

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机组成 |
| 姓 名： | 应周骏 |
| 学 院： | 计算机科学与技术学院 |
| 系： | 计算机科学与技术系 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3200103894 |
| 指导教师： | 马德 |

2022 年 3月 17 日

**浙江大学实验报告**

课程名称： 计算机组成 实验类型： 综合

实验项目名称： ALU和有限状态设计

学生姓名： 应周骏 专业：计算机科学与技术 学号：3200103894

同组学生姓名： 无 指导老师： 马德

实验地点： 东4-509实验日期：2022 年 3 月 17 日

**一、实验目的和要求**

Part Ⅰ

1. 复习寄存器传输控制技术；

2. 掌握CPU的核心组成：数据通路与控制器；

3. 设计数据通路的功能部件；

4. 进一步了解计算机系统的基本结构；

5. 熟练掌握IP核的使用方法；

Part Ⅱ

1. 复习有限状态机的基本概念；

2. 掌握有限状态机的两种模型；

3. 设计有限状态机解决实际问题；

**二、实验内容和原理**

**内容：**

1. 设计实现数据通路部件ALU；

2. 设计实现数据通路部件Register Files；

3. 设计有限状态机完成序列检测器并测试；

**原理：**

**1. 实验一：ALU设计**

实现5个基本运算，要求整理逻辑实验的ALU，实现verilog输入并仿真，并拓展ALU的功能。

表2.1.1 ALU功能

|  |  |  |
| --- | --- | --- |
| ALU Control Lines | Function | note |
| **000** | **And** | **兼容** |
| **001** | **Or** | **兼容** |
| **010** | **Add** | **兼容** |
| **110** | **Sub** | **兼容** |
| **111** | **Set on less than** | -- |
| **100** | **nor** | **扩展** |
| **101** | **srl** | **扩展** |
| **011** | **xor** | **扩展** |

