**《数字逻辑》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **姓名** | | 黄婧婧，蔡欣彤 | | **年级** | | 2020级 |
| **学号** | | 20204237，20204187 | | **专业、班级** | | 计算机科学与技术02班，信息安全01班 |
| **实验名称** | 16.米里状态机检测器 | | | | | |
| **实验时间** | **2021.12.1** | | **实验地点** | | **D1410** | |
| **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，掌握有限状态机的设计方法，并用米里状态机设计实现“1011”序列检测。 | | | | | | |
| 二、实验项目内容  　　设计实现一个米里状态机，能检测一个 8 位的二进制数据中是否存在“1011”序列。 | | | | | | |
| 1. 实验设计   　　状态表与状态图是用来表示同步时序电路的输入、输出、现态、次态之间转移关系的两种常用工具。如果同步时序电路的输出是输入和现态的函数，则称该电路为 Mealy 型电路。  　　序列检测器在很多数字系统中都不可缺少，尤其是在通信系统当中。序列检测器的作用就是从一系列的码流中找出用户希望出现的序列，序列可长可短。比如在通信系统中，数据流帧头的检测就属于一个序列检测器。序列检测器的类型有很多种，有逐比特比较的，有逐字节比较的，也有其他的比较方式，实际应用中需要采用何种比较方式，主要是看序列的多少以及系统的延时要求。  　　由于实际的拨码开关和按键开关都是机械式的设备，开关动作来回抖动多次后才能稳定下来，这个过程就会使得信号产生抖动。因此，如果用按键来产生时钟信号，为了一次按键得到一次上升沿（或下降沿），那么需要对按键输入先进行消抖处理。代码实现参考文档“按键消抖模块参考学习”。 | | | | | | |
| 1. 实验过程或算法 2. 设计“1011”序列检测的状态转换图。   S0：代表检测到零个有效信号的状态  S1：代表检测到一个有效信号的状态  S2：代表检测到两个有效信号的状态  S3：代表检测到三个有效信号的状态  S4：代表检测到三个有效信号的状态     1. 调用并转串输出模块将八个拨码开关作为外部二进制码流的输入。   端口设计：  ａ为八个拨码开关输入的数据，clk控制时钟脉冲，en控制是否选择重新读入八个拨码开关输入的数据，sdout为并转串的一位输出。  并转串模块代码设计：  module par2ser(  input [7:0] a,  input clk,  input en,  output reg sdout  );  reg f; //f表示是否正在输出的过程中  reg [7:0] temp; //储存输入，方便进行移位  always @(posedge clk or negedge en )  begin  if(!en)//en关闭时f回到0  begin  f = 0;  temp = a;  sdout=0;  end  else if(!f && en )//f变为1，并且输出序列第一位  begin  temp=a;  f = 1;  sdout = temp[0];  end  else if(f) //依次输出  begin  temp = (temp>>1);  sdout = temp[0];  end  end  endmodule  按键防抖模块：  module debkey(clk,key,debkey);  input clk;  input key;  output debkey;  //---------------------------------------------------------------  //100Hz 分频  parameter T100Hz = 499999;  integer cnt\_100Hz;  reg clk\_100Hz;  always @(posedge clk )  begin  cnt\_100Hz <= cnt\_100Hz + 1'b1;  if(cnt\_100Hz == T100Hz) begin  cnt\_100Hz <= 32'b0;  clk\_100Hz <= ~clk\_100Hz;  end  end  //---------------------------------------------------------------  //去抖模块  reg key\_rrr,key\_rr,key\_r;  always @(posedge clk\_100Hz )  begin  key\_rrr <= key\_rr;  key\_rr <= key\_r;  key\_r <= key;  end  //---------------------------------------------------------------  assign debkey = key\_rrr & key\_rr & key\_r;  endmodule  3、编写状态机设计文件，实现“1011”序列，用一个按键作为一个启动检测信号，另用一个按键来模拟 clk 信号，检测开关序列中是否存在“1011”序列，按下启动检测信号后，每按一次模拟 clk 的按键便传入一个开关的值，如果检测到“1011”序列则指定的 LED 灯亮。  端口设计：  clk控制时钟脉冲，set为重置按键（及开始按键），din为输入的八位待检测数据，key为输入按键，模拟时钟信号，ans为结果输出，ｘ为当前读入的数据，flag为展示现在的状态，开始检测与否。  module Mealy(  input clk,  input set,  input [7:0] din,  input key, //按键输入，模拟时钟信号  output ans, //结果输出  output x, //当前读入的数据  output reg flag  );  reg ans;  reg en;  reg [1:0] currentstate; //此时处于的状态  wire key\_out; //模拟时钟信号  wire set\_out;  //四种状态  parameter s0 = 2'b00;  parameter s1 = 2'b01;  parameter s2 = 2'b10;  parameter s3 = 2'b11;      par2ser u0(din,key\_out,en,x); //并转串模块调用  debkey u1(clk,key,key\_out); //按键消抖模块调用  debkey u2(clk,set,set\_out);    always@(negedge set\_out) begin  en=~en;  flag=en;  end    always@(negedge key\_out or negedge en)  begin  if(!en) begin //重置  currentstate<=s0;  ans<=0;  end  else begin  if(currentstate==s3 && x==1) //检查当前状态和输入  ans=1;  case(currentstate) //状态转换关系  s0:currentstate = x ? s1:s0;  s1:currentstate = x ? s1:s2;  s2:currentstate = x ? s3:s0;  s3:currentstate = x ? s1:s2;  default: currentstate = s0;  endcase  end  end  endmodule | | | | | | |
| 五、实验过程中遇到的问题及解决情况  1.在编写仿真代码时，一开始对如何对按键消抖模块进行仿真，很迷茫，后面查找资料以及询问同学，加上对按键消抖模块的思考，我们决定将在仿真时把按键消抖模块注释掉。 | | | | | | |
| 1. 实验结果及分析和（或）源程序调试过程   仿真代码：  仿真过程不需要按键消抖模块，因此我们将与按键消抖模块及其相关端口一起注释掉。仅留下clk时钟模拟每一次的串出，set模拟重置信号，din输入数据，ans显示是否检测到八位数据中是否有1011队列，x表示当前输入的信号的值。  module Mealy\_sim(  );  reg clk=0;  reg set=1;  reg [7:0] din=8'b1110\_1110;  wire ans;  wire x;  // wire flag;  always #20 clk=~clk;    Mealy u(clk,set,din,ans,x);    initial begin  #200 set=0;  #200 set=1;  end    endmodule  仿真图：    七、小组分工情况说明  黄婧婧负责构思设计，实现功能。  蔡欣彤负责仿真模拟，撰写实验报告。 | | | | | | |
|  | | | | | | |
|  | | | | | | |