

THE VERY LARGE SCALE INTEGRATED CIRCUIT
TESTER II

By

ROBERT IMARK

Bachelor of Science

Oklahoma State University

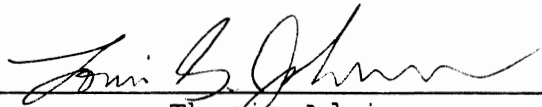
Stillwater, Oklahoma

1991

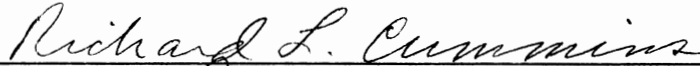
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1992

THE VERY LARGE SCALE INTEGRATED CIRCUIT
TESTER II

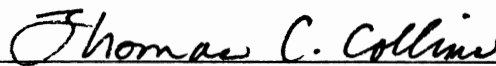
Thesis Approved:



Thesis Advisor







Dean of the Graduate College

PREFACE

Designing Very Large Integrated Circuit Tester II, making a prototype, and documenting it are a challenging yet rewarding experience. It is the milestone of my engineering education at Oklahoma State University. This project has involved many hours of hard work, and in the meantime I received help from several individuals and companies. Here I want to express my gratitude to Dr. Louis Johnson, my thesis advisor, for his guidance and assistance from the initiation of the idea to the completion of the design. I also want to thank Dr. Richard Cummins and Dr. R. G. Ramakumar for serving on my committee, and Dr. Chriswell Hutchens and Dr. Jerzy Krasinski for their technical advice and assistance. A special thanks goes to two companies, Motorola and Signetic, for providing free components for my prototype, and to OSU Electrical and Computer Engineering Department for covering most of the expenses of the prototype. I also want to thank Gerald Stotts for the materials and valuable information. Finally, I want to express my gratitude to my parents who have financed my education, and my wife who has encouraged and believed in me.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.....	1
II. HARDWARE DESIGN.....	3
Hardware Components.....	3
Working Example.....	13
III. THE COMPARATOR.....	15
Design.....	15
Design Breakdown.....	18
Comparator Example.....	19
IV. THE COMPUTER/HARDWARE INTERFACE.....	22
Design.....	22
Connectors.....	27
V. SOFTWARE DESIGN.....	32
Design.....	32
Software Execution.....	33
Test Data File.....	35
Example of Test Data File.....	41
Error Data File.....	45
VI. CONCLUSION.....	47
REFERENCES.....	49
APPENDIXES.....	50
APPENDIX A - THE CONTROLLER LAYOUT.....	50
APPENDIX B - THE HARDWARE LAYOUT.....	52
APPENDIX C - CODE FOR VLSI2.PAS.....	54
APPENDIX D - CODE FOR INOUT.ASM.....	69

Chapter	Page
APPENDIX E - THE USERS MANUAL.....	71
APPENDIX F - MANUFACTURERS LIST.....	74

Chapter	Page
APPENDIX E - THE USERS MANUAL.....	71
APPENDIX F - MANUFACTURERS LIST.....	74

LIST OF TABLES

Table	Page
I. The Control Function.....	6
II. State Table.....	9
III. Function Table.....	9
IV. The Addresses for I/O.....	25
V. The IOread Values.....	27
VI. Pin Functions.....	28
VII. The Cable.....	30
VIII. The VLSI tester II Addresses.....	31
IX. Chip Pin Numbers.....	37
X. Pin Assignment.....	41

LIST OF FIGURES

Figure	Page
1. Block Diagram of the Circuit.....	4
2. The Wave Form of the Timer.....	10
3. Timer Schematic.....	11
4. Timer Layout.....	12
5. Comparator Section.....	17
6. Comparator Example.....	21
7. Driver Schematic.....	23
8. Driver Layout.....	24
9. The Test Data File Format.....	39
10. The ZIF Pin Numbers.....	40
11. An Example of a Test Data File.....	43
12. Test Example.....	44
13. Error Data File.....	45
14. Hardware Diagram.....	POCKET

CHAPTER I

INTRODUCTION

The Very Large Scale Integrated Circuit Tester II is a tester that can test very large integrated circuits which are fabricated on a single chip. The Very Large Scale Integrated Circuit Tester II is the improvement of the Very Large Scale Integrated Circuit Tester I, my senior design project. The initial idea was originated by my thesis adviser Dr. Louis Johnson. Graduate students in the Electrical Engineering department here at Oklahoma State University design their own integrated circuits, and when these chips are returned after being fabricated, they must be tested to verify their operation. In the past, this had to be done manually one test at a time, which was time consuming. Dr. Johnson proposed the idea to build a device capable of rapid testing of such chips. The Very Large Scale Integrated Circuit Tester I was built to meet this need.

The VLSI tester I, however, was not satisfactory for testing dynamic latches, the electrical storage devices which keep the data only for short period of time. Consequently the VLSI tester II has been developed.

The working prototype of the VLSI tester II tests the logic and response time of chips with up to 64 pins. It is designed to test the response time of the chip greater than or equal to 10ns. The VLSI tester II has 512k bytes of memory for the storage of test data and it is capable of giving a new test data at the rate of 10Mhz. In addition, the device has the capacity to interface with both Texas Instruments and IBM personal computers. The software allows the user to enter the test data file and desired response time of the chip. The design contains its own power supply and computer interface board, thus making the device self sufficient. There are three areas in which the performance of the VLSI tester II surpasses that of the VLSI tester I. First of all, the new design has greater output of test data per unit time. Secondly, it can test smaller response time. Lastly the software is more user friendly.

CHAPTER II

HARDWARE DESIGN

Hardware Components

The VLSI tester II is composed of many hardware sections. It has eight main parts: the interface, the direction memories, the data memories, the write drivers, the control section, the timer, the comparator, and the zero insertion force sockets. Each part is discussed below to provide a clear understanding of how the hardware works. Figure 2.1 shows the block diagram of the circuit.

The first one is the interface. The interface is the signal translator between the computer and the VLSI tester II. The computer communicates through the driver, a device that amplifies the signal, with the interface. The computer can communicate by 11 I/O addresses. The first eight addresses are for the direction, data memory, and output driver. The last three are for the controller. The communication between the computer and the VLSI tester II is parallel. All the data signals are buffered by the 74LS245 chip driver.

The second part of the hardware is the direction memories and latches. The VLSI tester II uses direction

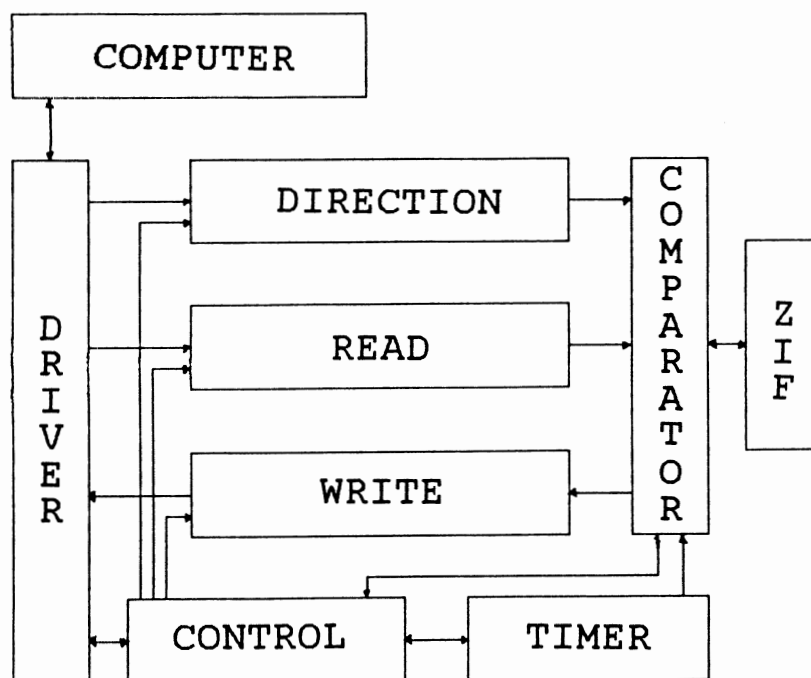


Figure 2.1 Block Diagram of the Circuit

memory to set the pins on the ZIF (Zero Insertion Force) socket as inputs or outputs. This is accomplished by using eight 62256 chips. Each one of these chips holds 32kbytes of data that correspond to eight pins on the ZIF socket. These memories are loaded through the 74LS244 chips by the computer, and are controlled by the control section of the design. The direction latches are for synchronization between the direction and the data signal of the timer.

The third is data memories and latches. The data memory is used to send test data to input pins on the ZIF socket, and hold the desired outputs that should result from the output pins of the ZIF socket. Similar to the direction memories, this section consists of eight 62256 chips. Each one of these chips holds 32Kbytes of data that correspond to eight pins on the ZIF socket. These memories are loaded through the 74LS244 chips by the computer, and are controlled by the control section of the design. The data latches are for synchronization between the data, and the data signal of the timer.

The forth one is the write drivers. If an error occurs during a test, the control section will load the write drivers with the locations of all errors. This driver section consists of eight 74LS245 chips. These drivers hold eight bits of data each which correspond to eight pins on the ZIF socket. This data will hold logic 1 if an error occurs, and logic 0 if the corresponding pin works

correctly.

The fifth part is the control section regulating the access among the read memories, the direction memories, the write driver, the comparator, the timer, the period, the address counter, and the error counter. The control of these sections is accomplished by the 11th I/O address of the VLSI tester II and with the proper codes as shown on table 2.1.

TABLE 2.1
THE CONTROL FUNCTION

Code	Direction	Action
FBH	OUT	For loading the address counter.
FDH	OUT	For loading the error counter.
FEH	OUT	For loading the timer
FEH	OUT	For loading the period
FDH	OUT	For loading the test data
FEH	OUT	For loading the direction data
07H	OUT	Start the test
FFH	OUT	Stop the test
FEH	OUT	For reading the output data
	IN	Read the status of VLSI2 tester

The VLSI tester II stops the test when the stop signal is high and the error counter is zero or the address counter is zero. If the value of the error counter is zero, the error indication is logic high, the address counter is not zero and the program changes the stop signal to high, the timer will be triggered. In addition, the address counter will be decreased by one, and the timer will send new data and sample signals. Consequently, the address counter must be increased by one and afterward the program can change the stop signal to high.

The other part of the controller is the period, which makes the delay between two data tests. It also coordinates the action of the timer, and the decrement of the error counter. To set the period, load the controller to the proper function and send the data to the 10th I/O address. The data of value 0 is the longest period and EFH is the shortest period.

The error counter counts number of errors during the test. To set the error counter, load the controller to the proper function and send the data to the 9th I/O address for the low byte, and 10th I/O address for the high byte, and must be in this order. Every time an error occurs, the error counter is decreased by one.

The stop counter gives the current address of the test data memories. To set the stop counter, load the controller to the proper function and send the data to the 9th I/O

address for the low byte, and 10th I/O address for the high byte, and this order must be followed. Every time the load signal from the timer goes from low to high, the stop counter is decreased by one.

The sixth part is the timer which has the function of making a programmable delay between two different signals. The user sets a programmable time delay through the computer. To set the timer, load the controller to the proper function and send the data to the 9TH I/O address. The data 0H correspond to the longest delay, and FEH for the shortest delay. The timer is accomplished by a 20MHz clock, a voltage controlled oscillator, a phase lock loop, counters and translators. The timer is triggered by the trigger signal. A synchronous sequential logic circuit receives the this signal, and transmits the incremental signal which is exactly one clock period long to the universal hexadecimal counters. The synchronous sequential logic circuit state table can be seen in the following table 2.2.

These counters change the carry out signal to high, and set the control to the level of the operational mode of increment. Furthermore, the timer starts to count at the same time the data signal goes high. The sample signal also goes high but only when the counter reaches zero. At this time the counter sets the control to preset and stops the counter. The function table of the universal hexadecimal counter is shown on the table 2.3.

TABLE 2.2
STATE TABLE

Present State		Next state				Output
		X=0		X=1		
DA	DB	DA	DB	DA	DB	Y
0	0	0	0	1	1	0
1	1	0	1	0	1	1
0	1	0	0	0	1	0

TABLE 2.3
FUNCTION TABLE

Cin	S1	S2	Operating Mode
X	L	L	Preset (Program)
L	L	H	Increment (Count Up)
H	L	H	Hold Count
L	H	L	Decrement (Count Down)
H	H	L	Hold Count
X	H	H	Hold (Stop Count)

The sample signal and data signal go low only when the trigger signal goes low. The voltage controlled oscillator generates a 100MHz clock for the counter. This frequency is divided by 10, and it is fed to the phase lock loop. The phase lock loop also compares this frequency with a 10MHz clock frequency from a crystal that is divided by two. The output of the phase lock loop is filtered, and fed to the voltage controlled oscillator. The 100MHz clock is used for the clock of the counter, delay flip-flop, and the frequency divider. All the circuits are Emitter Coupled Logic technology except for the 20MHz crystal driver. The wave forms coming to the timer and going from the timer are shown in the figure 2.2.

