

Experiment No. 7 & 8

DESIGN AND TESTING OF A SEVEN SEGMENT DECODER

Randy Dimmett
Department of Electrical Engineering
University of Missouri-Rolla
dimmett@umr.edu

EE112e
Randy Dimmett
Azeeta Abdullah
April 27th, 1998

Abstract:

Experiment No. 7 & 8: Design and Testing of a Seven-Segment Decoder

Author: Randy Dimmett

Date: April 27th, 1998

Affiliation: Department of Electrical Engineering, University of Missouri-Rolla

Contact: dimmett@umr.edu

In this lab, a seven-segment decoder was designed and simulated on a computer using Mentor's graphics software package. After its design was found to be working properly, it was constructed and tested. This seven segment decoder could display all hexadecimal characters (0-9 and A-F) on a seven segment display. This design is widely used in electronic devices such as digital clocks, digital calculators, and cell phones.

Table of Contents:

	page
Introduction	1
Background and Theory	1
Procedure	3
Results and Analysis	4
Conclusion	5
Appendix I – Equipment List	6
Appendix II – Seven Segment Display and Truth Table	7
Appendix III – Karnaugh Maps and Expressions	8
Appendix IV – Schematics	10
Appendix V – Verification Data	11
Appendix VI – Breadboard Connections	12
References	13

Introduction:

The seven-segment decoder designed and built in this lab simply uses a 4-bit input to display the characters 1-9 and a-f on a seven-segment display. The design was done on Mentor's design software and simulated to make sure the schematic derived using Karnaugh maps was correct. It was then built in lab and tested.

Background and Theory:

Most electronic devices begin on the basic level that was started on in this lab. That is, the designing using truth tables and Karnaugh maps. See Appendix II for an example of a truth table, and Appendix III for the corresponding Karnaugh maps.

Note that these are just representations in binary of what needs to be designed. For the seven segment decoder being designed and built in this lab, the basis for the given binary numbers is that the seven segment display in figure 1 is active low. This means that when the input to each segment is low (zero), that segment will light up. (For specifications of the display see Appendix II).

The display used in this lab (that in figure 1) just uses seven binary inputs that control which of the seven different segments (a through g) will light up. The decoder that will be designed and implemented will take the four bit input and convert it to the proper seven binary values that will give a display that corresponds to the four bit input. The decoder would cause the binary input 0000 to display a zero, 0001 would display a 1, and so on, all the way to 1001 for a nine. After nine, the characters a through f would then be displayed, thus using all possible combinations of a four bit binary input at 1111 (there are sixteen possible combinations). This decoder could really just be thought of as a hexadecimal display.

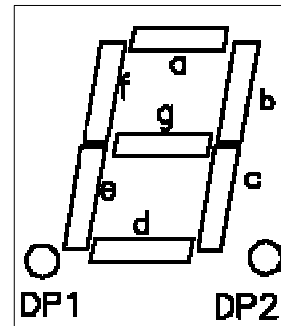


Figure 1

1234567890A b C d E F

Table 1

As discussed above, this design should take the four inputs and display the hexadecimal characters in the format seen in table 1.

After the Karnaugh maps are constructed, reduction can begin. This is a process that eliminates unnecessary expressions that would be derived from the maps (causing the use of more gates than needed in the hardware construction). See Appendix III for minimized expressions.

After the minimum expressions are derived from the Karnaugh maps, the actual design can begin. For the decoder built in this lab, only AND gates and OR gates were used (see Appendix IV).

This design can then be put on the computer and tested using Mentor's Graphics software package. Design Architect can be used for the computer design and Quicksim for the simulation and testing of the design. By doing this, it can be determined whether or not the correct design was actually created by checking the outputs of the simulation (see Appendix V for verifications).

If the design is working correctly, it can be implemented with hardware and tested in the lab using a Xilinx Field Programmable Gate Array (FPGA). The FPGA is reprogrammable for different schematics, and has to be loaded with data that configures it to a specific design, like a seven-segment decoder. Xilinx provides software for producing the data (a bit file) from a Mentor schematic.

Procedure:

Lab 7:

A truth table was constructed for the seven-segment decoder and placed in Appendix II. The truth table consisted of four inputs and seven outputs (one for each segment). Karnaugh maps were then constructed from the table and placed in Appendix III. Next, the minimized sum of products expressions were derived from these maps for each segment of the display (there were seven) and placed in Appendix III.

