**实验三 时序电路设计实验报告**

**姓名**：曾泇睷  **学号**：1820221053

**班级**：07812201 **手机**：15101578062

注：黑色字体内容不能改动，蓝色字体内容（为示例或说明）需删除和修改。

1. **实验题目**

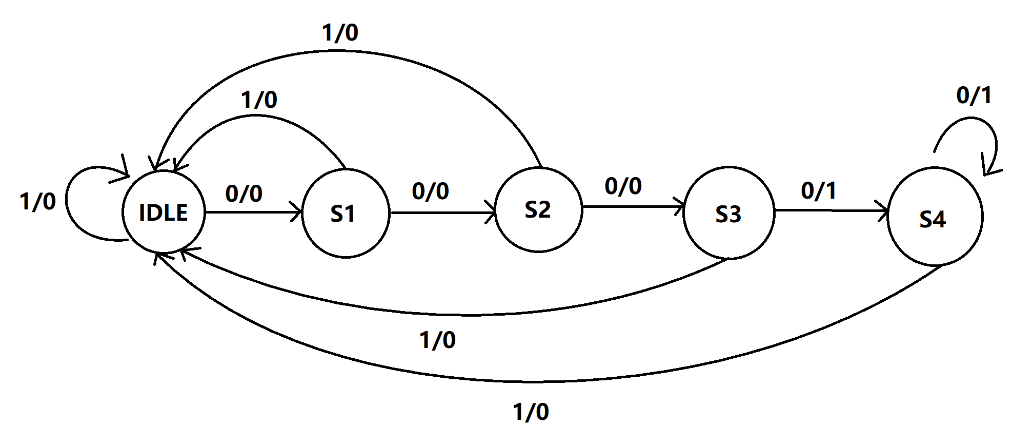
设计一个串行数据子序列检测器。当连续输入 4 个或 4 个以上的 0 时，输出为 1，其他情况下输出为 0。

1. **电路设计**
   1. **规范化**

输入：在每个时钟周期内接受1个0或1的输入。

输出：判断是否已经连续输入4个或以上的0，若是则输出1，否则输出0.

* 1. **形式化**



* 1. **状态分配**

IDLE状态：0000（0）

S1状态：0001（1）

S2状态：0010（2）

S3状态：0011（3）

S4状态：0100（4）

1. **电路实现**

`timescale 1ns / 1ps

module Exp2(

input clk,

input rst\_n,

input in,

output reg out

);

parameter STATE\_IDLE = 4'b0;

parameter STATE\_S1 = 4'b0001;

parameter STATE\_S2 = 4'b0010;

parameter STATE\_S3 = 4'b0011;

parameter STATE\_S4 = 4'b0100;

reg [3:0] current\_state;

reg [3:0] next\_state;

always @ (posedge clk or negedge rst\_n) begin

//复位则回到默认态

if(~rst\_n) begin

current\_state <= STATE\_IDLE;

end

else begin

current\_state <= next\_state;

end

end

always @(\*) begin

case (current\_state)

//只要输入0就进入下一状态

STATE\_IDLE: next\_state = (in=='b0) ? STATE\_S1 : STATE\_IDLE; //IDLE状态

STATE\_S1 : next\_state = (in=='b0) ? STATE\_S2 : STATE\_IDLE; //S1状态

STATE\_S2 : next\_state = (in=='b0) ? STATE\_S3 : STATE\_IDLE; //S2状态

STATE\_S3 : next\_state = (in=='b0) ? STATE\_S4 : STATE\_IDLE; //S3状态

//已经检测出来连续4个0输入，继续输入0仍然停在

STATE\_S4 : next\_state = (in=='b0) ? STATE\_S4 : STATE\_IDLE; //S4状态

default:

next\_state = STATE\_IDLE;

endcase

end

always @ (posedge clk or negedge rst\_n) begin

#clk

if(~rst\_n) begin

out <= 'b0;

end

//状态达到S4，输出1，代表得到目标子序列

else if(current\_state==STATE\_S4) begin

out <= 'b1;

end

//没有达到状态S4，输出0，代表没有得到目标子序列

else begin

out <= 'b0;

end

end

endmodule

1. **电路验证**
   1. **TestBench**

`timescale 1ns / 1ps

module testbench(

);

reg clk;

reg rst\_n;

reg in;

wire out;

always #5 clk<=~clk;

initial begin

clk <= 'b0;

rst\_n <= 'b0;

in <= 'b1;

#5 rst\_n <= 'b1;

//输入01

#10 in <= 'b0;

#10 in <= 'b1;

//输入001

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b1;

//输入0001

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b1;

//输入00001

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b1;

//输入000001

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b0;

#10 in <= 'b1;

end

Exp2 exp2(

.clk (clk ),

.rst\_n (rst\_n),

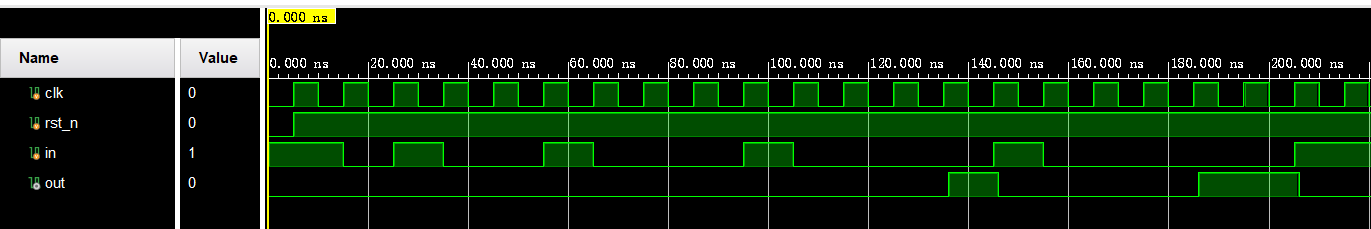
.in (in ),

.out (out )

);

endmodule

* 1. **仿真结果**

从15.000ns开始：输入一个0后输入一个1，输出保持为0

从35.000ns开始：输入两个0后输入一个1，输出保持为0

从65.000ns开始：输入三个0后输入一个1，输出保持为0

从105.000ns开始：输入四个0后输入一个1，输出在输入第四个0时变为1，输入1后又重置为0

从155.000ns开始：输入五个0后输入一个1，输出在输入第四个0时变为1，输入第5个0时保持输出为1，输入1后又重置为0

1. **实验心得**

本实验通过设计一个状态机，来检测连续输入的四个0，并在满足条件时输出1，这使我深入理解了有限状态机的工作原理和实现过程。通过参数化定义和状态转移，我掌握了如何使用Verilog HDL来描述复杂的逻辑电路。