**《数字逻辑》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **姓名** | |  | | **年级** | |  |
| **学号** | |  | | **专业、班级** | |  |
| **实验名称** | **摩尔状态机序列检测器** | | | | | |
| **实验时间** | **2019.12** | | **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，掌握有限状态机的设计方法，并用摩尔状态机设计实现“1101”序列检测。 | | | | | | |
| 二、实验项目内容  1、设计“1101”序列检测的状态转换图。  2、调用并转串输出模块，使用 Verilog HDL 语言的行为描述方式实现一个摩尔状态机，能检测一个 8 位的二进制数据中是否存在“1101”序列，如果检测到该序列则指定的 LED 灯亮。  3、综合、实现、生成 bit 流，下载到 Nexys4 开发板进行验证。 | | | | | | |
| 三、实验设计  1. 实验原理：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只与现态有关而与输入无关，则称该电路为 Moore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。   1. 状态图   五个状态，S4为最终态  s0 = 3'b000;  s1 = 3'b001;  s2 = 3'b010;  s3 = 3'b011;  s4 = 3'b100;  IMG_256 | | | | | | |
| 四、实验过程或算法  1. 根据设计要求，将所有可能的状态都列出来  2. 绘制状态表和原始状态图  3. 进行状态简化，绘制最简状态图  4. 进行状态分配，得到二进制状态表  5. 选用D触发器，求得输出函数和激励函数  1. 序列检测模块  module check(      input clk,//系统时钟      input clock\_in,// 并转串移位使能信号      input reset,//重置消抖      input clear,//重置状态机      input load\_in,// 读取并行序列的加载信号      input [7:0] data\_in, // 并行输入的序列      output reg dout // 检测结果信号      );      wire clock\_out; // 经过消抖处理后的移位使能信号      wire load\_out; // 经过消抖处理后的并转串加载信号      wire din; // 串行化处理后的信号      xiaodou U1(.clk(clk),.reset(reset),.load\_in(load\_in),.load\_out(load\_out),.clock\_in(clock\_in),.clock\_out(clock\_out));      par2ser PAR(.clk(clock\_out),.set(load\_out),.d(data\_in),.q(din));  // 五个状态，S4为最终态      parameter s0 = 3'b000;      parameter s1 = 3'b001;      parameter s2 = 3'b010;      parameter s3 = 3'b011;      parameter s4 = 3'b100;        reg [2:0] current\_state, next\_state;        always @(posedge clock\_out or negedge clear)      begin          if(clear==1)              current\_state <= s0;          else              current\_state <= next\_state;      end        always @ (current\_state or din)      begin          case(current\_state)              s0: next\_state <= (din == 1'b1) ? s1 : s0;              s1: next\_state <= (din == 1'b1) ? s2 : s0;              s2: next\_state <= (din == 1'b0) ? s3 : s2;              s3: next\_state <= (din == 1'b1) ? s4 : s0;              s4: next\_state <= (din == 1'b1) ? s2 : s0;          default: next\_state = s0;          endcase        end        always @(current\_state)      begin          if(current\_state == s4) dout = 1;          else dout = 0;      end  endmodule  2. 并转串模块  module par2ser(clk,set,d,q);      input clk, set; // 时钟信号和加载信号      input [7:0]d; // 并行输入序列      output q; //  串行输出        reg [7:0] d\_reg = 8'b0000\_0000;//初始化寄存器      always@ (posedge clk, posedge set)      begin          if(set)              d\_reg <= d; // 读入并行序列到寄存器中          else              d\_reg <= (d\_reg<<1);// 右移一位  q= d\_reg[7]; // 最高位的数据被串行输出      end  endmodule  3. 