《数字逻辑》实验报告

姓名	李燕琴 杨思怡	**年级**	2019 级
学号	20195633 20195217	**专业、班级**	计算机科学与技术（卓越）02 班 01 班
实验名称	实验十六 米里状态机序列检测器		
实验时间	2020/11/27	**实验地点**	DS1410
实验成绩		**实验性质**	**□验证性 □设计性 □综合性**

教师评价：

□算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；

□实验结果正确； □语法、语义正确； □报告规范；

评语：

评价教师签名（电子签名）：

一、实验目的

通过实验，掌握有限状态机的设计方法，并用米里状态机设计实现“1011” 序列检测。

二、实验项目内容

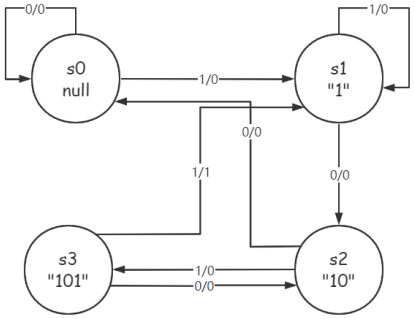
设计实现一个米里状态机，能检测一个 8 位的二进制数据中是否存在“1011”序列。

三、实验设计

1、实验原理

Mealy 型电路中，结果输出与输入和现态都有关。

状态转换图：



状态表：

现态	次态		输出	
	x=0	x=1	x=0	x=1
s0	s0	s1	0	0
s1	s2	s1	0	0
s2	s0	s3	0	0
s3	s2	s1	0	1

2、应用：序列检测器在很多数字系统中都不可缺少，尤其是在通信系统当中。序列检测器的作用就是从一系列的码流中找出用户希望出现的序列，序列可长可短。比如在通信系统中，数据流帧头的检测就属于一个序列检测器。序列检测器的类型有很多种，有逐比特比较的，有逐字节比较的，也有其他的比较方式，实际应用中需要采用何种比较方式，主要是看序列的多少以及系统的延时要求。

四、实验过程或算法

1、次现态转换，写在一个 always 模块中，根据状态表（图），得到相应的 case 语句。

```
// 次现态转换
always@(posedge myclk,posedge rst) begin
    if(rst) begin
        curr <= s0;
        next <= s0;
    end
    else begin
        case(curr)
            s0: next = x ? s1 : s0;
            s1: next = ~x ? s2 : s1;
            s2: next = x ? s3 : s0;
            s3: next = x ? s1 : s2;
        endcase
        curr <= next;
    end
end
```

2、时钟分频模块

板子自带时钟 100MHz，1 秒有 10^8 个时钟上升沿，通过计数 count=5*10^7 次时钟上升沿，自定义时钟翻转一次，可以得到周期为 1s（即频率为 1Hz）的新的时钟。

```
module fdivider(clk,f,myclk);
    input clk;
    input [31:0]f;
    output myclk;

    reg[31:0] clk_cnt=1'b0;
    reg inlineclk = 0;
    always@(posedge clk) begin
        if(clk_cnt == f[31:0]) begin
            clk_cnt<=1'b0;
            inlineclk <=~inlineclk;
        end
        else clk_cnt<=clk_cnt+1'b1;
    end
    assign myclk = inlineclk;
endmodule
```

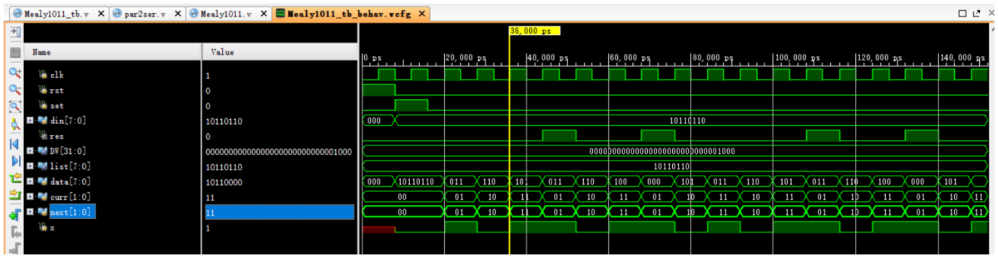
3、数据输入后进行并转串处理: 每一个时钟上升沿，通过获取最高位和逐步左移，得到并转串的数据 x。

```
// 内置左移模块（输入数据，得到右移结果，异步重置）
reg x;
reg [DW-1:0]data={DW{1'b0}};
always@(posedge myclk,posedge rst,posedge set) begin
    if(rst) begin
        data = {DW{1'b0}};
        x = 1'b0;
    end
    else if(set) begin
        data = din;
        x = 1'b0;
    end
    else begin
        x = data[DW-1];
        data = data << 1;
    end
end
```

