**实验2报告**

2017K8009908018

蔡润泽

实验箱：45

一、实验任务（10%）

本次实验主要是通过让我们利用仿真和上板，熟悉硬件设计与调试的流程，并了解Reg\_File, 同步RAM、异步RAM的特性，其中：

实验一：利用给出的寄存器堆相关代码，通过仿真调试，得出寄存器堆的特性。

实验二：利用同步RAM、异步RAM的相关代码，通过仿真、综合，查看两者的波形和资源利用率，学习同步和异步RAM的读写行为和异体同。

实验三：对给出的代码通过仿真和上板进行调试修改，找出存在的5处bug。

二、实验过程（90%）

（一）实验流水账

9月6日，19：00-20：00，实验一代码仿真，添加测试信号和marker，并保存相关文件

9月7日，10：00-11：30，实验二、实验三代码仿真、综合，添加测试信号和marker，并保存相关文件

9月7日，20：00-22：00，实验三代码调试、仿真、综合、上板，开始撰写实验报告

9月8日， 9：30-11：30，继续完成实验报告

（二）子任务一

图片包含 电视, 监视器, 屏幕, 墙壁

描述已自动生成

图1.1 寄存器堆写使能为1，时钟上升沿时的寄存器堆波形图

从此图中可以看出，第一根蓝线对应的时刻是we写使能信号变为1的时候，但此时clk未处于上升沿，因此，此时waddr对应的1号寄存器的值依然没有被wdata的值：1111ffff赋值；而图中黄线对应的时刻，即当we信号为1，即写使能时，且时钟为上升沿时，wdata的值1111ffff存入waddr指向的1号寄存器中。这体现了寄存器堆同步写的特点。

图中第二根蓝线是两个raddr改变的时刻，此时的rdata1和rdata2的值也立马改变，分别变为1号寄存器的值和2号寄存器的值。体现了寄存器堆异步读的特点，即不受时钟上升沿控制。

（三）子任务二

1. 仿真行为对比分析

图片包含 户外

描述已自动生成

图2.1同步RAM仿真波形图一

图片包含 屏幕截图

描述已自动生成

图2.2异步RAM仿真波形图一

1. 两种RAM相同点

从图2.1和图2.2中可以看出，第一根蓝线对应的时刻是ram\_wen写使能信号变为1的时候，但此时clk未处于上升沿，因此，此时wdata的值还未发生改变；而图中第二根黄线对应的时刻，即当wen信号为1，即写使能时，且时钟为上升沿时，wdata的值11223344存入ram\_addr指向的内存地址中，这从黄线时指向同一地址的ram\_rdata读数也变成11223344里体现出来。这体现了两个RAM均为同步输入的特点。

