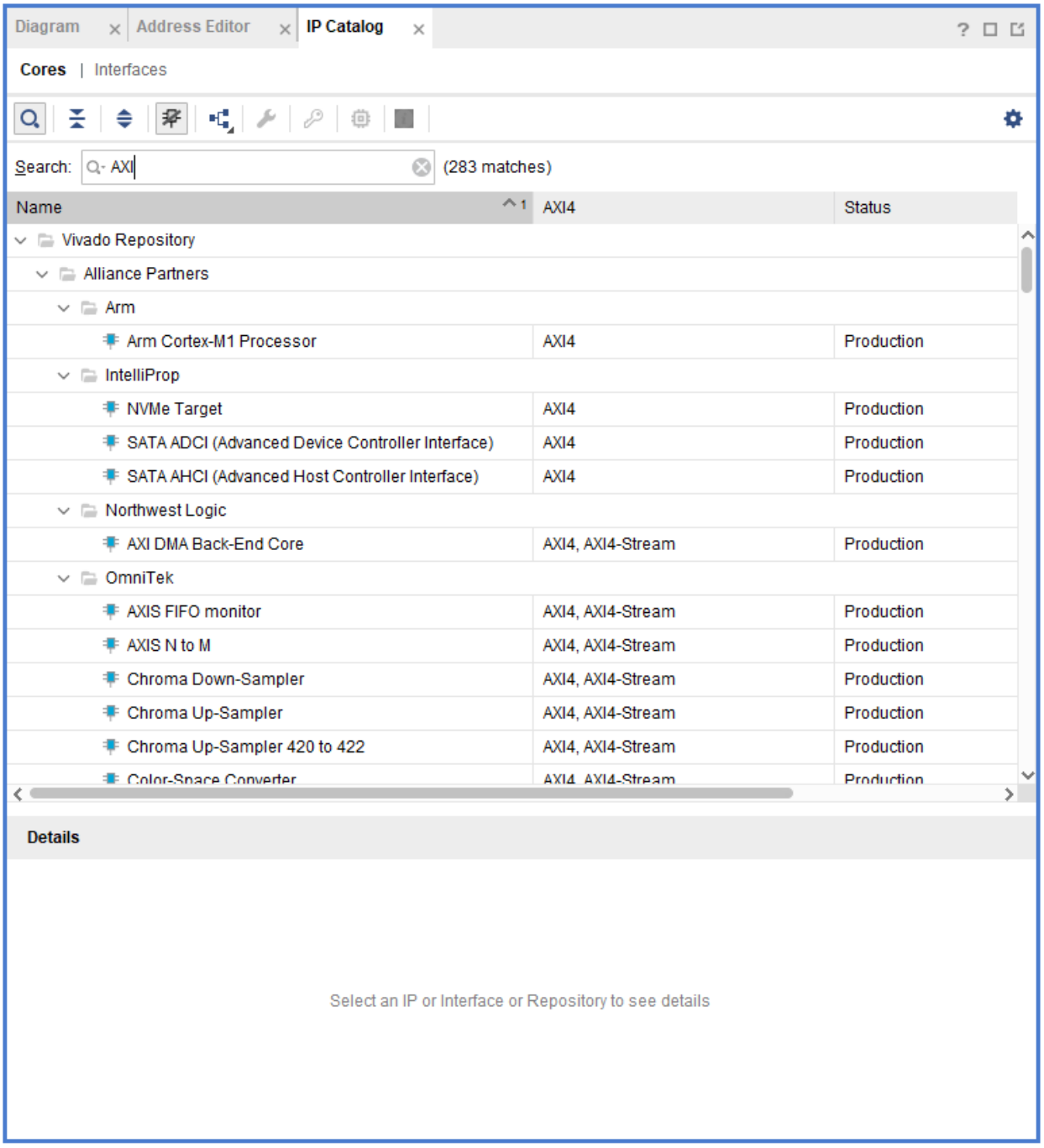
