上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

实验报告

课程名称：数字逻辑设计

项目名称：Lab2

学院(系)：电子信息与电气工程学院

专　　业：微电子科学与工程

学生姓名：王赟恺　　学号：522031910274

2024 年 05 月 04 日

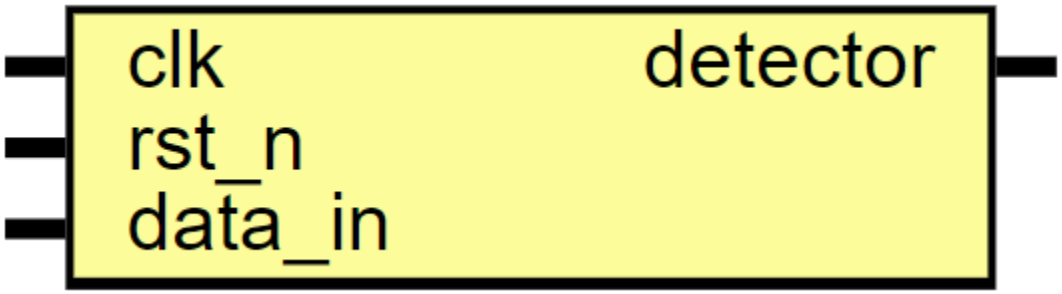
目录

1 序列检测

1.1 实验电路功能

随机序列中 xxx101101xxx 检测序列“101101”（左先），每检测到一个序列输出一个周期的脉冲，输入:clk,rstn,data_in，输出:detector， 可检测重叠的 101101 序列(改进后)。

1.2 实验电路模块端口图

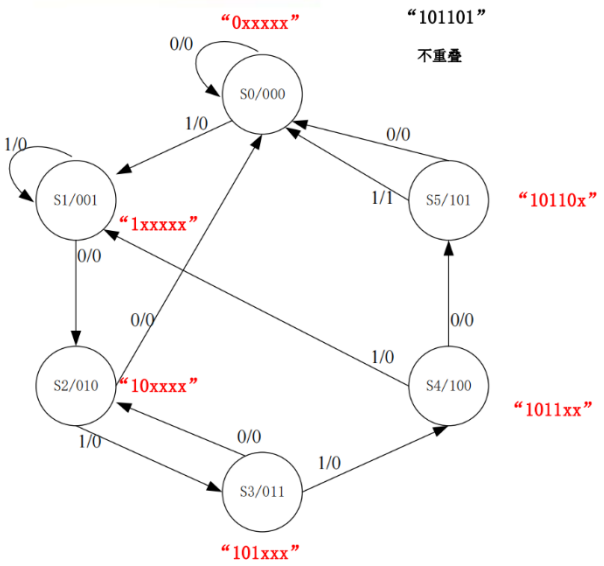


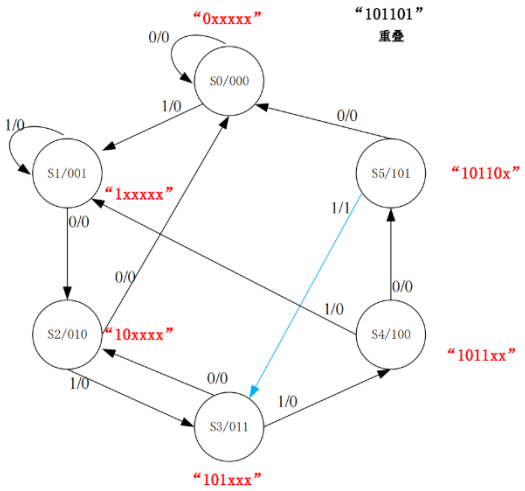
序列检测模块端口图

1.3 设计思路

代码的核心部分是状态机，功能由状态机的状态转移过程决定，在提供的初始代码中，状态机到达序列 101101 检测的最终状态 State5 后，下一状态默认回到了 State0;正是由于这部分代码忽略了最后状态的输入对状态转移的影响，导致了序列只能检测不重叠的目标序列，因此代码的更改思路在于在 State5 根据输入确定下一个状态，观察序列的重复规律，确定不同输入对应的下一个状态。

重叠检测和不重叠检测的状态转移图





1.4 代码分析

时序部分:

```
always @(posedge clk or negedge rst_n) begin
    if(!rst_n)begin
        state_cur <= STATE_0;
    end
    else begin
        state_cur <= state_next;
    end
    end
```

状态转移部分:

```
always @(*)begin
    case(state_cur)
        STATE_0:
            if(data_in) state_next = STATE_1;
            else        state_next = STATE_0;
        STATE_1:
            if(data_in) state_next = STATE_1;
            else        state_next = STATE_2;
        STATE_2:
```

```
                if(data_in) state_next = STATE_3;
                else        state_next = STATE_0;
            STATE_3:
                if(data_in) state_next = STATE_4;
                else        state_next = STATE_2;
            STATE_4:
                if(data_in) state_next = STATE_1;
                else        state_next = STATE_5;
            STATE_5:
                if(data_in) state_next = STATE_3;
                else        state_next = STATE_0;
            default:        state_next = STATE_0;
        endcase
        end
```

