Digital Waveform Generator using EEPROM ROM

By

Saeid Moslehpour
moslehpou@hartford.edu
Electrical & Computer Engineering
University of Hartford

Edward Filteau
filteau@hartford.edu
Electrical & Computer Engineering
University of Hartford

Dave Fairbanks
dftuna@aol .com
Electrical & Computer Engineering
University of Hartford

Sergio Nieto
oteins@msn.com
Electrical & Computer Engineering
University of Hartford

Abstract: *For a long time TTL (Transistor Transistor Logic) integrated circuits (7400 series) have been used to design digital circuits; with the creation of ROM (Read Only Memory), SPLD (Simple Programmable Logic Devices), CPLD (Complex Programmable Logic Devices) and FPGA (Field Programmable Gate Array), digital design has been easier than ever. This paper discusses the programming and simulation of ROM family devices such as EPROM (Erasable Programmable Read Only Memory), EEPROM (Electronically Erasable Programmable Read Only Memory), and Flash ROM and how to integrate them to designing a digital waveform generator. We are using PSpice software to program ROM family devices. The PSpice manual provides little information about the procedures of ROM programming. The following steps will include programming the chip in PSpice, which is similar to programming the actual ROM devices.*

I. Introduction

ROMs are memory devices used to store data permanently [1]. ROMs hold a bit pattern for each distinct address applied to their inputs. In dedicated microcontroller applications, often used in equipments like oscilloscope and logic analyzer, ROMs are used extensively. ROMs are generally used for read-only operations and are not written to after they are initially programmed. EEPROMs store their bit as charges held on gates and hence can be erased and reprogrammed using either UV light or voltage [2].

In an EPROM we have a grid of columns and rows [3]. The cell at each intersection has two transistors. The two transistors are separated from each other by a thin oxide layer [4]. One of the transistors is known as the floating gate and the other as the control gate. The floating gate's only link to the row (wordline) is through the control gate [5]. As long as this link is in place, the cell has a value of 1. To change the value to 0 requires a curious process called Fowler-Nordheim tunneling. Tunneling is used to alter the placement of electrons in the floating gate [6]. An electrical charge, usually 10 to 13 volts, is applied to the floating gate. The charge comes from the column (bitline), enters the floating gate, and drains to a ground.

II. Programming Description

Task 1-Programming a ROM device using PSpice ROM map

This shows how to create a memory map to program a ROM in PSpice for any types of waveforms using a 32Kx8 ROM from the PSpice [7] Breakout library to create a digital waveform generator from 16 pages of waveform inputs and outputs.

Task 2-Programming a ROM device using Intel Hex File

Using the TTROM compiler, create a Hex file to download into a device programmer. Then, program a ROM chip.

Task 1

Using Excel, create a memory map for the application then calculate how many memory locations needed for the ROM. First the examination of input waveforms takes place and then, they need to be divided into 32 pieces to effectively recreate the original waveforms. This meant that we would be using 32X16 = 512 locations of ROM. Each one of the outputs is assigned to an input waveform as shown below in Figure 1.

Table 1 - Memory Map for 512 X 8 ROM

MEMORY MAP FOR 512 x 8 ROM

Figure #	**Memory Address (Hex)**	**Out 0**	**Out 1**	**Out 2**	**Out 3**	**Out 4**
8-9	000-01F	E1not	A0	A1	A2	N/A
8-16	020-03F	I7not	I8not	I9not	N/A	N/A
8-19	040-05F	Einot	I6not	I7not	N/A	N/A
10-6	060-07F	S	R	Q	N/A	N/A
10-10	080-09F	G	S	R	N/A	N/A
10-11	0A0-0BF	G	S	R	N/A	N/A
10-13	0C0-0DF	G	D	N/A	N/A	N/A
10-16	0E0-0FF	E0-1	D0	N/A	N/A	N/A
10-19	100-11F	Cp	Sdnot	Rdnot	D	N/A
10-21	120-13F	Cp	Sdnot	Rdnot	D	N/A
10-21 (continued)	140-15F	Cp	Sdnot	Rdnot	D	N/A
10-31	160-17F	Cpnot	J	K	N/A	N/A
10-34	180-19F	Cpnot	Sdnot	Rdnot	J	K
10-36	1A0-1BF	Cp	Sdnot	Rdnot	J	K
10-38	1C0-1DF	Cp	Sdnot	Rdnot	J	Knot
10-43	1E0-1FF	Trigger	Rdnot	Data	N/A	N/A

N/A = Not Applicable

*Outputs not utilized held low in the ROM programming

** Outputs 5,6,7 are not needed for this experiment.

Creating the schematic in PSpice

First select a ROM32X8break from the breakout library as shown below in Figure 1.

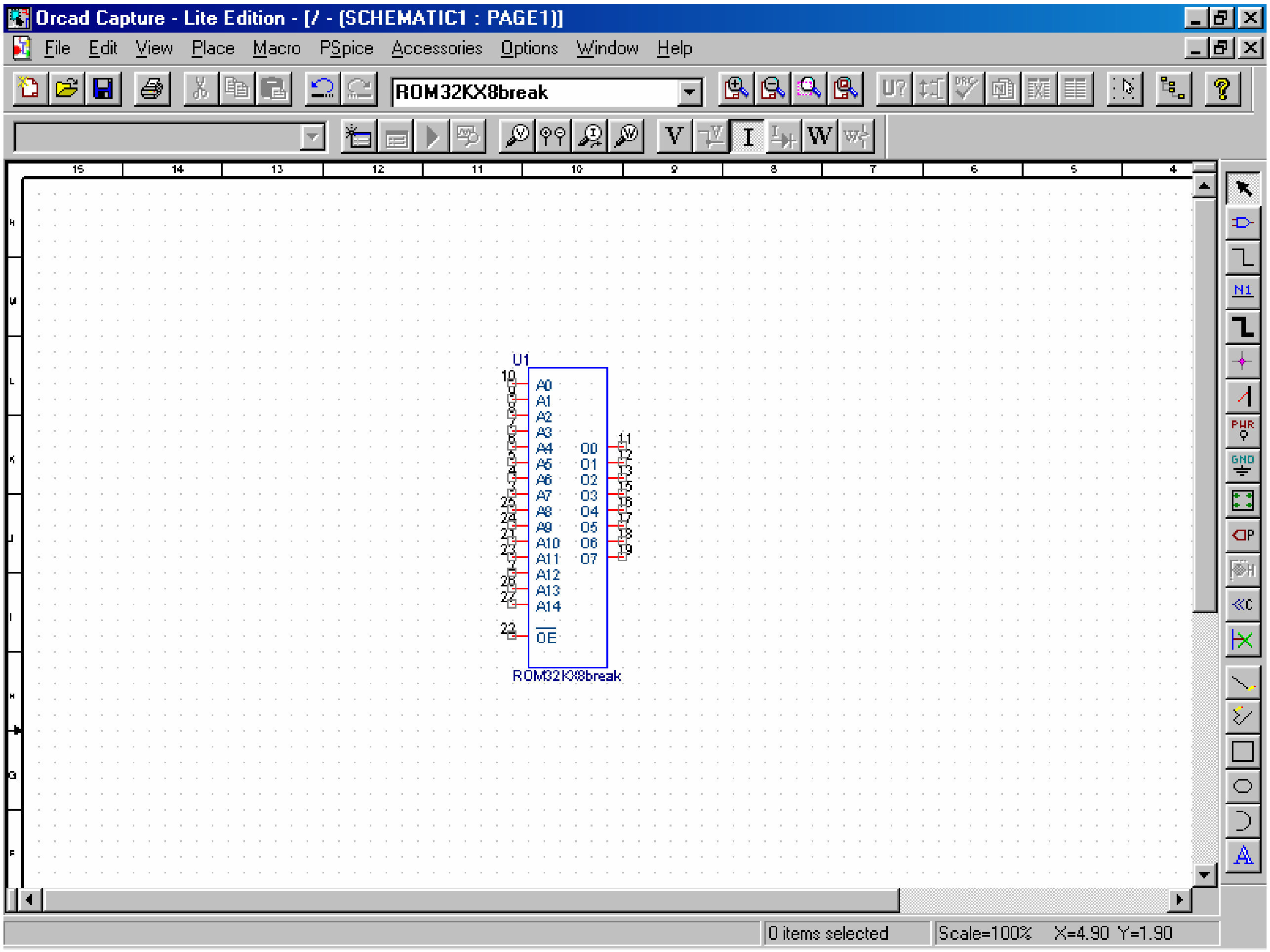

Orcad Capture - Lite Edition - [/ - (SCHEMATIC1 : PAGE1)]
File Edit View Place Macro PSpice Accessories Options Window Help
ROM32KX8break
U1
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
OE
O0 O1 O2 O3 O4 O5 O6 O7
ROM32KX8break
0 items selected
Scale=100%
X=4.90 Y=1.90

Figure 1 - ROM 32 by 8 from breakout library

Second, add the clock section to count up to 512 and the bus for the outputs as shown below in Figure 2.

