**《大规模可编程器件》实验报告**

实验四：变速数字时钟



**目录**

[一、实验目的 1](#_Toc27756)

[二、实验原理 1](#_Toc29187)

[三、实验内容 2](#_Toc7048)

[四、实验代码 2](#_Toc30923)

[五、实验结果 3](#_Toc31102)

[六、 实验心得 9](#_Toc31772)

# 

# 一、实验要求

设计一个变速数字时钟，要求数字时钟的速度有三个档位：

第一个档位为标准数字时钟，每隔1S秒计数器加1；

第二个档位为快速数字时钟，每隔0.1S秒计数器加1；

第三个档位为超快速数字时钟，每隔0.01S秒计数器加1；

三个档位可用按键切换；具备按键清零功能；具有整点报时功能，即在59分59秒时给出指示信息（LED灯亮），持续时间为1s/0.1s/0.01s，指示信号结束的时刻恰好为正点时刻。

说明：给定输入时钟为1kHz。

## 二、设计思路

总体程序由四个模块组成：按键消抖模块、时钟分频模块、数码管驱动选择模块、数码管计数显示模块。程序前端的按键消抖模块使用一个时钟分频模块将系统50MHz时钟分频获得5KHz时钟，用于按键消抖模块计数时钟使用，模块输出经消抖后的按键信号至时钟分频模块与数码管驱动选择模块。系统使用时钟分频模块对系统时钟分频，用于时钟计数，其频率由经消抖后的按键信号选择；按键经消抖模块输入至时钟分频模块，切换分频倍数。在数码管驱动模块中，计数器的计数值转化为时、分、秒的数值，其中，计数器计数来源由数码管驱动选择模块选择来源于外部按键输入或者时钟分频模块输出。程序流程如图1所示。

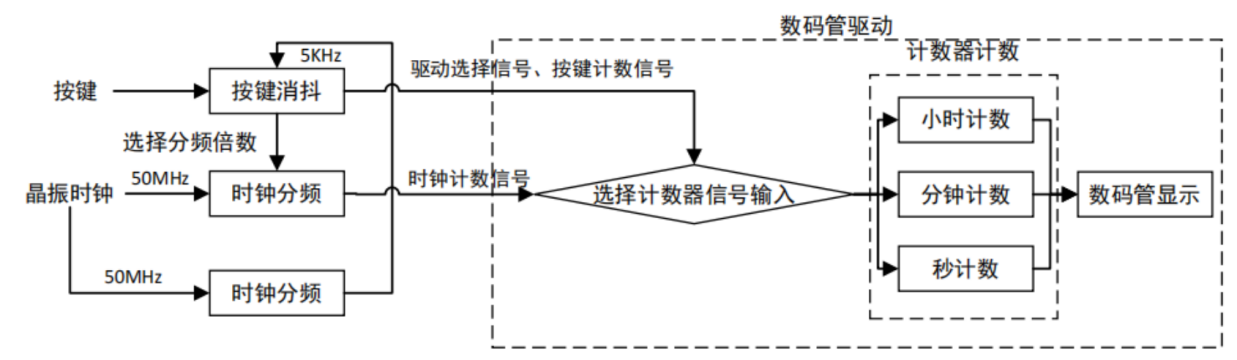


图 1 程序流程图

# 三、实验模块

1. 按键消抖模块：对按键进行消抖处理，使按键输入更加稳定。

2. 时钟分频模块：对50MHz时钟进行分频处理，输出1Hz、10Hz、100Hz时钟，用于后续数码管显示计数。

3. 数码管驱动选择模块：采用二选一多路选择器，通过按键输入选取数码管计数模块的计数输入脉冲，以达到时钟计时与自动调时的功能。

4. 数码管计数显示模块：将计数脉冲计数值累加进入模块内部寄存器处，随后实时显示时分秒数值至数码管。

# 四、实验代码

|  |
| --- |
| 顶层文件设计： |
| module clock(  input wire clk\_50MHz, //50MHz信号输入  input wire key\_A, //时钟速度调节键1  input wire key\_B, //时钟速度调节键2  input wire time\_switch, //时钟与按键输入选择按键  input wire time\_adj, //按键输入信号按键  input wire nRST, //复位信号  output [6:0]HEX0, //数码管显示  output [6:0]HEX1,output [6:0]HEX2,  output [6:0]HEX3,  output [6:0]HEX4,  output [6:0]HEX5,  output wire LED //LED输出  );  wire key\_key\_A; //按键消抖处理后的输出  wire key\_key\_B;  wire key\_time\_adj;  wire key\_time\_switch;  wire time\_set; //决定数码管显示的数字前进是时钟加一还是按键加一  wire time\_devide; //时钟分频模块，输出1Hz,10Hz,100Hz信号  wire [3:0]disp\_dat\_0; //数码管值存储变量  wire [3:0]disp\_dat\_1;  wire [3:0]disp\_dat\_2;  wire [3:0]disp\_dat\_3;  wire [3:0]disp\_dat\_4;  wire [3:0]disp\_dat\_5; |

