# AXI总线协议分析

AXI接口所涉及的详细引脚介绍如表 2.2所示。

表 2.2 AXI接口引脚介绍表

| 通道 | 名称 | 位宽 | 输入/出 | 说明 |
| --- | --- | --- | --- | --- |
| 读地址 | axi\_arready | 1 | 输入 | 表明“从”可以接收地址和对应的控制信号 |
| axi\_araddr | 32 | 输出 | 读地址，给出一次写突发传输的读地址 |
| axi\_arburst | 2 | 输出 | 突发类型 |
| axi\_arcache | 4 | 输出 | 内存类型，表明一次传输是怎样通过系统的 |
| axi\_arid | 4 | 输出 | 读地址ID，用来标志一组写信号 |
| axi\_arlen | 8 | 输出 | 突发长度，给出突发传输的次数 |
| axi\_arlock | 2 | 输出 | 总线锁信号，可提供操作的原子性 |
| axi\_arport | 3 | 输出 | 保护类型，表明一次传输的特权级及安全等级 |
| axi\_arsize | 3 | 输出 | 突发大小，给出每次突发传输的字节数 |
| axi\_arvaild | 1 | 输出 | 有效信号，表明此通道的地址控制信号有效 |
| 读数据 | axi\_rdata | 32 | 输入 | 读数据 |
| axi\_rid | 4 | 输入 | 读ID tag |
| axi\_rlast | 1 | 输入 | 突发的最后一次传输 |
| axi\_rresp | 2 | 输入 | 读响应，表明读传输的状态 |
| axi\_rvalid | 1 | 输入 | 表明此通道信号有效 |
| axi\_rready | 1 | 输出 | 表明主机能够接收读数据和响应信息 |
| 写地址 | axi\_awready | 1 | 输入 | 表明“从”可以接收地址和对应的控制信号 |
| axi\_awaddr | 32 | 输出 | 写地址，给出一次写突发传输的写地址 |
| axi\_awburst | 2 | 输出 | 突发类型 |
| axi\_awcache | 4 | 输出 | 内存类型，表明一次传输是怎样通过系统的 |
| axi\_awid | 4 | 输出 | 写地址ID，用来标志一组写信号 |
| axi\_awlen | 8 | 输出 | 突发长度，给出突发传输的次数 |
| axi\_awlock | 2 | 输出 | 总线锁信号，可提供操作的原子性 |
| axi\_awport | 3 | 输出 | 保护类型，表明一次传输的特权级及安全等级 |
| axi\_awsize | 3 | 输出 | 突发大小，给出每次突发传输的字节数 |
| axi\_awvalid | 1 | 输出 | 有效信号，表明此通道的地址控制信号有效 |
| 写数据 | axi\_wready | 1 | 输入 | 表明从机可以接收写数据 |
| axi\_wdata | 32 | 输出 | 写数据 |
| axi\_wlast | 1 | 输出 | 表明此次传输是最后一个突发传输 |
| axi\_wstrb | 4 | 输出 | 写数据有效的字节线，用来表明哪8bits数据是有效的 |
| axi\_wvalid | 1 | 输出 | 写有效，表明此次写有效 |
| 写响应 | axi\_bid | 4 | 输入 | 写响应的ID tag |
| axi\_bresp | 2 | 输入 | 写响应，表明写传输的状态 |
| axi\_bvalid | 1 | 输入 | 写响应有效 |
| axi\_bready | 1 | 输出 | 表明主机能够接收响应 |

# AXI总线接口设计

* 仲裁

采用固定优先级的仲裁方法，原本应该是存在有无cache和指令数据四种请求，但是由于有无在请求阶段以及进行过判断，所以在AXI总线上主要负责针对cache制定的仲裁机制，而不需要考虑uncache的情况。

在接口模块中主要考虑的情况则是指令cache和数据cache 的同时读以及指令cache的读和数据cache的写，数据cache优先执行，同时由于AXI协议是五通道读写独立传输，而指令存放的ram不会发生修改，因此可以实现指令cache读和数据cache写同时进行的处理。

