This is to certify that the
thesis entitled

ELECTRONICS SYSTEM FOR A LOW FREQUENCY
ULTRASOUND PHASED ARRAY

presented by

JASON DAVID BIEL

has been accepted towards fulfillment
of the requirements for the

M.S. degree in Electrical Engineering

Robert J. McGough
Major Professor's Signature

Sept. 24, 2009
Date

ELECTRONICS SYSTEM FOR A LOW FREQUENCY ULTRASOUND PHASED ARRAY

By

Jason David Biel

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Electrical Engineering

2009

ABSTRACT

ELECTRONICS SYSTEM FOR A LOW FREQUENCY ULTRASOUND PHASED ARRAY

By

Jason David Biel

An electronics system that allows for the independent control of all transducers in a 13 element low frequency (80kHz) phased array has been constructed and tested. The independent control of each transducer's driving waveform allows for the control of the position and intensity of the focused pressure field. Each channel can deliver up to 25W continuous wave to a 50Ω load over a frequency range of 20Hz to 200kHz. Arbitrary waveforms are defined by a numeric array of 2048 8-bit values on a computer and sent to the controlling electronics and stored in memory. The waveforms have a resolution of 320ns and 4mV. Pulsed waveform trains can have a programmed delay ranging from 0s to 167ms between cycling through the waveforms in memory. The pressure field in the water tank is measured over a 200x200x300 mm grid. The focus can be moved up to 30 mm from the geometric focus before grating lobes, due to the geometry of the transducer array, significantly distort the resulting pressure field.

This work is dedicated to my parents, David and Janet Biel, my wife, Saori Obayashi, and my children Benjamin and Ethan Biel.

ACKNOWLEDGMENT

I want to thank my parents, David and Janet Biel, who have encouraged and supported my education throughout my life. Without their love and support I would not have been able to complete my education.

I also thank Dr. Robert J. McGough for his support, patience and guidance in pursuing my masters degree. He gave me the freedom to explore and learn while providing direction when needed.

I also thank Dr. Gregory M. Wierzba for the serving on my committee, for sharing his knowledge of circuits and giving me the opportunity to learn and teach with him. I know as much as I do about circuits because of his classes. I feel that I could not have learned about circuits as thoroughly from anyone else. My time helping teach his laboratories was one of the greatest experiences in my education.

I thank Dr. Shantanu Chakrabartty for the serving on my committee and sharing his knowledge of circuits.

I thank Dr. Edwin Loh for his support and encouragement during my undergraduate education, without it I would not have been able to pursue a masters degree. I learned a tremendous amount from his patient mentoring while working with him.

I thank Brian Wright and Greg Mulder for their technical support and expertise. Their help allowed me to focus on my research.

I thank Don Chorman for building the ultrasound array.

I thank Don Vanderlaan, Matt Jennings, Marko Gerhard, Chris Mosser, and Paul Schmalenberg for their assistance in gathering data and building the system.

I would like to thank all of the members of the Biomedical Ultrasonics and Electromagnetics Laboratory that I have worked with over the years: Don Chorman, Don Vanderlaan, Matt Jennings, James Kelly, Duo Chen, Xiaozheng Zeng, Khawar Khurshid, Marko Gerhard, Chris Mosser, Lioung Wu, Ruihua Ding, Paul Schmalenberg,

Phil Gresock, and Josh Wong.

I thank Roxanne Peacock for her assistance in helping me get the supplies I needed to finish my project.

I thank Yasutaka and Michiko Obayashi for their support during my masters studies.

Finally I want to thank my wife, Saori Obayashi, and my children Benjamin and Ethan Biel. Without Saori's tough love to keep me on task I would not have been able to complete my thesis. Benjamin and Ethan have been a constant source of joy and inspiration that have provided needed stress relief and motivation during the difficult times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Cavitation is the generation of small bubbles in a liquid due to rarefraction [9, 16]. Cavitation has many medical applications such as non invasive surgery, tissue erosion, kidney stone disintegration and drug delivery [3, 2, 23, 20]. Since cavitation bubbles are generated during the negative pressure cycle, a longer period increases the chance for cavitation to happen and therefore lower frequencies should produce more cavitation at a given amount of power input from a transducer [9, 16]. This thesis describes a system that has been built to test and develop medical applications for cavitation.

Some systems can produce cavitation at higher frequencies and without the ability to adjust the output power [1, 8]. The electronics system described in this thesis has been designed to allow for the independent control of all transducers in the low frequency (80kHz) phased array, including the shape, amplitude and phase of the driving signal. The electronics system was also designed for the readout of the resulting pressure field from the water tank.

The low frequency ultrasound system is comprised of a transducer array and hydrophone in a water tank, a computer, and several custom and commercial electronic subsystems. The system block diagram is shown in Figure 1.1. A picture of the system

is shown in Figure 1.2. The computer controls the electronics that drive the transducer array, controls the position of the hydrophone in the water tank, and records the pressure reading from the hydrophone. A digital circuit stores digital waveforms in memory and outputs the digital waveforms and control signals to an array of digital to analog convertors (DACs), where the digital waveforms are transformed into analog waveforms. The analog waveforms are the input to an array of power amplifiers that drive the transducer array. A hydrophone is positioned at several points in the tank with a 3-D positioning system in order to measure the pressure throughout the water tank. At each point, the voltage waveform from the hydrophone is captured with an oscilloscope and read back and recorded by the computer.

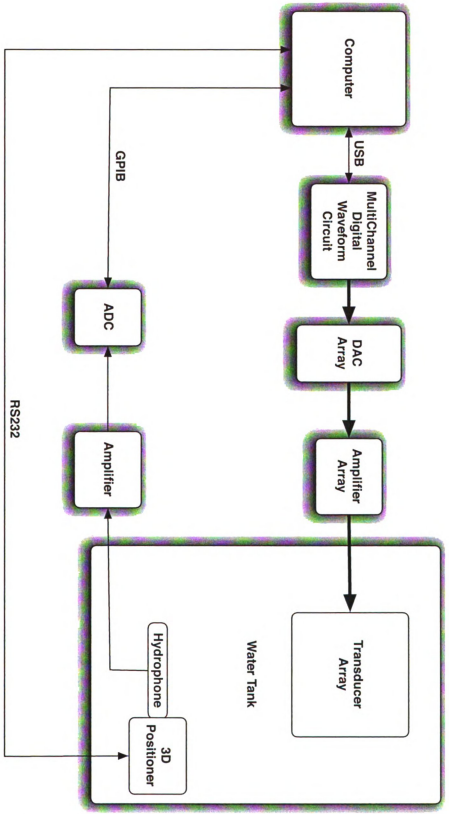


Figure 1.1: Block diagram of the low frequency ultrasound phased array system.

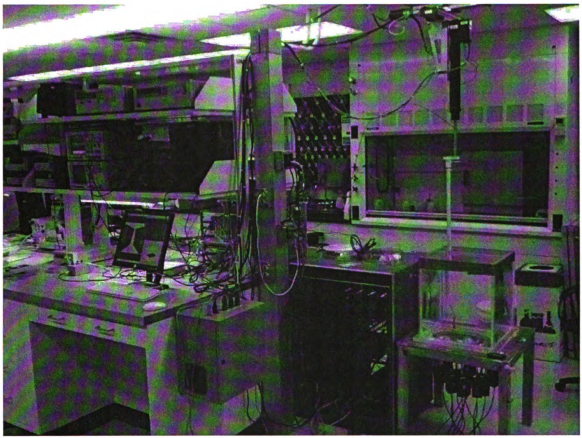


Figure 1.2: Low frequency ultrasound system including the water tank, hydrophone, positioning system, and driving electronics.

Chapter 2

Water Tank and Ultrasound Phased Array

The extent of the water tank is 380 by 380 by 470 mm. The water tank is shown in Figure 2.1. The transducer array is placed in the center bottom of the water tank. It is a 13 element spherically focused array as shown in Figure 2. The transducers are driven at 80kHz, which corresponds to a wavelength of 18.7mm in water. Each transducer is 40mm in diameter and they are spaced approximately 65mm from center to center. This spacing corresponds to a 3.5λ spacing in water at 80kHz. The transducers are labeled as shown in Figure 2.3.

A hydrophone is suspended in the water tank from a 3-D positioning system. The coordinate system is defined such that the X and Y axis are centered approximately above the center transducer. The Z axis is centered about 200 mm above the center transducer face. The X and Y axis definitions are shown in Figure 2.4. The positioning system can travel 200 mm in the X and Y direction and 300 mm in the Z direction. The resolution of the positioning system specified by Parker Hannifin is 200nm. The positioning system has a 195nm resolution. This was determined by instructing the positioning system to move 1,000,000 steps, measuring the distance travelled, and

dividing the distance travelled by 1,000,000 steps to determine the distance travelled in one step. The positioning system is shown in Figure 2.5.

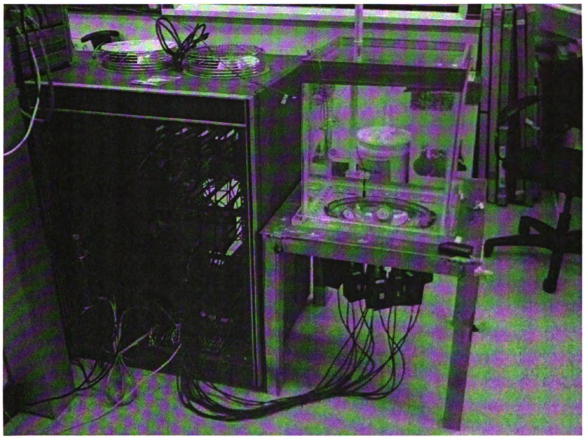
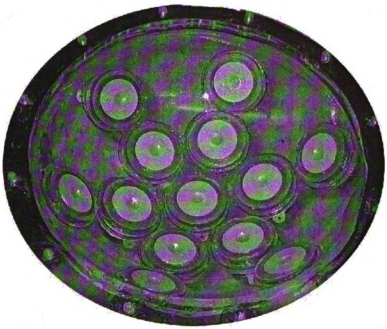
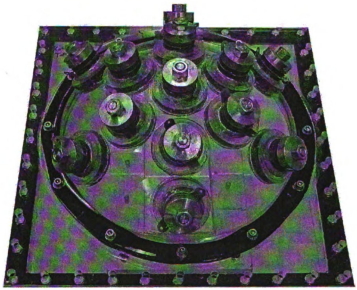


Figure 2.1: Water tank.



(a) Transducer array viewed from the top.



(b) Transducer array viewed from the bottom.

Figure 2.2: 80KHz transducer array.

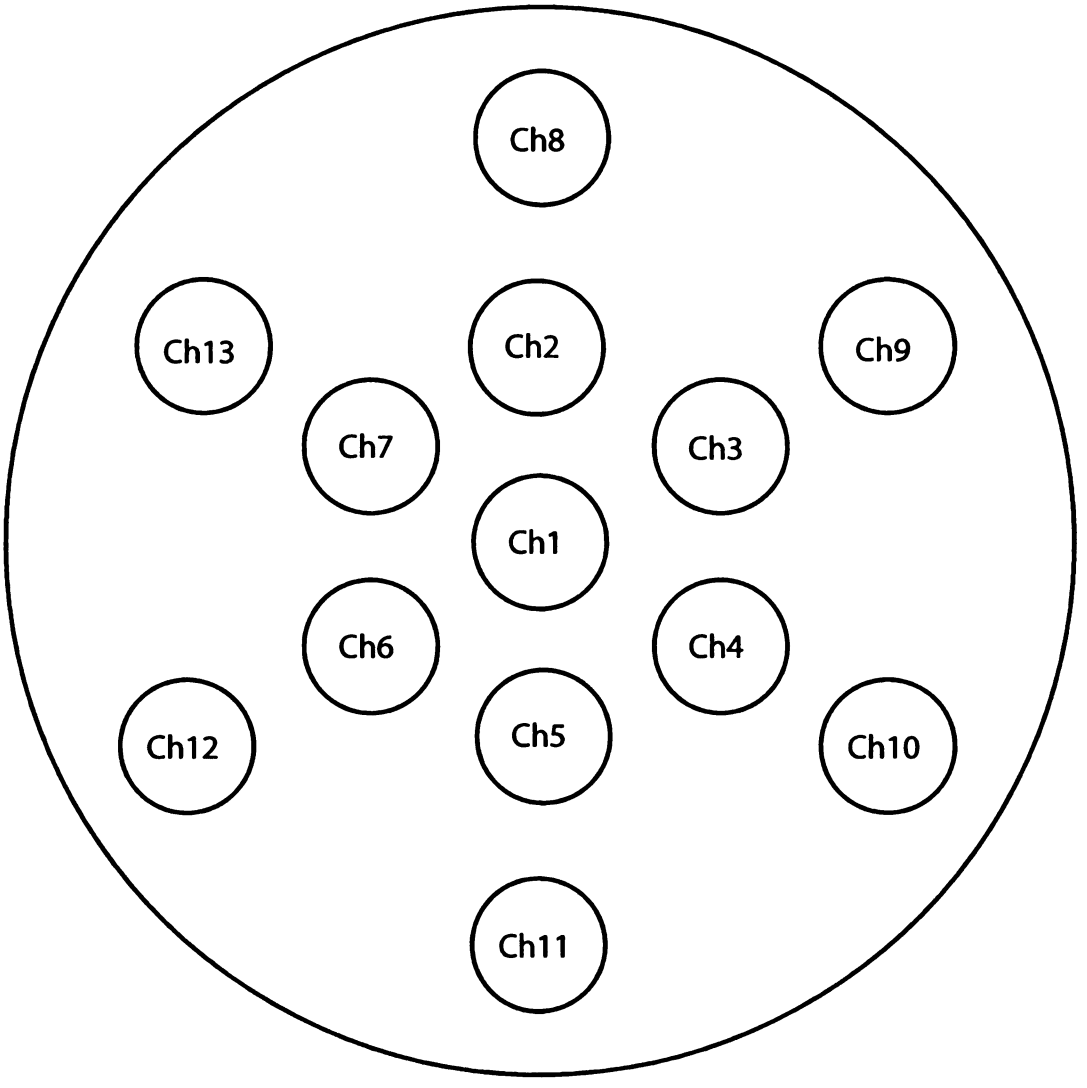


Figure 2.3: Transducer positions viewed from the top.

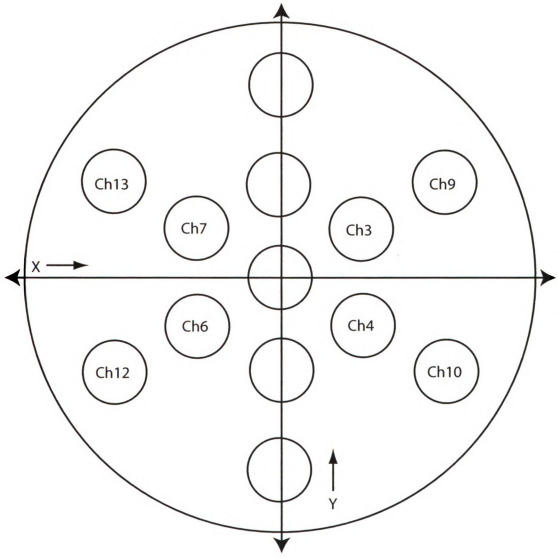


Figure 2.4: Transducer array X and Y axis definition viewed from the top.

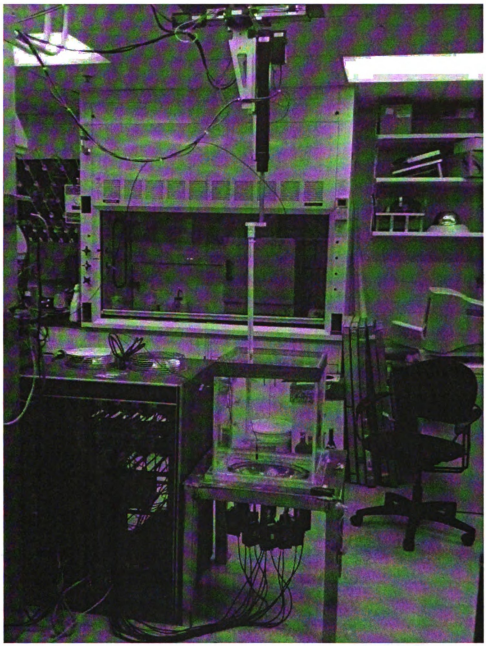


Figure 2.5: 3-D Positioning System with water tank and driving electronics.

Chapter 3

Control

The control circuit produces analog waveforms to drive the transducer array. The control portion of the system is made up of the following: a computer to define the waveforms, a digital circuit, called the arbitrary waveform generator (AWG), that cycles through the digital waveforms, an array of DACs that convert the digital waveforms into analog waveforms, and an array of matching networks and power amplifiers that drive the transducers. Figure 3.1 shows an overview of the control system.

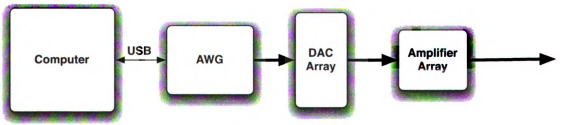


Figure 3.1: Overview of the control system.

3.1 Waveform Specification

The computer interface to control the transducer array and transducer readout is implemented in Matlab ® . Arrays defining the waveforms are generated and sent to the AWG using Matlab m-files. The waveforms are stored in memory on the AWG and output to the DACs.

3.1.1 Commands

Matlab M-files are used to store basic commands for the digital board. A list of these basic commands are shown in Figure 3.1. These commands read and write words from and to the AWG, setting the global and per-channel enable signals, and a command to define a wait period while cycling through values in memory. More complicated tasks can be created by calling these basic commands along with other Matlab code. These tasks include defining and programming some basic waveforms such as pulsed and continuous sine, ramp, square and triangle waves. A list of these complex functions are shown in Figure 3.2.

Command	Description
write_word	Write a single word to the AWG
read_word	Read a single word from the AWG
set_enable_mask	Enable any combination of channels
set_global_enable	Enable or disable all channels
Pad_delay	Set a time delay between cycling through digital waveforms stored in memory

Table 3.1: Low level AWG Matlab commands.

Function	Description
program_pulsed_sine_wave	Program a pulsed sine wave to a specified channel
program_pulsed_sine_wave _all_channels_cmdln	Program a pulsed sign wave to all channels. This version is self contained and can be run from the command line
program_pulsed_sine_waves _all_channels	Program a pulsed sine wave to all channels
program_sine_waves _all_channels	Program all channels with a specified sine wave
pulsed_ramp_array	Compute a numeric array that defines a pulsed ramp
pulsed_sine_array	Compute a numeric array that defines a pulsed sine wave
pulsed_square_array	Compute a numeric array that defines a pulsed square wave
pulsed_tria_array	Compute a numeric array that defines a pulsed triangle wave
ramp_array	Compute a numeric array that defines a repeating ramp
set_pad_delay	Program the AWG with the pad time between cycling through values stored in memory
sine_array	Program a sine wave to a specified channel
square_wave_array	Program a square wave to a specified channel
tria_wave_array	Program a triangle wave to a specified channel

Table 3.2: AWG high level commands.

3.2 USB Interface

A USB connection is used to communicate between the computer and the AWG. This makes connecting to the AWG easy since all modern computers have a USB port, so additional custom I/O cards for the computer are not needed. A USB connection also allows for faster communication that a standard RS232 interface. A USB to RS232 converter is used on the AWG to simplify the USB implementation on the AWG.

3.2.1 Software

On the computer, the virtual comm driver from FTDI® is used to interface with the external hardware through the USB port [15]. The driver makes the USB port look like a standard serial communications port (comm port). This simplifies the program interface because the only commands needed are read from and write to a standard serial port.

In Matlab, text strings are written to and read from a serial port. This is accomplished with the scanf and printf commands by passing a handle to the serial port that is associated with the USB controller.

3.2.2 Digital Board

The digital board uses an FTDI UM232R development board to convert between USB on the PC and RS-232 on the AWG digital board [14]. Figure 3.2 shows the schematic for the connection between the AWG and the FTDI UM232R development board.

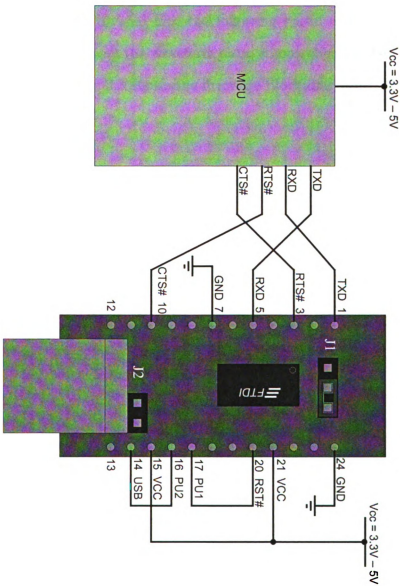


Figure 3.2: USB interface schematic [14].

3.3 Digital Board

The digital board is also called the arbitrary waveform generator (AWG). A MEMEC Spartan-3 LC Development Kit was used to implement the AWG hardware [4]. The firmware for AWG is implemented using VHDL and the Xilinx ISE design tools [10]. The AWG firmware is divided into the following parts: communications with the computer; memory to store the waveform data; a controller for the DAC signals; and clock generation. Figure 3.3 shows the firmware block diagram.

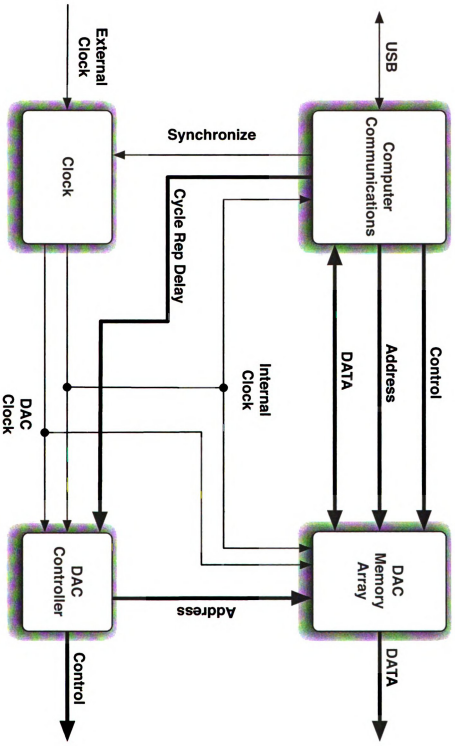


Figure 3.3: Block diagram of the AWG firmware.

3.3.1 Communications

The communications portion of the firmware communicates with the computer through the RS232 port. In the AWG, there is a receiver that triggers on new words transmitted into the serial port, converts them into an 8-bit parallel word and sends out clock signals to trigger and synchronize other parts of the firmware. The communications portion of the firmware also has a transmitter that sends data back to the computer. There is a state machine that parses words from the computer and sets address lines and control signals for the internal memory and the DAC controller. The communications firmware block diagram is shown in Figure 3.4.

Receiver

The receiver portion of the firmware triggers on new words arriving on the RS232 interface, sends a synchronize signal to the clock generator when a new word starts, and converts the serial word into a parallel word. The state diagram for the receiver state machine is shown in Figure 3.5. The timing diagram for the receiver signals is shown in Figure 3.6. The transcoding of the serial word to a parallel word is performed in the communications state machine. The communications state machine waits for the start bit on the RX signal from the RS232 port, and when the start bit is detected, each subsequent bit is put into ascending bits of the received parallel word. After the bit 7 is read, the New_Word_Available signal is asserted until the stop bit is detected, and the communications state machine starts over and waits for the next start bit. When the beginning of a new word is detected, the SYNC signal is asserted for one FPGA clock cycle in order to synchronize the external and internal baud clocks. The Baud_CLK signal is the internally generated baud clock. The discrepancy between the Baud_CLK timing and the RX signal as well as the need for the SYNC signal is described in Section 3.3.3. The New_Word_Available_AND_SYNC state diagram and signal waveforms are shown in Figure 3.7.

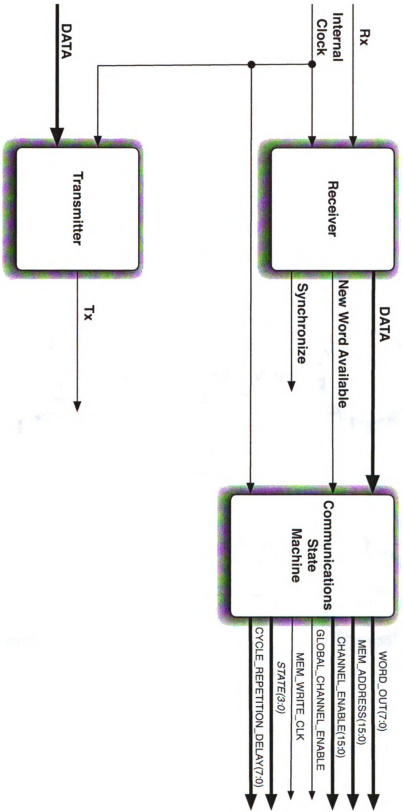


Figure 3.4: Block diagram of the communications portion of the firmware.

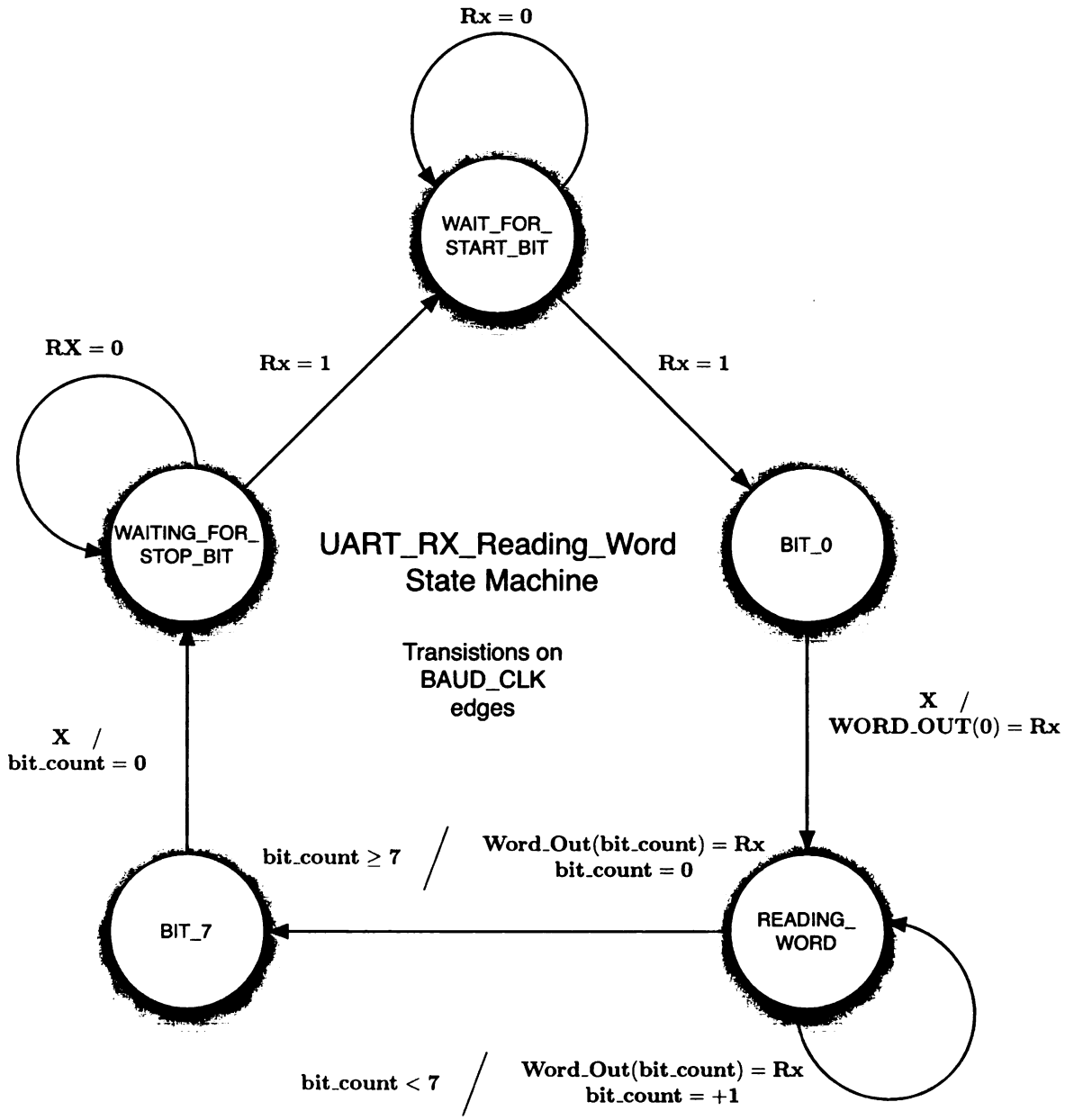


Figure 3.5: UART receiver reading word state machine. This state machine waits for a new word to start on the Rx signal, reads serial 8 bits into a register, and waits for the next word to start.

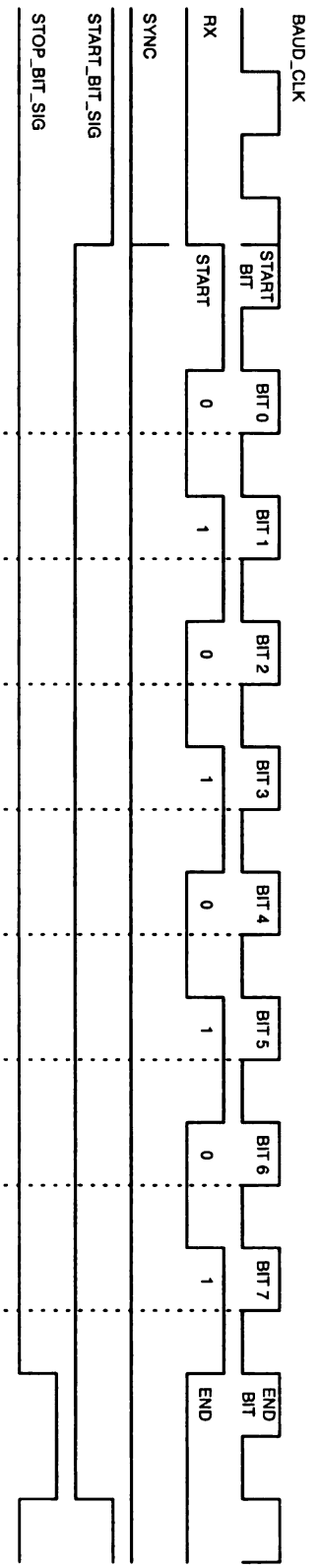


Figure 3.6: UART receiver reading word timing waveforms.

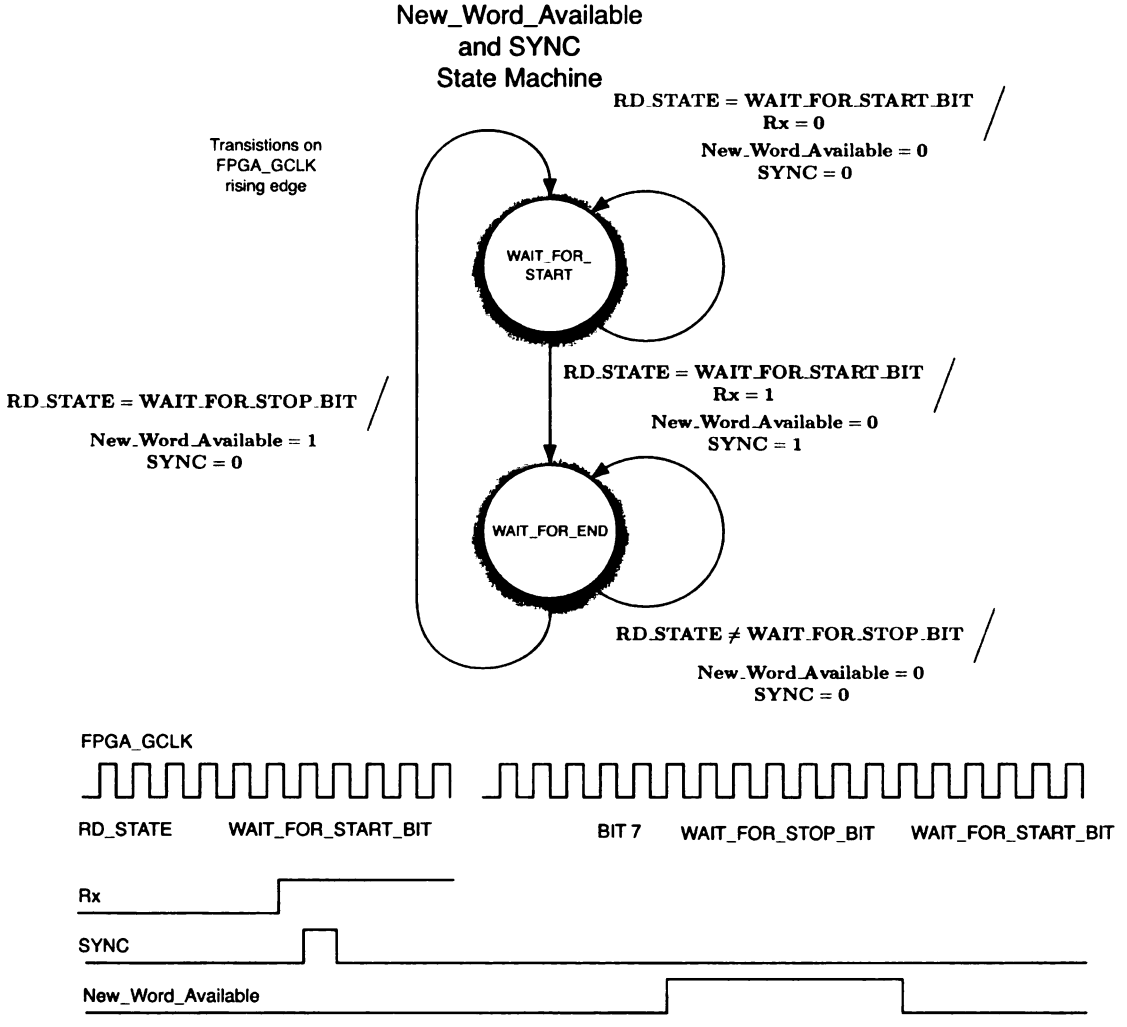


Figure 3.7: UART receiver new word available and sync state machine and timing waveforms. New_Word_Available is asserted after a new word is available from the receiver state machine. The sync signal is asserted for on FPGA clock cycle after the start of a new word is detected.

Transmitter

The transmitter portion of the AWG firmware takes an 8-bit parallel word, serializes it, and sends it back to the computer by way of RS232 communication with the FTDI UM232R development board [14]. Figure 3.8 shows the state diagram for the transmitter state machine and Figure 3.9 shows the timing diagram for the TX signal. Once the SEND_WORD signal is asserted by the communications state machine, the transmitter state machine take WORD_OUT, serializes it, and puts each bit on the TX serial signal. A start bit is asserted before the data word, and a stop bit is appended after the word.

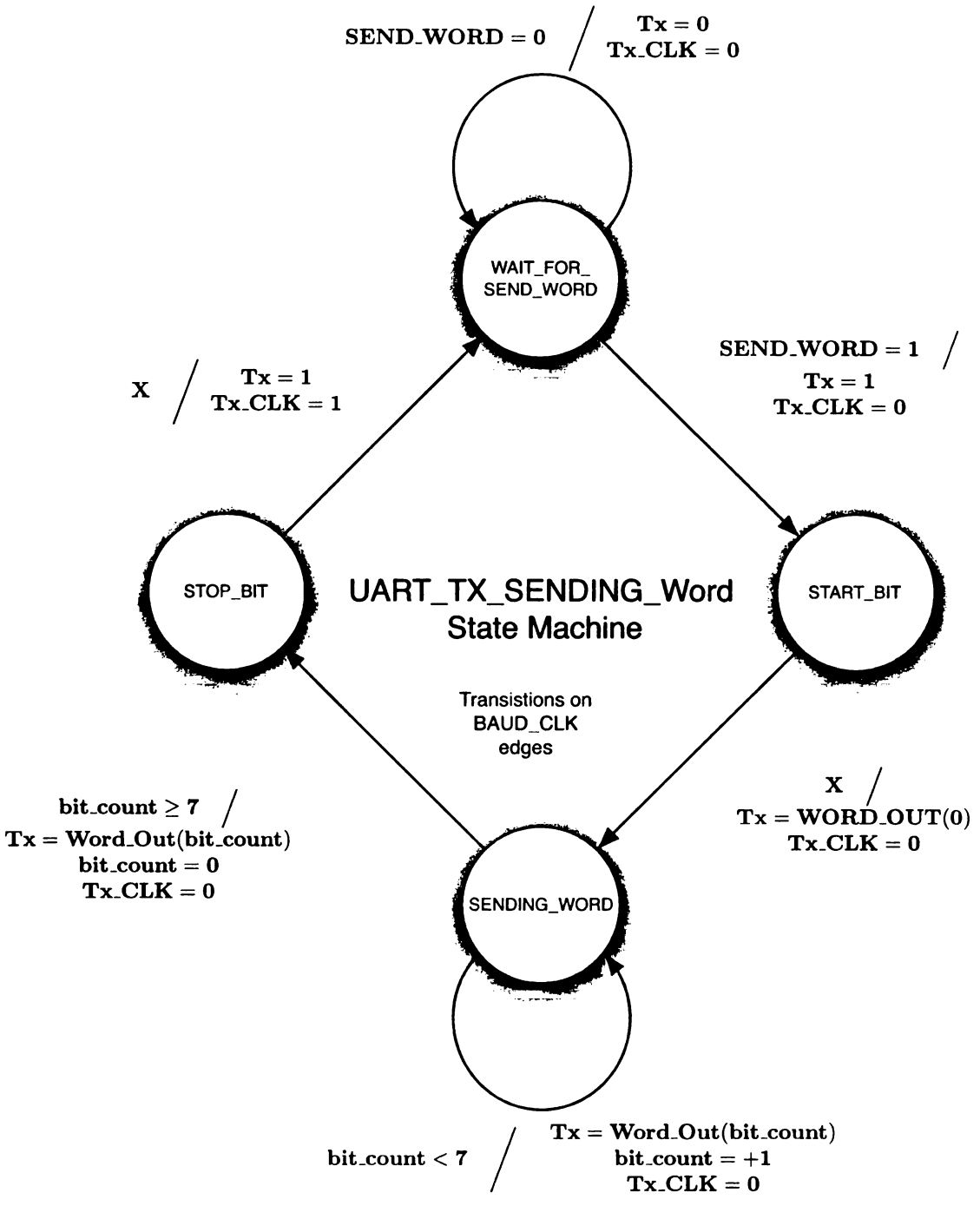


Figure 3.8: UART transmitter sending word state machine. When SEND_WORD is asserted each bit of WORD_OUT is placed on the TX serial signal.

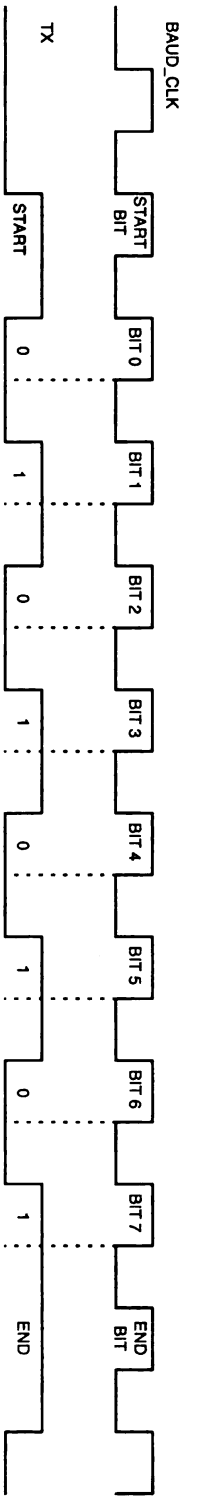


Figure 3.9: UART transmitter sending word waveforms. This waveforms shows the timing of the TX signal with respect to the baud clock for sending the value 0x33 including the start and stop bits.

3.3.2 Communications State Machine

The communications state machine waits for commands from the computer, parses the commands, and sets internal control signals and address values. The communication state machine also sends words back to the computer. The state diagram for the communication state machine is shown in Figure 3.10.

There are 9 commands that can be sent to the AWG. The commands are summarized in Table 3.3. Some commands require a single word, while others require additional words to be sent after the command.

Command(Hex)	Description	Secondary Word	Tertiary Word	4th Word
00	Reset			
01	Start			
02	Stop			
03	Set Enable Mask	Enable Mask High	Enable Mask Low	
04	Write	Address High	Address Low	Data
05	Status			
06	Read One	Address High	Address Low	
07	Read All			
08	Pad Cycle	Pad Value		

Table 3.3: AWG command summary.

The Reset command puts the AWG firmware into the startup state. This is the default state in case there is an error in communications. The Start command asserts the global enable signal. The Stop command deasserts the global enable signal. The Set Enable Mask command sets the enable mask on a per channels basis allowing for none, all or a subset of the channels to be enabled. When a channel is enabled, its memory contents are output to the DAC. When a channel is disabled, an idle value is output to the DAC. The Enable Mask is a 16-bit word; therefore, two 8-bit words must be sent from the computer to the AWG: Enable Mask High specifying bits 8-15 and a Enable Mask Low specifying bits 0-7. The Write command writes a word to an address in memory. There is 16k of addressable memory per channel; therefore, a

Figure 3.10: Communications state machine.

14 bit address is needed. The address format is shown in Figure 3.11. Bits 0-5 of the secondary command word sets bits 8-13 of the address. The tertiary command sets bits 0-7 of the address. The 4th command is the data word that is written to memory. The Status command sends the enable mask back to the computer. The Read One command reads one word from a specified address. Bits 0-5 of the secondary command word sets bits 8-13 of the address. The tertiary command sets bits 0-7 of the address. The Read All command sends all of the contents of memory back to the computer. The Pad Cycle command sets the delay time between cycles of outputting memory contents to the DACs. This is described in Section 3.3.5.

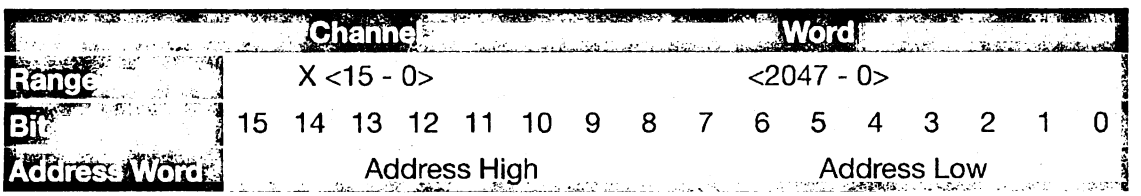


Figure 3.11: Memory addressing for the AWG.

3.3.3 Clock Generation

The clock for the AWG firmware is the external 50MHz FPGA clock. It is used to generate the other clocks needed for the AWG. The other clocks generated are the BAUD clock for RS232 communication and the DAC clock used to update the DAC control and data signals.

BAUD Clock

The BAUD clock generation state diagram is shown in Figure 3.12. The BAUD clock is generated from the FPGA clock with a clock divider implemented with a counter. A desired clock rate and duty cycle are specified in the firmware code, and a roll over count is computed for the clock divider. The relation is shown in Equation (3.1). The Divider signal is how many FPGA clock cycles there are in one baud clock cycle. The High_Cycles signal is how many FPGA clock cycles the baud clock is to be asserted. The FTDI UM232R can have baud rates defined by the Equation (3.4) [14].

$$Divider = \frac{Input\_Clock\_Freq}{BAUD\_Rate} \tag{3.1}$$

$$High\_Cycles = \frac{Divider * Duty\_Cycle}{100} \tag{3.2}$$

$$\tag{3.3}$$

$$BR = \frac{3000000}{n + x} \tag{3.4}$$

$$n \in [2^0, 2^{14}] \tag{3.5}$$

$$x \in \{0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8}\} \tag{3.6}$$

The counter increments until it reaches a value of Divider, then it resets to 0. The

BAUD_CLK is asserted when the counter is less than High_Cycles. If the count is greater than High_Cycles then BAUD_CLK is deasserted. The timing waveform for the BAUD_CLK is shown in Figure 3.12.

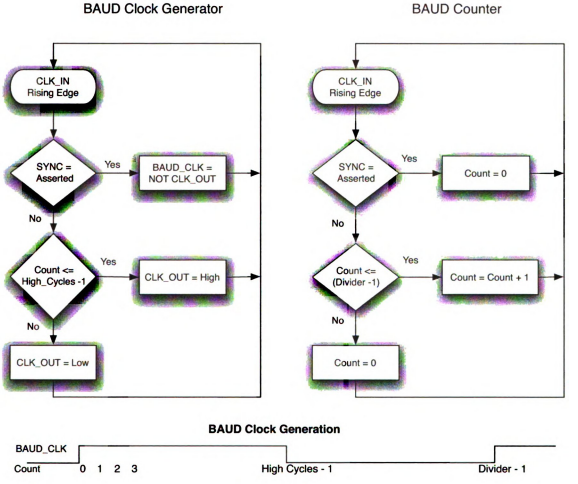


Figure 3.12: BAUD clock state machine

BAUD Clock synchronization

The FTDI UM232R has a reference clock of 48MHz, and the AWG has a reference clock of 50MHz [14]. This difference in clock speed can potentially cause a skew in the BAUD clocks in the UM232R and the AWG that would allow for a bit to be skipped. This is illustrated in Figure 3.13. In order to prevent a bit from being skipped, the clock edges of the two BAUD clocks on the FTDI UM232R and the AWG

are synchronized at the beginning of each word that is sent. This is accomplished with the SYNC signal from the receiver state machine that was described in Section 3.3.1. The SYNC signal is asserted for one FPGA clock cycle when a new word is received. This resets the internal BAUD clock so that its edges are coincident with the FTDI UM232R clock edges. The SYNC signal generation state diagram and timing waveforms are shown in Figure 3.14. The timing diagram for the synchronization is shown in Figure 3.7.

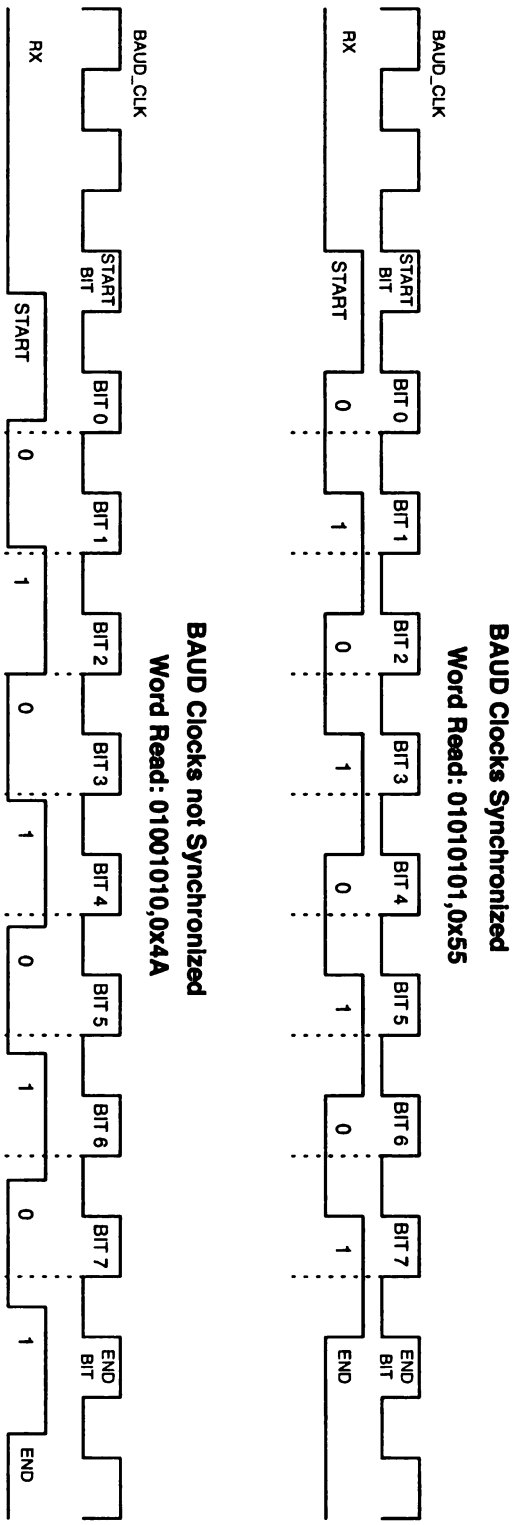


Figure 3.13: BAUD clock synchronization.

