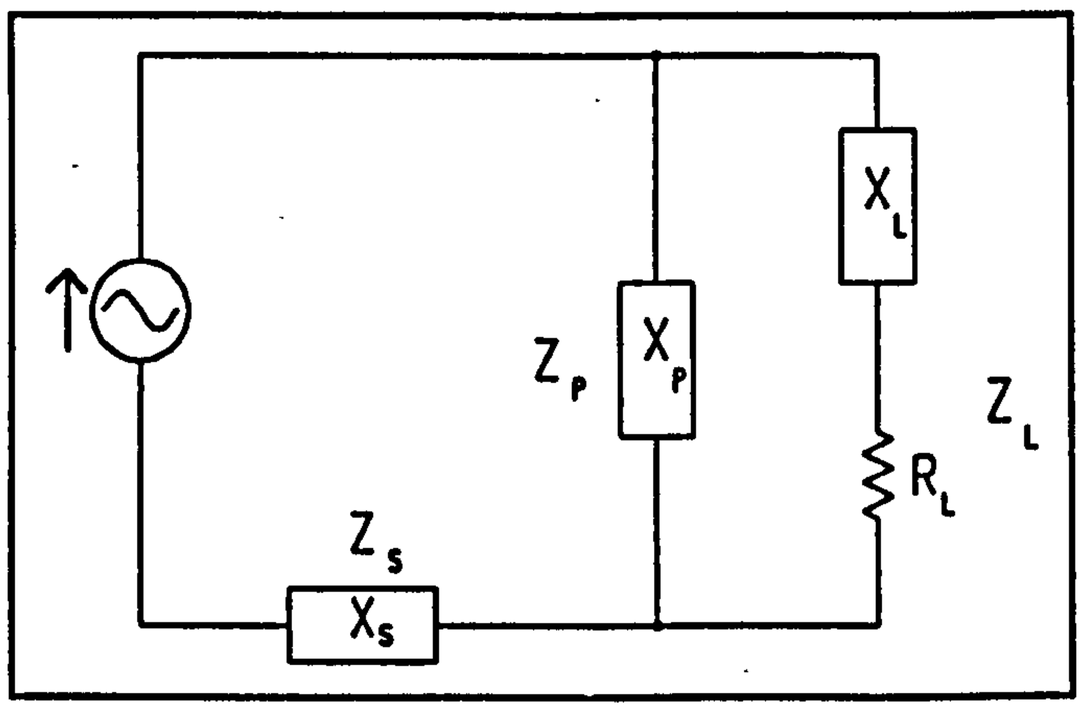


Figure 5.2 Simplified series-parallel resonant circuit with generalised reactance

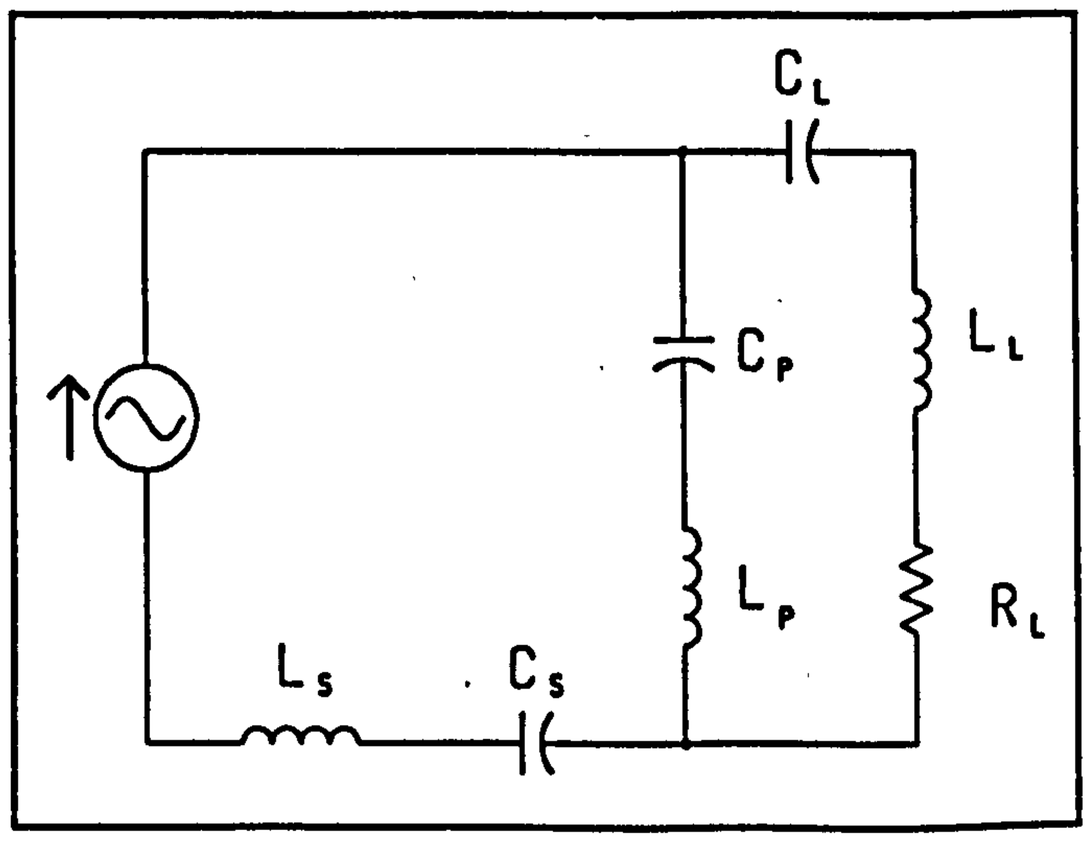


Figure 5.3 Simplified series-parallel resonant circuit with electrical components

5.2.2 Analysis of the circuit at resonance

The series-parallel resonant circuit is initially analysed using the simplified circuit, containing reactances, shown in Figure 5.2. At a particular frequency, the reactance values, X_s, X_p and X_L are given by

$$X_s = \omega L_s - \frac{1}{\omega C_s} \quad ,$$
$$X_p = \omega L_p - \frac{1}{\omega C_p} \quad , \qquad (5.1)$$
$$X_L = \omega L_L - \frac{1}{\omega C_L} \quad .$$

The total impedance, Z_{TOT}, (i.e. that presented to the generator) of the series-parallel resonant circuit in, Figure 5.2, is

$$Z_{TOT} = \frac{R_L X_P^2 \; + \; j \; (X_S R_L^2 + X_S X_L^2 + 2X_S X_P X_L + X_S X_P^2 + X_P X_L^2 + X_P^2 X_L + X_P R_L^2)}{R_L^2 + (X_L + X_P)^2} \quad . \qquad (5.2)$$

When Z_{TOT} is completely real the circuit is operating at a resonant frequency, so that the equivalent resistance, R_{TOT}, of the circuit, at resonance, is given by

$$R_{TOT} = \frac{R_L \; X_P^2}{R_L^2 + (X_L + X_P)^2} \quad . \qquad (5.3)$$

For a particular value of R_{TOT} at resonance, this equation can be rearranged to give a quadratic equation in X_P which can be solved to determine two values of X_P, viz.,

$$X_P = \frac{R_{TOT} X_L \pm \sqrt{R_{TOT} R_L (R_L^2 + X_L^2 - R_{TOT} R_L)}}{(R_L - R_{TOT})} \quad . \qquad (5.4)$$

These two values of X_P will each ensure that at a resonant frequency the overall circuit will have a specified equivalent resistance, R_{TOT}. A value of the reactance, X_s, of the series-leg can be found by setting the imaginary numerator of equation (5.2) to zero and rearranging to give

$$X_s = - \frac{X_P (X_P X_L + X_L^2 + R_L^2)}{R_L^2 + (X_L + X_P)^2} \quad . \qquad (5.5)$$

For each value of X_P, a unique value of X_S, can be calculated using equation (5.5)

It is possible, given certain component values, that the current flowing in the load can be greater than the current flowing in the series-leg (the switch current), the circuit exhibits current-gain. For the purposes of the subsequent analysis, the current-gain, A, is defined as the ratio of the current flowing in the load-leg to the current flowing in the series-leg, or source current, and is given by

$$A = \frac{I_{LOAD}}{I_s} = \frac{X_P(X_L+X_P) + jX_PR_L}{R_L^2+(X_L+X_P)^2} \quad . \tag{5.6}$$

This expression for current-gain is complex, indicating that the current through the load is not necessarily in phase with the source current. The magnitude of the current-gain is

$$|A| = \frac{|X_P|}{\sqrt{R_L^2+(X_L+X_P)^2}} \quad . \tag{5.7}$$

The analysis presented, so far, applies at any of the resonant frequencies of the circuit.

5.2.3 Analysis of the series-parallel resonant converter at any frequency

(i) The impedance equation in the frequency-domain

Figure 5.3 showed the simplified series-parallel-resonant circuit with all the relevant components. This model is used to analyze the series-parallel resonant circuit at a general frequency. The total impedance of the series-parallel resonant circuit in terms of the circuit components, at any angular frequency, ω, is

$$Z_{TOT} = \frac{(a_1\omega^5+a_2\omega^3+a_3\omega) + j\,(b_1\omega^6+b_2\omega^4+b_3\omega^2+b_4)}{(c_1\omega^5+c_2\omega^3+c_3\omega)} \; ; \tag{5.8}$$

where

$$a_1 = C_S\, C_P^{\,2}\, L_P^{\,2}\, C_L^{\,2}\, R_L;$$

$$a_2 = -2\, C_S\, C_P\, L_P\, C_L^{\,2}\, R_L;$$

$$a_3 = C_S\, C_L^{\,2}\, R_L;$$

$$b_1 = C_S C_P^2 C_L^2 (L_P L_L^2 + L_P^2 L_L + L_S L_L^2 + L_S L_P^2 + 2 L_S L_P L_L);$$

$$b_2 = C_S C_P^2 C_L(- L_P^2 - 2 L_S L_L - 2 L_P L_L - 2 L_S L_P)$$
$$+ C_S C_P C_L^2 (- L_L^2 - 2 L_P L_L - 2 L_S L_L - 2 L_S L_P)$$
$$+ C_P^2 C_L^2 (- L_L^2 - 2 L_P L_L - L_P^2)$$
$$+ C_S C_P^2 C_L^2 R_L^2 (L_P + L_S);$$

$$b_3 = C_S C_P (L_P C_P + L_S C_P + 2 L_P C_L + 2 L_L C_L + 2 L_S C_L - C_L^2 R_L^2)$$
$$+ C_S C_L^2 (L_L + L_S)$$
$$+ C_P^2 C_L (- C_L R_L^2 + 2 L_L + 2 L_P)$$
$$+ 2 C_P C_L^2 (L_L + L_P);$$

$$b_4 = - C_S C_P - C_S C_L - C_P^2 - 2 C_P C_L - C_L^2;$$

$$c_1 = C_S C_P^2 (C_L^2 L_L^2 + 2 L_P C_L^2 L_L + L_P^2 C_L^2);$$

$$c_2 = C_S C_P^2 (C_L^2 R_L^2 - 2 C_L L_L - 2 L_P C_L)$$
$$+ C_S C_P (-2 C_L^2 L_L - 2 L_P C_L^2);$$

$$c_3 = C_S C_P^2 + 2 C_S C_P C_L + C_S C_L^2.$$

This equation (5.8) is identical mathematically to equation (5.2) but explicitly shows the frequency dependence of impedance.

(ii) Calculation of the resonant frequencies of the series-parallel resonant circuit

Equation (5.8) gave the impedance of the series-parallel circuit in terms of angular frequency and component values. The circuit will exhibit resonance when the imaginary part of the impedance equation is zero;

$$\omega^6 + k_a \omega^4 + k_b \omega^2 + k_c = 0 \qquad (5.9)$$

where the coefficients are,

$$k_a = b_2 / b_1;$$

$$k_b = b_3 / b_1;$$

$$k_c = b_4 / b_1.$$

The imaginary part of the numerator of the impedance equation may be regarded as a polynomial of the sixth-order in ω, or a cubic polynomial in ω^2. It thus has three pairs of roots. The roots are the resonant frequencies of the circuit and are denoted by $\omega = \pm \omega_0$, $\omega = \pm \omega_1$ and $\omega = \pm \omega_2$. During a circuit design procedure, one of the

roots is usually set by choosing values of inductance and capacitance to give resonance at a particular frequency. This will be called the primary resonant frequency, ω_0. Thus given one root of the cubic, the remaining two roots, ω_1^2 and ω_2^2 can be calculated. Assuming $\omega_1^2 < \omega_2^2$ then the value of the primary resonant frequency ω_0 could be higher, lower or in-between the other two frequencies. Since the three roots of equation (5.9) are ω_0^2, ω_1^2, ω_2^2 the equation can be rewritten as

$$\omega^6 - \omega^4 (\omega_0^2+\omega_1^2+\omega_2^2) + \omega^2 (\omega_0^2\omega_1^2+\omega_1^2\omega_2^2+\omega_2^2\omega_0^2) - \omega_0^2\omega_1^2\omega_2^2 = 0 \qquad (5.10)$$

Equating the coefficients of equations (5.9) and (5.10) gives

$$\begin{aligned} k_a &= -\omega_0^2-\omega_1^2-\omega_2^2, \\ k_b &= \omega_0^2\omega_1^2+\omega_1^2\omega_2^2+\omega_2^2\omega_0^2, \\ k_c &= -\omega_0^2\omega_1^2\omega_2^2. \end{aligned} \qquad (5.11)$$

From this it can be seen that the coefficients k_a and k_c are always negative and the coefficient k_b is always positive. Combining the equations for k_a and k_c and eliminating ω_2^2 leads to a quadratic in ω_1^2

$$\omega_1^4 + \omega_1^2(k_a + \omega_0^2) - \frac{k_c}{\omega_0^2} = 0 \ , \qquad (5.12)$$

which has two solutions given by

$$\omega_1^2 = \frac{-(k_a + \omega_0^2) \pm \sqrt{(k_a + \omega_0^2)^2 + \dfrac{4\,k_c}{\omega_0^2}}}{2} \ . \qquad (5.13)$$

The third root of equation (5.10), ω_2^2, can now be found by rearranging the expression for k_a

$$\omega_2^2 = -k_a - \omega_0^2 - \omega_1^2 \ . \qquad (5.14)$$

Equations (5.13) and (5.14) therefore allow the two other resonant frequencies of the series-parallel resonant converter to be found given the primary resonant frequency of the circuit ω_0 and the component values.

(iii) Magnitude of impedance at resonant frequencies

Away from resonance at other frequencies, the impedance has real and imaginary parts and its magnitude is given by,

$$|Z_{TOT}| = \frac{[(a_1\omega^5+a_2\omega^3+a_3\omega)^2 + (b_1\omega^6+b_2\omega^4+b_3\omega^2+b_4)^2\ ]^{\frac{1}{2}}}{(c_1\omega^5+c_2\omega^3+c_3\omega)} \qquad (5.15)$$

Differentiating this expression gives

$$d\frac{|Z_{TOT}|}{d\omega} = \left[\left[\frac{2(a_1\omega^5+a_2\omega^3+a_3\omega)\ (5a_1\omega^4+3a_2\omega^2+a_3) + 2(b_1\omega^6+b_2\omega^4+b_3\omega^2+b_4)\ (6b_1\omega^5+4b_2\omega^3+2b_3\omega)}{2\ (c_1\omega^5+c_2\omega^3+c_3\omega)\ \sqrt{(a_1\omega^5+a_2\omega^3+a_3\omega)^2+(b_1\omega^6+b_2\omega^4+b_3\omega^2+b_4)^2}} \right] - \left[\frac{((a_1\omega^5+a_2\omega^3+a_3\omega)^2+(b_1\omega^6+b_2\omega^4+b_3\omega^2+b_4\omega)^2)^{\frac{1}{2}}\ (5c_1\omega^4+3c_2\omega^3+c_3)}{(c_1\omega^5+c_2\omega^3+c_3\omega)^2} \right] \right] \qquad (5.16)$$

This expression is zero at the points of maximum and minimum impedance. Setting the gradient equal to zero and solving for ω does not produce the same roots as the resonant frequencies. Thus, the maximum and minimum impedances of the series-parallel circuit do not occur at the defined resonance conditions. As further verification, the gradient of the impedance curve can be evaluated at the resonant frequencies and shown to be non-zero. At a resonant frequency the imaginary part of the impedance equation is equal to zero, i.e. $b_1\ \omega^6 + b_2\ \omega^4 + b_3\ \omega^2 + b_4 = 0$, and so the gradient of the impedance curve at a resonant frequency simplifies to

$$\frac{d|Z_{TOT}|}{d\omega} = \frac{(5a_1\omega^4+3a_2\omega^2+a_3)\ (c_1\omega^5+c_2\omega^3+c_3\omega) - (a_1\omega^5+a_2\omega^3+a_3\omega)\ (5c_1\omega^4+3c_2\omega^3+c_3}{(c_1\omega^5+c_2\omega^3+c_3\omega)^2} \qquad (5.17)$$

The fact that the maximum and minimum impedances of the series-parallel resonant circuit do not occur at the resonant frequencies, has serious implications for the use of frequency as a means of power control in series-parallel resonant circuits. In simple series resonant circuits as the frequency is taken away from resonance the power decreases. In the series-parallel circuit this is not necessarily the case. The power supplied by the circuit, to the load, depends on whether the gradient of the

impedance curve is greater, or less, than the rate-of-change of the cosine of the phase-angle.

(iv) Real average power in a series-parallel resonant circuit

Real average power in the circuit is given by equating the real part of the power

$$Power = V^2 \left[\frac{1}{Z_{TOT}}\right] \tag{5.18}$$

This can be evaluated for the series-parallel resonant circuit to give

$$Real\ average\ power = V^2 \left[\frac{(a_1\omega^5+a_2\omega^3+a_3\omega)\ (c_1\omega^5+c_2\omega^3+c_3\omega)}{(a_1\omega^5+a_2\omega^3+a_3\omega)^2-(b_1\omega^6+b_2\omega^4+b_3\omega^2+b_4\omega)^2}\right] \tag{5.19}$$

This function determines the variation in real average power with frequency for any series-parallel resonant circuit specified by the coefficients given by equation (5.8).

(v) Condition for valid solutions

The quadratic in ω_1^2 equation (5.12), has the solutions given in equation (5.13). ω_1^2 has a complex value if the term enclosed by the square root is negative. Complex frequencies have no physical significance in this context, and therefore conditions satisfying

$$(k_a + \omega_0^2)^2 + \frac{4\ k_c}{\omega_0^2} < 0 \tag{5.20}$$

are irrelevant to this study.

(vi) Conditions for the primary resonant frequency to be below, above or in between the other two resonant frequencies

It is observed that if at ω_0, any two of the three reactances, X_S, X_P or X_L are negative, then ω_0 is the lowest of the three resonant frequencies. Similarly when two of the three reactances are positive at ω_0, ω_0 is the highest resonant frequency. This is

explained by considering that, as the frequency increases, the reactances become dominated by the inductance, thus positive. The highest resonant frequency must therefore occur when two of the reactances are positive.

The conditions for ω_0 being the middle resonant frequency can be derived by finding the condition when two of the roots of the polynomial equation (5.10) are equal i.e. $\omega_0^2 = \omega_1^2$. Equation (5.10) then reduces to

$$2\omega_0^6 + k_a\,\omega_0^4 - k_c = 0 \quad . \tag{5.21}$$

Substituting for k_a and k_c gives

$$2\,b_1\,\omega_0^6 + b_2\,\omega_0^4 - b_4 = 0 \quad . \tag{5.22}$$

The coefficients b_1, b_2, and b_4 are defined in terms of the circuit components, whence, substituting and rearranging, a condition on C_S, the series capacitance, can be obtained which will make two of the resonant frequencies coincident, viz,

$$C_s = \frac{d_1}{d_0} \quad , \tag{5.23}$$

where,

$$\begin{aligned} d_1 = {} & -\omega_0^4 C_p^2 C_L^2 (L_L+L_P)^2 - (C_p+C_L)^2 + 2\omega_0^2 C_p^2 C_L (L_L+L_p) \\ & + 2\omega_0^2 C_p C_L^2 (L_L+L_P) - \omega_0^2 C_p^2 C_L^2 R_L^2 \end{aligned} \tag{5.24}$$

and,

$$\begin{aligned} d_0 = {} & 2\omega_0^6 C_P^2 C_L^2 (L_P L_L^2 + L_P^2 L_L) + \omega_0^4 (-C_P^2 C_L (L_P^2 + 2L_P L_L) - C_P C_L^2 (L_L^2 + 2L_P L_L) + C_P^2 C_L^2 R_L^2 L_P) + C_P + C_L \\ & + 2X_s \omega_0^5 C_P^2 C_L^2 (L_L+L_P)^2 - 2X_s \omega_0^3 C_p^2 C_L (L_L+L_P) - 2X_s \omega_0^3 C_P C_L^2 (L_L+L_P) + X_s \omega_0^3 C_P^2 C_L^2 R_L^2 \end{aligned} \tag{5.25}$$

As the roots, ω_1 and ω_2, are symmetric, equation (5.22) gives the condition for ω_0 being coincident with either ω_1 or ω_2. If two of the reactances, X_S, X_P, X_L are negative and one positive, when the condition occurs, then the value of C_S given in equation (5.23) will result in $\omega_0 = \omega_1$. If two of the reactances in the circuit are positive when the condition on C_S is met, then $\omega_0 = \omega_2$; i.e. the upper two resonant frequencies are coincident.

The different possible positions of the three resonant frequencies are summarised in Table 5.1. Also given in Table 5.1 is a reference number to an example circuit which exhibits these characteristics. These examples are considered in the subsequent section.

Position of ω_0, the primary resonant frequency	Conditions for this to occur	Example No.
ω_0 is the lowest resonant frequency i.e. $\omega_0 < \omega_1 < \omega_2$	Occurs when two of the reactances in the circuit at ω_0 are -ve and one is +ve	Example A
ω_0 occurs at the middle resonant frequency i.e. $\omega_1 < \omega_0 < \omega_2$	Occurs when $2\omega_0^6 + \omega_0^4 k_a - k_c < 0$ at all combinations of reactance	Example B
ω_0 is the highest resonant frequency i.e. $\omega_1 < \omega_2 < \omega_0$	Occurs when two of the reactances in the circuit are +ve and one is -ve	Example C
ω_0 is equal to ω_1	Occurs when $2\omega_0^6 + \omega_0^4 k_a - k_c = 0$ and two of the reactances in the circuit are negative	Example D
ω_0 is equal to ω_2	Occurs when $2\omega_0^6 + \omega_0^4 k_a - k_c = 0$ and two of the reactances in the circuit are positive	Example E

Table 5.1 Conditions for relative positions of the resonant frequencies

5.3 Example Circuits

Table 5.1 listed all the possible combinations of the positions of the resonant frequencies of a series-parallel resonant circuit and the conditions under which each combination occurred. Table 5.2 lists the component values of the five example circuits (A-E) that were designed to confirm and illustrate the conditions calculated in section 5.2.2.

Example name	Figure number	Component Values	X_L (Ω) at 90 kHz	X_s (Ω) at 90 kHz	X_p (Ω) at 90 kHz	R_{ser} (Ω)	f_0 (kHz)	f_1 (kHz)	f_2 (kHz)
A	Figure 5.4	L_s =2.9 μH C_s =0.4 μF L_p =0 μH C_p =69 nF L_L =7 μH C_L =∞ μF	4	-2.67	-24.94	7.5	90	184	442
B	Figure 5.5	L_s =8 μH C_s =1 μF L_p =17 μH C_p =.1 μF L_L =7 μH C_L =∞ μF	4	2.67	-8	7.5	90	48	107
C	Figure 5.6	L_s =21.6μH C_s =1 μF L_p =20 μH C_p =.1 μF L_L =22 μH C_L =∞ μF	12.44	10.45	-6.6	3.75	90	25	76
D	Figure 5.7	L_s =19 μH C_s =0.23μF L_p =10 μH C_p =0.1 μF L_L =8.56μH C_L =1.1μF	4	2.75	-7.96	7.5	90	90	101
E	Figure 5.8	L_s =10.9μH C_s =0.20μF L_p=17.2μH C_p =1 μF L_L =8.56μH C_L =0.2μF	-4	-2.75	7.964	7.5	90	68	90

Table 5.2 Numerical Values for Examples A to E

Figure 5.4 shows the frequency characteristic for Example A, obtained when the primary resonant frequency was specified as the lowest resonant frequency of the circuit. The primary resonant frequency, f_0, was 90 kHz. Table 5.2 shows that the impedance of two of the legs of the circuit at this frequency are negative and one is positive, confirming the condition given in Table 5.1.

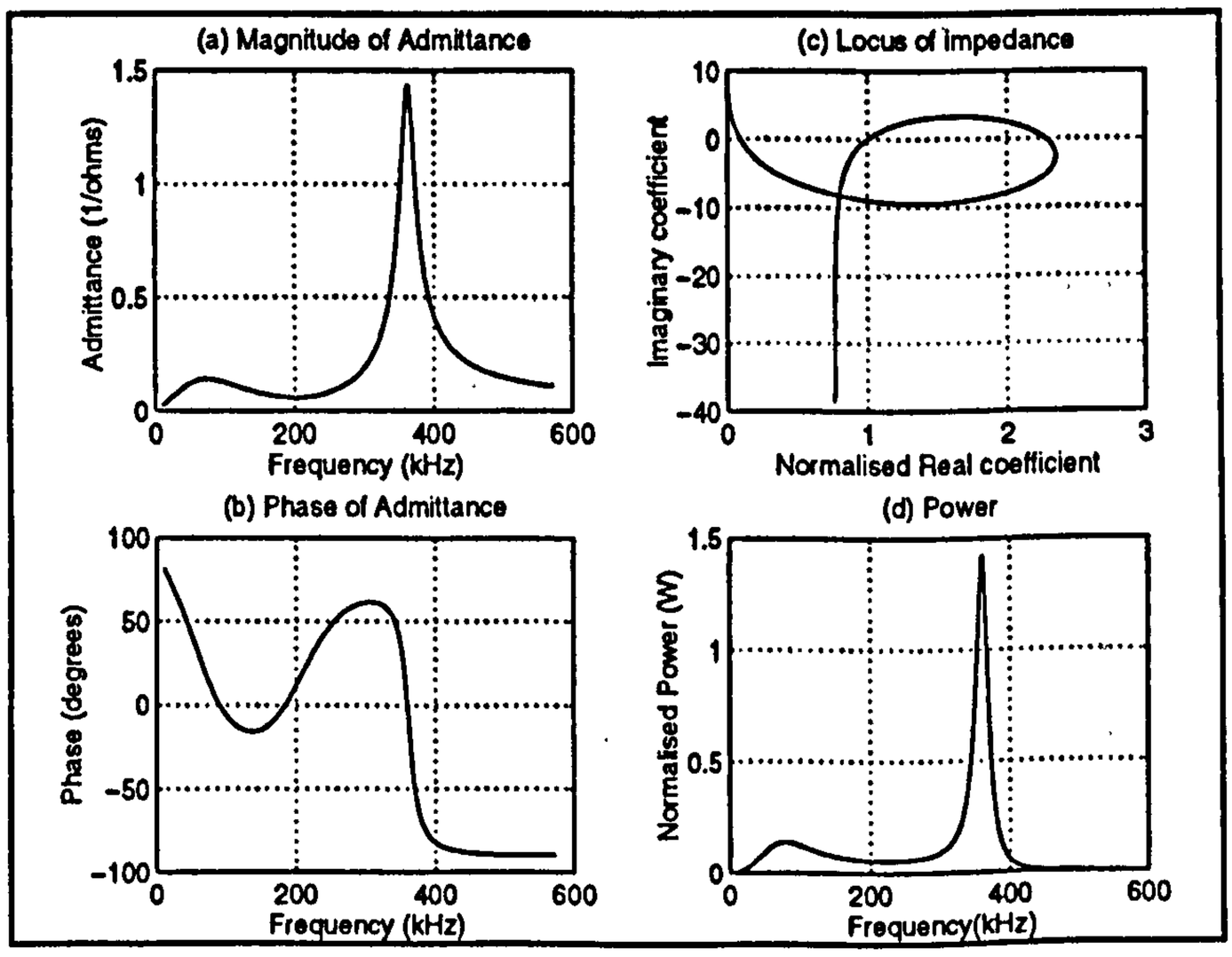


Figure 5.4 Example A Primary resonant frequency is the lowest resonant frequency i.e. $\omega_0 < \omega_1 < \omega_2$

In Figure 5.4, plots (a) and (b) show respectively the magnitude of the admittance and phase of the admittance (in degrees) plotted against frequency as calculated from equation (5.8). The primary resonant frequency (90 kHz) in plot (b) occurs when the phase plot passes through zero for the first time. The circuit looks capacitive below this frequency, is then inductive (negative phase angle of admittance) up to f_1 = 184 kHz when the circuit again appears capacitive until the upper resonant frequency (442 kHz) is exceeded. The locus of the impedance is shown in plot (c). In plot (c) the real part of the impedance was normalised by dividing the real part by its value at 90 kHz. This locus of the impedance shows that the impedance crosses the real axis three times corresponding to the three resonant frequencies of the circuit. Plot (d) shows the frequency variation of the power per unit voltage squared (calculated from equation 5.19).

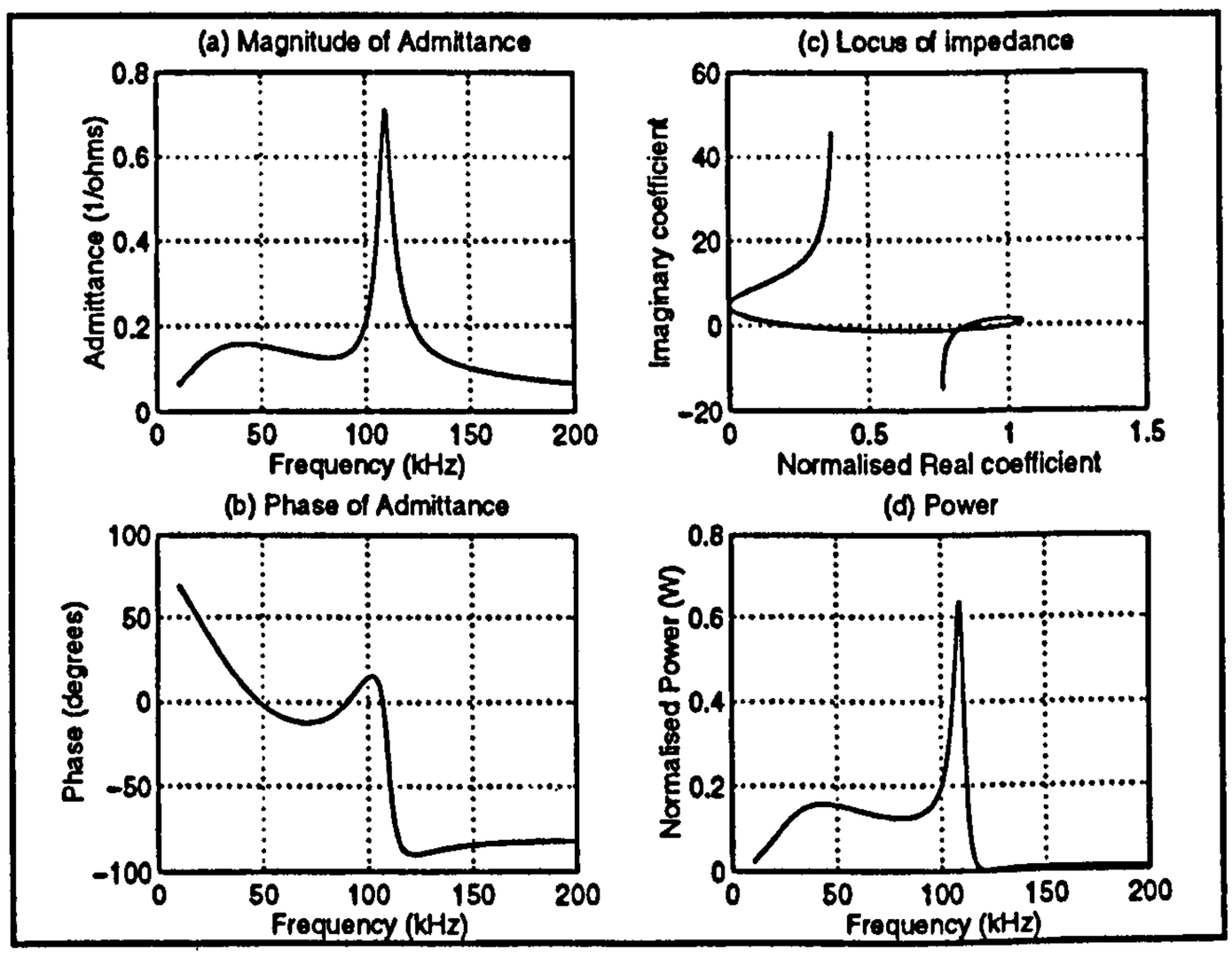


Figure 5.5 Example B Primary resonant frequency is the middle resonant frequency

i.e. $\omega_1 < \omega_0 < \omega_2$

Figure 5.5 shows the results for circuit Example B in which the primary resonant frequency at 90 kHz has been selected as a middle resonant frequency of the circuit. The condition in Table 5.1 is met. Table 5.2 gives the values of the impedance of each leg of the circuit at the primary resonant frequency. Plot (b) shows the primary resonant frequency occurring as the phase plot passes through zero for the second

time. Below the primary resonant frequency the circuit looks inductive and above this frequency the circuit looks capacitive. In plot (c) the locus of the impedance is seen to touch the imaginary axis of the plot. At this frequency the circuit appears totally reactive. This occurs when the impedance of the parallel-leg is zero (L_p and C_p are in resonance) and the load-leg is effectively short circuited. It should be notoed that this is not a resonant frequency of the whole circuit.

Figure 5.6 shows the results for circuit Example C in which the primary resonant frequency of 90 kHz has been selected as an upper resonant frequency of the circuit. Plot (b) shows the primary resonant frequency as the final point at which the phase plot crosses the x-axis. As in Example B there is a frequency, in this case just above 90 kHz, at which the circuit is totally imaginary and the load is shorted out by the parallel leg of the circuit.

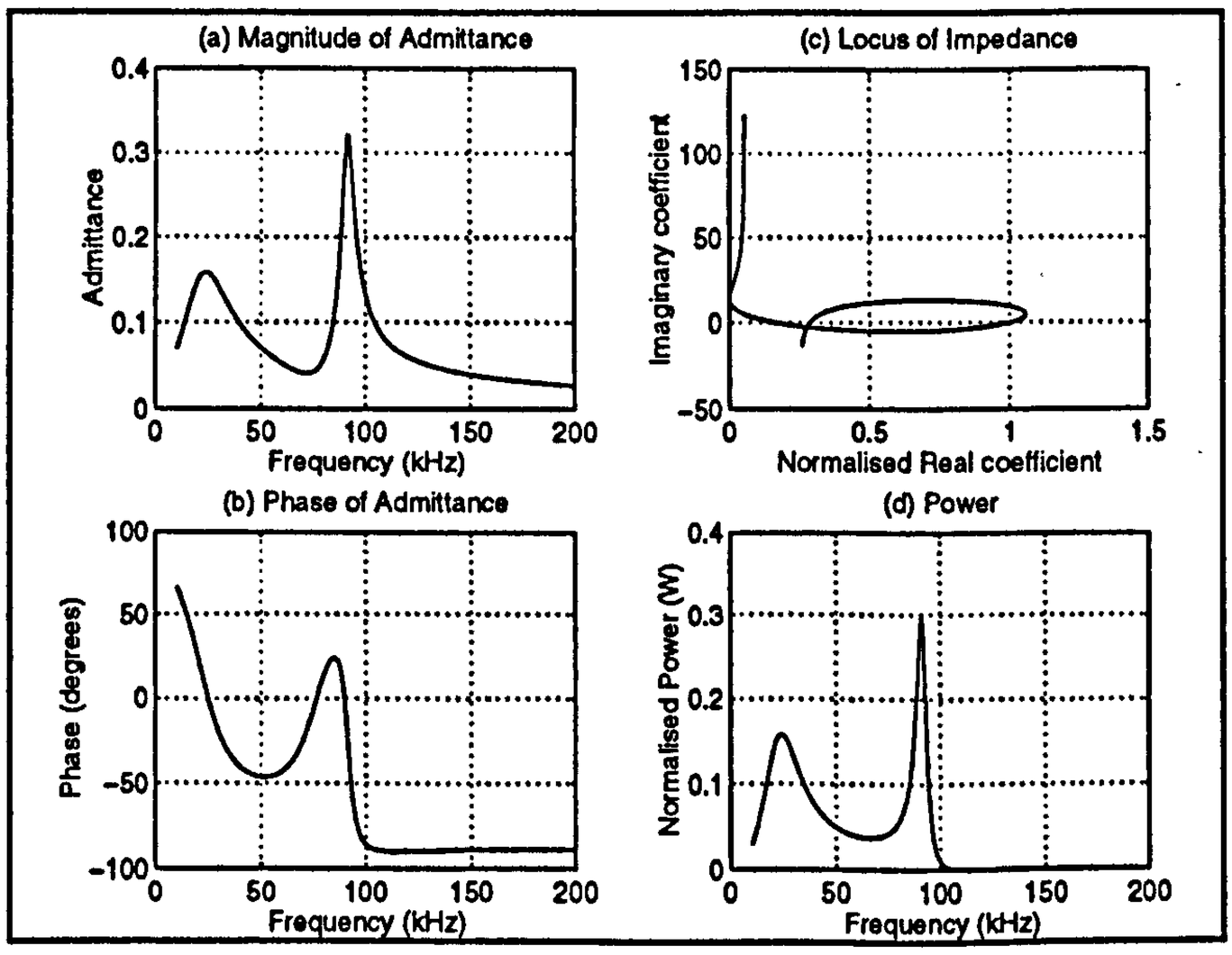


Figure 5.6 Example C Primary resonant frequency is the upper resonant frequency i.e. $\omega_1 < \omega_2 < \omega_0$

Close examination of the phase and power plots of each example circuit A-C has confirmed that the peak of the power is not always coincident with the resonant frequencies of the circuit. As stated earlier this has important implications if power control in the converter is achieved by moving the frequency of operation of the converter away from resonance. The power levels in the circuit could increase rather

than decrease as might be expected.

Example D was a circuit which was designed such that the two lower resonant frequencies were coincident. This was achieved by applying the condition given in Table 5.1. Figure 5.7, plot (b) shows that the actual design which has been achieved practically has all three resonant frequencies coincident. This could be of significant practical use. A series-parallel resonant circuit with its inherent short-circuit and open-circuit protection capability can be designed to look like a resonant circuit with a single resonant frequency. Concerns about exciting harmonics of other resonant frequencies (see section 5.8) would be removed by using such a design.

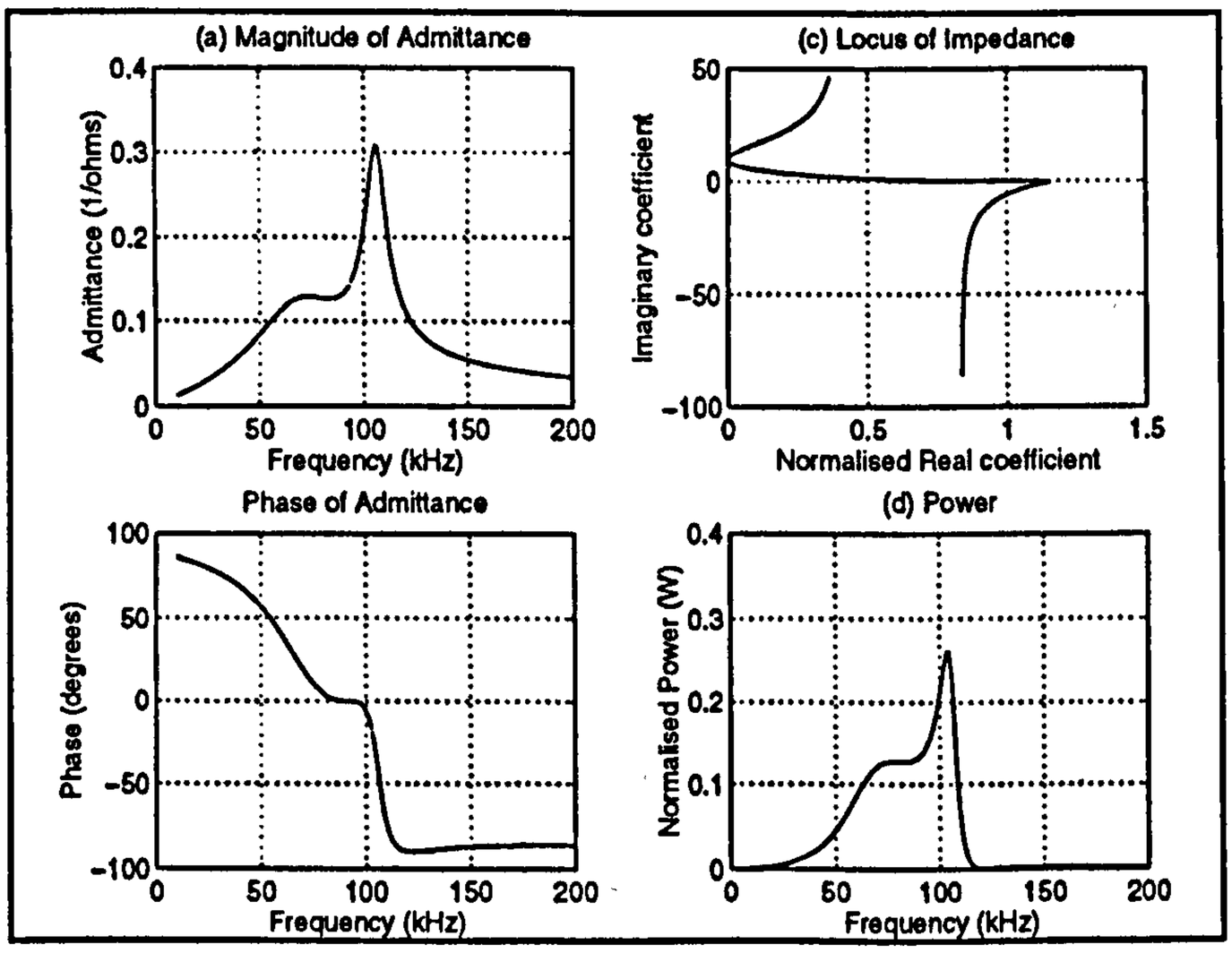


Figure 5.7 Example D Primary resonant frequency is coincident with lower resonant frequency i.e. $\omega_0 = \omega_1$

Figure 5.8 shows the results for example circuit E in which the primary resonant frequency and the upper resonant frequency are coincident. This is most evident from the plot of the phase of the admittance (plot (b)) in which the curve is seen to touch rather than cross the x-axis at the frequency corresponding to the co-incident middle and upper resonant frequencies.

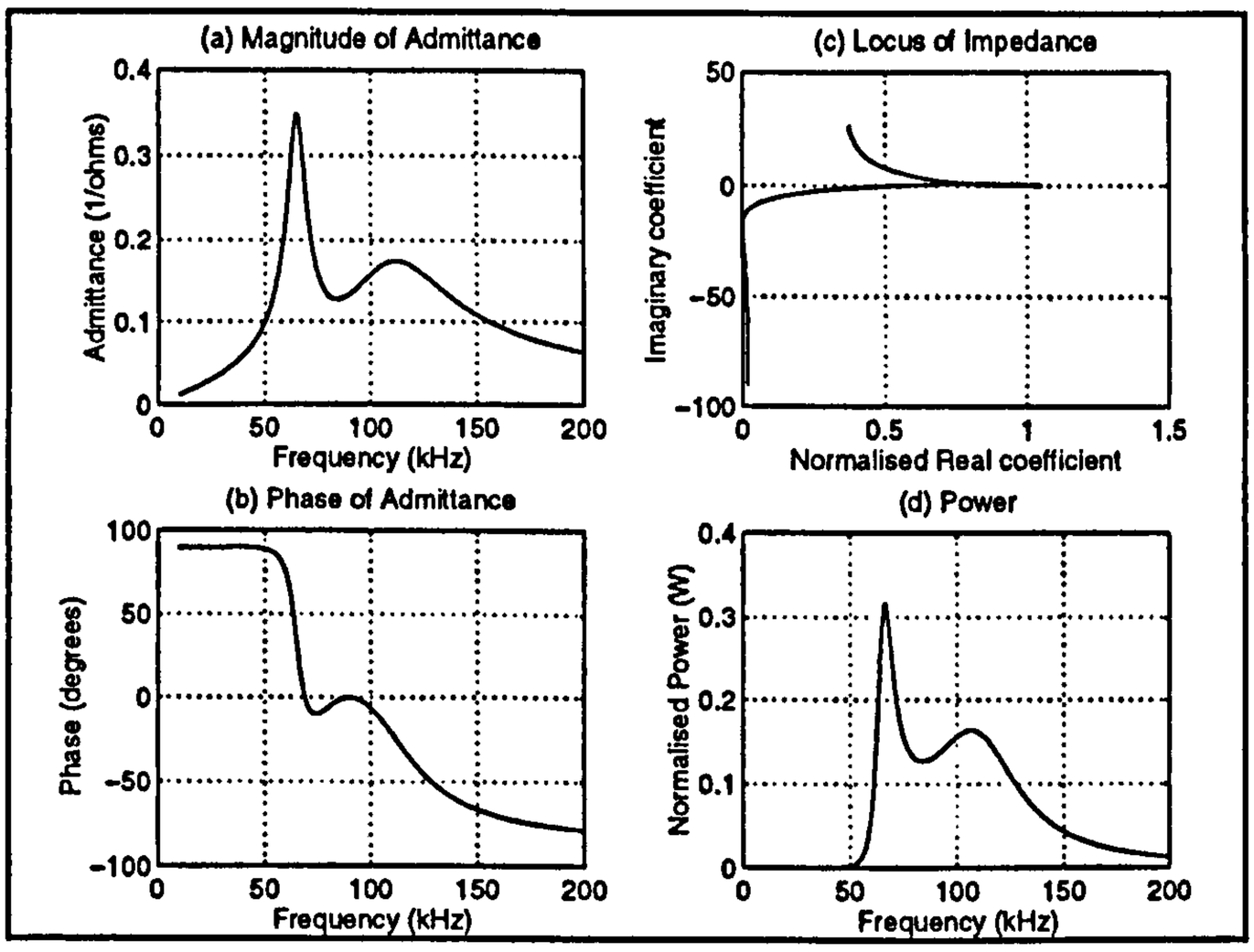


Figure 5.8 Example E Primary resonant frequency is coincident with upper resonant frequency i.e. $\omega_0 = \omega_2$

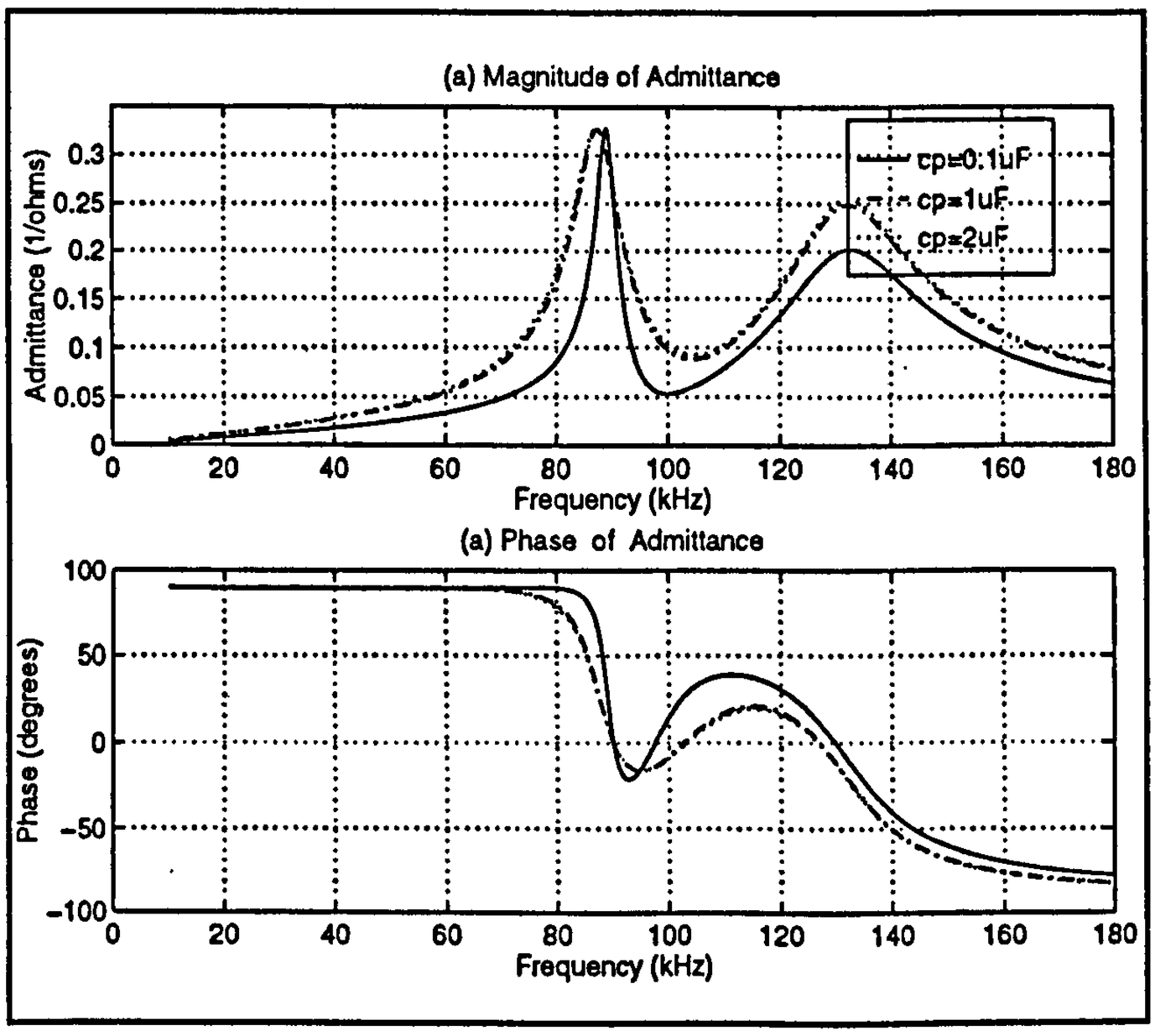
Table 5.3 gives the component values of two further example circuits F and G. These two examples were chosen to demonstrate the effect of varying, firstly, the value of the parallel-leg capacitance C_P and, secondly, the series-leg capacitance C_S. Example F was a circuit in which the primary resonant frequency of the circuit was the lowest resonant frequency.

Example name	Figure number	Component Values	X_L (Ω) at 90 kHz	X_S (Ω) at 90 kHz	X_P (Ω) at 90 kHz	R_{int} (Ω)	f_0 (kHz)	f_1 (kHz)	f_2 (kHz)
F	Figure 5.9	Average values C_S and C_P varied L_S =12.7μH C_S =0.1 μF L_P=14.8μH C_P =69 nF L_L =22μH C_L =71 nF	-12.44	-10.45	6.61	3.75	90 C_P=0.1nF C_P=1μF C_P=2μF C_P=50nF C_P=100nF C_P=150nF	 97.5 102 103 complex 102 100	 130 127 126 complex 127 150
G	Figure 5.10	Average values C_S and C_P varied L_S =21.6μH C_S =1 μF L_P=19.6μH C_P =0.1μF L_L =22μH C_L =∞ μF	12.44	10.45	-6.61		90 C_P=10nF C_P=100nF C_P=190nF C_P=10nF C_P=100nF C_P=190nF	 59 61 complex 59 24 18	 88 72 complex 88 88 88

Table 5.3 Component values for Examples F and G

Table 5.3 lists the average value of the components and the range of values of C_s and C_p over which the characteristics of the example circuits were evaluated. The primary resonant frequency of 90 kHz was maintained by adjusting the values of inductance in each leg of the circuit to maintain the same impedance values at 90 kHz.

Figure 5.9(a) shows the effect of varying the capacitance, C_p, on the magnitude and phase of the admittance in the circuit of Example F. Figure 5.9(b) shows the effect of varying the capacitance, C_s, in the same circuit.



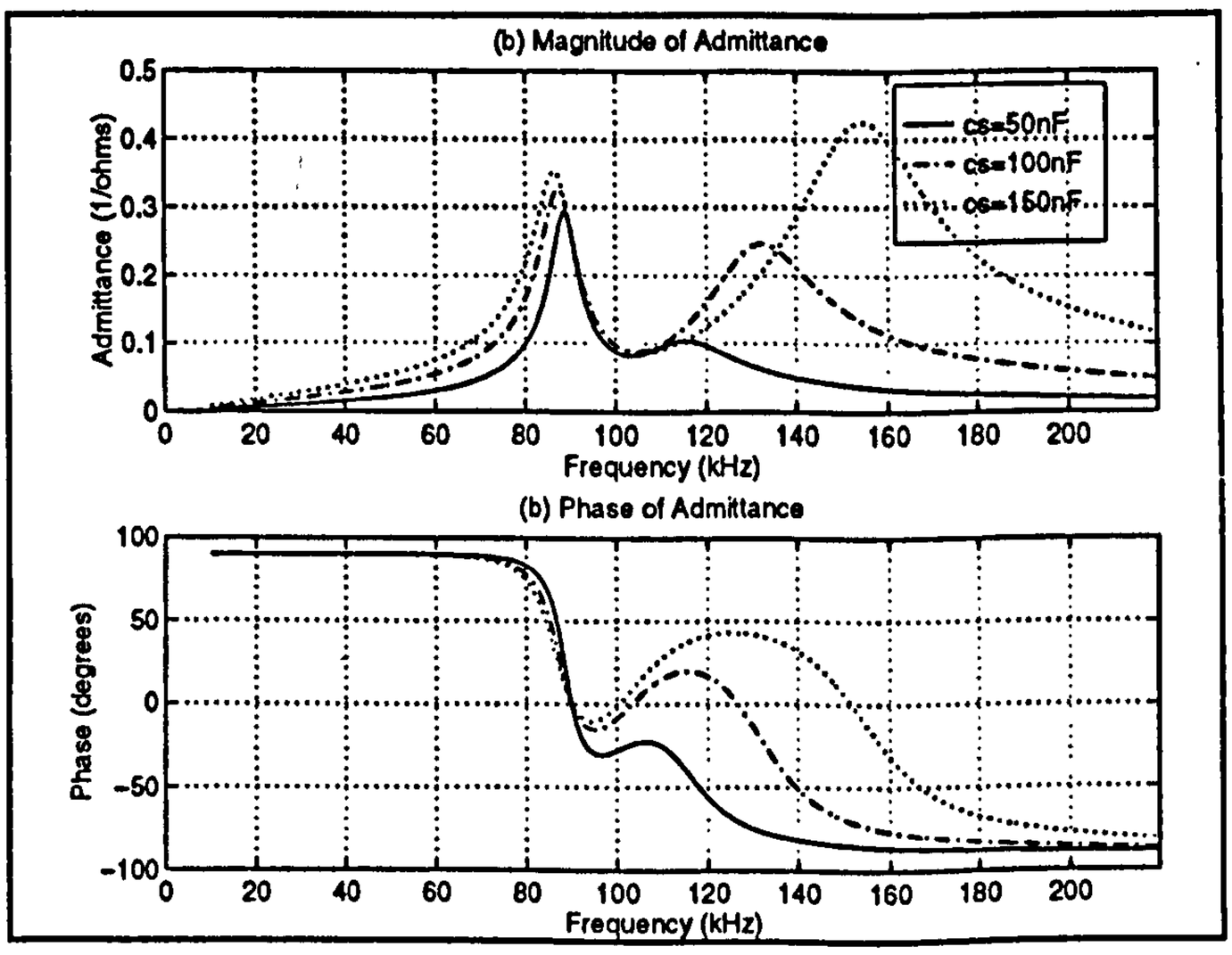


Figure 5.9 Example F (a) Varying C_p and (b) varying C_s when the primary resonant frequency is the lowest resonant frequency of the circuit

Increasing the value of the parallel capacitance causes the lower peak value of admittance to shift in frequency only slightly from 88.8 kHz to 87 kHz. Its magnitude does not change significantly. The magnitude of the minimum value of admittance (between the two maximum values) increases and the frequency at which it occurs shifts from 99.8 kHz to 104.6 kHz. The upper peak in the admittance plot changes only in magnitude and not in frequency.

In Figure 5.9(b) the magnitude of admittance plot shows that the frequencies at which the peaks in admittance occur diverge as C_s increases. The magnitude of the admittance at the peaks also increases. The phase of admittance shows the substantial shift in the upper resonant frequency of the circuit. At the lowest value of C_s, the two upper resonant frequencies of the circuit are complex, a condition not physically meaningful. As the value of C_s increases the two upper resonant frequencies of the circuit become real and the phase plot now passes through zero three times.

The results presented in Figure 5.10 are for the circuit in Example G which has the primary resonant frequency as the highest resonant frequency of the circuit. In Figure 5.10(a) increasing the value of the parallel capacitance causes the lower peak value of admittance to decrease in magnitude and shift to a lower resonant frequency. The upper admittance peak decreases in magnitude and the width of the peak increases. The phase of the admittance in Figure 5.10(b) shows how the two lower resonant frequencies become complex and therefore not physically meaningful, as the value of the parallel capacitor is increased. In Figure 5.10(b) increasing the value of the series capacitor C_s shifts the lower peak of admittance to a lower resonant frequency. The upper peak does not shift although the magnitude of the peak increases slightly. All the roots remain real.

In both the example circuits F and G a variation in the value of C_s causes a greater shift in the location of the resonant frequencies of the circuit than a variation in the value of C_p. This is because all the current flows through the series leg of the circuit. Variation in the value of C_p has less effect as the current in the circuit is divided between the parallel leg and the load leg of the circuit.

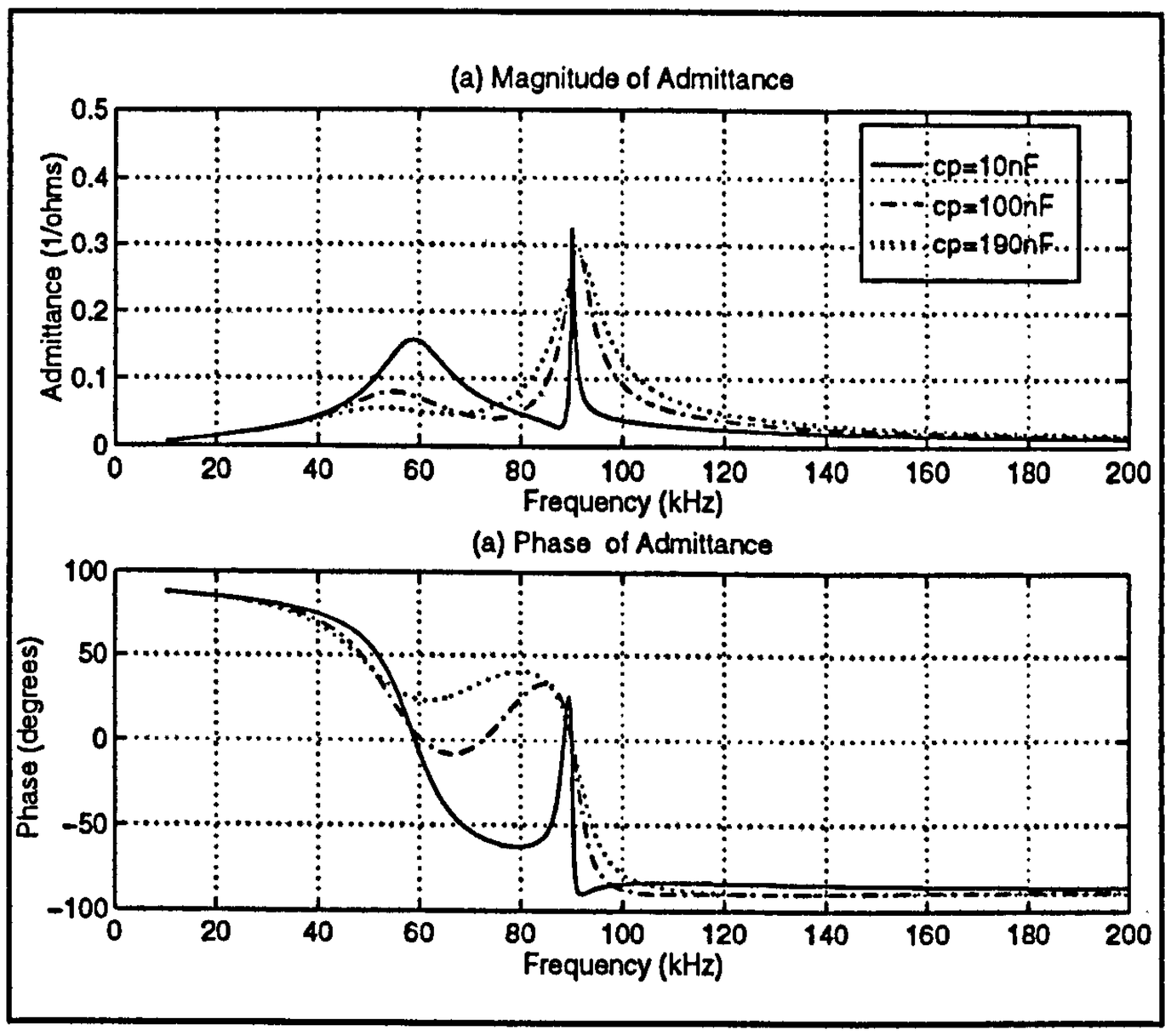


Figure 5.10 Example G (a) Varying C_P and (b) varying C_S when the primary resonant frequency is the highest resonant frequency of the circuit

5.4 Verification of Examples

The results illustrated by Figures 5.4 - 5.10 were all obtained directly from the mathematical expressions derived in the early part of this Chapter. In order to verify the mathematics and to illustrate the results of exciting these circuits in the time

domain, further simulation was done in Saber.

