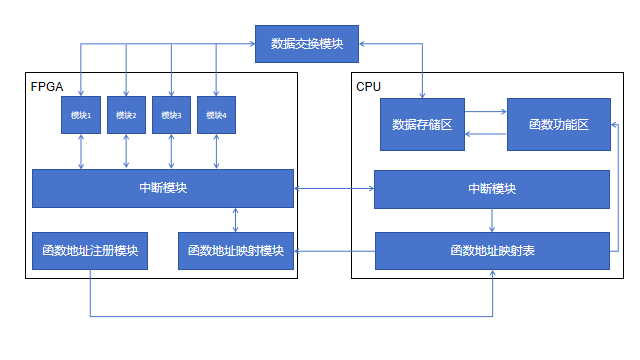
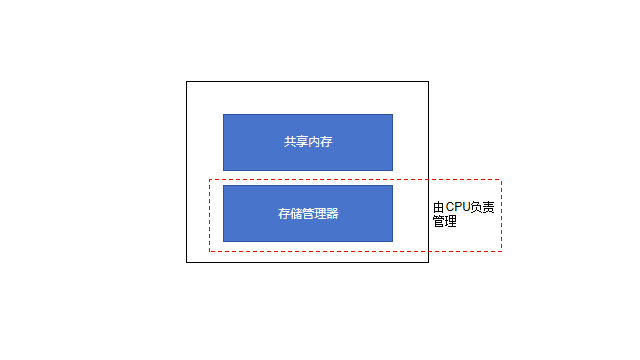
## 面向异构SoC的函数调用机制设计方案vol2

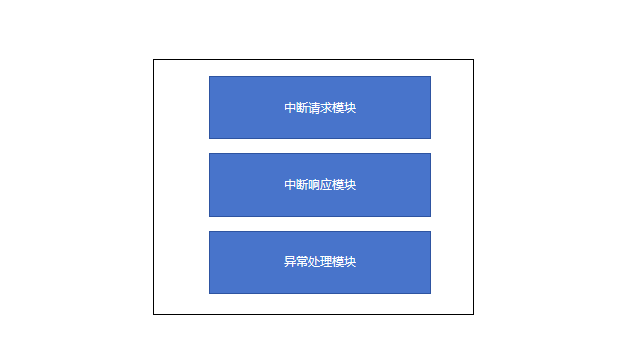
#### 整体结构：



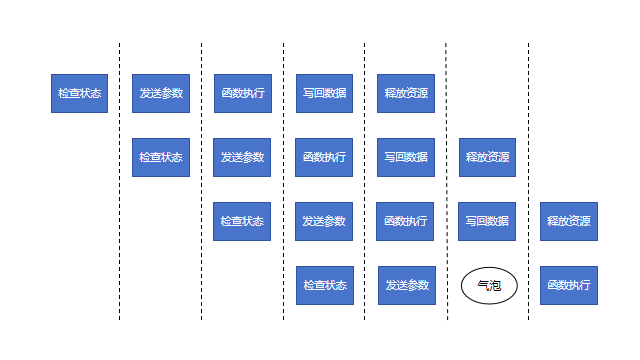
数据交换模块结构：



中断模块结构：



**流水处理流程：**



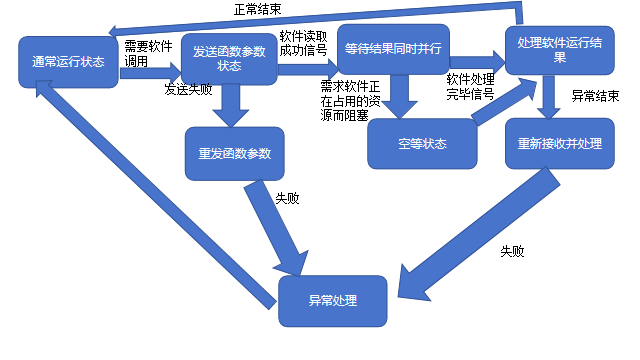
**接口的定义**：

* **FPGA到CPU的函数调用机制接口**

**执行流程**：

1. FPGA中模块进行初始化时，向CPU进行函数注册，即注册需要调用的函数。在CPU中应具有函数映射表以将函数与模块进行匹配注册，并将函数集中存放在一个单独的函数功能区，便于CPU顺利执行。
2. 在运行中，需要调用函数时，模块向中断请求模块发送中断请求，中断请求模块将对应模块号经过函数地址映射模块检查是否存在，若存在则发送中断请求到CPU的中断响应模块。
3. CPU响应中断，中断处理时，硬件向软件发送特定信号, 使得软件中断, 软件的中断处理程序先禁用其他中断, 数据规模小可以利用中断传输, 规模大考虑共享内存, 获取所需函数编号及数据, 根据实际情况判断是否需要进行校验设计, 完毕后中断结束, 向硬件发送信号, 使其可以进入下一个状态 (可能需要硬件备份一份数据在特定内存, 接收到接收/校验失败信号时可以进入到重发状态).判断CPU是否处于空闲状态，若空闲则响应，并发送数据获取信号，FPGA接收信号后，将模块对应数据发往数据交换模块中共享内存处，便于数据存取。
4. 调用对应函数，如果还没有加载对应编号的程序, 那么尝试加载, 成功后继续, 失败报错; 如果已经加载就跳过。参数获取采用外接 IP 数据交换模块读取总线, 数据规模过大IO瓶颈时可能考虑读取共享内存。所有软件程序都应当具备检测输入数据是否有效的函数 (\_\_valid\_\_), 然后在运行前检测数据是否有效, 无效向硬件报告, 硬件重发一次, 依旧无效直接报错, 交由硬件端处理 (将数据写入log) 软件端从程序首地址开始执行。
5. 函数运行结束，执行完毕后向硬件发送信号, 并使其准备接收数据. 硬件读取完毕后, 向软件发送信号, 软件即可释放所占用的资源(例如设为只读的内存)将运行结果和状态写入CPU数据存储区，并将存储区中数据发往数据交换模块共享内存中，FPGA将共享内存内数据送往各模块以完成计算。