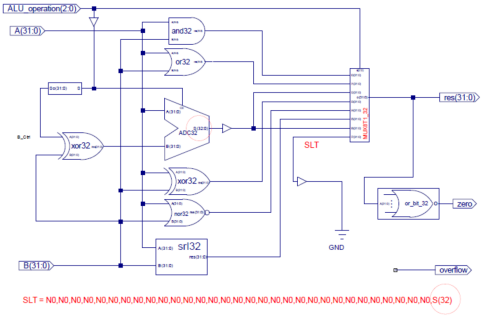


图2.1.1 ALU原理图

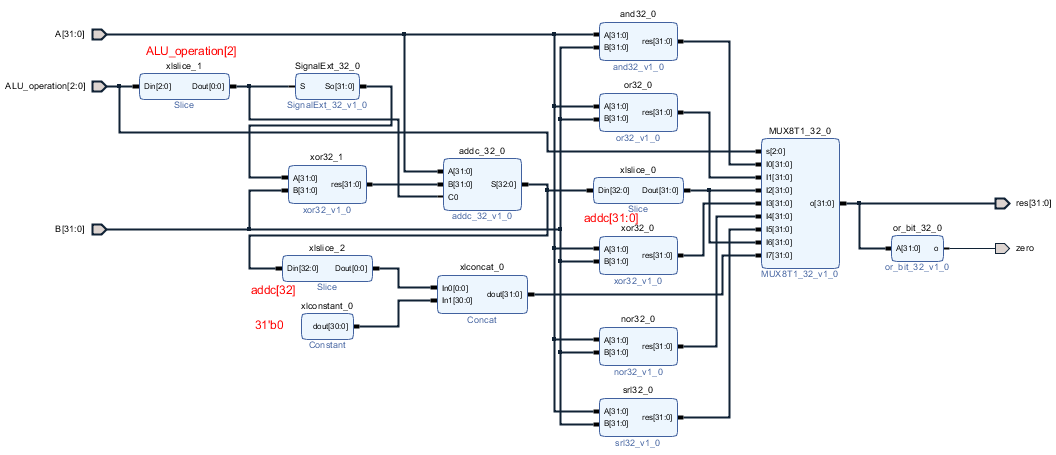


图2.1.2 ALU顶层逻辑连接

**2. 实验二：Register files 设计**

要求实现32×32bit寄存器组，优化逻辑实验Regs，并完成行为描述和仿真

端口要求：

二个读端口：Rs1\_addr,Rs1\_data；Rs2\_addr,Rs2\_data。

一个写端口，带写信号:Wt\_addr, Wt\_data, RegWrite。

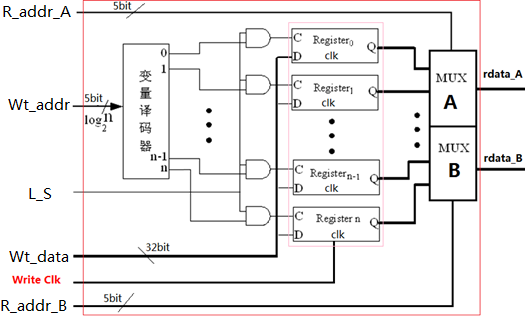


图2.2.1 Register Files原理图

**3. 实验三：有限状态机设计**

**3.1 状态机定义**

有限状态机（Finite State Machine，简称FSM）在有限个状态之间按一定规律转换的时序电路。

有限状态机通常是由寄存器组和组合逻辑组成时序电路，根据当前状态和输入信号可以控制下一个状态的跳转，有限状态机在电路中通常是作为控制模块，作为整个电路模块的核心而存在。

**3.2 状态机分类**

它主要包括两大类：Mealy型状态机和Moore型状态机。

Mealy型状态机:其组合逻辑的输出不仅与当前状态有关，还与输入有关。



图2.3.2.1 Mealy状态机示意图

Moore型状态机：其组合逻辑的输出只与当前的状态有关。



图2.3.2.2 Moore状态机示意图

状态寄存器由一组触发器组成，用来记忆状态机当前所处的状态，状态的改变只发生在时钟的跳变沿。状态是否改变、如何改变，取决于组合逻辑F的输出，F是当前状态和输入信号的函数。状态机的输出是由输出组合逻辑G提供的，G也是当前状态和输入信号的函数。

**3.3 状态机设计方法**

一段式描述：即状态跳转与输出信号都在同一个always块里面进行描述；

二段式描述：即将输出信号,与状态跳转分开描述，便于设计代码管理；

三段式描述：即将输出信号,与状态跳转分开描述，并且状态跳转用组合逻辑来控制；

**3.4 状态机设计步骤**

系统架构和接口定义：

表2.3.4.1 接口定义

|  |  |
| --- | --- |
| **接口** | **接口定义** |
| **clk** | 系统时钟 |
| **rst\_n** | 系统复位 |
| **X** | 序列输入 |
| **Y** | 检测输出 |

状态定义和编码：

状态机的编码方式主要包括二进制码（Binary），格雷码（gray），独热码（one hot）。格雷码相对于二进制码而言，在状态跳转的时候，只有单比特翻转，它的功耗相对比较低。独热码相对于格雷码或者二进制码而言，它增加了两个寄存器来表示状态，但是它会更节省组合逻辑电路，因为它在比较状态的时候，只需要比较一个比特位，那么其电路的速度和可靠性就会增加。

