Lab#4
14 segment display decoders

Digital design principles.

Work made by:
Christian Aaron Ortega Blanco

Hermosillo, Son. 5 de mayo de 2022.

In order to make a 14 segment decoder, we must know which combinations represents the character that we need, on this particular case, I need to represent the characters that conform my name; C-h-r-i-s-t-i-a-n has nine characters, so the first thing to do is a list of combinations. Im using the Figure 1 to “draw” each letter on a bit format.

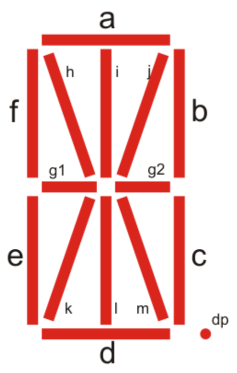


Figure 1 14 segment display

It’s easy to make the list of combinations with a table that contains the specifications of each bit.

		a	b	c	d	e	f	g1	g2	h	i	j	k	l	m
1	c	1	0	0	1	1	1	0	0	0	0	0	0	0	0
2	h	0	0	0	0	1	1	1	0	0	0	0	0	1	0
3	r	0	0	0	0	0	0	0	1	0	0	0	0	1	0
4	i	0	0	0	0	0	0	0	0	0	1	0	0	1	0
5	s	0	0	0	1	0	0	0	1	0	0	0	0	0	1
6	t	0	0	0	1	1	1	1	0	0	0	0	0	0	0
7	i	0	0	0	0	0	0	0	0	0	1	0	0	1	0
8	a	0	0	0	1	1	0	1	0	0	0	0	0	1	0
9	n	0	0	0	0	1	0	1	0	0	0	0	0	1	0

Table 1 Spelling table

Now we can create a txt document that contains only the bits of each character, this document is pretended to act has a ROM memory. In the design Scrib we only need to call the location of every character description in order to display it on his 14-segment format.

Design

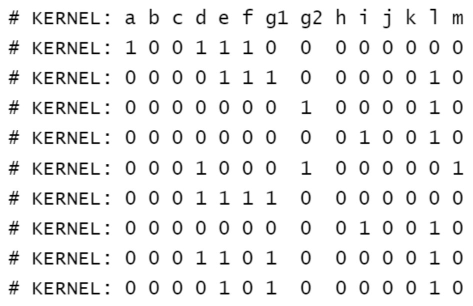
```
module Chris(input reg [3:0] Selection, output reg [13:0] char);

  reg [13:0] Mem [0:8];
  initial $readmemb("mem_c.txt",Mem);
    initial begin
      assign char=Mem[Selection];
  end

endmodule
```

Test Bench

```
`timescale 1ns/1ps
module Chris_TB;
  reg [3:0] Selection=4'b0;  /// selection vector
  wire [13:0] char;  ///14 segnments display connectors

  Chris UUT(Selection,char);  /// christian has 9 characters 14 bits per character

  initial
    begin
      $dumpfile("Chris.vcd");
      $dumpvars(1,Chris_TB);

      $display("a b c d e f g1 g2 h i j k l m");
      #2;
      for(int i=0 ; i<9; i++)begin
        Selection=i;
        #2;
        $display("%b   %b   %b   %b   %b   %b   %b     %b     %b   %b   %b   %b   %b   
%b",char[13],char[12],char[11],char[10],char[9],char[8],char[7],char[6],char[5],char
[4],char[3],char[2],char[1],char[0]);

      end

      $finish;
    end
 endmodule;
```

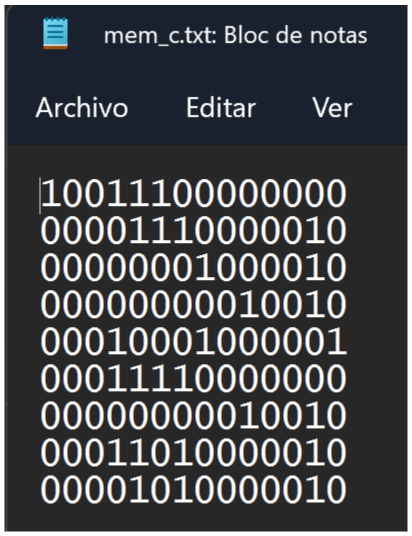
Result

If we run the code, we get a 14segment representation of the characters that contains the word “Christian”, we can confirm this comparing the ROM file with the table that the console displays after running the program, as above;



```
# KERNEL: a b c d e f g1 g2 h i j k l m
# KERNEL: 1 0 0 1 1 1 0  0  0 0 0 0 0 0
# KERNEL: 0 0 0 0 1 1 1  0  0 0 0 0 1 0
# KERNEL: 0 0 0 0 0 0 0  1  0 0 0 0 1 0
# KERNEL: 0 0 0 0 0 0 0  0  0 1 0 0 1 0
# KERNEL: 0 0 0 1 0 0 0  1  0 0 0 0 0 1
# KERNEL: 0 0 0 1 1 1 1  0  0 0 0 0 0 0
# KERNEL: 0 0 0 0 0 0 0  0  0 1 0 0 1 0
# KERNEL: 0 0 0 1 1 0 1  0  0 0 0 0 1 0
# KERNEL: 0 0 0 0 1 0 1  0  0 0 0 0 1 0
```

Table 2 Generated table



```
10011100000000
00001110000010
00000001000010
00000000010010
00010001000001
00011110000000
00000000010010
00011010000010
00001010000010
```

Table 3 ROM data