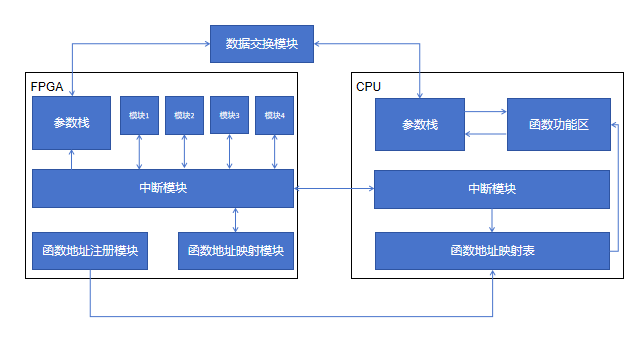
## 面向异构SoC的函数调用机制设计方案

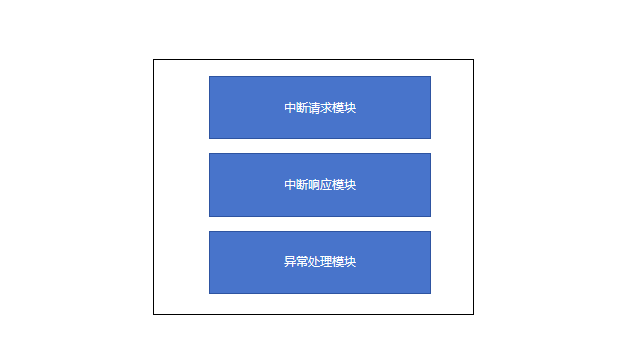
#### 整体结构：



参数栈结构：



中断模块结构：



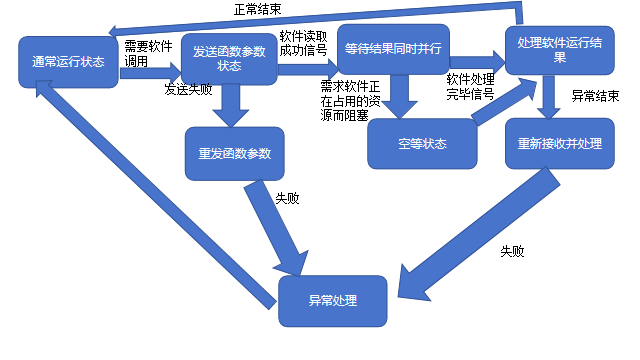
接口的定义：

* **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，参数栈中输出栈处数据通过数据交换模块发往CPU参数栈中输入栈。
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 核读取总线, 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参数栈处输出栈中，并将输出栈中数据发往FPGA内输入栈中，FPGA将输入栈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内参数栈的输入栈中，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或参数栈已溢出，若是则转异常处理，否则允许数据交换模块将CPU内输入栈中数据写往FPGA 内输出栈中。FPGA模块提取FPGA输出栈中数据，并运算后将结果写入FPGA内输入栈中。数据交换模块检查总线状态若空闲则将输入栈中数据写入CPU内输入栈中。CPU根据估算运行时间从输入栈内提取数据作为子模块函数调用运算结果。

程序编译运行过程：

1. 编译器不做修改，程序正常流程进行编译链接生成可执行文件
2. 程序运行开始，将程序读入进程，进行相关解释运行【这里需要对ELF文件执行流程进行详细了解】
3. 对于每一个程序段，根据FPGA中保存的函数地址映射关系，动态修改入口地址
4. FPGA和软件中的库不需要了解其功能是由谁调用的。