**《数字逻辑》实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓名** | **潘兴烨** | | **年级** | | **2022级** |
| **学号** | **20220589** | | **专业、班级** | | **计算机科学与技术 卓越01** |
| **实验名称** | **实验七 数字钟设计** | | | | |
| **实验时间** | **2023.10.26** | **实验地点** | | **DS1410** | |
| **实验成绩** |  | **实验性质** | | **□验证性 ☑设计性 □综合性** | |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  评语：  评价教师签名（电子签名）： | | | | | |
| 一、实验目的  通过实验，充分理解和掌握数字钟的原理，理解多进制计时器和数码管的工作原理，学会设计自己的数字钟。 | | | | | |
| 二、实验项目内容  1、实现一个六十进制数字时钟，秒到 60 则归零重加，同时让分加 1，分加到 60归零重加，并让小时加 1，小时加到 24 归零重加。  2、设计用于数字钟的计数器（60 进制，24 进制）并编写仿真文件，观察其  波形图并验证期正确性。  3、将设计好的计数器连接数码管并显示数值，数码管 1,0 显示秒值，数码管2,3 显示分值，数码管 4,5 显示小时（可以十六进制形式显示在 led 灯上）。  4、综合、实验、生成 bit 流，下载到开发板进行验证 | | | | | |
| 三、实验设计  整体分为四个模块，一是秒表计数器，用于统计1秒的时间；二是时分秒的进位和调时模块；三是数码管的显示模块，用于数码管的选择和数码管的数字显示。四是顶层模块，用于串接三个子模块和接收输入，传出输出。 | | | | | |
| 四、实验过程或算法  在是秒钟效应模块中，采用了计数器作为主体代码，以此产生一秒一次的信号输出，代码如下：module clk\_1s(  input clk,  input rstn,  output flag\_1s  );  parameter num\_1s = 100\_000\_000;  reg [26:0]cnt\_1s;  always@(posedge clk or negedge rstn)  begin  if(!rstn) begin  cnt\_1s <= 0;  end  else  begin  if(cnt\_1s == num\_1s) cnt\_1s <= 0;  else cnt\_1s <= cnt\_1s+1;  end  end  assign flag\_1s = (cnt\_1s == num\_1s) ? 1 : 0;  Endmodule  产生的输出信号又被第二个进位模块接受，在这个模块中运用了if else语句处理满60进一，满24进一的问题，区别于普通的实验要求，我们小组添加了调时功能，可以单独对时分秒进行一秒一次的增或减，这无疑使模块的编写更加繁琐，并在每次时钟沿到来的时候检索当前的时分秒的个位十位作为输出给数码管显示，模块代码如下：  module clock(  input clk,  input rstn,  input flag\_1s,  input stop,  input s0,  input s1,  input s2,  input s3,  input s4,  input s5,  output reg[3:0] data0,  output reg[3:0] data1,  output reg[3:0] data2,  output reg[3:0] data3,  output reg[3:0] data4,  output reg[3:0] data5  );  reg [5:0] second;  reg [5:0] minute;  reg [4:0] hour;  always@(posedge clk or negedge rstn)  begin  if(!rstn) begin  second <= 0;  end  else if(stop)  begin  if(s0)  begin  if(second==59&&flag\_1s) second <= 0;  else if(flag\_1s) second <= second + 1;  end  else if(s1)  begin  if(second==0&&flag\_1s) second <= 59;  else if(flag\_1s) second <= second - 1;  end  else second<=second;  end  else begin  if(second==59&&flag\_1s)begin  second <= 0;  end else if(flag\_1s)begin  second = second + 1;  end else begin  second <= second;  end  end  end  always@(posedge clk or negedge rstn)  begin  if(!rstn) minute <= 0;  else if(stop)  begin  if(s2)  begin  if(minute==59&&flag\_1s) minute <= 0;  else if(flag\_1s) minute <= minute + 1;  end  else if(s3)  begin  if(minute==0&&flag\_1s) minute <= 59;  else if(flag\_1s) minute <= minute - 1;  end  else minute<=minute;  end  else  begin  if(minute==59&&second==59&&flag\_1s) minute <= 0;  else if(second==59&&flag\_1s) minute <= minute + 1;  else minute <= minute;  end  end  always@(posedge clk or negedge rstn)  begin  if(!rstn) hour <= 0;  else if(stop)  begin  if(s4)  begin  if(hour==23&&flag\_1s) hour <= 0;  else if(flag\_1s) hour <= hour + 1;  end  else if(s5)  begin  if(hour==0&&flag\_1s) hour <= 23;  else