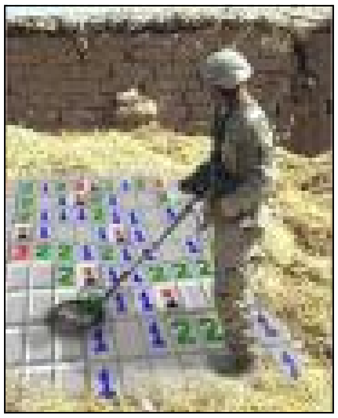


Report for

ECE 6710

Final Project – Minesweeper

Prepared by

Will Felt, Evan Neal, Vamsi Mudarapu, Jordan Savage

Date
4/25/06

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1.0 INTRODUCTION

The computer game Minesweeper is a game that was made popular by the Windows OS. The computer game Minesweeper starts the game by showing you a blank grid of squares. Some squares conceal mines; the rest are safe. Your task is to work out where the mines are without detonating any of them. You do this by choosing a square. If there's a mine underneath it, the mine is detonated and the game ends — - with a loss for you, of course. If there is no mine, however, the computer writes a number in that square, telling you how many mines there are in the eight immediately adjacent squares (horizontally, vertically, and diagonally).

If your first guess hits a mine, you're unlucky: you get no information except that you've lost. If it doesn't, though, then you get partial information about the location of nearby mines. You use this information to influence your next choice of square, and again either you detonate a mine and lose, or you gain information about the positions of nearby mines. If you wish, you can choose to mark a square as containing a mine: if you're wrong, you lose. Proceeding in this way, you can win the game by locating and marking all the mines.

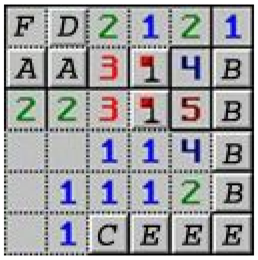


Figure 1 - Typical Minesweeper Board

For instance, after a few moves you might reach the position shown in Fig.1. Here a flag shows a known mine (position already deduced), the numbers are the information you've gotten from the computer, and the letters mark squares whose status is as yet untested. With a little thought, you can deduce that the squares marked A must contain mines, because of the 2's just below them. The squares marked B must also contain mines, because of the 4's and 5's nearby. In the same way, C must contain a mine; and it then follows that D and E do not. The status of F can then be deduced, after a few moves, by uncovering D and seeing what number appears. [1]

2.0 METHODS

In our project we have developed this game using only our cell library developed in ECE 6700. We developed modules to perform specific functions that when connected together create a display that can then be interacted with to create the game atmosphere and give the player the opportunity to select squares based on the knowledge of the surrounding squares given him. The inputs/outputs to the fabricated chip are listed below along with their corresponding purpose.

2.1.1 *Used Inputs:*

"Up_in": input controlled by user
"Down_in": input controlled by user
"Left_in": input controlled by user
"Right_in": input controlled by user
"Clk" : Clock
"Rst": Reset input controlled by user

The Cursor Position Module inside the chip takes input from the user and adjusts the cursor position accordingly. If the user asserts input Up_in, output vPos<2:0> increments one. If the user asserts input Down_in, output vPos<2:0> decrements one. If the user asserts Right_in, output hPos<2:0> increments one. If the user asserts Left_in, output hPos<2:0> decrements one. User is able to change cursor position as long as input Playing is high, indicating that the user has not already lost or won. Is the input Playing is low, indicating that the user has won or lost, the cursor position no longer updates the output according to the user input. Upon asserting reset, win and lose are set to zero, thus enabling input from user.

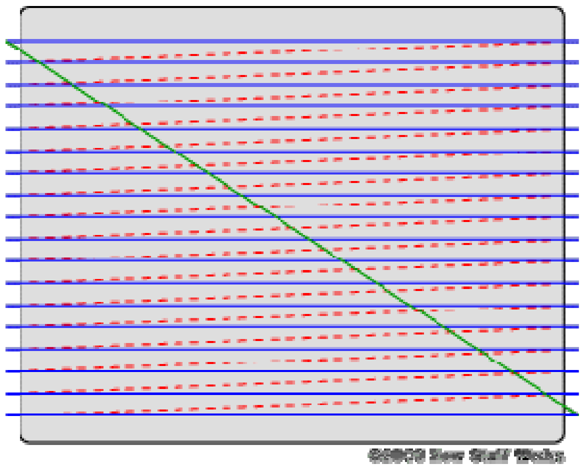


Figure 2 - CRT Scanning Pattern

2.1.2 *Used Outputs:*

"hVideo"
"vVideo"
"Vidout"

hVideo:

The hVideo Module is based on free-running 9 bit counter. This counter is used to create the timing cycle for the horizontal display. The 9 bit counter is called hcnt[8:0] with HA[6:0] being the unmodified lowest 7 of the 9 bits. The counter starts at 000 and counts up to 9'h12B which is 299. The cycle time of the counter is 30.0 microseconds (300 cycles @ 100 nsec/cycle). vCntEna is sent to the Vertical Module (vVideo) when hcnt is 6. The Vertical Module will only change state when vCntEna is asserted. hSync starts when hcnt is 1 and stops when hcnt is 9'h014. hBright starts when hcnt is 9'h080 and stops when hcnt is 9'h0FF as shown. [2] This enables the horizontal control of the graphics display as shown in Figure 2 - CRT Scanning Pattern.

vVideo:

The vVideo module is based on a 10 bit counter that changes value only when vCntEna is asserted. This counter will be used to create the timing cycle depicted in this cycle. The 10 bit counter, Vcnt[9:0], at 000 and counts up to 10'h22B which is 555. The count changes only in the cycle after vCntEna is asserted. Since vCntEna is asserted every 30 microseconds, the period of each frame is 16.68 milliseconds (30 microseconds/line * 556 lines). vBright starts when Vcnt is 0 and ends when it is 10'h0ff. vSync is asserted for three line times between Vcnt[9:0] == 10'h187 and Vcnt[9:0]==10'h189 . [2]

Vidout:

The Vidout is a video output signal. This signal is asserted in the bright parts of the image and not asserted where the image is dark.

