实验二：序列检测器设计

一 实验目的

1. 掌握有限状态机的实现原理与方法
2. 掌握序列检测的方法

二 实验原理

有限状态机

- 有限状态机是一种描述状态转移的理论装置
- 通过输入数据的变化，装置在有限的状态之间发生转移
- 当输入信号与状态同时满足一定条件时，执行特定的输出操作

移位寄存器

- 移位寄存器由多个触发器级联形成，前一个触发器的输出端是另一个触发器的输入端。
- 当在一端的触发器串行输入数据，数据将会根据时钟运行逐渐向寄存器移位
- 当并行输出的数据满足一定条件时，执行特定的输出操作

三 实验设计

有限状态机的实现

- 我们可以将当前输入的数据作为状态
- 注意到，因为目标序列已经明确（101011），分析状态时可以进行简化
    - 将复位状态设置为000000
    - 1位状态只有一种，设定为000001，复位状态输入为0，我们认为仍旧是原状态
    - 对于000001状态，输入0时，转变为000010状态，而输入1时，保持原状态，因为输入为11时，对于判定指定序列并没有意义。
- 原理如上，以下是完整的状态转移表

现态	x = 0	x = 1	输出（x = 0）	输出（x = 1）	状态输出表示
000000	000000	000001	0	0	000
000001	00010	000001	0	0	001
000010	000000	000101	0	0	010
000101	001010	000001	0	0	011
001010	000000	010101	0	0	100
010101	001010	101011	0	0	101
101011	000010	000001	0	0	111

- 状态用三个led灯表示出来

移位寄存器设计

- 设计由6个D触发器级联的移位寄存器
- 在寄存器的一端输入串行数据，同时设定wire变量保存各寄存器输出端口的值作为序列判定的依据
- 当待输入数据为1，而状态处于010101或者110101时，在下一次时钟上升沿，最左端信号因为左移离开寄存器，从而达到101011的状态，我们在此时输出判定信号为1