图片包含 电视, 室内, 监视器, 墙壁

描述已自动生成

图2.3同步RAM仿真波形图二

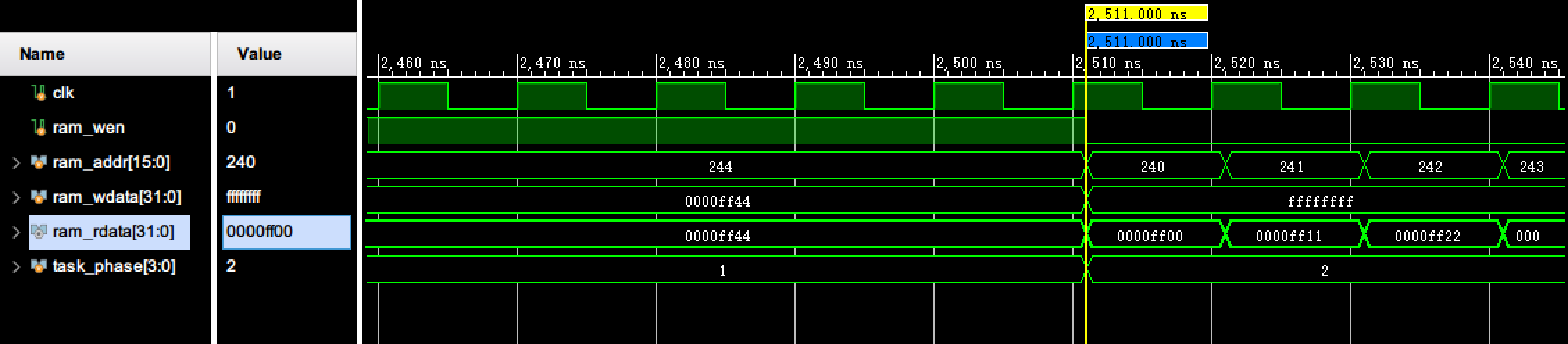


图2.4异步RAM仿真波形图二

1. 两种RAM不同点

从图2.3和图2.4中可以看出，在内存输出地址在240位置的值时，同步RAM是在时钟的上升沿才进行输出；而异步RAM是在读地址信号(ram\_rdata)发生改变时就异步输出了。这体现出同步RAM同步输出，异步RAM异步输出的特点。

1. 与寄存器堆进行比较

将两种RAM的读写行为与寄存器堆进行比较可以发现，两种RAM的写行为都与寄存器堆相同；而对于读行为，只有异步RAM与寄存器堆相同。

1. 时序、资源占用对比分析

（1）时序对比分析

图片包含 屏幕截图

描述已自动生成

图2.5同步RAM Design Runs

图片包含 屏幕截图

描述已自动生成

图2.6异步RAM Design Runs

通过图2.5和图2.6可以看出，同步RAM的最差负时序裕量 (WNS)为正0.56，而异步RAM的最差负时序裕量 (WNS)为负6.556；同步RAM的总负时序裕量 (TNS)为0，而异步RAM的总负时序裕量 (TNS)为负7449.666。对于时序裕量，应尽量做到TNS=THS。

而同步RAM对于LUT的占用率和FF的占用率极低；异步RAM对于LUT和LUTRAM的占用率比较高，特别是对于LUTRAM的占用率达到了71%。异步RAM的功耗也相对较高，是同步RAM的2.5倍

（2）资源占用对比分析

图片包含 屏幕截图

描述已自动生成

图2.7同步RAM资源利用率

图片包含 屏幕截图

描述已自动生成

图2.8异步RAM资源利用率

通过图2.7和图2.8可以看出，两种RAM的I/O和BUFG占用率相当。

而同步RAM对于LUT的占用率和FF的占用率极低；异步RAM对于LUT和LUTRAM的占用率比较高，特别是对于LUTRAM的占用率达到了71%。异步RAM的功耗也相对较高，是同步RAM的2.5倍。