3.0 RESULTS

Once all of the modules were in place and functioning correctly, testing began to see if the results would be what were planned. The following are several screen captures showing the game in various stages of play during a simulation.

1. Original screen after the player had pushed the reset button signaling the commencement of a new game. The curser begins in the upper left hand corner and is created by inverting the bits making a dark square with the silhouette of the appropriate symbol inside. The question marks show unselected squares.

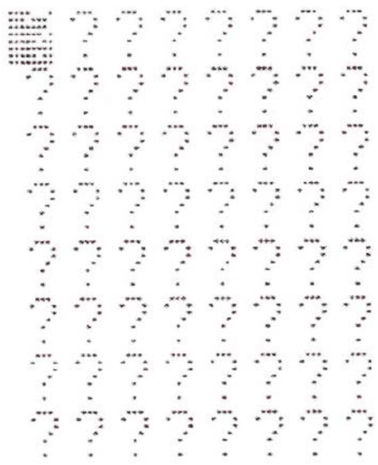
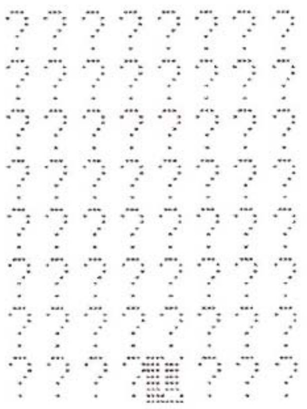


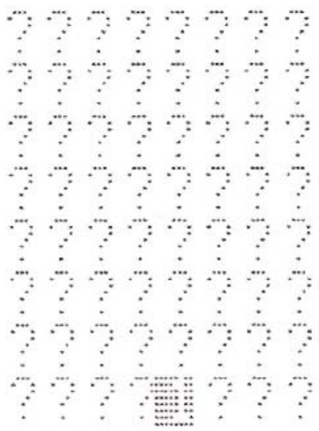
Figure 3 - Reset Game Board

2. The player has now and moved to the right four times, down seven times and has selected the square. Once revealed the square will either show a number or a mine depending on the configuration of the board in this case there were no mines nearby.



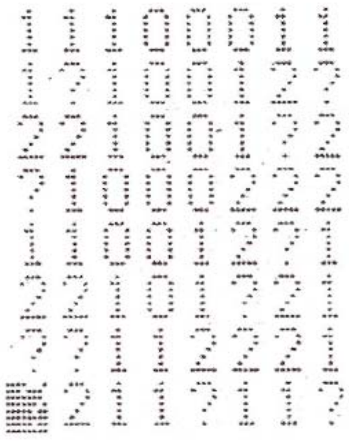
```
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ▦  ? ? ?
```

3. The player has now reset the board and moved to the right four times, down seven times and has selected the square. Once revealed the square will either show a number or a mine depending on the configuration of the board in this case there were no mines nearby. So we can see that the boards are changing each time the board is reset.



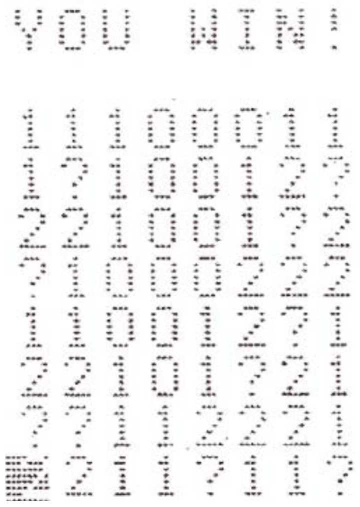
```
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ▦  ? ? ?
```

4. This shows regular game play just before the player has selected a winning or losing tile. This was to verify that the computer wouldn't pre-emptively display a winning or losing message.



```
1 1 1 0 0 0 1 1
1 ? 1 0 0 1 2 ?
2 2 1 0 0 1 ? 2
? 1 0 0 0 2 2 2
1 1 0 0 1 2 ? 1
2 2 1 0 1 ? 2 1
? ? 1 1 2 2 2 1
▦ 2 1 1 ? 1 1 ?
```

5. In this scenario the player has won the game. All numbered tiles have been revealed and no mines have been selected. Once the player has one the game, the cursor is locked in position. Despite attempts to move the cursor it remained in the same position.