四 实验代码

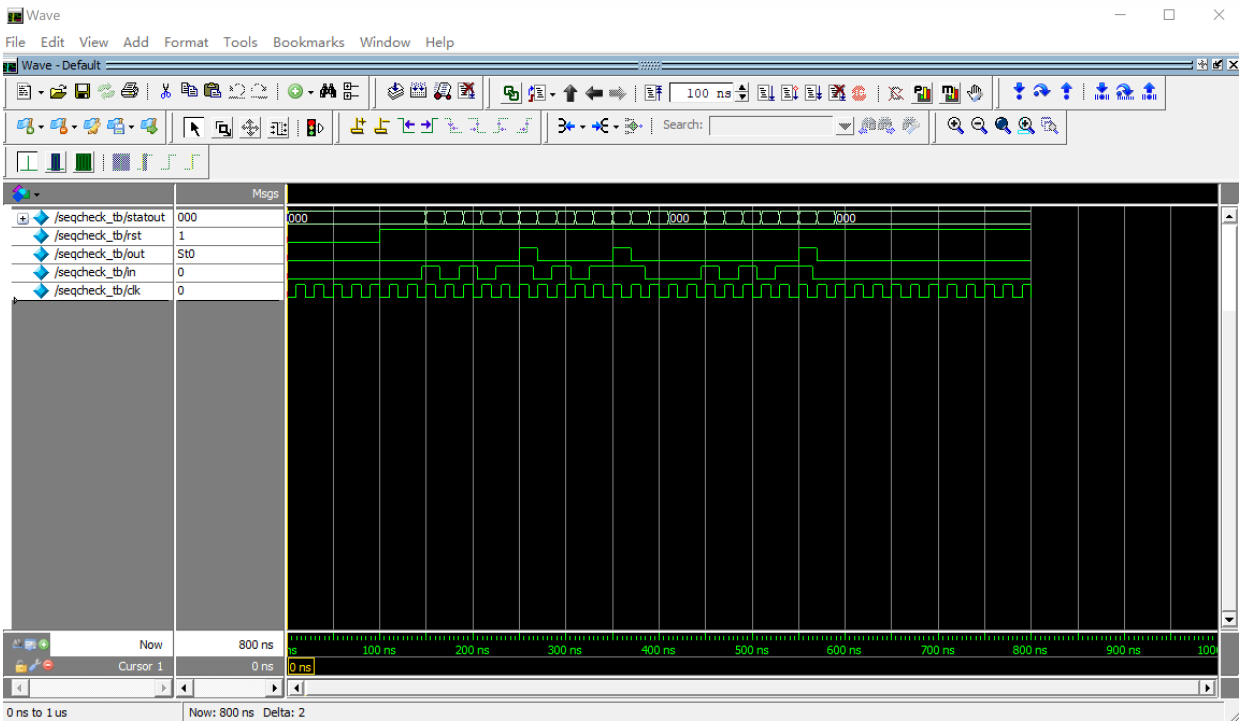
有限状态机实现

```
module seqcheck(in, out, rst, clk, statout);
input in, rst, clk;
output out, statout;
wire [2:0] statout;
reg [5:0] state;
reg out;
always @(posedge clk or negedge rst) begin
    if (~rst) begin
        state <= 6'b000000;
        out <= 0;
    end
    else if (clk) begin
    out <= 0;
        if(state == 6'b000000) begin
            if(in == 0) state <= 6'b000000;
            else if(in == 1) state <= 6'b000001;
        end
        else if (state == 6'b000001) begin
```

```
            if(in == 0) state <= 6'b000010;
            else if(in == 1) state <= 6'b000001;
        end
        else if (state == 6'b000010) begin
            if(in == 0) state <= 6'b000000;
            else if(in == 1) state <= 6'b000101;
        end
        else if (state == 6'b000101) begin
            if(in == 0) state <= 6'b001010;
            else if(in == 1) state <= 6'b000001;
        end
        else if (state == 6'b001010) begin
            if(in == 0) state <= 6'b000000;
            else if(in == 1) state <= 6'b010101;
        end
        else if (state == 6'b010101) begin
            if(in == 0) state <= 6'b001010;
            else if(in == 1) begin
                state <= 6'b101011;
                out <= 1;
            end
        end
        else if (state == 6'b101011) begin
            if(in == 0) state <= 6'b000010;
            else if(in == 1) state <= 6'b000001;
        end
    end
end
stat s0(.din(state), .dout(statout));
endmodule

module stat(din, dout);
input [5:0] din;
output [2:0]dout;
assign dout = (din==6'b000000)?3'b000:
              (din==6'b000001)?3'b001:
              (din==6'b000010)?3'b010:
              (din==6'b000101)?3'b011:
              (din==6'b001010)?3'b100:
              (din==6'b010101)?3'b101:
              (din==6'b101011)?3'b111:3'b000;
endmodule
```