The sum-of-product expressions were then used to design a schematic by hand for the seven-segment decoder. Design Architect, a program in the Mentor Graphics package, was then used to design the schematic on the computer. The inputs were labeled 1, 2, 4, and 8. The outputs were labeled a through g, corresponding to the segments of the display. All components used were found in the XC4000 library.

After the schematic was checked for errors, A symbol was created and named 7seg. Using this symbol, a top-level sheet was created and named *lab7*, and was saved in the *lab7* directory. The input and output buffers and pads were found under the IO category in the XC4000 library. See Appendix IV for the finished schematic.

The design was now ready for simulation. To prepare the design, the following command was entered at the command prompt:

```
p1d_dve_sim lab7 xc4000
```

This creates a design viewpoint with the appropriate parameters. Then, quicksim could be invoked using the command:

```
quicksim lab7
```

Quicksim was used to verify the design by creating all possible inputs and studying the resulting outputs of the simulation (the results were put in Appendix V).

After the design was simulated and proven to be correct, a bit file was created for programming the FPGA. To do this, the schematic was first converted into a xilinx netlist, which is just a textual representation of the graphical design, but without the graphics. For this, the following command was entered at the prompt:

```
men2xnf8 lab7 -p 4005pc84-5
```

Next a script was run that created numerous files, a couple of which are readable. The most important one created was the *lab7.bit* file, which contains the data for the FPGA.

A timing simulation was then done on the design with the *static timing analysis* technique, using Xilinx's static timing analysis tool *xdelay*. The following command was entered to carry this out:

```
xdelay -s -o lab7.time lab7
```

This produced a file *lab7.time* that contains a list of all the signal paths arranged in order of propagation delay.

Lab 8:

After computer design was completed and tested, the hardware representation was built. This was done using the FPGA, a breadboard, various pullup resistors, the seven segment display, and an octal dip switch (for providing up to sixteen different inputs).

The bit file was downloaded to the FPGA using *Xchecker*, a program for transferring data to the FPGA. As the input was changed, the output was viewed on the seven-segment display. See Appendix V for the breadboard connections used.

Results and Analysis:

The seven-segment decoder worked as planned. The results of the computer simulation can be seen in Appendix V.

The following outputs were gotten for the hardware testing (active low):

Input	9	0	8	A	3	C
8	Off	On	Off	Off	On	Off
4	On	On	On	On	On	Off
2	On	On	On	Off	Off	On
1	Off	On	On	On	Off	On
Output:	9	0	8	A	3	C

These are the outputs expected from the corresponding inputs.

The worst case propagation delay was found to be 30.5 ns, from *dip switch 5* (see Appendix IV) to segment *a*. This result came from running *xdelay*.

Conclusion:

In this experiment a seven-segment display was successfully designed and implemented using the bottom-up design technique, in which the small components are built first, and then added to a larger component until the design is complete. This worked quite well since each component designed could be tested before making the top-level design, which covered all the components.

The design began with the lowest level of design, that is, the truth table. It was arranged into the popular Karnaugh map format, allowing for reduction and minimization. From the maps, the expressions needed for the design were carefully derived and used for the decoder schematic. This step is crucial since one small error could largely effect the output.

The design was simulated on the computer using Mentor's computer software package. By doing this, errors could be detected and taken care of before hardware implementation. This is a very important step in designing that could save much time and money.

