

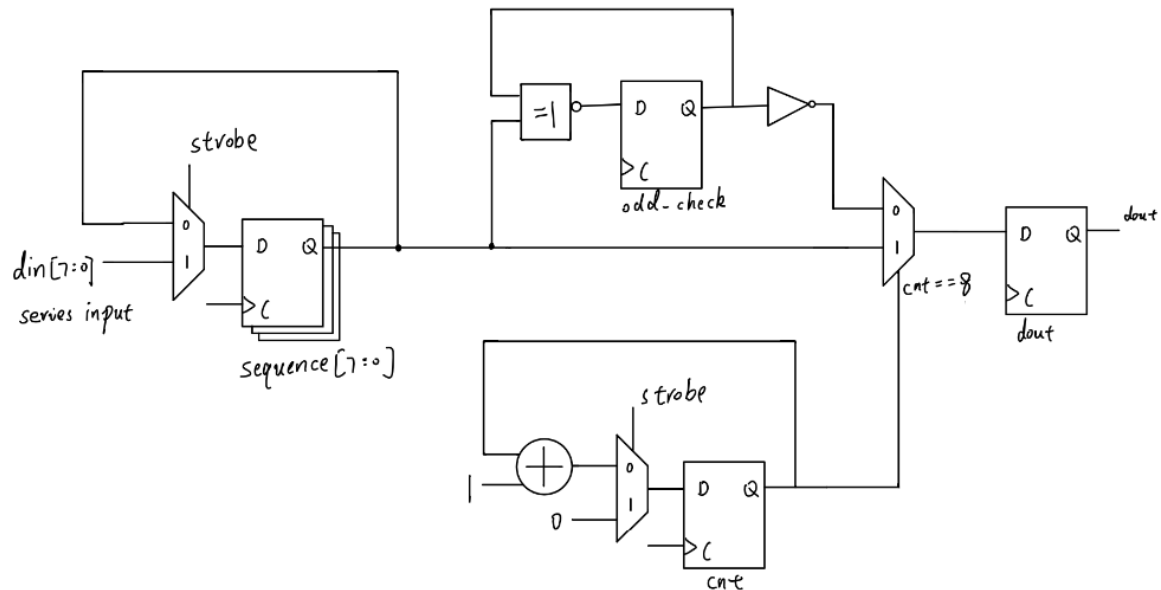
IC_HW_3

2021211039 刘沁雨

HW_3_1

原理示意图

dout输出部分为了和示例保持一致，加了一个寄存器，延迟一个周期输出



源代码

```
1 module hw_3_1 (  
2     input    [7:0]din,  
3     input    clk,rst,strobe,  
4     output   reg dout  
5 );  
6     reg [7:0]sequence;  
7     reg    odd_check;  
8     reg    [3:0]cnt;  
9  
10    always @(posedge clk) begin  
11        if(rst|strobe)  
12            odd_check <= 0;  
13        else  
14            odd_check <= (odd_check ^ dout);  
15        end  
16  
17    always @(posedge clk) begin  
18        if(rst|strobe)  
19            cnt <= 0;  
20        else  
21            cnt <= cnt + 1;  
22        end  
23
```