Example C was simulated in both the frequency- and time-domain. A frequency sweep confirmed the location of the resonant frequencies of the circuit as calculated from equations (5.12) and (5.13). In the time domain, a sine-wave voltage was used to excite the resonant circuit. At the three resonant frequencies of the circuit, 25 kHz, 76 kHz and 90 kHz, the switch current was sinusoidal and in phase with the voltage. The circuit was then driven as part of the half-bridge power converter shown in Figure 5.1. The resonant circuit was effectively excited by a square-wave voltage in this configuration. Figure 5.11 shows the voltage across the resonant circuit, V_S, the current, I_S, through the power switches and the load current, I_{LOAD}. Figure 5.11(a) shows the converter operating at 25 kHz, the lowest resonant frequency of the circuit. The switch current is in phase with the voltage.

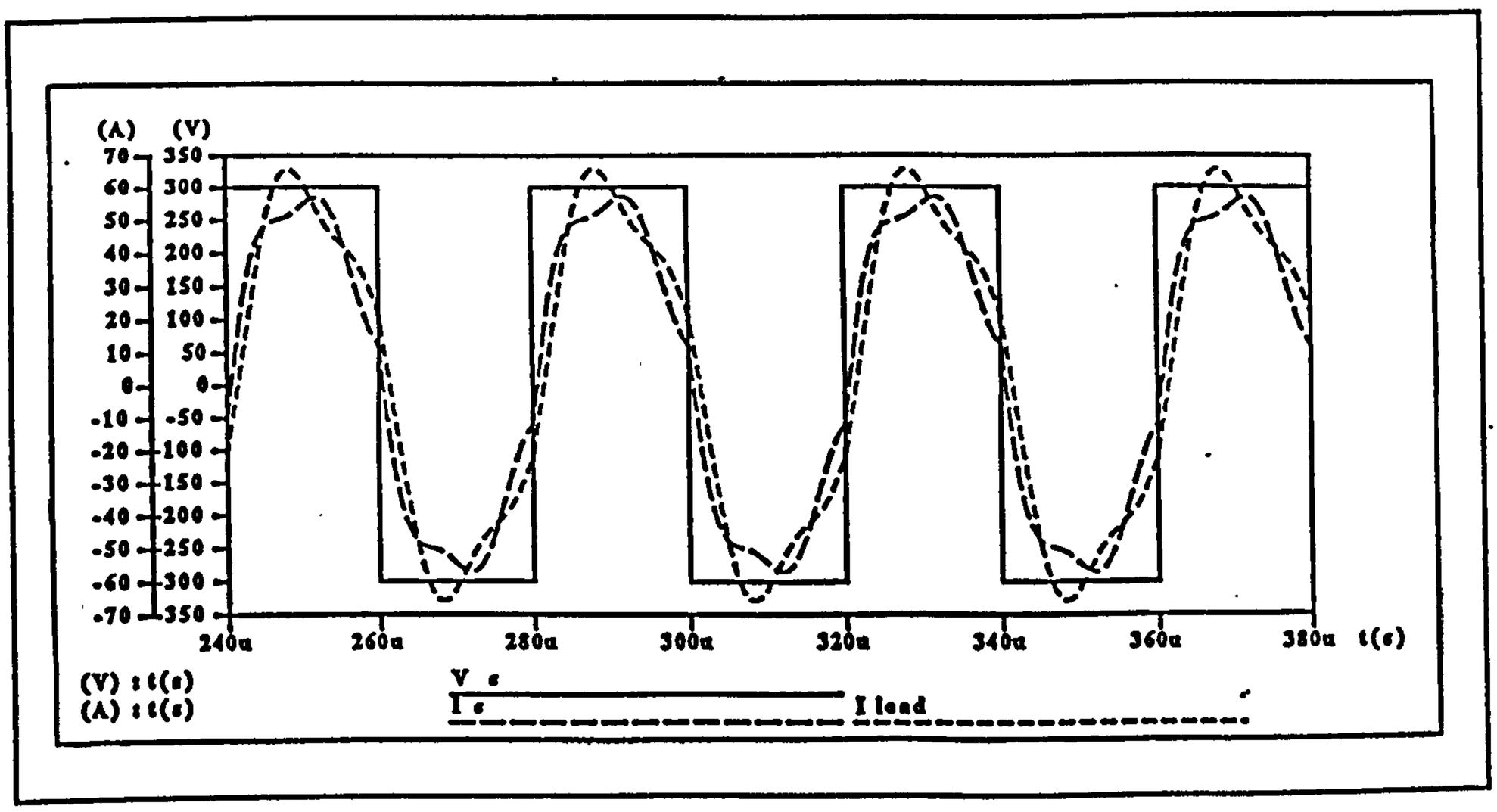


Figure 5.11(a) Example C Square-wave excitation at 25 kHz, the lowest resonant frequency of the circuit

Both the switch current and load current are flowing in the same sense. The load current is slightly larger than the switch current i.e. there is current-gain. The switch current has a high frequency ripple. A Fourier analysis of the switch current revealed that the ripple had components at 75 kHz, 90 kHz and 125 kHz. The frequency components at 75 kHz and 125 kHz correspond to the third and fifth harmonics of the

the driving square-wave voltage. The component at 90 kHz results from the excitation of a natural resonant frequency of the circuit.

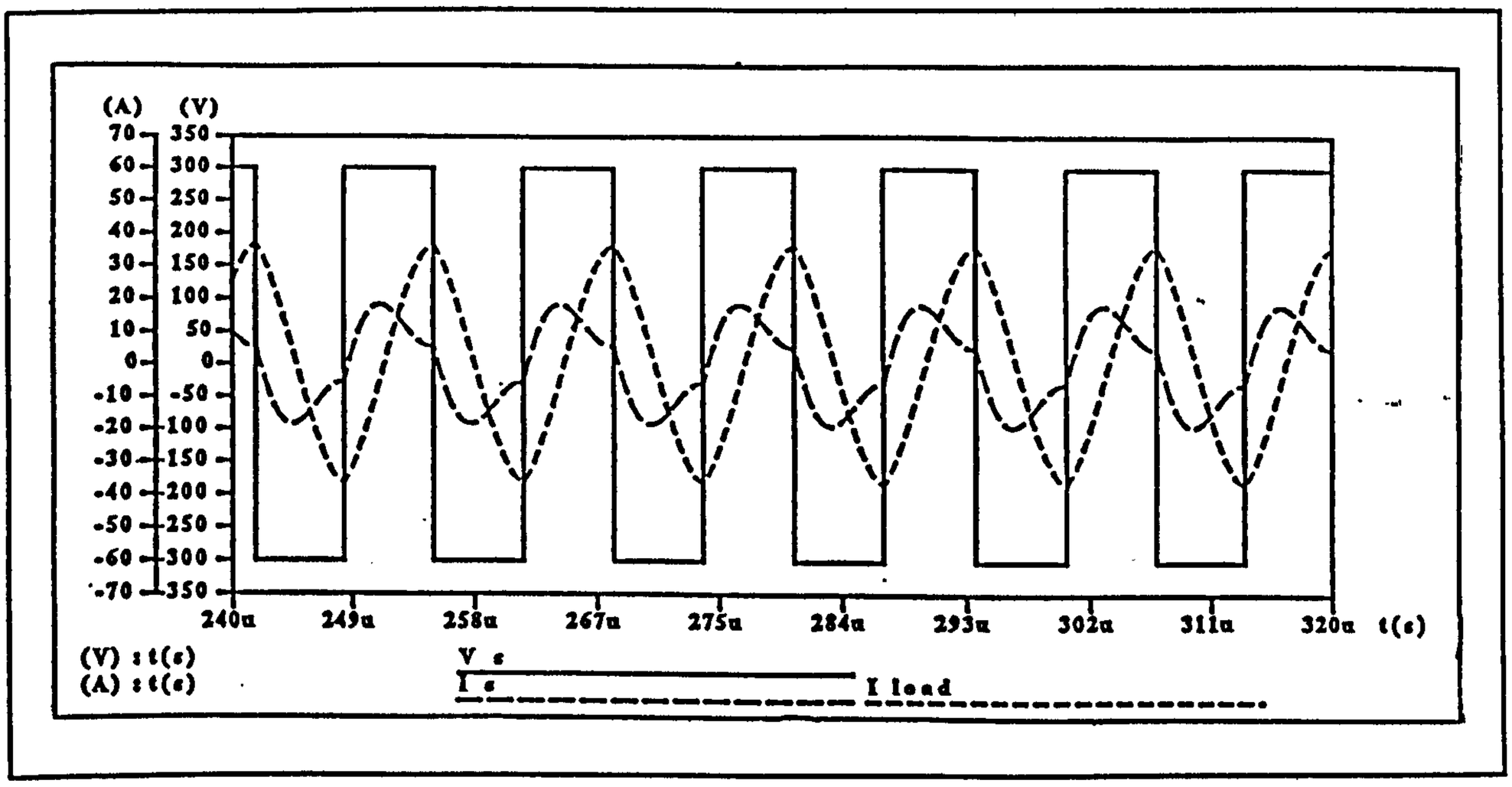


Figure 5.11(b) Example C Square-wave excitation at 76 kHz,
the middle resonant frequency of the circuit

Figure 5.11(b) shows the results obtained for the converter operating at 76 kHz, the middle resonant frequency of the resonant circuit. A Fourier analysis of the switch current, in this example, showed the current contains frequency components at the harmonics of the voltage and significant components at 25 kHz and 90 kHz, the highest and lowest resonant frequencies of the circuit. Unlike the result obtained while operating at the lowest resonant frequency of the circuit, only a fraction of the 25 kHz oscillation is completed before the supply voltage changes to the opposite polarity.

Figure 5.11(c) shows the converter operating at the upper resonant frequency of the circuit. The switch current is in phase with the voltage and it is sinusoidal. The load current flows in the opposite sense to the switch current. The switch current is larger in magnitude than the load current.

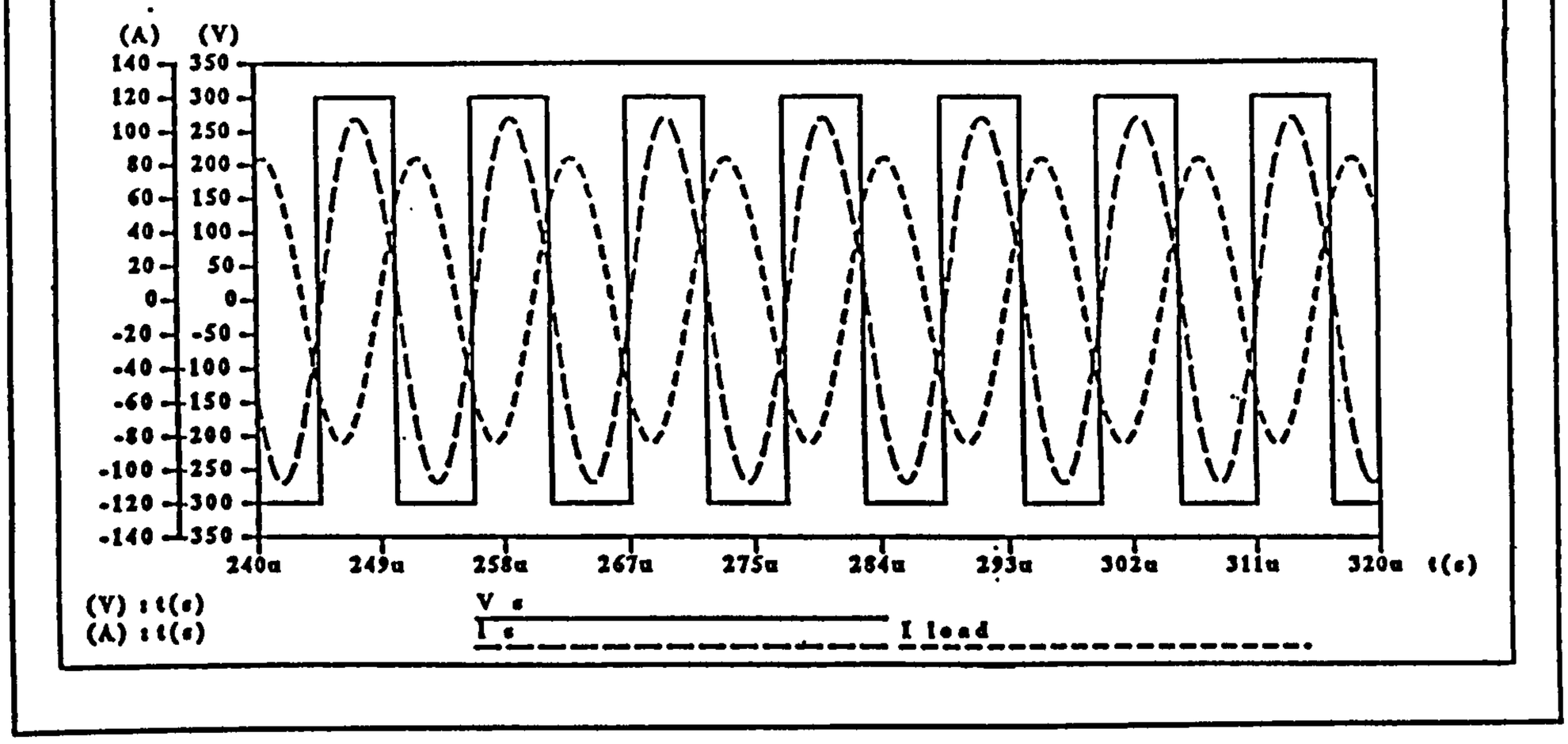


Figure 5.11(c) Example C Square-wave excitation at 90 kHz,
the highest resonant frequency of the circuit

In Figure 5.11(a), the switch current exhibited a high frequency ripple of 76 kHz and 90 kHz superimposed on the fundamental current at 25 kHz. The magnitude of the high-frequency ripple is dependent on the closeness of the upper resonant frequency to a harmonic of the fundamental frequency of the square-wave voltage. In the case of the circuit in example 3, with resonant frequencies at 25 kHz, 76 kHz and 90 kHz, the third harmonic of the square-wave exciting voltage occurs at 75 kHz and the fifth at 125 kHz. The upper resonant frequency of 90 kHz occurs midway between these two frequencies and therefore the voltage harmonics are attenuated by the circuit.

In Figure 5.12 the circuit given in example A is driven at 90 kHz, its lowest resonant frequency. Figure 5.12(a) shows the circuit excited by a pure sine wave at 90 kHz. The currents flowing in the circuit are sinusoidal and practically in phase. Figure 5.12(b) shows the currents in the circuit when the source voltage is a square-wave at 90 kHz. In this circuit the upper resonant frequency , 450 kHz, is coincident with the fifth harmonic of the driving square-wave voltage. The 450 kHz ripple in the current waveforms causes considerable distortion.

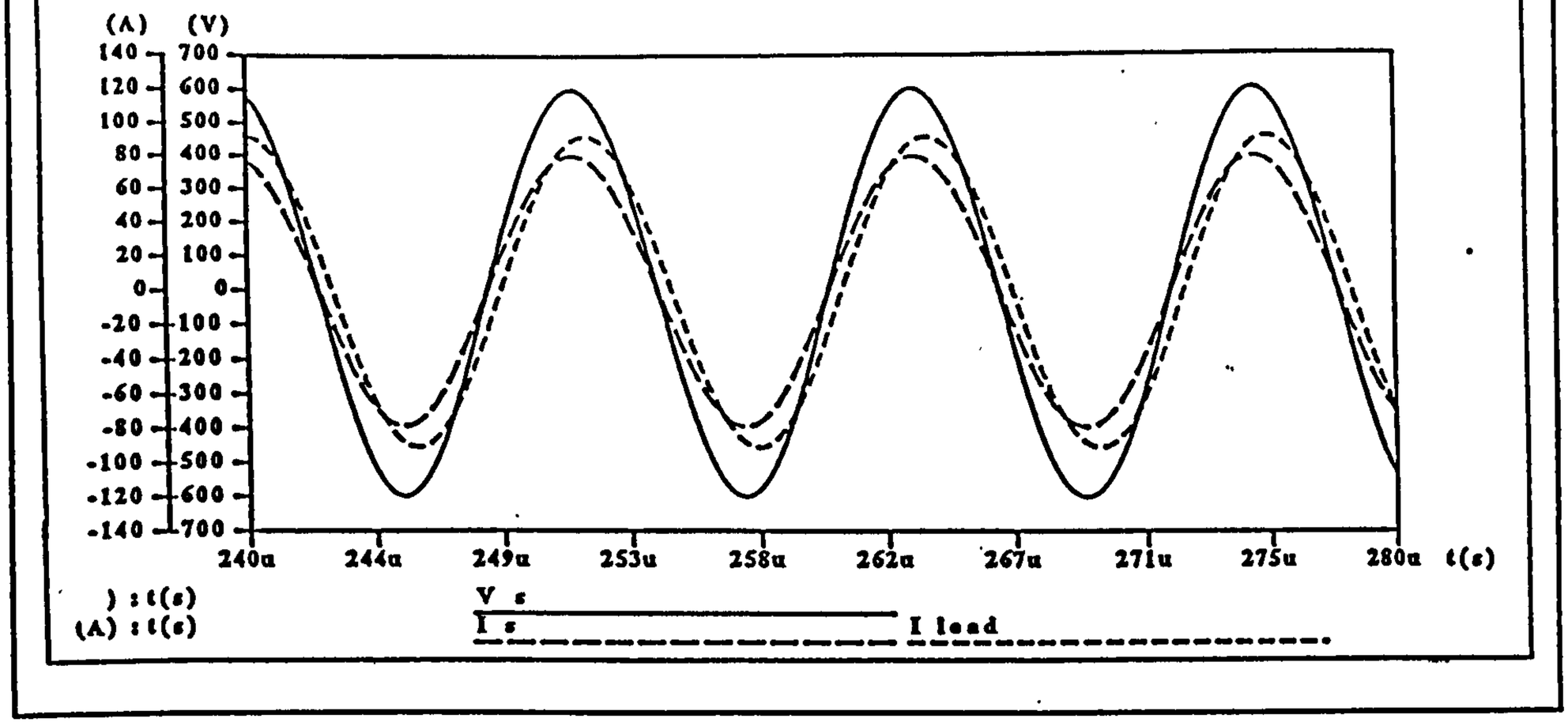


Figure 5.12(a) Example A Sine-wave excitation at 90 kHz,
the lowest resonant frequency of the circuit

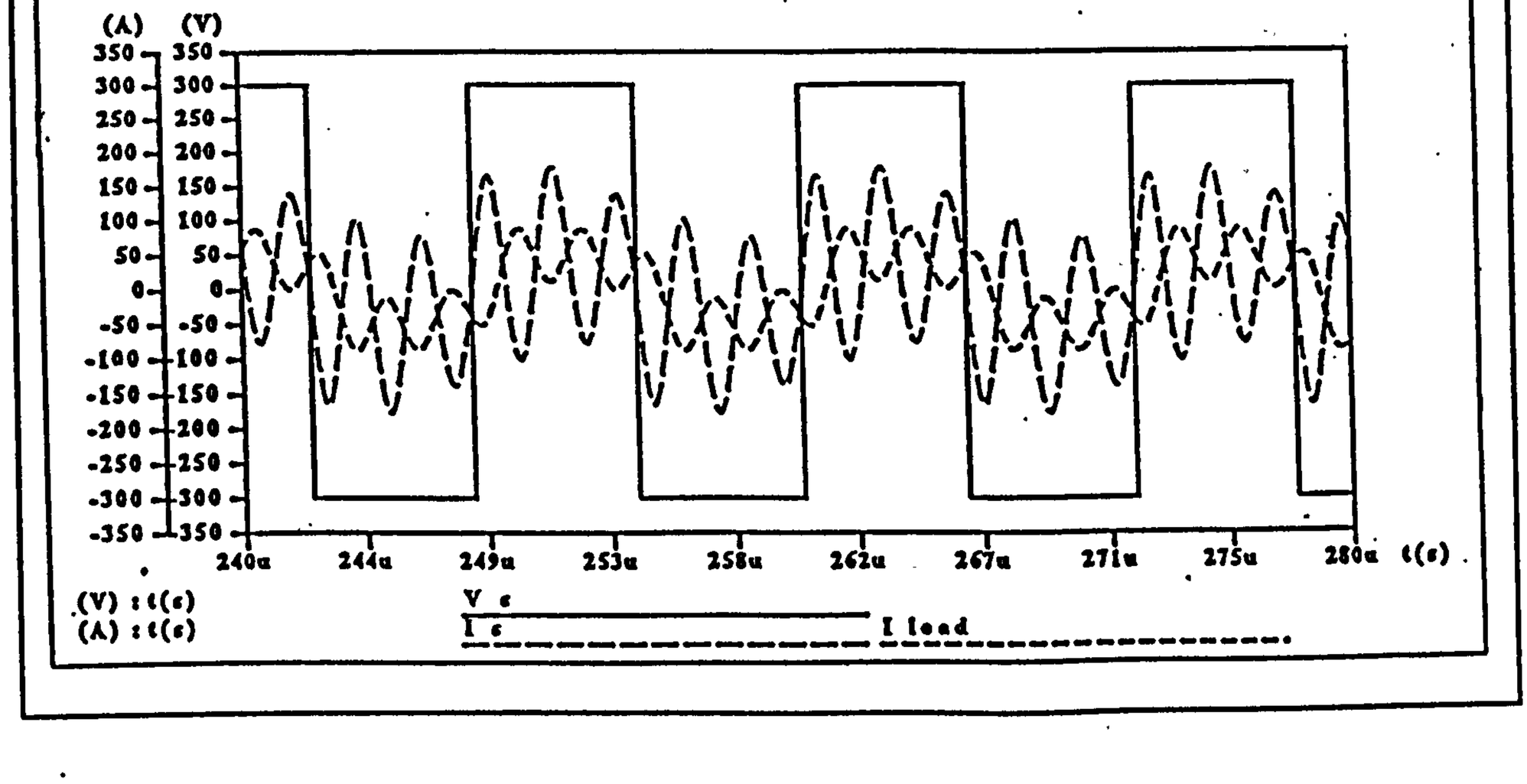


Figure 5.12(b) Example A Square-wave excitation at 90 kHz,
the lowest resonant frequency of the circuit

5.5 Calculation of component values for a series-parallel load-resonant converter.

The analysis so far in this chapter has allowed the frequency characteristics of a series-parallel resonant circuit to be predicted providing the component values are known. In order to design a series-parallel resonant circuit it is important to be able to specify the shape of the frequency characteristics to staisfy the requirements of the application and to then calculate the component values required for the resonant circuit. This way the the resonant frequencies of the circuit could be specified in the design stage. To achieve this the sequence for using the earlier equations needed to be reversed. The designer would specify the three resonant frequencies of the circuit, the referred resistance of the load R_L, the load inductance, L_L, the load capacitance, C_L and the desired resistance of the complete circuit, R_{TOT} at the resonant frequency of operation. It will be shown below that solution of the appropriate system of equations allow, the remaining four component values to be calculated.

Four independent equations are required in order to determine values for C_S, L_S, C_P, L_P so that the resulting circuit has the desired characteristic. Three equations are derived from the impedance, Z_{TOT}, of the circuit given in equation (5.8). The coefficients of the imaginary part of this expression are shown to be related to the resonant frequencies of the circuit in equations (5.11). To simplify the mathematics, the reactance of each leg of the circuit is introduced into the equations replacing the capacitive terms. The reactance of each leg of the circuit is related to the component values in that leg and the operating frequency as shown in equation (5.1).

Equations (5.1) were rearranged to give equations for C_S, C_P and C_L which when substituted into equation (5.9) to give an equation for each of k_a, k_b, k_c in terms of reactance, inductance and primary operating resonant frequency ω_0.

$$k_a = \frac{\left[\begin{array}{l} +(\omega_0^2 L_L - \omega_0 X_L)(-L_P^2 - 2L_S L_L - 2L_P L_L - 2L_S L_P) \\ +(\omega_0^2 L_P - \omega_0 X_P)(-L_L^2 - 2L_P L_L - 2L_S L_L - 2L_S L_P) \\ +(\omega_0^2 L_S - \omega_0 X_S)(-L_L^2 - 2L_P L_L - L_P^2) \\ +\ R_L^2 (L_P + L_S) \end{array}\right]}{L_P L_L^2 + L_P^2 L_L + L_S L_L^2 + L_S L_P^2 + 2L_S L_P L_L} \qquad (5.26)$$

$$k_b = \frac{\left[\begin{array}{l} +(\omega_0^2 L_L - \omega_0 X_L)^2 (L_P + L_S) \\ +(\omega_0^2 L_L - \omega_0 X_L)(\omega_0^2 L_P - \omega_0 X_P)(2L_P + 2L_L + 2L_S) \\ -(\omega_0^2 L_P - \omega_0 X_P) R_L^2 \\ +(\omega_0^2 L_P - \omega_0 X_P)^2 (L_L + L_S) \\ -(\omega_0^2 L_S - \omega_0 X_S) R_L^2 \\ +2(\omega_0^2 L_S - \omega_0 X_S)(\omega_0^2 L_L - \omega_0 X_L)(L_L + L_P) \\ +2(\omega_0^2 L_S - \omega_0 X_S)(\omega_0^2 L_P - \omega_0 X_P)(L_L + L_P) \end{array}\right]}{L_P L_L^2 + L_P^2 L_L + L_S L_L^2 + L_S L_P^2 + 2L_S L_P L_L} \qquad (5.27)$$

$$k_c = \frac{\left[\begin{array}{l} -(\omega_0^2 L_P - \omega_0 X_P)(\omega_0^2 L_L - \omega_0 X_L)^2 \\ -(\omega_0^2 L_L - \omega_0 X_L)(\omega_0^2 L_P - \omega_0 X_P)^2 \\ -(\omega_0^2 L_S - \omega_0 X_S)(\omega_0^2 L_S - \omega_0 X_L)^2 \\ -(\omega_0^2 L_S - \omega_0 X_S)(\omega_0^2 L_P - \omega_0 X_P)^2 \\ -2(\omega_0^2 L_S - \omega_0 X_S)(\omega_0^2 L_P - \omega_0 X_P)(\omega_0^2 L_L - \omega_0 X_L) \end{array}\right]}{L_P L_L^2 + L_P^2 L_L + L_S L_L^2 + L_S L_P^2 + 2L_S L_P L_L} \qquad (5.28)$$

These are the first three system equations. To solve for the fourth component in the circuit a further equation is required. The resistance of the circuit at resonance is

$$R_{TOT} = \frac{R_L X_P^2}{R_L^2 + (X_L + X_P)^2} . \qquad (5.29)$$

Equations (5.26), (5.27), (5.28) and (5.29) can now be solved simultaneously to obtain the required reactances and, subsequently, for the component values. Equations (5.26), (5.27) and (5.28) are first rearranged into three equations for L_S, the value of the series inductance, of the converter. It should be noted that k_a, k_b and k_c are known values related to the specified resonant frequencies of the circuit by equation (5.11).

$$L_S=\frac{\begin{bmatrix}+k_bL_PL_L^2+k_bL_P^2L_L-L_P(\omega_0^2L_L-\omega_0X_L)^2\\-2(L_P+L_L)(\omega_0^2L_L-\omega_0X_L)(\omega_0^2L_P-\omega_0X_P)\\+(\omega_0^2L_P-\omega_0X_P)R_L^2-L_L(\omega_0^2L_P-\omega_0X_P)^2\\-\omega_0X_SR_L^2\\+2\omega_0X_S(\omega_0^2L_L-\omega_0X_L)(L_L+L_P)\\+2\omega_0X_S(\omega_0^2L_P-\omega_0X_P)(L_L+L_P)\end{bmatrix}}{\begin{bmatrix}-k_bL_L^2-k_bL_P^2-2k_bL_PL_L+(\omega_0^2L_L-\omega_0X_L)^2\\+2(\omega_0^2L_L-\omega_0X_L)(\omega_0^2L_P-\omega_0X_P)\\+(\omega_0^2L_P-\omega_0X_P)^2-\omega_0^2R_L^2\\+2\omega_0^2(L_L+L_P)(\omega_0^2L_L-\omega_0X_L)\\+2\omega_0^2(L_L+L_P)(\omega_0^2L_P-\omega_0X_P)\end{bmatrix}} \tag{5.30}$$

$$L_S=\frac{\begin{bmatrix}+k_cL_PL_L^2+k_cL_P^2L_L\\+(\omega_0^2L_P-\omega_0X_P)(\omega_0^2L_L-\omega_0X_L)^2\\+(\omega_0^2L_L-\omega_0X_L)(\omega_0^2L_P-\omega_0X_P)^2\\-\omega_0X_S(\omega_0^2L_L-\omega_0X_L)^2-\omega_0X_S(\omega_0^2L_P-\omega_0X_P)^2\\-2\omega_0X_S(\omega_0^2L_P-\omega_0X_P)(\omega_0^2L_L-\omega_0X_L)\end{bmatrix}}{\begin{bmatrix}-k_cL_L^2-k_cL_P^2-2k_cL_PL_L\\-\omega_0^2(\omega_0^2L_L-\omega_0X_L)^2\\-\omega_0^2(\omega_0^2L_P-\omega_0X_P)^2\\-2\omega_0^2(\omega_0^2L_P-\omega_0X_P)(\omega_0^2L_L-\omega_0X_L)\end{bmatrix}} \tag{5.31}$$

$$L_S=\frac{\begin{bmatrix}+k_aL_PL_L^2+k_aL_P^2L_L\\+(L_P^2+2L_PL_L)(\omega_0^2L_L-\omega_0X_L)\\+(L_L^2+2L_PL_L)(\omega_0^2L_P-\omega_0X_P)\\+\omega_0X_S(-L_L^2-2L_PL_L-L_P^2)-R_L^2L_P\end{bmatrix}}{\begin{bmatrix}-k_aL_L^2-k_aL_P^2-2k_aL_PL_L\\+(-2L_L-2L_P)(\omega_0^2L_L-\omega_0X_L)\\+(-2L_L-2L_P)(\omega_0^2L_P-\omega_0X_P)\\+\omega_0^2(-L_L^2-2L_PL_L-L_P^2)+R_L^2\end{bmatrix}} \tag{5.32}$$

Equations (5.30), (5.31), (5.32) are equated, thereby eliminating L_S, to produce three quartics in L_P. The three quartics are shown in Appendix A.1. Each of these three

quartics has four roots for L_P. By a process of subtraction it can be proven that each of the quartics had the same four roots. The coefficients of the quartics can therefore be equated to eliminate the variable L_P. The L_P^2 term was equated. This produces a quadratic in X_S which has two solutions. The first solution for X_S is given in Appendix A.2; the second value of X_S is $X_S = -X_L$.

The solution for X_S in Appendix A.2. is a function of X_P, k_a, k_b, k_c, ω_0, L_L, and X_L, all are known with the exception of X_P. In order to solve for X_P and X_S a second equation for X_S dependent on X_P, R_{TOT}, X_L and R_L was obtained from equations (5.29 and (5.5).

$$X_S = -\frac{R_{TOT}(X_P X_L + X_L^2 + R_L^2)}{R_L X_P} . \qquad (5.33)$$

Equations in Appendix A.2 and (5.33) are then equated producing a cubic polynomial in X_P. This polynomial is listed in Appendix A.3. The three solutions to this polynomial were found using Mathematica, each solution being a function of the known design parameters.

A Mathematica program was written based on the solutions to this system of equations. A system specified in terms of X_L, L_L, R_L, R_{TOT}, ω_0, ω_1 and ω_2 may be solved to find the required component values L_S; C_S, L_P and C_P. Mathematica was chosen as the programming language so that the analysis can remain in a symbolic format. The Mathematica program is listed in Appendix A.4.

5.6 Conclusion

The analysis in this chapter uses simple frequency-domain analysis. Previous work in the literature uses a time-domain approach for the analysis or simplifies the circuit significantly so that the circuit appears to have but one resonant frequency. This present analysis has shown, as would be expected, that the circuit has a frequency response with multiple-resonant frequencies. The relationship between these resonant

frequencies and the component values of the circuit have been established. The circuit has been simulated with various component values to show the many general forms of frequency response which can be achieved using these circuits. Each frequency response has a different practical significance. This work has formed the basis of the welding power supply design and a new method of power control in the series-parallel load-resonant converter.

CHAPTER 6 DESIGN AND SIMULATION OF SERIES-PARALLEL LOAD-RESONANT CONVERTERS

6.1 Introduction

The analysis presented in Chapter 5 allowed the component values of a series-parallel load-resonant circuit to be calculated by specifying the three resonant frequencies and the load parameters of the circuit. This chapter shows how this analysis was used to design two series-parallel load-resonant circuits which were then simulated using the Saber simulation software. The first design, Circuit 1, was a circuit that achieved a current of 200 A in the load and operated at a frequency of 85 kHz. The current levels at the other resonant frequencies of the circuit were not specified. The second design, Circuit 2, specified the current levels in the load at two of the resonant frequencies of the circuit. This allowed a novel method of power control in the series-parallel load resonant converter to be developed.

6.2 Design of series-parallel load-resonant converters incorporating an output rectifier

In the proposed designs the output of the load-resonant converter is to be rectified before supplying the load. This means that the apparent power supply load resistance required for the design procedure is not the same as the actual value of the load resistance. According to Steigerwald [34], when the load on the output of a rectifier has some inductance, the current in the load is no longer a half-wave rectified sine-wave. The relationship between the apparent load resistance for a.c. analysis purposes (using the fundamental components only), R_{ac}, and the actual load resistance, R_{LOAD}, is given by

$$R_{ac} = \frac{\pi^2}{8} R_{LOAD} \tag{6.1}$$

The apparent effect of the rectifier in combination with the load inductance, is to reduce the current in the load. When rectifiers are in the circuit this must be taken into account if the a.c. analysis presented in Chapter 5 is to be valid.

6.3 Calculation of power in the series-parallel load-resonant converter

During the design procedure it was necessary to calculate the power delivered to the resonant circuit and the power absorbed by the load. The load-resonant converter was supplied by a half bridge inverter circuit with a maximum d.c. supply voltage of 600 V. Consequently, a maximum square-wave voltage of ± 300 V excited the resonant circuit. The r.m.s. value of the fundamental component of a ± 300 V square-wave voltage is

$$V_{S_{rms}} = 600 \frac{2}{\pi} \frac{1}{\sqrt{2}} = 270\ V \qquad (6.2)$$

This value of the fundamental assumes that the voltage waveform is a pure square-wave. In practice there would be a finite rise and fall time equivalent to the switching speed of the power devices. This becomes more significant at higher operating frequencies and would reduce the r.m.s. value quoted in (6.2).

The input power to the complete resonant circuit can be calculated using this r.m.s. voltage and the apparent circuit resistance at the resonant frequency. Output power from the circuit can be calculated from the average load current after rectification and the actual load resistance. In the following design procedure an efficiency of 80% was assumed for the converters. This value for estimated efficiency was chosen below the level normally expected of electronic power supplies because the low output voltage would make the on-state voltages in the output rectifier particularly significant. Furthermore, at the switching frequencies under consideration the switching losses in the output rectifiers are likely to be significant.

6.4 Design of series-parallel load-resonant circuit - Circuit 1

A series-parallel load-resonant circuit was designed to achieve 200 A d.c. to a resistive load with the load-resonant converter running at a frequency of 85 kHz. The other resonant frequencies were specified as 67.5 kHz and 40 kHz. The load resistance, R_{LOAD}, was chosen as 0.125 Ω, to model the resistance of a typical welding arc [9]. This value gives an arc voltage of 25 V with 200 A arc current, corresponding to an

output power of 5 kW. With the assumed efficiency of 80 % the input power required to meet this is 6250 W. Assuming an r.m.s. fundamental supply voltage of 270 V, the resistance of the complete circuit, R_{TOT} at the required resonant frequency of 85 kHz was calculated to be 11.7 Ω.

From (6.1) the apparent value of the load resistance R_{ac} is 0.154 Ω. A transformer with a turns ratio of 11:1 was chosen giving a referred apparent load resistance, R_L, of 18.6 Ω. The transformer had a leakage inductance of 24 μH. A further inductor was added in series with the primary of the isolation transformer to make a total load leg inductance of 80 μH.

The remaining four component values in the series-parallel load-resonant circuit were calculated using the analysis shown in Chapter 5 and the Mathematica program written for this purpose and given in Appendix B. The component values calculated by the programme are given in Table 6.1. This circuit has been validated by simulation (see section 6.8) and experimental testing (Chapter 7).

Parameters specified in Design Programme for Circuit 1			
Referred load resistance, corrected for a.c. analysis, R_L = 18.6 Ω			
Resonant frequencies of the circuit - 85 kHz, 67.5 kHz, and 40 kHz			
Total circuit resistance, R_{TOT} = 11.7 Ω at the resonant frequency of 85 kHz			
Load leg inductance inc. leakage inductance of the transformer, L_L= 80 μH			
Component values calculated by Design Programme			
C_S = 87 nF	L_S = 112 μH	C_P = 45 nF	L_P = 34 μH

Table 6.1 Specified parameters and calculated component values for Circuit 1

6.5 A new method of power control for series-parallel load-resonant converters

In Chapter 5, Figs. 5.4 - 5.8 showed the variation in admittance with frequency for a range of series-parallel load-resonant converters. A series-parallel load-resonant

converter was shown to have three pairs of real resonant frequencies with a different apparent resistance at each resonant frequency. A single valued load resistance can be transformed by the resonant circuit to give a different total circuit resistance at each resonant frequency. The power delivered to the circuit, and hence to the load, from a fixed d.c. supply voltage therefore varies depending on the resonant frequency at which the circuit has been excited. This is the basis of the new method of power control for series-parallel load-resonant converters. Unlike previous methods of power control for load-resonant converters (see Section 4.3), this method can be used to provide substantial changes in the power levels without introducing power losses in the switching devices since zero-current switching is maintained at each resonant frequency. One discrete change in power level is shown schematically in Figure 6.1.

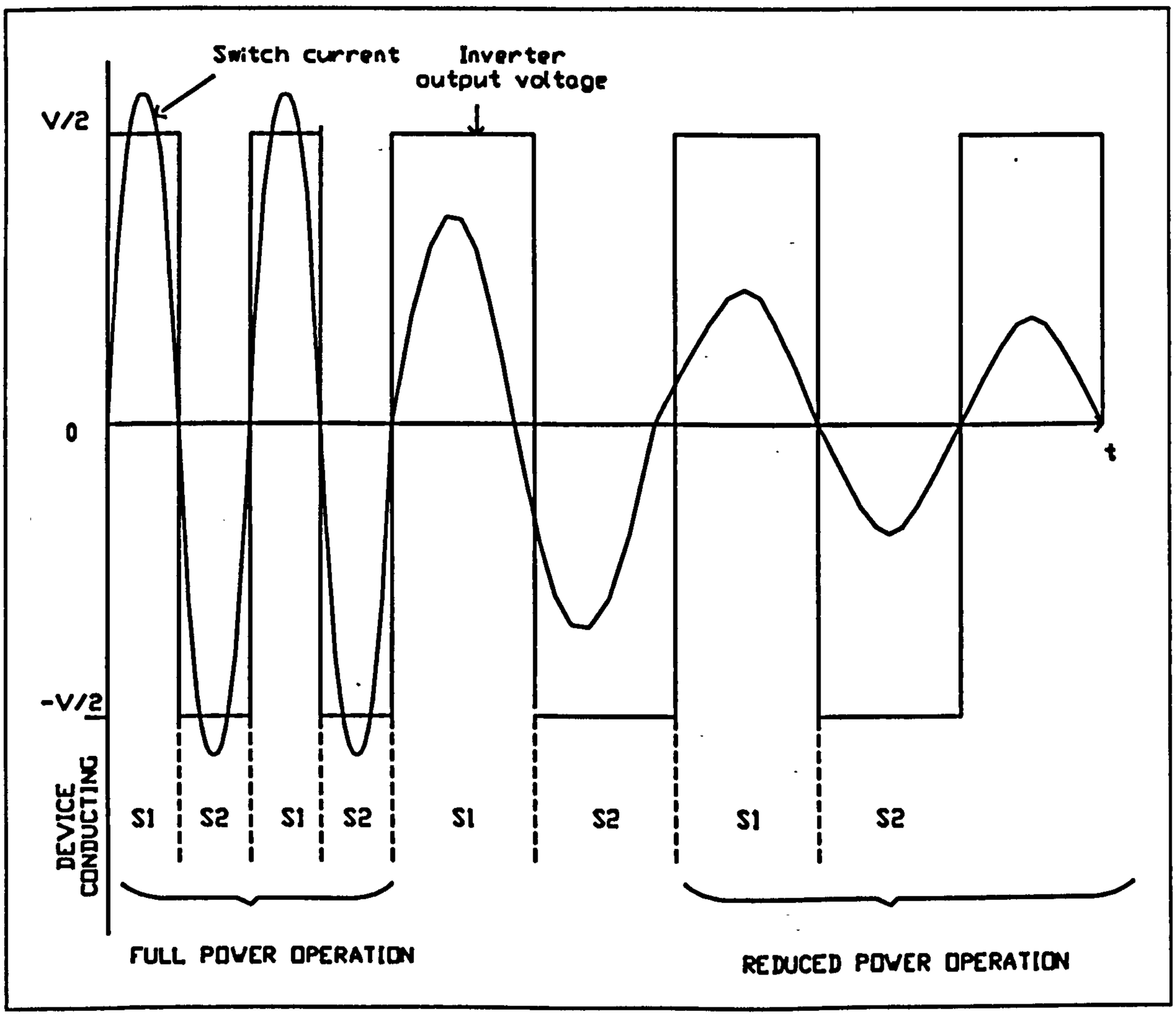


Figure 6.1 Change in switch current as circuit frequency is changed from high-power to low-power operation

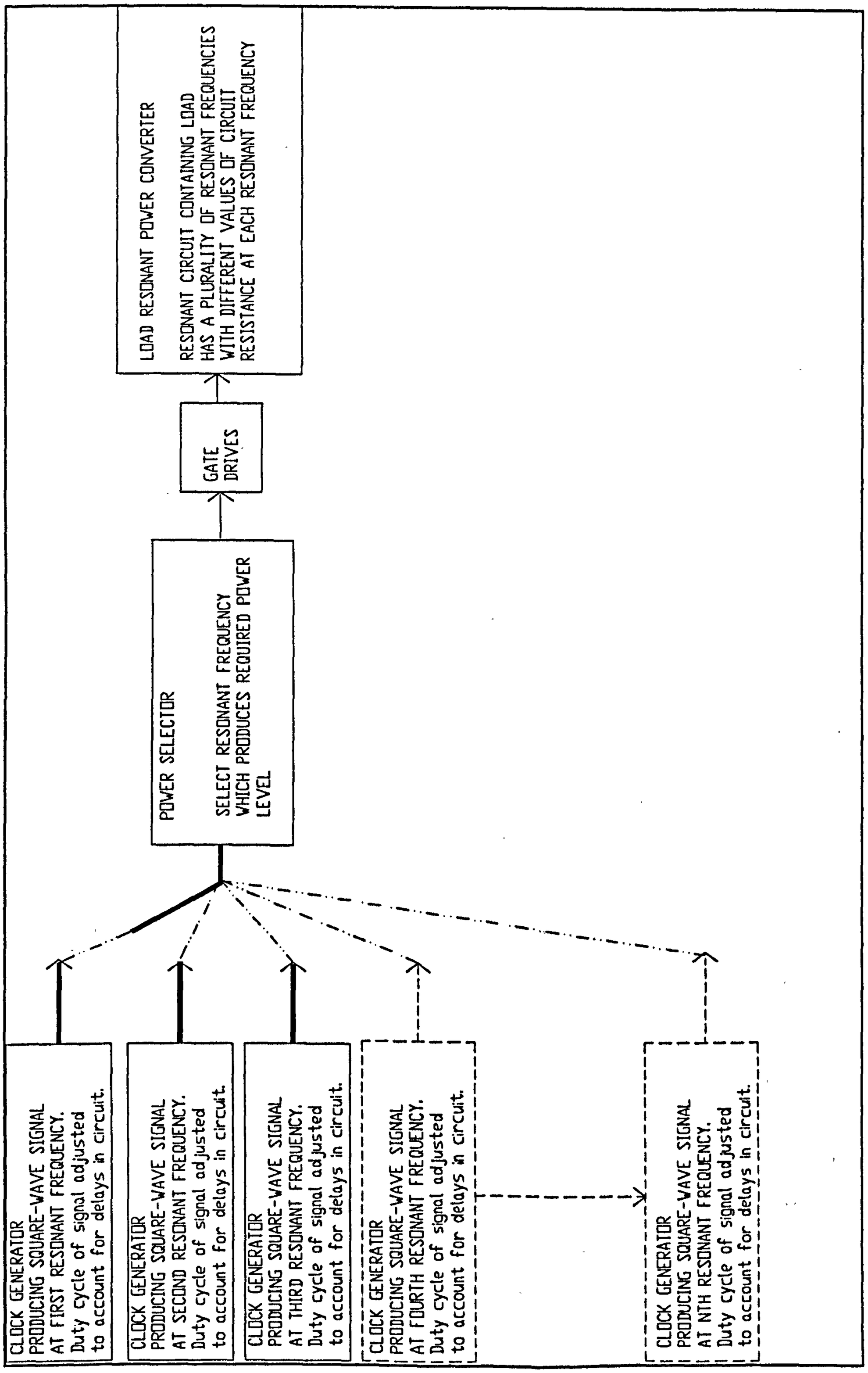


Figure 6.2 Simple control scheme to achieve pulsed output power

As the excitation of the circuit is changed from an upper resonant frequency with a low value of total circuit resistance to a lower resonant frequency with a higher value of circuit resistance the current automatically drops since the resistance of the circuit increases. Zero current switching is maintained in steady state operation at each power level at each resonant frequency.

The method is particularly suited to power supplies which have a requirement for pulsed power, although it will be shown later that intermediate power levels are also achievable. The method can be extended to resonant circuits with any number of resonant frequencies and a simple control scheme is shown schematically in Figure 6.2. Changes in the power level in the load are achieved by selecting the appropriate clock generator in order to excite the resonant circuit at the appropriate resonant frequency.

This new method of power control has been incorporated into the design of a new arc-welding power supply for pulsed operation, the design of which will now be described.

6.6 Design of a series-parallel load-resonant converter for the novel power control method - Circuit 2

Circuit 1 confirmed that the analysis presented in Chapter 5 could calculate realistic component values for a given set of resonant frequencies and load parameters. The computer package was then used to design a welding power supply suitable for pulsing between two power levels using the new method described above.

During a high power pulse the circuit would be running at one chosen resonant frequency. In a low-power mode, required to keep the welding arc simmering between high power pulses, the circuit would be running at a second resonant frequency. Unlike the design of the Circuit 1, this load resonant converter would have to be designed to run at more than one resonant frequency. In order to stay within the power limitations of a 13 A single phase a.c. supply, 3 kW of input average power

was considered as a maximum. However, as the power supply would be pulsing, the power drawn by the load-resonant converter could exceed this level during a high current pulse. In the high power mode the input power to the circuit was limited to 5 kW and an input power of 1 kW was chosen for the low power mode.

In the high-power mode, when 5 kW of power was to be delivered to the circuit, the total resistance of the resonant circuit, R_{TOT}, needed to be

$$R_{TOT} = \frac{V_{S_{rms}}^{\ 2}}{P_{HIGH}} = \frac{(270)^2}{5000} = 15.6\ \Omega\ . \tag{6.3}$$

In the low-power mode, when 1 kW of power was to be delivered to the circuit, the total resistance of the resonant circuit R_{TOT}, needed to be

$$R_{TOT} = \frac{V_{S_{rms}}^{\ 2}}{P_{LOW}} = \frac{(270)^2}{1000} = 72.9\ \Omega\ . \tag{6.4}$$

If the predicted efficiency of 80 % was achieved in practice, this would translate to average load currents of 178 A in the high-power mode and 80 A in the low-power mode.

The series-parallel load-resonant converter was designed using the value of R_{TOT} in the high-power mode. The upper resonant frequency was chosen as 100 kHz. This was chosen as the frequency at which the converter would operate in the high-power mode to minimise the size of the isolation transformer. The same isolation transformer as Circuit 1 with an 11:1 turns ratio was used for this design. The second and third resonant frequencies were specified as 75 kHz and 50 kHz respectively. This meant that odd harmonics of the forcing voltage, when running at either of the two lower resonant frequencies, would not excite a 100 kHz resonance.

6.7.1 Simple model simulation

The simple series-parallel load-resonant circuit model in the simulation is shown in Figure 6.3. The circuit contains an inductor, L_S, and a capacitor, C_S, in the series-leg, an inductor, L_P, and capacitor, C_P, in the parallel-leg and an inductor, L_L and a resistor, R_L, in the load-leg. R_L represents the resistance of the actual load referred to the primary-side of the isolation transformer. This model has the same configuration as the model used to develop the analysis of the circuit presented in Chapter 5.

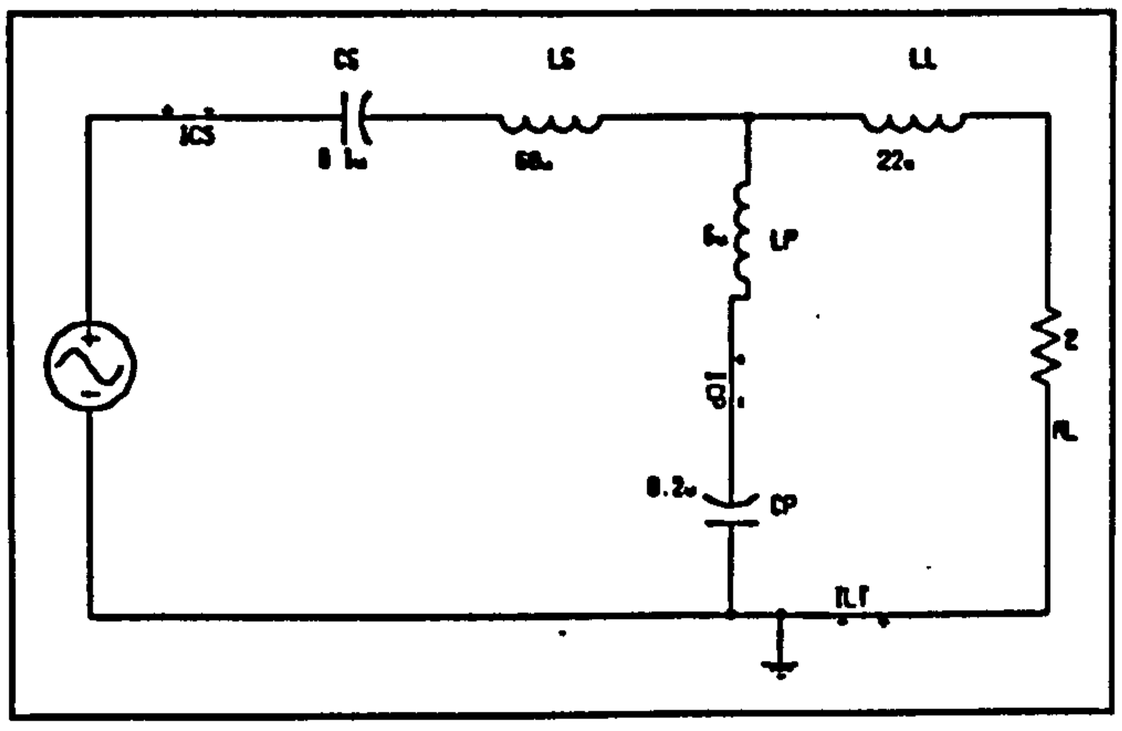


Figure 6.3 Simple series-parallel load-resonant circuit model in Saber

A frequency domain a.c. analysis was carried out using the simple model. This gave the frequency response of the circuit and confirmed the location of the resonant frequencies and the magnitude and phase of the current in each section of the circuit. A transient analysis of the circuit was obtained by driving the circuit with a square-wave voltage of ±300 V with its fundamental frequency at each resonant frequency of the circuit.

6.7.2 Full model of series-parallel load-resonant converter

The full series-parallel load-resonant converter model used in the Saber simulation is shown in Figure 6.4. The Saber files are given in Appendix C. To speed up the simulation the circuit was used with ideal models of the power switches and the circuit was controlled by algorithms written in the MAST development language. The

modelling of the circuit, using ideal devices, ignored the diode recovery of both the freewheeling diodes and the rectifier diodes, the switching characteristics of the IGBT and the gate drive requirements of the IGBT. This allowed greatly reduced simulation times while having a minimal effect on the accuracy of the results.

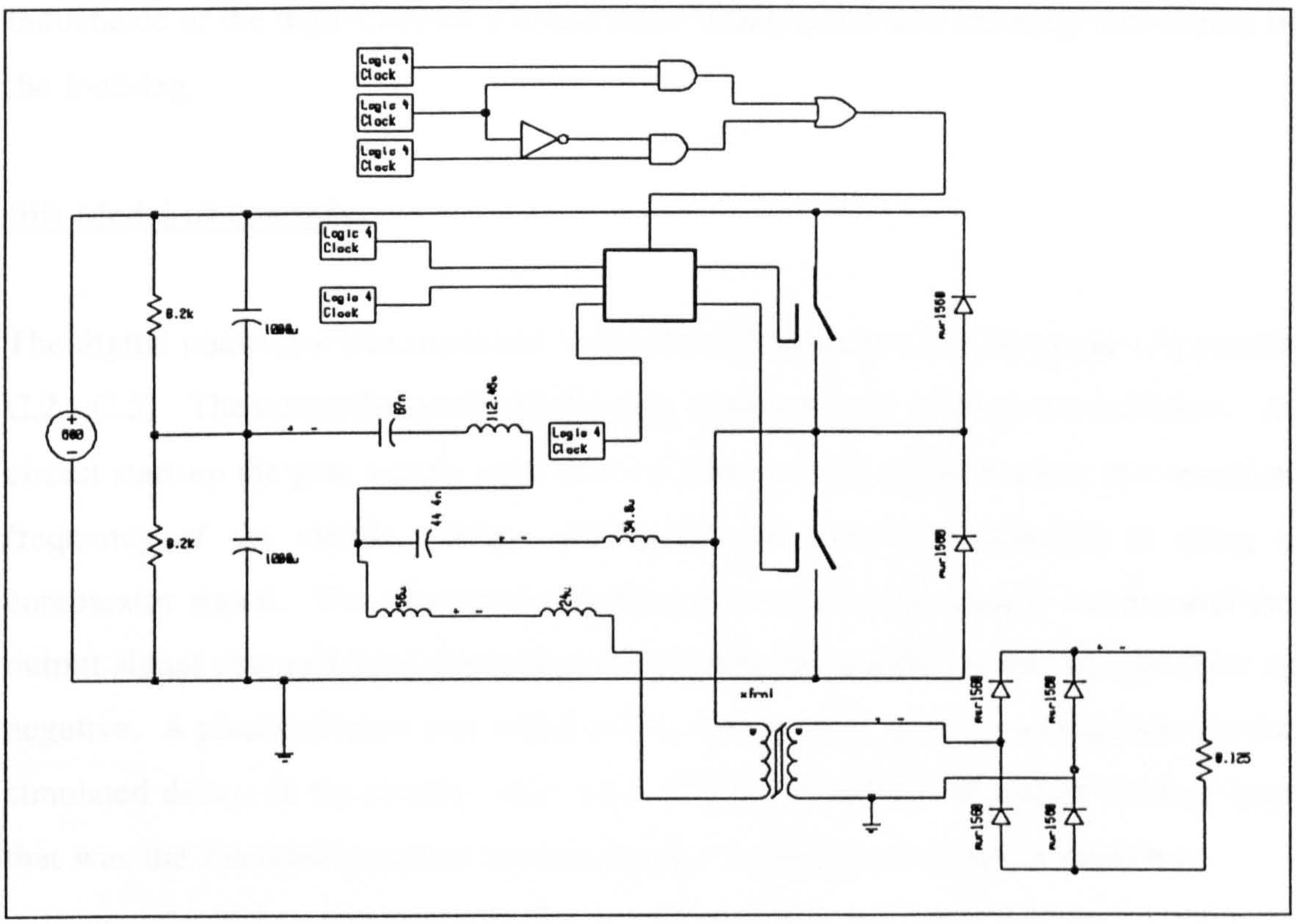


Figure 6.4 Complete series-parallel load-resonant converter model in Saber

(i) Model of power switches

The ideal switches were described in the MAST programming language as a two-pole switch that had two values of resistance (Appendix C.1). The switches were controlled by a logic signal representing the gate signal of the power devices. A logic level 'high' on the gate signal represented the switch being turned 'on'. Other components in the power circuit were standard models available within the simulator package.

(ii) Model of resonant components

The two winding transformer was an ideal transformer i.e. the model assumed infinite magnetising inductance and negligible leakage inductance. An extra inductor was placed in series with the primary winding of the transformer to simulate the leakage inductance of the high-frequency transformer which acted as a resonant component in the load-leg.

(iii) Model of controller

The digital controller was modelled in the MAST programming language (Appendix C.2 - C.5). The controller produced the gate signals for the ideal power switches. At circuit start-up the gate signals were derived from a clock signal running at a resonant frequency of the circuit. After 256 cycles the controller switched to using a comparator signal. The simulated comparator monitored the switch current and the output signal changed state depending on whether the switch current was positive or negative. A phase advance was added to this comparator signal to compensate for the simulated delays in the circuit and to allow for the modelling of a phase-locked loop that was the intended practical implementation of the phase-advance circuitry.

6.8 Results of the simulation of Circuit 1

Figure 6.5 gives the frequency response of the series-parallel load-resonant converter of Circuit 1. From this plot it was confirmed that the resonant frequencies were 82 kHz, 67 kHz and 38 kHz, all very close to the frequencies specified in the design procedure (Section 6.4).

Figure 6.6 shows the time domain, transient analysis results obtained for Circuit 1 when driven by a ± 300 V square-wave with fundamental frequencies at 82 kHz, 67 kHz and 38 kHz.

