Lab5任务

握手的必要性

类SRAM总线

AXI总线

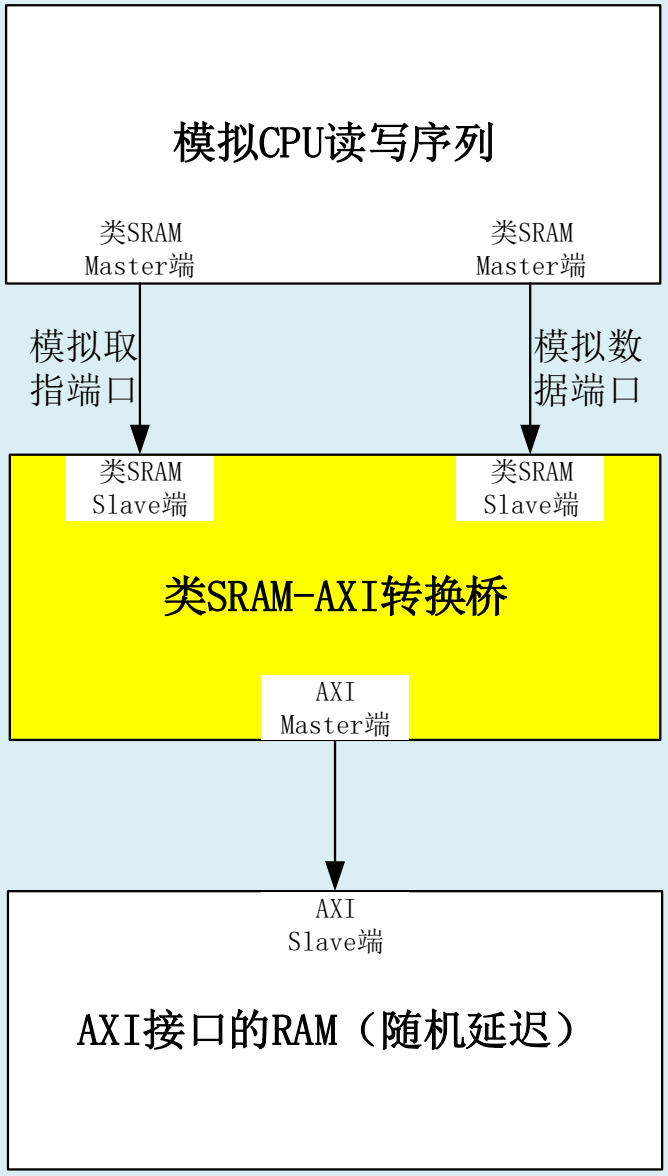
其他

Lab5任务

- 第一周，类SRAM接口到AXI接口的转换桥RTL：
- 第二周，CPU顶层修改为AXI接口：
  - 完成固定延迟的功能测试
- 第三周：
  - 完成随机延迟的功能测试

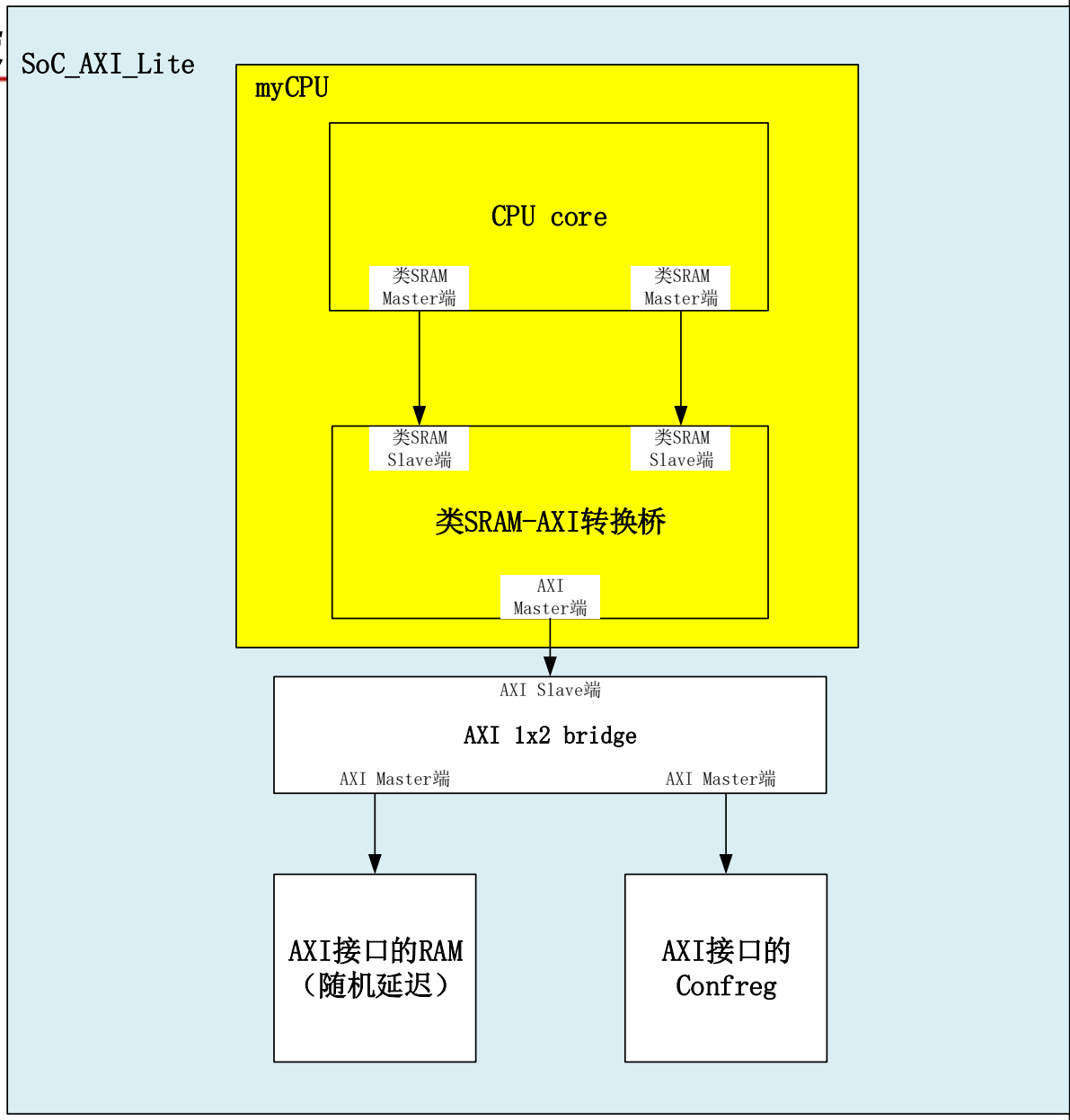
Lab5任务——第一阶段

- **类SRAM接口到AXI接口的转换桥RTL：**
  - **黄色部分为需要实现的**
  - **其他部分由我们提供**



Lab5任务——第二、三阶段

- **CPU顶层修改为AXI接口：**
  - **黄色部分为需要实现的**
  - **CPU内部可以使用类SRAM接口转换**



Lab5任务

握手的必要性

类SRAM总线

AXI总线

其他

握手的必要性

- **读写单周期返回的不足：**
  - **限制CPU频率**
- **传输握手：**
  - **可协调不同速度的设备间交换数据**
- **Master和slave区分：**
  - **Master：发起请求**
  - **Slave：响应请求，返回数据**
  - **Lab4中，CPU和inst ram，哪个是master，哪个是slave?**

