

实验三 时序电路设计实验报告

姓名：刘秉致

学号：1120220715

班级：070122011

手机：13051179979

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。

2. 电路设计

a) 规范化

输入：二进制字符 A

输出：二进制字符 B

时序行为：当连续输入连续 4 个或 4 个以上的 0 时，输出 1，其他情况下输出 0。

可能存在的状态：

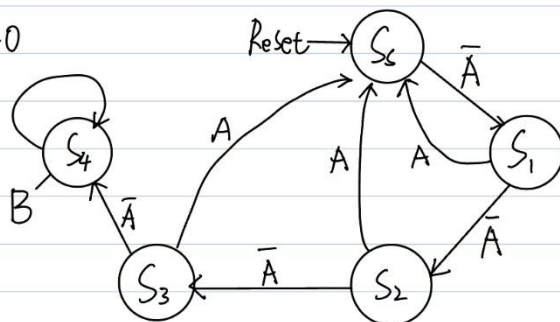
- S_0 : 初始状态，输出 0；
- S_1 : 连续接收到第一个 0，为未完成状态，输出 0；
- S_2 : 连续接收到第二个 0，未完成状态，输出 0；
- S_3 : 连续接收到第三个 0，为未完成状态，输出 0；
- S_4 : 连续接收到第四个 0，为完成状态，输出 1；

b) 形式化

状态机图绘制如下。

状态机图

Input: A
 Output: B
 Default: $B = 0$



根据状态机图，可以绘制出如下状态表。

状态表

现状态	下一状态		输出 B
	$A=1$	$A=0$	
S_0	S_0	S_1	0
S_1	S_0	S_2	0
S_2	S_0	S_3	0
S_3	S_0	S_4	0
S_4	S_4	S_4	1

c) 状态分配

采用格雷码赋值方法，将状态——赋值为：

- $S_0: 000$
- $S_1: 001$
- $S_2: 011$
- $S_3: 010$
- $S_4: 110$

因而电路需要采用三位寄存器 X 、 Y 、 Z 记录状态机状态。

获得如下真值表。

现状态 X Y Z	下一状态 A=1 A=0		输出B
0 0 0	0 0 0	0 0 1	0
0 0 1	0 0 0	0 1 1	0
0 1 1	0 0 0	0 1 0	0
0 1 0	0 0 0	1 1 0	0
1 1 0	1 1 0	1 1 0	1

对应获得该状态机的函数为

$$X_{next} = Y\bar{Z}\bar{A} + XY\bar{Z}$$

$$Y_{next} = Y\bar{Z}\bar{A} + XY\bar{Z} + \bar{X}Z\bar{A}$$

$$Z_{next} = \bar{X}\bar{Y}\bar{A}$$

$$B = XY\bar{Z}$$

3. 电路实现

```
4. `timescale 1ns/1ps
5.
6. module status(
7.     input wire A, //输入
8.     input wire clk, //时钟信号
9.     input wire reset, //复位信号
10.    output wire B //输出
11.);
12.    reg X,Y,Z; //三个寄存器，储存状态机状态
13.
14.    assign B=X&&Y&&!Z;
15.    always @(posedge clk) begin //时钟上沿触发
16.        if(!reset)begin
```

```

17.         X<=(Y&&!Z&&!A)|| (X&&Y&&!Z);
18.         Y<=(Y&&!Z&&!A)|| (X&&Y&&!Z)|| (!X&&Z&&!A);
19.         Z<=!X&&!Y&&!A;
20.     end
21.     else begin
22.         X<=0;
23.         Y<=0;
24.         Z<=0;
25.     end
26. end
27. endmodule

```

4. 电路验证

a) TestBench

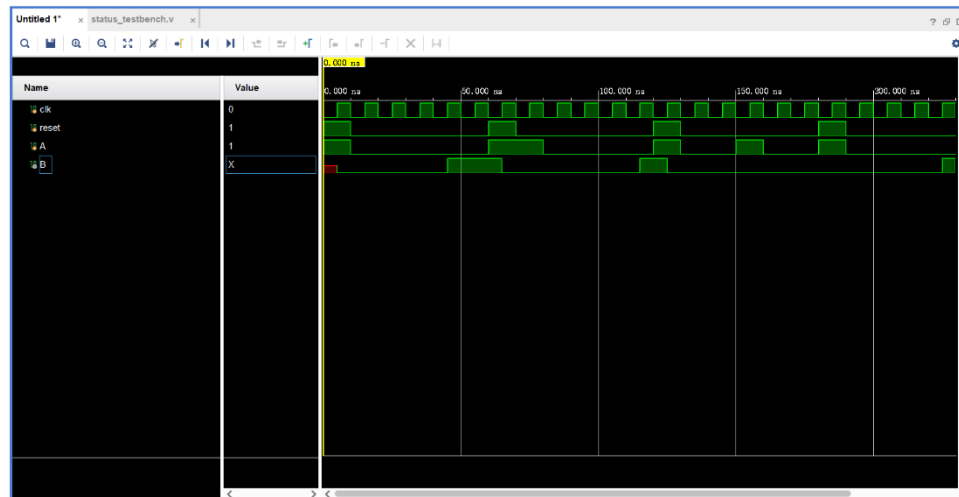
```

5. `timescale 1ns/1ps
6.
7. module status_testbench();
8.     parameter PERIOD = 10; //时间周期
9.     reg clk; //时钟信号，每 5ns 反转
10.    initial begin
11.        clk = 0 ;
12.        forever begin
13.            #(PERIOD/2) clk = ~clk;
14.        end
15.    end
16.
17.    reg reset; //复位信号
18.    initial begin
19.        reset=1;
20.        forever begin
21.            #(PERIOD) reset=0;
22.            #(PERIOD*5) reset=1;
23.        end
24.    end
25.
26.    reg A;
27.    initial begin
28.        //复位位置
29.        A=1;
30.        //测试串 1: 00000
31.        #PERIOD A=0;

```

```
32.         #PERIOD A=0;
33.         #PERIOD A=0;
34.         #PERIOD A=0;
35.         #PERIOD A=0;
36.         //复位位置
37.         #PERIOD A=1;
38.         //测试串 2: 10000
39.         #PERIOD A=1;
40.         #PERIOD A=0;
41.         #PERIOD A=0;
42.         #PERIOD A=0;
43.         #PERIOD A=0;
44.         //复位位置
45.         #PERIOD A=1;
46.         //测试串 3: 00100
47.         #PERIOD A=0;
48.         #PERIOD A=0;
49.         #PERIOD A=1;
50.         #PERIOD A=0;
51.         #PERIOD A=0;
52.         //复位位置
53.         #PERIOD A=1;
54.         //测试串 4: 00001
55.         #PERIOD A=0;
56.         #PERIOD A=0;
57.         #PERIOD A=0;
58.         #PERIOD A=0;
59.         #PERIOD A=1;
60.         //结束测试
61.         $stop;
62.     end
63.
64.     wire B;
65.
66.     //模块实例化
67.     status status_test(
68.         .A(A),
69.         .clk(clk),
70.         .reset(reset),
71.         .B(B)
72.     );
73.
74. endmodule
```

a) 仿真结果



根据波形，对几个测试字符串，输出结果都为正确结果。

5. 实验心得

本实验中，实现了一个比较简单的时序电路，实验过程中出现的最大问题在于 testbench 编写时的语法问题。本人也是通过这一实验，了解了在 initial 语句块中不能包含 always 块，即过程块不能相互包含的语法规则。