# 第2章 数字逻辑电路设计基础

## 2.1 实验二 寄存器堆仿真

### 1 实验目的

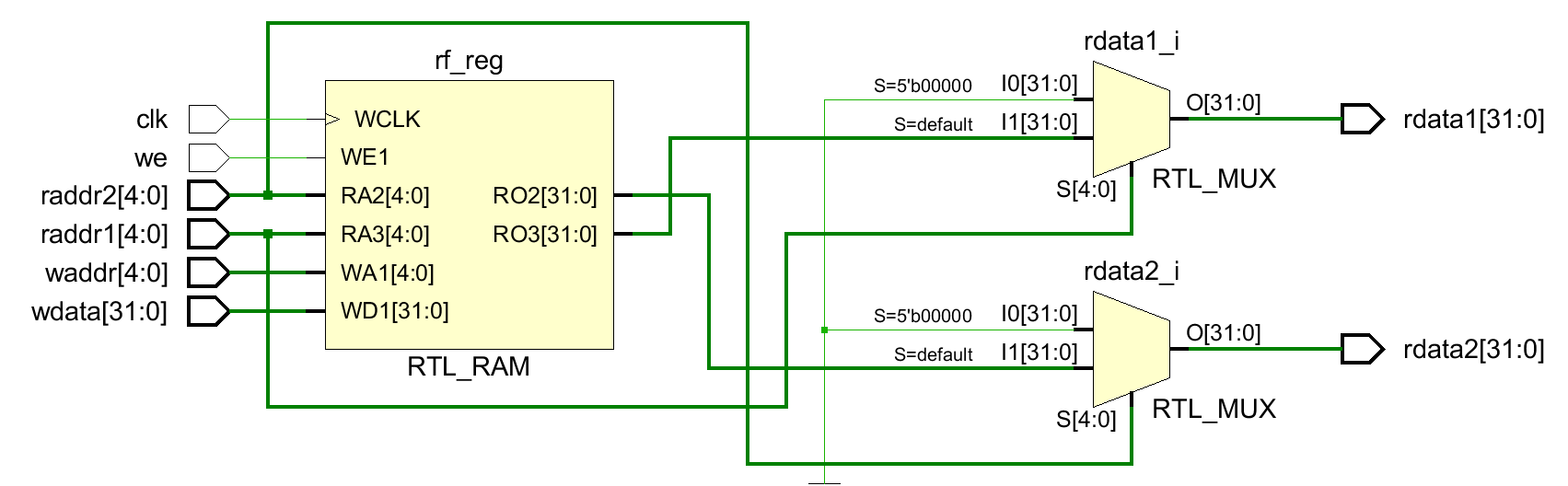
1. **掌握寄存器堆设计方法。**
2. **掌握功能仿真，能够通过观察其仿真波形了解行为特征。**

### 2 实验设备

Vivado硬件编程环境

### 3 实验内容与主要步骤

1. **画出结构框图，标出输入输出端口。**

****

1. **对工程进行仿真测试，结合波形观察寄存器堆的读写行为。**

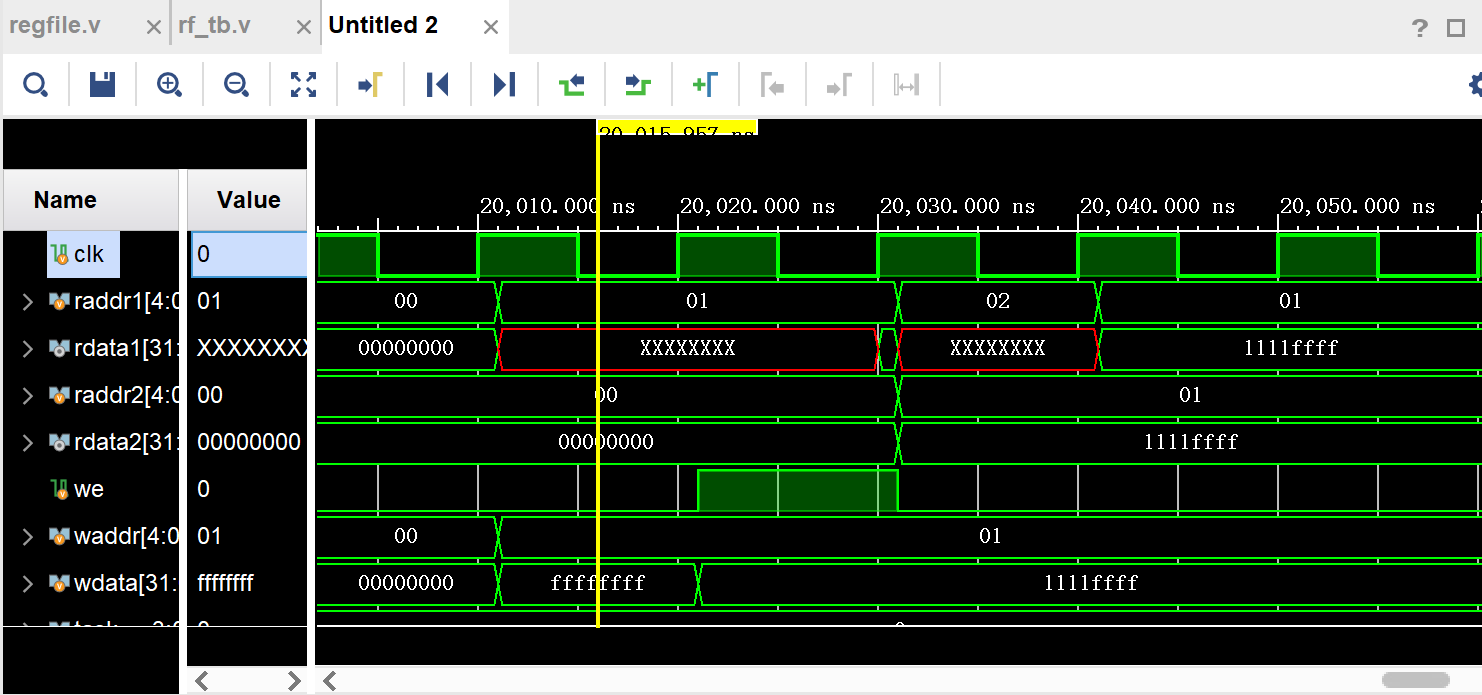
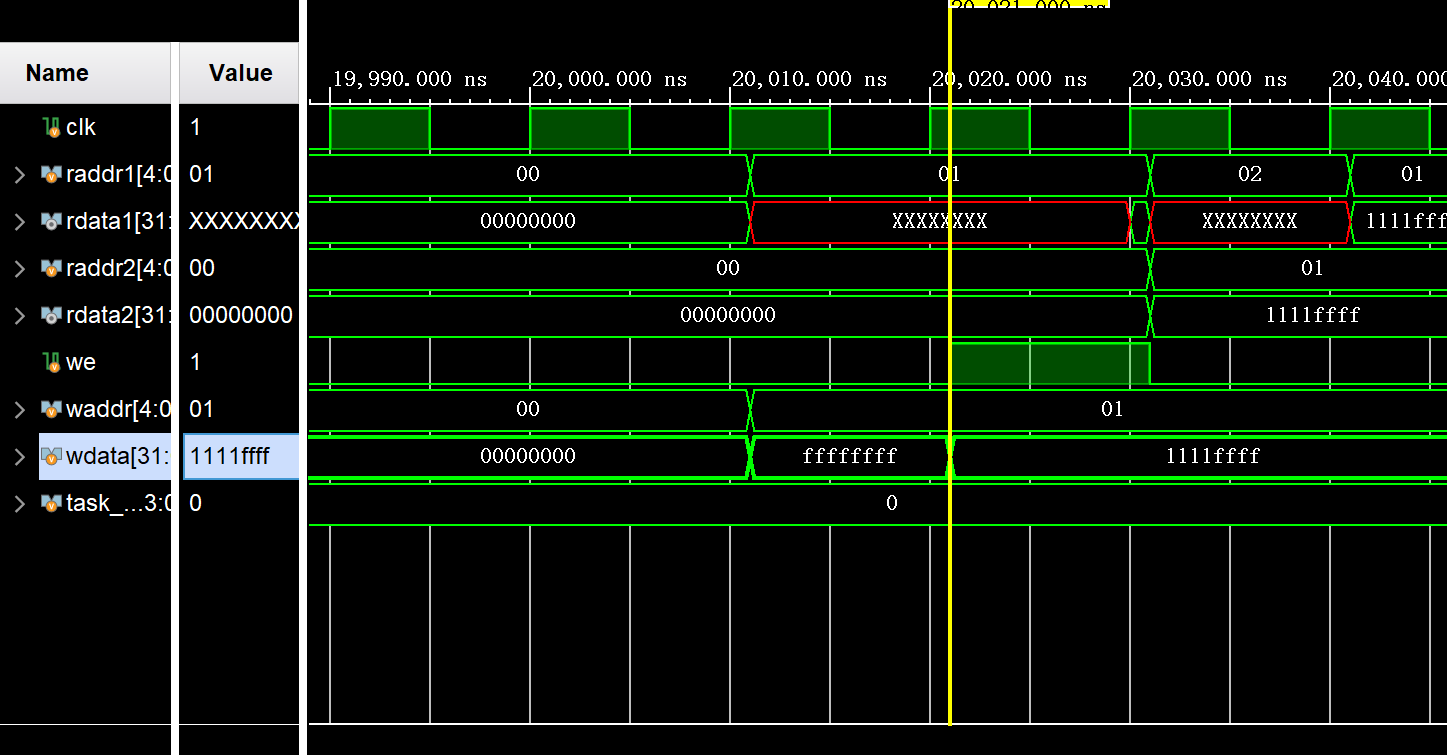
初始时，读地址、写地址、读数据、写数据、写有效等信号量均为0。****