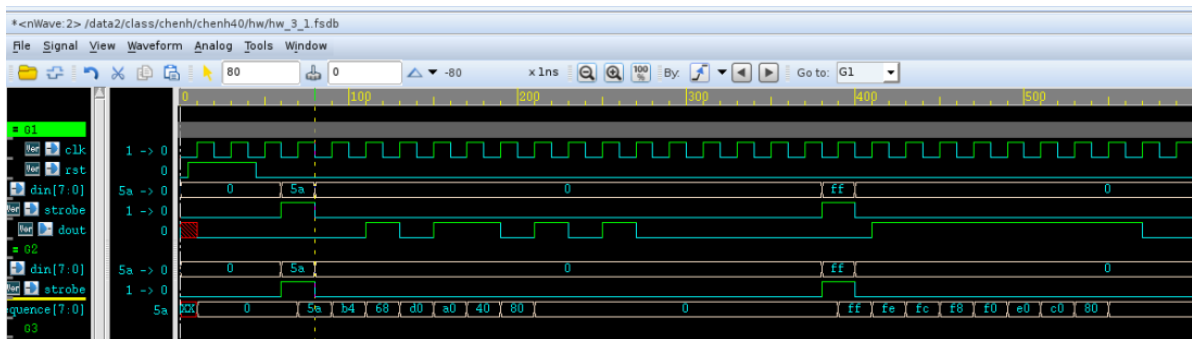
```

24     always @(posedge clk) begin
25         if(rst)
26             sequence <= 8'b0;
27         else if(strobe)
28             sequence <= din;
29         else
30             sequence <= {sequence[6:0],1'b0};
31     end
32
33     always @(posedge clk) begin
34         if(rst) dout <= 0;
35         else if(cnt == 8) dout <= ~odd_check;
36         else dout <= sequence[7];
37     end
38 endmodule

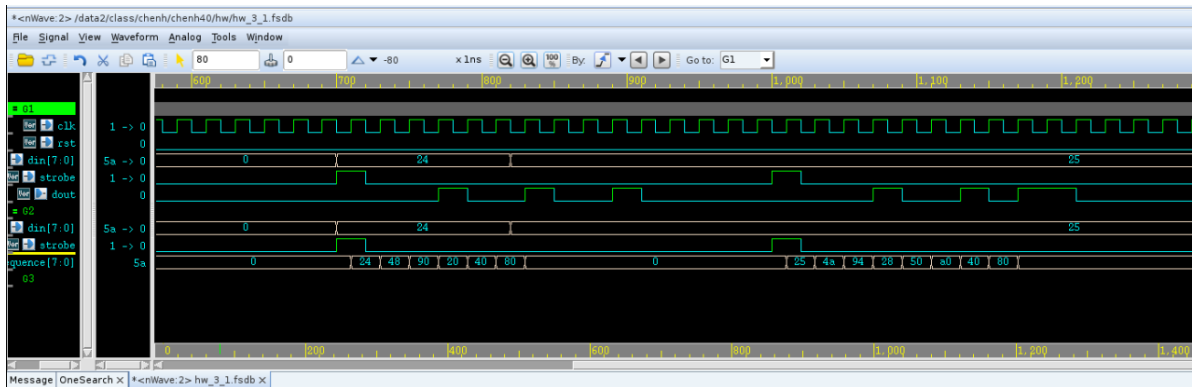
```