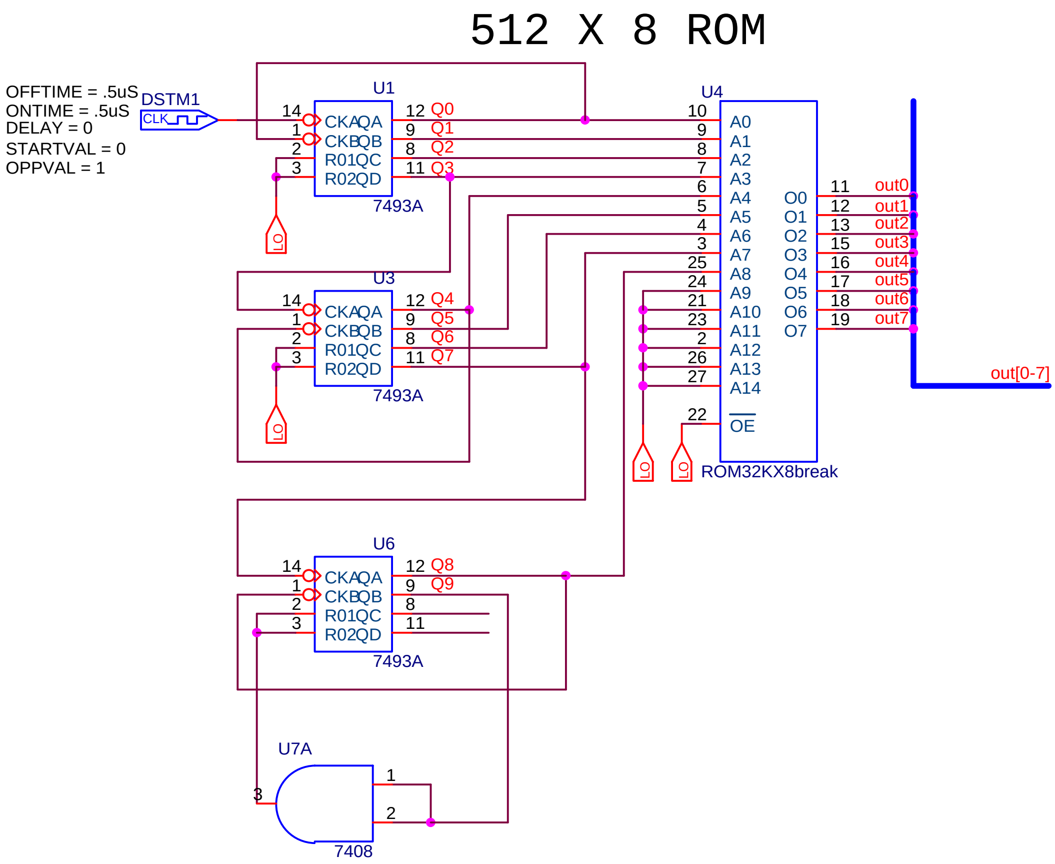


Figure 2 - Clock and 10 bit binary counter

Third, after creating the schematic, select the ROM chip and right click on it to bring up the "Edit PSpice Model" option, as shown below in Figure 3.

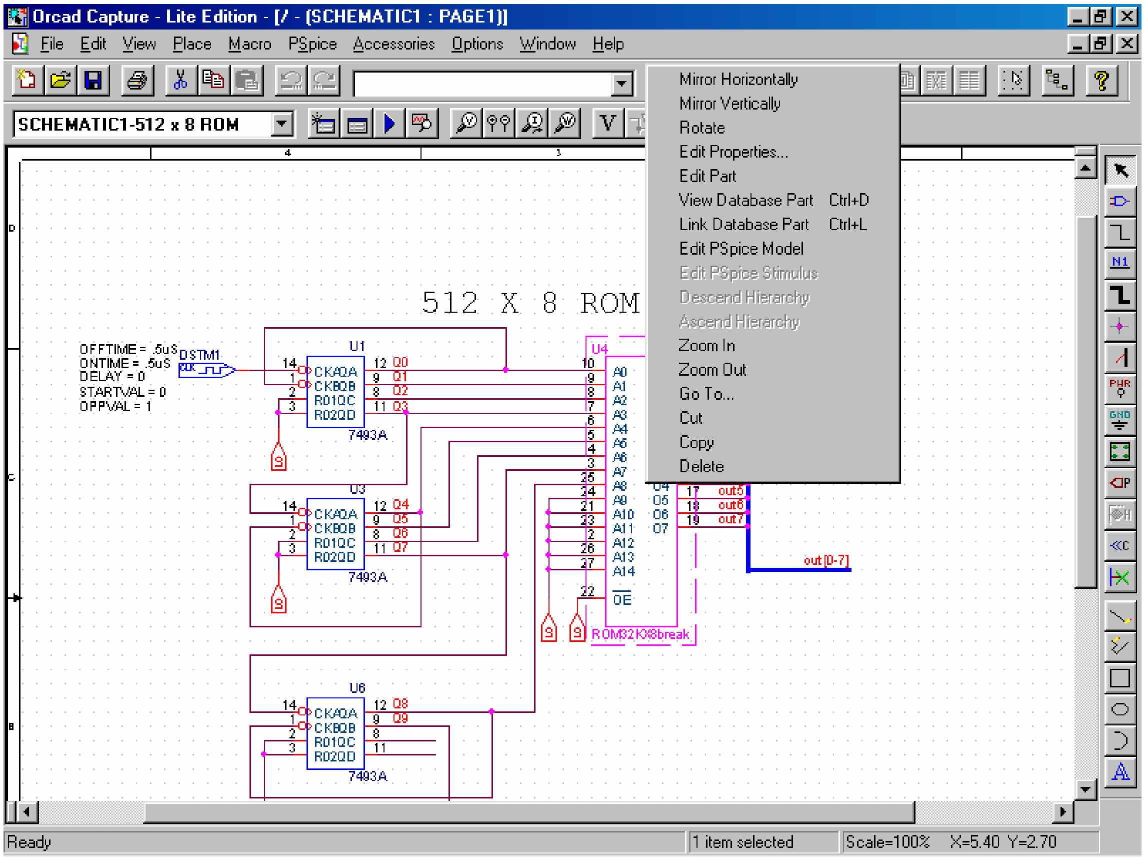
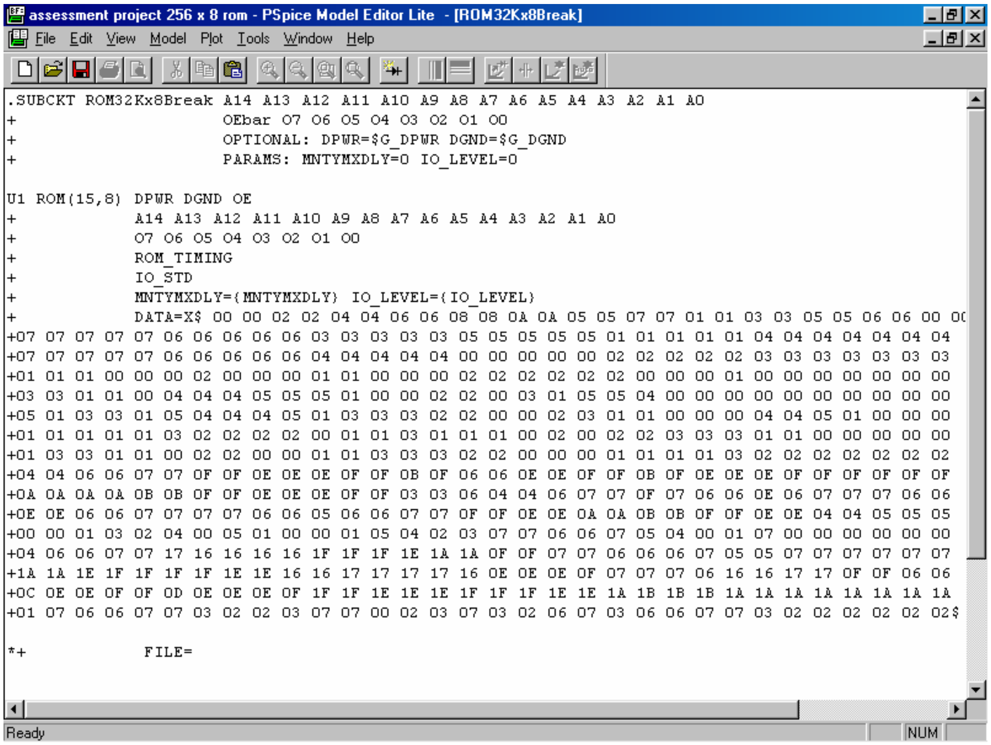