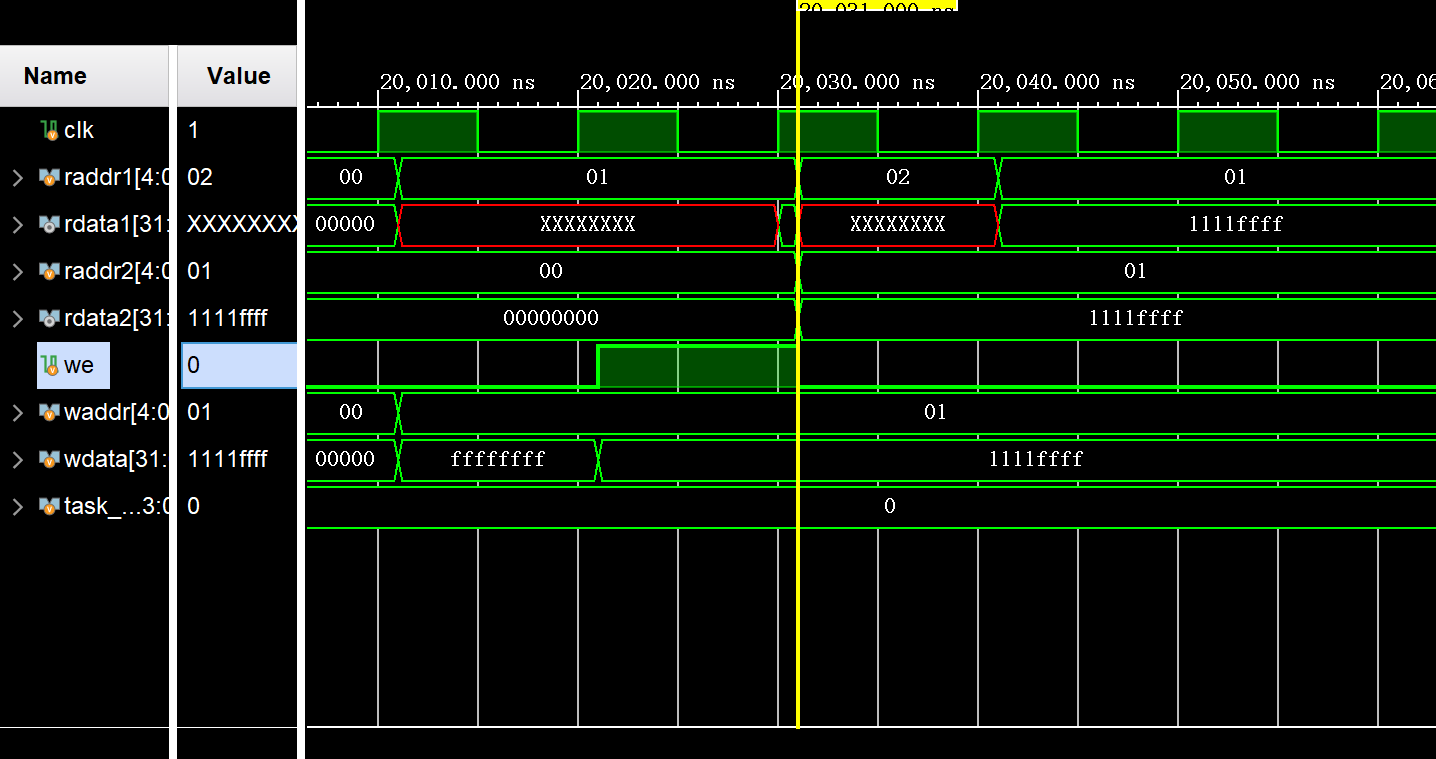
图2.1.1 初始波形图

1. **按仿真时间顺序，将仿真波形变化的关键时刻截图，观察记录各信号量的变化（时钟、读地址、读数据、写地址、写数据、写有效等），对读写行为做注释，填入实验数据表2.1。**