绘制状态转换图：



图2.3.4.1 状态转换图示例

RTL实现：通过HDL语言将状态转换图进行描述实现设计。

**三、实验过程和数据记录**

**实验一：ALU设计**

**1. 工程文件建立**

新建工程文件，命名为“Exp01\_ALU”。

**2. Top模块设计**

新建Verilog文件“ALU. v”，设为顶层模块，输入以下代码。

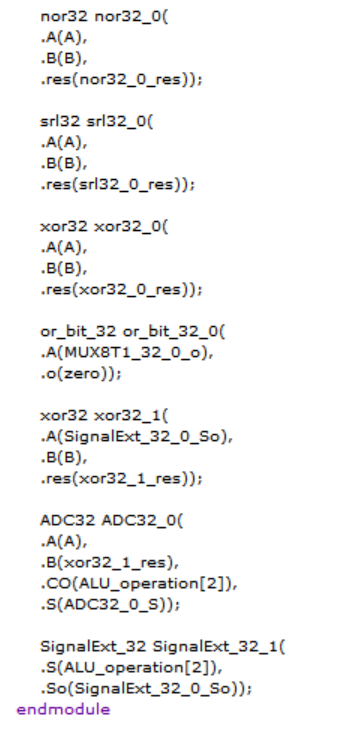
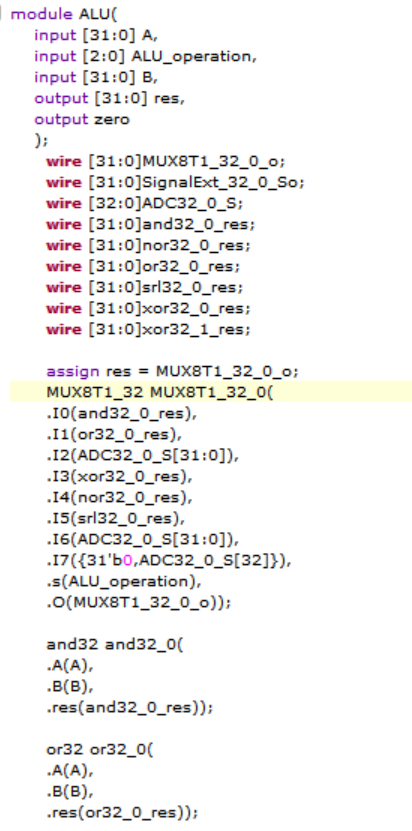


图3.1.1 ALU结构化描述

**3. 模块调用及仿真**

向工程中加入实验0完成的and32、or32、ADC32、xor32、nor32、srl32、SignalExt\_32、mux8to1\_32、or\_bit\_32模块，并对ALU进行仿真，输入仿真代码如下。