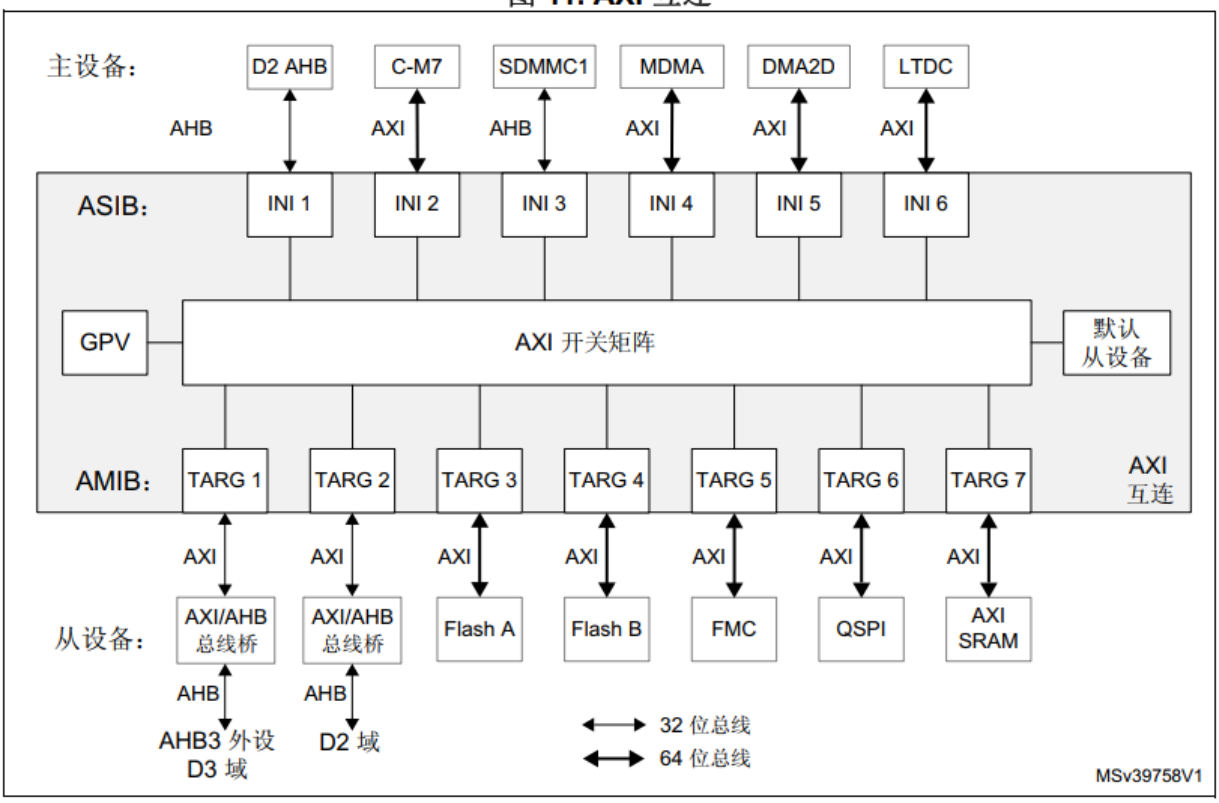
[L10]AXI总线协议初步学习