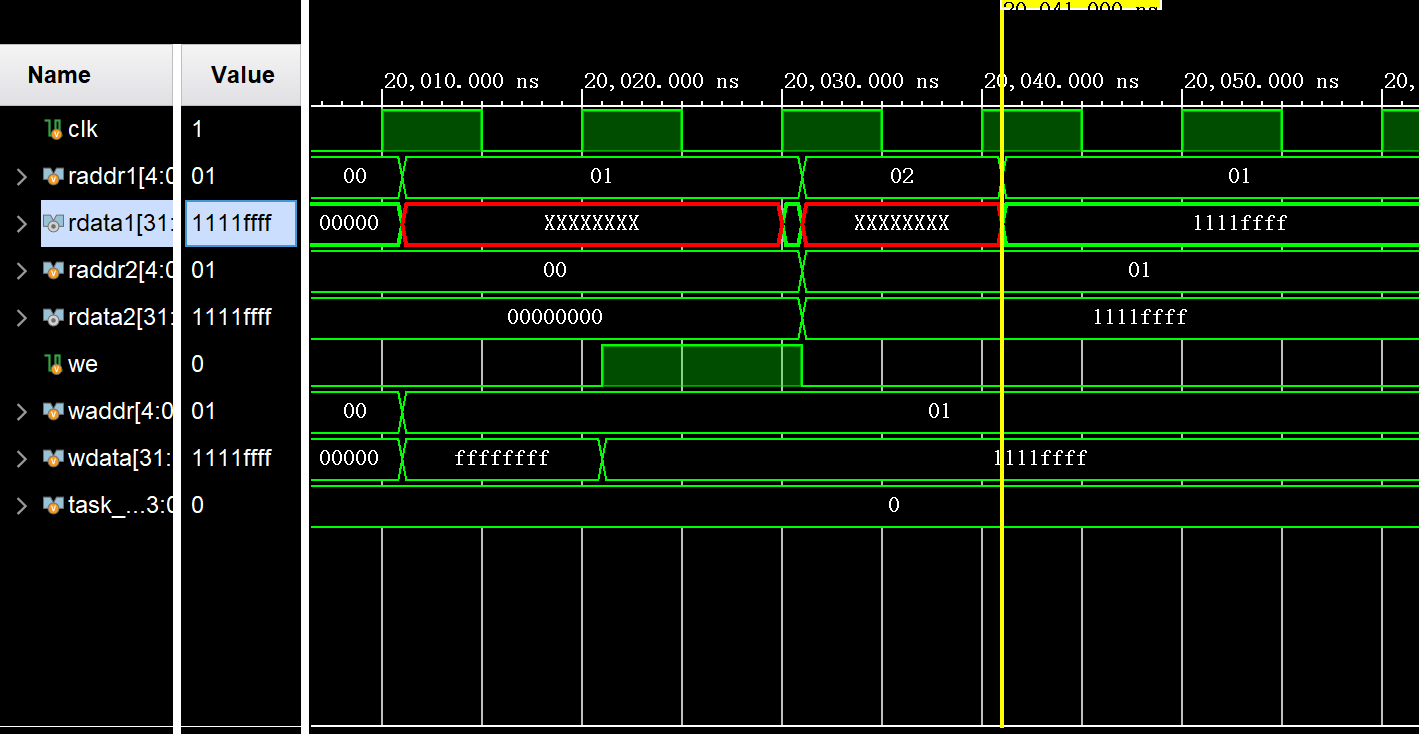
①20021ns时波形变化如图



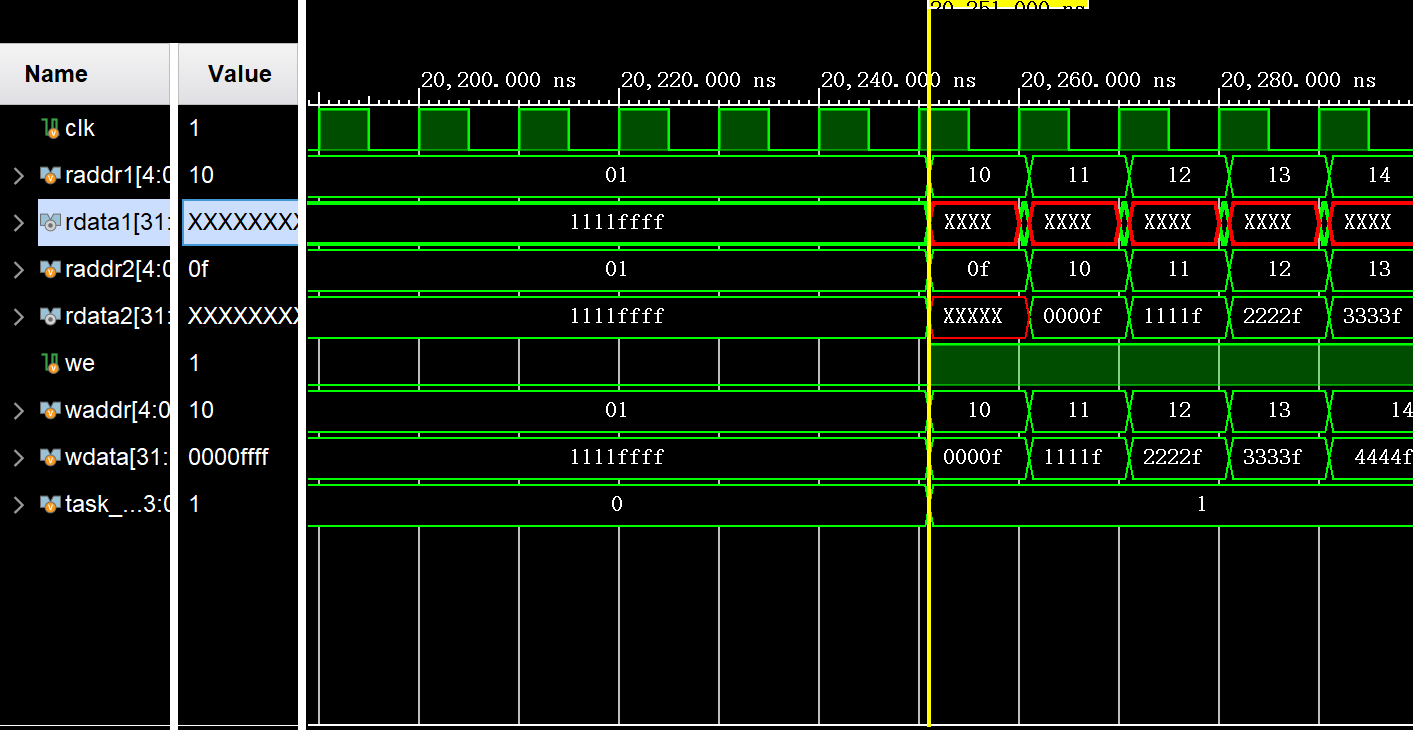
②20031ns时波形变化如图



③20041ns时波形变化如图

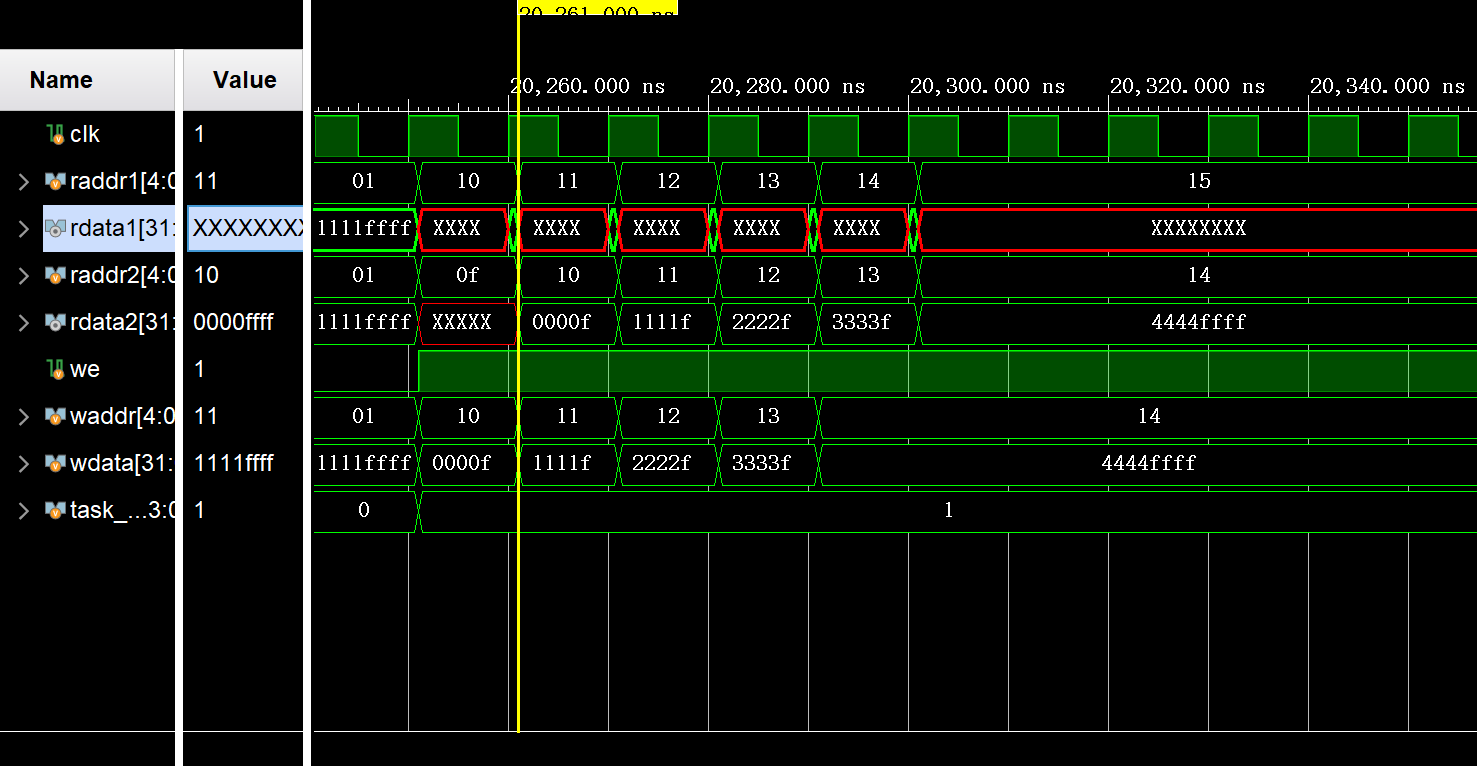


④20251ns时波形变化如图

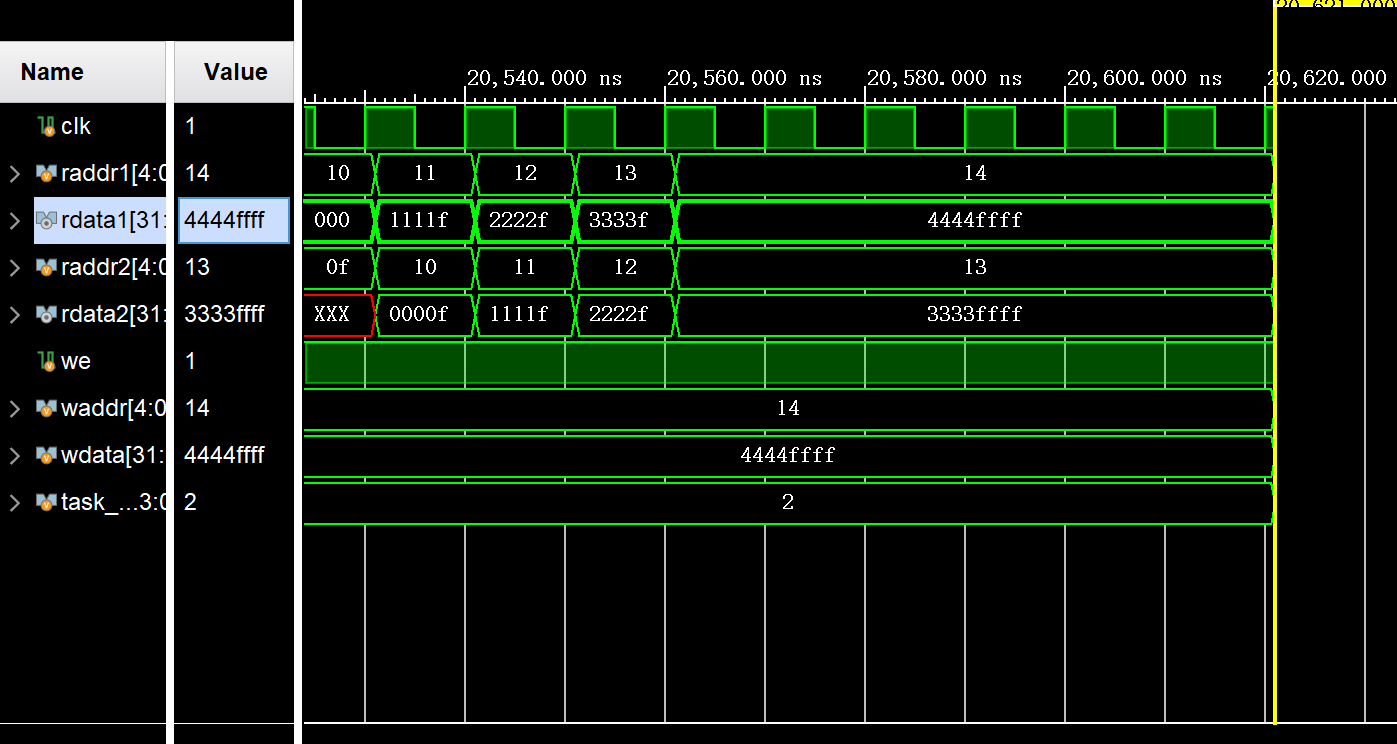


可以看到寄存器在20041ns至20251ns时，rdata1和rdata2的值都为1111ffff,说明此时写有效。

⑤20251ns至20301ns时波形变化如图



⑥20301ns至20621ns时波形变化如图



### 4 实验数据

表2.1 寄存器堆读写行为表

|  |  |  |
| --- | --- | --- |
| 时间顺序 | 信号变化 | 注释说明 |
| 20011ns | raddr从00变为01，rdata1从00000000变为XXXXXXXX，waddr1从00变为01，wdata从00000000变为fffff | clk高电平的时候，发生变化的一共有4个变量 |
| 20021ns | we从0变为1，wdata从fffff变为1111ffff | Clk高电平时，发生变化的一共有2个变量 |
| 20030ns | Clk从低电平变为高电平，rdata1从XXXXXXXX变为1111ffff | Clk从低电平变为高电平的时候，发生变化的一共有1个变量 |
| 20031ns | raddr1从01变为02，rdata1从1111ffff变为XXXXXXXX，we从1变为0，raddr2从00变为01，rdata2从00000000变为1111ffff | Clk高电平时，发生变化的一共有5个变量 |
| 20041ns | raddr1从02变为01，rdata1从XXXXXXXX变为1111ffff | Clk高电平时，发生变化的一共有2个变量 |
| 20251ns | raddr1从01变为10，rdata1从1111ffff变为XXXXXXXX，raddr2从01变为0f，rdata2从1111ffff变为000000，we从0变为1，waddr1从01变为10，wdata从1111ffff变为0000ffff，task\_phase从0变为1 | Clk高电平时，发生变化的一共有8个变量 |
| 20260ns | Clk从低电平变为高电平，rdata1从XXXXXXXX变为0000ffff | Clk从低电平变为高电平的时候，发生变化的一共有1个变量 |
| 20261ns | raddr1从10变为11，rdata1从0000ffff变为XXXXXXXX，raddr2从0f变为01，rdata2从XXXXXXXX变为0000ffff，waddr从10变为11，wdata从0000ffff变为1111ffff | Clk高电平时，发生变化的一共有6个变量 |
| 20270ns | Clk从低电平变为高电平，rdata从XXXXXXXX变为1111ffff | Clk从低电平变为高电平的时候，发生变化的一共有1个变量 |
| 20271ns | raddr1从11变为12，rdata1从1111ffff变为XXXXXXXX，raddr2从10变为11，rdata2从0000ffff变为1111ffff，waddr从11变为12，wdata从1111ffff变为2222ffff | Clk高电平时，发生变化的一共有6个变量 |
| 20280ns | Clk从低电平变为高电平，rdata1从XXXXXXXX变为2222ffff | Clk从低电平变为高电平的时候，发生变化的一共有1个变量 |
| 20281ns | raddr1从12变为13，rdata1从2222ffff变为XXXXXXXX，raddr2从11变为12，rdata2从1111ffff变为2222ffff，waddr从12变为13，wdata从2222ffff变为3333ffff | Clk高电平时，发生变化的一共有6个变量 |
| 20290ns | Clk从低电平变为高电平，rdata1从XXXXXXXX变为3333ffff | Clk从低电平变为高电平的时候，发生变化的一共有1个变量 |
| 20291ns | raddr1从13变为14，rdata1从3333ffff变为XXXXXXXX，raddr2从12变为13，rdata2从2222ffff变为3333ffff，waddr从13变为14，wdata从3333ffff变为4444ffff | Clk高电平时，发生变化的一共有6个变量 |
| 20300ns | Clk从低电平变为高电平，rdata1从XXXXXXXX变为4444ffff | Clk从低电平变为高电平的时候，发生变化的一共有1个变量 |
| 20301ns | raddr1从14变为15，rdata1从4444ffff变为XXXXXXXX，raddr2从13变为14，rdata2从3333ffff变为4444ffff | Clk高电平时，发生变化的一共有4个变量 |
| 20521ns | raddr1从15变为10，rdata1从XXXXXXXX变为0000ffff，raddr2从14变为0f，rdata2从4444ffff变为XXXXXXXX，task\_pharse从1变为2 | Clk高电平时，发生变化的一共有5个变量 |
| 20531ns | raddr1从10变为11，rdata1从0000ffff变为1111ffff，raddr2从0f变为10，rdata2从XXXXXXXX变为0000ffff | Clk高电平时，发生变化的一共有4个变量 |
| 20541ns | raddr1从11变为12，rdata1从1111ffff变为2222ffff，raddr2从10变为11，rdata2从0000ffff变为1111ffff | Clk高电平时，发生变化的一共有4个变量 |
| 20551ns | raddr1从12变为13，rdata1从2222ffff变为3333ffff，raddr2从11变为12，rdata2从1111ffff变为2222ffff | Clk高电平时，发生变化的一共有4个变量 |
| 20561ns | raddr1从13变为14，rdata1从2222ffff变为3333ffff，raddr2从12变为13，rdata2从2222ffff变为3333ffff | Clk高电平时，发生变化的一共有4个变量 |

### 5 实验小结（包括遇到的问题及解决办法）

问题：刚开始波形图全为0，不明白后续的波形为什么没有展现出来。



解决方法：询问别的同学后，明白了波形图未展现的原因。随后点击图中圈出的按钮，然后点击蓝色暂停键，即可展示需要的波形图。

## 2.2 实验三 同步RAM和异步RAM仿真、综合与实现

### 1 实验目的

1. **掌握调用 Xilinx库IP实例化同步RAM和异步RAM的方法。**
2. **理解同步RAM和异步 RAM读写行为的异同。**
3. **掌握查看时序结果和资源利用率的方法。**