握手的必要性

- **握手分类：**
  - 约定握手（不需要旗帜）：
  - 单向握手（一方摇旗帜）：
  - 双向握手（双方摇旗帜）：

  握手一旦成功，只会持续一拍！多个周期看到握手成功，那是针对不同传输的握手！

举个例子：

目标：两个山头被包围，同时约定突围

方式：

- 约定握手：天黑即突围
- 单向握手：山头A对山头B说：我随时可以，你摇旗帜，我们就突围。
- 双向握手：山头A和山头B同时摇旗帜，即突围

握手的必要性

- **所有传输或直接，或隐含都是有握手的：**
  - **单周期返回的RAM： 第一拍给地址，下一拍给数据**

    地址传输是单向握手，slave随时接受地址请求，片选en置上，就认为握手成功

    数据传输时约定握手，约定地址传输下拍数据就返回了。

- **握手分类：**
  - **约定握手：**
  - **单向握手：类SRAM数据传输，AHB协议数据传输(Hready)**
  - **双向握手：AXI协议是双向握手，valid & ready**

从上到下，握手越来越复杂，所实现的功能却越来越完善。

双向握手可实现约定握手、单向握手的功能。但反之却不然。

- Lab5任务
- 握手的必要性
- **类SRAM总线**
- AXI总线
- 其他

类SRAM总线

- 为SRAM接口增加地址传输握手信号addr_ok和数据传输握手信号data_ok
- 地址传输：双向握手，req & addr_ok同时有效
- 数据传输：单向握手，data_ok有效，master随时可以接受
- 握手成功只持续一拍。

信号	位宽	方向	功能
clk	1	input	时钟
req	1	master—>slave	请求信号，为1时有读写请求，为0时无读写请求
wr	1	master—>slave	该次请求是写
size	[1:0]	master—>slave	该次请求传输的字节数，0: 1byte；1: 2bytes；2: 4bytes。
addr	[31:0]	master—>slave	该次请求的地址
wdata	[31:0]	master—>slave	该次请求的写数据
addr_ok	1	slave—>master	该次请求的地址传输OK，读：地址被接收；写：地址和数据被接收
data_ok	1	slave—>master	该次请求的数据传输OK，读：数据返回；写：数据写入完成。
rdata	[31:0]	slave—>master	该次请求返回的读数据。

类SRAM总线

类SRAM接口与SRAM接口的不同

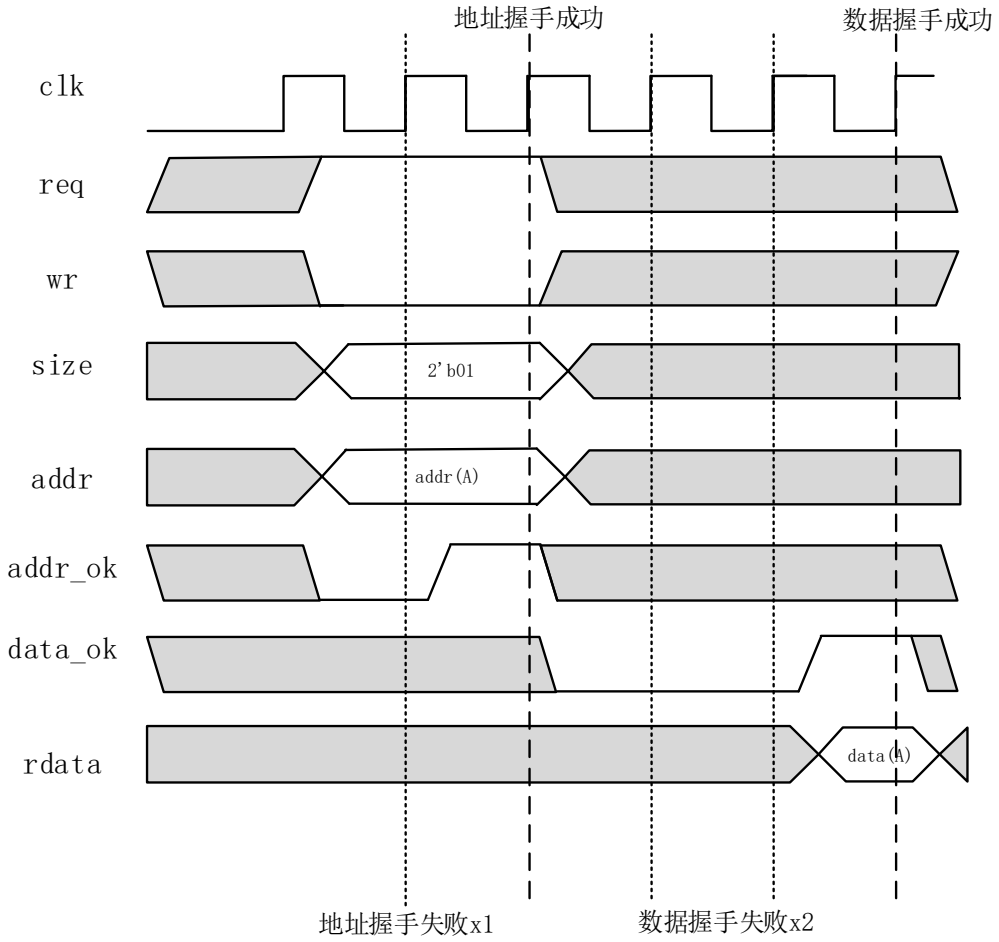
类SRAM地址为字节寻址

1. addr[1:0]=2’b00时，可能的组合：
   size=2’b00, size=2’b01, size=4’b10,
1. addr[1:0]=2’b01时，可能的组合：
   size=2’b00
1. addr[1:0]=2’b10时，可能的组合：
   size=2’b00, size=2’b01
1. addr[1:0]=2’b11时，可能的组合：
   size=2’b00

	data[31:24]	data[23:16]	data[15:8]	data[7:0]
size=2’b00,addr=2’b00	-	-	-	valid
size=2’b00,addr=2’b01	-	-	valid	-
size=2’b00,addr=2’b10	-	valid	-	-
size=2’b00,addr=2’b11	valid	-	-	-
size=2’b01,addr=2’b00	-	-	valid	valid
size=2’b01,addr=2’b10	valid	valid	-	-
size=2’b10,addr=2’b00	valid	valid	valid	valid