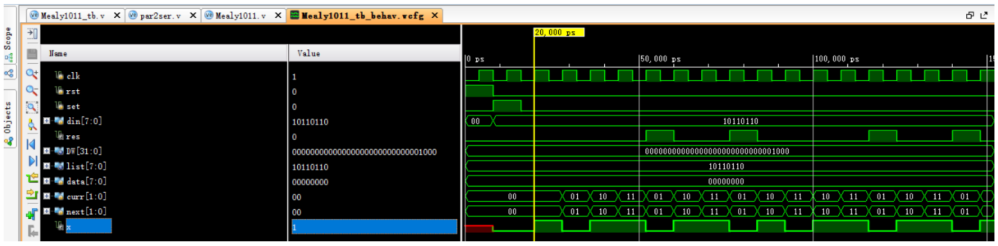
五、实验过程中遇到的问题及解决情况

1、实验最开始想通过调用 par2ser 并转串模块来进行并转串处理，但是仿真中发现调用 module 模块与嵌入 always 模块相比较会慢一个时钟，即检测到 1011 序列的下一个时钟才有 res=1。故为了能在检测到 curr==s3&x==1 时就输出信号，采用的是嵌入 always 模块。仿真记录如下：

1、使用内置always模块进行并转串(reg x)



2、调用子模块进行并转串wire x

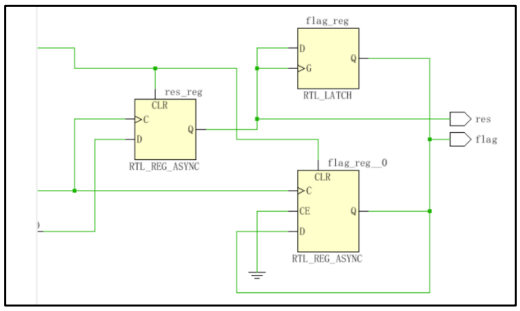


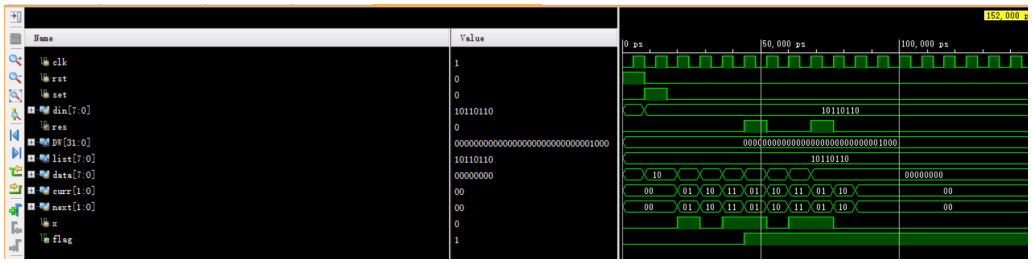
2、代码中的 res 信号表示一旦该时钟检测到信号就亮，否则就暗；而 flag 信号，表示如果检测到存在该序列，led 灯就常量；在设置 flag 时，最开始是将其单独放在一个 always 模块中，代码如下，但是结果 flag 并没有达到常亮的效果，而是和 res 同步变化。但是二者仿真效果均是常亮。之后问了老师，但仍没有懂其中的原因，不过通过将 res 和 flag 放在同一个模块中得到了常量的效

