# LoongArch CPU 设计实战

## 第3天：片上总线与异常处理

教师：杨建磊、万 寒
助教：苏阳、周振源
北京航空航天大学 计算机学院

2025年7月30日 四川 成都

# Day3 目录

# 1.1 计算机总线接口

- 一台完整的计算机包含什么?
  - 主机
  - 显示器、打印机
  - 鼠标、键盘
  - 内存条、U盘、移动硬盘
  - 网卡、显卡
  - ……

所有的外设都会以某种方式连接到主机上，那么：

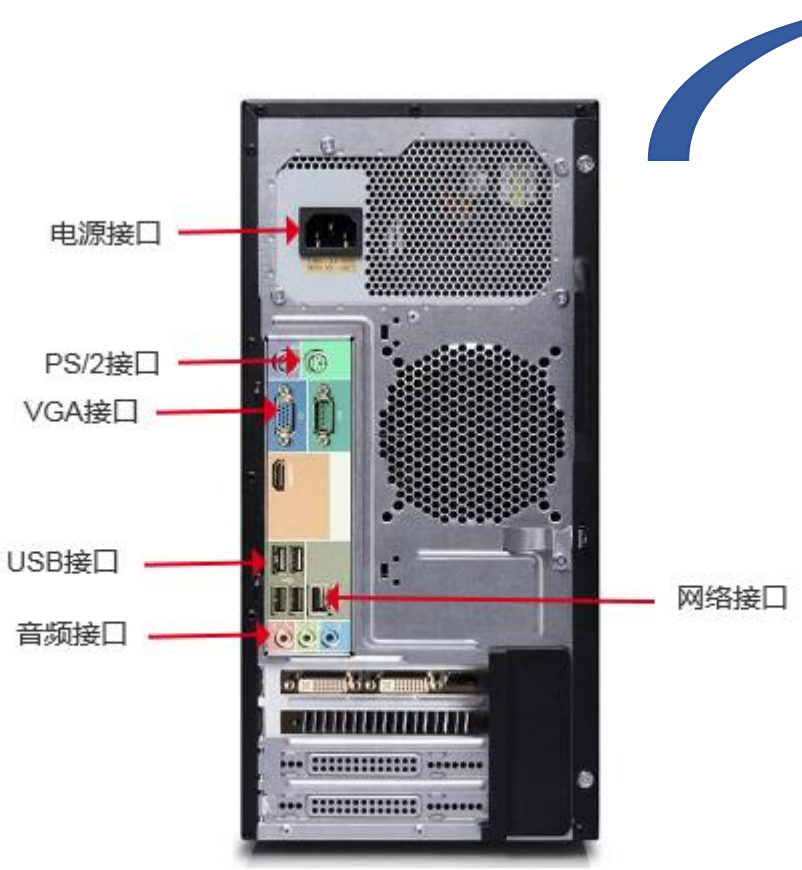
——

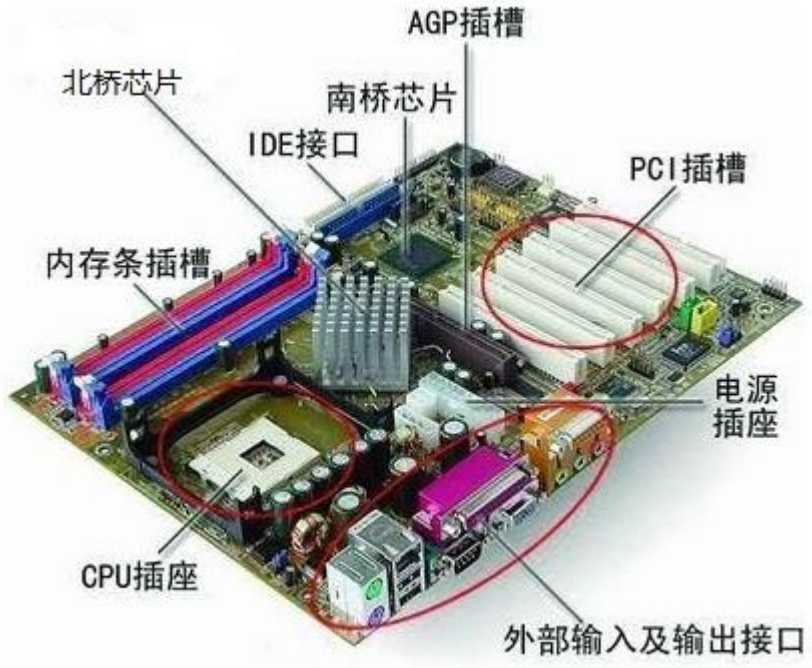
主机内部是什么?

主机是如何协同其他外设的?

# 1.1 计算机总线接口

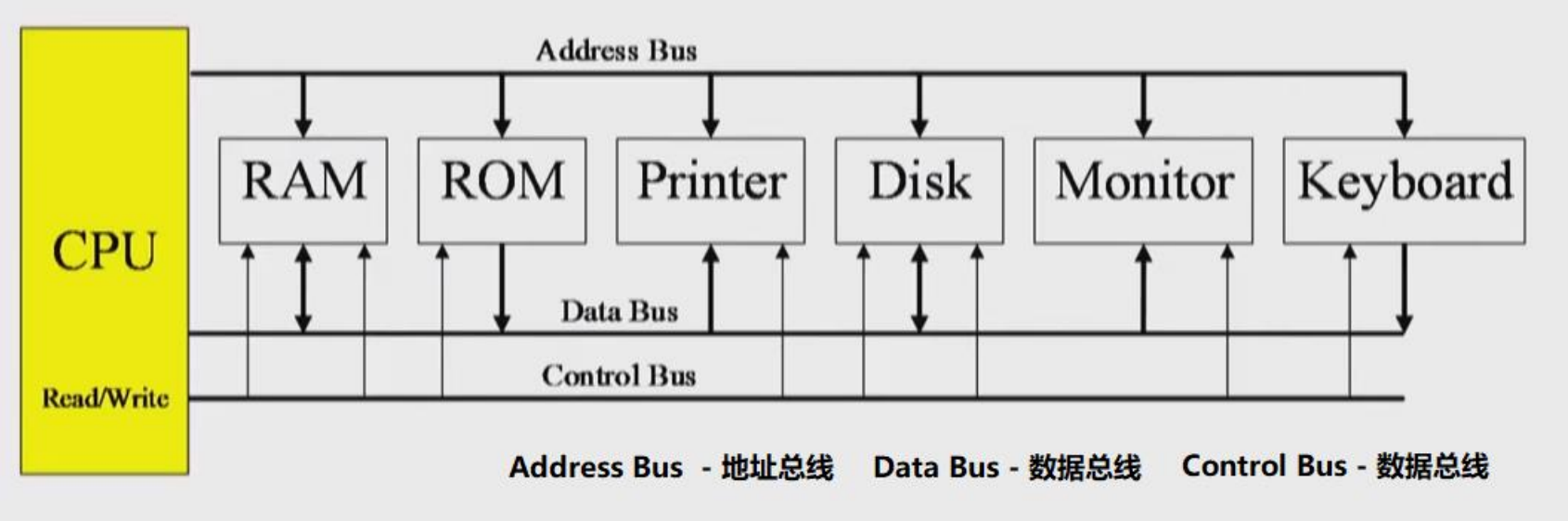
- 电源接口、USB接口、HDMI接口、VGA接口
- 各种各样的外设以各种各样的方式连接到主机（主板）
- CPU会以某种方式和连接的外设通信，什么方式呢?



# 1.1 计算机总线接口

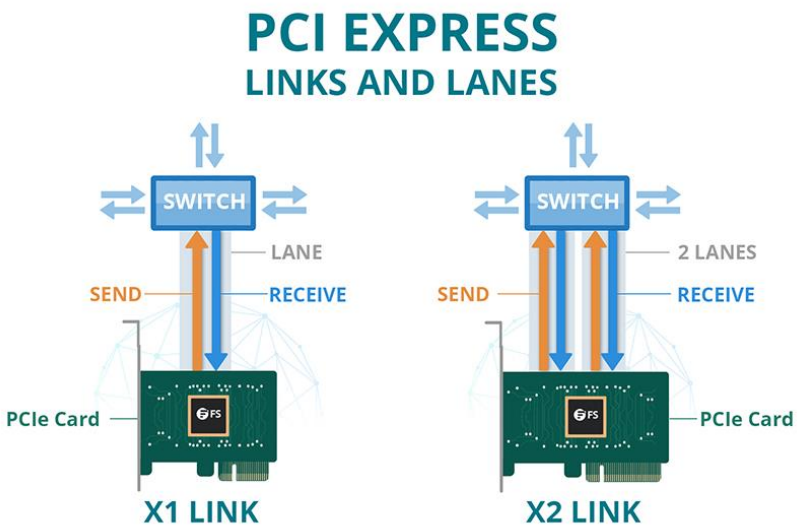
- 定义：连接多个部件的传输线
  - 数据、地址、控制
- 总线的关键特征：多设备共享的传输线
  - 主设备 master：任意时刻只能有一个设备向总线发送信息
  - 从设备 slave：多个部件可以同时从总线接受相同的信息，广播式
- 总线拓扑：单总线、多总线