Figure 3 - Editing the ROM Model

Forth, for programming the ROM on the device line, we include the data on the DATA line as shown below in Figure 4.


assessment project 256 x 8 rom - PSpice Model Editor Lite - [ROM32Kx8Break]
File Edit View Model Plot Tools Window Help
Ready
NUM

```
.SUBCKT ROM32Kx8Break A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
+                     OEbar O7 O6 O5 O4 O3 O2 O1 O0
+                     OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+                     PARAMS: MNTYMXDLY=0 IO_LEVEL=0

U1 ROM(15,8) DPWR DGND OE
+             A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
+             O7 O6 O5 O4 O3 O2 O1 O0
+             ROM_TIMING
+             IO_STD
+             MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+             DATA=X$ 00 00 02 02 04 04 06 06 08 08 0A 0A 05 05 07 07 01 01 03 03 05 05 06 06 00 0
+07 07 07 07 07 06 06 06 06 06 03 03 03 03 03 05 05 05 05 05 01 01 01 01 01 04 04 04 04 04 04 04
+07 07 07 07 07 06 06 06 06 06 04 04 04 04 04 00 00 00 00 00 02 02 02 02 02 03 03 03 03 03 03 03
+01 01 01 00 00 00 02 00 00 00 01 01 00 00 00 02 02 02 02 02 02 00 00 00 01 00 00 00 00 00 00 00
+03 03 01 01 00 04 04 04 05 05 05 01 00 00 02 02 00 03 01 05 05 04 00 00 00 00 00 00 00 00 00 00
+05 01 03 03 01 05 04 04 04 05 01 03 03 03 02 02 00 00 02 03 01 01 00 00 00 04 04 05 01 00 00 00
+01 01 01 01 01 03 02 02 02 02 00 01 01 03 01 01 01 00 02 00 02 02 03 03 03 01 01 00 00 00 00 00
+01 03 03 01 01 00 02 02 00 00 01 01 03 03 03 02 02 00 00 00 01 01 01 01 03 02 02 02 02 02 02 02
+04 04 06 06 07 07 0F 0F 0E 0E 0E 0F 0F 0B 0F 06 06 0E 0E 0F 0F 0B 0F 0E 0E 0E 0F 0F 0F 0F 0F 0F
+0A 0A 0A 0A 0B 0B 0F 0F 0E 0E 0E 0F 0F 03 03 06 04 04 06 07 07 0F 07 06 06 0E 06 07 07 07 06 06
+0E 0E 06 06 07 07 07 07 06 06 05 06 06 07 07 0F 0F 0E 0E 0A 0A 0B 0B 0F 0F 0E 0E 04 04 05 05 05
+00 00 01 03 02 04 00 05 01 00 00 01 05 04 02 03 07 07 06 06 07 05 04 00 01 07 00 00 00 00 00 00
+04 06 06 07 07 17 16 16 16 16 1F 1F 1F 1E 1A 1A 0F 0F 07 07 06 06 06 07 05 05 07 07 07 07 07 07
+1A 1A 1E 1F 1F 1F 1F 1E 1E 16 16 17 17 17 17 16 0E 0E 0E 0F 07 07 07 06 16 16 17 17 0F 0F 06 06
+0C 0E 0E 0F 0F 0D 0E 0E 0E 0F 1F 1F 1E 1E 1E 1F 1F 1F 1E 1E 1A 1B 1B 1B 1A 1A 1A 1A 1A 1A 1A 1A
+01 07 06 06 07 07 03 02 02 03 07 07 00 02 03 07 03 02 06 07 03 06 06 07 07 03 02 02 02 02 02 02$

*+              FILE=
```

Figure 4 - Programming the ROM in Model Editor

A. First remove the * sign, or the entire line will be considered a comment.
B. Next, enter "X$” after the = sign. The X refers to Hex and the $ signs are used to tell PSpice that’s the beginning of the data.
C. Then put a $ at the end of the data as shown above to tell PSpice the data is done.
Fifth, run the simulation in PSpice and examine the data to verify it is correct as shown below in Figure 5.

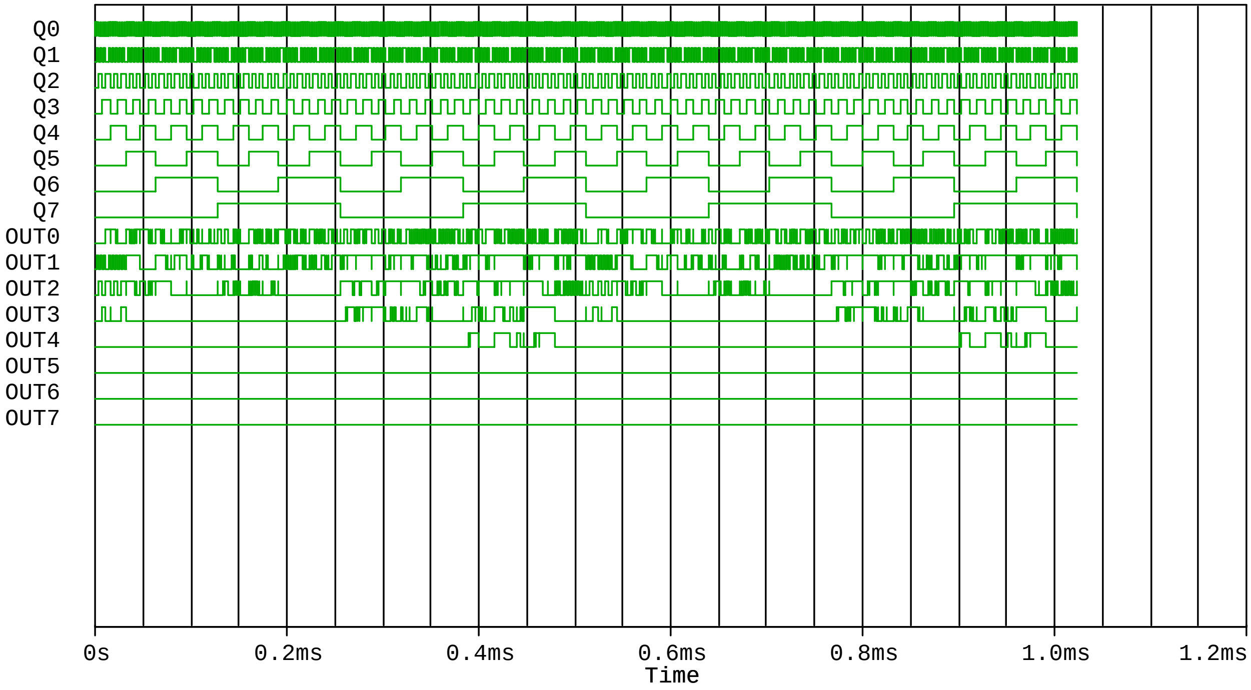
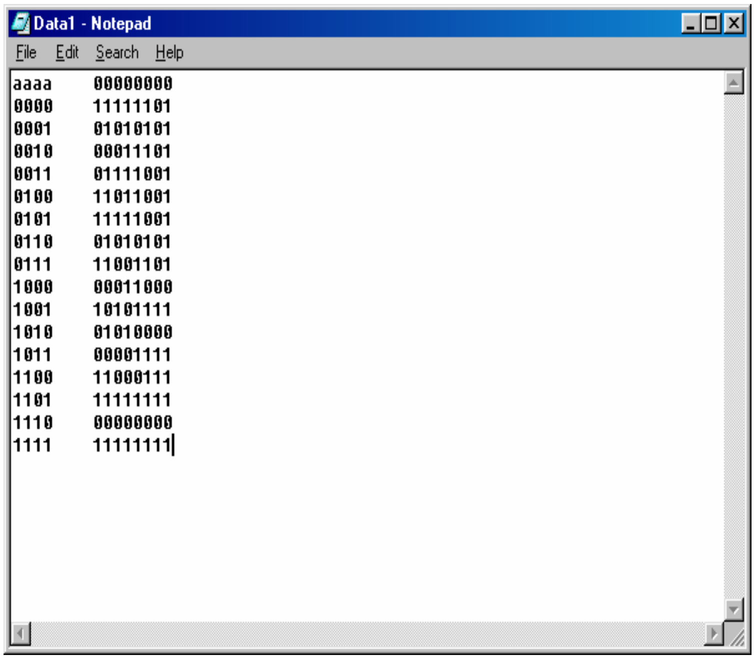