要想和外设通信,必须有一个总线,AXI是ARM发明的一个牛逼片上总线,我们需要大致地理解一下它,方便使用,如果想深入了解需要看UG1037文档.AXI优点很多,比如很多IP都是搞了AXI接口对接之类的,比如Vivado就有很多这样的IP.



其他ARM平台比如STM32上也用AXI必须进行通信,除非魔改核心.

图 11. AXI 互连



AXI协议特点(Not All):

- 地址/控制信号与数据处于不同阶段
- 支持非对齐数据传输
- 支持突发式传输,且仅需提供首地址
- 独立的读写通道,可提供低资源消耗DMA
- 支持发送多个待执行地址(outstanding address)
- 支持乱序数据包收发
- 允许插入寄存器以保证时序收敛

AXI区别于其他总线的地方在于它的通道分离特性,AXI4协议具有五个独立的通道:**读地址通道,读数据通道,写地址通道,写数据通道,写回应通道.**

通道分离带来了很多优点,比如支持在**未收到回应时发送多个待执行操作的地址信息,**这样的设置有利于流水线操作,提升传输效率与速度.

当多个从机操作完成时间不一致时,完成顺序与Master的控制顺序可以不同,这就使得AXI总线具有了**乱序数据包收发**的特性等等.

注:不同颜色代表不同部分讲解

通道	信号	方向	描述	通道	信号	方向	描述
Global	ACLK		全局时钟				
Global	ARESETn		全局复位，低有效				
WADDR	AWID	M2S	写地址ID号	RADDR	ARID	M2S	读地址ID号
WADDR	AWADDR	M2S	写地址	RADDR	ARADDR	M2S	读地址
WADDR	AWLEN	M2S	突发长度	RADDR	ARLEN	M2S	突发长度
WADDR	AWSIZE	M2S	突发数据包大小	RADDR	RWSIZE	M2S	突发数据包大小
WADDR	AWBURST	M2S	突发类型	RADDR	ARBURST	M2S	突发类型
WADDR	AWLOCK	M2S	锁定类型	RADDR	ARLOCK	M2S	锁定类型
WADDR	AWCACHE	M2S	存储类型	RADDR	ARCACHE	M2S	存储类型
WADDR	AWPROT	M2S	保护类型	RADDR	ARPROT	M2S	保护类型
WADDR	AWQOS	M2S	服务质量	RADDR	ARQOS	M2S	服务质量
WADDR	AWREGION	M2S	区域标志	RADDR	ARREGION	M2S	区域标志
WADDR	AWUSER	M2S	用户自定义	RADDR	ARUSER	M2S	用户自定义
WADDR	AWVALID	M2S	写地址有效	RADDR	ARVALID	M2S	写地址有效
WADDR	AWREADY	S2M	准备接收写地址	RADDR	ARREADY	S2M	准备接收写地址
WDATA	WID	M2S	写过程ID标签	RDATA	RID	S2M	读操作ID标签
WDATA	WDATA	M2S	写数据	RDATA	RDATA	S2M	读数据
WDATA	WSTRB	M2S	数据片选				
WDATA	WLAST	M2S	最后一个写数据	RDATA	RLAST	S2M	最后一个读数据
WDATA	WUSER	M2S	用户自定义	RDATA	RUSER	S2M	用户自定义
WDATA	WVALID	M2S	写操作有效	RDATA	RVALID	S2M	写操作有效
WDATA	WREADY	S2M	准备接收写数据	RDATA	RREADY	M2S	准备接收写数据
WRESP	BID	S2M	写回应ID标签				
WRESP	BRESP	S2M	写回应	RDATA	RRESP	S2M	读回应
WRESP	BUSER	S2M	用户自定义				
WRESP	BVALID	S2M	写回应有效				
WRESP	BREADY	M2S	准备接收写回应				

我们能够从上面的表中总结得到以下信息:

- AXI4协议将信号分为五个通道:分别是写地址通道(WADDR),写数据通道(WDATA),写回应通道(WRESP),读地址通道(RADDR),读数据通道(RDATA).
- 每个通道中均含有READY信号以及VALID信号(比如AWREADY,AWVALID...),并且除了READY信号外通道内的其他信号方向是一致的,如果抛开这两个握手信号来讲,**通道中信号方向是单向的.**

- 每个通道都具有ID标签,不同的传输任务具有不同的ID,这也是AXI总线支持乱序收发的基础.
- 总结来看,ID信号,VALID,READY是每个通道的共用信号,除了RDATA直接包含RRESP,这是协议设计的原因,至于为什么这么做,网上也没个统一的标准.

但是说了是简单了解,自然不会深入,但是也会让你了解到整个AXI时许的主要内容,如果你一定要想写了解,请点击:

https://developer.arm.com/documentation/ihi0022/hc/?lang=en (下面所有图都来自这个文档)

拆分每个信号出来理解,比如读事务:

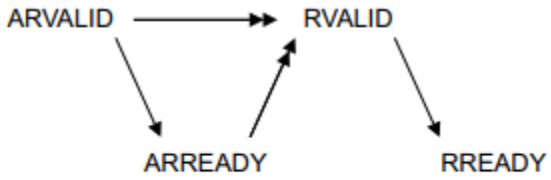


Figure A3-5 Read transaction handshake dependencies

我现在假设我开发的是一个外设,我是从机,主机会给我写一个地址,写到ARADDR(和其他地址相关端口,后面再说),然后就会拉高自己的ARVALID,我们收到地址,存好之后,就拉高ARREADY,相当于告诉主机,我收到地址了,然后准备好数据塞在数据端口RDATA,然后设置RVALID,然后主机设置RREADY告诉从机,我已经收到你的数据了.在最后一个时钟还需要设置RLAST,RLAST和RVALID同时设置,来告知这是最后一个传输了.