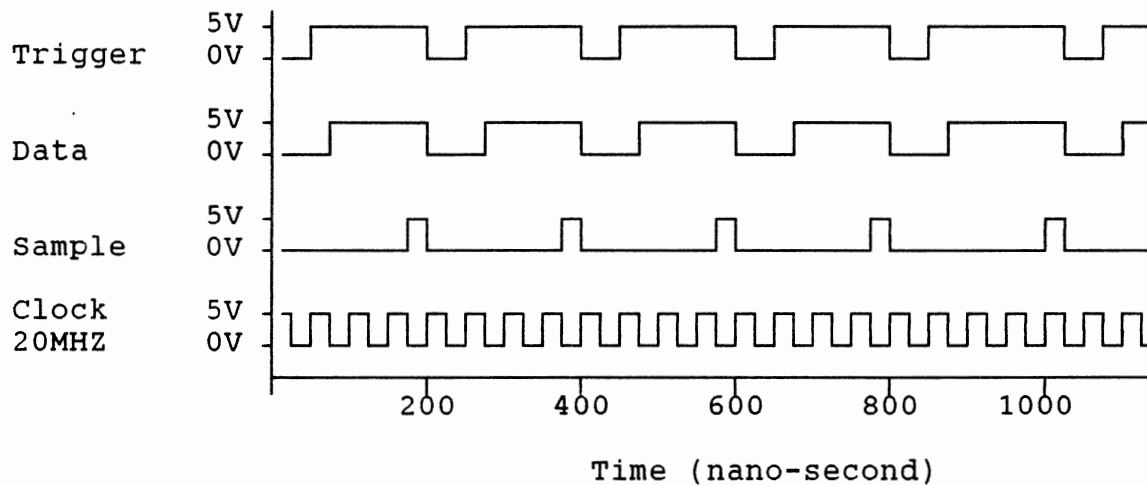


Figure 2.2 The wave form of the timer

The timer schematic is shown in the figure 2.3. The timer board layout is also shown in the figure 2.4.

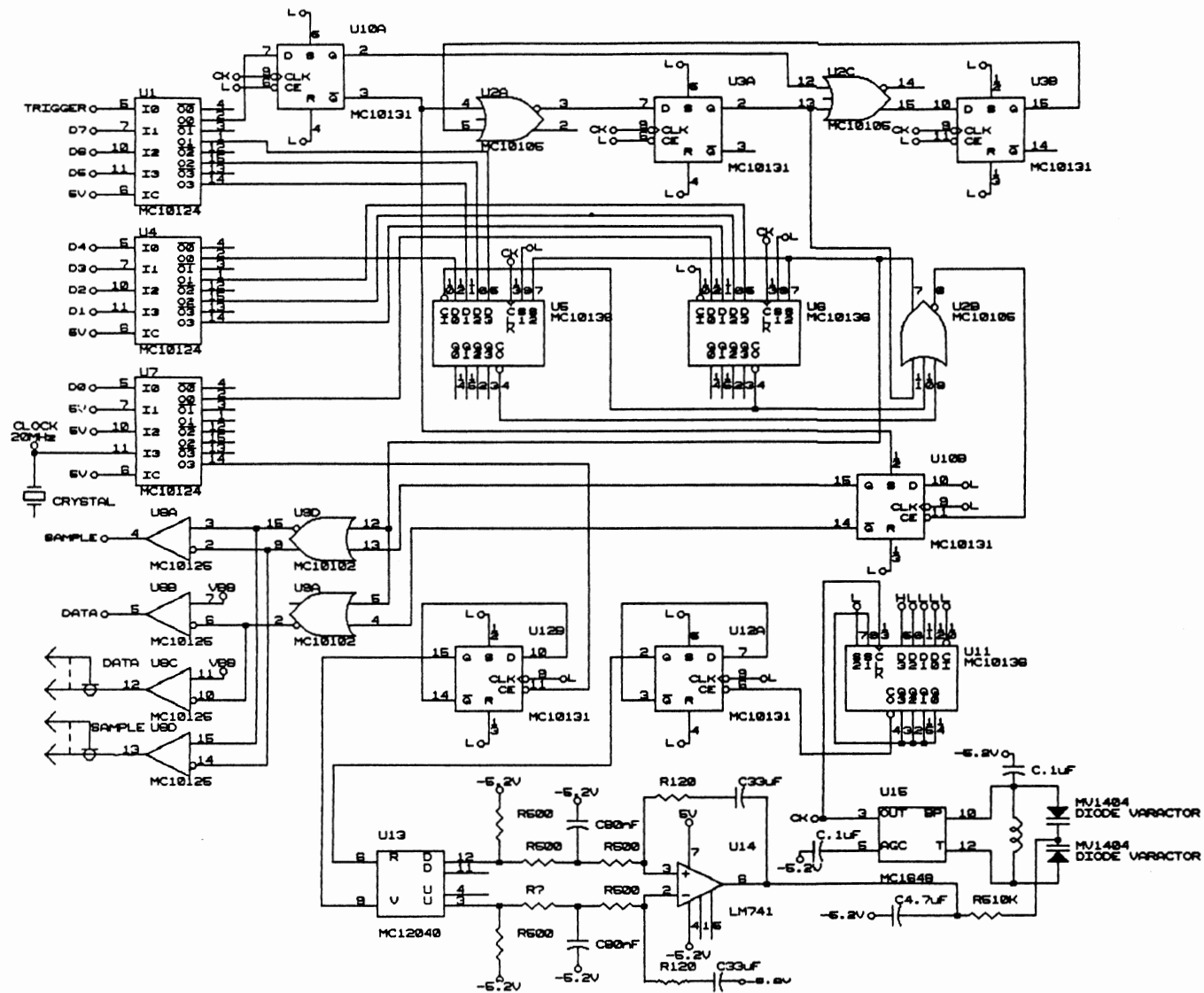


Figure 2.2 Timer Schematic

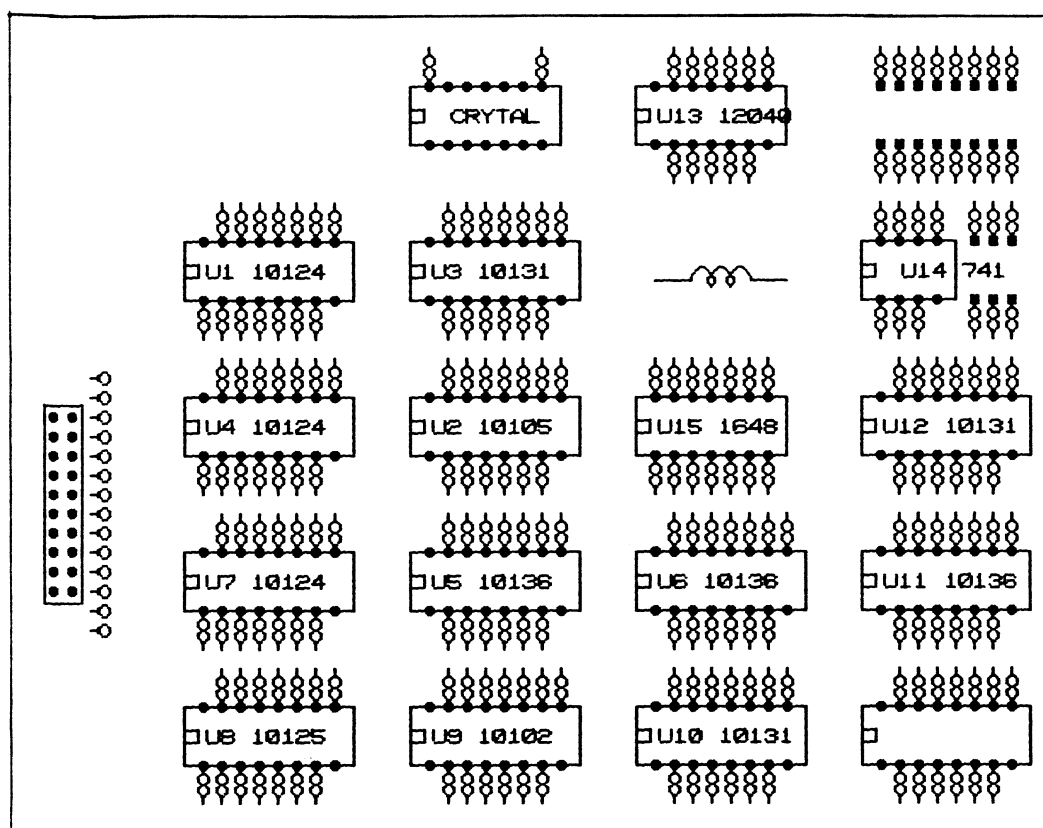


Figure 2.3 Timer Layout.

The seventh part is the comparator, an important portion of the hardware. Here is a brief explanation of its operation. More explanations about the comparator are given in chapter III. The comparator samples the chip output when the timer has counted down the necessary delay. Then the comparator compares the chip output with the test data, and sends an error indication signal. At the same time, the comparator sends the error indication of each individual pin to the write driver. It is accomplished by using 64 delay flip-flop (eight 74F374 chips), 64 three stage output drivers (sixteen 74F125 chips), 64 exclusive-or gates (sixteen 74LS86), and 9 eight input OR gates (ten 74HC4078 chips). The delay flip-flops are used for sampling the output of the chip on the ZIF socket. The 3-stage output drivers are used for setting the pins of the ZIF socket as input or output. The exclusive-or gates and the eight input OR gates are for the error indication signal. These error indication signals are transmitted to the write drivers.

The last part is the ZIF socket or Zero Insertion Force socket, a physical place where the chip is placed. The VLSI tester II has two ZIF sockets. The first is for chips up to 48 pins, and the second is for chips with up to 64 pins.

Working Example

During the first stage of the test, the computer communicates through the driver, loads the direction, and

the data memories. In addition, the user sets a programmable time delay through the computer. This delay is the maximum response time of the chip under test. When the test data are loaded in the memories, the computer sets the stop address, the error address, the period, and the timer. After this, it sends a start signal to the controller. At this time, the test data are transmitted to the latches. Furthermore, the controller checks whether the error indication signal is high, and whether the error counter is at the value zero. If so, the test stops. The test will also stop if the stop counter is at the value zero. If the test does not stop, then the error counter will be decreased by one, and the period sends a trigger signal to the timer. A few nano-seconds later, the data signal from the timer goes from low to high, and the data and direction latches are activated. The test data, therefore, are transmitted to the Zero Insertion Force socket. At the same time, the stop counter is decreased by one. When the sample signal from the timer goes from low to high, the outputs of the ZIF sockets are sampled by the sample latches. These outputs are compared by the comparator, and if any of the signals from the comparator is high, the error indication signal will be high. This cycle is repeated until the test is finished.

CHAPTER III

THE COMPARATOR

Design

The main function of the comparator section of the VLSI tester II device is to test the logic of the chips, integrated circuits. In other words, it tests whether the chip functions the way the designer intended. The comparator also plays an important role in testing the response time of the chip. The response time in this case is defined as the time required for the chip output to change from one logic state to the other after a chip input changes.

The design of the comparator section is broken down into parts. One part of the design includes two ZIF sockets. One of these sockets is capable of accepting the larger digital chips of up to 64 pins in size. The other has 48 pins and accepts the smaller integrated circuits. The chip to be tested is placed in the correct ZIF socket, and jumpers are used to supply the necessary power and ground connections. The VLSI tester II has a total of ten power and ground pins available. In addition, it has two different clocks for triggering sequential integrated

circuits. One clock is a rising edge clock, and the other is a falling edge clock.

