2022《FPGA 应用实验》实验报告

实验编号：lab 3　　实验时间：2022.04.02

实验名称：序列检测模块设计

班级：F1903604　学号：519021910917　姓名：费扬

1、实验平台

采用 Xilinx 公司的 FPGA 集成开发环境 Xilinx ISE Design Suite 10.1 sp3，实验开发板为 Xilinx Spartan-3E FPGA Starter Kit。

2、实验设计要求：

序列检测模块设计：

1. 检测出串行输入数字序列 d 中出现的指定的序列模式：10110
2. 当在输入序列中检测到“10110”时，则，发现标志（flag）输出 1，否 则，置 0；

不支持重叠形式的有效的序列：

1. 当指定序列模式重叠形式出现时，只有第一个序列有效，即：

d: 0101_1011_0110_1011_0100

flag: 0000_0100_0001_0000_1000

实验要求：

1. 分别采用 Mealy 型、Moore 型 FSM 实现序列检测器；
2. 序列 d 中出现：10110，标志：flag= 1，否则，置 0；
3. 输入/输出端口

使用在 Spartan－3E FPGA Starter Kit Board 上的 LED0 作为检出标志 flag；

使用滑动开关 SW3 作为复位开关；

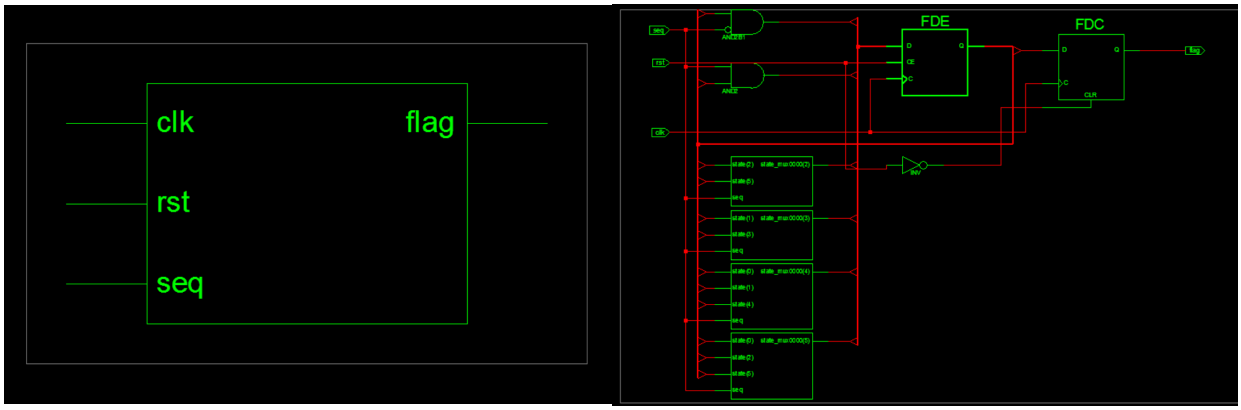
a）当 SW3 = 0 时，复位；

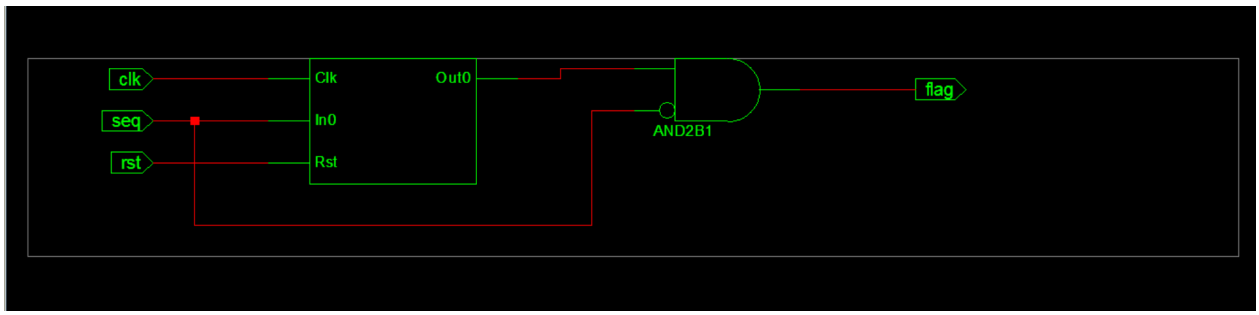
b）当 SW3 = 1 时，开始检测；

4. 使用开发板连接端口 6-pin header J1 的第 1 个管脚 J1<0>作为输入端口 d；

NET "J1<0>" LOC = "B4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;

