

# 点亮数字人生

计 93 王哲凡 2019011200

## 一、实验目的

通过数码管点亮程序，熟悉一种 HDL 语言，了解掌握硬件程序的编写规范，掌握 EDA 软件的使用方法和工作流程，进一步理解可编程芯片的工作原理。

## 二、实验内容与要求

1. 使用至多两个带译码的数码管和至少一个不带译码的数码管，循环有规律地显示奇数列、偶数列、自然数列。
2. 使用至少一个带译码的数码管和至少一个不带译码的数码管，循环有规律的显示实验日期以及桌号（2021040904）。

## 三、实验代码及简要原理分析

### 1. 奇数列、偶数列、自然数列

#### 1.1. 实验代码

```
1 module Light(  
2     input clk,  
3     input rst,  
4     output reg [3:0] even,  
5     output reg [3:0] odd,  
6     output reg [6:0] nature_encode  
7 );  
8  
9     reg [3:0] nature;  
10    reg [3:0] even_decode;  
11    reg [3:0] odd_decode;
```

```

12
13     initial
14     begin
15         even_decode <= 0;
16         odd_decode <= 1;
17         nature <= 0;
18     end
19
20     always@(even_decode)
21         even <= even_decode;
22
23     always@(odd_decode)
24         odd <= odd_decode;
25
26     always@(nature)
27         case (nature)
28             4'b0000: nature_encode = 7'b11111110;
29             4'b0001: nature_encode = 7'b01100000;
30             4'b0010: nature_encode = 7'b1101101;
31             4'b0011: nature_encode = 7'b1111001;
32             4'b0100: nature_encode = 7'b0110011;
33             4'b0101: nature_encode = 7'b1011011;
34             4'b0110: nature_encode = 7'b1011111;
35             4'b0111: nature_encode = 7'b1110000;
36             4'b1000: nature_encode = 7'b1111111;
37             4'b1001: nature_encode = 7'b1111011;
38             4'b1010: nature_encode = 7'b1110111;
39             4'b1011: nature_encode = 7'b0011111;
40             4'b1100: nature_encode = 7'b1001110;
41             4'b1101: nature_encode = 7'b0111101;
42             4'b1110: nature_encode = 7'b1001111;
43             4'b1111: nature_encode = 7'b1000111;
44             default: nature_encode = 7'b0000000;
45         endcase
46
47     always@(posedge clk or posedge rst)
48     begin
49         if (rst)
50         begin
51             even_decode <= 0;
52             odd_decode <= 1;
53             nature <= 0;
54         end
55         else
56         begin
57             if (even_decode == 4'b1000)
58                 even_decode <= 0;
59             else

```

```

60         even_decode <= even_decode + 2;
61     if (odd_decode == 4'b1001)
62         odd_decode <= 1;
63     else
64         odd_decode <= odd_decode + 2;
65     if (nature == 4'b1001)
66         nature <= 0;
67     else
68         nature <= nature + 1;
69     end
70 end
71 endmodule
72

```

## 1.2. 原理及功能简要说明

输入接口包括：

```

1 input clk,
2 input rst

```

分别表示计数器（每按一下就显示下一个数字）与复原器（按下后各管变为初始值）。

输出接口包括：

```

1 output reg [3:0] even,
2 output reg [3:0] odd,
3 output reg [6:0] nature_encode

```

分别表示偶数列（带译码）、奇数列（带译码）与自然数列（不带译码）的输出，其中 nature\_encode 从高到低表示 A 到 G，even 和 odd 均为从高到低表示 8 ~ 1。

其他寄存器包括：

```

1 reg [3:0] nature;
2 reg [3:0] even_decode;
3 reg [3:0] odd_decode;

```

分别用于记录三个数列的当前值，用于记录上一状态以及转换，通过下面三个 always 块负责改变对应数码管的输出：

```

1 always@(even_decode)
2     even <= even_decode;
3
4 always@(odd_decode)
5     odd <= odd_decode;
6

```

```

7  always@(nature)
8      case (nature)
9          4'b0000: nature_encode = 7'b11111110;
10         4'b0001: nature_encode = 7'b01110000;
11         4'b0010: nature_encode = 7'b1101101;
12         4'b0011: nature_encode = 7'b1111001;
13         4'b0100: nature_encode = 7'b0110011;
14         4'b0101: nature_encode = 7'b1011011;
15         4'b0110: nature_encode = 7'b1011111;
16         4'b0111: nature_encode = 7'b1110000;
17         4'b1000: nature_encode = 7'b1111111;
18         4'b1001: nature_encode = 7'b1111011;
19         4'b1010: nature_encode = 7'b1110111;
20         4'b1011: nature_encode = 7'b0011111;
21         4'b1100: nature_encode = 7'b1001110;
22         4'b1101: nature_encode = 7'b0111101;
23         4'b1110: nature_encode = 7'b1001111;
24         4'b1111: nature_encode = 7'b1000111;
25         default: nature_encode = 7'b0000000;
26     endcase