输出部分:

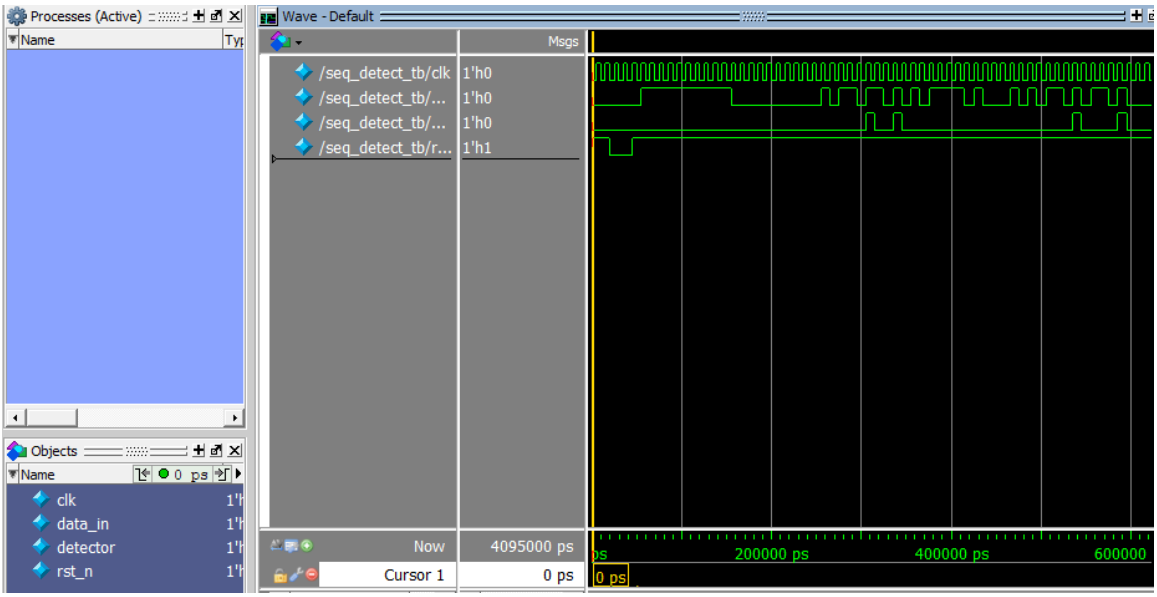
```
    assign detector = (state_cur == STATE_5) &&
data_in;
```

序列检测信号无延迟输出。

1.5 测试代码分析

测试部分代码涵盖了复位信号有效，非重叠和重叠 101101 序列出现的情况，仿真波形显示更改后的代码对非重叠序列和重叠序列都能实现检测功能。我的测试代码手工编写了重叠序列的几种情况，另一种标准的测试代码是用$random 函数生成大量随机输入序列，再将输出与标准模型比较。

1.6 波形图



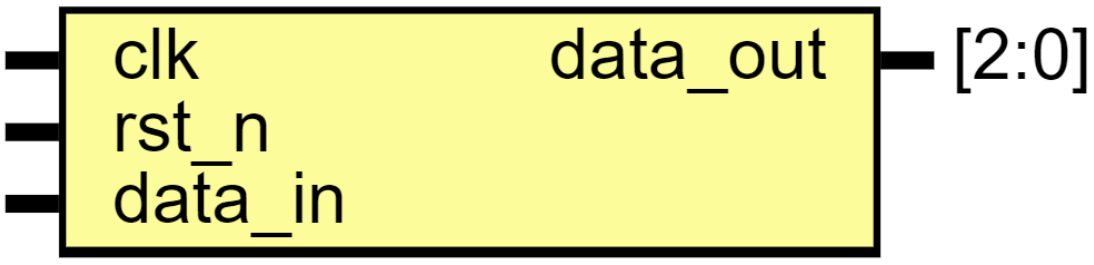
如图，复位后，能够检测出 101101 序列波形，对于重叠序列的两种情况都能完成检测功能。

2 求余运算

2.1 实验电路功能

现有 16 位寄存器。初始值为 0。每个时钟周期寄存器的值会左移 1 位，并且将输入的数据 data_in 作为寄存器的最低位，寄存器原来的最高位将被丢弃。要求每个周期实时输出该 16 位寄存器对 7 求余的余数 data_out[2:0]。