3、模块设计框图





4、实验原理：

1. Mealy 机和 Moore 机原理(FSM)

有限状态机（Finite-State Machine，FSM）：

简称状态机，是表示有限个状态以及在这些状态之间的转移和动作等行为的数学模型。状态机不仅是一种电路的描述工具，而且也是一种思想方法，在电路设计的系统级和 RTL 级有着广泛的应用。

状态机类型:

Verilog 中状态机主要用于同步时序逻辑的设计，能够在有限个状态之间按一定要求和规律切换时序电路的状态。状态的切换方向不但取决于各个输入值，还取决于当前所在状态。状态机可分为 2 类：Moore 状态机和 Mealy 状态机。

Moore 型状态机

Moore 型状态机的输出只与当前状态有关，与当前输入无关。输出会在一个完整的时钟周期内保持稳定，即使此时输入信号有变化，输出也不会变化。输入对输出的影响要到下一个时钟周期才能反映出来。这也是 Moore 型状态机的一个重要特点：输入与输出是隔离开来的。

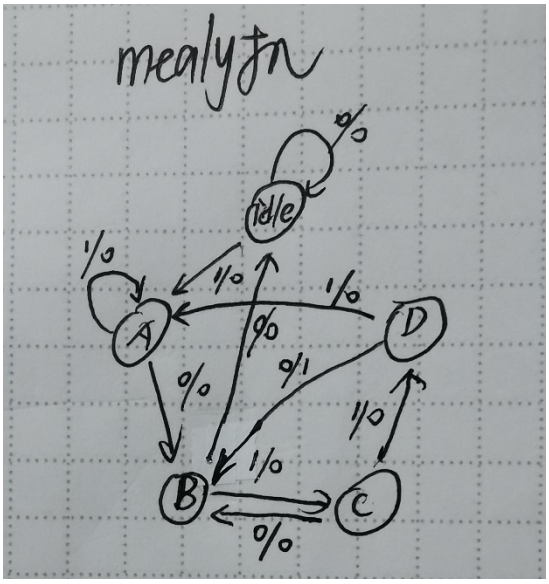
Mealy 型状态机

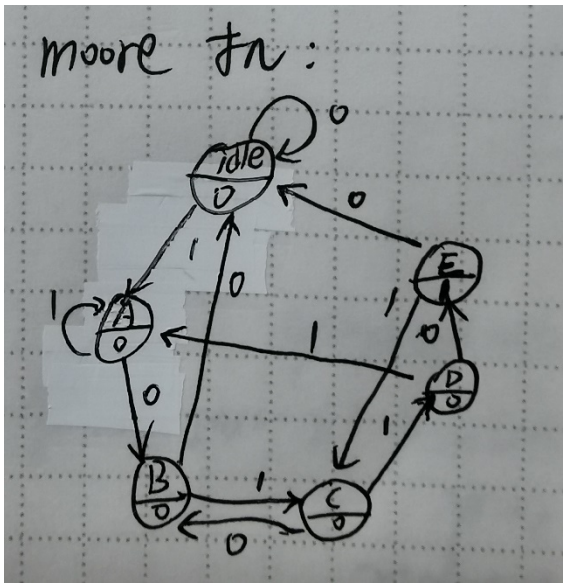
Mealy 型状态机的输出，不仅与当前状态有关，还取决于当前的输入信号。Mealy 型状态机的输出是在输入信号变化以后立刻发生变化，且输入变化可能出现在任何状态的时钟周期内。因此，同种逻辑下，Mealy 型状态机输出对输入的响应会比 Moore 型状态机早一个时钟周期。

状态机设计流程

根据设计需求画出状态转移图，确定使用状态机类型，并标注出各种输入输出信号，更有助于编程。一般使用最多的是 Mealy 型 3 段式状态机，下面用通过设计一个自动售卖机的具体实例来说明状态机的设计过程。

5. 序列检测器设计：





序列检测器顾名思义是用于检测一段给定的信号。可以使用状态机来直接描述，也可以采用移位寄存器的方式进行检测。

5、Verilog 模块设计

//米利型

/mealy.v

```
`timescale 10ns/1ns

module mealy(output reg flag, input seq, clk, rst);

  parameter IDLE=5'b0_0001, A=5'b0_0010, B=5'b0_0100, C=5'b0_1000, D=5'b1_0000;

  reg[4:0] p_state, n_state;

  always @(posedge clk,  negedge rst)
  if(!rst) p_state <= IDLE;
  else p_state <= n_state;

  always @(*) begin
     case (p_state)
  IDLE:  n_state = (seq) ? A:IDLE;
  A:  n_state = (seq) ? A:B;
  B:  n_state = (seq) ? C:IDLE;
  C:  n_state = (seq) ? D:B;
  D:  n_state = (seq) ? A:B;
default: n_state = IDLE;
endcase
flag = ((p_state == D) && (seq == 1'b0)) ? 1'b1 : 1'b0;
end

endmodule
```