类SRAM总线

一次读时序图

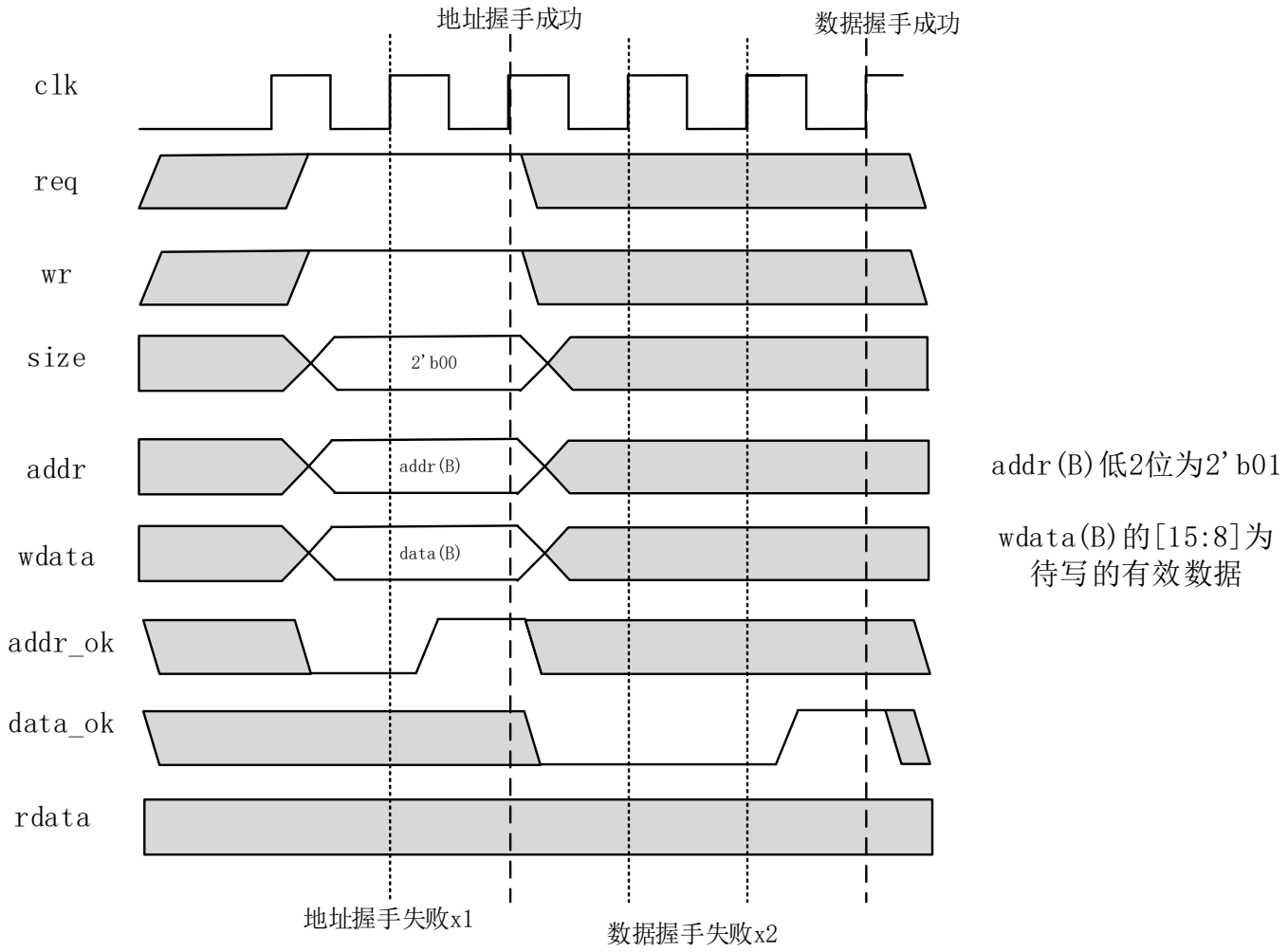


addr(A)低2位为2'b10

rdata(A)的[31:16]为读出的有效数据

类SRAM总线

一次写时序图



类SRAM总线

连续写读时序图

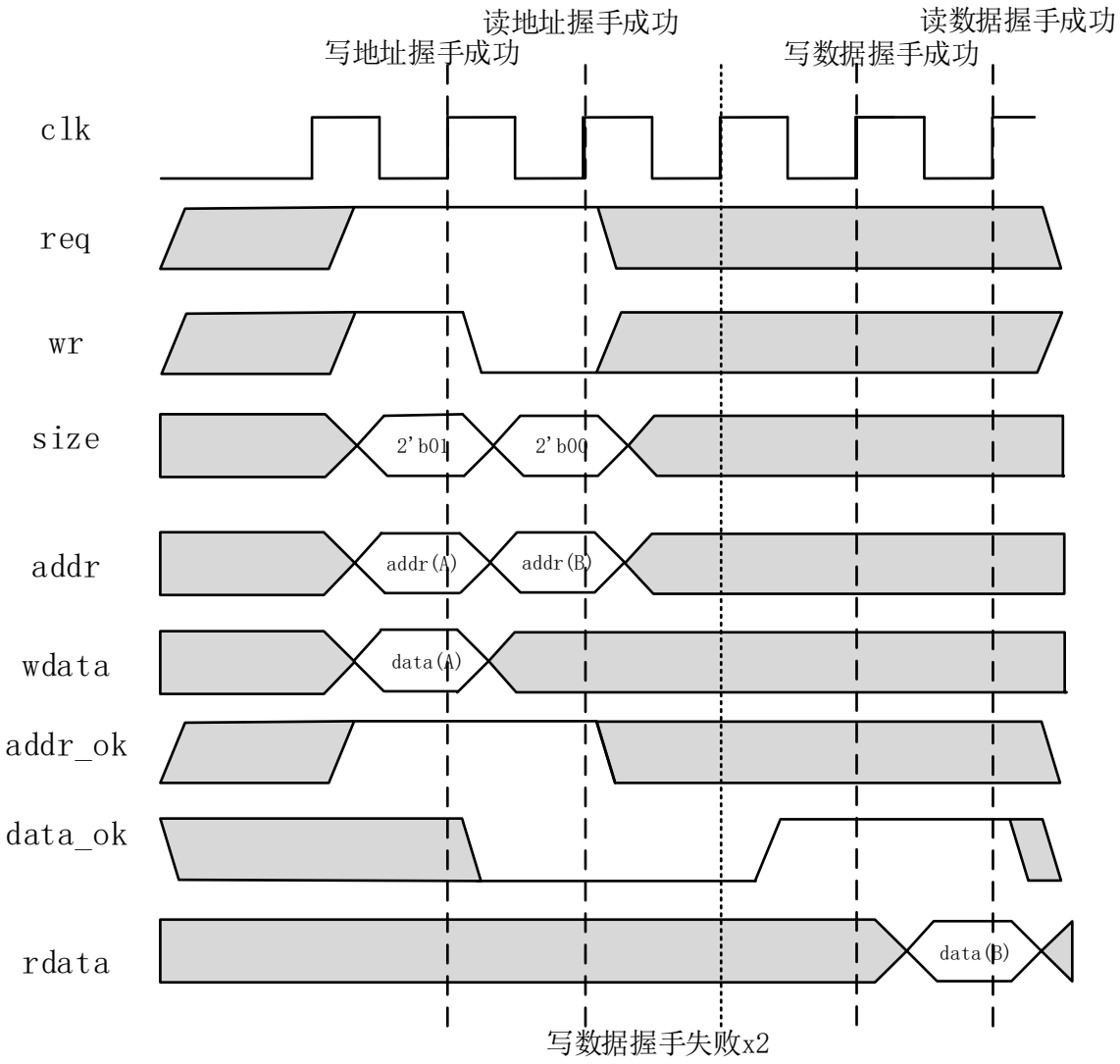
- slave返回的data_ok必须顺序返回的。
- 可能出现多次地址握手后，才会出线数据握手。