Figure 5 - The output file resulting from the binary counter

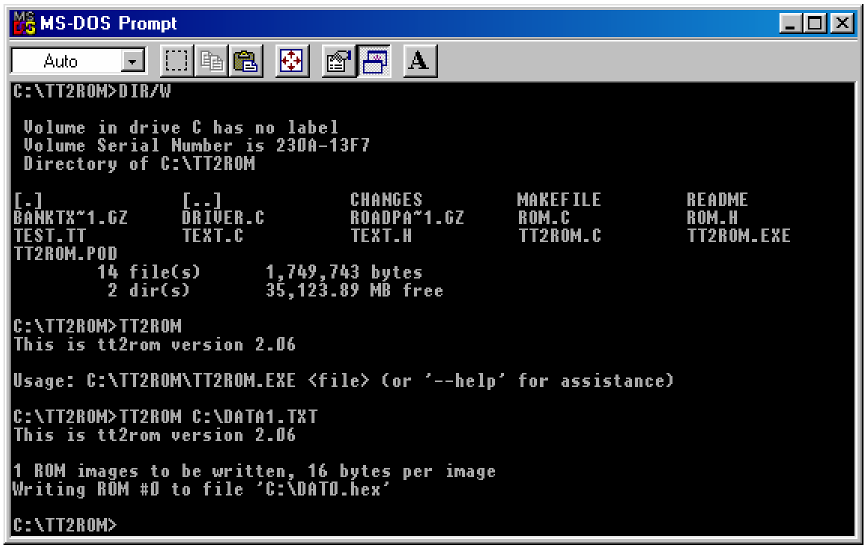
Creating an Intel Hex-File and Downloading Into a ROM Device
Using Notepad for small files, run the TTROM [8] complier and create a Hex file to download into a device programmer [9].
First copy or input the data into Notepad as shown below in Figure 6.



```
aaaa     00000000
0000     11111101
0001     01010101
0010     00011101
0011     01111001
0100     11011001
0101     11111001
0110     01010101
0111     11001101
1000     00011000
1001     10101111
1010     01010000
1011     00001111
1100     11000111
1101     11111111
1110     00000000
1111     11111111
```

Figure 6 - Creating the ROM table in notepad

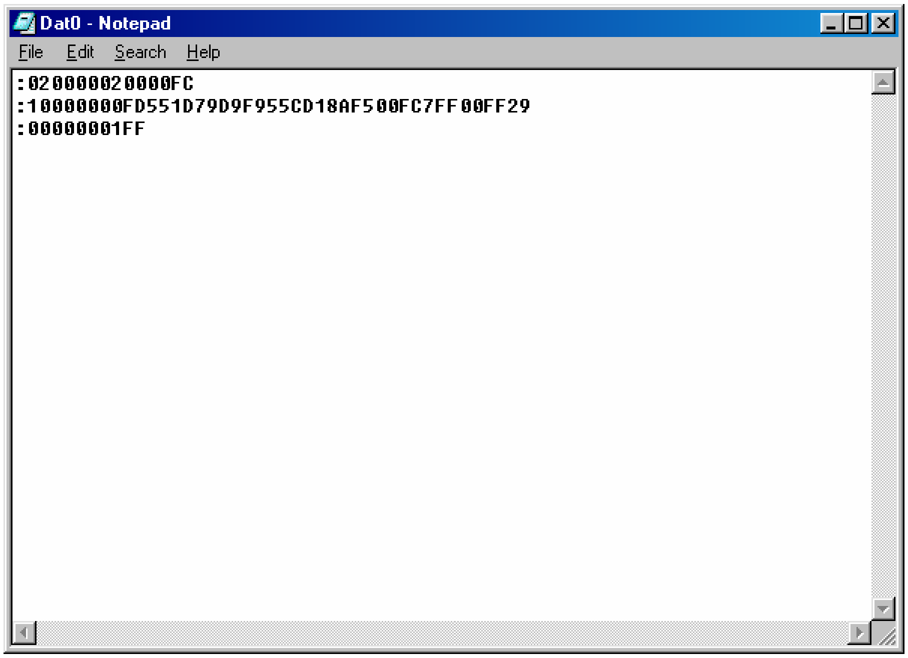
2. Run a DOS prompt window and go to the TTROM folder as shown below.
3. Enter the command: "tt2rom c:\ filename.txt"
4. The file has now been converted to Intel Hex format as shown below in Figure 7 [10].



```
C:\TT2ROM>DIR/W

 Volume in drive C has no label
 Volume Serial Number is 230A-13F7
 Directory of C:\TT2ROM

[.]             [..]            CHANGES         MAKEFILE        README
BANKTX~1.GZ     DRIVER.C        ROADPA~1.GZ     ROM.C           ROM.H
TEST.TT         TEXT.C          TEXT.H          TT2ROM.C        TT2ROM.EXE
TT2ROM.POD
        14 file(s)      1,749,743 bytes
         2 dir(s)       35,123.89 MB free

C:\TT2ROM>TT2ROM
This is tt2rom version 2.06

Usage: C:\TT2ROM\TT2ROM.EXE <file> (or '--help' for assistance)

C:\TT2ROM>TT2ROM C:\DATA1.TXT
This is tt2rom version 2.06

1 ROM images to be written, 16 bytes per image
Writing ROM #0 to file 'C:\DAT0.hex'

C:\TT2ROM>
```

Figure 7 - Running the TT2ROM program in DOS prompt

Note the actual Intel Hex file for the Digital Waveform Generator was not shown. This Figure is for reference, as shown below in Figure 8.



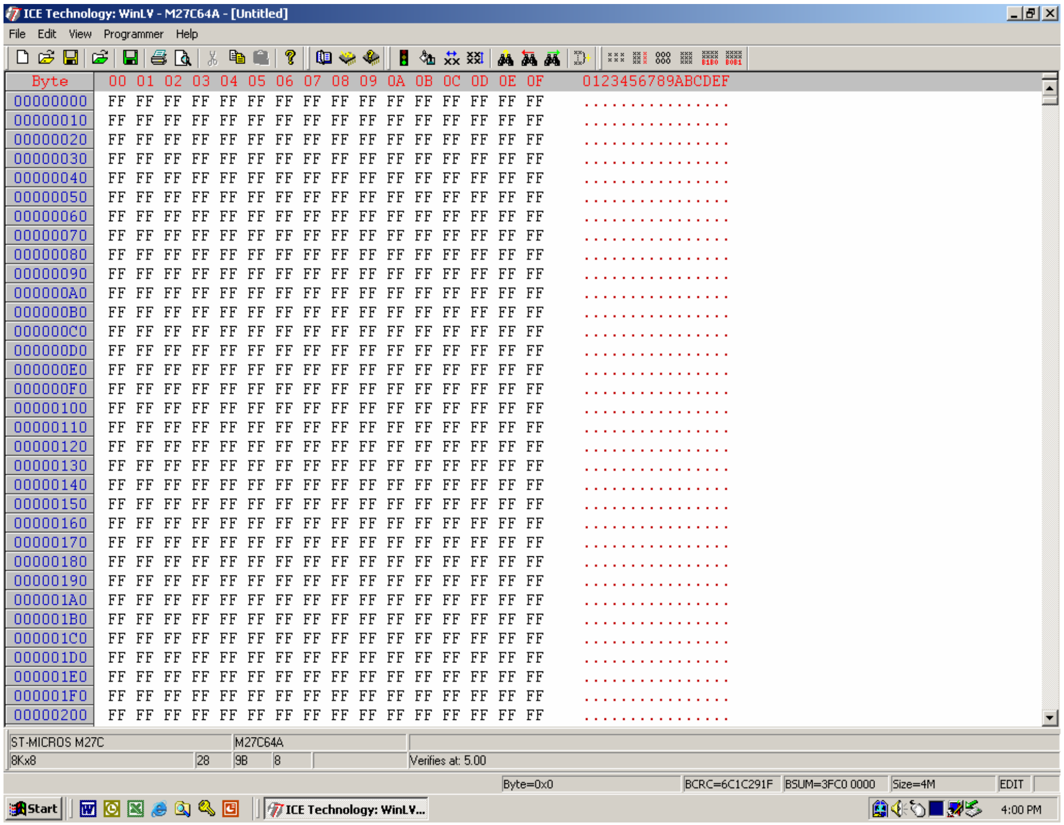
```
:020000020000FC
:10000000FD551D79D9F955CD18AF500FC7FF00FF29
:00000001FF
```

Figure 8 - Intel Hex File created by TT2ROM Program

Task 2

Now the Intel Hex file that you created can be read into the input buffer of programming software like ICROMMASTER- LV48 [5].

