实验 3 状态机实验

1 实验目的

1.1 熟悉 verilog 编程与调试

1.2 熟悉状态机的工作原理, 能熟练编写状态机程序

2 实验环境

AMD Vivado2022.2

3 原理说明

在本次实验课中, 主要学习了在 verilog 中编写状态机程序, 一般采用三段式来编写状态机程序, 也即

1. 阻塞的异步指数以及状态转换
2. 根据输入和当前状态产生下一状态
3. 根据输入和当前状态产生输出

在本实验中, 并未设计比较有挑战性或者创造性的电路, 因此在本次报告原理说明的部分主要内容是状态转移图以及思路讲解

3.1 序列检测器

两个序列检测器都能够检测重叠序列, 只需要改变状态转移即可, 在设计可检查重叠序列的序列检测器中, 要注意状态转移图中, 最后输出为 1 的状态再次接受输入后的状态转移并非回到初始, 而是根据子序列返回到之前某个状态中, 两个状态转移图如下

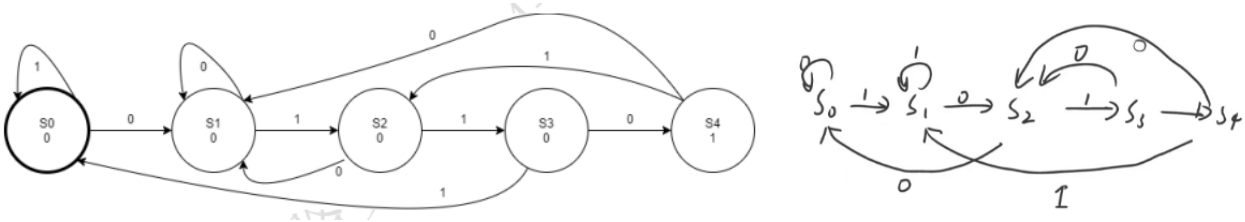

图 1: 左:0110 的序列检测　　右:1011 的序列检测

3.2 周期性输出

本次实验中, 采用了最简单的状态周期转移构成的 12 计数器, 只不过是改变了中间某些状态的输出, 不只是在最终状态的输出主要是 $S_0 \sim S_{11}$ 共 12 个状态, 一个时钟周期转换一个状态, 每个周期输出相应的输出在本实验中的状态转移图是简单的线性, 并无较大的参考价值, 在这里直接画出状态转移表

时钟周期	电路状态				输出
clk	S_3	S_2	S_1	S_0	Y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	0	0	0	0	0

3.3 报纸贩卖机

在本实验中采用的是当投入的币的金额超过 5 时, 直接 reset, 输出为 1 当且仅当正好金额为 5, 这个实验主要需要注意状态的转换, 状态转移图如下

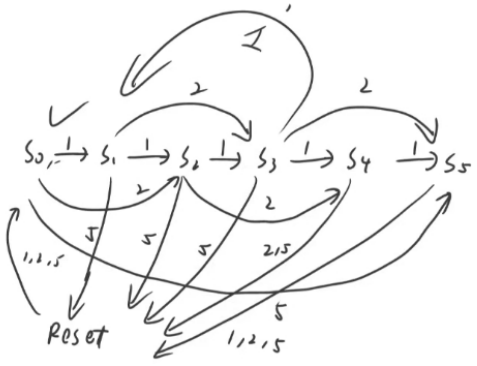

图 2: 报纸贩卖机的状态转移图

4 接口定义

在本次实验的每一个程序之中, 内部都利用 parameter 或者 localmeter 来定义了几个状态常量, 另外有两个内部的接线状态 state 以及 next_state, 分别表示当前状态以及下一状态, 下面是每一个接口

4.1 序列检测器

两个序列检测器的接口完全相同, 不同的二者的状态转移, 详见源代码部分

```
1    input clk,                  \\时钟的输入
2    input in,                   \\输入
3    input rstn,                 \\异步置数的输入
4    output reg out              \\输出，因为在always中赋值，需要使用reg
```