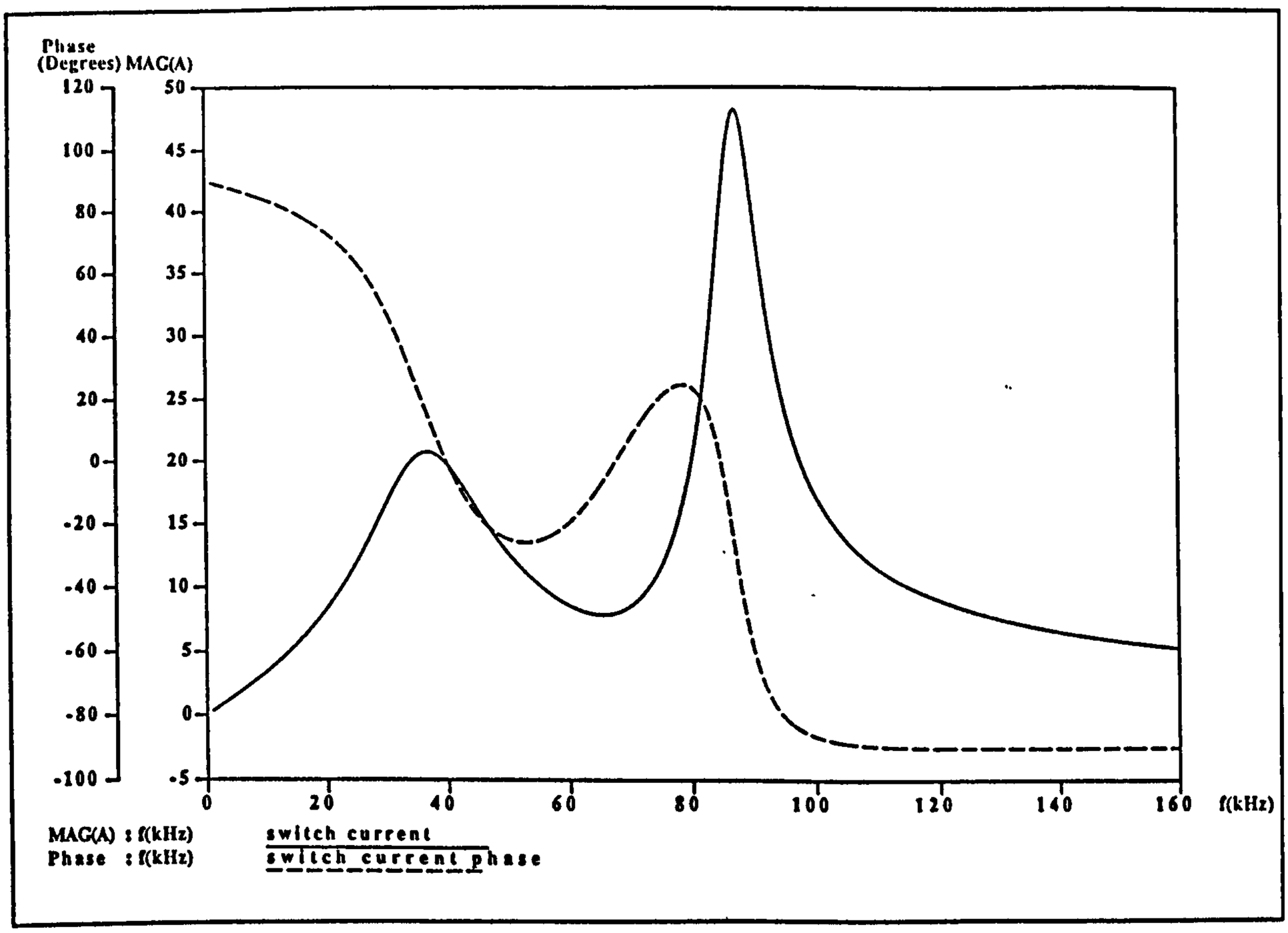


Figure 6.5 Frequency response of Circuit 1.

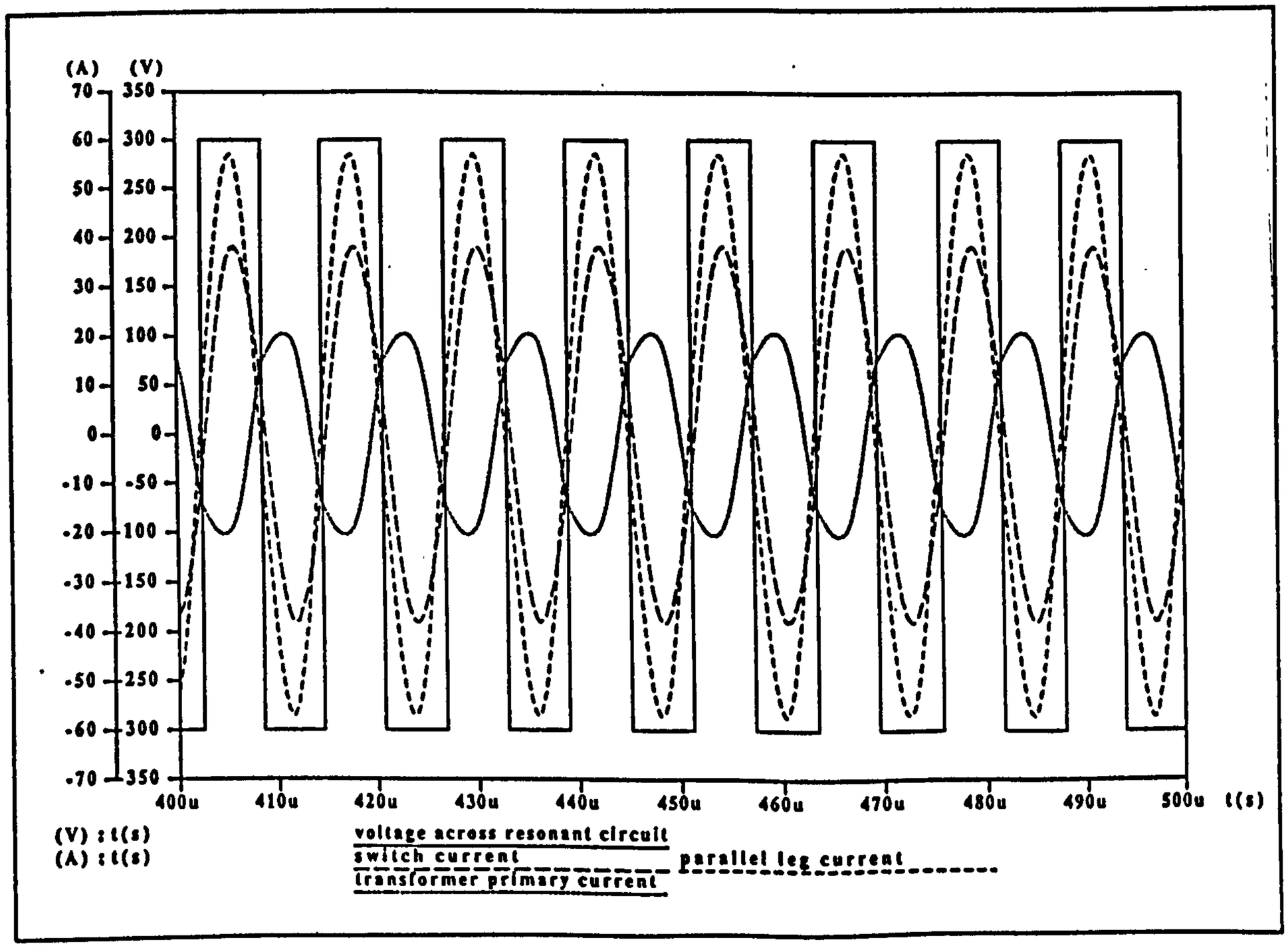


Figure 6.6 (a) Circuit 1 driven at 82 kHz.

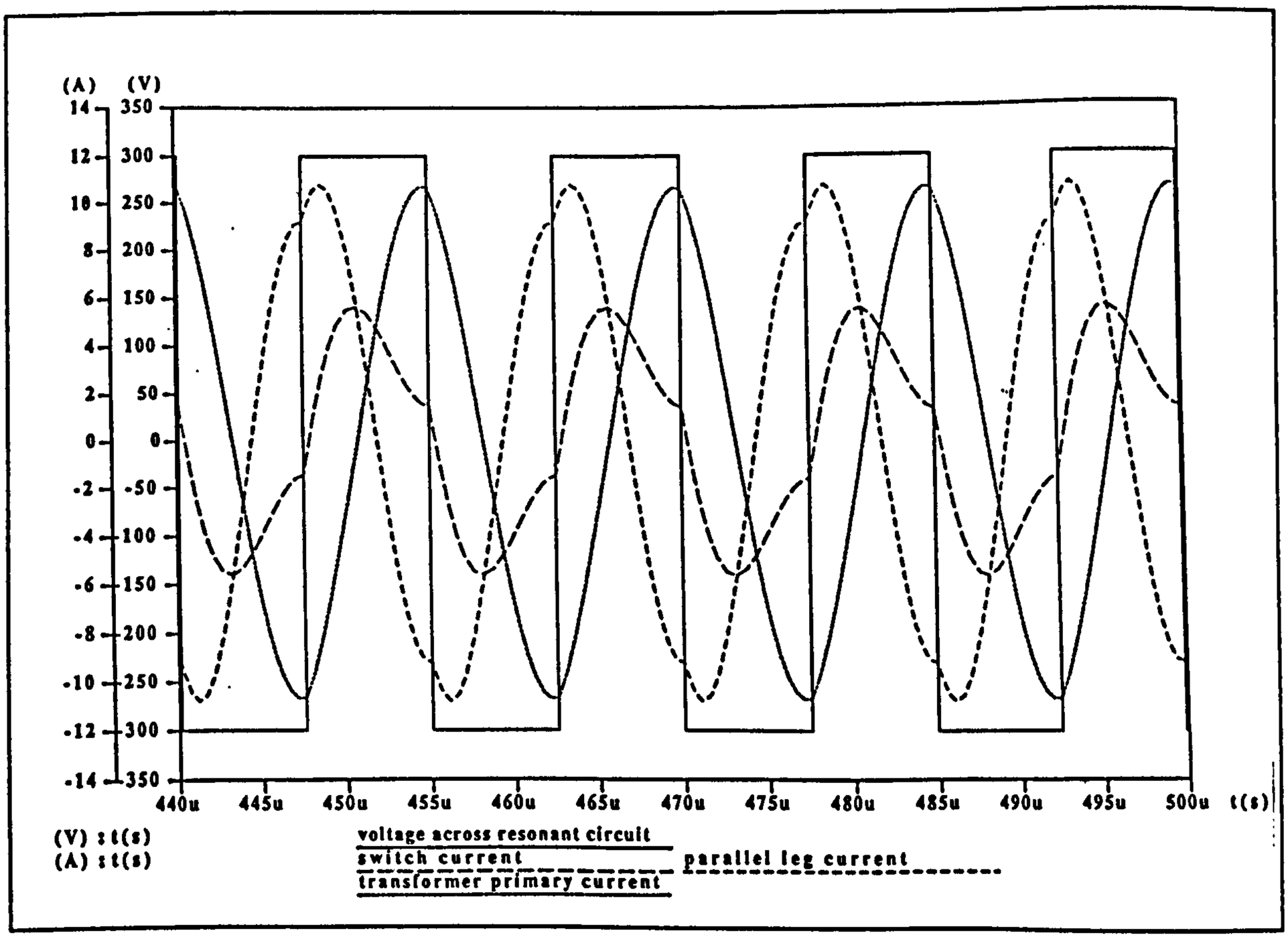


Figure 6.6 (b) Circuit 1 driven at 67 kHz

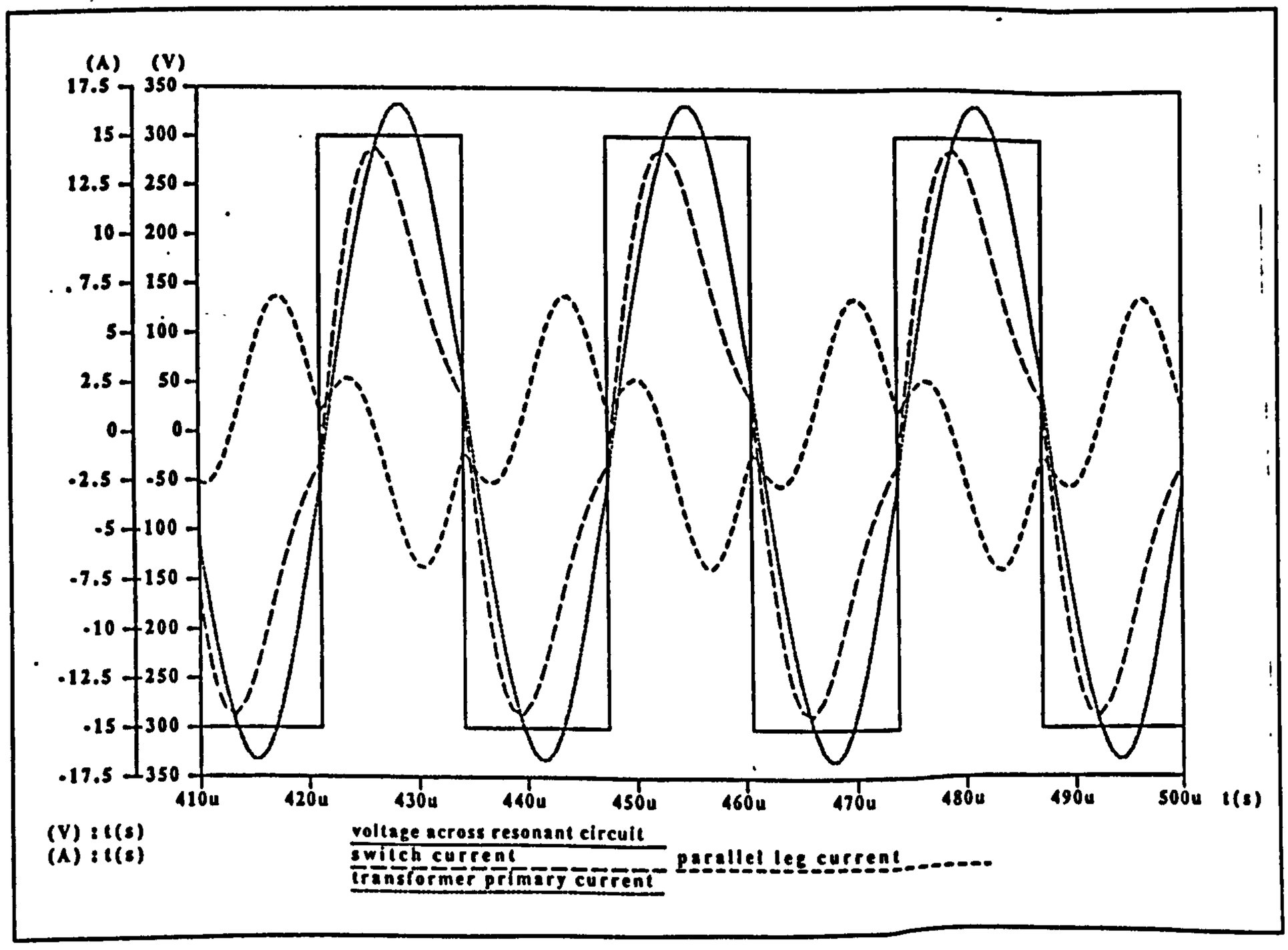


Figure 6.6 (c) Circuit 1 driven at 38 kHz

Figure 6.7 shows the result of the Circuit 1 simulation under normal operating conditions at the upper resonant frequency of 82 kHz, showing the rectified load current in addition to the switch voltage and switch current.

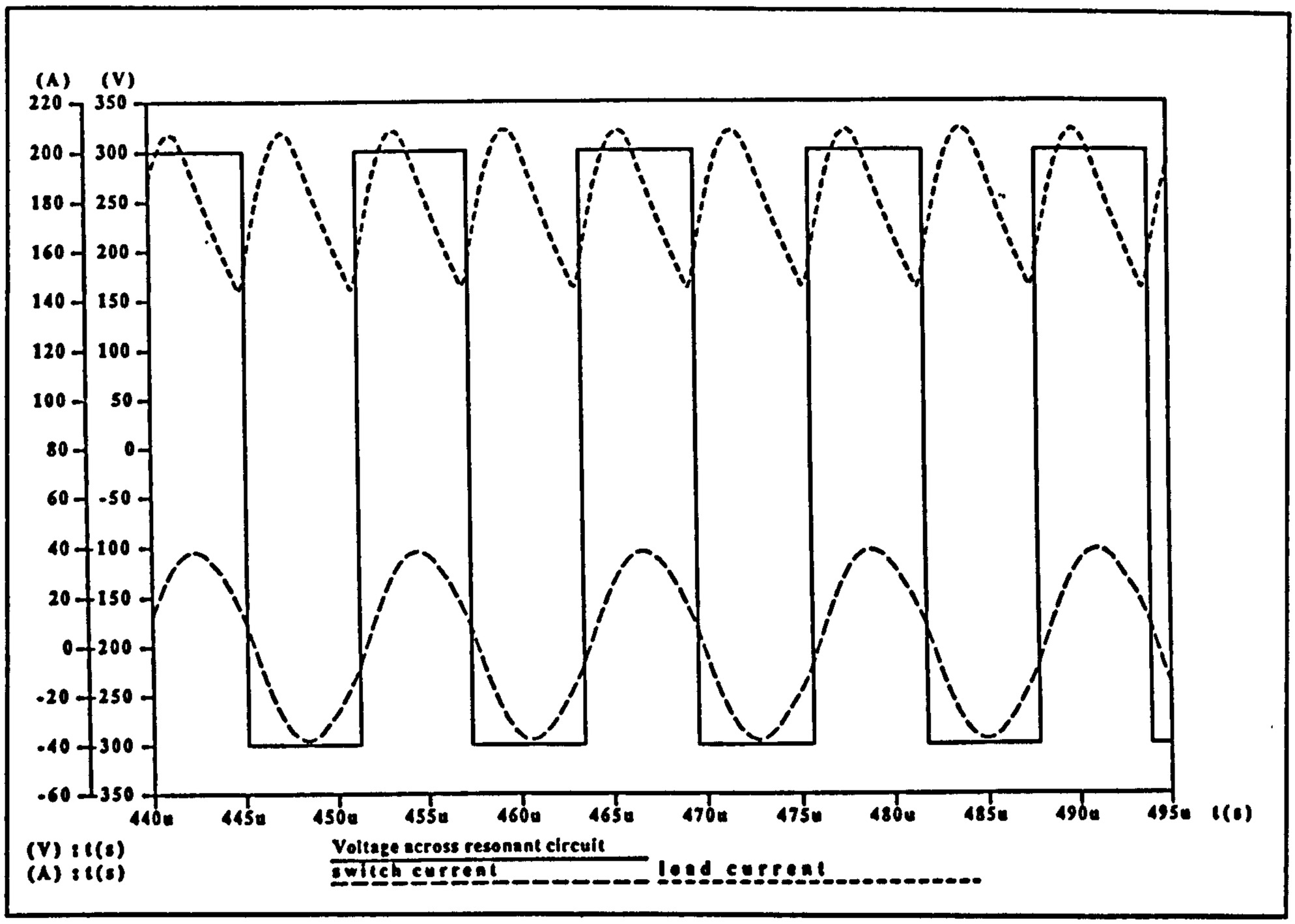


Figure 6.7 Full-circuit simulation of circuit 1

The rectified load current has a peak value of 208 A, dropping to 150 A near the zero crossing point of the switch current. This simulated load current (average 180 A) is slightly lower than the design value of 200 A. .This can be attributed to the fact that the component values used in Saber were rounded to the nearest whole number and that the large value of capacitor C_L predicted by Mathematica was ignored in Saber. Furthermore, Saber contains a real rectifier connected to a load of very low inductance, as can be observed from the load current waveform. The value of the load resistance used the Mathematica design program was calculated using [34] which assumes that the load current is constant with no ripple. In Fig. 6.6(a) the switch current is seen to be in phase with the switch voltage and to have a peak value of 38 A. Taking the r.m.s. value of the fundamental component of the voltage and the r.m.s. switch current, the simulated input power is 7255 W. This indicates that the value of R_{TOT} in the circuit as designed is 10 Ω rather than the desired value of 11.7 Ω.

The Saber simulation has supported the efficacy of the design mathematics presented in Chapter 5. The resonant frequencies occurred close to the design frequencies and the resistance value of the circuit at the specified resonant frequency was marginally lower.

6.9 Results of the simulation of Circuit 2

The series-parallel load-resonant circuit, designed to evaluate the novel pulsed power control technique, was simulated using the Saber simulation software models described in Section 6.7. Figure 6.8 shows the frequency response of the circuit showing the phase angle of the switch current and the magnitude of the switch current when excited with a sinusoidal voltage of 270 V r.m.s. Figure 6.9 (a,b,c) show time domain simulations for the circuit running at each of its resonant frequencies, 97 kHz, 63 kHz, and 50 kHz.

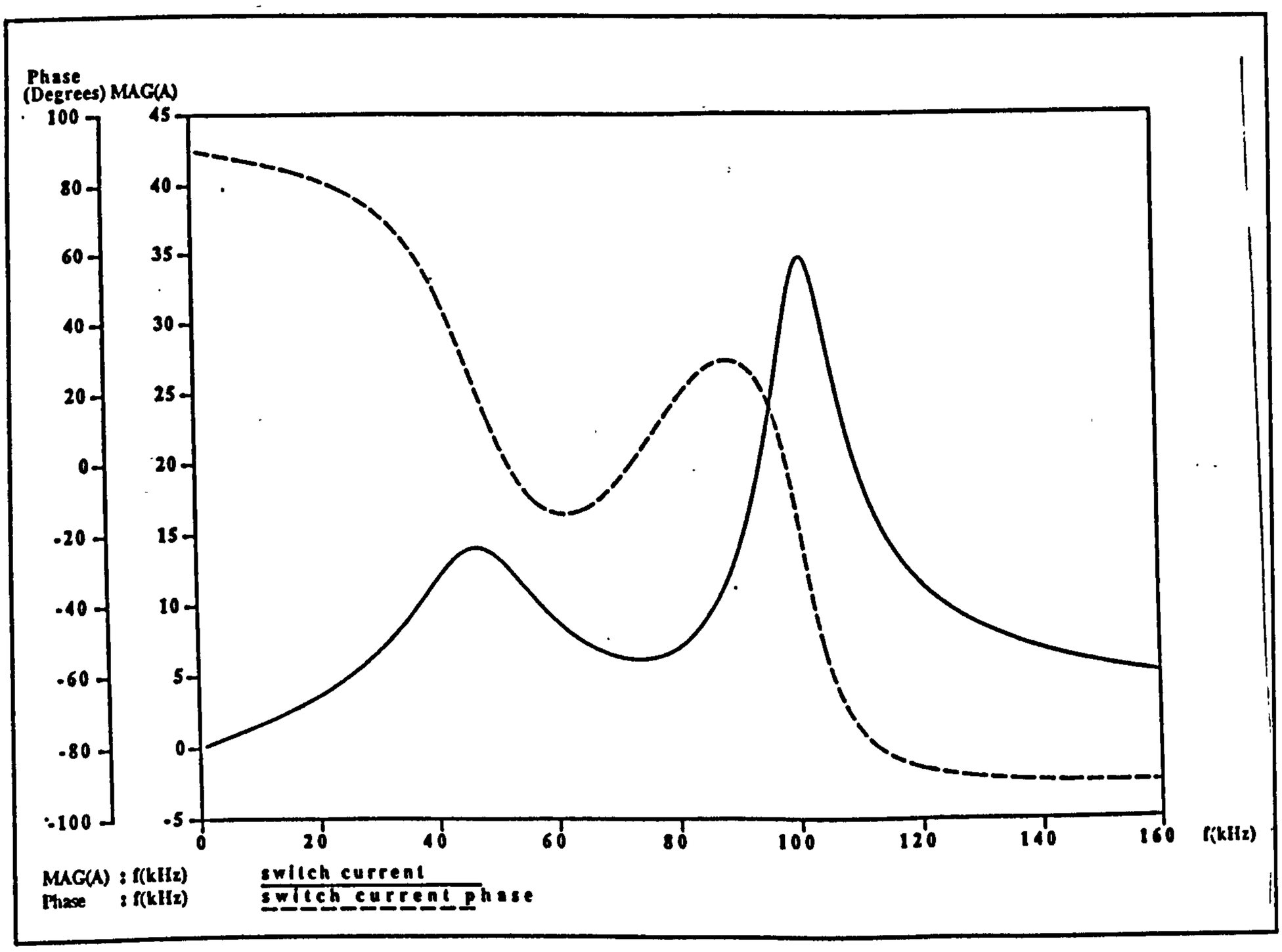


Figure 6.8 Frequency response of Circuit 2

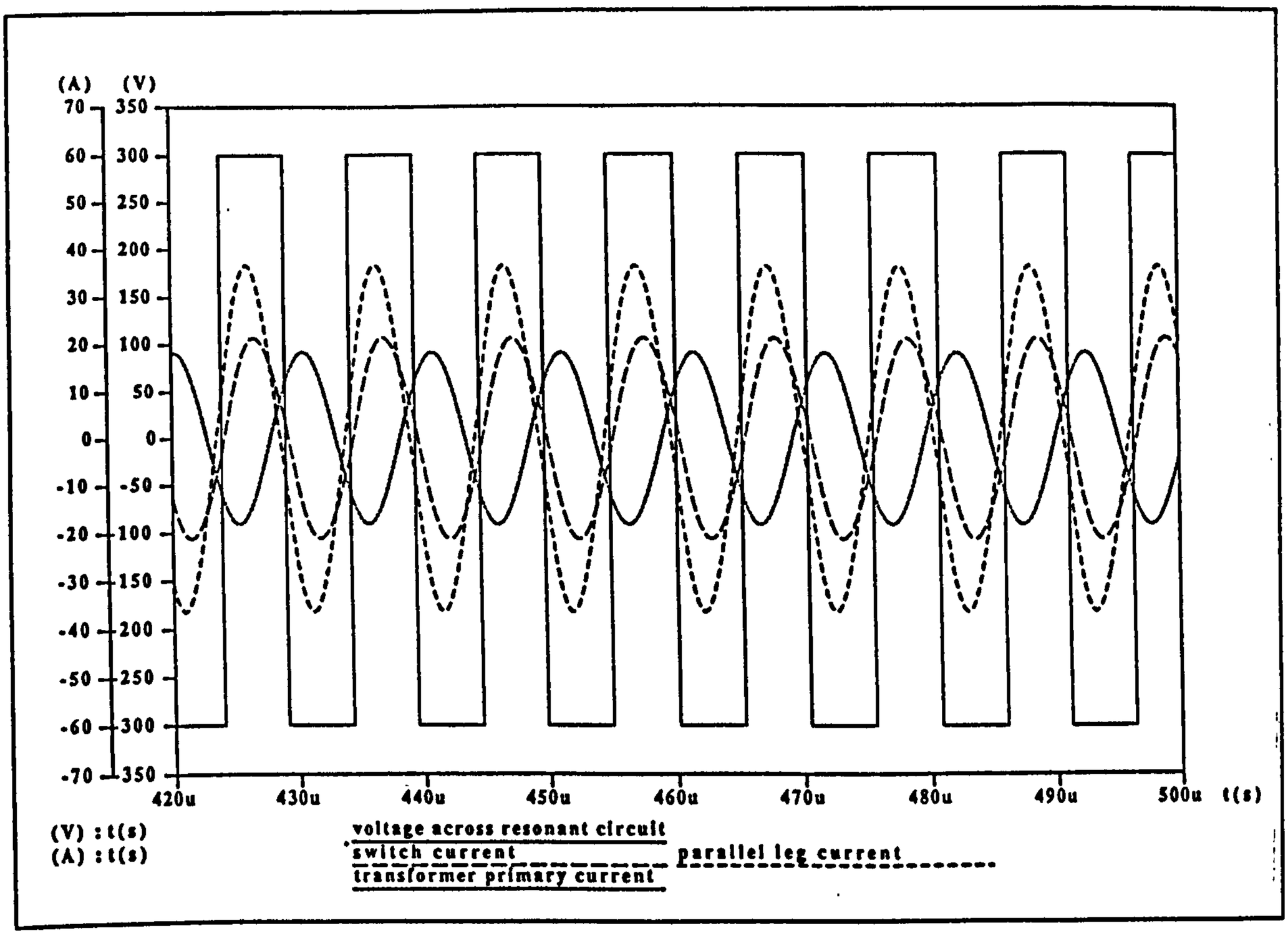


Figure 6.9(a) Circuit 2 driven at 97 kHz, the upper resonant frequency

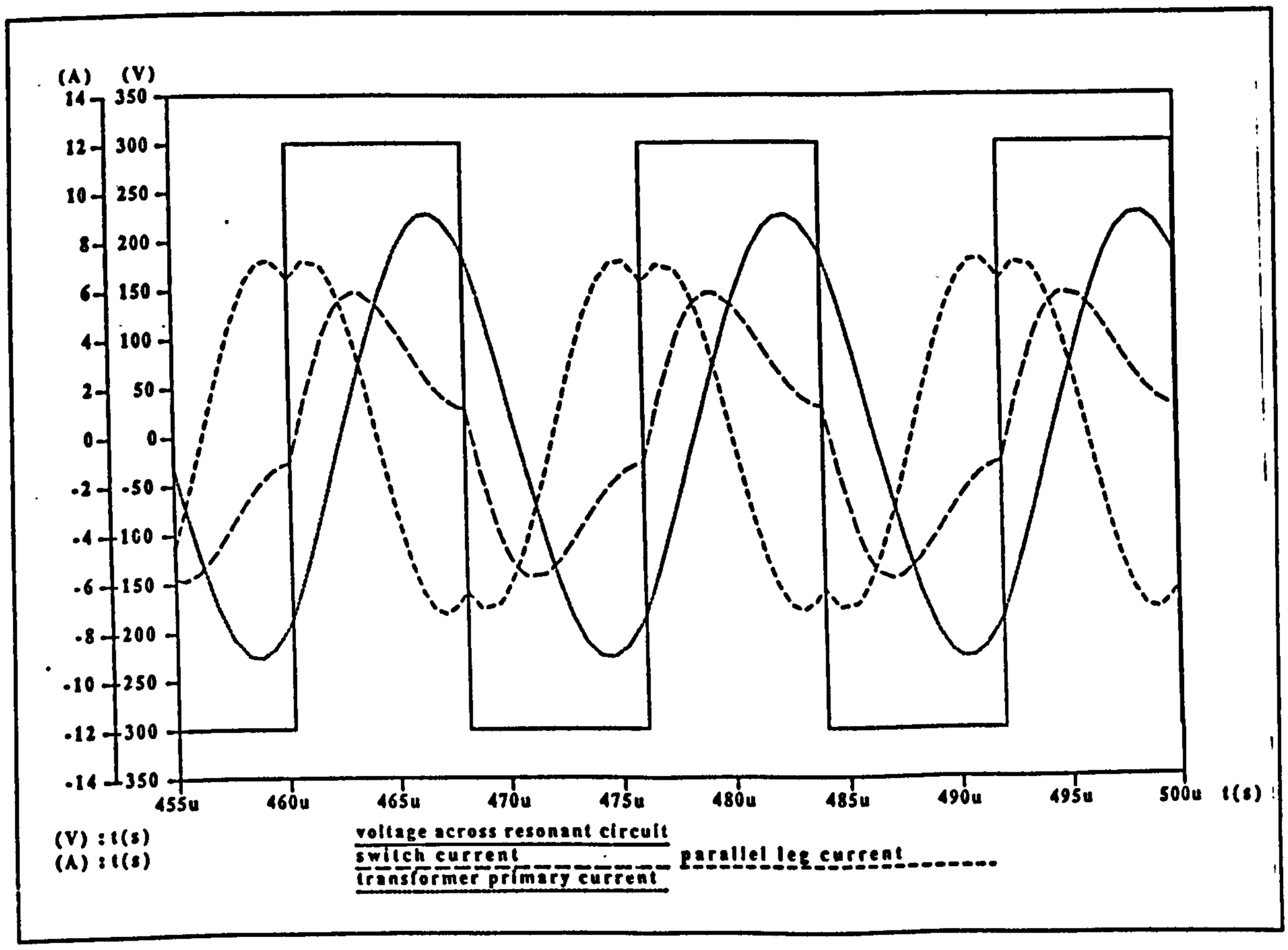


Figure 6.9(b) Circuit 2 driven at 63 kHz, the middle resonant frequency

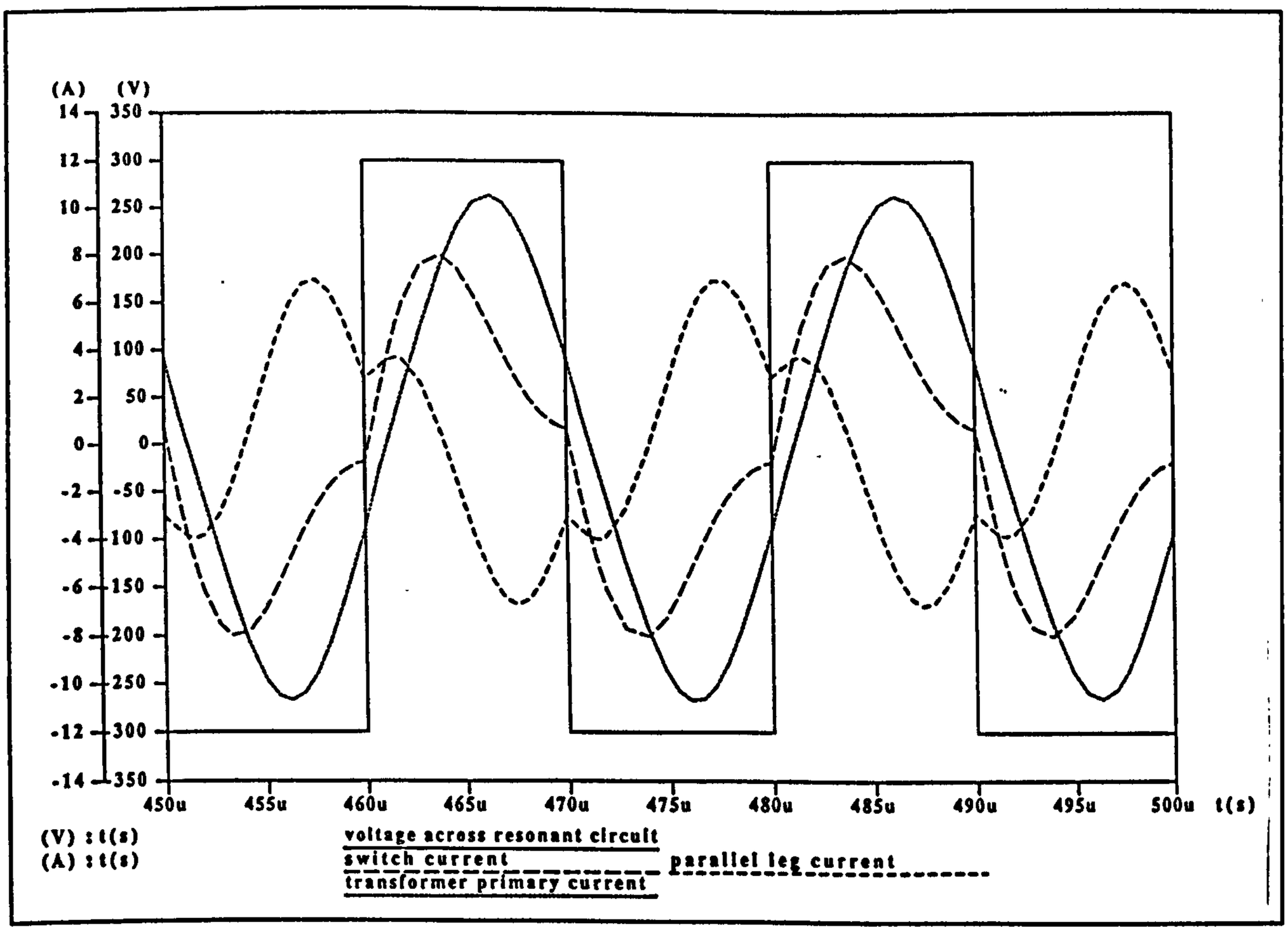


Figure 6.9(c) Circuit 2 driven at 50 kHz, the lower resonant frequency

The time domain results in Figure 6.9 (a,b,c) indicate that the upper and lower resonant frequencies of the load-resonant network as designed are very close to the values specified as inputs to the Mathematica Design program. The time domain simulation has however indicated that the middle resonant frequency is lower than the designed value of 75 kHz. When this was investigated further it was discovered that the same resonant components without the output rectifier did indeed have a middle resonant frequency of 75 kHz. It was found that increasing the load resistance in the simple model circuits had the same effect of shifting the middle resonant frequency as adding the output rectifier. It can be concluded therefore that the voltage drops in the output rectifier must be modelled as part of the load if the design mathematics is to be accurate. However, even with the slight changes in the resonant frequencies the load current magnitude in both high power and low power modes are very acceptable to justify construction of this design. The simulated load current, Fig. 6.10 (a and b) is 50 A ± 15 A in the low power mode and in high power mode is 150 A ± 30 A.

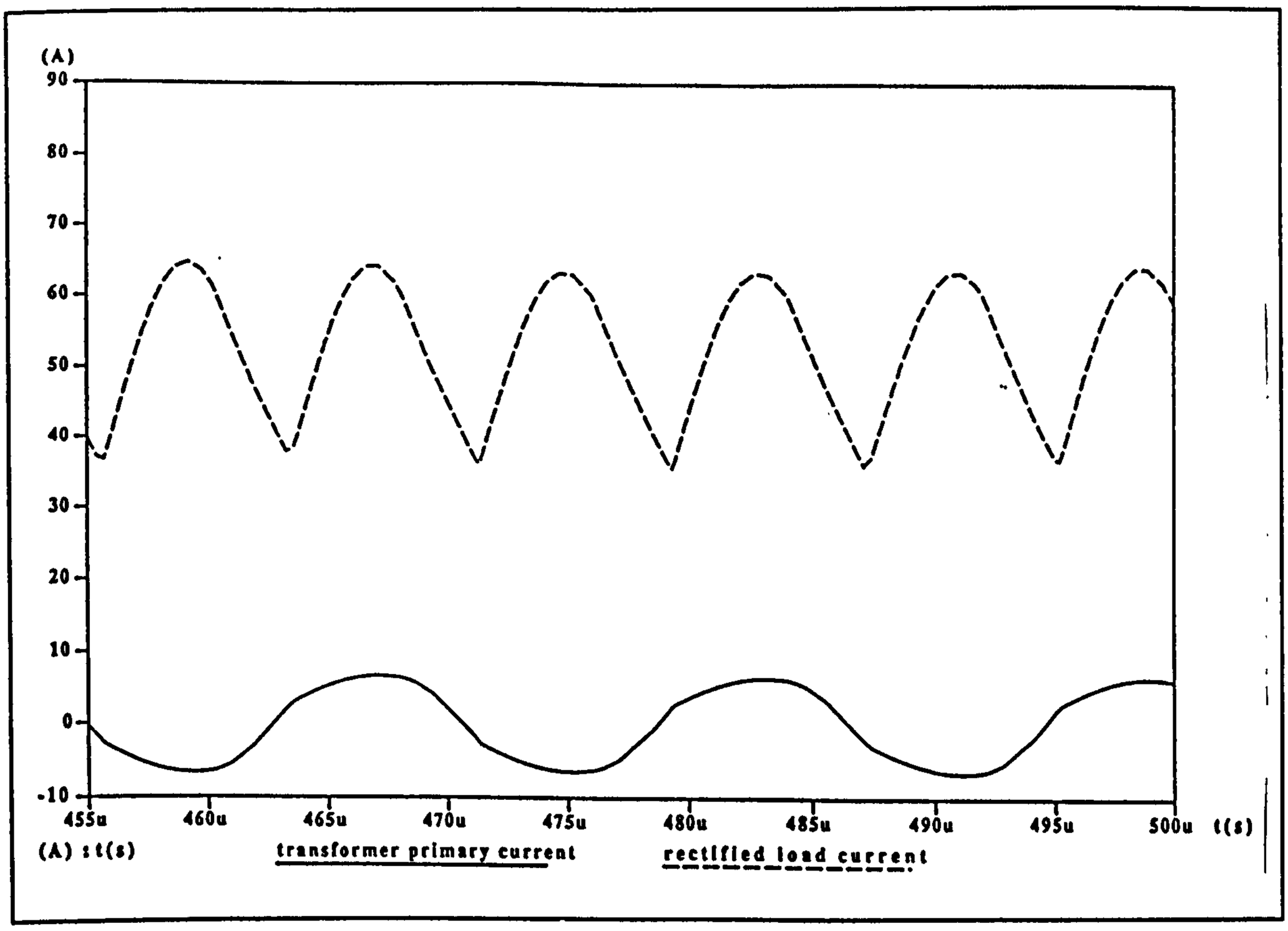


Figure 6.10(a) Complete circuit simulation of circuit 2 in low-power mode (63 kHz)

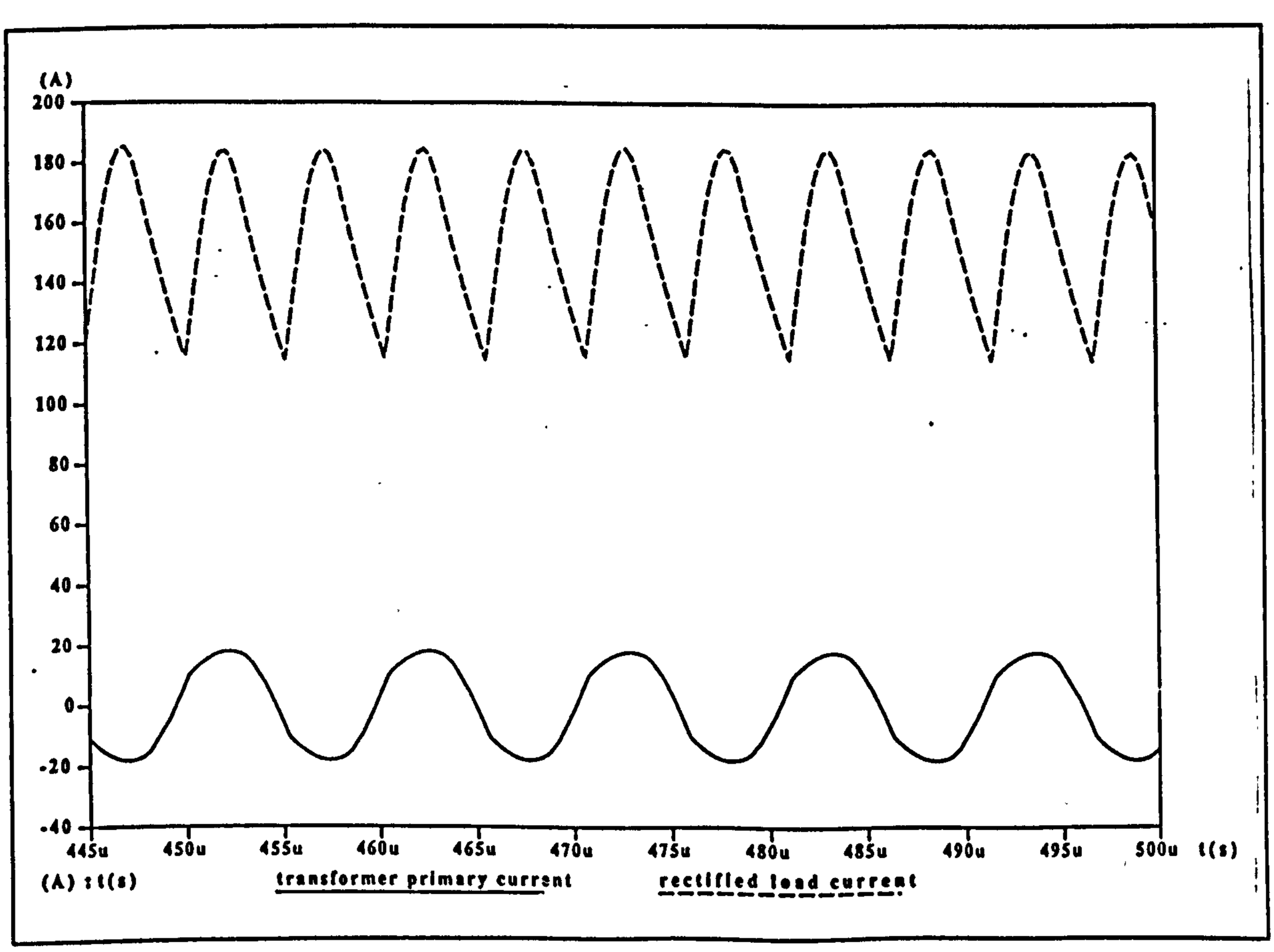


Figure 6.10(b) Complete circuit simulation of circuit 2 in high-power mode (97 kHz)

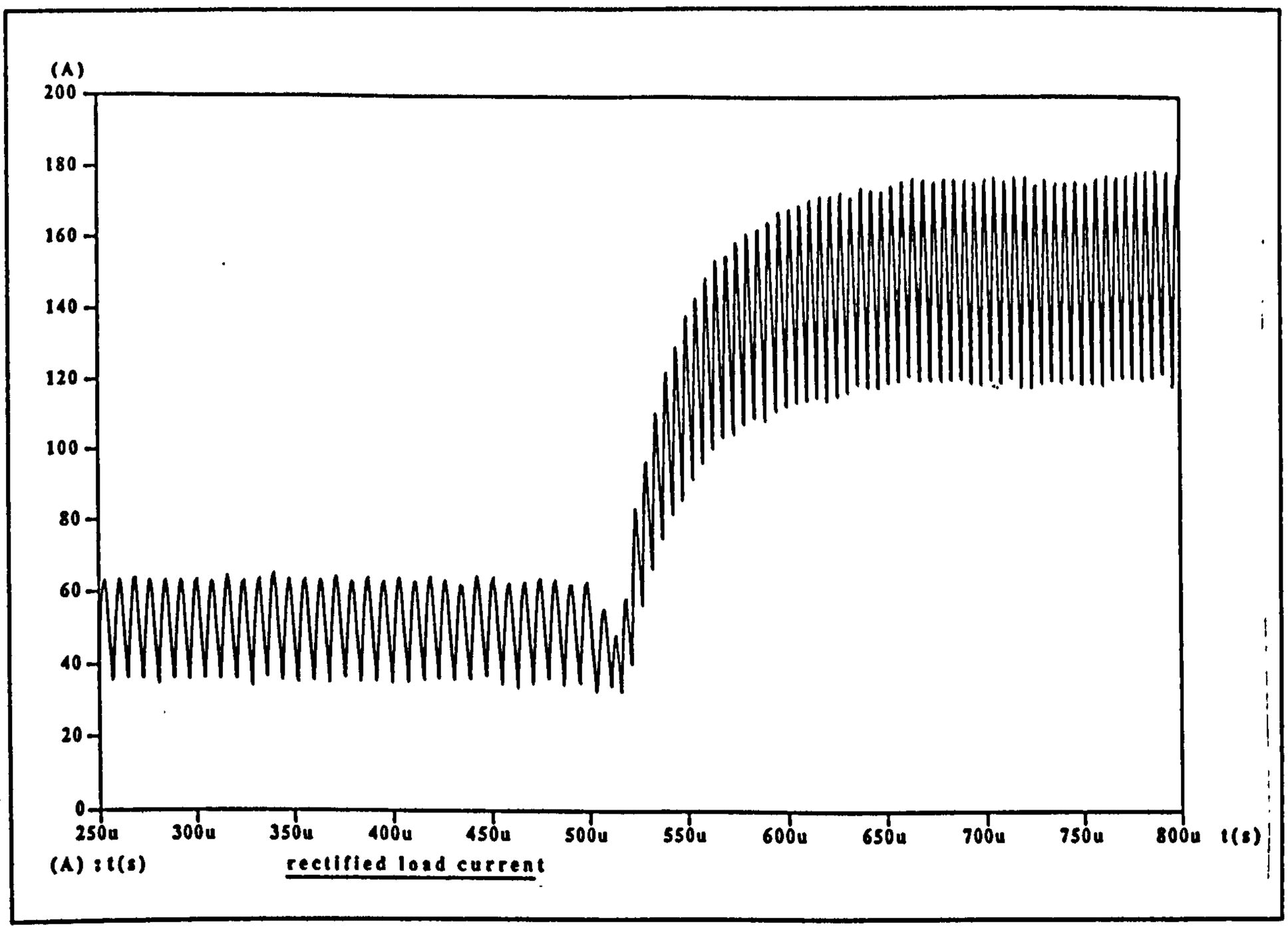


Figure 6.11 Simulation of Circuit 2 pulsing between 63 kHz and 97 kHz

Figure 6.11 shows the potential of the new control technique to achieve pulsed power output, the excitation frequency of the circuit was changed from 63 kHz to 97 kHz at the 500 μs point and the load current is seen to rise to the higher value within 100 μs.

To prove the capability of the series parallel resonant circuit under open-circuit and short-circuit conditions these situations were simulated for Circuit 2. Figure 6.12 shows the results of a frequency domain simulation with a short circuit applied across the input to the high-frequency rectifier. The resonant frequencies have changed slightly compared to the circuit under normal operating conditions (Figure 6.8). The switch current would be unacceptably high at the lower and upper resonant frequencies but is very well controlled at the middle resonant frequency. The circuit could therefore be designed to run continuously into a short circuit by selecting this frequency. Zero current switching would still be maintained. This is verified by the time domain simulation shown in Fig. 6.13 with a short circuit on the output of the transformer; the circuit was driven at 83 kHz. Although a peak current of 120 A flows in the short circuit, the switch current is a well controlled 3 A maximum.

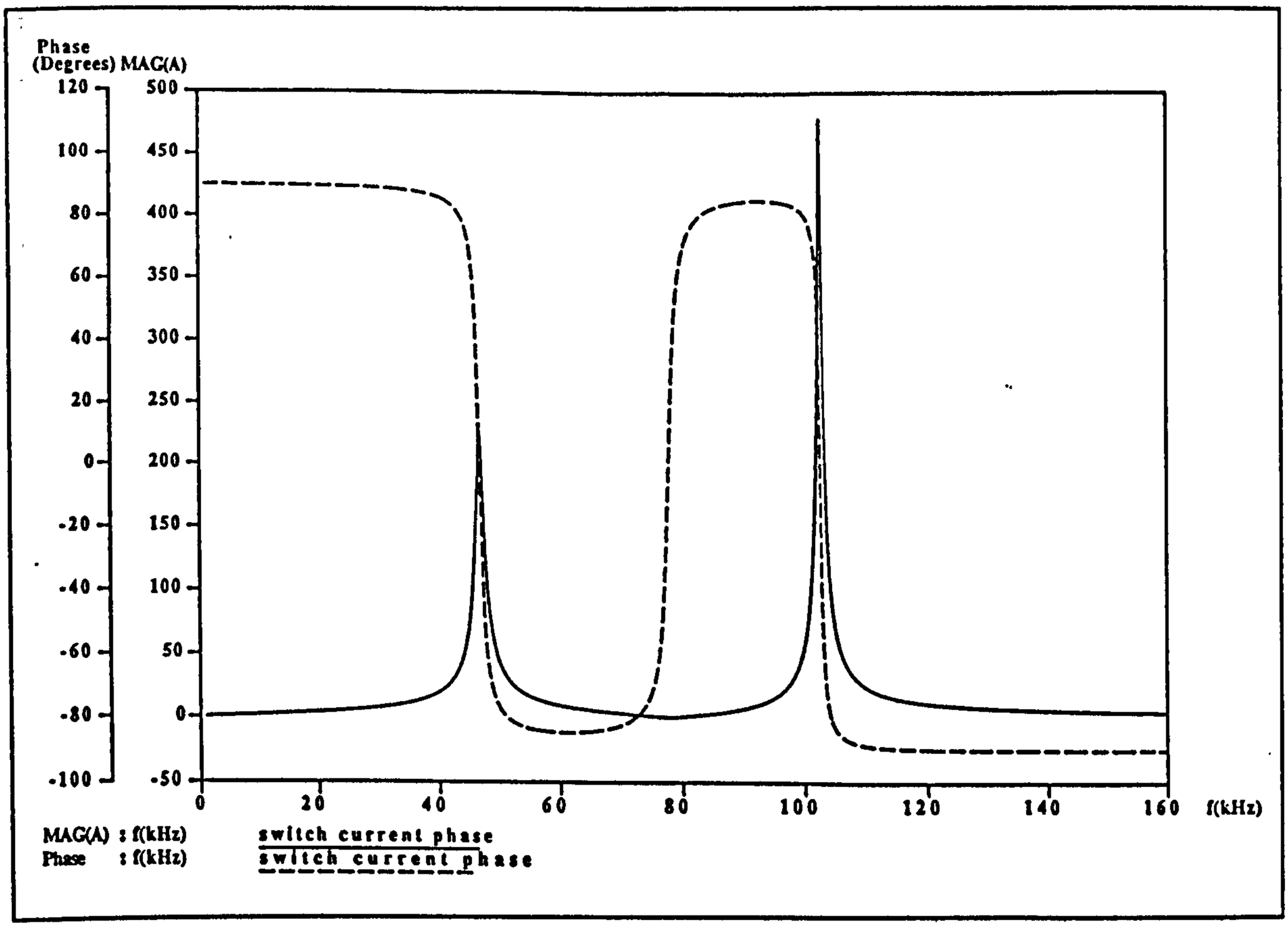


Figure 6.12 Circuit 2 under short-circuit conditions : Frequency domain simulation

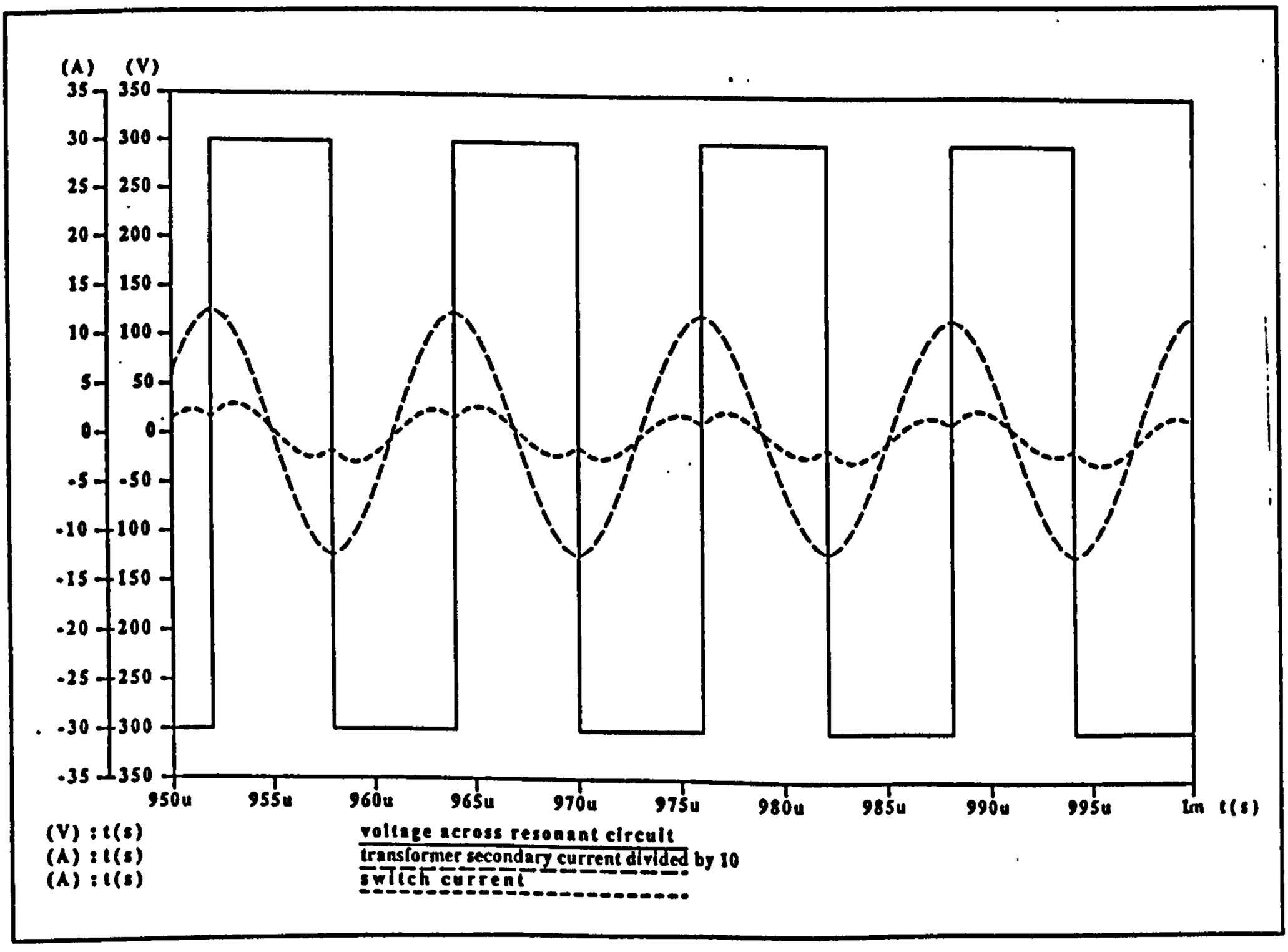


Figure 6.13 Circuit 2 under short-circuit conditions :
Time domain simulation running continuously at 83 kHz

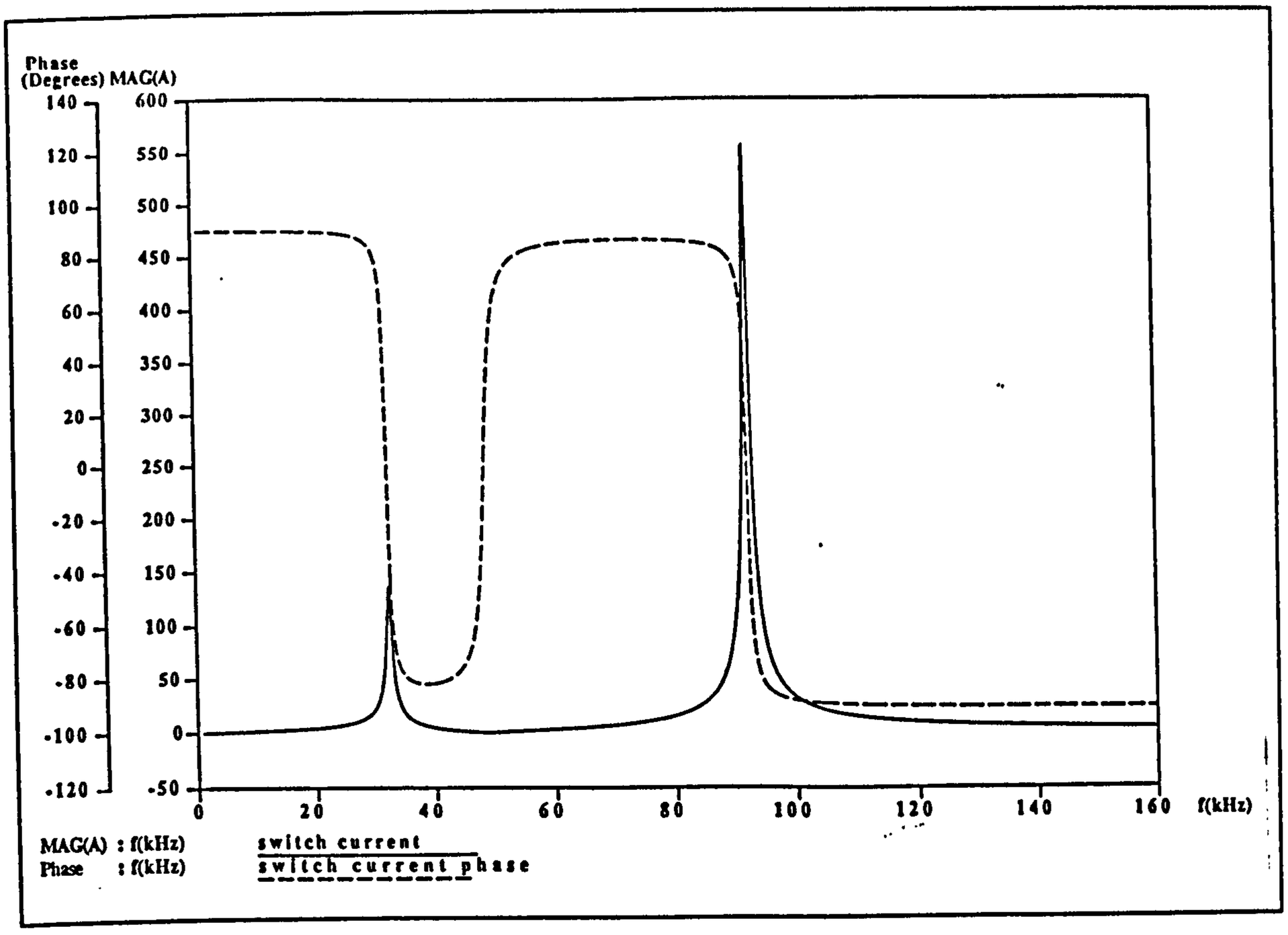


Figure 6.14 Circuit 2 under open-circuit conditions : Frequency domain simulation

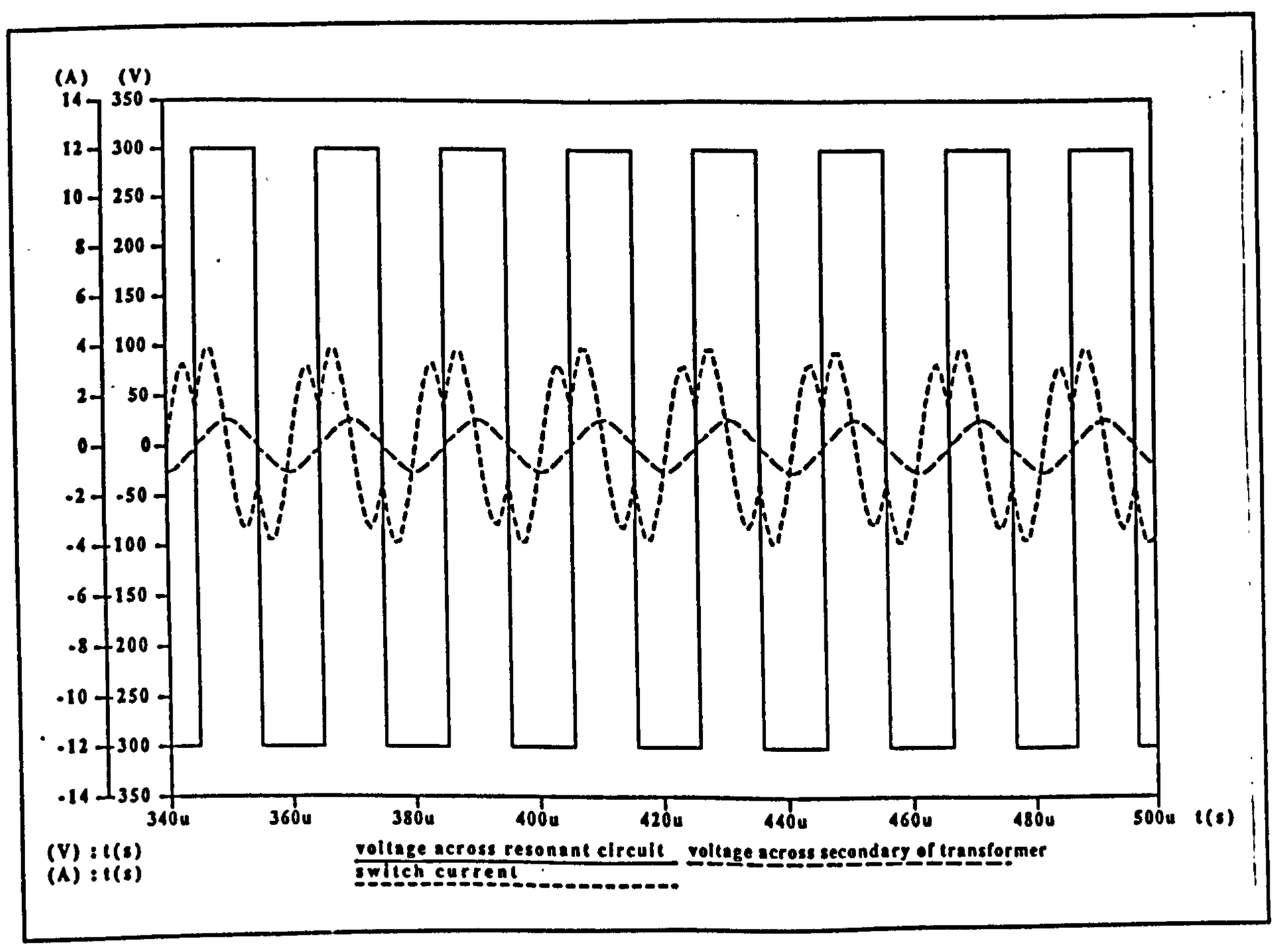


Figure 6.15 Circuit 2 under open-circuit conditions :

Time domain simulation driven at 49 kHz

With the load circuit open-circuit the frequency plot changes to the one shown in Figure 6.14. As with the short circuit condition, running the circuit for any length of time at either the upper or lower resonant frequencies would produce switch currents considerably in excess of those obtained in normal running modes. However, the time domain simulation shown in Figure 6.15 confirms that the circuit can run safely at the middle resonant frequency (49 kHz) with the load permanently open circuited, with switch currents no greater than 4 A. The open circuit voltage is no more than 25 V, with the converter running from a 600 V d.c. supply.