**程序运行时状态机**：



* **CPU到FPGA的模块调用接口**

**执行流程**：

1. FPGA内的模块在FPGA初始化之时先根据函数注册模块在CPU内生成函数映射表。
2. 用户程序在调用库函数时，根据函数映射表对被调函数入口地址进行匹配，匹配成功则同时调用FPGA内部模块，否则调用原函数地址。
3. 调用FPGA内模块时采用与FPGA到CPU相似的通信方式。先将参数写入CPU内数据存储区中，并向中断模块发送中断请求以调用FPGA子模块。
4. 中断模块将信号发往FPGA，检查模块寄存器是否已满，若是则转异常处理，否则允许数据交换模块将CPU存储区中数据写往数据交换模块共享内存中。FPGA模块提取数据交换模块共享内存中数据，并运算后将结果写入FPGA各模块寄存器中。数据交换模块检查总线状态若空闲则将寄存器中数据写入数据交换模块共享内存中。CPU则根据流水流程从共享内存处获得运算结果。

**流水逻辑：**

1. 进入流水（检查软件是否空闲）
2. 向软件发送参数（结束标志：软件接收数据ACK）
3. 软件执行（硬件可以同时并行）
4. 软件计算完成（结束标志：软件写入数据ACK）
5. 退出流水，解除资源锁（释放相关共享内存处资源）

**软件逻辑（轮询方式）：**

1. 定期轮询（有数据就接入队列）
2. 检查资源（确定数据量及资源需求量，允许则入队）
3. CPU直接使用函数功能区函数进行计算，数据写入数据存储区中。
4. CPU做任务调度，按计算次序交付给数据交换模块。

**数据交换时通信协议：**

* **通信时参数形式**

FPGA到数据交换模块：

Request（需要的内存大小，函数编号）

Direct\_send（数据，函数编号）

Indirect\_send（数据地址，函数编号）

CPU到数据交换模块：

Request（需要的内存大小，函数编号）

Direct\_send（数据，函数编号）

Indirect\_send（数据地址，函数编号）

数据交换模块到CPU：

Get\_request(数据地址，函数编号)

* **校验和定义**

写前校验和（sum1）：写前计算数据校验和

写后校验和（sum2）：写后数据交换模块计算校验和

读后校验和（sum3）：读数据之后计算校验和

数据正常：sum1==sum2==sum3

* **数据的生命周期**

分配完成等待写->写后->读后->释放

* **具体行为逻辑**

FPGA对共享内存处数据的写入分为两种方式，一种为直接写入，直接在参数中给出具体数据。一种为间接写入，此时数据存放在CPU中数据存储区中，数据交换模块需向CPU发送取数请求函数，CPU收到后若空闲则通过AXI协议将数据写入共享内存中。此时需规定一定的超时时间，以便异常时，数据交换模块能够正常取数。CPU对数据交换模块的写入则较为简单，通过AXI协议可直接将数据写入交换模块共享内存中。交换模块需具备管理功能以便管理共享内存，**目前的方法是将管理功能交由CPU执行并编写单独的管理器**。

* **管理器**

对内存空间的管理采用线段树的数据结构形式，可以提高管理效率。

出于安全考虑，管理器不允许对写入的数据进行释放空间之外的操作

要求管理器对于通信时的函数请求以尽可能快的速度进行回复。

* **双方通信时规定**

对于CPU与FPGA，在申请共享内存后，只能修改分配的部分

CPU与FPGA可同时读共享内存中数据

CPU与FPGA的通信需受数据交换模块中管理器的管理

* **可能的异常情况**
  + 超出共享内存大小
  + 超时
  + 数据等待时间过长

**软件处函数功能区函数形式：**

函数功能区的函数采用C++文件的方式进行编写：

./1 --start-memory 0x....

具体函数模板形式：



./main --funcNo=233 --start-address=... --size=...

**最小系统（数据自+1）：**

系统组成：FPGA，CPU，管理器

运行流程：

1. 初始化（由CPU发起）

FPGA调用CPU：

1. FPGA将数x写入共享内存
2. CPU读取函数功能区函数并且打印x（此处检查FPGA对CPU的IO通信）
3. CPU计算x+1，并输出x+1（此处检查函数功能）
4. CPU将x+1写入存储区并写入共享内存
5. FPGA读共享内存处x+1，并通过led等方式显示结果(此处检查CPU对FPGA的IO通信)

CPU调用FPGA：

1. CPU将数据x写入共享存储区，并向FPGA发送函数模块调用请求
2. FPGA内模块计算x+1，并存入模块寄存器中
3. FPGA将寄存器中内容发往数据交换模块然后发往CPU
4. CPU输出x+1

此后重复上述流程（初始化以外）

**测试例程：**

* 长时间循环任务
* 长循环+短循环
* 长循环+中循环+短循环，检查循环乱序对输出影响
* 内存压力测试
* 多个FPGA接往CPU时高并行度测试

**异常处理：**

FPGA: 得到日志(函数编号,内存,冲突类型)->将日志发给管理器->管理器发给CPU进行输出

CPU: 输出详细日志文件

**可能的问题：**

* **无法动态加载FPGA模块的函数**
* **效率不足（FPGA交由CPU进行函数运算效率不足）**
* **确定最小指令集（函数功能的构成部分）**