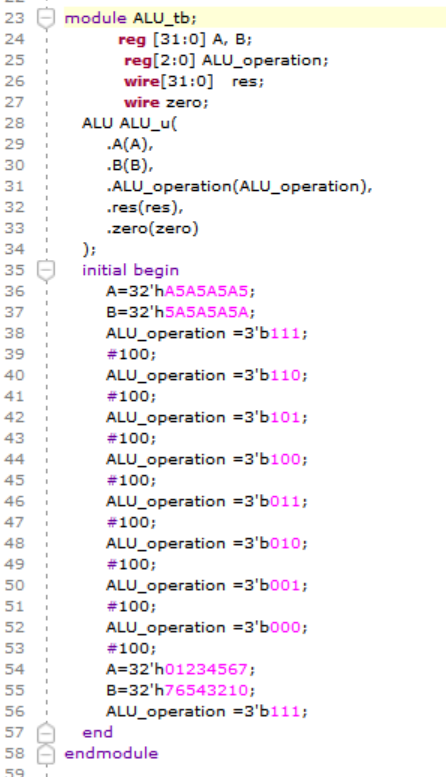


图3.1.2 ALU仿真代码

得到仿真结果图如下：

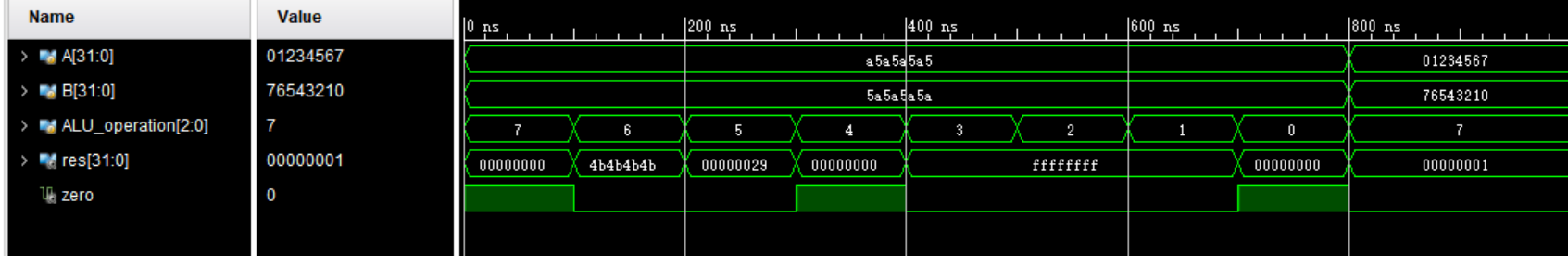


图3.1.3 ALU仿真结果图

**4. 补充：功能性描绘实现**

在结构化描述之外，用功能性描述重新设计ALU，修改ALU top模块，输入代码如下，得到的仿真结果一致。

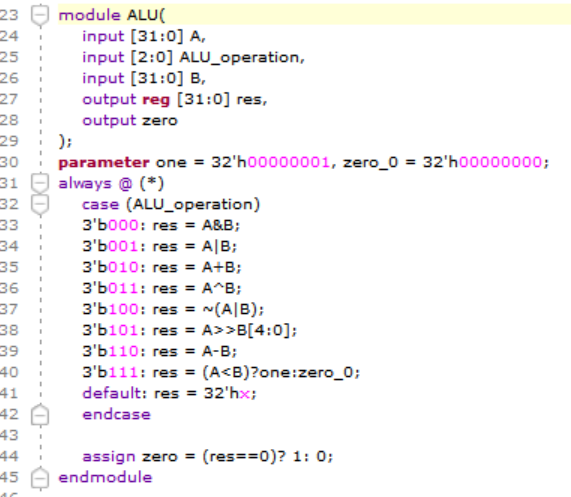


图3.1.4 ALU功能性描述

**实验二：Register File设计**

**1.工程文件建立**

新建工程文件“Exp01\_Reg”。

**2. Register模块**

新建Verilog文件“regs.v”，输入如下代码。