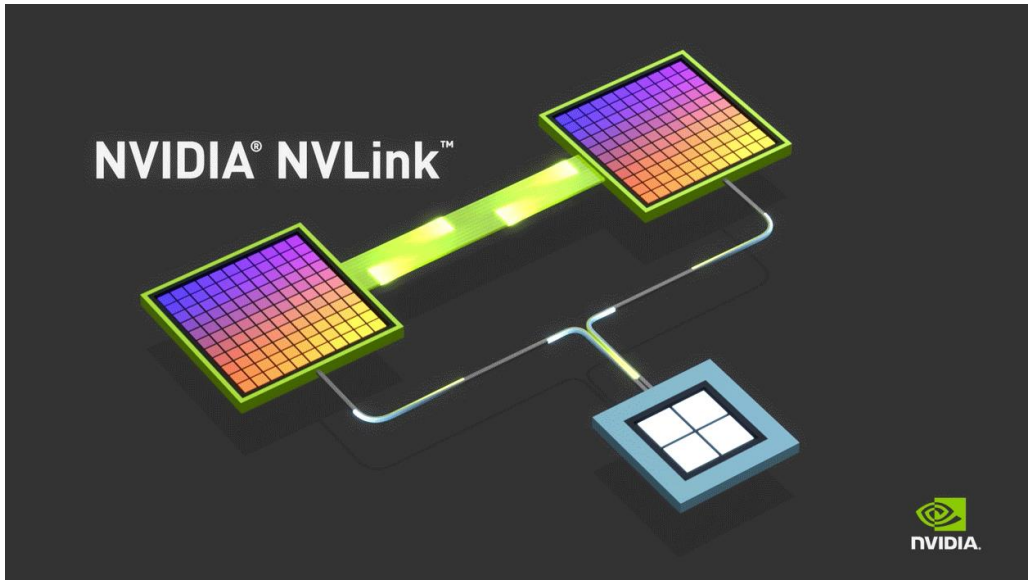


# 1.1 计算机总线接口

## 总线类型

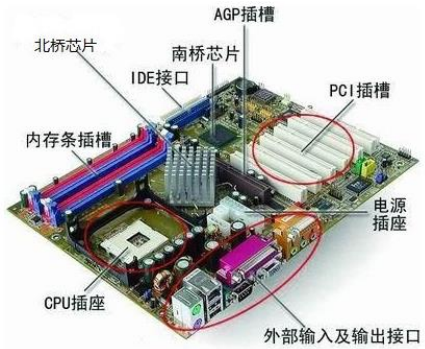
- 片内总线：指芯片内部用来连接不同模块的总线
- 片外总线：指芯片之间、板级之间主板上不同器件之间的通讯总线
- 显卡互联总线：主要指GPU与主板/CPU/其他GPU之间互联所用的高速总线



# 1.1 计算机总线接口

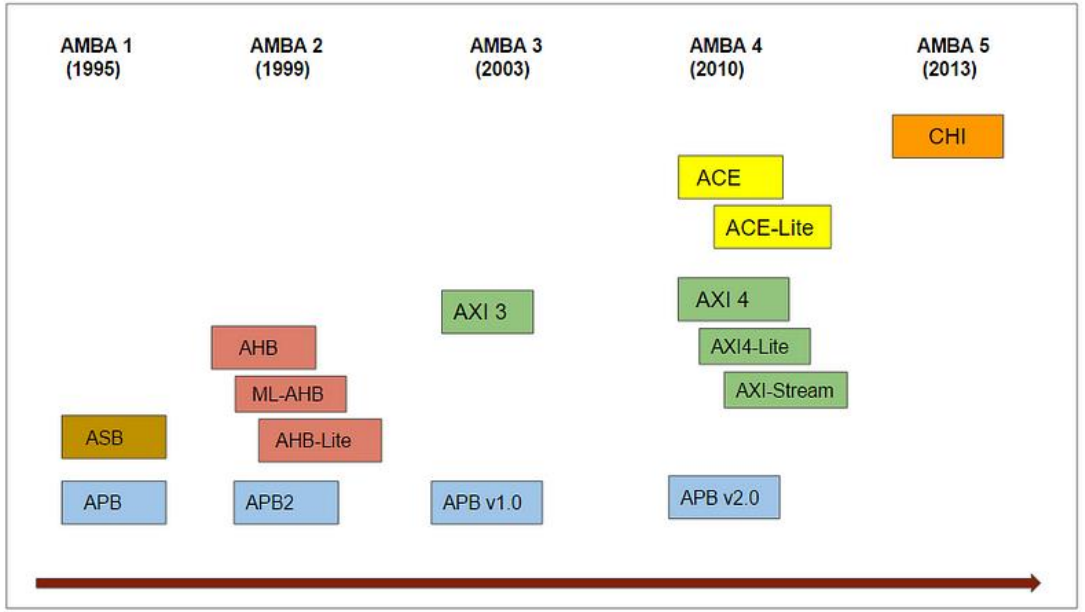
## □ SoC中的总线



# 1.2 AXI 总线

## AXI3总线—ARM AMBA3.0的核心

- 高性能、高带宽、低延迟的片内总线
- 能满足大部分器件的接口要求
- 适合高初始延时的存储控制器
- 向下兼容已有的 AHB 和 APB 接口
- 地址/控制阶段与数据阶段分离
- 使用基于突发（Burst）的事务，仅发布初始地址
- 支持无序事务完成
- AMBA4.0 将其修改升级为 AXI4 总线



# 1.2 AXI 总线

- AXI有五个独立的通道
  - 写地址通道（Write Address，AW）
  - 写数据通道（Write Data，W）
  - 写响应通道（Write Response，B）
  - 读地址通道（Read Address，AR）
  - 读数据通道（Read Data，R）

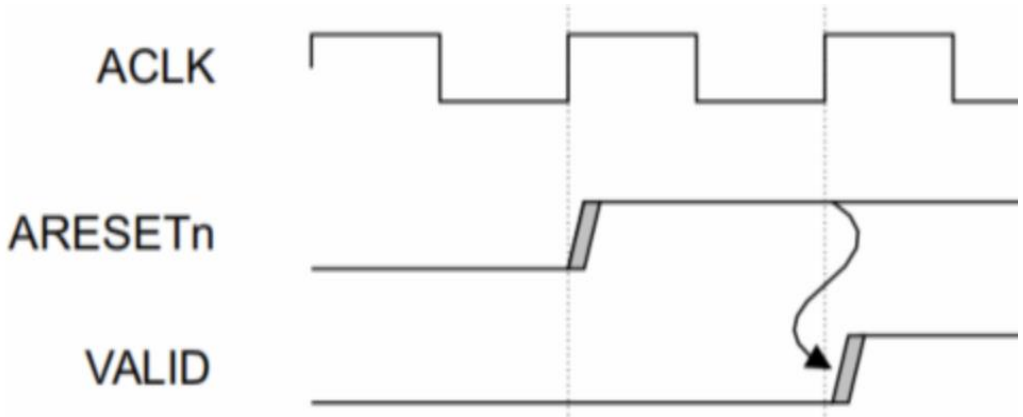
# 1.2 AXI 总线

- 写操作过程
  - 主机发送写地址/控制信号
  - 主机发送要写入的数据，即写数据信号
  - 从机发送写响应信号
- 读操作过程
  - 主机发送读地址/控制信号
  - 从机发送读地址信号

# 1.2 AXI 总线

## 全局信号

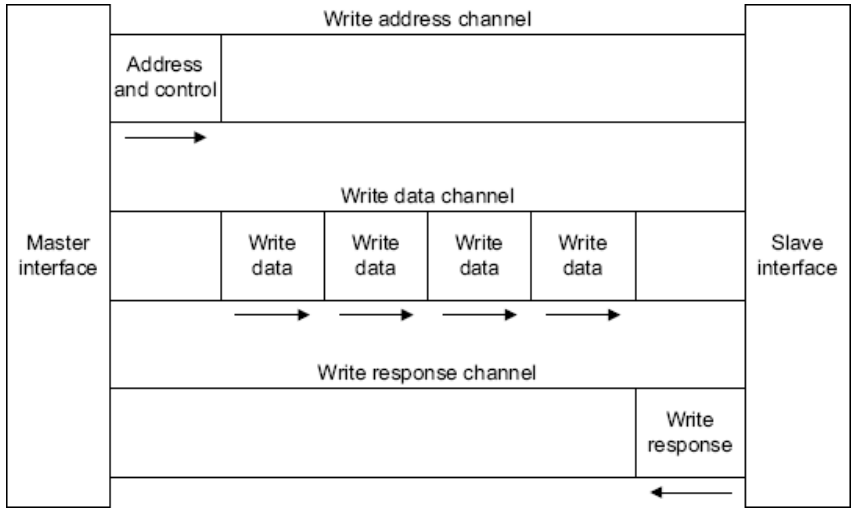
- ACLK：全局时钟信号，所有输入信号在 ACLK 上升沿采样。
- ARESETn：全局复位信号，异步复位同步置位。
  - 协议规定，ARESETn 低电平有效
  - 主机必须驱动 ARVALID，AWVALID，WVALID 为低电平
  - 从机必须驱动 BVALID 和 RVALID 为低电平
  - 其他信号能被驱动为任意值



# 1.2 AXI 总线

## □ 写地址通道信号（AW）

| 信号 | 源 | 必需/可选 | 位宽 | 默认值 | 描述 |
|---|---|---|---|---|---|
| AWID | 主 | 可选 | 可变，协议未定义位宽，一般为4 | 全0 | 写地址识别ID |
| AWADDR | 主 | 必需 | 可变，协议未定义位宽，一般为32/64 | - | 待写入数据的内存地址 |
| AWLEN | 主 | 可选 | 8（AXI4）<br>4（AXI3） | 全0 | 数据是按份传输的，此信号表示接下来要写入的数据量<br>实际传输数据量=AWLEN+1 |
| AWSIZE | 主 | 可选 | 3 | 数据总线宽度 | 表示每份数据占的字节数，n表示2^n |
| AWBURST | 主 | 可选 | 2 | 2’b01<br>INCR | 突发类型，表示在写事务中每次传输之间地址如何变化 |
| AWLOCK | 主 | 可选 | 1（AXI4）<br>2（AXI3） | 全0 | 0表示 Normal Access，表示主机访问从机是否是独占的 |
| AWCACHE | 主 | 可选 | 4 | 全0 | 表示不同的内存类型 |