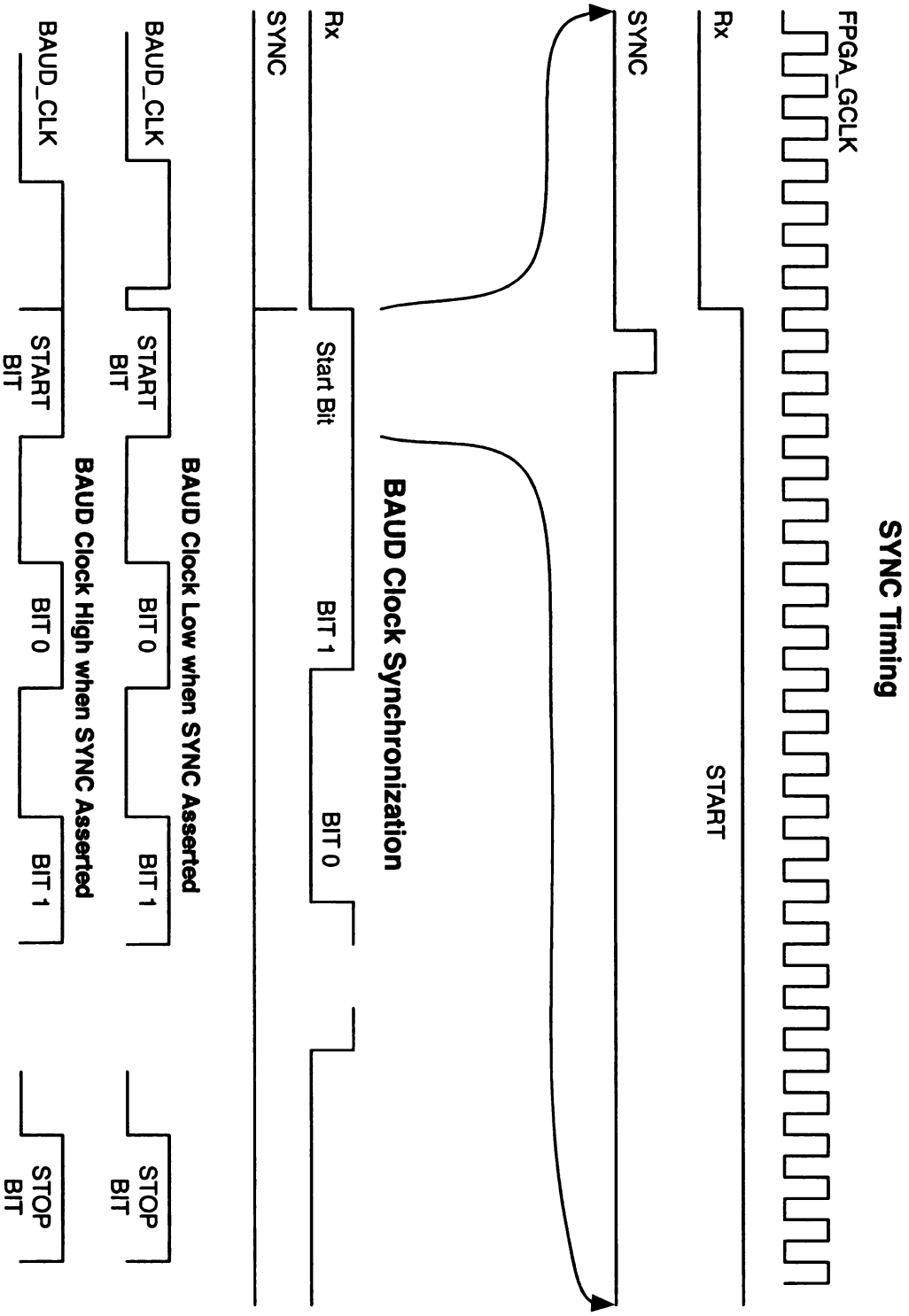


Figure 3.14: BAUD clock synchronization waveforms.

DAC Clock

The DAC clock generation state diagram and timing waveform is shown in Figure 3.15. The DAC clock is generated from the FPGA clock using a clock divider implemented with a counter. A desired clock rate and duty cycle are specified in the firmware code and a roll over count is computed for the clock divider. The relation is shown in Equation (3.7). The divider needs to be calculated in two ways since Divider * Duty_Cycle / 100 produces a negative number for frequencies 1 Hz and lower because Divider * Duty_Cycle sets bit 32 of Divider high, which is interpreted as a negative number. The sign is preserved when the next operation, divide by 100, occurs. If the divide by 100 operation is performed first, bit 32 is never asserted and the result is never considered negative. This does not work for frequencies less than Input_Clock_Frequency*100 because Divider/100 < 1, which is rounded to 0. Then, High_Cycles is 0, and the clock never changes.

$$Divider = \frac{Input\_Clock\_Freq}{Output\_Freq} \tag{3.7}$$

$$High\_Cycles = Divider * \frac{Duty\_Cycle}{100} \tag{3.8}$$

$$High\_Cycles = \frac{Divider}{100} * Duty\_Cycle \tag{3.9}$$

Figure 3.15: DAC clock generation.

3.3.4 Memory

The Xilinx Spartan 3 XC400 FPGA that is on the MEMEC Spartan-3 LC Development Kit has 16 RAM blocks that each have 16Kb of addressable data [11]. One block of RAM is used to store the digital waveforms for a given channel. Each block of memory can be configured for a different number of addressable words of different lengths as shown in Figure 3.16. The memory organization chosen for the RAM blocks is 2K words by 8-bits. Since the DACs inputs are 8-bit numbers, each address stores one value that is output to the DACs. Two bytes are needed in order to address the words in each channel. Eleven bits are needed to address a word in the block of RAM, and four bits are needed to select the block of RAM for desired channel. Figure 3.17 summarizes the memory block organization.

Total RAM bits, including parity	18,432 (16K data + 2K parity)
Memory Organizations	16Kx1 8Kx2 4Kx4 2Kx8 (no parity) 2Kx9 (x8 + parity) 1Kx16 (no parity) 1Kx18 (x16 + 2 parity) 512x32 (no parity) 512x36 (x32 + 4 parity) 256x72 (single-port only)
Parity	Available and optional only for organizations greater than byte-wide. Parity bits optionally available as extra data bits.
Performance	240+ MHz (refer to individual FPGA family data sheet)
Timing Interface	Simple synchronous interface. Similar to reading and writing from a register with a setup time for write operations and clock-to-output delay for read operations.
Single-Port	Yes

Figure 3.16: Xilinx Spartan-3 memory structure [11].

RAMB16_S9	2048 8-bit words															
	Channel								Word							
Range	X <15 - 0>								<2047 - 0>							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Address Word	Address High								Address Low							
	X	EN							Word Address							

Figure 3.17: Memory structure with 2048 addressable 8-bit words per block of RAM in 16 blocks of RAM.

Each memory block is implemented within dual port memory with independent address, enable, and clock signals. The schematic for the memory block is shown in Figure 3.18. For the AWG firmware, Port A is used to write values for the DAC waveforms into memory and to output the contents of memory back to the computer via the communications state machine. Port B is used to output the waveforms to the DACs from the contents of memory, where the input to Port B is not used.

There are two read/write modes for the Xilinx memory that were utilized in the AWG firmware: *read first* and *write first*. *Write first* is used on Port A because the new values being written into memory are also ouptut to the DACS. *Read first* is used on Port B because the input to Port B is not used and the contents of memory are always the source of the output. The memory timing requirements specified by Xilinx for dual port *read first* are shown in Figure 3.19 and the schematic is shown in Figure 3.20. When Write Enable (WE) is low, ENABLE is asserted, and there is a rising Clock (CLK) transition, data is read out from the specified address and the data input (DATA_in) is ignored. When WE is high, ENABLE is high, and there is a rising edge on CLK, Data_in is stored at the specified address and the old contents at that address are output to Data_out.

The memory timing requirements specified by Xilinx for dual port *write first* are shown in Figure 3.21 and the schematic is shown in Figure 3.22. When WE is low, ENABLE is asserted, and there is a rising CLK transition, data is read out from the

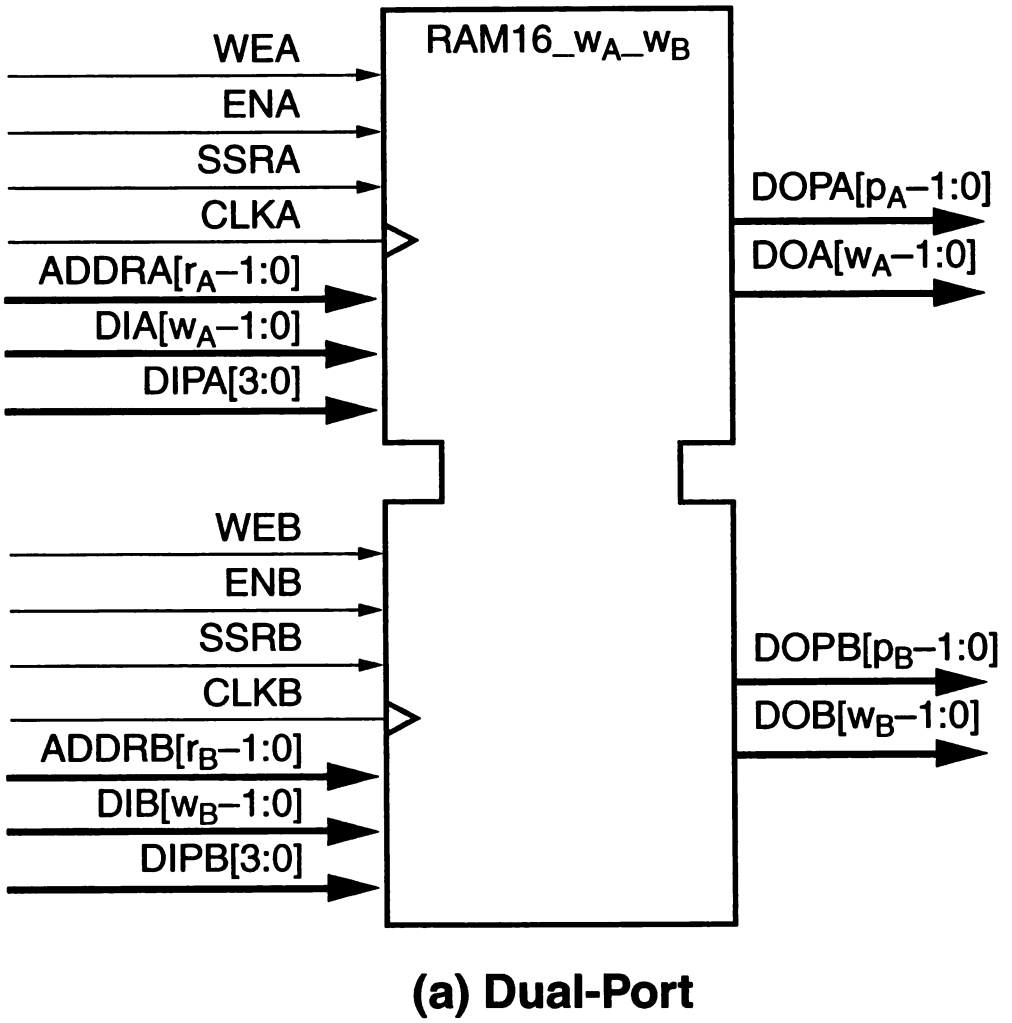


Figure 3.18: Xilinx memory schematic [11].

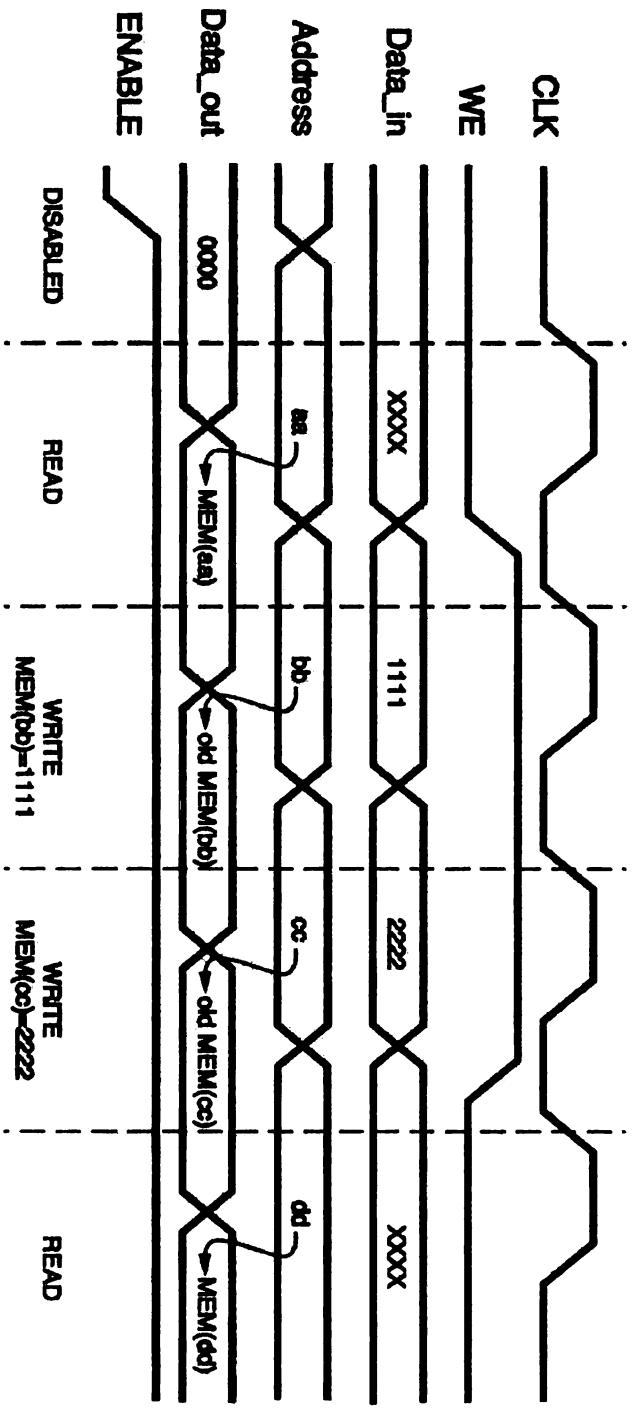


Figure 3.19: Memory timing requirements for dual port read first [11].

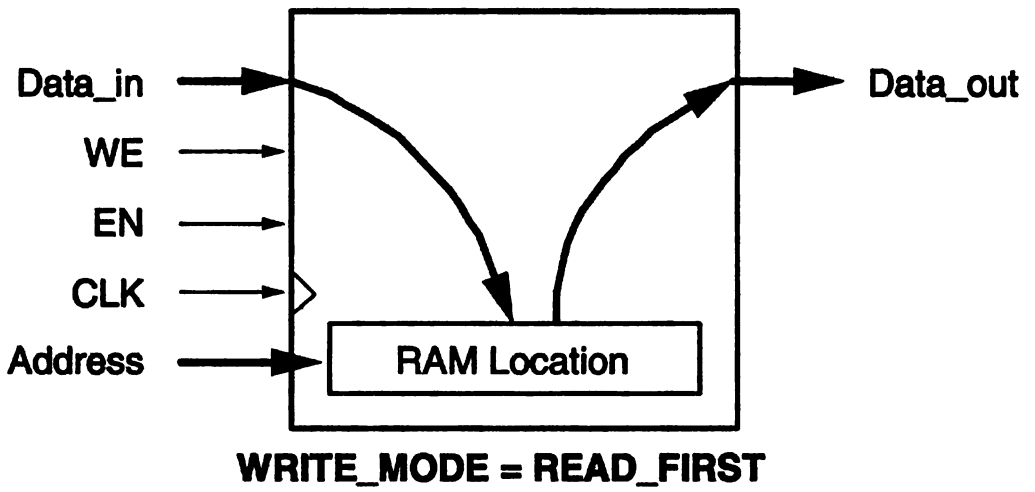


Figure 3.20: Schematic for dual port read first operation [11].

specified address and Data_in is ignored. When WE is high, ENABLE is high, and there is a rising edge on CLK, Data_in is stored at the specified address and is output to Data_out.

The timing waveforms generated in the AWG memory operations are shown in Figure 3.23. The address and data signals are set in the communications state machine for a write cycle. The enable signal is asserted, and one FPGA clock later, the write clock (WE signal) is asserted for one FPGA clock period. On the following rising edge of the FPGA clock, the memory at the address specified is changed to the new data.

The overall schematic for the memory interface is shown in Figure 3.24. Port A is used for read and writing information to memory from the computer via the communications state machine. Port B is used to output the contents of memory to the DACs.

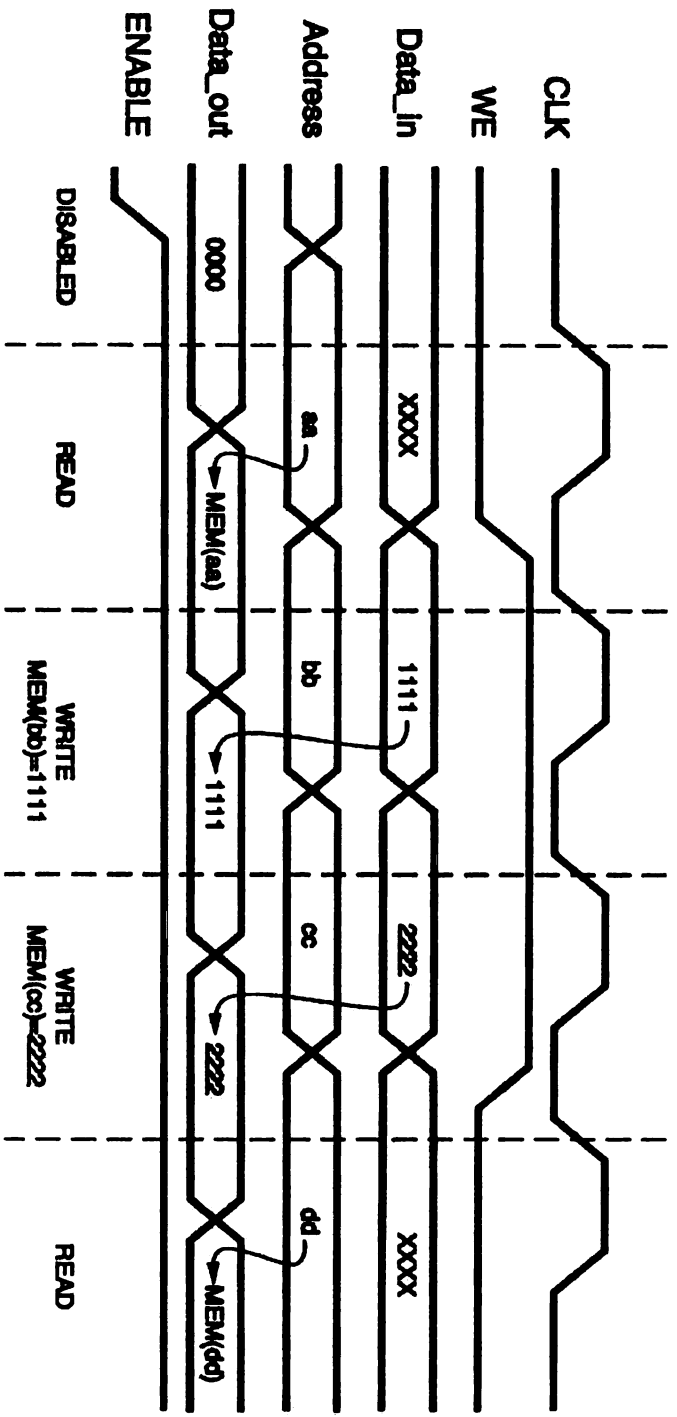


Figure 3.21: Memory timing requirements for dual port write first [11].

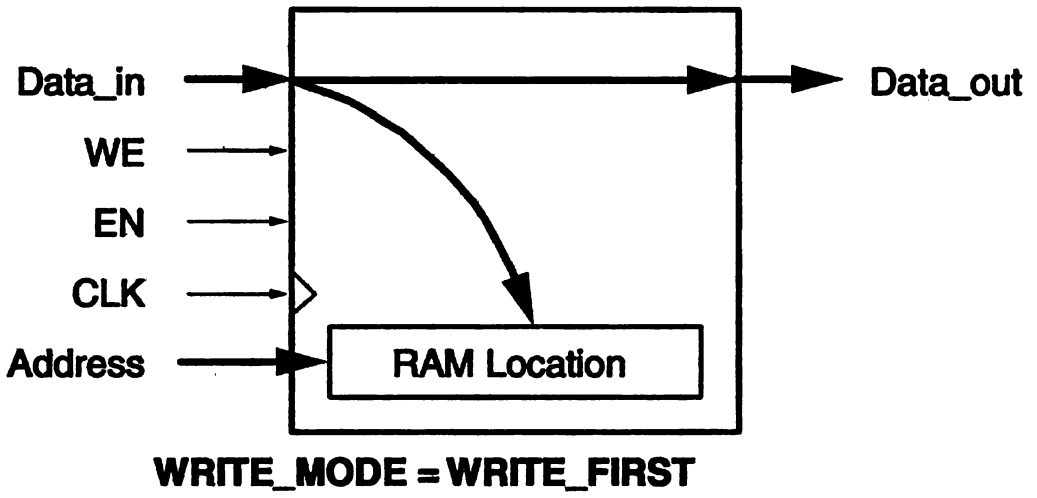


Figure 3.22: Schematic for dual port write first operation [11].

Port A: Computer Read and Write

The clocks that are fed into the blocks of RAM are the 50MHz FPGA_CLK attached to RAM block port CLK and the write clock (WR_CLK) attached to RAM block port WEA. The FPGA clock clocks both read and write actions and the WR_CLK clock selects a read or write cycle for Port A. The address in (ADDR_IN) is a 15-bit address that specifies which channel (block of RAM) to program and which word in that block of RAM to read and write. The 4 highest bits (ADDR_IN(14:11)) select which block of RAM is being addressed. They are passed through a 4:16 decoder to assert a single enable signal (ENA) to one of the blocks of RAM. The lower 11 bits (ADDR_IN(10:0)) are passed to ADDRA to specify which word in the block to address. The data input (D_IN) is an 8-bit word that is passed to DIA of each block of RAM as shown in Figure 3.24. Data parity is not used so port DIPA is held high [11]. Port A is not considered in a reset operation because its output goes to the transmitter and not to the DACs so SSRA is set low.

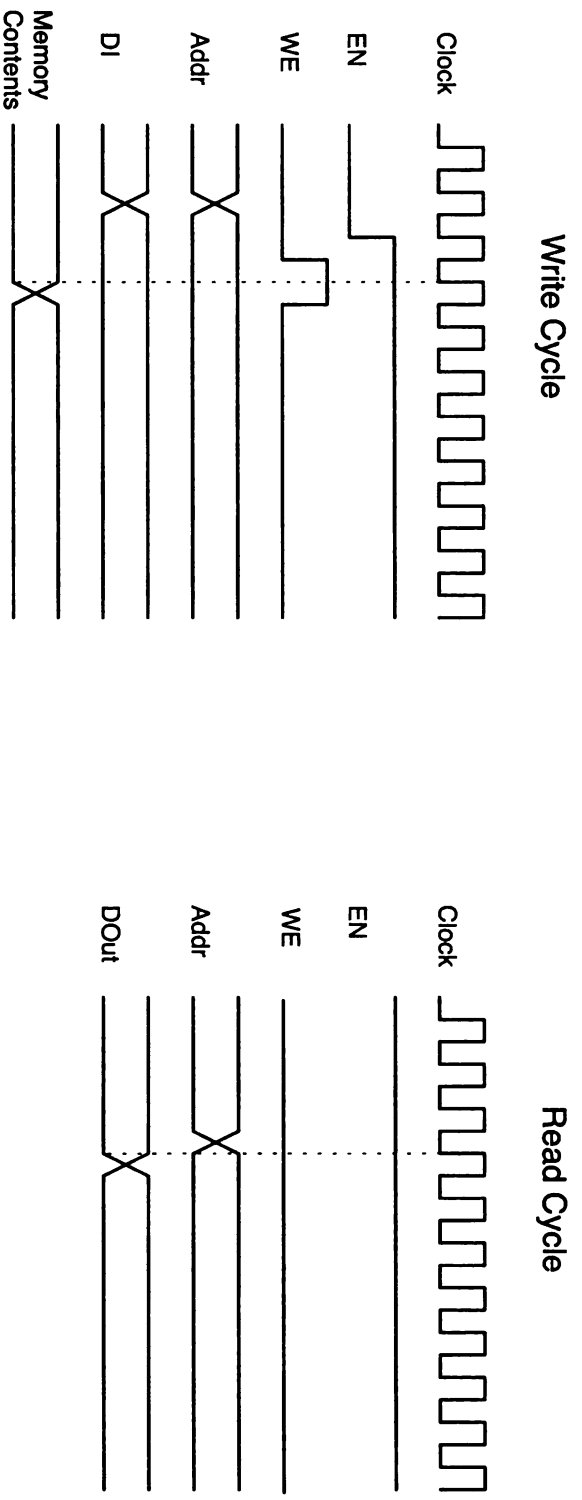


Figure 3.23: Memory timing waveforms

Port B: Computer Read and Write

The global enable (Global_Enable) and per channel enable (Per_Channel_Enable) signals allow the memory contents to be output to the DACs when asserted or output a predefined constant value when deasserted. The global enable and per channel enable signals are ANDed together and fed to SSRB for each block of RAM. If the global enable is low or the per channel enable for that block of RAM is low, the SSRB is low and the SSRVAL specified in the VHDL code is output to the DAC. Otherwise, if both are asserted, then SSRB is high and the contents of that block of memory are output to the DAC. The DAC clock signal is fed into the CLKB port of each block of RAM. The clock period is defined by the timing requirements of the DACs and is discussed in Section 3.3.6. The address out signal (ADDR_OUT) specifies which word in memory is output to the DACs. The bus select out signal specifies if even or odd channels are updated during the current memory cycle. The motivation for multiplexing between even and odd channels is discussed in Section 3.3.6. Since Port B is used only to output the memory contents to the DACs, write actions on Port B are disabled, and Port B is always set to read. This is done by setting WEB to low and ENB high. DIB is also held high in order to prevent the data inputs from floating. Data parity is not used so port DIPB is held high.

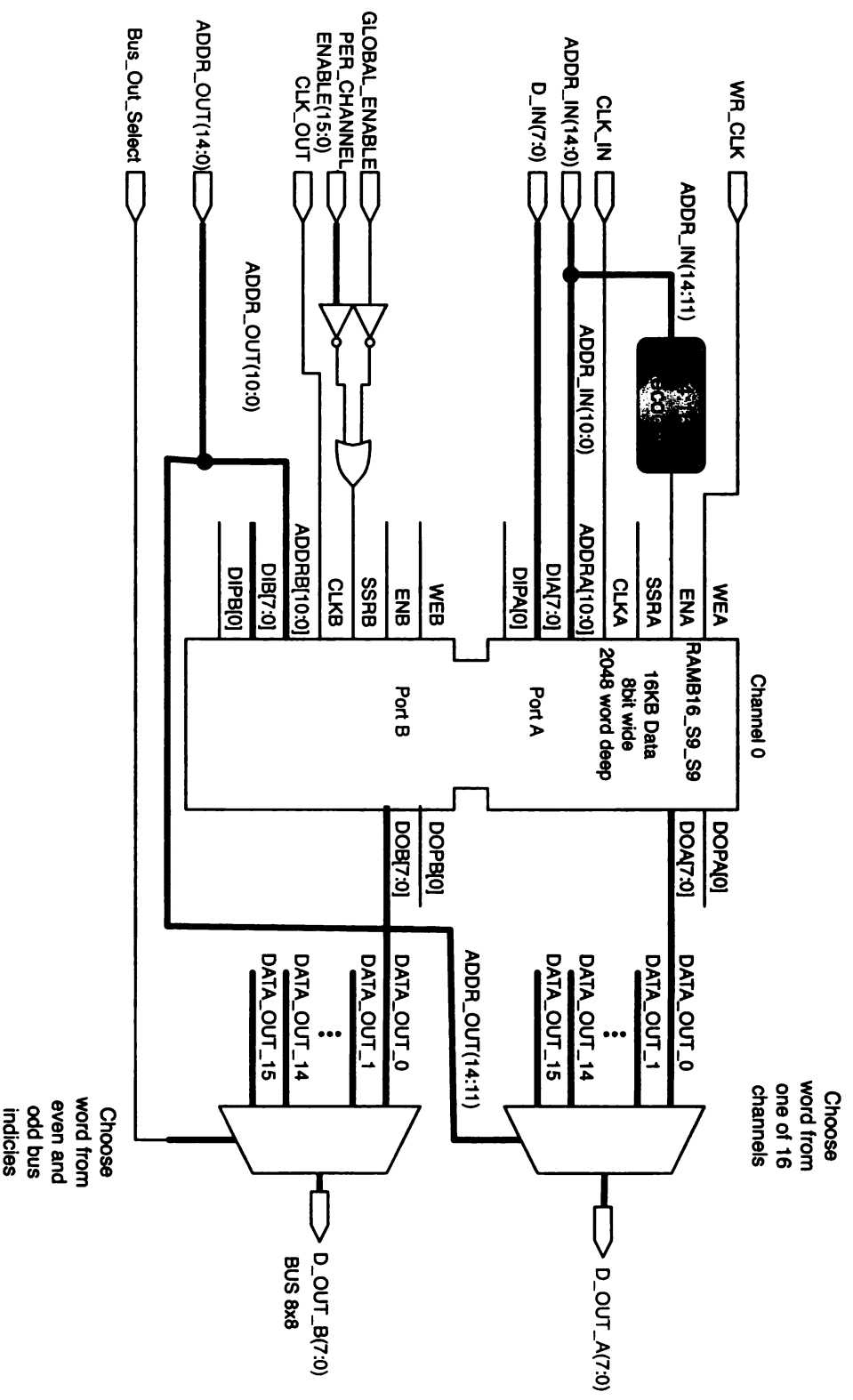


Figure 3.24: Memory schematic for channel 0. The same connections are made for channels 1-15.

3.3.5 DAC Controller

As shown in Figure 3.3, the DAC controller provides control signals to the DACs and addresses and a clock signal for the memory to select which words are output to the DACs. The control signals needed by the DACs are chip select bar ($\overline{CS}$) and write bar ($\overline{WR}$). The timing requirements for these signals and the data are shown in Figure 3.25. $\overline{WR}$ is always held low, so the timing requirements for it are not considered. $\overline{CS}$ must be held low for at least 40ns during a DAC update. The data must be stable for 15ns after ($\overline{CS}$) has been asserted, and the data must be held stable for at least 25ns. Figure 3.26 shows the timing that is implemented in the AWG. $\overline{CS}$ (CS_B) is held low for 40ns fulfilling the 40ns setup time requirement. The data is stable for at least 70ns prior to CS_B being asserted and is held 30ns after CS_B is deasserted, fulfilling the 15ns stable time and 25ns setup time requirements. The time between successive DAC updates is 320ns. As described in Section 3.3.6, DACs sharing a data bus are updated in an interleaving fashion where the data on the bus changes from one channel at the specified memory address when Out_Bus_Select is low to the other when it is high. CS_B is asserted in the middle on the Out_Bus_Select assertion in order to maximize the settling time. There is 160ns between DAC updates in a pair.

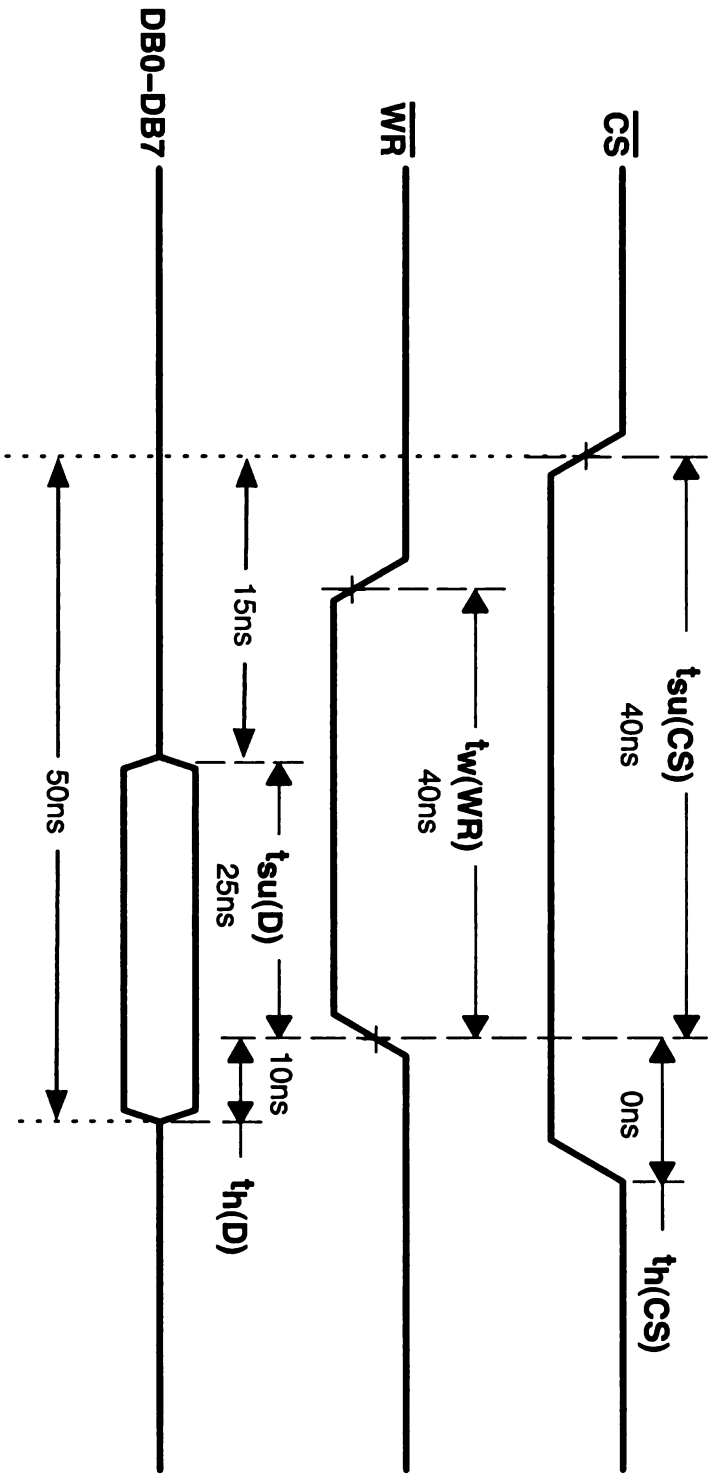


Figure 3.25: DAC timing requirements [13].

Figure 3.26: AWG DAC timing waveforms.

Figure 3.27 shows a pulsed waveform repeating. The maximum time between a pulsed wavefrom is $655\mu s$ because there are 2048 words per channel in memory and the DACs values are updated by cycling through the words every 320ns, where 320ns*2048 = $655\mu s$. If more time is required between pulses, then a wait time between cycling through values in memory is added. This added wait time is the padding. This padding is accomplished by adding a programmable counter into the address count. The counter increments in time units of $655\mu s$, or increments of memory cycle time. This is shown in Figure 3.27. The memory timing waveforms are shown in Figure 3.28. There is a master count that increments every FPGA clock cycle (20ns). There is a cycle count that increments every 16 FPGA clock cycles (320ns). This count generates the DAC memory clock. The DAC address updates every 320ns as well. The DAC address cycles from 0 to 2047. After cycling through all the DAC addresses, the master count is reset and held constant, the DAC address stops incrementing and the DAC memory clock stops pulsing. The pad count then starts incrementing to a programmed count. After the pad count stops, the pad count resets, the master count starts incrementing again, the DAC memory address starts incrementing, and the DAC memory clock starts pulsing again.

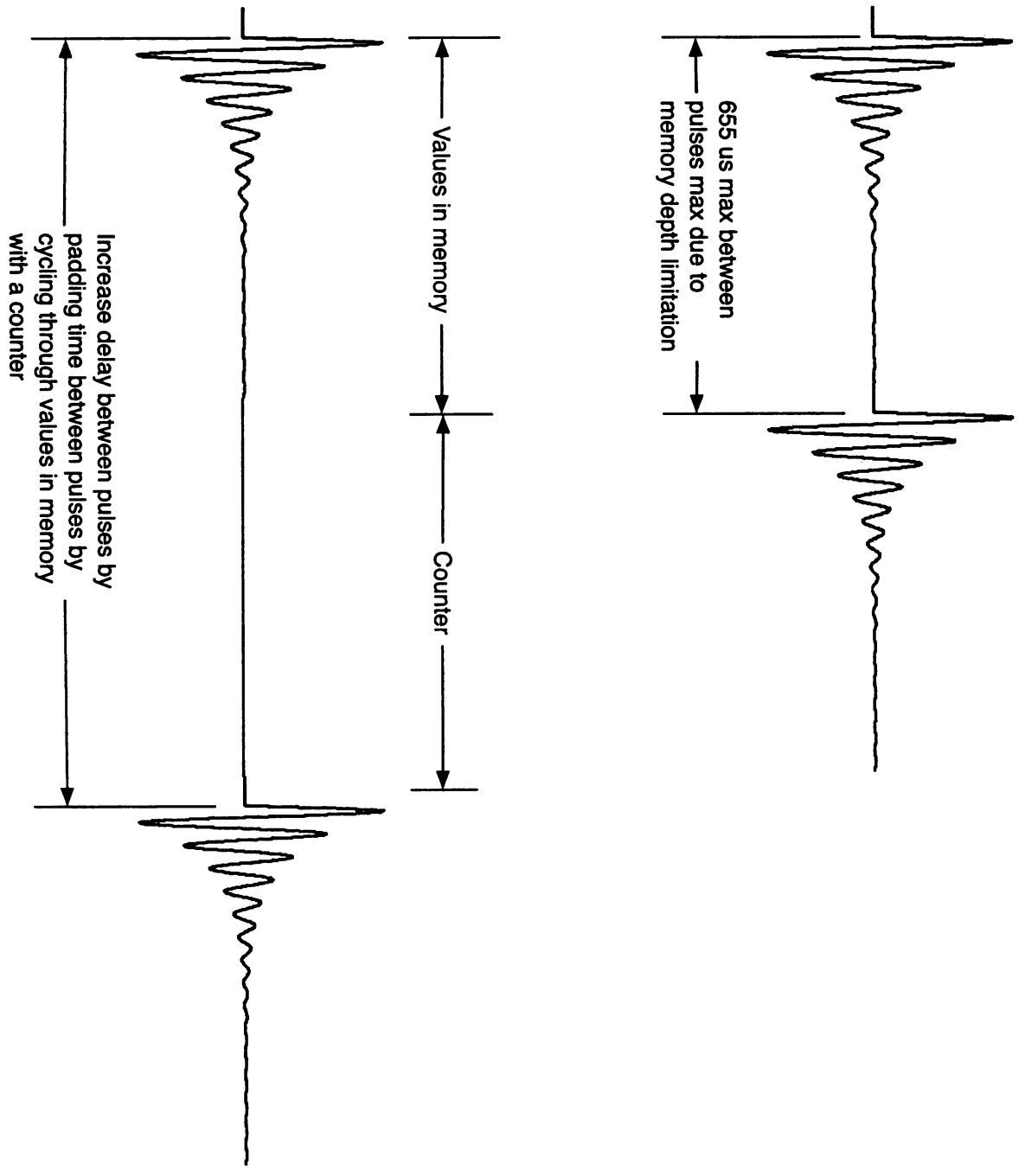


Figure 3.27: Pulsed waveform and cycle padding.

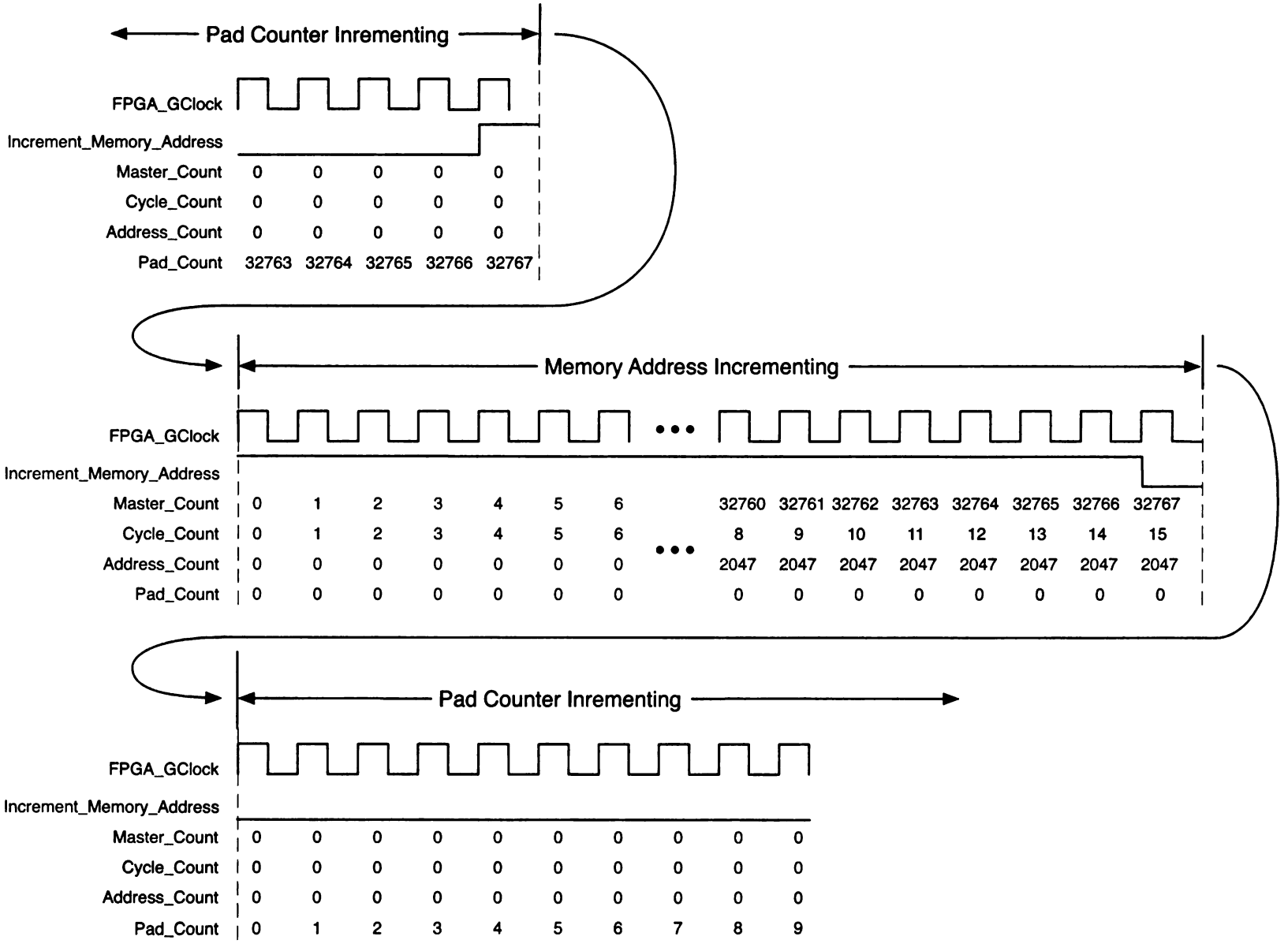


Figure 3.28: Cycle padding waveforms.

3.3.6 I/O

There are 97 IO pins available on the headers of the MEMEC Spartan-3 LC Development Kit using the P160 Proto Board [5]. The headers on the P160 Proto Board are standard 40 pin dual row with one-tenth inch spacing. The pinouts and associated FPGA pin nets are shown in Figures 3.29 - 3.32. The required signals for the AWG design are the following: communications signals Tx, Rx, CTS, and RTS, and DAC signals for 16 channels. The signals needed per channel are: Data(7:0), WR_B, and CS_B. If each DAC is updated independently and simultaneously, then 164 signals would be needed. This is summarized in Figure 3.33. If 2 DAC channels are updated serially, then they can share a common data bus and 8 8-bit data busses would be needed instead of 16 8-bit data busses, reducing the needed I/O lines by a factor of 2. 92 IO pins are needed to implement this arrangement, which is summarized in Figure 3.34. This arrangement would minimize the time between DAC updates by allowing for parallel updates to happen between 8 DACs at a time while reducing the number of pins needed by sharing a data bus between 2 DACs and satisfying the I/O pin constraint.

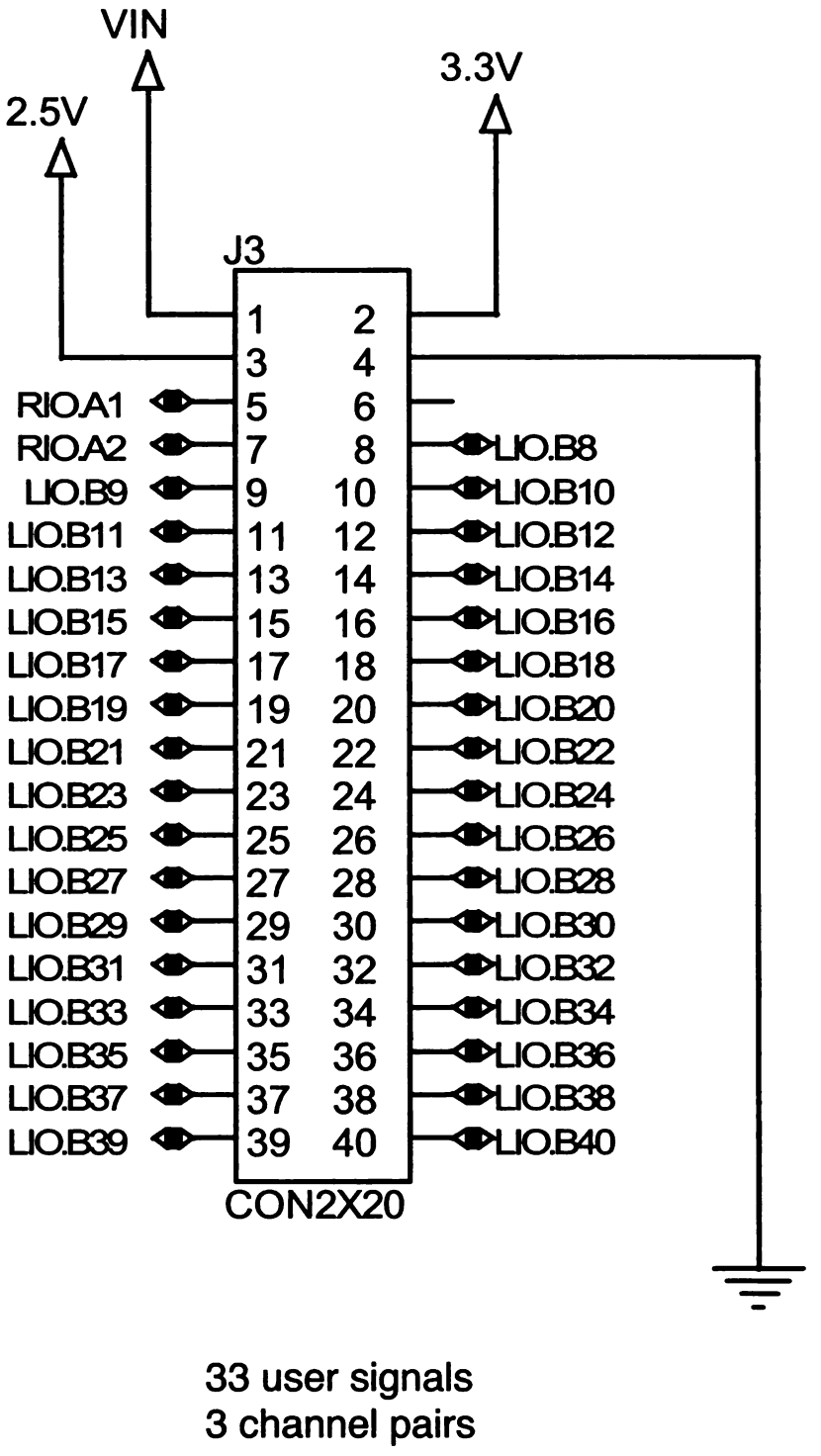


Figure 3.29: AWG I/O connector J3 [5].

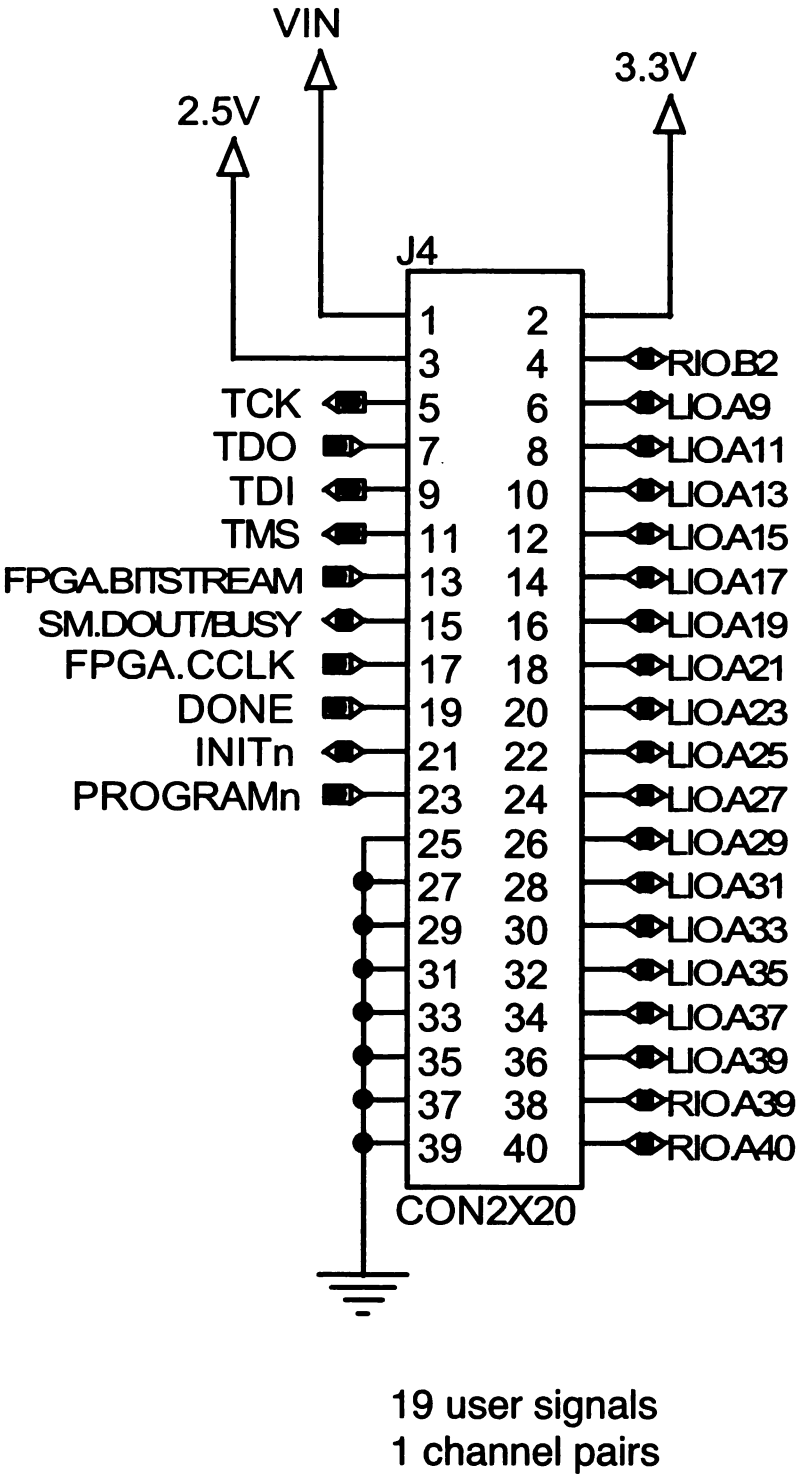


Figure 3.30: AWG I/O connector J4 [5].