Another portion of the design is 64 D flip-flops. When the test data are received by the chip through the test data latches, the response time is measured by means of the timer. This timer works in conjunction with the 64 D flip-flop gates. The outputs of the chip are latched in these flip-flops when the timer has counted down from the maximum response time. At the end of this time, the outputs are released, and the comparison is made with the desired outputs. This comparison is made using 64 exclusive-or gates. Testing of the 64 ZIF pin locations is broken down into eight bytes with one byte representing 8 pins of the ZIF socket. Testing one of these bytes is implemented as shown in figure 3.1. The test data from the directional latches are received by the 74F125 3-state buffers. A logic 0 at this portion will enable the buffer. This allows the test data from the read latch to be sent to the chip as an input. This data is also sent to both inputs of the XOR gate assuring that no errors can occur on the input pins of the chip under test. On the other hand, a logic 1 received by the buffer will disable it. This will designate the corresponding pin of the ZIF socket as an output. The test data from the read latch will then only be sent to one input of the XOR gate. This input will be the desired data that should come from the output of the pin in question. The

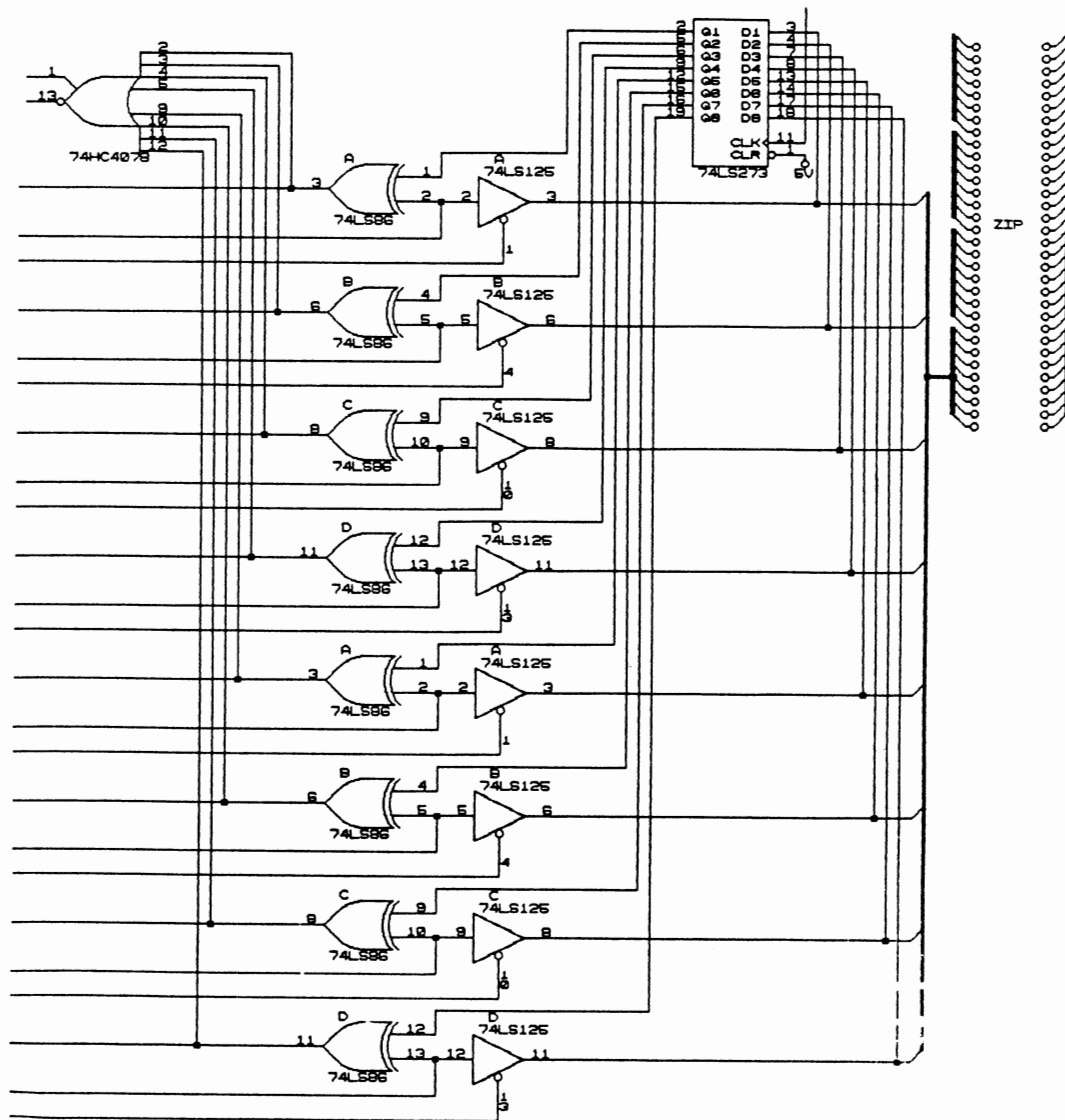


Figure 3.1 Comparator Section

actual output from the ZIF pin number will then be sent to the D flip flop. It will be held in this flip flop until the desired response time is reached. At this time the clock will trigger the D flip flop into releasing the actual output from the corresponding pin number. This output will then be sent to the other XOR input. Now the comparison is made. If an error has occurred, that is to say, the actual and desired outputs do not match, a logic 1 shall come from the XOR gate. Otherwise a logic 0 will come from the XOR gate. Each of these XOR outputs is sent to an eight input OR gate. This gives one byte of the error indicator signal. This procedure is duplicated eight times to accomplish the testing of each of the 64 pins on the ZIF socket. The end result is combined through another eight inputs or gates to get the entire error indicator signal. A logic 1 at this signal signifies an error occurred during a test. This error indicator signal is then sent to the error counter.

Design Breakdown

The design of the comparator section of the VLSI tester II device is composed of the following parts:

1. Two ZIF sockets. This is where the chips to be tested are inserted. A 48 pin socket is used for small IC chips, while a wider 64 pin socket is provided for larger chips.

2. 64 3-state buffers between the read latches and ZIF

sockets. One buffer is used for each of the 64 pin locations of the ZIF sockets. These buffers are enabled by data from the directional latches, either allowing the input test data to flow from the read latches to the chip under test, or blocking the desired output test data from the chip. Thus, these buffers determine whether a pin on the ZIF socket is an input or output.

3. 64 XOR gates. One gate is used for each pin location of the ZIF sockets. These gates compare all test data with the actual data from the chip. A logic 0 from an XOR gate signifies a correct response from the chip, while a logic 1 signifies an error has occurred.

4. 9 8-input OR gates. These are used to combine all the outputs from the XOR gates into one error signal. This signal is sent to the computer, and tells the software whether an error has occurred during a test.

5. 64 D flip-flop gates. These gates are triggered by the high speed clock to test the response time of the chip. These flip-flops will hold the outputs of the chip until the desired response time is reached by the clock. This assures that all comparisons are made simultaneously.

Comparator Example

An example will best display how the comparator functions. Refer to figure 3.2 where an inverter is placed in the ZIF socket for testing. For clarity this figure

refers to only one gate of the chip, thus only two pins. First, the directional data contains a logic 0 for ZIF pin number one. This enables the buffer, thus designating pin number one on the ZIF socket as an input. Similarly a logic 1 at pin number two disables its associated buffer, designating pin two as an output. Next the test data is loaded. The logic 0 at pin one is the input, and is sent to the chip under test as well as both the inputs of the XOR gate. The logic 1 at pin number two is the desired output, and will be received by only one input of the XOR gate. After the timer reaches the desired response time, the actual output from the inverter is released. This allows the signal from this inverter output to enter the other input of the XOR gate corresponding to that ZIF pin number. The XOR gate will respond with a logic 0 if the actual output and desired output match, while a logic 1 would indicate an error resulted.

TESTING AN INVERTER GATE

INVERTER
GATE IN
ZIF SOCKET

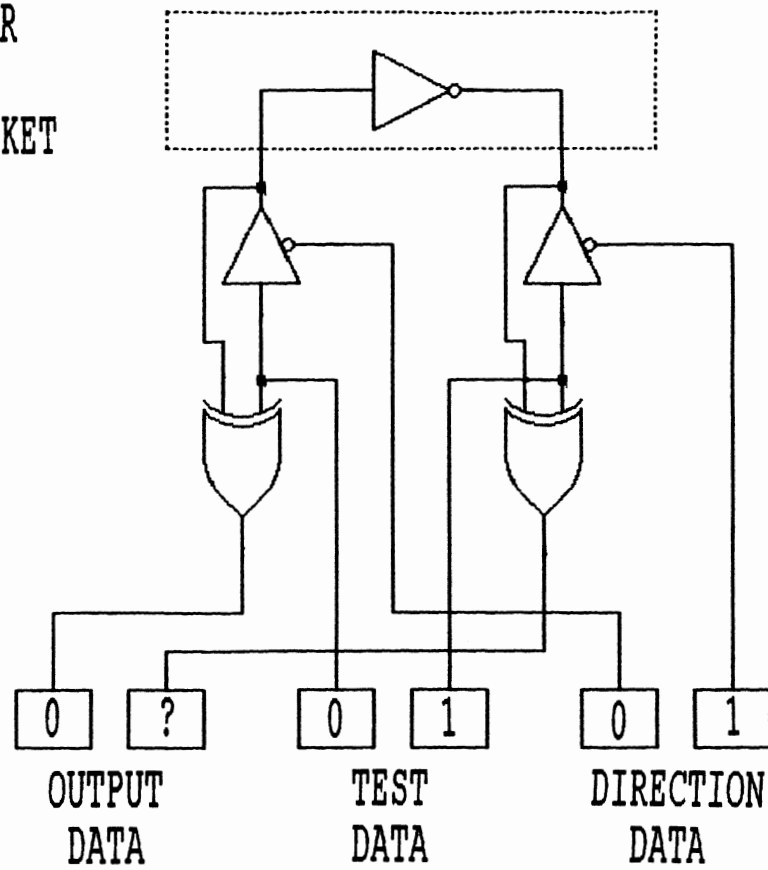


Figure 3.2 Comparator Example

CHAPTER IV

THE COMPUTER/HARDWARE INTERFACE

Design

The VLSI tester II is conceived as a digital chip tester which can test the functionality of chips. The software is used to transfer data to the hardware which is needed to run the tests. To accomplish this communication between the devices, an interface must be used. The interface for VLSI tester II is called driver, and it is placed inside the computer. The basic function of this driver is to buffer the signal coming and going from the computer, to select the proper I/O address, and to modify the control logic of the signals. The driver board is used to determine what addresses should be used to communicate with the hardware. A schematic of this is shown in figure 4.1, and the driver board layout is shown in figure 4.2.

To determine which addresses will allow I/O, dip switches are used. These switches are either left open or closed which send a value of logic 1 or 0 respectively to their corresponding address comparator. For example, to