|  |
| --- |
| 按键消抖模块： |
| module eliminate\_jitters(  input clk, // 50MHz时钟信号输入  input key, // 原始的按键输入  input nRST, // 复位信号输入  output clk\_50Hz, // 分频后的时钟信号输出  output reg key\_o // 经过消抖后的按键输出  );  reg [20:0]cnt; //定义一个21位宽的计数器  reg key\_d1; //定义一个位宽为1位的寄存器  reg key\_d2; //定义一个位宽为1位的寄存器 |
| 时钟分频模块： |
| module time\_devide\_timer(  input clk, //50MHz时钟  input switch\_1, //切换频率  input switch\_2,  output clk\_timer, //分频时钟输出  input nRST //复位信号  );  reg [24:0]count\_1 = 25'd0;//计数器  reg [24:0]count\_10 = 25'd0;  reg [24:0]count\_100 = 25'd0;  reg clk\_1\_r = 0;//控制电平  reg clk\_10\_r = 0;  reg clk\_100\_r = 0;  reg clk\_timer\_r =0;  assign clk\_timer = clk\_timer\_r;  ////////////////////////////////////////代码块  always @(posedge clk)//0.5s反转一次  begin  if(!nRST) // 时分秒复位  begin  count\_1 <= 25'd0;  end  else  begin  if(count\_1 == 25'd24999999)  begin  count\_1 <= 25'd0;  clk\_1\_r <= ~clk\_1\_r;  end  else  count\_1 <= count\_1 + 1;  end  end  always @(posedge clk)//0.05s反转一次  begin  if(!nRST) // 时分秒复位  begin  count\_10 <= 25'd0;  end  else  begin  if(count\_10 == 25'd2499999)  begin  count\_10 <= 25'd0;  clk\_10\_r <= ~clk\_10\_r;  end  else  count\_10 <= count\_10 + 1;  end  end  always @(posedge clk)//0.005s反转一次  begin  if(!nRST) // 时分秒复位  begin  count\_100 <= 25'd0;  end  else  begin  if(count\_100 == 25'd249999)  begin  count\_100 <= 25'd0;  clk\_100\_r <= ~clk\_100\_r;  end  else  count\_100 <= count\_100 + 1;  end;  end  ////////////////////////////////////////  always @(clk\_1\_r or clk\_10\_r or clk\_100\_r or switch\_1 or switch\_2)//选择输出信号  begin  case({switch\_1,switch\_2})  2'b00 :clk\_timer\_r <= clk\_1\_r;  2'b01 :clk\_timer\_r <= clk\_10\_r;  2'b11 :clk\_timer\_r <= clk\_100\_r;  default:clk\_timer\_r <= clk\_1\_r;  endcase  end  endmodule |
| 数码管驱动选择模块： |
| module key\_time(  input key\_time\_adj, //消除抖动后按键输入,按键输入  input key\_time\_switch, //消除抖动后按键输入,决定数字  前进是时钟还是按键  input key\_clk\_timer, //消除抖动后按键输入,时间输入  output key\_o //经过选择后的按键输出  );  assign key\_o=(key\_time\_switch ? key\_time\_adj:key\_clk\_timer);  endmodule |
| 数码管计数显示模块： |
| module digital\_show(  input clk\_timer, // 计数时钟信号输入  input nRST, // 复位信号输入  output [3:0]disp\_dat\_0, // 显示数值  output [3:0]disp\_dat\_1,  output [3:0]disp\_dat\_2,  output [3:0]disp\_dat\_3,  output [3:0]disp\_dat\_4,  output [3:0]disp\_dat\_5,  output LED //LED信号输出  );  reg [4:0]hour; //0-23小时  reg [5:0]minute; //0-59分钟  reg [5:0]second; //0-59秒  // 时钟计数  always @(posedge clk\_timer or negedge nRST)  begin  if(!nRST) // 时分秒复位  begin  hour <= 5'd0;  minute <= 6'd0;  second <= 6'd0;  end  else  begin  second <= second + 1; // 每个时钟信号到来时，秒+1  // 如果秒到达59，对其清零且分钟+1  if(second == 6'd59)  begin second <= 6'd0; minute <= minute+1'b1; end  // 如果分钟到达60，对其清零且小时+1  if(minute == 6'd59 & second == 6'd59) begin minute <= 6'd0; hour  <= hour + 1'b1; end  // 如果小时到达24，对其清零  if(hour == 5'd24) begin hour <= 5'd0; end  end  end  assign LED = (second == 6'd59 & minute == 6'd59)?1'b1:1'b0;  assign disp\_dat\_0 = second % 10; // 秒个位  assign disp\_dat\_1 = second / 10; // 秒十位  assign disp\_dat\_2 = minute % 10; // 分个位  assign disp\_dat\_3 = minute / 10; // 分十位  assign disp\_dat\_4 = hour % 10; // 时个位  assign disp\_dat\_5 = hour / 10; // 时十位  endmodule  module data\_show(  input [3:0]disp\_dat, // 显示数值  output reg [6:0]disp\_show  );  always @(disp\_dat)  begin  case(disp\_dat) // 将显示的数值映射为对应段码  4'h0:disp\_show = 7'b100\_0000; // 0  4'h1:disp\_show = 7'b111\_1001; // 1  4'h2:disp\_show = 7'b010\_0100; // 2  4'h3:disp\_show = 7'b011\_0000; // 3  4'h4:disp\_show = 7'b001\_1001; // 4  4'h5:disp\_show = 7'b001\_0010; // 5  4'h6:disp\_show = 7'b000\_0010; // 6  4'h7:disp\_show = 7'b111\_1000; // 7  4'h8:disp\_show = 7'b000\_0000; // 8  4'h9:disp\_show = 7'b001\_0000; // 9  default:disp\_show = 7'b100\_0000;  endcase  end  endmodule |