写事务稍微复杂一些:

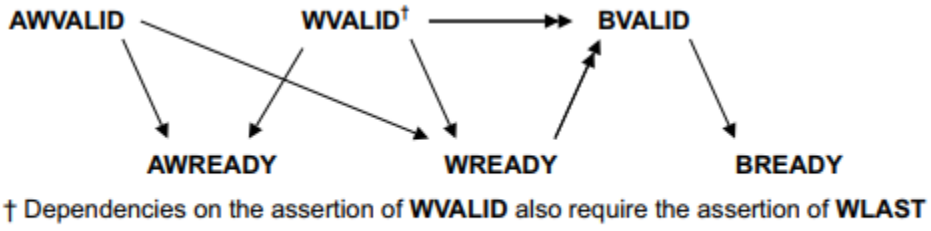


Figure A3-6 AXI3 write transaction handshake dependencies

同样先写入到AWADDR(和其他地址相关端口,后面再说),然后设置AWVALID,从机接收确认后设置AWREADY,然后主机把数据塞在WDATA中,并设置WVALID有效,从机收到后表示

WREADY有效,然后到从机真正把数据写入后,设置BVALID,告诉主机,我写完了,主机也回报BREADY表示好的,我知道了,当然最后一个写也要告知哦,不告知不能停止.

另外VALID和READY的发生时机可以是VALID先拉高,READY先拉高,或者同时拉高.各条件之间也不严格限定这个顺序,具体也可以看手册.

注意:上面不管是读还是写,地址过程仅一次,之后便会按照多种不同模式传输.

刚才说到地址数据,这里地址多少位数据多少位,那么肯定是根据设计来的,看看一开始STM32的图,有些AXI线路以32位连接,有些以64位连接,为了简化情况,暂时认为都是32位连接.这样每一个从机的地址空间大小就是4KB,每个寄存器占用4B.当然这里也特指的是AXI4.

地址传输中用到了ID字段,即AWID,BID,ARID,RID,如果传输的是同一件事,那么应该设置同一个ID,ID的宽度是可配置的,具体看实际应用,同一个ID时候,任务会按照传进来的先后顺序传输,AXI是带有缓冲机制的,比如说写入数据,可以先行写到总线矩阵,然后让从机挨个获取,不过前提是有这么一个设计,不同ID还可以区分不同任务,比如像STM32框图中,CPU和DMA同时访问一个外设,他们应用不同ID来区分,这样就由总线仲裁等等决定各种访问.

当然地址信号组里信号那么多,不可能仅仅是传输个地址那么简单,比如写方向的AWLEN,AWSIZE,AWBURST,AWLOCK,AWCACHE,AWPROT,AWQOS,AWREGION,读方向也有同样的信号.

AWLEN/RWLEN 指示要传输的数据长度,在INCR模式可以传输1~256个数据,FIXED模式只能传输1-16个数据,WARP模式只能传输2,4,8,16个数据.因为他的宽度是8,当LEN=0指示传输1个数据.

AWSIZE/ARSIZE指示传输的数据包大小,位宽3位,定义如下.

Table A3-2 Burst size encoding

AxSIZE[2:0]	Bytes in transfer
0b000	1
0b001	2
0b010	4
0b011	8
0b100	16
0b101	32
0b110	64
0b111	128

AWBURST/ARBURST指示传输类型.

Table A3-3 Burst type encoding

AxBURST[1:0]	Burst type
0b00	FIXED
0b01	INCR
0b10	WRAP
0b11	Reserved

固定传输意思就是在一个地址不断写,递增传输指的是写入0x00地址,下次0x04,下一次0x08...,WARP传输是指计算到指定个数后,会到最初地址重新写.

AWLOCK/ARLOCK指示传输是否独占,如果这个外设传输是独占的,那么就设置为1,当你访问这个外设区域时候,其他人就无法访问了.

AWCACHE/ARCACHE是不是很像ARM的MPU一样.