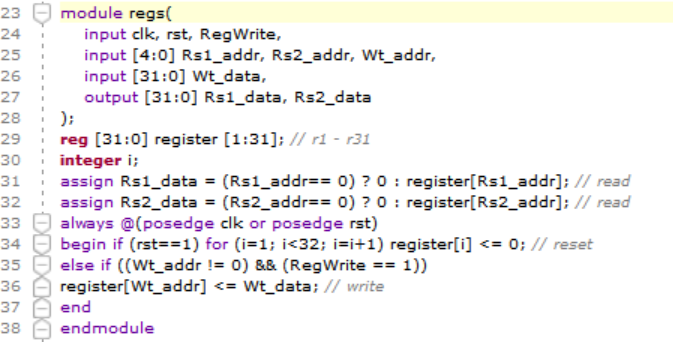


图3.2.1 Register代码

对该顶层文件进行仿真，输入如下仿真代码。

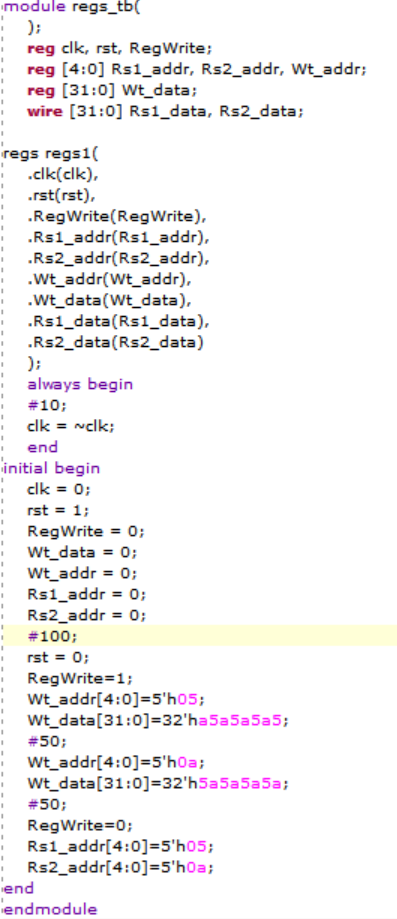


图3.2.2 Register仿真代码

得到仿真结果如下。

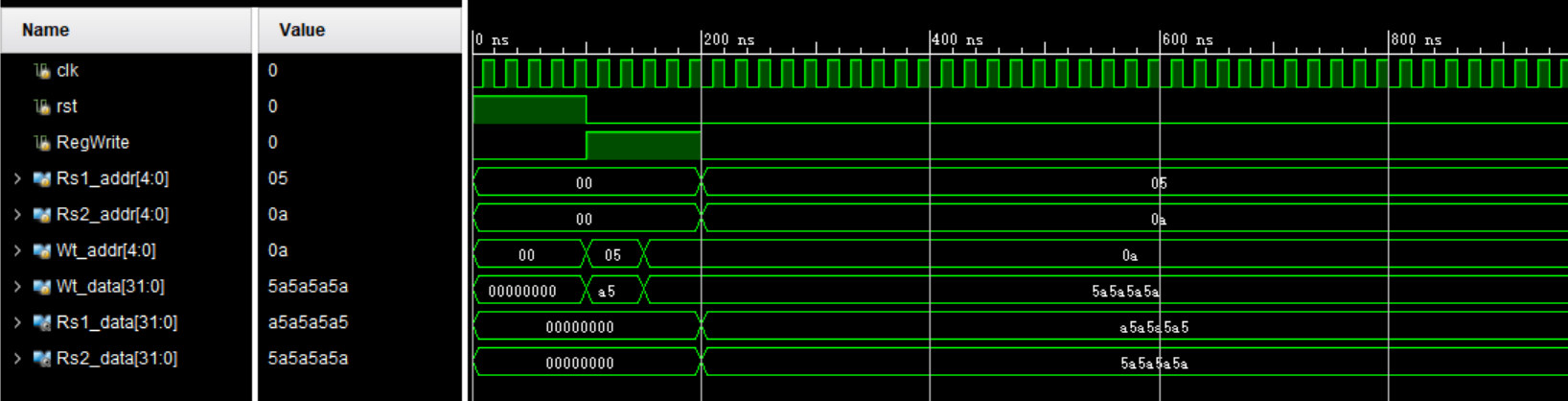


图3.2.3 Register仿真结果图

**实验三：有限状态机设计**

**1. 工程文件建立**

新建工程文件，命名“Exp01\_FSM”。

**2. 状态机设计**

新建Verilog文件“seq.v”，输入以下代码。

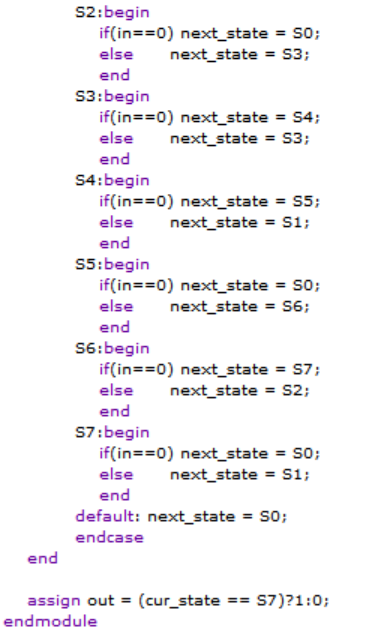
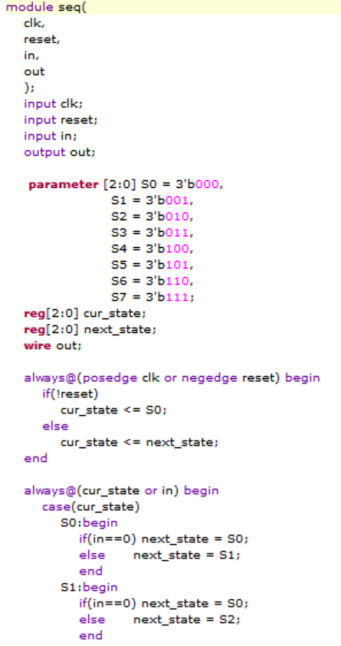