1. 总结

通过上述对比分析，在生成大块RAM时，利用同步RAM更优。虽然异步RAM灵活方便，但同步RAM的稳定性更高，资源占用率更低，因此大规模的使用更优。

（三）子任务三

1、错误1：变量命名错误

（1）错误现象

Vivado软件有该变量命名行出现高亮，num\_csn信号为ZZ，如图3.1。

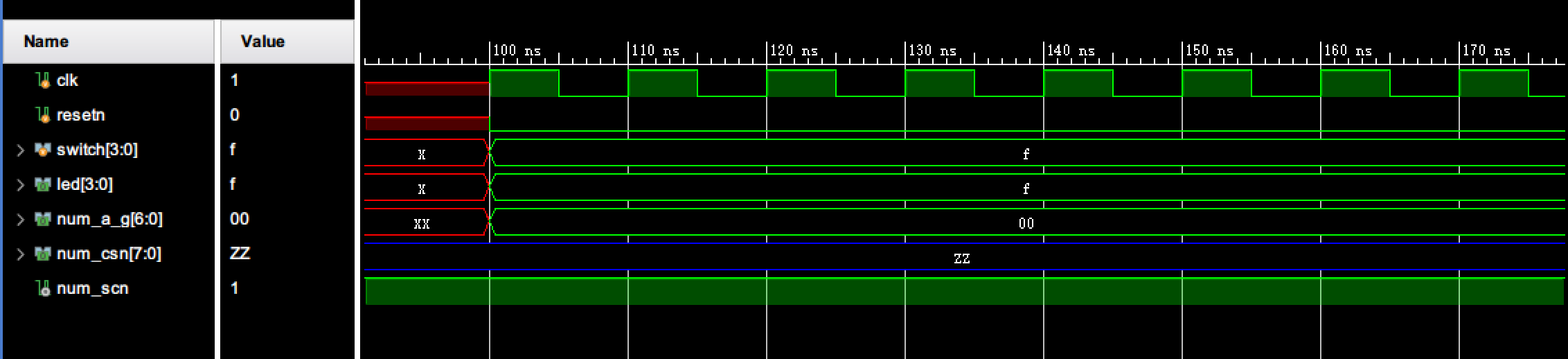


图3.1错误一导致的波形

（2）分析定位过程

Vivado软件变量命名行出现高亮，通过高亮部分定位到错误变量名num\_scn，通过与前面部分代码比对发现，应该修改为“num\_csn”。

（3）错误原因

由于粗心导致的同一变量的变量名前后不一致。

（4）修正效果

修改变量名为“num\_csn”高亮消失，num\_csn波形恢复正常，如图3.2。

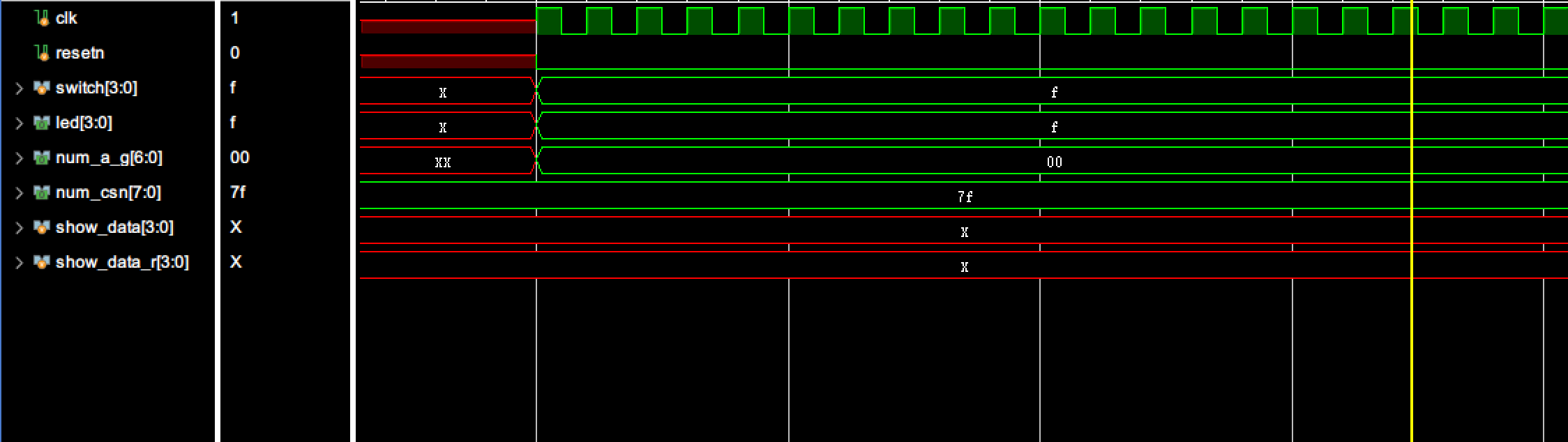


图3.2 错误一修复后的波形

（5）归纳总结（可选）

可以用一些IDE工具，通过变量补全的插件，自动补全变量。写代码时应该要时刻保持细心。

2、错误2：赋值语句被注释

（1）错误现象

波形中的show\_data和show\_data\_r为X。

（2）分析定位过程

通过找show\_data和show\_data\_r的赋值语句，发现“// show\_data <= ~switch;”，即赋值语句被注释掉。

（3）错误原因

由于失误，“//”将show\_data的赋值语句给注释掉，导致未给show\_data赋值。

（4）修正效果