仿真结果

第一张图G1验证示例功能

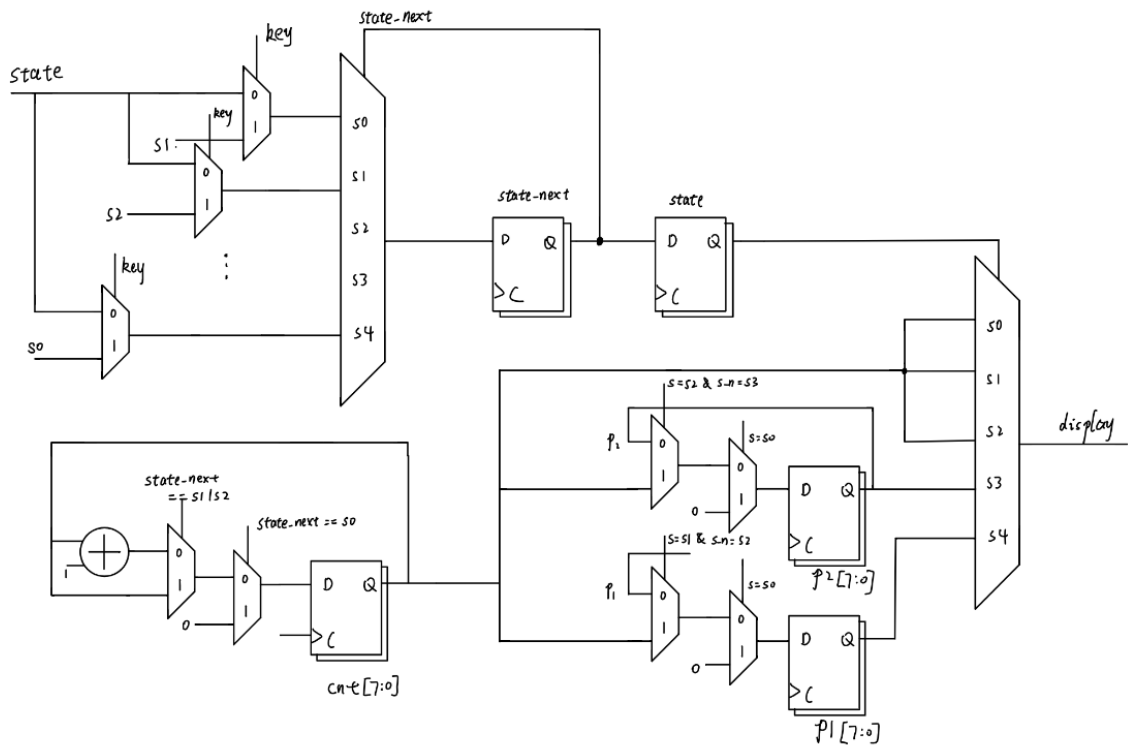


第二张图G2，控制strobe信号晚于输入信号输入，验证strobe的低电平信号期间，输入信号无效。可以看到25等到strobe信号置一才被输入到寄存器串中，对输入的隔绝符合预期。



HW_3_2

原理示意图



源代码

```

1 module hw_3_2 (
2     input clk, key, rst,
3     output reg [7:0] display
4 );
5     parameter s0 = 3'b000, //清零
6               s1 = 3'b001, //计数开始
7               s2 = 3'b010, //记住p1
8               s3 = 3'b011, //计数停止, 输出p2
9               s4 = 3'b100; //输出p1
10
11     reg [2:0] state, state_next;
12     reg [7:0] cnt, p1, p2;
13
14     always@(posedge clk) begin
15         if(rst) begin
16             cnt <= 0;
17             state_next <= s0;
18         end
19         else
20             cnt <= cnt + 1;
21             state <= state_next;
22         end
23
24     always @(*) begin
25         case(state)
26             s0: begin display = cnt; if(key) state_next = s1; else
state_next = s0; end
27             s1: begin display = cnt; if(key) state_next = s2; else
state_next = s1; end
28             s2: begin display = cnt; if(key) state_next = s3; else
state_next = s2; end
29             s3: begin display = p2; if(key) state_next = s4; else
state_next = s3; end

```

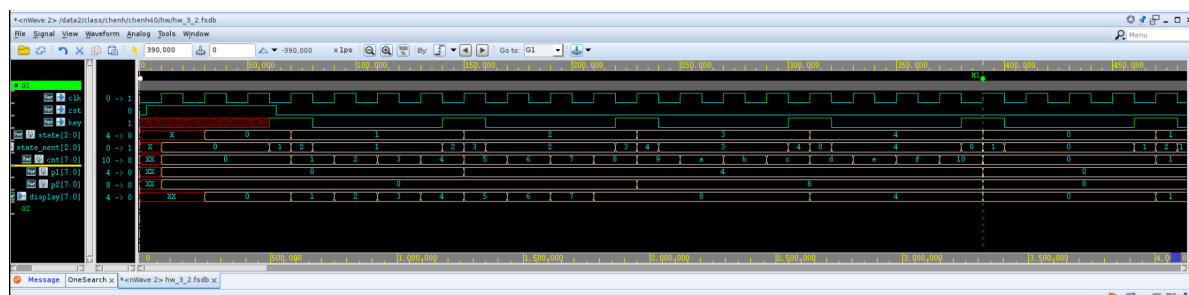
```

30         s4: begin display = p1; if(key) state_next = s0; else
state_next = s4; end
31     endcase
32 end
33
34     always @(posedge clk) begin
35         if(rst|state_next == s0) begin
36             cnt <= 0;
37             p1 <= 0;
38             p2 <= 0;
39         end
40         else if(state_next == s1|s2)
41             cnt <= cnt + 1;
42             if(state == s1 && state_next == s2) p1 <= cnt;
43             if(state == s2 && state_next == s3) p2 <= cnt;
44         end
45     endmodule