图3.3.1 有限状态机代码

对该原理图进行仿真，输入如下仿真代码。

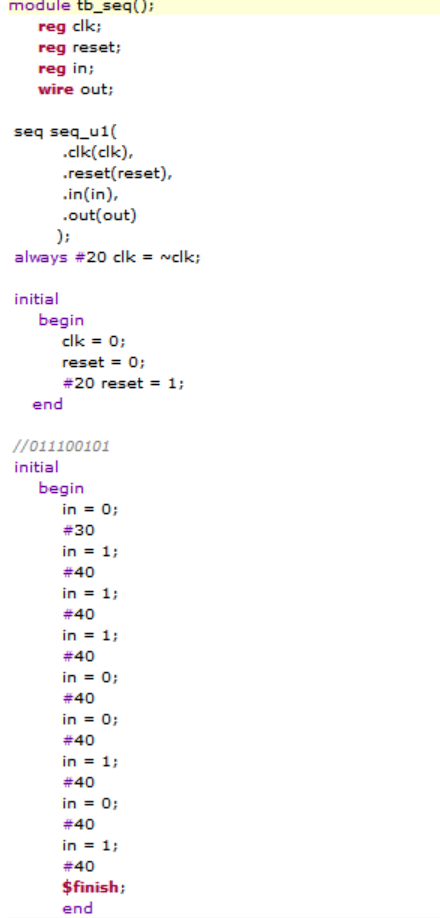


图3.3.2 有限状态机仿真代码

得到仿真结果。

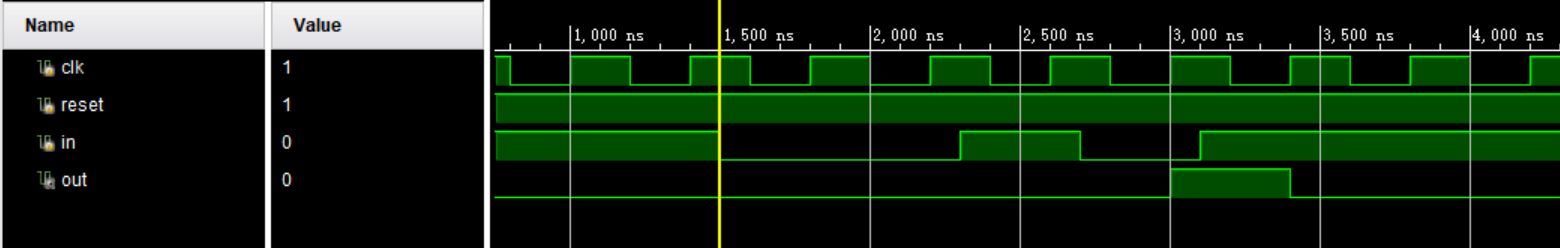


图3.3.3 有限状态机仿真结果图

**四、实验结果分析**

**实验一：ALU设计**

**1. ALU代码分析**

对于结构化描述代码，此处不做展开，其原理即根据ALU顶层逻辑图，描述图中给出的电路，用Verilog语言代替电路连接。

对于功能性描述代码，可以看到其整体用一个always语句，对于operation的改变做出响应。由于前期实验0中实现的元件的Verilog代码大多十分简洁，使用功能性描述，可以很大提高编程效率，但是整体上架构上，使用元件能够使得结构更加清晰，有助于后续其他复杂模块设计