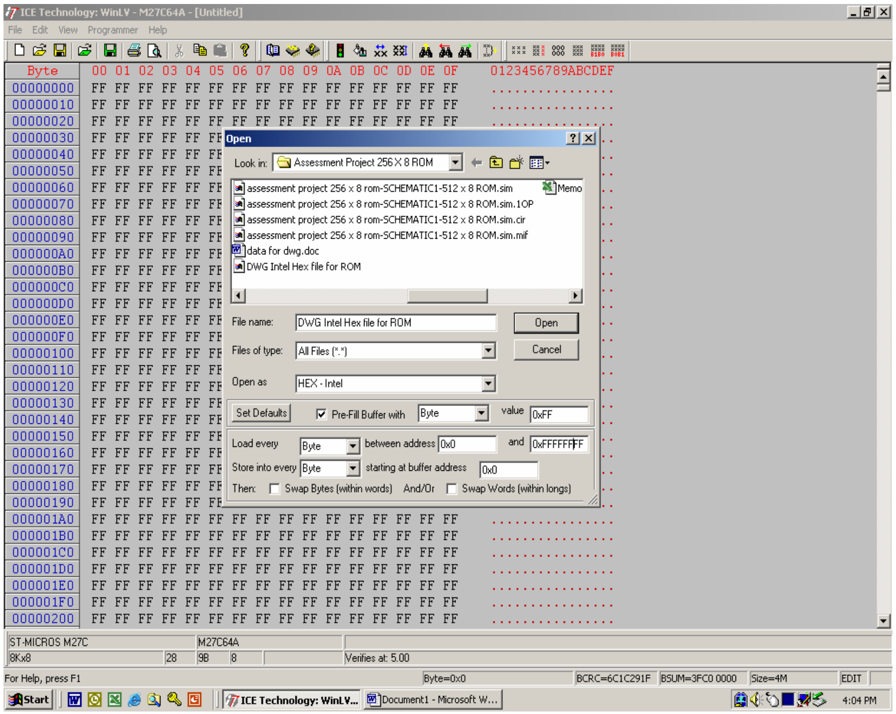
1. The type of programmer used for this program run was the ICROMMASTER- LV48
(A) More information can be found at http://www.icetech.com/
2. Open programmer by selecting the icon for WinLV as shown below in Figure 10.
2. Open Intel Hex file loading buffer with Hex codes, as shown in Figure 9 and10 [6].



```
  Byte    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000030  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000040  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000050  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000060  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000070  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000080  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000090  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000100  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000110  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000120  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000130  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000140  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000150  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000160  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000170  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000180  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000190  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000200  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

```
ST-MICROS M27C              M27C64A
8Kx8              28   9B   8           Verifies at: 5.00
                                  Byte=0x0   BCRC=6C1C291F   BSUM=3FC0 0000   Size=4M   EDIT
```

Figure 9 - EEPROM Programmer


ICE Technology: WinLV - M27C64A - [Untitled]
File Edit View Programmer Help

```
Byte      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F    0123456789ABCDEF
00000000  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
00000010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
00000020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
00000030  FF FF FF FF FF FF
00000040  FF FF FF FF FF FF
00000050  FF FF FF FF FF FF
00000060  FF FF FF FF FF FF
00000070  FF FF FF FF FF FF
00000080  FF FF FF FF FF FF
00000090  FF FF FF FF FF FF
000000A0  FF FF FF FF FF FF
000000B0  FF FF FF FF FF FF
000000C0  FF FF FF FF FF FF
000000D0  FF FF FF FF FF FF
000000E0  FF FF FF FF FF FF
000000F0  FF FF FF FF FF FF
00000100  FF FF FF FF FF FF
00000110  FF FF FF FF FF FF
00000120  FF FF FF FF FF FF
00000130  FF FF FF FF FF FF
00000140  FF FF FF FF FF FF
00000150  FF FF FF FF FF FF
00000160  FF FF FF FF FF FF
00000170  FF FF FF FF FF FF
00000180  FF FF FF FF FF FF
00000190  FF FF FF FF FF FF
000001A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
000001B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
000001C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
000001D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
000001E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
000001F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
00000200  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
```

Open

Look in: Assessment Project 256 X 8 ROM

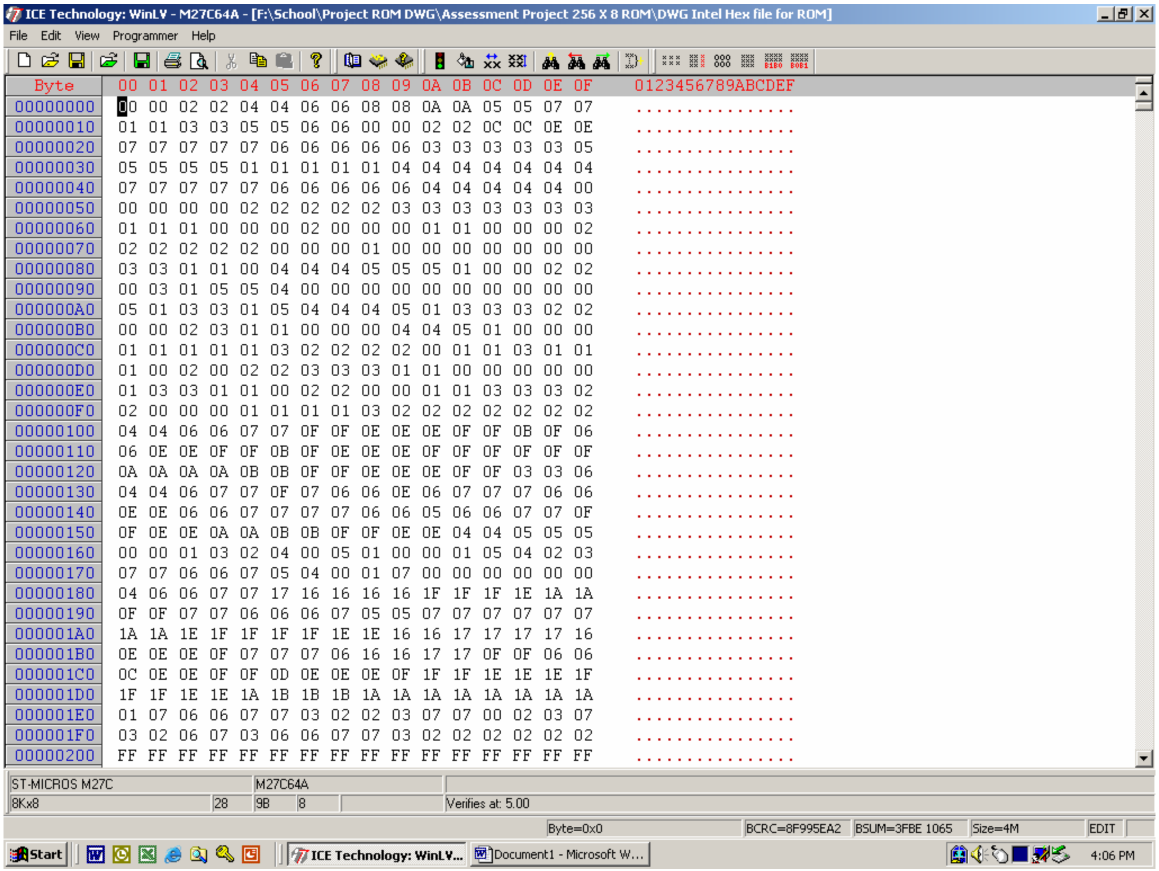
- assessment project 256 x 8 rom-SCHEMATIC1-512 x 8 ROM.sim
- assessment project 256 x 8 rom-SCHEMATIC1-512 x 8 ROM.sim.1OP
- assessment project 256 x 8 rom-SCHEMATIC1-512 x 8 ROM.sim.cir
- assessment project 256 x 8 rom-SCHEMATIC1-512 x 8 ROM.sim.mif
- data for dwg.doc
- DWG Intel Hex file for ROM
- Memo

File name: DWG Intel Hex file for ROM
Files of type: All Files (*.*)
Open as: HEX - Intel
☑ Pre-Fill Buffer with: Byte value: 0xFF
Load every: Byte between address: 0x0 and: 0xFFFFFFFF
Store into every: Byte starting at buffer address: 0x0
Then: □ Swap Bytes (within words) And/Or □ Swap Words (within longs)