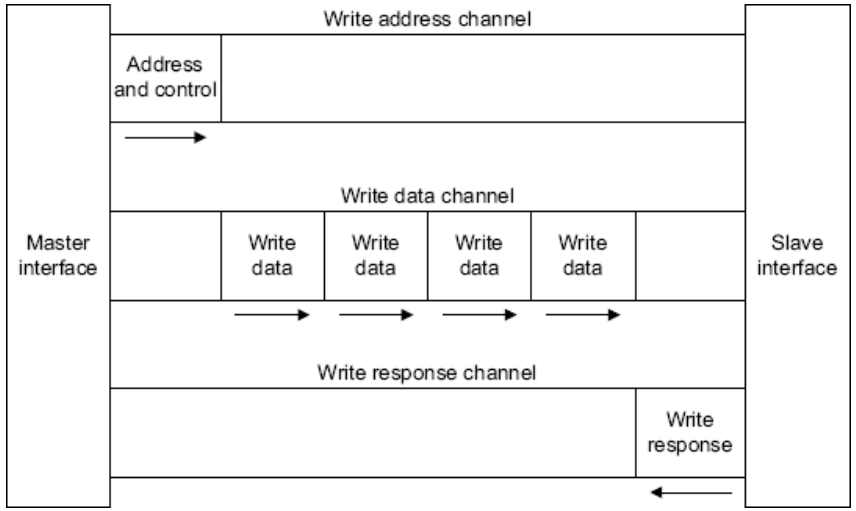



参考：https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi

# 1.2 AXI 总线

## □ 写地址通道信号（AW）

| 信号 | 源 | 必需/可选 | 位宽 | 默认值 | 描述 |
|---|---|---|---|---|---|
| AWPORT | 主 | 必需 | 3 | 全0 | 写事务的保护属性：特权，安全级别和访问类型 |
| AWQOS | 主 | 可选 | 4 | 全0 | 服务质量标识符，AXI3 中未实现。AXI4 中未指定 AxQOS 的确切用途，但建议用作读/写优先级指示符，QOS 越大优先级越高 |
| AWREGION | 主 | 可选 | 4 | 全0 | 写入事务的区域指示器，AXI3 未实现 |
| AWUSER | 主 | 可选 | 可变，协议未定义位宽 | 全0 | AXI4 一般不建议使用用户字段，AXI3 中未实现 |
| AWVALID | 主 | 必需 | 1 | - | valid |
| AWREADY | 从 | 必需 | 1 | - | ready |




参考：https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi

# 1.2 AXI 总线

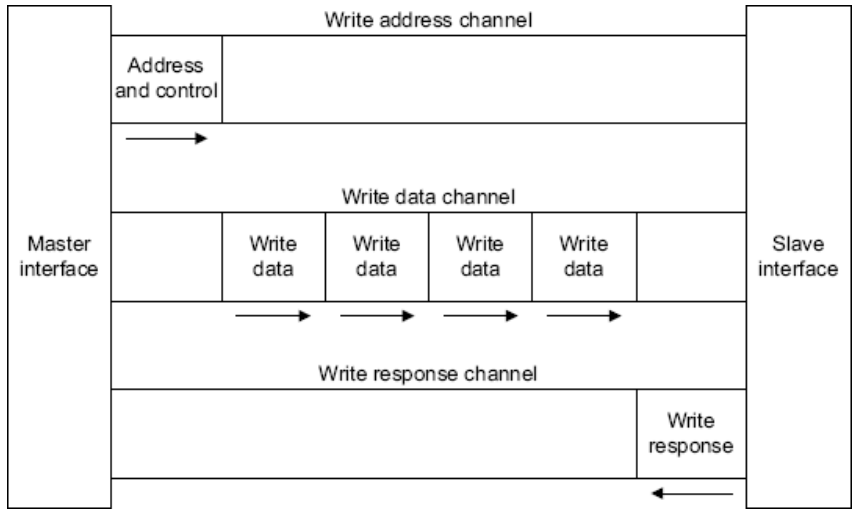
## AxBURST

| 值 | 名 | 描述 |
| --- | --- | --- |
| 2’b00 | FIXED | 在固定突发中，<br>1. 每次传输的地址都相同<br>2. 在每份数据的有效字节通道是固定的，但与已声明 WSTRB 的实际字节可能不同。<br>此突发类型用于重复访问同一位置，例如在加载或清空 FIFO 时。 |
| 2’b01 | INCR | 递增。在递增突发中，突发中每次传输的地址都是前一次传输的地址的增量。增量值取决于传输的大小。此突发类型用于访问常规顺序存储器。 |
| 2’b10 | WRAP | 包装。包装突发与增量突发类似，不同之处在于，如果达到地址上限，地址会绕到一个较低的地址。以下限制适用于包装突发，<br>1. 起始地址必须与每次传输的大小对齐。<br>2. 突发长度必须为2、4、8或16，即 AWLEN 必须为1、3、7或15。 |
| 2’b11 | 保留 | - |

# 1.2 AXI 总线

## □ 写数据通道信号（W）

| 信号 | 源 | 必需/可选 | 位宽 | 默认值 | 描述 |
|---|---|---|---|---|---|
| WID | 主 | 可选 | 可变，协议未定义位宽 | - | 写地址识别ID，仅在AXI3中实现。通过AWID，WID和BID一致来对应地址和数据，故AXI3可支持乱序传输，而AXI4只能顺序传输，数据紧跟地址，或地址紧跟数据。 |
| WDATA | 主 | 必需 | 与 AWSIZE 指定的数据位宽保持一致，2^AWSIZE | - | 要写入的数据。 |
| WSTRB | 主 | 可选 | 2^AWSIZE | 全0 | 表示哪些字节通道保存有效数据，WSTRB为高表示数据总线的对应的字节是有效数据。 |
| WLAST | 主 | 必需 | 1 | - | 表示写事务是否为最后一次数据传输，高电平代表此时刻的数据是最后一份。 |
| WUSER | 主 | 可选 | 可变，协议未定义位宽 | - | 一般不建议使用用户字段，AXI3中未实现。 |
| WVALID | 主 | 必需 | 1 | - | valid |
| WREADY | 从 | 必需 | 1 | - | ready |

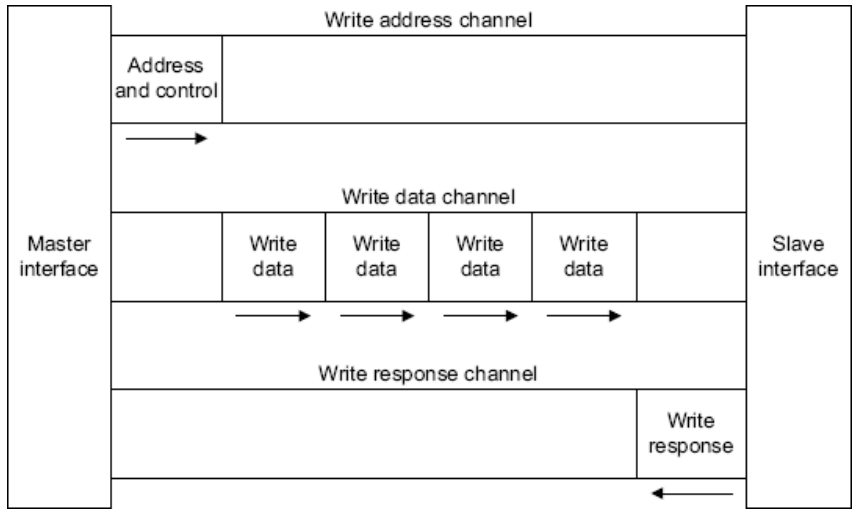



参考：https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi

# 1.2 AXI 总线

## □ 写响应通道信号（B）

| 信号 | 源 | 必需/可选 | 位宽 | 默认值 | 描述 |
|---|---|---|---|---|---|
| BID | 主 | 可选 | 可变，协议未定义位宽 | - | 写响应识别ID，与AWID保持一致。 |
| BRESP | 主 | 必需 | 2 | 全0 | 写响应，表示写事务状态。详细用法参见后表。 |
| BUSER | 主 | 可选 | 可变，协议未定义位宽 | 全0 | 一般不建议使用用户字段，AXI3中未实现。 |
| BVALID | 主 | 必需 | 1 | - | valid |
| BREADY | 从 | 必需 | 1 | - | ready |