**2.ALU仿真结果分析**

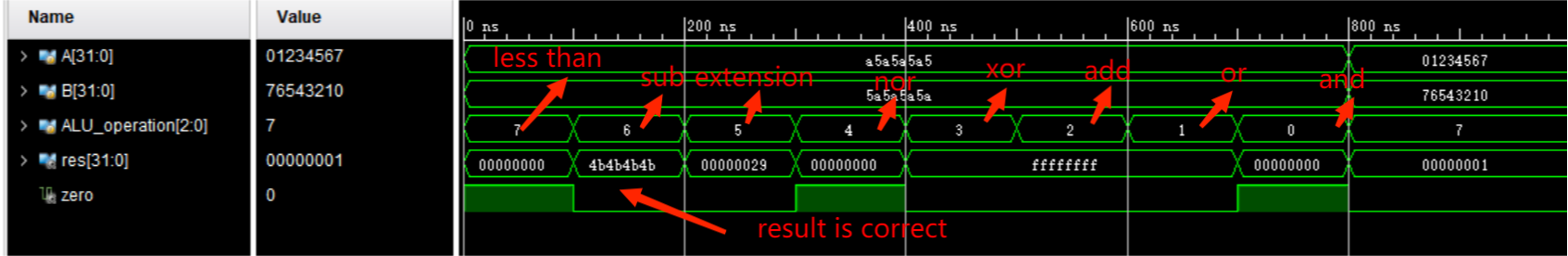


图4.1.1 ALU仿真结果分析

在仿真时，ALU\_operation表示的操作可以对照ALU设计要求，上图也进行了标注，可以看到算数操作结果达到预想要求。

**实验二：Register Files**

**1. 代码分析**

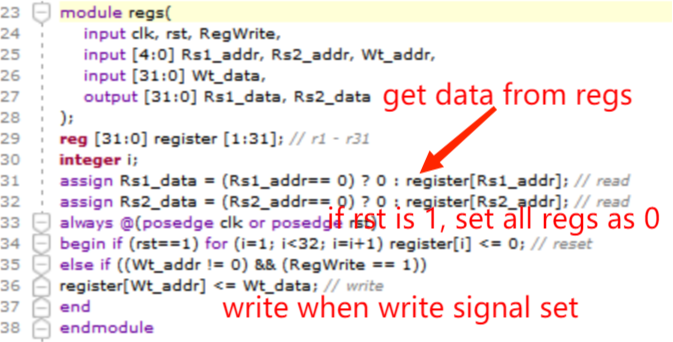


图4.2.1 Register代码分析

**2. 仿真结果分析**

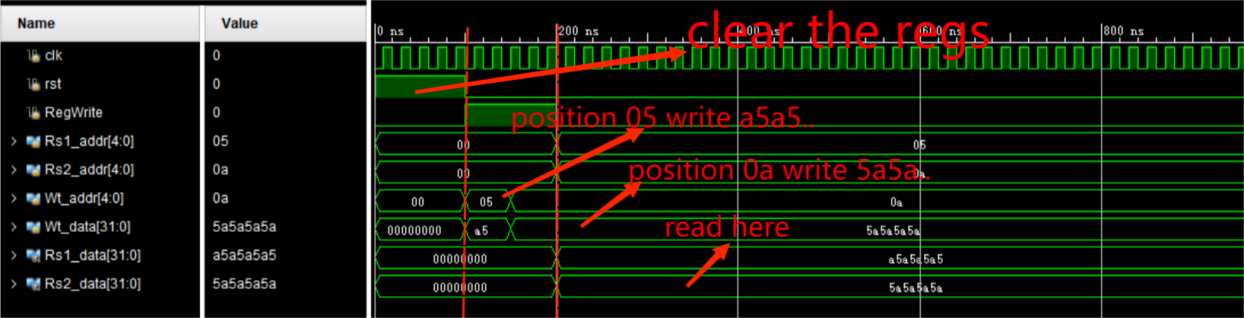


图4.2.2 Register仿真分析

首先，在最开始，rst置1，表示清零；此后write信号置1，此时，我们可以看到对于地址0x05，我们写入a5a5a5a5；对于地址0x0a，我们写入了5a5a5a5a。此时Rs\_1，Rs\_2的读取地址为0，则不读内容。当write信号置0后，我们对两个寄存器的读取地址做更改，则读到的内容符合我们写入的内容，符合实验预期。