```
ST-MICROS M27C    M27C64A
8Kx8    28    9B    8    Verifies at: 5.00
For Help, press F1    Byte=0x0    BCRC=6C1C291F    BSUM=3FC0 0000    Size=4M    EDIT
```

Figure 10- Uploading file to EEPOM Programmer

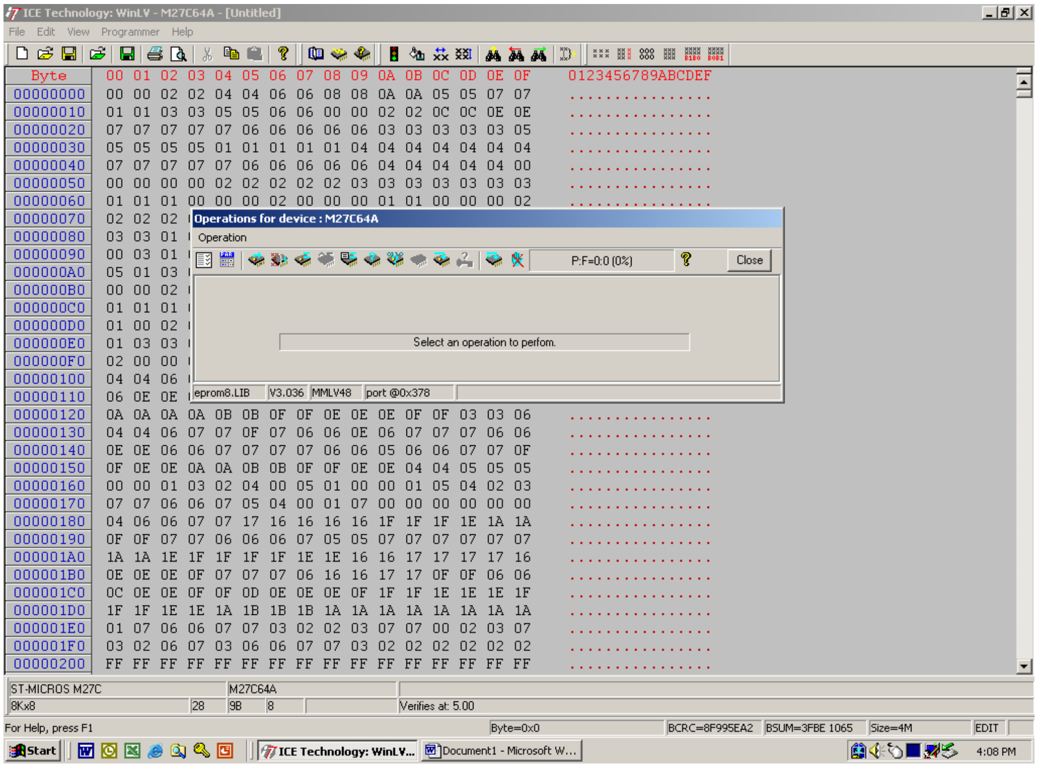
3. The data buffer has the required information for our Digital Waveform Generator ROM chip in the program buffer starting at location 0000 Hex, as shown in Figure 11.


ICE Technology: WinLV - M27C64A - [F:\School\Project ROM DWG\Assessment Project 256 X 8 ROM\DWG Intel Hex file for ROM]
File
Edit
View
Programmer
Help

```
Byte      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F    0123456789ABCDEF
00000000  00 00 02 02 04 04 06 06 08 08 0A 0A 05 05 07 07    ................
00000010  01 01 03 03 05 05 06 06 00 00 02 02 0C 0C 0E 0E    ................
00000020  07 07 07 07 07 06 06 06 06 06 03 03 03 03 03 05    ................
00000030  05 05 05 05 01 01 01 01 01 04 04 04 04 04 04 04    ................
00000040  07 07 07 07 07 06 06 06 06 06 04 04 04 04 04 00    ................
00000050  00 00 00 00 02 02 02 02 02 03 03 03 03 03 03 03    ................
00000060  01 01 01 00 00 00 02 00 00 00 01 01 00 00 00 02    ................
00000070  02 02 02 02 02 00 00 00 01 00 00 00 00 00 00 00    ................
00000080  03 03 01 01 00 04 04 04 05 05 05 01 00 00 02 02    ................
00000090  00 03 01 05 05 04 00 00 00 00 00 00 00 00 00 00    ................
000000A0  05 01 03 03 01 05 04 04 04 05 01 03 03 03 02 02    ................
000000B0  00 00 02 03 01 01 00 00 00 04 04 05 01 00 00 00    ................
000000C0  01 01 01 01 01 03 02 02 02 02 00 01 01 03 01 01    ................
000000D0  01 00 02 00 02 02 03 03 03 01 01 00 00 00 00 00    ................
000000E0  01 03 03 01 01 00 02 02 00 00 01 01 03 03 03 02    ................
000000F0  02 00 00 00 01 01 01 01 03 02 02 02 02 02 02 02    ................
00000100  04 04 06 06 07 07 0F 0F 0E 0E 0E 0F 0F 0B 0F 06    ................
00000110  06 0E 0E 0F 0F 0B 0F 0E 0E 0E 0F 0F 0F 0F 0F 0F    ................
00000120  0A 0A 0A 0A 0B 0B 0F 0F 0E 0E 0E 0F 0F 03 03 06    ................
00000130  04 04 06 07 07 0F 07 06 06 0E 06 07 07 07 06 06    ................
00000140  0E 0E 06 06 07 07 07 07 06 06 05 06 06 07 07 0F    ................
00000150  0F 0E 0E 0A 0A 0B 0B 0F 0F 0E 0E 04 04 05 05 05    ................
00000160  00 00 01 03 02 04 00 05 01 00 00 01 05 04 02 03    ................
00000170  07 07 06 06 07 05 04 00 01 07 00 00 00 00 00 00    ................
00000180  04 06 06 07 07 17 16 16 16 16 1F 1F 1F 1E 1A 1A    ................
00000190  0F 0F 07 07 06 06 06 07 05 05 07 07 07 07 07 07    ................
000001A0  1A 1A 1E 1F 1F 1F 1F 1E 1E 16 16 17 17 17 17 16    ................
000001B0  0E 0E 0E 0F 07 07 07 06 16 16 17 17 0F 0F 06 06    ................
000001C0  0C 0E 0E 0F 0F 0D 0E 0E 0E 0F 1F 1F 1E 1E 1E 1F    ................
000001D0  1F 1F 1E 1E 1A 1B 1B 1B 1A 1A 1A 1A 1A 1A 1A 1A    ................
000001E0  01 07 06 06 07 07 03 02 02 03 07 07 00 02 03 07    ................
000001F0  03 02 06 07 03 06 06 07 07 03 02 02 02 02 02 02    ................
00000200  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................

ST-MICROS M27C          M27C64A
8Kx8            28      9B   8            Verifies at: 5.00

Byte=0x0    BCRC=8F995EA2    BSUM=3FBE 1065    Size=4M    EDIT
```

Figure 11 - Uploaded HEX codes

Next, select operations icon displaying several selections, as shown in Figure 12.



ICE Technology: WinLV - M27C64A - [Untitled]