if(flag\_1s) hour<= hour - 1;  end  else hour<=hour;  end  else  begin  if(hour==23&&minute==59&&second==59&&flag\_1s) hour<=0;  else if(minute==59&&second==59&&flag\_1s) hour <= hour + 1;  else hour <= hour;  end  end  always@(posedge clk or negedge rstn)  begin  if(!rstn) begin  data0 <= 0;  data1 <= 0;  data2 <= 0;  data3 <= 0;  data4 <= 0;  data5 <= 0;  end else begin  data0 <= hour/10;  data1 <= hour%10;  data2 <= minute/10;  data3 <= minute%10;  data4 <= second/10;  data5 <= second%10;  end  end  Endmodule  数码管显示模块是实验过程中最难的一部分，因为数码管串接在一起，一次只能点亮一个数码管，所以需要对四个数码管在很短的时间内不断切换，以此达到持续点亮的显示效果。所以我们设计了六个不同的状态，一一对应四个数码管和两串led灯，切换不同的状态也就切换了不同的数码管和数据导入，对变化的六个数据进行动态扫描。在这之前我们对影响显示的时钟信号进行了分频，由100mHz分到了1000Hz，具体代码如下：  module digital(  input clk,  input rstn,  input [3:0]data0,  input [3:0]data1,  input [3:0]data2,  input [3:0]data3,  input [3:0]data4,  input [3:0]data5,  output reg[7:0]seg,  output reg[3:0]sel,  output reg[3:0]hours1,  output reg[3:0]hours2  );  reg[15:0]cnl;  reg clk1k;  always@(posedge clk or negedge rstn)  begin  if(!rstn)  begin  cnl<=0;  clk1k<=0;  end  else if(cnl>=24999)  begin  clk1k<=!clk1k;  cnl<=0;  end  else cnl<=cnl+1;  end  reg [3:0]tub;  reg [2:0]state;  always@(posedge clk1k or negedge rstn)  begin  if(!rstn)  begin  tub<=0;  state<=0;  sel<=0;  hours1<=0;  hours2<=0;  end  else  begin  case(state)  0:begin hours1<=data0;sel<=4'b1111;state<=1;end  1:begin hours2<=data1;sel<=4'b1111;state<=2;end  2:begin tub<=data2;sel<=4'b0111;state<=3;end  3:begin tub<=data3;sel<=4'b1011;state<=4;end  4:begin tub<=data4;sel<=4'b1101;state<=5;end  5:begin tub<=data5;sel<=4'b1110;state<=0;end  default:state<=0;  endcase  end  end  always@(\*)  if(!rstn)  seg<=8'b1100\_0000;  else  case(tub)  0:seg<=8'b1100\_0000;  1:seg<=8'b1111\_1001;  2:seg<=8'b1010\_0100;  3:seg<=8'b1011\_0000;  4:seg<=8'b1001\_1001;  5:seg<=8'b1001\_0010;  6:seg<=8'b1000\_0010;  7:seg<=8'b1111\_1000;  8:seg<=8'b1000\_0000;  9:seg<=8'b1001\_0000;  default:seg<=8'b1100\_0000;  endcase  Endmodule  最后设计了顶层模块起到串接作用，代码如下  module top(  input clk,  input rstn,  input stop,  input s0,  input s1,  input s2,  input s3,  input s4,  input s5,  output [7:0]seg,//输出用于数码管数字的显示  output [3:0]sel,//输出控制六个数码管的显示  output [3:0]hours1,//输出时的  output [3:0]hours2  );  wire flag\_1s;  wire [3:0]data0;  wire [3:0]data1;  wire [3:0]data2;  wire [3:0]data3;  wire [3:0]data4;  wire [3:0]data5;  clk\_1s clk\_1s\_u(  .clk(clk),  .rstn(rstn),  .flag\_1s(flag\_1s)  );  clock clock\_u(  .clk(clk),  .rstn(rstn),  .stop(stop),  .s0(s0),  .s1(s1),  .s2(s2),  .s3(s3),  .s4(s4),  .s5(s5),  .flag\_1s(flag\_1s),  .data0(data0),  .data1(data1),  .data2(data2),  .data3(data3),  .data4(data4),  .data5(data5)  );  digital digital\_u(  .clk(clk),  .rstn(rstn),  .data0(data0),  .data1(data1),  .data2(data2),  .data3(data3),  .data4(data4),  .data5(data5),  .seg(seg),  .sel(sel),  .hours1(hours1),  .hours2(hours2)  );  Endmodule  design文件夹显示如下：  屏幕截图 2023-10-30 235758  