addr_ok->addr_ok->addr_ok->addr_ok->...->data_ok。

此时数据握手是对应第一次的传输。

master端避免这一情况的出现可以通过拉低req信号。

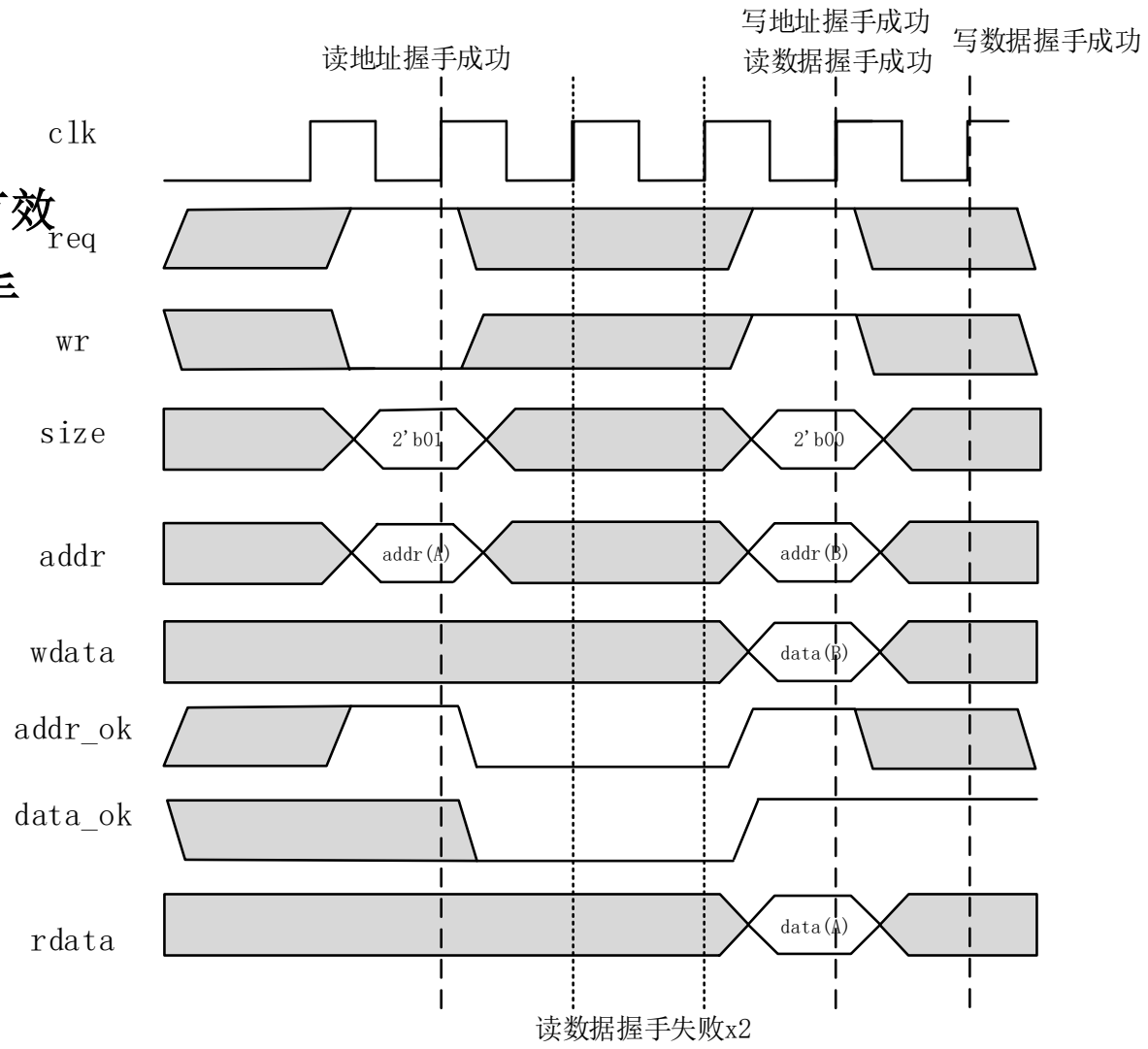
slave避免这一情况的出现可以通过拉低addr_ok



类SRAM总线

- **连续读写时序图**

- 当addr_ok和data_ok同时有效时，是针对不同请求的握手



Lab5任务

握手的必要性

类SRAM接口

AXI总线

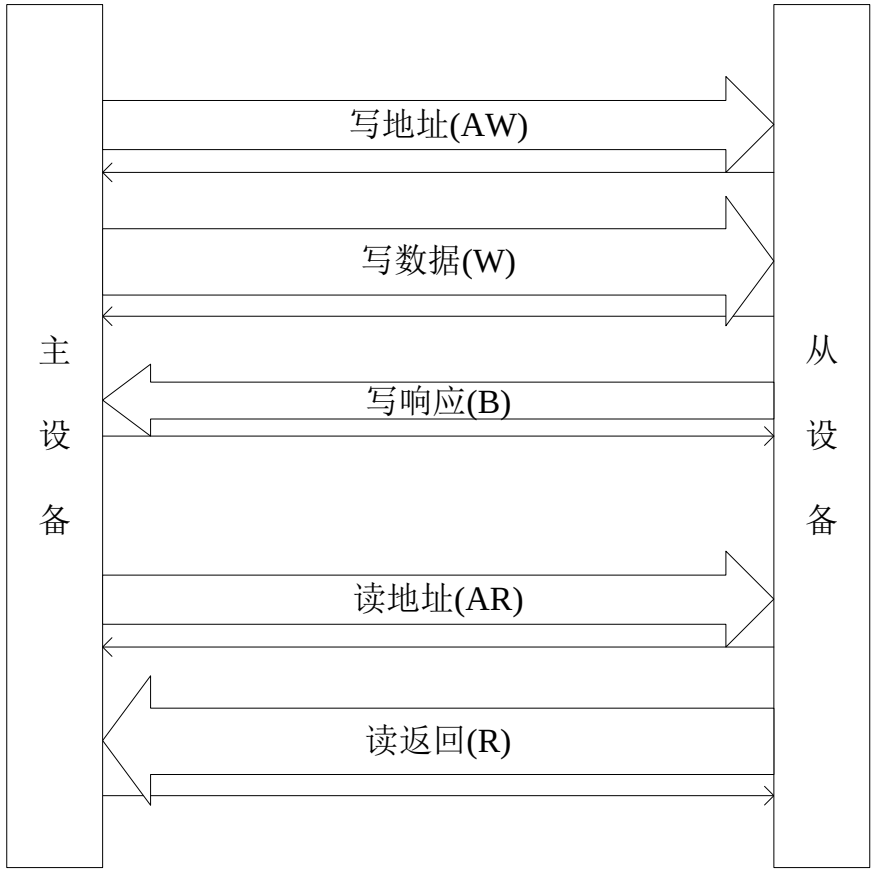
其他

AXI总线

- **由五个通道组成**
  - 写地址、写数据、写响应
  - 读地址、读返回

  读写分开、命令数据分开

  每个通道有自己的握手信号
- **基本定义**
  - 总线事务(transaction):
    - 一次完整读写过程
  - 传输(transfer):
    - 传输一个周期的数据
    - 在VALID、READY同时为高的周期完成一次传输



AXI总线

- **AXI接口信号：写地址以aw开头，写数据以w开头，写响应以b开头**
  读地址以ar开头， 读数据以r开头
- **时钟复位 与 握手信号**

名称	宽度	方向	描述
aclk	1	—>mster/slave	时钟
hresetn	1	—>mster/slave	复位信号，低电平有效