删除//，做到修改完这一步，会发现一个新的问题，如图3.3,因此修正效果需要将新问题解决后才能查看。

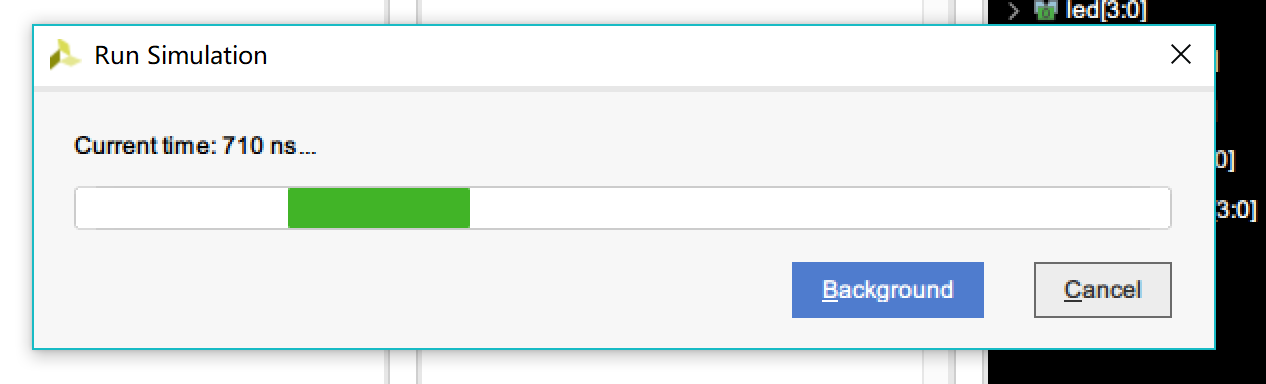


图3.3错误二修复后的仿真时出现新问题

（5）归纳总结（可选）

由于粗心导致的问题，编写代码时还需要更加细心。

3、错误3：由于逻辑环锁住导致仿真不能进行

（1）错误现象

仿真卡在710ns，不能顺利进行，如图3.3。

（2）分析定位过程

Vivado产生高亮，如图3.4。

图片包含 屏幕截图

描述已自动生成

图3.4错误三仿真失败后的代码部分产生高亮

通过分析发现，“assign keep\_a\_g = num\_a\_g + nxt\_a\_g;”和”assign nxt\_a\_g = show\_data==…=keep\_a\_g ;”中nxt\_a\_g参与了keep\_a\_g的赋值，而keep\_a\_g参与了nxt\_a\_g，因此形成了锁环。通过对代码的语义分析，可以将“assign keep\_a\_g = num\_a\_g + nxt\_a\_g；”改为“assign keep\_a\_g = num\_a\_g；”。

（3）错误原因

nxt\_a\_g参与了keep\_a\_g的赋值，而keep\_a\_g参与了nxt\_a\_g，导致形成锁环。

（4）修正效果

如图3.5，将“assign keep\_a\_g = num\_a\_g + nxt\_a\_g；”改为“assign keep\_a\_g = num\_a\_g；”后，仿真正常执行，错误2的修改也得到了验证。