Table A4-5 Memory type encoding

ARCACHE[3:0]	AWCACHE[3:0]	Memory type
0b0000	0b0000	Device Non-bufferable
0b0001	0b0001	Device Bufferable
0b0010	0b0010	Normal Non-cacheable Non-bufferable
0b0011	0b0011	Normal Non-cacheable Bufferable
0b1010	0b0110	Write-Through No-Allocate
0b1110 (0b0110)	0b0110	Write-Through Read-Allocate
0b1010	0b1110 (0b1010)	Write-Through Write-Allocate
0b1110	0b1110	Write-Through Read and Write-Allocate
0b1011	0b0111	Write-Back No-Allocate
0b1111 (0b0111)	0b0111	Write-Back Read-Allocate
0b1011	0b1111 (0b1011)	Write-Back Write-Allocate
0b1111	0b1111	Write-Back Read and Write-Allocate

AWPORT和ARPORT也是类似,区分是否安全访问,是否优先访问,数据还是指令访问.

Table A4-6 Protection encoding

AxPROT	Value	Function
[0]	0	Unprivileged access
	1	Privileged access
[1]	0	Secure access
	1	Non-secure access
[2]	0	Data access
	1	Instruction access

AWQOS/ARQOS指的是优先级,一般来说数值越高越优先,比如STM32H7上就说自己挂的AHB1是带QOS支持的,也就是这个支持.

AWREGION/ARREGION是分区标志,一个分区最小4KB,一个从机也最小4KB,所以一个从机最小可以有1个分区,当然一个从机可以有很多分区和很多个连续4KB组成,这个就是分区标记,比如片上从机有0x0000~0x00FF...0x1000~0x10FF,并且设置分区标志0访问0x0000~0x00FF,地址设置12位0x000~0xFFF(12位是因为4KB),那么实际访问就是0x00~0xFF,如果区域标记是1,那么实际访问就是0x1000~0x10FF.

那么地址阶段能传递的所有信息,就已经标注完了.

数据阶段有一个非常特别的WDATA,没有对应的RDATA,

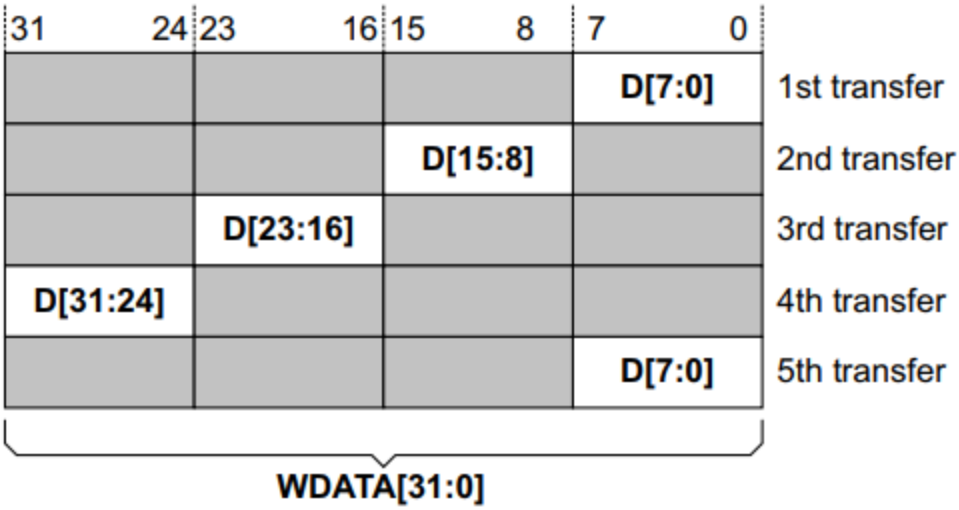


Figure A3-8 Narrow transfer example with 8-bit transfers

比如主机一次只能发送8B,我如何把32B传过去呢,那分批传不就行了,比如WDATA设置为'b0001,那么传递D[7:0],如果设置为'b0010,那么传递D[15:8],如此类推.

那么最后要解决的就是BRESP/RRESP,为什么要这两个信号呢,首先从机应该尽力保证提供数据或者储存主机写入的数据,但是这不是保证,如果出现任何错误都应该用这个位来响应.