完成了设计文件之后就是对针脚的绑定： 约束文件的编写因为有数码管所以需要去查表进行约束，采用图像化约束会简单一些，并且需要查表完成  屏幕截图 2023-10-31 000057  屏幕截图 2023-10-31 000102  代码如下：  set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[7]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[6]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[5]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[4]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {sel[5]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {sel[4]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {sel[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {sel[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {sel[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {sel[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports clk]  set\_property IOSTANDARD LVCMOS33 [get\_ports rstn]  set\_property PACKAGE\_PIN W5 [get\_ports clk]  set\_property PACKAGE\_PIN R2 [get\_ports rstn]  set\_property PACKAGE\_PIN U2 [get\_ports {sel[0]}]  set\_property PACKAGE\_PIN U4 [get\_ports {sel[1]}]  set\_property PACKAGE\_PIN V4 [get\_ports {sel[2]}]  set\_property PACKAGE\_PIN W4 [get\_ports {sel[3]}]  set\_property PACKAGE\_PIN V7 [get\_ports {seg[7]}]  set\_property PACKAGE\_PIN W7 [get\_ports {seg[0]}]  set\_property PACKAGE\_PIN W6 [get\_ports {seg[1]}]  set\_property PACKAGE\_PIN U8 [get\_ports {seg[2]}]  set\_property PACKAGE\_PIN V8 [get\_ports {seg[3]}]  set\_property PACKAGE\_PIN U5 [get\_ports {seg[4]}]  set\_property PACKAGE\_PIN V5 [get\_ports {seg[5]}]  set\_property PACKAGE\_PIN U7 [get\_ports {seg[6]}]    set\_property PACKAGE\_PIN P3 [get\_ports {hours1[2]}]  set\_property PACKAGE\_PIN U3 [get\_ports {hours1[1]}]  set\_property PACKAGE\_PIN W3 [get\_ports {hours1[0]}]  set\_property PACKAGE\_PIN V3 [get\_ports {hours2[2]}]  set\_property PACKAGE\_PIN V13 [get\_ports {hours2[1]}]  set\_property PACKAGE\_PIN V14 [get\_ports {hours2[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {hours1[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {hours1[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {hours1[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {hours2[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {hours2[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {hours2[2]}]  在约束文件编写后，对程序进行了上板验证和操作，效果图如下：  63E1E92CAE58D448C38651B27396BFF3  73442F3E59FA95F95851270D29DBE2D3  4A3005AAA468D5933F4FDFD09405DBBE  实验效果图从录制的效果视频中截得。 | | | | | |
| 五、实验过程中遇到的问题及解决情况  在实验过程中，主要是数码管显示模块中的分频部分出现问题，一开始没有进行分频，数码管显示过亮，甚至无法看清数字，第一次分频后又出现了只有一个数码管亮的问题，这是由于刷新频率太低导致的，在调频到了1000hz后就解决了问题。 | | | | | |
| 六、实验结果及分析和（或）源程序调试过程  在尝试生成比特流的过程中遇到了报错。经过分析，我们想到可能是由于某些端口未被指派造成的。因此，我们在综合过程结束后，选择手动选择端口的方式来分配引脚。这样，我们能够更加细致地检查每个端口是否都被正确地指派，然后发现我们遗漏了seg[0]。  在将代码部署到base3上执行之后，发现数码管的显示效果过于频繁，。经过仔细的分析，我们认识到这是因为刷新率设置得太高导致的。为了解决这个问题，添加了一个分频功能，以降低刷新率。然而，由于参数设置不当，刷新率反而变得过低。为了解决这个问题，我们进行了多次参数调整，最终将刷新率设置为1000hz，从而得到了清晰且不过量的显示效果。  七、小组分工情况说明  由潘兴烨完成四个代码块的编写  由张斯宇和曹泽阳完成实验报告的编写和程序的调试以及上板测试优化 | | | | | |