4.2 周期性输出

```
1    input clk,                  \\时钟的输入
2    input rstn,                 \\异步置数，此程序不需要输入
3    output reg out              \\输出
```

4.3 报纸贩卖机

```
1    input [1:0]in,              \\因为有1,2,5三个输入，因此需要2位输入
2    input clk,                  \\时钟的输入
3    input rstn,                 \\异步置数
4    output reg out              \\输出
```

5 调试过程以及结果

本次实验比较顺畅, 并未出现较大的困难, 相比于实验 2.5 给出的代码, 本次实验在代码中的状态转移并没有使用 if-else 语句来描述, 而是利用了 cases 的嵌套, 个人认为这样的代码更加简洁, 而且可读性更高.

另外, 一个比较明显的困难是在随机数 0,1,2 的生成, 由于 verilog 中 random() 语句生成的随机数是有符号的, 因此 $random() % 2 会生成显示为 3 的-1, 因此这里需要随机生成无符号数, 经过资料查找, 该效果可以用语句 $urandom_range(2, 0); 来实现

5.1 序列检测器

5.1.1 0110 序列检测

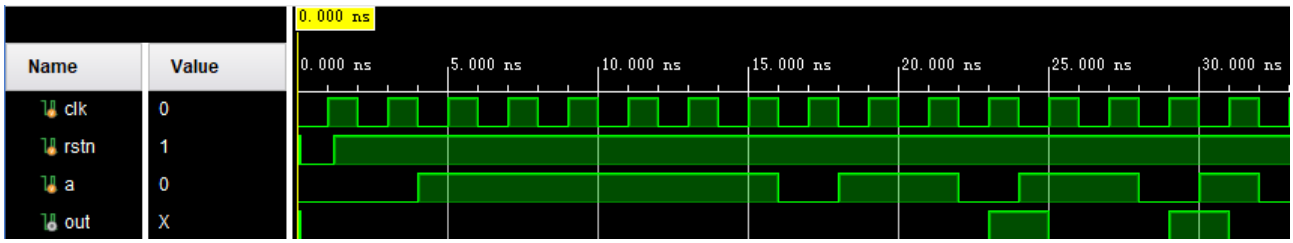

图 3: 0110 序列检测器的波形图

5.1.2 1011 序列检测器

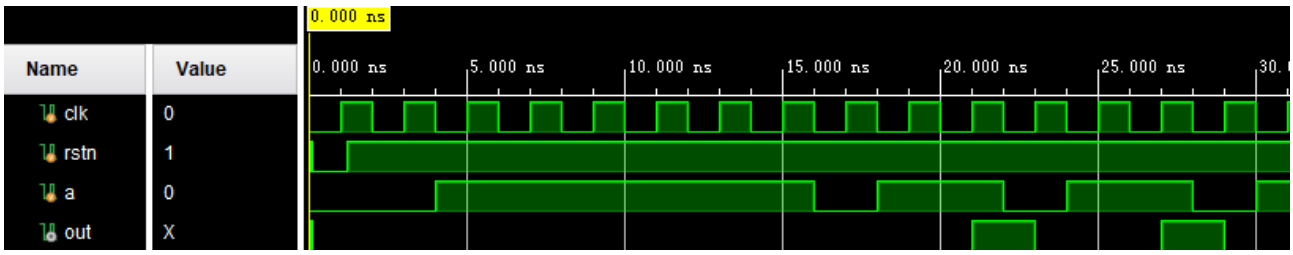

图 4: 1011 序列检测器的波形图

5.2 周期输出序列

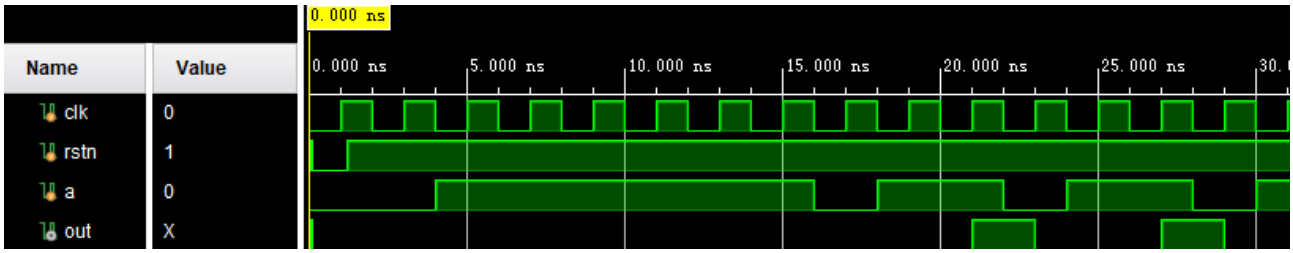

图 5: 周期输出序列波形图

5.3 报纸贩卖机波形图

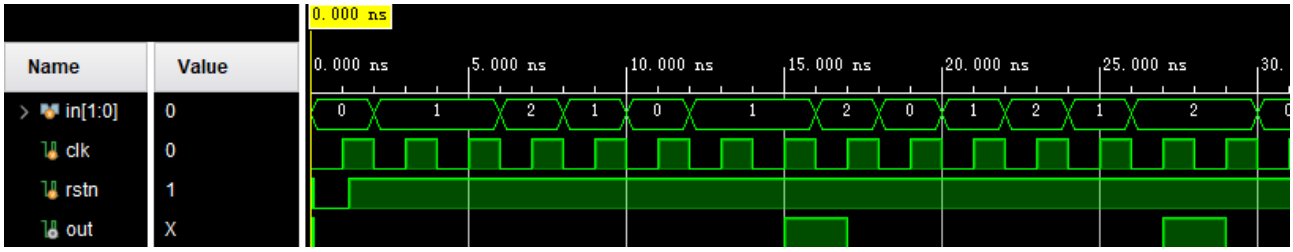

图 6: 报纸贩卖机波形图

6 实验总结

经过本次实验, 极大程度上达成了实验目的, 熟练掌握了三段式状态机的工作以及编写原理, negedge 和 posedge 分别表示下降, 上升沿触发同时对与 testbench 的编写有了更加深刻的体会, 比如 clk 的编写, 初始化输入以及异步指数等, 了另外对于 testbench 中的随机数生成有了使用 $urandom_range(m, n); 的办法

7 源代码

由于本次实验的 testbench 比较重要, 因此本次报告中的源代码部分分为设计文件和激励文件两部分

7.1 设计文件

7.1.1 0110 序列检测

```
module check_str(
    input clk,
    input in,
    input rstn,
    output reg out
);
    parameter s0 = 3'b000;
    parameter s1 = 3'b001;
    parameter s2 = 3'b010;
    parameter s3 = 3'b011;
    parameter s4 = 3'b100;
    //无关状态
```

```