* 读状态机

依据读操作涉及的通道信号依赖关系搭建状态机，如图 2.10所示。

State0：读地址握手阶段

等待地址握手，当产生读请求时，将axi\_arvalid置为1表示读请求有效，等待axi\_arready信号，读地址握手成功则撤销axi\_arvalid信号，并切换状态至State1，并将axi\_rready置为1，表示读数据就绪。

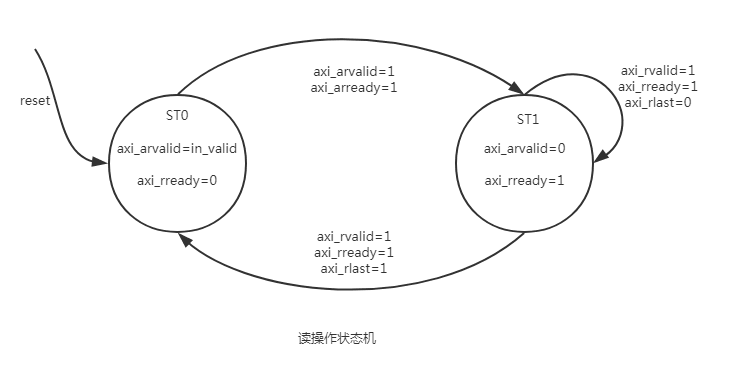


图 2.10 AXI总线读状态机

State1：读数据传输阶段

等待数据传输过程中的数据有效信号axi\_rvalid，读数据握手成功则发送一次cache写入信号，将读取数据载入cache，并在最后一次传输axi\_rlast后切换状态至State0，并将仲裁锁信号撤销。

* 写状态机

依据写操作涉及的三通道信号依赖关系搭建状态机如图 2.11所示。

State0：地址握手阶段

等待地址握手，当产生写请求时，将axi\_awvalid置为1,表示写请求有效，等待axi\_awready信号，写地址握手成功则撤销axi\_awvalid信号，同时切换状态置State1。

State1：写数据阶段

等待cache提供的写数据，每写一次数据，将axi\_wvalid置为1表示当前数据有效，等待axi\_wready信号，写数据握手成功后，向cache发送进行下一次数据写操作的信号，直到接收到cache发送的最后一次写请求wlast，最后一次握手成功后切换状态至State2，同时提前提醒cache撤销stall停机信号，但并不释放总线，将axi\_bready置为1，由总线接口等待写响应。

State2：写响应阶段

等待写响应，等待axi\_bvalid信号，响应握手成功后才能撤销仲裁的写锁，进而准备等待下一次写操作的仲裁。

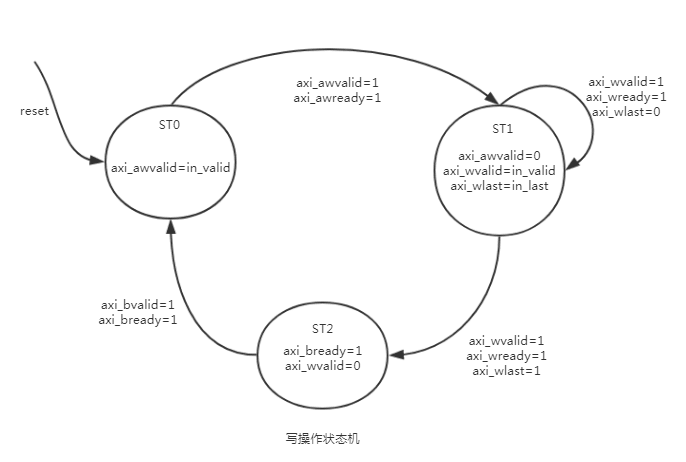


图 2.11 AXI总线写操作状态机

# AXI接口测试遇到的问题：

* 搭建框架过程中通过Verilog代码进行crossbar实例化接口AXI协议不同会有差异，需要检查各项参数，其中ID宽度参数设置为4，否则会导致接口不对齐
* 此处选择的axi协议为AXI3，实际上用到的参数len仅有低四位，但实际len位宽为八位，其中高四位注意不能悬空，否则会仿真出错。