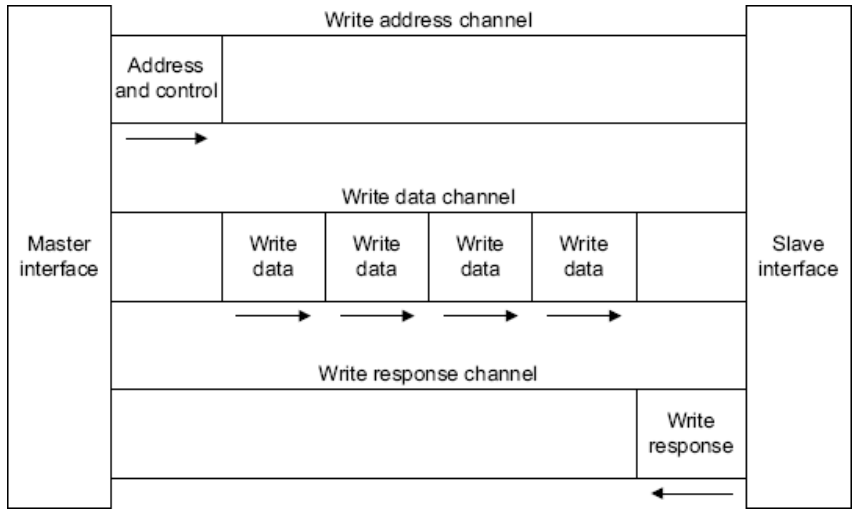



参考：https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi

# 1.2 AXI 总线

## □ 写响应通道信号（B）

| 信号 | 源 | 必需/可选 | 位宽 | 默认值 | 描述 |
|---|---|---|---|---|---|
| BID | 从 | 可选 | 可变，协议未定义位宽 | - | 写响应识别ID，与AWID保持一致。 |
| BRESP | 从 | 必需 | 2 | 全0 | 写响应，表示写事务状态。详细用法参见后表。 |
| BUSER | 从 | 可选 | 可变，协议未定义位宽 | 全0 | 一般不建议使用用户字段，AXI3中未实现。 |
| BVALID | 从 | 必需 | 1 | - | valid |
| BREADY | 主 | 必需 | 1 | - | ready |

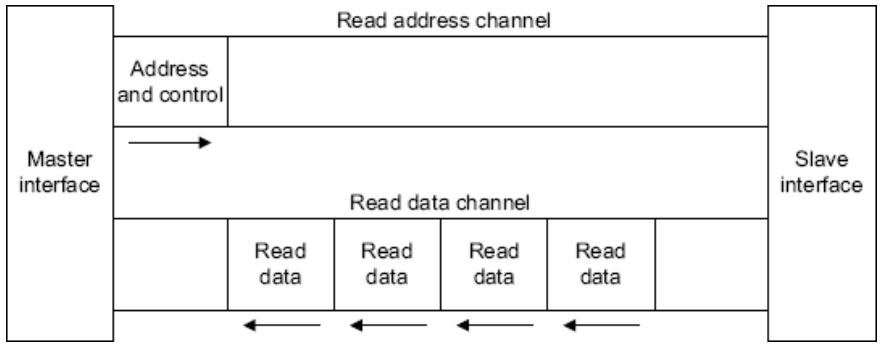



参考：
https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi

# 1.2 AXI 总线

## □ 读地址通道信号（AR）

| 信号 | 源 | 必需/可选 | 位宽 | 默认值 | 描述 |
|---|---|---|---|---|---|
| ARID | 主 | 可选 | 可变，协议未定义位宽，一般为4 | 全0 | 读地址识别ID |
| ARADDR | 主 | 必需 | 可变，协议未定义位宽，一般为32/64 | - | 读数据的内存地址 |
| ARLEN | 主 | 可选 | 8（AXI4）<br>4（AXI3） | 全0 | 数据是按份传输的，此信号表示接下来要读取的数据量<br>实际传输数据量=ARLEN+1 |
| ARSIZE | 主 | 可选 | 3 | 数据总线宽度 | 表示每份数据占的字节数，n表示2^n |
| ARBURST | 主 | 可选 | 2 | 2’b01<br>INCR | 突发类型，表示在读事务中每次传输之间地址如何变化 |
| ARLOCK | 主 | 可选 | 1（AXI4）<br>2（AXI3） | 全0 | 0表示 Normal Access，表示主机访问从机是否是独占的 |
| ARCACHE | 主 | 可选 | 4 | 全0 | 表示不同的内存类型 |

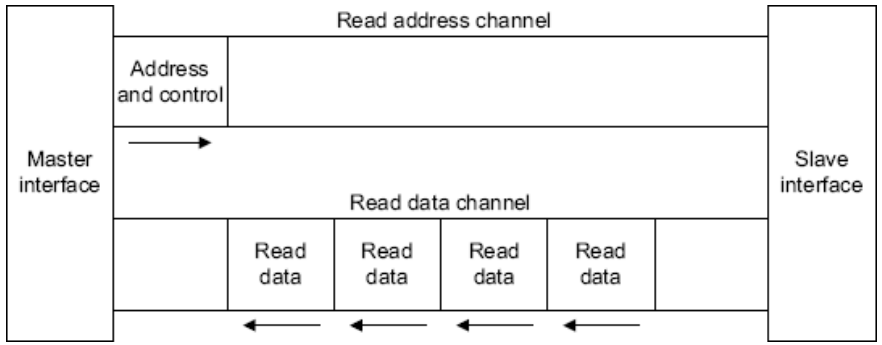



参考：
https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi

# 1.2 AXI 总线

## □ 读地址通道信号（AR）

| 信号 | 源 | 必需/可选 | 位宽 | 默认值 | 描述 |
|---|---|---|---|---|---|
| ARPORT | 主 | 必需 | 3 | 全0 | 读事务的保护属性：特权，安全级别和访问类型 |
| ARQOS | 主 | 可选 | 4 | 全0 | 服务质量标识符，AXI3 中未实现。AXI4 中未指定 AxQOS 的确切用途，但建议用作读/写优先级指示符，QOS 越大优先级越高 |
| ARREGION | 主 | 可选 | 4 | 全0 | 读取事务的区域指示器，AXI3 未实现 |
| ARUSER | 主 | 可选 | 可变，协议未定义位宽 | 全0 | AXI4 一般不建议使用用户字段，AXI3 中未实现 |
| ARVALID | 主 | 必需 | 1 | - | valid |
| ARREADY | 从 | 必需 | 1 | - | ready |

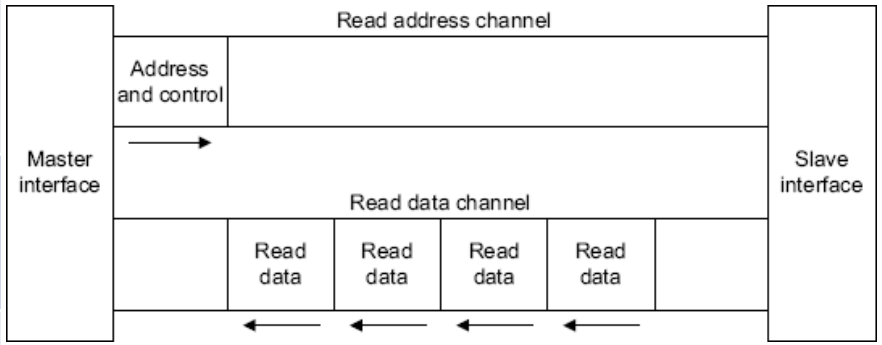



参考：https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi

# 1.2 AXI 总线

## □ 读数据通道信号（R）

| 信号 | 源 | 必需/可选 | 位宽 | 默认值 | 描述 |
|---|---|---|---|---|---|
| RID | 从 | 可选 | 可变，协议未定义位宽 | - | 读数据识别ID，与ARID保持一致用来对应地址和数据。 |
| RDATA | 从 | 必需 | 可变，协议未定义位宽 | - | 读数据。 |
| RRESP | 从 | 可选 | 2 | 全0 | 读响应，表示读事务状态。 |
| RLAST | 从 | 必需 | 1 | - | 表示读事务是否为最后一次数据传输，高电平代表此时刻的数据是最后一份。 |
| RUSER | 从 | 可选 | 可变，协议未定义位宽 | 全0 | 一般不建议使用用户字段，AXI3中未实现。 |
| RVALID | 从 | 必需 | 1 | - | valid |
| RREADY | 主 | 必需 | 1 | - | ready |

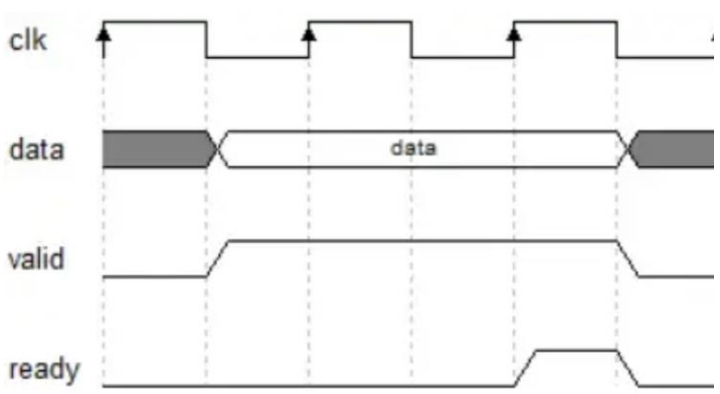


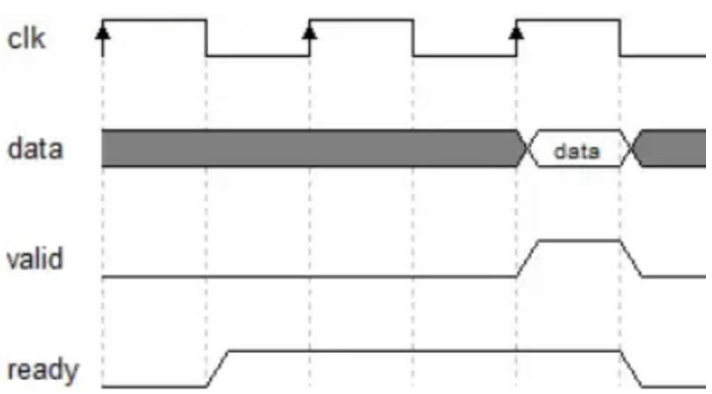

参考：https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi

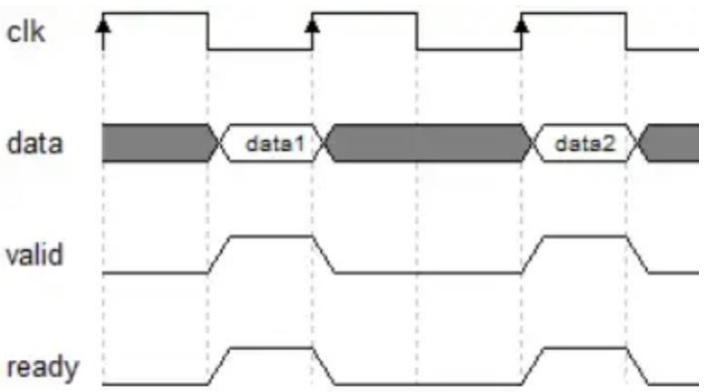
# 1.2 AXI 总线

## 握手信号

- AXI总线的五个通道都通过VALID和READY进行握手
- VALID由数据发送端驱动，VALID置高电平表示发送数据有效，接收端可以接收
- READY由数据接收端驱动，READY置高电平表示接收端已准备好接收数据
- 为避免出现死锁的情况，VALID与READY需要满足以下条件：
  - VALID信号不得依赖于READY信号
  - READY信号可以依赖VALID信号