# 五、实验结果

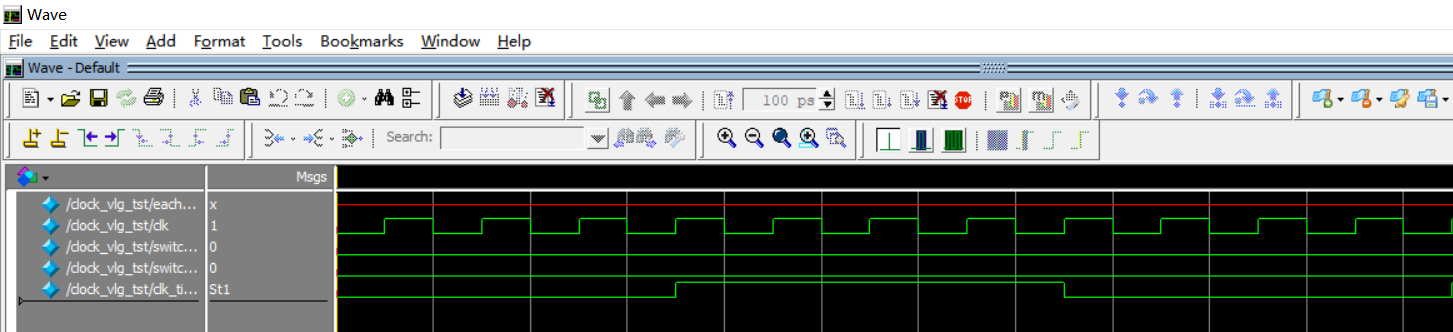


图 2 结果图

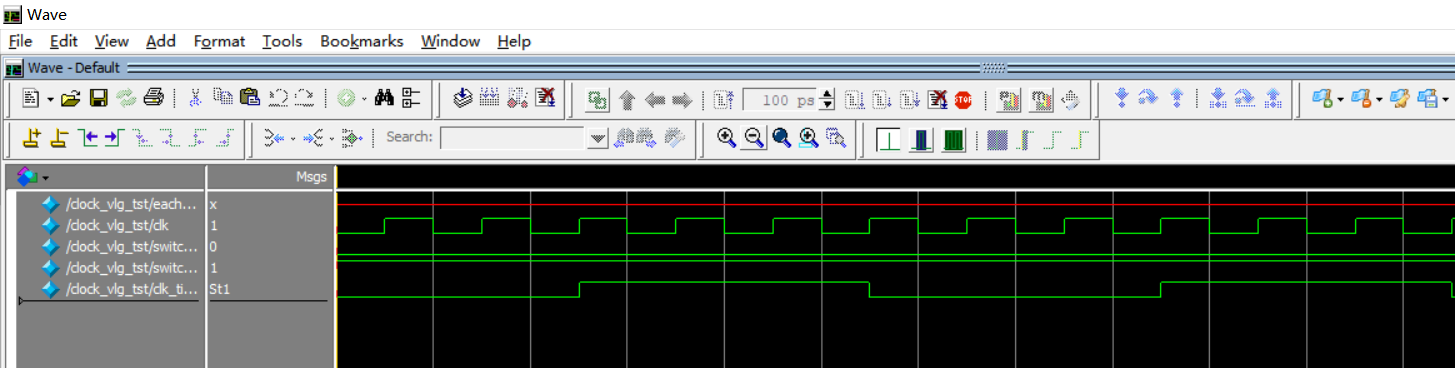


图 3 结果图

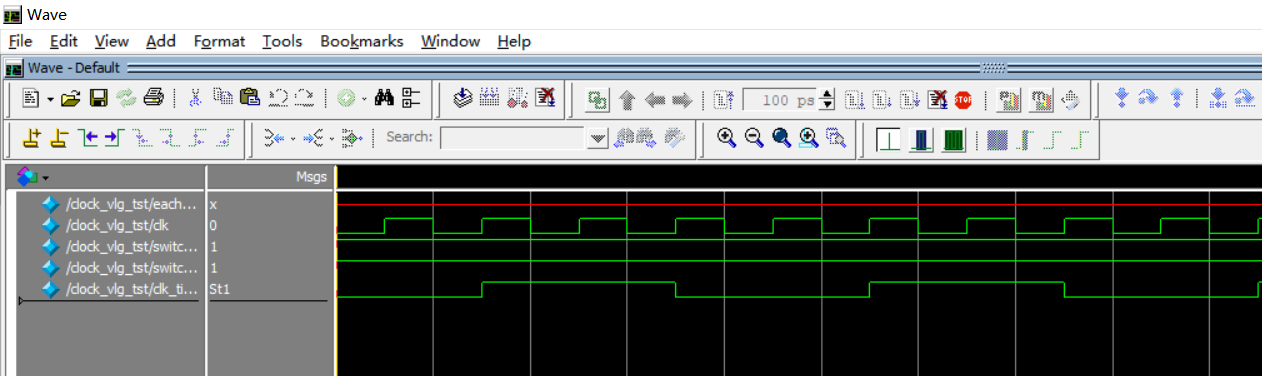


图 4 结果图

从波形图中可以看出，不同模式下，模块输出频率将会改变，并且符合预先要求，可见该模块的设计满足要求。

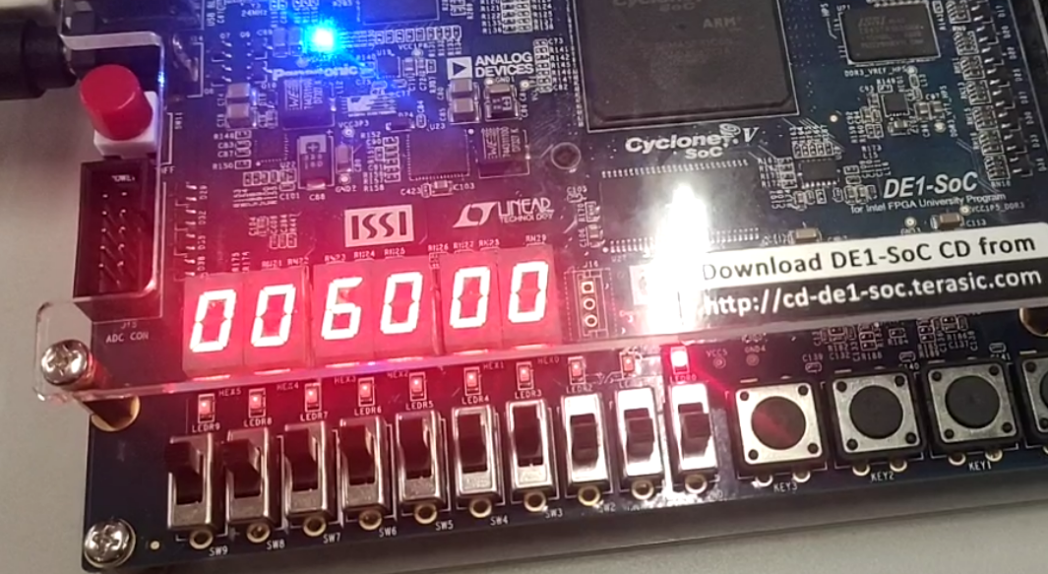