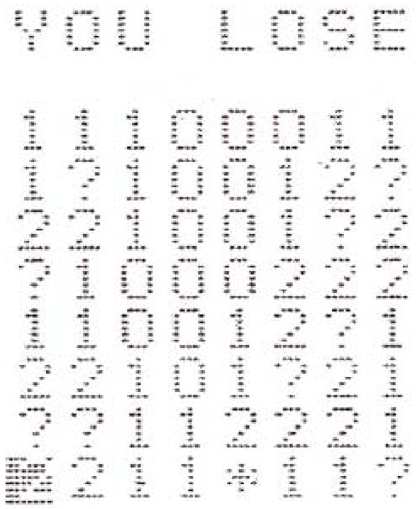


```
YOU WIN!

11100011
1?10012?
2210017?2
?1000222
110012?1
22101?21
??112221
▓211?11?
```

Figure 4 - "You Win!" Game board

6. The player has now selected a mine so the "You Lose" message is displayed and the player has quickly moved 4 tiles to the left. The scenario has showed the game was lost but it was discovered if the player moves the cursor before the screen refreshes the cursor can still move. Once the screen has refreshed and the player was unable to move the cursor.



```
YOU LOSE

11100011
1?10012?
22100172
?1000222
110012?1
22101?21
??112221
▓211*11?
```

Figure 5 - "You Lose" Game board

These screens were as planned and seemed to do what they were designed to do. This was major milestone in the completion of our project.

4.0 DISCUSSION

The first thing that was done when the chip arrived was to build a test fixture (See Figure 6) to see if it was working correctly (See attached schematic in APPENDIX B). This was designed by Evan. Once built the chip was placed in the fixture according to the documentation provided by MOSIS. After several attempts to seat the chip correctly it was discovered that the chip was manufactured with a 90 degree rotation. Once this was discovered, and the chip was reinserted, everything worked except two pins. These were then found by trial and error using a power supply set to 5 volts.

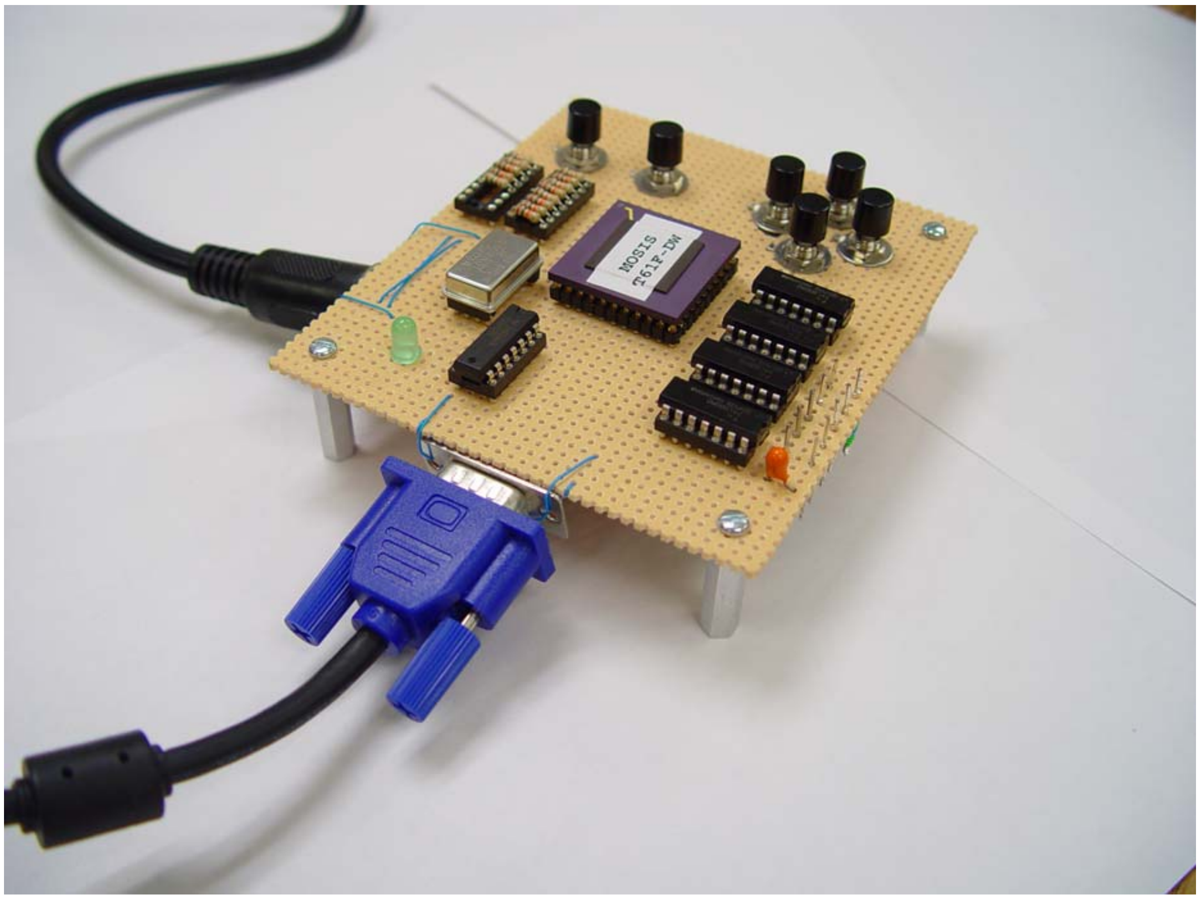
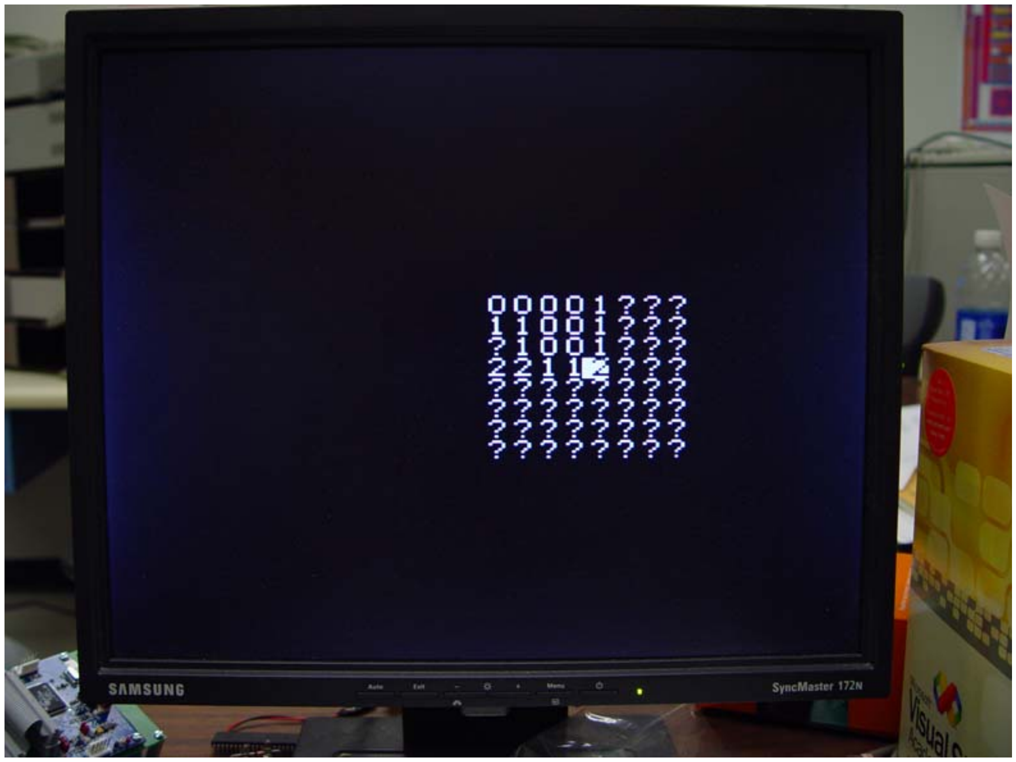


