Verilog 第_7_次实验报告

实验名称	数码管显示模块的设计				
学生姓名	邢清画	学号	2211999	指导老师	董前琨
实验地点	津南实验楼 A 区 308		实验时间	2023 年 12 月 19 日	

一、实验项目名称

Verilog 中多位数码管显示的模块设计

二、实验目的

使用 Verilog 完成一个模块，接受八位拨码开关信号输入并在数码管上显示对应的十进制数字。

三、必修或选修

选修

四、实验平台

Vivado

五、实验内容及步骤

实验内容：

结合 CPU 试验箱指导手册，完成并改进 8 段数码管实验，要求和注意事项如下：

1、原始实验的 8 段数码管为组合逻辑电路，只能显示一个数字（或者多个一样的数字），请将代码修改成数码管显示 2 个不同数字，要求八个拨码开关分两组，一组控制显示一个数字。

2、数码管的显示引脚是复用的，所有数码管只是由于片选信号不同才才不同位置显示，所以修改时需要把代码改成时序逻辑电路。实验箱上有统一的时钟信号（100Mhz）,通过时钟信号不断刷新，在不同时刻让片选不同的数码管，选择的同时显示对应的数据，从而达到不断刷新过程中显示多个数字的目的。

设计思路：

通过将输入的数字分解为个位和十位，并使用时钟信号来控制数码管的动态刷新。代码中实现了一个计数器来交替切换两个数码管的激活状态，同时根据每个数字的 BCD（二进制编码的十进制数）编码来控制数码管的显示。这样，尽管物理上只有一个数码管被激活，观察者看到的效果却是两个数码管同时显示不同的数字。

1.输入和输出定义:
bcd_num: 七位输入，用于提供要显示的数字。
clk: 时钟信号输入，用于控制数码管的刷新。
scan_select: 用于选择哪个数码管被激活。
seg7: 控制七段数码管显示的输出。
2.初始状态设定:
初始设置 seg7 为显示某个数字的状态。
初始化其他控制变量。
3.数字分解:
通过对输入的 bcd_num 进行运算，将其分解为个位和十位的 BCD 编码（二进制编码的十进制数），分别存储在 dig_0 和 dig_1 中。
4.时钟信号处理:
使用 clk（时钟信号）来生成一个计数器 jud，用于控制数码管的刷新频率。
使用 jud 的某一位作为触发信号，反转 judge 变量。这里 judge 用于切换当前活跃的数码管。
5.数码管选择和显示:
通过 judge 的状态，选择当前激活的数码管（使用 scan_select）。
根据当前激活的数码管，使用 case 语句来选择对应的数字显示。例如，当显示个位数时，根据 dig_0 的值来设置 seg7 的状态。
6.动态刷新:
通过不断切换 judge 状态，交替显示两个不同的数字。
这种动态刷新给观察者的感觉是两个数码管同时显示不同的数字，尽管实际上在任何给定时间点只有一个数码管被激活。

design

```
module mul(input[6:0] bcd_num, input clk, output reg [1:0] scan_select, output reg [7:0] seg7 // 7 段数码管显示数据
);
initial begin seg7 =8'b01111110;jud=0;judge=0;end
reg [3:0] dig_0; // 个位数字的 BCD 编码
reg [3:0] dig_1; // 十位数字的 BCD 编码
reg judge;
reg [7:0] jud;
always @ (posedge clk) begin
jud=jud+1;
end
always @(posedge jud[7])begin
 judge=~judge;
end
always @ (posedge clk) begin
if(bcd_num>=7'b1100011)begin
dig_0 <= 4'b1001; // 取出个位数字的 BCD 编码
dig_1 <= 4'b1001;
end
```

```
else begin
dig_0 <= bcd_num%4'b1010; // 取出个位数字的 BCD 编码
dig_1 <= bcd_num/4'b1010;
 end // 取出十位数字的 BCD 编码
end
// jud下降沿处理
always @ (judge) begin
if (~judge) begin
scan_select<=2'b10;
// 选择扫描第一个数码管
case(dig_0)
// 根据个位数字的 BCD 编码进行七段数码管的显示
4'b0000: seg7 <= 8'b01111110; // 显示"0"到“9”，其他不显示
4'b0001: seg7 <= 8'b00110000;
4'b0010: seg7 <= 8'b01101101;
4'b0011: seg7 <= 8'b01111001;
4'b0100: seg7 <= 8'b00110011;
4'b0101: seg7 <= 8'b01011011;
4'b0110: seg7 <= 8'b01011111;
4'b0111: seg7 <= 8'b01110000;
4'b1000: seg7 <= 8'b01111111;
4'b1001: seg7 <= 8'b01111011;
endcase
end
```

```
else begin
scan_select<=2'b01;
// 选择扫描第二个数码管
case(dig_1)
// 根据十位数字的 BCD 编码进行七段数码管的显示
4'b0000: seg7 <= 8'b01111110; // 显示"0"到“9”，其他不显示
4'b0001: seg7 <= 8'b00110000;
4'b0010: seg7 <= 8'b01101101;
4'b0011: seg7 <= 8'b01111001;
4'b0100: seg7 <= 8'b00110011;
4'b0101: seg7 <= 8'b01011011;
4'b0110: seg7 <= 8'b01011111;
4'b0111: seg7 <= 8'b01110000;
4'b1000: seg7 <= 8'b01111111;
4'b1001: seg7 <= 8'b01111011;
endcase
end

endmodule
```

