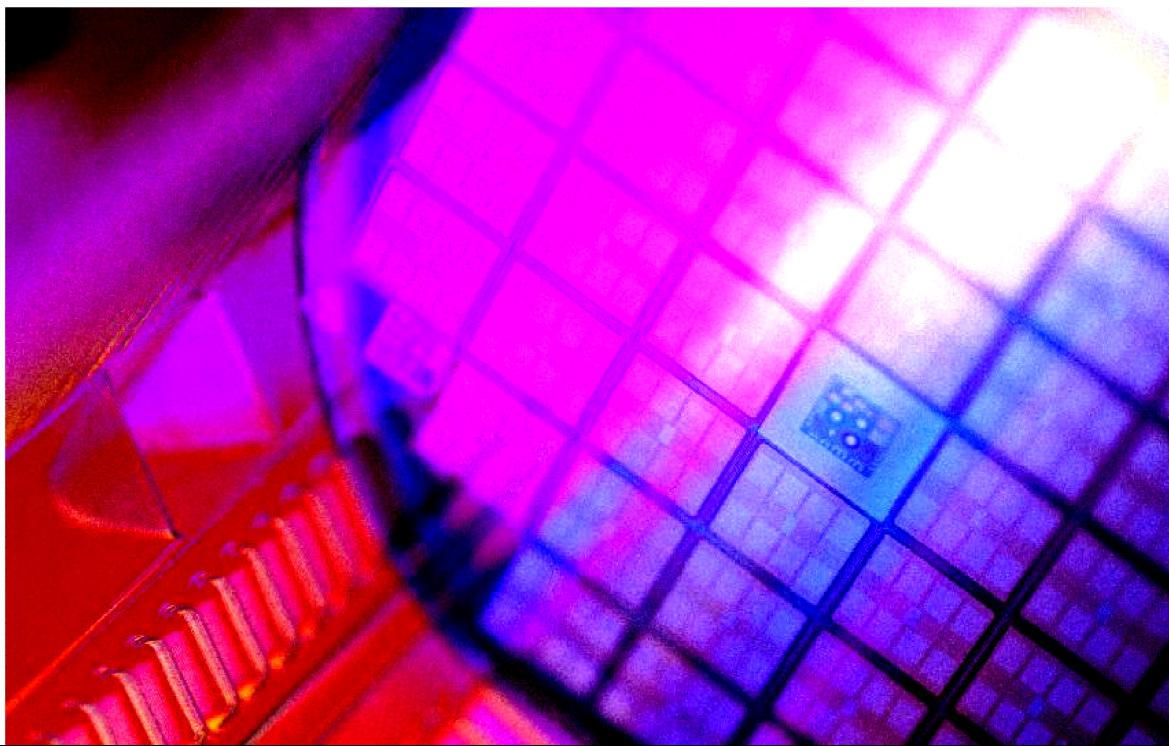


FEQ1301T03-V0.00



ZRtech

# FPGA 开发套件 HDL 实验教程

—数码管实验



[www.zr-tech.com](http://www.zr-tech.com)

# 实验 3 数码管实验

## 1.1 第 3 例-数码管

### 1.1.1 静态数码管 ( example\_dt\_1 )

同 led 的例化一样，对 beep 蜂鸣器也作出同样的例化处理，并加入了控制的 En 管脚。

通常的子模块中都会加入“使能信号 En”或者“数据输出有效 Valid”。这里控制不仅仅是为了高端大气，而是 FPGA 中所有的模块都是并行的。这是和 C 的重要区别，所有模块只要例化就会生成，也就是在 FPGA 资源中有了一席之地，无时无刻这个模块都在工作。不是你调用或者不调用的事了。

简洁的代码总能使人编程的心情变得愉快，数码管驱动之前先将代码再一次封装：

```
13  led_module led_ct(
14      .clk(CLK),
15      .en(1'b1),
16      .led(led)
17  );
18
19  beep_module beep_ct(
20      .clk(CLK),
21      .en(1'b1),
22      .beep(BP1)
23 );
```

这里只要将使能信号置 0 就可以控制模块的输出，具体请参考代码。你也可以用其他的方式去控制，这里只提供你一个途径而已。

数码管的驱动较为简单，因为 FPGA 天然的并行性，只要对数码管进行 7 段编码的参数化设置就可以分别驱动 7 段码了。代码如下：

```
28 parameter [6:0]NUM_0=7'b0111111;
29 parameter [6:0]NUM_1=7'b0000110;
30 parameter [6:0]NUM_2=7'b1011011;
31 parameter [6:0]NUM_3=7'b1001111;
32 parameter [6:0]NUM_4=7'b1100110;
33 parameter [6:0]NUM_5=7'b1101101;
34 parameter [6:0]NUM_6=7'b1111101;
35 parameter [6:0]NUM_7=7'b0000111;
36 parameter [6:0]NUM_8=7'b1111111;
37 parameter [6:0]NUM_9=7'b1101111;
38 parameter [6:0]NUM_B=7'b0000000;
39
40 parameter [7:0]EN_1=4'b1110;
41 parameter [7:0]EN_2=4'b1101;
42 parameter [7:0]EN_3=4'b1011;
43 parameter [7:0]EN_4=4'b0111;
44 parameter [3:0]EN_A=4'b0000;
45
46 assign {DS_G,DS_F,DS_E,DS_D,DS_C,DS_B,DS_A} = NUM_8 ;
47 assign {DS_EN1,DS_EN2,DS_EN3,DS_EN4} = EN_A;
```