读地址握手			
arvalid	1	—>slave	读请求地址有效
arready	1	—>master	从设备准备好，已接受读地址
读数据握手			
rvalid	1	—>mater	从设备返回数据，读数据有效
rready	1	—>slave	主设备准备好，已接受返回的读数据

时钟复位 与 握手信号(续)

名称	宽度	方向	描述
写地址握手			
awvalid	1	—>slave	写请求地址有效
awready	1	—>master	从设备准备好，已接受写地址
写数据握手			
wvalid	1	—>slave	写数据有效
wready	1	—>master	从设备准备好，已接受写数据
写响应握手			
bvalid	1	—>master	从设备回应写结果，写回应有效
bready	1	—>slave	主设备准备好，已接受写回应

握手信号无先后

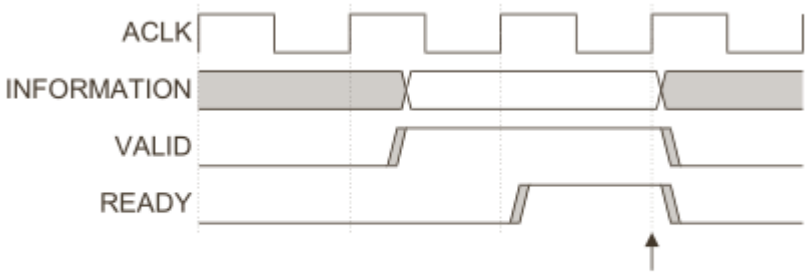


Figure 3-1 VALID before READY handshake

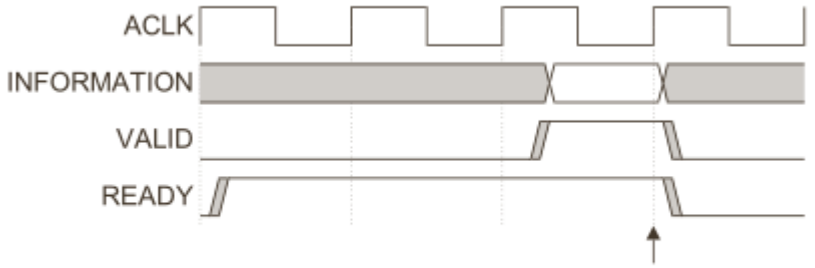


Figure 3-2 READY before VALID handshake

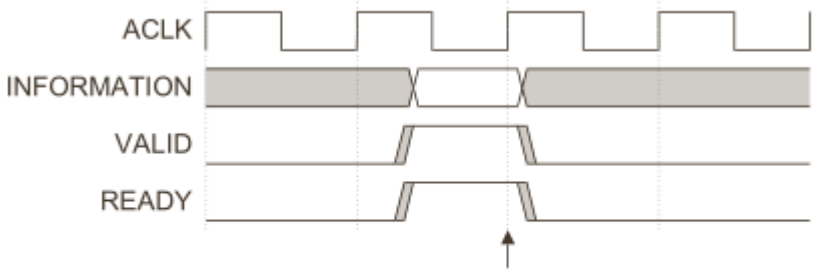
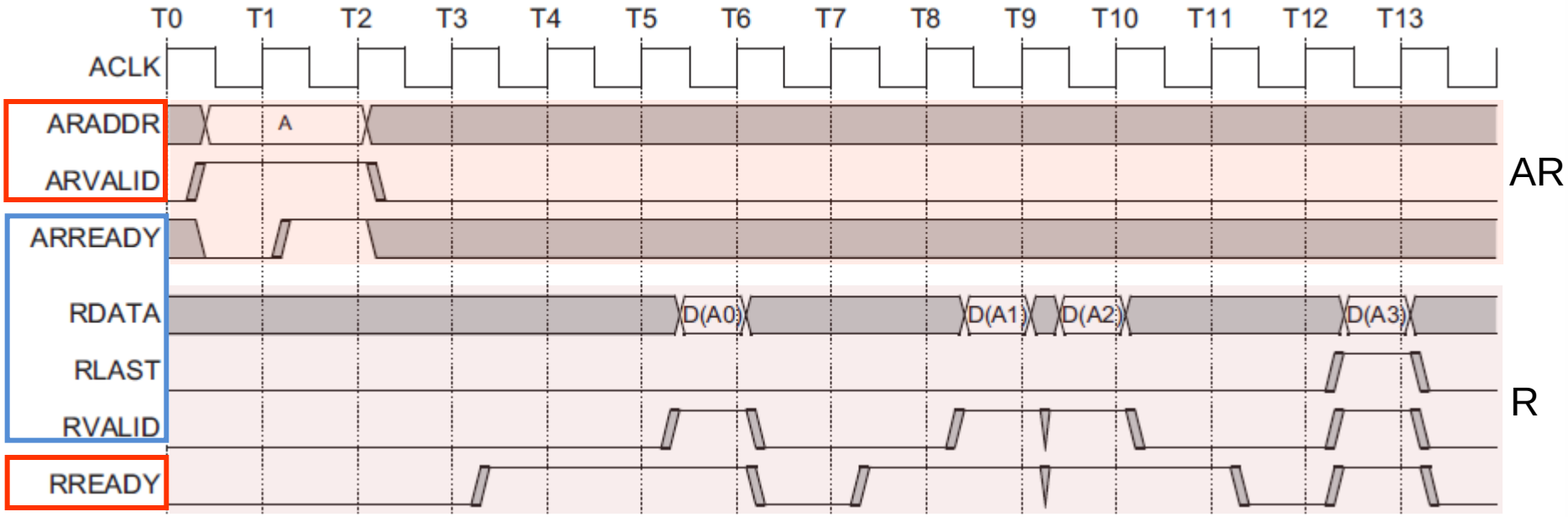