constrain

```
set_property PACKAGE_PIN AC19 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property PACKAGE_PIN AA7 [get_ports {bcd_num[6]}]
set_property PACKAGE_PIN W6 [get_ports {bcd_num[5]}]
set_property PACKAGE_PIN AB6 [get_ports {bcd_num[4]}]
set_property PACKAGE_PIN AC23 [get_ports {bcd_num[3]}]
set_property PACKAGE_PIN AC22 [get_ports {bcd_num[2]}]
set_property PACKAGE_PIN AD24 [get_ports {bcd_num[1]}]
set_property PACKAGE_PIN AC21 [get_ports {bcd_num[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[0]}]
set_property PACKAGE_PIN D3 [get_ports {scan_select[1]}]
set_property PACKAGE_PIN D25 [get_ports {scan_select[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {scan_select[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {scan_select[1]}]
set_property PACKAGE_PIN C4 [get_ports {seg7 [7]}]
set_property PACKAGE_PIN A2 [get_ports {seg7 [6]}]
set_property PACKAGE_PIN D4 [get_ports {seg7 [5]}]
set_property PACKAGE_PIN E5 [get_ports {seg7 [4]}]
set_property PACKAGE_PIN B4 [get_ports {seg7 [3]}]
set_property PACKAGE_PIN B2 [get_ports {seg7 [2]}]
set_property PACKAGE_PIN E6 [get_ports {seg7 [1]}]
```

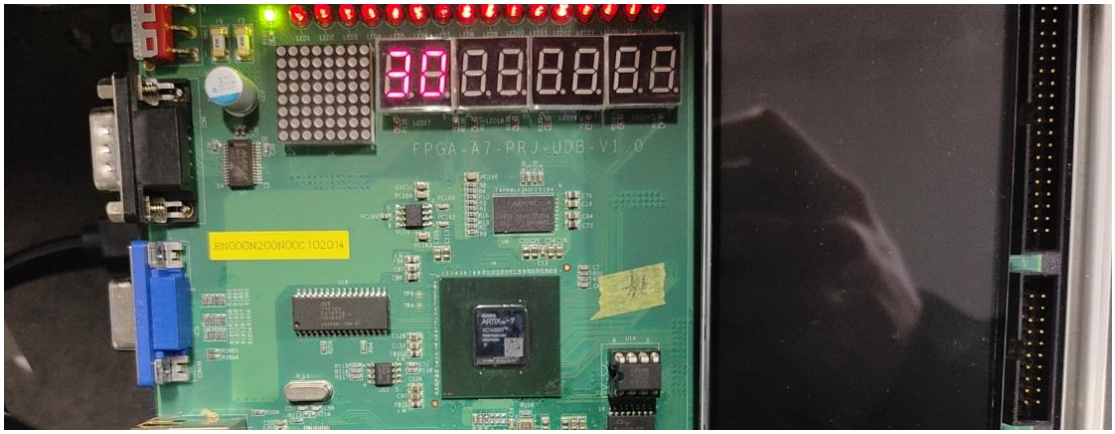
```
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd_num[0]}]
set_property PACKAGE_PIN D3 [get_ports {scan_select[1]}]
set_property PACKAGE_PIN D25 [get_ports {scan_select[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {scan_select[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {scan_select[1]}]
set_property PACKAGE_PIN C4 [get_ports {seg7 [7]}]
set_property PACKAGE_PIN A2 [get_ports {seg7 [6]}]
set_property PACKAGE_PIN D4 [get_ports {seg7 [5]}]
set_property PACKAGE_PIN E5 [get_ports {seg7 [4]}]
set_property PACKAGE_PIN B4 [get_ports {seg7 [3]}]
set_property PACKAGE_PIN B2 [get_ports {seg7 [2]}]
set_property PACKAGE_PIN E6 [get_ports {seg7 [1]}]
set_property PACKAGE_PIN C3 [get_ports {seg7 [0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg7 [7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg7 [6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg7 [5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg7 [4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg7 [3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg7 [2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg7 [1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg7 [0]}]
```

