**实验10报告**

蔡润泽、付轶凡

箱子号：45

一、实验任务（10%）

在 lab10 的实验环境中，完成:

a) 带握手的类 SRAM 接口到 AXI 接口的转换桥 RTL 代码编写。

b) 通过简单的读写测试。  
二、实验设计（40%）

（一）总体设计思路

读、写状态机如下：

**A close up of a logo

Description automatically generated**

图1.1 AXI读、写状态机设计图

如图1.1的状态机示意图，在代码设计中，AXI对于读事务和写事务分别由两个状态机。

1. 读事务

对于读事务，一共存在五个状态，FREE、VALID、READY、OK、END：

FREE:表示读操作请求为空，可以接受读请求。

VALID:表示接收到了读请求，于是从FREE跳转到了VALID状态。在VALID状态下发出arvalid、addr\_ok以及有效的arsize、araddr。

READY:表示接收到了arready，于是从VALID跳转到了READY状态。在READY状态下接收并存储rdata，并发出rready。

OK：表示接收到了rvalid，于是从READY跳转到了OK状态，控制data\_ok不会与addr\_ok错误对应。

END：如果从OK状态跳到END状态，则表示可以发出data\_ok信号，一个读事务结束。

1. 写事务

对于写事务，一共存在六个状态，FREE、VALID、READY、BVALID、OK、END：

FREE:表示写请求为空，可以接受写请求。

VALID:表示接收到了写请求，于是从FREE跳转到了VALID状态。在VALID状态下发出awvalid、addr\_ok以及有效的awsize、awaddr。获得正确wstrb并在下一拍发出。

READY:表示接收到了awready，于是从VALID跳转到了READY状态。在READY状态下发出wdata，并发出wvalid。

BVALID：表示接收到了wready，于是从READY跳转到了BVALID状态。在BVALID状态下发出bready信号。

OK：表示接收到了bvalid，于是从BVALID跳转到了OK状态，控制data\_ok不会与addr\_ok错误对应。

END：如果从OK状态跳到END状态，则表示可以发出data\_ok信号，一个写事务结束。

（二）重要模块1设计：指令类SRAM(inst sram-like)

1. 工作原理

inst\_addr\_ok 信号用来和 req 信号一起完成读写请求的握手。只有在 clk 的上升沿同时看到 inst\_req 和 inst\_addr\_ok 都为 1 的时候才是一次成功的请求握手。inst\_data\_ok对应读事务的时候，它是数据返回的有效信号;对应写事务的时候，它写响应有效信号。当握手成功时，传递数据。

1. 接口定义

input inst\_req,

input inst\_wr,

input [ 1:0] inst\_size,

input [31:0] inst\_addr,

input [31:0] inst\_wdata,

output [31:0] inst\_rdata,

output inst\_addr\_ok,

output inst\_data\_ok

1. 功能描述

接收指令的读写请求，并在握手成功时传递数据。

1. 重要模块2设计：数据类SRAM(data sram-like)
2. 工作原理

data\_addr\_ok 信号用来和 req 信号一起完成读写请求的握手。只有在 clk 的上升沿同时看到 data\_req 和 data\_addr\_ok 都为 1 的时候才是一次成功的请求握手。 data\_data\_ok对应读事务的时候，它是数据返回的有效信号;对应写事务的时候，它写响应有效信号。当握手成功时，传递数据。