Figure 3-3 VALID with READY handshake

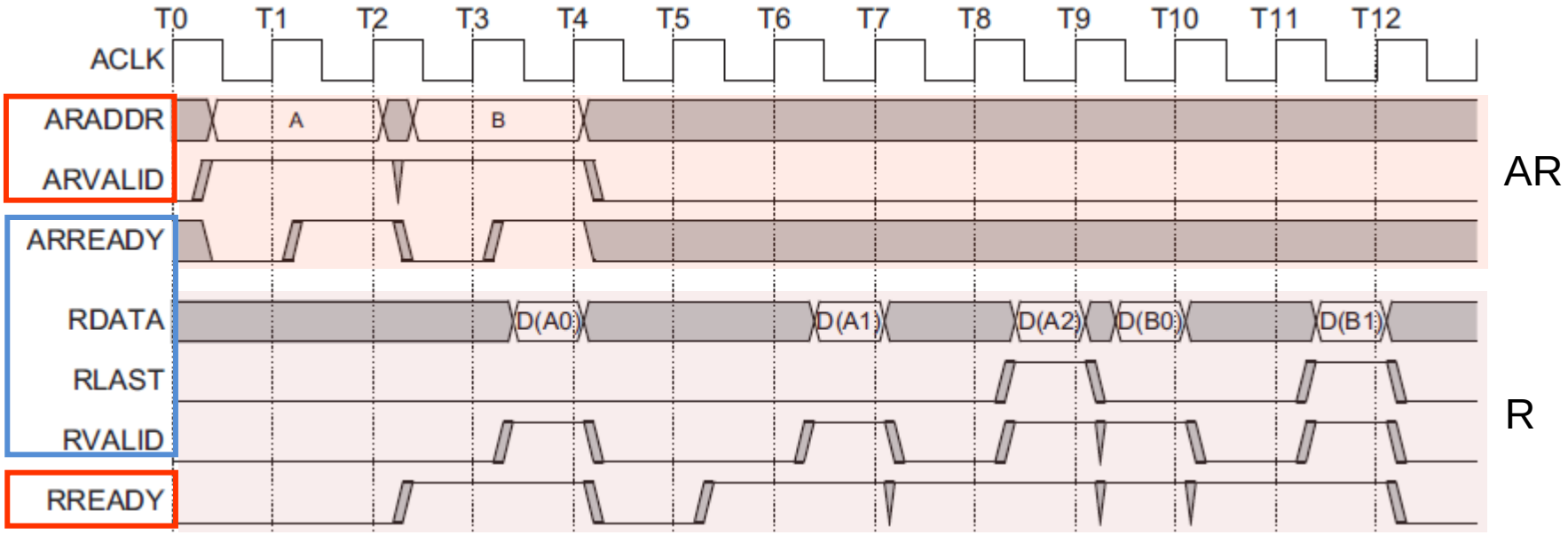
AXI总线——AXI读事务示例



master out

slave out

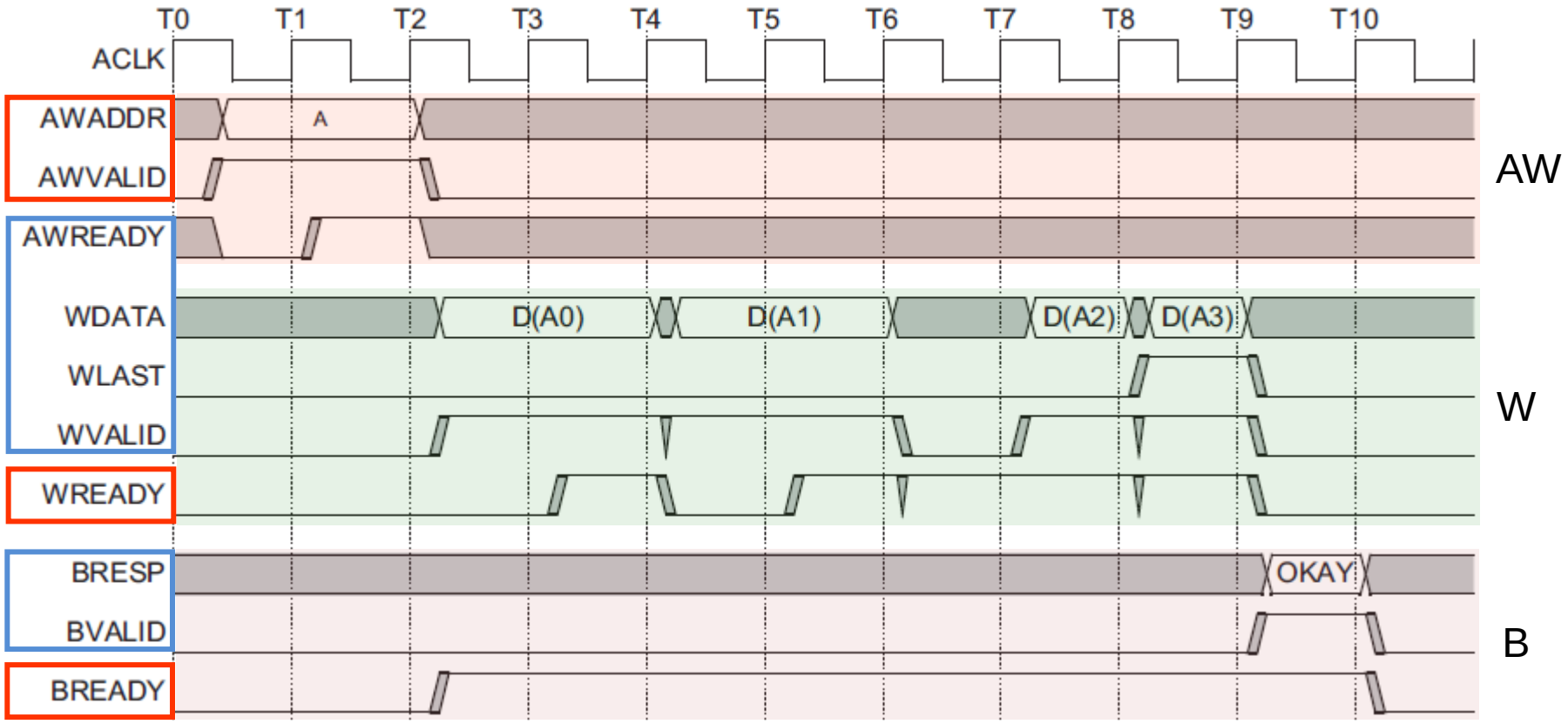
AXI总线——AXI重叠读事务示例



master out

slave out

AXI总线——AXI写事务示例



master out

slave out

- **读通道握手依赖关系**

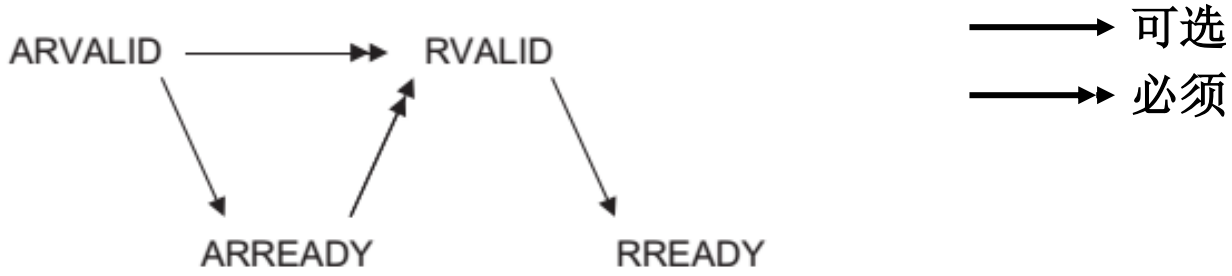


Figure 3-4 Read transaction handshake dependencies

- **写通道握手依赖关系**

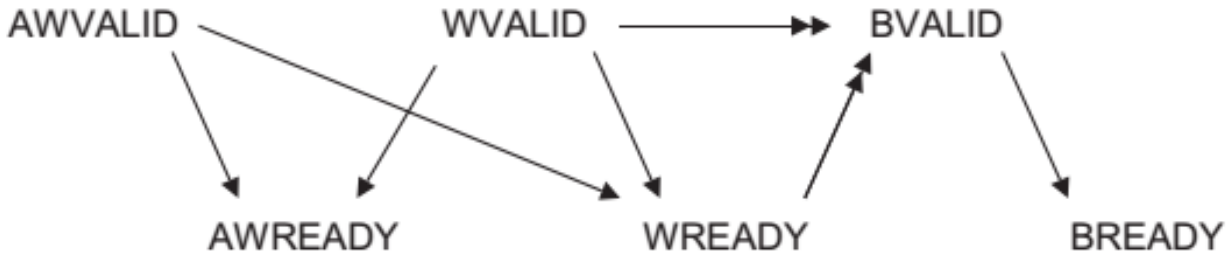


Figure 3-5 Write transaction handshake dependencies

AXI总线——AXI其他关键信号

- **ID: 支持读写乱序 （建议指令、数据对ID进行区分）**
  - 每个事务都有ID标签(ARID/RID/AWID/WID/BID)
  - 同ID的事务必须按序
  - 不同ID的事务可以乱序
- **BURST/LEN/SIZE: 突发传输控制 （暂不需要关心）**
  - 突发传输类型: FIXED, INCR, WRAP
  - 突发传输长度: 1~16
  - 突发传输单位: 1,2,4~BW
- **WSTRB: 写掩码 （尽量与AWSIZE对应，通常可以三字节写）**
  - 为高的位对应的字节写有效
- **RESP: 读/写响应状态 （不支持BUS Error）**
  - 响应类型：OKAY、EXOKAY、SLVERR、DECERR

AXI总线——AXI其他关键信号（不需要关心）

- **ARLOCK/AWLOCK: 原子性操作**
  - 00: 普通
  - 01: 独占
    - 独占地对某个地址先读后写，若期间无其它主设备写，则成功写入，否则失败，通过RESP返回状态。类似LL/SC
  - 10: 加锁
    - 从第一个加锁事务开始，到第一个普通事务结束