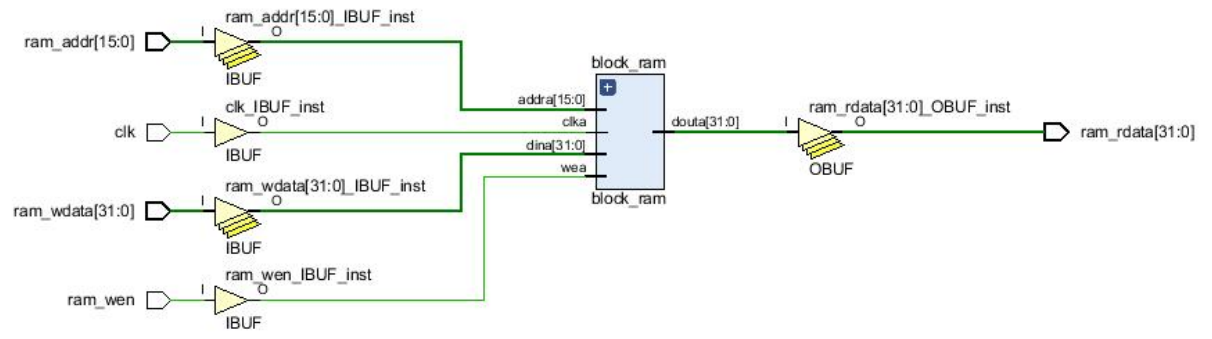
### 2 实验设备

安装了 Vivado 的计算机环境，龙芯实验箱

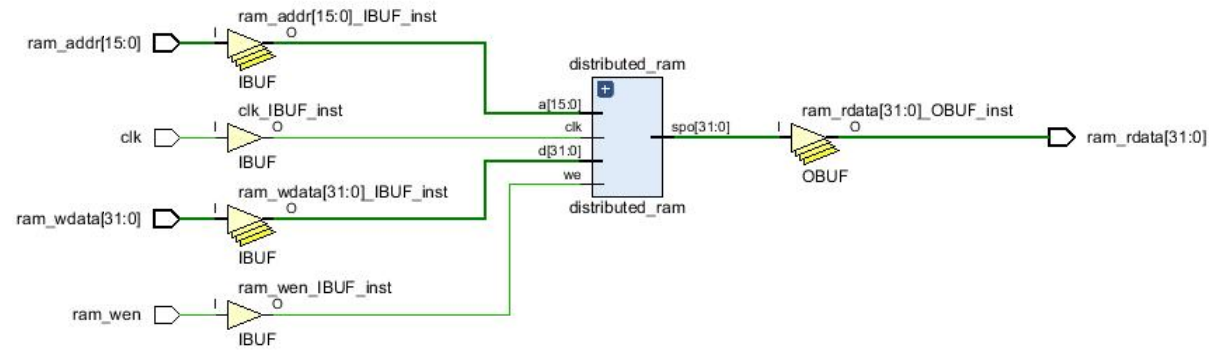
### 3 实验内容与主要步骤

1. **分别画出同步RAM和异步RAM结构框图，标出输入输出端口。**

* 同步RAM

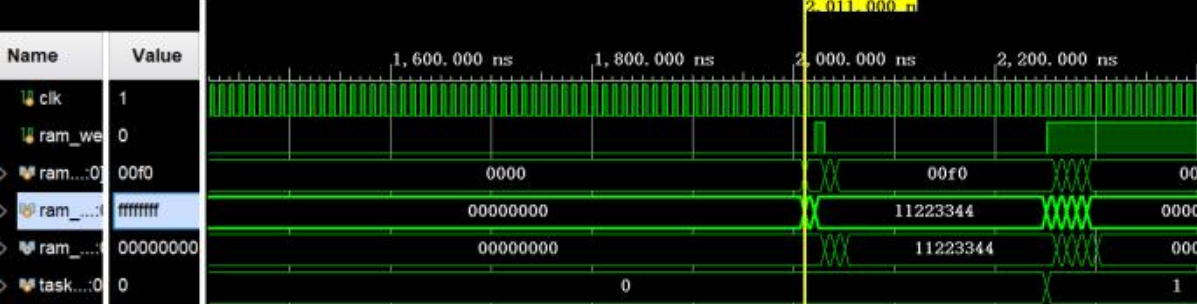
****

* 异步RAM

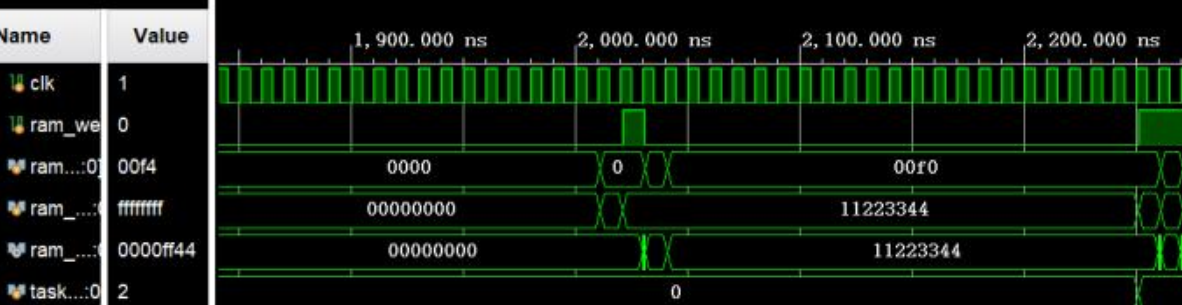
****

1. **对同步RAM和异步RAM分别创建工程，进行仿真，并将仿真波形截图，对比两者读写行为的异同,列入实验数据表。**

* 同步RAM

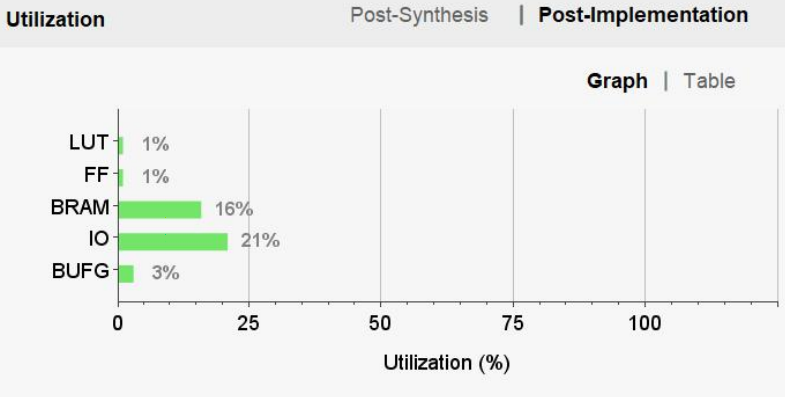
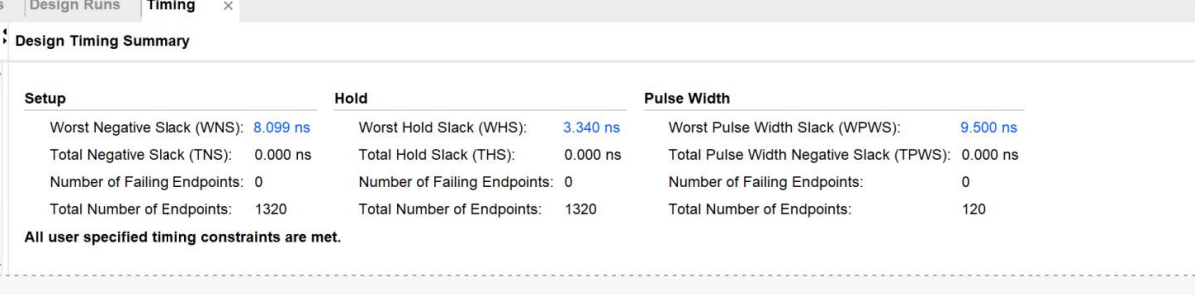
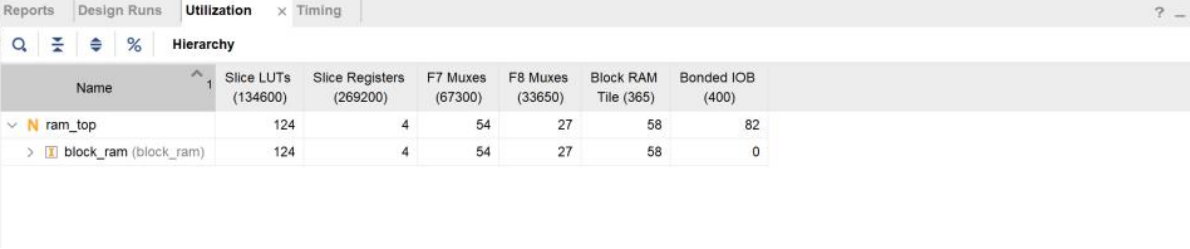
****

* 异步RAM

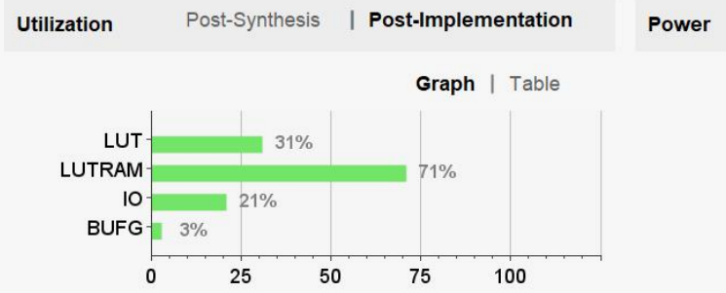
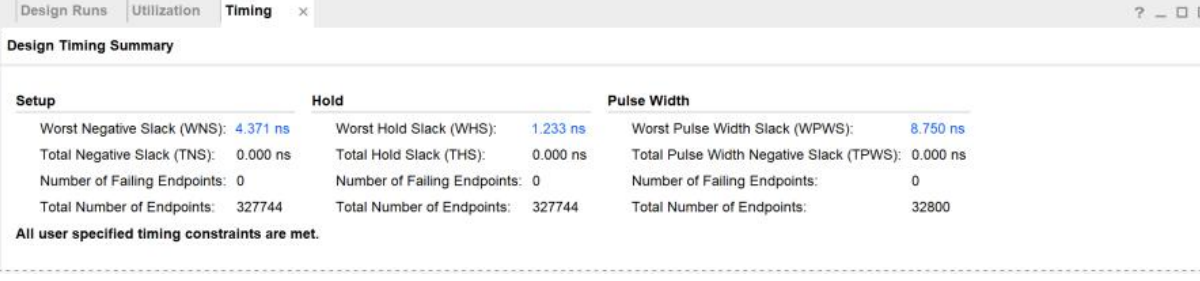


1. **对同步RAM和异步RAM分别进行综合和实现，查看分析时序结果和资源利用率（截图）。**

* 同步RAM

****

* 异步RAM



1. **自行调研、参考同步/异步 RAM 定制的资料，并根据仿真波形对比参数的作用。（选做）**

### 4 实验数据

表2.2 同步RAM和异步RAM对比分析表

|  |  |  |
| --- | --- | --- |
| 同步RAM | 异步RAM | 对比分析 |
| 读数据时，同步 RAM 的数据读取依赖于时钟的上升沿。 | 读数据时，异步RAM的数据读取并不依赖于时钟的上升沿。 | 读数据时，同步和异步的区别主要在于异步RAM可以与时 钟信号步调不一致，而同步RAM需要与时钟信号步调一 致。 |
| 写数据时，同步 RAM 的数据读取依赖于时钟的上升沿。 | 写数据时，异步RAM的数据读取依赖于时钟的上升沿。 | 写数据时，同步和异步RAM的写入都需要与时钟信号步调一致。 |
| 同步 RAM 在进行仿 真时的 WNS 违约值 约是异步 RAM 的两倍，而同步 RAM 的 资源占用率相对异步 RAM 较少。 | 异步 RAM 在进行仿真时的WNS违约值约是同步RAM的一半，而异步RAM的资源占用率相对同步RAM 较多。 | 异步RAM在满足时序条件时有着更好的表现，稳定性可能更好，但资源占用更大，可能会使运行性能下降。 |

### 5 实验小结（包括遇到的问题及解决办法）

问题：不理解RAM同步和异步的区别

解决方法：RAM 同步和异步不是读写周期的区别，而应该联系时钟周期信号进行分析。 在本次实验中，我实际掌握了调用 Xilinx 库 IP 实例化同步 RAM 和异步 RAM 的方法，理解了同步RAM和异步 RAM 读写行为的异同。