6.10 Conclusions

This Chapter has used the Mathematica Design program to design two different series-parallel load resonant converters. The first was designed to produce load currents of 200 A whereas the second has been designed primarily to illustrate the novel method of power control. Both circuits have been simulated in Saber to varying levels of complexity allowing the design Mathematics to be verified and the proposed control algorithms to be verified before the circuits were constructed. Minor variations between the specified circuit resistances and position of the resonant frequencies have been shown to be almost entirely due to the simple model of the rectifier used in the analysis.

CHAPTER 7 CONSTRUCTION OF TWO SERIES-PARALLEL LOAD-RESONANT CONVERTERS AND TEST RESULTS

7.1 Design of an active rectifier.

The two circuits designed in Chapter 6 operate from a controlled d.c. voltage. An active rectifier was used to produce a variable d.c. voltage from the incoming single phase a.c. supply. This provided the variable d.c. output voltage and drew sinusoidal current from the single phase a.c. supply. The active rectifier was a full-bridge diode rectifier and a buck-boost converter. This is shown in Figure 7.1.

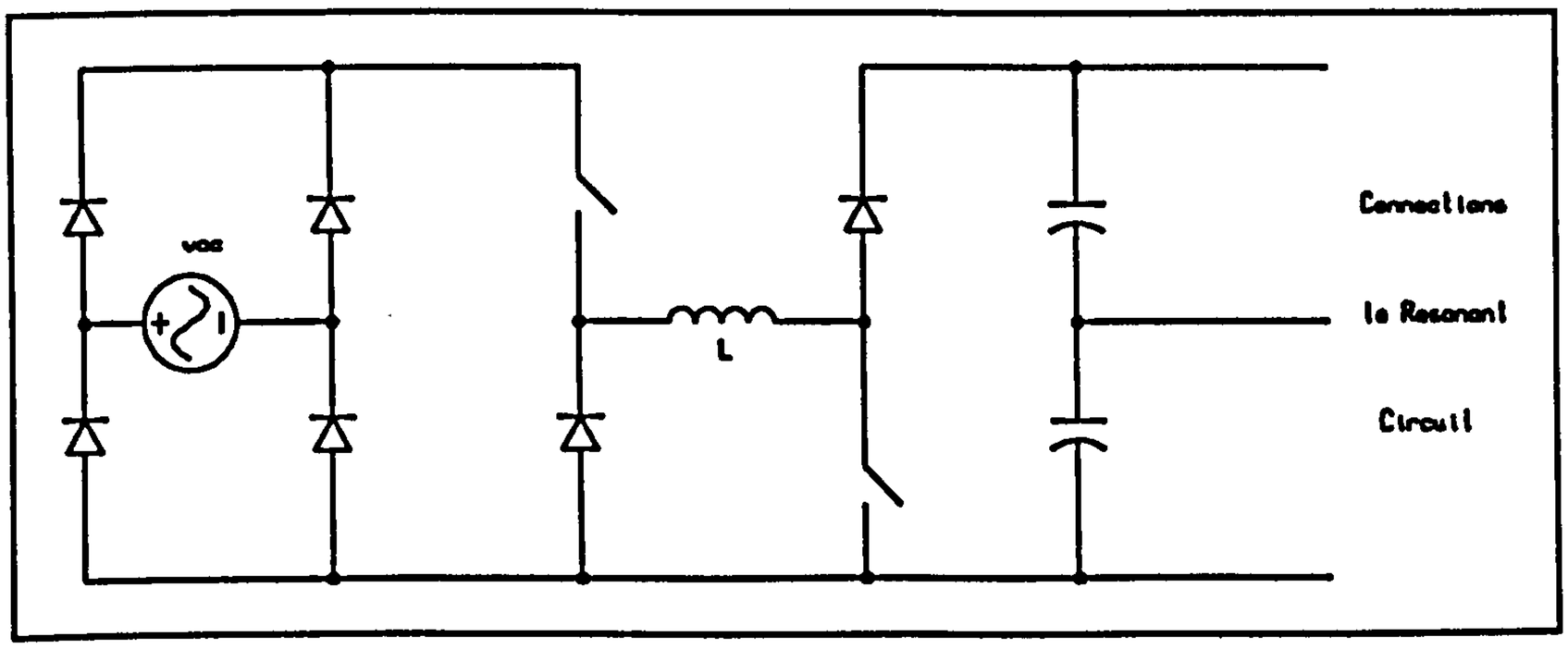


Figure 7.1 Active rectifier

In the buck-boost converter, the output voltage could be greater or less than the input voltage and was controlled by simultaneous repetitive switching of both power switches in the circuit. The duty cycle of the power switches determined the ratio of the output voltage to input voltage for a given switching frequency, load resistance and value of inductance.

The equations relating the output and input voltage of the converter are given in Appendix D.

To achieve unity-power factor the buck-boost converter was operated in the discontinuous-current mode i.e. the current in the inductor returned to zero during each switching cycle of the converter. The magnitude of the current in the inductor in each

switching cycle, and hence the magnitude of the current drawn from the supply, was then proportional to the a.c. supply voltage. Typical a.c. supply voltage and inductor current waveforms for this circuit operating in discontinuous current mode are shown in Figure 7.2.

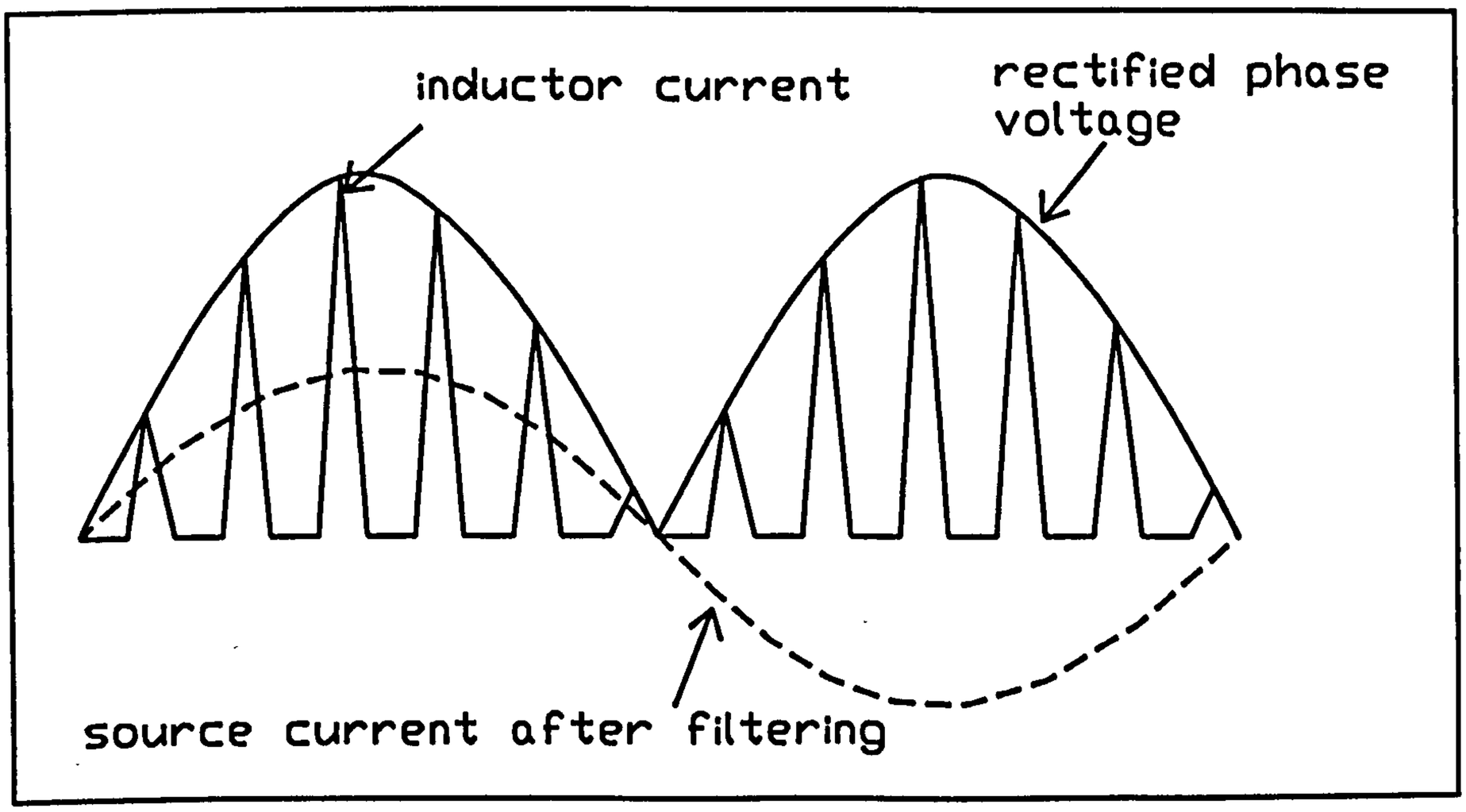


Figure 7.2 Typical voltage and inductor current waveforms for active rectifier operating in discontinuous-current mode

To achieve discontinuous current operation the inductor, in the active rectifier was designed so that, at the peak of the supply voltage, 340 V, the current in the inductor still had sufficient time to fall to zero before the next switching cycle began. The maximum value of L to maintain discontinuous current is shown in Appendix D to be given by

$$L_{boundary} = \left(\frac{E^2 D^2 T_S}{2}\right) P_{av} \tag{7.1}$$

A value of L of 110 μH was chosen for an average output power of 3 kW. The peak value of current at the peak-input voltage was calculated to be 73.7 A.
In the circuit 1 design the average power of 3 kW was achieved by pulsing the resonant circuit on and off at a 50% duty cycle.

7.2 Construction of complete series-parallel load-resonant converter

7.2.1 Power Component Selection

The complete series-parallel load-resonant converter is shown in Figure 7.3. The same power electronics and controller design were used for the experimental evaluation of both Circuit 1 and Circuit 2.

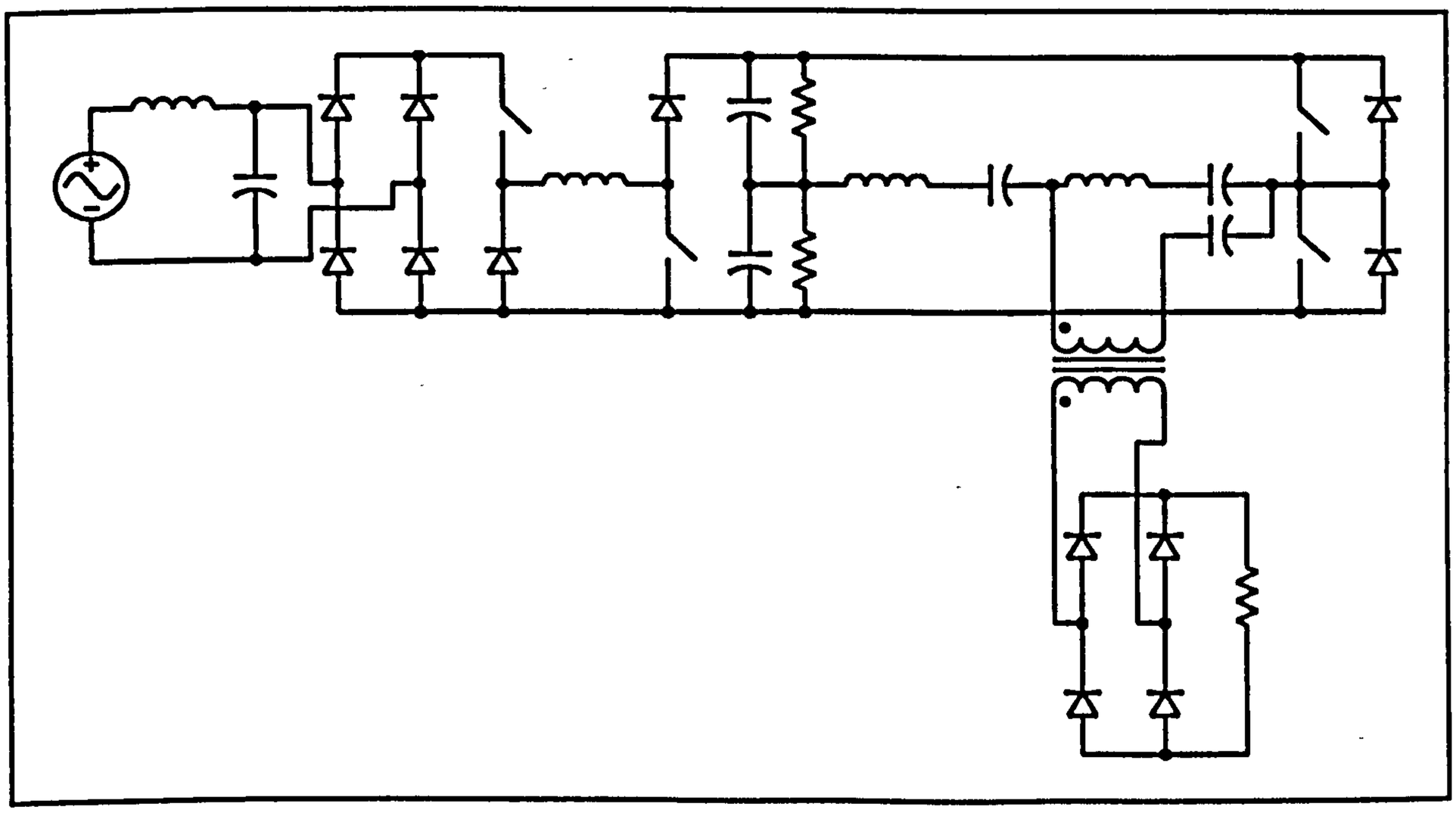


Figure 7.3 Complete power converter

The power switches used in the series-parallel load-resonant converter were IGBTs. These were chosen due to their low on-state losses, their availability in a module, the SKM50GB100D and their low cost as discussed in Chapter 3. This module contained two series connected 1000 V, 50 A IGBTs and a freewheeling diode in parallel with each IGBT. The module construction reduced the inductance between the two power switches to a minimum and controlled the current ringing and voltage overshoot during switching.

The resonant components in the circuit were polypropylene capacitors, inductors, wound on ferrite cores, and the leakage inductance of the isolation transformer. The

number of turns on the primary and secondary windings of the isolation transformer were chosen to minimise leakage inductance (small number of turns). This was achieved by using a single turn copper sheet for the secondary winding. The core size was not optimised again because it was more important to retain flexibility in subsequent use of the high-frequency transformer in different circuits. Additional inductance was added to the load-leg in series with the primary of the transformer to give the required value of inductance in the load-leg of the circuit. Polypropylene capacitors were used due to their approximately constant value of capacitance over a wide-frequency range and low value of inductance. It was important that the components chosen could operate with a number of different circuit arrangements and so they were all over-rated so as to avoid over-heating and saturation problems.

The secondary circuit of the isolation transformer consisted of a full bridge diode rectifier constructed from fast-recovery epitaxial diodes. A full bridge diode rectifier was used to keep the design of the transformer simple. These were paralleled to increase the current rating of the rectifier. The load used for testing both Circuits were ten 1.2Ω 150 W resistors in parallel.

7.2.2 Resonant frequency tracking controller design

The frequency analysis of the resonant circuits showed the variation in impedance of the resonant circuit against frequency at which the circuit was excited. The impedance of the circuit specified in the design procedure only occurred at resonance. In both of the constructed designs moving slightly away from resonance changed the impedance of the circuit significantly. A more sophisticated controller was developed which could take into account the switching delays in the circuit so that the frequency of operation of the circuit remained at resonance and, indeed, would automatically adjust for slight changes in the circuit parameters. The logic of the control circuits were implemented in a Xilinx logic cell array clocked by a 10 MHz crystal oscillator. A block diagram of the circuit is shown in Figure 7.4. Using the Xilinx system allowed easy modification of the design and proved suitability for possible eventual implementation in an ASIC (application specific integrated circuit).

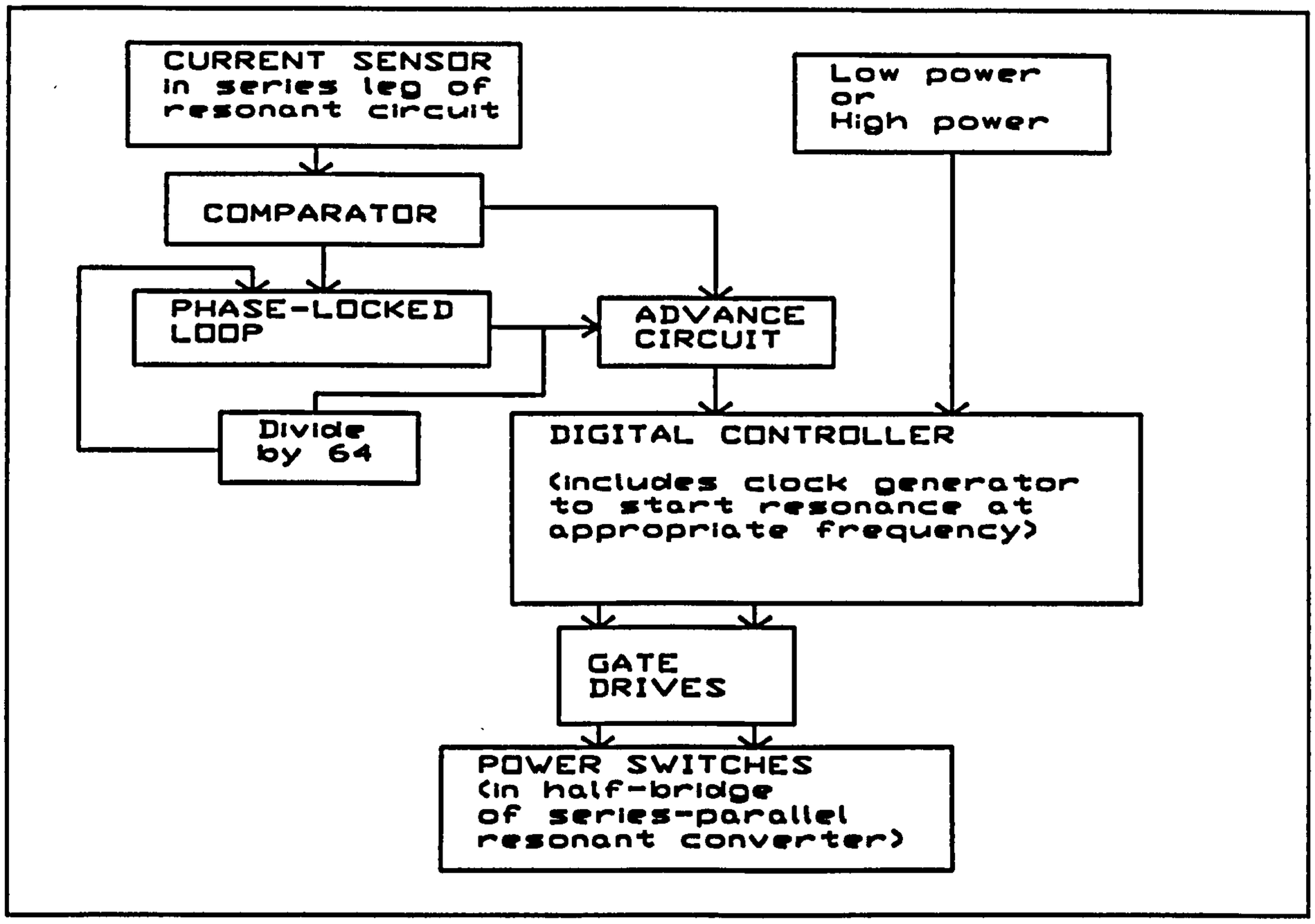


Figure 7.4 Block diagram of the resonant control circuit

(i) Current feedback

The current feedback circuit was the same as used in the series load-resonant converter in Chapter 4. The current monitored was the current in the series-leg of the circuit. A 5 V logic signal was generated which changed state each time the current passed through zero.

(ii) Phase-locked loop

To achieve optimum switching of the power devices, the comparator signal generated from the current feedback circuit was advanced. This was achieved using a phase-locked loop. The phase-locked loop device contained a linear voltage-controlled oscillator (v.c.o.). The voltage controlled oscillator signal was divided by 64 so that the v.c.o. signal could run at a frequency 64 times that of the input comparator signal (i.e. 6.4 MHz when operating at 100 kHz). The comparator signal and the v.c.o. output were held in-phase by the phase-locked loop. If the frequency of the comparator signal varied, the v.c.o. signal tracked the frequency, and each cycle of the

v.c.o. signal remained at 1/64th of the comparator cycle, regardless of the frequency of operation of the circuit.

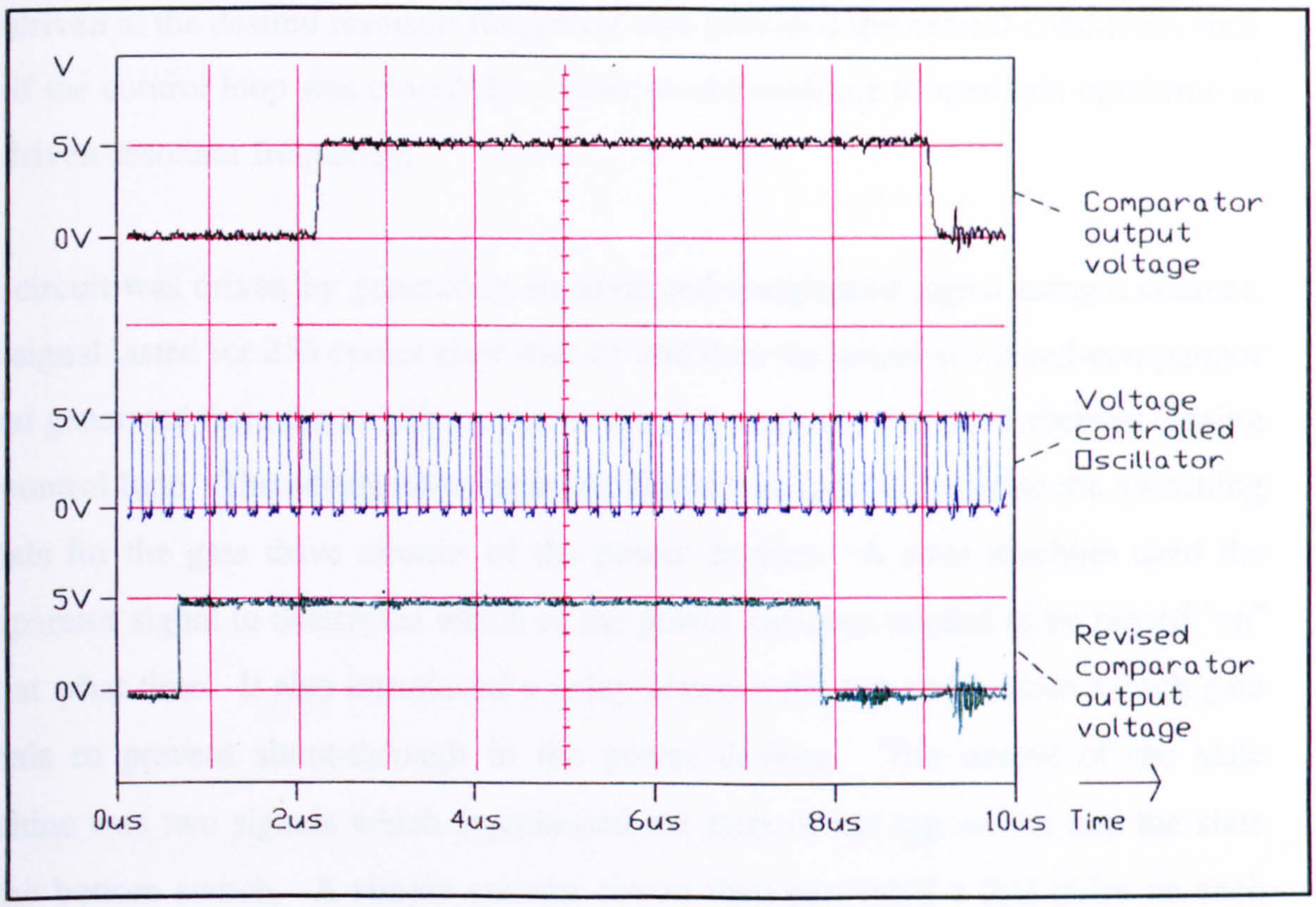


Figure 7.5 Comparator signal and advanced-comparator signal

Figure 7.5 shows a comparator signal derived from the resonant switch current, the v.c.o signal and the advanced-comparator signal. The comparator signal could be advanced by a different number of v.c.o. cycles on each edge. In Figure 7.5 the comparator signal was advanced by five v.c.o. cycles on each edge. The advance was increased until the zero- crossing of the power-switch current occurred at the same instant as the switching of the IGBTs. The final advance was 16 v.c.o cycles on the rising comparator edge and 15 on the falling edge. The asymmetry of the advanced-comparator signal was due to the difference in speed of the pulse transformer gate drive circuit and bottom switch gate drive circuit.

(iii) Digital Controller employing state machines

At start-up the resonant circuit was driven at a resonant frequency of the circuit. This was so that the switch current, monitored by the Hall-effect device, could build-up and

have a sufficient enough magnitude to be detected by the Hall-effect device before the comparator signal was used in the control circuit. It had been found that if the circuit was driven at the desired resonant frequency, this provided the correct conditions such that if the control loop was closed the circuit would continue to maintain operation at the driven resonant frequency.

The circuit was driven by generating an advanced-comparator signal using a counter. The signal lasted for 256 cycles after start-up and then the actual advanced-comparator signal generated from the switch current in the power circuit was used, thereby closing the control loop. The advanced-comparator signal was used to generate the switching signals for the gate drive circuits of the power devices. A state machine used the comparator signal to determine which of the power switches needed to be turned 'on' and at what time. It also introduced a delay between the top and bottom switch gate signals to prevent shoot-through in the power devices. The output of the state machine was two signals which represented the state of the top switch and the state of the bottom switch. A simple counter circuit then generated a 2μs pulse on each edge of the top switch signal to turn the pulse transformer gate drive on and off. State machines were used in the logic design to minimise the problems caused by noise in asynchronous logic design. The state machine was written in the PAL logic language and implemented in the Xilinx logic-cell array. Details are given in Appendix E.1.

(iv) Pulse-transformer gate drive

Figure 7.6 shows the isolated pulse-transformer circuit used to drive the top switch of the series-parallel resonant converter. This had the same configuration as the isolated gate drive circuit of the series load-resonant converter but the circuit was modified to operate at frequencies up to, and including, 100 kHz. The primary stage of the gate drive was modified to be driven from a ± 15 V supply. A dual channel power driver device, UC3707, containing two totem pole outputs was used on the primary side of the isolation transformer to produce a voltage swing of 30 V across the primary of the pulse-transformer. The pulse-transformer had a turns ratio of 2:1. The secondary circuit used the positive and negative voltage pulses to produce a + 15 V and - 5 V

swing at the gate of the IGBT as before. These modifications improved the speed of the gate drive at high switching currents. The bottom switch gate drive circuit remained a conventional non-isolated transistor push-pull circuit.

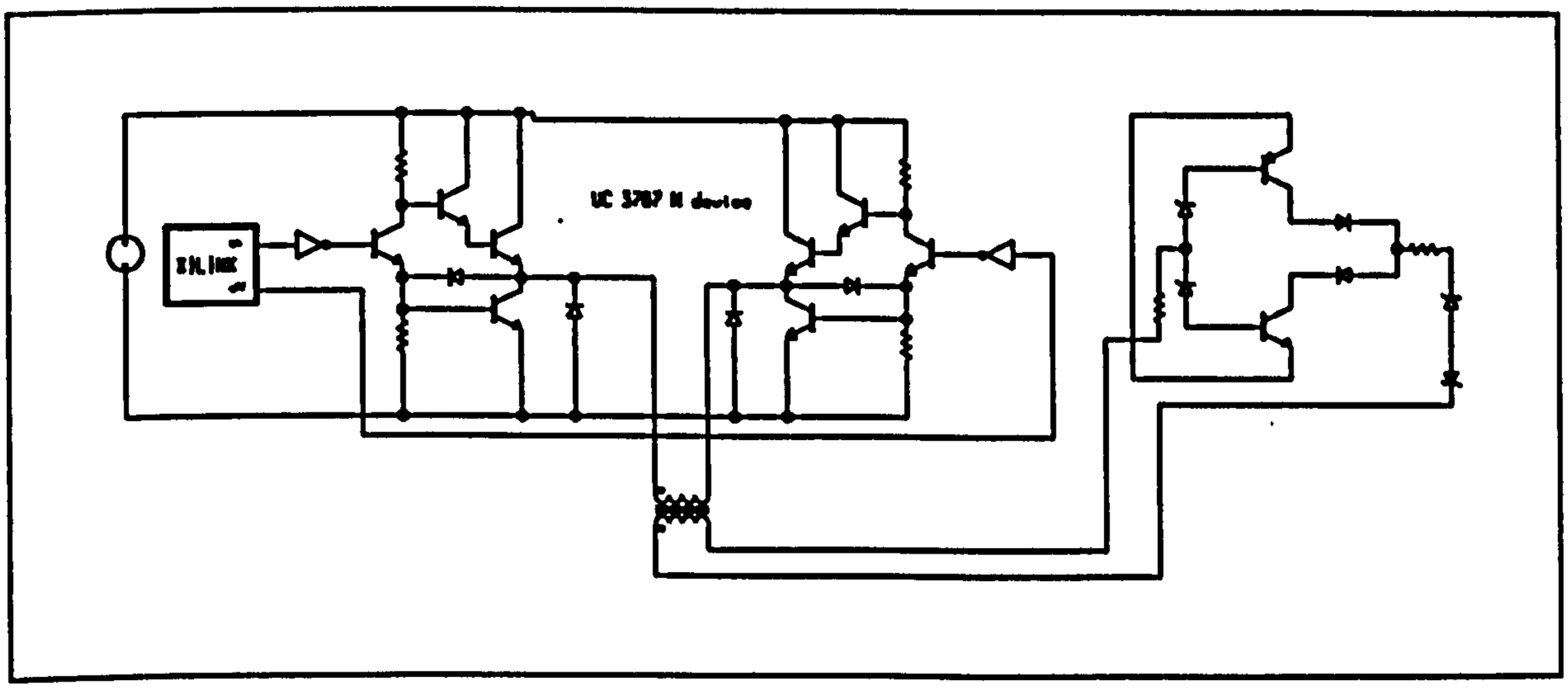


Figure 7.6 Revised pulse-transformer gate drive

7.3 Results of series-parallel load-resonant converter - Circuit 1

Circuit 1 was constructed. The component values were chosen as close as possible to the values calculated and specified in Table 6.1. The resonant frequencies of the circuit were accurately located by first driving the resonant circuit and the inverter from a variable frequency signal generator. This allowed the circuit to be excited at a range of frequencies, determining the frequencies at which the current and voltages across the circuit were in-phase. In Circuit 1 the upper resonant frequency was found to be at 83 kHz, compared with 82 kHz predicted from the simulation. The other resonant frequencies were identified using this technique at 65 kHz and 40 kHz (67 kHz and 38 kHz respectively in simulation).

Figure 7.7 (a,b,c) shows the experimental circuit waveforms of Circuit 1 operating at its highest resonant frequency of 83 kHz. Figure 7.7 (a) shows that the voltage across the bottom switch of the power converter is in-phase with the switch current at a frequency of 83 kHz. Figure 7.7(b) shows the relative magnitudes of the currents in each leg of the circuit, at the top resonant frequency of the circuit, when the circuit was in full operation.

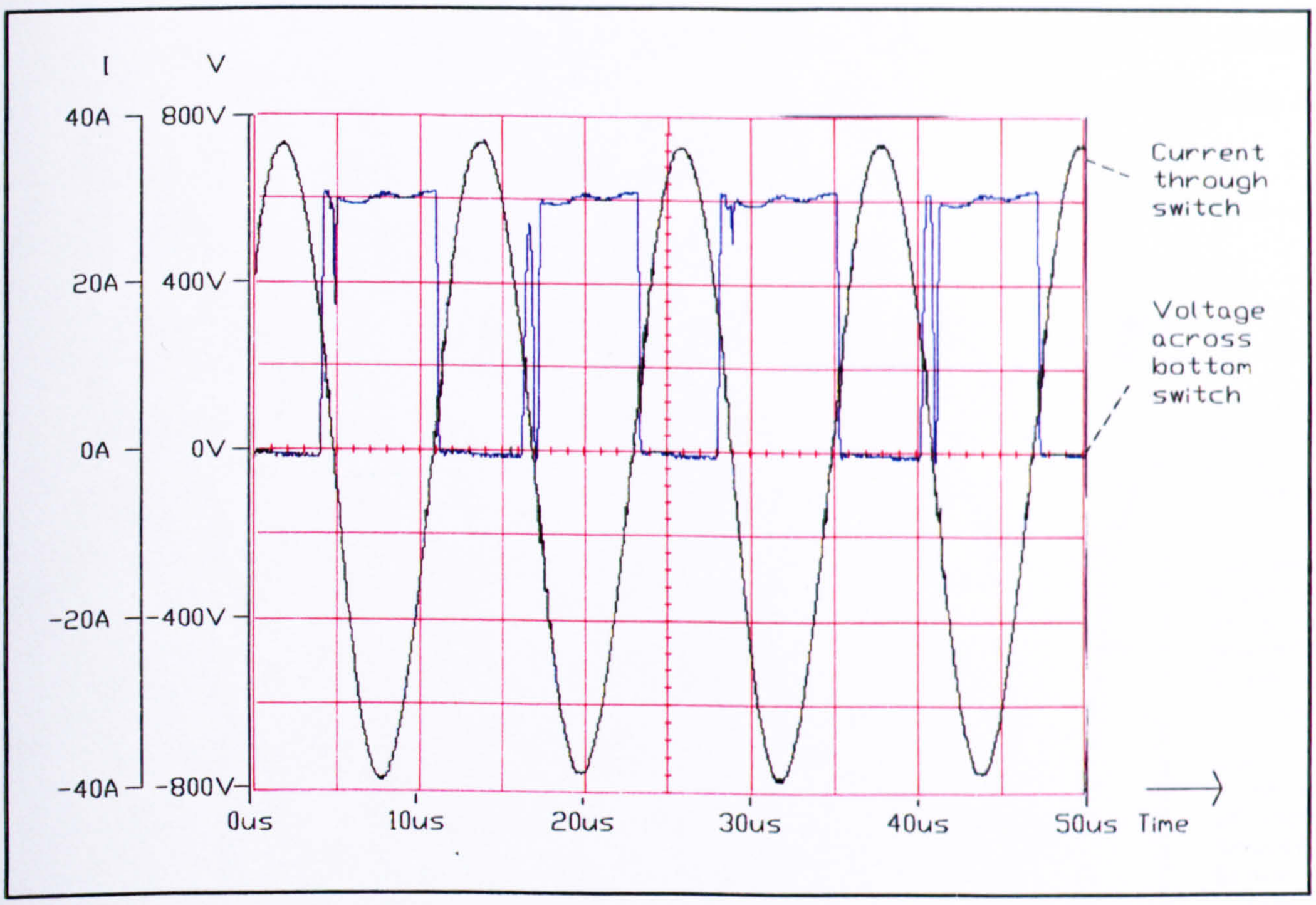


Figure 7.7 (a) Circuit 1, voltage across and currents through switch at highest-resonant frequency

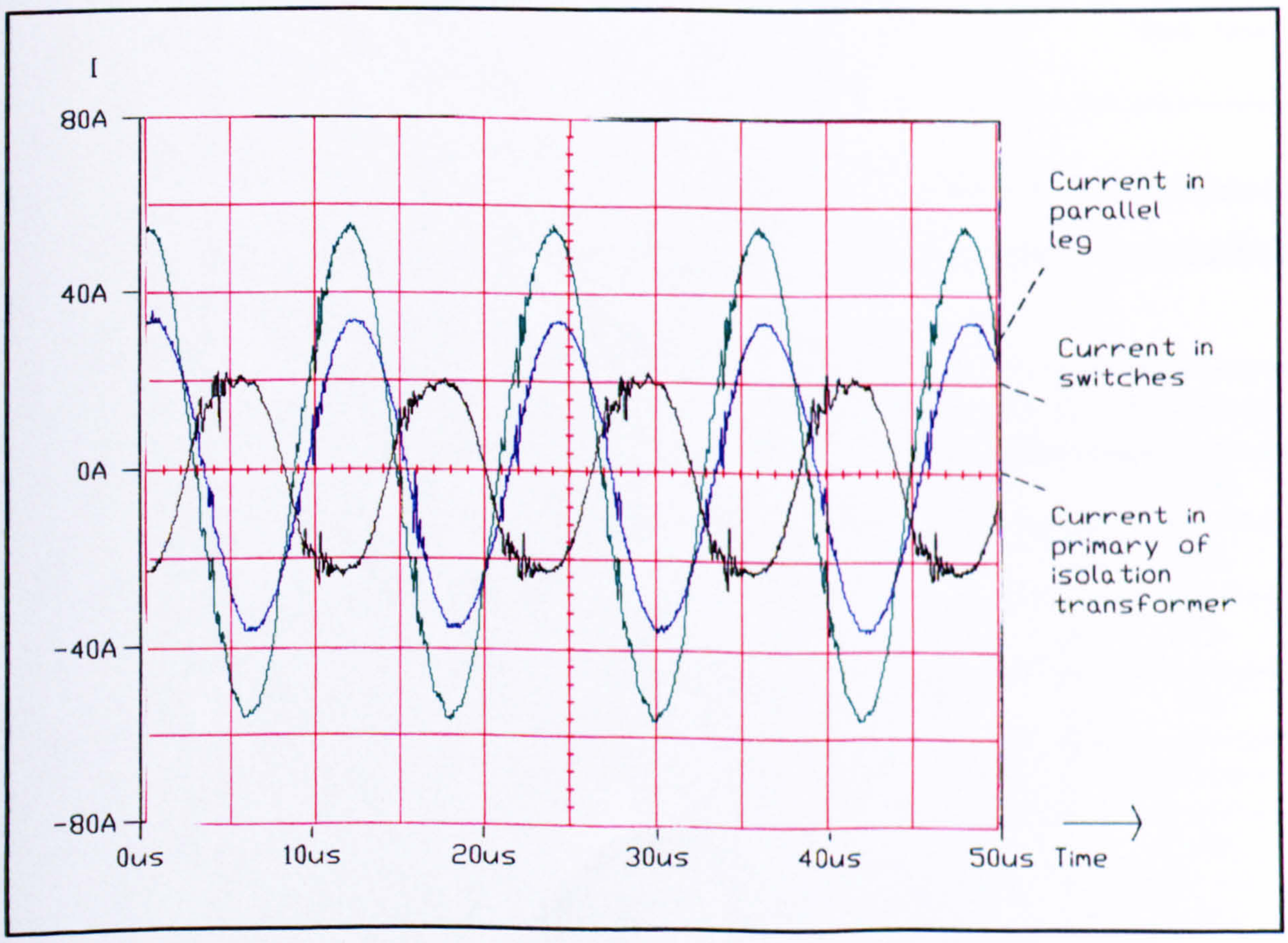


Figure 7.7 (b) Circuit 1, currents in each leg of the circuit at highest-resonant frequency

Figure 7.5(c) shows the currents in the primary and secondary of the isolation transformer and the rectified current in the load. All results were taken when the d.c. supply voltage to the converter was 600 V.

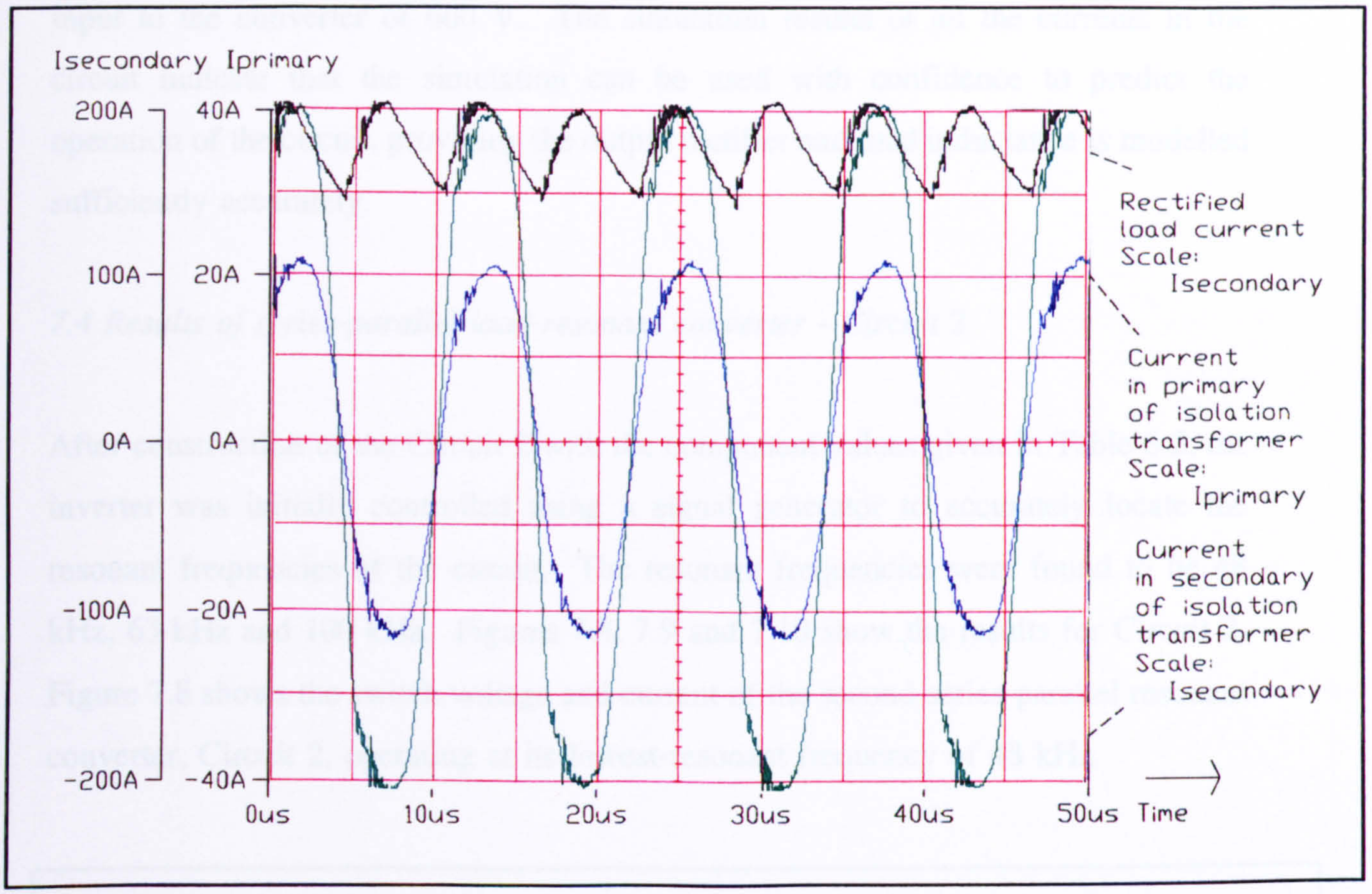


Figure 7.7 (c) Circuit 1, transformer and rectified load current at highest-resonant frequency

The comparison between the experimental and simulated results is summarised in Table 7.1.

	Experimental result	Simulated result
Upper resonant frequency	83 kHz	82 kHz
Peak Switch Current @ Upper Resonant Frequency	34 A	38 A
Peak current in parallel leg at upper resonant frequency	55 A	57 A
Peak Current in primary of transformer at upper resonant frequency	20 A	20 A
Load Current at upper resonant frequency	180 A ± 30 A	180 A ± 30 A
Middle Resonant Frequency	65 kHz	67 kHz
Lower Resonant Frequency	40 kHz	38 kHz
Apparent circuit resistance, R_{TOT}, at the upper resonant frequency	11.2 Ω	10 Ω

Table 7.1 Comparison between experimental and simulated results for Circuit 1

These results showed that the design procedure of the series-parallel resonant converter has produced a circuit with very close to the specified resonant frequencies. A rectified load current of peak value in excess of 200 A was achieved from a d.c. input to the converter of 600 V. The simulation results of all the currents in the circuit indicate that the simulation can be used with confidence to predict the operation of the circuit, providing the output rectifier and load inductance is modelled sufficiently accurately.

7.4 Results of series-parallel load-resonant converter - Circuit 2

After construction of the Circuit 2 with the component values given in Table 6.2, the inverter was initially controlled using a signal generator to accurately locate the resonant frequencies of the circuit. The resonant frequencies were found to be 48 kHz, 63 kHz and 100 kHz. Figures 7.8, 7.9 and 7.10 show the results for Circuit 2. Figure 7.8 shows the switch voltage and current of the second series-parallel resonant converter, Circuit 2, operating at its lowest-resonant frequency of 48 kHz.

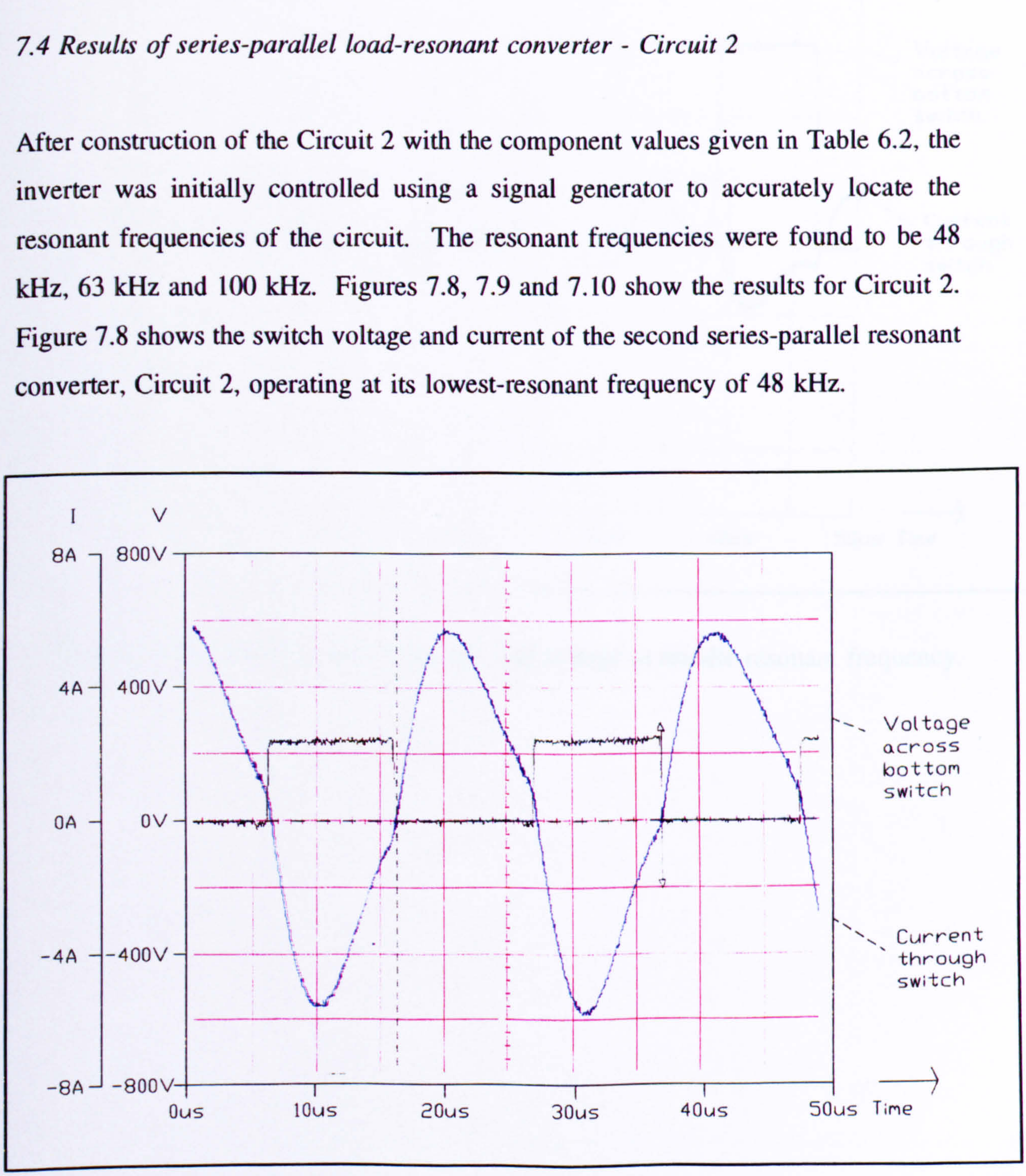


Figure 7.8 Circuit 2, switch current and voltage at lowest-resonant frequency.

Figure 7.9(a) shows the switch voltage and current of the converter operating at its middle-resonant frequency, 63 kHz, and Figure 7.9(b) shows the currents in each leg of the circuit at this frequency. Figure 7.9 (c) shows the current in the primary of the transformer and the d.c. current in the load when operating from a 600 V d.c. supply.

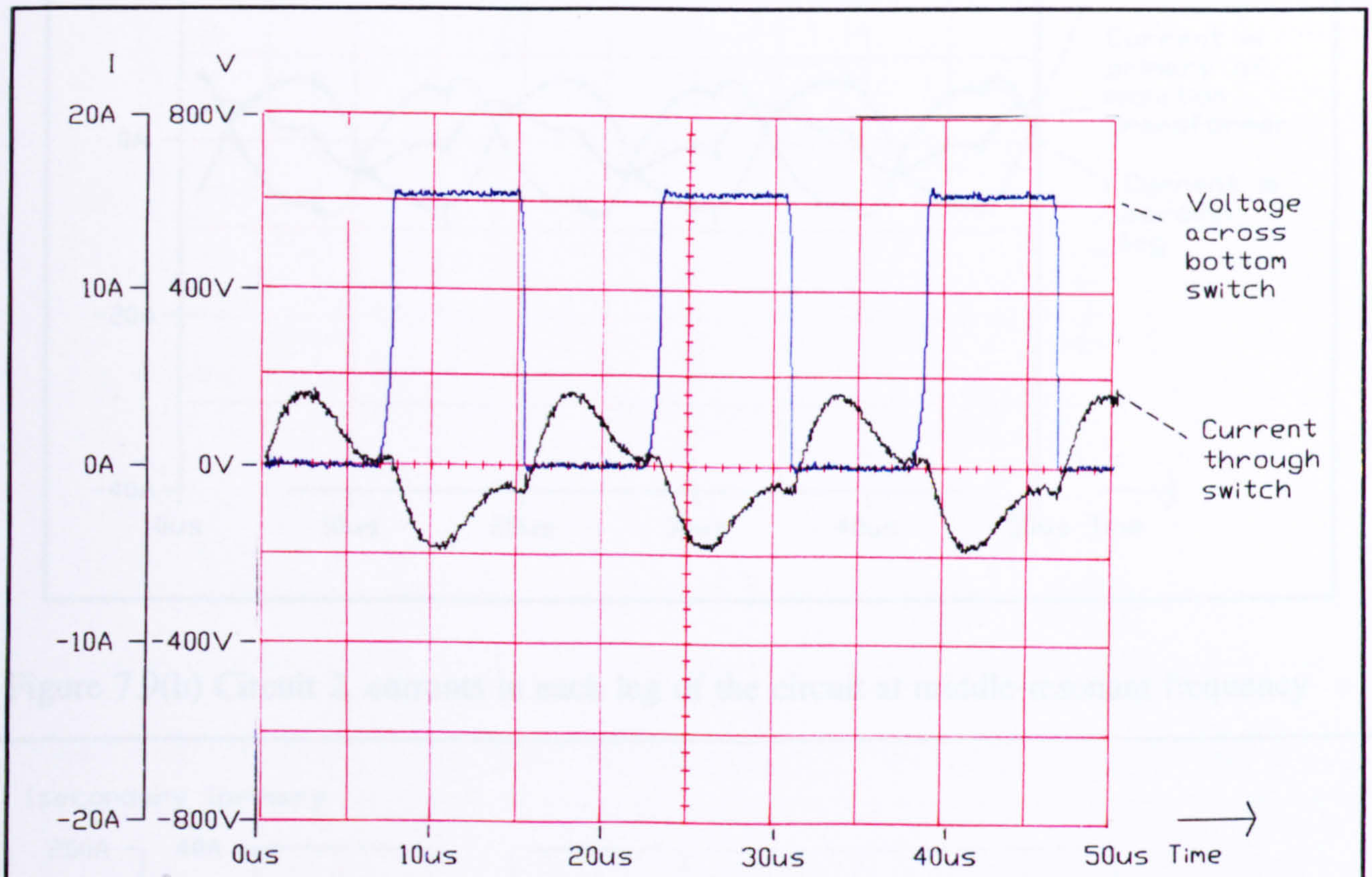


Figure 7.9 (a) Circuit 2, switch current and voltage at middle-resonant frequency.