//摩尔型

//moore.v

```
`timescale 10 ns / 1 ns
module moore(
    output reg flag,
    input seq, clk, rst
);
```

```
 parameter  IDLE=6'b00_0001,  A=6'b00_0010,  B=6'b00_0100,  C=6'b00_1000,  D=6'b01_0000,
E=6'b10_0000;

reg [5:0] state;

 always @(posedge clk or negedge rst)
  begin
    if(!rst) begin
      flag <= 1'b0;
  end
else begin
     flag <= (state == E) ? 1'b1 : 1'b0 ;
        case (state)
            IDLE:  state <= (seq) ? A:IDLE;
            A:  state <= (seq) ? A:B;
            B:  state <= (seq) ? C:IDLE;
            C:  state <= (seq) ? D:B;
            D:  state <= (seq) ? A:E;
            E:   state <= (seq) ? C:IDLE;
            default: state <= IDLE;
        endcase
end
end
endmodule
```

//引脚约束
//mealy.ucf/moore.ucf

```
NET "flag" LOC="F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8;
NET "clk" LOC = "C9" | IOSTANDARD = LVCMOS33;
NET "seq" LOC = "B4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6;
NET "rst" LOC = "L13" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6;
```

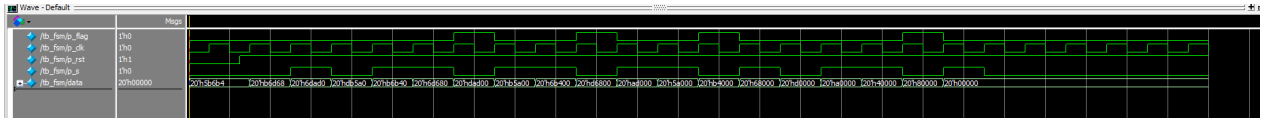
//test_bench
//tb_fsm.v

```
`timescale 10ns/1ns
`include "mealy.v"
//`include "moore.v"
```

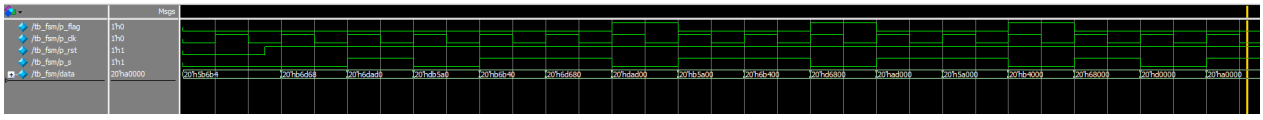
```
module tb_fsm;
    wire p_flag;
    reg p_clk, p_rst, p_s;

    initial begin p_rst = 1'b0; #25 p_rst = 1'b1; end
    initial begin p_clk = 0; forever #10 p_clk = ~p_clk; end

    parameter SIZE = 20;
    reg [SIZE-1 : 0] data = 20'b0101_1011_0110_1011_0100;

    initial begin: SERIES
        integer i;
        p_s = 0;
        #30;
        for (i=0;i<SIZE;i=i+1) begin
            p_s = data[SIZE-1];
            data=data<<1;
            #20;
        end
    end

    mealy u ( .flag(p_flag), .seq(p_s), .clk(p_clk), .rst(p_rst) );
    //moore v ( .flag(p_flag), .seq(p_s), .clk(p_clk), .rst(p_rst) );

    initial
  begin
  $monitor($time, "seq=%b, rst=%b, flag=%b", p_s, p_rst, p_flag);
  end
endmodule
```