具体区别：

一、对于同步RAM：

时钟驱动：SRAM的读写操作与时钟信号同步，通常每个时钟周期能完成一个数据传输。

存储单元：使用多个晶体管（通常是六个）构成一个存储单元，具有较快的存取速度，但占用空间较大。

速度：SRAM速度较快，适合高速缓存和需要快速响应的应用，数据传输速率可达几百MHz。

二、异步RAM

无时钟依赖：ARAM的读写操作不依赖时钟，信号由控制电路直接管理，因此在某些情况下会导致速度较慢。

存储单元：一般使用较少的晶体管（通常是四个），结构简单，适合实现低成本的存储解决方案。

速度：相对较慢，通常用于低速应用，数据传输速率通常在几十MHz到几百MHz之间。

三、总结

SRAM适合高性能应用，具有更快的响应和更复杂的控制逻辑。

ARAM更简单且成本低，适合对速度要求不高的场合。

## 2.3 实验四 数字逻辑电路的设计与测试

### 1 实验目的

1. **了解数字逻辑电路功能仿真的常见错误。**
2. **掌握功能仿真的波形分析方法。**
3. **掌握波形异常类错误的常用调试方法。**

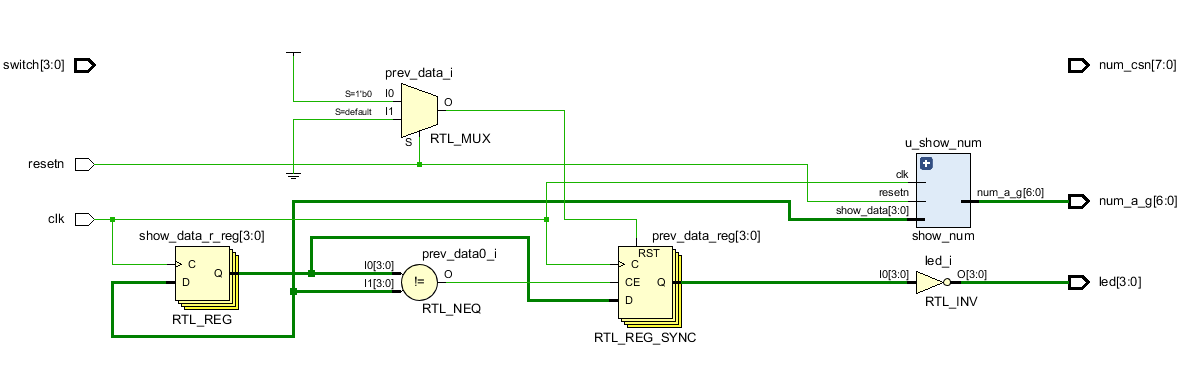
### 2 实验设备

A.Vivado软件

B.FPGA实验板

### 3 实验内容与主要步骤

1. **画出结构框图，标出输入输出端口。**



1. **为项目创建工程，进行仿真，调试bug。（将调试改正的bug列表说明）**

提供的设计源码中包含 5 个 bug，其中 4 个是第3.2.2节中提到的波形异常的前 4 种情况：波形为 “Z”、波形为 “X”、波形停止和越沿采样，另外的 1 个 bug 是功能 bug。

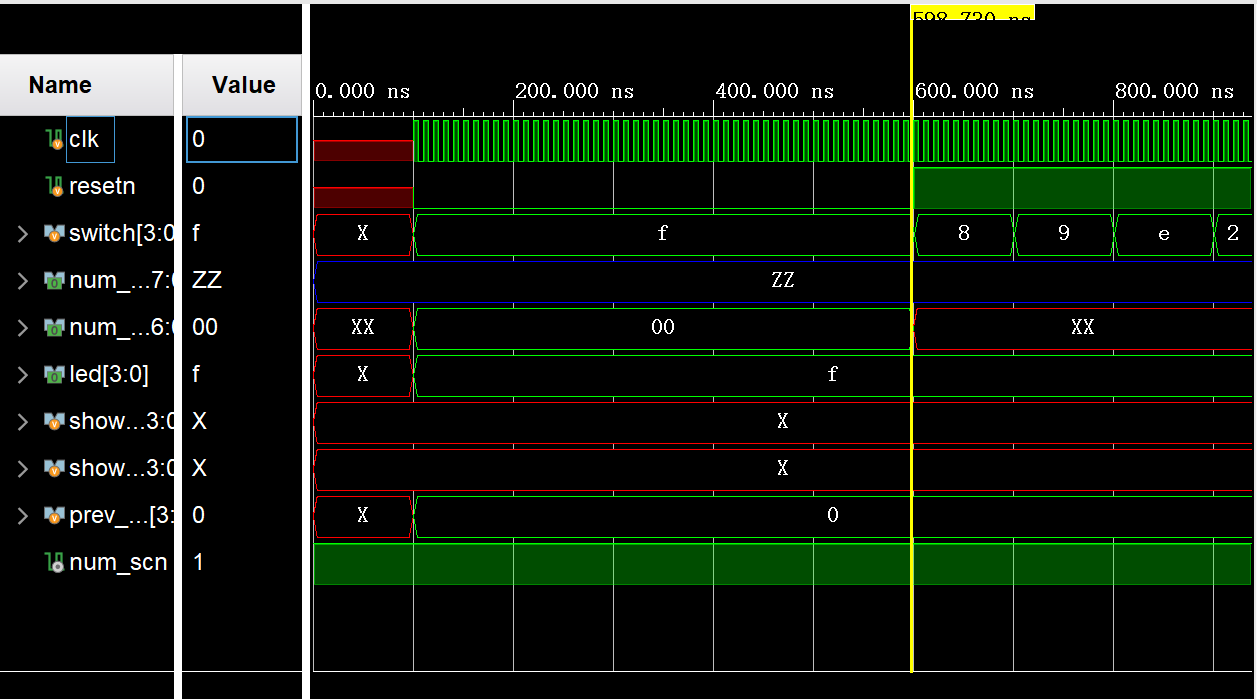
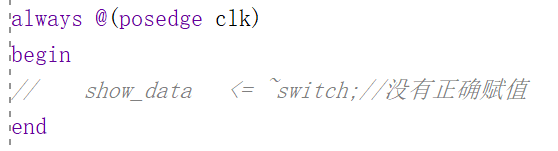


图2.3.1 修改前的截图

①我们可以发现，仿真状态下show\_data[3:0]信号量报X波形异常。查看代码可以看到show\_data被注释掉了，因此没有正确赋值。



②修改完第一处错误后，仿真加载不出来，有波形停止错误。通过断点调试，发现nxt\_a\_g和keep\_a\_g存在循环使用。

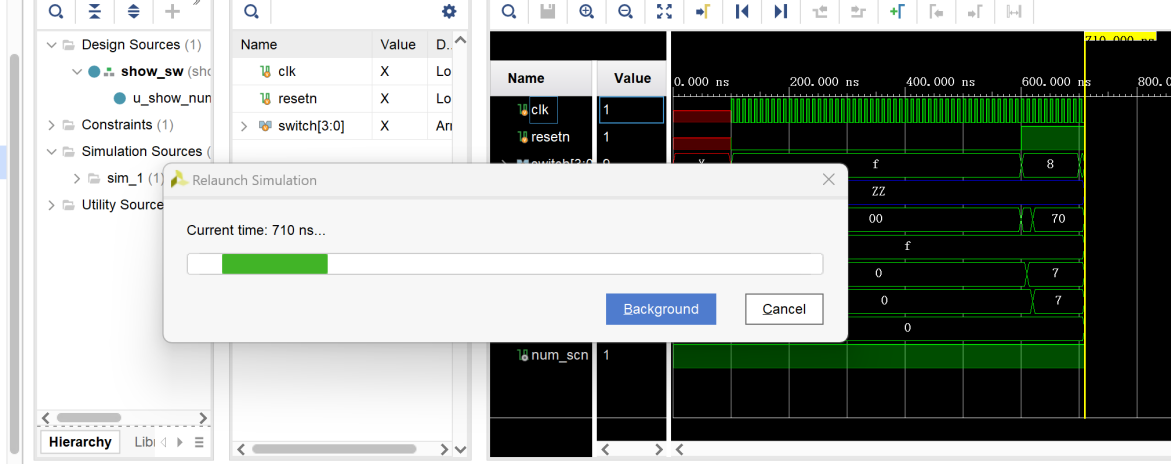


图2.3.2 第二处错误现象

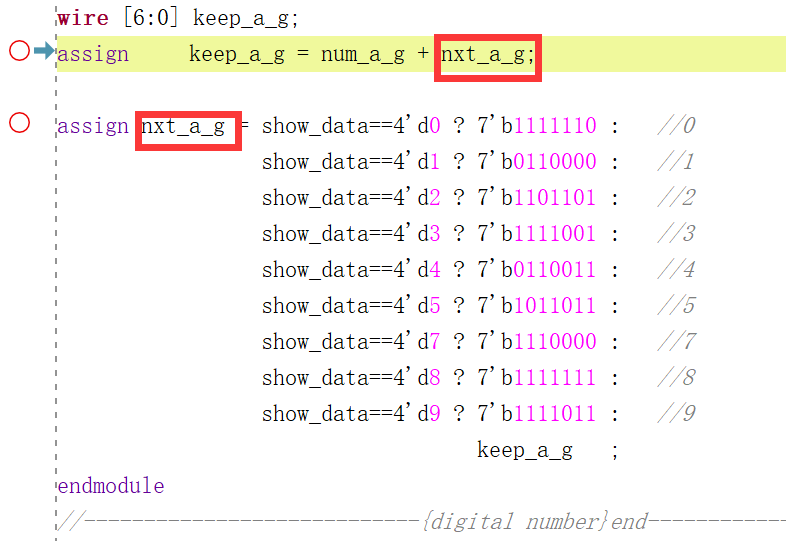


图2.3.3 第二处错误原因

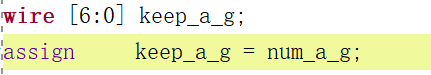


图2.3.4 第二处错误修改

③修改完前面的两处错误后，num\_csn[7:0]还存在Z形错误。查看代码发现，代码中的变量名有错误。

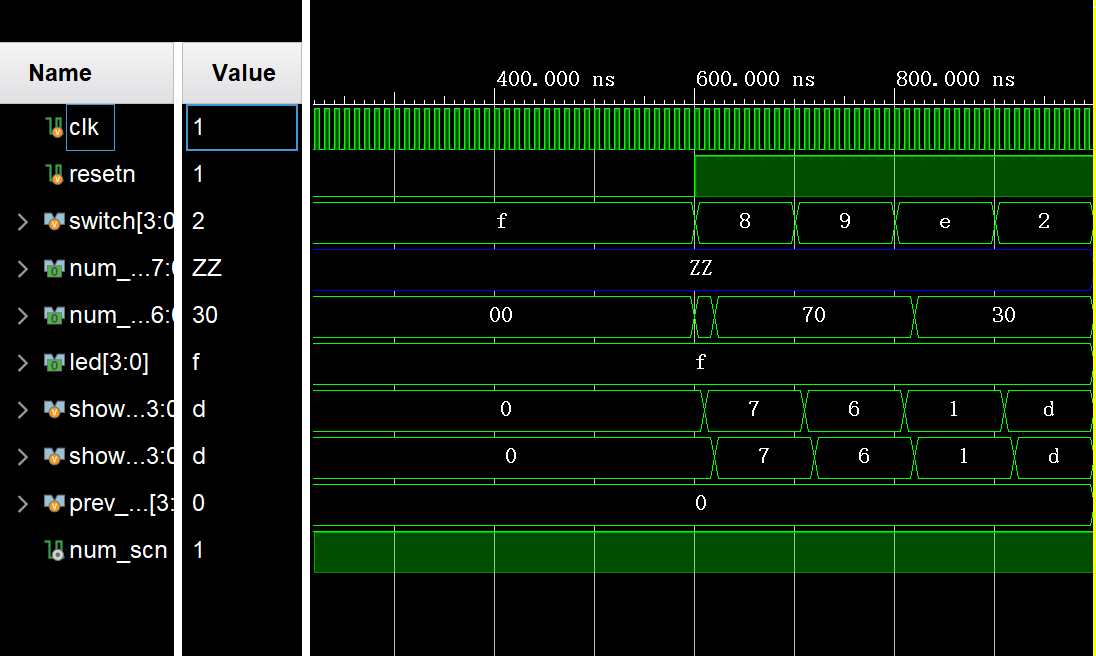


图2.3.5 Z形错误

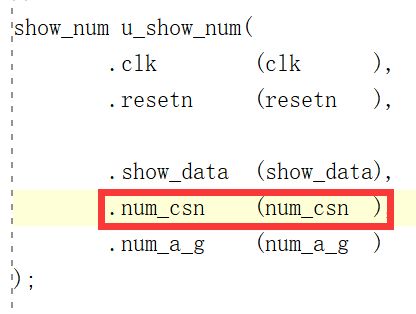
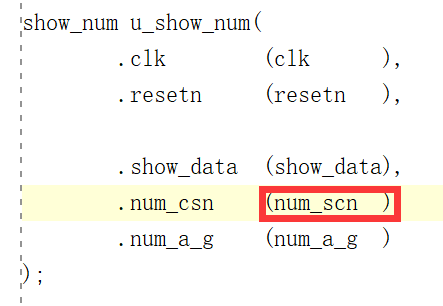