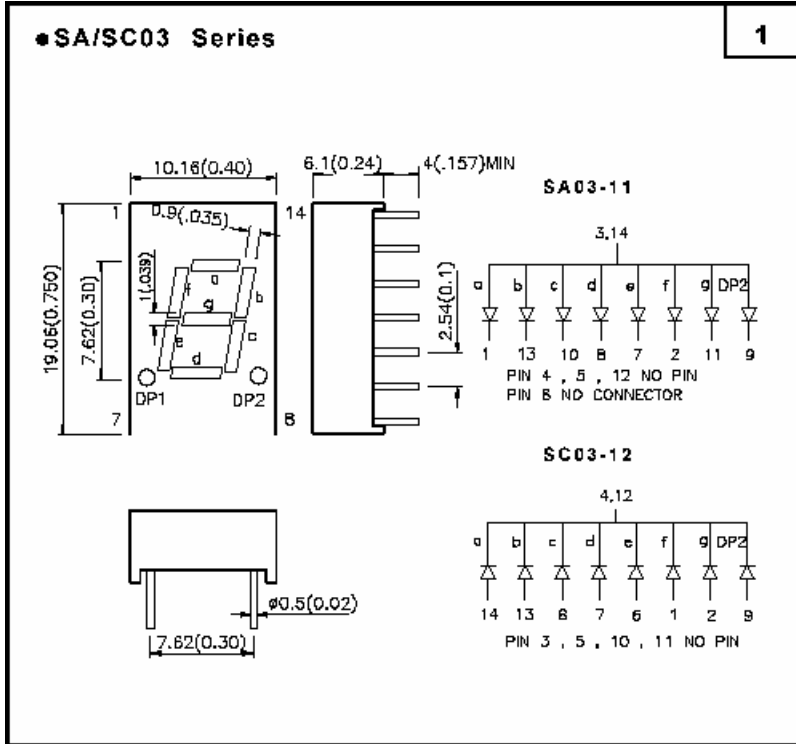
The finished seven-segment display is very useful for displaying alpha-numeric characters and can be found in many electronic devices such as calculators, cell phones, and watches.

Appendix I – Equipment List:

Sun Workstation
Mentor Graphics Software Package
Xilinx XACT software
xc4005 Field Programmable Gate Array
PLCC adapter
Breadboard
Seven segment display
Dip switch
Various resistors and connectors

Appendix II – Seven Segment Display and Truth Table:

Seven segment display



Truth Table

w	x	y	z	Hex	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	0	1	1	1	1
0	0	1	0	2	0	0	1	0	0	1	0
0	0	1	1	3	0	0	0	0	1	1	0
0	1	0	0	4	1	0	0	1	1	0	0
0	1	0	1	5	0	1	0	0	1	0	0
0	1	1	0	6	0	1	0	0	0	0	0
0	1	1	1	7	0	0	0	1	1	1	1
1	0	0	0	8	0	0	0	0	0	0	0
1	0	0	1	9	0	0	0	0	1	0	0
1	0	1	0	A	0	0	0	1	0	0	0
1	0	1	1	b	1	1	0	0	0	0	0
1	1	0	0	C	0	1	1	0	0	0	1
1	1	0	1	d	1	0	0	0	0	1	0
1	1	1	0	E	0	1	1	0	0	0	0
1	1	1	1	f	0	1	1	1	0	0	0

Appendix III – Karnaugh Maps and Expressions:

Note: w and x are on each column and y and z are on each row.