**实验三：有限状态机设计**

**1. 状态机代码分析**

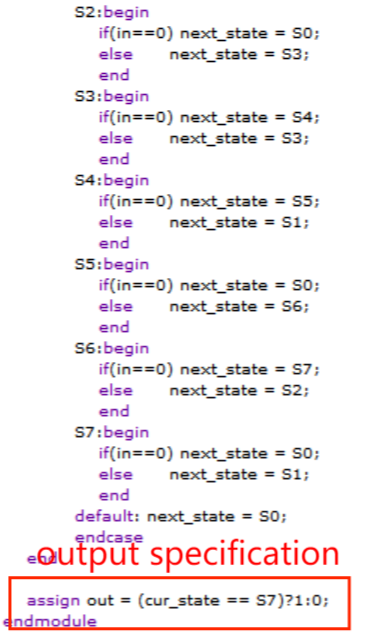
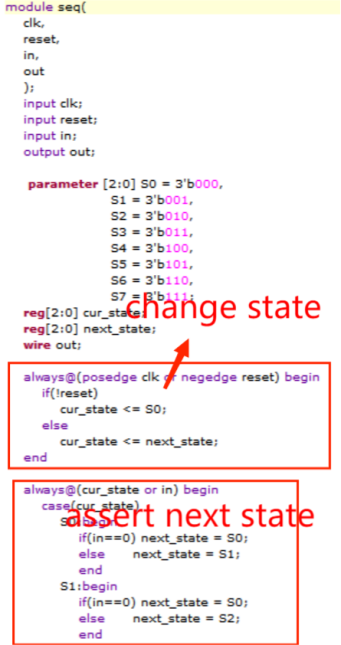
****

图4.3.1 FSM代码分析

**2. 状态机仿真分析**

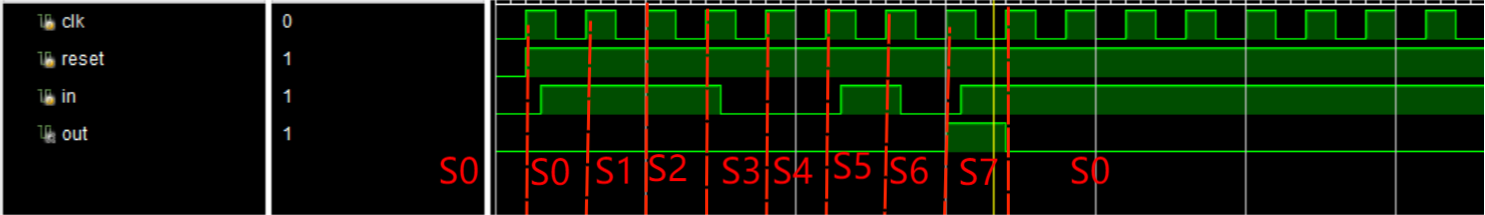
****

图4.3.2 FSM仿真结果分析

对于该段仿真，我们可以通过代码中第二段，即next state确定这段代码为突破点，分析各时钟周期状态机状态值，具体分析见上图。

**五、讨论与心得**

1. 通过本次实验，我复习了数字逻辑设计实验的相关内容，也进一步熟悉了Vivado实验平台的操作方法。同时，注意到本次实验对于Register Files的设计，不同于之前数字逻辑实验，本次实验采用的是Write的数据/地址与Register本身分离，且一个地址对应于一个数值，这样的设计比起之前原理图的寄存器设计更简洁，且能够方便的操作。

2. 本次实验也是对于实验0设计的几个元件的检查，发现其中依然存在一些错误，也顺利通过本次实验，纠正了一些Verilog代码的问题，能够方便后续进一步调用。

3. 本次实验整体相对顺利，为后续CPU设计做好铺垫。