```

仿真结果

设定每四个周期按压一次key，可以看到输出结果符合预期，

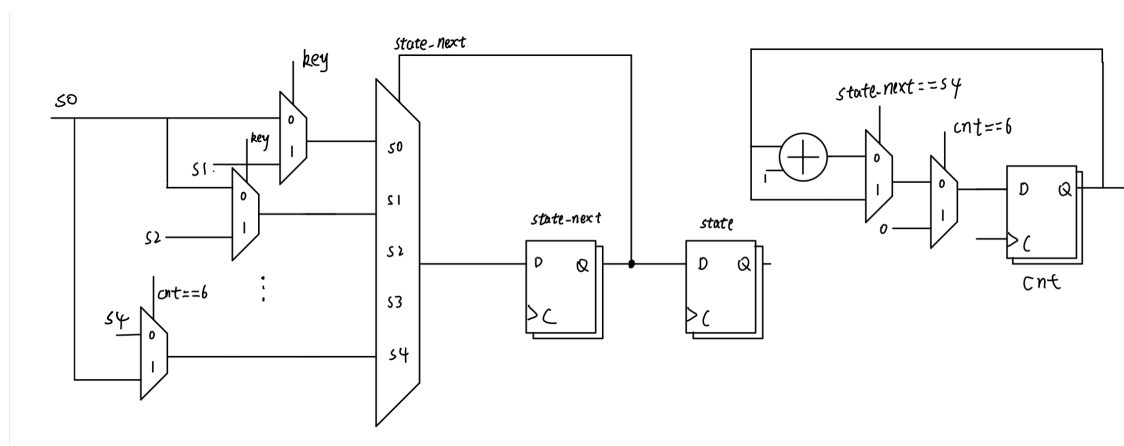


HW_3_3

原理示意图

这块模块间互联有点麻烦，就分小模块展示

1. 状态机模块 + 计数器模块



2. 串入并出模块以及求奇校验位的模块


```

30         else begin
31             case (state)
32                 s0: if(din) state_next = s1; else state_next = s0;
33                 s1: if(!din) state_next = s2; else state_next = s0;
34                 s2: if(din) state_next = s3; else state_next = s0;
35                 s3: if(!din) state_next = s4; else state_next = s0;
36                 s4: if(cnt == 6) state_next = s0;
37                 default: state_next = s0;
38             endcase
39         end
40     end
41
42     always @(posedge clk) begin
43         if(rst|cnt==6) cnt <= 0;
44         else if(state_next == s4)
45             cnt <= cnt + 1;
46     end
47
48     always @(*) begin
49         if(cnt == 5) flag = check^din;
50         else flag = 0;
51     end
52
53     always @(posedge clk) begin
54         if(rst)
55             sequence <= 0;
56         else
57             sequence <= {sequence[2:0],din};
58     end
59
60     always @(posedge clk) begin
61         if(rst|flag) begin
62             dout <= 0;
63             strobe <= 0;
64         end
65         else if(cnt == 5) begin
66             dout <= sequence;
67             strobe <= 1;
68         end
69         else begin
70             dout <= 0;
71             strobe <= 0;
72         end
73     end
74 endmodule

```

仿真结果

G1框图中M1前部分展示的是示例部分的输出。

G2展示的是在M1后，各个模块的子状态被成功复位到初始状态，并能够正常继续工作（检测到新的1010序列），不受干扰。

输入序列数据保持不变，奇偶校验位改变（1010_1011_1），最终输出始终保持为0，可见数据校验功能正常。