图2.3.6 第三处代码错误及修改

④修改完第三个错误后，发现有越沿采样的问题。查看代码发现，是因为赋值时将“<=”写成了“=”。

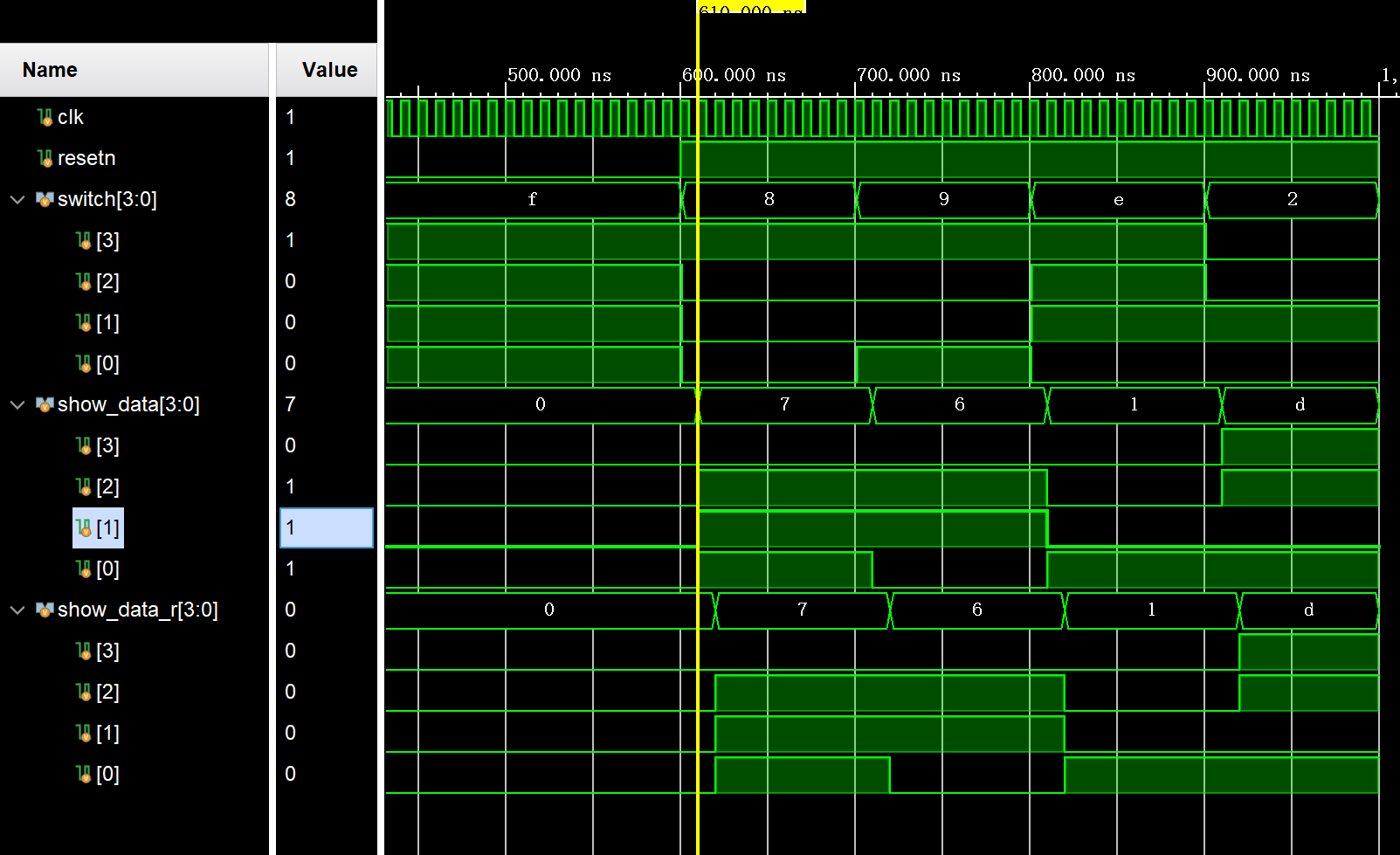


图2.3.7 越沿采样问题

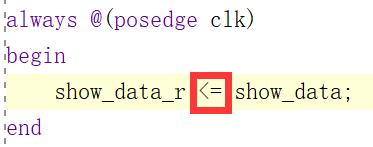
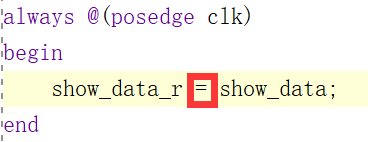


图2.3.8 第四处代码错误及修改

⑤最后一处是功能形错误，代码中少了一个编号为6的情形。

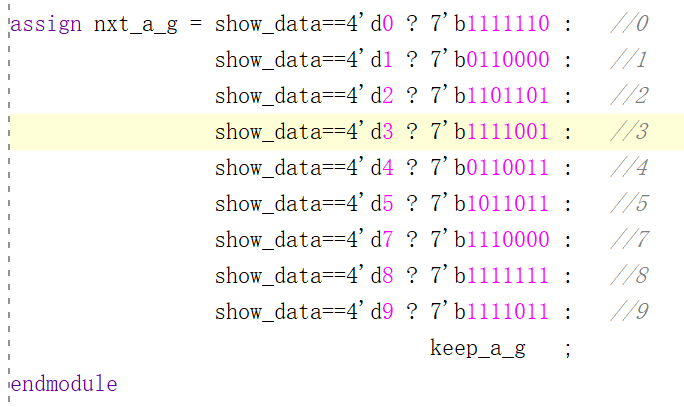


图2.3.9 第五处代码错误

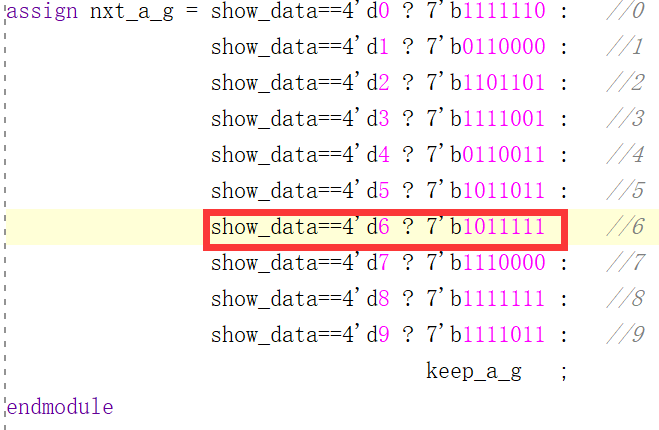


图2.3.10 第五处代码修改

⑥综上，将调试的代码错误列表如下。

表2.3.1 bug列表总览

|  |  |  |
| --- | --- | --- |
| 修改前 | 修改后 | 说明 |
| //show\_data <= ~switch; | show\_data <= ~switch; | 将//注释删除，  使其正确赋值 |
| .num\_csn (num\_scn ), | .num\_csn (num\_csn ), | 变量名写错，波形出现Z形错误 |
| assign keep\_a\_g = num\_a\_g + nxt\_a\_g; | assign keep\_a\_g = num\_a\_g; | 逻辑问题，导致循环 |
| num\_a\_g = nxt\_a\_g; | num\_a\_g <= nxt\_a\_g; | 赋值错误，出现越沿采样问题 |
| 无 | show\_data==4'd6 ? 7'b1011111 : //6 | 遗漏编码为6的情况 |