13      parameter s5n = 3'b101;
14      parameter s6n = 3'b110;
15      parameter s7n = 3'b111;
16
17      reg [2:0]state;
18      reg [2:0]next_state;
19
20
21
22      always @(negedge rstn or posedge clk)begin
23          if(!rstn)begin
24              state <= s0;
25          end
26          else begin
27              state <= next_state;
28          end
29      end
30
31
32
33      always @(state or in)begin
34          case(state)
35          s0:begin
36              case(in)
37              0 : next_state = s1;
38              1 : next_state = s0;
39              endcase
40          end
41          s1:begin
42              case(in)
43              0 : next_state = s1;
44              1 : next_state = s2;
45              endcase
46          end
47          s2:begin
48              case(in)
49              0 : next_state = s1;
```

```
            1 : next_state = s3;
            endcase
        end
        s3:begin
            case(in)
            0 : next_state = s4;
            1 : next_state = s0;
            endcase
        end
        s4:begin
            case(in)
            0 : next_state = s1;
            1 : next_state = s2;
        default: next_state=s0;
            endcase
        end
        endcase
    end



    always@(state)begin
        case(state)
        s0: out = 1'b0;
        s1: out = 1'b0;
        s2: out = 1'b0;
        s3: out = 1'b0;
        s4: out = 1'b1;
        default: out=1'bx;
        endcase
    end

endmodule
```