Sum of Products is abbreviated with SOP

Segment a

	00	01	11	10
00	0	1	0	0
01	1	0	1	0
11	0	0	0	1
10	0	0	0	0

$$\text{SOP: } w'x'y'z + w'xy'z' + wxy'z + wx'yz$$

Segment b

	00	01	11	10
00	0	0	1	0
01	0	1	0	0
11	0	0	1	1
10	0	1	1	0

$$\text{SOP: } xyz' + wyz + w'xy'z + wxy'z'$$

Segment c

	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	0	0	1	0
10	1	0	1	0

$$\text{SOP: } w'x'yz' + wxy'z' + wxy$$

Segment d

	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	0	1	1	0
10	0	0	0	1

$$\text{SOP: } w'xy'z' + w'x'y'z + wx'yz' + xyz$$

Segment e

	00	01	11	10
00	0	1	0	0
01	1	1	0	1
11	1	1	0	0
10	0	0	0	0

$$\text{SOP: } x'y'z + w'xy' + w'z$$

Segment f

	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	1	0	0
10	1	0	0	0

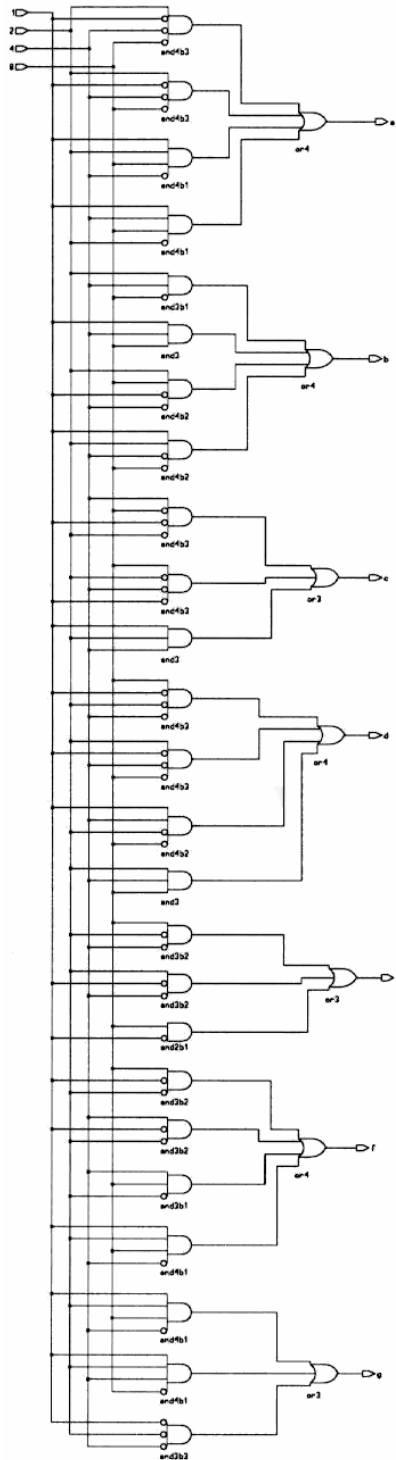
$$\text{SOP: } w'x'z + w'x'y + w'yz + wxy'z$$

Segment g

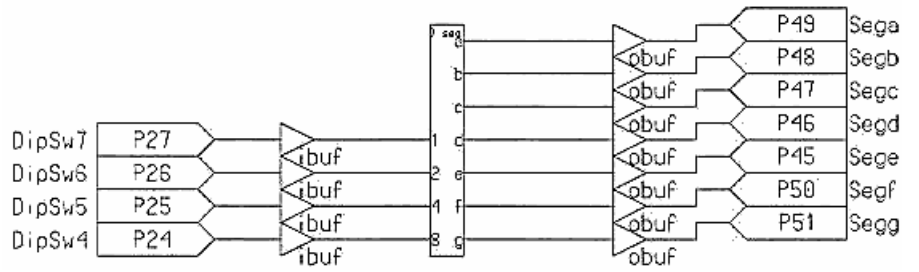
	00	01	11	10
00	1	0	1	0
01	1	0	0	0
11	0	1	0	0
10	0	0	0	0

$$\text{SOP: } wxy'z' + w'xyz + w'x'y'$$

Appendix IV – Schematics:



Seven segment decoder



Top level schematic

Appendix IV – Verification Data:

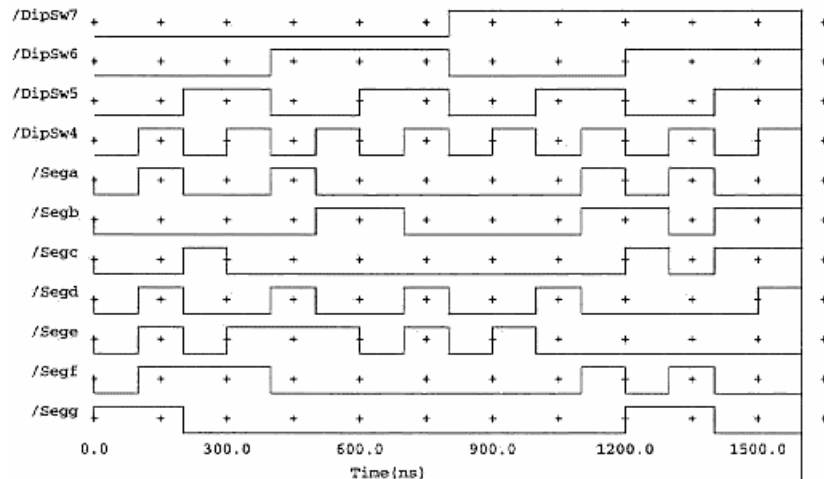
List output from Quicksim

```

    0.0 0 0 0 0 Xr Xr Xr Xr Xr Xr Xr
    0.1 0 0 0 0 X X X X X X X
    0.4 0 0 0 0 0 0 0 0 0 0 1
  100.0 0 0 0 1 0 0 0 0 0 0 1
  100.4 0 0 0 1 1 0 0 1 1 1 1
  200.0 0 0 1 0 1 0 0 1 1 1 1
  200.4 0 0 1 0 0 0 1 0 0 1 0
  300.0 0 0 1 1 0 0 1 0 0 1 0
  300.4 0 0 1 1 0 0 0 0 1 1 0
  400.0 0 1 0 0 0 0 0 0 1 1 0
  400.4 0 1 0 0 1 0 0 1 1 0 0
  500.0 0 1 0 1 1 0 0 1 1 0 0
  500.4 0 1 0 1 0 1 0 0 1 0 0
  600.0 0 1 1 0 0 1 0 0 1 0 0
  600.4 0 1 1 0 0 1 0 0 0 0 0
  700.0 0 1 1 1 0 1 0 0 0 0 0
  700.4 0 1 1 1 0 0 0 1 1 0 0
  800.0 1 0 0 0 0 0 0 1 1 0 0
  800.4 1 0 0 0 0 0 0 0 0 0 0
  900.0 1 0 0 1 0 0 0 0 0 0 0
  900.4 1 0 0 1 0 0 0 0 1 0 0
 1000.0 1 0 1 0 0 0 0 0 1 0 0
 1000.4 1 0 1 0 0 0 0 1 0 0 0
 1100.0 1 0 1 1 0 0 0 1 0 0 0
 1100.4 1 0 1 1 1 1 0 0 0 1 0
 1200.0 1 1 0 0 1 1 0 0 0 1 0
 1200.4 1 1 0 0 0 1 1 0 0 0 1
 1300.0 1 1 0 1 0 1 1 0 0 0 1
 1300.4 1 1 0 1 1 0 0 0 0 1 1
 1400.0 1 1 1 0 1 0 0 0 0 1 1
 1400.4 1 1 1 0 0 1 1 0 0 0 0
 1500.0 1 1 1 1 0 1 1 0 0 0 0
 1500.4 1 1 1 1 0 1 1 1 0 0 0
Time(ns) ^/DipSw7 ^/DipSw4 ^/Segc ^/Segf
          ^/DipSw6 ^/Sega ^/Segd ^/Segg
          ^/DipSw5 ^/Segb ^/Sege

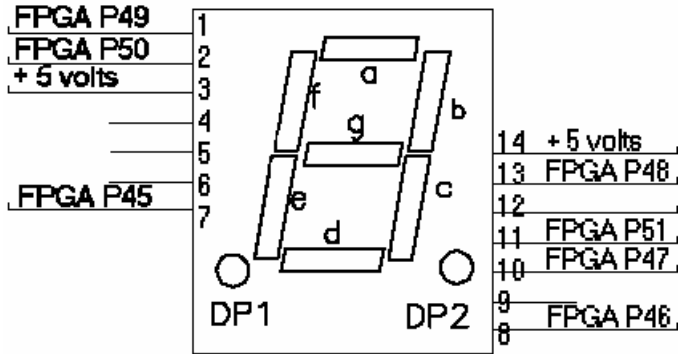
```

Trace Output from Quicksim

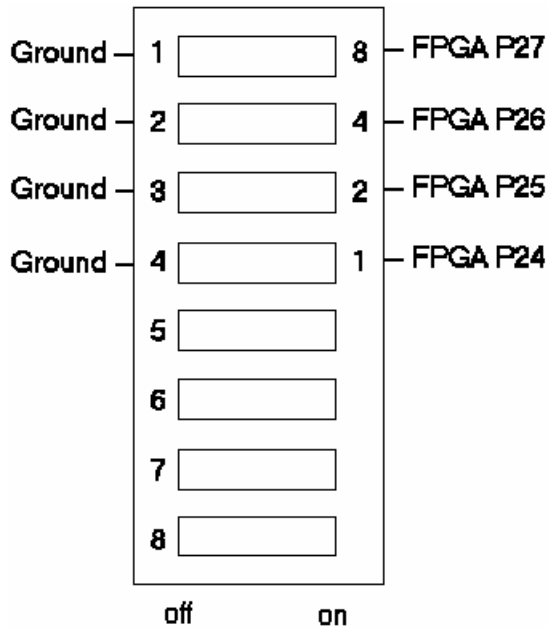


Appendix V – Breadboard Connections:

Connections of the Seven-Segment Display (each connection through a 420 Ω resistor)



Connections for Octal Dip Switch (each FPGA connection through a pullup resistor)



References:

The seven segment images were found at Kingbright® LED on the web:

<http://www.kingbright.com/>

All other expressions, graphs, and figures were created by the author.