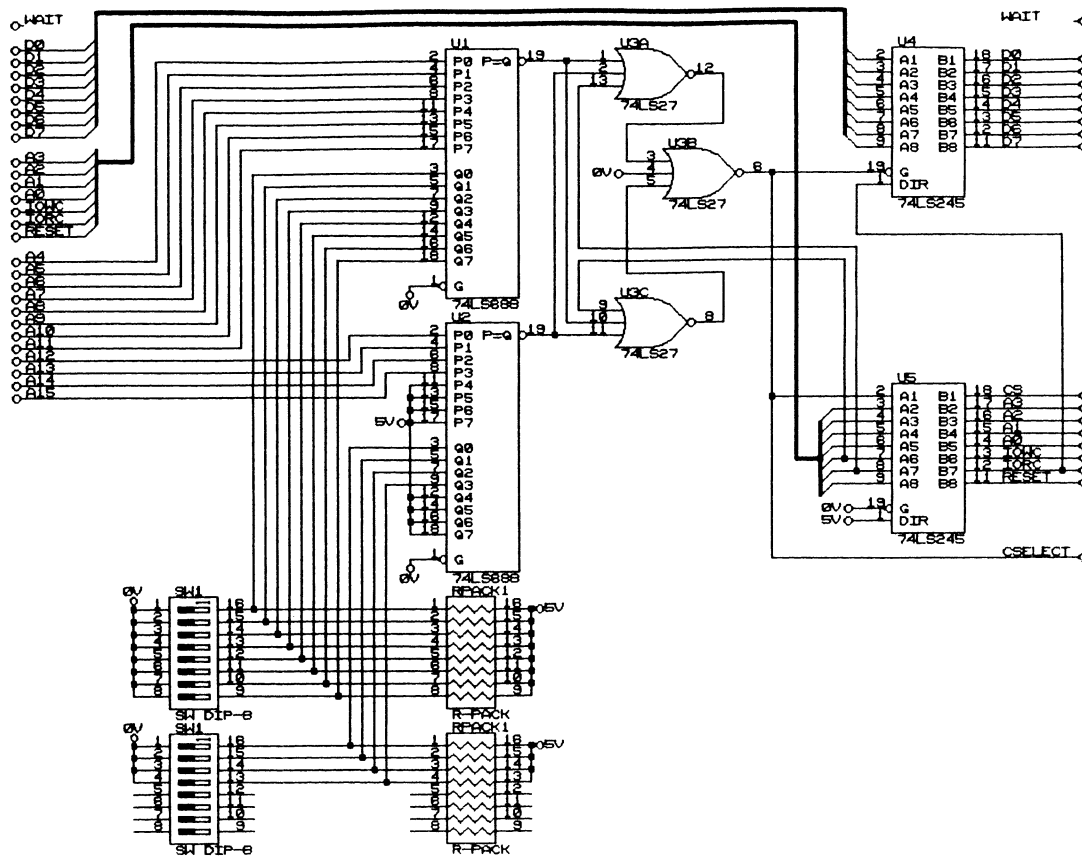


Figure 4.1 Driver Schematic

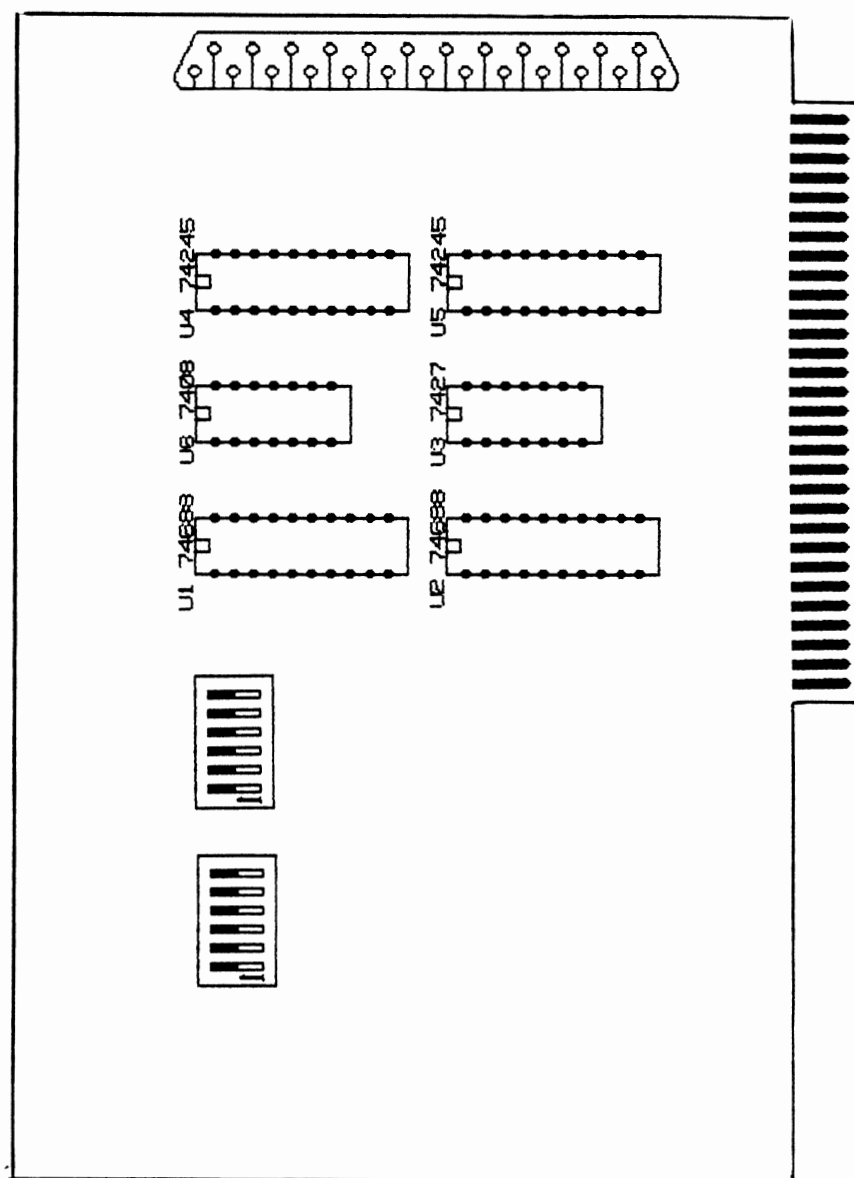


Figure 4.2 Driver Layout

address the board for a base address of 300H switches, SW 1-5 and SW 1-6 must be open, and all other switches of SW 1 and SW 2 must be closed. For the Texas Instrument computer, only 10 address pins are used for I/O, however; for IBM computers all 16 I/O address pins are used. Thus, the board needs to account for all I/O address pins. The address used for I/O and their corresponding dip switches are shown in table 4.1.

TABLE 4.1
THE ADDRESSES FOR I/O

Address Bit	Switch
A15	SW 2-6
A14	SW 2-5
A13	SW 2-4
A12	SW 2-3
A11	SW 2-2
A10	SW 2-1
A9	SW 1-6
A8	SW 1-5
A7	SW 1-4
A6	SW 1-3
A5	SW 1-2
A4	SW 1-1

TABLE 4.1 (Continued)

Address Bit	Switch
A3	N/A
A2	N/A
A1	N/A
A0	N/A

These switches are used to set the board to respond to a proper address. The base address currently set on the board is 300H. This address will work for both TI and IBM computers, so it does not need to be changed. However, other addresses may be set by adjusting the dip switches. We must be careful not to use an address taken by the computer. In addition, the program must change accordingly. Notice that since A0-A3 do not have switches, the addresses used are 300H through 30FH.

The functionality of address comparator (74LS688) is to indicate that the computer wants to communicate to the VLSI tester II. However, before this signal is sent to the VLSI tester II, it must check whether either the IOREAD or IOWRITE is low. Then the chip selects the signal which goes low. The reason for this complex logic is to avoid false communication between the computer and the VLSI tester II. Once the chip selected goes low, the driver allows the

communication with the hardware and the computer. The IOread signal from the computer is also used to determine whether the software is writing to the hardware or reading from the hardware. The logic values of the IOread signal that determine this are shown in table 4.2.

TABLE 4.2
THE IOREAD VALUES

IOread Value	Result
1	Software writes to hardware.
0	Software reads from hardware.

These values are sent to the 74LS245 chips, U4 and U5 on schematic. These chips are octal bus transceivers with 3-state outputs. Therefore, the data flow from A to B when writing to the hardware, or B to A when reading from the hardware.

Connectors

The connectors that are attached to the internal board have the pin functions shown in table 4.3.

TABLE 4.3
PIN FUNCTIONS

PIN	SIGNAL	PIN	SIGNAL
A01	NMI-	B01	GROUND
A02	DATA 7	B02	RESET
A03	DATA 6	B03	+ 5 V power
A04	DATA 5	B04	IR0 (interrupt 0)
A05	DATA 4	B05	No connection
A06	DATA 3	B06	No connection
A07	DATA 2	B07	-12 V power
A08	DATA 1	B08	DMA- (CPU enable)
A09	DATA 0	B09	+12 V power
A10	WAIT-	B10	GROUND
A11	LOGIC GROUND	B11	AMWC- (memory write)
A12	ADDRESS 19 (MSB)	B12	MRDC- (memory read)
A13	ADDRESS 18	B13	AIOWC- (I/O write)
A14	ADDRESS 17	B14	IORC- (I/O read)
A15	ADDRESS 16	B15	No connection
A16	ADDRESS 15	B16	No connection
A17	ADDRESS 14	B17	No connection
A18	ADDRESS 13	B18	No connection
A19	ADDRESS 12	B19	No connection
A20	ADDRESS 11	B20	PCLK (5-MHz clock)
A21	ADDRESS 10	B21	IR6 (interrupt 6)

TABLE 4.3 (Continued)

PIN	SIGNAL	PIN	SIGNAL
A22	ADDRESS 9	B22	IR5 (interrupt 5)
A23	ADDRESS 8	B23	IR4 (interrupt 4)
A24	ADDRESS 7	B24	IR2 (interrupt 2)
A25	ADDRESS 6	B25	IR1 (interrupt 1)
A26	ADDRESS 5	B26	No connection
A27	ADDRESS 4	B27	RFSH (refreshing)
A28	ADDRESS 3	B28	ALE (address latch)
A29	ADDRESS 2	B29	+5 V power
A30	ADDRESS 1	B30	OSC (15-MHz clock)
A31	ADDRESS 0 (LSB)	B31	GROUND

However, the VLSI tester II does not need all the signals that are available from the computer. Thus, the following signals which appear on the cable coming from the computer are shown in table 4.4.

TABLE 4.4

THE CABLE

Pin	Function	Pin	Function
1	WAIT	13	A1
2	D0	14	A0
3	D1	15	IOWC
4	D2	16	IORC
5	D3	17	RESET
6	D4	18	CSELECT
7	D5	19	GROUND
8	D6	20	GROUND
9	D7	21	GROUND
10	CS	22	GROUND
11	A3	23	GROUND
12	A2	24	GROUND
		25	GROUND

The addresses for sending this data are shown in table 4.5.

TABLE 4.5
THE VLSI TESTER II ADDRESSES

ADDRESS		FUNCTION
(base address)	300H	MEMORY
	301H	MEMORY
	302H	MEMORY
	303H	MEMORY
	304H	MEMORY
	305H	MEMORY
	306H	MEMORY
	307H	MEMORY
	308H	CONTROL
	309H	CONTROL
	30AH	CONTROL

CHAPTER V

SOFTWARE DESIGN

Design

A portion of the VLSI tester II system is the computer software. The software is needed to send the test data to the data and direction memories. It sets the period and the timer to a proper delay. In addition, it loads the stop counter and error counter. Furthermore, the software monitors the tests performed for any errors that occur. It then takes the necessary actions if an error has occurred. The program was written in Pascal Language, and two subroutines were written in Assembly Language. The files that were linked together for this interface are:

- (1) VLSI2.PAS - CONTROLS THE FLOW OF THE TESTS.
- (2) INOUT.ASM - COMMUNICATES WITH THE HARDWARE.

The code for these sections are found in appendix 2 and appendix 3 respectively. This interface allows the programmer to use the Pascal Language to read the tests that will be performed upon a chip, to convert data, to write

the output data, and to control the controller in the VLSI tester II.

Assembly Language program is used to communicate with the hardware. The purpose of the program is to write or read I/O port to the hardware. It accomplishes this writing from the hardware by using the following instruction:

```
OUT  AL, DX
```

It accomplishes this reading from the hardware by using the following instructions:

```
IN   AL, DX
```

Software Execution

When the software begins execution, it first prints a title page to the screen. This page holds simple information about the VLSI tester II. Next, the name of the test data file should be entered by the user. The entire file name should be entered at this time. The location of the disk drive should proceed the filename if it is different from the current drive. An example of this is:

```
A:Test.Dat
```

This file is previously created by the user (see Test Data File), and holds all the tests to be performed upon a chip. Once the filename has been entered, you should place the chip to be tested on the ZIF socket. In addition, the software asks the user to input the maximum response time of the chip. This is an integer value from 10 to 2550 representing the delay of 10 nano-seconds to 2550 nano-seconds respectively. This delay is the maximum time the chip has to respond to the inputs. If the chip under test does not respond to the proper logic level within this time, an error will occur. Press return key. Now the testing procedure begins.

First, the computer loads the test data into the memories. Then the computer sets the period, the timer, the stop address, and the error address. Finally, it sends the start signal to the VLSI tester II, and wait until the test is finished. After the test is finished, the computer reads the status of the VLSI tester II. If an error occurs, then the software prompts the user to enter the proper error testing mode. Notice that the VLSI tester II terminates before the end of the test if an error occurs during the test procedure. There are four modes that can be entered. These are:

- (1) It stops if an error occurs. The test reads the output data and restarts.
- (2) It stops if an error occurs. The tester reads the

output data and resumes the test.

(3) It accounts the number of errors that has occurred.

(4) It exits.

If test mode (3) is chosen, the test procedure will occur 15 times. It works like binary search technique because of the inability to access the output of the error counter. The first (1) and the second (2) mode allow the user to pin-point the location of the error. Also the computer allows the user to output the location of the errors to a file (see Error Data File). If the test mode (1) is chosen, the software stops and restarts from the beginning every time that an error occurs. Its mode of testing is mainly for chips that contain dynamic latches. The second (2) mode, the program will stop and resume the same address where an error occurs. Its mode of testing is mainly for chips that contain static latches and combination logic. The four (4) mode is to exit the error test procedure.

Once the mode of the test is chosen, the computer loads the VLSI tester II with proper error counter and stop counter. It continues until the error test procedure is terminated.

Test Data File

A major portion of the testing procedure is the test data file. This file holds all the tests that will be performed upon the chip under test. This file is created by

the user following a strict format.

First, the input format must contain the pin corresponding between the ZIF socket and the data input of the file. It does not need to be in order and it can have multiple assignments for the same pin but only the last one will be considered. In addition, pins that are not specified will be considered as zero. Consequently all these unspecified pins will drive logic low at ZIF socket because the VLSI tester II must be prevented from false error indication signal from unused pins.