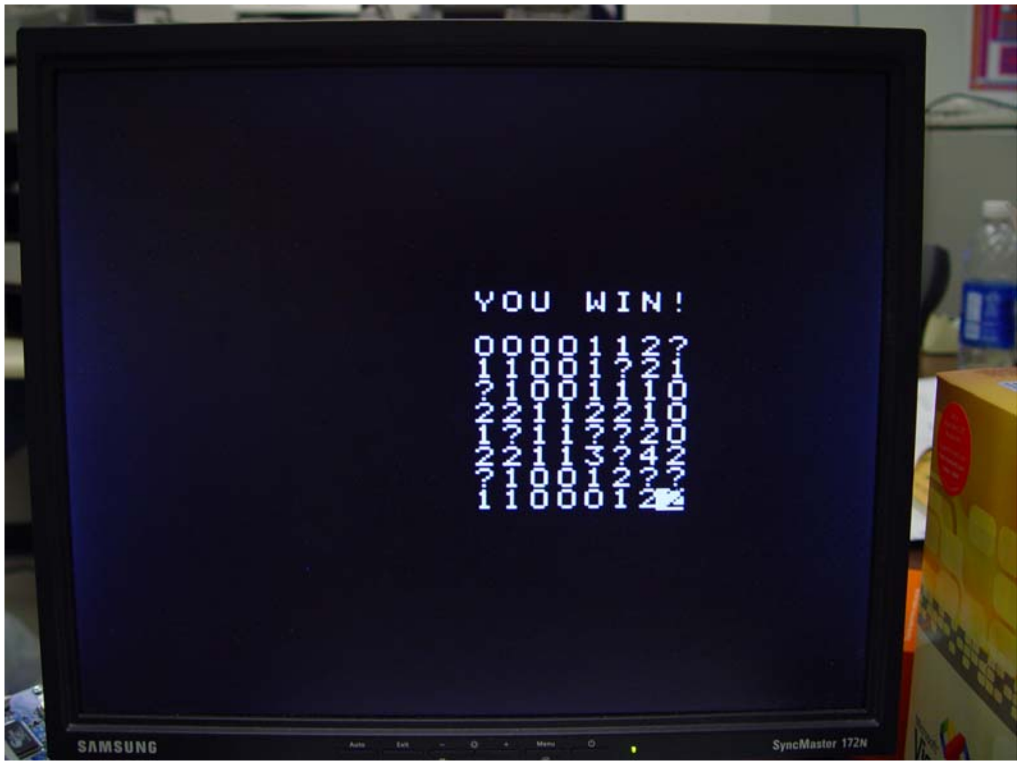
Figure 6 - Circuit Card Test Fixture

Once all of these fabrication errors were found the chip worked perfect and tests were a performed in a like manner to the above simulations and were a complete success. A few screens are shown below.



```
00001???
11001???
?1001???
22112???
????????
????????
????????
????????
```

