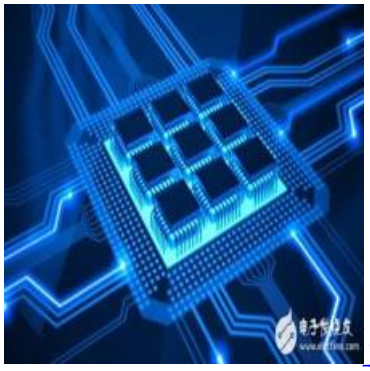
数码管显示实验（一）（初步明白片选、段选）

原创

李锐博恩 于 2018-08-17 17:57:40 发布 20271 收藏 59
分类专栏： Verilog/FPGA 实用总结区

本文链接：https://blog.csdn.net/Reborn_Lee/article/details/81780998



背景：

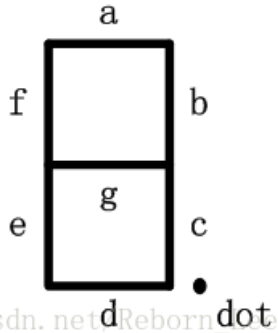
这是一个让我感到耻辱而又欣慰的实验，大约花了我4个半小时时间才调通我下面要写的这篇博文的内容，这篇博文的内容比较简单，所以花这么久时间让我感到自己很弱，故而耻辱，而通过自己的各种改动，适合自己的一块没有资料的FPGA开发板，又是很欣慰的。

一个带小数点的数码管的所有8个发光二极管的正极或负极有一个公共端，通常必须接GND（共阴极数码管）或者接VCC（共阳极数码管），而另一个非公共端的8个引脚就留给用户的I/O直接控制了。

所以，你做实验之前要明白自己的开发板的数码管是共阳极的还是共阴极的，由于板子是师兄留给我的，我就不知道我的板子是共阳极的还是供阴极的，所以显示数字出现乱码时，我就花了很长时间确定是共阳极的还是共阴极的，乱码是这个原因还是引脚约束出问题了等等，很烦恼，当然，最后成功的显示数字之后，心里还是很舒服的。

我的Spartan-6 FPGA开发板中的数码管是共阳极的，因此段选是低电平有效，也就是低电平时，每一段对应的发光二极管点亮。

如下图是数码管的示意图：



如果是共阴极的，那么译码表为：

数字/字符	0	1	2	3	4	5	6	7
编码(16进制)	3f	06	5b	4f	66	6d	7d	07
数字/字符	8	9	A	B	C	D	E	F
编码(16进制)	7f	6f	77	7c	39	5e	79	71

每个数字或字符的编码是怎么得到的呢？

如下表：

段	a	b	c	d	e	f	g	dot	16进制
0	1	1	1	1	1	1	0	0	3f(0011_1111)
1	0	1	1	0	0	0	0	0	06(0000_0110)

上表举了0、1这两个数字的编码方式，应该很明白了吧，从dot开始到a，依次编码，亮为1，灭为0，如此规律，可以找到各个数字或字符对应的16进制编码，该16进制编码最高位赋值给dot，然后是g、f依次到a，这样的话，就应该到时候分配引脚的时候，dot引脚对应的编码位数是最高位，a引脚对应的是编码位数的最低位。

不明白这一点，弄错了，就会出现乱码。

上面说的是共阴极的情况，那么共阳极的情况呢？对应的编码是多少呢？

可知根据规律自己推，这里就直接给出了：

数字/字符	0	1	2	3	4	5	6	7
编码(16进制)	C0	F9	A4	B0	99	92	82	F8
数字/字符	8	9	A	B	C	D	E	F
编码(16进制)	80	90	88	83	C6	A1	86	8E