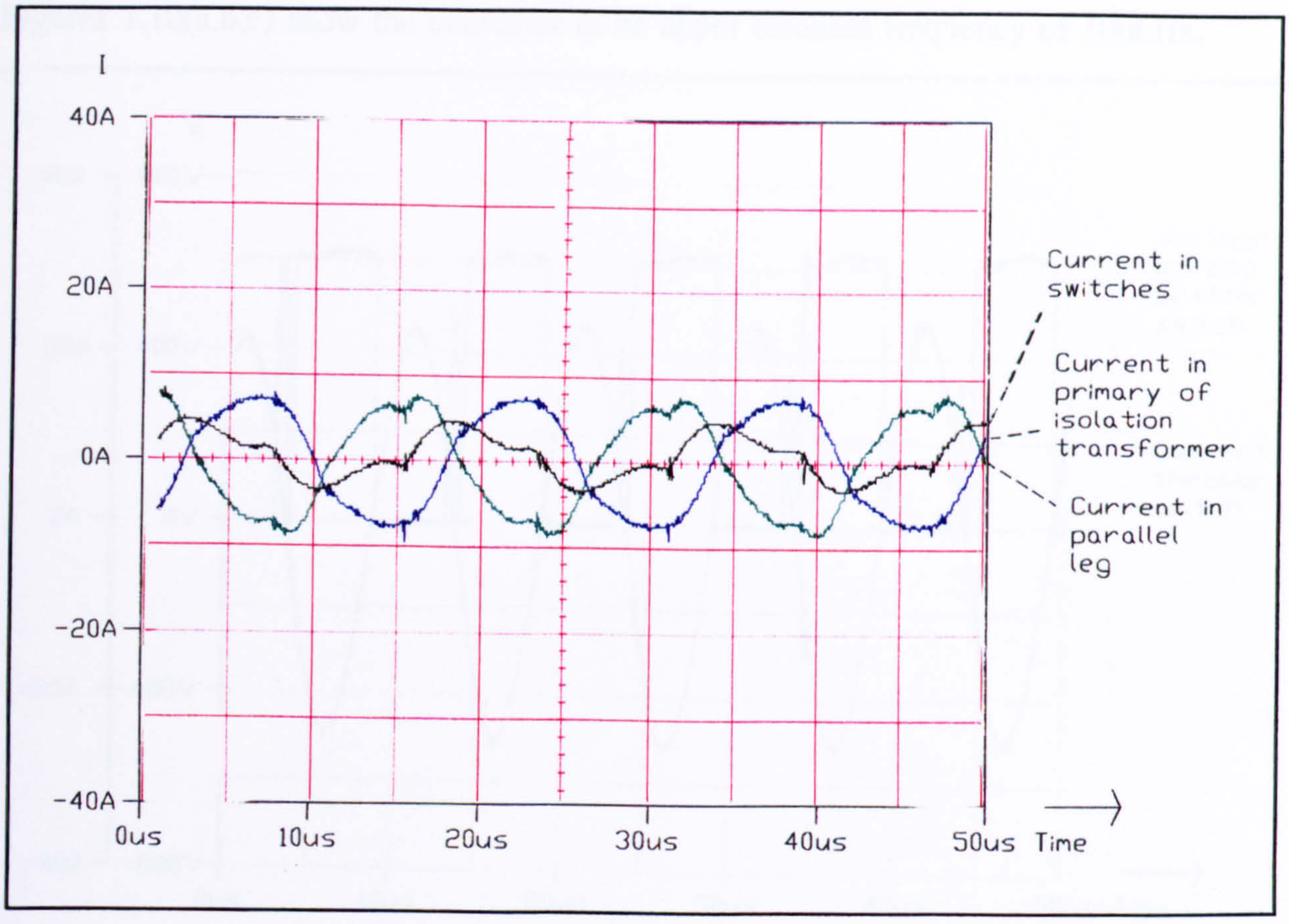


Figure 7.9(b) Circuit 2, currents in each leg of the circuit at middle-resonant frequency

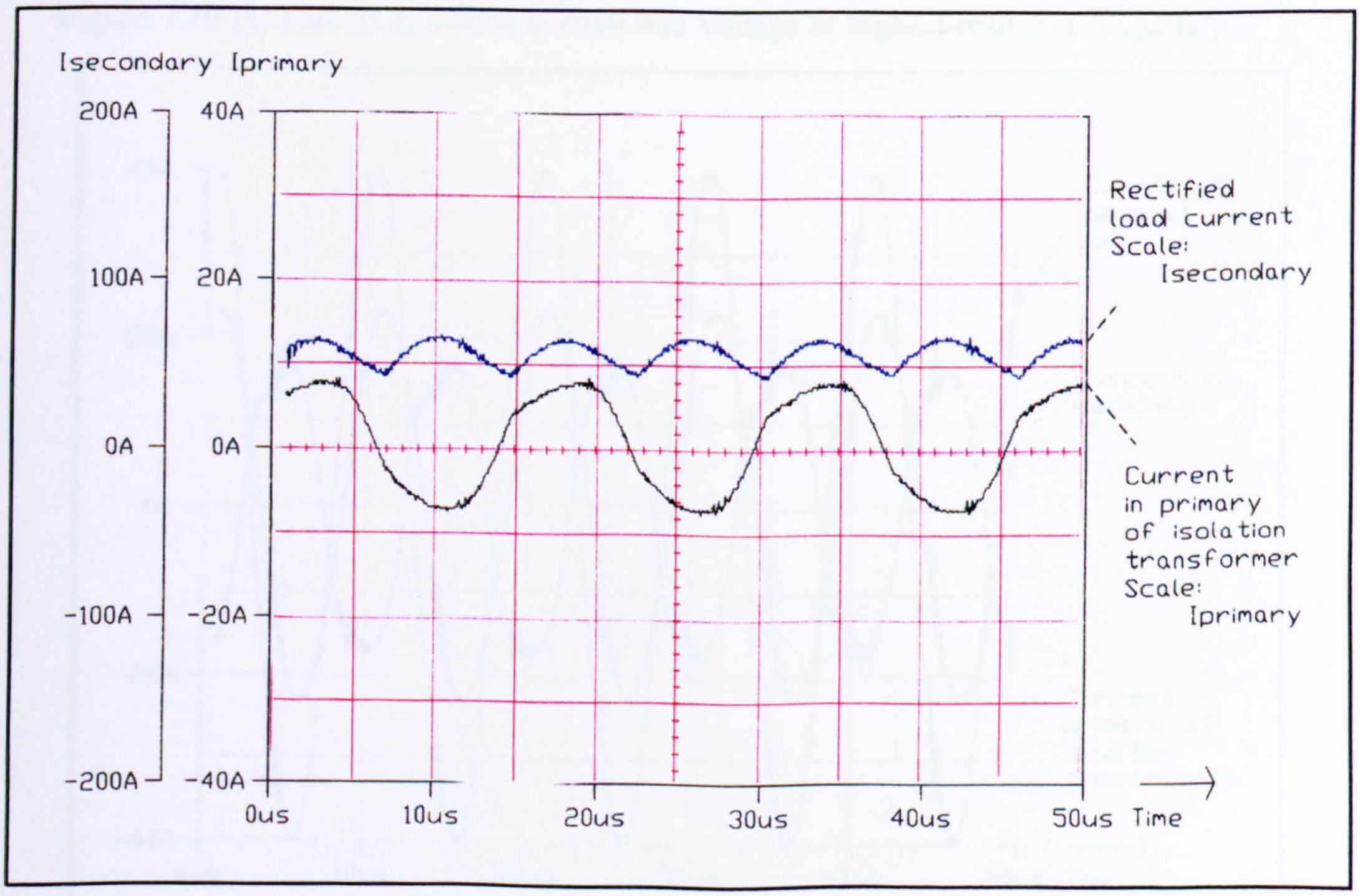


Figure 7.9(c) Circuit 2, transformer currents and rectified load current at middle-resonant frequency

Figures 7.10(a,b,c) show the converter at its upper resonant frequency of 100kHz.

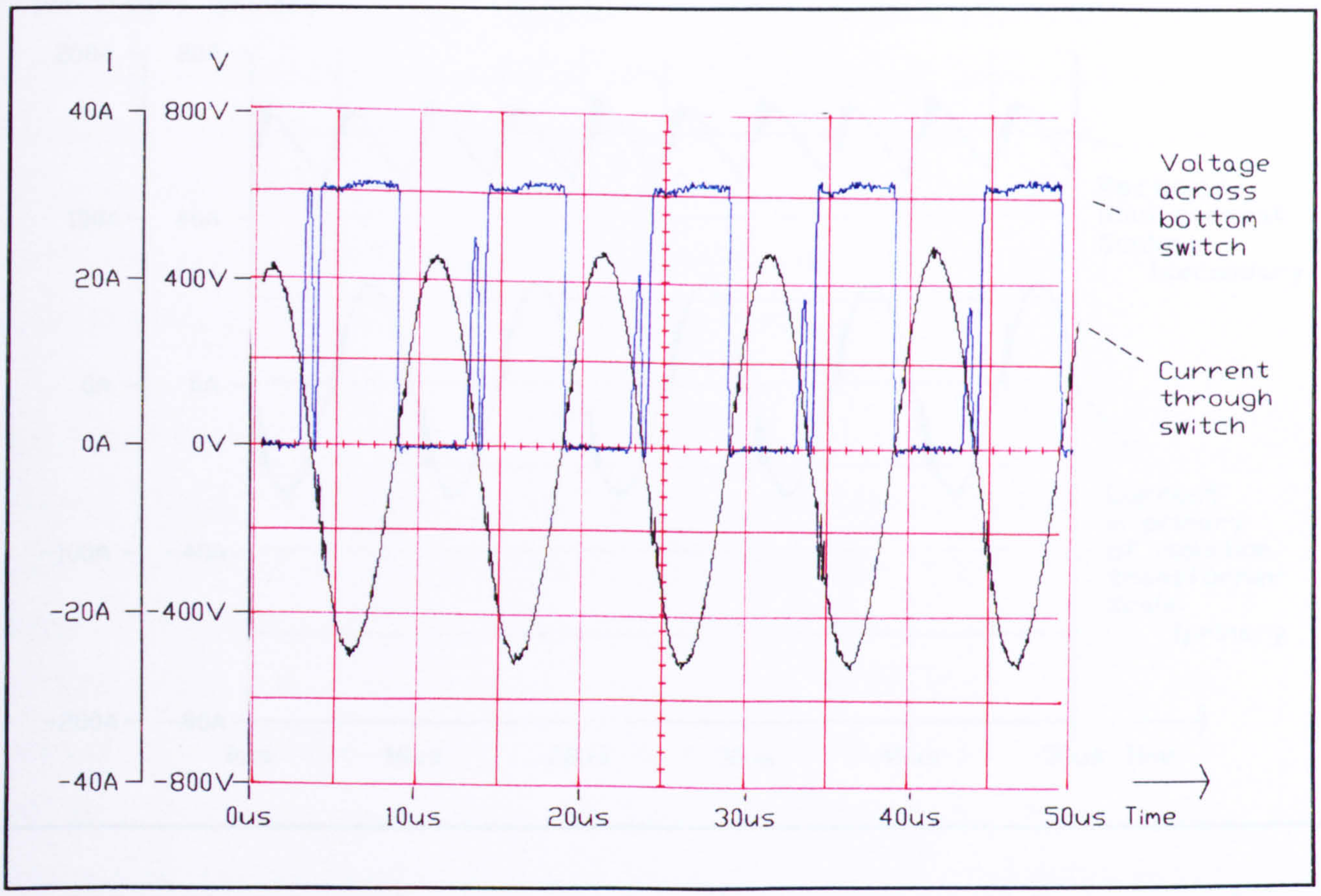


Figure 7.10 (a) Circuit 2, switch current and voltage at highest-resonant frequency

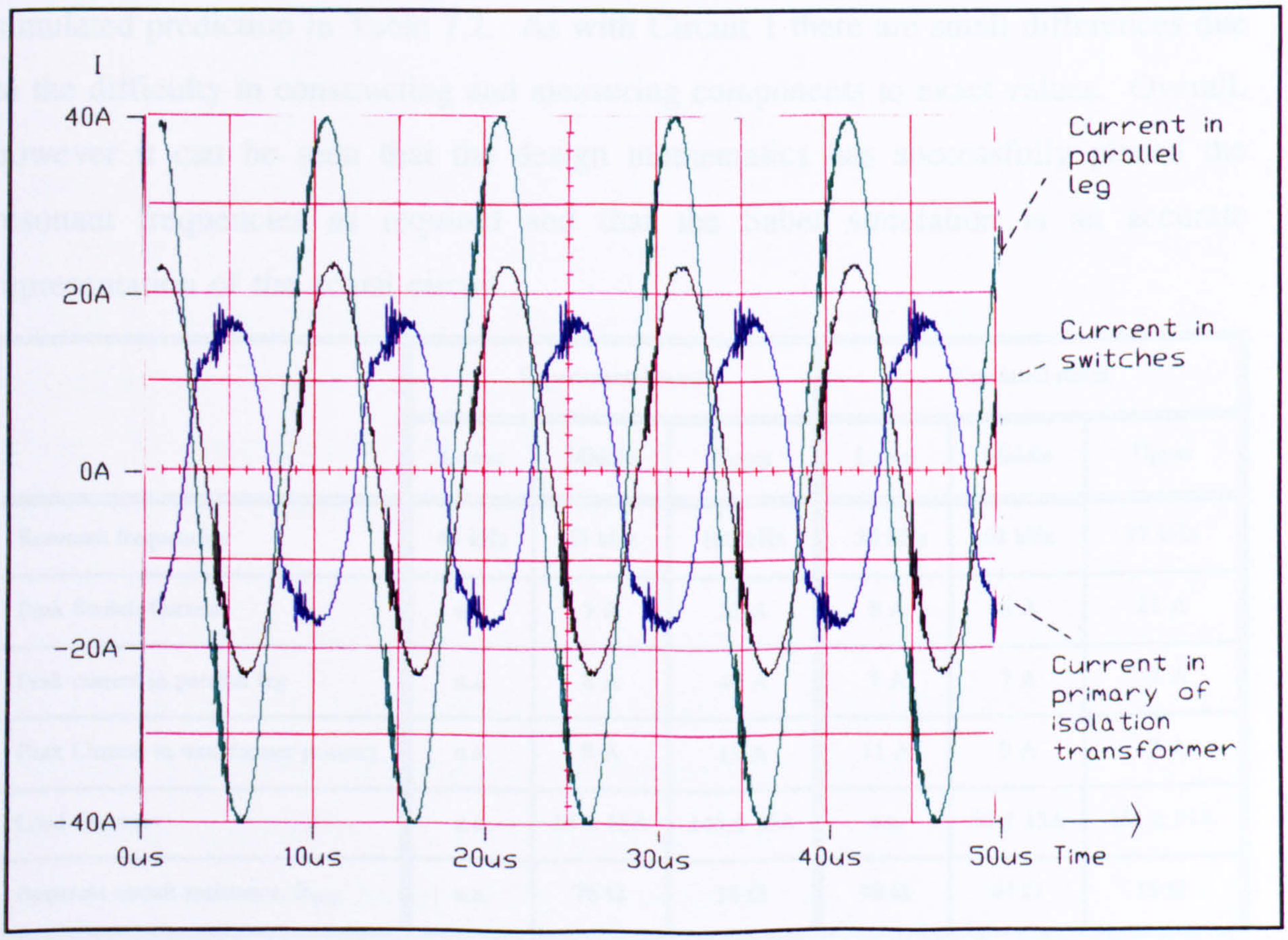


Figure 7.10 (b) Circuit 2, currents in the circuit at highest-resonant frequency

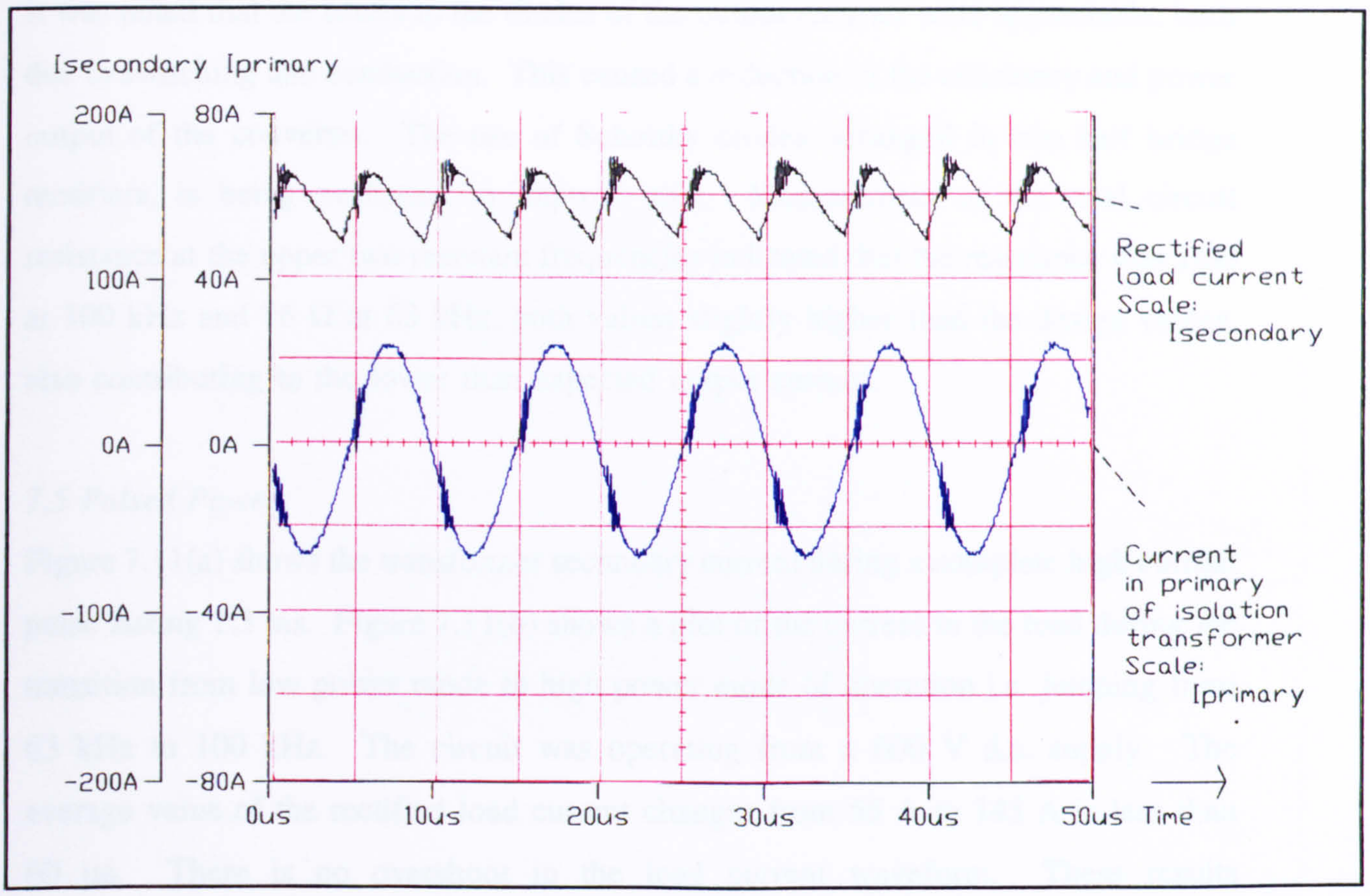


Figure 7.10 (c) Circuit 2, transformer currents & rectified load current at highest-resonant frequency

The results obtained from the experimental testing of Circuit 2 are compared to the simulated prediction in Table 7.2. As with Circuit 1 there are small differences due to the difficulty in constructing and measuring components to exact values. Overall, however it can be seen that the design mathematics has successfully placed the resonant frequencies as required and that the Saber simulation is an accurate representation of the actual circuit.

	Experimental result			Simulated result		
	Lower	Middle	Upper	Lower	Middle	Upper
Resonant frequencies	48 kHz	63 kHz	100 kHz	50 kHz	63 kHz	97 kHz
Peak Switch Current	n.a.	5 A	23 A	8 A	6 A	21 A
Peak current in parallel leg	n.a.	8 A	40 A	7 A	7 A	37 A
Peak Current in transformer primary	n.a.	8 A	17 A	11 A	9 A	17 A
Load Current	n.a.	55 ± 15A	145 ± 25A	n.a.	50 ± 13A	151 ± 31A
Apparent circuit resistance, R_{TOT}	n.a.	76 Ω	16 Ω	48 Ω	64 Ω	18 Ω

Table 7.2 Comparison between experimental and simulated results for Circuit 2

It was noted that the losses in the diodes of the output rectifier were appreciable, both due to switching and conduction. This caused a reduction in the efficiency and power output of the converter. The use of Schottky diodes, arranged in two half bridge rectifiers, is being evaluated to improve this. Measurement of the total circuit resistance at the upper two resonant frequencies indicated that the resistance was 16Ω at 100 kHz and 76 Ω at 63 kHz, both values slightly higher than the design values, also contributing to the lower than expected output current.

7.5 Pulsed Power

Figure 7.11(a) shows the transformer secondary current during a complete high current pulse lasting 1.3 ms. Figure 7.11(b) shows a plot of the current in the load during the transition from low power mode to high power mode of operation i.e. jumping from 63 kHz to 100 kHz. The circuit was operating from a 600 V d.c. supply. The average value of the rectified load current changes from 55 A to 145 A in less than 60 μs. There is no overshoot in the load current waveform. These results demonstrate that this circuit is capable of delivering a 7 : 1 change in output power while maintaining zero current switching. The transient response of the converter is very fast due to the high operating frequency and minimal output filtering.

The results clearly demonstrate the potential for this new technique to obtain substantial changes in average current and hence power while operating from a fixed d.c. supply voltage and maintaining zero current switching throughout. The circuit's IGBT devices had negligible switching losses during these tests and yet were switching at frequencies five times their normal recommended frequency. It is also important to note that the pulsed current waveform has been achieved with no output filtering. This accounts for the high frequency ripple on the rectified current but demonstrates the possibility for a very rapid rise and fall between low power and high power modes of operation. Since the operating frequency of the circuit is very high, the amount of filtering required to achieve a smooth output voltage or current is small. Either capacitive or inductive filters could be added to the rectified output depending on whether a particular application required constant current or constant voltage output. In the application of the welding power supply it was felt that this level of current ripple was acceptable providing the current did not drop to zero. No filtering was therefore added to the circuit other than the inductance of the output leads.

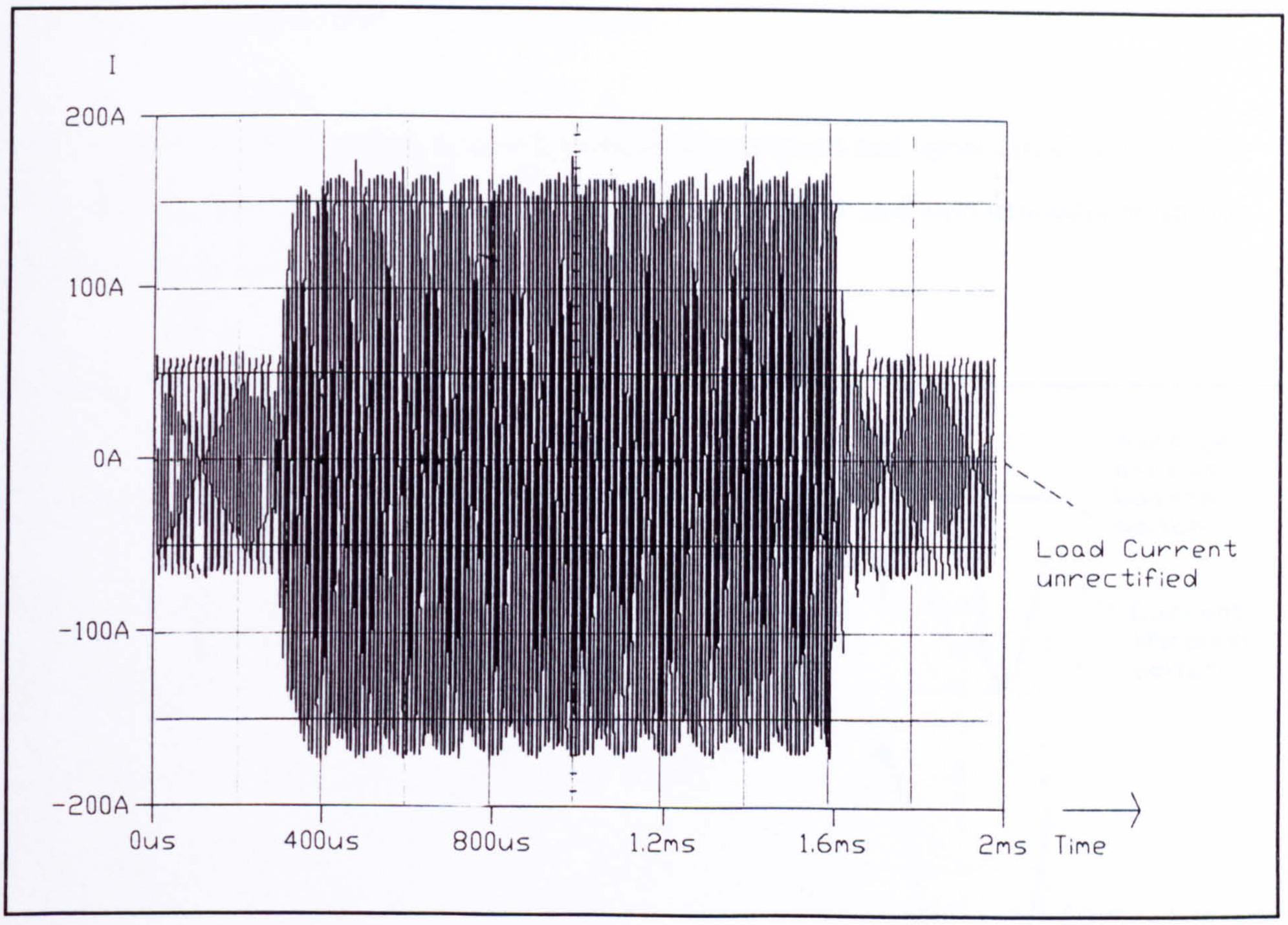


Figure 7.11(a) Transformer secondary current as the circuit changes from low power to high power to low power

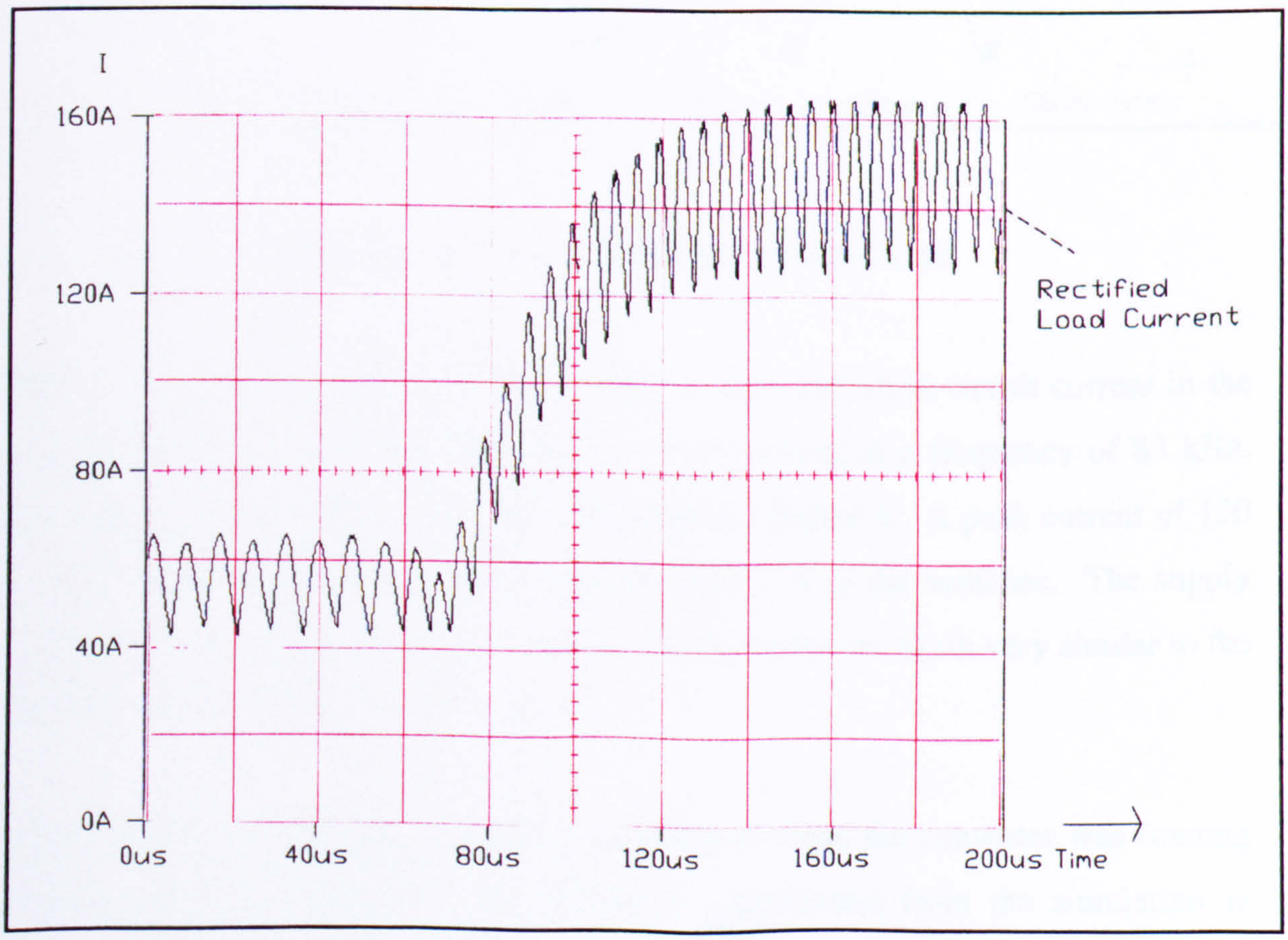


Figure 7.11 (b) Rectified load current during the rising edge of a high current pulse

7.6 Open-circuit and short-circuit operation

Circuit 2 has been tested experimentally under short-circuit and open-circuit conditions to verify the simulated predictions that it could run safely and continuously with the load both open-circuited or short-circuited.

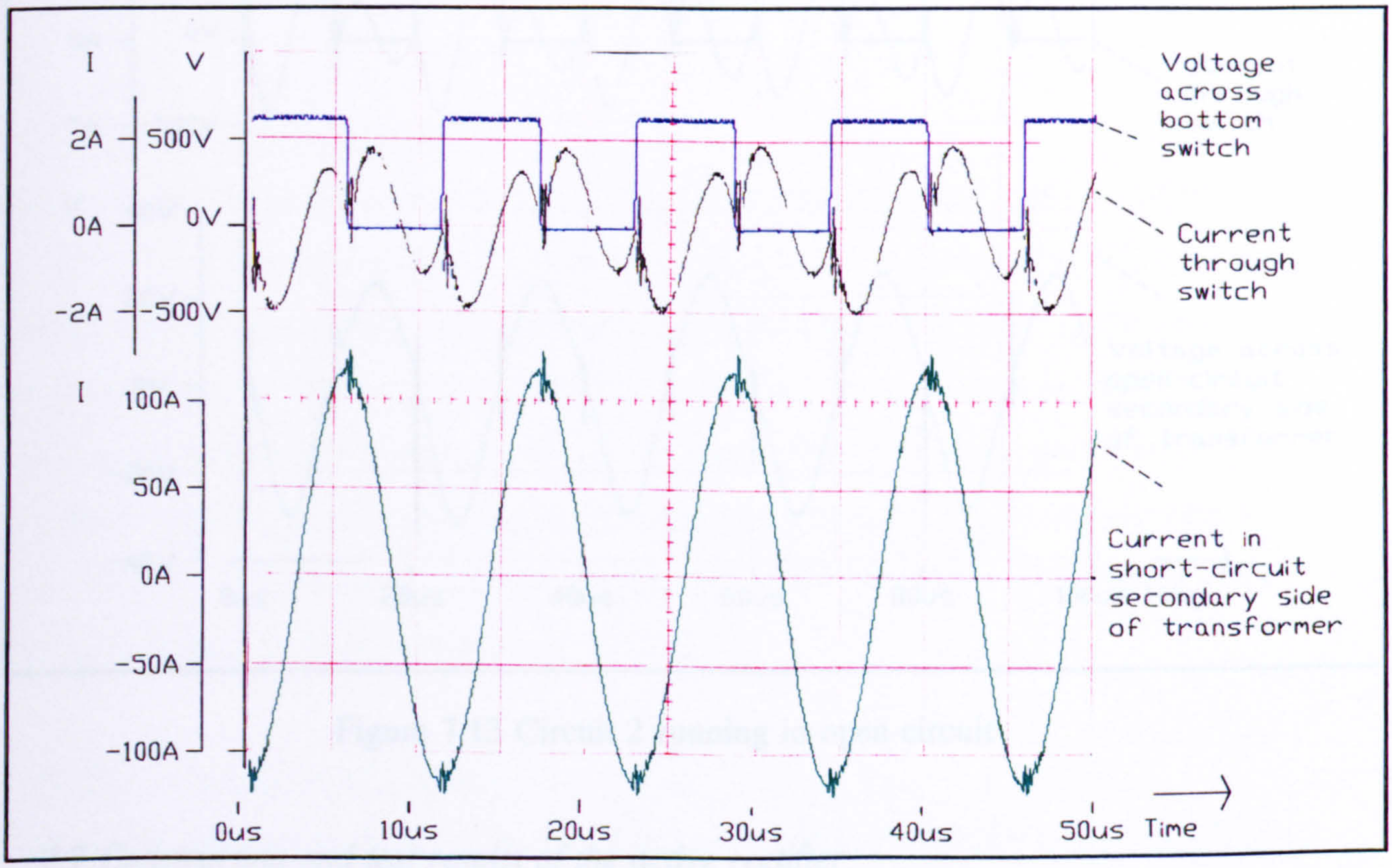


Figure 7.12 Circuit 2 running in short-circuit

Figure 7.12 shows the switch current, switch voltage and short circuit current in the secondary of the transformer when the circuit is running at a frequency of 83 kHz. This frequency was chosen from the simulation in Chapter 6. A peak current of 120 A was recorded in the short circuit with less than 2 A in the switches. The supply voltage was the rated d.c. supply of 600 V. These results are again very similar to the simulation result shown in Figure 6.13.

Figure 7.13 shows the experimental result obtained when the converter was running with the load open-circuited. The frequency was chosen from the simulation in Chapter 6 to be 49 kHz. The peak open circuit voltage recorded on the secondary of

the transformer was about 27 V, and the switch current was a modest 3 A, again comparing very well with the simulation in Figure 6.15.

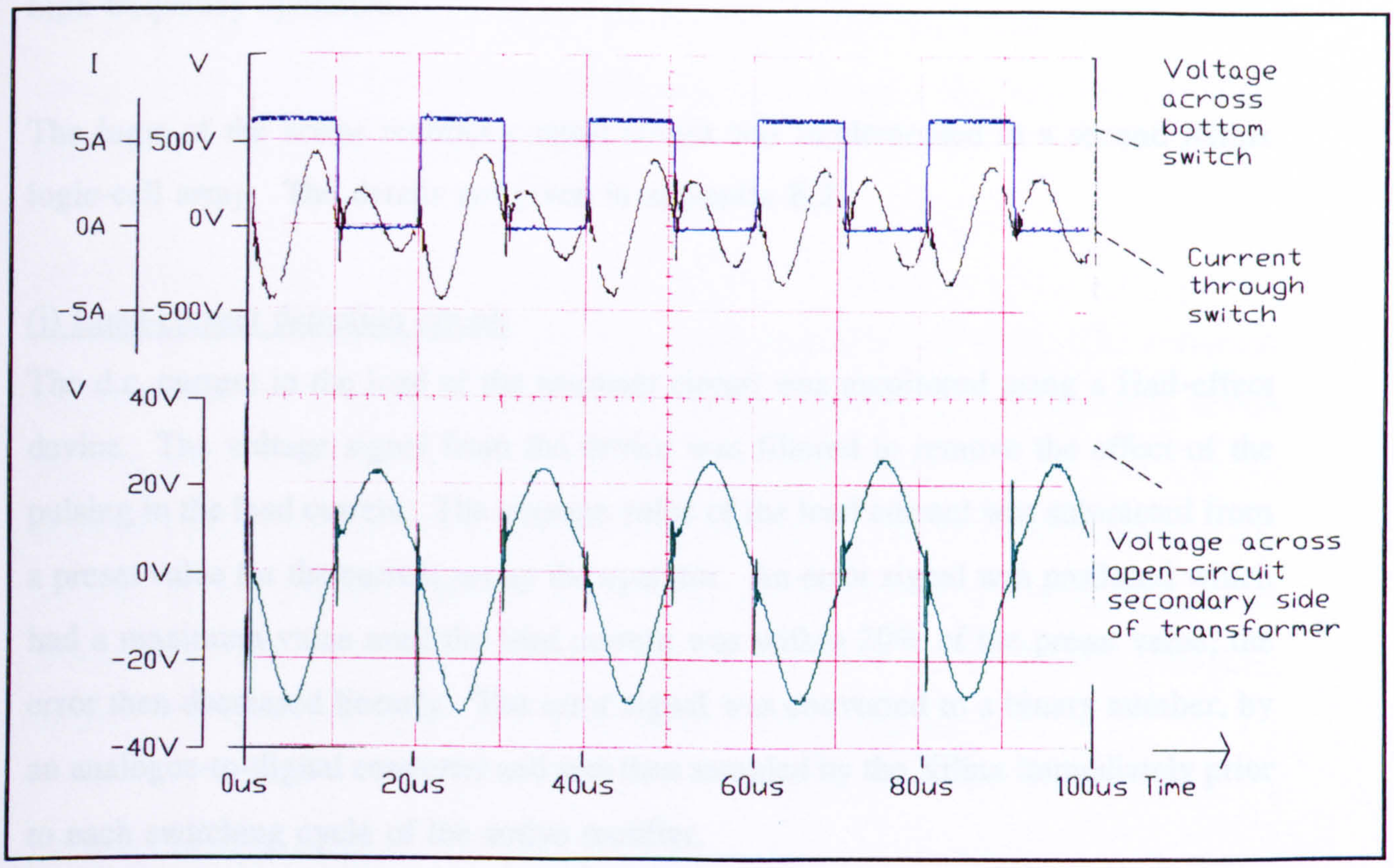


Figure 7.13 Circuit 2 running in open-circuit

7.7 Construction and test results of the active rectifier

7.7.1 Construction of active rectifier

The active rectifier was constructed using a 25 A, 600 V single phase diode rectifier bridge. The buck-boost converter used two SKM181F power MOSFETS and two BYT30PI800 freewheeling diodes. The inductor was constructed from a U and I 3C80 ferrite core combination with an air gap to avoid saturation.

7.7.2 Control of active rectifier

The active rectifier was used to achieve complete power control in the resonant converter. The d.c. current in the load of the resonant circuit was monitored and the duty-cycle of the power switches in the active rectifier adjusted to maintain the load current required. This method of power control varied the d.c supply voltage to the resonant circuit. This avoided the need to move the frequency of operation of the

resonant circuit away from resonance this being the more conventional method of power control in resonant circuits. In this converter the resonant circuit continued to operate at a resonant frequency of the circuit, maintaining zero current switching and high-frequency operation.

The logic of the active rectifier control circuit was implemented in a second Xilinx logic-cell array. The details are given in appendix E.2.

(i) Load-current detection circuit

The d.c. current in the load of the resonant circuit was monitored using a Hall-effect device. The voltage signal from the device was filtered to remove the effect of the pulsing in the load current. The average value of the load current was subtracted from a preset value for the current set by the operator. An error signal was produced which had a maximum value until the load current was within 20% of the preset value, the error then decreased linearly. The error signal was converted to a binary number, by an analogue-to-digital converter and was then sampled by the Xilinx immediately prior to each switching cycle of the active rectifier.

(ii) Steady-state control circuit

The active rectifier in steady-state was designed to run at 20 kHz (Section 6.6). A 20 kHz, 0.5 duty cycle signal was generated in Xilinx. The duty cycle of this signal was modified by preloading a counter with the digital value of the error. The result was a switching signal that had a duty cycle of 0.4, when the value of the load current was less than 80% of the preset-value, a switching signal with decreasing duty-cycle, from 0.4 to 0, as the load current increased to within 20% of the preset value, and a duty-cycle of 0 when the load current was greater than or equal to the preset-value.

The switching signal was converted to the 'on' signals required by the pulse-transformer gate drive. The 'off' signal was generated using the 20 kHz signal. This ensured that regardless of the state of the switching signal the power switches were turned 'off' each cycle.

(iii) Gate-drive circuit

Both power switches were driven by pulse-transformer gate drive circuits. The switching signals generated in the Xilinx logic cell array were converted from a 0-5 V signal to a -15 V to -10 V signal using comparators. This resulted in a ±15 V swing across the gate -drive pulse-transformers and ensured a negative bias across the gate of each device when turned 'off'.

(iv) Start-up circuit

At start-up the output voltage of the buck-boost converter is 0 volts. The converter will therefore run in the continuous current mode, building up large currents, unless the frequency of operation of the circuit is reduced below 20 kHz. A start-up procedure was designed which operated while the output voltage remained below 250 V. During the start-up procedure the switching frequency of the power devices is reduced to 5 kHz by blocking out three out of every four of the 20 kHz switching cycles. The maximum duty cycle achieved at 5 kHz is therefore 0.1.

(v) Over-voltage circuit

If the load current was never to reach the preset-value, because the value of the load resistance in the resonant circuit was too large, the control circuit would continue to drive the switches of the buck-boost converter at maximum duty-cycle building up a larger and larger output voltage. Eventually the circuit would fail. An over-voltage circuit was therefore designed. This used a resistor divider to create a 0-15 V signal in proportion to the output voltage of the converter. A potentiometer is set with a preset-value equivalent to an output voltage of 600 V. A comparator, with feedback, is used to compare the output voltage with the preset voltage. The signal was high while the output voltage was less than the preset voltage. If the signal goes low the on-pulse signal to the pulse-transformer gate drive circuits is overridden and only the turn-off pulses remain so that both power switches are turned 'off' and kept off. The output of the resistor divider is also used to generate the start-up signal and a signal to commence operation of the resonant circuit. This uses two further comparators. The reference potentiometer for each was set to a value equivalent to an output voltage of 300 V.

7.7.3 Test results showing unity power factor

Figure 7.14 shows the results of the full converter in operation when running continuously in low power mode. The active rectifier is driven from a 240 V single phase a.c. supply. An r.m.s. current of 3.5 A is taken from the supply in phase with the voltage. The setting on the current demand pot held the d.c. link voltage at about 530 V d.c..

Table 7.3 shows the current harmonics obtained at this voltage and current compares them with the limits of harmonic currents as specified in the British Standards BS 5406: Part 2 [53]. This British standard is for household appliances and portable tools. The requirement is therefore more stringent than those of an industrial environment. It is predicted at a 3 kW power level the harmonic levels would still be within the standards.

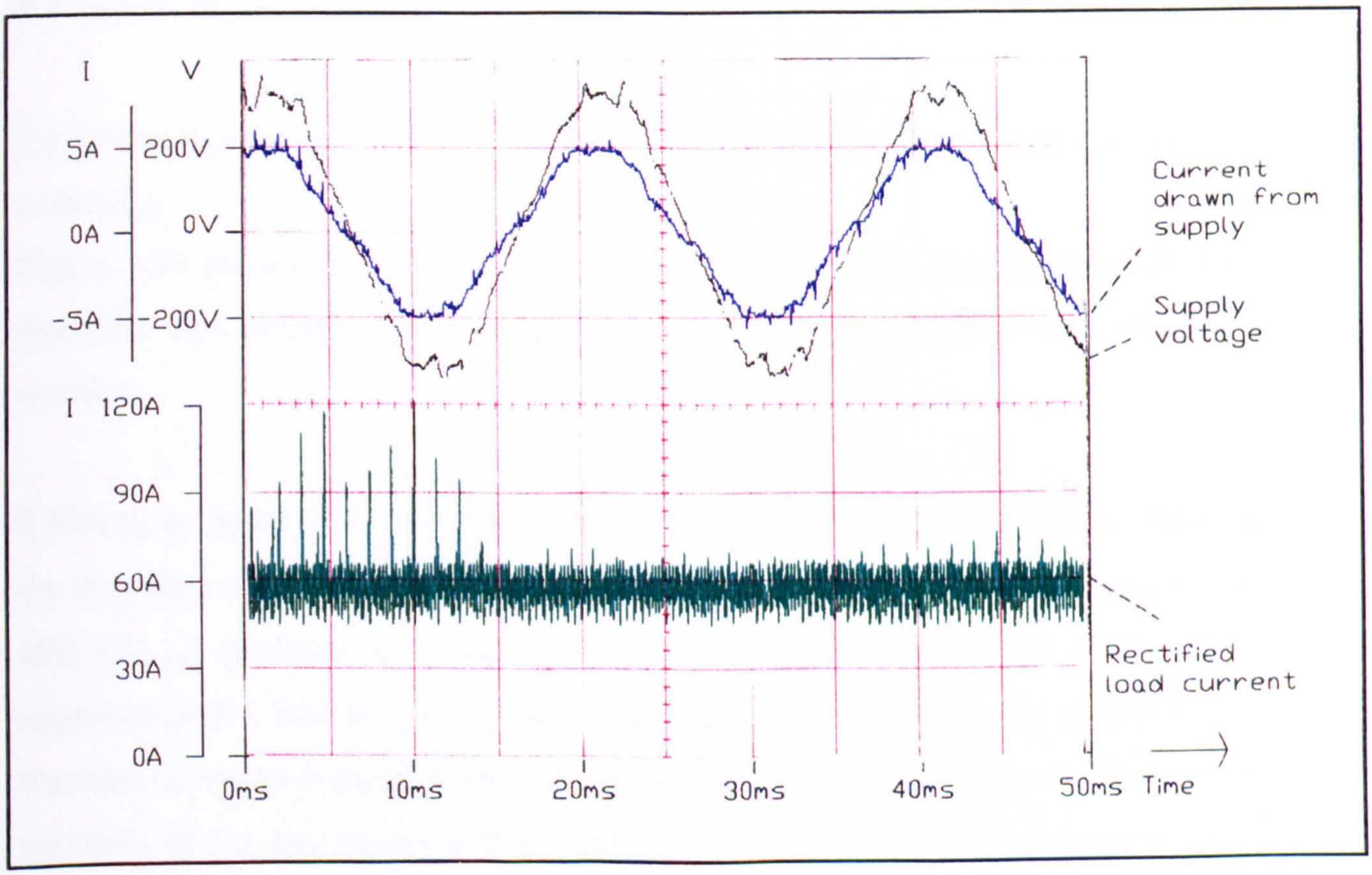


Figure 7.14 Full converter operation with current feedback and unity power factor operation at 240 V r.m.s.

	Supply voltage 240 V r.m.s	British Standard
Fundamental	3.4 A	
2nd harmonic	0.007 A	1.08 A
3rd harmonic	0.072 A	2.30 A
5th harmonic	0.07 A	1.14 A
7th harmonic	0.004 A	0.77 A
9th harmonic	0.024 A	0.40 A
11th harmonic	0.018 A	0.33 A
13th harmonic	0.005 A	0.21 A

Table 7.3. Harmonic content of supply current

7.8 Smooth power control

7.8.1 Smooth power control by varying d.c. link voltage and maintaining zero current switching

Figure 7.15 shows the range of power achievable from the experimental resonant converter with minor variation of the d.c. supply voltage obtained from the active rectifier.

Referring to figure 7.15, at an r.m.s. fundamental supply voltage of 300 V, changing the frequency of operation of the circuit from 63 kHz (R_{TOT3}) to 48 kHz (R_{TOT2}) to 100 kHz (R_{TOT1}) produces a change in the output power level of the circuit from approximately 1 kW to 2.7 kW to 6 kW. However, since the d.c. supply to the resonant converter is controllable, any power between these levels can be obtained by variation of the d.c. supply voltage and hence the r.m.s. value of the fundamental component. A 16:1 variation in power delivered to the circuit can be obtained by appropriate choice of excitation frequency and a modest variation in d.c. supply voltage. Zero current switching is maintained in the load resonant converter over this entire operating range. The ability of the circuit to change its apparent resistance has reduced the range of voltage required to produce a large range of output power.

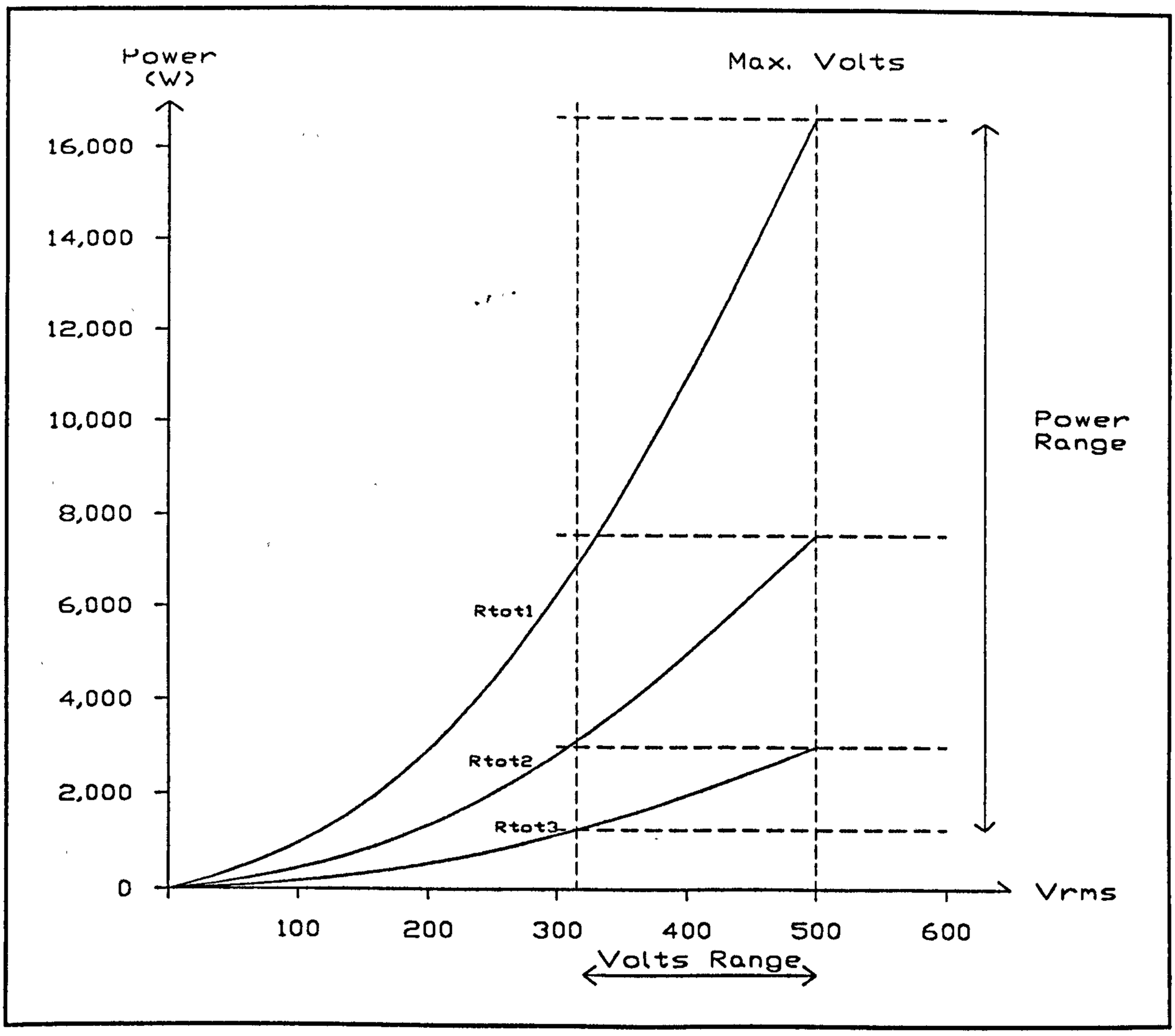


Figure 7.15 Power range of Circuit 2

7.8.2 Smooth power control by running the circuit at intermediate frequencies

Smooth variation of the output power and operation of the circuit at power levels between those obtained at the resonant frequencies can be obtained by fine variation of the operating frequency. In this mode of operation the proposed circuit has a major advantage over prior art resonant converters in that it is not necessary to move far from each resonant frequency before more efficient operation of the converter can be obtained at, or near, one of the circuit's other resonant frequencies. As is usual for load-resonant converters, operation in the inductive region is preferred to reduce diode losses. In the experimental load-resonant converters this can be done by changing the advance on the comparator signal.

CHAPTER 8 CONCLUSIONS

8.1 Conclusions drawn from the work

In recent years there has been a growth in the market for electric arc-welding equipment, much of this growth owing to the advances made in power electronics, thereby allowing automation and improved weld quality. The background to this thesis was the need for a more-compact, more-controllable, arc welding power supply using state of the art power electronics. This can be achieved, in principle, by raising the frequency of operation in the power converter.

The limitations of 'hard-switched' power supplies and many existing 'soft-switched' power supplies restrict the operation of such circuits at higher frequencies and higher output powers, making them expensive. Coupled with the particular requirements of an arc-welding power supply to operate under both short-circuit and open-circuit conditions, it became apparent that the series-parallel load-resonant converter was an attractive circuit for this application. Hence the interest of commercial manufacturers in this work.

Extensive mathematical analysis on the series-parallel resonant network has revealed several features which have not been previously referred to in the literature. The multiplicity of resonant frequencies in such circuits has been ignored by many authors and yet this has been the key to the novel method of power control proposed for series-parallel resonant converters. This new and innovative method of power control provides the possibility of substantial modulation of the power output while maintaining zero current switching. This is achieved by designing a load-resonant converter to have multiple resonant frequencies and different values of apparent circuit resistance at each of these frequencies. Operation of the circuit at each frequency delivers a different amount of power to the load, but zero current switching can be maintained.

A Mathematica Design Programme has been written which takes a specified frequency

characteristic for a series-parallel load-resonant circuit and calculates the set of passive component values required to create that circuit. This breakthrough in mathematical analysis has allowed circuits to be designed with very particular characteristics.

Two series-parallel load-resonant converters were designed and simulated. The simulation in both time domain and frequency domain confirmed the efficacy of the design mathematics and allowed the non-ideal effects particularly in the output rectifier to be modelled in a way which was not possible in the a.c. circuit based design mathematics. These two load-resonant converters were constructed, further verifying the mathematics and simulation. The first converter [55] produced currents of 200 A with the IGBTs in the inverter switching at 82 kHz, nearly four times their normal maximum operating frequency. The losses in the IGBTs due to switching were minimal. The second circuit was designed to demonstrate the new power control technique and pulsed currents of 145 A and a simmer current of 55 A were achieved from a fixed d.c. supply of 600 V [56]. This corresponds to a pulsed power ratio of 7 : 1, with a very rapid transition between the power levels.

The circuit was subsequently supplied by an active rectifier. The converter then drew near unity power factor and gave a full range of power control while maintaining zero current switching.

The circuit has been demonstrated running continuously from its rated d.c. supply voltage with the load both on short-circuit and on open-circuit. Appropriate choice of operating frequency has allowed the switch current, open-circuit voltage and short-circuit current to be limited to safe values. This is a very important feature which is unique to this type of load-resonant converter. Other converters could only survive under such conditions by detecting the fault and shutting down. In welding this is not appropriate as short-circuits and open-circuits occur during normal operation.

The successful demonstration of these circuits at power levels appropriate for welding means that more controllable and lighter welding power supplies will now be possible. The novel control method achieves significant changes in power maintaining zero-

current switching in the power devices at each power level. This has not been considered in previous power control schemes for load-resonant converters and offers significant potential for further development and industrial exploitation.

8.2 Areas for further work

In its present form the analysis of the series-parallel load-resonant converter allows the three resonant frequencies to be placed at particular frequencies and the resistance of the circuit at one of these frequencies to be specified. Following the successful verification of the new pulsed-power technique it would now be more advantageous to specify the resistance of the circuit at the second and third frequencies and thus specify a more accurate profile of the complete frequency response. The system of equations would therefore need to be rewritten to take this requirement into account.

Resonant circuits with more than three resonant frequencies could also be investigated so that the power range over which resonant operation occurred could be increased.

With the assistance of a welding equipment manufacturer the circuits will need to be evaluated for welding different materials to determine the control requirements for the pulsed-current circuit, and the design of the present resonant components adjusted as appropriate. A three-phase welding power supply could be developed. The single phase rectifier could be replaced by a suitable sinusoidal rectifier for three phase systems.

Other applications of this technology could be investigated, for example, laser power supplies where pulsed power is commonly used to reduce the average heat input to a workpiece. For this application, a different transformer design would be necessary as the laser flashlamp is a higher voltage gas discharge. Pulsed sonar applications could be another area of interest.

Other applications requiring benign open-circuit and short-circuit conditions would be attractive areas for future exploitation.

REFERENCES

(1) H. B. Cary, "Modern Welding Technology", Prentice Hall, 1989.

(2) Anon, "Arc welding Power sources", Welding and Metal Fabrication, vol. 58, no. 5, June 1990, pp. 47-52.

(3) J. Norrish, "Arc welding power source designs to meet the needs of industry", Welding and Metal Fabrication, vol. 57, no. 2, March 1989, pp. 61, 63, 65, 67.

(4) S. S. Huang, Y. H. Wu, X. P. Wu, S. B. Li, "Analysis on electrical properties of MOSFET inverter for arc welding", Schweissen and Schneiden vol. 40, no. 2, February 1988, pp. 79-81.

(5) I. K. Pokhodnya, I. Zaruba, V. Ponomarev, V. Latanskii, T. Gvenetadze, N. Troitskaya, "Methods of comparative evaluation of the technological properties of welding equipment and materials", Paton Welding Journal, vol. 2, no. 5, May 1990, pp. 319-322.

(6) H. Yamamoto, S. Harada, T. Yasuda, "The development of welding current control systems for spatter reduction", Welding International, vol. 4, no. 5, 1990, pp. 398-407.

(7) T. Mita, "Reducing spatter in CO_2 gas-shielded welding - waveform control", Welding International, vol. 3, no. 3, 1989, pp. 227-232.

(8) British Standards Institution, "Arc welding power sources, equipment and accessories", British Safety Standard 638: Part 9:1990.

(9) G. E. Cook, G. J. Merrick, "Arc energy relations in Pulse Current Welding Processes", Advanced Welding Techniques, Proceedings 2nd International Symposium of the Japanese Welding Society, 25-27 August 1975, Osaka, Japan, pp. 291-296.

(10) S. Huang, S. Li, Z. Wang, Y. Wu, "An investigation on classification methods of arc welding power sources", Schweissen und Schneiden, vol. 41, no. 3, March 1989, pp. 127-131.

(11) S. Gilbert, "Modern welding power supplies", Pacrim Weldcon '92, Transferring Technology and Knowhow. Proceedings, 40th WTIA National Conference, Darwin, Australia, 29 June-3 July 1992, pp. 23-55.

(12) B. F. Kuvin, "Here's what's new in welding power sources", Welding Design and Fabrication, vol. 65, no. 7, July 1992, pp. 22-29.

(13) Anon., "Developments in power supplies for MIG welding", Welding Review International, vol. 10, no. 4, November 1991, pp. 195-196.

(14) A. Sakabe, T. Kashima, T. Mita, T. Araya, "Inverter-controlled arc welding equipment", Welding International, vol. 1, no. 7, 1987, pp. 629-638.

(15) S. S. Huang, "A study of the MOSFET arc welding inverter (Untersuchung des MOSFET...)", Schweissen und Schneiden, vol. 37, no. 7, July 1985, pp. 302-304.

(16) M. Pixley, "Power sources for pulsed MIG welding", Joining and Materials, vol. 2, no. 6, June 1989, pp. 268-269, 271.

(17) D. Mckeown, "Synergic control-another gimmick or practical solution?", Joining and Materials, vol. 2, no. 6, June 1989, pp. 271.

(18) H. Yamamoto, S. Harada, T. Ueyama, S. Ogawa, "Development of low-frequency pulsed MIG welding for aluminium alloys", Welding International, vol. 6, no. 7, 1992, pp. 580-583.

(19) J. Norrish, J. Nixon, "A history of pulsed MIG welding", Joining and Materials, vol. 2, no. 6, June 1989, pp. 264-266, 268.

(20) C. J. Allum, A. Morris, W. Spray, A. Crane, "Solid state power control for pulsed MIG welding systems", Advanced welding systems, Proceedings 1st International Conference, London 19-21 November 1985, paper no. 38, pp. 397-406.

(21) H. Akagi, T. Sawae, A. Nabae, "130 kHz 7.5 kW current source inverters using static induction transistors for induction heating applications", IEEE Transactions on Power Electronics, vol. 3, July 1988, pp. 303-309.

(22) The SIT Handbook, Tokin Corporation, Japan.

(23) J. Nishizawa and K. Yamamoto, "High-Frequency High Power Static Induction Transistor", IEEE Transactions on Electron Devices, vol. 25, No. 3, March 1978, pp. 314-322.

(24) J. Nishizawa, T. Terasaki and J. Shibata, "Field-Effect Transistor versus Analog Transistor (Static Induction Transistor)", IEEE Transactions on Electron Devices, vol. 22, No. 4, April 1975, pp. 185-197.

(25) SIT data sheet, Tokin Corporation, Japan.

(26) Correspondence with Tokin Corporation, Japan, November 1992.

(27) J. Nishizawa, K. Mitsui, K. Mitamura, S. Maruyama, M. Ikehara and T. Tamamushi, "Low distortion, high efficiency, static induction transistor (SIT) type sinusoidal pwm inverter for uninterruptible power supplies (UPS)", IEEE Industry Applications Society Conference Proceedings 1986, pp. 623-629.

(28) B. K. Bose, "Recent Advances in Power Electronics", IEEE Transactions on Power Electronics, vol. 7, no. 1, January 1992, pp. 2-16.

(29) H. Pollock, J.O Flower, "The Static Induction Transistor", IEE Colloquium Active and Passive components for power electronics, paper no. 3, 25th November 1992, pp. 1-5.

(30) G. Hua, F.C. Lee, "Soft-switching Techniques in PWM Converters", IEEE Transactions on Industrial Electronics, vol. 42, no. 6, December 1995, pp. 595-603.

(31) K. Liu, R. Oruganti, F.C. Lee, "Quasi-resonant Converters - Topologies and Characteristics", IEEE Transactions on Power Electronics, vol. 2, no. 1, 1987, pp. 62-71.

(32) S. J. Finney, T. C. Green, B. W. Williams, "Review of resonant link topologies for inverters", IEE Proceedings Part B, vol. 140, no. 2, March 1993, pp. 103-114.

(33) N. Mohan, T. Underland, W. Robbins, "Power Electronics: Converters, Applications and Design", J.Wiley & Sons, 1989.

(34) R. L. Steigerwald, "A Comparison of Half-Bridge Resonant Converter Topologies", IEEE Transactions on Power Electronics, vol. 3, no. 2, April 1988, pp. 174-182.

(35) A. K. S. Bhat, "Fixed frequency pwm series-parallel resonant converter", IEEE Transactions on Industry Applications, vol. 28, no. 5, September/October 1992, pp. 1002-1009.