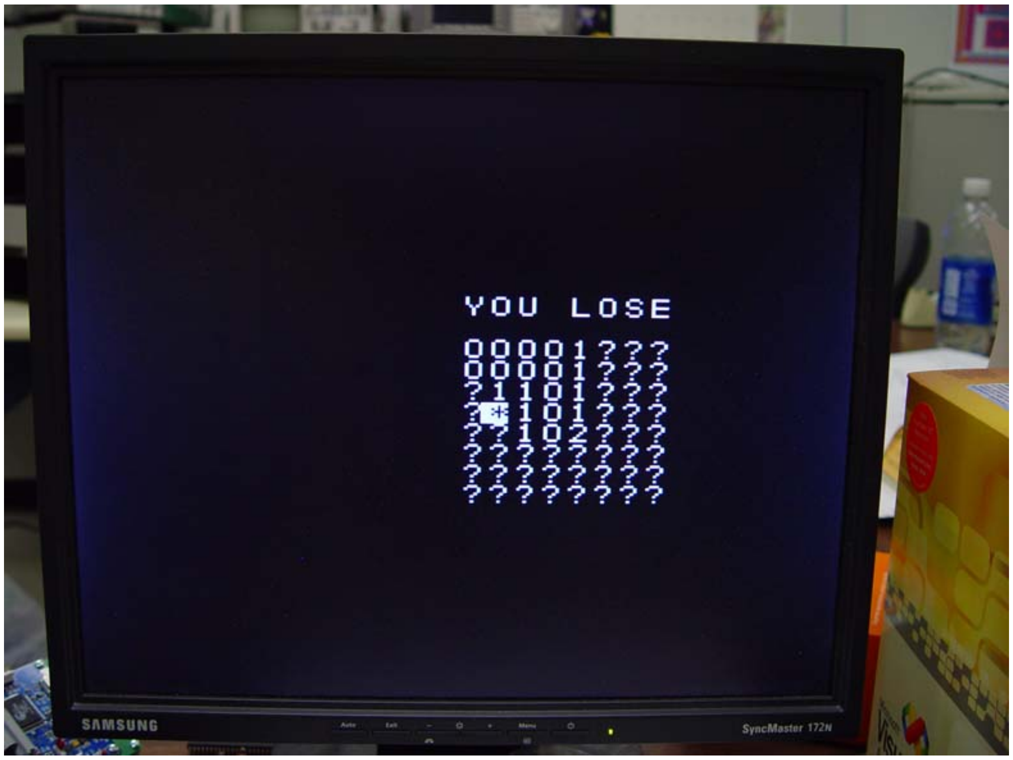
Figure 7 - Actual Screen



```
YOU WIN!

0000112?
11001?21
?1001110
22112210
1?11??20
22113?42
?10012??
11000122
```

Figure 8 - "You Win" Screen



```
YOU LOSE

00001???
00001???
?1101???
?*101???
??102???
????????
????????
????????
```

Figure 9 - "You Lose" Screen

In order to further characterize the chip, a Schmoo plot was created using a Tektronix LV500 chip tester and logic analyzer LV500. After several attempts to create the Schmoo plot and a few repairs to the LV500, the Schmoo plot was obtained. The plot is of power supply voltage vs. flip-flop hold time. Based on this test, the maximum clock frequency the device is able to work at is 1/8ns = 125MHz and can work at power supply voltages above 3V. The result can be seen in Figure 10.