下面说说这个实验的要求，由于只是初步了解数码管的段选片选，所以就不要求那么复杂了，相对简单一些：

看看你的开发板上有几个数码管，让它们同时显示数字从0 到 F。

下面给出我的FPGA设计的Verilog HDL硬件描述语言：

这个代码是特权同学的，我改动了下，适应我的FPGA开发板，（数码管是共阳极的，片选是低电平有效，8个数码管。）

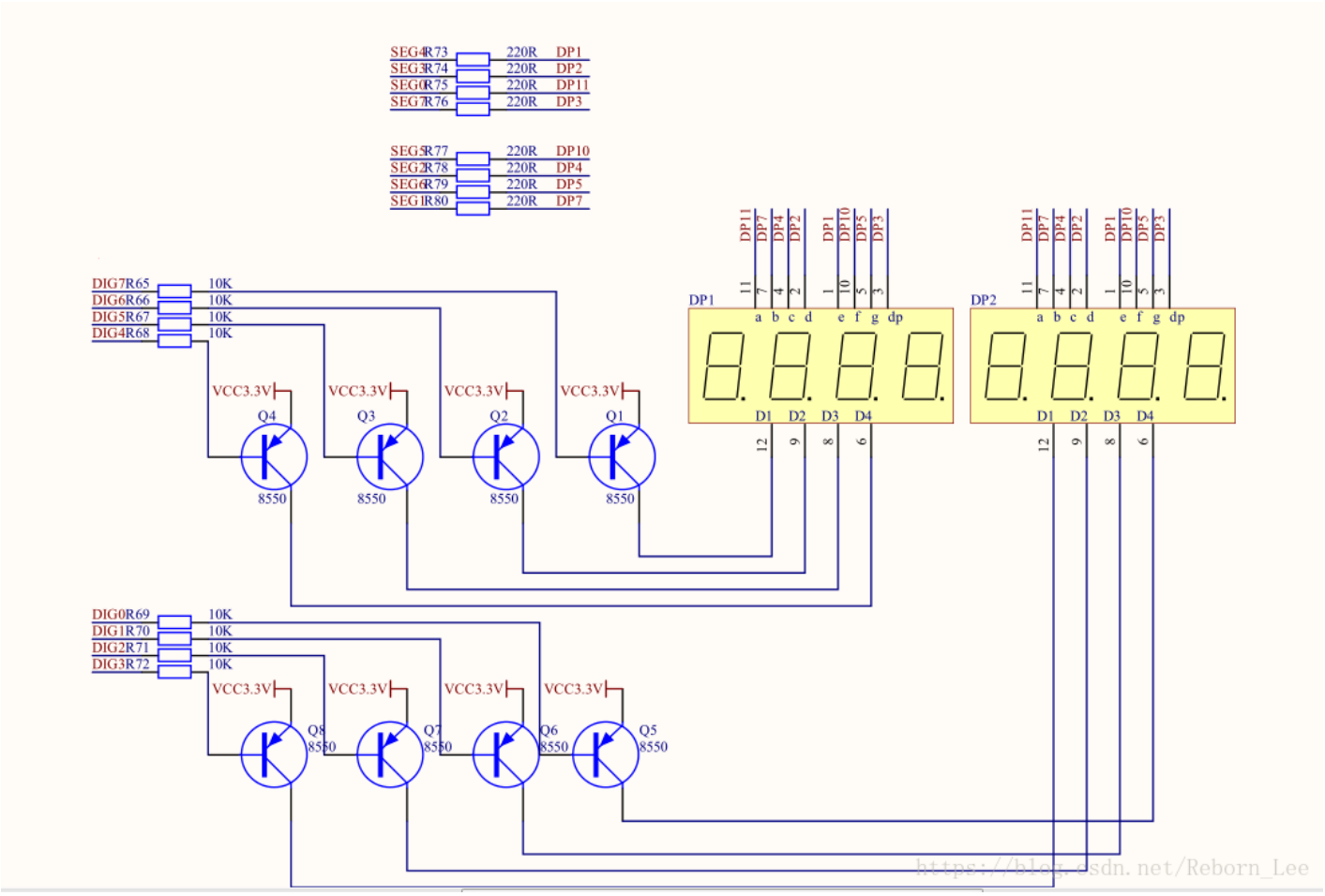
```
`timescale 1ns / 1ps
//
// Company:
// Engineer:
//
// Create Date:    14:03:17 08/17/2018
// Design Name:
// Module Name:    leg_seg7
// Project Name:
```

```
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
module led_seg7(
   clk,rst_n,
   sm_cs1_n,sm_db
  );

input clk;  // 50MHz
input rst_n; // 复位信号，低有效

output[7:0] sm_cs1_n; //数码管片选信号，低有效
output[7:0] sm_db; //8段数码管（包括小数点）

reg[24:0] cnt; //计数器，最大可以计数到2的25次方*20ns=640ms

always @ (posedge clk or negedge rst_n)
 if(!rst_n) cnt <= 25'd0;
 else cnt <= cnt+1'b1; //循环计数

reg[3:0] num; //显示数值

always @ (posedge clk or negedge rst_n)
 if(!rst_n) num <= 4'd0;
 else if(cnt == 25'h1ffffff) num <= num+1'b1; //每640ms增一

//-------------------------------------------------------------------------------
/* 共阳级 :带小数点
              ;0,  1,  2,  3,  4,  5,  6,  7,
      db      C0,  F9, A4, B0, 99, 92, 82, F8
              ;8,  9,  a,   b,   c,   d,   e,   f , 灭
      db       80, 90, 88,  83,  C6,  A1,  86,  8E, ff*/
parameter seg0 = 7'hC0,
   seg1 = 7'hF9,
   seg2 = 7'hA4,
   seg3 = 7'hB0,
   seg4 = 7'h99,
   seg5 = 7'h92,
   seg6 = 7'h82,
   seg7 = 7'hF8,
   seg8 = 7'h80,
   seg9 = 7'h90,
   sega = 7'h88,
   segb = 7'h83,
   segc = 7'hC6,
   segd = 7'hA1,
   sege = 7'h86,
   segf = 7'h8E;

reg[7:0] sm_dbr;  //8段数码管（包括小数点）
```

```
always @ (num)
  case (num) //NUM值显示在数码管上
   4'h0: sm_dbr <= seg0;

   4'h1: sm_dbr <= seg1;
   4'h2: sm_dbr <= seg2;
   4'h3: sm_dbr <= seg3;
   4'h4: sm_dbr <= seg4;
   4'h5: sm_dbr <= seg5;
   4'h6: sm_dbr <= seg6;
   4'h7: sm_dbr <= seg7;
   4'h8: sm_dbr <= seg8;
   4'h9: sm_dbr <= seg9;
   4'ha: sm_dbr <= sega;
   4'hb: sm_dbr <= segb;
   4'hc: sm_dbr <= segc;
   4'hd: sm_dbr <= segd;
   4'he: sm_dbr <= sege;
   4'hf: sm_dbr <= segf;
   default: ;
   endcase

assign sm_db = sm_dbr;
assign sm_cs1_n = 8'b0000_0000;  //数码管常开


endmodule
```