7.1.2 1011 序列检测

```
module check_str_1(
```

```
    input clk,
    input in,
    input rstn,
    output reg out
);
    parameter s0 = 3'b000;
    parameter s1 = 3'b001;
    parameter s2 = 3'b010;
    parameter s3 = 3'b011;
    parameter s4 = 3'b100;
    //无关状态
    parameter s5n = 3'b101;
    parameter s6n = 3'b110;
    parameter s7n = 3'b111;

    reg [3:0]state;
    reg [3:0]next_state;



    always @(negedge rstn or posedge clk)begin
        if(!rstn)begin
            state <= s0;
        end
        else begin
            state <= next_state;
        end
    end



    always @(state or in)begin
        case(state)
        s0:begin
            case(in)
            0 : next_state = s0;
            1 : next_state = s1;
```

```
            endcase
        end
        s1:begin
            case(in)
            0 : next_state = s2;
            1 : next_state = s1;
            endcase
        end
        s2:begin
            case(in)
            0 : next_state = s0;
            1 : next_state = s3;
            endcase
        end
        s3:begin
            case(in)
            0 : next_state = s2;
            1 : next_state = s4;
            endcase
        end
        s4:begin
            case(in)
            0 : next_state = s2;
            1 : next_state = s1;
        default: next_state=s0;
            endcase
        end
        endcase
    end



    always@(state)begin
        case(state)
        s0: out = 1'b0;
        s1: out = 1'b0;
        s2: out = 1'b0;
```

```
76          s3: out = 1'b0;
77          s4: out = 1'b1;
78          default: out=1'bx;
79          endcase
80      end
81
82  endmodule
```

7.1.3 周期输出序列

```
1   module periodic_output(
2       input clk,
3       input rstn,
4       output reg out
5       );
6
7       parameter s0 = 4'b0000;
8       parameter s1 = 4'b0001;
9       parameter s2 = 4'b0010;
10      parameter s3 = 4'b0011;
11      parameter s4 = 4'b0100;
12      parameter s5 = 4'b0101;
13      parameter s6 = 4'b0110;
14      parameter s7 = 4'b0111;
15      parameter s8 = 4'b1000;
16      parameter s9 = 4'b1001;
17      parameter s10 = 4'b1010;
18      parameter s11 = 4'b1011;
19      parameter s12 = 4'b1100;
20      parameter s13 = 4'b1101;
21      parameter s14 = 4'b1110;
22      parameter s15 = 4'b1111;
23
24      reg [3:0]state;
25      reg [3:0]next_state;
26
27      always @(negedge rstn or posedge clk)begin
```

```
28          if(!rstn)begin
29              state <= s0;
30          end
31          else begin
32              state <= next_state;
33          end
34      end
35
36
37      always @(state)begin
38          case(state)
39              s0: next_state = s1;
40              s1: next_state = s2;
41              s2: next_state = s3;
42              s3: next_state = s4;
43              s4: next_state = s5;
44              s5: next_state = s6;
45              s6: next_state = s7;
46              s7: next_state = s8;
47              s8: next_state = s9;
48              s9: next_state = s10;
49              s10: next_state = s11;
50              s11: next_state = s0;
51              default: next_state = s0;
52          endcase
53      end
54
55
56
57      always @(state)begin
58          case(state)
59              s0: out = 0;
60              s1: out = 0;
61              s2: out = 1;
62              s3: out = 0;
63              s4: out = 1;
64              s5: out = 0;
```

```
            s6: out = 0;
            s7: out = 1;
            s8: out = 1;
            s9: out = 0;
            s10: out = 1;
            s11: out = 1;
            default: out = 1'bx;
        endcase
    end

endmodule
```