果。

放在不同 always 模块的代码、生成的 rtl 电路、仿真效果：

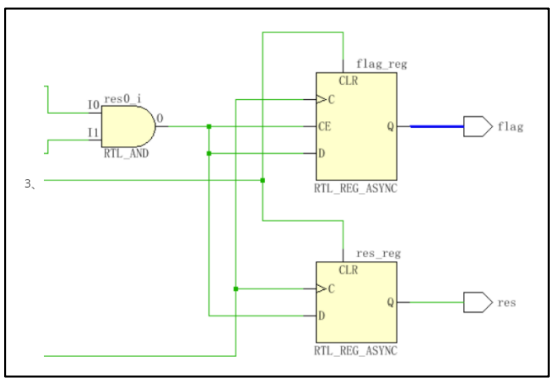
```
//    always@(res) begin
//        if(res) flag = 1'b1;
//        else flag = flag;
//    end
```

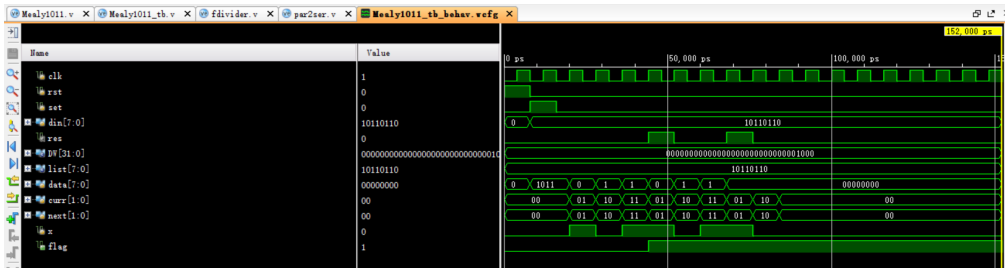




放在同一 always 模块的代码以及生成的 rtl 电路：

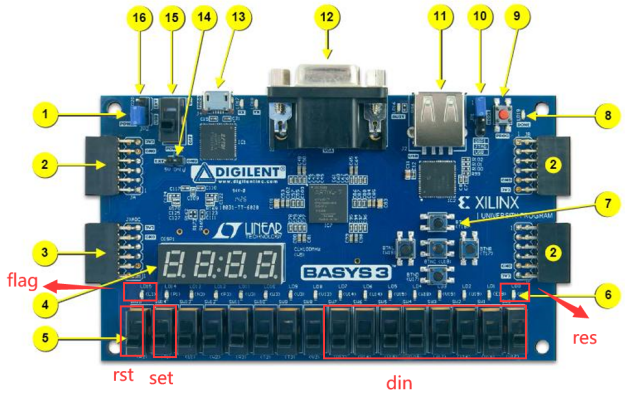
```
        else begin
            if(curr==s3 && x==1) begin
                res = 1;
            end
            else res = 0;
            if(res) flag = 1'b1;
        end
    end
```



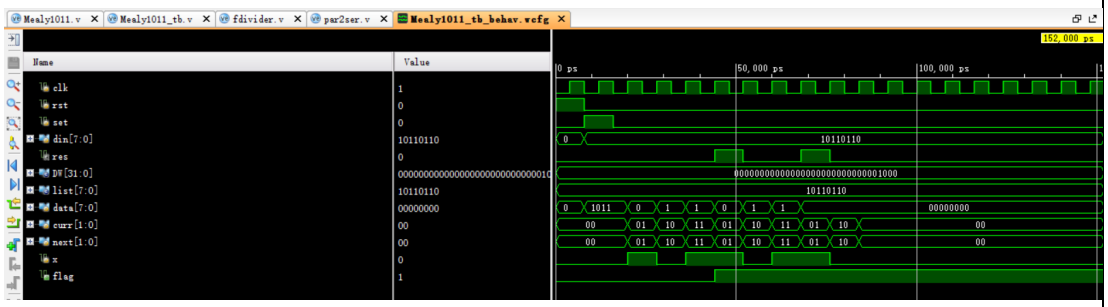


六、实验结果及分析和（或）源程序调试过程

1、端口设计



2、仿真结果



3、上板实验

见附件视频，视频中 din = 8’b1011_0110。

七、小组分工情况说明

李燕琴：代码编写

杨思怡：代码优化，仿真上板