对上述代码需要解释的是片选信号有8个，低电平有效，我把8个数码区的片选全部赋值为0了，这样8个数码管就全部有效，同步显示0到F.（片选的意思是选择哪一个数码管有效）

这是我的数码管的电路图：



对应的FPGA引脚列表：

	数码管接口	FPGA引脚号
	DIG0	100
	DIG1	99
	DIG2	102
	DIG3	101
	DIG4	9
	DIG5	10
	DIG6	11
	DIG7	12
	SEG0	119
	SEG1	116
	SEG2	118
	SEG3	124
	SEG4	121
	SEG5	117
	SEG6	115
	SEG7	120

4		
5	复位按键RESET	62（底板KEY6）
6		104
7	全局时钟FPGA_CLK	55

给出我的引脚约束：

可根据实际情况改动用于自己的FPGA开发板：

```
# PlanAhead Generated physical constraints

NET "rst_n" LOC = P62;
NET "clk" LOC = P55;
NET "sm_cs1_n[0]" LOC = P100;
NET "sm_cs1_n[1]" LOC = P99;
NET "sm_cs1_n[2]" LOC = P102;
NET "sm_cs1_n[3]" LOC = P101;
NET "sm_cs1_n[4]" LOC = P9;
NET "sm_cs1_n[5]" LOC = P10;
NET "sm_cs1_n[6]" LOC = P11;
NET "sm_cs1_n[7]" LOC = P12;



NET "sm_db[0]" LOC = P119;
NET "sm_db[1]" LOC = P116;
NET "sm_db[2]" LOC = P118;
NET "sm_db[3]" LOC = P124;
NET "sm_db[4]" LOC = P121;
NET "sm_db[5]" LOC = P117;
NET "sm_db[6]" LOC = P115;
NET "sm_db[7]" LOC = P120;
```

根据以上信息，在ISE中进行综合、实现、上板子调试，结果很完美。