7.1.4 报纸贩卖机

```
module newspaper_machine(
    input [1:0]in,
    input clk,
    input rstn,
    output reg out
    );

    reg [2:0]state;
    reg [2:0]next_state;

    parameter s0 = 3'b000;
    parameter s1 = 3'b001;
    parameter s2 = 3'b010;
    parameter s3 = 3'b011;
    parameter s4 = 3'b100;
    parameter s5 = 3'b101;
    parameter s6 = 3'b110;
    parameter reset = 3'b111;

    always @(negedge rstn or posedge clk)begin
        if(!rstn)begin
            state <= s0;
        end
```

```
24          else begin
25              state <= next_state;
26          end
27      end
28
29
30      always @(in or state)begin//in中0,1,2分别表示1,2,5分钱
31          case(state)
32              s0:case(in)
33                      2'b00: next_state = s1;
34                      2'b01: next_state = s2;
35                      2'b10: next_state = s5;
36                  endcase
37              s1:case(in)
38                      2'b00: next_state = s2;
39                      2'b01: next_state = s3;
40                      2'b10: next_state = reset;
41                  endcase
42              s2:case(in)
43                      2'b00: next_state = s3;
44                      2'b01: next_state = s4;
45                      2'b10: next_state = reset;
46                  endcase
47              s3:case(in)
48                      2'b00: next_state = s4;
49                      2'b01: next_state = s5;
50                      2'b10: next_state = reset;
51                  endcase
52              s4:case(in)
53                      2'b00: next_state = s5;
54                      2'b01: next_state = reset;
55                      2'b10: next_state = reset;
56                  endcase
57              reset: next_state = s0;
58              default: next_state = reset;
59          endcase
60      end
```

```
61
62      always @(state) begin
63          case(state)
64              s5: out = 1;
65              reset: out = 1'b0;
66              default: out = 0;
67          endcase
68      end
69
70  endmodule
```

7.2 激励文件

7.2.1 序列检测器

```
1       module test_check_str(
2
3       );
4
5       reg clk;
6       reg rstn;
7       reg a;
8       wire out;
9
10      check_str check(
11          .clk(clk),
12          .rstn(rstn),
13          .in(a),
14          .out(out)
15      );
16
17      initial begin
18          clk=0;
19          rstn=1;
20          a=0;
21          #0.1 rstn=0;
22          #1.1 rstn=1;
23      end
```

```
24
25      always begin
26          #1 clk = ~clk;
27      end
28
29      always begin
30          #2 a = $random() % 2;
31      end
32
33  endmodule
```

7.2.2 周期输出序列

```
1   module test_periodic_output(
2
3   );
4
5   reg clk;
6   reg rstn;
7   wire out;
8
9   periodic_output start(
10      .clk(clk),
11      .rstn(rstn),
12      .out(out)
13  );
14
15  initial begin
16      clk = 0;
17      rstn = 1;
18      #0.1 rstn = 0;
19      #1.1 rstn = 1;
20  end
21
22  always begin
23      #1 clk = ~clk;
24  end
```

```
25
26 endmodule
```

7.2.3 报纸贩卖机

```
1  module check_newpaper_machine(
2      );
3
4  reg [1:0]in;
5  reg clk;
6  reg rstn;
7  wire out;
8
9  newspaper_machine start(
10     .clk(clk),
11     .in(in),
12     .rstn(rstn),
13     .out(out)
14 );
15
16 initial begin
17     clk = 0;
18     in = 0;
19     rstn = 1;
20     #0.1 rstn = 0;
21     #1.1 rstn = 1;
22 end
23
24 always begin
25     #1 clk = ~clk;
26 end
27
28 always begin
29     #2 in = $urandom_range(2, 0);
30 end
31
32 endmodule
```