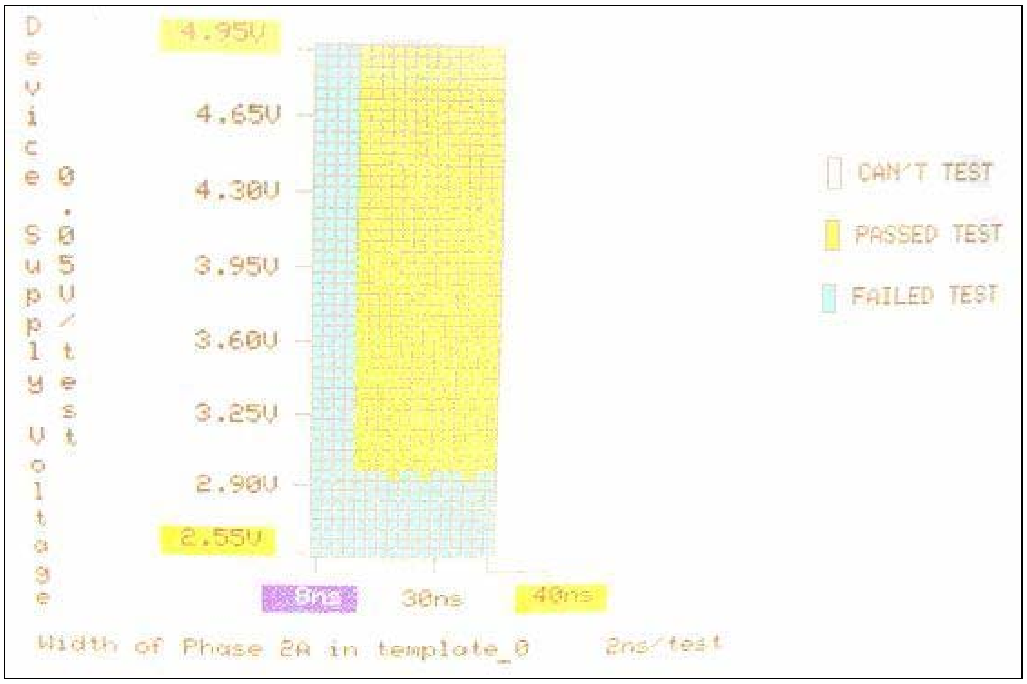


Figure 10 - Schmoo Plot

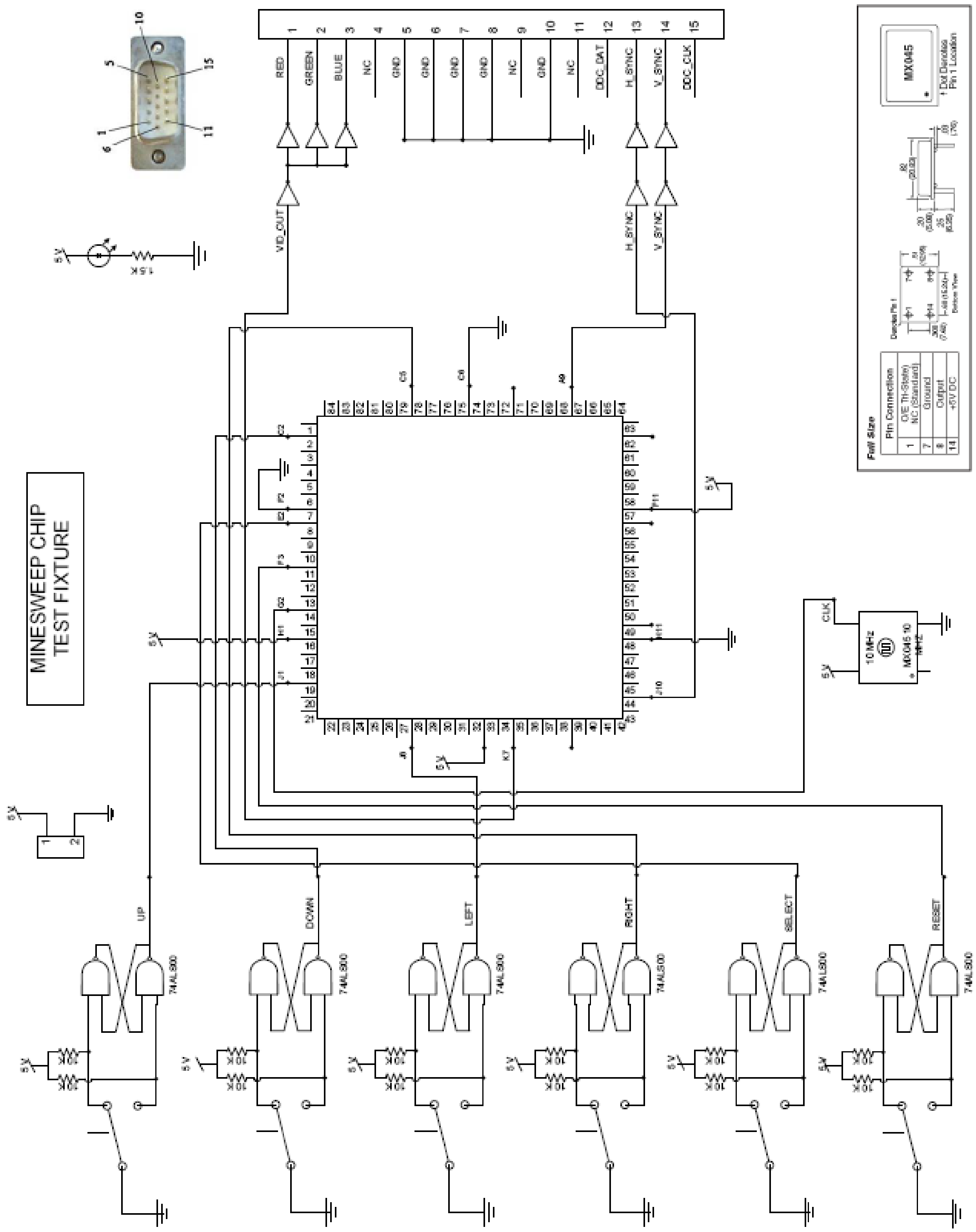
5.0 CONCLUSION AND RECOMMENDATIONS

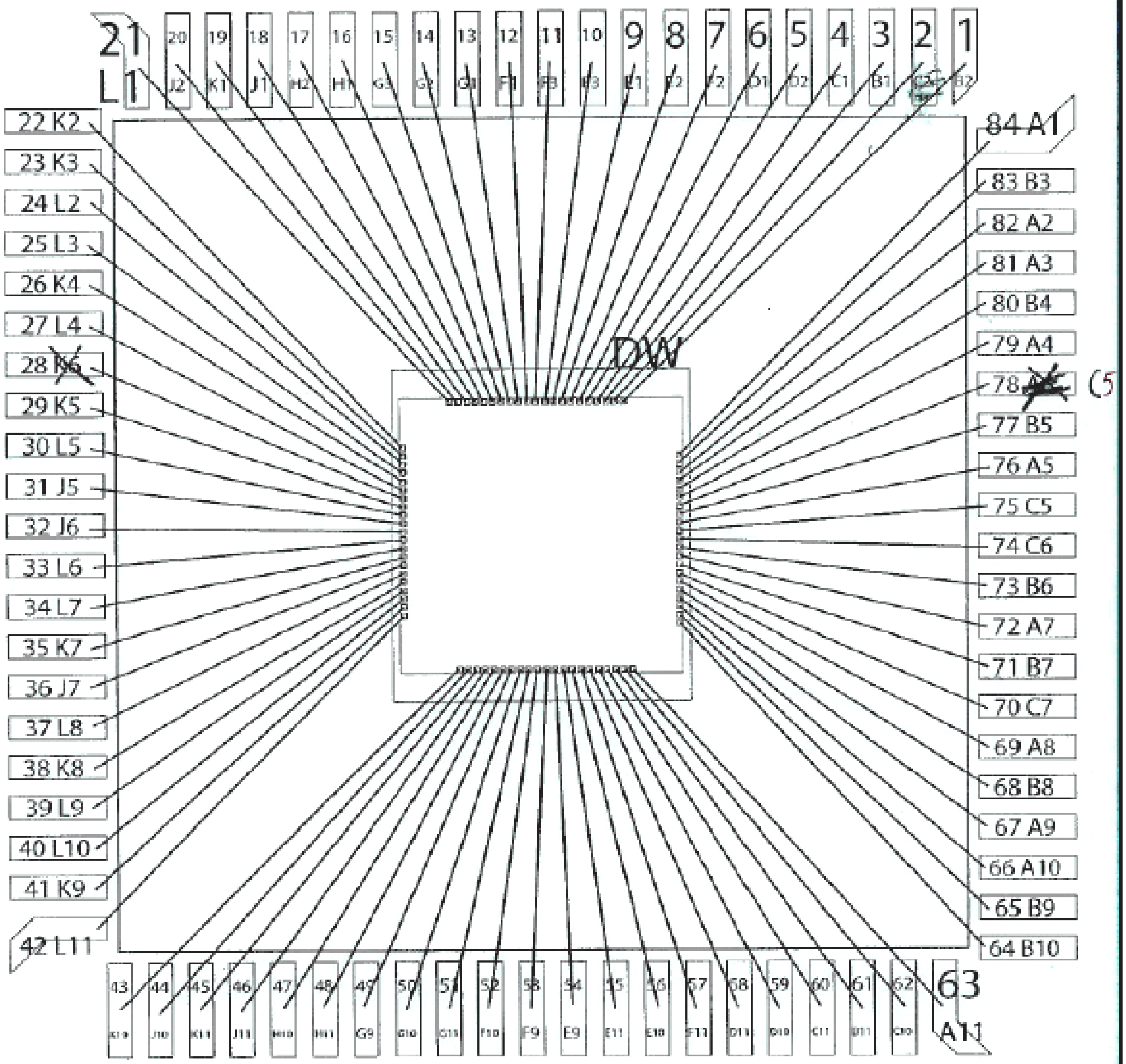
The chip was fully functional once the proper inputs and outputs were identified. The chip tester also could have been a little newer but it seemed to finally do what it was designed to do. In the end, the design was a success and everything seemed to perform in the way it was designed. Overall this was excellent design experience from start to finish.