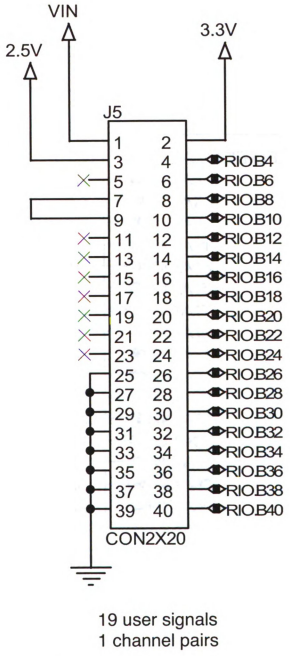


Figure 3.31: AWG I/O connector J5 [5].

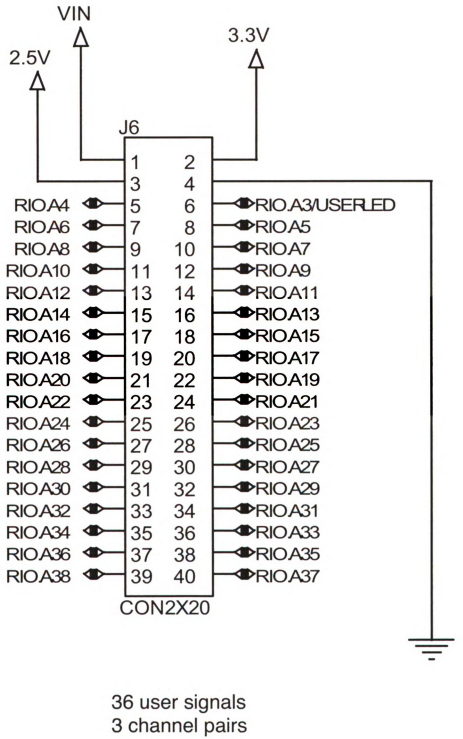


Figure 3.32: AWG I/O Connector J6 [5].

16 Independent DAC Channels				
Function	Signal	Per Channel	Per Chip	
RS-232/USB	Tx		1	
	Rx		1	
	CTS		1	
	RTS		1	
DAC	Data(7:0)	8	128	
	WR_B	1	16	
	CS_B	1	16	
TOTAL			164	97 available

Figure 3.33: 16 channel DAC I/O resources.

8 Channel Pairs				
Function	Signal	Per Channel	Per Chip	
RS-232/USB	Tx		1	
	Rx		1	
	CTS		1	
	RTS		1	
DAC	Data(7:0)	8	64	
	WR_B	1	16	
	CS_B	1	8	
TOTAL			92	97 available

Figure 3.34: 8 channel pairs DAC I/O resources.

3.4 DAC Board

3.4.1 Digital to Analog Converter

The Digital to Analog Converter (DAC) converts 8-bit words from the AWG to analog voltages. There are 16 channels in the DAC array. As described in section 3.3.6, 2 channels share the same data bus. A DAC channel consists of a DAC circuit followed by a Butterworth filter to reduce ringing in the output. Figure 3.35 shows the DAC circuit overview. The DAC circuit consists of a Texas Instruments TLC7254 8-bit DAC chips and an op-amp. The Butterworth filter is implemented with 3 op-amps and several resistors and capacitors.

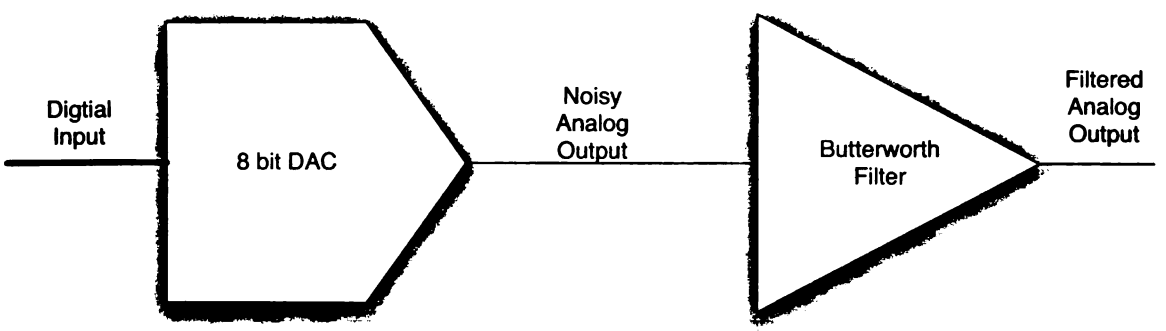


Figure 3.35: DAC with Butterworth filter.

Theory of Operation

The DAC circuit functions as an inverting amplifier as shown in Figure 3.36. The TLC7254 varies the resistance R_{int}, a resistor between a constant reference voltage, and an output pin attached to ground. This varies the current flowing through R_{int} and the feedback resistor R_{fb} and in turn the output voltage. The output of the DAC is hooked up to the negative input of an op-amp with the positive input attached to ground. This provides the ground reference for R_{int}. The output of the op-amp is fed back to the DAC, where an internal resistor connects it to the current output of

R_{fb}. In this configuration, the DAC and op-amp form an inverting amplifier with a constant input voltage and a variable gain. The transfer function of the amplifier is

$$V_{out} = -V_{ref} * \frac{R_{fb}}{R_{int}}. \tag{3.10}$$

The resistance R_{int} is controlled by the digital input. A negative voltage reference is used so that the output varies from 0V to $|V_{int}|$. A potentiometer is placed in series with V_{int} to provide an output range calibration. The potentiometer can increase the resistance between V_{int} and ground, changing R_{int} in the transfer function and thus changing the gain and output voltage range.

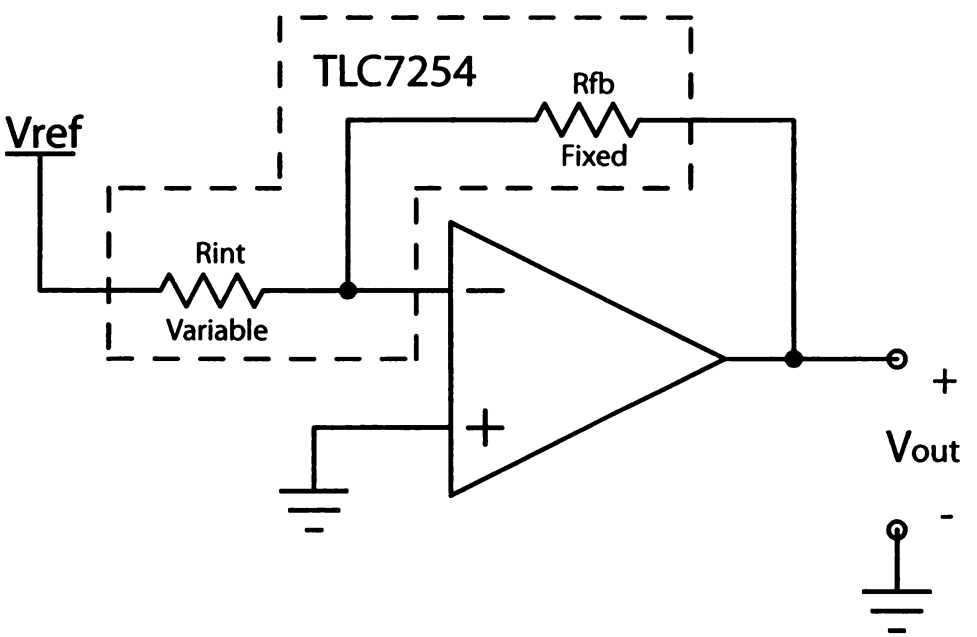


Figure 3.36: DAC simplified schematic

Stability

The output of the DAC circuit was initially unstable because of the high bandwidth of the op-amp. The unstable output is demonstrated in Figure 3.38. A pole was added to the transfer function by adding a 2pF capacitor between the op-amp output and the negative input as shown in Figure 3.37. The stabilized output is shown in Figure 3.39.

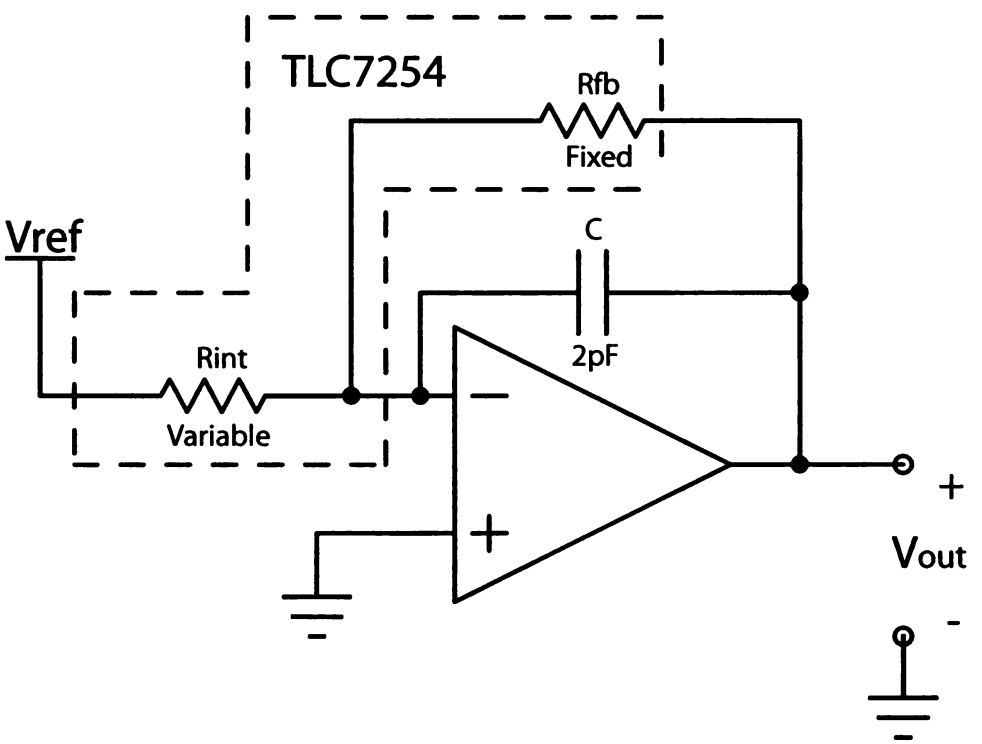


Figure 3.37: DAC circuit with stabilizing capacitor.

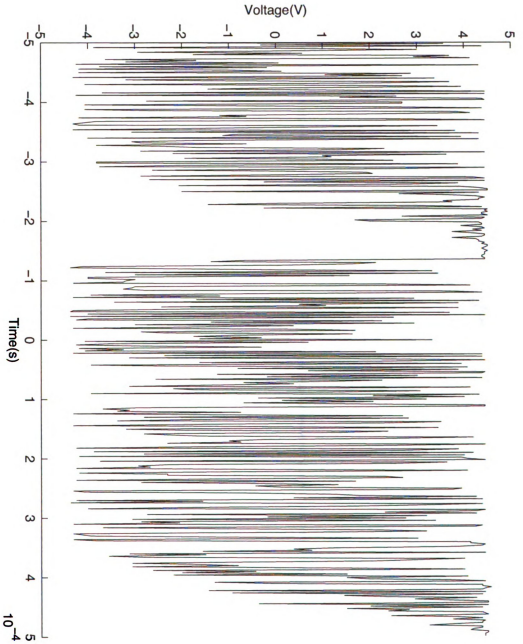


Figure 3.38: Example of DAC instability for a ramp function input.

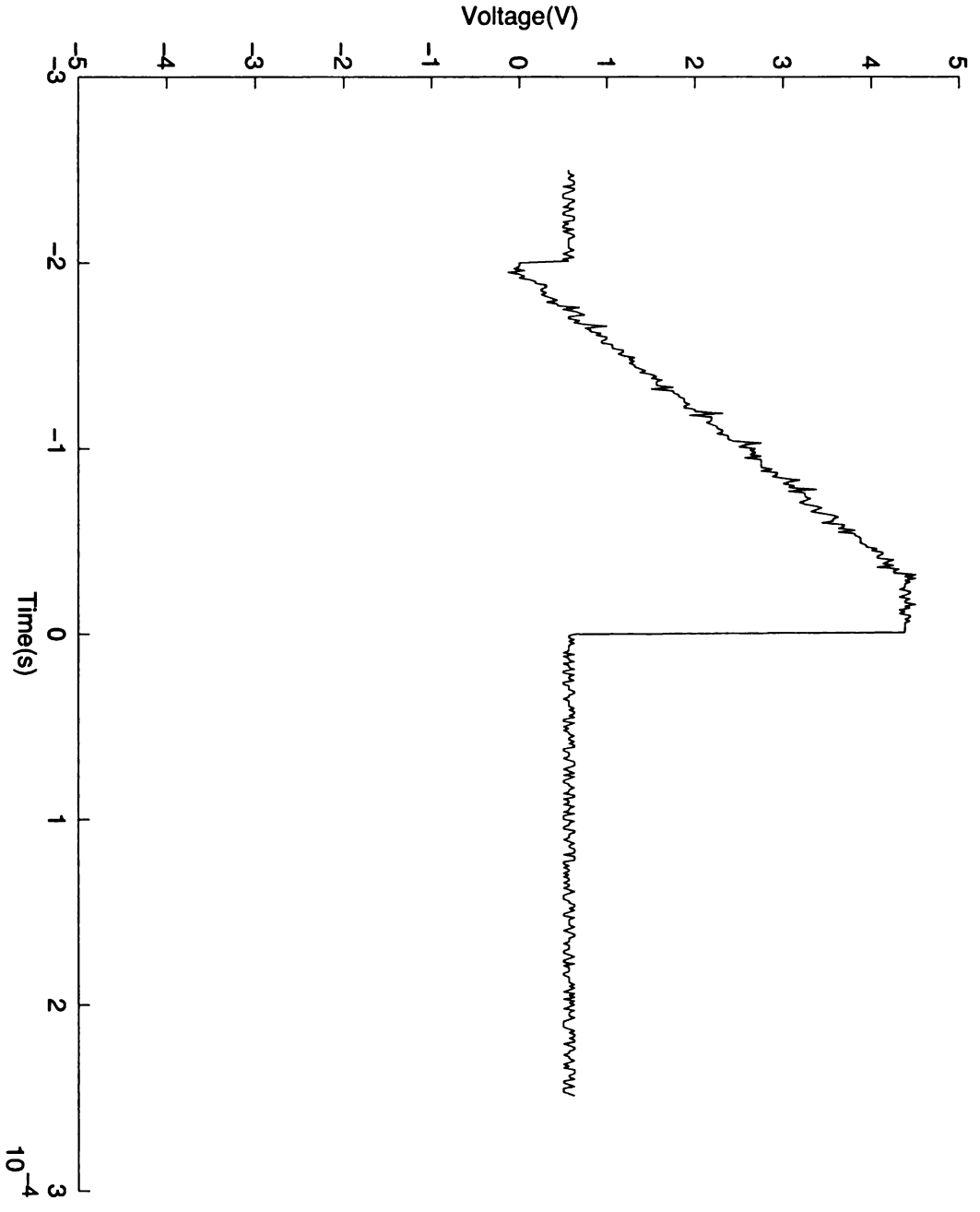


Figure 3.39: Example of stabilized DAC for a ramp function input.

3.4.2 Output Filter

The output of the DAC contains a significant amount of ringing. The ringing is shown in Figure 3.40. A filter was added after the DAC circuit to reduce the ringing on the output. The filter chosen was a Butterworth filter because of the balance between the filter cutoff and step response [21]. A cutoff frequency of 200Hz was chosen and the component values were scaled by 100. A fifth order filter was implemented because a 3rd did not reduce the ringing to less than 15mV. The filtering result with a 3rd order filter is shown in Figure 3.41. The filtering result with a 5th order filter, implemented by cascading 2nd and 3rd order filters, is shown in Figure 3.42. A lookup table was used to determine the values of the components. The schematic for the filter is shown in Figure 3.43. The calculations for the capacitor values are as follows:

$$C_1 = \frac{1.753}{100 * 2 * \pi * 200e3} = 13.9579nF \tag{3.11}$$

$$C_2 = \frac{1.354}{100 * 2 * \pi * 200e3} = 10.7748nF \tag{3.12}$$

$$C_3 = \frac{0.4214}{100 * 2 * \pi * 200e3} = 3.3534nF \tag{3.13}$$

$$C_4 = \frac{3.235}{100 * 2 * \pi * 200e3} = 25.7433nF \tag{3.14}$$

$$C_5 = \frac{0.3089}{100 * 2 * \pi * 200e3} = 2.4581nF \tag{3.15}$$

$$\tag{3.16}$$

The closest standard values for the capacitors are listed in Table 3.4.

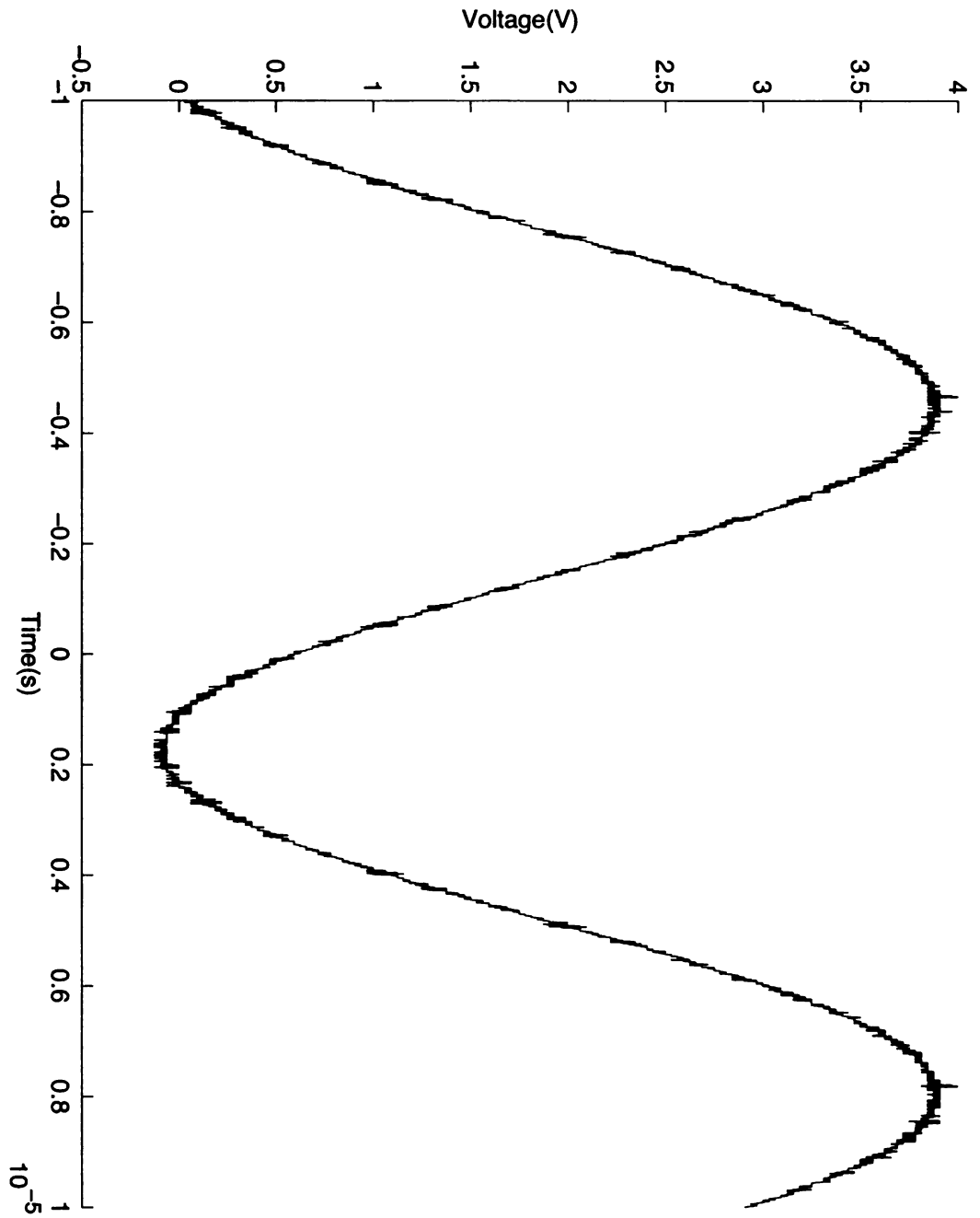


Figure 3.40: DAC output without filtering.

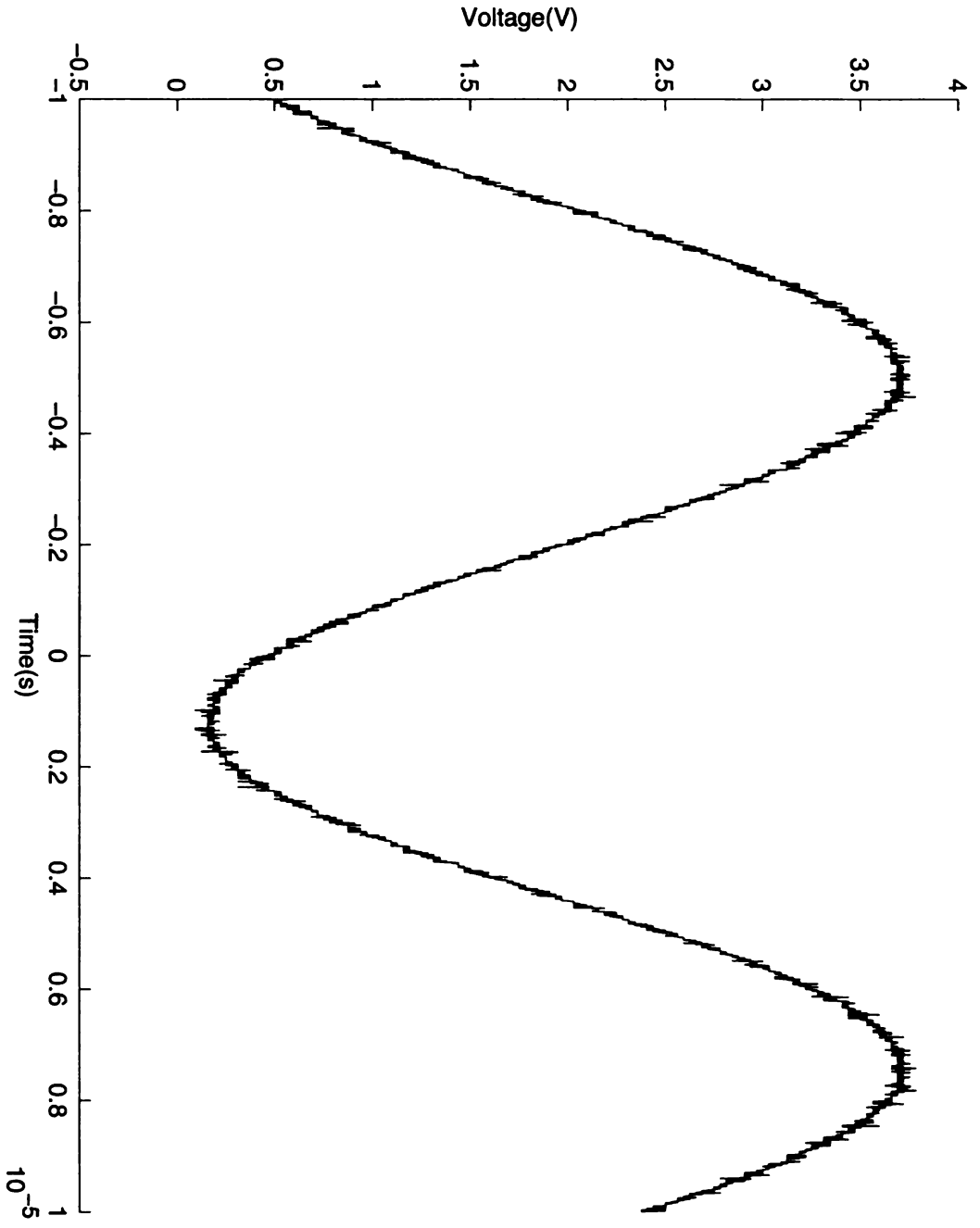


Figure 3.41: DAC output after 3rd order Butterworth filter.

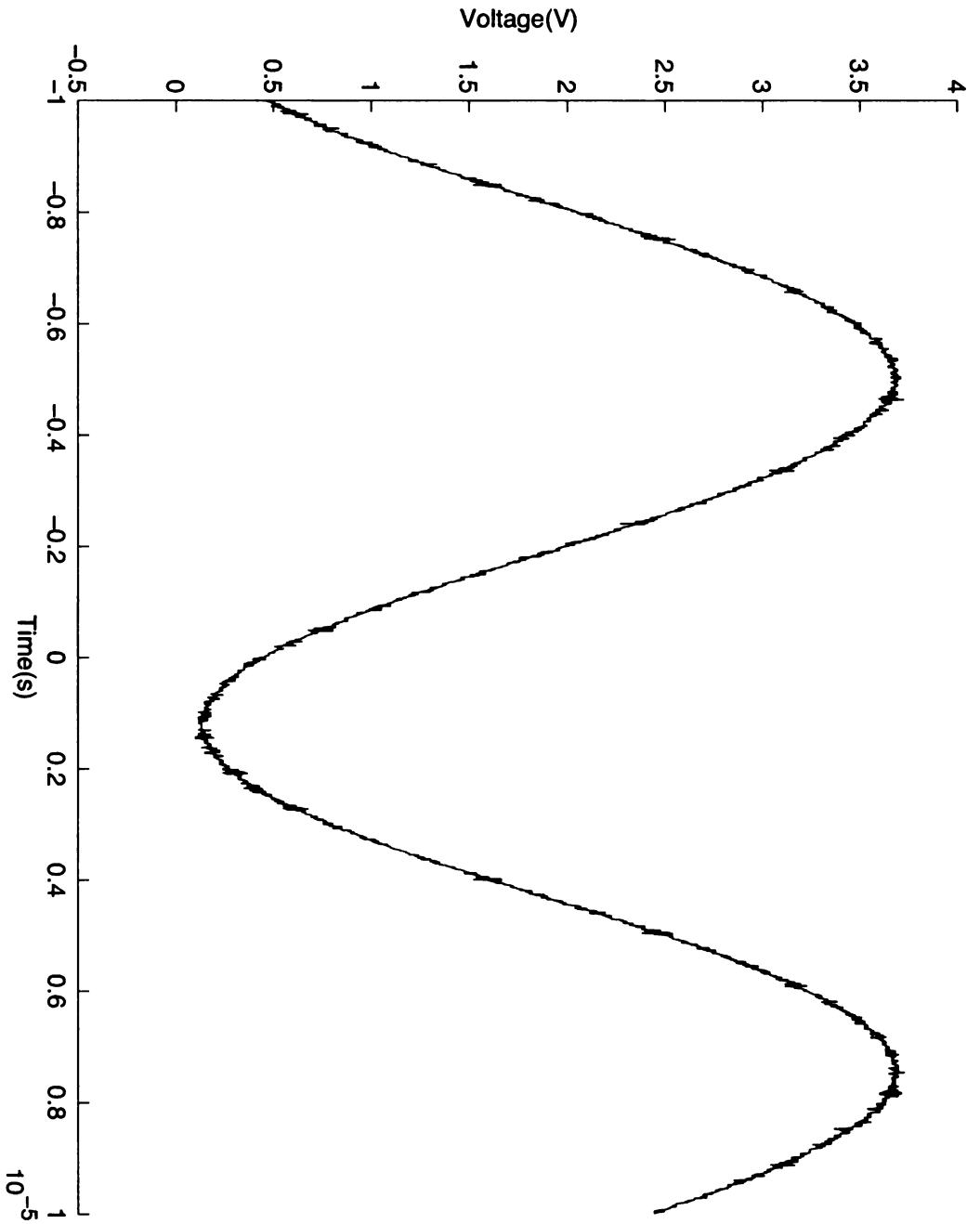


Figure 3.42: DAC output after 5th order Butterworth filter.

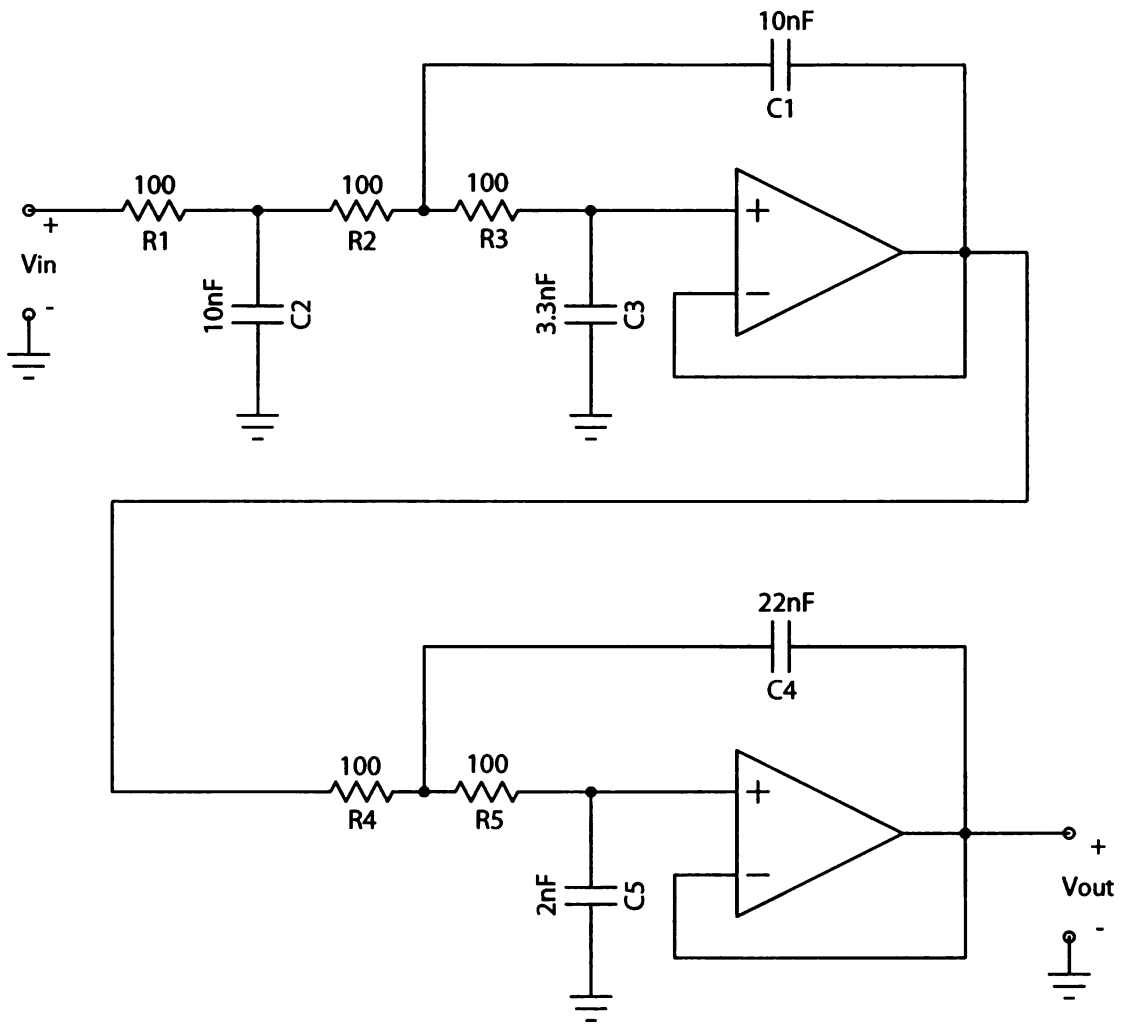


Figure 3.43: Butterworth filter schematic.

Capacitor	Calculated	Standard
C1	13.9579nF	10nF
C2	10.7748nF	10nF
C3	3.3534nF	3.3nF
C4	25.7433nF	22nF
C5	2.4581nF	2nF

Table 3.4: Butterworth filter standard capacitor values.

Implementation

The DAC board schematics were captured using Cadence® design tools. The design was implemented in a hierarchical fashion in order to be able to reuse as much work between the channels as possible. The hierarchical design meant that the schematic for the DAC and Butterworth filter only had to be entered once and instanced for each channel. It was also possible to reuse or copy the layout of the first channel for the remaining channels. The DAC circuit boards were designed to fit in a rack mount cage that had 6.25 inch tall slots. Only 6 channels could be fit into this space, therefore two boards were designed: a 2-channel board that utilized the connection to the AWG with connectors J4 and J5 since they had only enough pins for one channel pair each, and a 6-channel board that utilized J3 and J6 on the AWG because those connectors had enough pins for 3 channel pairs each. The layout of the 2-channel DAC board is shown in Figure 3.44, and the layout for the 8-channel DAC board is shown in Figure 3.45. The assembled 2-channel board is shown in Figure 3.4.2, and the assembled 6-channel board is shown in Figure 3.4.2.

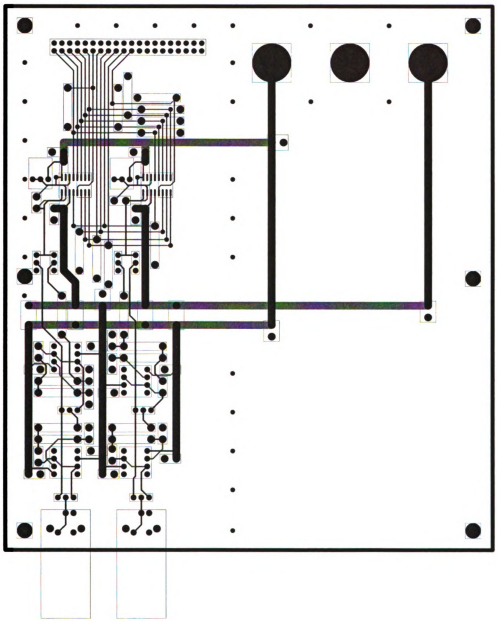


Figure 3.44: 2-channel DAC layout.

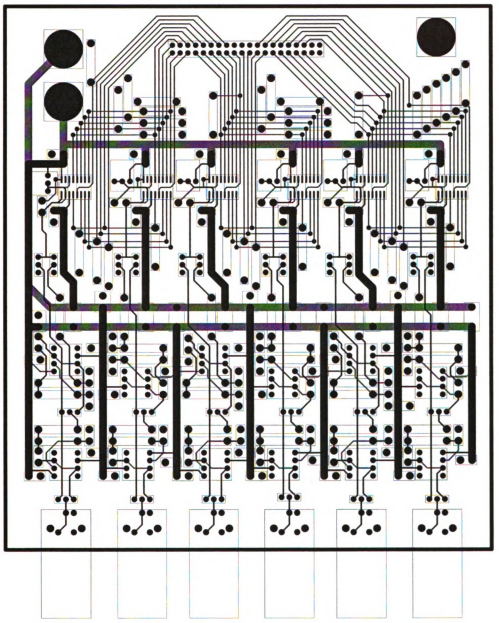
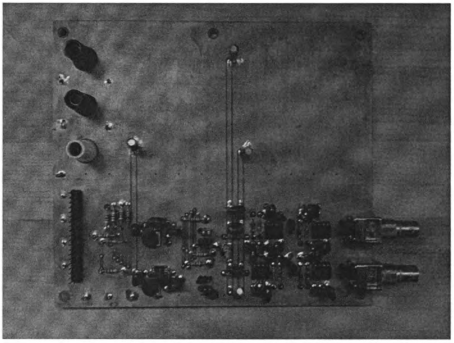
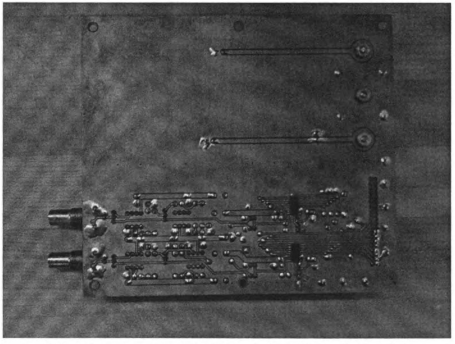


Figure 3.45: 6-channel DAC layout

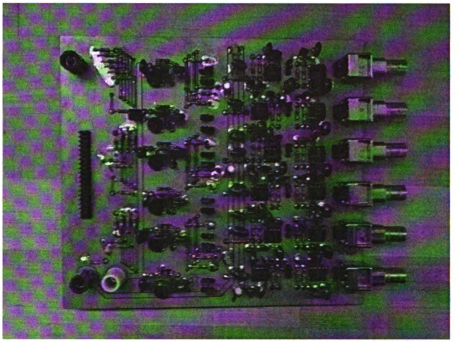


(a) 2-channel assembled DAC board viewed from the top.

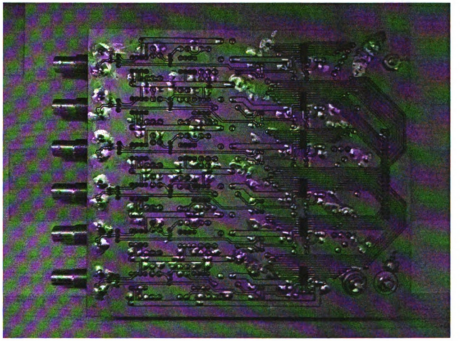


(b) 2-channel assembled DAC board viewed from the bottom.

Figure 3.46: 2-channel DAC board.



(a) 6-channel assembled DAC board viewed from the top.



(b) 6-channel assembled DAC board viewed from the bottom.

Figure 3.47: 2-channel DAC board.

3.5 Amplifier

The power amplifier takes the signal from the analog output of the DAC and provides a voltage and current gain so that the output of the power amplifier can drive up to 25W into a 50Ω load.

3.5.1 Types of Power Amplifiers

There are many different types of amplifiers with tradeoffs between signal integrity, efficiency and design complexity. The amplifier types are summarized in Table 3.5.

Class	Complexity	Signal Integrity	Efficiency
A	Low	High	Low
B	Low	Low	Medium
C	Low	Low	High
Complementary Pair B	Medium	Medium	Medium
D	High	High	High
AB	High	High	Medium

Table 3.5: Amplifier classification summary.

Class A amplifiers are simple with great signal integrity but poor efficiency. They have good signal integrity because they pass the whole signal without the transistor changing its region of operation. Class A has poor efficiency because the transistor is biased so that it stays on when there is no signal. A Class A amplifier is shown in Figure 3.48.

Class B amplifiers are as simple and are more efficient than Class A but have poor signal integrity. Class B amplifiers are simple because have the same components as Class A. The difference between Class A and B amplifiers is how the transistors are biased. Class B are more efficient than Class A because when there is no signal the transistor does not draw any current. Class B has poor signal integrity because it only passes half of the input signal. A Class B amplifier is shown in Figure 3.49.

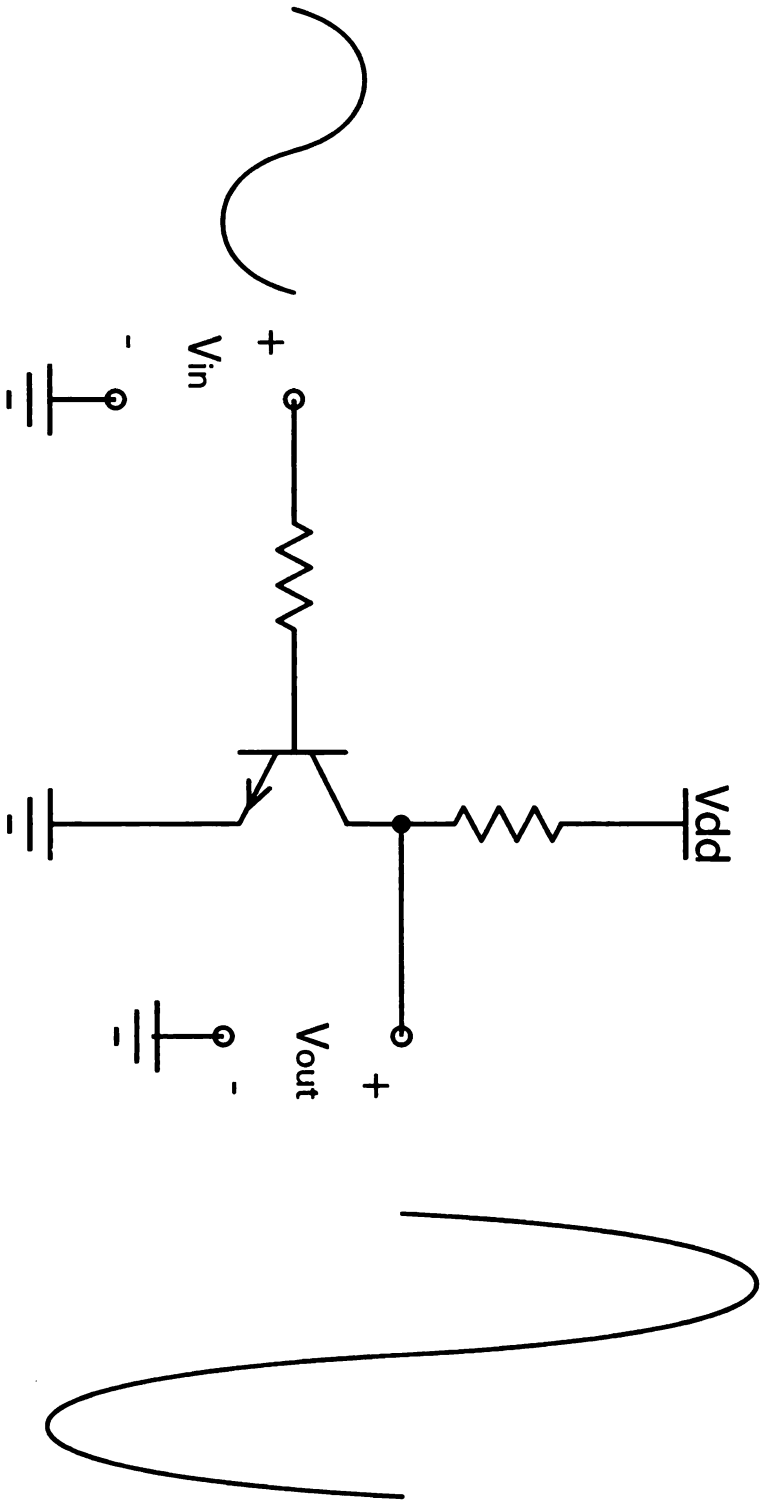


Figure 3.48: Class A amplifier.

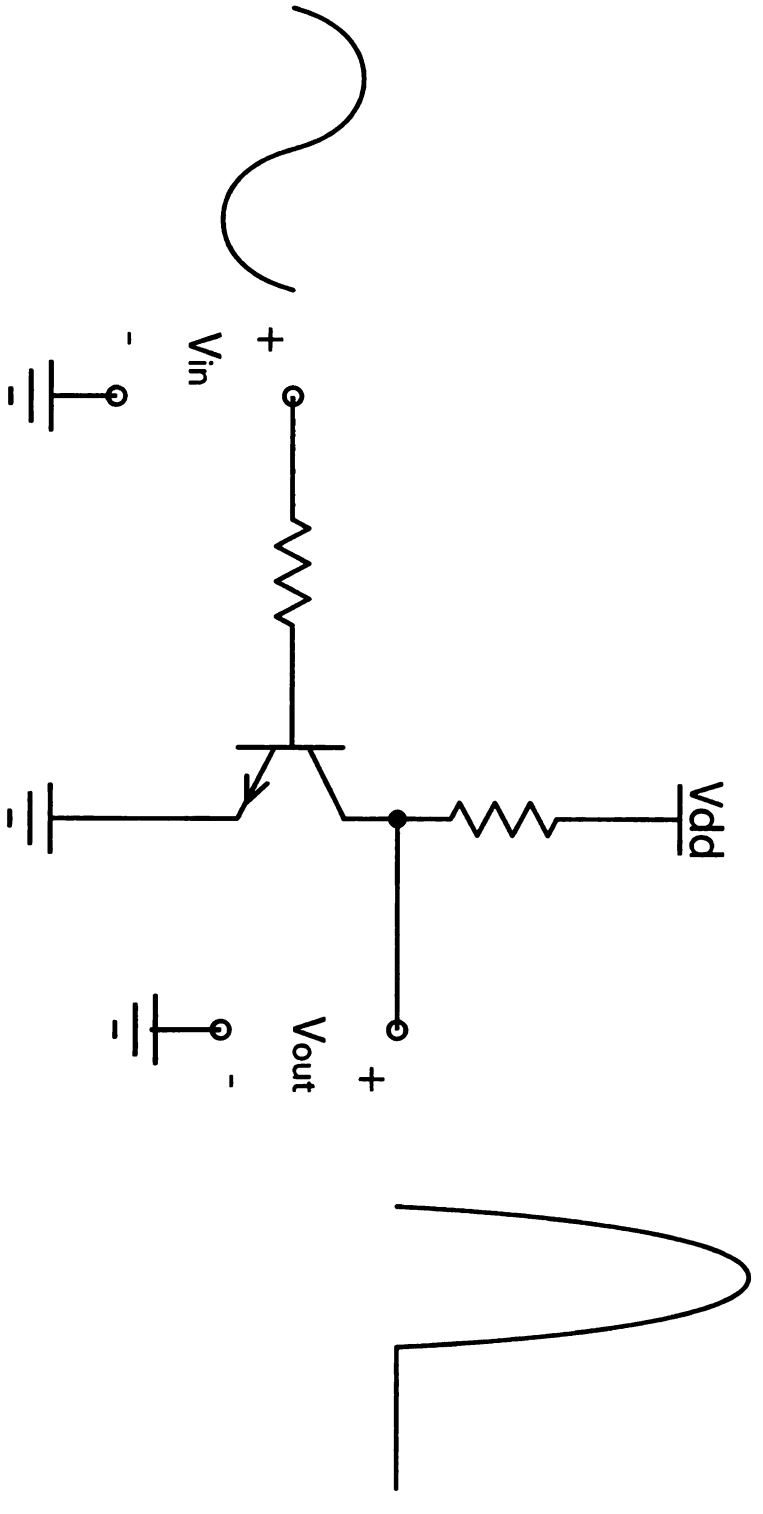


Figure 3.49: Class B amplifier.

Class C are as simple as the Class A and B amplifiers with high efficiency and poor signal integrity. Again, the difference is how the transistor is biased. They are similar to Class B in that they turn on when the input exceeds a certain threshold voltage. They pass less than half the input signal so their signal integrity is poor. They have high efficiency becasue they pass so little of the input signal. A Class C amplifier is shown in Figure 3.50.

A Complementary Pair Class B amplifier design is more complex than a Class B but has similar efficiency and improved signal integrity. A Complementary Pair Class B amplifier consists of two Class B amplifiers placed in series between the power supplies. One amplifier passes the positive half of the signal and the other passes the negative half of the signal. Ideally, the two amplifiers would pass exactly half of the input signal, but because of the voltage needed to turn on transistors, there is a dead zone around zero input where neither amplifier follows the input. This effect is known as cross over distortion. This can be minimized with providing high gain feedback from the output of the amplifier to the input. This reduces the input voltage range that causes no change at the output. A Complementary Pair Class B amplifier is shown in Figure 3.51.

Class D is a complex design with good signal integrity and efficiency. Class D uses the concept of pulse width modulation. The output transistors pull the output to V_{supply} or 0V. The output is averaged over time, so for a higher signal, a higher duty cycle is used. Likewise, for a lower output value, a lower duty cycle is used. The output needs to be filtered in order to average the signal and to eliminate the undesired higher frequencies that arise from switching. A Class D amplifier is shown in Figure 3.52.

Class AB is a complex design with good signal integrity and efficiency. Class AB is a combination of Class A and Class B. Class AB is a complementary pair configuration where one amplifier drives the output when the signal is positive and

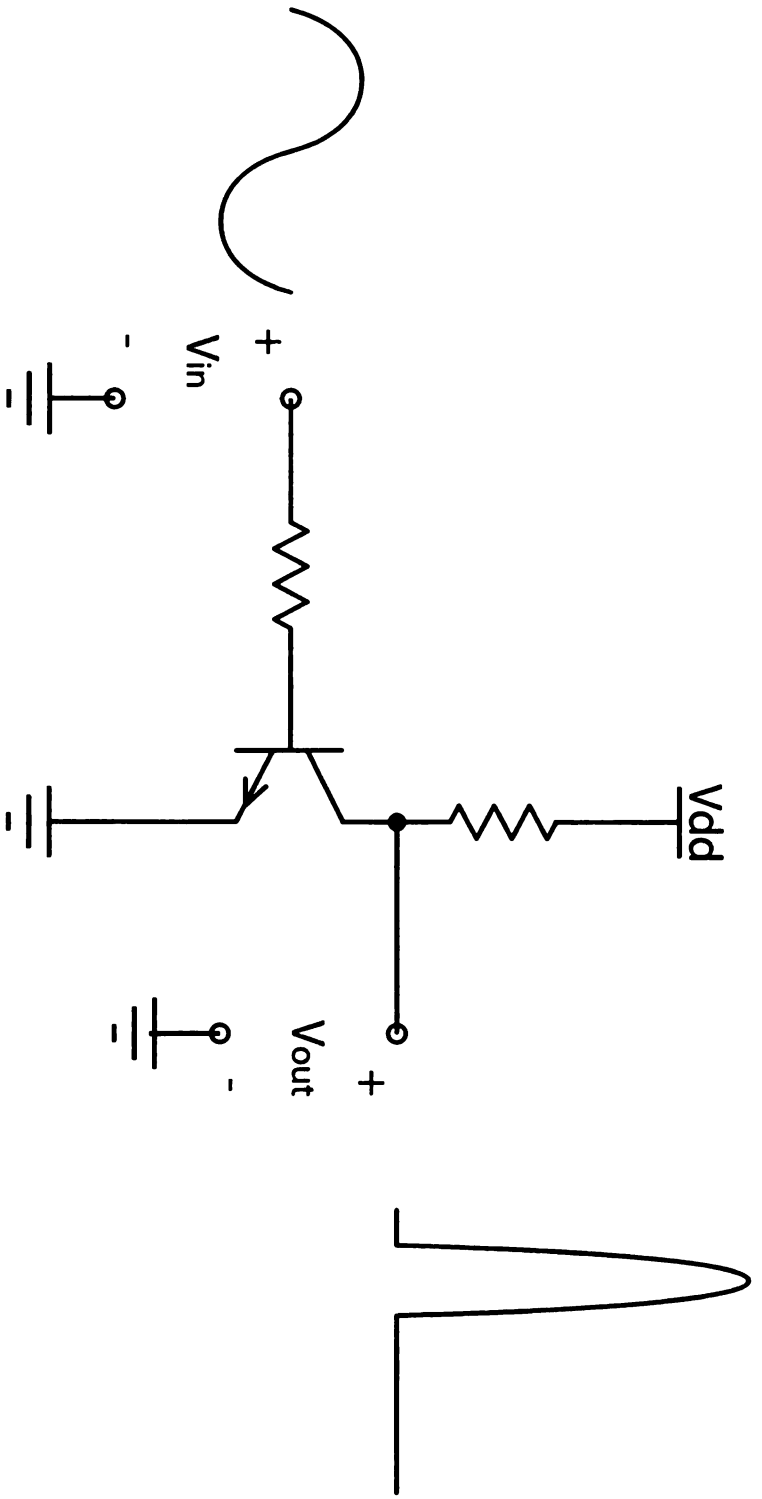


Figure 3.50: Class C amplifier.