(36) P. Jain, "A constant frequency resonant d.c.-d.c. converter with zero switching losses", IEEE Industry Applications Society Conference Proceedings, October 1991, pp. 1067-1073.

(37) D. Czarkowski and M. K. Kamimierczuk, "Phase-controlled Series-Parallel Resonant Converter", IEEE Transactions on Power Electronics, vol. 8, no. 3, July 1993, pp. 309-319.

(38) N. Aouda, Y. Cheron, M. Metz and H. Foch, "Improved operating area of the Zero-voltage-switching Series-resonant converter using Non-linear Capacitive Snubbers", European Power Electronics Conference, Seville, vol. 2, 1995, pp. 658-663.

(39) Y. Murai and S. Mochizuki, "Current pulse control of high frequency series resonant d.c. link power converter", IEEE Industry Applications Society Conference Proceedings, San Diego, 1989 pp. 1023-1030.

(40) R. L. Steigerwald, "High-Frequency Resonant Transistor DC-DC Converters", IEEE Transactions on Power Electronics, vol. 31, no. 2, May 1984, pp. 181-191.

(41) A. K. S. Bhat, "Analysis and design of a High-Frequency Resonant Converter using LCC-Type Commutation", IEEE Transactions on Power Electronics, vol. 2, no. 2, October 1987, pp. 291-300.

(42) A. K. S. Bhat and S. B. Dewan, "A Generalized Approach for the Steady-State Analysis of Resonant Inverters", IEEE Transactions on Industry Applications, vol. 25, no. 2, March/April 1989, pp. 326-338.

(43) I. Batarseh, R. Liu, C. Q. Lee and A. K. Upadhyay, "Theoretical and Experimental Studies of the LCC-Type Parallel Resonant Converter", IEEE Transactions on Power Electronics, vol. 5, no. 2, April 1990, pp. 140-150.

(44) A. K. S. Bhat, V. Agarwal, "Operating modes and simulation of the series-parallel resonant converter", IEEE International Symposium on Circuits and Systems, 1992, pp. 1883-1886.

(45) L. Malesani, P. Mattavelli, L. Rossetto, P. Tenti, W. Marin, A. Pollmann, "Electronic welder with high-frequency resonant inverter", IEEE Industry Applications Society Conference Record, Toronto, Canada, October 1993, pp. 1073-1080.

(46) A. K. S. Bhat, "Analysis and Design of a Series-Parallel Resonant Converter", IEEE Transactions on Power Electronics, vol. 8, no. 1, January 1993, pp. 1-11.

(47) M. K. Kazimierczuk, N. Thirunarayan, S. Wang, "Analysis of Series-Parallel Resonant Converter", IEEE Transactions on Aerospace and Electronic Systems, vol. 29, no. 1, January 1993, pp. 88-98.

(48) A. K. S. Bhat, "Analysis, optimisation and design of a series-parallel resonant converter", IEEE Applied Power Electronics Conference, Los Angeles, CA, March 11-19 1990, pp 155-164.

(49) S. D. Johnson, A. F. Witulski, R. W. Erickson, "Comparison of Resonant Topologies in High voltage DC Applications", IEEE Transactions on Aerospace and Electronic Systems, vol. 24, no. 3, May 1988, pp. 263-273.

(50) S. Shah and A. K. Upadhyay, "Analysis and Design of a Half-Bridge Resonant Converter operating in Discontinuous conduction mode", IEEE Applied Power Electronics Conference, 11-16 March 1990, Los Angeles, California, July 1993, pp. 309-319.

(51) A.K.S. Bhat, A. Biswas and B.S.R. Iyengar, "Analysis and Design of (LC)(LC)-type series-parallel resonant converter", in Conf. Record of IEEE Industry Applications Society Annual Meeting, Toronto, October 1993, pp. 1098-1105.

(52) I. Batarseh, "Resonant converter topologies with three and four energy storage elements", IEEE Trans. on Power Electronics, vol 9, no. 1, January 1994, pp. 64 - 73.

(53) British Standards Institution, "Disturbances in supply systems caused by household appliances and similar electrical equipment", BS 5406:Part 2 :1988 EN 60 555-2:1987.

(54) H. Pollock and J.O. Flower, "Design, simulation and testing of a series resonant converter for pulsed load applications", IEE 5th International Conference on Power Electronics and Variable Speed Drives, London, October 1994, pp. 256-261.

(55) H. Pollock and J.O. Flower, "A series-parallel load-resonant converter for a controlled-current arc welding power supply", accepted for publication in IEE Proceedings Pt.B.

(56) H. Pollock and J.O. Flower, "New method of power control for series parallel load-resonant converters maintaining zero current switching and unity power factor operation", accepted for publication in IEEE Transactions on Power Electronics.

APPENDIX A WELDING PROCESSES

There are approximately 50 distinct welding processes. These are sub-divided into seven groups and summarised in Table A.1.

GROUP	WELDING PROCESS		DEFINITION
Arc-welding Non-consumable Electrode:	Gas tungsten arc	GTAW (TIG)	TIG:uses an arc between a tungsten electrode (non-consumable) and the weld pool.
	Plasma arc	PAW	
Consumable Electrode:	Carbon arc	CAW	MIG: uses an arc between a continuous filler metal electrode and the weld pool.
	Electro gas	EGW	
	Flux cored arc	FCAW	Both techniques use an externally supplied gas without the application of pressure.
	Gas metal arc	GMAW (MIG)	
	Shielded metal arc	SMAW	
	Stud arc	SW	
	Submerged arc	SAW	
Brazing	Diffusion brazing	DFB	Materials are heated to brazing temperature in the presence of a filler metal which melts above 450 C. The filler is bronze or brass and is not distributed by capillary action.
	Dip brazing	DB	
	Furnace brazing	FB	
	Induction brazing	IB	
	Infrared brazing	IRB	
	Resistance brazing	RB	
	Torch brazing	TB	
Oxyfuel gas welding	Oxy-acetalyene	OAW	Materials are heated by an oxyfuel gas flame. Processes occur with or without filler metal or pressure.
	Oxy-hydrogen	OHW	
	Air acetylene		
	Pressure gas	PGW	
Resistance welding	Flash welding	FW	Surfaces are heated by the resistance of the workpieces to the flow of welding current in the circuit of which they are a part and by the application of pressure.
	Projection	RPW	
	Resistance seam	RSEW	
	Resistance spot	RSW	
	Upset welding	UW	
Solid-state welding	Cold welding		The surfaces coalesce by the application of pressure at temperatures below the melting point of the base metal without the addition of brazing or solder filler metal
	Diffusion welding		
	Explosion welding		
	Forge welding		
	Friction welding		
	Hot pressure		
	Roll welding		
	Ultrasonic		
Soldering	Dip		The materials are heated to soldering temperature and a filler metal with a melting point of less than 450 C is used.
	Furnace		
	Induction		
	Infrared		
	Iron		
	Resistance		
	Torch		
	Wave		
Other welding processes	Electron-beam		
	Electoslag		
	Flow		
	Induction		
	Laser-beam		
	Percussion		
	Thermit		

Table A.1 Summary of welding processes

Table A.2 summarises the type of conventional welding power sources used for arc-welding. Both the static and dynamic performance of power supplies have great influence on the properties of the arc-welding process.

ARC-WELDING PROCESSES	DIRECT CURRENT (d.c.)		ALTERNATING CURRENT (a.c.)
NON-CONSUMABLE ELECTRODE	Constant-current drooping	Constant-voltage flat	Constant-current drooping
Gas tungsten arc-welding	✓	×	✓
Plasma arc-welding	✓	×	×
Carbon arc-welding	✓	×	✓
Stud arc-welding	✓	Possible	×
CONSUMABLE ELECTRODE			
Shielded-metal arc-welding	✓	×	✓
Gas metal arc-welding:			
Inert gas-nonferrous MIG,	Possible	✓	×
Spray arc transfer,	Possible	✓	×
Globular transfer,	Possible	✓	×
Short-circuiting transfer,	×	✓	×
Pulsed-arc transfer.	Special	Special	Possible
Flux-cored arc-welding	✓	✓	Experimental
Submerged arc-welding	✓	✓	✓

KEY

✓ used
× not used

Table A.2 Summary of power supply requirements of arc-welding processes

APPENDIX B EQUATIONS FOR DESIGN PROCEDURE

```
-(kc*ll^2*rl^2*w*xp) + 2*kc*ll^4*w^3*xp + kb*ll^4*w^5*xp - ll^4*w^9*xp -
  2*kc*ll^3*w^2*xl*xp - 2*kb*ll^3*w^4*xl*xp + 4*ll^3*w^8*xl*xp +
  kb*ll^2*w^3*xl^2*xp - 6*ll^2*w^7*xl^2*xp + 4*ll*w^6*xl^3*xp - w^5*xl^4*xp -
  kc*ll^3*w^2*xp^2 - kb*ll^3*w^4*xp^2 + ll*rl^2*w^6*xp^2 + 2*ll^3*w^8*xp^2 +
  kb*ll^2*w^3*xl*xp^2 - rl^2*w^5*xl*xp^2 - 7*ll^2*w^7*xl*xp^2 +
  8*ll*w^6*xl^2*xp^2 - 3*w^5*xl^3*xp^2 - rl^2*w^5*xp^3 - ll^2*w^7*xp^3 +
  4*ll*w^6*xl*xp^3 - 3*w^5*xl^2*xp^3 - w^5*xl*xp^4 - kc*ll^2*rl^2*w*xs +
  2*kc*ll^4*w^3*xs + kb*ll^4*w^5*xs - ll^4*w^9*xs - 2*kc*ll^3*w^2*xl*xs -
  2*kb*ll^3*w^4*xl*xs + 4*ll^3*w^8*xl*xs + kb*ll^2*w^3*xl^2*xs -
  6*ll^2*w^7*xl^2*xs + 4*ll*w^6*xl^3*xs - w^5*xl^4*xs - 2*kc*ll^3*w^2*xp*xs -
  2*kb*ll^3*w^4*xp*xs + 4*ll^3*w^8*xp*xs + 2*kb*ll^2*w^3*xl*xp*xs -
  12*ll^2*w^7*xl*xp*xs + 12*ll*w^6*xl^2*xp*xs - 4*w^5*xl^3*xp*xs +
  kb*ll^2*w^3*xp^2*xs - 6*ll^2*w^7*xp^2*xs + 12*ll*w^6*xl*xp^2*xs -
  6*w^5*xl^2*xp^2*xs + 4*ll*w^6*xp^3*xs - 4*w^5*xl*xp^3*xs - w^5*xp^4*xs +
  lp^4*(2*kc*w^3*xl + kb*w^5*xl - w^9*xl + 2*kc*w^3*xs + kb*w^5*xs -
     w^9*xs) + lp^3*(kc*rl^2*w^2 + rl^2*w^8 + 4*kc*ll*w^3*xl +
     2*kb*ll*w^5*xl - 2*ll*w^9*xl - kc*w^2*xl^2 - kb*w^4*xl^2 + 2*w^8*xl^2 -
     2*kc*w^2*xl*xp - 2*kb*w^4*xl*xp + 4*w^8*xl*xp + 8*kc*ll*w^3*xs +
     4*kb*ll*w^5*xs - 4*ll*w^9*xs - 2*kc*w^2*xl*xs - 2*kb*w^4*xl*xs +
     4*w^8*xl*xs - 2*kc*w^2*xp*xs - 2*kb*w^4*xp*xs + 4*w^8*xp*xs) +
  lp^2*(kc*ll*rl^2*w^2 + ll*rl^2*w^8 + 2*kc*ll^2*w^3*xl + kb*ll^2*w^5*xl -
     rl^2*w^7*xl - ll^2*w^9*xl - kc*ll*w^2*xl^2 - kb*ll*w^4*xl^2 +
     2*ll*w^8*xl^2 - w^7*xl^3 - kc*rl^2*w*xp + 2*kc*ll^2*w^3*xp +
     kb*ll^2*w^5*xp - 3*rl^2*w^7*xp - ll^2*w^9*xp - 4*kc*ll*w^2*xl*xp -
     4*kb*ll*w^4*xl*xp + 8*ll*w^8*xl*xp + kb*w^3*xl^2*xp - 7*w^7*xl^2*xp +
     kb*w^3*xl*xp^2 - 6*w^7*xl*xp^2 - kc*rl^2*w*xs + 12*kc*ll^2*w^3*xs +
     6*kb*ll^2*w^5*xs - 6*ll^2*w^9*xs - 6*kc*ll*w^2*xl*xs -
     6*kb*ll*w^4*xl*xs + 12*ll*w^8*xl*xs + kb*w^3*xl^2*xs - 6*w^7*xl^2*xs -
     6*kc*ll*w^2*xp*xs - 6*kb*ll*w^4*xp*xs + 12*ll*w^8*xp*xs +
     2*kb*w^3*xl*xp*xs - 12*w^7*xl*xp*xs + kb*w^3*xp^2*xs - 6*w^7*xp^2*xs) +
  lp*(-2*kc*ll*rl^2*w*xp + 4*kc*ll^3*w^3*xp + 2*kb*ll^3*w^5*xp -
     2*ll*rl^2*w^7*xp - 2*ll^3*w^9*xp - 4*kc*ll^2*w^2*xl*xp -
     4*kb*ll^2*w^4*xl*xp + 2*rl^2*w^6*xl*xp + 8*ll^2*w^8*xl*xp +
     2*kb*ll*w^3*xl^2*xp - 10*ll*w^7*xl^2*xp + 4*w^6*xl^3*xp -
     kc*ll^2*w^2*xp^2 - kb*ll^2*w^4*xp^2 + 3*rl^2*w^6*xp^2 +
     2*ll^2*w^8*xp^2 + 2*kb*ll*w^3*xl*xp^2 - 10*ll*w^7*xl*xp^2 +
     8*w^6*xl^2*xp^2 + 4*w^6*xl*xp^3 - 2*kc*ll*rl^2*w*xs + 8*kc*ll^3*w^3*xs +
     4*kb*ll^3*w^5*xs - 4*ll^3*w^9*xs - 6*kc*ll^2*w^2*xl*xs -
     6*kb*ll^2*w^4*xl*xs + 12*ll^2*w^8*xl*xs + 2*kb*ll*w^3*xl^2*xs -
     12*ll*w^7*xl^2*xs + 4*w^6*xl^3*xs - 6*kc*ll^2*w^2*xp*xs -
     6*kb*ll^2*w^4*xp*xs + 12*ll^2*w^8*xp*xs + 4*kb*ll*w^3*xl*xp*xs -
     24*ll*w^7*xl*xp*xs + 12*w^6*xl^2*xp*xs + 2*kb*ll*w^3*xp^2*xs -
     12*ll*w^7*xp^2*xs + 12*w^6*xl*xp^2*xs + 4*w^6*xp^3*xs)
```

B.1 Three quartics in L_p (a) First quartic in L_p

```
-(kc*ll^4*w*xp) + ka*ll^4*w^5*xp - ll^2*rl^2*w^5*xp + 2*ll^4*w^7*xp -
  2*ka*ll^3*w^4*xl*xp + 2*ll*rl^2*w^4*xl*xp - 6*ll^3*w^6*xl*xp +
  ka*ll^2*w^3*xl^2*xp - rl^2*w^3*xl^2*xp + 6*ll^2*w^5*xl^2*xp -
  2*ll*w^4*xl^3*xp - ka*ll^3*w^4*xp^2 + ll*rl^2*w^4*xp^2 - 3*ll^3*w^6*xp^2 +
  ka*ll^2*w^3*xl*xp^2 - rl^2*w^3*xl*xp^2 + 7*ll^2*w^5*xl*xp^2 -
  4*ll*w^4*xl^2*xp^2 + ll^2*w^5*xp^3 - 2*ll*w^4*xl*xp^3 - kc*ll^4*w*xs +
  ka*ll^4*w^5*xs - ll^2*rl^2*w^5*xs + 2*ll^4*w^7*xs - 2*ka*ll^3*w^4*xl*xs +
  2*ll*rl^2*w^4*xl*xs - 6*ll^3*w^6*xl*xs + ka*ll^2*w^3*xl^2*xs -
  rl^2*w^3*xl^2*xs + 6*ll^2*w^5*xl^2*xs - 2*ll*w^4*xl^3*xs -
  2*ka*ll^3*w^4*xp*xs + 2*ll*rl^2*w^4*xp*xs - 6*ll^3*w^6*xp*xs +
  2*ka*ll^2*w^3*xl*xp*xs - 2*rl^2*w^3*xl*xp*xs + 12*ll^2*w^5*xl*xp*xs -
  6*ll*w^4*xl^2*xp*xs + ka*ll^2*w^3*xp^2*xs - rl^2*w^3*xp^2*xs +
  6*ll^2*w^5*xp^2*xs - 6*ll*w^4*xl*xp^2*xs - 2*ll*w^4*xp^3*xs +
  lp^4*(-(kc*w*xl) + ka*w^5*xl + 2*w^7*xl - kc*w*xs + ka*w^5*xs + 2*w^7*xs) +
  lp^3*(-(kc*rl^2) - rl^2*w^6 - 2*kc*ll*w*xl + 2*ka*ll*w^5*xl + 4*ll*w^7*xl -
     ka*w^4*xl^2 - 3*w^6*xl^2 - 2*ka*w^4*xl*xp - 6*w^6*xl*xp - 4*kc*ll*w*xs +
     4*ka*ll*w^5*xs + 8*ll*w^7*xs - 2*ka*w^4*xl*xs - 6*w^6*xl*xs -
     2*ka*w^4*xp*xs - 6*w^6*xp*xs) +
  lp^2*(-(kc*ll*rl^2) - ll*rl^2*w^6 - kc*ll^2*w*xl + ka*ll^2*w^5*xl +
     rl^2*w^5*xl + 2*ll^2*w^7*xl - ka*ll*w^4*xl^2 - 3*ll*w^6*xl^2 +
     w^5*xl^3 - kc*ll^2*w*xp + ka*ll^2*w^5*xp + 2*rl^2*w^5*xp +
     2*ll^2*w^7*xp - 4*ka*ll*w^4*xl*xp - 12*ll*w^6*xl*xp + ka*w^3*xl^2*xp +
     7*w^5*xl^2*xp + ka*w^3*xl*xp^2 + 6*w^5*xl*xp^2 - 6*kc*ll^2*w*xs +
     6*ka*ll^2*w^5*xs - rl^2*w^5*xs + 12*ll^2*w^7*xs - 6*ka*ll*w^4*xl*xs -
     18*ll*w^6*xl*xs + ka*w^3*xl^2*xs + 6*w^5*xl^2*xs - 6*ka*ll*w^4*xp*xs -
     18*ll*w^6*xp*xs + 2*ka*w^3*xl*xp*xs + 12*w^5*xl*xp*xs + ka*w^3*xp^2*xs +
     6*w^5*xp^2*xs) + lp*(-2*kc*ll^3*w*xp + 2*ka*ll^3*w^5*xp +
     4*ll^3*w^7*xp - 4*ka*ll^2*w^4*xl*xp - 12*ll^2*w^6*xl*xp +
     2*ka*ll*w^3*xl^2*xp + 10*ll*w^5*xl^2*xp - 2*w^4*xl^3*xp -
     ka*ll^2*w^4*xp^2 - rl^2*w^4*xp^2 - 3*ll^2*w^6*xp^2 +
     2*ka*ll*w^3*xl*xp^2 + 10*ll*w^5*xl*xp^2 - 4*w^4*xl^2*xp^2 -
     2*w^4*xl*xp^3 - 4*kc*ll^3*w*xs + 4*ka*ll^3*w^5*xs - 2*ll*rl^2*w^5*xs +
     8*ll^3*w^7*xs - 6*ka*ll^2*w^4*xl*xs + 2*rl^2*w^4*xl*xs -
     18*ll^2*w^6*xl*xs + 2*ka*ll*w^3*xl^2*xs + 12*ll*w^5*xl^2*xs -
     2*w^4*xl^3*xs - 6*ka*ll^2*w^4*xp*xs + 2*rl^2*w^4*xp*xs -
     18*ll^2*w^6*xp*xs + 4*ka*ll*w^3*xl*xp*xs + 24*ll*w^5*xl*xp*xs -
     6*w^4*xl^2*xp*xs + 2*ka*ll*w^3*xp^2*xs + 12*ll*w^5*xp^2*xs -
     6*w^4*xl*xp^2*xs - 2*w^4*xp^3*xs)
```

(b) Second quartic in L_p

```
-(kb*ll^4*w*xp) + ka*ll^2*rl^2*w*xp - rl^4*w*xp - 2*ka*ll^4*w^3*xp +
  4*ll^2*rl^2*w^3*xp - 3*ll^4*w^5*xp + 2*ka*ll^3*w^2*xl*xp -
  4*ll*rl^2*w^2*xl*xp + 6*ll^3*w^4*xl*xp - 3*ll^2*w^3*xl^2*xp +
  ka*ll^3*w^2*xp^2 - 3*ll*rl^2*w^2*xp^2 + 3*ll^3*w^4*xp^2 -
  4*ll^2*w^3*xl*xp^2 - ll^2*w^3*xp^3 - kb*ll^4*w*xs + ka*ll^2*rl^2*w*xs -
  rl^4*w*xs - 2*ka*ll^4*w^3*xs + 4*ll^2*rl^2*w^3*xs - 3*ll^4*w^5*xs +
  2*ka*ll^3*w^2*xl*xs - 4*ll*rl^2*w^2*xl*xs + 6*ll^3*w^4*xl*xs -
  3*ll^2*w^3*xl^2*xs + 2*ka*ll^3*w^2*xp*xs - 4*ll*rl^2*w^2*xp*xs +
  6*ll^3*w^4*xp*xs - 6*ll^2*w^3*xl*xp*xs - 3*ll^2*w^3*xp^2*xs +
  lp^4*(-(kb*w*xl) - 2*ka*w^3*xl - 3*w^5*xl - kb*w*xs - 2*ka*w^3*xs -
     3*w^5*xs) + lp^3*(-(kb*rl^2) - ka*rl^2*w^2 - 2*kb*ll*w*xl -
     4*ka*ll*w^3*xl - 6*ll*w^5*xl + ka*w^2*xl^2 + 3*w^4*xl^2 +
     2*ka*w^2*xl*xp + 6*w^4*xl*xp - 4*kb*ll*w*xs - 8*ka*ll*w^3*xs -
     12*ll*w^5*xs + 2*ka*w^2*xl*xs + 6*w^4*xl*xs + 2*ka*w^2*xp*xs +
     6*w^4*xp*xs) + lp^2*(-(kb*ll*rl^2) - ka*ll*rl^2*w^2 - kb*ll^2*w*xl -
     2*ka*ll^2*w^3*xl - rl^2*w^3*xl - 3*ll^2*w^5*xl + ka*ll*w^2*xl^2 +
     3*ll*w^4*xl^2 - w^3*xl^3 - kb*ll^2*w*xp + ka*rl^2*w*xp -
     2*ka*ll^2*w^3*xp + rl^2*w^3*xp - 3*ll^2*w^5*xp + 4*ka*ll*w^2*xl*xp +
     12*ll*w^4*xl*xp - 4*w^3*xl^2*xp - 3*w^3*xl*xp^2 - 6*kb*ll^2*w*xs +
     ka*rl^2*w*xs - 12*ka*ll^2*w^3*xs + 4*rl^2*w^3*xs - 18*ll^2*w^5*xs +
     6*ka*ll*w^2*xl*xs + 18*ll*w^4*xl*xs - 3*w^3*xl^2*xs +
     6*ka*ll*w^2*xp*xs + 18*ll*w^4*xp*xs - 6*w^3*xl*xp*xs - 3*w^3*xp^2*xs) +
  lp*(-2*kb*ll^3*w*xp + 2*ka*ll*rl^2*w*xp - 4*ka*ll^3*w^3*xp +
     6*ll*rl^2*w^3*xp - 6*ll^3*w^5*xp + 4*ka*ll^2*w^2*xl*xp -
     2*rl^2*w^2*xl*xp + 12*ll^2*w^4*xl*xp - 4*ll*w^3*xl^2*xp +
     ka*ll^2*w^2*xp^2 - rl^2*w^2*xp^2 + 3*ll^2*w^4*xp^2 - 4*ll*w^3*xl*xp^2 -
     4*kb*ll^3*w*xs + 2*ka*ll*rl^2*w*xs - 8*ka*ll^3*w^3*xs +
     8*ll*rl^2*w^3*xs - 12*ll^3*w^5*xs + 6*ka*ll^2*w^2*xl*xs -
     4*rl^2*w^2*xl*xs + 18*ll^2*w^4*xl*xs - 6*ll*w^3*xl^2*xs +
     6*ka*ll^2*w^2*xp*xs - 4*rl^2*w^2*xp*xs + 18*ll^2*w^4*xp*xs -
     12*ll*w^3*xl*xp*xs - 6*ll*w^3*xp^2*xs)
```

(c) Third quartic in L_p

X_s =

```
(ka*kc*l1*rl^2*w + ka*kb*l1*rl^2*w^3 + 3*kc*l1*rl^2*w^3 + ka^2*l1*rl^2*w^5 +
    3*kb*l1*rl^2*w^5 + 4*ka*l1*rl^2*w^7 + 3*l1*rl^2*w^9 - kc*rl^2*w^2*xl -
    kb*rl^2*w^4*xl - ka*rl^2*w^6*xl - rl^2*w^8*xl + ka*kc*l1*w*xl^2 +
    ka*kb*l1*w^3*xl^2 + 3*kc*l1*w^3*xl^2 + ka^2*l1*w^5*xl^2 +
    3*kb*l1*w^5*xl^2 + 4*ka*l1*w^7*xl^2 + 3*l1*w^9*xl^2 - kc*w^2*xl^3 -
    kb*w^4*xl^3 - ka*w^6*xl^3 - w^8*xl^3 + ka*kc*rl^2*xp + kc*rl^2*w^2*xp -
    ka^2*rl^2*w^4*xp - 2*kb*rl^2*w^4*xp - 7*ka*rl^2*w^6*xp - 8*rl^2*w^8*xp +
    4*ka*kc*l1*w*xl*xp + 4*ka*kb*l1*w^3*xl*xp + 12*kc*l1*w^3*xl*xp +
    4*ka^2*l1*w^5*xl*xp + 12*kb*l1*w^5*xl*xp + 16*ka*l1*w^7*xl*xp +
    12*l1*w^9*xl*xp - ka*kb*w^2*xl^2*xp - 4*kc*w^2*xl^2*xp -
    2*ka^2*w^4*xl^2*xp - 7*kb*w^4*xl^2*xp - 13*ka*w^6*xl^2*xp -
    13*w^8*xl^2*xp - ka*kb*w^2*xl*xp^2 - 3*kc*w^2*xl*xp^2 -
    2*ka^2*w^4*xl*xp^2 - 6*kb*w^4*xl*xp^2 - 12*ka*w^6*xl*xp^2 - 12*w^8*xl*xp^2
    )/(-(ka*kc*rl^2) - 4*kc*rl^2*w^2 + ka^2*rl^2*w^4 - kb*rl^2*w^4 +
    4*ka*rl^2*w^6 + 5*rl^2*w^8 - 6*ka*kc*l1*w*xl - 6*ka*kb*l1*w^3*xl -
    18*kc*l1*w^3*xl - 6*ka^2*l1*w^5*xl - 18*kb*l1*w^5*xl - 24*ka*l1*w^7*xl -
    18*l1*w^9*xl + ka*kb*w^2*xl^2 + 3*kc*w^2*xl^2 + 2*ka^2*w^4*xl^2 +
    6*kb*w^4*xl^2 + 12*ka*w^6*xl^2 + 12*w^8*xl^2 - 6*ka*kc*l1*w*xp -
    6*ka*kb*l1*w^3*xp - 18*kc*l1*w^3*xp - 6*ka^2*l1*w^5*xp -
    18*kb*l1*w^5*xp - 24*ka*l1*w^7*xp - 18*l1*w^9*xp + 2*ka*kb*w^2*xl*xp +
    6*kc*w^2*xl*xp + 4*ka^2*w^4*xl*xp + 12*kb*w^4*xl*xp + 24*ka*w^6*xl*xp +
    24*w^8*xl*xp + ka*kb*w^2*xp^2 + 3*kc*w^2*xp^2 + 2*ka^2*w^4*xp^2 +
    6*kb*w^4*xp^2 + 12*ka*w^6*xp^2 + 12*w^8*xp^2)
```

B.2 First solution of quadratic in X_s

```
-(ka*kc*rl^4*rtot) - 4*kc*rl^4*rtot*w^2 + ka^2*rl^4*rtot*w^4 -
  kb*rl^4*rtot*w^4 + 4*ka*rl^4*rtot*w^6 + 5*rl^4*rtot*w^8 -
  6*ka*kc*ll*rl^2*rtot*w*xl - 6*ka*kb*ll*rl^2*rtot*w^3*xl -
  18*kc*ll*rl^2*rtot*w^3*xl - 6*ka^2*ll*rl^2*rtot*w^5*xl -
  18*kb*ll*rl^2*rtot*w^5*xl - 24*ka*ll*rl^2*rtot*w^7*xl -
  18*ll*rl^2*rtot*w^9*xl - ka*kc*rl^2*rtot*xl^2 + ka*kb*rl^2*rtot*w^2*xl^2 -
  kc*rl^2*rtot*w^2*xl^2 + 3*ka^2*rl^2*rtot*w^4*xl^2 +
  5*kb*rl^2*rtot*w^4*xl^2 + 16*ka*rl^2*rtot*w^6*xl^2 +
  17*rl^2*rtot*w^8*xl^2 - 6*ka*kc*ll*rtot*w*xl^3 - 6*ka*kb*ll*rtot*w^3*xl^3 -
  18*kc*ll*rtot*w^3*xl^3 - 6*ka^2*ll*rtot*w^5*xl^3 - 18*kb*ll*rtot*w^5*xl^3 -
  24*ka*ll*rtot*w^7*xl^3 - 18*ll*rtot*w^9*xl^3 + ka*kb*rtot*w^2*xl^4 +
  3*kc*rtot*w^2*xl^4 + 2*ka^2*rtot*w^4*xl^4 + 6*kb*rtot*w^4*xl^4 +
  12*ka*rtot*w^6*xl^4 + 12*rtot*w^8*xl^4 +
  (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w + ka*kb*ll*rl^3*w^3 +
     3*kc*ll*rl^3*w^3 - 6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
     ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 - 6*ka^2*ll*rl^2*rtot*w^5 -
     18*kb*ll*rl^2*rtot*w^5 + 4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
     3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl -
     kc*rl^3*w^2*xl + 2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
     kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl + 11*kb*rl^2*rtot*w^4*xl -
     ka*rl^3*w^6*xl + 28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
     29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 - 12*ka*kc*ll*rtot*w*xl^2 +
     ka*kb*ll*rl*w^3*xl^2 + 3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
     36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 + 3*kb*ll*rl*w^5*xl^2 -
     12*ka^2*ll*rtot*w^5*xl^2 - 36*kb*ll*rtot*w^5*xl^2 +
     4*ka*ll*rl*w^7*xl^2 - 48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
     36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 + 3*ka*kb*rtot*w^2*xl^3 +
     9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
     18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 + 36*ka*rtot*w^6*xl^3 -
     rl*w^8*xl^3 + 36*rtot*w^8*xl^3)*xp +
  (ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 + 3*kc*rl^2*rtot*w^2 -
     ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 + 2*ka^2*rl^2*rtot*w^4 +
     6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
     12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
     4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
     18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
     6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
     24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl -
     ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
     9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
     6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
     36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*xp^2 +
  (-(ka*kb*rl*w^2*xl) - 3*kc*rl*w^2*xl + ka*kb*rtot*w^2*xl +
     3*kc*rtot*w^2*xl - 2*ka^2*rl*w^4*xl - 6*kb*rl*w^4*xl +
     2*ka^2*rtot*w^4*xl + 6*kb*rtot*w^4*xl - 12*ka*rl*w^6*xl +
     12*ka*rtot*w^6*xl - 12*rl*w^8*xl + 12*rtot*w^8*xl)*xp^3
```

B.3 Polynomial in X_p

```
(* MATHEMATICA PROGRAM TO CALCULATE THE COMPONENT VALUES FOR*)
(*A SERIES-PARALLEL RESONANT CONVERTER*)
(*Section 1 Input*)
(* xl the load impedance*)
(* Minimum value of xl, Maximum value of xl, no. of increments*)
minxl = 59.690; (* True value -0.5*)
maxxl = 59.690;  (* True value -2*)
nnxl = 1;
nxl = 1;
(* ll the load inducatnce*)
(* Minimum value of ll, Maximum value of ll, no. of increments*)
minll = 100*10^-6;
maxll = 100*10^-6;
nnll = 1;
nll = 1;
(* rl the load resistance*)
(* Minimum value of rl, Maximum value of rl, no. of increments *)
minrl = 15.125;
maxrl = 15.125;
nnrl = 1;
nrl = 1;
(* rtot the resistance value of the circuit at resonance *)
(* Minimum value of rtot, Maximum value of rtot, no. of increments*)
minrtot = 14.59;
maxrtot = 14.59;
nnrtot = 1;
nrtot = 1;
(* f the resonant frequency range 20000 to 100000*)
(* Minimum value of f, Maximum value of f, no. of increments*)
minf = 95000;
maxf = 95000;
nnf =  1;
nf = 1;
w = N[2*Pi*f];
(* f1 the lower resonant frequency 10000 to 500000*)
(* Minimum value of f1, Maximum value of f1, no. of increments*)
minf1 = 60000;
maxf1 = 60000;
nnf1=  1 ;
nf1 = 1;
w1 = N[ 2*Pi*f1];
(* f2 the upper resonant frequency 10000 to 500000*)
(* Minimum value of f2, Maximum value of f2, no. of increments*)

minf2 = 85000;
maxf2 = 85000;
nnf2 = 1;
nf2 = 1;
w2 = N [2*Pi*f2];
(* set a value j to start incrementing to control opening and appending of
results file *)
j = 1;

scale:=
Module[{},
xl = N[(((maxxl-minxl)/nnxl)*(nxl))+ minxl];
ll = N[(((maxll-minll)/nnll)*(nll))+ minll];
rl = N[(((maxrl-minrl)/nnrl)*(nrl))+ minrl];
rtot = N[(((maxrtot-minrtot)/nnrtot)*(nrtot))+ minrtot];
f = N[(((maxf-minf)/nnf)*(nf))+ minf];
w = N[2*Pi*f];
f1 = N[(((maxf1-minf1)/nnf1)*(nf1))+ minf1];
w1 = N[ 2*Pi*f1];
f2 = N[(((maxf2-minf2)/nnf2)*(nf2))+ minf2];
w2 = N [2*Pi*f2];];
```