```
Byte      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F    0123456789ABCDEF
00000000  00 00 02 02 04 04 06 06 08 08 0A 0A 05 05 07 07    ................
00000010  01 01 03 03 05 05 06 06 00 00 02 02 0C 0C 0E 0E    ................
00000020  07 07 07 07 07 06 06 06 06 06 03 03 03 03 03 05    ................
00000030  05 05 05 05 01 01 01 01 01 04 04 04 04 04 04 04    ................
00000040  07 07 07 07 07 06 06 06 06 06 04 04 04 04 04 00    ................
00000050  00 00 00 00 02 02 02 02 02 03 03 03 03 03 03 03    ................
00000060  01 01 01 00 00 00 02 00 00 00 01 01 00 00 00 02    ................
00000070  02 02 02
00000080  03 03 01
00000090  00 03 01
000000A0  05 01 03
000000B0  00 00 02
000000C0  01 01 01
000000D0  01 00 02
000000E0  01 03 03
000000F0  02 00 00
00000100  04 04 06
00000110  06 0E 0E
00000120  0A 0A 0A 0A 0B 0B 0F 0F 0E 0E 0E 0F 0F 03 03 06    ................
00000130  04 04 06 07 07 0F 07 06 06 0E 06 07 07 07 06 06    ................
00000140  0E 0E 06 06 07 07 07 07 06 06 05 06 06 07 07 0F    ................
00000150  0F 0E 0E 0A 0A 0B 0B 0F 0F 0E 0E 04 04 05 05 05    ................
00000160  00 00 01 03 02 04 00 05 01 00 00 01 05 04 02 03    ................
00000170  07 07 06 06 07 05 04 00 01 07 00 00 00 00 00 00    ................
00000180  04 06 06 07 07 17 16 16 16 16 1F 1F 1F 1E 1A 1A    ................
00000190  0F 0F 07 07 06 06 06 07 05 05 07 07 07 07 07 07    ................
000001A0  1A 1A 1E 1F 1F 1F 1F 1E 1E 16 16 17 17 17 17 16    ................
000001B0  0E 0E 0E 0F 07 07 07 06 16 16 17 17 0F 0F 06 06    ................
000001C0  0C 0E 0E 0F 0F 0D 0E 0E 0E 0F 1F 1F 1E 1E 1E 1F    ................
000001D0  1F 1F 1E 1E 1A 1B 1B 1B 1A 1A 1A 1A 1A 1A 1A 1A    ................
000001E0  01 07 06 06 07 07 03 02 02 03 07 07 00 02 03 07    ................
000001F0  03 02 06 07 03 06 06 07 07 03 02 02 02 02 02 02    ................
00000200  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    ................
```

Operations for device : M27C64A

P:F=0:0 (0%)

Select an operation to perfom.

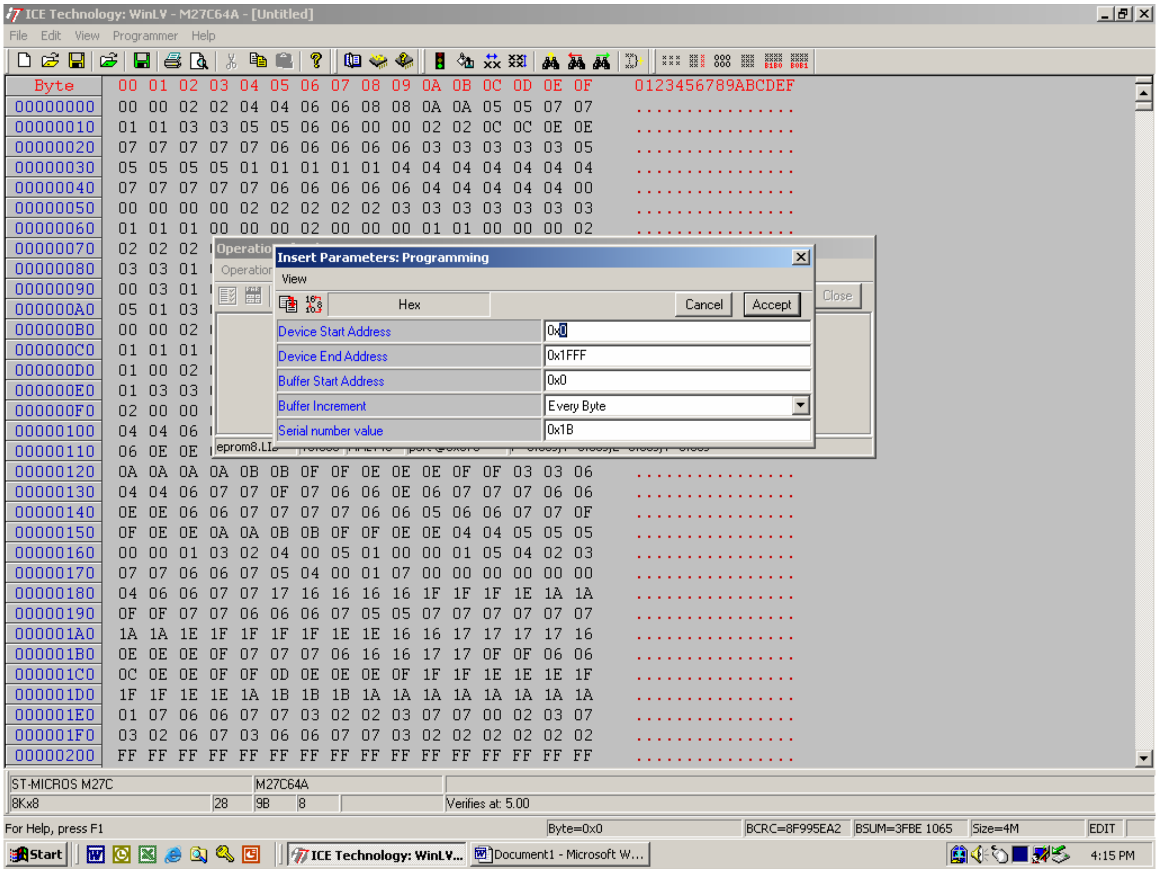
eprom8.LIB | V3.036 | MMLV48 | port @0x378

ST-MICROS M27C | M27C64A
8Kx8 | 28 | 9B | 8 | Verifies at: 5.00

For Help, press F1 | Byte=0x0 | BCRC=8F995EA2 | BSUM=3FBE 1065 | Size=4M | EDIT

Figure 12 - Operation selection

With parameters menu changes can be made to locations for data storage into ROM but in this case no changes are necessary so select accept, as shown below in Figure 13.


ICE Technology: WinLV - M27C64A - [Untitled]
File Edit View Programmer Help

```
Byte      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000  00 00 02 02 04 04 06 06 08 08 0A 0A 05 05 07 07   ................
00000010  01 01 03 03 05 05 06 06 00 00 02 02 0C 0C 0E 0E   ................
00000020  07 07 07 07 07 06 06 06 06 06 03 03 03 03 03 05   ................
00000030  05 05 05 05 01 01 01 01 01 04 04 04 04 04 04 04   ................
00000040  07 07 07 07 07 06 06 06 06 06 04 04 04 04 04 00   ................
00000050  00 00 00 00 02 02 02 02 02 03 03 03 03 03 03 03   ................
00000060  01 01 01 00 00 00 02 00 00 00 01 01 00 00 00 02   ................
00000070  02 02 02
00000080  03 03 01
00000090  00 03 01
000000A0  05 01 03
000000B0  00 00 02
000000C0  01 01 01
000000D0  01 00 02
000000E0  01 03 03
000000F0  02 00 00
00000100  04 04 06
00000110  06 0E 0E
00000120  0A 0A 0A 0A 0B 0B 0F 0F 0E 0E 0E 0F 0F 03 03 06   ................
00000130  04 04 06 07 07 0F 07 06 06 0E 06 07 07 07 06 06   ................
00000140  0E 0E 06 06 07 07 07 07 06 06 05 06 06 07 07 0F   ................
00000150  0F 0E 0E 0A 0A 0B 0B 0F 0F 0E 0E 04 04 05 05 05   ................
00000160  00 00 01 03 02 04 00 05 01 00 00 01 05 04 02 03   ................
00000170  07 07 06 06 07 05 04 00 01 07 00 00 00 00 00 00   ................
00000180  04 06 06 07 07 17 16 16 16 16 1F 1F 1F 1E 1A 1A   ................
00000190  0F 0F 07 07 06 06 06 07 05 05 07 07 07 07 07 07   ................
000001A0  1A 1A 1E 1F 1F 1F 1F 1E 1E 16 16 17 17 17 17 16   ................
000001B0  0E 0E 0E 0F 07 07 07 06 16 16 17 17 0F 0F 06 06   ................
000001C0  0C 0E 0E 0F 0F 0D 0E 0E 0E 0F 1F 1F 1E 1E 1E 1F   ................
000001D0  1F 1F 1E 1E 1A 1B 1B 1B 1A 1A 1A 1A 1A 1A 1A 1A   ................
000001E0  01 07 06 06 07 07 03 02 02 03 07 07 00 02 03 07   ................
000001F0  03 02 06 07 03 06 06 07 07 03 02 02 02 02 02 02   ................
00000200  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

Insert Parameters: Programming

Device Start Address	0x0
Device End Address	0x1FFF
Buffer Start Address	0x0
Buffer Increment	Every Byte
Serial number value	0x1B

