**杭州电子科技大学计算机学院**

**实验报告**

实验项目：交通灯设计

课程名称：数字电路课程设计

姓名：易永豪 学号：17052247 同组姓名：郭佳炜 学号：17052246 实验位置（机号）：91

实验日期： 2019.11.18 指导教师：戴钧

|  |  |
| --- | --- |
| 实验  目的 | （1）学习交通灯设计的原理与设计方法  （2）掌握灵活运用Verlog HDL语言进行各种描述与建模的技巧和方法 |
| 实验  设备 | 1.PC机 一台  2.FPGA实验板 一块 |
| 实验  内容  和  原理  （  原理  图、  原理  说明、  算法、  程序、  步骤、  和方  法等  ）  实验  代码 | 本实验要求设计一个十字路口的交通灯系统.  系统输入:时钟CLK、系统复位CLR,交通灯正常启动start,主干道阻塞不通行stopa,次干道阻塞不通行stopb.系统暂停 pause 按钮用于暂停系统,便于观察实验结果。  系统输出:在数码管(高2片)上显示主干道读秒倒计时,在数码管(低2片)上显示次干道读秒倒计时,6个LED灯分别显示主干道和次干道的红绿黄灯。  系统的运行规则是:  (1)按下复位按钮,主干道和次干道的红灯皆亮,且主干道和次干道都不显示读秒,4  个数码管显示“-------”;  (2)当主干道阻塞时,主干道和次干道都不显示读秒,数码管无显示,主干道的红灯  亮,次干道绿灯长亮:  (3)当次干道阻塞时,主干道和次干道都不显示读秒,数码管无显示,次干道的红灯  亮,主干道绿灯长亮:  (4)当按下启动按钮,系统启动;  (5)交通灯正常工作时,主干道和次干道都显示读秒,系统在四个状态之间切换,这4  个状态分别是:statel:主干道通行次干道禁  表7.2 交通灯系统状态转换表  行,主干道绿灯亮,次干道红灯亮,时长35  状态 主干道次干道 时间(秒)  秒:state2:主干道黄灯亮,次干道红灯亮,  statel 绿灯亮红灯亮35  时长5秒;state3:次干道通行,主干道禁行,state2 黄灯亮红灯亮5s主干道红灯亮,次干道绿灯亮,时长25秒;state3 红灯亮绿灯亮25state4:主干道红灯亮,次干道黄灯亮,时长state4 红灯亮黄灯亮55秒.状态转换表如表7.2所示。  (6)当暂停按钮按下时,主干道和次干道计数暂停,红绿黄灯保持不变,观察结果。  松开暂停按钮,系统继续运行。  表一 交通灯系统状态转换表   |  |  |  |  | | --- | --- | --- | --- | | 状态哦】 | 主干道 | 次干道 | 时间 | | State1 | 绿灯亮 | 红灯亮 | 35 | | State2 | 黄灯亮 | 红灯亮 | 5 | | State3 | 红灯亮 | 绿灯亮 | 25 | | State4 | 红灯亮 | 黄灯亮 | 5 |   **可逆计数器实验代码：**  module top(  input CLK\_100M,  input CLR,  input start,  input stopA,  input stopB,  input pause,    output wire [2:0]LEDA,  output wire [2:0]LEDB,  output wire [3:0]An,  output wire [7:0]Seg  );    parameter time\_200ms = 10000000;    wire CLK\_5HZ;  wire [3:0]D0,D1,D2,D3;  wire [7:0]Disp\_buf0,Disp\_buf1,Disp\_buf2,Disp\_buf3;    fenpin fp1(.n(time\_200ms),.rst(CLR),.CLK\_in(CLK\_100M),.CLK\_out(CLK\_5HZ));  kongzhi kz(CLK\_5HZ,CLR,stopA,stopB,start,pause,LEDA,LEDB,D0,D1,D2,D3);  zhuanhuan zh1(D0,Disp\_buf0);  zhuanhuan zh2(D1,Disp\_buf1);  zhuanhuan zh3(D2,Disp\_buf2);  zhuanhuan zh4(D3,Disp\_buf3);  xianshi xs(CLK\_100M,Disp\_buf0,Disp\_buf1,Disp\_buf2,Disp\_buf3,An,Seg);  endmodule  module fenpin(  input [32:1]n,  input rst,  input CLK\_in,  output reg CLK\_out  );  reg [32:1] counter;  always@(posedge rst or posedge CLK\_in)  begin  if(rst==1)  begin  counter <= 32'd0;  CLK\_out <= 1'b0;  end  else if(counter==n)  begin  CLK\_out <= ~CLK\_out;  counter <= 32'd0;  end  else  counter <= counter+1'd1;  end  endmodule  module kongzhi(  input CLK,  input CLR,  input stopA,  input stopB,  input start,  input pause,  output reg [2:0]LEDA,  output reg [2:0]LEDB,  output reg [3:0]D0,  output reg [3:0]D1,  output reg [3:0]D2,  output reg [3:0]D3  );    reg [5:0]secondA;  reg [5:0]secondB;  reg [1:0]stateA;  reg [1:0]stateB;  parameter A1 = 2'b00;  parameter A2 = 2'b01;  parameter A3 = 2'b10;  parameter B1 = 2'b00;  parameter B2 = 2'b01;  parameter B3 = 2'b10;  parameter A4 = 35;  parameter A5 = 5;  parameter A6 = 30;  parameter B4 = 40;  parameter B5 = 25;  parameter B6 = 5;    initial  begin  secondA = A4;  secondB = B4;  stateA = A3;  stateB = B1;  LEDA <= 3'b000;  LEDB <= 