```

最后通过一个 always 块处理 clk 和 rst 的信号触发，来对应改变三个计数器：

```

1  always@(posedge clk or posedge rst)
2  begin
3      if (rst)
4          begin
5              even_decode <= 0;
6              odd_decode <= 1;
7              nature <= 0;
8          end
9      else
10         begin
11             if (even_decode == 4'b1000)
12                 even_decode <= 0;
13             else
14                 even_decode <= even_decode + 2;
15             if (odd_decode == 4'b1001)
16                 odd_decode <= 1;
17             else
18                 odd_decode <= odd_decode + 2;
19             if (nature == 4'b1001)
20                 nature <= 0;
21             else
22                 nature <= nature + 1;
23         end
24     end

```

优先判断，如果是 rst 的上升沿，则置零，否则说明是 clk 的上升沿，进行三个数列的计数。

此处将 rst 和 clk 的两个上升沿触发放到一个 always 块，是因为 Verilog 不能在不同的 always 块修改同一个寄存器的值。

## 2. 特定循环数列

### 2.1. 实验代码

```
1  module Light(  
2      input clk,  
3      input rst,  
4      output reg [3:0] even,  
5      output reg [3:0] odd,  
6      output reg [6:0] nature_encode  
7  );  
8  
9      reg [3:0] nature;  
10     reg [3:0] cnt;  
11     reg [3:0] num;  
12  
13     initial  
14     begin  
15         cnt <= 0;  
16     end  
17  
18     always@(cnt)  
19         case (cnt)  
20             4'b0000: num = 2;  
21             4'b0001: num = 0;  
22             4'b0010: num = 2;  
23             4'b0011: num = 1;  
24             4'b0100: num = 0;  
25             4'b0101: num = 4;  
26             4'b0110: num = 0;  
27             4'b0111: num = 9;  
28             4'b1000: num = 0;  
29             4'b1001: num = 4;  
30             default: num = 0;  
31         endcase  
32  
33     always@(num)  
34     begin  
35         even <= num;  
36         odd <= num;  
37         case (num)  
38             4'b0000: nature_encode = 7'b1111110;  
39             4'b0001: nature_encode = 7'b0110000;
```

```

40         4'b0010: nature_encode = 7'b1101101;
41         4'b0011: nature_encode = 7'b1111001;
42         4'b0100: nature_encode = 7'b0110011;
43         4'b0101: nature_encode = 7'b1011011;
44         4'b0110: nature_encode = 7'b1011111;
45         4'b0111: nature_encode = 7'b1110000;
46         4'b1000: nature_encode = 7'b1111111;
47         4'b1001: nature_encode = 7'b1111011;
48         4'b1010: nature_encode = 7'b1110111;
49         4'b1011: nature_encode = 7'b0011111;
50         4'b1100: nature_encode = 7'b1001110;
51         4'b1101: nature_encode = 7'b0111101;
52         4'b1110: nature_encode = 7'b1001111;
53         4'b1111: nature_encode = 7'b1000111;
54         default: nature_encode = 7'b0000000;
55     endcase
56 end
57
58 always@(posedge clk or posedge rst)
59 begin
60     if (rst)
61     begin
62         cnt = 0;
63     end
64     else
65     begin
66         if (cnt == 4'b1001)
67             cnt <= 0;
68         else
69             cnt <= cnt + 1;
70     end
71 end
72 endmodule
73

```

## 2.2. 原理及功能简要说明

输入输出类似，但输出均代表译码或未译码的规定数列。

```

1  reg [3:0] cnt;
2  reg [3:0] num;

```

cnt 代表当前是循环数列的第几位，num 代表是具体是什么数，二者通过一个 always 块联系：

```

1  always@(cnt)
2      case (cnt)
3          4'b0000: num = 2;

```

```

4      4'b0001: num = 0;
5      4'b0010: num = 2;
6      4'b0011: num = 1;
7      4'b0100: num = 0;
8      4'b0101: num = 4;
9      4'b0110: num = 0;
10     4'b0111: num = 9;
11     4'b1000: num = 0;
12     4'b1001: num = 4;
13     default: num = 0;
14 endcase

```

nature\_encode 的译码依然同上，而 clk 与 rst 的上升沿须对应修改为对于 cnt 的修改：

```

1 always@(posedge clk or posedge rst)
2 begin
3     if (rst)
4     begin
5         cnt = 0;
6     end
7     else
8     begin
9         if (cnt == 4'b1001)
10            cnt <= 0;
11        else
12            cnt <= cnt + 1;
13    end
14 end

```

## 四、实验收获

### 1. Verilog 收获

在模块中的不同 always 块中不能修改同一个寄存器的值，这个设计主要是为了防止数据冒险。

Verilog 中的非阻塞赋值只能用于寄存器，因此上述变量基本均采用寄存器而非线网。

同一个 always 块中尽量不出现一个信号的上升沿与另一个的下降沿，在实际板子中难以实现。

## 2. 环境配置

在 Quartus 中需要指定 Modelsim 目录，选择应该是 modelsim\_ase 目录对应的文件夹。

在仿真时，需要去除 -novopt 选项以防止 Error 产生。

安装驱动时，可能需要通过系统设置的高级选项，进行禁用驱动程序签名强制的启动，再进行对应驱动 USB-Blaster。

## 3. 实验心得

Verilog 作为硬件语言，有很多跟软件语言不一样的地方，特别是它的运行顺序逻辑。

但是整体来说，仍然有类似于函数、变量的结构，可以一定程度上类比学习。