ST-MICROS M27C | M27C64A

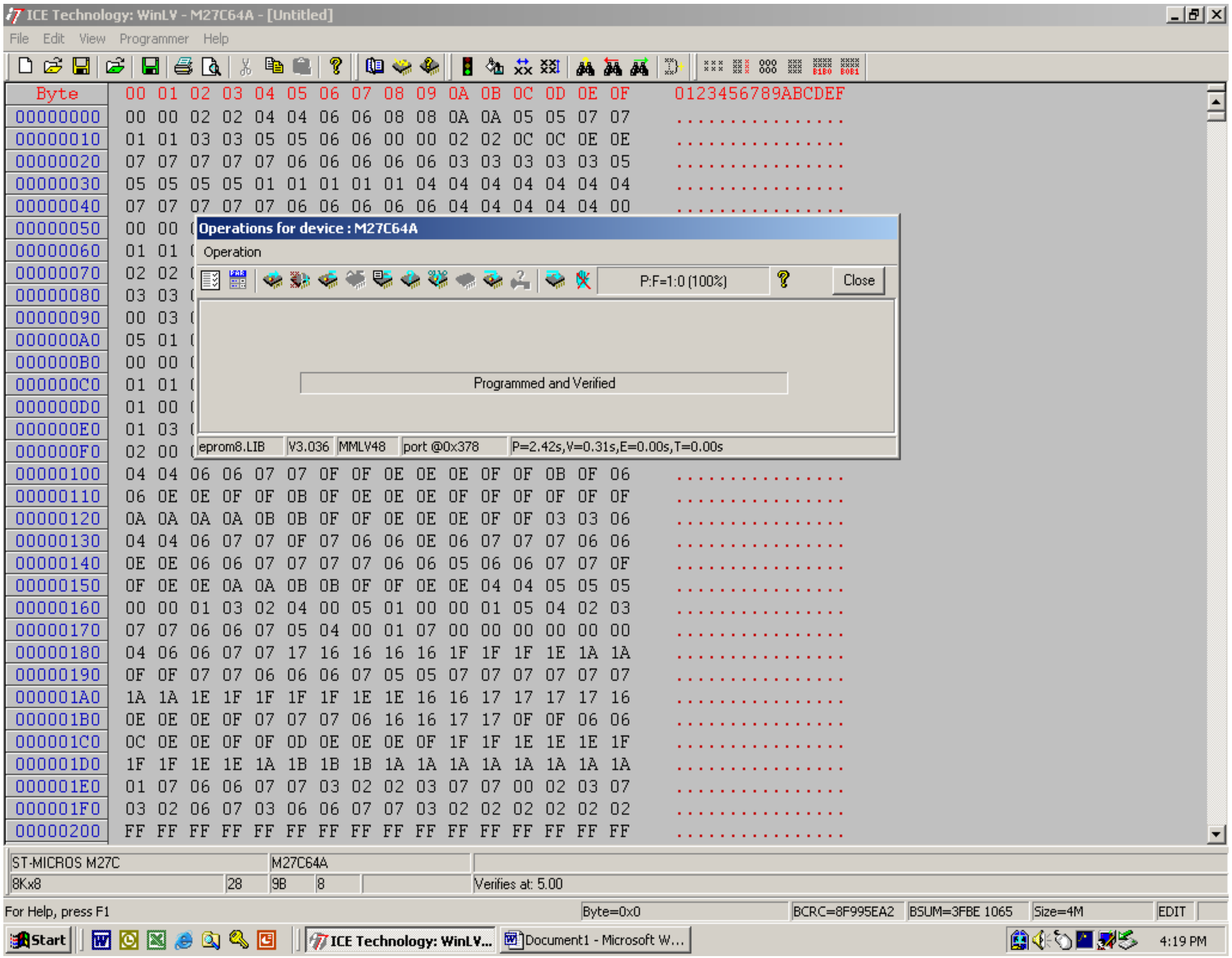
8Kx8 | 28 | 9B | 8 | Verifies at: 5.00

For Help, press F1 | Byte=0x0 | BCRC=8F995EA2 | BSUM=3FBE 1065 | Size=4M | EDIT

Figure 13 - Device selection

The data has now been loaded into the ROM.

7. At the end of the operation the program data has been programmed into the ROM and it should also report that the data was verified as shown below in Figure 14.


ICE Technology: WinLV - M27C64A - [Untitled]
File Edit View Programmer Help

```
Byte      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000  00 00 02 02 04 04 06 06 08 08 0A 0A 05 05 07 07   ................
00000010  01 01 03 03 05 05 06 06 00 00 02 02 0C 0C 0E 0E   ................
00000020  07 07 07 07 07 06 06 06 06 06 03 03 03 03 03 05   ................
00000030  05 05 05 05 01 01 01 01 01 04 04 04 04 04 04 04   ................
00000040  07 07 07 07 07 06 06 06 06 06 04 04 04 04 04 00   ................
00000050  00 00
00000060  01 01
00000070  02 02
00000080  03 03
00000090  00 03
000000A0  05 01
000000B0  00 00
000000C0  01 01
000000D0  01 00
000000E0  01 03
000000F0  02 00
00000100  04 04 06 06 07 07 0F 0F 0E 0E 0E 0F 0F 0B 0F 06   ................
00000110  06 0E 0E 0F 0F 0B 0F 0E 0E 0E 0F 0F 0F 0F 0F 0F   ................
00000120  0A 0A 0A 0A 0B 0B 0F 0F 0E 0E 0E 0F 0F 03 03 06   ................
00000130  04 04 06 07 07 0F 07 06 06 0E 06 07 07 07 06 06   ................
00000140  0E 0E 06 06 07 07 07 07 06 06 05 06 06 07 07 0F   ................
00000150  0F 0E 0E 0A 0A 0B 0B 0F 0F 0E 0E 04 04 05 05 05   ................
00000160  00 00 01 03 02 04 00 05 01 00 00 01 05 04 02 03   ................
00000170  07 07 06 06 07 05 04 00 01 07 00 00 00 00 00 00   ................
00000180  04 06 06 07 07 17 16 16 16 16 1F 1F 1F 1E 1A 1A   ................
00000190  0F 0F 07 07 06 06 06 07 05 05 07 07 07 07 07 07   ................
000001A0  1A 1A 1E 1F 1F 1F 1F 1E 1E 16 16 17 17 17 17 16   ................
000001B0  0E 0E 0E 0F 07 07 07 06 16 16 17 17 0F 0F 06 06   ................
000001C0  0C 0E 0E 0F 0F 0D 0E 0E 0E 0F 1F 1F 1E 1E 1E 1F   ................
000001D0  1F 1F 1E 1E 1A 1B 1B 1B 1A 1A 1A 1A 1A 1A 1A 1A   ................
000001E0  01 07 06 06 07 07 03 02 02 03 07 07 00 02 03 07   ................
000001F0  03 02 06 07 03 06 06 07 07 03 02 02 02 02 02 02   ................
00000200  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

Operations for device : M27C64A

Operation

P:F=1:0 (100%)

Close

Programmed and Verified

eprom8.LIB | V3.036 | MMLV48 | port @0x378 | P=2.42s,V=0.31s,E=0.00s,T=0.00s

ST-MICROS M27C | M27C64A

8Kx8 | 28 | 9B | 8 | Verifies at: 5.00

For Help, press F1 | Byte=0x0 | BCRC=8F995EA2 | BSUM=3FBE 1065 | Size=4M | EDIT

Figure14 - Data verification

Conclusion

Teaching students real industrial hardware/software has been a challenging one, the step by step illustration of usage of these hardware/software with proper screenshots improves students learning. There has been good student feedback about usage of software simulation for digital courses. PSpice now knows as Cadence SPB, is leading electronic automation design software, a student version can be obtain form the company’s website. The company also has a “university program” that allows educational institutions to buy full version of this software at extremely low price.

References

[1] "How ROM Works" http://computer.howstuffworks.com/rom.htm, 1998-2005 How Stuff Work .Inc. 2.

[2] "Microprocessor", EDN, Vol 49 Issue 25, p110, 2p 12/7/2004

[3] Kleitz, W., "Digital Electronics with VHDL", Chapter. 16 Semiconductors, Magnetic, and Optical Memory, Prentice Hall 2005.

[4] Manes, S., "The Weird New World of Hardware Fixes", PC World, Vol. 23 Issue pg 176, 1p, 1c.

[5] Schweber, B., "Mems Make a Nonvolatile-Memory Match", EDN, Vol.49 Issue 16 pg 20, 1/4p, 8/5/2004

[6] Tocci, R, N. Widmer, and G. L. Moss, "Digital Systems: Principles and Applications, 9/E, Chpt. 12 Memory Devices, Prentice Hall 2003

[7] "Orcad Capture", http://www.orcad.com/orcadcapture.aspx , Cadence Design Systems, Inc. 2004

[8] "Using tt2rom to Synthesize Truth Table in ROM". http://engineering.dartmouth.edu/~Brian_W_Pogue/demos/tt2rom.pdf , Spring 2002

[9] Ameen, M., Brown, R., Rashik R, Sundaran S., Zeeshan, M. "Programming Read Only Memory with Orcad", unpublished manual.

[10] Cook, N., "Electronics: A Complete Course, 2/E", Chapter. 27 Semiconductor Memory, Prentice Hall 2004.