B.4. Mathematica program. A System specified in terms of X_L, L_L, R_L, R_{TOT}, ω_0, ω_1, ω_2 may be solved to find component values L_S, C_S, L_P and C_P.

```
(* Section 2 Calculation of values of k*)
(* Calculate value of ka *)
kcalc:=
Module[{},
ka =N[ -(w*w)-(w1*w1)-(w2*w2)];
(* Calculate value of kb *)
kb =N[(w*w*w1*w1)+(w1*w1*w2*w2)+(w2*w2*w*w)];
(* Calculate value of kc *)
kc = N[-(w*w*w1*w1*w2*w2)];];

(*Section 3 Calculation of impedances and values of inductances*)
(*Calculate possible values of xp*)
xpcalc:=
Module[{xppa,xppb,xppc},
xppa = N[(-(ka*kc*r1^3 + kc*r1^3*w^2 + ka*kb*r1^2*rtot*w^2 + 3*kc*r1^2*rtot*w^2
        ka^2*r1^3*w^4 - 2*kb*r1^3*w^4 + 2*ka^2*r1^2*rtot*w^4 +
     6*kb*r1^2*rtot*w^4 - 7*ka*r1^3*w^6 + 12*ka*r1^2*rtot*w^6 -
     8*r1^3*w^8 + 12*r1^2*rtot*w^8 + 4*ka*kc*l1*r1*w*xl -
     6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*r1*w^3*xl + 12*kc*l1*r1*w^3*xl -
     6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*r1*w^5*xl +
     12*kb*l1*r1*w^5*xl - 6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
     16*ka*l1*r1*w^7*xl - 24*ka*l1*rtot*w^7*xl + 12*l1*r1*w^9*xl -
     18*l1*rtot*w^9*xl - ka*kb*r1*w^2*xl^2 - 4*kc*r1*w^2*xl^2 +
     3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 - 2*ka^2*r1*w^4*xl^2 -
     7*kb*r1*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
     13*ka*r1*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*r1*w^8*xl^2 +
     36*rtot*w^8*xl^2)/
   (3*w^2*(-(ka*kb*r1) - 3*kc*r1 + ka*kb*rtot + 3*kc*rtot - 2*ka^2*r1*w^2 -
       6*kb*r1*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 - 12*ka*r1*w^4 +
       12*ka*rtot*w^4 - 12*r1*w^6 + 12*rtot*w^6)*xl) -
  (-(ka*kc*r1^3 + kc*r1^3*w^2 + ka*kb*r1^2*rtot*w^2 + 3*kc*r1^2*rtot*w^2 -
        ka^2*r1^3*w^4 - 2*kb*r1^3*w^4 + 2*ka^2*r1^2*rtot*w^4 +
        6*kb*r1^2*rtot*w^4 - 7*ka*r1^3*w^6 + 12*ka*r1^2*rtot*w^6 -
        8*r1^3*w^8 + 12*r1^2*rtot*w^8 + 4*ka*kc*l1*r1*w*xl -
        6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*r1*w^3*xl +
        12*kc*l1*r1*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
        18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*r1*w^5*xl + 12*kb*l1*r1*w^5*xl -
        6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
        16*ka*l1*r1*w^7*xl - 24*ka*l1*rtot*w^7*xl + 12*l1*r1*w^9*xl -
        18*l1*rtot*w^9*xl - ka*kb*r1*w^2*xl^2 - 4*kc*r1*w^2*xl^2 +
        3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 - 2*ka^2*r1*w^4*xl^2 -
        7*kb*r1*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
        13*ka*r1*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*r1*w^8*xl^2 +
        36*rtot*w^8*xl^2)^2/
  (9*w^4*(-(ka*kb*r1) - 3*kc*r1 + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*r1*w^2 - 6*kb*r1*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 -
       12*ka*r1*w^4 + 12*ka*rtot*w^4 - 12*r1*w^6 + 12*rtot*w^6)^2*xl^2)\
 + (ka*kc*l1*r1^3*w - 6*ka*kc*l1*r1^2*rtot*w + ka*kb*l1*r1^3*w^3 +
   3*kc*l1*r1^3*w^3 - 6*ka*kb*l1*r1^2*rtot*w^3 -
   18*kc*l1*r1^2*rtot*w^3 + ka^2*l1*r1^3*w^5 + 3*kb*l1*r1^3*w^5 -
   6*ka^2*l1*r1^2*rtot*w^5 - 18*kb*l1*r1^2*rtot*w^5 +
   4*ka*l1*r1^3*w^7 - 24*ka*l1*r1^2*rtot*w^7 + 3*l1*r1^3*w^9 -
   18*l1*r1^2*rtot*w^9 - ka*kc*r1^2*rtot*xl - kc*r1^3*w^2*xl +
   2*ka*kb*r1^2*rtot*w^2*xl + 2*kc*r1^2*rtot*w^2*xl - kb*r1^3*w^4*xl +
   5*ka^2*r1^2*rtot*w^4*xl + 11*kb*r1^2*rtot*w^4*xl - ka*r1^3*w^6*xl +
   28*ka*r1^2*rtot*w^6*xl - r1^3*w^8*xl + 29*r1^2*rtot*w^8*xl +
   ka*kc*l1*r1*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
   ka*kb*l1*r1*w^3*xl^2 + 3*kc*l1*r1*w^3*xl^2 -
   12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
   ka^2*l1*r1*w^5*xl^2 + 3*kb*l1*r1*w^5*xl^2 -
   12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
   4*ka*l1*r1*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 + 3*l1*r1*w^9*xl^2 -
   36*l1*rtot*w^9*xl^2 - kc*r1*w^2*xl^3 + 3*ka*kb*rtot*w^2*xl^3 +
   9*kc*rtot*w^2*xl^3 - kb*r1*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
   18*kb*rtot*w^4*xl^3 - ka*r1*w^6*xl^3 + 36*ka*rtot*w^6*xl^3 -
   r1*w^8*xl^3 + 36*rtot*w^8*xl^3)/
```

```
   (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 -
       12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)*xl))/
(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 + 3*kc*rl^2*rtot*w^2 -
      ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 + 2*ka^2*rl^2*rtot*w^4 +
      6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 -
        8*rl^3*w^8 + 12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
      6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
      12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
      18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
      12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
      18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
      24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl -
      ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
      6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^3/
   (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 -
       12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)^3*xl^3)
    + ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
      12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
      4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
      4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
      6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
      4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
      6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
      16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
      18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
      3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
      2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 -
      13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
     (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w + ka*kb*l1*rl^3*w^3 +
      3*kc*l1*rl^3*w^3 - 6*ka*kb*l1*rl^2*rtot*w^3 -
      18*kc*l1*rl^2*rtot*w^3 + ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
      6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
      4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
      18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
      2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
      kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
      11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
      28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
      ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
      ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
      12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
      ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
      12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
      4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
      3*l1*rl*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 +
      6*ka^2*rtot*w^4*xl^3 + 18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
  (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 -
       12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)^2*xl^2)
   - (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
      kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
      6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
      18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
      18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
      18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 + ka*kb*rl^2*w^2*xl^2 -
      kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
      16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 - 6*ka*kc*l1*w*xl^3 -
```

```
      6*ka*kb*l1*w^3*xl^3 - 18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 -
      18*kb*l1*w^5*xl^3 - 24*ka*l1*w^7*xl^3 - 18*l1*w^9*xl^3 +
      ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 +
      6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 + 12*w^8*xl^4))/
   (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 -
      12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)*xl) +
  ((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
            3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
            2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
            7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
            12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
            6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
            12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
            18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
            12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
            18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
            24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
            18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
            3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
            2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
            6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
            13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
            36*rtot*w^8*xl^2)^2/
   (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
  (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w + ka*kb*l1*rl^3*w^3 +
     3*kc*l1*rl^3*w^3 - 6*ka*kb*l1*rl^2*rtot*w^3 -
     18*kc*l1*rl^2*rtot*w^3 + ka^2*l1*rl^3*w^5 +
     3*kb*l1*rl^3*w^5 - 6*ka^2*l1*rl^2*rtot*w^5 -
     18*kb*l1*rl^2*rtot*w^5 + 4*ka*l1*rl^3*w^7 -
     24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
     18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
     2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
     kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
     11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
     28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
     ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
     ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
     12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
     ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
     12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
     4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
     3*l1*rl*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
     3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
     kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
     18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 + 36*ka*rtot*w^6*xl^3 -
     rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
   (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
  (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
        3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
        2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
        7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
        12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
        6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
        12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
        18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
        12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
        18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
        24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
        18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
```

```
      3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
      2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
      6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
      13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)^3/
   (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
       12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
  ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
      7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
      12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
      6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
      12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
      18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
      12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
      18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
      24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
      18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
      3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
      2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
      6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
      13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)*
    (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
      ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
      6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
      ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
      6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
      4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
      3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl -
      kc*rl^3*w^2*xl + 2*ka*kb*rl^2*rtot*w^2*xl +
      2*kc*rl^2*rtot*w^2*xl - kb*rl^3*w^4*xl +
      5*ka^2*rl^2*rtot*w^4*xl + 11*kb*rl^2*rtot*w^4*xl -
      ka*rl^3*w^6*xl + 28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
      29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
      12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
      3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
      36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
      3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
      36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
      48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
      36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
      kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
      18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
   (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
       12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
  (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
      kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
      6*ka*kc*ll*rl^2*w*xl - 6*ka*kb*ll*rl^2*w^3*xl -
      18*kc*ll*rl^2*w^3*xl - 6*ka^2*ll*rl^2*w^5*xl -
      18*kb*ll*rl^2*w^5*xl - 24*ka*ll*rl^2*w^7*xl -
      18*ll*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
      ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
      3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
      16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
      6*ka*kc*ll*w*xl^3 - 6*ka*kb*ll*w^3*xl^3 -
      18*kc*ll*w^3*xl^3 - 6*ka^2*ll*w^5*xl^3 -
      18*kb*ll*w^5*xl^3 - 24*ka*ll*w^7*xl^3 - 18*ll*w^9*xl^3 +
      ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 +
```

```
                6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 + 12*w^8*xl^4))/
           (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
                2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
                6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
                12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3) +
      (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 + 3*kc*rl^2*rtot*w^2 -
           ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 + 2*ka^2*rl^2*rtot*w^4 +
           6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 -
           8*rl^3*w^8 + 12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
           6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
           12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
           18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
           6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
           16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
           18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
           3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
           7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
           13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
           36*rtot*w^8*xl^2)^3/
         (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
            2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 -
            12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)^3*xl^3)\
       + ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
           3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
           2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
           12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
           4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
           4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
           6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
           4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
           6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
           16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
           18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
           3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
           7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
           13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
           36*rtot*w^8*xl^2)*(ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
           ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 - 6*ka*kb*ll*rl^2*rtot*w^3 -
           18*kc*ll*rl^2*rtot*w^3 + ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
           6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
           4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 + 3*ll*rl^3*w^9 -
           18*ll*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
           2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
           kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
           11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
           28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
           ka*kc*ll*rl*w*xl^2 - 12*ka*kc*ll*rtot*w*xl^2 +
           ka*kb*ll*rl*w^3*xl^2 + 3*kc*ll*rl*w^3*xl^2 -
           12*ka*kb*ll*rtot*w^3*xl^2 - 36*kc*ll*rtot*w^3*xl^2 +
           ka^2*ll*rl*w^5*xl^2 + 3*kb*ll*rl*w^5*xl^2 -
           12*ka^2*ll*rtot*w^5*xl^2 - 36*kb*ll*rtot*w^5*xl^2 +
           4*ka*ll*rl*w^7*xl^2 - 48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
           36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 + 3*ka*kb*rtot*w^2*xl^3 +
           9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
           18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 + 36*ka*rtot*w^6*xl^3 -
           rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
         (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
            2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 -
            12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)^2*xl^2)\
       - (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
           kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 - 6*ka*kc*ll*rl^2*w*xl -
           6*ka*kb*ll*rl^2*w^3*xl - 18*kc*ll*rl^2*w^3*xl -
           6*ka^2*ll*rl^2*w^5*xl - 18*kb*ll*rl^2*w^5*xl -
           24*ka*ll*rl^2*w^7*xl - 18*ll*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
           ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 +
           5*kb*rl^2*w^4*xl^2 + 16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
```

```
     6*ka*kc*l1*w*xl^3 - 6*ka*kb*l1*w^3*xl^3 - 18*kc*l1*w^3*xl^3 -
     6*ka^2*l1*w^5*xl^3 - 18*kb*l1*w^5*xl^3 - 24*ka*l1*w^7*xl^3 -
     18*l1*w^9*xl^3 + ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 +
     2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 + 12*w^8*xl^4))/
 (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 -
     12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)*xl) +
((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
           3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
           2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
           12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
           4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
           4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
           6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
           4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
           6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
           16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
           12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
           4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
           9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
           7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
           18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
           36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^2/
 (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
(ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w + ka*kb*l1*rl^3*w^3 +
   3*kc*l1*rl^3*w^3 - 6*ka*kb*l1*rl^2*rtot*w^3 -
   18*kc*l1*rl^2*rtot*w^3 + ka^2*l1*rl^3*w^5 +
   3*kb*l1*rl^3*w^5 - 6*ka^2*l1*rl^2*rtot*w^5 -
   18*kb*l1*rl^2*rtot*w^5 + 4*ka*l1*rl^3*w^7 -
   24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
   18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
   2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
   kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
   11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
   28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
   ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
   ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
   12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
   ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
   12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
   4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
   3*l1*rl*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
   3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 +
   6*ka^2*rtot*w^4*xl^3 + 18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
   36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
 (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
     12*rtot*w^6)*xl))^3 +
(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
     3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
     2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
     12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
     4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
     4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
     6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
     4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
     6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
     16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
     12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
     4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
     9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
     7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
```

```
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^3/
   (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
  ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
      12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
      4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
      4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
      6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
      4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
      6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
      16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
      12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
      7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
    (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w +
      ka*kb*l1*rl^3*w^3 + 3*kc*l1*rl^3*w^3 -
      6*ka*kb*l1*rl^2*rtot*w^3 - 18*kc*l1*rl^2*rtot*w^3 +
      ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
      6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
      4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
      18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
      2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
      kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
      11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
      28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
      29*rl^2*rtot*w^8*xl + ka*kc*l1*rl*w*xl^2 -
      12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*rl*w^3*xl^2 +
      3*kc*l1*rl*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
      36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*rl*w^5*xl^2 +
      3*kb*l1*rl*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
      36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*rl*w^7*xl^2 -
      48*ka*l1*rtot*w^7*xl^2 + 3*l1*rl*w^9*xl^2 -
      36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
      kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
      18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
   (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
  (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
      kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
      6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
      18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
      18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
      18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 + ka*kb*rl^2*w^2*xl^2 -
      kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 +
      5*kb*rl^2*w^4*xl^2 + 16*ka*rl^2*w^6*xl^2 +
      17*rl^2*w^8*xl^2 - 6*ka*kc*l1*w*xl^3 -
      6*ka*kb*l1*w^3*xl^3 - 18*kc*l1*w^3*xl^3 -
      6*ka^2*l1*w^5*xl^3 - 18*kb*l1*w^5*xl^3 -
      24*ka*l1*w^7*xl^3 - 18*l1*w^9*xl^3 + ka*kb*w^2*xl^4 +
      3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 +
      12*ka*w^6*xl^4 + 12*w^8*xl^4))/
  (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
```

```
                    12*rtot*w^6)*xl))^2)^(1/2))^(1/3))];

xppb = N[(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
      12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
      4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
      12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
      4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
      18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
      12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
      2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 -
      13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)/
    (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot - 2*ka^2*rl*w^2 -
       6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 - 12*ka*rl*w^4 +
       12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)*xl) -
    (-((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
            3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
            2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
            12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
            4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
            4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
            6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
            4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
            6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
            16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
            12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
            4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
            9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
            6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
            13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
            36*rtot*w^8*xl^2)^2/
          (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
             2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
             6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
             12*rtot*w^6)^2*xl^2) +
          (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w + ka*kb*l1*rl^3*w^3 +
             3*kc*l1*rl^3*w^3 - 6*ka*kb*l1*rl^2*rtot*w^3 -
             18*kc*l1*rl^2*rtot*w^3 + ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
             6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
             4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
             18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
             2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
             kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
             11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
             28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
             ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
             ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
             12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
             ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
             12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
             4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
             3*l1*rl*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
             3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 +
             6*ka^2*rtot*w^4*xl^3 + 18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
             36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
          (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
             2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
             6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
             12*rtot*w^6)*xl))/
       (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
            3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
            2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
            12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
```

```
      4*ka*kc*l1*r1*w*xl - 6*ka*kc*l1*rtot*w*xl +
      4*ka*kb*l1*r1*w^3*xl + 12*kc*l1*r1*w^3*xl -
      6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
      4*ka^2*l1*r1*w^5*xl + 12*kb*l1*r1*w^5*xl -
      6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
      16*ka*l1*r1*w^7*xl - 24*ka*l1*rtot*w^7*xl +
      12*l1*r1*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*r1*w^2*xl^2 -
      4*kc*r1*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*r1*w^4*xl^2 -
      7*kb*r1*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*r1*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*r1*w^8*xl^2 + 36*rtot*w^8*xl^2)^3/
   (27*w^6*(-(ka*kb*r1) - 3*kc*r1 + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*r1*w^2 - 6*kb*r1*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*r1*w^4 + 12*ka*rtot*w^4 -
        12*r1*w^6 + 12*rtot*w^6)^3*xl^3) +
  ((ka*kc*r1^3 + kc*r1^3*w^2 + ka*kb*r1^2*rtot*w^2 +
    3*kc*r1^2*rtot*w^2 - ka^2*r1^3*w^4 - 2*kb*r1^3*w^4 +
    2*ka^2*r1^2*rtot*w^4 + 6*kb*r1^2*rtot*w^4 - 7*ka*r1^3*w^6 +
    12*ka*r1^2*rtot*w^6 - 8*r1^3*w^8 + 12*r1^2*rtot*w^8 +
    4*ka*kc*l1*r1*w*xl - 6*ka*kc*l1*rtot*w*xl +
    4*ka*kb*l1*r1*w^3*xl + 12*kc*l1*r1*w^3*xl -
    6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
    4*ka^2*l1*r1*w^5*xl + 12*kb*l1*r1*w^5*xl -
    6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
    16*ka*l1*r1*w^7*xl - 24*ka*l1*rtot*w^7*xl +
    12*l1*r1*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*r1*w^2*xl^2 -
    4*kc*r1*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
    9*kc*rtot*w^2*xl^2 - 2*ka^2*r1*w^4*xl^2 -
    7*kb*r1*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
    18*kb*rtot*w^4*xl^2 - 13*ka*r1*w^6*xl^2 +
    36*ka*rtot*w^6*xl^2 - 13*r1*w^8*xl^2 + 36*rtot*w^8*xl^2)*
  (ka*kc*l1*r1^3*w - 6*ka*kc*l1*r1^2*rtot*w +
    ka*kb*l1*r1^3*w^3 + 3*kc*l1*r1^3*w^3 -
    6*ka*kb*l1*r1^2*rtot*w^3 - 18*kc*l1*r1^2*rtot*w^3 +
    ka^2*l1*r1^3*w^5 + 3*kb*l1*r1^3*w^5 -
    6*ka^2*l1*r1^2*rtot*w^5 - 18*kb*l1*r1^2*rtot*w^5 +
    4*ka*l1*r1^3*w^7 - 24*ka*l1*r1^2*rtot*w^7 + 3*l1*r1^3*w^9 -
    18*l1*r1^2*rtot*w^9 - ka*kc*r1^2*rtot*xl - kc*r1^3*w^2*xl +
    2*ka*kb*r1^2*rtot*w^2*xl + 2*kc*r1^2*rtot*w^2*xl -
    kb*r1^3*w^4*xl + 5*ka^2*r1^2*rtot*w^4*xl +
    11*kb*r1^2*rtot*w^4*xl - ka*r1^3*w^6*xl +
    28*ka*r1^2*rtot*w^6*xl - r1^3*w^8*xl +
    29*r1^2*rtot*w^8*xl + ka*kc*l1*r1*w*xl^2 -
    12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*r1*w^3*xl^2 +
    3*kc*l1*r1*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
    36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*r1*w^5*xl^2 +
    3*kb*l1*r1*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
    36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*r1*w^7*xl^2 -
    48*ka*l1*rtot*w^7*xl^2 + 3*l1*r1*w^9*xl^2 -
    36*l1*rtot*w^9*xl^2 - kc*r1*w^2*xl^3 +
    3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
    kb*r1*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
    18*kb*rtot*w^4*xl^3 - ka*r1*w^6*xl^3 +
    36*ka*rtot*w^6*xl^3 - r1*w^8*xl^3 + 36*rtot*w^8*xl^3))/
  (6*w^4*(-(ka*kb*r1) - 3*kc*r1 + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*r1*w^2 - 6*kb*r1*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*r1*w^4 + 12*ka*rtot*w^4 -
       12*r1*w^6 + 12*rtot*w^6)^2*xl^2) -
  (rtot*(-(ka*kc*r1^4) - 4*kc*r1^4*w^2 + ka^2*r1^4*w^4 -
       kb*r1^4*w^4 + 4*ka*r1^4*w^6 + 5*r1^4*w^8 -
       6*ka*kc*l1*r1^2*w*xl - 6*ka*kb*l1*r1^2*w^3*xl -
       18*kc*l1*r1^2*w^3*xl - 6*ka^2*l1*r1^2*w^5*xl -
       18*kb*l1*r1^2*w^5*xl - 24*ka*l1*r1^2*w^7*xl -
       18*l1*r1^2*w^9*xl - ka*kc*r1^2*xl^2 + ka*kb*r1^2*w^2*xl^2 -
       kc*r1^2*w^2*xl^2 + 3*ka^2*r1^2*w^4*xl^2 +
```

```
      5*kb*rl^2*w^4*xl^2 + 16*ka*rl^2*w^6*xl^2 +
      17*rl^2*w^8*xl^2 - 6*ka*kc*ll*w*xl^3 -
      6*ka*kb*ll*w^3*xl^3 - 18*kc*ll*w^3*xl^3 -
      6*ka^2*ll*w^5*xl^3 - 18*kb*ll*w^5*xl^3 -
      24*ka*ll*w^7*xl^3 - 18*ll*w^9*xl^3 + ka*kb*w^2*xl^4 +
      3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 +
      12*ka*w^6*xl^4 + 12*w^8*xl^4))/
   (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
      12*rtot*w^6)*xl) +
  ((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
        3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
        2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
        7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
        12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
        6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
        12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
        18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
        12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
        18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
        24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
        18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
        4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
        9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
        7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
        18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
        36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
        36*rtot*w^8*xl^2)^2/
     (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
    (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
      ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
      6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
      ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
      6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
      4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
      3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 -
      ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
      2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
      kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
      11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
      28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
      29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
      12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
      3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
      36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
      3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
      36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
      48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
      36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
      kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
      18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
   (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
  (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
      7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
      12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
```

```
      6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
      12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
      18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
      12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
      18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
      24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
      18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
      7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)^3/
 (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
     12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
   3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
   2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
   7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
   12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
   6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
   12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
   18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
   12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
   18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
   24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
   18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
   4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
   9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
   7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
   18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
   36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
   36*rtot*w^8*xl^2)*
  (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w +
   ka*kb*l1*rl^3*w^3 + 3*kc*l1*rl^3*w^3 -
   6*ka*kb*l1*rl^2*rtot*w^3 - 18*kc*l1*rl^2*rtot*w^3 +
   ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
   6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
   4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 +
   3*l1*rl^3*w^9 - 18*l1*rl^2*rtot*w^9 -
   ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
   2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
   kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
   11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
   28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
   29*rl^2*rtot*w^8*xl + ka*kc*l1*rl*w*xl^2 -
   12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*rl*w^3*xl^2 +
   3*kc*l1*rl*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
   36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*rl*w^5*xl^2 +
   3*kb*l1*rl*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
   36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*rl*w^7*xl^2 -
   48*ka*l1*rtot*w^7*xl^2 + 3*l1*rl*w^9*xl^2 -
   36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
   3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
   kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
   18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
   36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))
/(6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot +
    3*kc*rtot - 2*ka^2*rl*w^2 - 6*kb*rl*w^2 +
    2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 - 12*ka*rl*w^4 +
    12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
    kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
    6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
```

```
                    18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
                    18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
                    18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
                    ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
                    3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
                    16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
                    6*ka*kc*l1*w*xl^3 - 6*ka*kb*l1*w^3*xl^3 -
                    18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 -
                    18*kb*l1*w^5*xl^3 - 24*ka*l1*w^7*xl^3 -
                    18*l1*w^9*xl^3 + ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 +
                    2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 +
                    12*w^8*xl^4))/
                (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
                    2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
                    6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
                    12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3)) +
        (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
            3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
            2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
            12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
            4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
            4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
            6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
            4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
            6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
            16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
            18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
            3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
            2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
            18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
            36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^3/
         (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
             2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
             6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
             12*rtot*w^6)^3*xl^3) +
        ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
            3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
            2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
            12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
            4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
            4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
            6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
            4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
            6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
            16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
            18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
            3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
            2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
            18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
            36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
          (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w + ka*kb*l1*rl^3*w^3 +
            3*kc*l1*rl^3*w^3 - 6*ka*kb*l1*rl^2*rtot*w^3 -
            18*kc*l1*rl^2*rtot*w^3 + ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
            6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
            4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
            18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
            2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
            kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
            11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
            28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
            ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
            ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
            12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
            ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
            12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
            4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
```

```
      3*l1*r1*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*r1*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 - kb*r1*w^4*xl^3 +
      6*ka^2*rtot*w^4*xl^3 + 18*kb*rtot*w^4*xl^3 - ka*r1*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - r1*w^8*xl^3 + 36*rtot*w^8*xl^3))/
   (6*w^4*(-(ka*kb*r1) - 3*kc*r1 + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*r1*w^2 - 6*kb*r1*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*r1*w^4 + 12*ka*rtot*w^4 - 12*r1*w^6 +
       12*rtot*w^6)^2*xl^2) -
  (rtot*(-(ka*kc*r1^4) - 4*kc*r1^4*w^2 + ka^2*r1^4*w^4 -
      kb*r1^4*w^4 + 4*ka*r1^4*w^6 + 5*r1^4*w^8 -
      6*ka*kc*l1*r1^2*w*xl - 6*ka*kb*l1*r1^2*w^3*xl -
      18*kc*l1*r1^2*w^3*xl - 6*ka^2*l1*r1^2*w^5*xl -
      18*kb*l1*r1^2*w^5*xl - 24*ka*l1*r1^2*w^7*xl -
      18*l1*r1^2*w^9*xl - ka*kc*r1^2*xl^2 + ka*kb*r1^2*w^2*xl^2 -
      kc*r1^2*w^2*xl^2 + 3*ka^2*r1^2*w^4*xl^2 + 5*kb*r1^2*w^4*xl^2 +
      16*ka*r1^2*w^6*xl^2 + 17*r1^2*w^8*xl^2 - 6*ka*kc*l1*w*xl^3 -
      6*ka*kb*l1*w^3*xl^3 - 18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 -
      18*kb*l1*w^5*xl^3 - 24*ka*l1*w^7*xl^3 - 18*l1*w^9*xl^3 +
      ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 +
      6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 + 12*w^8*xl^4))/
   (2*w^2*(-(ka*kb*r1) - 3*kc*r1 + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*r1*w^2 - 6*kb*r1*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*r1*w^4 + 12*ka*rtot*w^4 - 12*r1*w^6 +
      12*rtot*w^6)*xl) +
  ((-(ka*kc*r1^3 + kc*r1^3*w^2 + ka*kb*r1^2*rtot*w^2 +
         3*kc*r1^2*rtot*w^2 - ka^2*r1^3*w^4 - 2*kb*r1^3*w^4 +
         2*ka^2*r1^2*rtot*w^4 + 6*kb*r1^2*rtot*w^4 -
         7*ka*r1^3*w^6 + 12*ka*r1^2*rtot*w^6 - 8*r1^3*w^8 +
         12*r1^2*rtot*w^8 + 4*ka*kc*l1*r1*w*xl -
         6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*r1*w^3*xl +
         12*kc*l1*r1*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
         18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*r1*w^5*xl +
         12*kb*l1*r1*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
         18*kb*l1*rtot*w^5*xl + 16*ka*l1*r1*w^7*xl -
         24*ka*l1*rtot*w^7*xl + 12*l1*r1*w^9*xl -
         18*l1*rtot*w^9*xl - ka*kb*r1*w^2*xl^2 -
         4*kc*r1*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
         9*kc*rtot*w^2*xl^2 - 2*ka^2*r1*w^4*xl^2 -
         7*kb*r1*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
         18*kb*rtot*w^4*xl^2 - 13*ka*r1*w^6*xl^2 +
         36*ka*rtot*w^6*xl^2 - 13*r1*w^8*xl^2 + 36*rtot*w^8*xl^2)^
       2/
   (9*w^4*(-(ka*kb*r1) - 3*kc*r1 + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*r1*w^2 - 6*kb*r1*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*r1*w^4 + 12*ka*rtot*w^4 -
       12*r1*w^6 + 12*rtot*w^6)^2*xl^2) +
  (ka*kc*l1*r1^3*w - 6*ka*kc*l1*r1^2*rtot*w +
     ka*kb*l1*r1^3*w^3 + 3*kc*l1*r1^3*w^3 -
     6*ka*kb*l1*r1^2*rtot*w^3 - 18*kc*l1*r1^2*rtot*w^3 +
     ka^2*l1*r1^3*w^5 + 3*kb*l1*r1^3*w^5 -
     6*ka^2*l1*r1^2*rtot*w^5 - 18*kb*l1*r1^2*rtot*w^5 +
     4*ka*l1*r1^3*w^7 - 24*ka*l1*r1^2*rtot*w^7 +
     3*l1*r1^3*w^9 - 18*l1*r1^2*rtot*w^9 - ka*kc*r1^2*rtot*xl -
     kc*r1^3*w^2*xl + 2*ka*kb*r1^2*rtot*w^2*xl +
     2*kc*r1^2*rtot*w^2*xl - kb*r1^3*w^4*xl +
     5*ka^2*r1^2*rtot*w^4*xl + 11*kb*r1^2*rtot*w^4*xl -
     ka*r1^3*w^6*xl + 28*ka*r1^2*rtot*w^6*xl - r1^3*w^8*xl +
     29*r1^2*rtot*w^8*xl + ka*kc*l1*r1*w*xl^2 -
     12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*r1*w^3*xl^2 +
     3*kc*l1*r1*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
     36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*r1*w^5*xl^2 +
     3*kb*l1*r1*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
     36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*r1*w^7*xl^2 -
     48*ka*l1*rtot*w^7*xl^2 + 3*l1*r1*w^9*xl^2 -
     36*l1*rtot*w^9*xl^2 - kc*r1*w^2*xl^3 +
     3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
```

```
      kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
      18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
    (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
       12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
  (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
       3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
       2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
       7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
       12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
       6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
       12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
       18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
       12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
       18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
       24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
       18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
       4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
       9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
       7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
       18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
       36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^
     3/
   (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
  ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
       3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
       2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
       7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
       12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
       6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
       12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
       18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
       12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
       18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
       24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
       18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
       4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
       9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
       7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
       18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
       36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
     (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
       ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
       6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
       ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
       6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
       4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
       3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 -
       ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
       2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
       kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
       11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
       28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
       29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
       12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
       3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
       36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
       3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
       36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
       48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
       36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
```

```
     3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
     kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
     18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
     36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
 (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
     kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
     6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
     18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
     18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
     18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
     ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
     3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
     16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
        6*ka*kc*l1*w*xl^3 - 6*ka*kb*l1*w^3*xl^3 -
        18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 -
        18*kb*l1*w^5*xl^3 - 24*ka*l1*w^7*xl^3 - 18*l1*w^9*xl^3 +
        ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 +
        6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 + 12*w^8*xl^4))/
     (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3))/2 +
((-3)^(1/2)*((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
        3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
        2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
        12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
        4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
        4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
        6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
        4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
        6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
        16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
        12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
        4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
        9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
        6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
        13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
        36*rtot*w^8*xl^2)^2/
     (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
        12*rtot*w^6)^2*xl^2) +
    (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w + ka*kb*l1*rl^3*w^3 +
       3*kc*l1*rl^3*w^3 - 6*ka*kb*l1*rl^2*rtot*w^3 -
       18*kc*l1*rl^2*rtot*w^3 + ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
       6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
       4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
       18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
       2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
       kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
       11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
       28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
       ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
       ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
       12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
       ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
       12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
       4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
       3*l1*rl*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
       3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 +
       6*ka^2*rtot*w^4*xl^3 + 18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
       36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
```

```
   (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
       12*rtot*w^6)*xl))/
(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
       3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
       2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
       12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
       4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
       4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
       6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
       4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
       6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
       16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
       12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
       4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
       9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
       7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
       18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
       36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^3/
 (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
     3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
     2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
     12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
     4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
     4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
     6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
     4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
     6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
     16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
     12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
     4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
     9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
     7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
     18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
     36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
   (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
     ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
     6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
     ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
     6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
     4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 + 3*ll*rl^3*w^9 -
     18*ll*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
     2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
     kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
     11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
     28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
     29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
     12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
     3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
     36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
     3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
     36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
     48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
     36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
     3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
     kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
     18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
     36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
(6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
```

```
      12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
     kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
     6*ka*kc*ll*rl^2*w*xl - 6*ka*kb*ll*rl^2*w^3*xl -
     18*kc*ll*rl^2*w^3*xl - 6*ka^2*ll*rl^2*w^5*xl -
     18*kb*ll*rl^2*w^5*xl - 24*ka*ll*rl^2*w^7*xl -
     18*ll*rl^2*w^9*xl - ka*kc*rl^2*xl^2 + ka*kb*rl^2*w^2*xl^2 -
     kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 +
     5*kb*rl^2*w^4*xl^2 + 16*ka*rl^2*w^6*xl^2 +
     17*rl^2*w^8*xl^2 - 6*ka*kc*ll*w*xl^3 -
     6*ka*kb*ll*w^3*xl^3 - 18*kc*ll*w^3*xl^3 -
     6*ka^2*ll*w^5*xl^3 - 18*kb*ll*w^5*xl^3 -
     24*ka*ll*w^7*xl^3 - 18*ll*w^9*xl^3 + ka*kb*w^2*xl^4 +
     3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 +
     12*ka*w^6*xl^4 + 12*w^8*xl^4))/
 (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
     12*rtot*w^6)*xl) +
   ((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
          3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
          2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
          7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
          12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
          6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
          12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
          18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
          12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
          18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
          24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
          18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
          4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
          9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
          7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
          18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
          36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
          36*rtot*w^8*xl^2)^2/
        (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
          2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
          6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
          12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
      (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
         ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
         6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
         ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
         6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
         4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
         3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 -
         ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
         2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
         kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
         11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
         28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
         29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
         12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
         3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
         36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
         3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
         36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
         48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
         36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
         3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
         kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
         18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
         36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
      (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
```

```
          2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
          6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
          12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
          3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
          2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
          7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
          12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
          6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
          12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
          18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
          12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
          18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
          24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
          18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
          4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
          9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
          7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
          18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
          36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
          36*rtot*w^8*xl^2)^3/
 (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
           2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
           6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
           12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
          3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
          2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
          7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
          12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
          6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
          12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
          18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
          12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
          18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
          24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
          18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
          4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
          9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
          7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
          18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
          36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
          36*rtot*w^8*xl^2)*
(ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
          ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
          6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
          ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
          6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
          4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
          3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 -
          ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
          2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
          kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
          11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
          28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
          29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
          12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
          3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
          36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
          3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
          36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
          48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
          36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
          3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
          kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
          18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
```

```
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))
   /(6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot +
       3*kc*rtot - 2*ka^2*rl*w^2 - 6*kb*rl*w^2 +
       2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 - 12*ka*rl*w^4 +
       12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
  (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
      kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
      6*ka*kc*ll*rl^2*w*xl - 6*ka*kb*ll*rl^2*w^3*xl -
      18*kc*ll*rl^2*w^3*xl - 6*ka^2*ll*rl^2*w^5*xl -
      18*kb*ll*rl^2*w^5*xl - 24*ka*ll*rl^2*w^7*xl -
      18*ll*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
      ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
      3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
      16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
      6*ka*kc*ll*w*xl^3 - 6*ka*kb*ll*w^3*xl^3 -
      18*kc*ll*w^3*xl^3 - 6*ka^2*ll*w^5*xl^3 -
      18*kb*ll*w^5*xl^3 - 24*ka*ll*w^7*xl^3 -
      18*ll*w^9*xl^3 + ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 +
      2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 +
      12*w^8*xl^4))/
    (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
       12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3) +
 (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
      12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
      4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
      4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
      6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
      4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
      6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
      16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
      12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
      6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
      13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)^3/
   (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
      12*rtot*w^6)^3*xl^3) +
  ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
      12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
      4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
      4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
      6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
      4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
      6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
      16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
      12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
      6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
      13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)*
    (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
      ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
      6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
      ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
      6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
      4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 + 3*ll*rl^3*w^9 -
```

```
      18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
      2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
      kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
      11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
      28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
      ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
      ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
      12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
      ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
      12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
      4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
      3*l1*rl*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
      kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
      18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 + 36*ka*rtot*w^6*xl^3 -
      rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
   (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
       12*rtot*w^6)^2*xl^2) -
   (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
       kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
       6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
       18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
       18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
       18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 + ka*kb*rl^2*w^2*xl^2 -
       kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 +
       5*kb*rl^2*w^4*xl^2 + 16*ka*rl^2*w^6*xl^2 +
       17*rl^2*w^8*xl^2 - 6*ka*kc*l1*w*xl^3 - 6*ka*kb*l1*w^3*xl^3 -
       18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 - 18*kb*l1*w^5*xl^3 -
       24*ka*l1*w^7*xl^3 - 18*l1*w^9*xl^3 + ka*kb*w^2*xl^4 +
       3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 +
       12*ka*w^6*xl^4 + 12*w^8*xl^4))/
    (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
       12*rtot*w^6)*xl) +
   ((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
       3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
       2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
       7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
       12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
       6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
       12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
       18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
       12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
       18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
       24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
       18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
       4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
       9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
       7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
       18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
       36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2
       )^2/
    (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
       12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
   (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w +
   ka*kb*l1*rl^3*w^3 + 3*kc*l1*rl^3*w^3 -
   6*ka*kb*l1*rl^2*rtot*w^3 - 18*kc*l1*rl^2*rtot*w^3 +
   ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
   6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
   4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 +
   3*l1*rl^3*w^9 - 18*l1*rl^2*rtot*w^9 -
```

```
  ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
  2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
  kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
  11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
  28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
  29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
  12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
  3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
  36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
  3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
  36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
  48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
  36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
  3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
  kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
  18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
  36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
(3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
     12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
     3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
     2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
     7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
     12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
     6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
     12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
     18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
     12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
     18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
     24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
     18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
     4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
     9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
     7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
     18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
     36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2
    )^3/
 (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
     3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
     2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
     7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
     12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
     6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
     12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
     18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
     12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
     18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
     24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
     18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
     4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
     9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
     7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
     18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
     36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2
    )*(ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
    ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
    6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
    ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
    6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
    4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
```

```
3*l1*rl^3*w^9 - 18*l1*rl^2*rtot*w^9 -
ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
29*rl^2*rtot*w^8*xl + ka*kc*l1*rl*w*xl^2 -
12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*rl*w^3*xl^2 +
3*kc*l1*rl*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*rl*w^5*xl^2 +
3*kb*l1*rl*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*rl*w^7*xl^2 -
48*ka*l1*rtot*w^7*xl^2 + 3*l1*rl*w^9*xl^2 -
36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
(6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
6*ka*kc*l1*w*xl^3 - 6*ka*kb*l1*w^3*xl^3 -
18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 -
18*kb*l1*w^5*xl^3 - 24*ka*l1*w^7*xl^3 -
18*l1*w^9*xl^3 + ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 +
2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 +
12*w^8*xl^4))/
(2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3)))/2)];


xppc = N[(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 -
13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)/
(3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot - 2*ka^2*rl*w^2 -
6*kb*rl*w^2 + 2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 - 12*ka*rl*w^4 +
12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)*xl) -
(-((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
```

```
      6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
      4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
      6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
      16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
      12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
      6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
      13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)^2/
   (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
        12*rtot*w^6)^2*xl^2) +
  (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w + ka*kb*l1*rl^3*w^3 +
    3*kc*l1*rl^3*w^3 - 6*ka*kb*l1*rl^2*rtot*w^3 -
    18*kc*l1*rl^2*rtot*w^3 + ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
    6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
    4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
    18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
    2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
    kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
    11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
    28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
    ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
    ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
    12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
    ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
    12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
    4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
    3*l1*rl*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
    3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 +
    6*ka^2*rtot*w^4*xl^3 + 18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
    36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
  (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
      12*rtot*w^6)*xl))/
   (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
        3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
        2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
        12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
        4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
        4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
        6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
        4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
        6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
        16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
        12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
        4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
        9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
        7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
        18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
        36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^3/
     (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
         2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
         6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
         12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
    ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
    3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
    2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
    12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
    4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
    4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
    6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
    4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
```

```
  6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
  16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl +
  12*l1*rl*w^9*xl - 18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
  4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
  9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
  7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
  18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
  36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
(ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w +
  ka*kb*l1*rl^3*w^3 + 3*kc*l1*rl^3*w^3 -
  6*ka*kb*l1*rl^2*rtot*w^3 - 18*kc*l1*rl^2*rtot*w^3 +
  ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
  6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
  4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
  18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
  2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
  kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
  11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
  28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
  29*rl^2*rtot*w^8*xl + ka*kc*l1*rl*w*xl^2 -
  12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*rl*w^3*xl^2 +
  3*kc*l1*rl*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
  36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*rl*w^5*xl^2 +
  3*kb*l1*rl*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
  36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*rl*w^7*xl^2 -
  48*ka*l1*rtot*w^7*xl^2 + 3*l1*rl*w^9*xl^2 -
  36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
  3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
  kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
  18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
  36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
 (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
     12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
    kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
    6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
    18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
    18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
    18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 + ka*kb*rl^2*w^2*xl^2 -
    kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 +
    5*kb*rl^2*w^4*xl^2 + 16*ka*rl^2*w^6*xl^2 +
    17*rl^2*w^8*xl^2 - 6*ka*kc*l1*w*xl^3 -
    6*ka*kb*l1*w^3*xl^3 - 18*kc*l1*w^3*xl^3 -
    6*ka^2*l1*w^5*xl^3 - 18*kb*l1*w^5*xl^3 -
    24*ka*l1*w^7*xl^3 - 18*l1*w^9*xl^3 + ka*kb*w^2*xl^4 +
    3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 +
    12*ka*w^6*xl^4 + 12*w^8*xl^4))/
 (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
    2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
    6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
    12*rtot*w^6)*xl) +
((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
          3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
          2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
          7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
          12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
```

```
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
      7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)^2/
   (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
  (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
     ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
     6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
     ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
     6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
     4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
     3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 -
     ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
     2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
     kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
     11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
     28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
     29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
     12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
     3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
     36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
     3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
     36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
     48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
     36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
     3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
     kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
     18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
     36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
    (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
       12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
  (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
       3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
       2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
       7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
       12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
       6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
       12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
       18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
       12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
       18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
       24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
       18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
       4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
       9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
       7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
       18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
       36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
       36*rtot*w^8*xl^2)^3/
  (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
  ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
      7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
      12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
      6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
      12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
```

```
  18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
  12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
  18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
  24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
  18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
  4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
  9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
  7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
  18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
  36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
  36*rtot*w^8*xl^2)*
(ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
  ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
  6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
  ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
  6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
  4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
  3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 -
  ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
  2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
  kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
  11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
  28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
  29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
  12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
  3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
  36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
  3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
  36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
  48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
  36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
  3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
  kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
  18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
  36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))
/(6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot +
    3*kc*rtot - 2*ka^2*rl*w^2 - 6*kb*rl*w^2 +
    2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 - 12*ka*rl*w^4 +
    12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
    kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
    6*ka*kc*ll*rl^2*w*xl - 6*ka*kb*ll*rl^2*w^3*xl -
    18*kc*ll*rl^2*w^3*xl - 6*ka^2*ll*rl^2*w^5*xl -
    18*kb*ll*rl^2*w^5*xl - 24*ka*ll*rl^2*w^7*xl -
    18*ll*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
    ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
    3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
    16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
    6*ka*kc*ll*w*xl^3 - 6*ka*kb*ll*w^3*xl^3 -
    18*kc*ll*w^3*xl^3 - 6*ka^2*ll*w^5*xl^3 -
    18*kb*ll*w^5*xl^3 - 24*ka*ll*w^7*xl^3 -
    18*ll*w^9*xl^3 + ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 +
    2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 +
    12*w^8*xl^4))/
 (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
    2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
    6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
    12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3)) +
 (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
       3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
       2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
       12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
       4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
       4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
       6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
       4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
```

```
      6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
      16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
      18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
      3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
      2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^3/
   (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
       12*rtot*w^6)^3*xl^3) +
  ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
      12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
      4*ka*kc*l1*rl*w*xl - 6*ka*kc*l1*rtot*w*xl +
      4*ka*kb*l1*rl*w^3*xl + 12*kc*l1*rl*w^3*xl -
      6*ka*kb*l1*rtot*w^3*xl - 18*kc*l1*rtot*w^3*xl +
      4*ka^2*l1*rl*w^5*xl + 12*kb*l1*rl*w^5*xl -
      6*ka^2*l1*rtot*w^5*xl - 18*kb*l1*rtot*w^5*xl +
      16*ka*l1*rl*w^7*xl - 24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
      18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 - 4*kc*rl*w^2*xl^2 +
      3*ka*kb*rtot*w^2*xl^2 + 9*kc*rtot*w^2*xl^2 -
      2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
    (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w + ka*kb*l1*rl^3*w^3 +
      3*kc*l1*rl^3*w^3 - 6*ka*kb*l1*rl^2*rtot*w^3 -
      18*kc*l1*rl^2*rtot*w^3 + ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
      6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
      4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 + 3*l1*rl^3*w^9 -
      18*l1*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
      2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
      kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
      11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
      28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
      ka*kc*l1*rl*w*xl^2 - 12*ka*kc*l1*rtot*w*xl^2 +
      ka*kb*l1*rl*w^3*xl^2 + 3*kc*l1*rl*w^3*xl^2 -
      12*ka*kb*l1*rtot*w^3*xl^2 - 36*kc*l1*rtot*w^3*xl^2 +
      ka^2*l1*rl*w^5*xl^2 + 3*kb*l1*rl*w^5*xl^2 -
      12*ka^2*l1*rtot*w^5*xl^2 - 36*kb*l1*rtot*w^5*xl^2 +
      4*ka*l1*rl*w^7*xl^2 - 48*ka*l1*rtot*w^7*xl^2 +
      3*l1*rl*w^9*xl^2 - 36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 +
      6*ka^2*rtot*w^4*xl^3 + 18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
   (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
       2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
       6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
       12*rtot*w^6)^2*xl^2) -
  (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
      kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
      6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
      18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
      18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
      18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 + ka*kb*rl^2*w^2*xl^2 -
      kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
      16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 - 6*ka*kc*l1*w*xl^3 -
      6*ka*kb*l1*w^3*xl^3 - 18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 -
      18*kb*l1*w^5*xl^3 - 24*ka*l1*w^7*xl^3 - 18*l1*w^9*xl^3 +
      ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 +
      6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 + 12*w^8*xl^4))/
  (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
      12*rtot*w^6)*xl) +
```

```
((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
            3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
            2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
            7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
            12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
            6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
            12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
            18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
            12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
            18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
            24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
            18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
            4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
            9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
            7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
            18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
            36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^
           2/
          (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
              2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
              6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
              12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
         (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
            ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
            6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
            ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
            6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
            4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
            3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl -
            kc*rl^3*w^2*xl + 2*ka*kb*rl^2*rtot*w^2*xl +
            2*kc*rl^2*rtot*w^2*xl - kb*rl^3*w^4*xl +
            5*ka^2*rl^2*rtot*w^4*xl + 11*kb*rl^2*rtot*w^4*xl -
            ka*rl^3*w^6*xl + 28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
            29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
            12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
            3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
            36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
            3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
            36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
            48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
            36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
            3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
            kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
            18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
            36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
          (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
              2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
              6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
              12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
        (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
            3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
            2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
            7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
            12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
            6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
            12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
            18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
            12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
            18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
            24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
            18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
            4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
            9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
            7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
            18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
            36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)^
```

```
  3/
(27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
     12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
    3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
    2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
    7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
    12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
    6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
    12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
    18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
    12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
    18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
    24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
    18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
    4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
    9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
    7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
    18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
    36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
  (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
    ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
    6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
    ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
    6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
    4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
    3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 -
    ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
    2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
    kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
    11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
    28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
    29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
    12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
    3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
    36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
    3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
    36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
    48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
    36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
    3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
    kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
    18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
    36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
 (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
     12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
    kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
    6*ka*kc*ll*rl^2*w*xl - 6*ka*kb*ll*rl^2*w^3*xl -
    18*kc*ll*rl^2*w^3*xl - 6*ka^2*ll*rl^2*w^5*xl -
    18*kb*ll*rl^2*w^5*xl - 24*ka*ll*rl^2*w^7*xl -
    18*ll*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
    ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
    3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
    16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
    6*ka*kc*ll*w*xl^3 - 6*ka*kb*ll*w^3*xl^3 -
    18*kc*ll*w^3*xl^3 - 6*ka^2*ll*w^5*xl^3 -
    18*kb*ll*w^5*xl^3 - 24*ka*ll*w^7*xl^3 - 18*ll*w^9*xl^3 +
    ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 +
    6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 + 12*w^8*xl^4))/
 (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
    2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
```

```
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3))/2 -
(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
36*rtot*w^8*xl^2)^2/
 (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
     12*rtot*w^6)^2*xl^2) +
(ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w + ka*kb*ll*rl^3*w^3 +
   3*kc*ll*rl^3*w^3 - 6*ka*kb*ll*rl^2*rtot*w^3 -
   18*kc*ll*rl^2*rtot*w^3 + ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
   6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
   4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 + 3*ll*rl^3*w^9 -
   18*ll*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
   2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
   kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
   11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
   28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
   ka*kc*ll*rl*w*xl^2 - 12*ka*kc*ll*rtot*w*xl^2 +
   ka*kb*ll*rl*w^3*xl^2 + 3*kc*ll*rl*w^3*xl^2 -
   12*ka*kb*ll*rtot*w^3*xl^2 - 36*kc*ll*rtot*w^3*xl^2 +
   ka^2*ll*rl*w^5*xl^2 + 3*kb*ll*rl*w^5*xl^2 -
   12*ka^2*ll*rtot*w^5*xl^2 - 36*kb*ll*rtot*w^5*xl^2 +
   4*ka*ll*rl*w^7*xl^2 - 48*ka*ll*rtot*w^7*xl^2 +
   3*ll*rl*w^9*xl^2 - 36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
   3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 - kb*rl*w^4*xl^3 +
   6*ka^2*rtot*w^4*xl^3 + 18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
   36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
  (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
      12*rtot*w^6)*xl))/
 (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
      12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
      4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
      4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
      6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
      4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
      6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
      16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
      12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
      7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*
(27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rto
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
     12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
```

```
((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
     3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
     2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
     12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
     4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
     4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
     6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
     4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
     6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
     16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
     12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
     4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
     9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
     7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
     18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
     36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2)*
   (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
    ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
    6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
    ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
    6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
    4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 + 3*ll*rl^3*w^9 -
    18*ll*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
    2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
    kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
    11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
    28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
    29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
    12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
    3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
    36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
    3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
    36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
    48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
    36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
    3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
    kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
    18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
    36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
  (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
     12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
 (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
     kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
     6*ka*kc*ll*rl^2*w*xl - 6*ka*kb*ll*rl^2*w^3*xl -
     18*kc*ll*rl^2*w^3*xl - 6*ka^2*ll*rl^2*w^5*xl -
     18*kb*ll*rl^2*w^5*xl - 24*ka*ll*rl^2*w^7*xl -
     18*ll*rl^2*w^9*xl - ka*kc*rl^2*xl^2 + ka*kb*rl^2*w^2*xl^2 -
     kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 +
     5*kb*rl^2*w^4*xl^2 + 16*ka*rl^2*w^6*xl^2 +
     17*rl^2*w^8*xl^2 - 6*ka*kc*ll*w*xl^3 -
     6*ka*kb*ll*w^3*xl^3 - 18*kc*ll*w^3*xl^3 -
     6*ka^2*ll*w^5*xl^3 - 18*kb*ll*w^5*xl^3 -
     24*ka*ll*w^7*xl^3 - 18*ll*w^9*xl^3 + ka*kb*w^2*xl^4 +
     3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 +
     12*ka*w^6*xl^4 + 12*w^8*xl^4))/
  (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
     12*rtot*w^6)*xl) +
 ((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
        3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
        2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
        7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
        12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
```

```
      6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
      12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
      18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
      12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
      18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
      24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
      18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
      7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)^2/
   (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
  (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w +
   ka*kb*l1*rl^3*w^3 + 3*kc*l1*rl^3*w^3 -
   6*ka*kb*l1*rl^2*rtot*w^3 - 18*kc*l1*rl^2*rtot*w^3 +
   ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
   6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
   4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 +
   3*l1*rl^3*w^9 - 18*l1*rl^2*rtot*w^9 -
   ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
   2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
   kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
   11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
   28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
   29*rl^2*rtot*w^8*xl + ka*kc*l1*rl*w*xl^2 -
   12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*rl*w^3*xl^2 +
   3*kc*l1*rl*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
   36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*rl*w^5*xl^2 +
   3*kb*l1*rl*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
   36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*rl*w^7*xl^2 -
   48*ka*l1*rtot*w^7*xl^2 + 3*l1*rl*w^9*xl^2 -
   36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
   3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
   kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
   18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
   36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
  (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
 (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
      7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
      12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
      6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
      12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
      18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
      12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
      18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
      24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
      18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
      7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
      36*rtot*w^8*xl^2)^3/
 (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
```

```
      12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
    3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
    2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
    7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
    12*rl^2*rtot*w^8 + 4*ka*kc*ll*rl*w*xl -
    6*ka*kc*ll*rtot*w*xl + 4*ka*kb*ll*rl*w^3*xl +
    12*kc*ll*rl*w^3*xl - 6*ka*kb*ll*rtot*w^3*xl -
    18*kc*ll*rtot*w^3*xl + 4*ka^2*ll*rl*w^5*xl +
    12*kb*ll*rl*w^5*xl - 6*ka^2*ll*rtot*w^5*xl -
    18*kb*ll*rtot*w^5*xl + 16*ka*ll*rl*w^7*xl -
    24*ka*ll*rtot*w^7*xl + 12*ll*rl*w^9*xl -
    18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
    4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
    9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
    7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
    18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
    36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
    36*rtot*w^8*xl^2)*
  (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
    ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
    6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
    ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
    6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
    4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 +
    3*ll*rl^3*w^9 - 18*ll*rl^2*rtot*w^9 -
    ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
    2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
    kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
    11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
    28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
    29*rl^2*rtot*w^8*xl + ka*kc*ll*rl*w*xl^2 -
    12*ka*kc*ll*rtot*w*xl^2 + ka*kb*ll*rl*w^3*xl^2 +
    3*kc*ll*rl*w^3*xl^2 - 12*ka*kb*ll*rtot*w^3*xl^2 -
    36*kc*ll*rtot*w^3*xl^2 + ka^2*ll*rl*w^5*xl^2 +
    3*kb*ll*rl*w^5*xl^2 - 12*ka^2*ll*rtot*w^5*xl^2 -
    36*kb*ll*rtot*w^5*xl^2 + 4*ka*ll*rl*w^7*xl^2 -
    48*ka*ll*rtot*w^7*xl^2 + 3*ll*rl*w^9*xl^2 -
    36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
    3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
    kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
    18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
    36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))
 /(6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot +
    3*kc*rtot - 2*ka^2*rl*w^2 - 6*kb*rl*w^2 +
    2*ka^2*rtot*w^2 + 6*kb*rtot*w^2 - 12*ka*rl*w^4 +
    12*ka*rtot*w^4 - 12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
    kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
    6*ka*kc*ll*rl^2*w*xl - 6*ka*kb*ll*rl^2*w^3*xl -
    18*kc*ll*rl^2*w^3*xl - 6*ka^2*ll*rl^2*w^5*xl -
    18*kb*ll*rl^2*w^5*xl - 24*ka*ll*rl^2*w^7*xl -
    18*ll*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
    ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
    3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
    16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
    6*ka*kc*ll*w*xl^3 - 6*ka*kb*ll*w^3*xl^3 -
    18*kc*ll*w^3*xl^3 - 6*ka^2*ll*w^5*xl^3 -
    18*kb*ll*w^5*xl^3 - 24*ka*ll*w^7*xl^3 -
    18*ll*w^9*xl^3 + ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 +
    2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 +
    12*w^8*xl^4))/
(2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
    2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
    6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
    12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3) +
```

```
(-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
        3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
        2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
        12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
        4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
        4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
        6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
        4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
        6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
        16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
        12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
        4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
        9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
        6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
        13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
        36*rtot*w^8*xl^2)^3/
    (27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
         2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
         6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
         12*rtot*w^6)^3*xl^3) +
   ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
       3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
       2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 - 7*ka*rl^3*w^6 +
       12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 + 12*rl^2*rtot*w^8 +
       4*ka*kc*ll*rl*w*xl - 6*ka*kc*ll*rtot*w*xl +
       4*ka*kb*ll*rl*w^3*xl + 12*kc*ll*rl*w^3*xl -
       6*ka*kb*ll*rtot*w^3*xl - 18*kc*ll*rtot*w^3*xl +
       4*ka^2*ll*rl*w^5*xl + 12*kb*ll*rl*w^5*xl -
       6*ka^2*ll*rtot*w^5*xl - 18*kb*ll*rtot*w^5*xl +
       16*ka*ll*rl*w^7*xl - 24*ka*ll*rtot*w^7*xl +
       12*ll*rl*w^9*xl - 18*ll*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
       4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
       9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 - 7*kb*rl*w^4*xl^2 +
       6*ka^2*rtot*w^4*xl^2 + 18*kb*rtot*w^4*xl^2 -
       13*ka*rl*w^6*xl^2 + 36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 +
       36*rtot*w^8*xl^2)*
     (ka*kc*ll*rl^3*w - 6*ka*kc*ll*rl^2*rtot*w +
      ka*kb*ll*rl^3*w^3 + 3*kc*ll*rl^3*w^3 -
      6*ka*kb*ll*rl^2*rtot*w^3 - 18*kc*ll*rl^2*rtot*w^3 +
      ka^2*ll*rl^3*w^5 + 3*kb*ll*rl^3*w^5 -
      6*ka^2*ll*rl^2*rtot*w^5 - 18*kb*ll*rl^2*rtot*w^5 +
      4*ka*ll*rl^3*w^7 - 24*ka*ll*rl^2*rtot*w^7 + 3*ll*rl^3*w^9 -
      18*ll*rl^2*rtot*w^9 - ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
      2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
      kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
      11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
      28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl + 29*rl^2*rtot*w^8*xl +
      ka*kc*ll*rl*w*xl^2 - 12*ka*kc*ll*rtot*w*xl^2 +
      ka*kb*ll*rl*w^3*xl^2 + 3*kc*ll*rl*w^3*xl^2 -
      12*ka*kb*ll*rtot*w^3*xl^2 - 36*kc*ll*rtot*w^3*xl^2 +
      ka^2*ll*rl*w^5*xl^2 + 3*kb*ll*rl*w^5*xl^2 -
      12*ka^2*ll*rtot*w^5*xl^2 - 36*kb*ll*rtot*w^5*xl^2 +
      4*ka*ll*rl*w^7*xl^2 - 48*ka*ll*rtot*w^7*xl^2 +
      3*ll*rl*w^9*xl^2 - 36*ll*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
      kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
      18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 + 36*ka*rtot*w^6*xl^3 -
      rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
    (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
         2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
         6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
         12*rtot*w^6)^2*xl^2) -
   (rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
       kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
       6*ka*kc*ll*rl^2*w*xl - 6*ka*kb*ll*rl^2*w^3*xl -
       18*kc*ll*rl^2*w^3*xl - 6*ka^2*ll*rl^2*w^5*xl -
```

```
      18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
      18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 + ka*kb*rl^2*w^2*xl^2 -
      kc*rl^2*w^2*xl^2 + 3*ka^2*rl^2*w^4*xl^2 +
      5*kb*rl^2*w^4*xl^2 + 16*ka*rl^2*w^6*xl^2 +
      17*rl^2*w^8*xl^2 - 6*ka*kc*l1*w*xl^3 - 6*ka*kb*l1*w^3*xl^3 -
      18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 - 18*kb*l1*w^5*xl^3 -
      24*ka*l1*w^7*xl^3 - 18*l1*w^9*xl^3 + ka*kb*w^2*xl^4 +
      3*kc*w^2*xl^4 + 2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 +
      12*ka*w^6*xl^4 + 12*w^8*xl^4))/
   (2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 - 12*rl*w^6 +
      12*rtot*w^6)*xl) +
  ((-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
            3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
            2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
            7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
            12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
            6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
            12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
            18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
            12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
            18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
            24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
            18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
            4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
            9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
            7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
            18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
            36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2
            )^2/
     (9*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
         2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
         6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
         12*rl*w^6 + 12*rtot*w^6)^2*xl^2) +
    (ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w +
      ka*kb*l1*rl^3*w^3 + 3*kc*l1*rl^3*w^3 -
      6*ka*kb*l1*rl^2*rtot*w^3 - 18*kc*l1*rl^2*rtot*w^3 +
      ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
      6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
      4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 +
      3*l1*rl^3*w^9 - 18*l1*rl^2*rtot*w^9 -
      ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
      2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
      kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
      11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
      28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
      29*rl^2*rtot*w^8*xl + ka*kc*l1*rl*w*xl^2 -
      12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*rl*w^3*xl^2 +
      3*kc*l1*rl*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
      36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*rl*w^5*xl^2 +
      3*kb*l1*rl*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
      36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*rl*w^7*xl^2 -
      48*ka*l1*rtot*w^7*xl^2 + 3*l1*rl*w^9*xl^2 -
      36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
      kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
      18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3)/
     (3*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
        2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
        6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
        12*rl*w^6 + 12*rtot*w^6)*xl))^3 +
   (-(ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
         3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
         2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
```

```
      7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
      12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
      6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
      12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
      18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
      12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
      18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
      24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
      18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
      7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2
      )^3/
(27*w^6*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
     2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
     6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
     12*rl*w^6 + 12*rtot*w^6)^3*xl^3) +
  ((ka*kc*rl^3 + kc*rl^3*w^2 + ka*kb*rl^2*rtot*w^2 +
      3*kc*rl^2*rtot*w^2 - ka^2*rl^3*w^4 - 2*kb*rl^3*w^4 +
      2*ka^2*rl^2*rtot*w^4 + 6*kb*rl^2*rtot*w^4 -
      7*ka*rl^3*w^6 + 12*ka*rl^2*rtot*w^6 - 8*rl^3*w^8 +
      12*rl^2*rtot*w^8 + 4*ka*kc*l1*rl*w*xl -
      6*ka*kc*l1*rtot*w*xl + 4*ka*kb*l1*rl*w^3*xl +
      12*kc*l1*rl*w^3*xl - 6*ka*kb*l1*rtot*w^3*xl -
      18*kc*l1*rtot*w^3*xl + 4*ka^2*l1*rl*w^5*xl +
      12*kb*l1*rl*w^5*xl - 6*ka^2*l1*rtot*w^5*xl -
      18*kb*l1*rtot*w^5*xl + 16*ka*l1*rl*w^7*xl -
      24*ka*l1*rtot*w^7*xl + 12*l1*rl*w^9*xl -
      18*l1*rtot*w^9*xl - ka*kb*rl*w^2*xl^2 -
      4*kc*rl*w^2*xl^2 + 3*ka*kb*rtot*w^2*xl^2 +
      9*kc*rtot*w^2*xl^2 - 2*ka^2*rl*w^4*xl^2 -
      7*kb*rl*w^4*xl^2 + 6*ka^2*rtot*w^4*xl^2 +
      18*kb*rtot*w^4*xl^2 - 13*ka*rl*w^6*xl^2 +
      36*ka*rtot*w^6*xl^2 - 13*rl*w^8*xl^2 + 36*rtot*w^8*xl^2
      )*(ka*kc*l1*rl^3*w - 6*ka*kc*l1*rl^2*rtot*w +
      ka*kb*l1*rl^3*w^3 + 3*kc*l1*rl^3*w^3 -
      6*ka*kb*l1*rl^2*rtot*w^3 - 18*kc*l1*rl^2*rtot*w^3 +
      ka^2*l1*rl^3*w^5 + 3*kb*l1*rl^3*w^5 -
      6*ka^2*l1*rl^2*rtot*w^5 - 18*kb*l1*rl^2*rtot*w^5 +
      4*ka*l1*rl^3*w^7 - 24*ka*l1*rl^2*rtot*w^7 +
      3*l1*rl^3*w^9 - 18*l1*rl^2*rtot*w^9 -
      ka*kc*rl^2*rtot*xl - kc*rl^3*w^2*xl +
      2*ka*kb*rl^2*rtot*w^2*xl + 2*kc*rl^2*rtot*w^2*xl -
      kb*rl^3*w^4*xl + 5*ka^2*rl^2*rtot*w^4*xl +
      11*kb*rl^2*rtot*w^4*xl - ka*rl^3*w^6*xl +
      28*ka*rl^2*rtot*w^6*xl - rl^3*w^8*xl +
      29*rl^2*rtot*w^8*xl + ka*kc*l1*rl*w*xl^2 -
      12*ka*kc*l1*rtot*w*xl^2 + ka*kb*l1*rl*w^3*xl^2 +
      3*kc*l1*rl*w^3*xl^2 - 12*ka*kb*l1*rtot*w^3*xl^2 -
      36*kc*l1*rtot*w^3*xl^2 + ka^2*l1*rl*w^5*xl^2 +
      3*kb*l1*rl*w^5*xl^2 - 12*ka^2*l1*rtot*w^5*xl^2 -
      36*kb*l1*rtot*w^5*xl^2 + 4*ka*l1*rl*w^7*xl^2 -
      48*ka*l1*rtot*w^7*xl^2 + 3*l1*rl*w^9*xl^2 -
      36*l1*rtot*w^9*xl^2 - kc*rl*w^2*xl^3 +
      3*ka*kb*rtot*w^2*xl^3 + 9*kc*rtot*w^2*xl^3 -
      kb*rl*w^4*xl^3 + 6*ka^2*rtot*w^4*xl^3 +
      18*kb*rtot*w^4*xl^3 - ka*rl*w^6*xl^3 +
      36*ka*rtot*w^6*xl^3 - rl*w^8*xl^3 + 36*rtot*w^8*xl^3))/
 (6*w^4*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
      2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
      6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
      12*rl*w^6 + 12*rtot*w^6)^2*xl^2) -
(rtot*(-(ka*kc*rl^4) - 4*kc*rl^4*w^2 + ka^2*rl^4*w^4 -
     kb*rl^4*w^4 + 4*ka*rl^4*w^6 + 5*rl^4*w^8 -
```

```
    6*ka*kc*l1*rl^2*w*xl - 6*ka*kb*l1*rl^2*w^3*xl -
    18*kc*l1*rl^2*w^3*xl - 6*ka^2*l1*rl^2*w^5*xl -
    18*kb*l1*rl^2*w^5*xl - 24*ka*l1*rl^2*w^7*xl -
    18*l1*rl^2*w^9*xl - ka*kc*rl^2*xl^2 +
    ka*kb*rl^2*w^2*xl^2 - kc*rl^2*w^2*xl^2 +
    3*ka^2*rl^2*w^4*xl^2 + 5*kb*rl^2*w^4*xl^2 +
    16*ka*rl^2*w^6*xl^2 + 17*rl^2*w^8*xl^2 -
    6*ka*kc*l1*w*xl^3 - 6*ka*kb*l1*w^3*xl^3 -
    18*kc*l1*w^3*xl^3 - 6*ka^2*l1*w^5*xl^3 -
    18*kb*l1*w^5*xl^3 - 24*ka*l1*w^7*xl^3 -
    18*l1*w^9*xl^3 + ka*kb*w^2*xl^4 + 3*kc*w^2*xl^4 +
    2*ka^2*w^4*xl^4 + 6*kb*w^4*xl^4 + 12*ka*w^6*xl^4 +
    12*w^8*xl^4))/
(2*w^2*(-(ka*kb*rl) - 3*kc*rl + ka*kb*rtot + 3*kc*rtot -
    2*ka^2*rl*w^2 - 6*kb*rl*w^2 + 2*ka^2*rtot*w^2 +
    6*kb*rtot*w^2 - 12*ka*rl*w^4 + 12*ka*rtot*w^4 -
    12*rl*w^6 + 12*rtot*w^6)*xl))^2)^(1/2))^(1/3)))/2)];
xppd = N[ (-((2 - (2*rl)/(rl - rtot))*xl) +
       ((2 - (2*rl)/(rl - rtot))^2*xl^2 -
          4*(rl^2 - rl^3/(rl - rtot) + xl^2 - (rl*xl^2)/
      (rl - rtot)))^(1/2))/2];
xppe = N[ (-((2 - (2*rl)/(rl - rtot))*xl) -
       ((2 - (2*rl)/(rl - rtot))^2*xl^2 -
          4*(rl^2 - rl^3/(rl - rtot) + xl^2 - (rl*xl^2)/
       (rl - rtot)))^(1/2))/2];
           xpa=Re[xppa];
           xpb=Re[xppb];
           xpc=Re[xppc];
           xpd=Re[xppd];
           xpe=Re[xppe];];


(*Calculation of values of xs*)
xscalc[xp_,xs_]:=
Module[{xss},
xss = N[(ka*kc*l1*rl^2*w + ka*kb*l1*rl^2*w^3 + 3*kc*l1*rl^2*w^3 + ka^2*l1*rl^2*w
    3*kb*l1*rl^2*w^5 + 4*ka*l1*rl^2*w^7 + 3*l1*rl^2*w^9 - kc*rl^2*w^2*xl -
kb*rl^2*w^4*xl - ka*rl^2*w^6*xl - rl^2*w^8*xl + ka*kc*l1*w*xl^2 +
ka*kb*l1*w^3*xl^2 + 3*kc*l1*w^3*xl^2 + ka^2*l1*w^5*xl^2 +
3*kb*l1*w^5*xl^2 + 4*ka*l1*w^7*xl^2 + 3*l1*w^9*xl^2 - kc*w^2*xl^3 -
kb*w^4*xl^3 - ka*w^6*xl^3 - w^8*xl^3 + ka*kc*rl^2*xp + kc*rl^2*w^2*xp -
ka^2*rl^2*w^4*xp - 2*kb*rl^2*w^4*xp - 7*ka*rl^2*w^6*xp - 8*rl^2*w^8*xp +
4*ka*kc*l1*w*xl*xp + 4*ka*kb*l1*w^3*xl*xp + 12*kc*l1*w^3*xl*xp +
4*ka^2*l1*w^5*xl*xp + 12*kb*l1*w^5*xl*xp + 16*ka*l1*w^7*xl*xp +
12*l1*w^9*xl*xp - ka*kb*w^2*xl^2*xp - 4*kc*w^2*xl^2*xp -
2*ka^2*w^4*xl^2*xp - 7*kb*w^4*xl^2*xp - 13*ka*w^6*xl^2*xp -
13*w^8*xl^2*xp - ka*kb*w^2*xl*xp^2 - 3*kc*w^2*xl*xp^2 -
2*ka^2*w^4*xl*xp^2 - 6*kb*w^4*xl*xp^2 - 12*ka*w^6*xl*xp^2 - 12*w^8*xl*xp^2
)/(-(ka*kc*rl^2) - 4*kc*rl^2*w^2 + ka^2*rl^2*w^4 - kb*rl^2*w^4 +
4*ka*rl^2*w^6 + 5*rl^2*w^8 - 6*ka*kc*l1*w*xl - 6*ka*kb*l1*w^3*xl -
18*kc*l1*w^3*xl - 6*ka^2*l1*w^5*xl - 18*kb*l1*w^5*xl - 24*ka*l1*w^7*xl -
18*l1*w^9*xl + ka*kb*w^2*xl^2 + 3*kc*w^2*xl^2 + 2*ka^2*w^4*xl^2 +
6*kb*w^4*xl^2 + 12*ka*w^6*xl^2 + 12*w^8*xl^2 - 6*ka*kc*l1*w*xp -
6*ka*kb*l1*w^3*xp - 18*kc*l1*w^3*xp - 6*ka^2*l1*w^5*xp -
18*kb*l1*w^5*xp - 24*ka*l1*w^7*xp - 18*l1*w^9*xp + 2*ka*kb*w^2*xl*xp +
6*kc*w^2*xl*xp + 4*ka^2*w^4*xl*xp + 12*kb*w^4*xl*xp + 24*ka*w^6*xl*xp +
24*w^8*xl*xp + ka*kb*w^2*xp^2 + 3*kc*w^2*xp^2 + 2*ka^2*w^4*xp^2 +
6*kb*w^4*xp^2 + 12*ka*w^6*xp^2 + 12*w^8*xp^2)]; xs=Re[xss];];


(*Calculation of lp*)
lpcalc[xp_,xs_,lpa_,lpb_,lpc_,lpd_]:=
Module[{a5,a4,a3,a2,a1,lppa,lppb,lppc,lppd},
a5 = N[(-(kb*l1^4*w*xp) + ka*l1^2*rl^2*w*xp - rl^4*w*xp - 2*ka*l1^4*w^3*xp +
  4*l1^2*rl^2*w^3*xp - 3*l1^4*w^5*xp + 2*ka*l1^3*w^2*xl*xp -
```

```
  4*l1*rl^2*w^2*xl*xp + 6*l1^3*w^4*xl*xp - 3*l1^2*w^3*xl^2*xp +
  ka*l1^3*w^2*xp^2 - 3*l1*rl^2*w^2*xp^2 + 3*l1^3*w^4*xp^2 -
  4*l1^2*w^3*xl*xp^2 - l1^2*w^3*xp^3 - kb*l1^4*w*xs + ka*l1^2*rl^2*w*xs -
  rl^4*w*xs - 2*ka*l1^4*w^3*xs + 4*l1^2*rl^2*w^3*xs - 3*l1^4*w^5*xs +
  2*ka*l1^3*w^2*xl*xs - 4*l1*rl^2*w^2*xl*xs + 6*l1^3*w^4*xl*xs -
  3*l1^2*w^3*xl^2*xs + 2*ka*l1^3*w^2*xp*xs - 4*l1*rl^2*w^2*xp*xs +
  6*l1^3*w^4*xp*xs - 6*l1^2*w^3*xl*xp*xs - 3*l1^2*w^3*xp^2*xs )] ;

a1 =N[(-(kb*w*xl) - 2*ka*w^3*xl - 3*w^5*xl - kb*w*xs - 2*ka*w^3*xs -
     3*w^5*xs)] ;
a2 =N[(-(kb*rl^2) - ka*rl^2*w^2 - 2*kb*l1*w*xl -
     4*ka*l1*w^3*xl - 6*l1*w^5*xl + ka*w^2*xl^2 + 3*w^4*xl^2 +
     2*ka*w^2*xl*xp + 6*w^4*xl*xp - 4*kb*l1*w*xs - 8*ka*l1*w^3*xs -
     12*l1*w^5*xs + 2*ka*w^2*xl*xs + 6*w^4*xl*xs + 2*ka*w^2*xp*xs +
     6*w^4*xp*xs)];

a3 =N[(-(kb*l1*rl^2) - ka*l1*rl^2*w^2 - kb*l1^2*w*xl -
     2*ka*l1^2*w^3*xl - rl^2*w^3*xl - 3*l1^2*w^5*xl + ka*l1*w^2*xl^2 +
     3*l1*w^4*xl^2 - w^3*xl^3 - kb*l1^2*w*xp + ka*rl^2*w*xp -
     2*ka*l1^2*w^3*xp + rl^2*w^3*xp - 3*l1^2*w^5*xp + 4*ka*l1*w^2*xl*xp +
     12*l1*w^4*xl*xp - 4*w^3*xl^2*xp - 3*w^3*xl*xp^2 - 6*kb*l1^2*w*xs +
     ka*rl^2*w*xs - 12*ka*l1^2*w^3*xs + 4*rl^2*w^3*xs - 18*l1^2*w^5*xs +
     6*ka*l1*w^2*xl*xs + 18*l1*w^4*xl*xs - 3*w^3*xl^2*xs +
     6*ka*l1*w^2*xp*xs + 18*l1*w^4*xp*xs - 6*w^3*xl*xp*xs -
     3*w^3*xp^2*xs)];

a4 = N[(-2*kb*l1^3*w*xp + 2*ka*l1*rl^2*w*xp - 4*ka*l1^3*w^3*xp +
     6*l1*rl^2*w^3*xp - 6*l1^3*w^5*xp + 4*ka*l1^2*w^2*xl*xp -
     2*rl^2*w^2*xl*xp + 12*l1^2*w^4*xl*xp - 4*l1*w^3*xl^2*xp +
     ka*l1^2*w^2*xp^2 - rl^2*w^2*xp^2 + 3*l1^2*w^4*xp^2 -
     4*l1*w^3*xl*xp^2 -
     4*kb*l1^3*w*xs + 2*ka*l1*rl^2*w*xs - 8*ka*l1^3*w^3*xs +
     8*l1*rl^2*w^3*xs - 12*l1^3*w^5*xs + 6*ka*l1^2*w^2*xl*xs -
     4*rl^2*w^2*xl*xs + 18*l1^2*w^4*xl*xs - 6*l1*w^3*xl^2*xs +
     6*ka*l1^2*w^2*xp*xs - 4*rl^2*w^2*xp*xs + 18*l1^2*w^4*xp*xs -
     12*l1*w^3*xl*xp*xs - 6*l1*w^3*xp^2*xs)];

lppa= N[((-a2/(2*a1) + (-(a4/a1) +
          (a2*(a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1)))/
               (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                  (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                  ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
                (1/3) + (a3^3/(27*a1^3) -
                 (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                 (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                 ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
                (1/3)))/(2*a1))/
          (2*(-(a5/a1) + (a3/(3*a1) -
                   (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1)))/
                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                       ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                          (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                             (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
           )^(1/3) + (a3^3/(27*a1^3) -
          (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
          (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
          ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
             (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))
^(1/3))^2/4)^(1/2)) +
```

```
((a2/(2*a1) - (-(a4/a1) +
          (a2*(a3/(3*a1) -
               (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                   ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                     (a3^3/(27*a1^3) -
                     (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                     (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                   )^(1/3) +
               (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                  (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                  ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                     (a3^3/(27*a1^3) -
                     (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                     (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                  )^(1/3)))/(2*a1))/
        (2*(-(a5/a1) + (a3/(3*a1) -
                (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                 (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                    (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                    ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                    (a3^3/(27*a1^3) -
                    (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                    (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                    )^(1/3) +
                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                   ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                    (a3^3/(27*a1^3) -
                    (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                    (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                   )^(1/3))^2/4)^(1/2)))^2 -
  4*((a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
        (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
           (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
           ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
            (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
              (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))
    ^(1/3) + (a3^3/(27*a1^3) -
    (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
    (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
    ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
      (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
 (1/3))/2 - (-(a5/a1) +
(a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
        ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
          (a3^3/(27*a1^3) -
          (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
          (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
        )^(1/3) +
    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
       ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
         (a3^3/(27*a1^3) -
         (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
       )^(1/3))^2/4)^(1/2)))^(1/2))/2)];
lppb = N[((-a2/(2*a1) + (-(a4/a1) +
          (a2*(a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
               (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                  (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                  ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
```

```
                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
              (1/3) + (a3^3/(27*a1^3) -
               (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
               (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
               ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                  (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                     (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
            (1/3)))/(2*a1))/
(2*(-(a5/a1) + (a3/(3*a1) -
          (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
           (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
              (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
              ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                 (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                    (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
              )^(1/3) + (a3^3/(27*a1^3) -
             (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
             (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
             ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))
             ^(1/3))^2/4)^(1/2)) -
    ((a2/(2*a1) - (-(a4/a1) +
                (a2*(a3/(3*a1) -
                   (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                       ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                         (a3^3/(27*a1^3) -
                         (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                       )^(1/3) +
                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                       ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                         (a3^3/(27*a1^3) -
                         (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                     )^(1/3)))/(2*a1))/
          (2*(-(a5/a1) + (a3/(3*a1) -
                    (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                        ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                        (a3^3/(27*a1^3) -
                        (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                        )^(1/3) +
                     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                        ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                         (a3^3/(27*a1^3) -
                         (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                       )^(1/3))^2/4)^(1/2)))^2 -
4*((a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
      (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
      ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
        (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
           (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))
^(1/3) + (a3^3/(27*a1^3) -
(a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
(-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
```

```
                                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
                             (1/3))/2 - (-(a5/a1) +
                           (a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                                   ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                                      (a3^3/(27*a1^3) -
                                      (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                      (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                                   )^(1/3) +
                                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                                   ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                                      (a3^3/(27*a1^3) -
                                      (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                      (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                                  )^(1/3))^2/4)^(1/2)))^(1/2))/2) ];
lppc = N[((-a2/(2*a1) - (-(a4/a1) +
                          (a2*(a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                               (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                  (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                                  ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                                     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))1^2)^(1/2))^
                            (1/3) + (a3^3/(27*a1^3) -
                             (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                             (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                             ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
                          (1/3)))/(2*a1))/
                    (2*(-(a5/a1) + (a3/(3*a1) -
                             (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                              (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                 (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                                 ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                                )^(1/3) + (a3^3/(27*a1^3) -
                               (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                               (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                               ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                                  (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                     (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))
                            ^(1/3))^2/4)^(1/2)) +
                  ((a2/(2*a1) + (-(a4/a1) +
                           (a2*(a3/(3*a1) -
                                (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                                 (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                    (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                                    ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                                      (a3^3/(27*a1^3) -
                                      (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                      (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                                    )^(1/3) +
                                 (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                    (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                                    ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                                       (a3^3/(27*a1^3) -
                                       (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                                   )^(1/3)))/(2*a1))/
                          (2*(-(a5/a1) + (a3/(3*a1) -
                                   (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
```

```
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                        ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                        (a3^3/(27*a1^3) -
                        (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                        )^(1/3) +
                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                       ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                        (a3^3/(27*a1^3) -
                        (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                       )^(1/3))^2/4)^(1/2)))^2 -
          4*((a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                   ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                      (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))
                  ^(1/3) + (a3^3/(27*a1^3) -
                  (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                  (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                  ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                  )^(1/3) +
                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                   ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                      (a3^3/(27*a1^3) -
                      (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                      (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                   )^(1/3))^2/4)^(1/2)))^(1/2))/2)];
lppd= N[((-a2/(2*a1) - (-(a4/a1) +
          (a2*(a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                   ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                      (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
                 (1/3) + (a3^3/(27*a1^3) -
                  (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                  (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                  ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
                (1/3)))/(2*a1))/
           (2*(-(a5/a1) + (a3/(3*a1) -
                 (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                  (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                  (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                  ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                  )^(1/3) + (a3^3/(27*a1^3) -
                 (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                 (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                 ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
```

```
                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))
                ^(1/3))^2/4)^(1/2)) -
((a2/(2*a1) + (-(a4/a1) +
          (a2*(a3/(3*a1) -
               (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                   (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                   ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                     (a3^3/(27*a1^3) -
                     (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                     (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                   )^(1/3) +
               (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                  (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                  ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                     (a3^3/(27*a1^3) -
                     (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                     (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                  )^(1/3)))/(2*a1))/
           (2*(-(a5/a1) + (a3/(3*a1) -
                    (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
                     (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                        ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                        (a3^3/(27*a1^3) -
                        (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                        )^(1/3) +
                    (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                       (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
                       ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
                        (a3^3/(27*a1^3) -
                        (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
                        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
                       )^(1/3))^2/4)^(1/2))}^2 -
  4*((a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
       (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
        (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
        ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
          (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
            (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))
     ^(1/3) + (a3^3/(27*a1^3) -
     (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
     (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
     ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
       (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2))^
   (1/3))/2 + (-(a5/a1) +
  (a3/(3*a1) - (-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))/
      (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
         ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
           (a3^3/(27*a1^3) -
           (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
           (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
         )^(1/3) +
      (a3^3/(27*a1^3) - (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
         (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2) +
         ((-a3^2/(9*a1^2) + ((a2*a4)/a1 - 4*a5)/(3*a1))^3 +
           (a3^3/(27*a1^3) -
            (a3*((a2*a4)/a1 - 4*a5))/(6*a1^2) -
            (-a4^2 - (a2^2*a5)/a1 + 4*a3*a5)/(2*a1^2))^2)^(1/2)
         )^(1/3))^2/4)^(1/2))]^(1/2))/2)];

lpa=Re[lppa];
```

```
        lpb=Re[lppb];
        lpc=Re[lppc];
        lpd=Re[lppd];];


(*Calculation of ls*)
lscalc[xp_,xs_,lp_,ls__]:=
Module[{lss},

lss = N[  ((    ka*lp*ll*ll + ka*lp*lp*ll
                 + (lp*lp+2*lp*ll)*(w*w*ll - w*xl)
                 + (ll*ll+2*lp*ll)*(w*w*lp - w*xp)
                 -   w*xs*(ll*ll + 2*ll*lp + lp*lp)-rl*rl*lp )/
               (  - 1*ka*ll*ll - ka*lp*lp - 2*ka*lp*ll
                  - 2*(ll + lp)*(w*w*ll - w*xl)
                  - 2*(ll + lp)*(w*w*lp - w*xp)
                  - w*w*(ll*ll+2*ll*lp+lp*lp) + rl*rl))];
      ls=Re[lss];];


(* Calculation of cs,cp*)
cspcalc[xp_,xs_,lp_,ls_,cp_,cs_]:=
Module[{},
cs=N[1/(w*w*ls-w*xs)];
cp=N[1/(w*w*lp-w*xp)];
];


(* Calculation of cl*)
clcalc[xl_,ll_,cl_]:=
Module[{},
cl=N[1/(w*w*ll-w*xl)];
];

(* Results written to output file*)
wresult:=
Module[{},

sresult = OpenAppend["result1", FormatType ->OutputForm];

Write[sresult,"D(",j,",1)=  ",xl,";"];
Write[sresult,"D(",j,",2)=  ",ll*10^6,"E-6;"];
Write[sresult,"D(",j,",3)=  ", f,";"];
Write[sresult,"D(",j,",4)=  ", f1,";"];
Write[sresult,"D(",j,",5)=  ", f2,";"];
Write[sresult,"D(",j,",6)=  ",rl,";"];
Write[sresult,"D(",j,",7)=  ",rtot,";"];
Write[sresult,"D(",j,",8)=  ",cl*10^6,"E-6;"];

(*Write[sresult,"D(",j,",9)=  ", ka*10^-11,"E+11;"];
Write[sresult,"D(",j,",10)=  ", kb*10^-22,"E+22;"];
Write[sresult,"D(",j,",11)=  ", kc*10^-32,"E+32;"];*)
Write[sresult,"                                  "];
Write[sresult,"    %  First set of solutions  % "];
Write[sresult,"D(",j,",12)=  ",xpa,";"];
Write[sresult,"D(",j,",13)=  ",xsa,";"];
Write[sresult,"D(",j,",14)=  ",lpaa*10^6,"E-6;"];
Write[sresult,"D(",j,",15)=  ",cpaa*10^6,"E-6;"];
Write[sresult,"D(",j,",16)=  ",lsaa*10^6,"E-6;"];
Write[sresult,"D(",j,",17)=  ",csaa*10^6,"E-6;"];
Write[sresult,"D(",j,",18)=  ",lpba*10^6,"E-6;"];
Write[sresult,"D(",j,",19)=  ",cpba*10^6,"E-6;"];
Write[sresult,"D(",j,",20)=  ",lsba*10^6,"E-6;"];
Write[sresult,"D(",j,",21)=  ",csba*10^6,"E-6;"];
```

```
Write[sresult,"D(",j,",22)=  ",lpca*10^6,"E-6;"];
Write[sresult,"D(",j,",23)=  ",cpca*10^6,"E-6;"];
Write[sresult,"D(",j,",24)=  ",lsca*10^6,"E-6;"];
Write[sresult,"D(",j,",25)=  ",csca*10^6,"E-6;"];
Write[sresult,"D(",j,",26)=  ",lpda*10^6,"E-6;"];
Write[sresult,"D(",j,",27)=  ",cpda*10^6,"E-6;"];
Write[sresult,"D(",j,",28)=  ",lsda*10^6,"E-6;"];
Write[sresult,"D(",j,",29)=  ",csda*10^6,"E-6;"];

Write[sresult,"  %  Second set of solutions  %  "];
Write[sresult,"D(",j,",30)=  ",xpb,";"];
Write[sresult,"D(",j,",31)=  ",xsb*10^13,"E-13;"];
Write[sresult,"D(",j,",32)=  ",lpab*10^6,"E-6;"];
Write[sresult,"D(",j,",33)=  ",cpab*10^6,"E-6;"];
Write[sresult,"D(",j,",34)=  ",lsab*10^6,"E-6;"];
Write[sresult,"D(",j,",35)=  ",csab*10^6,"E-6;"];
Write[sresult,"D(",j,",36)=  ",lpbb*10^6,"E-6;"];
Write[sresult,"D(",j,",37)=  ",cpbb*10^6,"E-6;"];
Write[sresult,"D(",j,",38)=  ",lsbb*10^6,"E-6;"];
Write[sresult,"D(",j,",39)=  ",csbb*10^6,"E-6;"];
Write[sresult,"D(",j,",40)=  ",lpcb*10^6,"E-6;"];
Write[sresult,"D(",j,",41)=  ",cpcb*10^6,"E-6;"];
Write[sresult,"D(",j,",42)=  ",lscb*10^6,"E-6;"];
Write[sresult,"D(",j,",43)=  ",cscb*10^6,"E-6;"];
Write[sresult,"D(",j,",44)=  ",lpdb*10^6,"E-6;"];
Write[sresult,"D(",j,",45)=  ",cpdb*10^6,"E-6;"];
Write[sresult,"D(",j,",46)=  ",lsdb*10^6,"E-6;"];
Write[sresult,"D(",j,",47)=  ",csdb*10^6,"E-6;"];
Write[sresult,"  % Third set of solutions  %   "];
Write[sresult,"D(",j,",48)=  ",xpc,";"];
Write[sresult,"D(",j,",49)=  ",xsc,";"];
Write[sresult,"D(",j,",50)=  ",lpac*10^6,"E-6;"];
Write[sresult,"D(",j,",51)=  ",cpac*10^6,"E-6;"];
Write[sresult,"D(",j,",52)=  ",lsac*10^6,"E-6;"];
Write[sresult,"D(",j,",53)=  ",csac*10^6,"E-6;"];
Write[sresult,"D(",j,",54)=  ",lpbc*10^6,"E-6;"];
Write[sresult,"D(",j,",55)=  ",cpbc*10^6,"E-6;"];
Write[sresult,"D(",j,",56)=  ",lsbc*10^6,"E-6;"];
Write[sresult,"D(",j,",57)=  ",csbc*10^6,"E-6;"];
Write[sresult,"D(",j,",58)=  ",lpcc*10^6,"E-6;"];
Write[sresult,"D(",j,",59)=  ",cpcc*10^6,"E-6;"];
Write[sresult,"D(",j,",60)=  ",lscc*10^6,"E-6;"];
Write[sresult,"D(",j,",61)=  ",cscc*10^6,"E-6;"];
Write[sresult,"D(",j,",62)=  ",lpdc*10^6,"E-6;"];
Write[sresult,"D(",j,",63)=  ",cpdc*10^6,"E-6;"];
Write[sresult,"D(",j,",64)=  ",lsdc*10^6,"E-6;"];
Write[sresult,"D(",j,",65)=  ",csdc*10^6,"E-6;"];
j++;
Close[sresult];];

(* Order of calculations calling previous modules *)
increment:=
Module[{},
scale1=scale;
kc1=kcalc;
xp1=xpcalc;
xs1=xscalc[xpa,xsa];
xs2=xscalc[xpb,xsb];
xs3=xscalc[xpc,xsc];
(*xsd=N[-xl];
xse=N[-xl];*)
lp1 = lpcalc[xpa,xsa,lpaa, lpba, lpca, lpda];
lp2 = lpcalc[xpb,xsb,lpab, lpbb, lpcb, lpdb];
lp3 = lpcalc[xpc,xsc,lpac, lpbc, lpcc, lpdc];
(*lp4 = lpcalc[xpd,xsd,lpad, lpbd, lpcd, lpdd];
lp5 = lpcalc[xpe,xse,lpae, lpbe, lpce, lpde];*)
```

```
ls1=lscalc[xpa,xsa,lpaa,lsaa];
ls2=lscalc[xpa,xsa,lpba,lsba];
ls3=lscalc[xpa,xsa,lpca,lsca];
ls4=lscalc[xpa,xsa,lpda,lsda];
ls5=lscalc[xpb,xsb,lpab,lsab];
ls6=lscalc[xpb,xsb,lpbb,lsbb];
ls7=lscalc[xpb,xsb,lpcb,lscb];
ls8=lscalc[xpb,xsb,lpdb,lsdb];
ls9=lscalc[xpc,xsc,lpac,lsac];
ls10=lscalc[xpc,xsc,lpbc,lsbc];
ls11=lscalc[xpc,xsc,lpcc,lscc];
ls12=lscalc[xpc,xsc,lpdc,lsdc];
(*ls13=lscalc[xpd,xsd,lpad,lsad];
ls14=lscalc[xpd,xsd,lpbd,lsbd];
ls15=lscalc[xpd,xsd,lpcd,lscd];
ls16=lscalc[xpd,xsd,lpdd,lsdd];
ls17=lscalc[xpe,xse,lpae,lsae];
ls18=lscalc[xpe,xse,lpbe,lsbe];
ls19=lscalc[xpe,xse,lpce,lsce];
ls20=lscalc[xpe,xse,lpde,lsde];*)
cv1=cspcalc[xpa,xsa,lpaa,lsaa,cpaa,csaa];
cv2=cspcalc[xpa,xsa,lpba,lsba,cpba,csba];
cv3=cspcalc[xpa,xsa,lpca,lsca,cpca,csca];
cv4=cspcalc[xpa,xsa,lpda,lsda,cpda,csda];
cv5=cspcalc[xpb,xsb,lpab,lsab,cpab,csab];
cv6=cspcalc[xpb,xsb,lpbb,lsbb,cpbb,csbb];
cv7=cspcalc[xpb,xsb,lpcb,lscb,cpcb,cscb];
cv8=cspcalc[xpb,xsb,lpdb,lsdb,cpdb,csdb];
cv9=cspcalc[xpc,xsc,lpac,lsac,cpac,csac];
cv10=cspcalc[xpc,xsc,lpbc,lsbc,cpbc,csbc];
cv11=cspcalc[xpc,xsc,lpcc,lscc,cpcc,cscc];
cv12=cspcalc[xpc,xsc,lpdc,lsdc,cpdc,csdc];
(*cv13=cspcalc[xpd,xsd,lpad,lsad,cpad,csad];
cv14=cspcalc[xpd,xsd,lpbd,lsbd,cpbd,csbd];
cv15=cspcalc[xpd,xsd,lpcd,lscd,cpcd,cscd];
cv16=cspcalc[xpd,xsd,lpdd,lsdd,cpdd,csdd];
cv17=cspcalc[xpe,xse,lpae,lsae,cpae,csae];
cv18=cspcalc[xpe,xse,lpbe,lsbe,cpbe,csbe];
cv19=cspcalc[xpe,xse,lpce,lsce,cpce,csce];
cv20=cspcalc[xpe,xse,lpde,lsde,cpde,csde];*)
cv21=clcalc[xl,ll,cl];
report=wresult;
ClearAll[xpa, xpb,xpc,xsa,xsb,xsc,lpaa, lsaa, cpaa, csaa, lpba, lsba, cpba,
csba, lpca, lsca, cpca, csca, lpda, lsda, cpda, csda, lpab, lsab, cpab, csab,
lpbb, lsbb, cpbb, csbb, lpcb, lscb, cpcb, cscb, lpdb,lsdb,cpdb,csdb,
lpac,lsac,cpac,csac,xpc,xsc,lpbc,lsbc,cpbc,csbc,lpcc,lscc,cpcc,cscc,
lpdc,lsdc,cpdc,csdc,cl];];
Do [increment, {nxl,1,nnxl+1},{nll,1,nnll},{nrl,1,nnrl},
                {nrtot,1,nnrtot},{nf,1,nnf},{nf1,1,nnf1},{nf2,1,nnf2}];
```