图 5 整点报时效果图

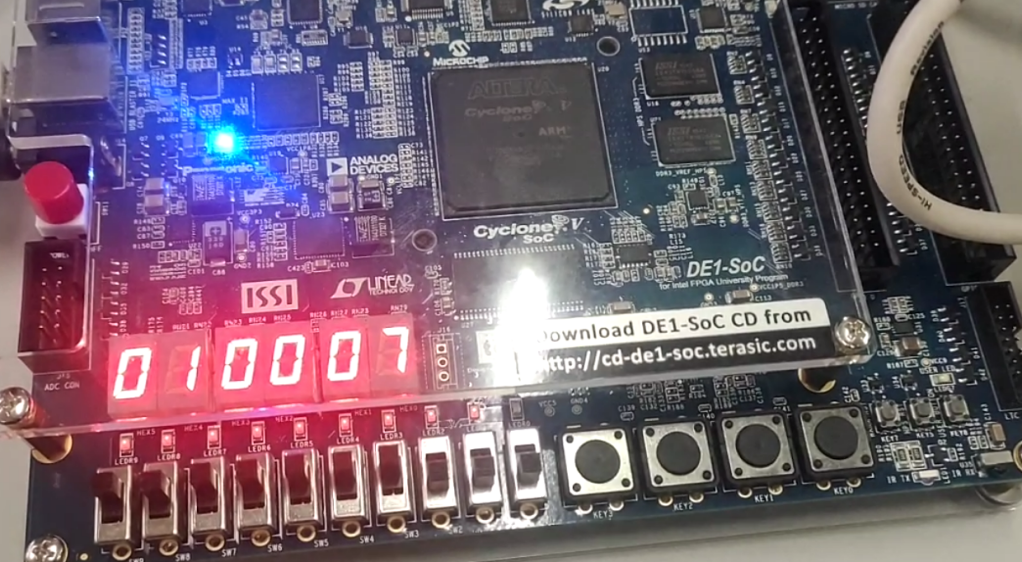


图 6 小时调节

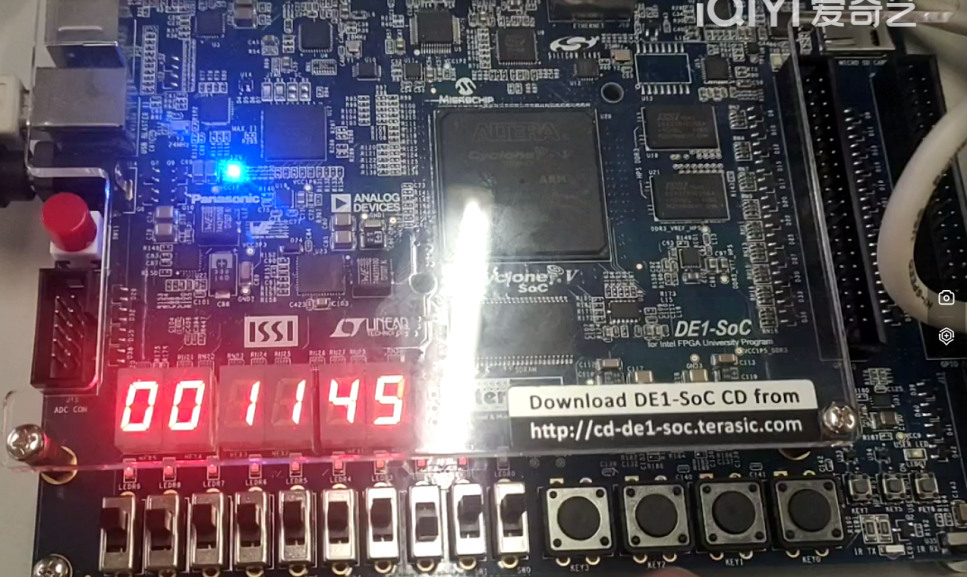


图 7 分钟调节

# 实验心得

在完成该程序的过程中，我深刻了解了分模块设计的优点以及缺点，例如优点有方便设计，思路清晰，并且一个错误一般不会带动其他错误。缺点有接口设置比较复杂等。此次程序设计要考虑的东西比较多，在犯错的过程中通过解决一些问题也掌握了一些必要的知识例如不能在多个模块里面对一个寄存器进行赋值，不能多次声明一个寄存器等。经过此次学习收获良多，深刻地感受到了软件和硬件控制之间的关系。

此次实验比较成功，但是因为后期出现了一些bug，所以完成的时间比较晚，动态的显示代码量比较大，不过在看过动态显示的范例代码之后，的确感觉是动态显示比较好。

通过此次实验掌握了基本的quartus编程以及modelsim仿真，自己敲代码也出现了大大小小的错误，不过在自己百度查阅资料以及通过老师和同学的帮助已经解决了问题，得到了准确的结果。