Second, the input format must contain the direction and the data. To specify direction, you must put the d or D at the first character of the line. Each bit tells the corresponding ZIF pin number whether it is an input or an output. To specify data, you must put the letter t or T at the first character of the line. Each bit sets the corresponding ZIF pin where zero is for low logic level (0V) and one for high logic level (5V). Furthermore, after you specify the letter, you must enter the proper logic level for each pin that is specified. In addition, these numbers may have blank spaces between, but all the data or direction data must be contained in the same line. Every time that you specify the letter t or T, a new test data will be started. When you set the direction, this direction will copy into the direction memory every time you specify a new data until you specify another new direction data. The

direction data must be specified before the data for the test data.

The format of a test data is shown in figure 5.1. Note that for smaller chips, the ZIF pin numbers do not always coincide with the chip pin numbers. See figure 5.2 for the layout of the pin numbers. Notice that if a 14 pin chip is placed into ZIF socket (A), the chip pin numbers and ZIF pin numbers are shown in the table 5.1.

TABLE 5.1
CHIP PIN NUMBERS

Chip Pin #	ZIF Pin #
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	42
9	43
10	44
11	45
12	46

TABLE 5.1 (Continued)

Chip Pin #	ZIF Pin #
13	47
14	48

Thus, the user needs to be especially careful when making a test data file.

- 0 means the ZIF socket pin number is a chip input or the signal goes from VLSI tester II to the ZIF socket.
- 1 means the ZIF socket pin number is a chip output or the signal goes from the ZIF socket to VLSI tester II.

The two lines which contain the letter the t or T are the test data. These are the actual logic states.

```
1 1
2 2
3 3
4 4
48 5
47 6
46 7
45 8
d 01010101
t 11111111
T 00000000
```

Figure 5.1. The Test Data File Format

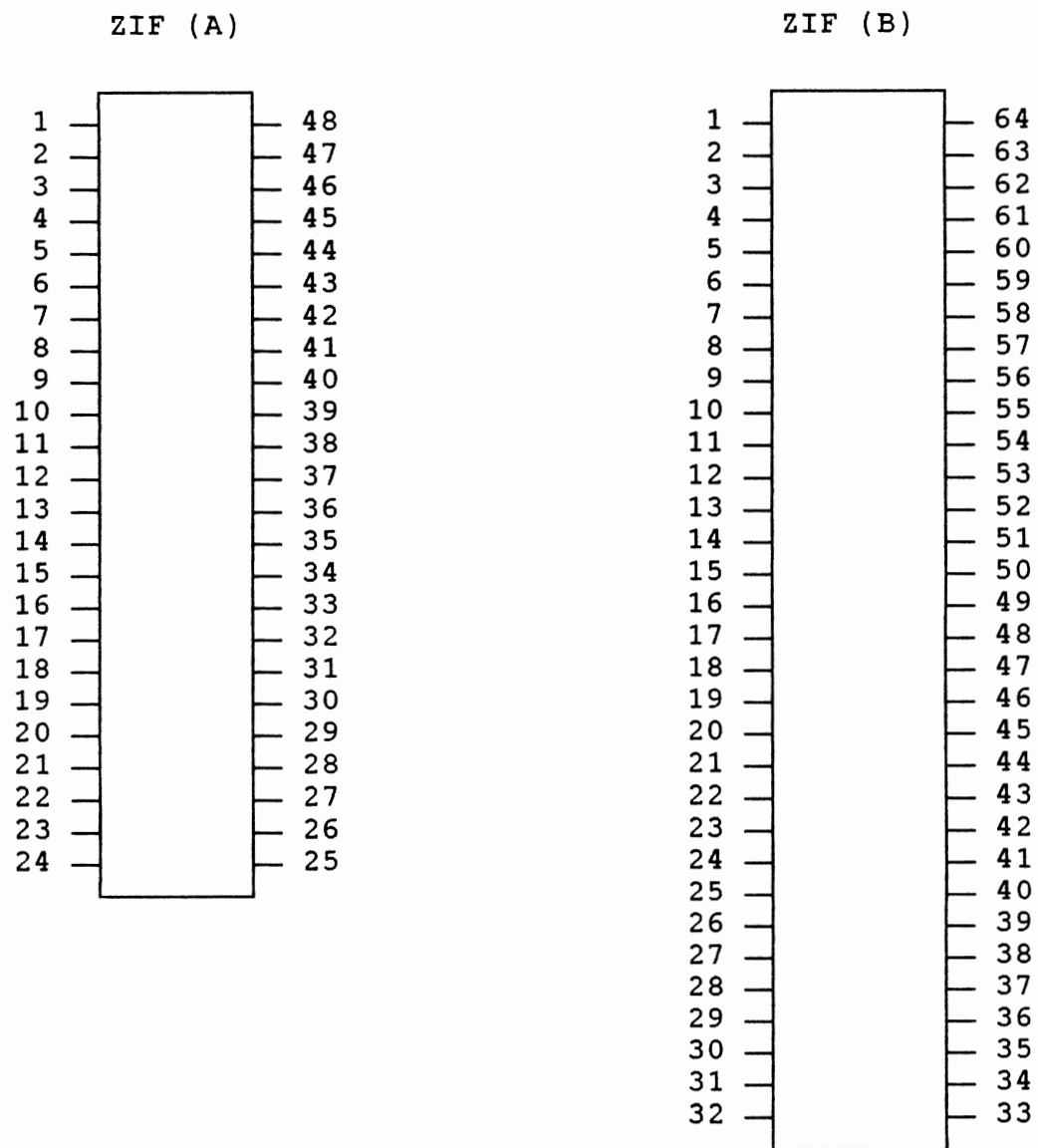


Figure 5.2. The ZIF Pin Numbers

Example of Test Data File

To demonstrate the procedure for creating a test file, a file for a 14 pin 74LS08 2-input AND gate will be created. For the first test of this AND gate, the following data will be used as shown in the table 5.2. (The inputs and outputs for this gate are the same as those from the TTL 1985 Data Book.)

TABLE 5.2
PIN ASSIGNMENT

Chip Pin Number	Directional	Test	Chip Pin Number	Directional	Test
1	Input	0	8	Output	0
2	Input	0	9	Input	0
3	Output	0	10	Input	0
4	Input	0	11	Output	0
5	Input	0	12	Input	0
6	Output	0	13	Input	0
7	GND	0	14	Vcc	1

The Vcc and GND pins should have jumpers attached to the ZIF socket pin numbers. These two pins (GND and Vcc) should be

considered as inputs with the following test data.

GND	-	0
Vcc	-	1

A test file that will send the data shown in figure 5.3 to this chip is shown in figure 5.4.

```
1 1
2 2
3 3
4 4
5 5
6 6
47 13
46 12
45 11
44 10
43 9
42 8
d 001 001 100 100
t 000 000 000 000
t 100 100 010 010
t 010 011 001 001
t 111 111 111 111
t 000 000 000 000
t 100 100 010 010
t 010 010 001 001
t 111 111 111 111
```

Figure 5.3. An Example of a Test Data File

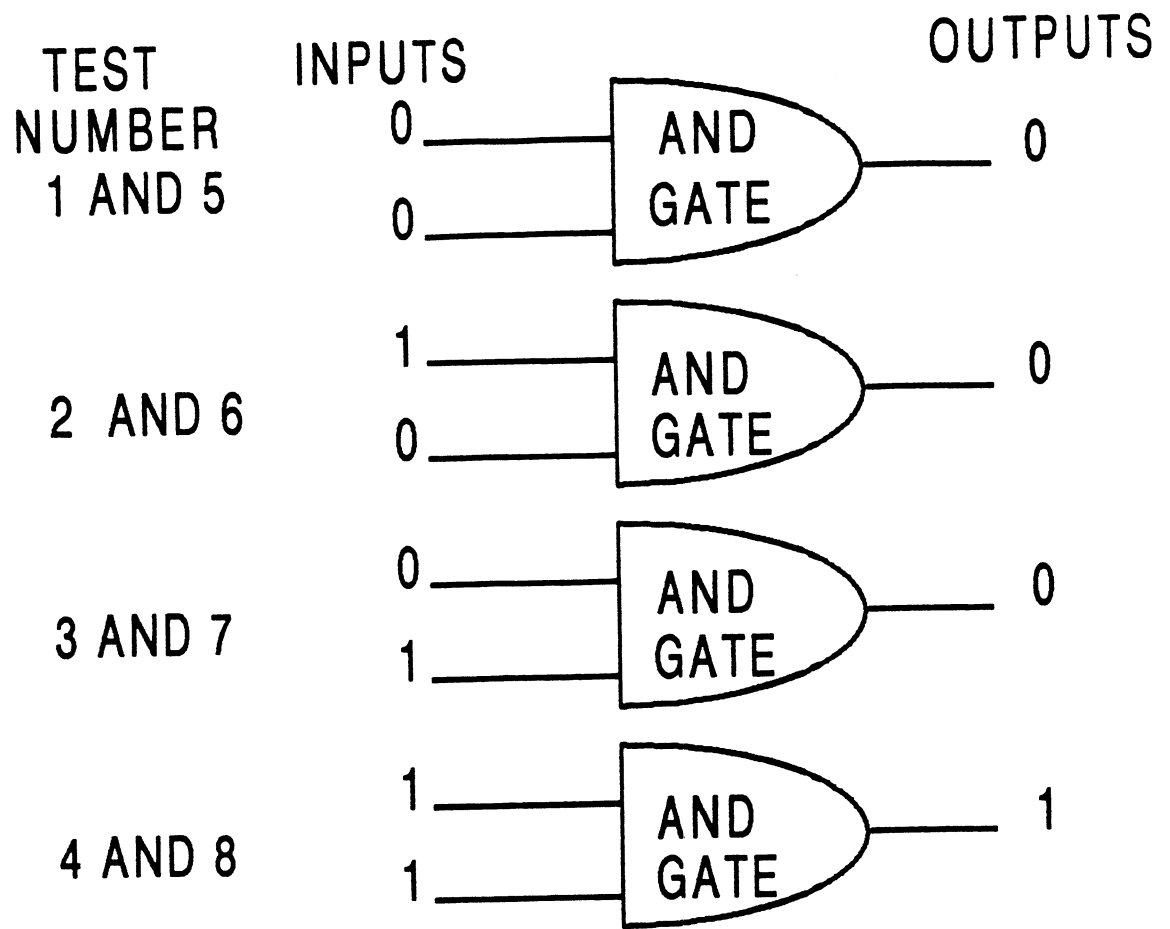


Figure 5.4 Test Example

Error Data File

When the location of the errors found during a test is written to a file, it appears in the following manner as shown in the figure 5.5.

```
2
3
4
5
6
4
4
4
4
4
4
4
d
t
t
t0000001000000
```

Figure 5.5. Output data

This form of output error file allows the user to easily locate the ZIF pin numbers where errors have occurred. All the first character of each line of the input file will be printed in the output file, and only the error occurred will be printed on the pin number. The error will be shown as a 1 in the corresponding ZIF pin number where the error occurs. A 0 will be placed at the ZIF pin number location that acts correctly.

CHAPTER VI

CONCLUSION

The working prototype of VLSI tester II tests the logic and response time of chips with up to 64 pins. It also tests the response time of the chip greater than or equal to 10ns. The device is capable of interfacing with both Texas Instruments and IBM personal computers. The software allows the user to enter the test data file and desired response time of the chip. The user can also select three modes of testing. The first mode counts the number of errors while the second mode stops when an error occurs. Then the tester reads the output data and resumes the test. The third mode stops if an error occurs. Then the tester reads the output data and restarts. The second and the third modes allow the user to pin-point the location of the error. In addition, the design contains its own power supply and computer interface board, thus making the device self sufficient.

There are three distinct differences between the VLSI tester I and VLSI tester II. First of all, the new design has greater output of test data per unit time. Secondly it can test smaller response time. Lastly the software is more user friendly. The design met the following specifications.