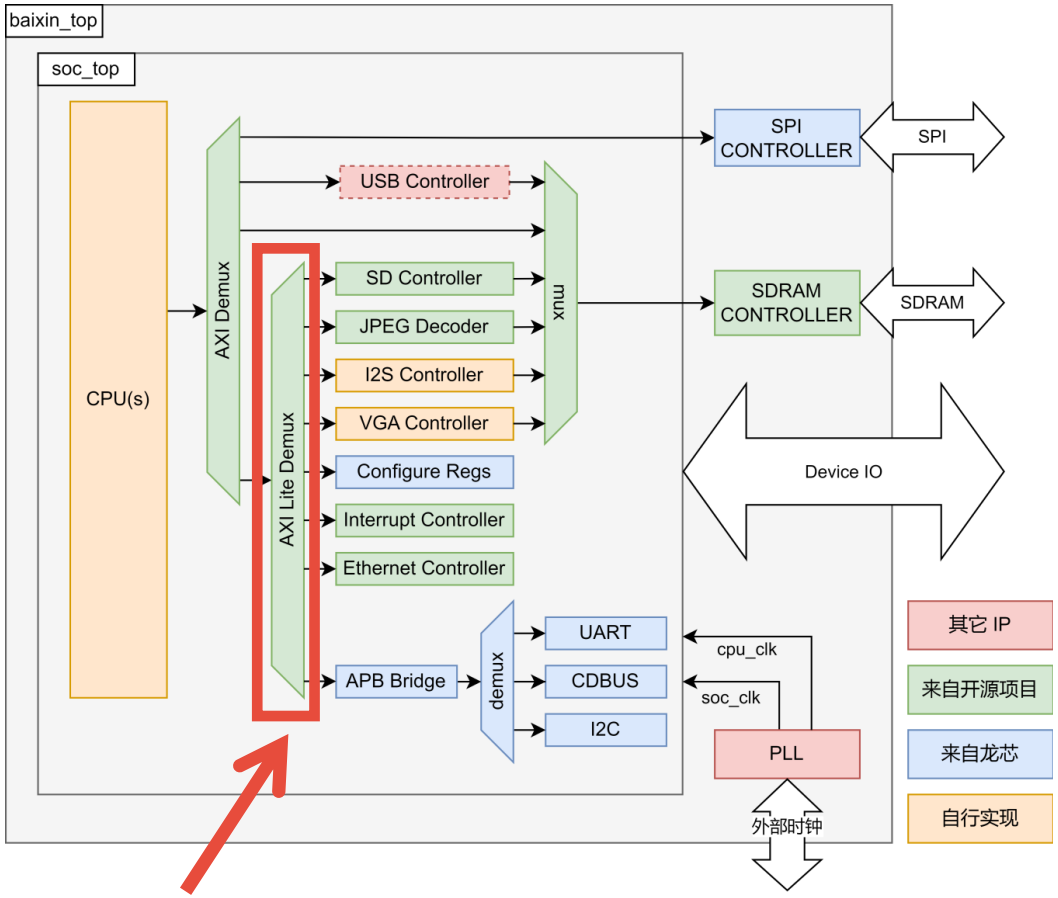



# 1.3 AXI-Lite 总线

- AXI4-Lite 是 AXI4 总线协议的精简版
  - 适用于不需要 AXI4 完整功能的简单控制寄存器的接口
  - 所有读写事务的突发长度均为1
  - 仅支持32/64位数据总线宽度
  - 访问都是不可修改的，不可缓冲的
  - 不支持独占访问

参考：https://github.com/BUAA-CI-LAB/MegaSoC/tree/master/General/pulp-axi/src/axi_to_axi_lite.sv

| Global | Write address channel | Write data channel | Write response channel | Read address channel | Read data channel |
|---|---|---|---|---|---|
| ACLK | AWVALID | WVALID | BVALID | ARVALID | RVALID |
| ARESETn | AWREADY | WREADY | BREADY | ARREADY | RREADY |
| – | AWADDR | WDATA | BRESP | ARADDR | RDATA |
| – | AWPROT | WSTRB | – | ARPROT | RRESP |




AXI<-->AXI Lite

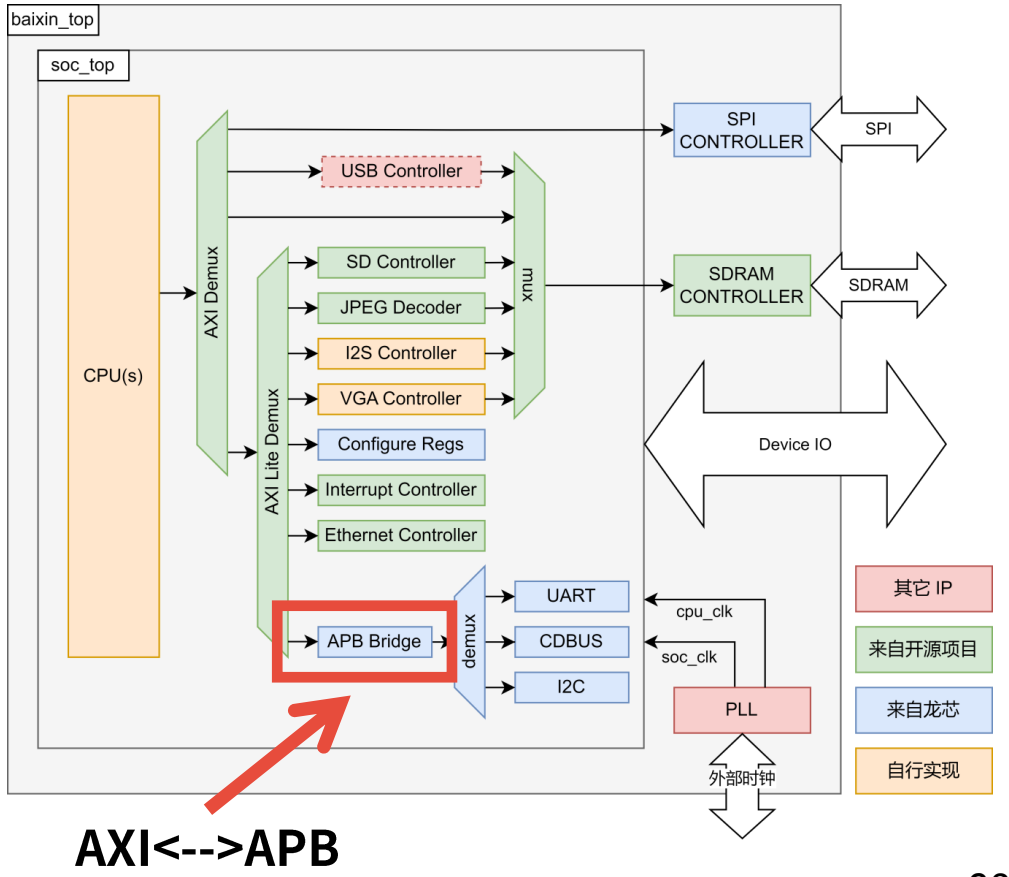
# 1.4 APB 总线

## □ AMBA 总线协议之一，最基本的总线协议

- 主要应用于低带宽、低功耗的周边外设连接，如：I2C、UART等

参考：MegaSoC/General/pulp-axi/src/my_axi_lite_to_apb.sv

| 信号名 | 源 | 说明 |
|---|---|---|
| PCLK | 时钟源 | 系统时钟 |
| PRESETn | 复位源 | 复位信号，低使能 |
| PADDR | APB桥 | 地址信号 |
| PPORT | APB桥 | 保持类型，详见手册 |
| PSELx | APB桥 | 片选信号，表示x从机是否被选中 |
| PENABLE | APB桥 | 使能信号 |
| PWRITE | APB桥 | 读/写控制信号，高电平写，低电平读 |
| PWDATA | APB桥 | 写数据 |
| PSTRB | APB桥 | 写选通，表示哪个字节是有效的 |