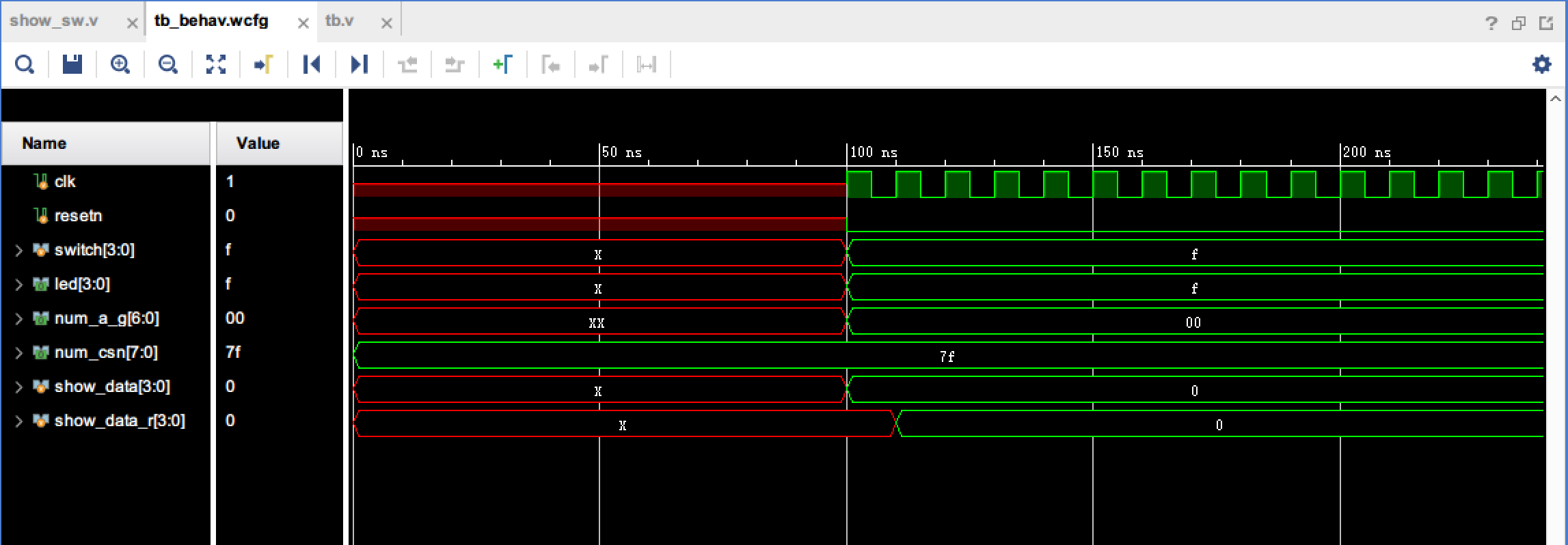


图3.5错误三修改后的波形图

4、错误4： show\_data\_r在always语句中未用非阻塞赋值

（1）错误现象

通过观察代码，发现show\_data\_r的赋值语句在always块中，但采取的是阻塞赋值方式

（2）分析定位过程

通过找show\_data\_r的赋值语句，发现“show\_data\_r = show\_data;”，即show\_data\_r没用使用非阻塞赋值，而在时序逻辑里，需要使用非阻塞复制，因此，我将“show\_data\_r = show\_data;”改为“show\_data\_r <= show\_data;”

（3）错误原因

show\_data\_r的赋值语句在always块中，但采取的是阻塞赋值方式。而在时序逻辑里，需要使用非阻塞复制。

（4）修正效果

将“show\_data\_r = show\_data;”改为“show\_data\_r <= show\_data;”

5、错误5：nxt\_a\_g条件赋值语句中少了一行

（1）错误现象

上板后，如图3.6从左至右拨第2、第3个开关，数字“6”无法显示。

图片包含 电子产品, 电路

描述已自动生成

图3.6错误五无法显示数字“6”的上板图

（2）分析定位过程

通过找nxt\_a\_g的条件赋值语句，发现缺少了show\_data==4'd6？这一个选择条件，因此，代码加上了“show\_data==4'd6？7'b 1011111：”

（3）错误原因

nxt\_a\_g的条件赋值语句，发现缺少了show\_data==4'd6？这一个选择条件，导致板上数字“6”无法显示

（4）修正效果

代码加上了“show\_data==4'd6？7'b 1011111：”后，如图3.7上板数字“6”可以显示。

图片包含 户外, 电子产品, 浅色, 建筑物

描述已自动生成

图3.7错误五修正后显示数字“6”的上板图

四、实验总结（可选）

通过完成这次实验，在硬件部分，我较之前增加了对同步RAM和异步RAM的了解。发现同步、异步RAM只是在读数据时有区别。异步RAM的异步只指读异步。通过两者的对比，也了解到生成大块RAM时，同步RAM的优势。

而子任务三对于测试代码的调试，让我了解到细致在编程时的重要性。做实验之前，本以为这5个bug会藏得很深，但发现这5处bug只是非常常见的错误，甚至我在第一次接触测试代码，在没有经过仿真和上板的测试的情况下，就找出了这5处bug。当然，“当局者迷”，找别人的bug可能相对容易，但自己写代码的时候，可能一不小心也是会犯很多自己并不容易察觉的小错误，所以今后写代码的确需要更加细致。

另外，这次调试虽然没有测试自己编写的代码，但上板调试的过程能看到代码能实现在实物上，还是非常具有体验感。这也让我对今后的实验充满了期待。当然了，还是希望自己能少写一些bug，争取少调几次就能通过。