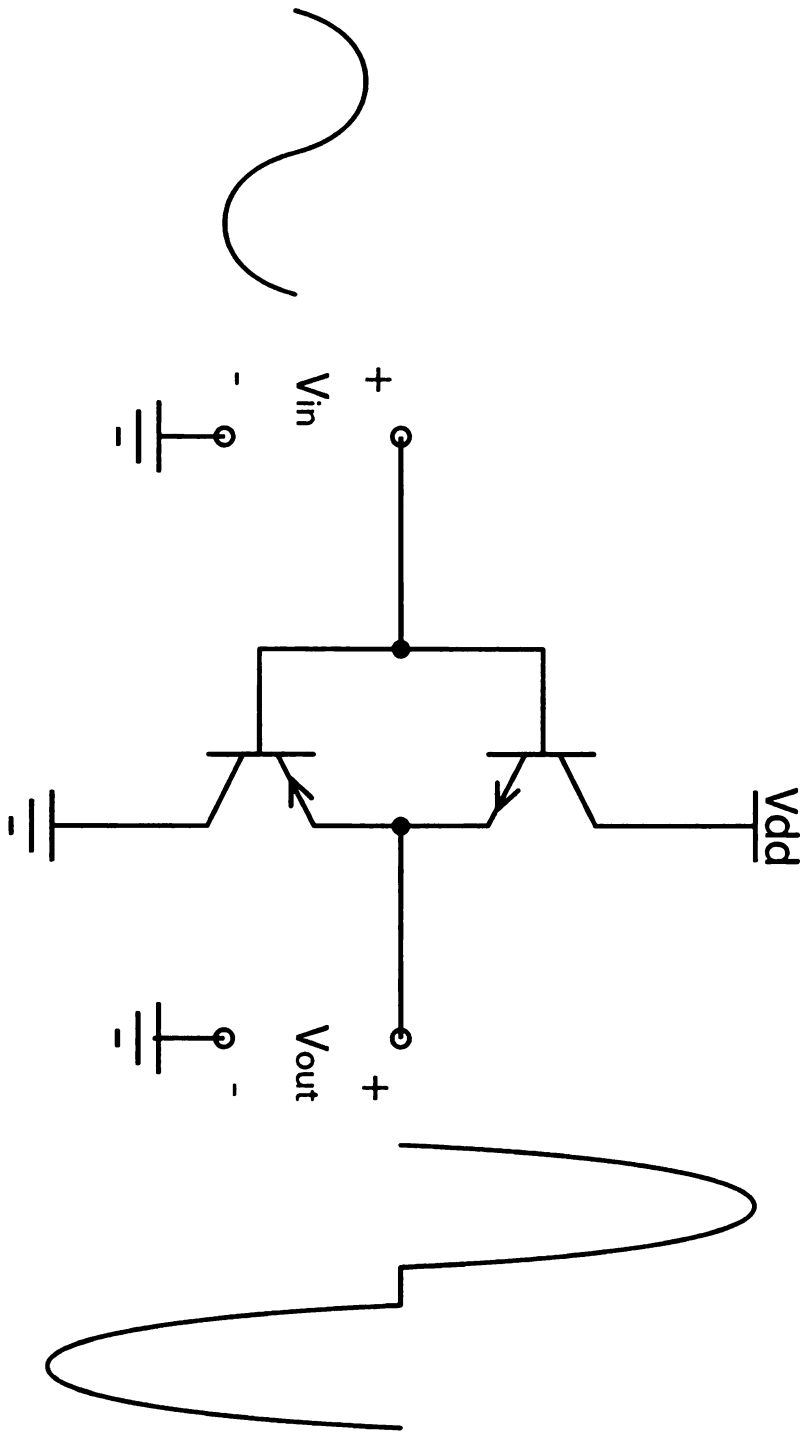


Figure 3.51: Complementary Pair Class B amplifier.

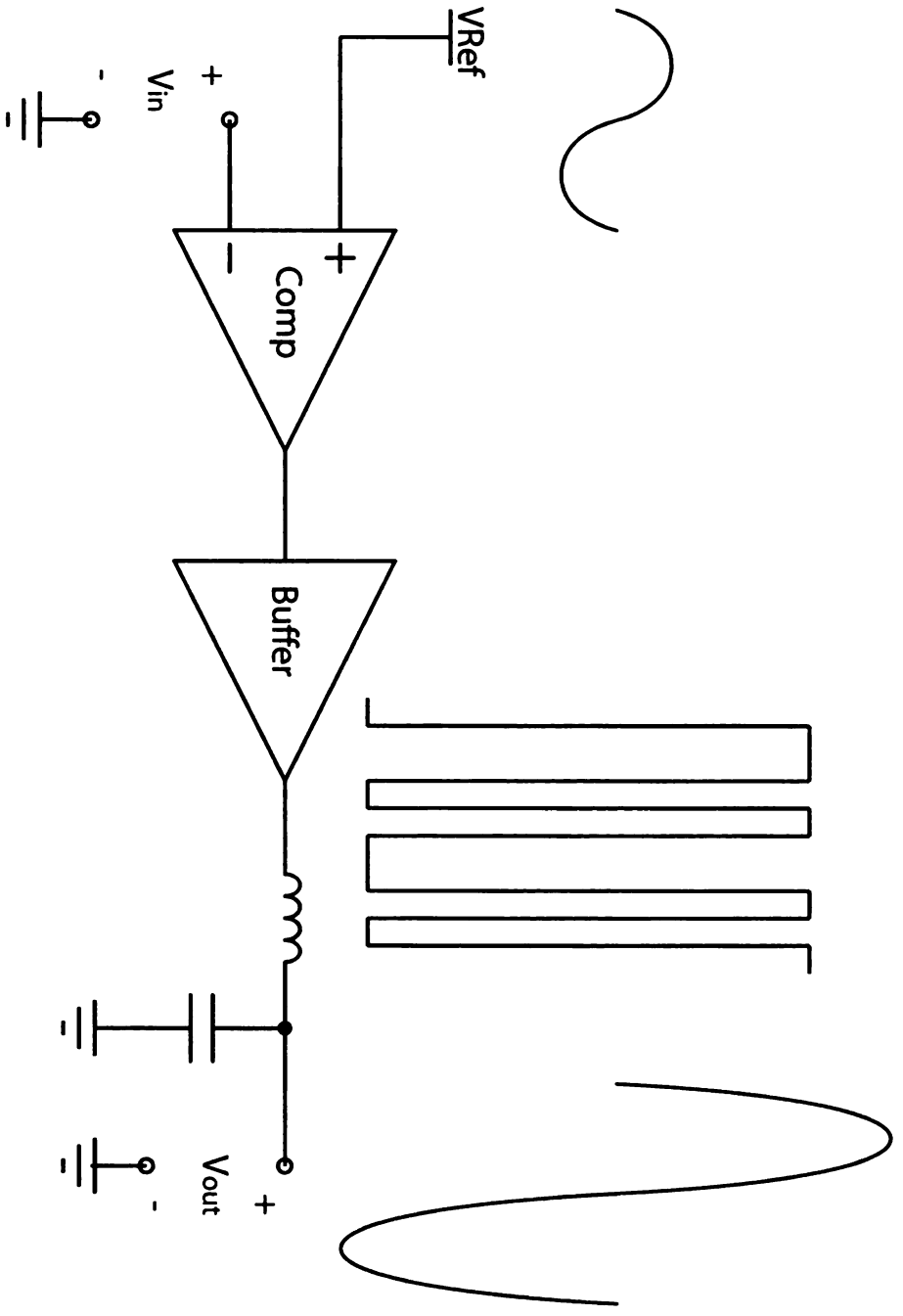


Figure 3.52: Class D amplifier.

the other drives the output when the signal is negative. Both amplifiers conduct a small amount of current when there is no signal. This eliminates the cross over distortion but decreases the amplifier efficiency. Also, a Class AB amplifier needs a biasing network that dissipates additional energy. Since Class AB is a linear amplifier and not a switching amplifier like Class D, there is not any switching noise that needs to be filtered. A Class AB amplifier is shown in Figure 3.53.

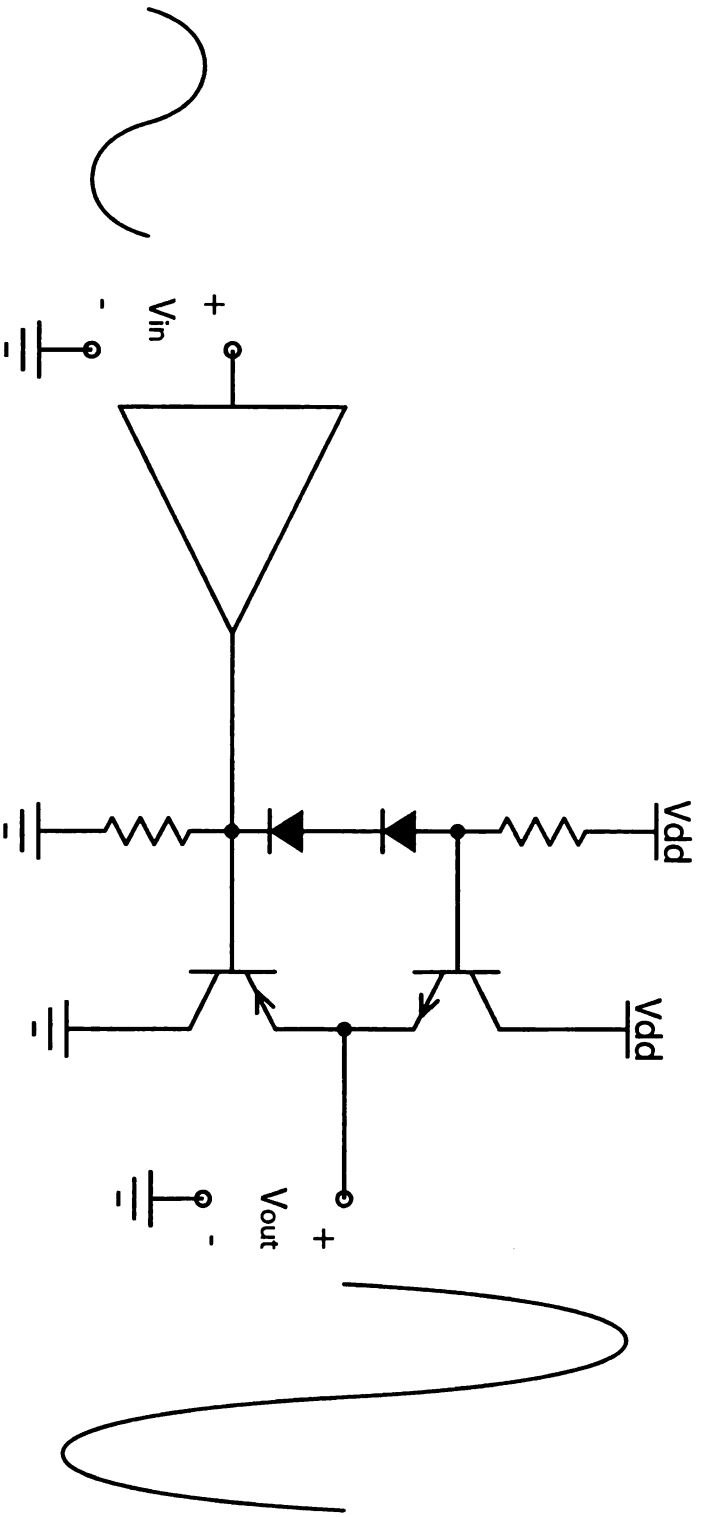


Figure 3.53: Class AB amplifier.

3.5.2 Amplifier Design

A Class AB amplifier was implemented because of the improved signal integrity with moderate efficiency and design complexity and the lack of harmonic noise from a switching design. The load is assumed to be 50Ω and the largest desired output is 25W when driven by a continuous sine wave. In order to get 25W out of 50Ω, a $70V_{rms}$ or $100V_{pk-pk}$ sine wave is needed.

In order to maximize the voltage output, two amplifiers are connected together in a bridge topology instead of a single ended configuration. This topology in shown in Figure 3.54. Instead of attaching one side of the load to ground, one amplifier output is attached to one side of a load and another amplifier, with a gain of -1 relative to the first amplifier, is attached to the other side of the load. This allows for twice the voltage to be applied across the load for a given power supply voltage and eliminates any DC offset on the output.

Each of the amplifiers has two parts: a voltage gain stage and a current gain stage. The voltage gain stage amplifies the input voltage, where the input voltage is at most $1V_{pk-pk}$. In order to obtain a $100V_{pk-pk}$ output across the load, each amplifier in the bridge must produce $50V_{pk-pk}$. With an input of $1V_{pk-pk}$, a gain of 50 is needed to produce $50V_{pk-pk}$. An op-amp is used to provide the voltage gain because op-amps are inexpensive and small , and they admit straightforward designs. The OPA552 is chosen because of its availability, cost, and performance. It has a bandwidth of 12MHz, a large supply voltage range of $\pm 30V$, a large output current, and an available spice model. The large bandwidth facilitate the design of this 80kHz amplifier. The large supply range makes a $100V_{pk-pk}$ output possible, even with dropout voltage across the op-amp and output transistors, and the ability to simulate the circuit in Spice allows for fast design refinement with fewer design cycles.

The current gain stage consists of a biasing network and a complimentary pair of

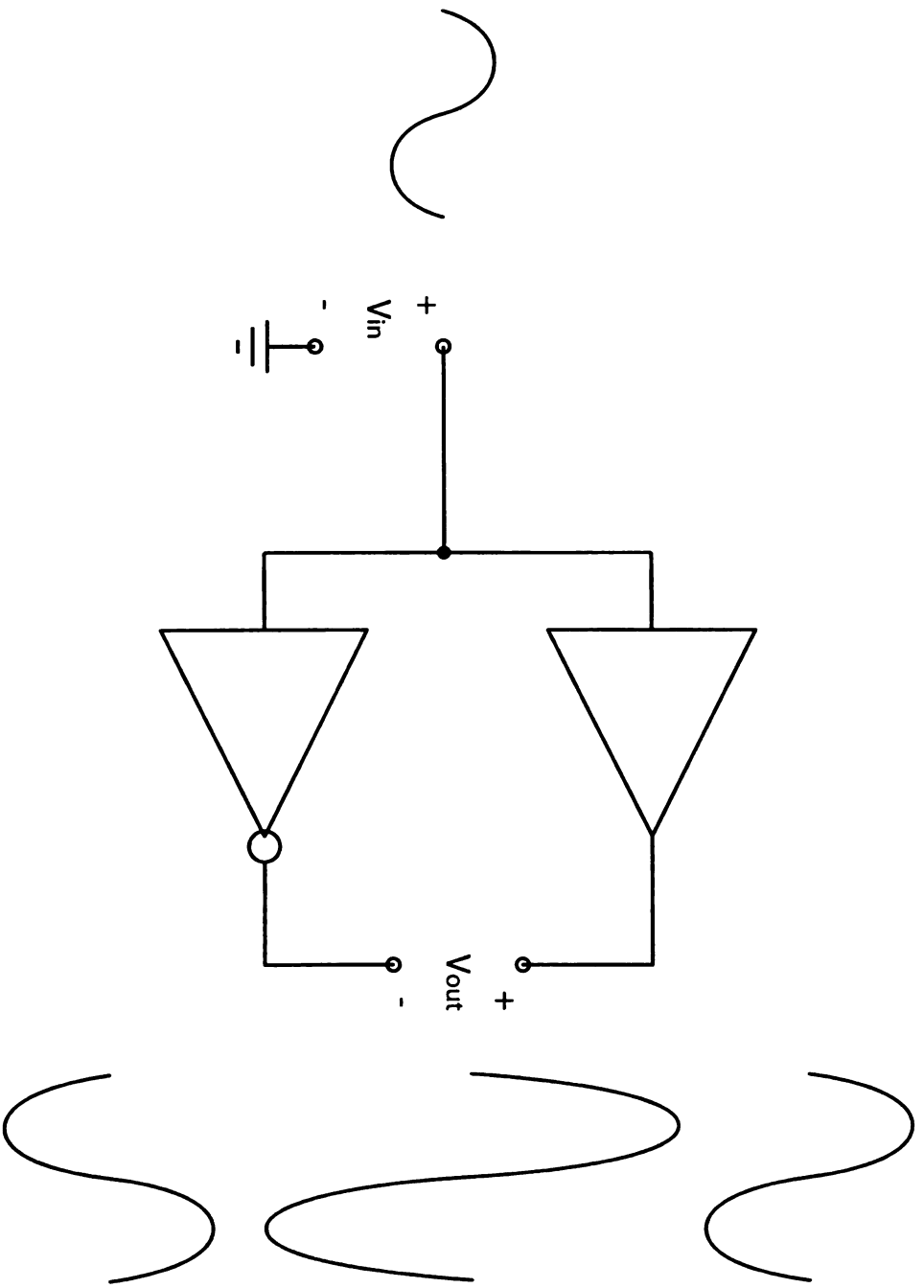


Figure 3.54: Bridged amplifier topology.

power transistors. The biasing network makes the amplifier Class AB. The network biases the output transistors so that each transistor is conducting a small amount of current when the input is 0V. This configuration consumes energy all of the time, even without a signal, but signal distortion is minimized. The output transistors, MJH11021(PNP) and MJH11022(NPN), are a complimentary pair of Darlington BJTs. The complimentary pair aspect of the transistors means that they have similar characteristics, such as base-emmitter on voltage (V_{BE-on}) and current gain, so they can be easily used in a complementary pair topology and the waveform will still be symmetric. The Darlington aspect of the BJTs means that two BJTs are cascaded as shown in Figure 3.55. This increases the current gain dramatically, which is needed in a power amplifier. The output of the op-amp can supply 200mA of current. The current needed for a $50V$ signal across a 50Ω load is 1A, therefore a current gain of 5 is needed. This is well within their minimum rated AC current gain of 75.

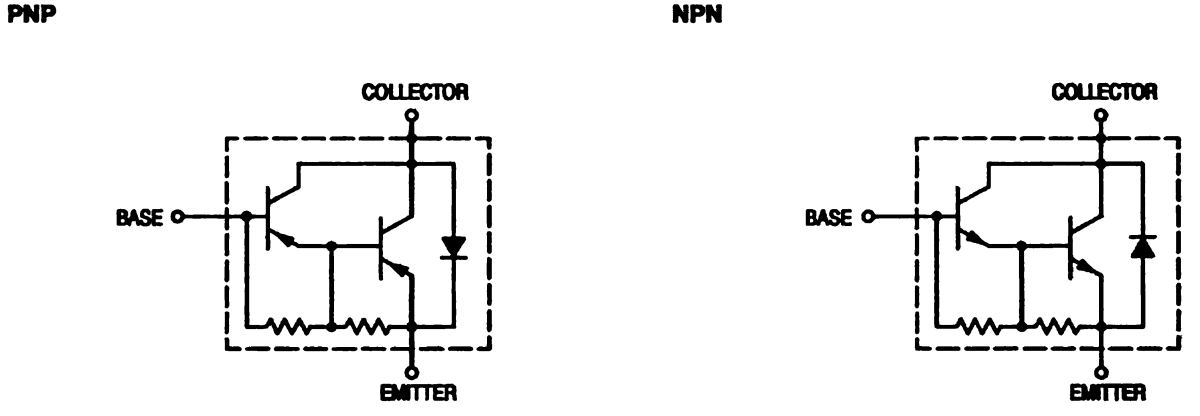


Figure 3.55: Darlington pair transistors [18]

The complete amplifier design for the inverting amplifier is shown in Figure 3.56 and the non-inverting amplifier is shown in Figure 3.57. The amplifiers are attached to the load as shown in Figure 3.54. The layout of the amplifier is shown in Figure 3.58. The assembled board is shown in Figure 3.59.

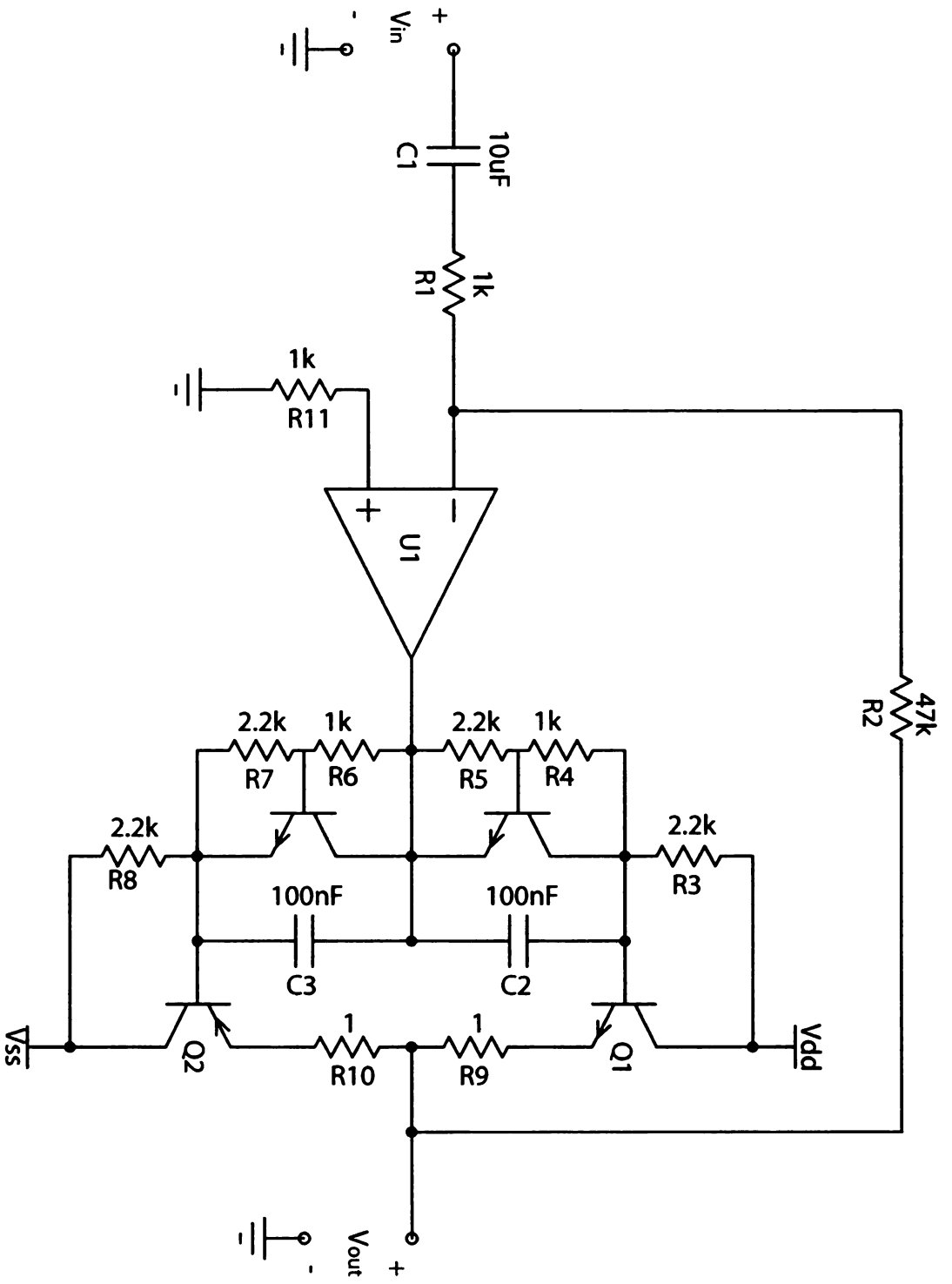


Figure 3.56: Inverting amplifier design.

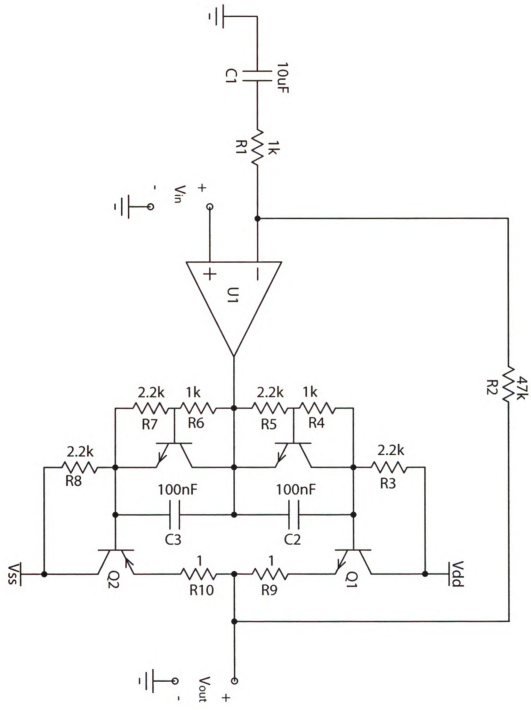


Figure 3.57: Non-inverting amplifier design.

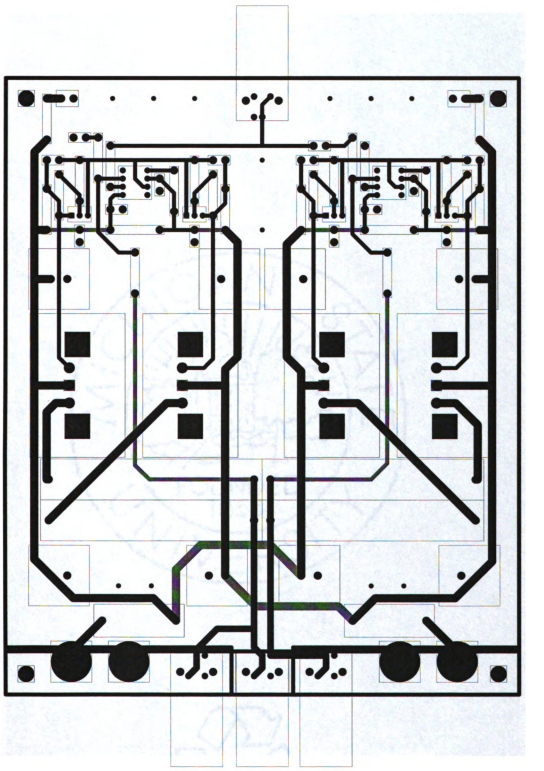


Figure 3.58: Amplifier layout.

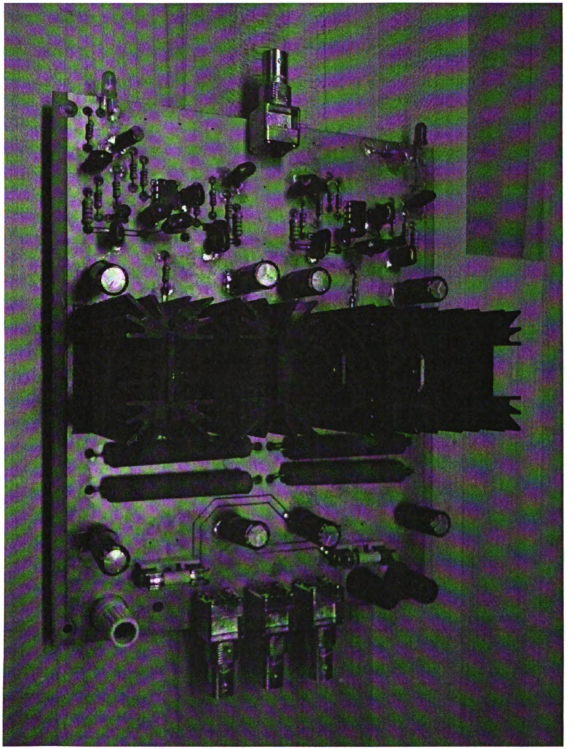


Figure 3.59: Assembled amplifier.

3.5.3 Stability

Circuit stability is analyzed using Rate of Closure (ROC) [21] which predicts the stability of an amplifier from inspecting the open loop gain of the amplifier and the gain of the feedback network. The transfer function of an op-amp circuit with feedback is shown in Equation (3.17).

$$\frac{V_{out}}{V_{in}} = K\frac{1}{1+\frac{1}{A_{dm}\beta}} \tag{3.17}$$

K is the ideal gain. The values of K for different topologies are shown in Table 3.6. The op-amp topologies are shown in figures 3.60 - 3.62. The error multiplier (EM) is defined in Equation (3.18).

$$EM = \frac{1}{1+\frac{1}{A_{dm}\beta}} \tag{3.18}$$

A_{dm} is the open loop gain of the op-amp. β is the gain of the feedback network from the output of the op-amp to the input terminals of the op-amp as shown in figures 3.63 - 3.65. β is defined for each topology in Equation (3.19).

$$\beta = \frac{V_2}{V_1} = \frac{Z_1}{Z_1+Z_2} \tag{3.19}$$

The term $A_{dm}\beta$ is the loop gain G. The phase margin (PM) is defined as $180° - \angle|G|$ at the frequency f_c where $|G| = 1$. If $PM \leq 0°$ then the amplifier is unstable. If $0° < PM < 45°$ then the amplifier is marginally stable. If $PM \geq 45°$ then the amplifier is stable [21].

PM can be predicted by inspecting the magnitude Bode plots of A_{dm} and $\frac{1}{\beta}$. Each pole of A_{dm} and and zero of $\frac{1}{\beta}$ decrease PM by 90°. Each pole of A_{dm} changes the slope of the magnitude Bode plot of A_{dm} by $-20°$ and each zero of $\frac{1}{\beta}$ changes the

slope of the magnitude Bode plot of $\frac{1}{\beta}$ by $+20°$. Taking the difference in the slopes of the magnitude Bode plots of A_{dm} and $\frac{1}{\beta}$ at f_c will determine how many poles and zeros and are in A_{dm} and $\frac{1}{\beta}$ up to f_c and determine PM. This difference is the ROC and is shown in Equation (3.20). Figure 3.66 shows an example plot of $|A_{dm}|$ and $\frac{1}{\beta}$ for different feedback networks and the corresponding ROC and PM values. An example plot for an unstable amplifier is shown in Figure 3.67 and Figure 3.68 shows the Bode plot for a stabilized amplifier. An amplifier is stable if $ROC < 40dB/dec$ because $PM \geq 45°$ [21].

$$\begin{aligned} ROC &= slope\left|\frac{1}{\beta}\right| - slope|A_{dm}| \\ \text{Evaluated at } f_c &\rightarrow |A_{dm}| = \left|\frac{1}{\beta}\right| \end{aligned} \tag{3.20}$$

Topology	**Ideal Gain(K)**
Inverting	$\frac{-Z_2}{Z_1}$
Non inverting	$1+\frac{Z_2}{Z_1}$
Differential	$\frac{Z_2}{Z_1}$

Table 3.6: Ideal gain of common op-amp topologies.

The ROC for the designed amplifier is $20dB/dec$ and is shown in Figure 3.69.

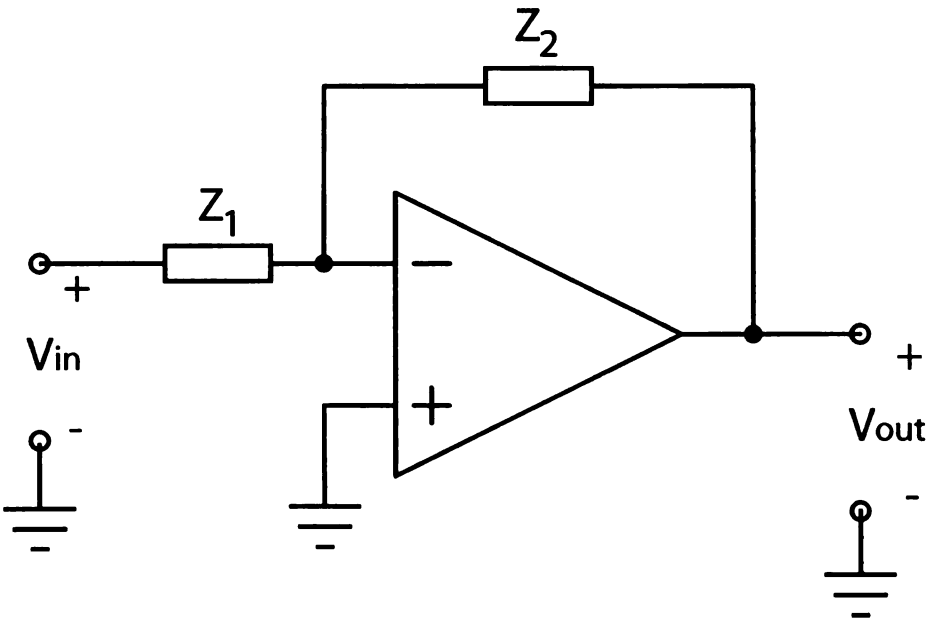


Figure 3.60: Inverting amplifier topology.

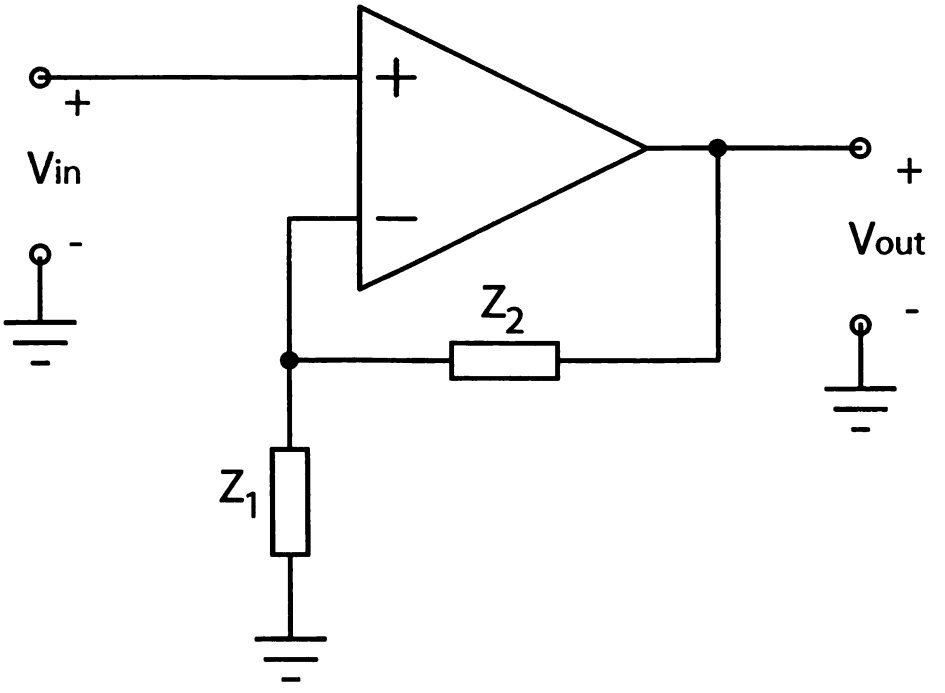


Figure 3.61: Non inverting amplifier topology.

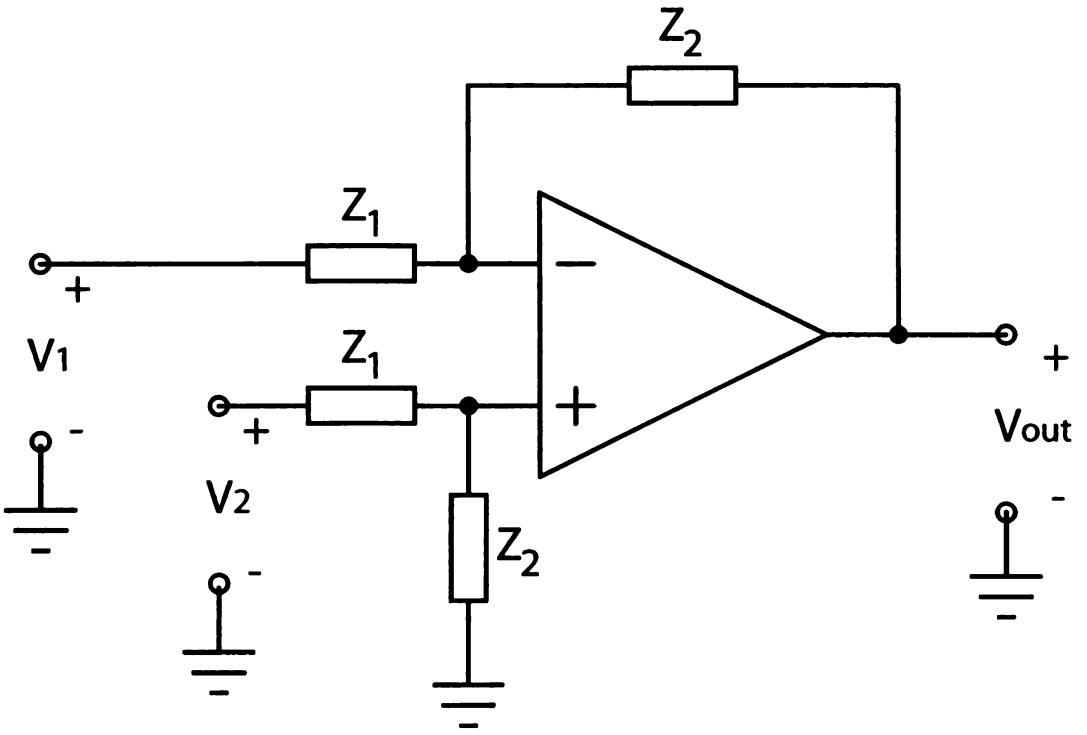


Figure 3.62: Differential amplifier topology.

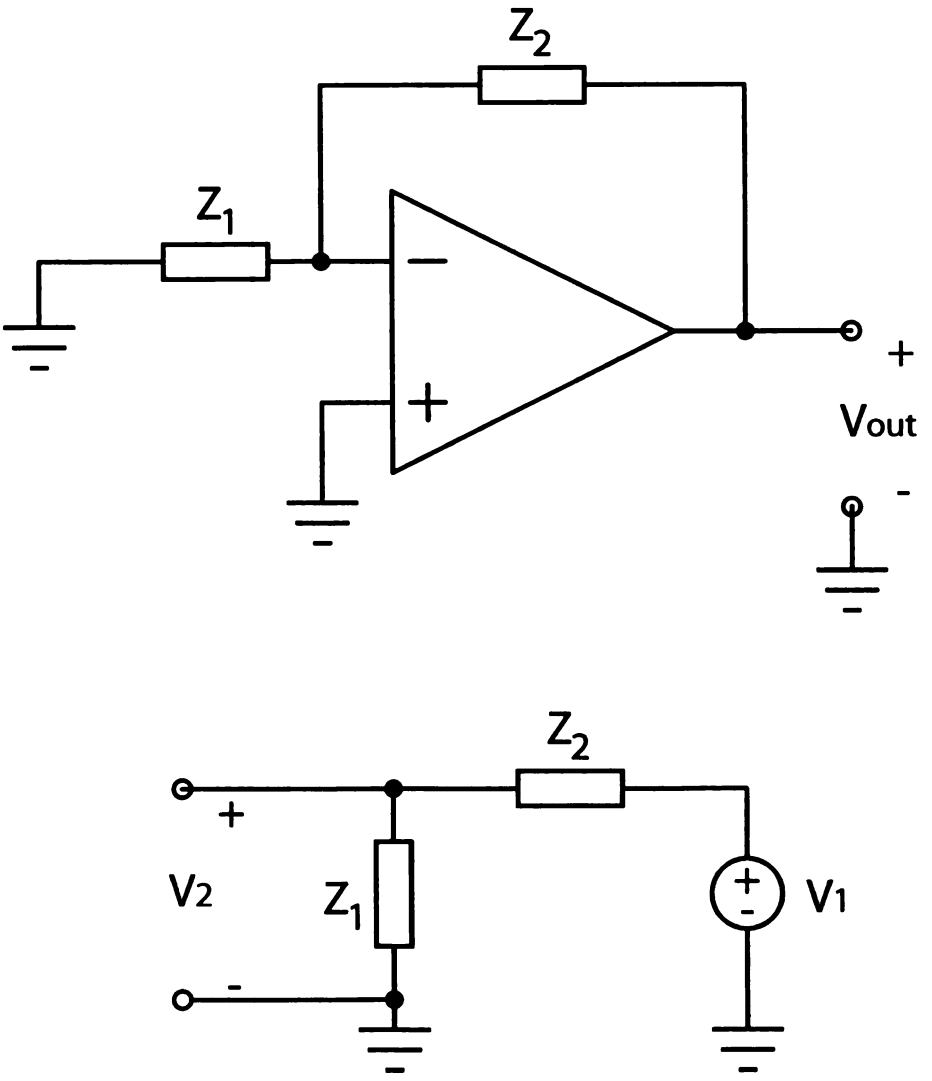


Figure 3.63: Inverting amplifier β network.

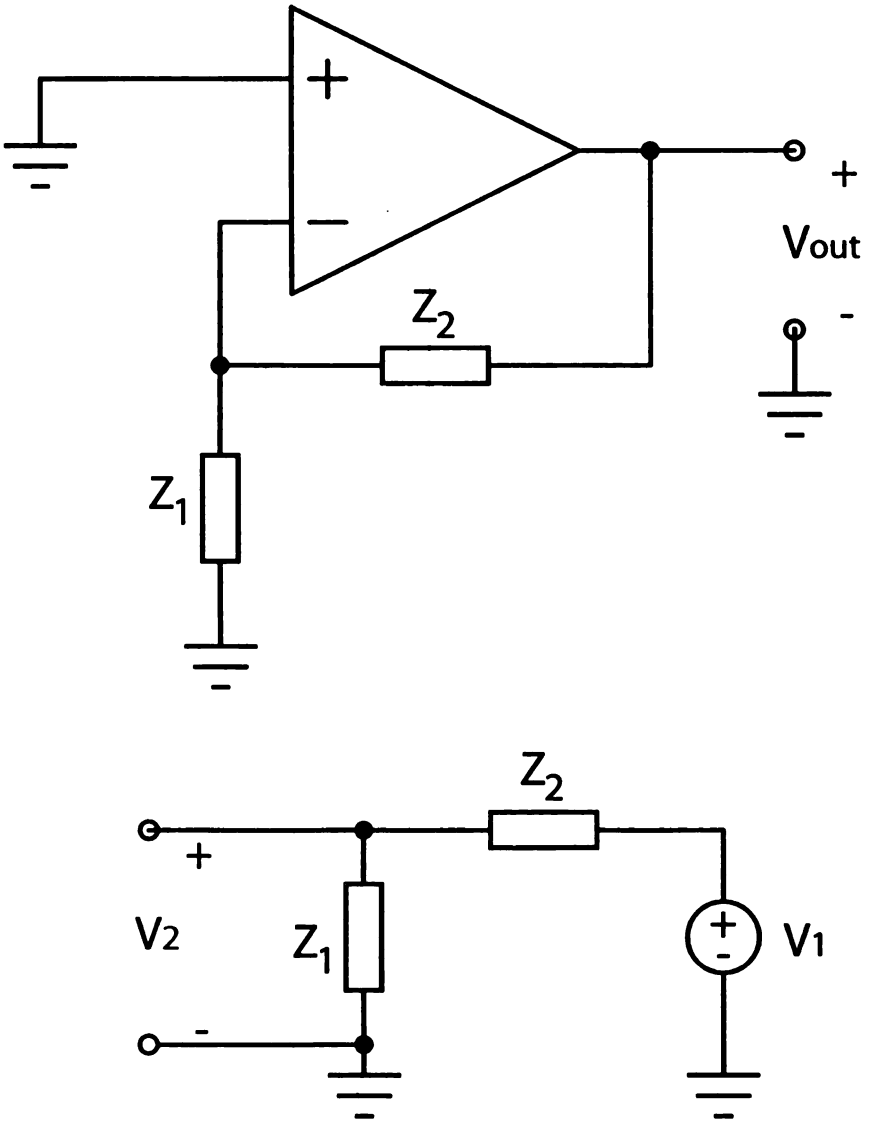


Figure 3.64: Non inverting amplifier β network.

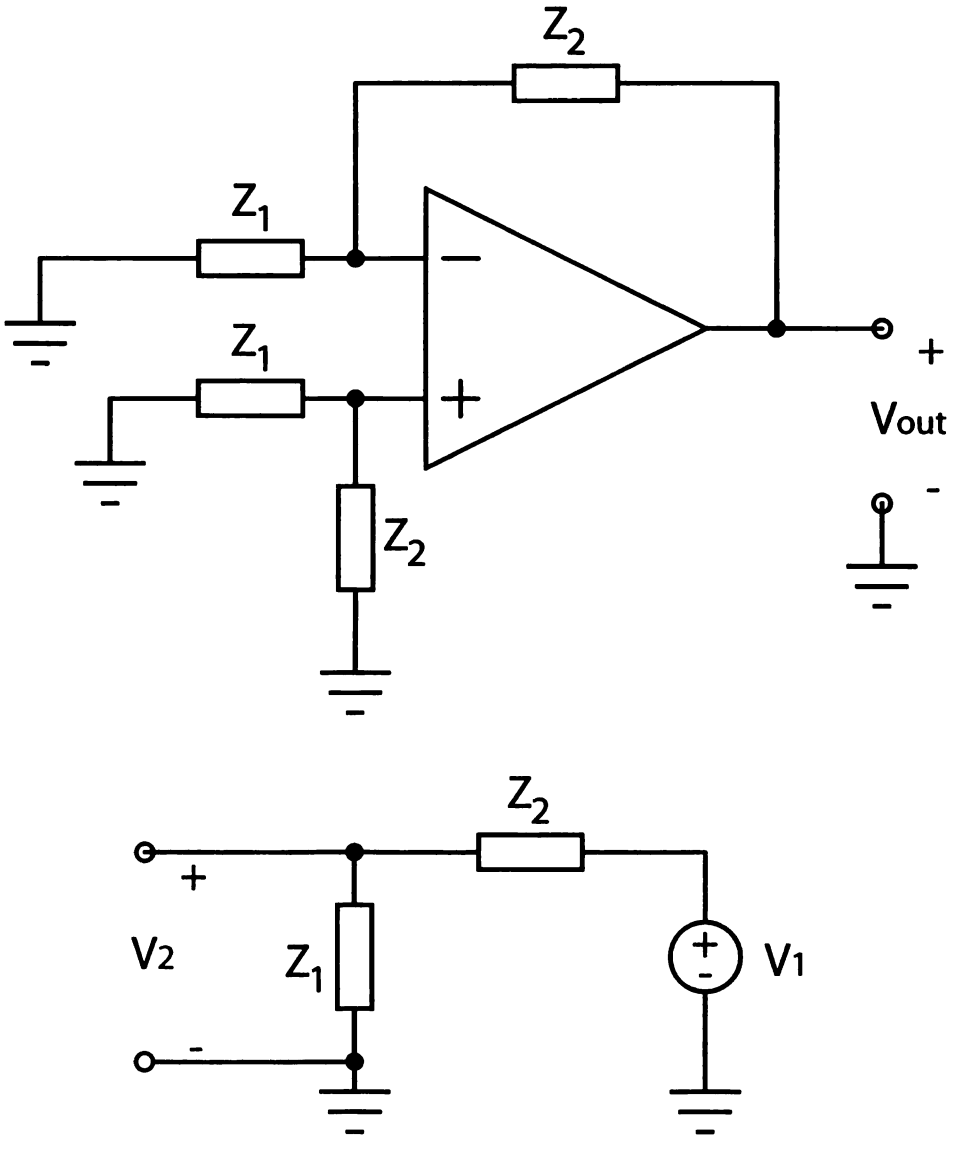


Figure 3.65: Differential amplifier β network.

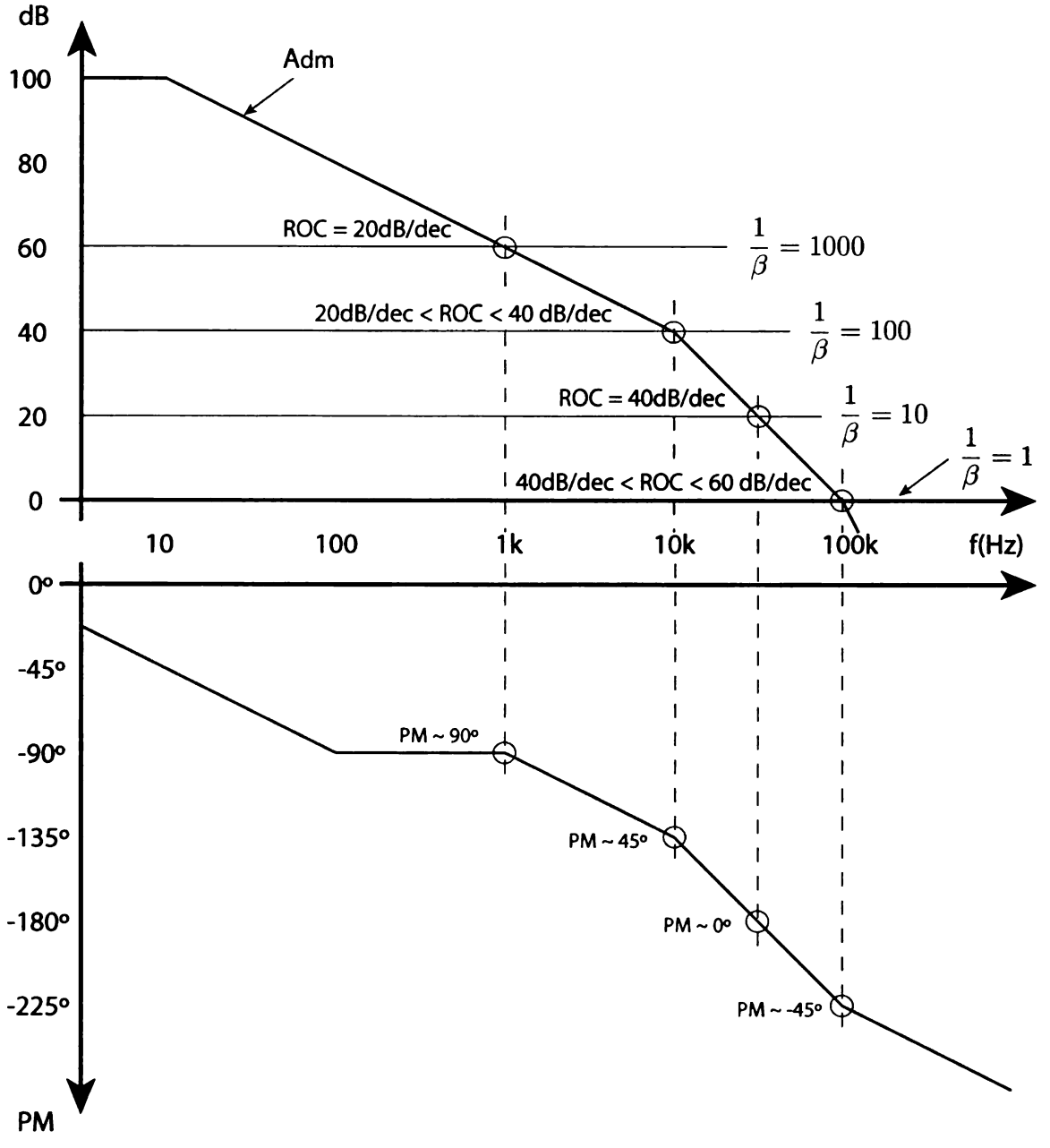


Figure 3.66: Bode plot of $|A_{dm}|$ and $\frac{1}{\beta}$ for different feedback networks and the corresponding plot of PM [21].