    - 将所访问的从设备区域加锁，阻止其它主设备访问
- **ARPROT/AWPROT: 访问保护**
  - [0]: 普通/特权
  - [1]: 安全/非安全
  - [2]: 数据/指令
- **ARCACHE/AWCACHE: 缓存控制**
  - [0]: 可写缓冲，未到达目的即回响应
  - [1]: 可缓存，cached/uncached，读预取，写合并
  - [2]: 读分配，如果读发生缓存缺失则分配一个缓存块
  - [3]: 写分配，如果写发生缓存缺失则分配一个缓存块

AXI总线

- AXI对比购物

AXI总线

- **红色为重点关注，黑色有固定分配**

表 1-3 32 位 AXI 接口信号

信号	位宽	方向	功能	备注
AXI 时钟与复位信号				
aclk	1	input	AXI 时钟	
aresetn	1	input	AXI 复位，低电平有效	
读请求地址通道，（以 ar 开头）				
arid	[3:0]	master—>slave	读请求的 ID 号	取指为 0 取数为 1
araddr	[31:0]	master—>slave	读请求的地址	
arlen	[7:0]	master—>slave	读请求控制信号，请求传输的长度(数据传输拍数)	固定为 0
arsize	[2:0]	master—>slave	读请求控制信号，请求传输的大小(数据传输每拍的字节数)	
arburst	[1:0]	master—>slave	读请求控制信号，传输类型	固定为 2'b01
arlock	[1:0]	master—>slave	读请求控制信号，原子锁	固定为 0
arcache	[3:0]	master—>slave	读请求控制信号，CACHE 属性	固定为 0
arprot	[2:0]	master—>slave	读请求控制信号，保护属性	固定为 0
arvalid	1	master—>slave	读请求地址握手信号，读请求地址有效	
arready	1	slave—>master	读请求地址握手信号，slave 端准备好接受地址传输	
读请求数据通道，（以 r 开头）				
rid	[3:0]	slave—>master	读请求的 ID 号，同一请求的 rid 应和 arid 一致	指令回来为 0