6、试验仿真结果和分析

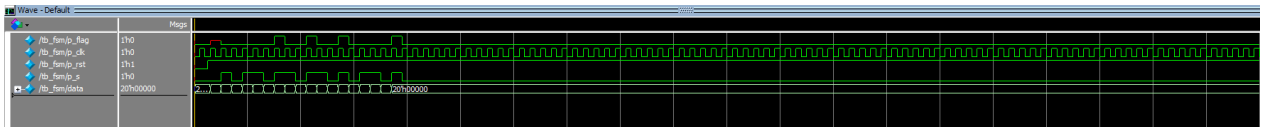
如图：功能仿真中，正确显示 flag。



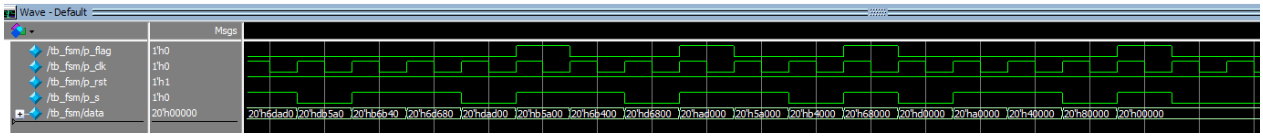
Mealy：Test 1



Mealy：Test 2



Moore：Test 1

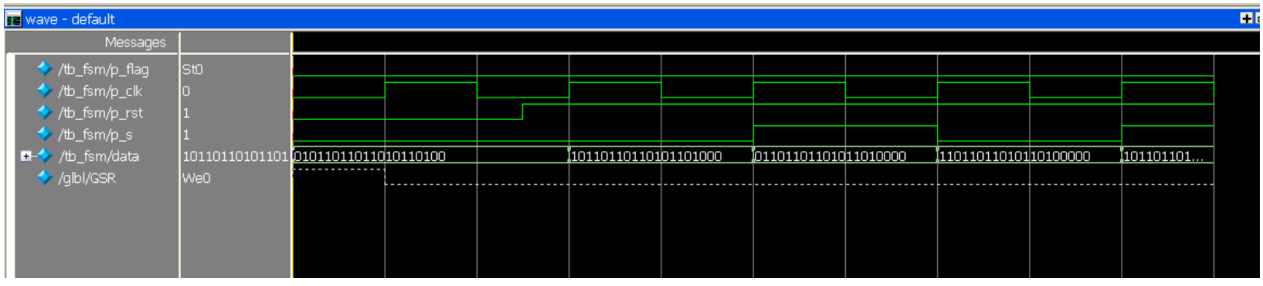


Moore：Test 2

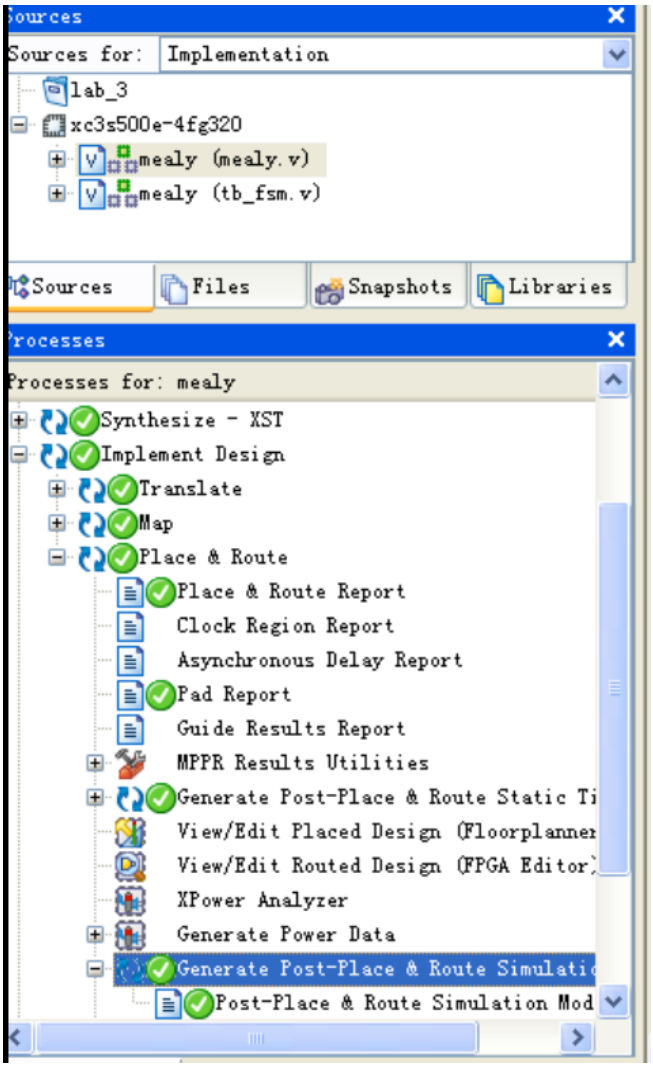
```
  0 rst=0, flag=0
 25 rst=1, flag=0
130 rst=1, flag=1
150 rst=1, flag=0
190 rst=1, flag=1
210 rst=1, flag=0
250 rst=1, flag=1
270 rst=1, flag=0
350 rst=1, flag=1
370 rst=1, flag=0
```

```
 50 rst=1, flag=0
150 rst=1, flag=1
170 rst=1, flag=0
210 rst=1, flag=1
230 rst=1, flag=0
270 rst=1, flag=1
290 rst=1, flag=0
370 rst=1, flag=1
390 rst=1, flag=0
```

在对应时刻显示 Flag 为 1 的情况 (左 mealy 右 moore)



后仿真(但没有明显延时)



以 mealy 机为例，执行仿真和实现之后的全仿真过程