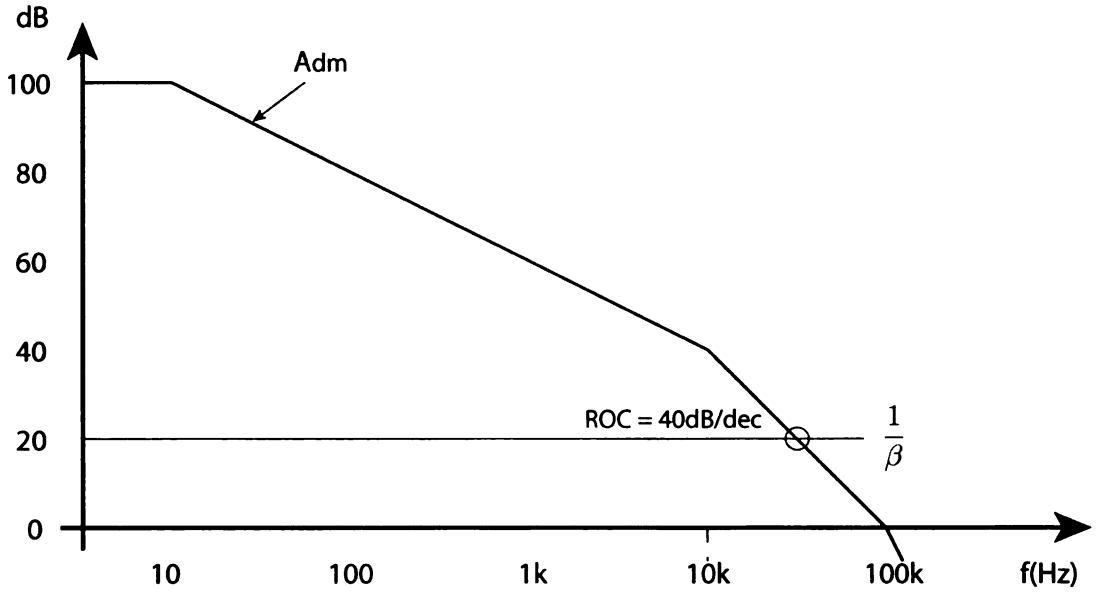


Figure 3.67: Bode plot showing the ROC of unstable amplifier.

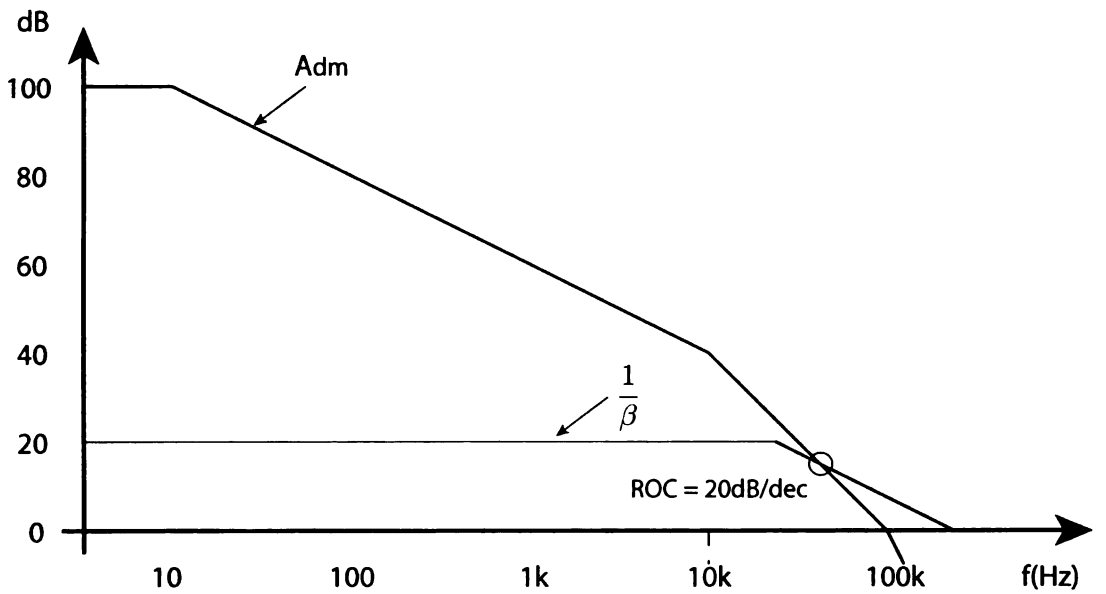


Figure 3.68: Bode plot showing the ROC of a stabilized amplifier.

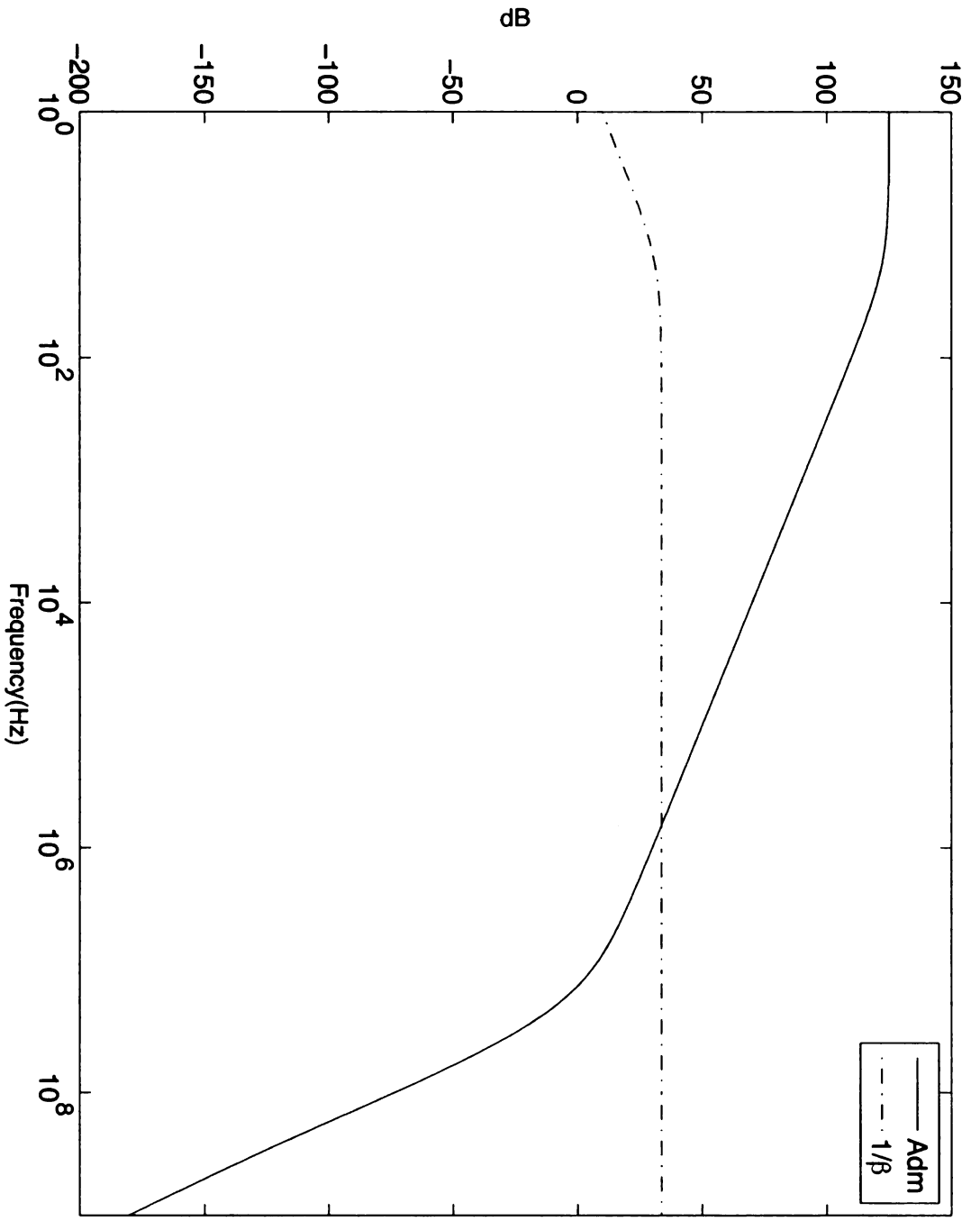


Figure 3.69: Plot of the open loop Bode plot and $1/\beta$ to determine the ROC of power amplifier.

3.5.4 Performance

The pulsed output of the amplifier with a $1V_pk-pk$ input and a $100V_pk-pk$ output across a 50Ω load is shown in Figure 3.70. The simulated bandwidth of the amplifier is 380kHz. The lower -3_{dB} frequency is 15Hz and the upper -3_{dB} frequency is 380kHz. The Bode plot of the amplifier is shown in Figure 3.71. The measured bandwidth of the constructed amplifier is 160kHz. The limiting elements in the simulation are the output BJTs. A discrepancy in the model and the acutall transistors could explain the difference in simulated and measured bandwidth.

3.6 Impedance Matching

Impedance matching is used to maximize power transfer and eliminate reflections [22].

3.6.1 transducer impedance.s

The impedance of each transducer was measured with a HP4194A Impedance Analyzer and is shown in Figures 3.72 -3.84. Each channel has a unique impedance. The impedance of each channel at 80kHz is summarized in Table 3.7. The resistance varies from 151Ω to 318Ω and the reactance varies from -167Ω to 25Ω. This is a large variation in the impedances of the transducers, so each channel will be uniquely matched.

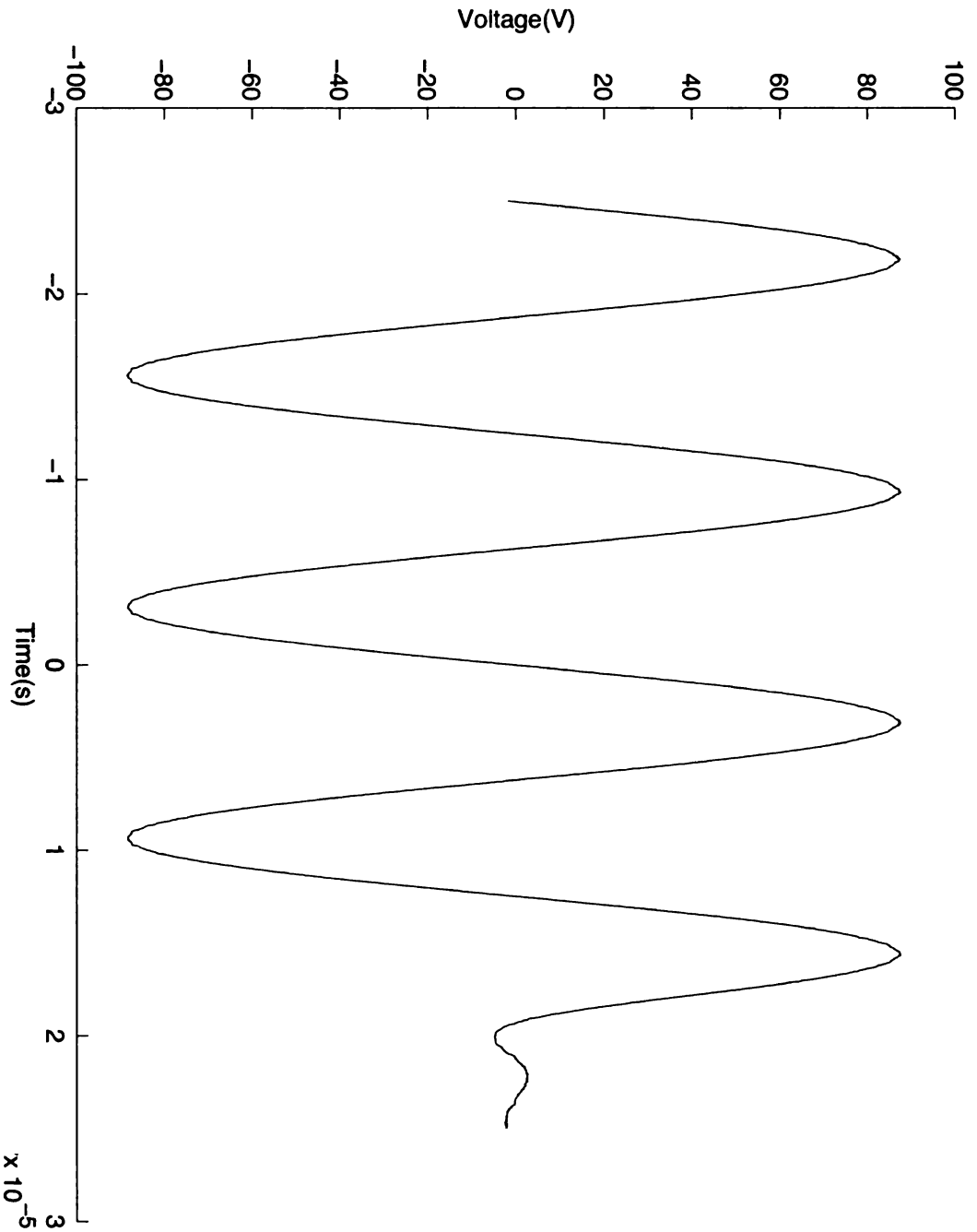


Figure 3.70: Amplifier output.

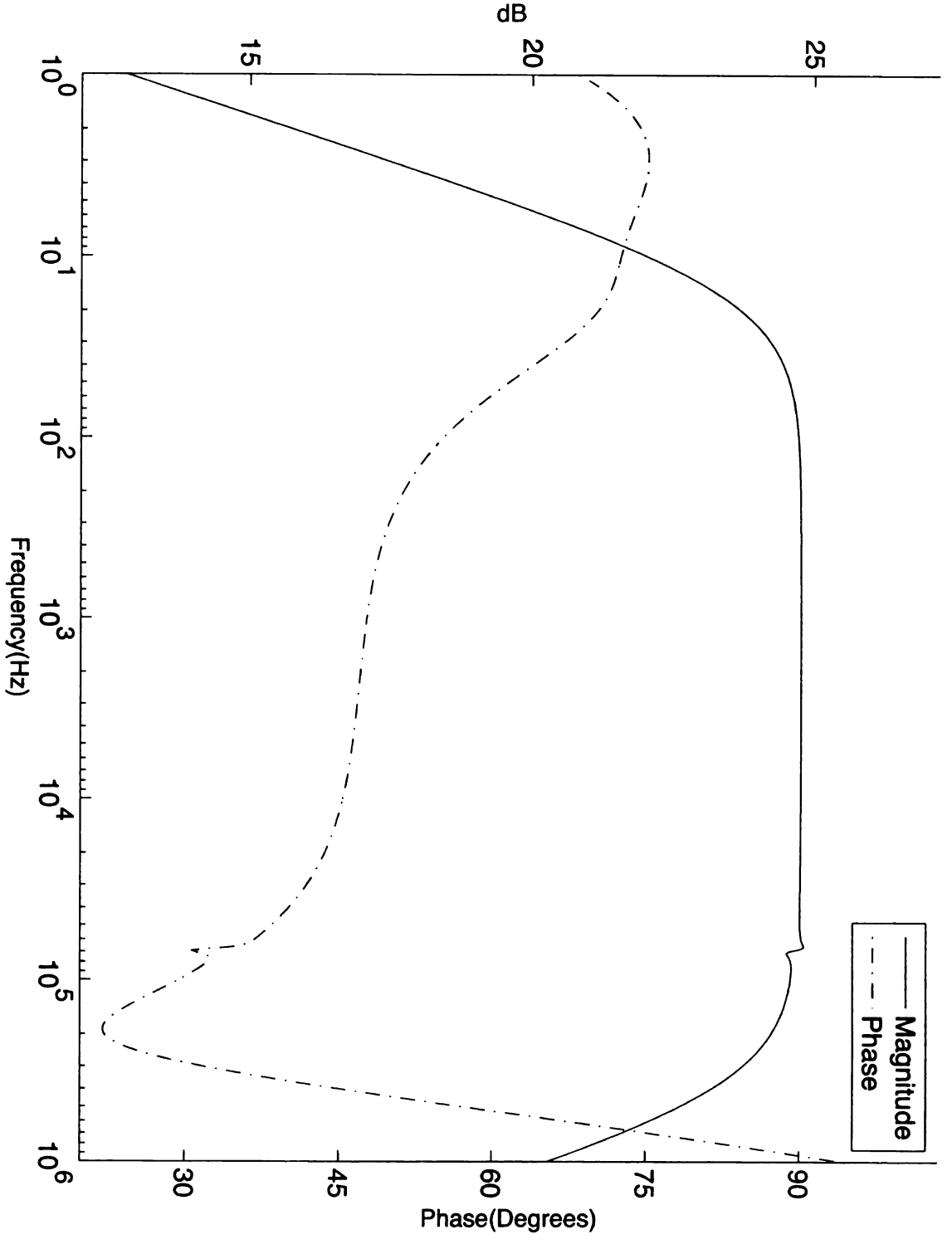


Figure 3.71: Amplifier bode plot.

Channel	Resistance(Ω)	Reactance(Ω)
1	226.60	15.36
2	231.60	25.31
3	245.11	-35.31
4	218.07	-22.96
5	217.69	-29.76
6	254.65	12.52
7	318.20	-5.58
8	318.23	-167.74
9	151.90	-6.53
10	238.58	-8.65
11	196.82	-34.99
12	237.13	-35.50
13	232.82	-66.26

Table 3.7: Transducer impedances at 80kHz.

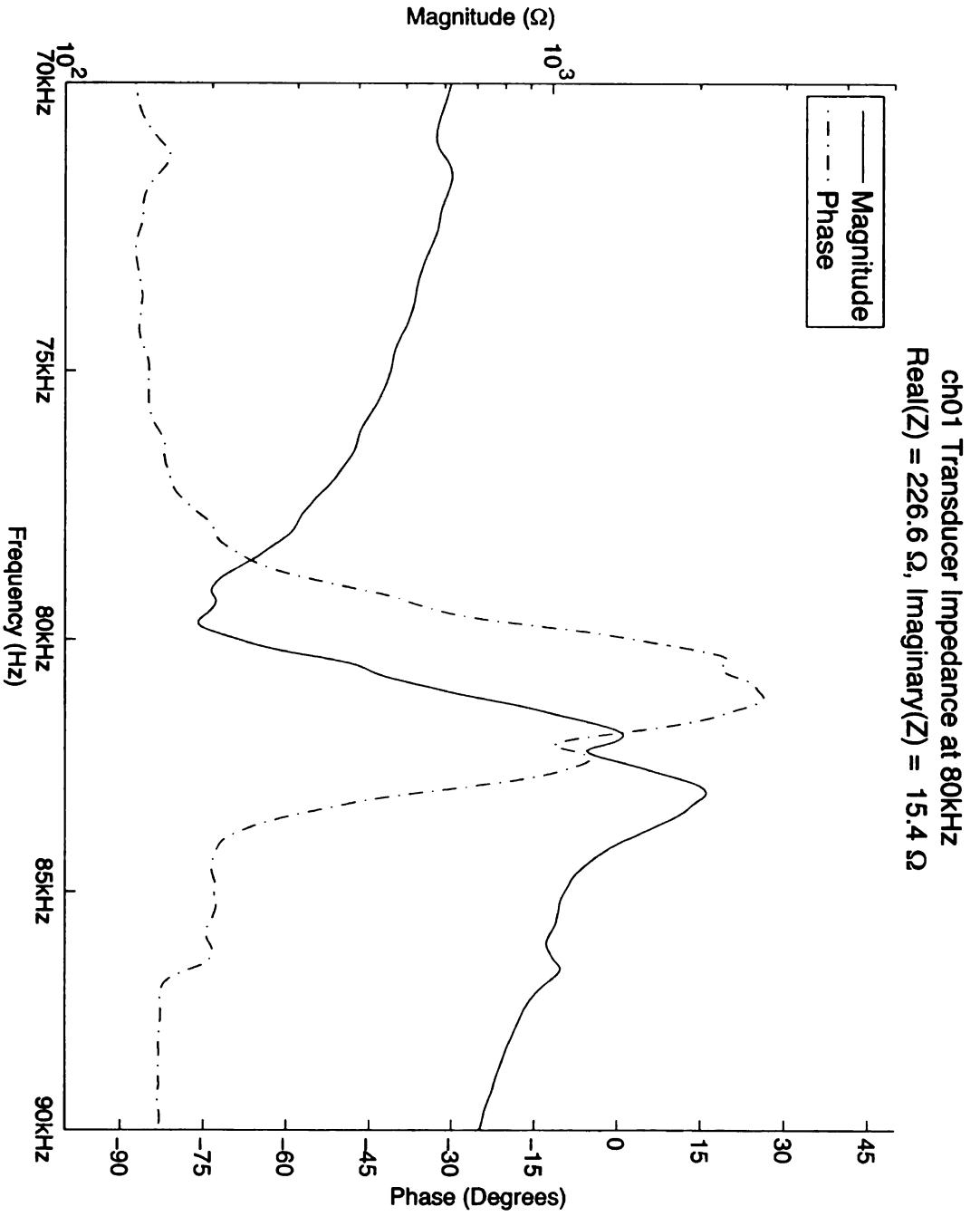


Figure 3.72: Channel 1 transducer impedance.

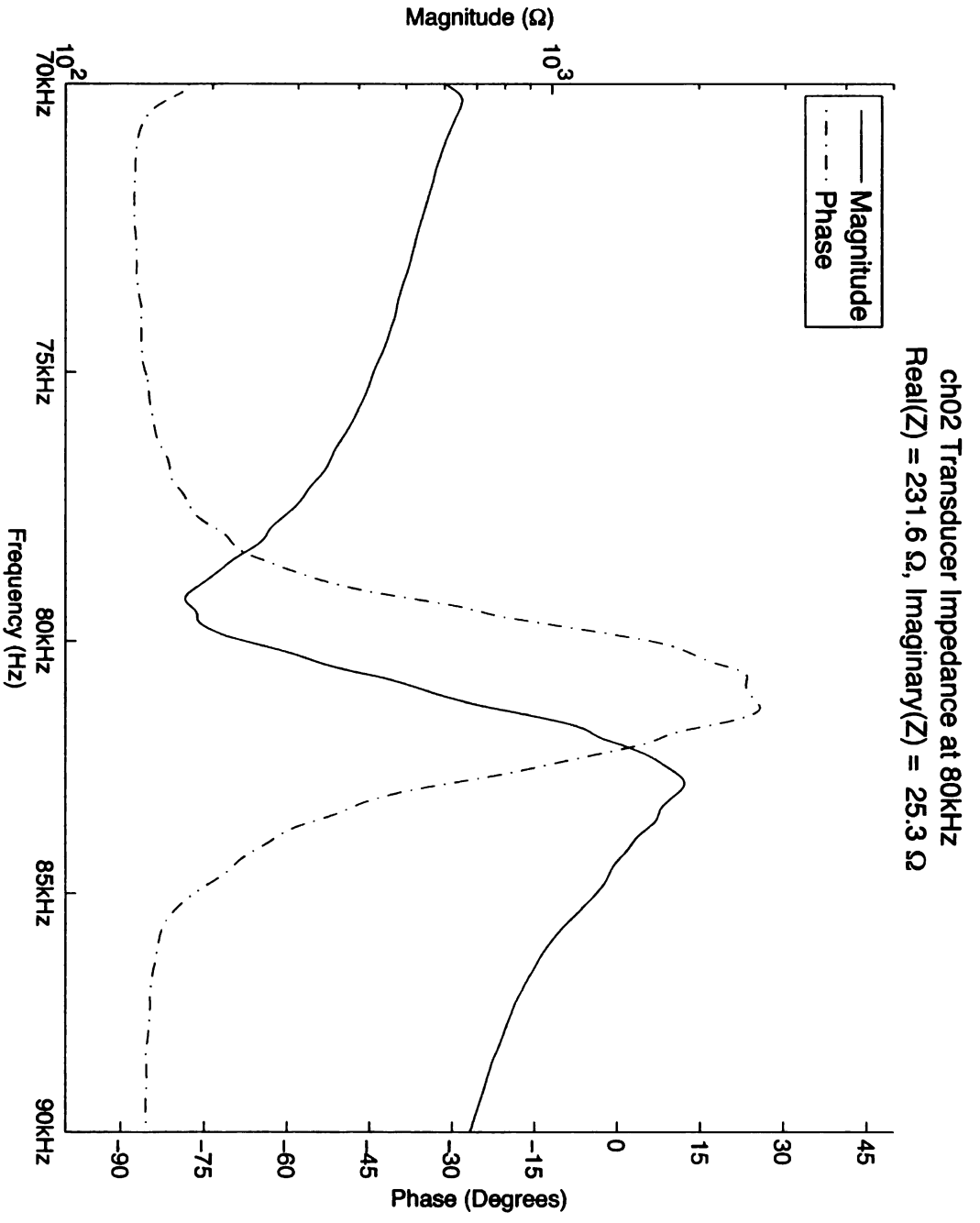


Figure 3.73: Channel 2 transducer impedance.

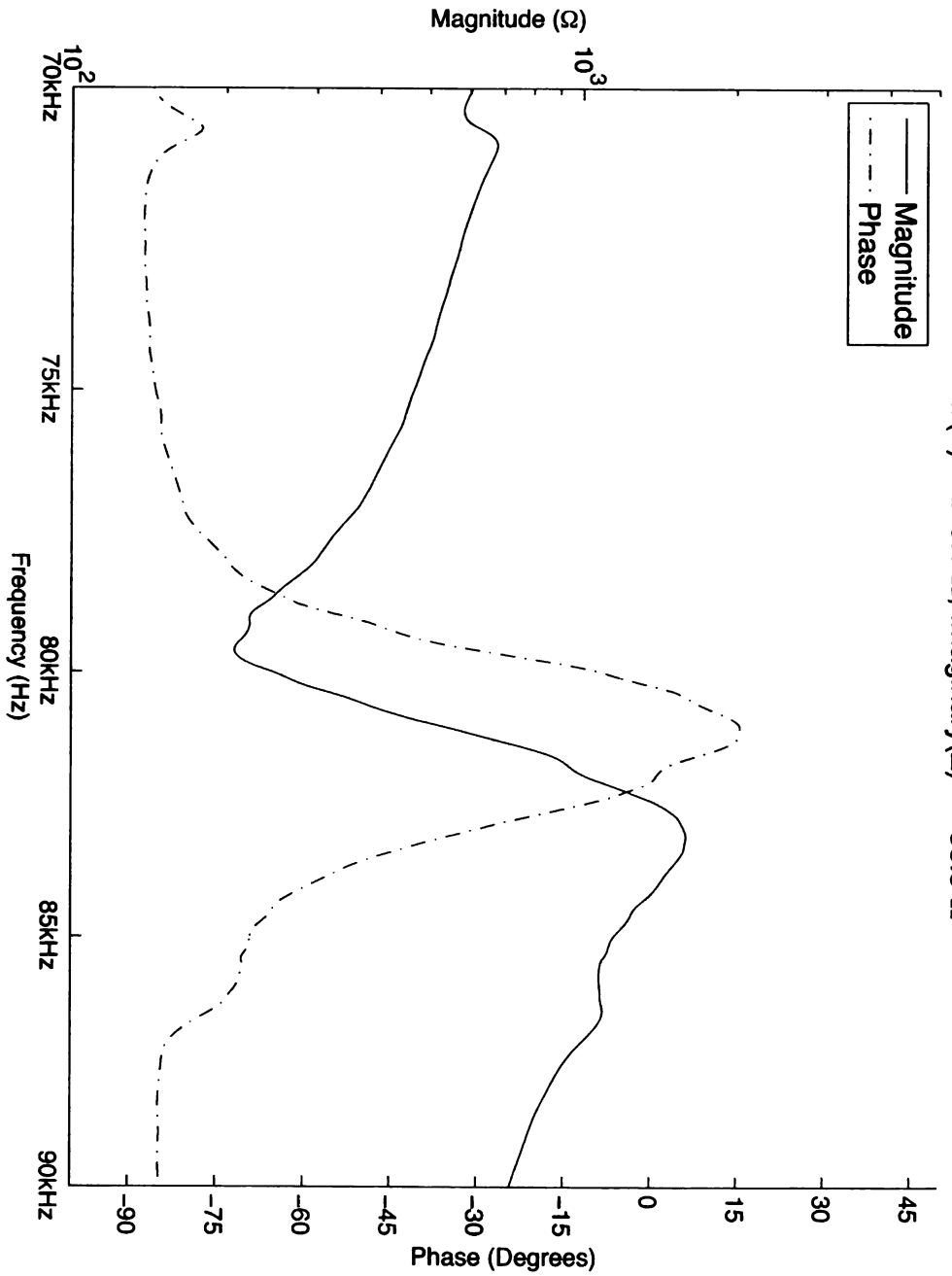


Figure 3.74: Channel 3 transducer impedance.

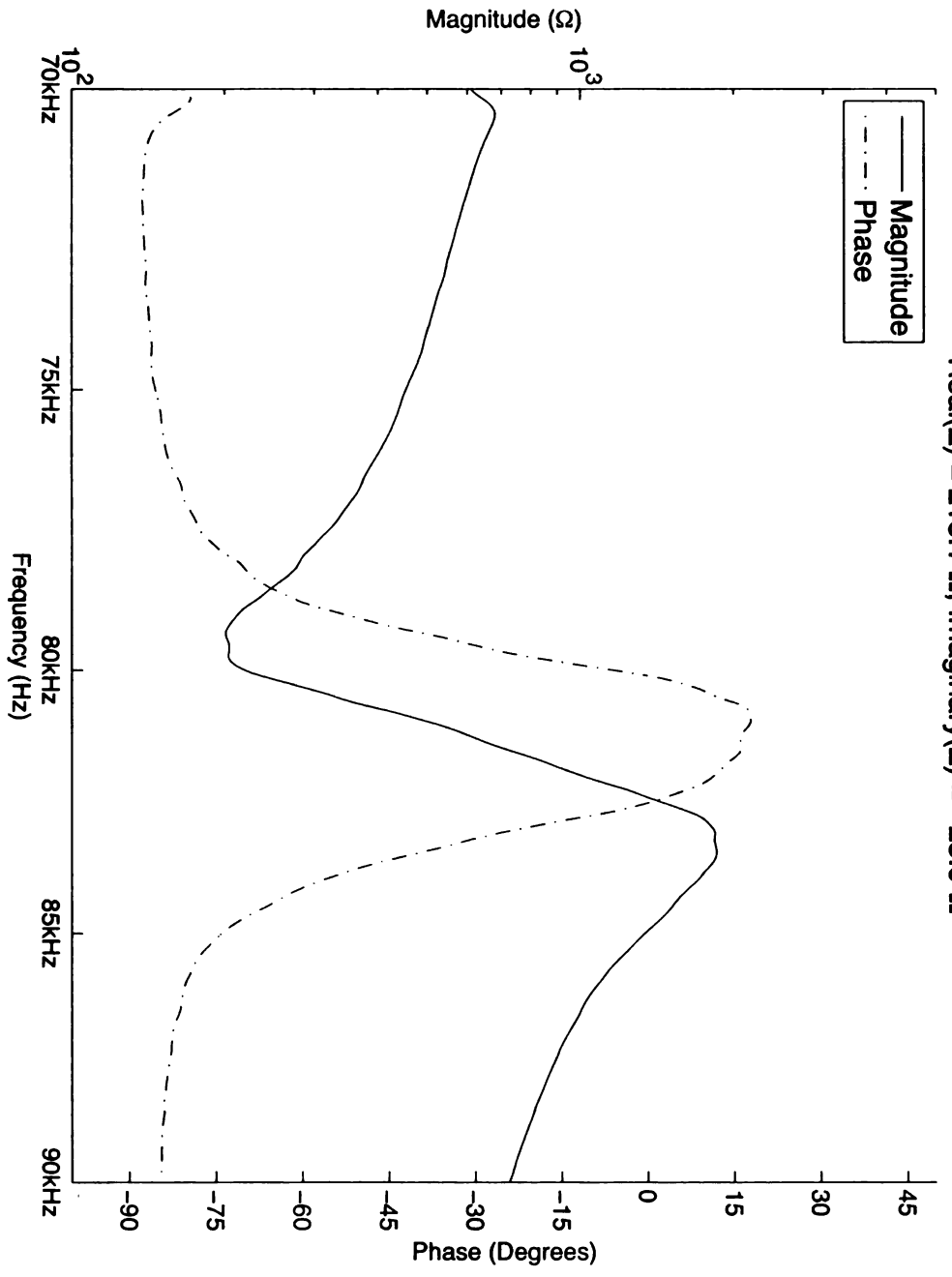


Figure 3.75: Channel 4 transducer impedance.

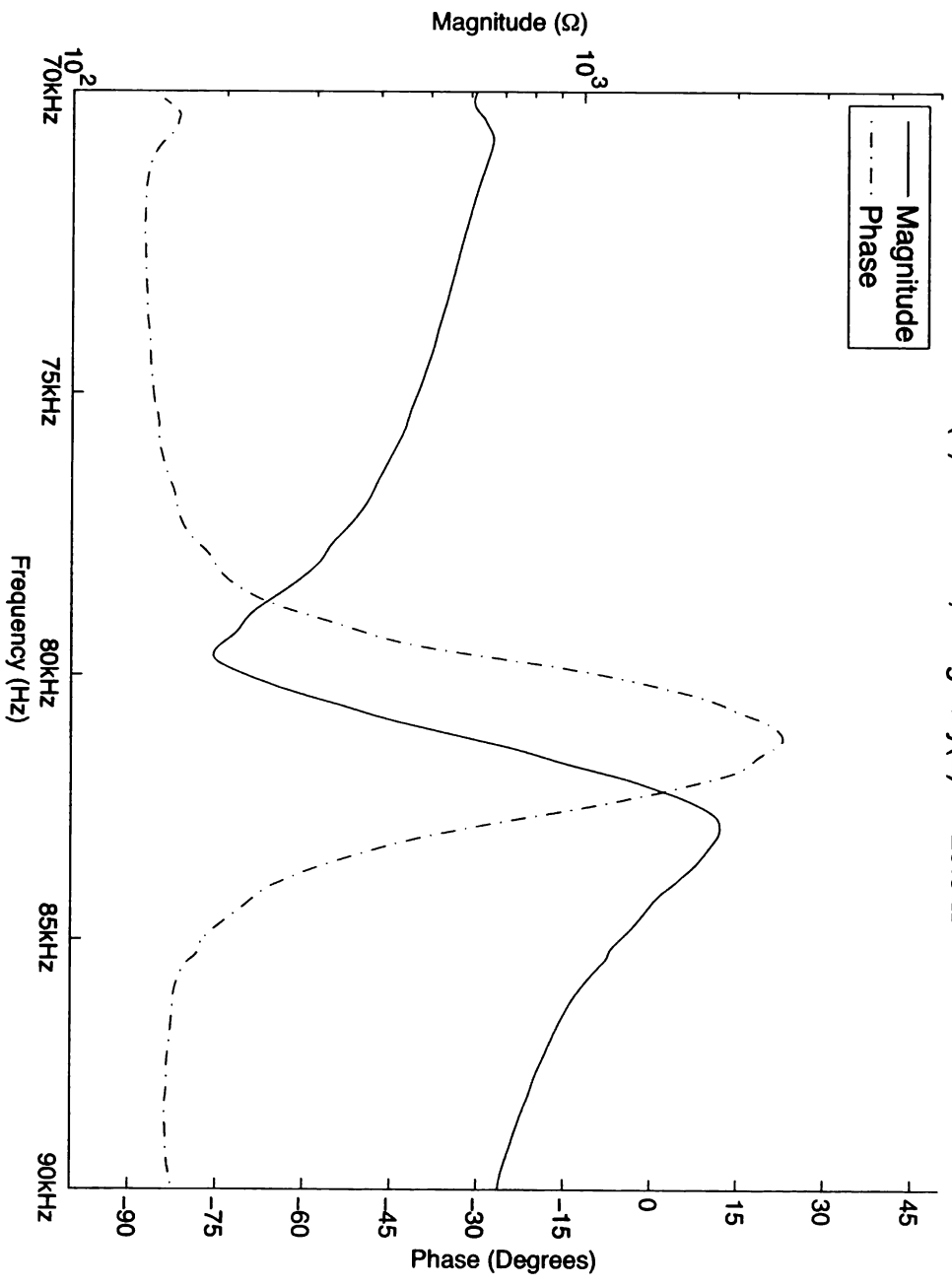


Figure 3.76: Channel 5 transducer impedance.

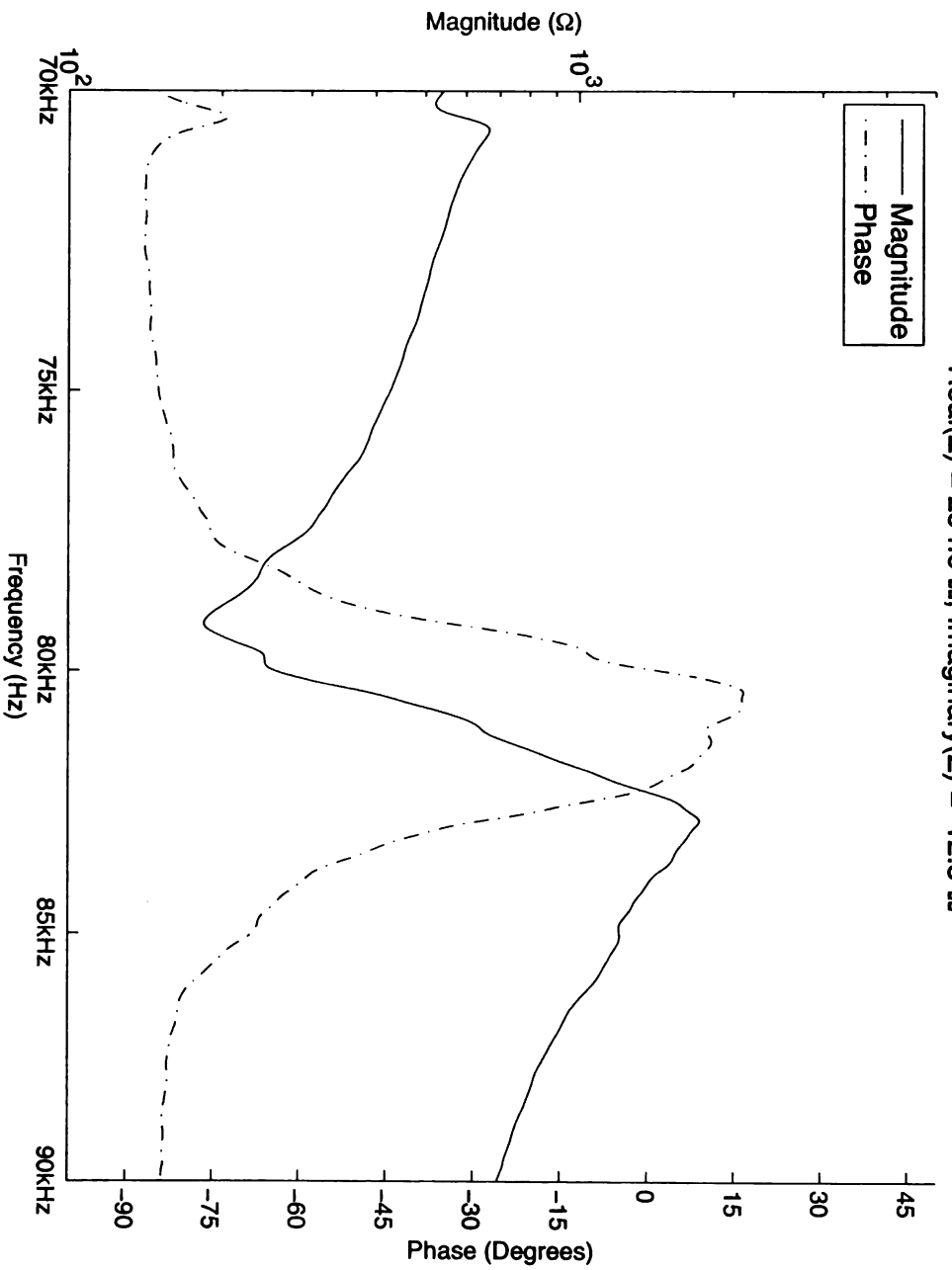


Figure 3.77: Channel 6 transducer impedance.

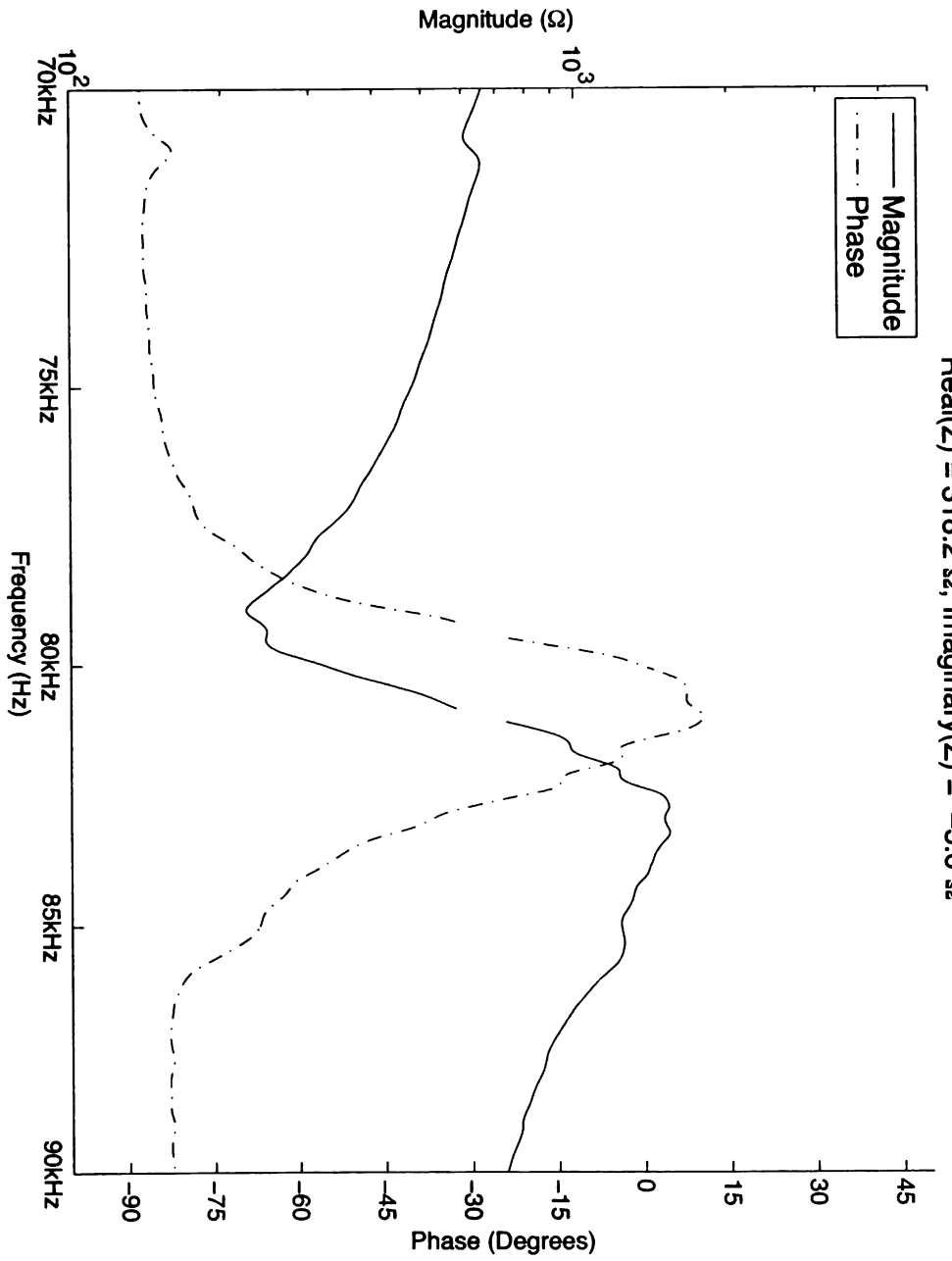


Figure 3.78: Channel 7 transducer impedance.

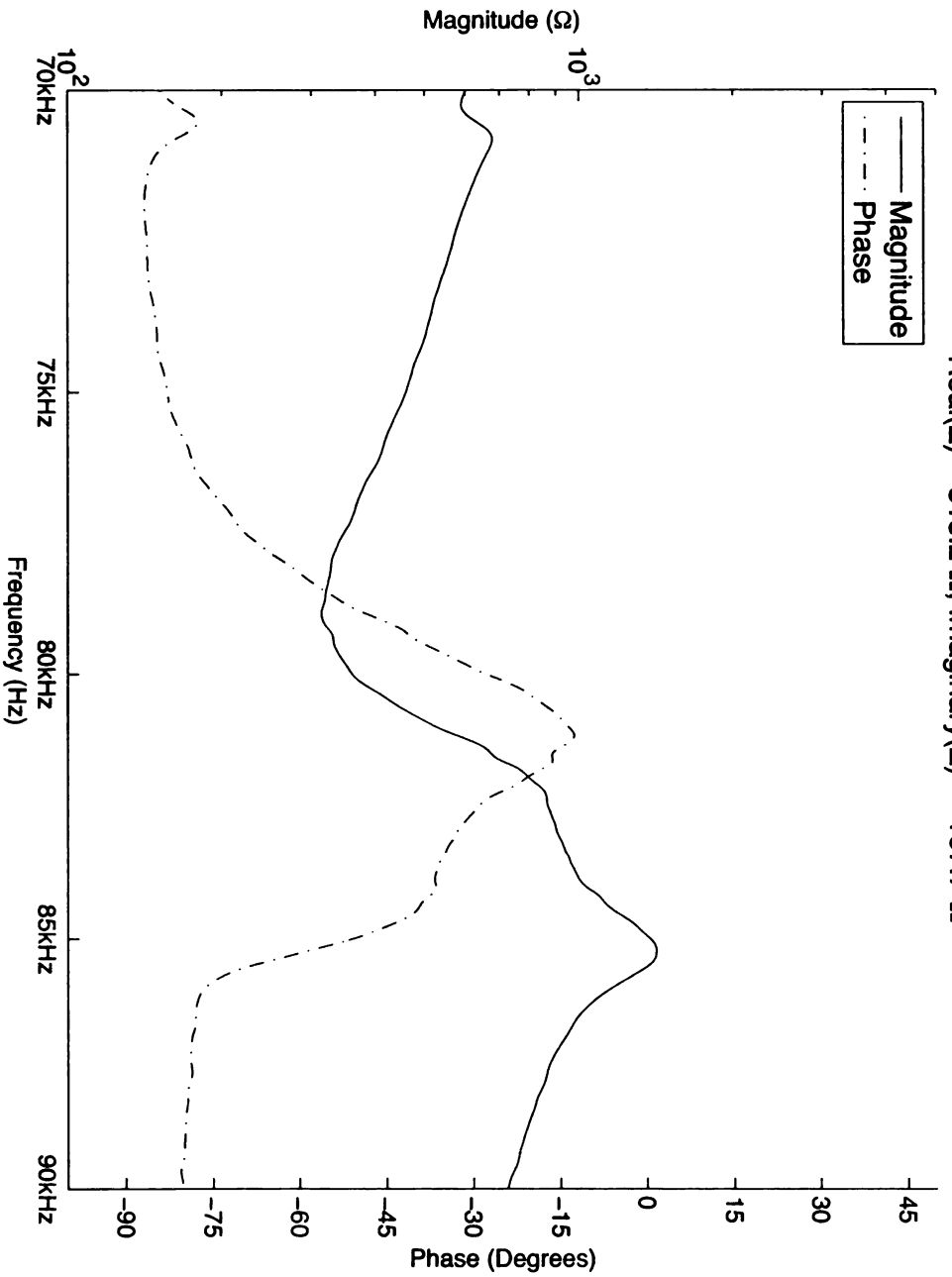


Figure 3.79: Channel 8 transducer impedance.