1. **改正后的功能仿真测试时序图截图。**

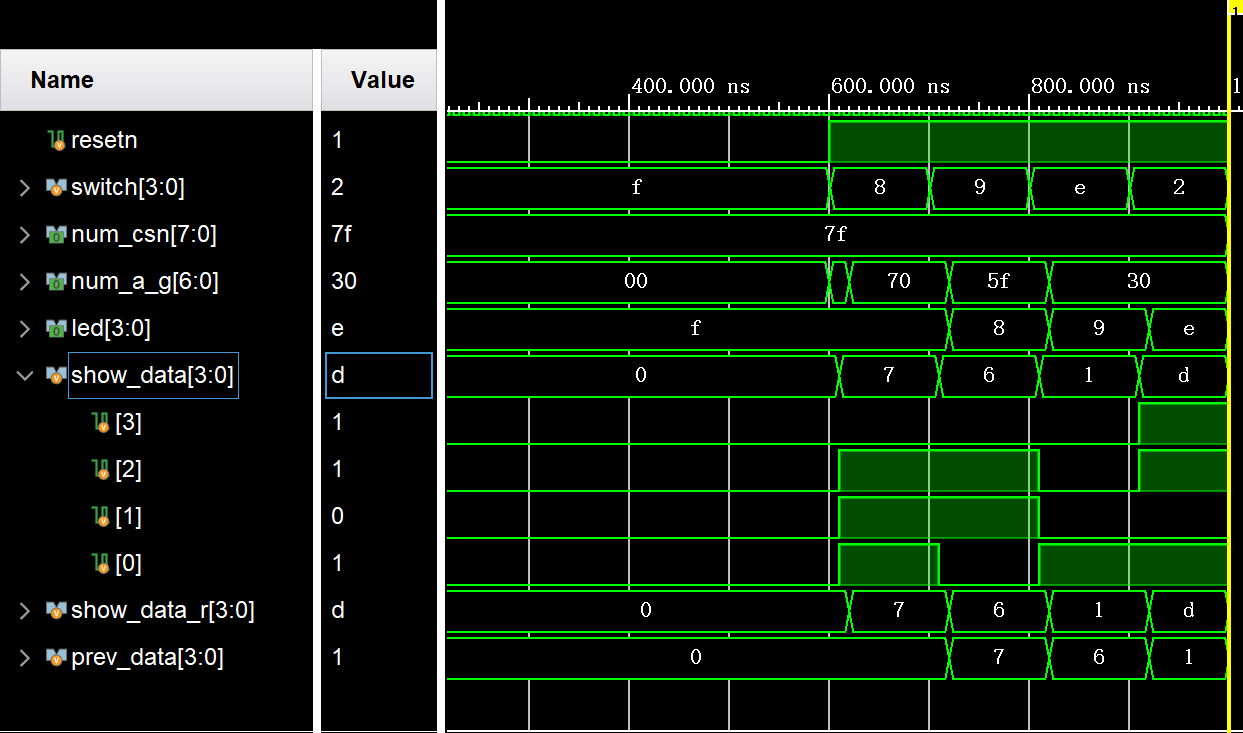
****

图2.3.11 修改正确的仿真测试时序图

1. **综合实现，生成比特流文件，连接开发板，进行上板验证。**
2. **将板上的运行结果拍照。**

**①**按reset复位键，并将所有开关拨下，所有LED灯都为关闭，数码管显示为0。

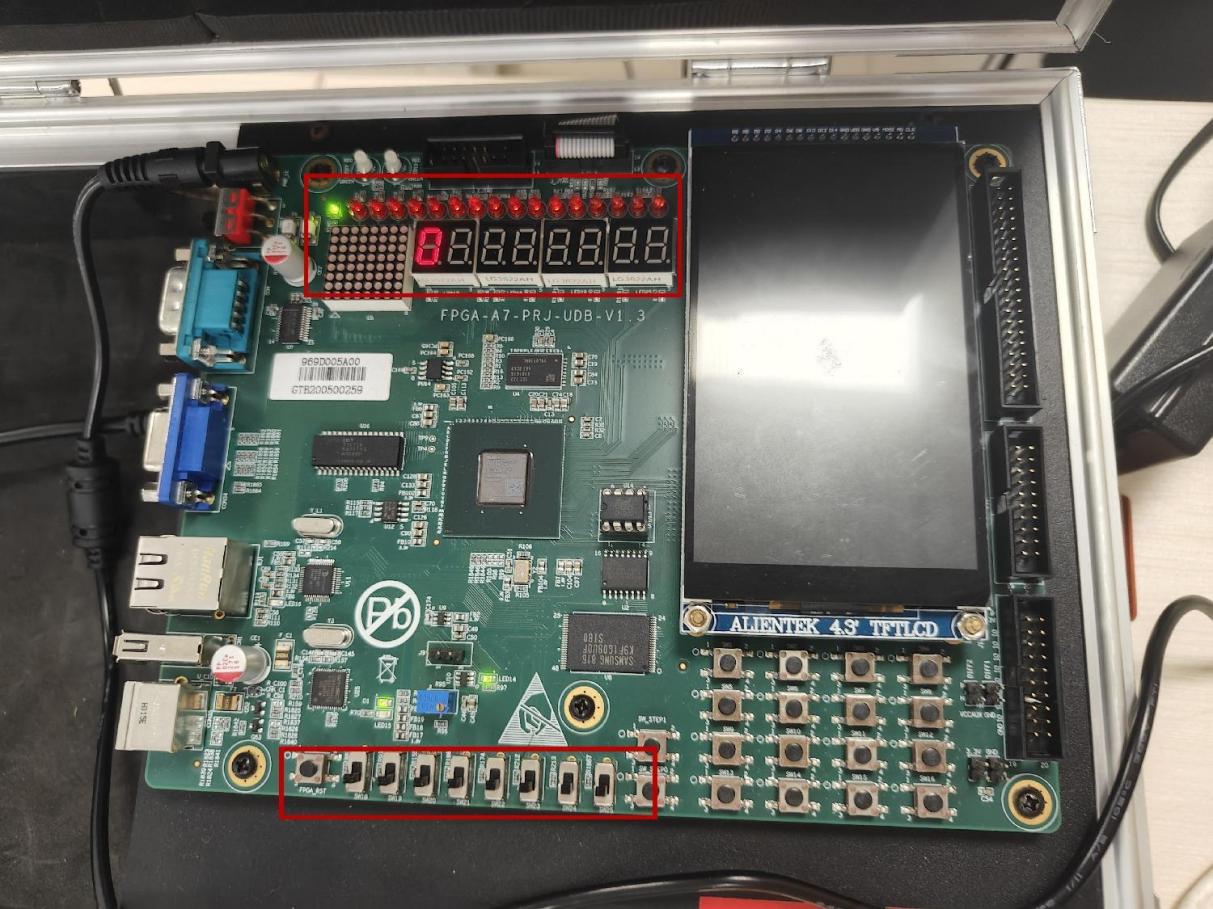


图2.3.12 结果1

②将最右边的1个开关拨上，所有LED灯都为关闭，数码管显示为1。

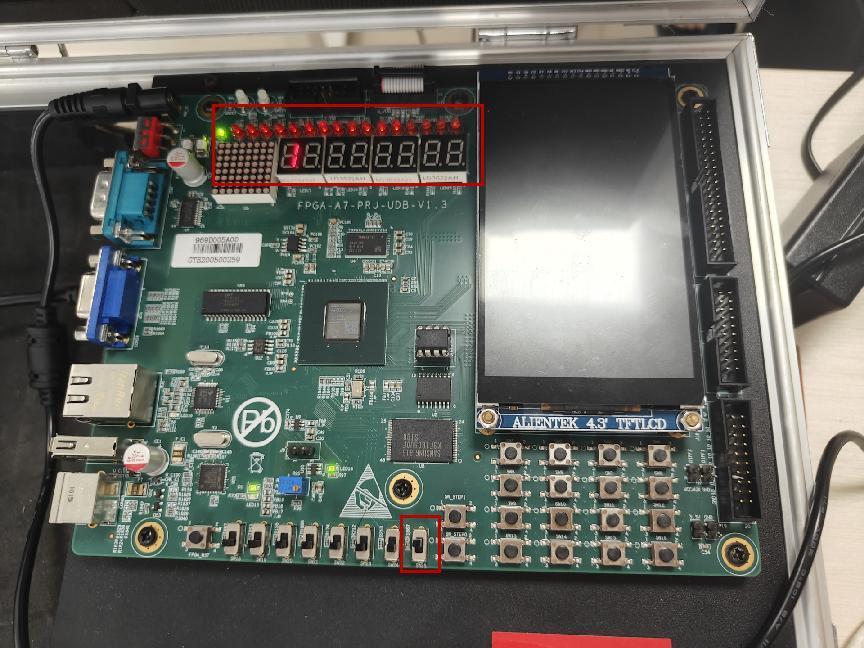


图2.3.13 结果2

③将最右边的2个开关拨上，LED灯最右边1个灯亮，数码管显示为3。

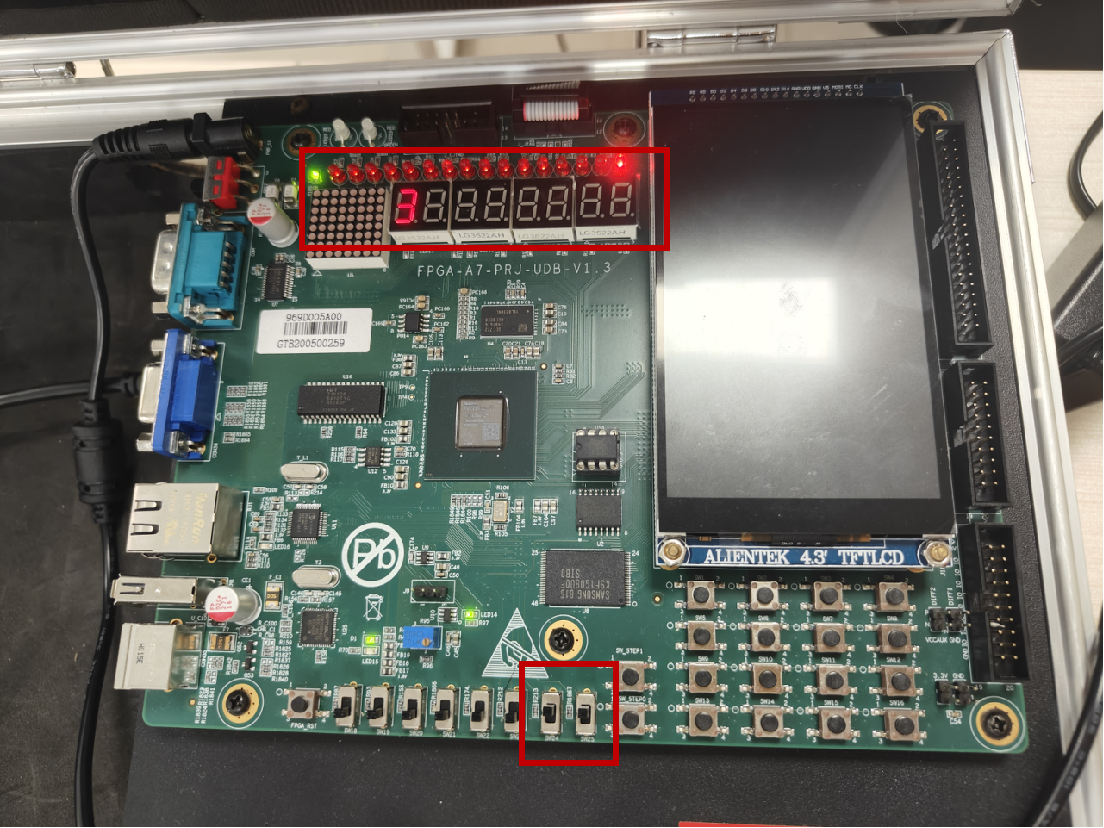


图2.3.14 结果3

④将最右边的3个开关拨上，LED灯最右边2个灯亮，数码管显示为7。

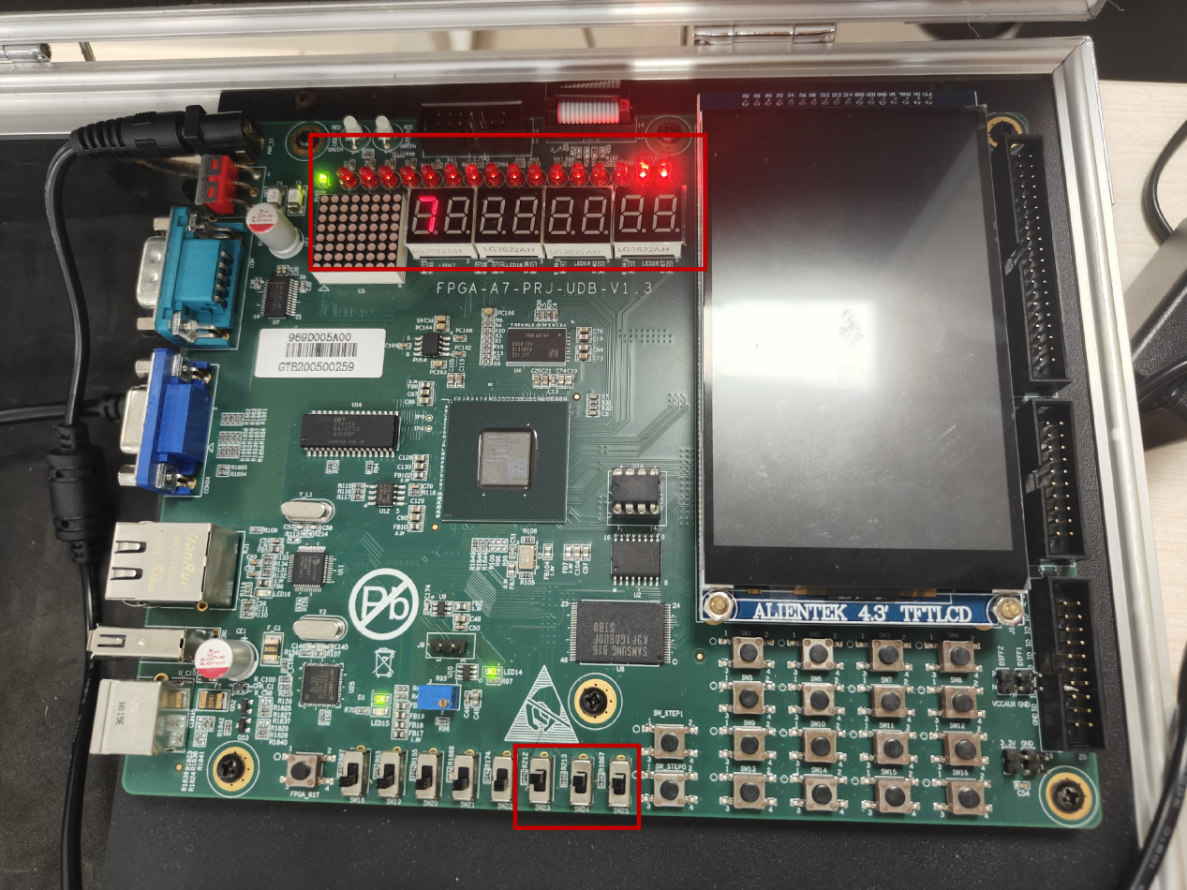
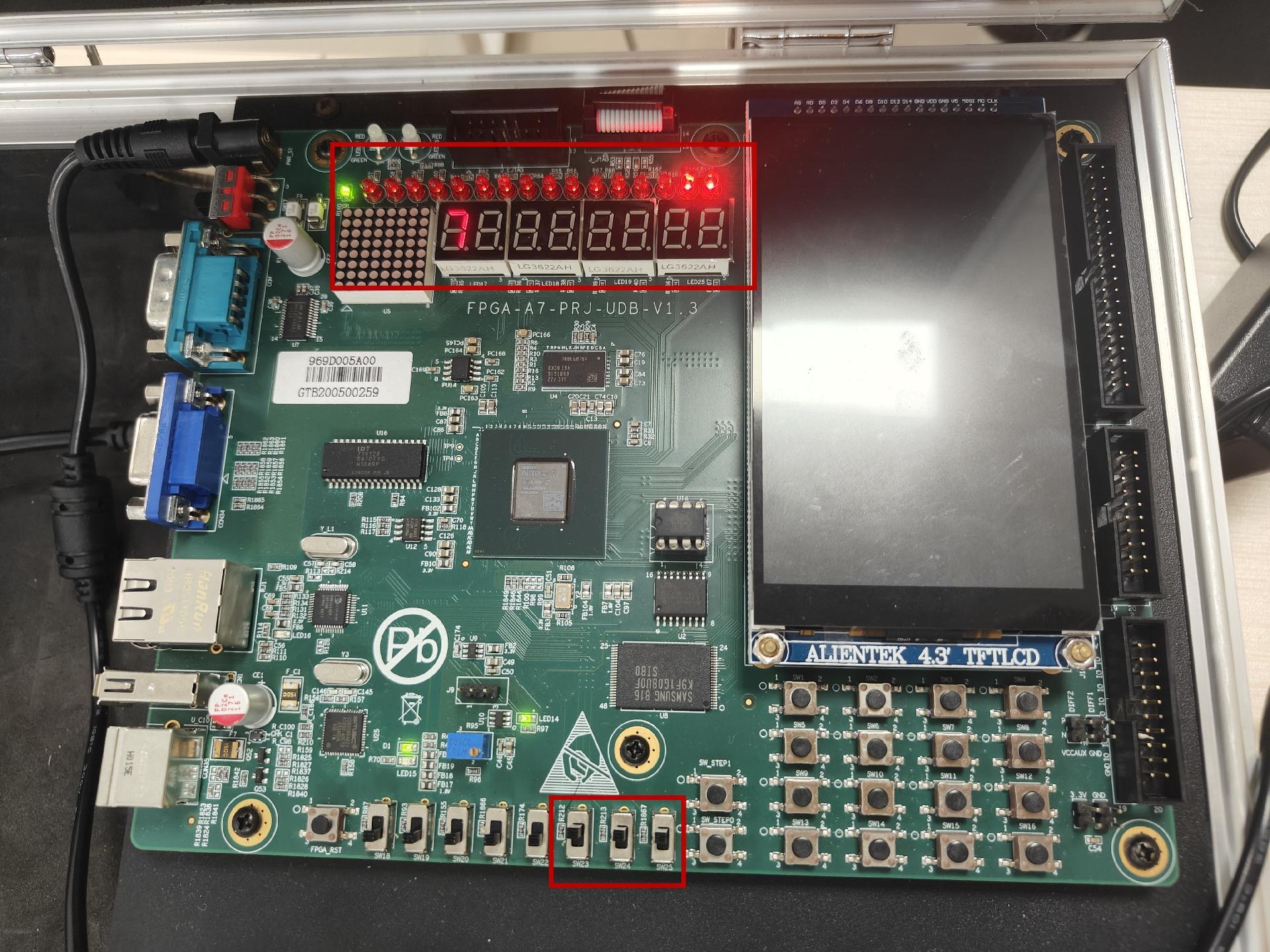