APPENDIX A - REFERENCES

[1] Ian Stewart: http://www.claymath.org/Popular_Lectures/Minesweeper/
[2] Allen Tanner:http://www.cs.utah.edu/classes/cs5710/MiniProject/CAD7aht.ht

APPENDIX B – DOCUMENTATION





Qty: 5	T61F-DW	PGA84M
	(74693)	

Customer Providing Diagram (approved)

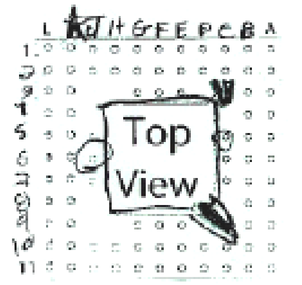
MOSIS

Design_name: wholechipname

Customer Account: 2818

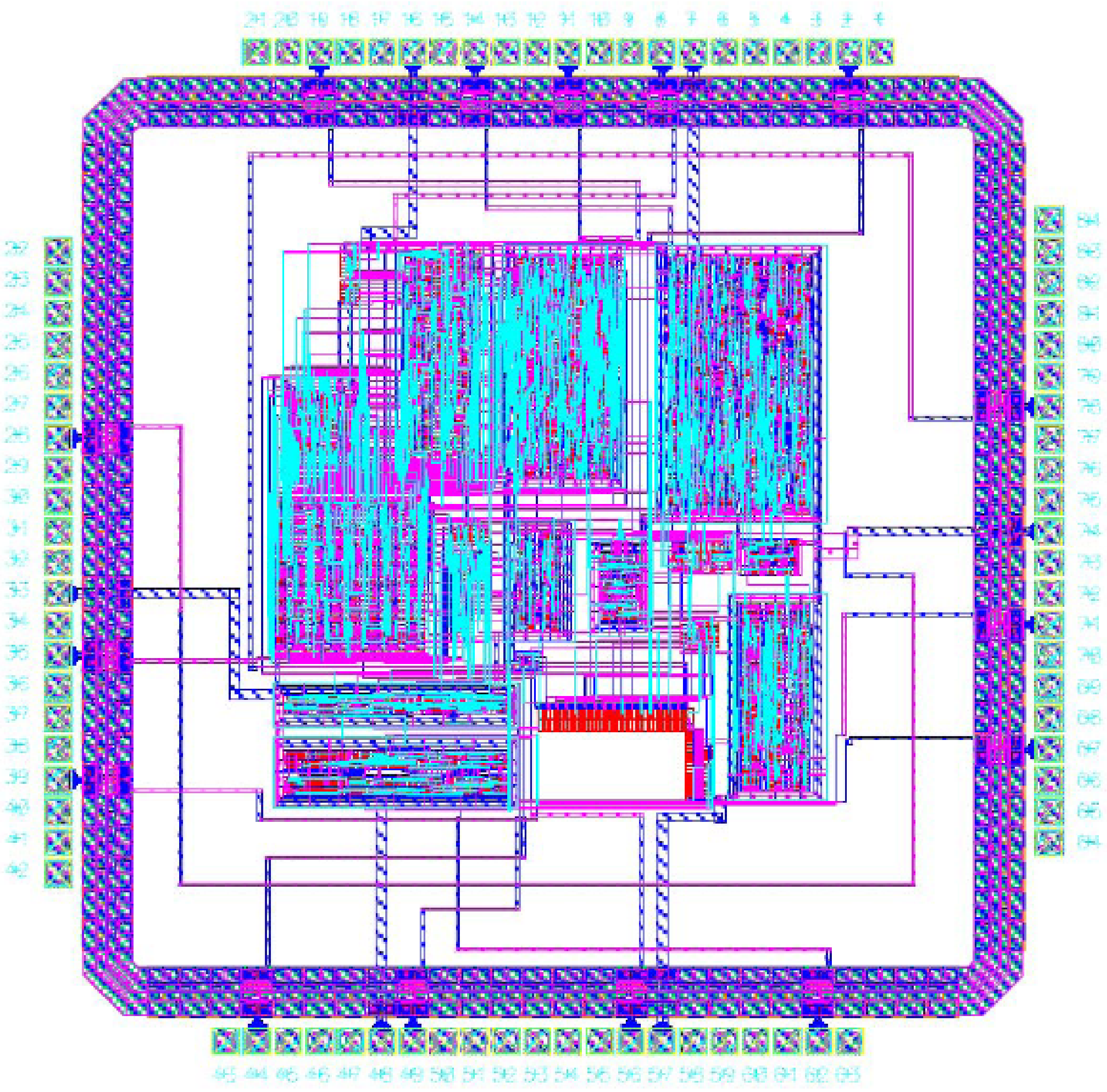
Approximate Max Die Size: 3099 um x 3556 um. Cutting might produce larger die.