仿真结果



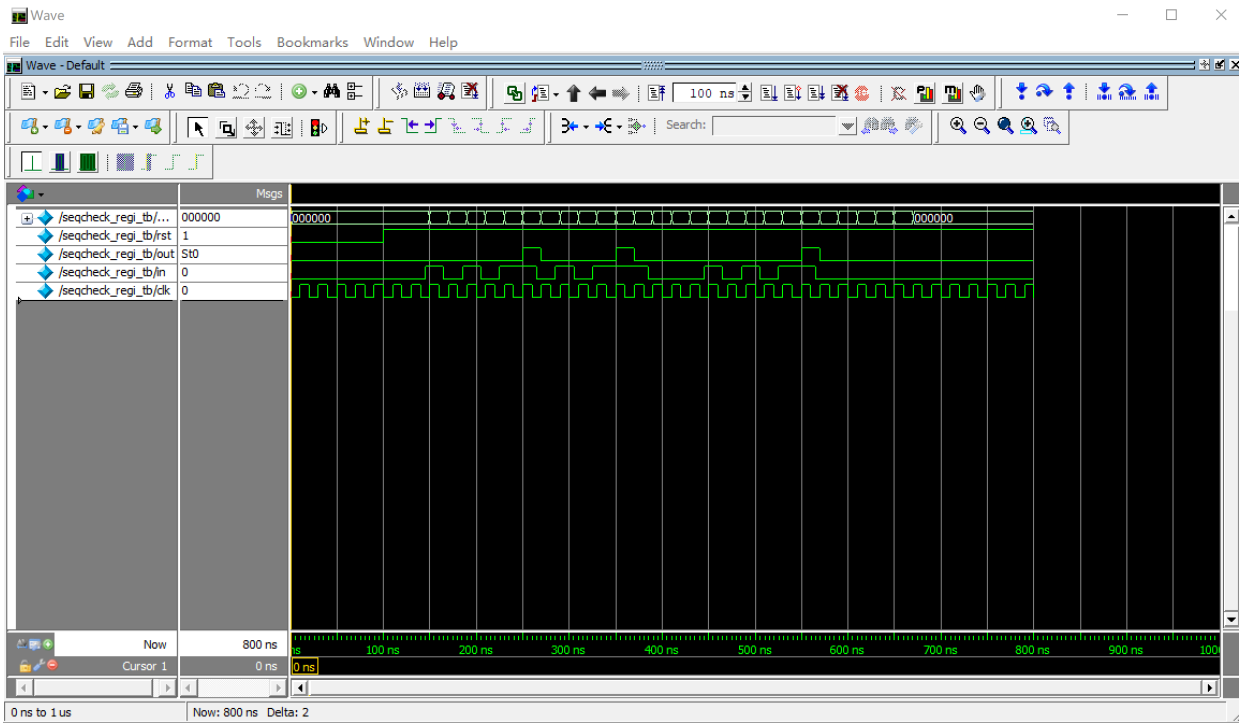
移位寄存器实现

```
`timescale 1ns/1ns
module seqcheck(in, out, clk, rst, state);
input in, clk, rst;
output out;
output state;
reg out;
wire [5:0] state;
dtricker D0(.d(in), .q(state[0]), .rst(rst), .clk(clk));
dtricker D1(.d(state[0]), .q(state[1]), .rst(rst), .clk(clk));
dtricker D2(.d(state[1]), .q(state[2]), .rst(rst), .clk(clk));
dtricker D3(.d(state[2]), .q(state[3]), .rst(rst), .clk(clk));
dtricker D4(.d(state[3]), .q(state[4]), .rst(rst), .clk(clk));
dtricker D5(.d(state[4]), .q(state[5]), .rst(rst), .clk(clk));
always @(posedge clk or negedge rst) begin
    if (~rst) begin
        out <= 0;
    end
    else if (clk) begin
        out <= 0;
        if(((state == 6'b010101) || (state == 6'b110101))
         && (in == 1)) out <= 1;
    end
end
endmodule

module dtricker(d, q, rst, clk);
input d, rst, clk;
output q;
```

```
29  reg q;
30  always @(posedge clk or negedge rst) begin
31      if (~rst) begin
32          q <= 0;
33      end
34      else begin
35          #0 q <= d;
36      end
37  end
38  endmodule
```

仿真结果



- 从上述两个仿真结果看出,当输入一段序列后(实验中样例序列),输出信号在正确的地方出现脉冲

五.实验小结

1. 在进行移位寄存器的设计时,一开始因为没有考虑清楚判定输入数据的时间与条件,导致高电平的信号输出比目标序列输入滞后一个周期,之后,通过增加条件语句,得以在串行数据尚未进入寄存器时便将其数据作为状态判定,从而能够在序列出现同时,输出检测信号.
2. always的时钟沿触发时,一般不需要写else if(clk), 这样有时可能出现错误.