APPENDIX C MAST FILES FOR SABER SIMULATION

```
template hpigbt p m cntl = ron, roff, von, voff
electrical p, m
state logic_4 cntl
number ron=0.0291, roff=1meg, von=1.4, voff=0
{
state r r
state v v
when (event_on(cntl)) {
     if (cntl ==14_1) {
                      r = ron
                      v = von
                      }
     else             {
                      r = roff
                      v = voff
                      }
     schedule_next_time(time)
     }
equations {
     i(p->m) += (v(p) - v(m) - v)/r
     }
}     ,
```

C.1 MAST file for IGBT model

```
template feedback in ct cb cp clk drive
state logic_4 in, ct, cb, cp, clk, drive
{
state logic_4 set
when (dc_init){
     schedule_event (time, cb, 14_0)
     schedule_event (time, ct, 14_0)
     schedule_event (time, set, 14_0)
     }
when ( event_on (clk)) {

   if ((in == 14_0) & (ct == 14_0) & (cb == 14_0) &
          (set == 14_0)) {
                schedule_event (time, set, 14_1)
                schedule_event (time, ct, 14_0)
                schedule_event (time, cb, 14_1)
                schedule_event (time+1u, set, 14_0)
                }
   if ((in == 14_0) & (ct == 14_0) & (cb == 14_1) &
        (drive == 14_0) & (set == 14_0)) {
                schedule_event (time, set, 14_1)
                schedule_event (time, ct, 14_1)
                schedule_event (time, cb, 14_0)
                schedule_event (time+1u, set, 14_0)
                }
   if ((in == 14_0) & (ct == 14_1) & (cb == 14_0) &
        (drive == 14_1) & (set == 14_0)) {
                schedule_event (time, ct, 14_0)
                schedule_event (time, cb, 14_0)
                }
   if ((in == 14_1) & (ct == 14_0) & (cb == 14_0) &
          (set == 14_0)) {
                schedule_event (time, set, 14_1)
                schedule_event (time, ct, 14_0)
                schedule_event (time, cb, 14_1)
                schedule_event (time+1u, set, 14_0)
                }
   if ((in == 14_1) & (ct == 14_0) & (cb == 14_1) &
        (cp == 14_0) & (set == 14_0)) {
                schedule_event (time, set, 14_1)
                schedule_event (time, ct, 14_1)
                schedule_event (time, cb, 14_0)
                schedule_event (time+1u, set, 14_0)
                }
   if ((in == 14_1) & (ct == 14_1) & (cb == 14_0) &
        (cp == 14_1) & (set == 14_0)) {
                schedule_event (time, ct, 14_0)
                schedule_event (time, cb, 14_0)
                }
          }
}
```

C.2 MAST file for power switch controller

```
template comp ci cp
ref i ci
state logic_4 cp
{
state nu before,after
when (dc_init){
      schedule_event (time, cp, 14_0)
      }
when (threshold (ci, 0, before, after)) {
       if (after >= 0) {schedule_event (time, cp, 14_1)
                        }
       else            {schedule_event (time, cp, 14_0)
                        }
                    }
}
```

C.3 MAST file for comparator

```
template phase cp ph change = advance1, advance2

state logic_4 cp, ph, change
number advance1=1
number advance2=1

{
state time cold=0, cnew
state nu advance
when ( event_on (cp)){
            cold = cnew
            cnew = time

           if (change == 14_0){
            advance = advance1
                  }
           else advance = advance2

           if (cp == 14_1) {
              schedule_event(time+((cnew-cold)*advance), ph, 14_0)
                    }
           if (cp == 14_0){
               schedule_event(time+((cnew-cold)*advance), ph, 14_1)
                     }

}

}
```

C.4 MAST file for phase-advance circuit

APPENDIX D BUCK-BOOST CONVERTER

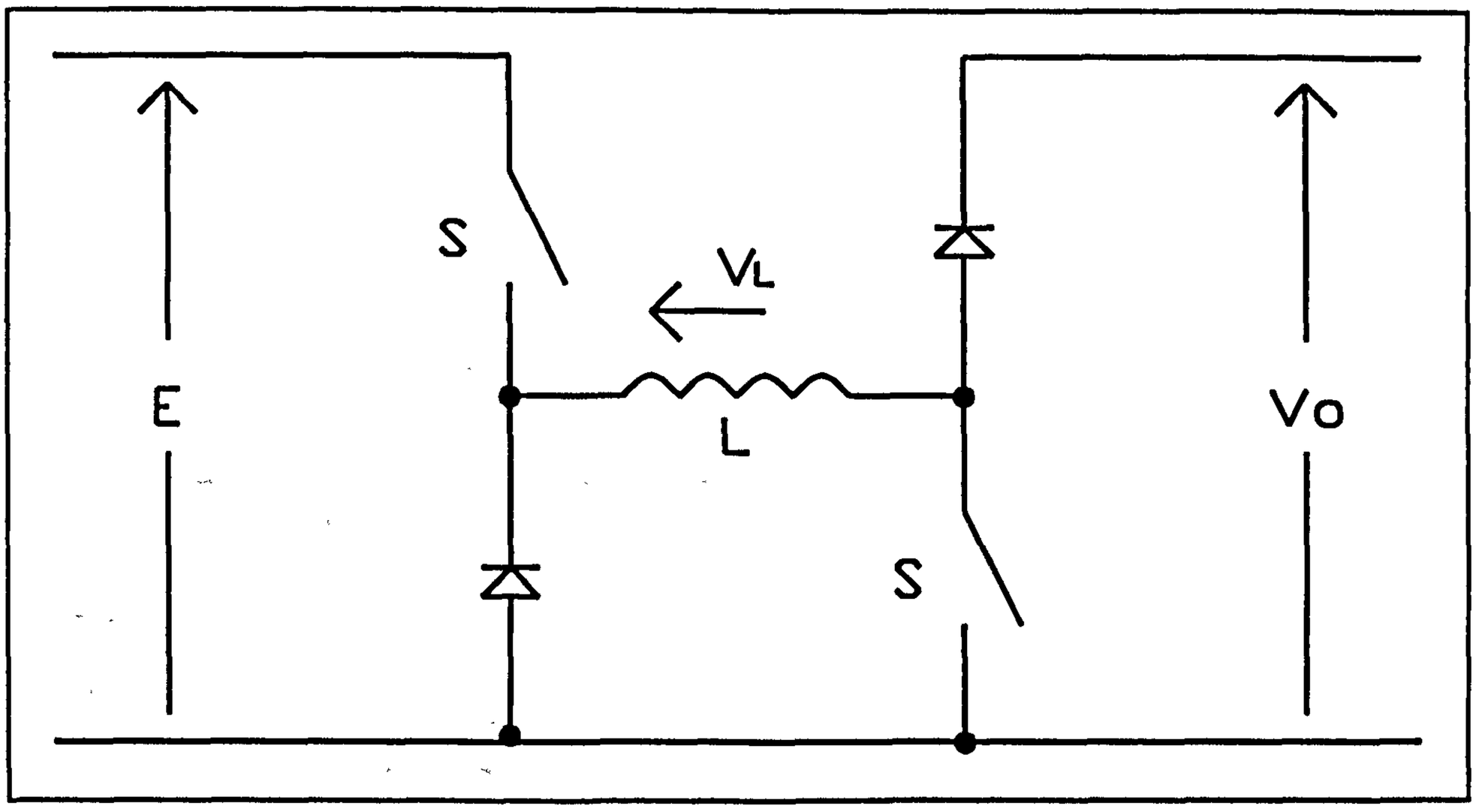


Figure D.1 Buck-Boost converter circuit

Figure D.1 shows a buck-boost converter in which the output voltage, V_o, is in the same sense as the input voltage, E. The switches, S, are operated concurrently with a duty cycle D. The inductor has a value L and the circuit is connected to a load with resistance R.

The circuit can be operated in two modes continuous-current mode or discontinuous-current mode. The current mode is discontinuous if the inductor current falls to zero before the switches are turned 'on' again. The current and voltage waveforms are shown in Figure D.2.

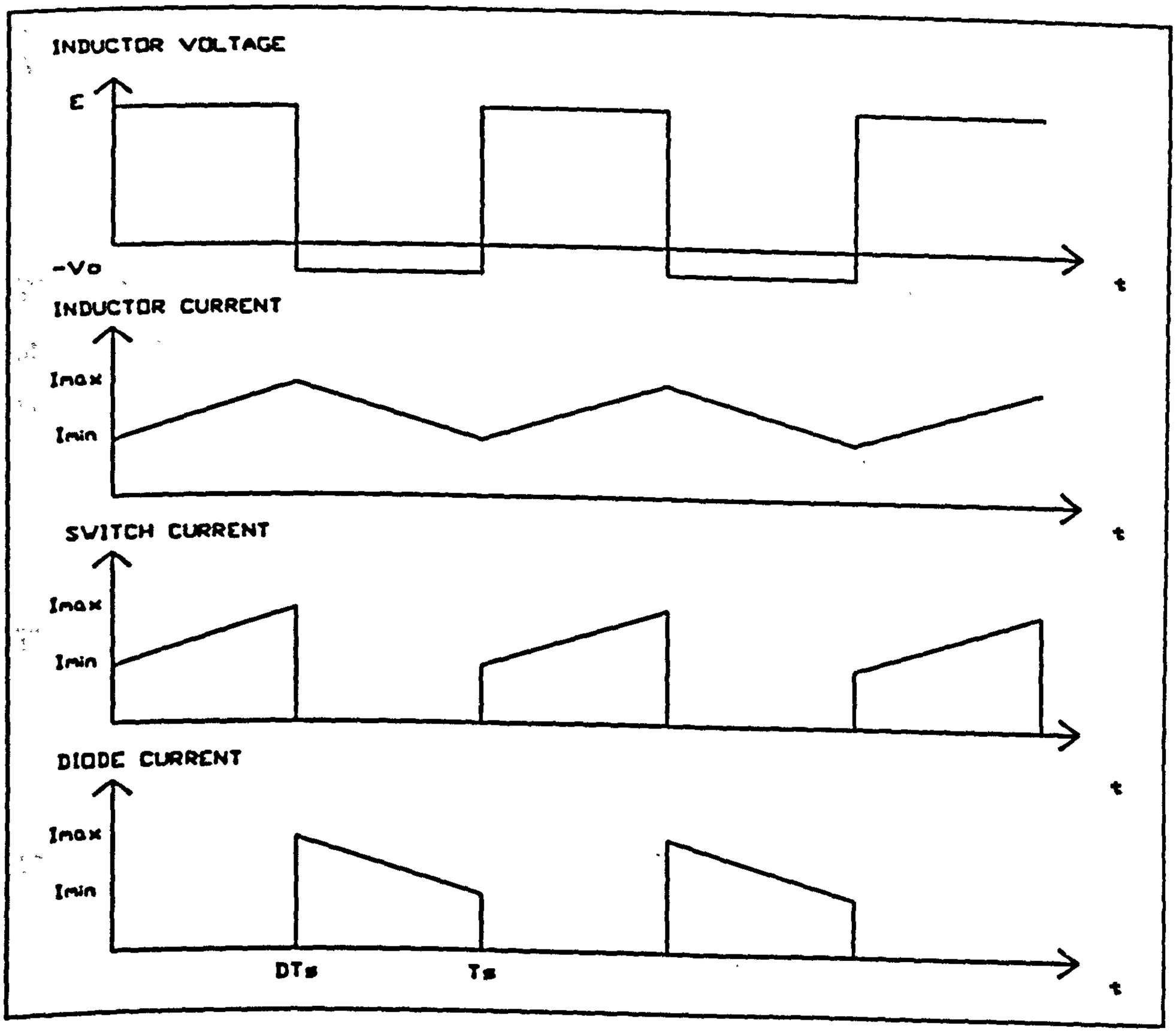


Figure D.2 Waveforms for step-up/down converter

Analysis of buck-boost converter working in continuous-inductor current mode.

When the switches, S, are closed current flows through the switches and the inductor. The voltage equation is

$$V_L = E \quad and \quad V_L = L\frac{di_L}{dt} \quad . \tag{D.1}$$

Rearranging equations (D.1) results in

$$\frac{di_L}{dt} = \frac{E}{L} \quad . \tag{D.2}$$

The inductor current increases from I_{min} to I_{max} during the time the switches are 'on' therefore

$$I_{max}-I_{min} = \frac{E}{L} \quad . \tag{D.3}$$

When the switches are opened current continues to flow in the inductor (as current in an inductor cannot change instantaneously) and the diodes come into conduction. The voltage equation is

$$V_L = - V_O \quad . \tag{D.4}$$

Therefore

$$\frac{di_L}{dt} = - \frac{V_o}{L} \tag{D.5}$$

and

$$I_{min}-I_{max} = \frac{V_o}{L} (1 - D)T_S \quad . \tag{D.6}$$

Combining equations (D.3) and (D.6) the ratio of output voltage to input voltage is given by

$$\frac{V_O}{E} = \frac{D}{1 - D} \quad . \tag{D.7}$$

For values of D < 0.5, the output voltage is less than the input voltage.

For values of D > 0.5, the output voltage is greater than the input voltage.

Analysis of converter working in discontinuous conduction mode.

If the current in the inductor falls to zero, at a time $D_2 T_s$ during each cycle before the switches are turned on again at T_s, the converter is operating in a discontinuous current mode.

When the switch is closed, $I_{min} = 0$ in equation (D.3)

$$I_{max} = \frac{EDT_S}{L} \tag{D.8}$$

and when the switch is open, $I_{min} = 0$ in equation (D.6)

$$I_{max} = \frac{V_o}{L}(D_2 - D)T_S \quad . \tag{D.9}$$

The time taken for the current to fall from I_{max} to 0 is $(D_2 - D)\ T_s$.

The average input power during the interval the switch is closed is the product of the input voltage, the average input current and the fraction of time when current is drawn from the supply, i.e.

$$\frac{E\ I_{max}\ D}{2} \quad . \tag{D.10}$$

The average output power, P_{av} (assuming V_O is constant) is

$$P_{av} = \frac{V_o^2}{R} \quad . \tag{D.11}$$

The power equation is therefore

$$\frac{E\ I_{max}\ D}{2} = \frac{V_0^2}{R} = P_{av} \quad . \tag{D.12}$$

The equations (D.8) and (D.12) are combined to give an expression for V_O/E when the inductor current is discontinuous which is

$$\frac{V_O}{E} = \sqrt{\frac{D^2RT_S}{2\ L}} \quad . \tag{D.13}$$

Condition for boundary between continuous and discontinuous operation.

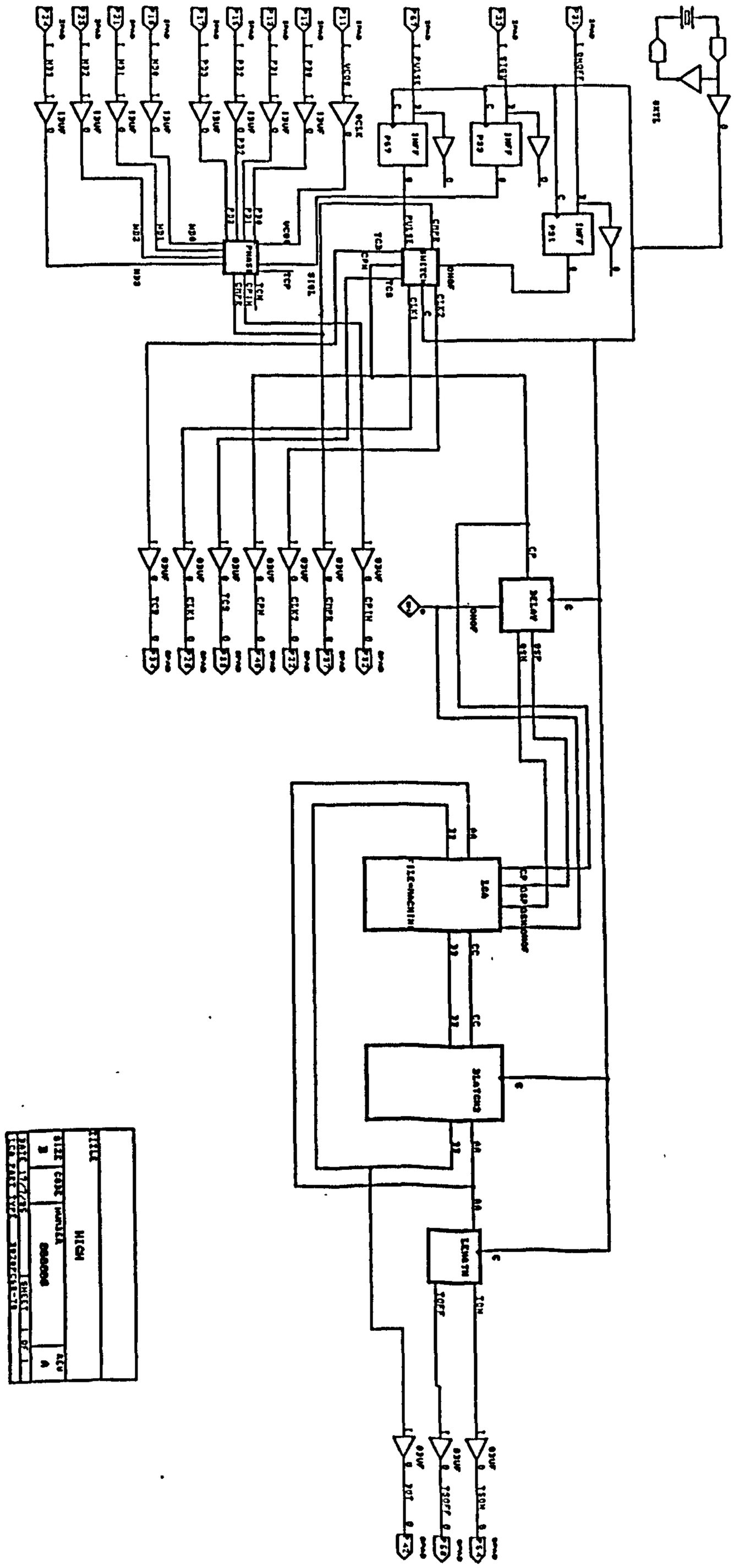
At the boundary between continuous and *discontinuous operation* $D_2 = 1$ but I_{min} will still be zero and so the equations for discontinuous operation still hold. Combining equations (D.8) and (D.12) and eliminating I_{max}

$$\frac{E}{2}\left(\frac{EDT_s}{L}\right)D = P_{av} \quad . \tag{D.14}$$

Rearranging to find the value of L at the boundary

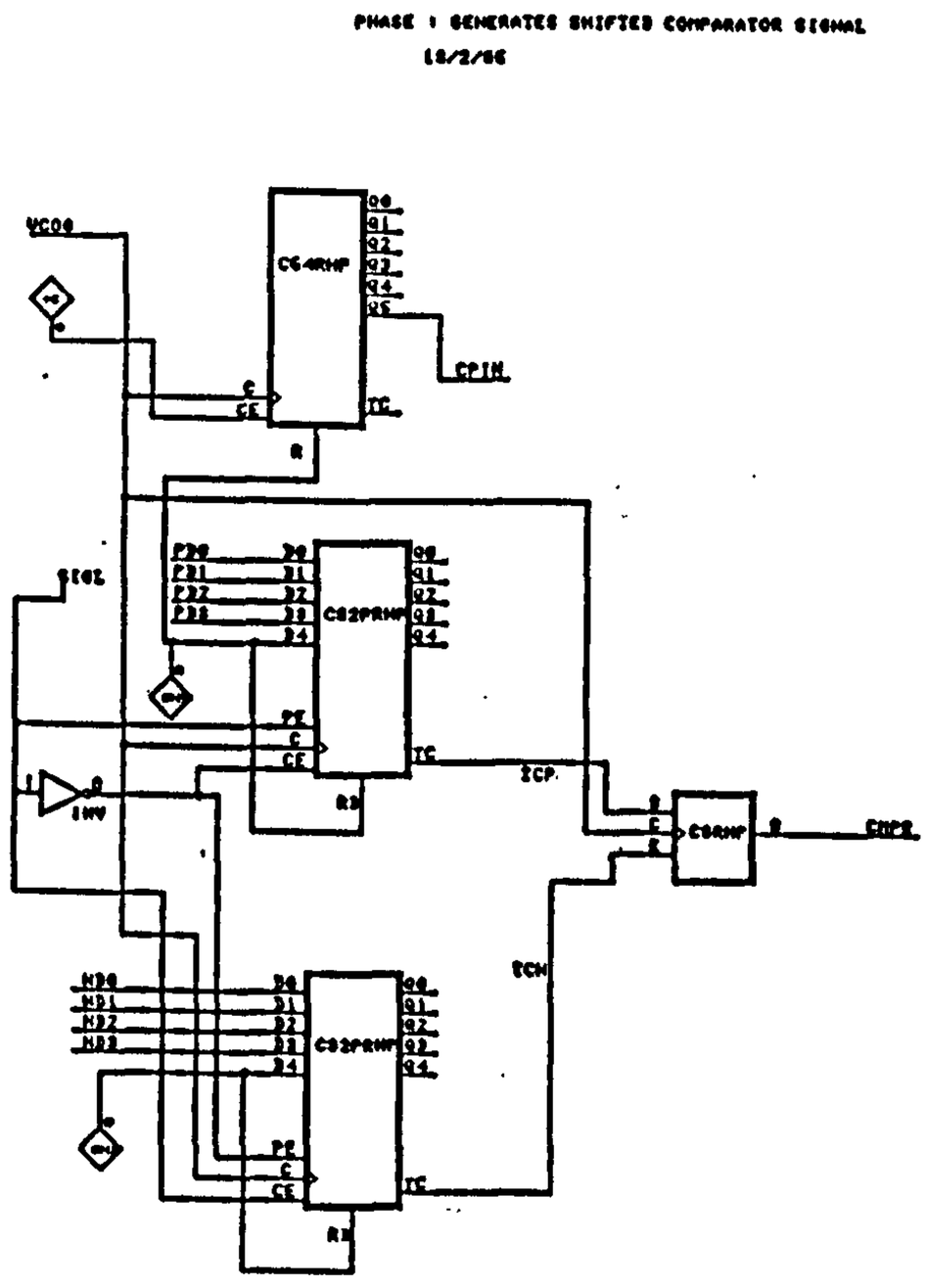
$$L_{boundary} = \left(\frac{E^2D^2T_s}{2}\right)P_{av} \quad . \tag{D.15}$$

An inductor value less than this value of $L_{boundary}$ will ensure that the converter operates in discontinuous mode and when fed from a rectified a.c. input will draw approximately unity power factor.



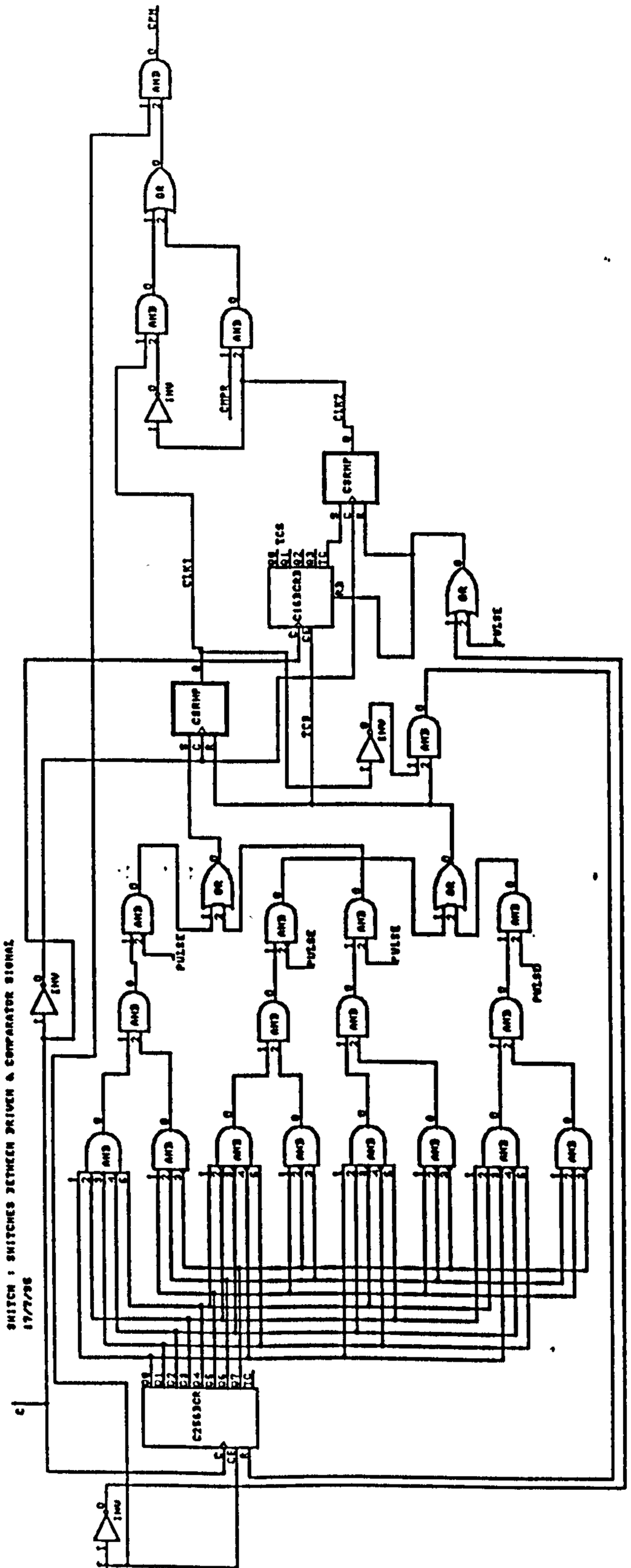
E.1 Controller of resonant converter

(a) Top-level Xilinx diagram 'HIGH'



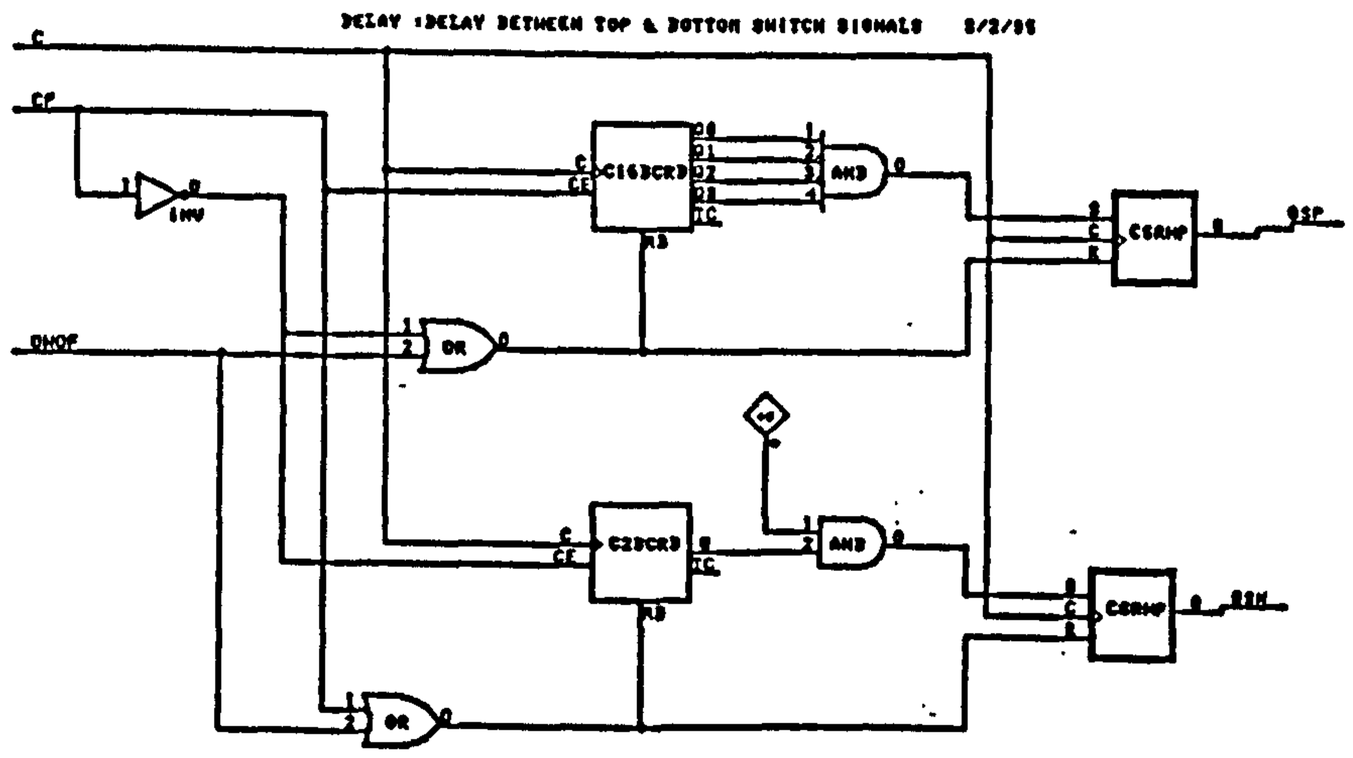
(b) Sub-level diagram of block 'PHASE'.

Generates advanced comparator signal.



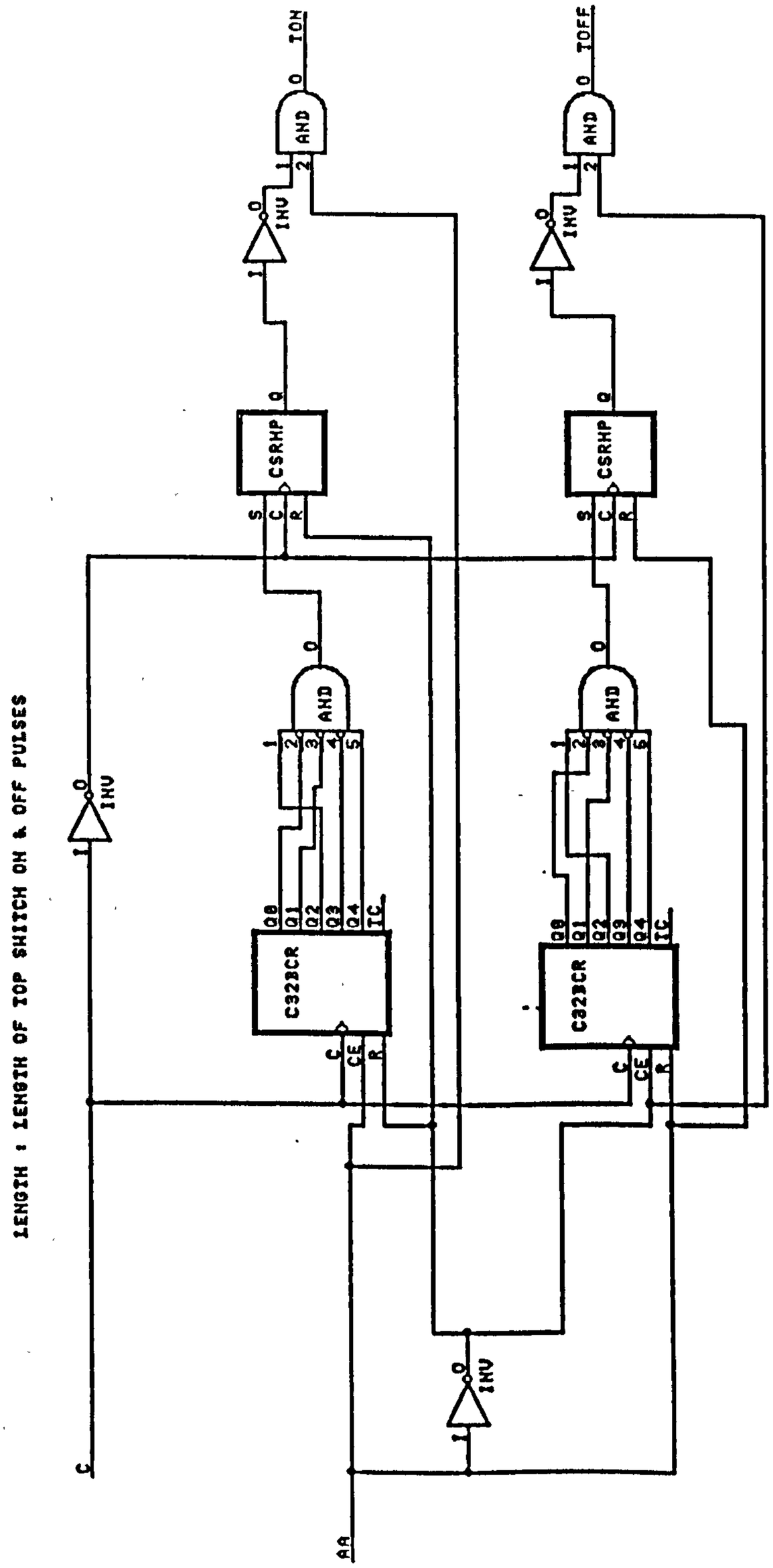
(c) Sub-level diagram of block 'SWITCH'.

Generates start-up comparator signal and switches in advanced comparator signal.



(d) Sub-level diagram of block 'DELAY'.

Introduces a time-delay between the top switch and bottom switch gate signals to prevent shoot-through.



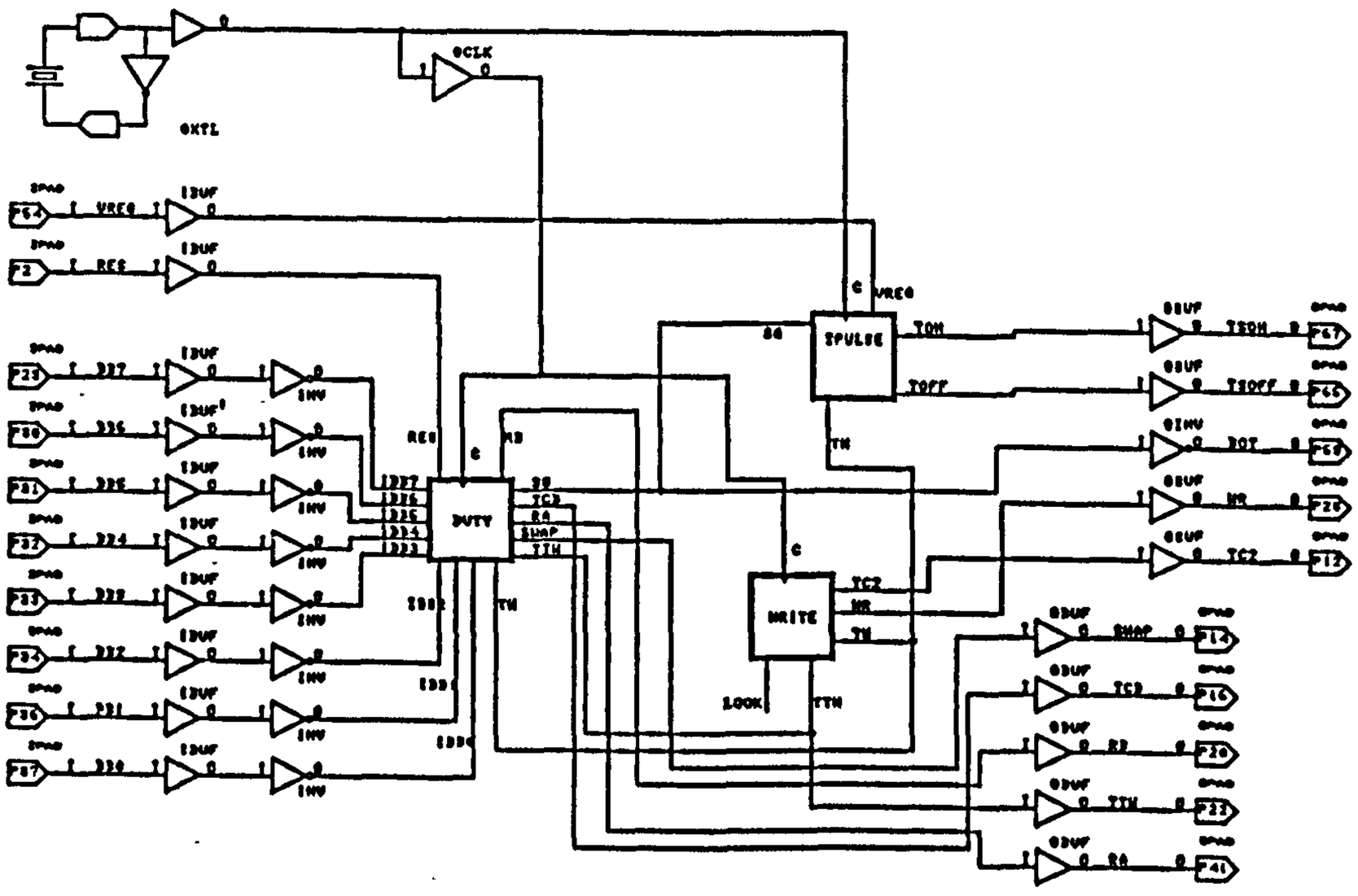
(e) Sub-level diagram of block 'LENGTH'.

Generates the 'on' and 'off' pulses required to operate the pulse-transformer gate drive circuit.

```
TITLE MACHINE.PDS
AUTHOR H.POLLOCK
DATE 2ND MARCH 1994
CHIP MACHINE LCA
;Input Pins  1       2       3       4       5       6
             CP      OSP     OSN     AA      BB      ONOF
;Output Pins 7       8
             CC      DD
;Input combinations
;
STRING CCA '/CP * /OSP * /OSN *  AA * /BB * /ONOF'
STRING CCB '/CP * /OSP *  OSN * /AA * /BB * /ONOF'
STRING CCC '/CP * /OSP *  OSN *  AA * /BB * /ONOF'
STRING CCD '/CP *  OSP * /OSN *  AA * /BB * /ONOF'
STRING CCE '/CP *  OSP *  OSN * /AA * /BB * /ONOF'
STRING CCF '/CP *  OSP *  OSN *  AA * /BB * /ONOF'
STRING DDA ' CP * /OSP * /OSN * /AA *  BB * /ONOF'
STRING DDB ' CP * /OSP *  OSN * /AA *  BB * /ONOF'
STRING DDC ' CP *  OSP * /OSN * /AA * /BB * /ONOF'
STRING DDD ' CP *  OSP * /OSN * /AA *  BB * /ONOF'
STRING DDE ' CP *  OSP *  OSN * /AA * /BB * /ONOF'
STRING DDF ' CP *  OSP *  OSN * /AA *  BB * /ONOF'

EQUATIONS

CC = CCA + CCB + CCC + CCD + CCE + CCF

DD = DDA + DDB + DDC + DDD + DDE + DDF
```

(f) Sub-level PAL file 'MACHINE'

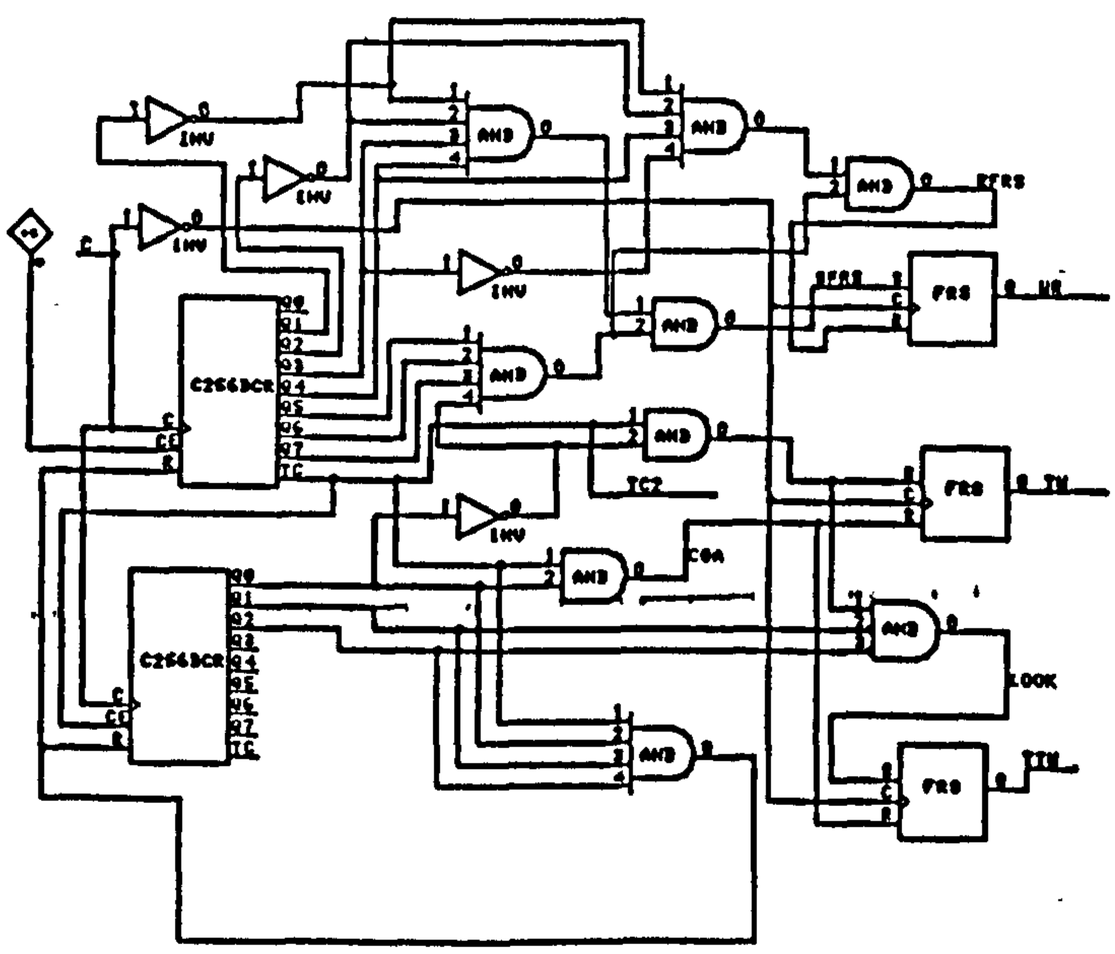
Generates switch signals from the advanced comparator signal.



TITLE			
ACTIVE			
SIZE: B	CODE:	NUMBER: 000000	REV: A
DATE: 26/4/95		SHEET 1 OF 1	
LCA PART TYPE: 3020PC68-70			

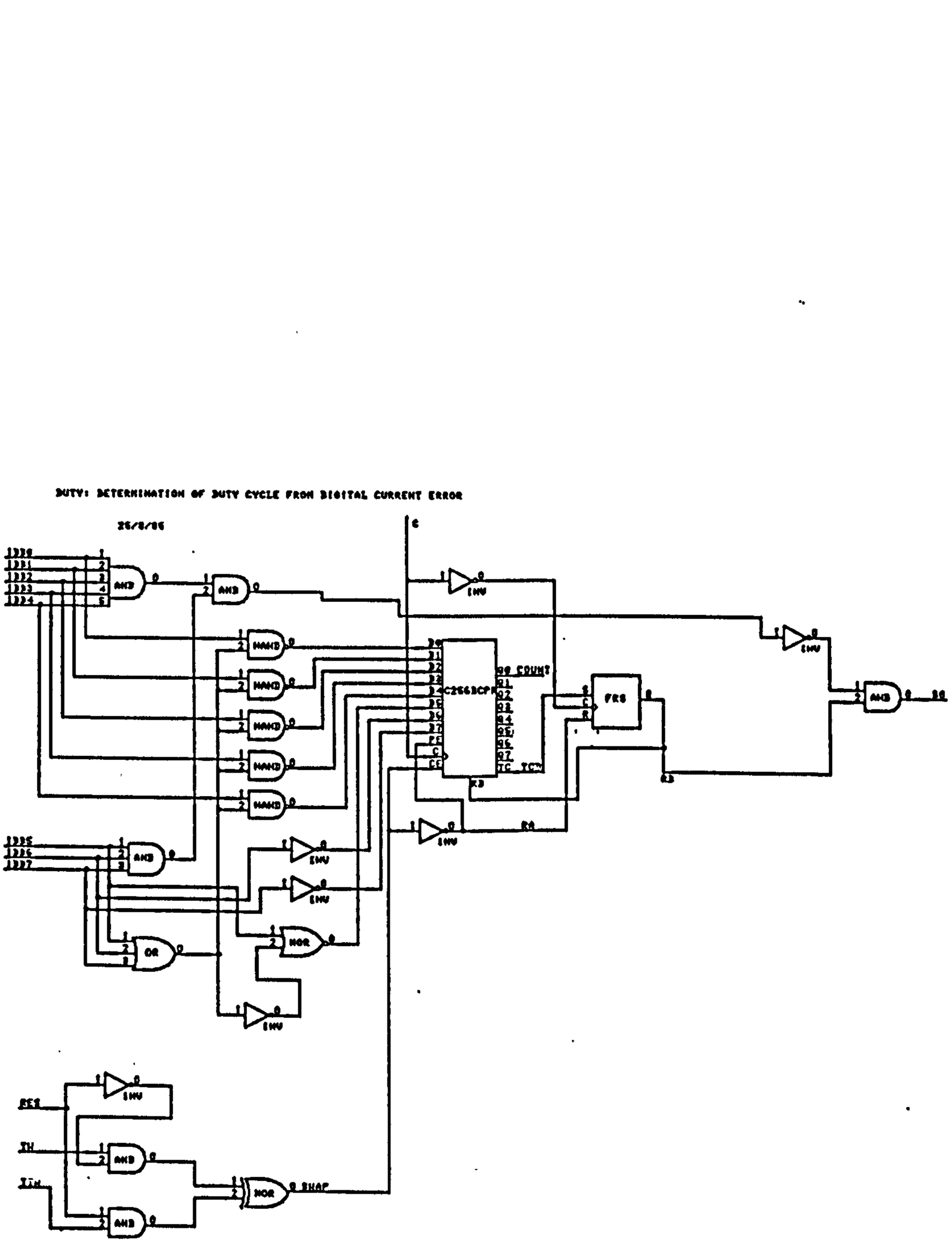
E.2 Controller of Active Rectifier

(a) Top-level Xilinx diagram ' ACTIVE'



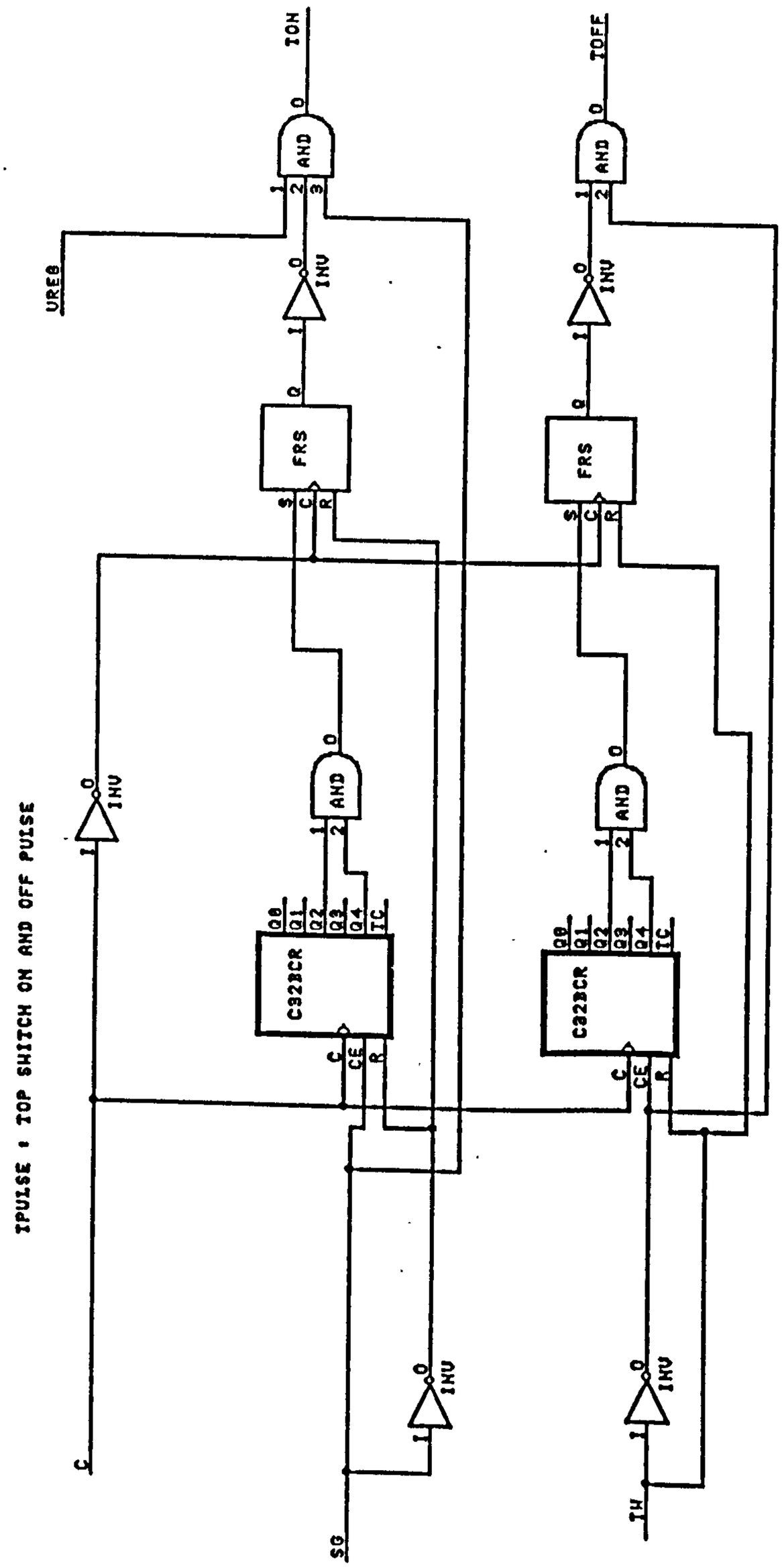
(b) Sub-level diagram of block 'WRITE'

Generates a 5 kHz 0.25 duty-cycle signal and a 20 kHz 0.5 duty-cycle signal.



(c) Sub-level diagram of block 'DUTY'

Modifies duty-cycle of 20 kHz signal to achieve required current in load.



(d) Sub-level diagram of block 'TPULSE'

Generates the 'on' and 'off' pulses required to operate the pulse-transformer gate drive circuit.