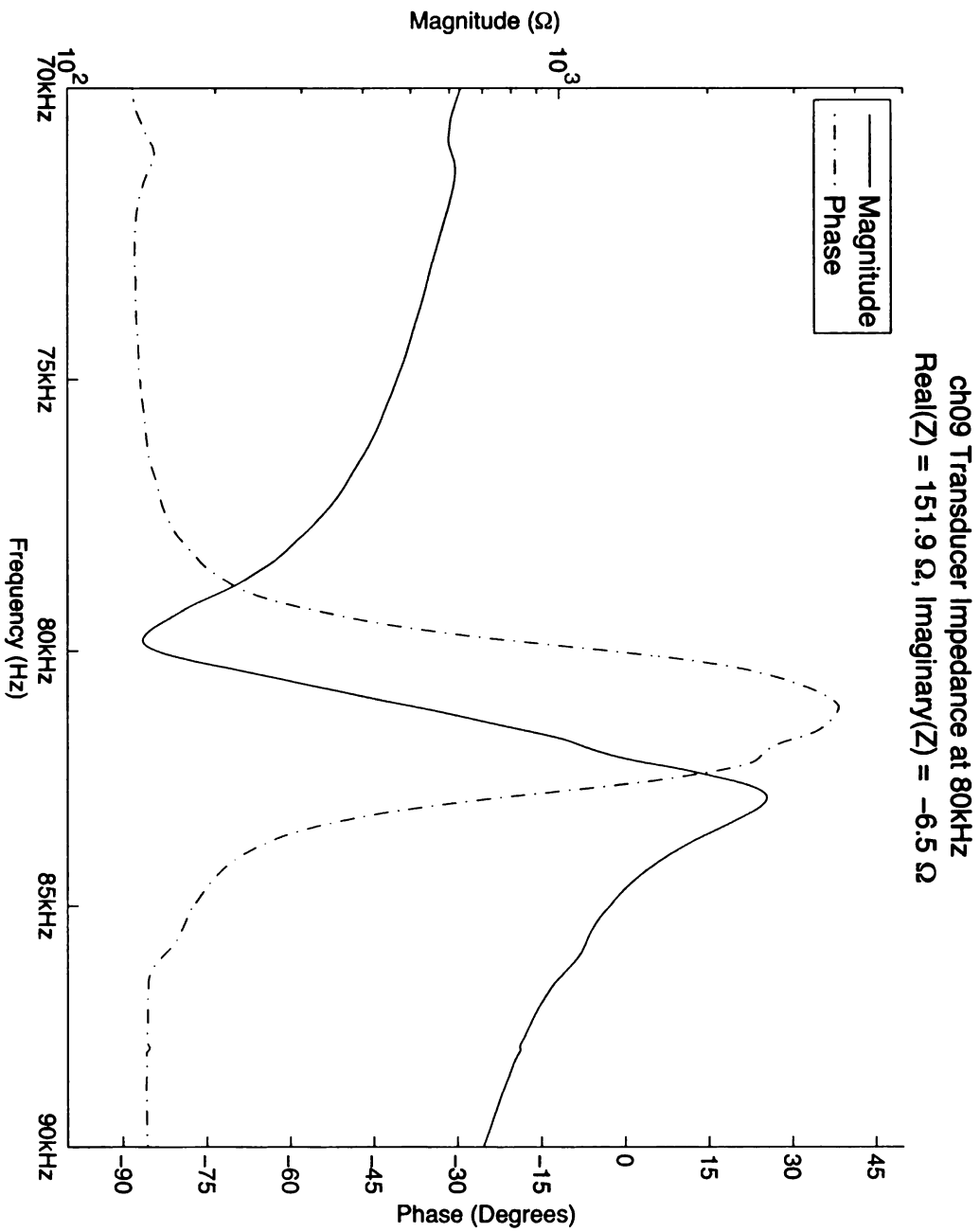


Figure 3.80: Channel 9 transducer impedance.

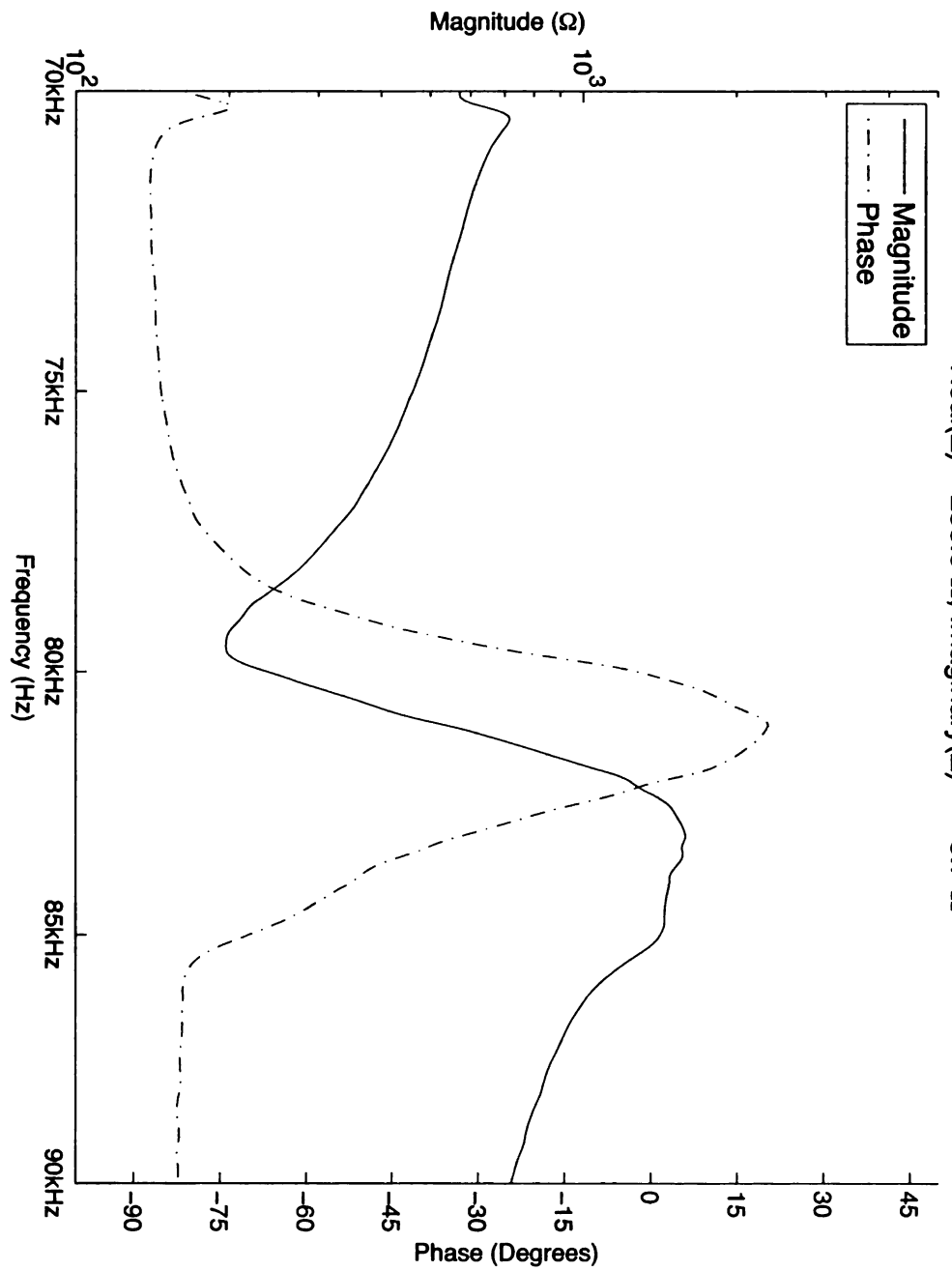


Figure 3.81: Channel 10 transducer impedance.

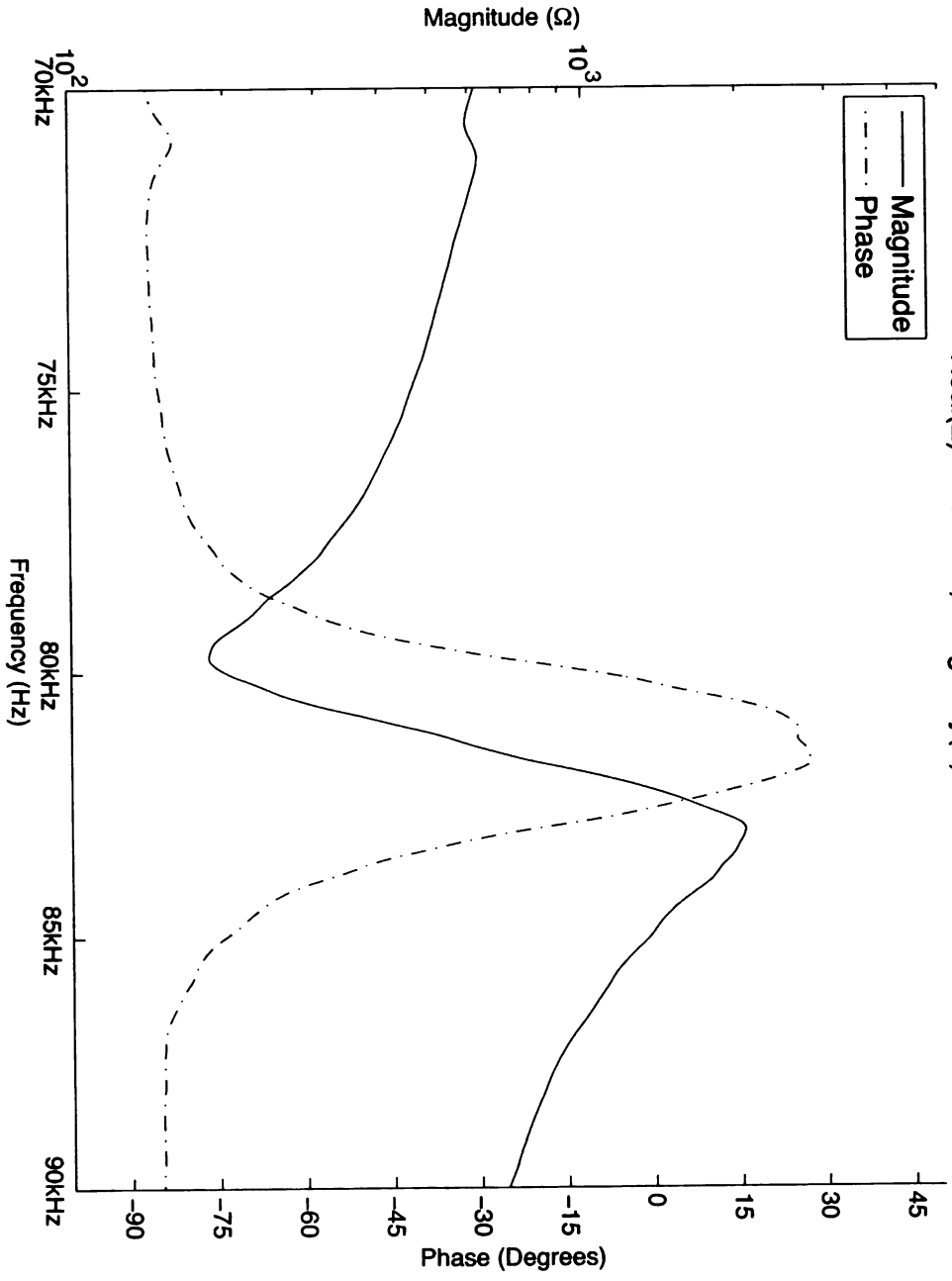


Figure 3.82: Channel 11 transducer impedance.

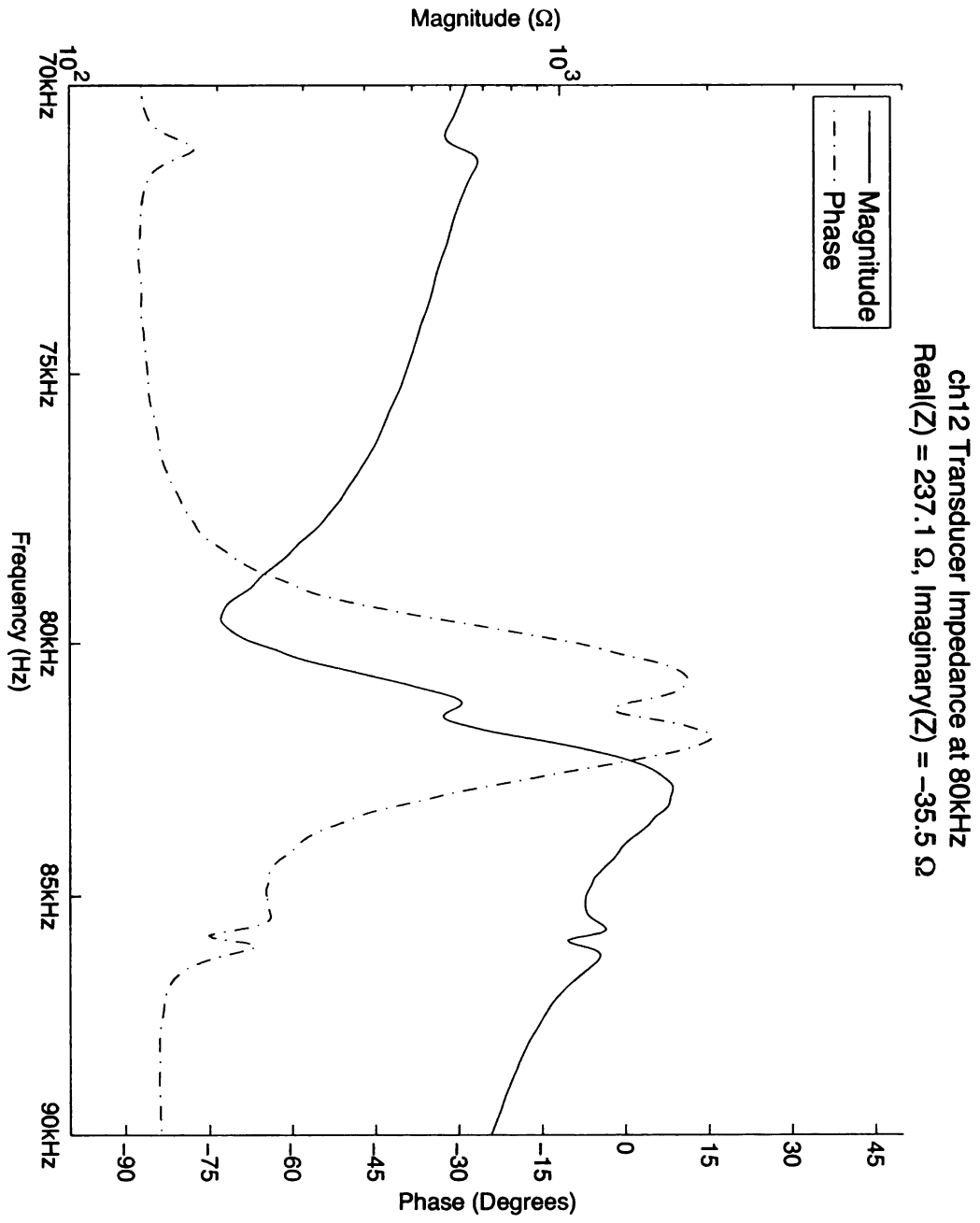


Figure 3.83: Channel 12 transducer impedance.

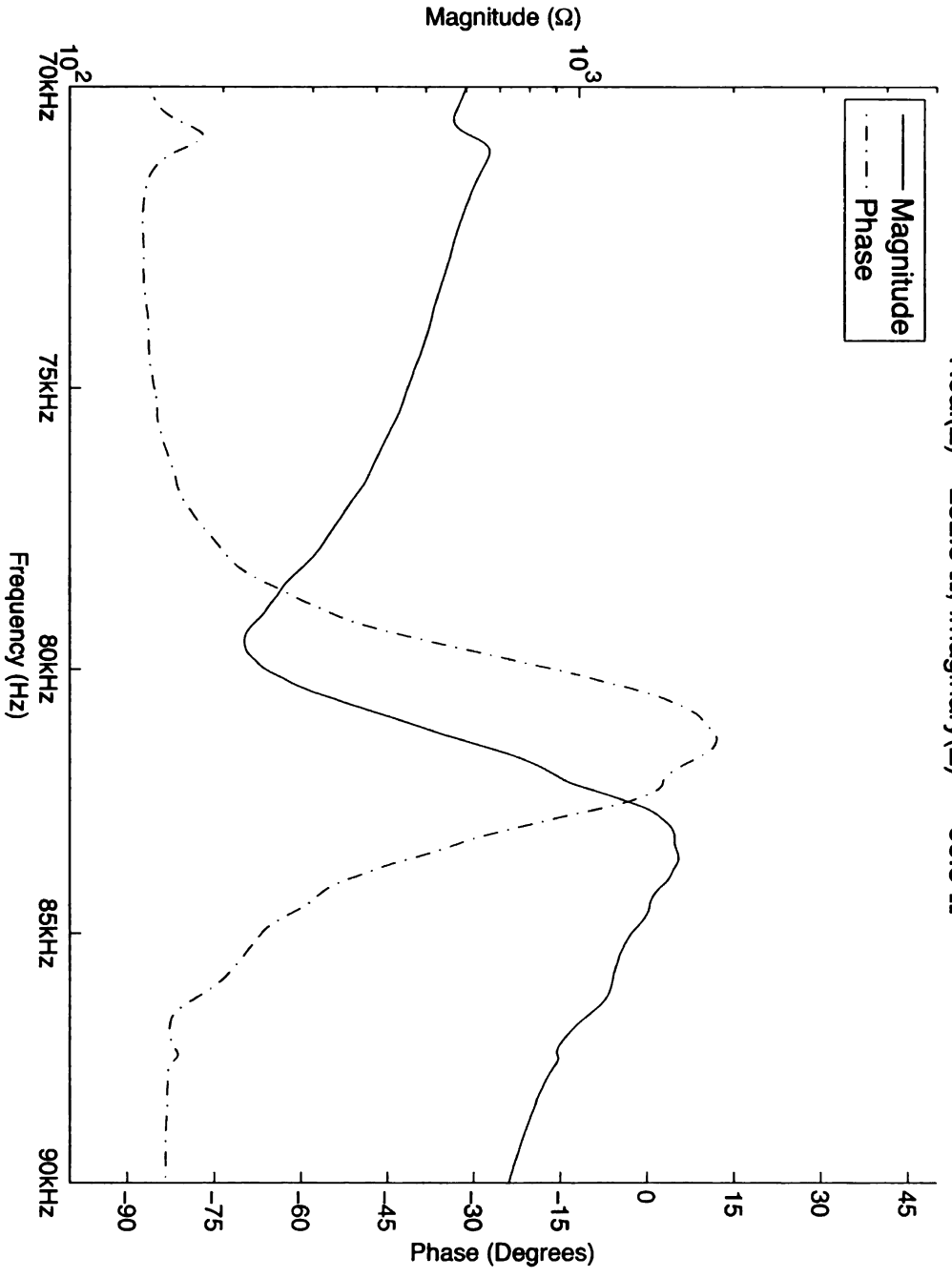
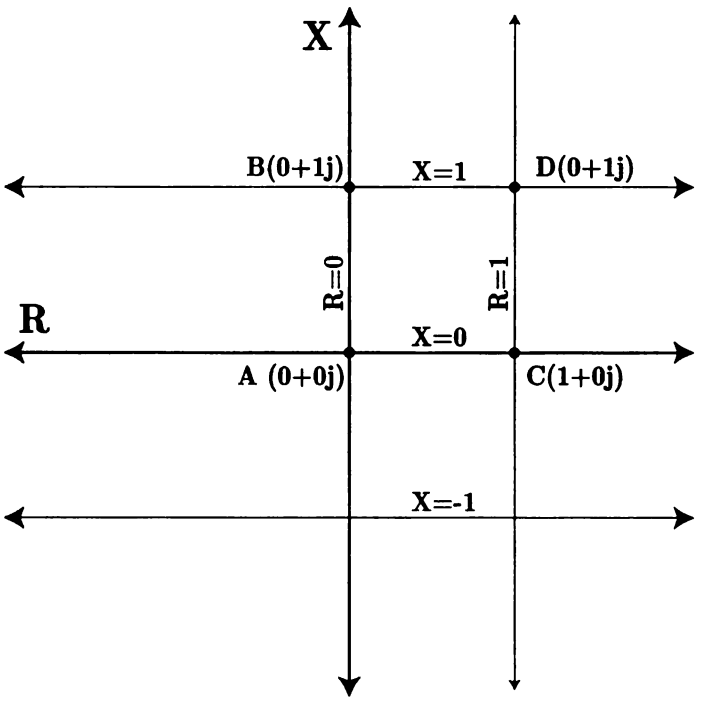


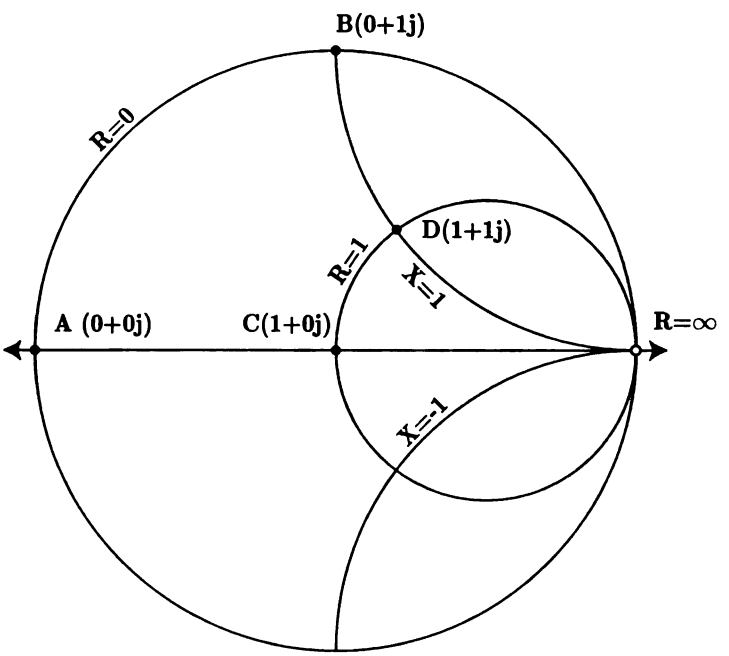
Figure 3.84: Channel 13 transducer impedance.

3.6.2 Matching Networks

A Smith chart can be used to determine the elements needed to transform the transducer impedance to 50Ω. A Smith chart is a graph of the impedance and admittance planes where impedances and admittances are mapped directly. The impedance (Z) plane in Figure 3.85(a) is transformed to the impedance (Z) Smith chart in Figure 3.85(b). The circles in Figure 3.85(b) are of constant resistance. The arcs in Figure 3.85(b) are of constant reactance. The admittance plane is transformed in to the Smith chart in Figure 3.86. The circles in Figure 3.86(b) are of constant conductance. The arcs in Figure 3.86(b) are of constant susceptance. The impedance and admittance Smith charts can be overlaid such that impedances and admittances map directly as is shown in Figure 3.87. This is because $Z = \frac{1}{Y}$, so point A ($Z = 0 + 0j$) on the Z graphs maps to infinity on the Y graphs. This point is on the far left of Figure 3.87. Likewise, point E ($Y = 0 + 0j$) on the Y graphs maps to infinity on the Z graphs, which is on the far right in Figure 3.87.

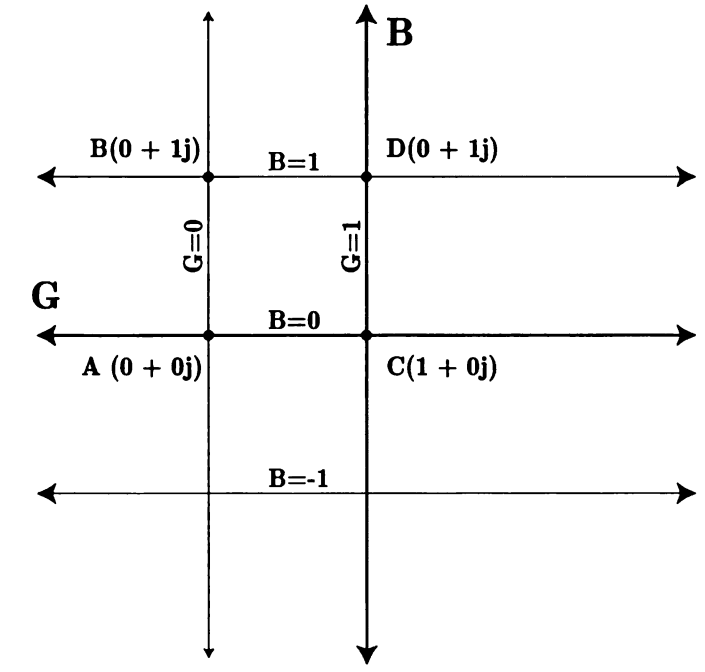


(a) Z cartesian plane.

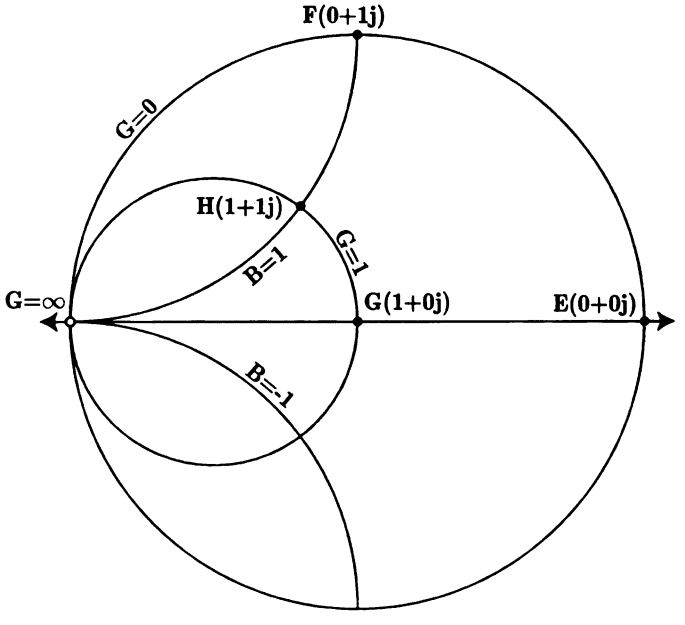


(b) Z Smith graph.

Figure 3.85: Z-plane to Z-Smith chart mapping.



(a) Y cartesian plane.



(b) Y Smith graph.

Figure 3.86: Y-plane to Y-Smith chart mapping.

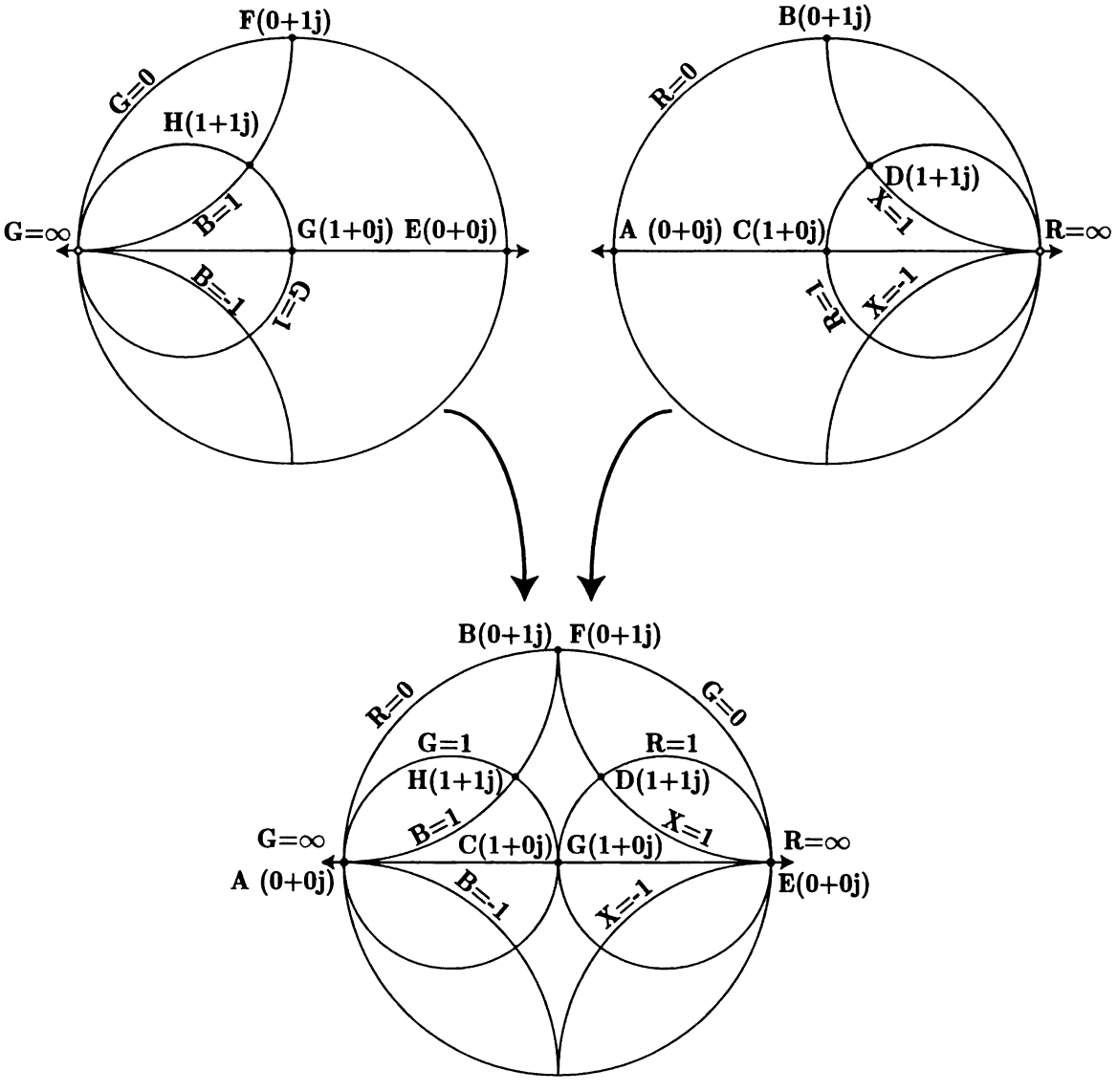


Figure 3.87: Combined Y and Z Smith chart mapping.

Computing the impedance transformation of a passive network can be accomplished using Smith charts. Inductors placed in series with a circuit moves the impedance upwards along the lines of constant resistance. A capacitor placed in series with a circuit moves the impedance downwards along the lines of constant resistance. An inductor placed in parallel with a circuit moves the admittance upwards along the lines of constant conductance. A capacitor placed in parallel with a circuit moves the admittance downwards along lines of constant conductance. These paths are shown in Figure 3.88.

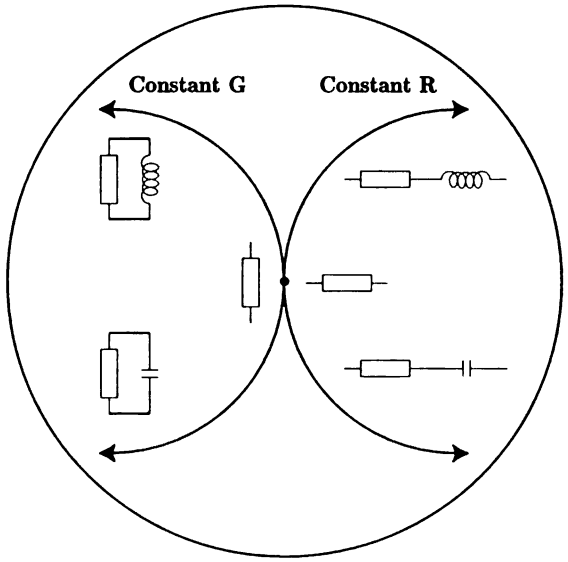


Figure 3.88: Effect of adding a capacitor or inductor in series or parallel with an impedance.

In order to transform an impedance using a Smith chart, a point is plotted for the starting impedance and one for the conjugate of the desired impedance on the Smith chart. Arcs are drawn along paths of constant conductance and resistance through the points of starting and desired conjugate impedance. This is shown in Figure 3.89. Matching networks can be realized with paths from arcs of both points that intersect. Figure 3.90 shows two possible paths for the given example.

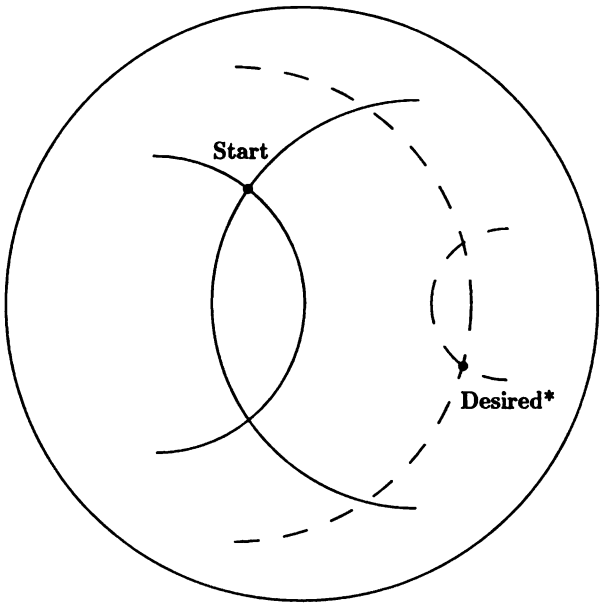


Figure 3.89: Arcs of constant resistance and conductance plotted through the start and conjugate desired impedances.

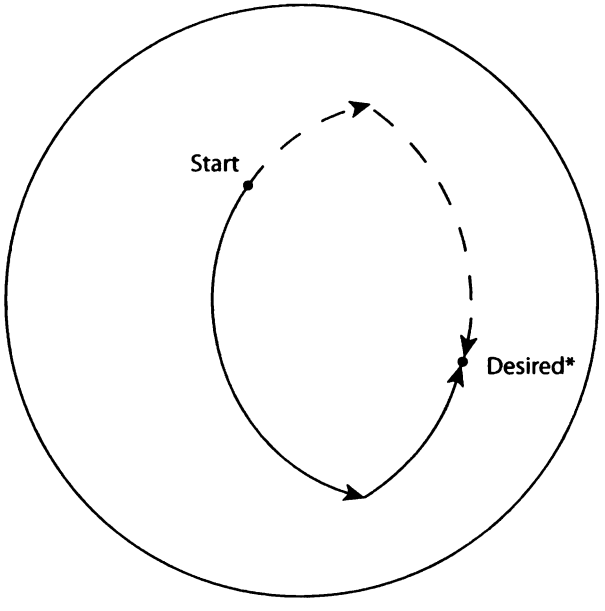


Figure 3.90: Two possible paths on the Smith chart transforming impedance start to desired conjugate impedance.

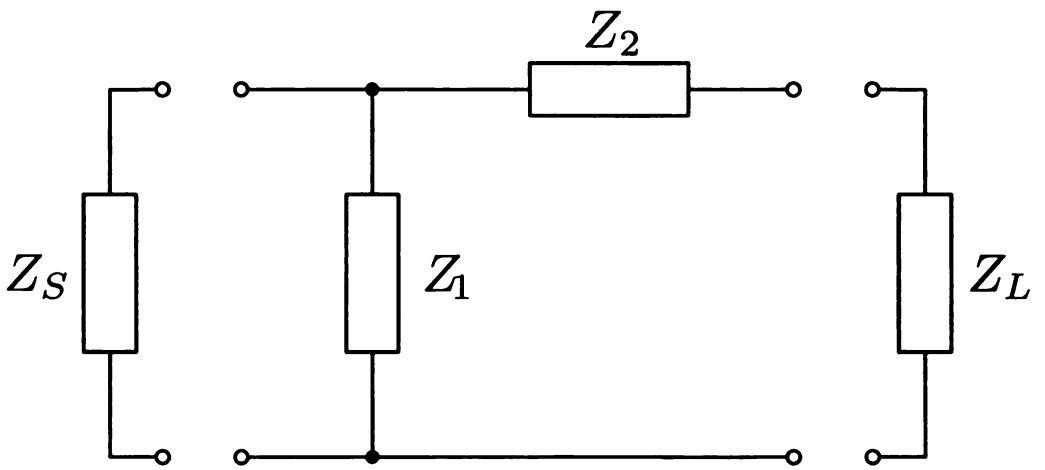
There are several topologies for impedance matching: 2 element L-networks; 3 element π and T-networks; and 4 element cascaded 2 element networks. All networks can transform the magnitude of the source to load impedances. L-networks are simple but have a fixed selectivity and pass high or low frequencies. T and π-networks are more complicated but allow for a designed selectivity. Cascaded networks allow for a band pass filters and larger source to load impedance ratios. For simplicity, the initial matching is done with L-networks.

L-networks are simple 2 element networks that match the impedance and act as a low-pass or high-pass filters. Figure 3.91 shows the two basic topologies of an L-network. Z_S is the output impedance of the source and Z_L is the input impedance of the load. Z_1 and Z_2 can be capacitors, inductors or resistors, but inductors and capacitors are typically used so that power is not lost in a resistive element.

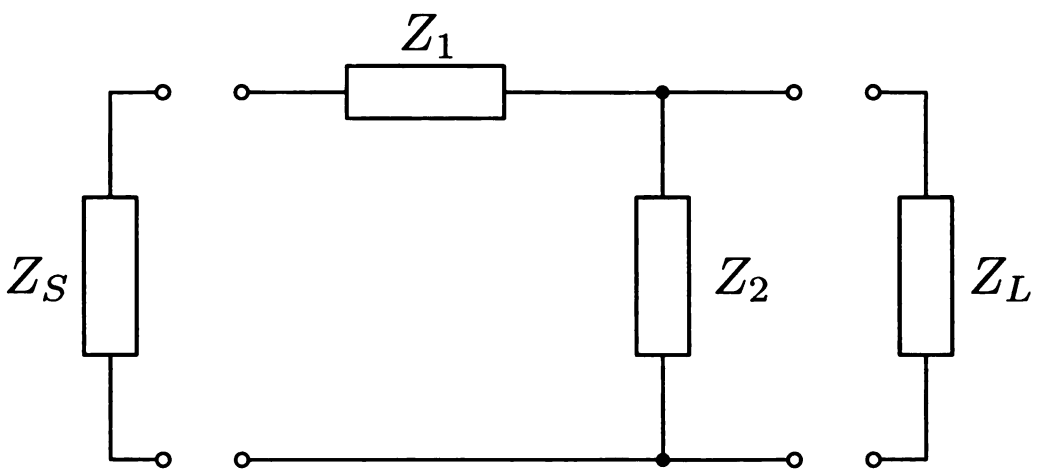
Choosing which topology to use depends on the difference of the real parts of $Z_S(R_S)$ and $Z_L(R_L)$. If R_S is larger than R_L then Z_1 is placed in parallel with Z_S to lower the source impedance and Z_2 is placed in series with R_L to raise the load impedance as is shown in Figure 3.91(a). If R_L is larger than R_S, then Z_2 is placed in parallel with Z_L in order to lower the load impedance and Z_1 is placed in series with Z_S raise the source impedance as shown in Figure 3.91(b). This can be seen on the Smith charts in Figure 3.6.2. Impedances with larger R, to the right on the Smith chart, move along arcs of constant conductance implying reactive elements being placed in parallel. Impedances with smaller R, to the left on the Smith chart, move along arcs of constant resistance implying reactive elements being placed in series.

A 2 element L-network is either a high pass or low pass filter. Whether it is high pass or low pass depends on the choice of where the inductor and capacitor is placed. For the case of $R_S < R_L$ in Figure 3.92, if a capacitor is used for Z_1 and an inductor is used for Z_2, then the L-nework is a low pass filter. Conversely, if an inductor is

used for Z_1 and a capacitor is used for Z_2, then the L-network is a high pass filter. For the case of $R_S > R_L$ in Figure 3.6.2, if a capacitor is used for Z_1 and an inductor is used for Z_2, then the L-nework is a high pass filter. Conversely, if an inductor is used for Z_1 and a capacitor is used for Z_2, then the L-network is a low pass filter.



(a) L-network for $Z_S > Z_L$.



(b) L-network for $Z_S < Z_L$.

Figure 3.91: L-Network Topologies

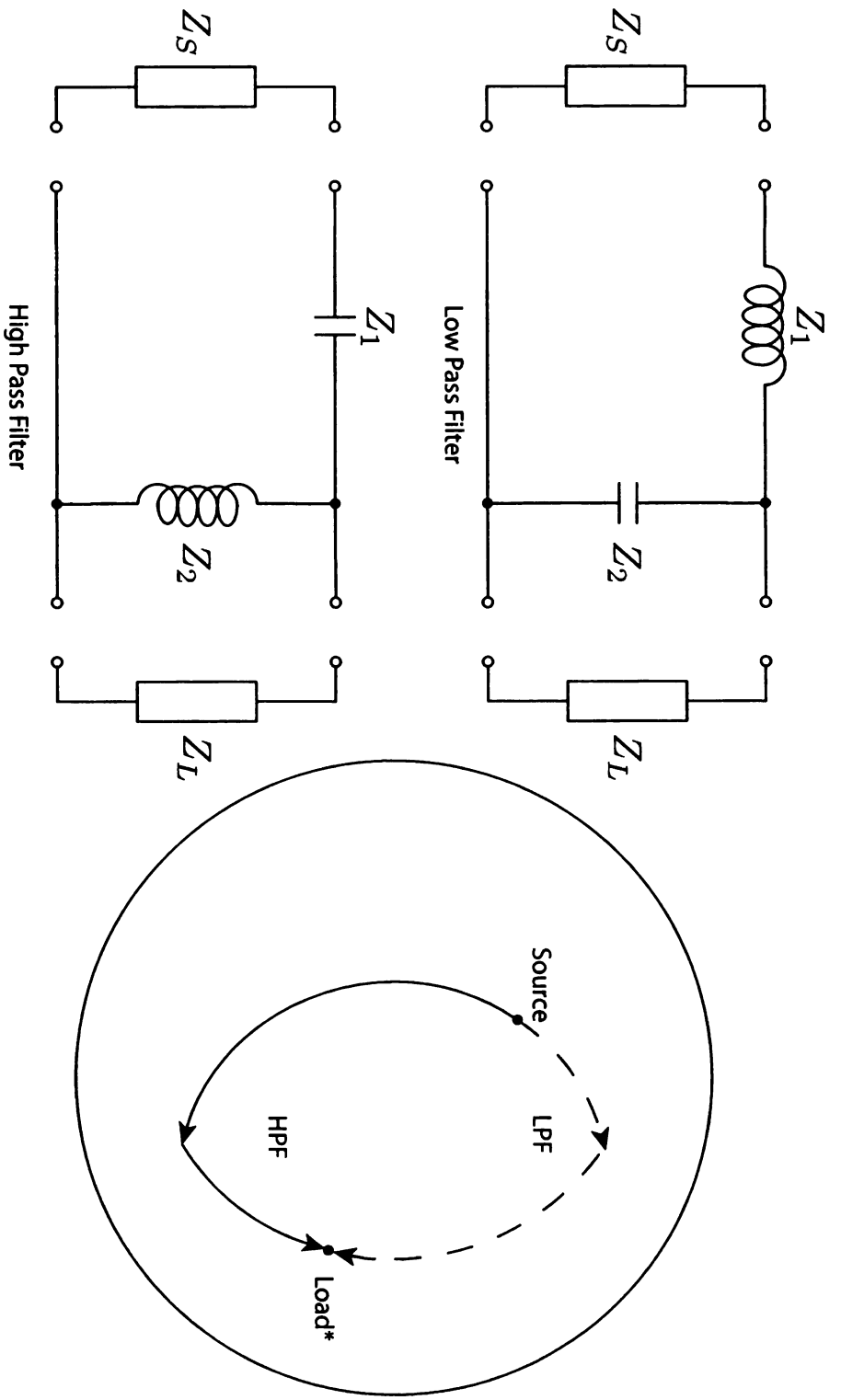


Figure 3.92: L-network impedance matching on a Smith chart for $R_S < R_L$.

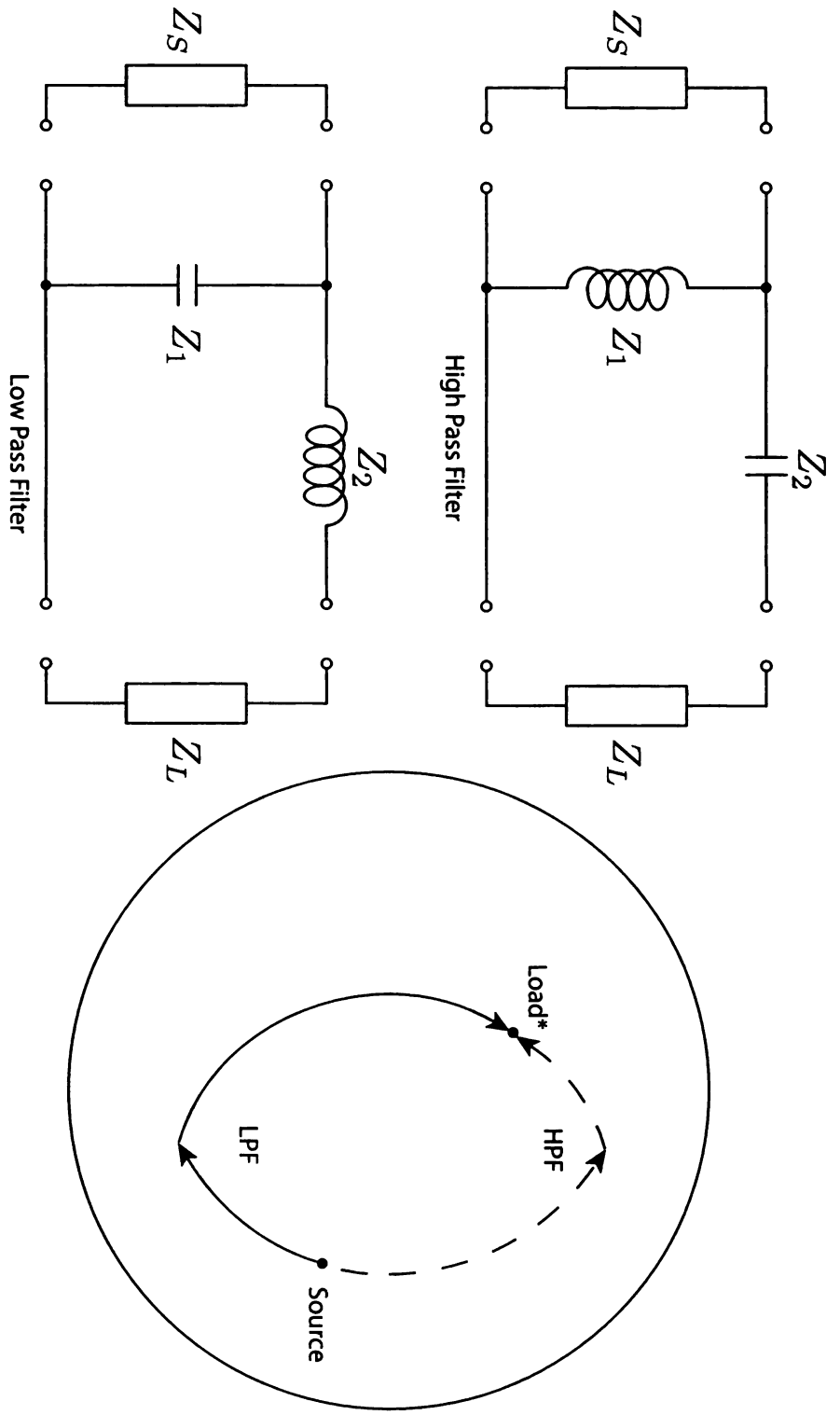


Figure 3.93: L-network impedance matching on a Smith chart for $R_S > R_L$.

Whether X_1 and X_2 is a capacitor or an inductor depends on if $R_S > R_L$ or $R_S < R_L$. First consider a matching network where $R_S > R_L$. To calculate the values of X_1 and X_2, the impedances of the source is transformed with X_1 to Z_1 at point C and the impedance Z_1 is transformed with X_2 from point C to *Load** in Figure 3.94. The transformed impedance Z_2 is set equal to the conjugate load impedance and the values X_1 and X_2 are calculated.

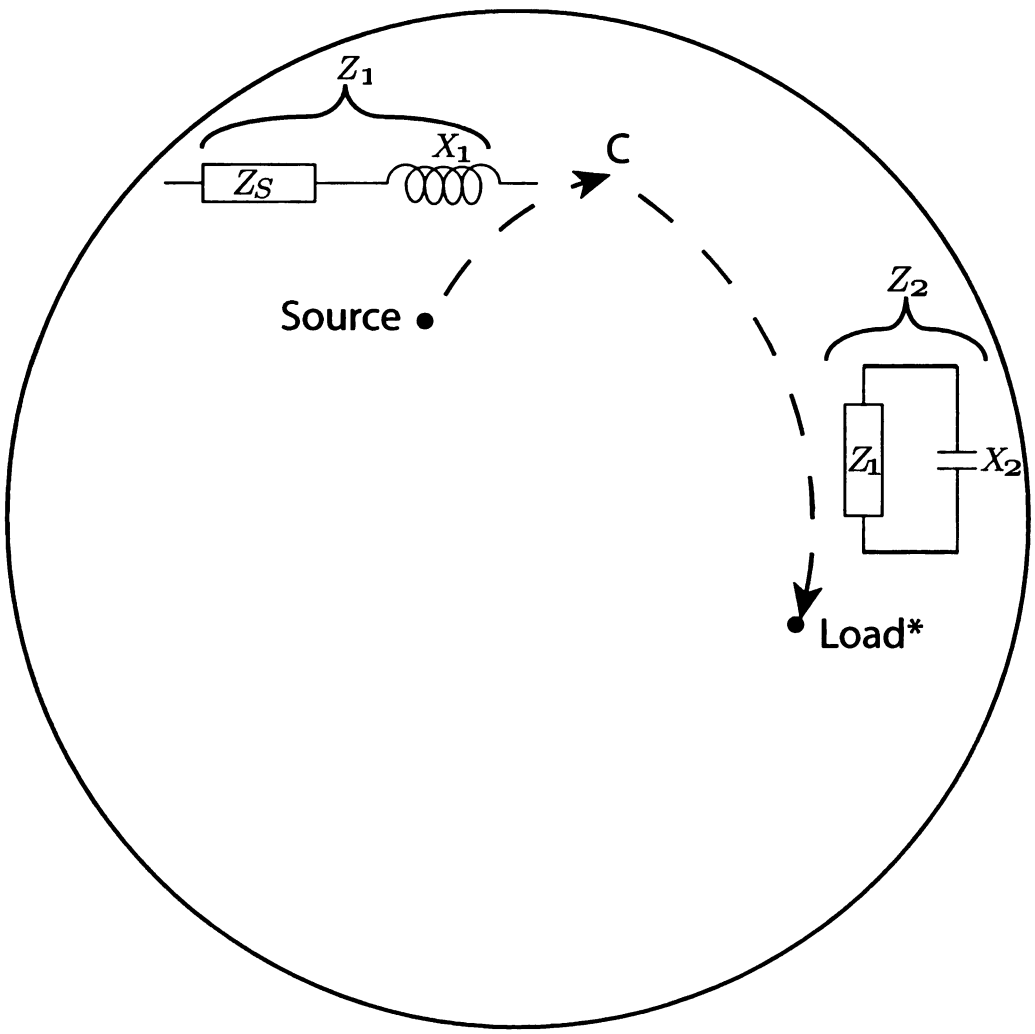


Figure 3.94: Transformed start and load impedances on a Smith chart.