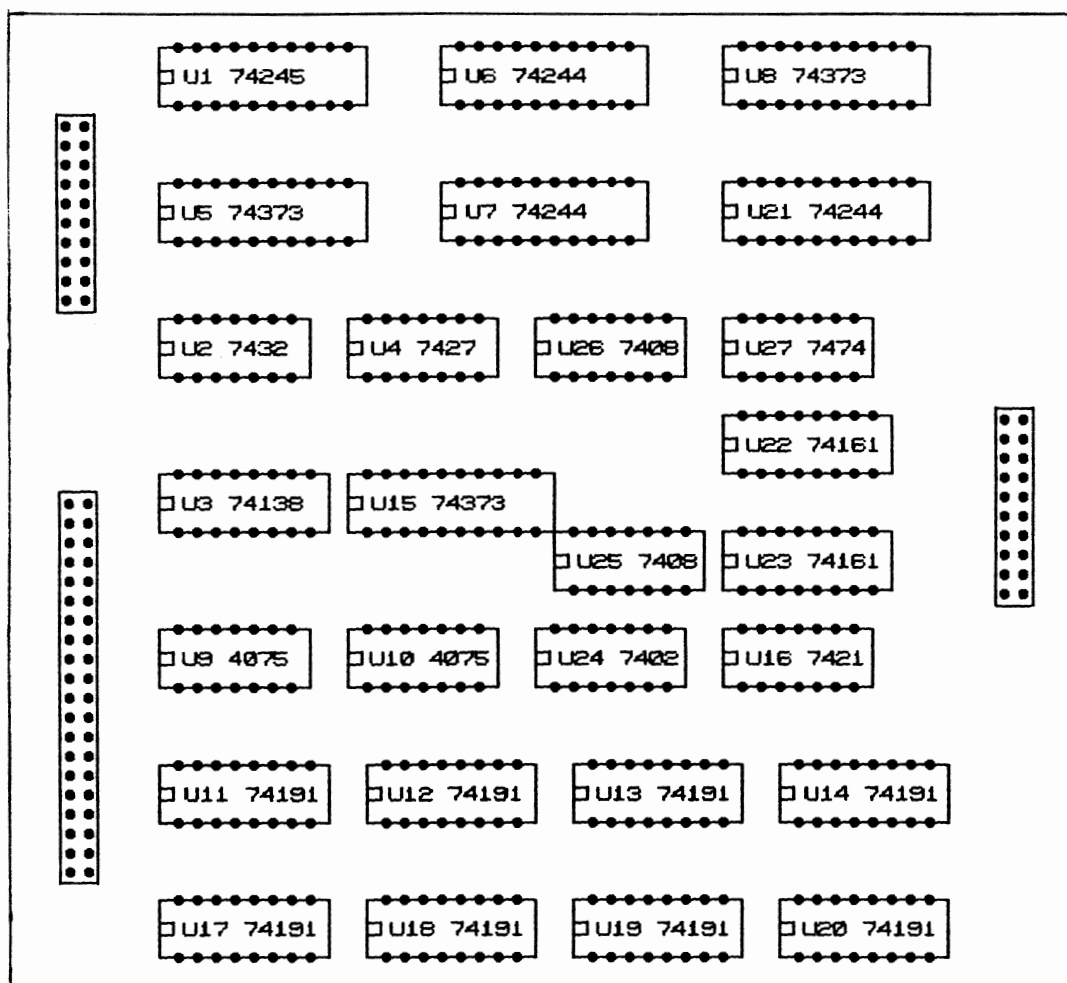
- (1) Ability to test the logic of the chip.
- (2) Ability to test chips with 64 pins or less.
- (3) Test the response time of the chip greater than or equal to 10ns.
- (4) Ability to send a new test data at the period of 100ns or greater.
- (5) Ability to interface with Texas Instruments and IBM personal computers.
- (6) Ability to user defined pins between the ZIF socket and the chip for ease writing of the test data.

REFERENCES

- ALS/AS Logic Data Book, Texas Instruments, 1985.
- Konvalina, John and Wileman, Stanley. Programming With Pascal. New York: McGraw-Hill, Inc., 1987.
- Liu, Yu-Cheng and Gibson, Glenn A. Microcomputer Systems: The 8086/8088 Family. Englewood Cliffs, NJ: Prentice-Hall, 1986.
- MECL System Design Handbook, Motorola, 1988.
- MECL Device Data, Motorola, 1989.
- Microprocessor Data Book, OKI Semiconductor, Texas Instruments, 1987.
- TTL Logic Data Book, Texas Instruments, 1985.

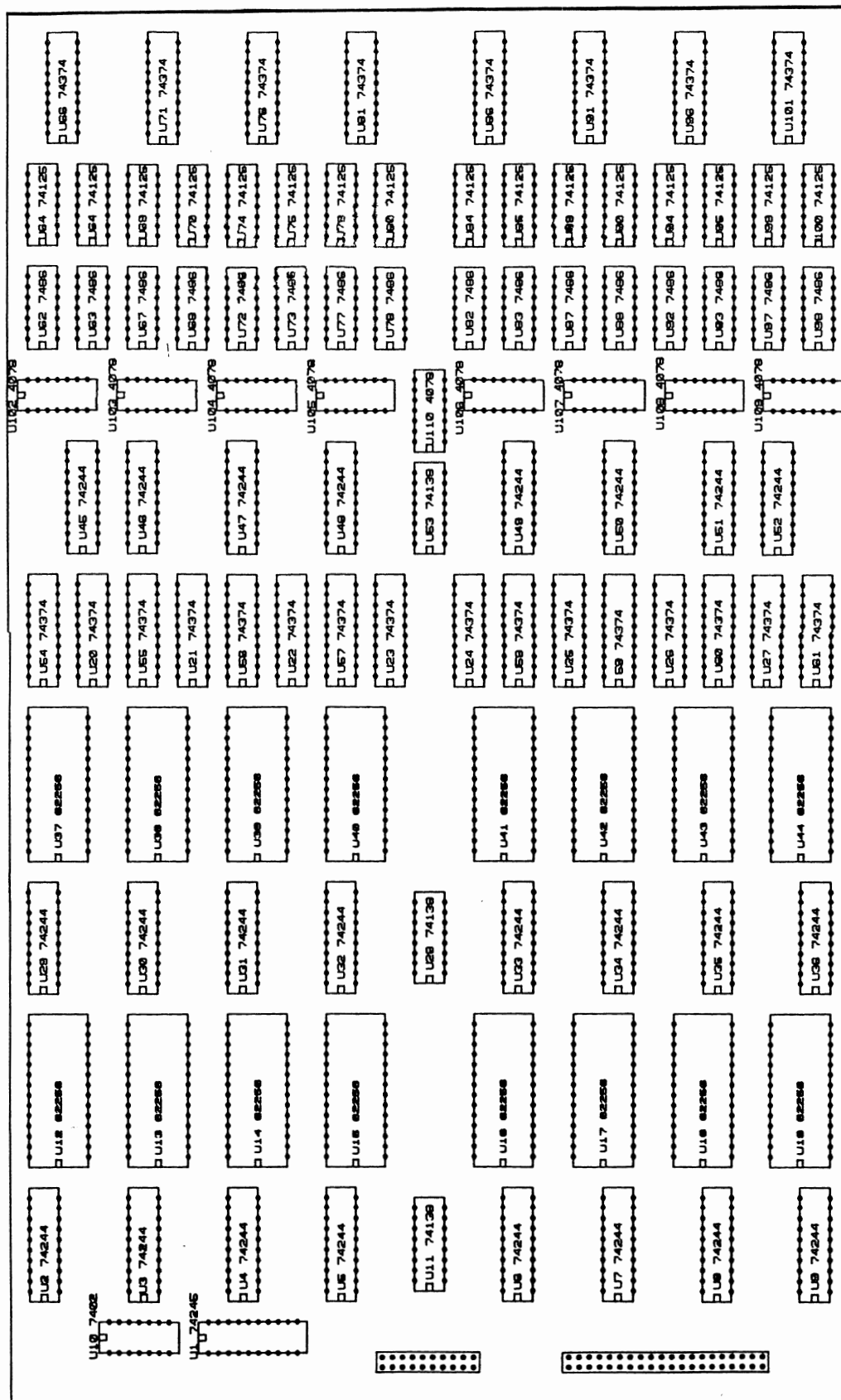
APPENDIX A

THE CONTROLLER LAYOUT



APPENDIX B

THE HARDWARE LAYOUT



APPENDIX C

CODE FOR FILE VLSI2.PAS

```

(*****)
(* PURPOSE: This program is the software for the VLSI tester II. *)
(* PROCEDURE: INN = Read data from the VLSI tester II *)
(* OUT = Write data to the VLSI tester II *)
(* BITON = Verify in a certain bit of a byte is on *)
(* DATAOUT = Send a group of bytes to the VLSI tester *)
(* with the proper encoding of the pin *)
(* position *)
(* DATAIN = Receive a group of bytes from the VLSI *)
(* tester II with the proper decoding of the *)
(* pin position *)
(* ERRORRESTART = It makes the error output file of *)
(* the chip that is being tested *)
(* ERRORCONT = It makes the error output file of the *)
(* chip that is being tested *)
(* ERRORACCOUNT = It accounts the numbers of errors *)
(* that has occurred during the test *)
(* INITIATION = Initialize some variables and the *)
(* introduction of the software *)
(* HIGHIMPEDANCE = It sets the pins of the ZIF socket *)
(* in the high impedance state *)
(* SENDDATAMAIN = It sends to the VLSI tester II the *)
(* direction and test data *)
(* VARIABLE: INFILE = The name of the input test file *)
(* OUTFILE = The name of the output error file *)
(* FILENAME = Temporary variable for the file name *)
(* ACHAR = Temporary variable for the input character *)
(* STATUS = The status of the VLSI tester II *)
(* SECOND = The second number of the pin vector *)
(* position *)
(* DELAY = The amount of delay for the chip to *)
(* respond *)
(* STARTADDRESS = The start address of the test data *)
(* in the memory *)
(* TESTADDRESS = Temporary address of the test data *)
(* NUMBER = Temporary number variable *)
(* INDEX = The index variable for the data *)
(* NUMPIN = The numbers of pins that are defined *)
(* TESTCOND = The value of the controller status *)
(* TESTOPTION = The test option *)
(* TEMPO = temporary variable *)
(* PINPOSITION = The relation between the pin of the *)
(* ZIF socket and the column test data *)
(* DATA = The data array *)
(* DATADIREC = The data direction array *)
(* EXP2 = The array of exponents of base 2 *)
(*****)

```

```
PROGRAM VLSI(INPUT,OUTPUT);
```

```
TYPE
```

```
NUM64 = ARRAY[0..64] OF INTEGER;
```

```

NUM8 = ARRAY[0..8] OF INTEGER;

VAR
  INFILE,OUTFILE : TEXT;
  FILENAME : LSTRING(30);
  ACHAR,STATUS : CHAR;
  SECOND : BOOLEAN;
  DELAY,STARTADDRESS,TESTADDRESS,NUMBER,INDEX:INTEGER;
  NUMPIN,TESTCOND,TESTOPTION,TEMPO:INTEGER;
  PINPOSITION,DATA,DATADIREC : NUM64;
  EXP2 : ARRAY[1..8] OF INTEGER;
CONST
  A0 = 768;
  A8 = 776;
  A9 = 777;
  AA = 778;
  LADDRESS = 251;
  LERROR = 253;
  LTIMER = 254;
  LPERIOD = 254;
  LDIRECTION = 254;
  LDATA = 253;
  LSTART = 7;
  LSTOP = 255;
  ROUTPUT =251;

(*procedure debug;
var
loop:integer;
begin
  for loop:=1 to 64 do
    if data[loop] = 0
    then write('0')
    else write('1');
writeln;
end;*)

(*****
(* PROCEDURE: INN *)
(* PURPOSE: Read data from the VLSI tester II. It reads twice *)
(* and if they are equal, then it returns with the data*)
(* VARIABLE: ADDRESS = The address of the I/O port *)
(* VALU = The read data *)
(*****)

PROCEDURE INN(ADDRESS:INTEGER; VAR VALU:INTEGER);EXTERN;

(*****
(* PROCEDURE: OUT *)
(* PURPOSE: Write data to the VLSI tester II. It writes twice *)
(* to the VLSI tester II. *)
(*****)

```



```

(* VARIABLE: ADDRESS = The address of the I/O port *)
(*          VALU = The output data *)
(*****)

```

```
PROCEDURE OUT(ADDRESS,VALU:INTEGER);EXTERN;
```

```

(*****)
(* PROCEDURE: BITON *)
(* PURPOSE: It is to check if the bit of a byte is on *)
(* VARIABLE: NUM = The byte to be tested *)
(*          BITPOSITION = Which bit to be checked *)
(*****)

```

```
FUNCTION BITON(NUM,BITPOSITION:INTEGER):BOOLEAN;
```

```

BEGIN
  BITON := ODD(NUM DIV EXP2[BITPOSITION])
END;
```

```

(*****)
(* PROCEDURE: DATAOUT *)
(* PURPOSE: It sends a group of bytes to the VLSI tester II with*)
(*          the proper encoding from the pin position array *)
(* VARIABLE: DATA = The data to be sent to the VLSI tester II *)
(*          PINPOSITION = The relation between the ZIF socket *)
(*                   and the data column *)
(*****)