按键去抖模块  module xiaodou(      input clk, // 系统时钟      input reset, // 重置信号      input load\_in, // 开始读取加载信号      input clock\_in, // 移位使能信号      output load\_out, // 处理后的加载信号      output clock\_out // 处理后的使能信号      );      reg key\_rrr,key\_rr,key\_r;      reg load\_rrr,load\_rr,load\_r;      always @(posedge clk or negedge reset)         if(!reset)            begin // key 和 load均为高电平有效，故初始化为0               key\_rrr <=0;               key\_rr <=0;               key\_r <=0;               load\_rrr <=0;               load\_rr <=0;               load\_r <=0;            end          else            begin // 在三个时钟周期内进行取样，由于采用非阻塞式赋值，更新彼此有一个时钟周期的延迟                key\_rrr <= key\_rr;                key\_rr <= key\_r;                key\_r <= clock\_in;                load\_rrr <= load\_rr;                load\_rr <= load\_r;                load\_r <= load\_in;            end      assign clock\_out = key\_rrr & key\_rr & key\_r; // 三个时钟周期内的三次取样均为高电平，说明稳定      assign load\_out = load\_rrr & load\_rr & load\_r;  endmodule | | | | | | |
| 五、实验过程中遇到的问题及解决情况  1.画状态转移图时，中间状态的转移开始不够简洁  解决方法： 进行了状态简化，绘制出最简状态图  2.并转串模块本想运用系统时钟进行移位寄存器的移位，但是通过时序仿真可以实现，在FPGA板上无法实现  解决方法：选择使用自定义时钟，最后成功实现 | | | | | | |
| 1. 实验结果及分析和（或）源程序调试过程 2. 序列检测测试模块   module check\_1101\_sim;      reg clk;      wire dout;      reg clock\_in, reset, clear, load\_in;      reg [7:0] data\_in;        initial clk = 1;        always #20 clk = ~clk;        always          begin              #200 clock\_in = ~clock\_in;   load\_in = ~load\_in;          end      initial          begin              #10 clock\_in = 0; reset = 0; clear = 1; load\_in = 0; //clock\_in,reset为0时重置消抖，clear为1时重置状态机，load\_in为1时开始并转串              #40 clear = 0; reset = 1;          end      initial          begin              data\_in = 8'b11010101;          end        check c(               clk,//系统时钟               clock\_in,// 并转串移位使能信号               reset,//重置消抖               clear,//重置状态机               load\_in,// 读取并行序列的加载信号               data\_in, // 并行输入的序列               dout // 检测结果信号              );  endmodule  {F017C82B-DC3F-C780-9BF9-601A01B66AC4}  成功实现检测1101序列的功能，上板测试后确认无误。   1. 并转串测试模块   module par2ser\_sim;      reg clk;      wire q;      reg set;      reg [7:0] d;        initial clk = 0;      always #20 clk = ~clk;        initial      begin          set = 1;    //set为1将数据全部读入，为0开始并转串          d = 8'b11010101;          #20 set = 0;      end        par2ser p2s(clk,set,d,q);  endmodule     1. **按键去抖模块**   module xiaodou\_sim;      reg clk; // 系统时钟      reg reset; // 重置信号      reg load\_in; // 开始读取加载信号      reg clock\_in; // 移位使能信号      wire load\_out; // 处理后的加载信号      wire clock\_out; // 处理后的使能信号        initial          begin              clk = 1; reset = 1;              clock\_in = 0; load\_in = 0;              #50 reset = 0;          end        always #20 clk = ~clk;        always          begin              #400 clock\_in = ~clock\_in;   load\_in = ~load\_in;          end        xiaodou x(       clk, // 系统时钟       reset, // 重置信号       load\_in, // 开始读取加载信号       clock\_in, // 移位使能信号       load\_out, // 处理后的加载信号       clock\_out // 处理后的使能信号      );  endmodule  {54011FFE-A992-F71F-E649-AF293E41A543}   1. **开发板** 2. 并行输入序列为11011010     发现读入四次之后（即检测到1101），输出为1；且继续读三次，输出又为1。说明可以正常检测序列1101，且状态设计正确。   1. 并行输入序列为11010110     发现读入四次之后（即检测到1101），输出为1。说明可以正常检测序列1101，且状态设计正确。  经过如上充分测试可以说明本次实验设计正确，功能正常。  七、小组分工情况说明  蔡嘉轩、尹宇慧：负责实验结果部分（开发板的操作演示等）、实验设计部分（包括原理图、电路图等）。  田润泽：负责协同小组其他成员调试程序，完善、补充相应的部分  姚语涵：负责实验结果部分（编写仿真代码等），撰写报告 | | | | | | |