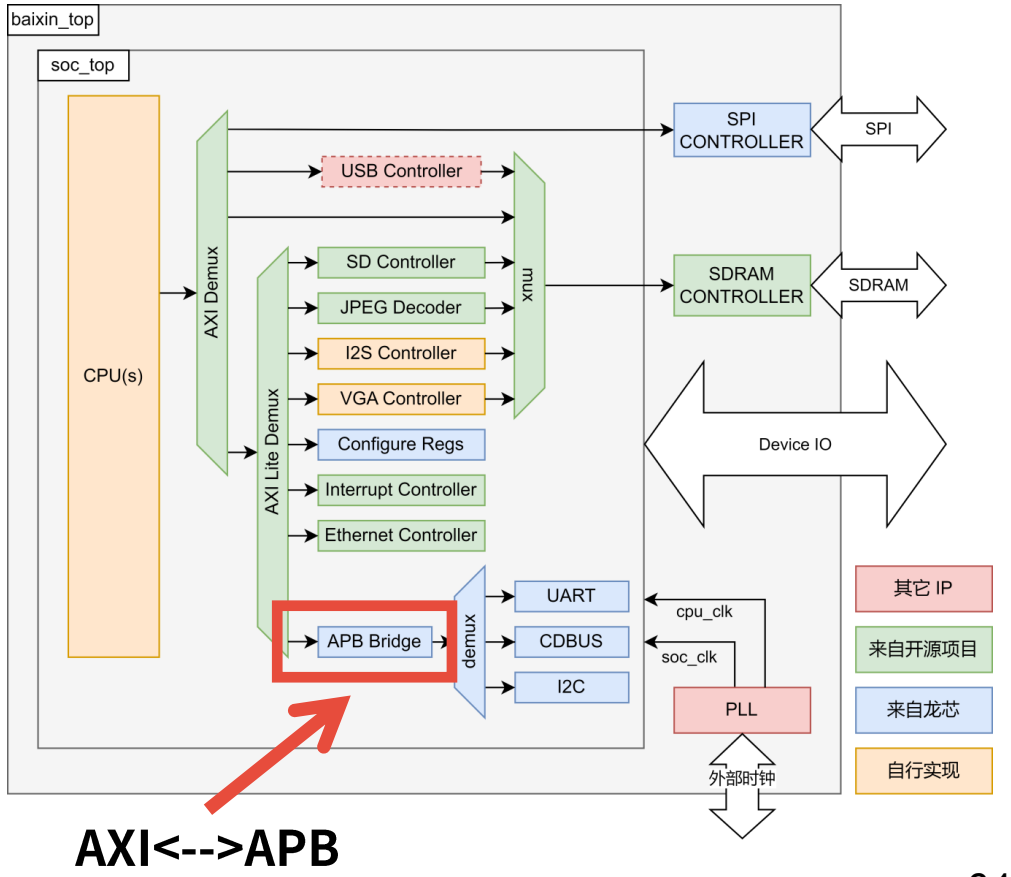
# 1.4 APB 总线

## □ AMBA 总线协议之一，最基本的总线协议

- 主要应用于低带宽、低功耗的周边外设连接，如：I2C、UART等

参考：MegaSoC/General/pulp-axi/src/my_axi_lite_to_apb.sv

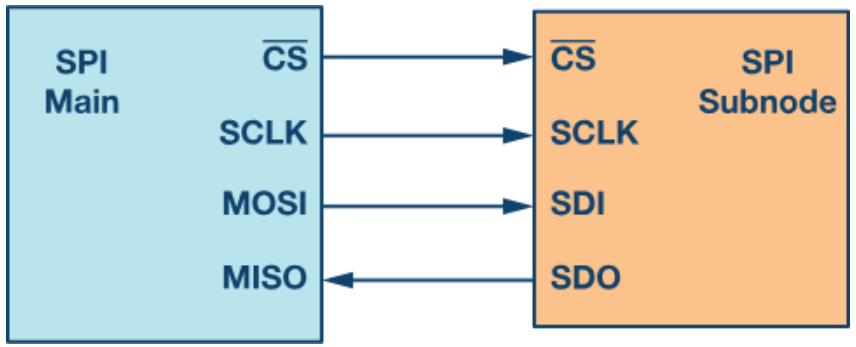
| 信号名 | 源 | 说明 |
|---|---|---|
| PREADY | 从 | 标记从机是否已将数据发送到总线 |
| PRDATA | 从 | 读数据 |
| PSLVERR | 从 | 故障信号，高电平位发生故障 |