```

```
PROCEDURE DATAOUT(DATA,PINPOSITION:NUM64);
```

```

VAR
  BYTE,VALU,LOOP:INTEGER;
```

```

BEGIN
  DATA[0]:=0;
  FOR BYTE := 0 TO 7 DO
    BEGIN
      VALU := 0;
      FOR LOOP := 1 TO 8 DO
        IF DATA[PINPOSITION[8*BYTE+LOOP]] <> 0
          THEN VALU := VALU + EXP2[LOOP];
        OUT(A0 + BYTE,VALU);
      END
    END
  END;
```

```

(*****)
(* PROCEDURE: DATAIN *)
(* PURPOSE: It receives a group of bytes from the VLSI tester II*)
(*          with the proper encoding from the pin position array*)

```

```
(* VARIABLE: DATA = The data to be sent to the VLSI tester II *)
(*          PINPOSITION = The relation between the ZIF socket *)
(*          and the data column *)
(*****)
```

```
PROCEDURE DATAIN(VAR DATA:NUM64; PINPOSITION:NUM64);
```

```
VAR
```

```
    BYTE,VALU,LOOP:INTEGER;
    DATASOCKET : NUM64;
```

```
BEGIN
```

```
    FOR LOOP := 1 TO 64 DO
        DATA[LOOP] := 0;
        FOR BYTE := 0 TO 7 DO
            BEGIN
                INN(A0 + BYTE,VALU);
                FOR LOOP := 1 TO 8 DO
                    BEGIN
                        DATASOCKET[8*BYTE+LOOP]:= VALU MOD 2;
                        VALU := VALU DIV 2
                    END;
                END;
            FOR LOOP := 1 TO 64 DO
                DATA[PINPOSITION[LOOP]] := DATASOCKET[LOOP];
            END;
```

```
(*****
(* PROCEDURE: ERRORRESTART *)
(* PURPOSE: It makes the error output file of the chip that is *)
(*          being tested *)
(* VARIABLE: ERRORLOOP = Number of errors to stop the test *)
(*          TESTCOND = The value of the controller status *)
(*          FILEPOSITION = The position pointer of the input *)
(*                      file *)
(*          TEMPO = temporary variable *)
(*          DELTADDRESS = The difference between the address of*)
(*                      the test error and the position *)
(*                      pointer of the input file *)
(*          LOOP = The loop variable *)
(*          ERRORADDRESS = The address of the test error *)
(*****)
```

```
PROCEDURE ERRORRESTART;
```

```
VAR
```

```
    ERRORLOOP,TESTCOND,FILEPOSITION,TEMPO,DELTADDRESS,LOOP:INTEGER;
    ERRORADDRESS:INTEGER;
    ACHAR : CHAR;
```

```

BEGIN
  RESET(INFILE);
  FILEPOSITION := -1;
  OUT(AA,LADDRESS);
  OUT(A8,(ERRORLOOP MOD 256));
  OUT(A9,(ERRORLOOP DIV 256));
  OUT(AA,LSTOP);
  OUT(AA,LERROR);
  OUT(A8,1);
  INN(AA,TESTCOND);
  IF BITON(TESTCOND,1) (*increment the error counter if error *)
  THEN ERRORLOOP := 1 (*indication is on *)
  ELSE ERRORLOOP := 0;
  TESTCOND := 0;
  WHILE NOT(BITON(TESTCOND,2)) DO (*repeat until the test *)
  BEGIN (*reaches the end *)
    OUT(AA,LADDRESS);
    OUT(A8,(ERRORLOOP MOD 256));
    OUT(A9,(ERRORLOOP DIV 256));
    OUT(AA,LSTOP);
    OUT(AA,LERROR);
    OUT(A8,1);
    INN(AA,TESTCOND);
  (*write(testcond);*)
  IF BITON(TESTCOND,1) (*increment the error counter if error*)
  THEN BEGIN (*indication is on *)
    OUT(A8,((ERRORLOOP + 1) MOD 256));
    OUT(A9,((ERRORLOOP + 1) DIV 256));
  END
  ELSE BEGIN
    OUT(A8,(ERRORLOOP MOD 256));
    OUT(A9,(ERRORLOOP DIV 256));
  END;
  OUT(A9,0);
  OUT(AA,LADDRESS);
  OUT(A8,(STARTADDRESS MOD 256));
  OUT(A9,(STARTADDRESS DIV 256));
  OUT(AA,LSTART);
  INN(AA,TESTCOND);
  IF NOT(BITON(TESTCOND,2)) (*if has an error then print on *)
  THEN BEGIN (*the error output file *)
    INN(A8,TEMPO);
    INN(A9,ERRORADDRESS);
    ERRORADDRESS := 256*ERRORADDRESS + TEMPO + 1;
    DELTADDRESS := STARTADDRESS - ERRORADDRESS - FILEPOSITION;
  (*writeln(testcond,erroraddress);*)
    IF DELTADDRESS > 0
    THEN BEGIN
      FILEPOSITION := FILEPOSITION + DELTADDRESS;
      FOR LOOP := 1 TO DELTADDRESS DO
        BEGIN (*advance deltaaddress positions in the input file*)

```

```

        READLN(INFILE);
        READ(INFILE,ACHAR);
        WRITELN(OUTFILE);
        WRITE(OUTFILE,ACHAR);
        WHILE (ACHAR <> 'T') AND (ACHAR <> 't') DO
        BEGIN
            READLN(INFILE);
            READ(INFILE,ACHAR);
            WRITELN(OUTFILE);
            WRITE(OUTFILE,ACHAR);
        END;
    END;
    OUT(AA,ROUTPUT);
    DATAIN(DATA,PINPOSITION);
    FOR LOOP := 1 TO NUMPIN DO (*write to the output file *)
    BEGIN
        IF DATA[LOOP] = 0
        THEN WRITE(OUTFILE,'0')
        ELSE WRITE(OUTFILE,'1');
    END;
    ERRORLOOP := ERRORLOOP + 1; (*increment the error loop *)
(*debug;*)
END
ELSE BEGIN
    WRITELN('ERRORS ARE INCONSISTENT');
    ERRORLOOP := ERRORLOOP +1;
END;
END;
END;
END;
END;

```

```

(*****
(* PROCEDURE: ERRORCONT *)
(* PURPOSE: It makes the error output file of the chip that is *)
(*           being tested *)
(* VARIABLE: ERRORLOOP = Number of errors to stop the test *)
(*           TESTCOND = The value of the controller status *)
(*           FILEPOSITION = The position pointer of the input *)
(*                       file *)
(*           TEMPO = temporary variable *)
(*           DELTADDRESS = The difference between the address of*)
(*                       the test error and the position *)
(*                       pointer of the input file *)
(*           LOOP = The loop variable *)
(*           ERRORADDRESS = The address of the test error *)
(*****

```

```
PROCEDURE ERRORCONT;
```

```
VAR
```

```
    TESTCOND, TEMPO, ERRORADDRESS, DELTADDRESS, LOOP: INTEGER;
```

```

FILEPOSITION:INTEGER;
ACHAR:CHAR;

BEGIN
  RESET(INFILE);
  TESTCOND := 0;
  FILEPOSITION := -1;
  ERRORADDRESS := STARTADDRESS;
  WHILE NOT(BITON(TESTCOND,2)) DO
    BEGIN
      OUT(AA,LADDRESS);
      OUT(A8,(ERRORADDRESS MOD 256));
      OUT(A9,(ERRORADDRESS DIV 256));
      OUT(AA,LSTOP);
      OUT(AA,LEERROR);
      OUT(A8,1);
      INN(AA,TESTCOND);
      IF BITON(TESTCOND,1)
      THEN OUT(A8,1)
      ELSE OUT(A8,0);
      OUT(A9,0);
      OUT(AA,LADDRESS);
      OUT(A8,(ERRORADDRESS MOD 256));
      OUT(A9,(ERRORADDRESS DIV 256));
      OUT(AA,LSTART);
      INN(AA,TESTCOND);
      IF NOT(BITON(TESTCOND,2))
      THEN BEGIN
        INN(A8,TEMPO);
        INN(A9,ERRORADDRESS);
        ERRORADDRESS := 256*ERRORADDRESS + TEMPO + 1;
        (*writeln(testcond,erroraddress);*)
        DELTADDRESS := STARTADDRESS - ERRORADDRESS - FILEPOSITION;
        IF DELTADDRESS > 0
        THEN BEGIN
          FILEPOSITION := FILEPOSITION + DELTADDRESS;
          FOR LOOP :=1 TO DELTADDRESS DO
            BEGIN
              READLN(INFILE);
              READ(INFILE,ACHAR);
              WRITELN(OUTFILE);
              WRITE(OUTFILE,ACHAR);
              WHILE (ACHAR <> 'T') AND (ACHAR <> 't') DO
                BEGIN
                  READLN(INFILE);
                  READ(INFILE,ACHAR);
                  WRITELN(OUTFILE);
                  WRITE(OUTFILE,ACHAR);
                END;
              END;
            OUT(AA,ROUTPUT);
            DATAIN(DATA,PINPOSITION);

```

```

        FOR LOOP := 1 TO NUMPIN DO
            IF DATA[LOOP] = 0
                THEN WRITE(OUTFILE,'0')
                ELSE WRITE(OUTFILE,'1');
(*debug;*)
        END
    ELSE BEGIN
        ERRORADDRESS := ERRORADDRESS - 1;
    END;
END;
END;
END;

(*****
(* PROCEDURE: ERRORACCOUNT *)
(* PURPOSE: It accounts the errors that it has occurred during *)
(*           the test *)
(* VARIABLE: ERRORLOOP = Number of errors to stop the test *)
(*           TESTCOND = The value of the controller status *)
(*           LOOP = The loop variable *)
(*****)

PROCEDURE ERRORACCOUNT;

VAR
    VALU, ERRORLOOP, TESTCOND, LOOP: INTEGER;

BEGIN
    VALU := 16384;
    ERRORLOOP := VALU;
    FOR LOOP := 0 TO 13 DO (*it makes the test 14 times for the *)
        BEGIN (*binary search *)
            OUT(AA,LERROR);
            OUT(A8,(ERRORLOOP MOD 256));
            OUT(A9,(ERRORLOOP DIV 256));
            OUT(AA,LADDRESS);
            OUT(A8,(STARTADDRESS MOD 256));
            OUT(A9,(STARTADDRESS DIV 256));
            OUT(AA,LSTART);
            INN(AA,TESTCOND);
            VALU := VALU DIV 2;
            IF (BITON(TESTCOND,2) AND NOT(BITON(TESTCOND,4)))
                THEN ERRORLOOP := ERRORLOOP - VALU
                ELSE ERRORLOOP := ERRORLOOP + VALU;
            END;
            WRITELN('NUMBER OF ERROR = ',ERRORLOOP);
        END;
    END;

(*****
(* PROCEDURE: INITIATION *)

```

```
(* PURPOSE: Initiate some variables and the introduction of      *)
(*           the software                                         *)
(*****)
```

```
PROCEDURE INITIATION;
```

```
BEGIN
```

```
EXP2[1] := 1;
EXP2[2] := 2;
EXP2[3] := 4;
EXP2[4] := 8;
EXP2[5] := 16;
EXP2[6] := 32;
EXP2[7] := 64;
EXP2[8] := 128;
```

```
WRITELN(' +-----+');
WRITELN(' |                                     |');
WRITELN(' |               VLSI2 CHIP TESTER       |');
WRITELN(' |                                     |');
WRITELN(' |      ADVISOR:  DR. LOUIS JOHNSON        |');
WRITELN(' |                                     |');
WRITELN(' |      SPONSOR:  OKLAHOMA STATE UNIVERSITY |');
WRITELN(' |               SCHOOL OF ELECTRICAL ENGINEERING |');
WRITELN(' |                                     |');
WRITELN(' |      PROJECT FROM  ROBERT IMARK         |');
WRITELN(' |                                     |');
WRITELN(' | REMINDER: READ THE USER MANUAL BEFORE OPERATING |');
WRITELN(' |               THE DEVICE.                 |');
WRITELN(' |                                     |');
WRITELN(' +-----+');
WRITELN;
END;
```

```
(*****
(* PROCEDURE: HIGHIMPEDANCE                                     *)
(* PURPOSE: It sets the pins of the ZIF socket in the high    *)
(*           impedance state                                    *)
(*****)
```

```
PROCEDURE HIGHIMPEDANCE;
```

```
VAR
```

```
LOOP: INTEGER;
```

```
BEGIN
```

```
  OUT(AA, LADDRESS);
  OUT(A8, 1);
  OUT(A9, 0);
  OUT(AA, LDIRECTION);
  FOR LOOP := 0 TO 7 DO
```

```

    OUT(A0 + LOOP,255);
OUT(AA,LERROR);
OUT(A8,255);
OUT(AA,LTIMER);
OUT(A8,1);
OUT(AA,LPERIOD);
OUT(A9,1);
OUT(AA,LSTART);
OUT(AA,LSTOP);
END;