1. 接口定义

input data\_req,

input data\_wr,

input [ 1:0] data\_size,

input [31:0] data\_addr,

input [31:0] data\_wdata,

output [31:0] data\_rdata,

output data\_addr\_ok,

output data\_data\_ok

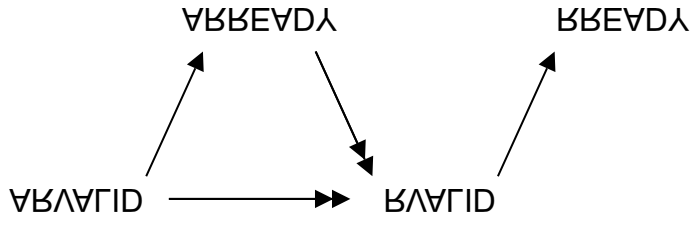
1. 功能描述

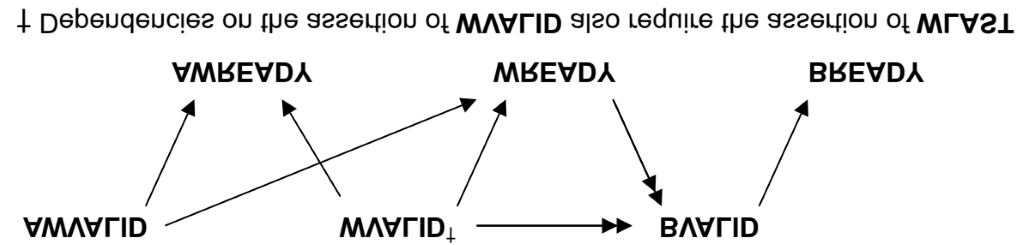
接收数据的读写请求，并在握手成功时传递数据。

（四）重要模块3设计：AXI模块

1. 工作原理

总线的两端可以分为主方master和从方slave，对于读操作来说，主方提出请求，从方接收请求返回数据；对于写操作来说，主方提出请求并发出写数据，从方接收请求和数据，AXI 总线协议采用握手机制完成主方与从方的交互，AXI总线有五个通道，写地址请求、写数据请求、写响应、读地址请求、读数据请求，每一个通道都有一对valid和ready信号用来实现握手。这些通道的valid和ready依赖关系如下图：





1. 接口定义
2. //ar
3. output  [ 3:0]  arid,
4. output  [31:0]  araddr,
5. output  [ 7:0]  arlen,
6. output  [ 2:0]  arsize,
7. output  [ 1:0]  arburst,
8. output  [ 1:0]  arlock,
9. output  [ 3:0]  arcache,
10. output  [ 2:0]  arprot,
11. output          arvalid,
12. input           arready,
13. //r
14. input   [ 3:0]  rid,
15. input   [31:0]  rdata,
16. input   [ 1:0]  rresp,
17. input           rlast,
18. input           rvalid,
19. output          rready,
20. //aw
21. output  [ 3:0]  awid,
22. output  [31:0]  awaddr,
23. output  [ 7:0]  awlen,
24. output  [ 2:0]  awsize,
25. output  [ 1:0]  awburst,
26. output  [ 1:0]  awlock,
27. output  [ 3:0]  awcache,
28. output  [ 2:0]  awprot,
29. output          awvalid,
30. input           awready,
31. //w
32. output  [ 3:0]  wid,
33. output  [31:0]  wdata,
34. output  reg[ 3:0]   wstrb,
35. output          wlast,
36. output          wvalid,
37. input           wready,
38. //b
39. input   [ 3:0]  bid,
40. input   [ 1:0]  bresp,
41. input           bvalid,
42. output          bready
43. 功能描述

接收CPU的读、写指令和数据请求，并将请求传递给类SRAM，当相应的指令和数据准备好时。完成对应的读事务和写事务操作。

三、实验过程（50%）

（一）实验流水账

11月16日 20：00-20：30 阅读讲义

11月16日 20：30-22：00 设计代码

11月17日 10：00-11：30 继续实现代码

11月17日 14：00-15：30 调试bug

11月18日19：30-22：30 撰写实验报告

（二）错误记录

1、错误1：data\_ok和addr\_ok不对应。

（1）错误现象

原来的版本里运行仿真通过，上板后修改随机种子发现有不通过的情况，在vivado上重现该错误，随机种子修改为16’h030f，运行仿真，出现了如图3.1的两个读data\_sram\_like的错误：

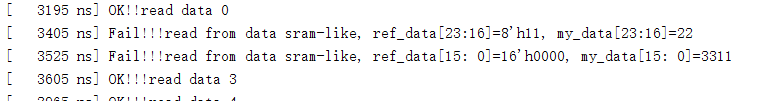


图3.1 read data 1和read data 2 测试错误

（2）分析定位过程

通过查看波形，发现data\_data\_ok与data\_addr\_ok握手有误，如图3.2的波形所示，data\_addr\_ok在read状态机的valid阶段发出addr\_ok，在write状态机的valid阶段也发出了一个addr\_ok，也就是先有读data的请求，再有写data的请求，由于该随机种子下是长延迟，由波形图可以看出rvalid延迟了很晚才来，而wready很快就来了，就导致write的状态机比read的状态机跳转更快，于是write先到了end态发出了data\_ok，read后到end态发出data\_ok，所以就成了write发出的data\_ok对应read发出的addr\_ok，read发出的data\_ok对应write发出的addr\_ok，导致读写响应的错误：