The source impedance is

$$Z_S = R_S + jX_S. \tag{3.21}$$

The load impedance conjugate is given by

$$Z_L = R_L + jX_L, \tag{3.22}$$

$$Z_L^* = R_L - jX_L. \tag{3.23}$$

The source impedance is transformed to Z_1 at point C by adding X_1 in series as is shown in Equation (3.24).

$$Z_1 = R_S + j(X_S + X_1) \tag{3.24}$$

$$\tag{3.25}$$

The admittance Y_1 at point C is

$$Y_1 = \frac{1}{Z_1} = \frac{1}{R_S + j(X_S + X_1)} \tag{3.26}$$

$$= \frac{1}{R_S + j(X_S + X_1)} \frac{R_S - j(X_S + X_1)}{R_S - j(X_S + X_1)} \tag{3.27}$$

$$= \frac{R_S}{R_S^2 + (X_S + X_1)^2} - j\frac{(X_s + X_1)}{R_S^2 + (X_s + X_1)^2} \tag{3.28}$$

The admittance Y_1 is transformed to the admittance Y_2 by adding $G_2 = 1/X_2$ in parallel with Y_1.

$$Y_2 = \frac{R_S}{R_S^2 + (X_S + X_1)^2} - j\left[\frac{(X_s + X_1)}{R_S^2 + (X_s + X_1)^2} + G_2\right] \tag{3.29}$$

The admittance $Y(L^*)$ from eq (3.23) is

$$Y_L^* = \frac{1}{Z_L^*} = \frac{1}{R_L - jX_L} \tag{3.30}$$

$$= \frac{1}{R_L - jX_L} \frac{R_L + jX_L}{R_L + jX_L} \tag{3.31}$$

$$= \frac{R_L + jX_L}{R_L^2 + X_L^2} \tag{3.32}$$

$$Y_L^* = \frac{R_L}{R_L^2 + X_L^2} + j\frac{X_L}{R_L^2 + X_L^2} \tag{3.33}$$

$$Y_L^* = G_L^* + jB_L^* \tag{3.34}$$

The admittances Y_L^* and Y_2 are equal. Setting the real and imaginary parts equal

$$\frac{R_L}{R_L^2 + X_L^2} = \frac{R_S}{R_S^2 + (X_S + X_1)^2} \tag{3.35}$$

$$G_L^* = \frac{R_S}{R_S^2 + (X_S + X_1)^2} \tag{3.36}$$

$$j\frac{X_L}{R_L^2 + X_L^2} = -j\left[\frac{(X_s + X_1)}{R_S^2 + (X_s + X_1)^2} + G_2\right] \tag{3.37}$$

$$B_L^* = -j\left[\frac{(X_s + X_1)}{R_S^2 + (X_s + X_1)^2} + G_2\right] \tag{3.38}$$

In Equation 3.36 the known quantities are the conjugate load conductance (G_L^*), the source resistance (R_S), and the source reactance (X_S). Solving Equation 3.36 for X_1 yields two answers, a value for a low pass and a high pass matching network. In Equation 3.38 the known quantities are the conjugate load susceptance (B_L^*), the source reactance (X_S), the source resistance (R_S), and the reactance X_1 from Equation 3.36. Solving Equation 3.38 for two answers, a value for a low pass and a high pass matching network.

The Matlab code to implement the L network matching component calculations is shown in Listing 3.1. If $R_S < R_L$ then a series element is attached between the

source and load (X_1) and an element is placed in parallel with the load (G_2). If $R_S > R_L$ then a series element is attached between the source and load (X_2) and an element is placed in parallel with the source (G_1). If X_1 is negative then E1 is a capacitor and E2 is an inductor. If X_1 is positive then E1 is an inductor and E2 is a capacitor. If X_2 is negative then E3 is a capacitor and E4 is an inductor. If X_2 is positive then E3 is an inductor and E4 is a capacitor.

Listing 3.1: L matching network calculations Matlab code.

```
function [E1,E2,E3,E4] = L_matching(z_source,z_load,freq)

% L_matching(z_source,z_load,freq)
%
% This function determines the capacitance and inductance
% for elements in L matching networks. The inputs are the
% source and load impedance and the operating frequency.
% The outputs are the capacitance and inductance of the low
% pass L matching network followed by the capacictance and
% inductance of the high pass L matching network.
%
% The elements are returned in the following structure:
% struct('value',0,'prefix','','unit','','topology','')
% value:    {Numeric value of component}
% prefix:   Unused
% unit:     {'H','F'}. This indicates if the element is an
%           inductor, H for Henries, or a capacitor, F for
%           Farads.
% topology: {'Source_Parallel', 'Source_Series',
%            'Load_Parallel', 'Load_Series'}.
%           This indicates where the element is in the L
%           network. 'Source_Parallel' indicates in
%           parallel with the source. 'Load_Parallel'
%           indicates in parallel with the load.
%           'Source_Series' and 'Load_Series' indicate the
%           element is in series between the load and
```

```
%           source.
%
% Example: [E1 E2 E3 E4] = L_matching(50, 25 + 10j, 1e6)

E1 = struct( 'value',0,'prefix','','unit','','topology','');
E2 = struct(E1);
E3 = struct(E1);
E4 = struct(E1);

if real(z_source) < real(z_load)
% Series element attached to source.
% Parallel element attached to load.
        R = real(z_source);
        X = imag(z_source);
        y = 1/z_load;
        ys = conj(y);
        Gs = real(ys);
        Bs = imag(ys);
else
% Series element attached to load.
% Parallel element attached to source.
        R = real(z_load);
        X = imag(z_load);
        y = 1/z_source;
        ys = conj(y);
        Gs = real(ys);
        Bs = imag(ys);
```

```
end


        Be = subs(solve('R=(Gs)/(Gs^2 + (Bs + B)^2)','B'));
        Xe = (-(Bs + Be)./(Gs^2 + (Bs + Be).^2)) - X;



if real(z_source) > real(z_load)
% Source element moves along constant conductance circles.
% Load element moves along constant resistance circles.
% Series element attached to source.
% Parallel element attached to load.
        if Xe(1) < 0
        % X1 Capacitive
        % X2 Inductive
                E1.value = -1/(Be(1)*2*pi*freq);
                E1.unit = 'H';
                E1.topology='Source Parallel';
                E2.value = 1/(Xe(1)*2*pi*freq);
                E2.unit = 'F';
                E2.topology='Load Series';
        else
        % X1 Inductive
        % X2 Capacitive
                E1.value = Be(1)/(2*pi*freq);
                E1.unit = 'F';
                E1.topology='Source Parallel';
                E2.value = -Xe(1)/(2*pi*freq);
```

```
            E2.unit = 'H';
            E2.topology='Load_Series';
        end

        if Xe(2) < 0
        % X3 Capacitive
        % X4 Inductive
            E3.value = -1/(Be(2)*2*pi*freq);
            E3.unit = 'H';
            E3.topology='Source_Parallel';
            E4.value = 1/(Xe(2)*2*pi*freq);
            E4.unit = 'F';
            E4.topology='Load_Series';
        else
        % X3 Inductive
        % X4 Capacitive
            E3.value = Be(2)/(2*pi*freq);
            E3.unit = 'F';
            E3.topology='Source_Parallel';
            E4.value = -Xe(2)/(2*pi*freq);
            E4.unit = 'H';
            E4.topology='Load_Series';
        end
else
% Source element moves along constant resistance circles.
% Load element moves along constant conductance circles.
% Series element attached to load.
```

```
% Parallel element attached to source.
        if Xe(1) < 0
        % X1 Capacitive
        % X2 Inductive
                E1.value = -1/(Xe(1)*2*pi*freq);
                E1.unit = 'F';
                E1.topology='Source Series';
                E2.value = 1/(Be(1)*2*pi*freq);
                E2.unit = 'H';
                E2.topology='Load Parallel';
        else
        % X1 Inductive
        % X2 Capacitive
                E1.value = Xe(1)/(2*pi*freq);
                E1.unit = 'H';
                E1.topology='Source Series';
                E2.value = -Be(1)/(2*pi*freq);
                E2.unit = 'F';
                E2.topology='Load Parallel';
        end

        if Xe(2) < 0
        % X3 Capacitive
        % X4 Inductive
                E3.value = -1/(Xe(2)*2*pi*freq);
                E3.unit = 'F';
                E3.topology='Source Series';
```

```
            E4.value = 1/(Be(2)*2*pi*freq);
            E4.unit = 'H';
            E4.topology='Load Parallels';
        else
        % X3 Inductive
        % X4 Capacitive
            E3.value = Xe(2)/(2*pi*freq);
            E3.unit = 'H';
            E3.topology='Source Series';
            E4.value = -Be(2)/(2*pi*freq);
            E4.unit = 'F';
            E4.topology='Load Parallels';
        end
end
```

Figure 3.95 shows the matching networks. The L matching network calculated for each channel is listed in Table 3.8. The output impedance of the amplifier (Amp_{out}) is assumed to be 50Ω and the transducer impedance is assumed to be larger than 50Ω. The matching network with values E1 and E2 is used because it is a high pass filter and any DC signal across Amp_{out} will be blocked.

The resulting impedance transformations of the transducers with the matching networks are summarized in Table 3.9 and shown in Figures 3.96 - 3.108.

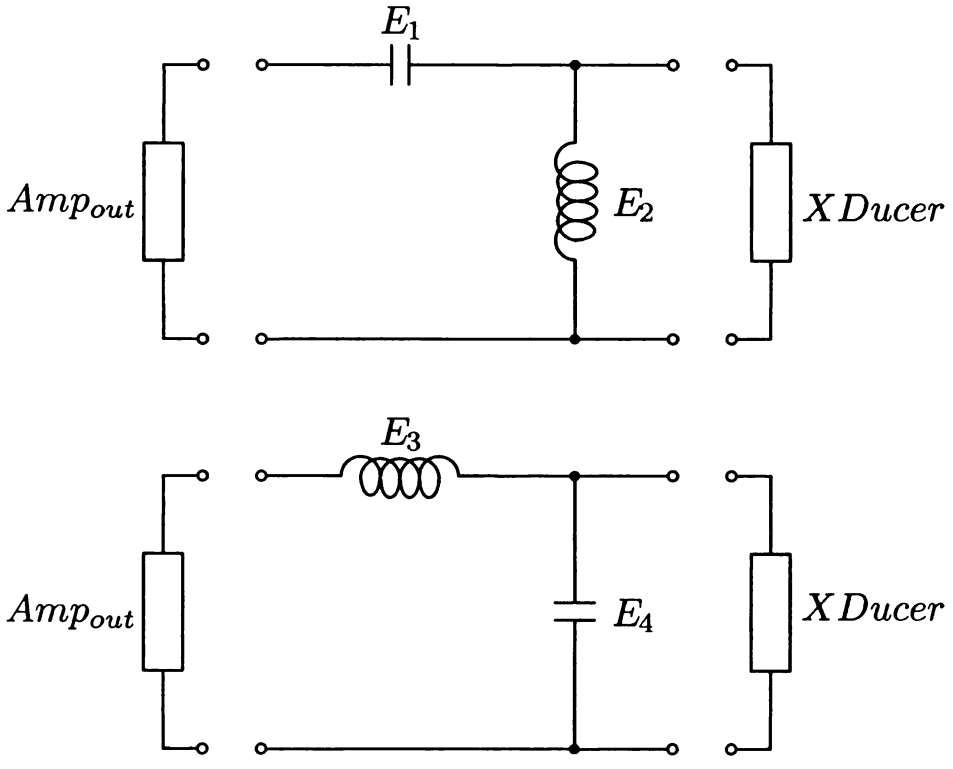


Figure 3.95: L matching networks for the array transducers.

Channel	E1(Ω)	E2(Ω)	E3(Ω)	E4(Ω)
1	2.11e-08F	2.49e-04H	1.87e-04H	1.71e-08F
2	2.07e-08F	2.57e-04H	1.91e-04H	1.72e-08F
3	1.99e-08F	2.32e-04H	1.99e-04H	1.48e-08F
4	2.15e-08F	2.25e-04H	1.84e-04H	1.57e-08F
5	2.15e-08F	2.22e-04H	1.84e-04H	1.54e-08F
6	1.96e-08F	2.57e-04H	2.02e-04H	1.62e-08F
7	1.72e-08F	2.71e-04H	2.30e-04H	1.44e-08F
8	1.49e-08F	2.53e-04H	2.66e-04H	1.05e-08F
9	2.78e-08F	2.06e-04H	1.42e-04H	1.81e-08F
10	2.05e-08F	2.40e-04H	1.93e-04H	1.59e-08F
11	2.27e-08F	2.10e-04H	1.74e-04H	1.54e-08F
12	2.03e-08F	2.28e-04H	1.95e-04H	1.49e-08F
13	1.98e-08F	2.18e-04H	2.00e-04H	1.36e-08F

Table 3.8: Matching network element values.

Channel	Magnitude(Ω)	Phase(Degrees)
1	50.32	0.84
2	50.19	-0.37
3	49.65	-0.31
4	50.31	0.34
5	50.18	0.19
6	48.81	0.93
7	49.09	0.30
8	50.12	-0.55
9	50.96	0.28
10	49.79	0.54
11	49.71	0.66
12	50.13	0.80
13	49.70	0.34

Table 3.9: Impedances of matched transducers.

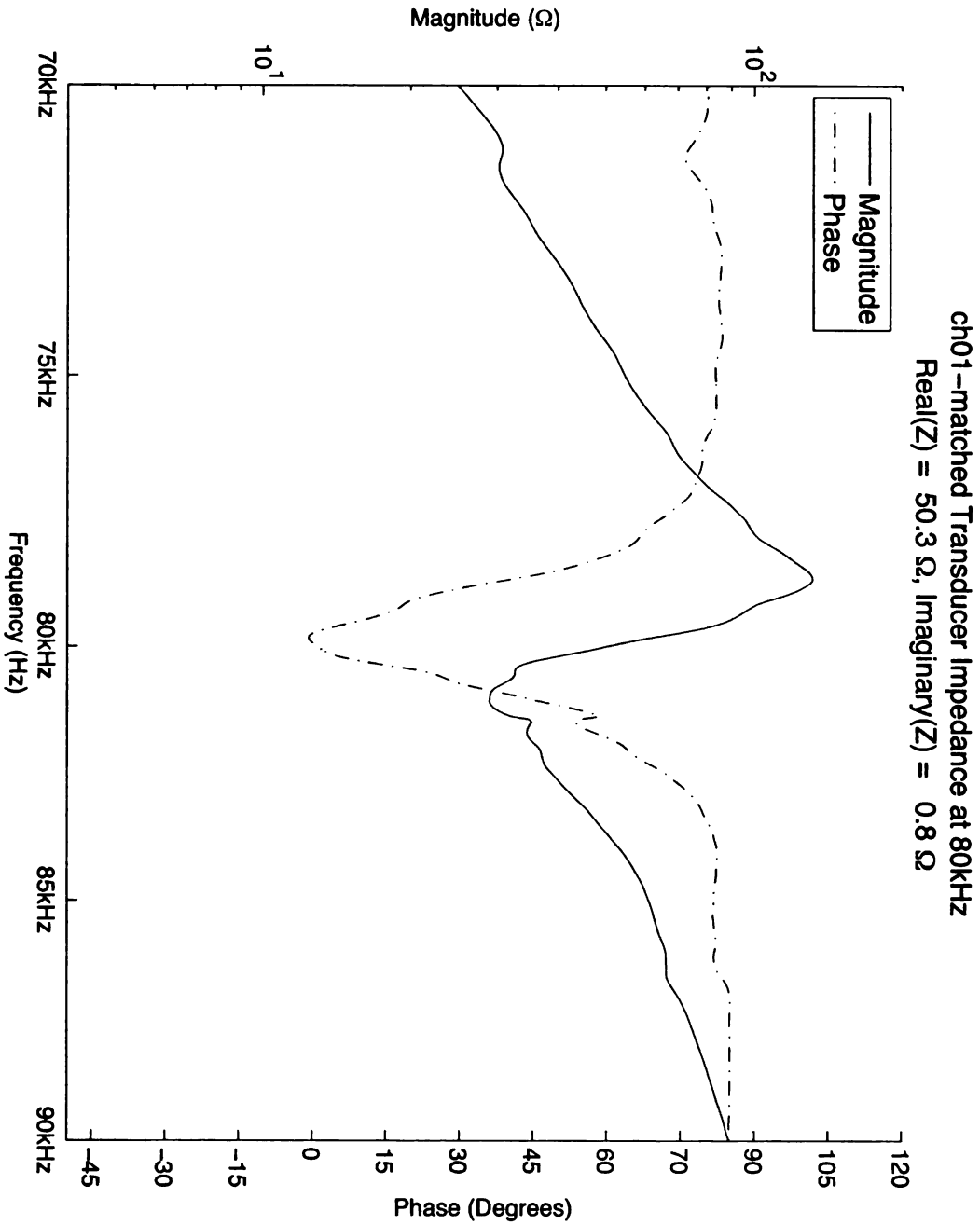


Figure 3.96: Channel 1 matched transducer impedance.

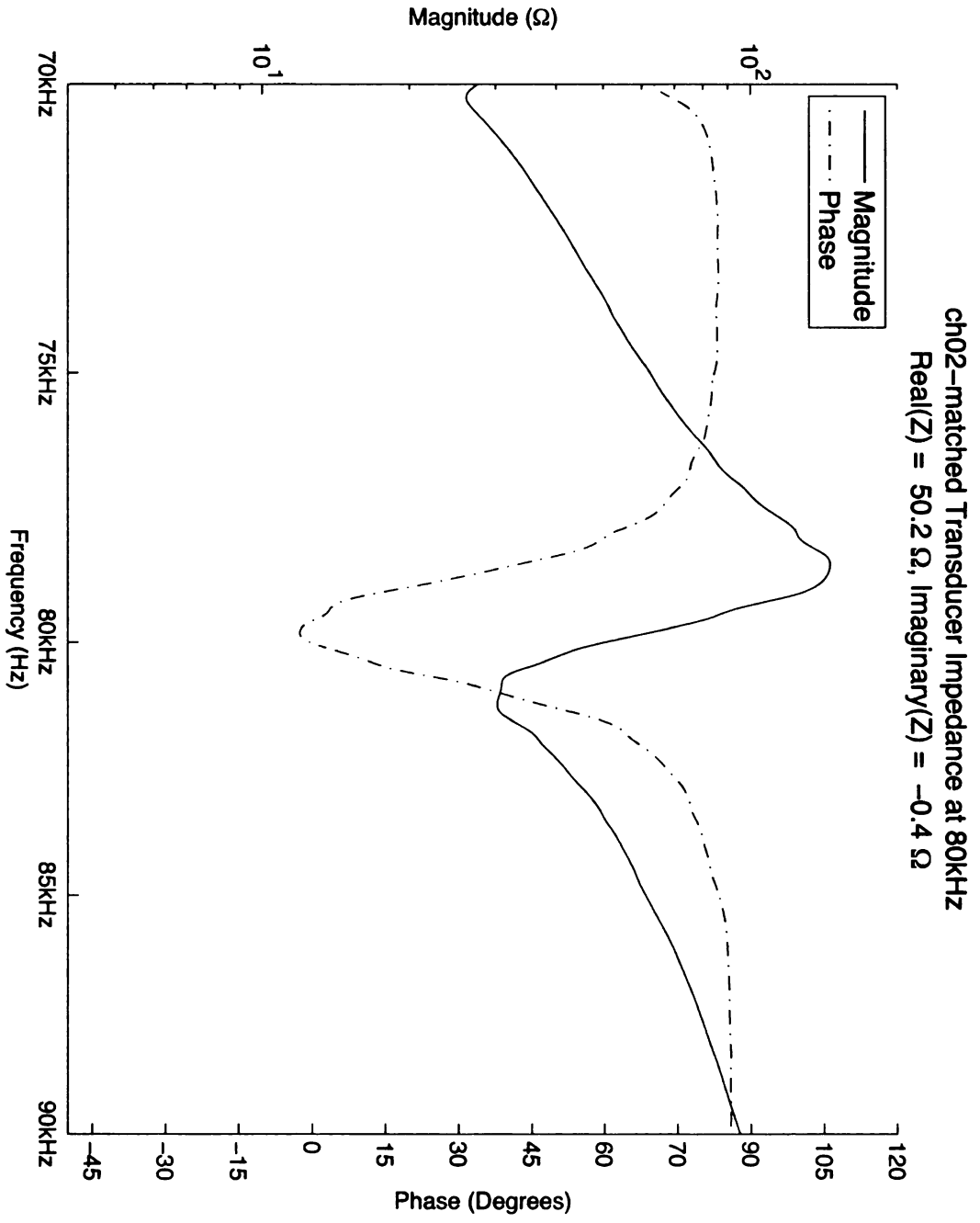


Figure 3.97: Channel 2 matched transducer impedance.

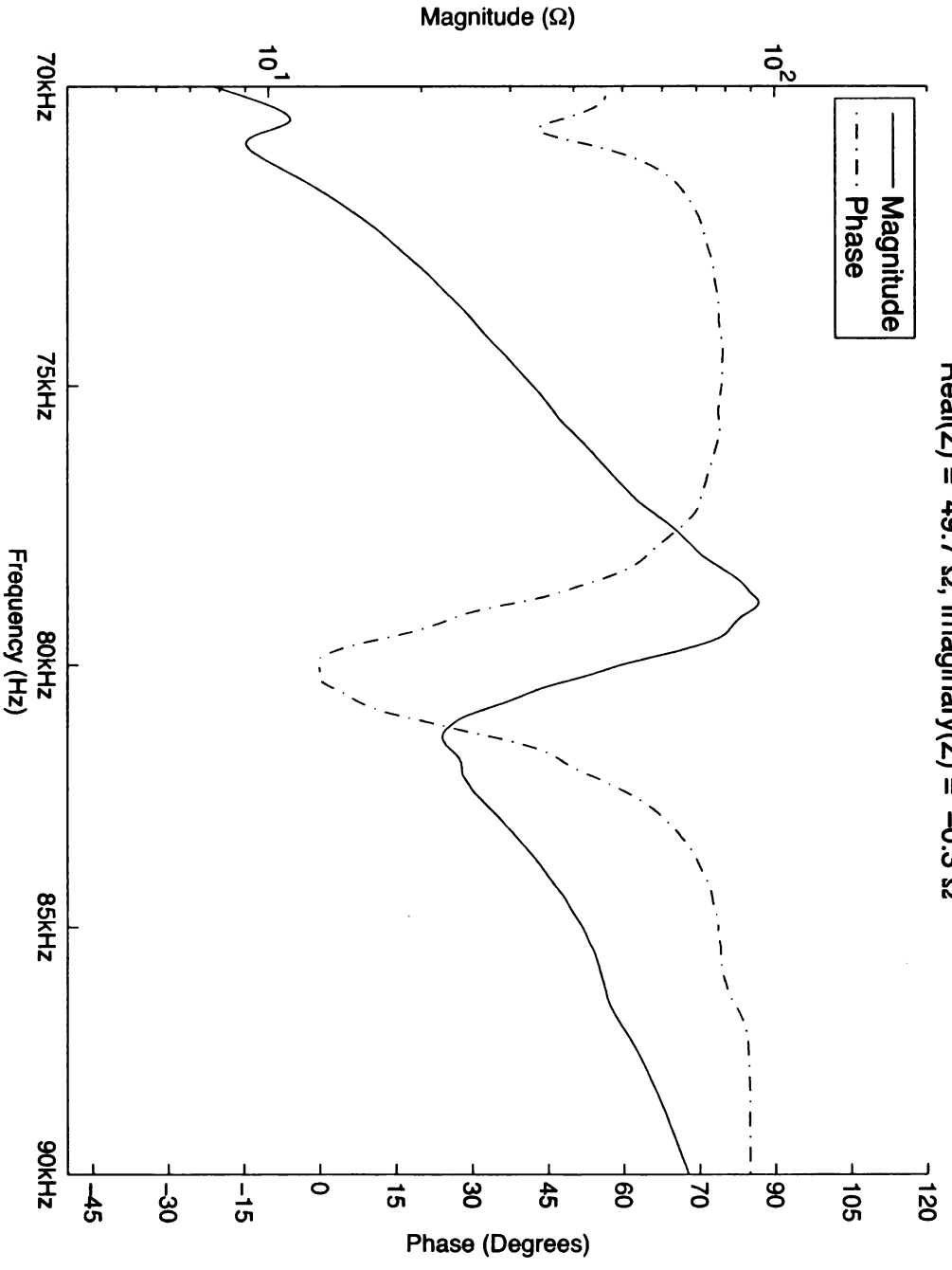


Figure 3.98: Channel 3 matched transducer impedance.

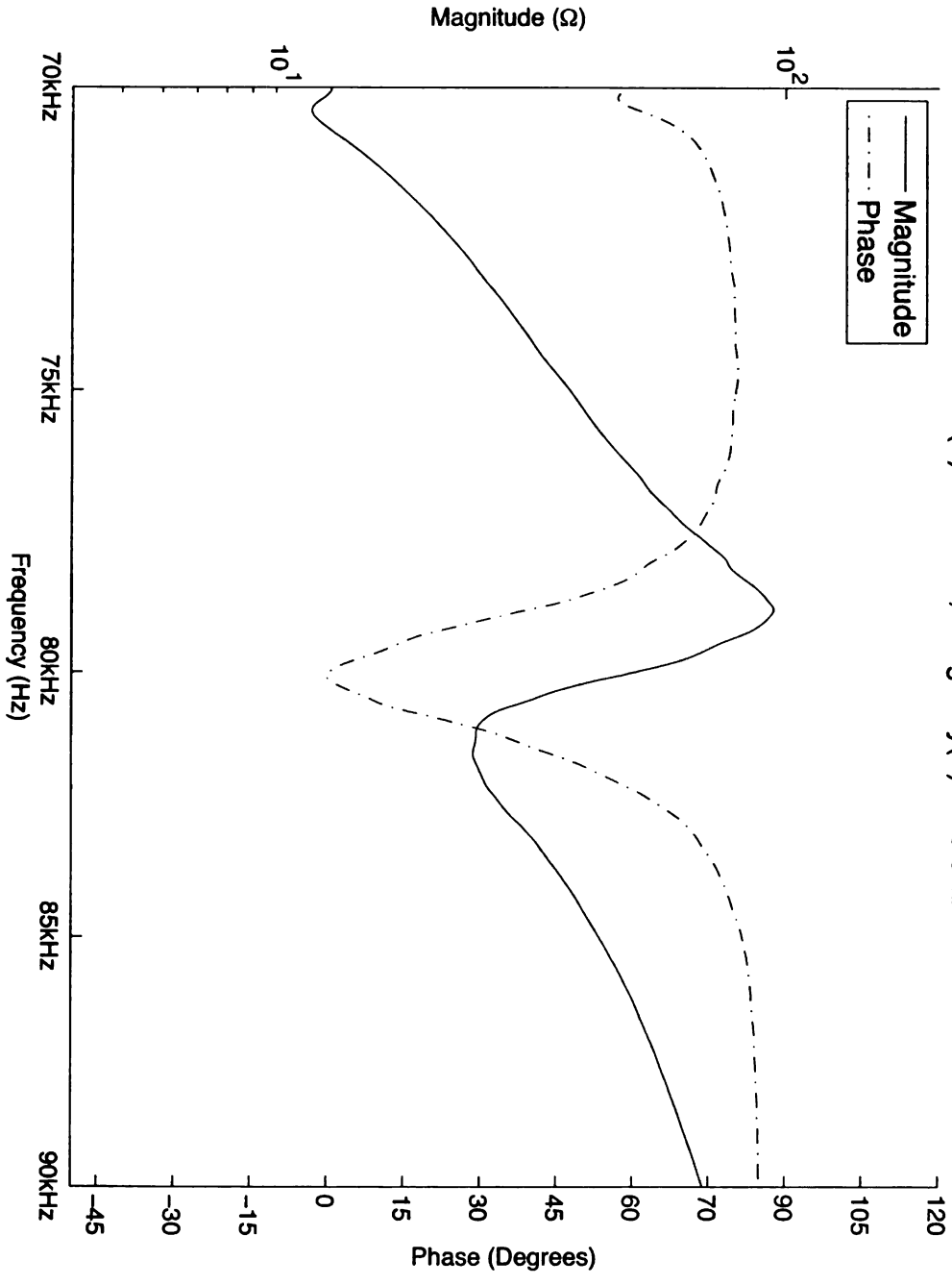


Figure 3.99: Channel 4 matched transducer impedance.

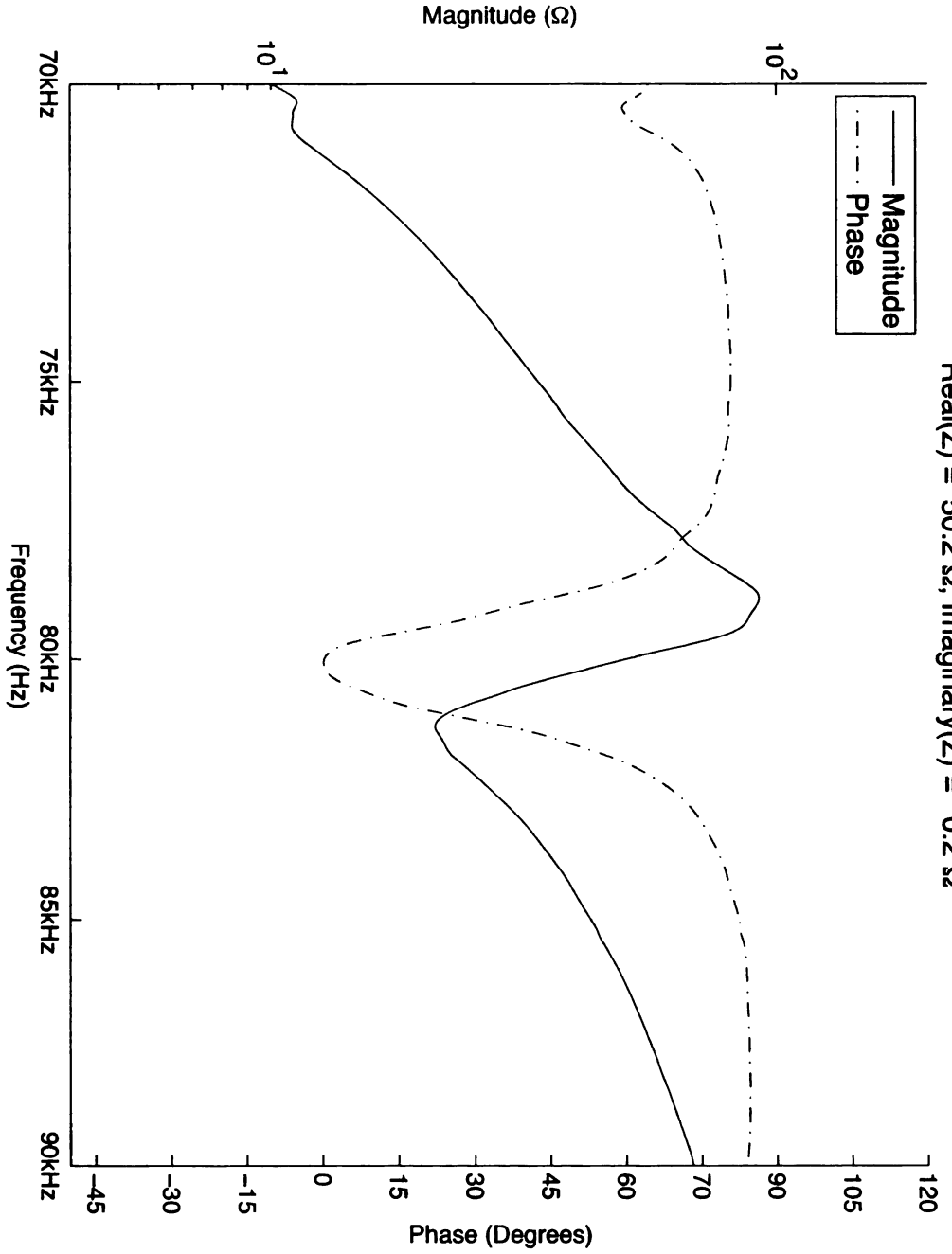


Figure 3.100: Channel 5 matched transducer impedance.

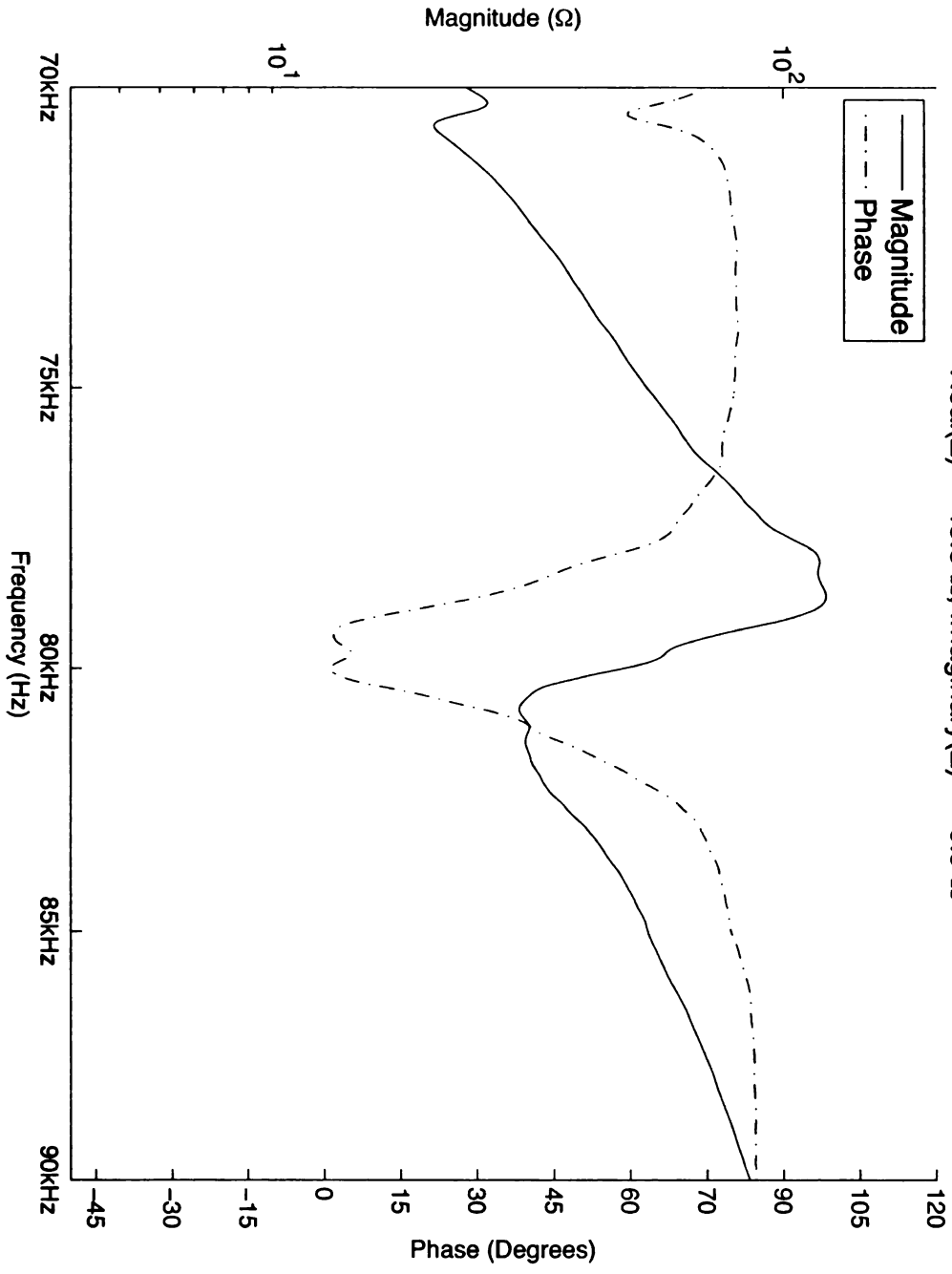


Figure 3.101: Channel 6 matched transducer impedance.

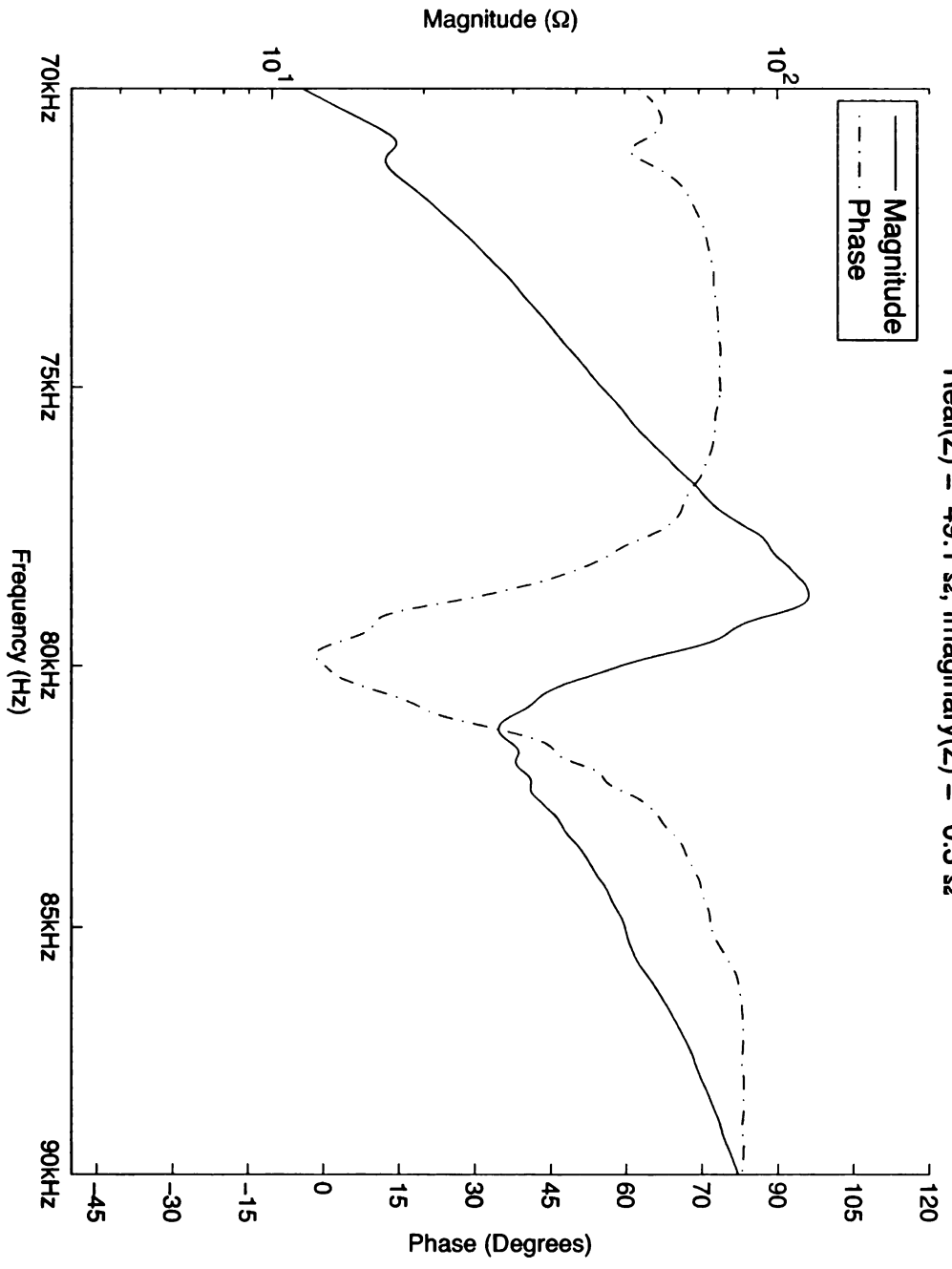


Figure 3.102: Channel 7 matched transducer impedance.

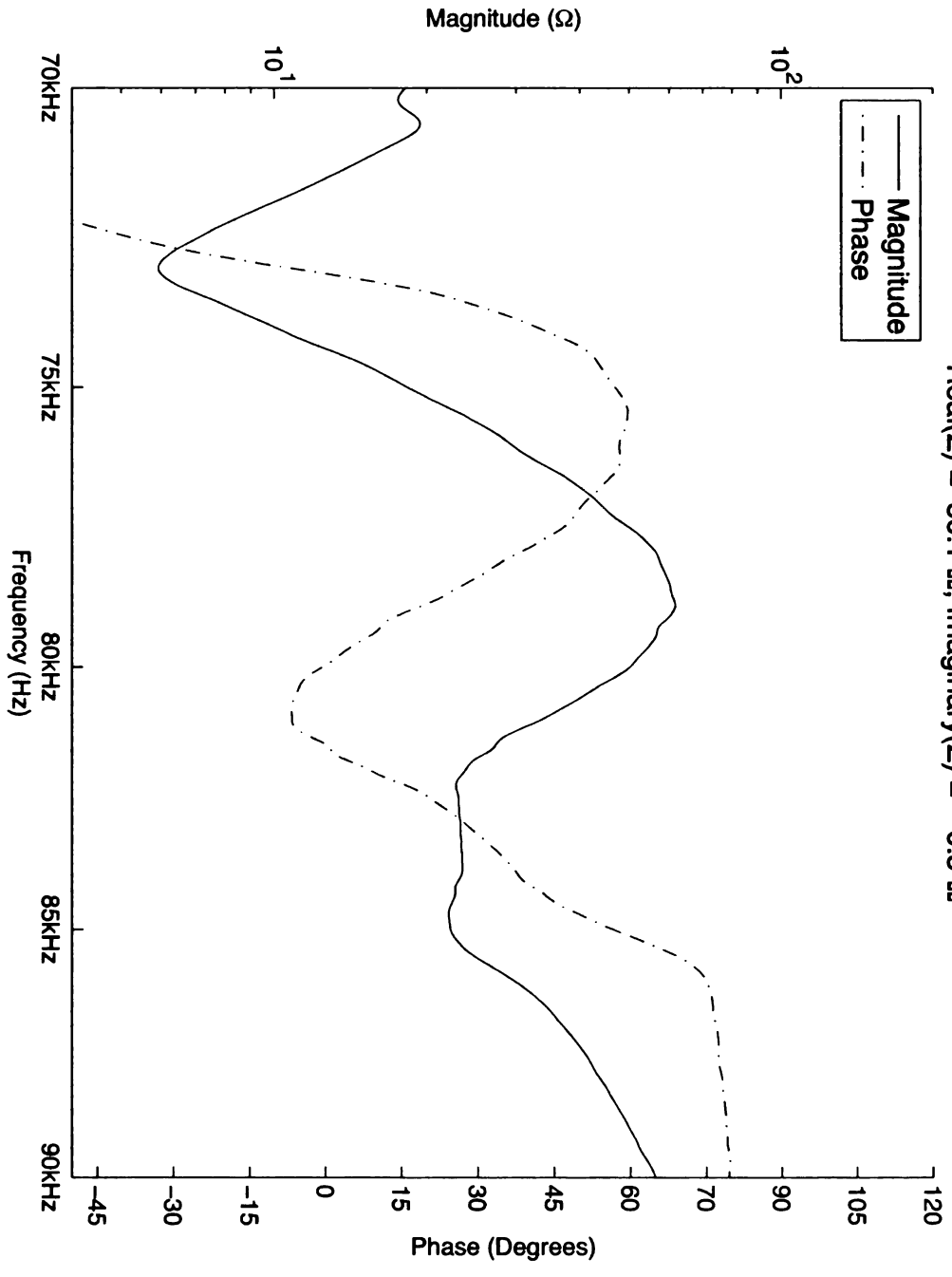


Figure 3.103: Channel 8 matched transducer impedance.

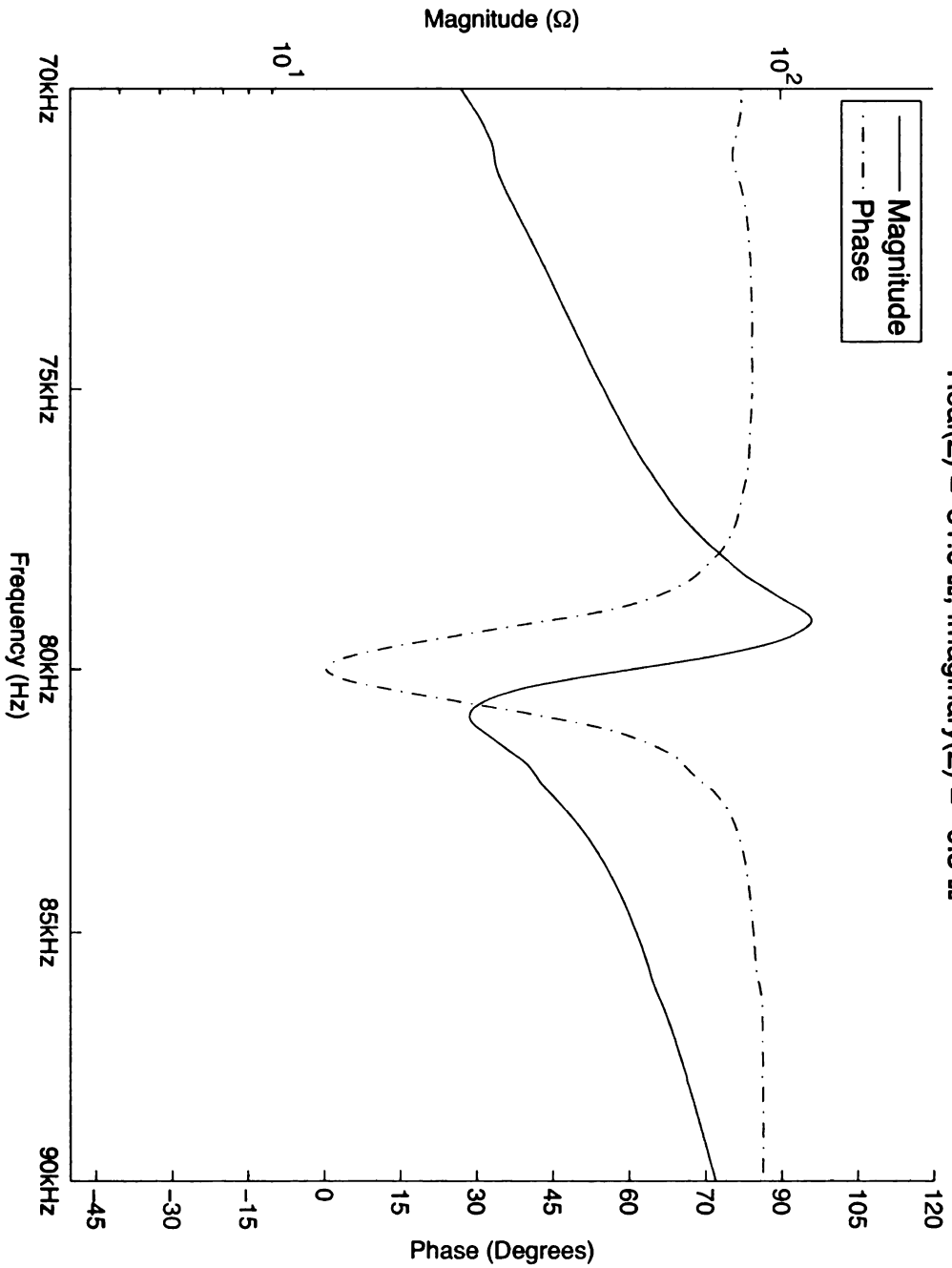


Figure 3.104: Channel 9 matched transducer impedance.

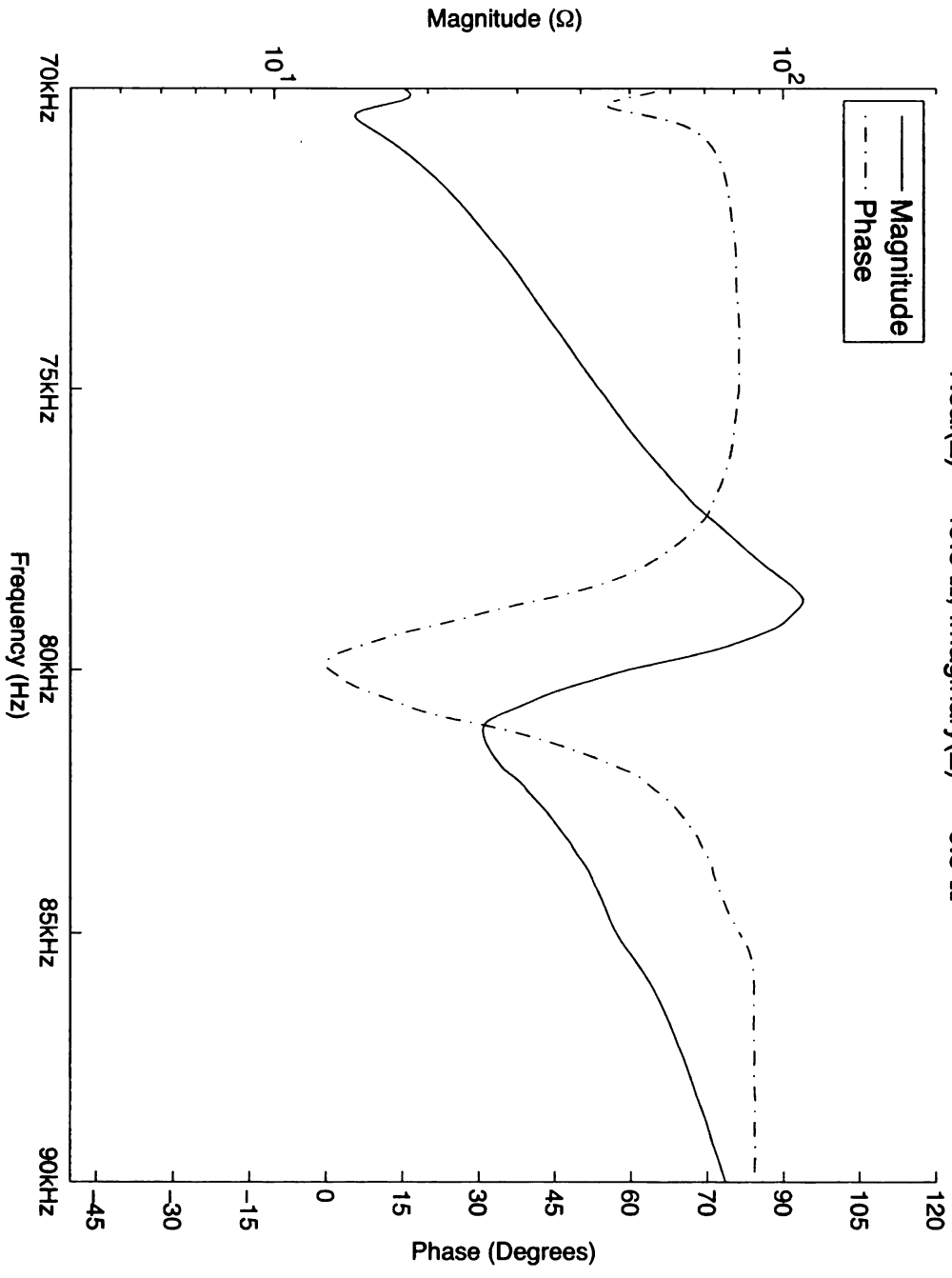


Figure 3.105: Channel 10 matched transducer impedance.