2.2 实验电路模块端口图



2.3 设计思路

核心部分在于移位寄存器的实现，根据课上的代码规范，可以用拼接信号的方法非阻塞赋

值，。每个周期该语句执行一次，将新的输入信号赋值至最低位寄存器，将寄存器的原先低位赋值到高位，从而实现对寄存器最高位的丢弃。取余数部分，电路利用 Melay 状态机实现，定义七个状态 STATE0~STATE6，分别对应当前寄存器值除以七的余数，根据寄存器最高位和 data_in 的值执行状态转换。

2.4 代码分析

```
always@(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        tmp[15:0] <= 16'b0;
    else
    begin
        tmp[15:0] <= {tmp[14:0], data_in};
    end
    end
```

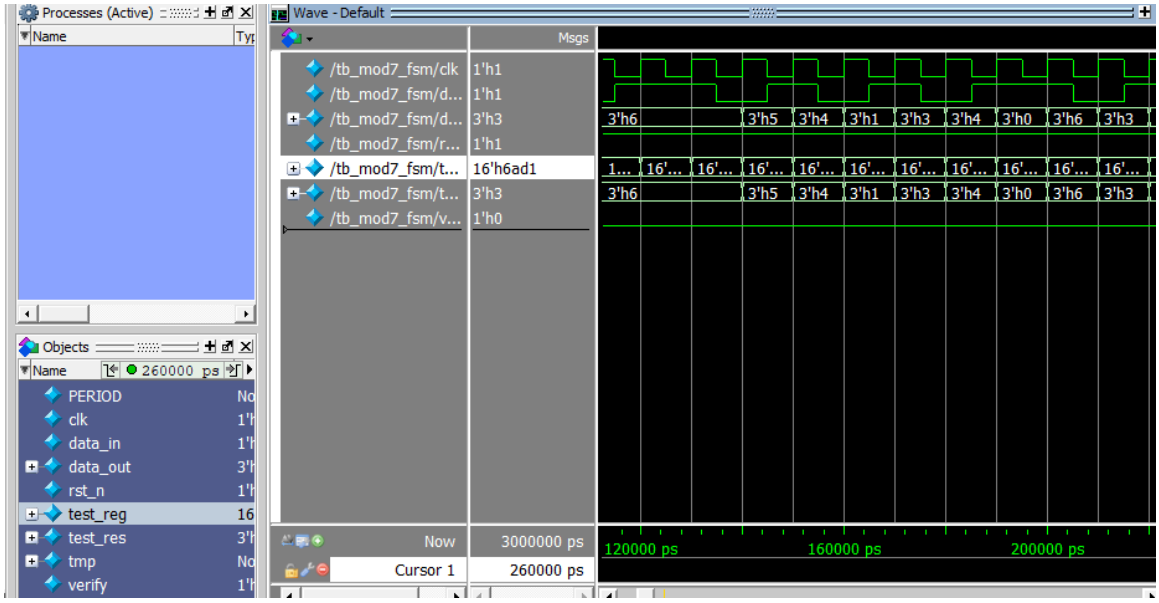
用时序电路实现移位寄存器，复位信号有效后，16 位寄存器全部置为 0。

状态机部分代码见代码文件。

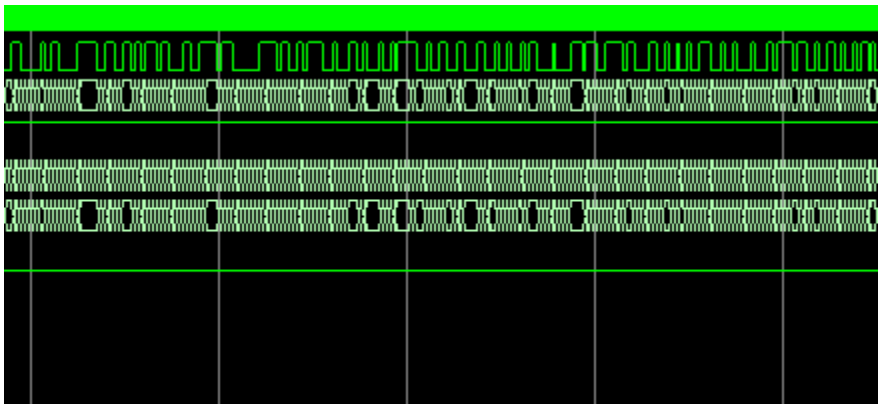
2.5 测试代码分析

测试代码用$random 生成输入信号序列 data_in 激励，在模块外用同样跟踪一组移位寄存器，并采用更简单的取余数运算%7 得到标准结果，verify 信号为 0 是表示模块结果与标准模型结果一致，为 1 表示不一致。注意输入信号序列激励在时钟下降沿产生，防止产生亚稳态的情况。

2.6 波形图



如图，模块结果和标准模型结果相一致，verify 输出为 0.



更多的测试样例截图
可见代码正确地实现了模块功能。