AXI总线

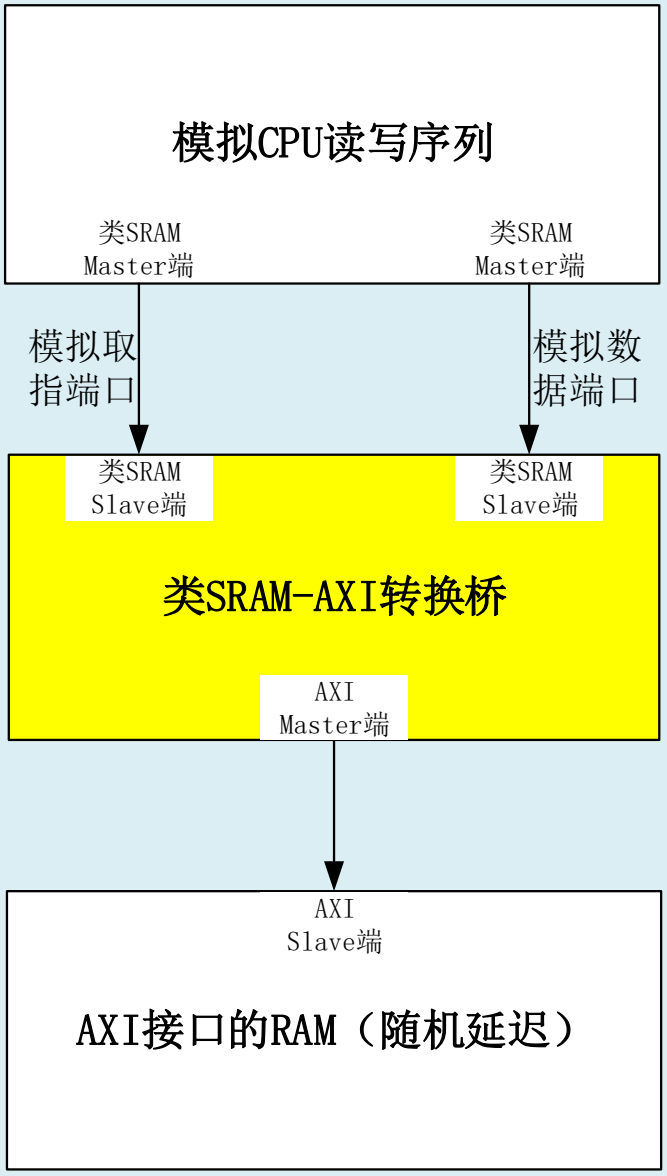
- **红色为重点关注，黑色有固定分配**

				数据回来为 1
rdata	[31:0]	slave—>master	读请求的读回数据	
rresp	[1:0]	slave—>master	读请求控制信号，本次读请求是否成功完成	可忽略
rlast	1	slave—>master	读请求控制信号，本次读请求的最后一拍数据的指示信号	可忽略
rvalid	1	slave—>master	读请求数据握手信号，读请求数据有效	
rready	1	master—>slave	读请求数据握手信号，master 端准备好接受数据传输	
写请求地址通道，（以 aw 开头）				
awid	[3:0]	master—>slave	写请求的 ID 号	固定为 1
awaddr	[31:0]	master—>slave	写请求的地址	
awlen	[7:0]	master—>slave	写请求控制信号，请求传输的长度(数据传输拍数)	固定为 0
awsize	[2:0]	master—>slave	写请求控制信号，请求传输的大小(数据传输每拍的字节数)	
awburst	[1:0]	master—>slave	写请求控制信号，传输类型	固定为 2'b01
awlock	[1:0]	master—>slave	写请求控制信号，原子锁	固定为 0
awcache	[3:0]	master—>slave	写请求控制信号，CACHE 属性	固定为 0
awprot	[2:0]	master—>slave	写请求控制信号，保护属性	固定为 0
awvalid	1	master—>slave	写请求地址握手信号，写请求地址有效	
awready	1	slave—>master	写请求地址握手信号，slave 端准备好接受地址传输	
写请求数据通道，（以 w 开头）				
wid	[3:0]	master—>slave	写请求的 ID 号	固定为 1
wdata	[31:0]	master—>slave	写请求的写数据	
wstrb	[3:0]	master—>slave	写请求控制信号，字节选通位	
wlast	1	master—>slave	写请求控制信号，本次写请求的最后一拍数据的指示信号	固定为 1
wvalid	1	master—>slave	写请求数据握手信号，写请求数据有效	
wready	1	slave—>master	写请求数据握手信号，slave 端准备好接受数据传输	
写请求响应通道，（以 b 开头）				
bid	[3:0]	slave—>master	写请求的 ID 号，同一请求的 bid、wid 和 awid 应一致	可忽略
bresp	[1:0]	slave—>master	写请求控制信号，本次写请求是否成功完成	可忽略
bvalid	1	slave—>master	写请求响应握手信号，写请求响应有效	
bready	1	master—>slave	写请求响应握手信号，master 端准备好接受写响应	

- Lab5任务
- 握手的必要性
- 类SRAM总线
- AXI总线
- 其他

其他——第一阶段任务

- **2x1: 两个类SRAM的Slave到一个AXI的Master的桥，**
- **仲裁：**



总线接口设计思考

- **问题1：每个总线事务需要完整执行完毕**
  - 复位期间不要对外发起总线请求

    此时外部总线或从设备可能尚未结束复位，不能响应总线请求，有可能造成各状态机进入错误状态。
  - 不要重复发请求

    如果因为控制的不精细造成同一条指令的取指或同一条访存指令的访存请求重复发出多次，对于普通的SRAM而言没有风险，但是一旦发到总线上，则每个请求都是一个独立的事务，都必须有始有终的处理完，而且每个事务都会产生其执行效果。
  - Store 或 Uncache load，不能够推测执行。

    即如果一条uncache load指令会被例外取消，那么这条uncache load指令不能发出总线请求。
  - 请求一旦置起后，如果没有响应，则不能更换请求。

总线接口设计思考

- **问题2：AXI读、写通道分离**
  - 同地址读写不保证顺序性

    AXI读、写通道分离，导致可能出现一种情况：当存在地址相关的总线读写序列是，在Master端写地址和写数据请求先于读请求交互完毕，但是在Slave端读请求先于写地址和写数据请求到达，从而导致读返回的数据并不是写入的新值，出错。
  - 解决的方式：Master端在发起读请求时，先要确保该请求与“已发出请求但尚未接收到写响应的写请求”不存在地址相关。
  - 先读后写：CPU解决
  - 先写后读：

    暴力：所有的请求必须确保前一个总线事务完全完成。

    高效：可以记录那些“已发出请求但尚未接收到写响应的写请求”的地址、类型等信息，供后续读请求查询判断用。

谢谢！