1. OKAY 一切安好.
2. EXOKAY 独占访问时,一切安好.
3. SLVERR 下级错误,比如我们从机无法实际执行,储存写入的数据.

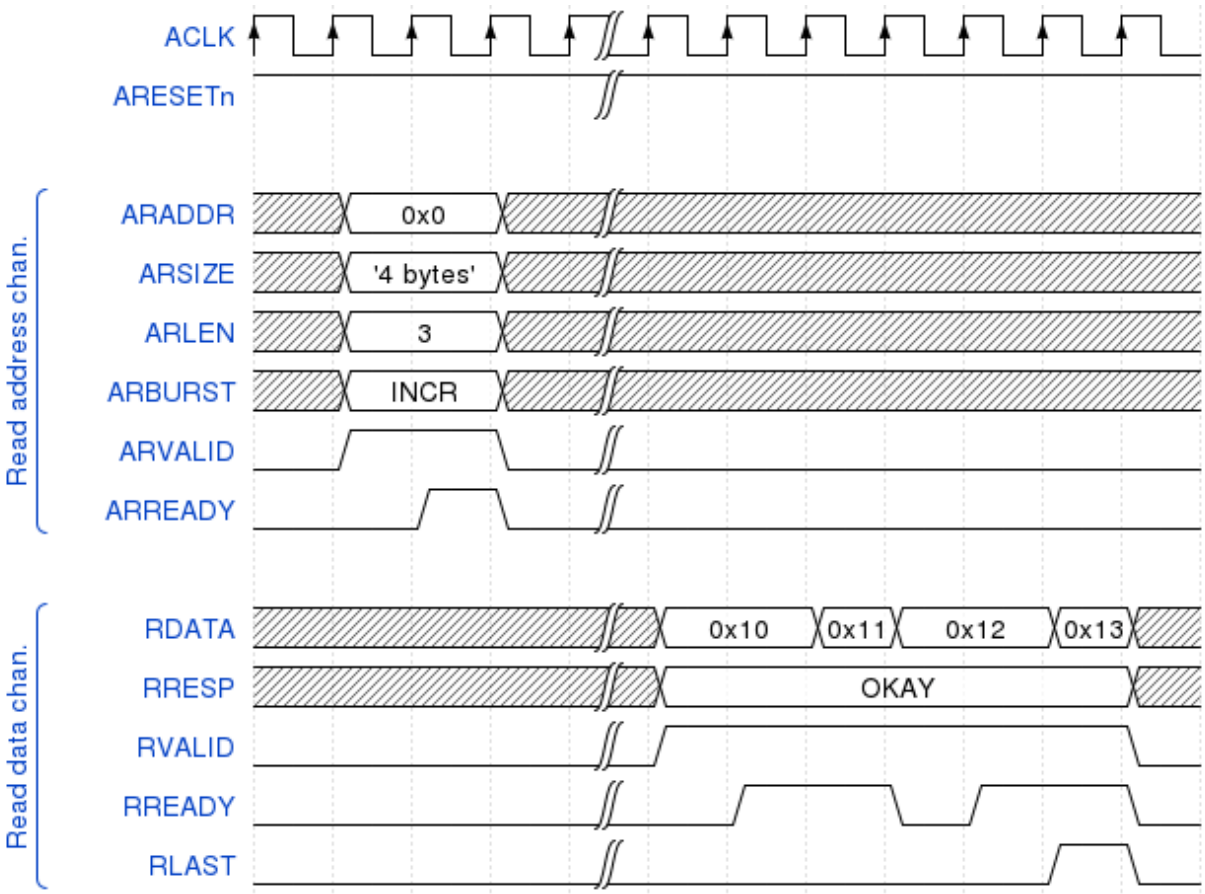
4. DECERR 传说过来的数据,并不能解码,通常是内部使用,所以错误一般用SLVERR就行.