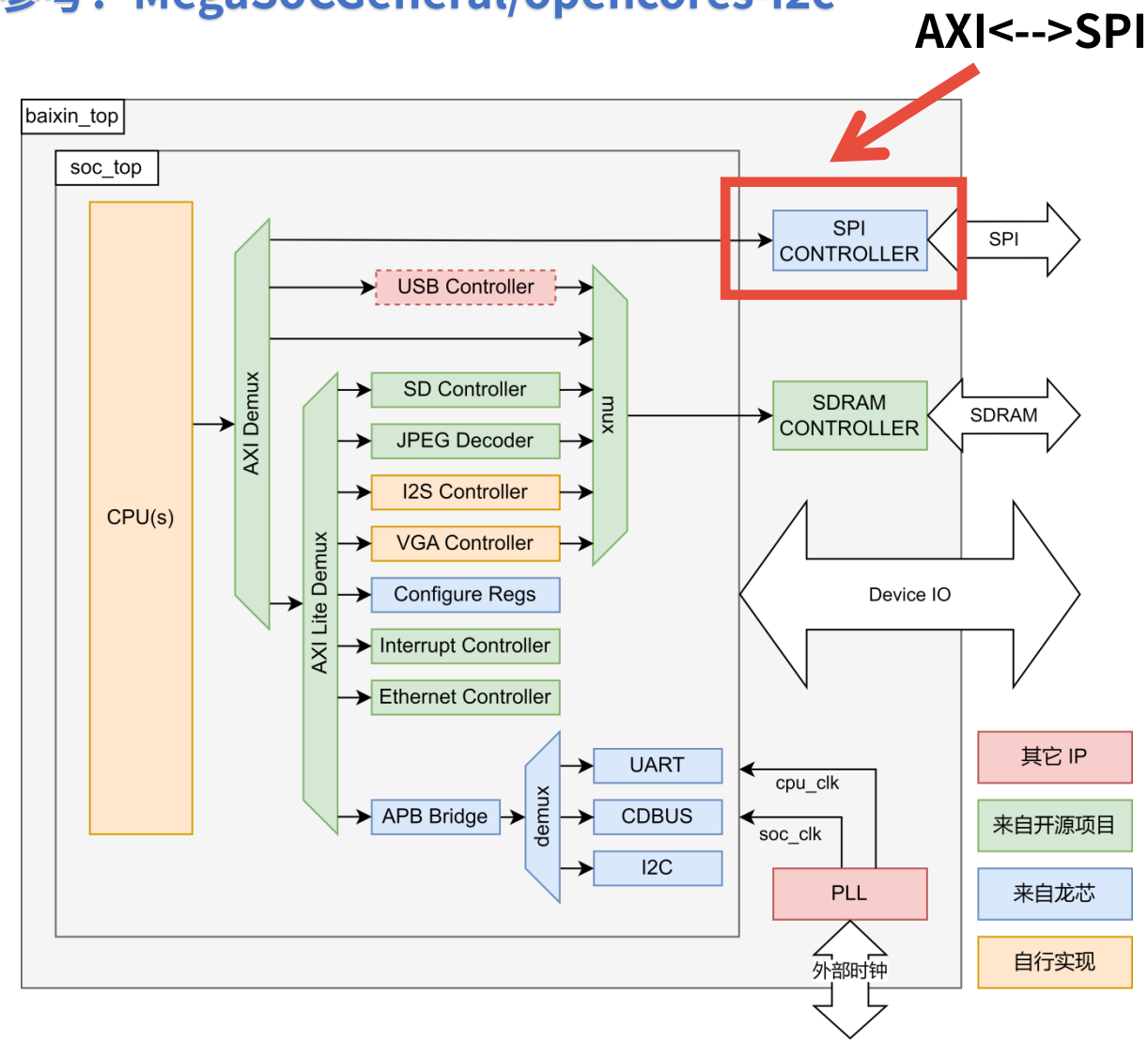
# 1.5 SPI 总线

## 串行外围设备接口

- 通信简单，支持全双工通信
- 在芯片的管脚上只占用四根线
- 采用主-从模式的控制方式
- 采用同步方式传输数据
- 通常用于 Flash、实时时钟等



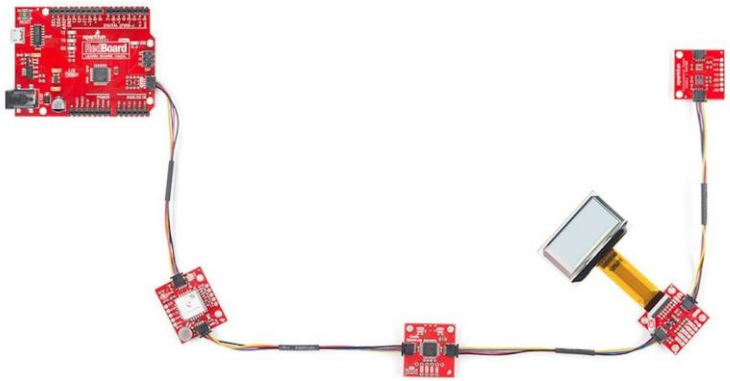

参考：MegaSoCGeneral/opencores-i2c

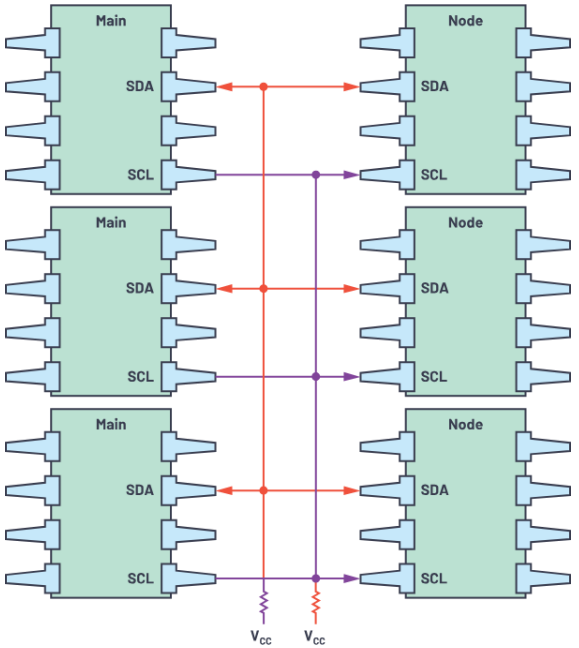


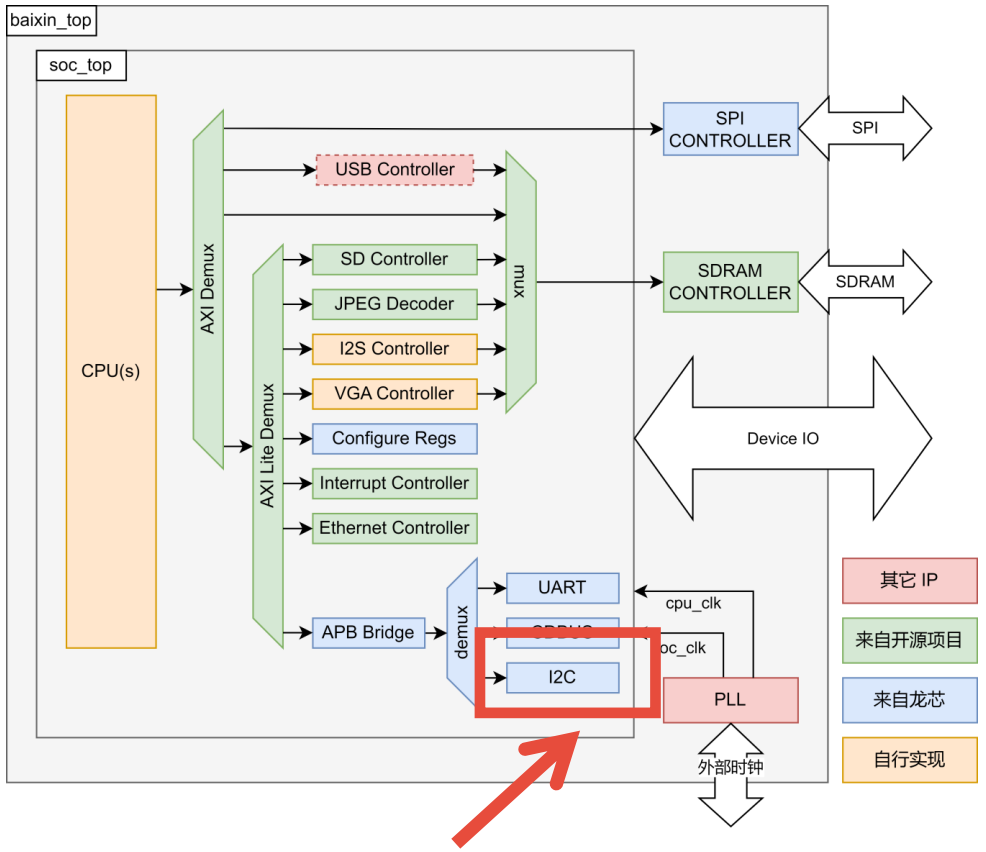
# 1.6 I2C 总线

- Philips 公司在八十年代初推出的一种串行半双工总线
  - 用于近距离、低速的芯片之间的通信
  - 多主机总线
  - I2C总线有两根双向的信号线：
    - 数据线SDA用于收发数据
    - 时钟线SCLK用于通信双方时钟的同步

参考：MegaSoCGeneral/i2c-slave





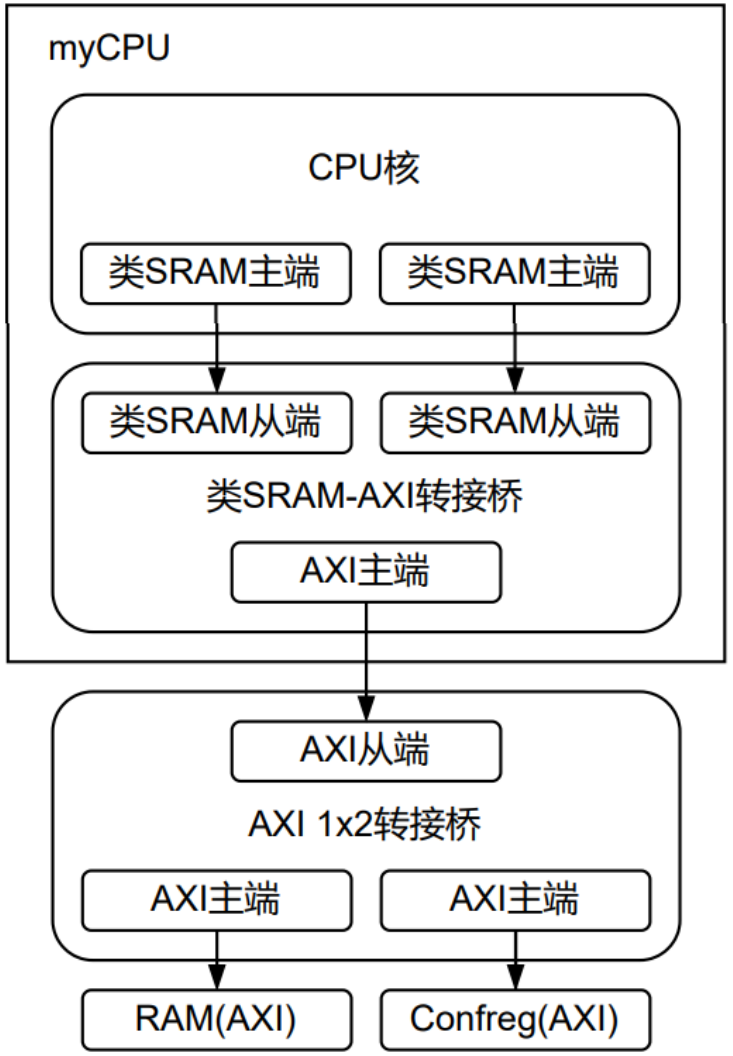




APB<-->I2C

# 1.7 类 SRAM 总线

## □ 类SRAM总线和转接桥（简化访存）

| 信号名 | 位宽 | 源 | 说明 |
|---|---|---|---|
| clk | 1 | 时钟源 | 系统时钟 |
| req | 1 | 主 | 请求信号，高电平代表有读写请求 |
| wr | 1 | 主 | 高电平代表写请求，低电平代表读请求 |
| size | 2 | 主 | 请求传输的字节数 |
| addr | 32 | 主 | 请求地址 |
| wstrb | 4 | 主 | 写请求的字节写使能 |
| wdata | 32 | 主 | 写请求的写数据 |
| addr_ok | 1 | 从 | 请求地址传输完成，读：地址被接受；写：地址和数据被接收 |
| data_ok | 1 | 从 | 请求的数据传输完成，读：数据返回；写：数据写入完成 |
| rdata | 32 | 从 | 请求返回的读数据 |