```

```

(*****
(* PROCEDURE: SENDDATAMAIN *)
(* PURPOSE: It sends to the VLSI tester II the direction and the*)
(*          test data *)
(*****

```

```

PROCEDURE SENDDATAMAIN;

```

```

BEGIN
  RESET(INFILE);
  WHILE NOT EOF(INFILE) DO
    BEGIN
      NUMBER := 0;
      STATUS := 'N';
      SECOND := FALSE;
      INDEX := 0;
      WHILE NOT EOLN(INFILE) DO
        BEGIN
          READ(INFILE,ACHAR);
          IF (ACHAR = 'D') OR (ACHAR = 'd')
            THEN STATUS := 'D'
          ELSE IF (ACHAR = 'T') OR (ACHAR = 't')
            THEN STATUS := 'T'
          ELSE IF (ACHAR <= '9') AND (ACHAR >= '0')
            THEN CASE ACHAR OF
              '0': NUMBER := NUMBER*10;
              '1': NUMBER := NUMBER*10 + 1;
              '2': NUMBER := NUMBER*10 + 2;
              '3': NUMBER := NUMBER*10 + 3;
              '4': NUMBER := NUMBER*10 + 4;
              '5': NUMBER := NUMBER*10 + 5;
              '6': NUMBER := NUMBER*10 + 6;
              '7': NUMBER := NUMBER*10 + 7;
              '8': NUMBER := NUMBER*10 + 8;
              '9': NUMBER := NUMBER*10 + 9
            END;
          IF STATUS = 'N'
            THEN IF (ACHAR = ' ') OR EOLN(INFILE)
              THEN BEGIN
                IF SECOND = TRUE

```



```

        THEN PINPOSITION[TEMPO] := NUMBER
        ELSE BEGIN
            SECOND := TRUE;
            TEMPO := NUMBER;
            NUMBER := 0
        END
    END;
    IF (STATUS <> 'N') AND (ACHAR <> ' ')
    THEN BEGIN
        DATA[INDEX] := NUMBER;
        NUMBER := 0;
        INDEX := INDEX +1;
    END;
    END;
    IF STATUS <> 'N'
    THEN BEGIN
        (*write(testaddress);*)
        (*debug;*)
        IF STATUS = 'T'
        THEN BEGIN
            OUT(AA,LADDRESS);
            OUT(A8,(TESTADDRESS MOD 256));
            OUT(A9,(TESTADDRESS DIV 256));
            OUT(AA,LDATA);
            DATAOUT(DATA,PINPOSITION);
            OUT(AA,LDIRECTION);
            DATAOUT(DATADIREC,PINPOSITION);
            TESTADDRESS := TESTADDRESS - 1;
        END
        ELSE DATADIREC := DATA;
    END;
    READLN(INFILE);
    END;
END;

PROCEDURE RUNCONTINUOUS;

VAR LOW,HIGH:INTEGER;

BEGIN
    LOW := STARTADDRESS MOD 256;
    HIGH := STARTADDRESS DIV 256;
    WRITELN('TO STOP TURN OFF THE VLSI TESTER II');
    WHILE TESTCOND < 31 DO
    BEGIN
        OUT(AA,249);
        OUT(A8,LOW);
        OUT(A9,HIGH);
        OUT(AA,LSTART);
        INN(AA,TESTCOND);
        WHILE NOT(BITON(TESTCOND,2)) DO

```

```

    INN(AA,TESTCOND);
END;
WRITELN('DO YOU WISH TO RUN ANOTHER TEST (Y,N)?');
READLN(ACHAR);
END;

(* MAIN PROGRAM *)

BEGIN
  INITIATION;
  REPEAT (*reset the pinposition and the data *)
    FOR INDEX := 1 TO 64 DO
      BEGIN
        PINPOSITION[INDEX] := 0;
        DATA[INDEX] := 0
      END;
    HIGHIMPEDANCE;
    WRITELN('PLEASE PLACE THE CHIP ON THE ZIF SOCKET');
    WRITELN('INPUT THE TEST DATA FILENAME. ');
    READLN(FILENAME); (*enter the input file and the time delay*)
    ASSIGN(INFILE,FILENAME);
    RESET(INFILE);
    REPEAT
      WRITELN('INPUT THE RESPONSE TIME DELAY (10 to 2550ns)');
      READLN(TEMPO)
    UNTIL (TEMPO >= 10) AND (TEMPO <= 2550);
    DELAY:= TRUNC(TEMPO/10);
    WRITELN('THE DELAY TIME IS = ',(DELAY*10),'ns');
    OUT(AA,LTIMER);
    OUT(A8,(255-DELAY));
    DELAY:= TRUNC((DELAY*10 + 130 )/50 + 0.9);
    IF DELAY < 2 THEN DELAY := 2;
    WRITELN('THE DATA FREQUENCY IS = ',(1/(DELAY*5E-8)),'Hz');
    DELAY:= 241 - DELAY;
    OUT(AA,LPERIOD);
    OUT(A9,DELAY);
    STARTADDRESS := 0;
    NUMPIN := 0;
    WHILE NOT EOF(INFILE) DO (*counts the number of tests and *)
      BEGIN (*the number of defined pins *)
        READLN(INFILE,ACHAR);
        IF (ACHAR = 'T') OR (ACHAR = 't')
          THEN STARTADDRESS := STARTADDRESS + 1
        ELSE IF (ACHAR >= '0') AND (ACHAR <= '9')
          THEN NUMPIN := NUMPIN + 1;
        END;
      TESTADDRESS := STARTADDRESS;
      SENDDATAMAIN;
      OUT(AA,LADDRESS);
      OUT(A8,(STARTADDRESS MOD 256));
      OUT(A9,(STARTADDRESS DIV 256));

```

```

OUT(AA,LERROR);
OUT(A8,1);
INN(AA,TESTCOND);
IF BITON(TESTCOND,1)
THEN OUT(A8,1)
ELSE OUT(A8,0);
OUT(A9,0);
OUT(AA,LADDRESS);
OUT(A8,(STARTADDRESS MOD 256));
OUT(A9,(STARTADDRESS DIV 256));
OUT(AA,LSTART);    (*make a test
REPEAT
    INN(AA,TESTCOND);
    UNTIL BITON(TESTCOND,2) OR BITON(TESTCOND,4);
(*writeln(testcond);*)
    WRITELN;
    IF NOT(BITON(TESTCOND,4))
    THEN WRITELN('NO ERROR HAS OCCURRED')
    ELSE BEGIN
        WRITELN('ERROR HAS OCCURRED');
        REPEAT
WRITELN;
WRITELN('DO YOU WANT TO SEE THE OUTPUT ERROR OF THE CHIP?');
WRITELN('IT WILL RESTART THE TEST EVERY TIME THAT AN ERROR OCCURS');
WRITELN('TYPE THE NUMBER 1 AND THE RETURN KEY. ');
WRITELN;
WRITELN('DO YOU WANT TO SEE THE OUTPUT ERROR OF THE CHIP?');
WRITELN('IT WILL CONTINUE THE TEST AT THE SAME POINT AN ERROR OCCURS. ');
WRITELN('EVERY TIME THAT AN ERROR OCCURS, THE TEST STOPS FOR 1
MILLISECOND. ');
WRITELN('TYPE THE NUMBER 2 AND THE RETURN KEY. ');
WRITELN;
WRITELN('DO YOU WANT TO COUNT THE NUMBER OF ERRORS?');
WRITELN('TYPE THE NUMBER 3 AND THE RETURN KEY. ');
WRITELN;
WRITELN('TO STOP THE TEST. ');
WRITELN('TYPE THE NUMBER 4 AND THE RETURN KEY. ');
        READLN(TESTOPTION);
        IF TESTOPTION <= 2
        THEN BEGIN
            WRITELN('INPUT THE ERROR DATA FILENAME. ');
            READLN(FILENAME);
            ASSIGN(OUTFILE,FILENAME);
            REWRITE(OUTFILE);
        END;
        IF TESTOPTION = 1
        THEN ERRORRESTART
        ELSE IF TESTOPTION = 2
            THEN ERRORCONT;
        IF TESTOPTION = 3
        THEN ERRORACCOUNT;
        IF TESTOPTION <= 2

```

```
        THEN CLOSE(OUTFILE);
    UNTIL (TESTOPTION = 4)
END;
CLOSE(INFILE);
WRITELN('DO YOU WISH TO RUN ANOTHER TEST (Y,N)?');
WRITELN('OR DO YOU WANT TO RUN THE TEST CONTINUALLY AND');
WRITELN('BE ABLE TO SEE THE OUTPUT RESPONSE OF THE CHIP');
WRITELN('WITH A OSCILLOSCOPE ?      ENTER THE LETTER      C');
READLN(ACHAR);
IF (ACHAR = 'C') OR (ACHAR = 'c')
    THEN RUNCONTINUOUS;
UNTIL ((ACHAR <> 'Y') AND (ACHAR <> 'y'));
END.
```

APPENDIX D

CODE FOR FILE INOUT.ASM

```

NAME      INOUT

DATA1     SEGMENT
DATA1     ENDS

STSEG     SEGMENT STACK
STSEG     ENDS

CODE1     SEGMENT
          ASSUME CS:CODE1, DS:DATA1, SS:STSEG

          PUBLIC OUT
          PUBLIC INN
OUT        PROC FAR
          PUSH BP
          PUSH AX
          PUSH DX
          MOV BP,SP
          MOV AX,6+4[BP]
          MOV DX,8+4[BP]
          OUT DX,AL
          OUT DX,AL
          POP DX
          POP AX
          POP BP
          RET 4
OUT        ENDP
INN        PROC FAR
          PUSH BP
          PUSH AX
          PUSH BX
          PUSH DX
          MOV BP,SP
          MOV DX,8+6[BP]
LOOP:      IN AL,DX
          MOV BL,AL
          IN AL,DX
          CMP AL,BL
          JNZ LOOP
          MOV AH,0
          MOV BX,6+6[BP]
          MOV [BX],AX
          POP DX
          POP BX
          POP AX
          POP BP
          RET 4
INN        ENDP

CODE1     ENDS
          END

```

APPENDIX E

THE USER`S MANUAL

Operating the VLSI tester II device is rather simple, just follow the steps below. Do not skip any of the steps, since it could cause damage to the chip under test.

STEPS

- (1) Create a test data file that holds all the tests to be performed upon the chip. Make sure that the data file is correct. Refer to Chapter V to learn how to create a test data file.
- (2) Attach the ribbon cable to the VLSI tester II device. If the computer locks up during a test, the ribbon cable may not be attached properly.
- (3) Determine which pins on the ZIF socket correspond to power and ground pins. Take the female connector from the corresponding ZIF pin number male connector. Use the jumpers provided to attach these pins to the power and ground pins available. Make sure the jumper is connected to the male connector closest to the ZIF socket. A jumper to the available clock pins can optionally be used.

- (4) Power up the VLSI tester II device.
- (5) Run the VLSI2 executable file on the computer.
- (6) Follow the instructions given by the software.

WARNING: Make sure that the chip is placed correctly within the socket and the jumpers are attached to the correct pins. Otherwise the chip under test may be damaged.

APPENDIX F

MANUFACTURERS LIST

Manufacturers List

- A. Texas Instrument
7613 East 63rd place
Tulsa, Oklahoma 74133
Tel. (918) 250-0611
- B. National Semiconductor
2900 Semiconductor Drive
Santa Clara, California 95051
Tel. (408) 339-9240
- C. Digi-Key
P.O. Box 677
Thief River Falls, MN 56701
Tel. (800) 344-4539
- D. JAMECO
- E. MOTOROLA
- F. SIGNETIC

VITA

Robert R. Imark

Candidate for the Degree of
Master of Science

Thesis: THE VERY LARGE SCALE INTEGRATED CIRCUIT TESTER II

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Curitiba, Brazil, July 3, 1967.

Education: Graduated from Santa Maria High School, Brazil in December 1985; received Bachelor of Science in Electrical and Computer Engineering from Oklahoma State University in May 1991; and completed requirements for the Master of Science degree at Oklahoma State University in July, 1992.

Professional Experience: Summer work in the Frontier Engineering Company and International Ceramics in 1990.