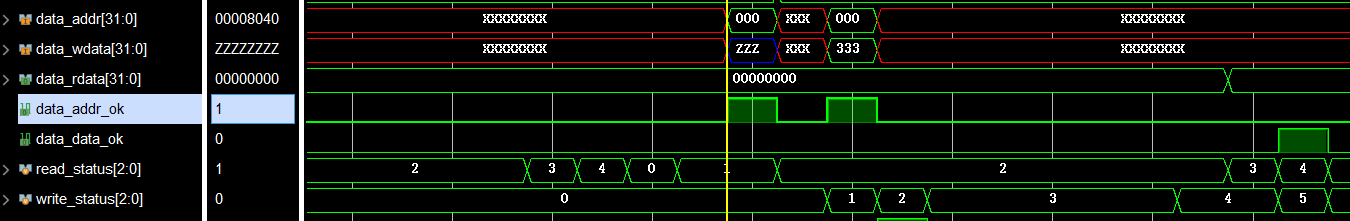


图3.2 相关波形

（3）错误原因

造成这样错误的原因是read先发addr\_ok后write也发addr\_ok且write的data\_ok比read更早到达，所以需要在read发出addr\_ok后将write的状态机阻塞住。

（4）修正效果

我们采用了一种较为粗暴的方式阻塞状态机，如图3.3的代码，只要read进入了其他的状态，就将write阻塞在了free态不允许它发addr\_ok，这样就不会有错误的对应了。

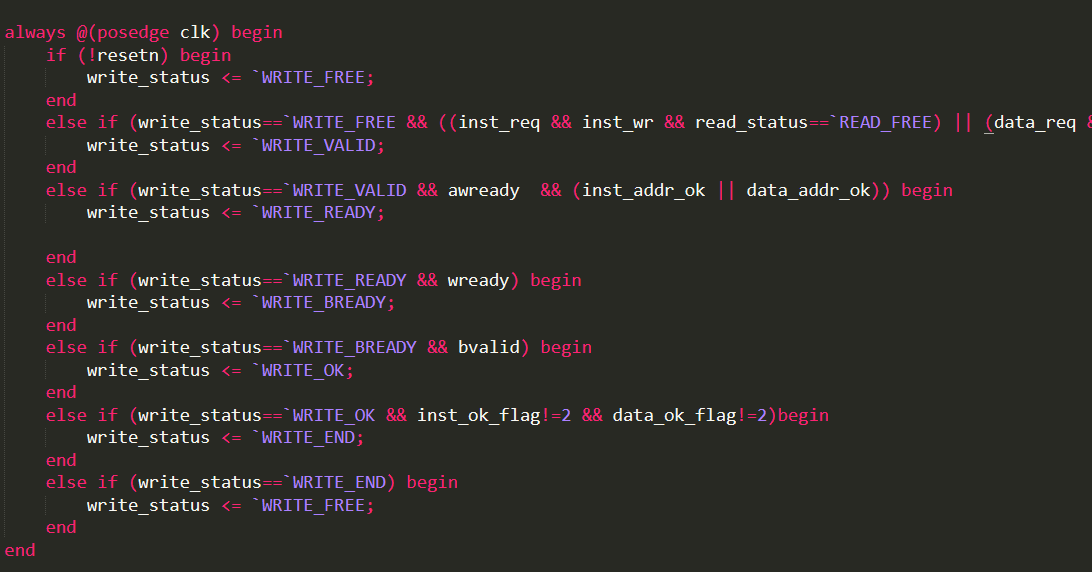
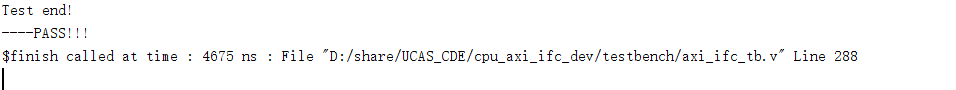


图3.3 增加阻塞后的状态机跳转

修改后，该随机种子下的测试顺利通过。

图3.4 测试通过

综合、布局布线、生成bit文件，然后上板验证，我们一个一个地拨了256个随机种子，全部正常显示。

四、实验总结（可选）

本次实验较前几次实验难度突然增大，一是因为上学期组成原理实验没有设计总线，二是因为讲义比之之前的部分更加难懂，我们这次设计虽然通过了测试，但仍然认为有可以改进的地方，甚至有些地方的理解可能仍然是错误的，只是顺利通过了测试而已。我们之前版本的代码在进行测试时，总是随机种子仿真时可以通过，但是上板采样随机种子时总会不成功，为此一遍一遍地修改设计，大概上板有4到5次才最终成功地通过了所有的256个随机种子采样，但是核心的错误只有上面所述的一个，就是读写的data\_ok和addr\_ok不能对应的问题，这恰好就是随机延迟带来的，我们想了很多个修改的办法，最后才确定了用这种很“粗暴”的阻塞方案，之后如果有时间，也会考虑重新修改这一部分的设计。