这是一段静态的编码，点亮了所有段位，因此显示 4 个 8。

### 1.1.2 动态数码管 ( example\_dt\_2 )

动态的只是在静态的基础上在不同时钟节拍显示不同的数字。这两种本质的区别就是组合逻辑和时序逻辑的关系。由于 FPGA 的并行性，每时每刻数码管都能得到驱动，MCU 控制数码管所用的动态扫描方法和 FPGA 动态驱动的区别。

FPGA 的重要优势之一就在于并发的控制，这种控制不像 MCU 需要核心运算单元干预，因此不存在占用 CPU 时间片的说法。需要用到高速扫描电路驱动时，例如 VGA、RGB LCD 显示等领域，除了专用的 ASIC 之外，FPGA 具有相当大的优势并且兼具灵活性和兼容性。

说远了，数码管动态驱动的代码如下，里面加入了 16 进制的编码 A-F：

```
54     reg [3:0]num_dt[3:0];
55     reg [6:0]ds_reg;
56     reg [3:0]ds_en;
57     assign {DS_G,DS_F,DS_E,DS_D,DS_C,DS_B,DS_A} = ds_reg ;
58     assign {DS_EN1,DS_EN2,DS_EN3,DS_EN4} = ds_en;
59
60     reg [31:0]cnt_dt;
61     always @ (posedge CLK)
62     begin
63         cnt_dt = cnt_dt + 1'b1;
64     end
65
66     always @ (posedge CLK)
67     begin
68         case (cnt_dt[27:24])
69             4'h0 : begin ds_reg = NUM_0; ds_en=EN_A;end
70             4'h1 : begin ds_reg = NUM_1; ds_en=EN_A;end
71             4'h2 : begin ds_reg = NUM_2; ds_en=EN_A;end
72             4'h3 : begin ds_reg = NUM_3; ds_en=EN_A;end
73             4'h4 : begin ds_reg = NUM_4; ds_en=EN_A;end
74             4'h5 : begin ds_reg = NUM_5; ds_en=EN_A;end
75             4'h6 : begin ds_reg = NUM_6; ds_en=EN_A;end
76             4'h7 : begin ds_reg = NUM_7; ds_en=EN_A;end
77             4'h8 : begin ds_reg = NUM_8; ds_en=EN_A;end
78             4'h9 : begin ds_reg = NUM_9; ds_en=EN_A;end
79             4'ha : begin ds_reg = NUM_A; ds_en=EN_A;end
80             4'hb : begin ds_reg = NUM_B; ds_en=EN_A;end
81             4'hc : begin ds_reg = NUM_C; ds_en=EN_A;end
82             4'hd : begin ds_reg = NUM_D; ds_en=EN_A;end
83             4'he : begin ds_reg = NUM_E; ds_en=EN_A;end
84             4'hf : begin ds_reg = NUM_F; ds_en=EN_A;end
85             default: begin ds_reg = NUM_BLK; ds_en=EN_A;end
86         endcase
87     end
```

### 1.1.3 数码管元件化 ( example\_dt\_3 )

数码管相比 LED 和蜂鸣器来说 ,控制比较复杂 ,因此元件化改动比较大 ,为了封装数码管 ,首先要考虑到四个数字是独立控制显示的。因此在做控制时需要将输入进行独立处理这里选择将 4 个 7 段码数据寄存器作为独立的 4 组输入。如下 ,具体的代码部分不再详述了 ,请参考例子 :

```
1 //      (0)
2 //      -----
3 //      |      |
4 //      (5) | (6) | (1)
5 //      -----
6 //      |      |
7 //      (4) |      | (2)
8 //      -----
9 //      (3)

10
11 module dt_module(
12     //Clock Input:48M
13     input  clk,
14     input [3:0]num1,
15     input [3:0]num2,
16     input [3:0]num3,
17     input [3:0]num4,
18     output reg [3:0]ds_en,
19     //Digital tube
20     output reg [6:0]ds_reg
21 );
```

顶层的调用中使用了寄存器组的形式，如下：

```
35     reg [3:0]num_dt[3:0];
36     dt_module dt_ct(
37         .clk(CLK),
38         .num1(num_dt[0]),
39         .num2(num_dt[1]),
40         .num3(num_dt[2]),
41         .num4(num_dt[3]),
42         .ds_en(ds_en),
43         .ds_reg(ds_reg)
44     );
45
46     reg [31:0]cnt;
47     always @ (posedge CLK)
48     begin
49         cnt = cnt + 1;
50         num_dt[0] = cnt[27:24];
51         num_dt[1] = cnt[27:24];
52         num_dt[2] = cnt[27:24];
53         num_dt[3] = cnt[27:24];
54     end
55
```

寄存器组固定定义为这种格式，表示 4 个 4bit 的寄存器组成的寄存器组。通常在 FPGA 中较大的寄存器组应该作为 Memory 使用，否则会占用大量寄存器资源并且降低整个设计的最大运行频率。

至此，讲了基本的例子，做了一些简单的和稍微复杂些的例化工作。在以后的例子中，会对资源控制进行示例，前面已经了解的资源会直接调用不再赘述。

文档内部编号 : FEQ1301T03

编号说明 :

首一字母 : F-FPGA 系列

首二字母 : L-理论类 E-实验类 T-专题类

首三字母 : C-普及类 Q-逻辑类 S-软核类

数字前两位 : 代表年度

数字后两位 : 同类文档顺序编号

尾字母/数字 : C 目录 , T 正文 , 数字表示章节号

## 修订记录

版本号	日期	描述	修改人
0.00	2013.9.25	FEQ1301T03 文档建立	kdy