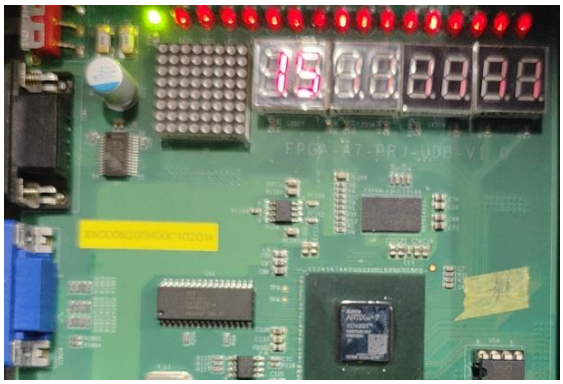
六、关键问题讨论

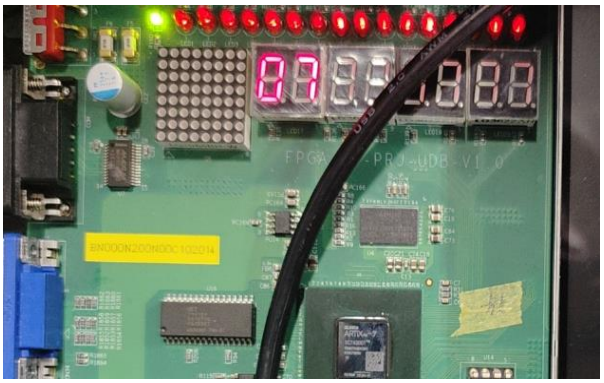
代码如下：

```
module mul (
input[6:0] bcd_num, // 7 位输入，表示要显示的数字
input clk, output reg [1:0] scan_select, // 输出，用于选择当前激活的数码管
output reg [7:0] seg7 // 7 段数码管显示数据 );
initial begin seg7 =8'b01111110; // 初始化 seg7 以显示一个特定数字
jud=0;judge=0;end
reg [3:0] dig_0; // 个位数字的 BCD 编码
reg [3:0] dig_1; // 十位数字的 BCD 编码
reg judge; // 用于切换数码管显示的变量
reg [7:0] jud; // 计数器，用于控制数码管的刷新
// 使用时钟信号更新计数器
always @ (posedge clk) begin
jud=jud+1;
end
// 当计数器的某一位变化时，切换 judge 的状态
always @(posedge jud[7])begin
 judge=~judge;
end
// 时钟信号驱动的处理，用于更新数码管显示的数字
always @ (posedge clk) begin
if(bcd_num>=7'b1100011)begin
dig_0 <= 4'b1001; // 如果输入值大于特定值，固定显示特定数字
dig_1 <= 4'b1001;
end
else begin
dig_0 <= bcd_num%4'b1010;  // 否则，计算个位数字的 BCD 编码
dig_1 <= bcd_num/4'b1010; // 计算十位数字的 BCD 编码
 end // 取出十位数字的 BCD 编码
end
// jud 下降沿处理
always @ (judge) begin
if (~judge) begin
scan_select<=2'b10;
// 选择扫描第一个数码管
case(dig_0)
// 根据个位数字的 BCD 编码进行七段数码管的显示
4'b0000: seg7 <= 8'b01111110; // 显示"0" 到“9”，其他不显示
4'b0001: seg7 <= 8'b00110000;
4'b0010: seg7 <= 8'b01101101;
4'b0011: seg7 <= 8'b01111001;
4'b0100: seg7 <= 8'b00110011;
4'b0101: seg7 <= 8'b01011011;
4'b0110: seg7 <= 8'b01011111;
4'b0111: seg7 <= 8'b01110000;
```

```
4'b1000: seg7 <= 8'b01111111;
4'b1001: seg7 <= 8'b01111011;
endcase
end
else begin
scan_select<=2'b01;
// 选择扫描第二个数码管
case(dig_1)
// 根据十位数字的 BCD 编码进行七段数码管的显示
4'b0000: seg7 <= 8'b01111110; // 显示"0" 到“9”，其他不显示
4'b0001: seg7 <= 8'b00110000;
4'b0010: seg7 <= 8'b01101101;
4'b0011: seg7 <= 8'b01111001;
4'b0100: seg7 <= 8'b00110011;
4'b0101: seg7 <= 8'b01011011;
4'b0110: seg7 <= 8'b01011111;
4'b0111: seg7 <= 8'b01110000;
4'b1000: seg7 <= 8'b01111111;
4'b1001: seg7 <= 8'b01111011;
endcase
end
end endmodule
```







七、总结

这个实验通过 Verilog 编程实现了在单个七段数码管上交替显示两个不同数字的功能。它展示了时序逻辑电路设计的基本原理，包括时钟信号管理、信号同步、以及状态控制。实验提供了对数字电路设计中重要概念的深入理解，特别是在涉及时钟驱动的电路和动态显示系统时。通过这个实验，加深对硬件描述语言的理解，特别是在设计和实现复杂的逻辑功能时。学会了使用多个模块进行集成设计。