3'b000;  D0 <= 4'b1111;  D1 <= 4'b1111;  D2 <= 4'b1111;  D3 <= 4'b1111;  end  always@(posedge CLK or posedge CLR)  begin  if(pause==1)  begin  LEDA <= LEDA;  LEDB <= LEDB;  D0 <= D0;  D1 <= D1;  D2 <= D2;  D3 <= D3;  end  else if(CLR==1)  begin  LEDA <= 3'b100;  LEDB <= 3'b100;  D0 <= 4'b1101;  D1 <= 4'b1101;  D2 <= 4'b1101;  D3 <= 4'b1101;  end  else if(stopA==1)  begin  LEDA <= 3'b100;  LEDB <= 3'b001;  D0 <= 4'b1111;  D1 <= 4'b1111;  D2 <= 4'b1111;  D3 <= 4'b1111;  end  else if(stopB==1)  begin  LEDA <= 3'b001;  LEDB <= 3'b100;  D0 <= 4'b1111;  D1 <= 4'b1111;  D2 <= 4'b1111;  D3 <= 4'b1111;  end  else if(start==1)  begin  case(stateA)  A1:LEDA <= 3'b100;  A2:LEDA <= 3'b010;  A3:LEDA <= 3'b001;  endcase  case(stateB)  B1:LEDB <= 3'b100;  B2:LEDB <= 3'b010;  B3:LEDB <= 3'b001;  endcase  D0 <= secondA / 10;  D1 <= secondA % 10;  D2 <= secondB / 10;  D3 <= secondB % 10;  secondA = secondA-1;  secondB = secondB-1;  if(secondA==0)  begin  case(stateA)  A1:secondA = A4;  A2:secondA = A6;  A3:secondA = A5;  endcase  case(stateA)  A1:stateA = A3;  A2:stateA = A1;  A3:stateA = A2;  endcase  end  if(secondB==0)  begin  case(stateB)  B1:secondB = B5;  B2:secondB = B4;  B3:secondB = B6;  endcase  case(stateB)  B1:stateB = B3;  B2:stateB = B1;  B3:stateB = B2;  endcase  end  end  end  endmodule  module zhuanhuan(  input [3:0]D\_in,  output reg [7:0]D\_out  );  always@(D\_in)  begin  case(D\_in)  0: D\_out = 8'b00000011;  1: D\_out = 8'b10011111;  2: D\_out = 8'b00100101;  3: D\_out = 8'b00001101;  4: D\_out = 8'b10011001;  5: D\_out = 8'b01001001;  6: D\_out = 8'b01000001;  7: D\_out = 8'b00011111;  8: D\_out = 8'b00000001;  9: D\_out = 8'b00001001;  13: D\_out = 8'b11111101;  default:D\_out = 8'b11111111;  endcase  end  endmodule  module xianshi(  input CLK,  input [7:0] Dispbuf1,  input [7:0] Dispbuf2,  input [7:0] Dispbuf3,  input [7:0] Dispbuf4,    output reg [3:0]An,  output reg [7:0]Seg  );    reg [2:0]state;    initial  begin  state = 0;  end    always@(posedge CLK)  begin  case(state)  0:  begin  An <= 4'b1000;  Seg <= Dispbuf1;  end  1:  begin  An <= 4'b1001;  Seg <= 8'b11111111;  end  2:  begin  An <= 4'b1001;  Seg <= Dispbuf2;  end  3:  begin  An <= 4'b1010;  Seg <= 8'b11111111;  end  4:  begin  An <= 4'b1010;  Seg <= Dispbuf3;  end  5:  begin  An <= 4'b1011;  Seg <= 8'b11111111;  end  6:  begin  An <= 4'b1011;  Seg <= Dispbuf4;  end  7:  begin  An <= 4'b1000;  Seg <= 8'b11111111;  end  endcase  case(state)  0:state = 1;  1:state = 2;  2:state = 3;  3:state = 4;  4:state = 5;  5:state = 6;  6:state = 7;  7:state = 0;  endcase  end  endmodule |
| 生成  的电  路图  (可以截图） | **可逆计数器模块**    **内部电路**    **移位寄存器模块** |
| 调试过程  中遇到的问题与解决方法 | 数码管闪烁过快，使用分频来解决。 |
| 结论  、  结果 | 结论：  功能完全符合。 |
| 实验  心得  与  小结 | 通过本次实验,了解了可逆计数器的工作原理,对二进制加减法有了更深刻的认识。 |
| 教师  评议 | 实验报告评议：  实验报告规范/比较规范/不规范，原理叙述清晰/比较清晰/模糊，实验过程表达清楚/比较清楚/一般，测试数据完整/比较完整/不完整。图表规范/比较规范/不规范。数据记录完整，调试能力强（良好，一般，差），代码结构合理/比较合理/不合理，变量符号定义合理/比较合理/不合理，实验体会深刻/一般。 |
| 成绩： A /B/C/D/E 指导教师签名： |