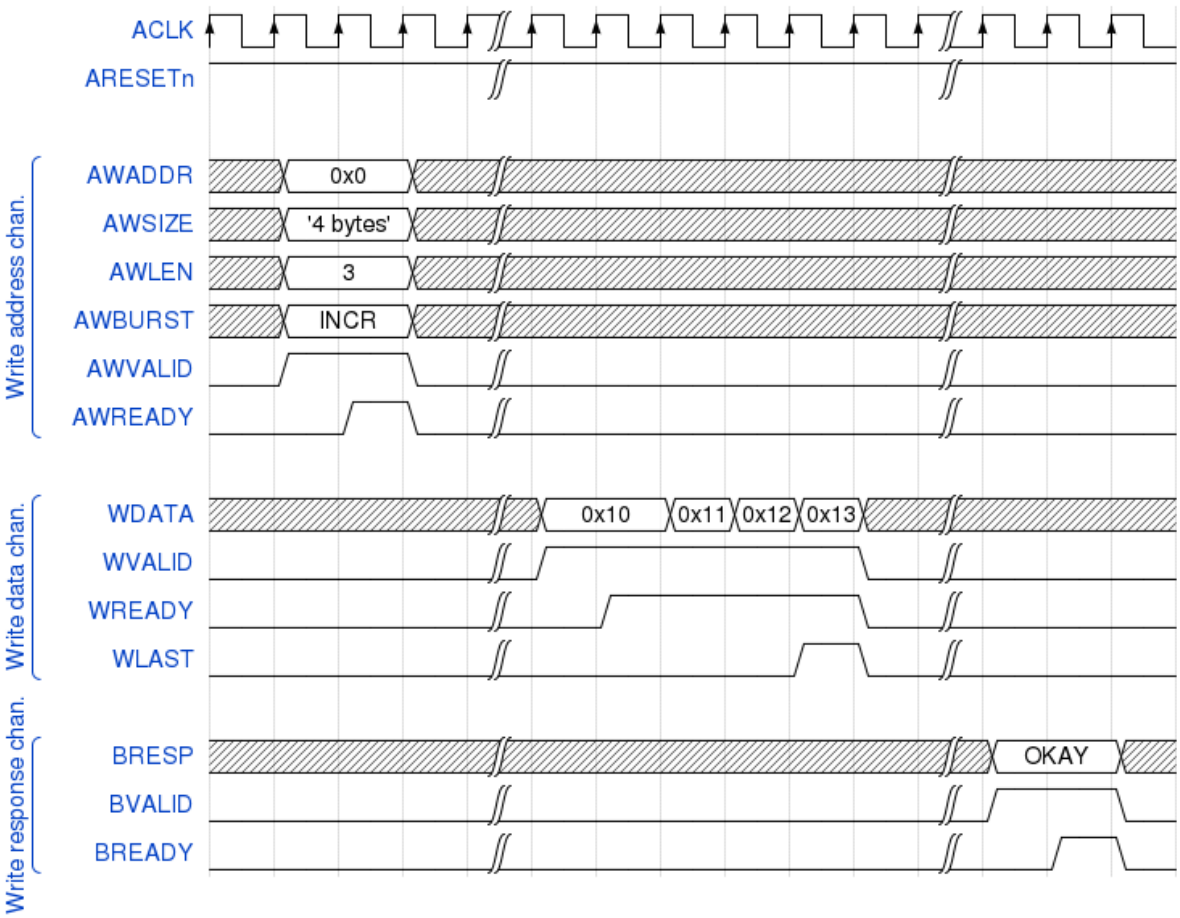
Table A3-5 RRESP and BRESP encoding

RRESP[1:0] BRESP[1:0]	Response
0b00	OKAY
0b01	EXOKAY
0b10	SLVERR
0b11	DECERR

附加的用户自定义数据,总线会原封不动转发他们.

现在附上维基的两个图,就很容易理解了(省略了部分信号).


ACLK
ARESETn
Read address chan.
ARADDR
0x0
ARSIZE
'4 bytes'
ARLEN
3
ARBURST
INCR
ARVALID
ARREADY
Read data chan.
RDATA
0x10
0x11
0x12
0x13
RRESP
OKAY
RVALID
RREADY
RLAST


ACLK
ARESETn
Write address chan.
AWADDR
0x0
AWSIZE
'4 bytes'
AWLEN
3
AWBURST
INCR
AWVALID
AWREADY
Write data chan.
WDATA
0x10
0x11
0x12
0x13
WVALID
WREADY
WLAST
Write response chan.
BRESP
OKAY
BVALID
BREADY