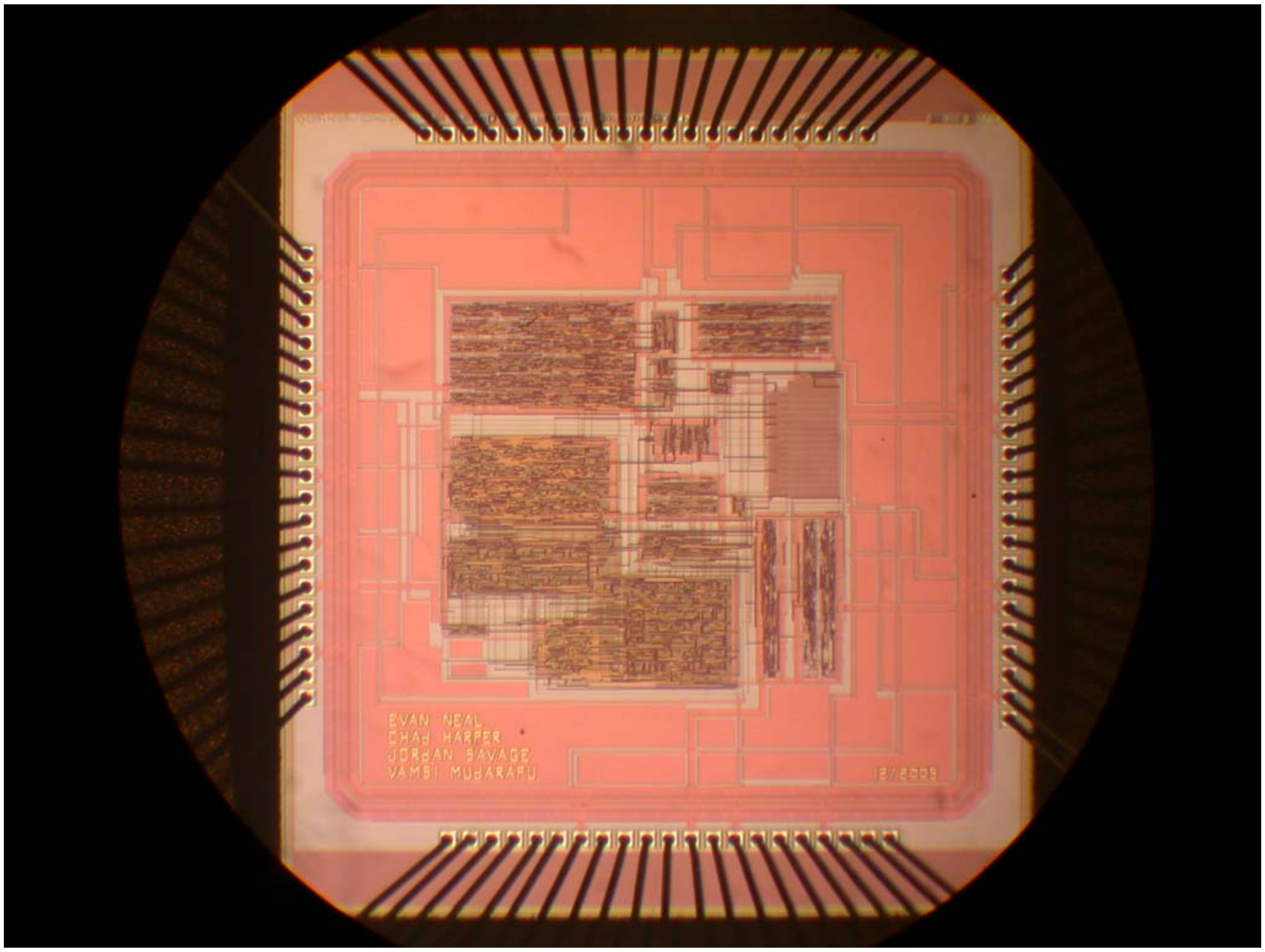
Cavity Size: 8890 um x 8890 um



03-FEB-2006 11:11:55

wholechip2_Layout_plot.pdf




EVAN NEAL
CHAD HARPER
JORDAN SAVAGE
VAMSI MUDARAPU
12/2005