图2.3.15 结果4

⑤将右边第一个和第三个开关拨上，LED灯最右边3个灯亮，数码管显示为5。



图2.3.16 结果5

⑥将右边第三个开关拨上，LED灯右边第1个和第3个灯亮，数码管显示为4。

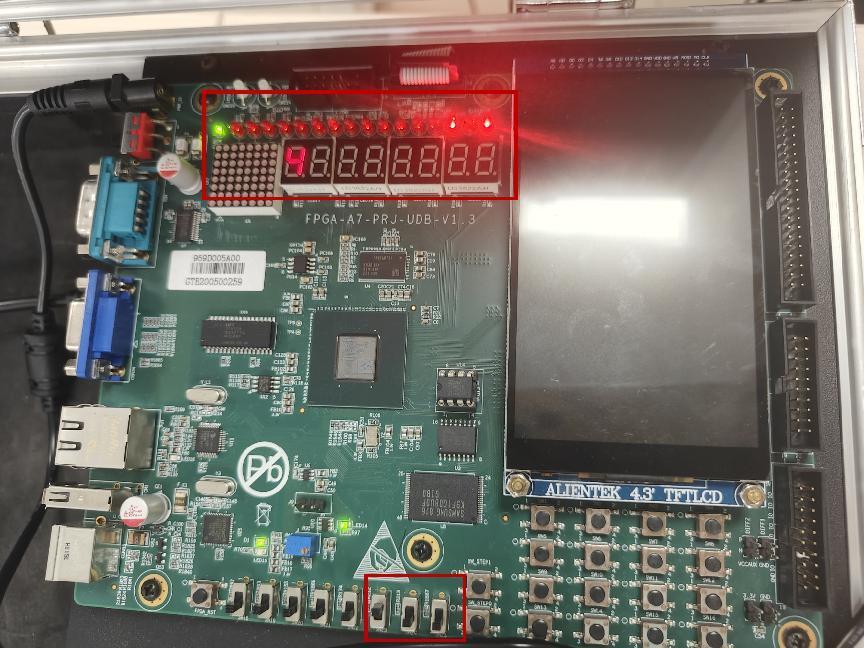


图2.3.17 结果6

⑦将开关全部拨下，LED灯右边第3个灯亮，数码管显示为0。

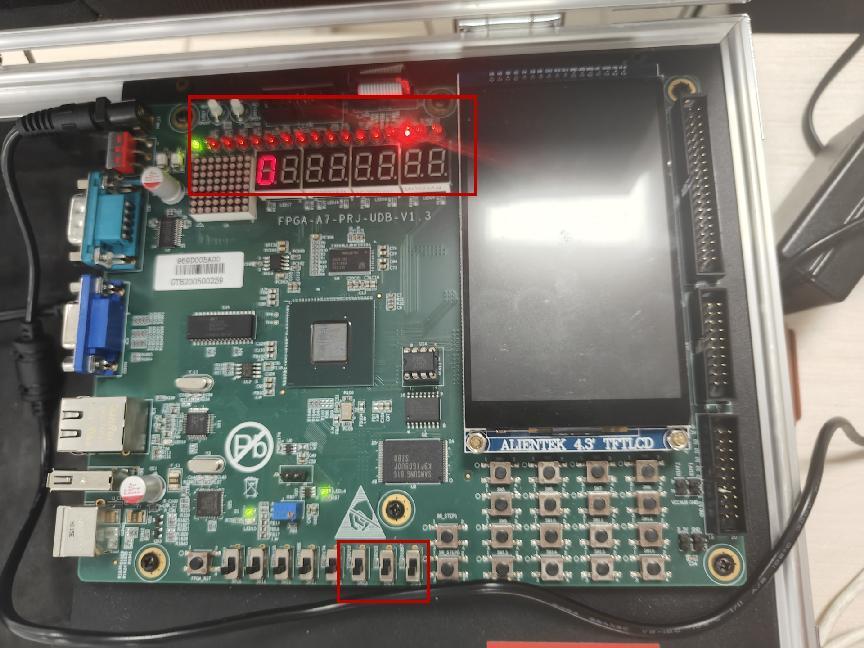


图2.3.18 结果7

### 4 实验数据

表2.3.2 拨码显示实验现象表

|  |  |  |
| --- | --- | --- |
| 实验操作 | 实验现象 | 分析 |
| 按reset复位键，并将所有开关拨下 | 所有LED灯都为关闭，数码管显示为0 | 开关全部拨下为1，LED灯高电平不亮 |
| 最右边的1个开关拨上 | 所有LED灯都为关闭，数码管显示为1 | 最右侧1个开关拨上状态为1，LED灯显示上一次拨码开关的状态为0，LED灯不亮 |
| 最右边的2个开关拨上 | LED灯最右边1个灯亮，数码管显示为3 | 最右侧2个开关拨上状态为3，LED灯显示上一次拨码开关的状态为1 |
| 最右边的3个开关拨上 | LED灯最右边2个灯亮，数码管显示为7 | 最右侧3个开关拨上状态为7，LED灯显示上一次拨码开关的状态为3 |
| 右边第一个和第三个开关拨上 | LED灯最右边3个灯亮，数码管显示为5 | 右边第一个和第三个开关拨上状态为5，LED灯显示上一次拨码开关的状态为7 |
| 右边第三个开关拨上 | LED灯右边第1个和第3个灯亮，数码管显示为4 | 右边第三个开关拨上状态为4，LED灯显示上一次拨码开关的状态为5 |
| 将开关全部拨下 | LED灯右边第3个灯亮，数码管显示为0 | 全部拨下状态为0，LED灯显示上一次拨码开关的状态为4 |

### 5 实验小结（包括遇到的问题及解决办法）

问题：对越沿采样的概念不理解

解决方法：

查找实验指导书，从中我明白了越沿采样的波形是什么样的，也了解了阻塞赋值和非阻塞赋值的区别。

以下是指导书内容：

越沿采样是指一个被采样的信号在上升沿采样到了其在上升沿后的值，一般情况下认为这是一个错误，是 RTL 里阻塞赋值 “=” 和非阻塞赋值 “<=” 使用不当导致的。

越沿采样是一个隐藏较深的错误，往往可能和逻辑错误混在一起。初看起来，其波形是很正常的，而且在发生越沿采样后，要再执行很长时间才会出错。因此，大家可以先按照逻辑错误进行调试，如果发现数据采样有异常，就需要甄别是否出现了越沿采样的错误。



如图所示，在 105ns 时刻，clk 上升沿到来，a\_r 和 a\_r\_r 同时变为 1（也就是 a 的值）。a\_r 在 105ns 时刻前是 0，在 105ns 时刻后是 1。从源码来看，a\_r\_r 是在上升沿采样 a\_r 的值，结果在 105ns 时刻采样到 a\_r 为 1 的值，也就是采样到 a\_r 在同一上升沿后的值。这就属于越沿采样。

造成这一现象更深层的原因是 Verilog 里阻塞赋值 “=” 和非阻塞赋值 “<=” 混用。在该图的源码中，a\_r 采用阻塞赋值，而 a\_r\_r 采用非阻塞赋值。每一次赋值分为两步：第一步是计算等式右侧的表达式；第二步是赋值给左侧的信号。这两步简记为计算和赋值。在一个上升沿到来时，所有由上升沿驱动的信号按以下顺序进行处理：

1. 先处理阻塞赋值，即完成计算和赋值，同一信号完成计算后立刻完成赋值。同一 always 块里的阻塞赋值从上到下按顺序串行执行，不同 always 块里的阻塞赋值根据所用工具实现确定顺序的串行执行，一一完成计算和赋值。

2. 进行非阻塞赋值的计算。对于所有非阻塞赋值，其等式右侧的值都同时计算好。

3. 上升沿结束时，所有非阻塞赋值同时完成最终的赋值动作。

从以上描述可以看到，非阻塞赋值是在上升沿的最后一个时间步里完成处理的，晚于阻塞赋值的处理。所以在图3.15的示例中，a\_r\_r 的赋值晚于 a\_r 的赋值，造成了越沿采样的情况。除非特意设计，一般认为越沿采样是一个设计错误。针对越沿采样错误，我们有以下几点建议：

• 编写 RTL 时注意代码规范，所有 always 写的时序逻辑只允许采用非阻塞赋值。

• 一旦发现越沿采样的情况，追溯被采样信号，直到追溯到某一个阻塞赋值的信号，随后进行修正。