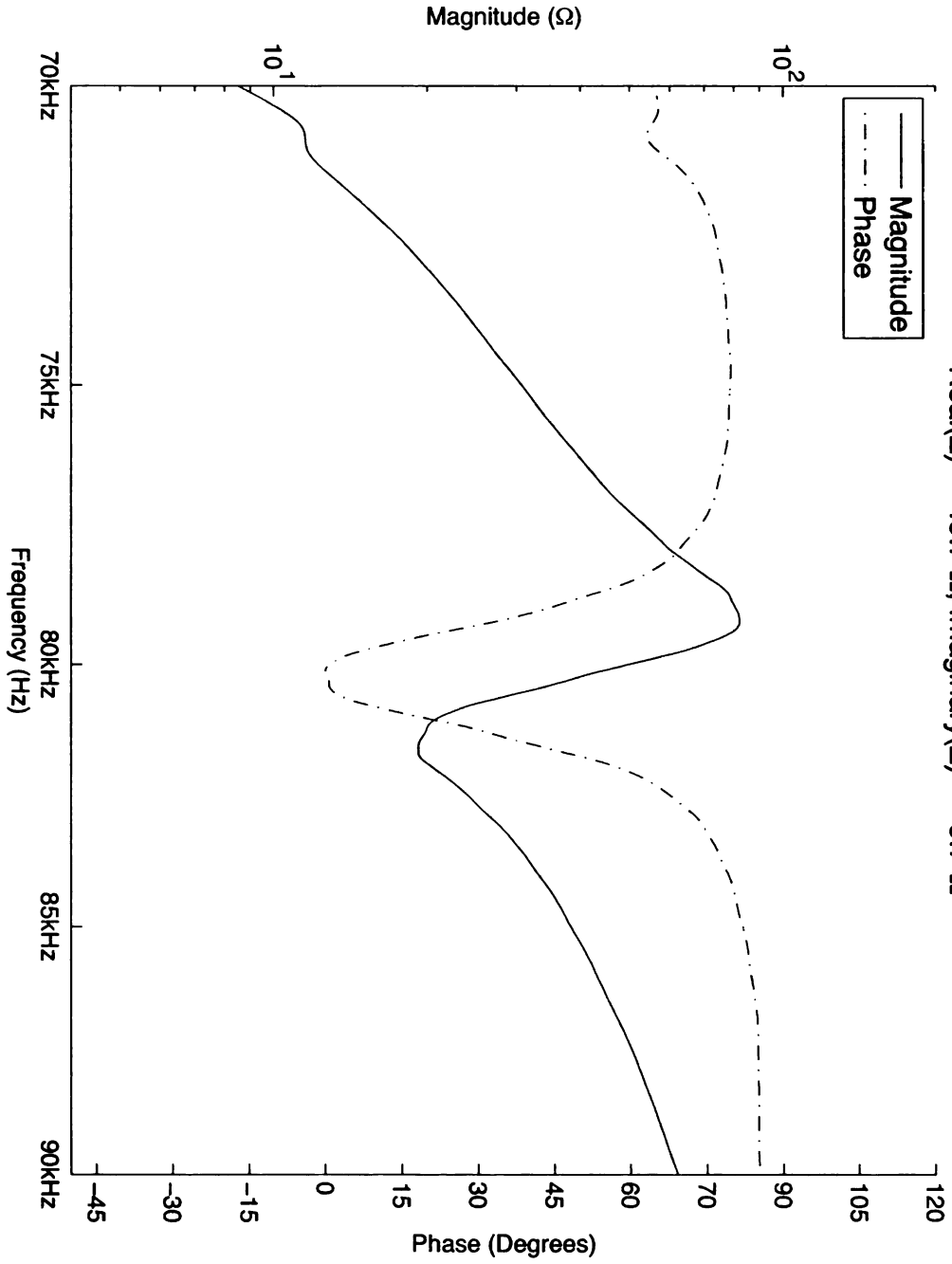


Figure 3.106: Channel 11 matched transducer impedance.

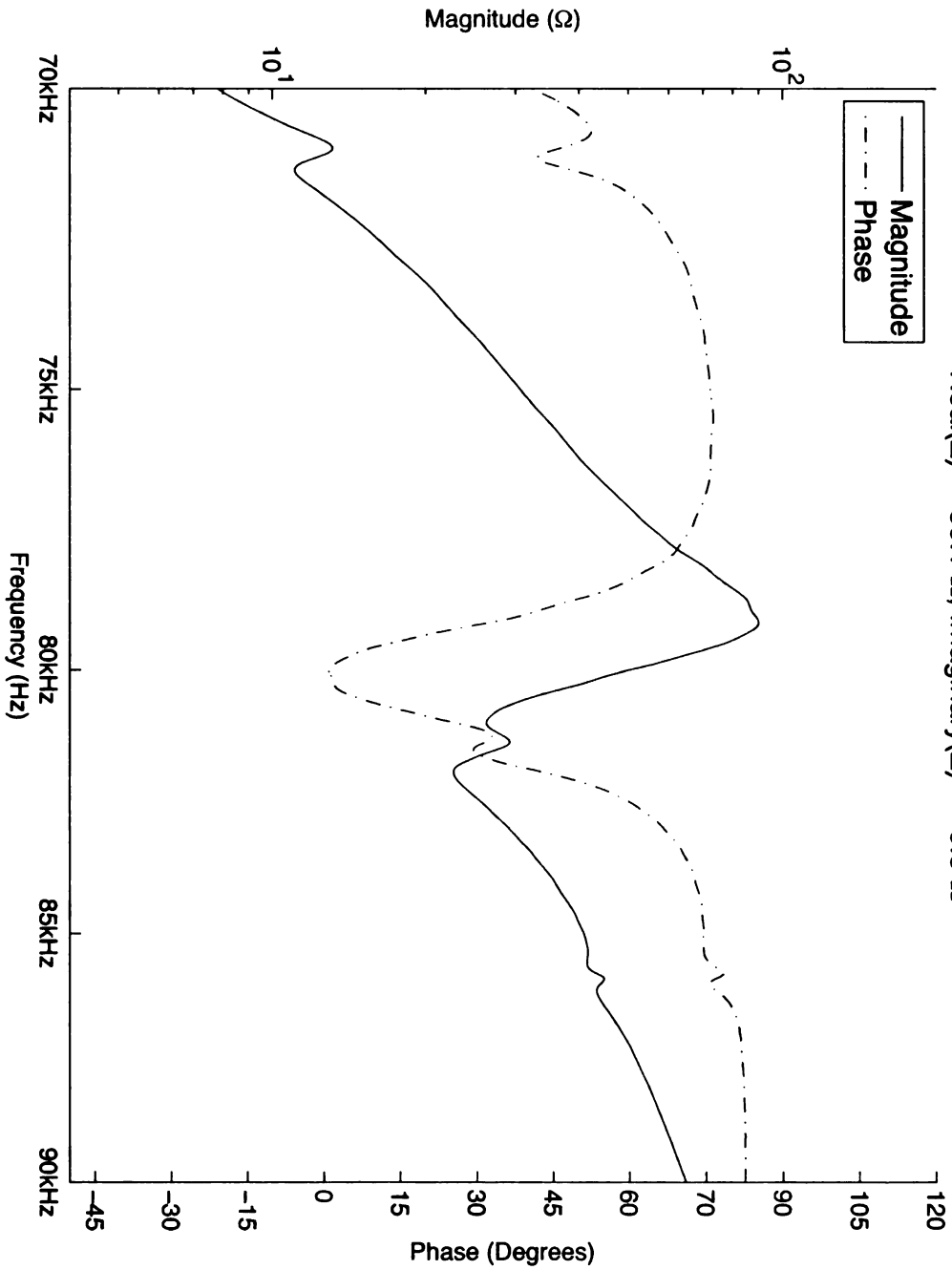


Figure 3.107: Channel 12 matched transducer impedance.

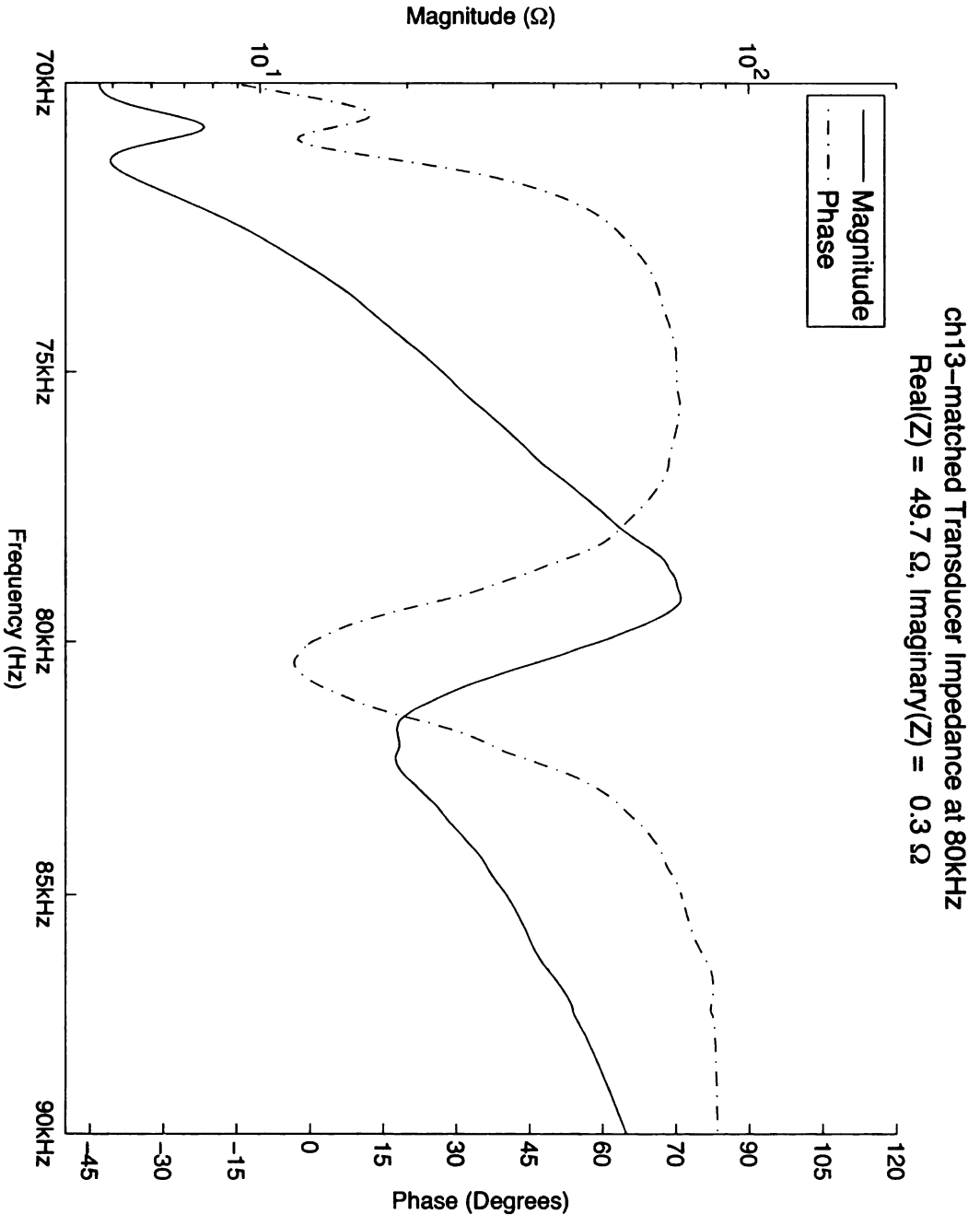


Figure 3.108: Channel 13 matched transducer impedance.

Chapter 4

Pressure Field Measurement

A pressure field measurement entails positioning the hydrophone in the desired location, exciting the transducer array, and capturing the resulting pressure data. This process is repeated for each location in the desired area over which the pressure field is measured. In order to focus the field in a particular location, the delays for each transducer must be set such that the pressure from each transducer constructively interferes at the desired location.

4.1 Hydrophone Positioning

The hydrophone is positioned using a Parker® 3-D positioning system. The positioner is controlled by the computer over an RS-232 port. Each axis is independently controlled by passing character string commands over the serial port. Each controller will echo the command and send back any requested information such as the status of the controller. Control was implemented using Matlab m-files. Table 4.1 summarizes the commands that control the positioning system.

Command	Description
pos_home	Send the positioner to the home position.
pos_in_position	Check to see if a specified axis or all axis are in position.
pos_init	Initialize the positioner communication settings and motor profiles and send to the home position.
pos_is_moving	Check to see is a specified axis or all axis are moving.
pos_map	Define the geometry of the positioning system.
pos_move	Move the positioner in 3 dimensions specified in mm.
pos_move_axis1	Move the positioner along the X-axis specified in mm.
pos_move_axis2	Move the positioner along the Y-axis specified in mm.
pos_move_axis3	Move the positioner along the Z-axis specified in mm.
pos_move_step	Move the positioner in 3 dimensions specified in steps.
pos_read	Read a line from the motor controller and return it as a string.
pos_read_all	Read lines from the motor controller and return them as several strings until an empty string is read.
pos_read_register	Read the contents of a specified register.
pos_write	Write a command over the serial port to the motor controllers and return the response.

Table 4.1: Positioner Matlab commands.

4.2 Hydrophone Measurements

The relative pressure field value is collected from the output of the hydrophone, which passes through a pre-amplifier into an oscilloscope. The oscilloscope digitizes the analog waveform, and the resulting digital waveform is sent to the computer over the GPIB bus. Basic commands to interface with the GPIB bus were implemented with Matlab using m-files. The commands are summarized in Table 4.2. Using the GPIB commands and other Matlab commands, higher function m-files were created to control and communicate with the oscilloscope. These are summarized in Table 4.3.

4.3 Calibration

In order to focus the field in a particular location, the delays for each transducer must be set such that the pressure from each transducer constructively interferes at

Command	Description
GPIB_close	Close the specified GPIB port.
GPIB_init	Open and initialize the specified GPIB port.
GPIB_query	Send a command and read back the response into a string.
GPIB_read	Read a single line from the GPIB port and place it in a string.
GPIB_read_all	Read a line from the GPIB port and place it in a string until a null string is read.
GPIB_write	Write a command to a specified GPIB port.

Table 4.2: GPIB Matlab commands.

Command	Description
scope_capture_wave_fast	Capture the waveform from the oscilloscope and return the data array. This function does not set the GPIB address, save the data, or return the time array.
scope_init	Initialize the oscilloscope to a predefined state. This file should be copied and editied for each experiment so that the state is preserved for each experiment and not overwritten by subsequent ones.
scope_make_time_array	Query the oscilloscope to determine the time settings and number of points a waveform and then return a time array that matches this criteria.
scope_set_channel	Change the selected channel on the oscilloscope.
scope	This is a macro command that sets the oscilloscope resolution, captures the waveform from the oscilloscope, saves the data to a specified file, and displays a plot of the data.

Table 4.3: Oscilloscope Matlab commands.

the desired location. There are two ways to determine the delays needed to focus the pressure field at a specific location: measure the delays from each transducer at the desired location and use correlation to determine the needed delays; calculate the delays based on the geometry of the transducer array, the frequency at which the transducers are driven at, and the position of the desired focus [17]. The geometry of the array has yet to be measured in detail so the former method is used. Figure 4.1 shows the responses from each transducer superimposed on each other. The delays were calculated, and Figure 4.2 shows the resulting waveforms measured at the same location with the applied delays. Table 4.4 contains a summary of commands that

are used for the calibration process. Listing 4.1 shows an example m-file that focuses the array at a single location.

Command	Description
calibrate_all_transducers	A macro function to calibrate all transducers at a specified position and save the calibration data to a specified directory.
calibrate_reference	This function measures and records the hydrophone waveforms at a specified location.
calibrate_single_transducer	A macro function to calibrate a single transducer at a specified position and save the calibration data to a specified directory.
delay_calculations	This function calculates the delays needed to make the pressure from each transducer coherent at a sampled position. The waveforms from a calibration test with no delays applied are read in from the specified directory and root file name.
delay_check	This function measures and records the hydrophone waveforms at a specified location using calculated delays.
delay_reference	This function calculates the delays from saved waveform responses for each transducer at one location and saves the results to a file.
plot_calibrated_delayed	This function plots the waveforms saved from the calibrate_all_transducers function. The directory and file names should be the same that were used when calling calibrate_all_transducers. The waveforms are saved in PDF format.
plot_calibrated_reference	This function plots the waveforms saved from the calibrate_reference function. The directory and file names should be the same that were used when calling calibrate_reference. The waveforms are saved in PDF format.

Table 4.4: Calibration Matlab commands.

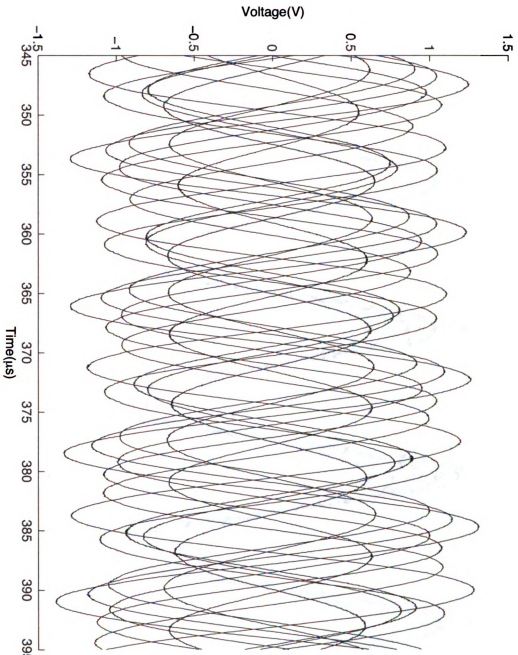


Figure 4.1: Superimposed responses of each individually excited transducer with no time delay.

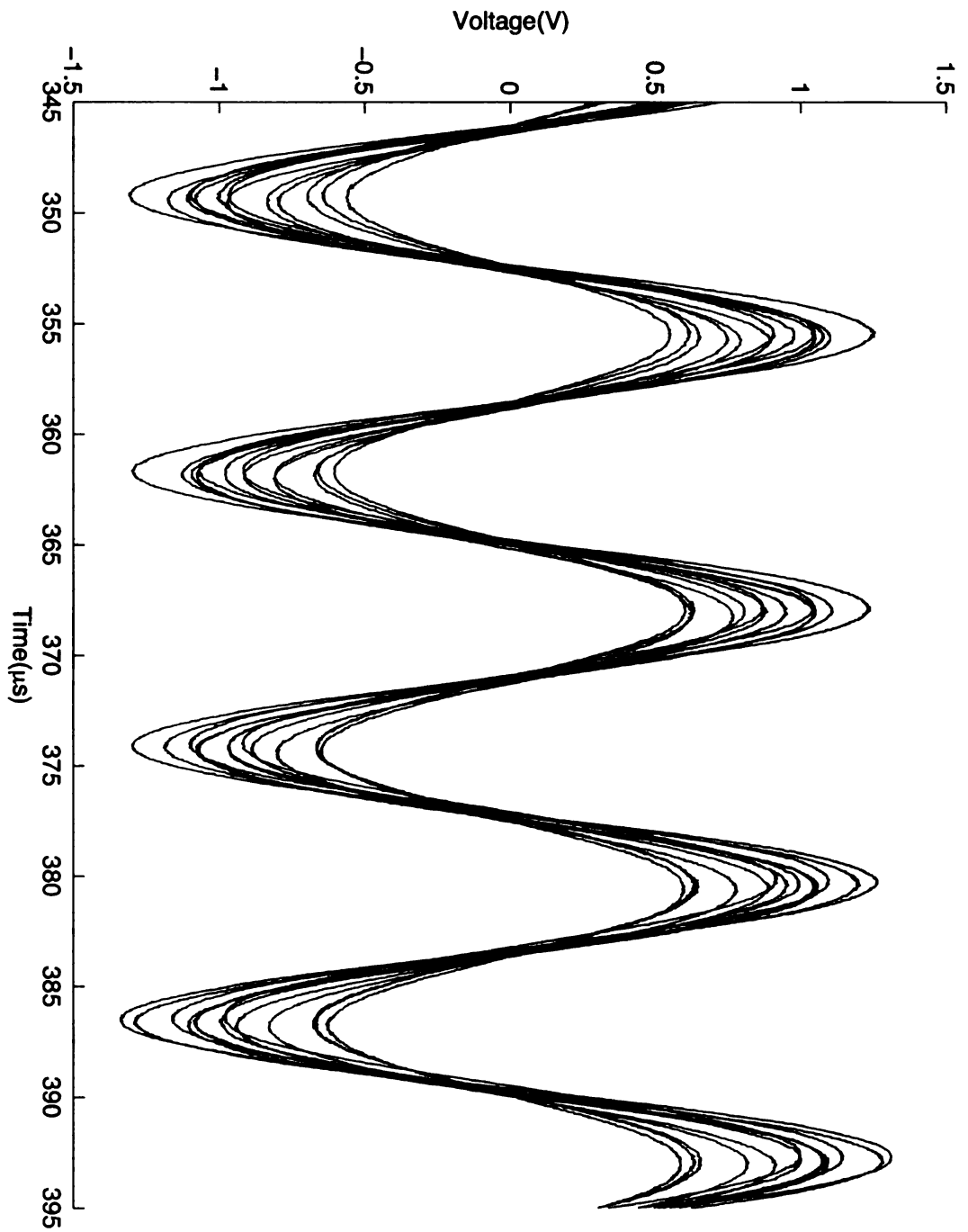


Figure 4.2: Superimposed responses of each individually excited transducer with calculated delays.

Listing 4.1: Example Matlab m-file that measures the transducer response.

```
position = [0 -10 -25]
directory = '(x,x,-25)';
file =    '(0,-10,-25)';
calibrate_reference(position,directory,file)
delay_reference(directory,file)
delay_check(position,directory,file)
plot_calibrated_reference(directory,file)
plot_calibrated_delayed(directory,file)
```

4.4 Pressure Observation

The positioner and oscilloscope commands are used with other Matlab commands to define functions that measure the pressure in the water tank. These commands are summarized in Table 4.5. Listing 4.2 is an m-file example of a pressure field observation. The delays from the calibration observation performed in Listing 4.1 are load into the array and the pressure field is measured over the specified plane. The variable *dimension* specifies how large the plane is on each side in mm. The variable *resolution* specifies the distance between each observation in mm. The variable *origin* specifies the least coordinate in the plane to be measured. The *directory* and *file* need to be the same as specified in Listing 4.1.

Command	Description
pm_calc	Calculate the magnitude of a voltage waveform.
pos_scan_cube	Scan a cube of a specified dimension, resolution, and starting location and record the data to a specified directory.
measure	Capture a waveform from the oscilloscope and save the data to a specified file.
scan_line	Scan a line of pressure parallel to the X-axis in a specified direction, distance, and resolution.
measure_pressure_field	Measure the pressure over the specified dimension starting at the specified origin and moving in a positive direction. The waveforms from the pressure field measurements are stored in the specified directory with the specified root file name.
scan_plane	Scan a plane of pressure normal to the Z-axis in a specified direction, distance and resolution.
plot_one	Plot the results from a single plane pressure field scan.
transient_movie	Make a movie out of the oscilloscope traces over a plane. This movie shows the pressure field in the plane over time.

Table 4.5: Pressure field measurementss Matlab commands.

Listing 4.2: Example Matlab m-file to measure a pressure field.

```
clear all
% Define paths to matlab m-files
common_path = '../../Common/';
addpath(common_path)
path_defs(common_path)
%-----------------------------------------
% Load delays
position = [0 -10 -25]
directory = '(x,x,-25)';
file =    '(0,-10,-25)';

data_file_name = ...
sprintf('../Calibrate/%s/%s_ref_delays.mat',directory,file);

load(data_file_name);
%-----------------------------------------
% Focus at home position
% 100mm plane with 2mm resolution
%
dimension = 160;
resolution = 2;
origin = [-80 -80 -25];
%
measure_pressure_field(delays ...
          ,directory,file,dimension,resolution,origin,.5);
```

Chapter 5

Results

5.1 Pressure Linearity

Since the pressure from each transducer adds linearly, the pressure measured from exciting each transducer individually should add up to the pressure measured from exciting all the transducers. This assumption is confirmed and shown in Figure 5.1.

5.2 Focussing

In order to determine where the geometric focus of the transducer array is, a cube of the pressure field was scanned without any delays applied to the transducers. The resulting pressure field is shown in Figure 5.2. The geometric focus was estimated to be at -25 mm below the hydrophone home position.

The array was calibrated to focus at points evenly spaced in the geometric focal plane between (0,0,-25) mm and (0,60,-25) mm. The resulting pressure fields are shown in Figures 5.3 - 5.9. There are large grating lobes about 60 mm from the focus. As the focus moves away from the geometric focus and a grating lobe moves

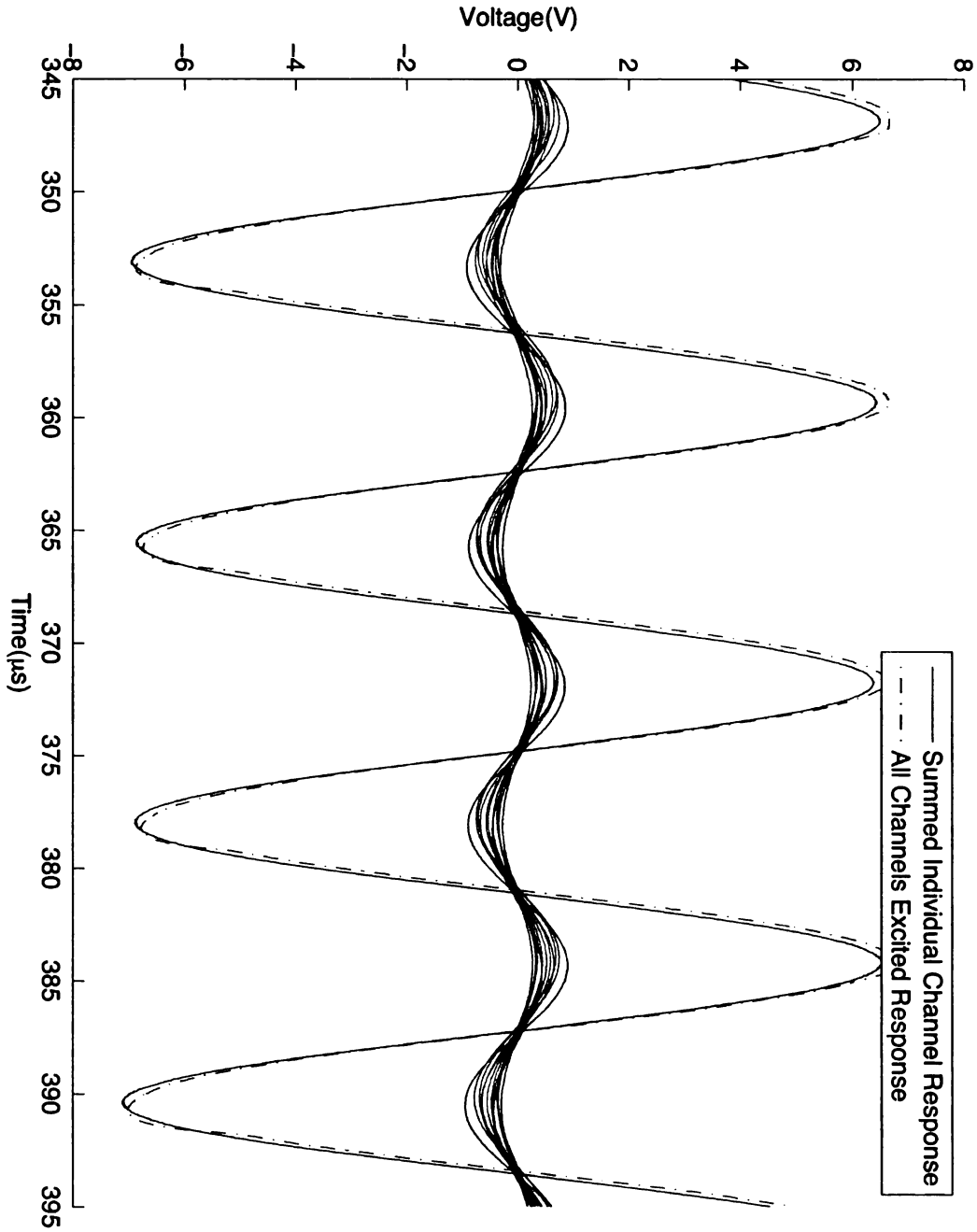


Figure 5.1: Summed response of individually excited transducers and the response of all transducers excited.

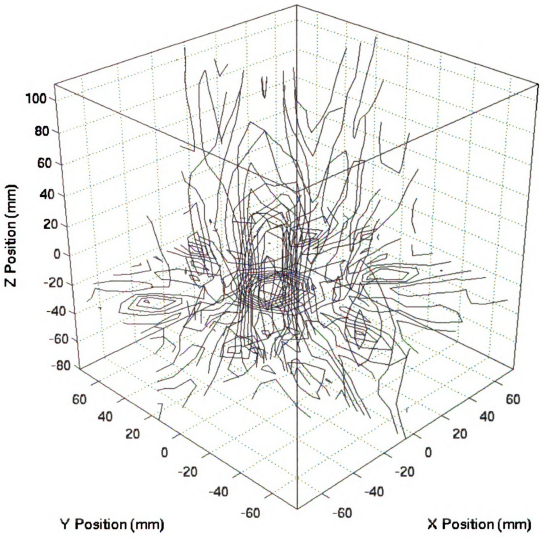


Figure 5.2: 3 dimensional scan of the pressure field with no delays applied to the transducers.

toward the geometric focus, the pressure at the intended focal point decreases and the pressure at the grating lobe that is moving closer to the geometric focus increases. The focus can be moved approximately 30 mm from the geometric focus before the grating lobe has a higher pressure than the intended focus.

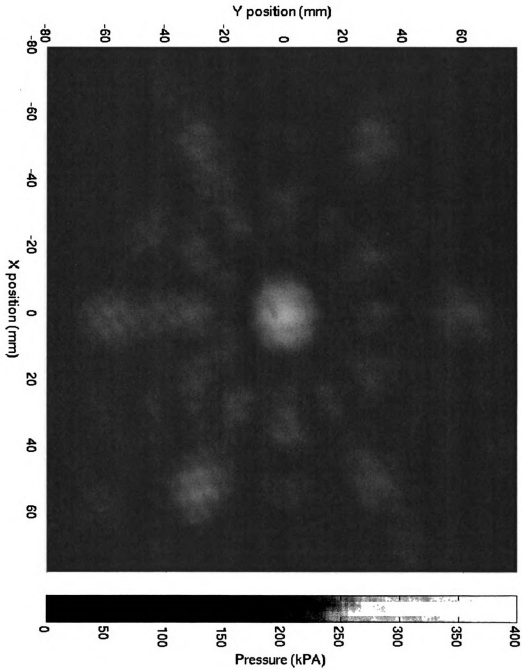


Figure 5.3: Array focused at (0,0,-25).

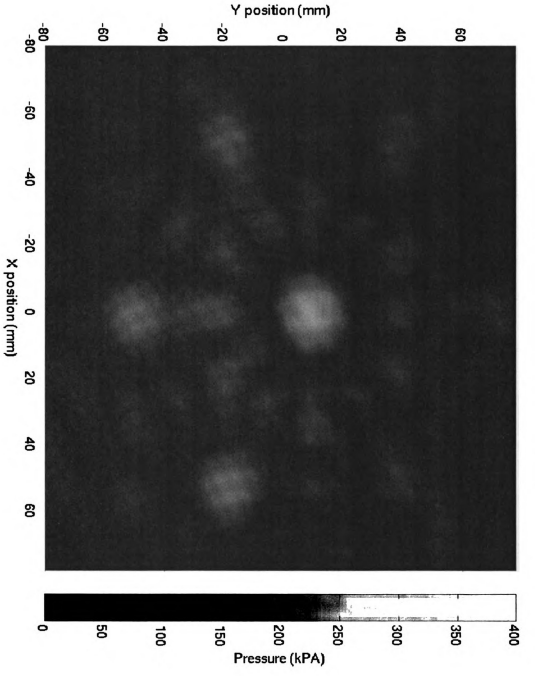


Figure 5.4: Array focused at (0,10,-25).

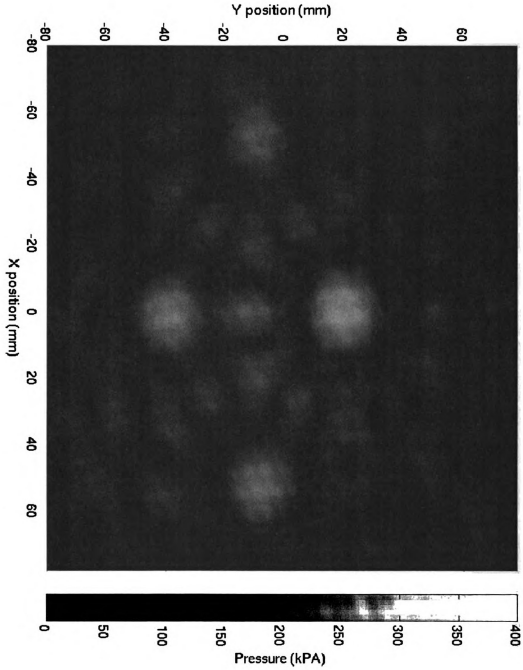


Figure 5.5: Array focused at (0,20,-25).

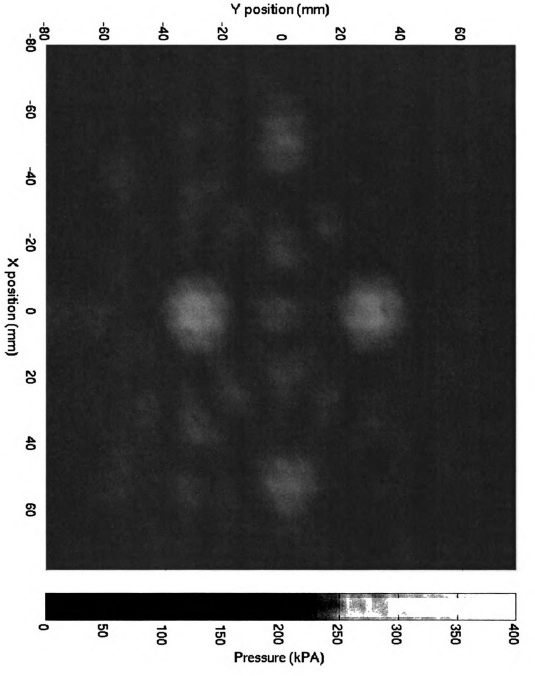


Figure 5.6: Array focused at (0,30,-25).

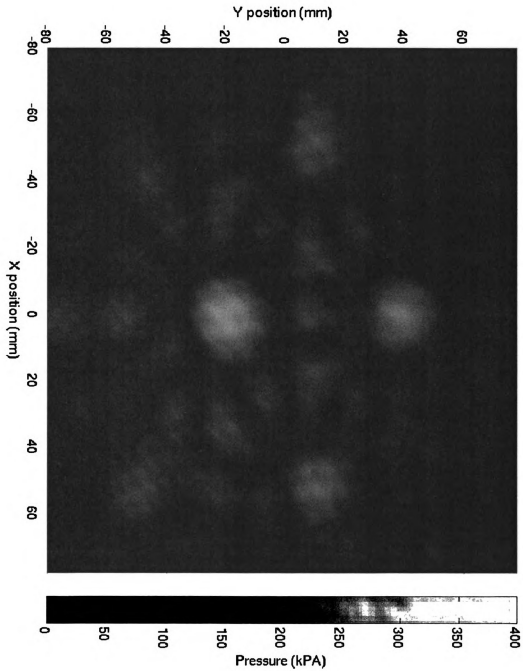


Figure 5.7: Array focused at (0,40,-25).

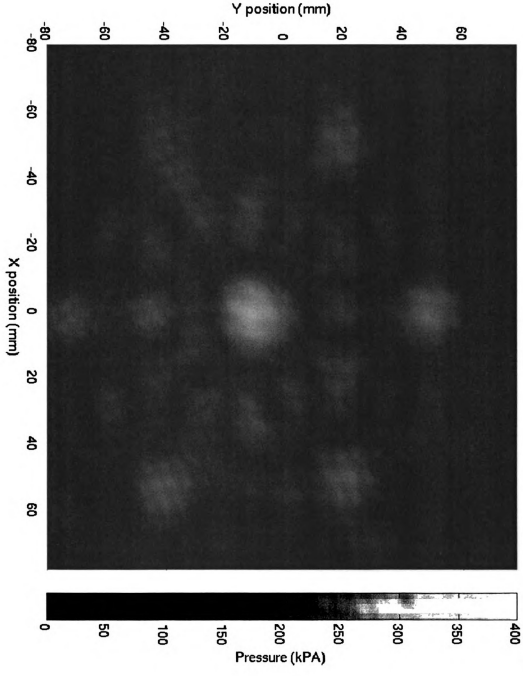


Figure 5.8: Array focused at (0,50,-25).

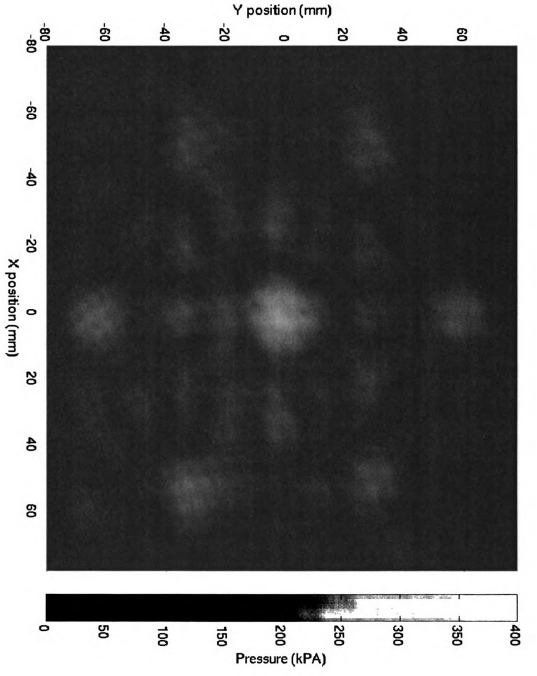


Figure 5.9: Array focused at (0,60,-25).

Chapter 6

Discussion

The system can be improved in several ways with future implementations. More channels can be implemented by addressing FPGA board I/O limitations using serial DACs instead of parallel DACs and by increasing the memory capacity by using FPGAs with more memory or by using memory external to the FPGAs. Another improvement is to implement multiple delay patterns. This can be accomplished by changing the existing firmware or by increasing the computer serial interface throughput in order to decrease the time needed to program the channel waveform data and allow for delay patterns to be loaded while the array is running. Lastly, the power amplifier can be modified slightly to drastically increase the output power.

6.1 Serial DAC

Initially, the TLC7254 parallel DAC was chosen because of its speed, cost, and availability. Since the initial design, there are serial DACS that have become available that are comparable in speed and cost such as the AD5426 or AD5450. These serial DACs require 2 signals per channel as apposed to 9 signals per channel for the TLC7254 [13] [7] [6]. This allows the number of DAC channels to increase from 16 to 46 with

the current FPGA circuit board I/O resources as shown in Figure 6.1. This would also simplify the DAC circuit board by reducing the number of traces. The reduced number of traces may may also allow for 8 channels to be implemented in a single circuit board that fits in the 6.25 inch tall slots that are presently.

Function	Signal	Serial DAC Per Channel	Per Chip	
RS-232/USB	Tx		1	
	Rx		1	
	CTS		1	
	RTS		1	
DAC	Din	1	46	
	SYNC_B	1	46	
	SClock		1	
TOTAL			97	97 available

Figure 6.1: Serial DAC I/O resources.

6.2 Additional Memory

More memory is needed in order to implement more channels. Increased memory capacity can come from larger capacity FPGAs such as the Xilinx XC6SLX100, which has 268 blocks of 16Kb block RAM, or the Xilinx XC3S5000, which has 104 blocks of 16Kb block RAM, or memory external to the FPGA [11] [12]. Each 16Kb block of memory can store the waveform for one transducer; therefore, the XC3S5000 can implement 104 channels and the XC6SLX100 can implement 268 channels. In order to use these higher capacity FPGAs, development boards for these FPGAs need to be found and purchased or custom boards need to be constructed. Since the FPGA slice resources are presently underutilized, using lower capacity FPGAs with external

memory is another option. External memory would take up I/O resources on the FPGA and reduce the number of DAC channels that can be implemented. In order to utilize external memory, development boards that support external memory need to be found and purchased or custom circuit boards need to be fabricated.

6.3 Multiple Delay Patterns

6.3.1 Firmware Implementation

Future applications of the ultrasound phased array motivate the implementation multiple delay patterns. Only one delay pattern is possible with the current firmware, but multiple delays are possible with slight modifications in the firmware. The waveforms would still be stored in memory, but different delays would be stored for each pattern. These different delays could be implemented with cascaded counters. There are 3340 unused slices in the FPGA with the AWG design. The counters needed to implement the delays for one pattern for each channel require 42 slices; therefore, approximately 79 total delays/channels can be implemented or 6 delay patterns patterns for the 13 transducer array can be implemented with the current FPGA. This estimate is conservative because there are some elements of the delay counters that can be shared instead of replicated for each channel.

6.3.2 Live Updating Implementation

Updating the delay information from the computer while running the array would allow for unlimited delay patterns. With the existing firmware, it takes one minute to update 16 2K-word channels of memory. This corresponds to about 546 words per second. The current serial communication baud clock rate is .5 MBaud but the FTDI UM232R can go up to 3 MBaud [14]. This would increase the throughput by a factor

of 6 to 3.2K words per sec. There are two words needed for to specify each delay, one delay per channel per pattern, and 13 channels in the array means there are 26 words need to be sent from the computer to the AWG per pattern. Therefore programming a pattern would take approximately 7.9ms.

6.4 Power

More output power can be achieved with the existing amplifier design with few changes. The LME49811 is a high bandwidth, high voltage op-amp [19]. It was not used initially because it was twice the cost of the OPA552 and there is not a spice model available, so the design could not be simulated prior to building the circuit and this could potentially increase the design time and cost. The LME49811 can handle a supply voltage up to 200V. The LME49811 bandwidth is approximately 100MHz with a gain of approximately 58dB at 80kHz[19]. This is greater than the gain of 34dB needed in the existing amplifier design. The LME49811 can source up to 9mA of output current [19]. Assuming a load matched to 50Ω with an output voltage of $400V_{pk-pk}$ max, a miximum output current of approximately 8A is needed. The current gain of the output transistors would need to be 8A/9mA = 889. The existing output transistors can supply up to 15A continuous current, can withstand 250V across the collector-emitter junction, and have a DC current gain of approximately 1300 at a collector current of 8A at 25C [18]. Higher voltage power transistors will be needed or the maximum output voltage will need to be limited to 250_{pk-pk}. 400W will be delivered to a 50Ω load when driven with a $400V_{pk-pk}$ sinusoid. 156W will be delivered to a 50Ω load when driven with a $250V_{pk-pk}$ sinusoid.

Chapter 7

Conclusion

An electronics system that allows for the independent control of all transducers in a 13 element low frequency (80kHz) phased array has been constructed and tested. The system comprises an ultrasound transducer phased array in a water tank, a hydrophone suspended in the water tank from a 3-D positioning system, a computer to control the ultrasound array, 3-D positioning system, and record the pressure from the hydrophone, and a set of custom and commercial electronics to drive the phased array.

The water tank is 300x300x500 mm, and the transducer array is positioned at the center bottom of the tank. The 3-D positioning systems is approximately centered in the water tank. It can travel 200x200x300 mm and has a resolution of 200nm. The transducer array is made up of thirteen 80kHz ultrasonic transducers that are 40mm in diameter and are placed in a radial pattern with 65mm spacing. This corresponds to a 3.5λ center-to-center spacing at 80kHz in water.

The computer controls the hydrophone position and the electronics that drive the transducer array. The control is implemented with m-files in Matlab. There are m-files that: perform a pressure field measurement, calibrate and test the transducer array delays, control the 3-D positioning system, read and record the voltage data

from the hydrophone, and program waveforms that drive the transducer array.

The electronics that drive the transducer array are divided into the following: a commercial FPGA development board that stores digital waveforms and controls an array of DACs, an array of DACs on a set of custom circuit boards that convert the digital waveforms into analog waveforms, an array of power amplifiers on a set of custom circuit boards that amplify the analog waveforms in order to drive the transducers, and a set of matching networks. The commercial FPGA board is a MEMEC Spartan-3 LC Development Kit that contains a Spartan 3 XC400 FPGA. The digital waveforms are stored in the RAM on the FPGA with the waveform for each channel stored in a 16Kb data block. The waveforms are 2048 8-bit words that update the DAC voltages every 320ns. The DACs are implemented using a Texas Intruments TLC7524 and an op-amp. The initial output of the DAC circuit was noisy, so a 5th order Butterworth filter is used to reduce the noise. The output of the DACs, which have a resolution of 4mV for an 8-bit input, are set to $1V_{pk-pk}$. The output of the DAC circuits are amplified with an array of power amplifiers. The power amplifiers are Class AB implemented using Burr-Brown OPA552 op-amps and On-Semiconductor MJH11021/2 Complementary Darlington Silicon Power Transistors. The power amplifier can drive 25W into a 50Ω load. Each transducer has a matching network that transforms its impedance to 50Ω.

The geometric focus of the transducer array was observed and the focus was moved in the X-Y plane by changing the phase of each transducer. The focus can be moved up to 30 mm from the geometric focus before grating lobes, due to the geometry of the transducer array, significantly distort the resulting pressure field.

BIBLIOGRAPHY

[1] K Agbossou, J-L Dion, S Carignan, M Abdelkrim, and A Cheriti. Class d amplifier for a power piezoelectric load. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 47:1036–1041, 2000.

[2] E Cathignol, J Tavakkoli, A Birer, and A Arefiev. Comparison between the effects of cavitation induced by two different pressure-time shock waveform pulses. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 45:788–799, 1998.

[3] M Cooper, Z Xu, E Rothman, AM Levin, AP Advincula, JB Fowlkes, and CA Cain. Controlled ultrasound tissue erosion: the effects of tissue type, exposure parameters and the role of dynamic microbubble activity. volume 3, pages 1808–1811 Vol.3, 2004.

[4] Memec Design. Memec spartan-3 lc users guide.

[5] Memec Design. P160 proto user guide.

[6] Analog Devices. 8-/10-/12-/14-bit high bandwidth multiplying dacs with serial interface AD5450/AD5451/AD5452/AD5453. 2006.

[7] Analog Devices. 8-/10-/12-bit high bandwidth multiplying dacs with serial interface AD5426/AD5432/AD5443. 2009.

[8] T Hall and C Cain. A low cost compact 512 channel therapeutic ultrasound system for transcutaneous ultrasound surgery. pages 1–5, Oct 2005.

[9] T F Hueter and R H Bolt. *Sonics Techniques For The Use Of Sound And Ultrasound In Engineering And Science.* 1955.

[10] Xilinx Inc. ISE design suite.

[11] Xilinx Inc. Spartan-3 generation FPGA user guide. 2007.

[12] Xilinx Inc. Spartan-6 generation FPGA user guide. 2009.

[13] Texas Instruments. TLC7524C, TLC7524E, TLC7524I 8-bit multiplying digital-to-analog converters. (SLAS061D), 2007.

[14] Future Technology Devices International Ltd. UM232R dip module. 2009.

[15] Future Technology Devices International Ltd. Virtual com port drivers. 2009.

[16] W P Mason and H Flynn. Physics of acoustic cavitation in liquids. *Physical Acoustics*, I(B):58–172, 1964.

[17] R McGough, D Cindric, and T Samulski. Shape calibration of a conformal ultrasound therapy array. *IEEE Transactions on Ultrasonics*, Dec 2001.

[18] On Semi. Complementary darlington silicon power transistors.

[19] National Semiconductor. LME49811 audio power amplifier series high fidelity 200 volt power amplifier input stage with shutdown. 2008.

[20] BC Tran, J Seo, TL Hall, JB Fowlkes, and CA Cain. Microbubble-enhanced cavitation for noninvasive ultrasound surgery. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 50:1296–1304, 2003.

[21] G M Wierzba. ECE 402: Applications of analog integrated circuits. 2007.

[22] G M Wierzba. ECE 404: Radio frequency electronic circuits. 2007.

[23] Z Xu, A Ludomirsky, LY Eun, TL Hall, BC Tran, JB Fowlkes, and CA Cain. Controlled ultrasound tissue erosion. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 51:726–736, 2004.