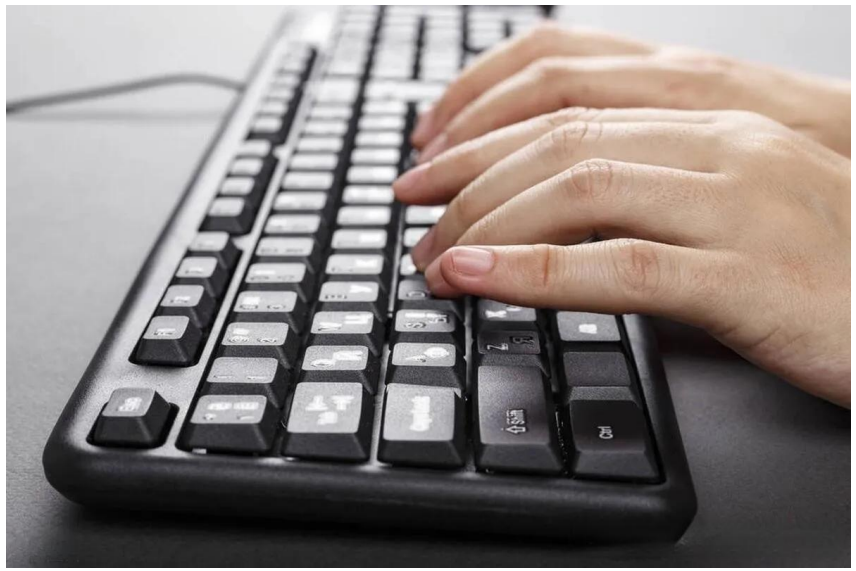


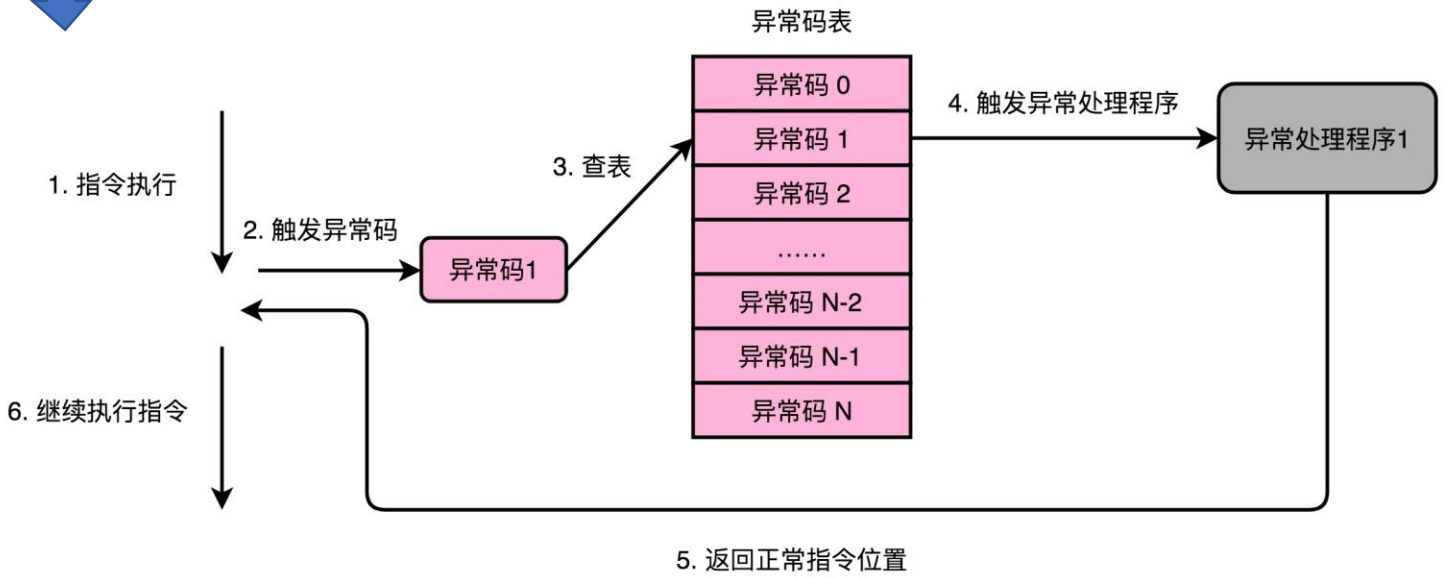
# Day3 目录

# 2.1 例外与中断

## 什么是中断和异常

- 中断和异常是一种事件处理机制，通过中断或异常发出一个信号， **操作系统**会**打断当前的操作**，然后**根据信号找到对应的处理程序**处理这个中断或异常，处理完毕之后再根据处理结构是否要**返回到源程序**接着往下执行

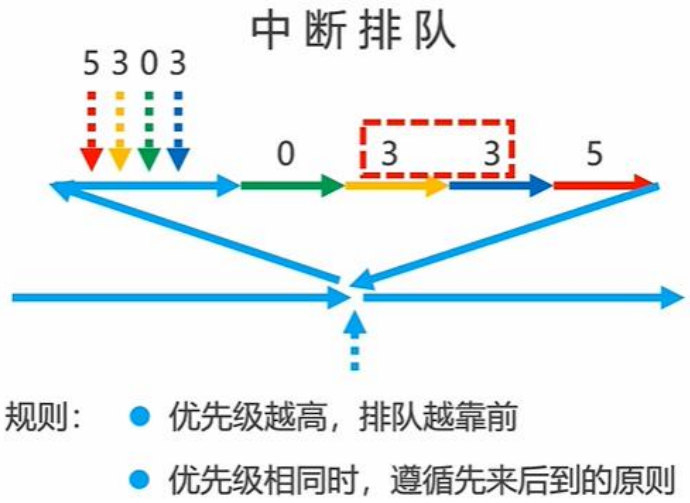




# 2.1 例外与中断

## 中断类型

- 每个处理器核内部可记录12个线中断，分别是：
  - 1个核间中断（IPI）
  - 1个定时器中断（TI）
  - 8个硬中断（HWI0-HWI7）
  - 2个软中断（SWI0-SWI1）




| 中断类型 | 中断源 | 采样记录位 | 示例 |
|---|---|---|---|
| IPI | 核外中断控制器 | CSR.ESTAT.IS[12] | CPU1通知CPU2 |
| TI | 核内恒定频率定时器 | CSR.ESTAT.IS[11] | 操作系统进程切换 |
| HWI | 处理器核外部 | CSR.ESTAT.IS[9:2] | 鼠标、键盘 |
| SWI | 处理器核内部 | CSR.ESTAT.IS[1:0] | 异步任务处理 |

# 2.1 例外与中断

## 例外入口

- TLB重填例外的入口来自于CSR.TLBRENTRY
- 其他例外入口来自于CSR.EENTRY

## 例外优先级

- 中断
- 取指阶段例外（取指地址错、取指TLB相关例外）
- 译码阶段例外
- 执行阶段例外（地址对齐错、TLB相关例外）

| Ecode | EsubCode | 例外代号 | 例外类型 |
|---|---|---|---|
| 0x0 | 0 | INT | 中断。 |
| 0x1 | 0 | PIL | load 操作页无效例外 |
| 0x2 | 0 | PIS | store 操作页无效例外 |
| 0x3 | 0 | PIF | 取指操作页无效例外 |
| 0x4 | 0 | PME | 页修改例外 |
| 0x7 | 0 | PPI | 页特权等级不合规例外 |
| 0x8 | 0 | ADEF | 取指地址错例外 |
| | 1 | ADEM | 访存指令地址错例外 |
| 0x9 | 0 | ALE | 地址非对齐例外 |
| 0xB | 0 | SYS | 系统调用例外 |
| 0xC | 0 | BRK | 断点例外 |
| 0xD | 0 | INE | 指令不存在例外 |
| 0xE | 0 | IPE | 指令特权等级错例外 |
| 0xF | 0 | FPD | 浮点指令未使能例外 |
| 0x12 | 0 | FPE | 基础浮点指令例外 |
| 0x1A–0x3E | | | 保留编码 |
| 0x3F | 0 | TLBR | TLB 重填例外 |

# 2.2 特权架构

## □ 控制与状态寄存器（CSR）

- CSR寄存器用于管理处理器和系统的各种控制信息与状态信息。用户程序或操作系统通过读写这些寄存器，来获得系统状态，或者配置/控制CPU及相关外设的行为。

标志处理器状态（模式，使能等）

| 地址 | 名称 | |
|---|---|---|
| 0x0 | 当前模式信息 | CRMD |
| 0x1 | 例外前模式信息 | PRMD |
| 0x2 | 扩展部件使能 | EUEN |
| 0x4 | 例外配置 | ECFG |
| 0x5 | 例外状态 | ESTAT |
| 0x6 | 例外返回地址 | ERA |
| 0x7 | 出错虚地址 | BADV |
| 0xc | 例外入口地址 | EENTRY |
| 0x10 | TLB 索引 | TLBIDX |
| 0x11 | TLB 表项高位 | TLBEHI |
| 0x12 | TLB 表项低位 0 | TLBELO0 |
| 0x13 | TLB 表项低位 1 | TLBELO1 |

例外控制

TLB控制

————类似MIPS中的CP0寄存器

| | | |
|---|---|---|
| 0x18 | 地址空间标识符 | ASID |
| 0x19 | 低半地址空间全局目录基址 | PGDL |
| 0x1A | 高半地址空间全局目录基址 | PGDH |
| 0x1B | 全局目录基址 | PGD |
| 0x20 | 处理器编号 | CPUID |
| 0x30~0x33 | 数据保存 | SAVE0~SAVE3 |
| 0x40 | 定时器编号 | TID |
| 0x41 | 定时器配置 | TCFG |
| 0x42 | 定时器值 | TVAL |
| 0x44 | 定时中断清除 | TICLR |
| 0x60 | LLBit 控制 | LLBCTL |
| 0x88 | TLB 重填例外入口地址 | TLBRENTRY |
| 0x98 | 高速缓存标签 | CTAG |
| 0x180~0x181 | 直接映射配置窗口 | DMW0~DMW1 |

进程控制

软件控制定时器

原子访存映射配置

# 2.3 例外硬件处理

## □ 例外硬件处理通用过程

- CSR.CRMD的PLV和IE分别存到CSR.PRMD的PPLV、PIE，然后将CSR.CRMD的PLV置为0，IE置为0，将触发例外指令的PC值记录到CSR.ERA
- 跳转到例外入口处取值
- 执行ERTN后，将CSR.PRMD中的PPLV、PIE恢复至CSR.CRMD的PLV、IE
- 跳转到CSR.ERA所记录的地址处取指

用户态

① 异常处理准备
- □ 记录触发异常指令 PC 至 CSR.ERA
- □ 硬件保存部分现场（PPLV←PLV, PIE←IE）
- □ 提升特权等级至最高（PLV←0）
- □ 记录其它异常相关信息（e.g. CSR.BADV）

② 确定异常来源，进入异常处理入口。
- □ 记录异常类型（CSR.ESTAT.Ecode、EsubCode）
- □ TLB 重填异常入口（CSR.TLBRENTRY），其余异常入口（CSR.EENTRY）

内核态

I. 保存执行状态
II. 处理异常
III. 恢复执行状态
IV. 执行 ertn 指令返回
- □ 恢复硬件保存的那部分现场（PLV←PPLV, IE←PIE）
- □ 跳转到 CSR.ERA 所存的返回地址处

# 2.4 中断硬件处理

## 处理器硬件响应中断的处理过程

- 中断信号被处理器采样至CSR.ESTAT.IS域
- 与CSR.ECFG.LIE域中的局部中断使能信号按位与，得到多位中断向量
- 当CSR.CRMD.IE=1且中断向量不全为0时，处理器挑选指令标记中断
- 随后处理器硬件的处理过程与普通例外的处理过程一致

# Day3 目录