|  |  |
| --- | --- |
| 文档编号 |  |
| 文档版本 |  |
| 文档管控 | 内部公开 |
| 存档日期 |  |

昆明湖V1项目ExuBlock模块AS

|  |  |
| --- | --- |
| 编 写： | 胡轩 |
| 校 对： |  |
| 审 核： |  |
| 批 准： |  |

昆明湖V1项目

2024年XX月XX日

文档修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **版本编号** | **变化状态** | **变更说明** | **作者** | **日期** |
| 1 | V0.1 | C | 语雀导出 | 胡轩 | 2023.6.13 |
|  |  |  |  |  |  |
| 3 |  |  |  |  |  |
|  |  |  |  |  |  |

\*变化状态：C—创建，A—增加，M—修改，D—删除

文档审批信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **审核** | **会签** | **批准** | **备注** |
| V1.0 |  |  |  |  |
|  |  |  |  |  |

目 录

[1 简介 1](#_Toc107389929)

[1.1 文档介绍 1](#_Toc107389930)

[1.2 参考文档 1](#_Toc107389931)

[1.3 术语说明 1](#_Toc107389932)

[1.4 技术背景 1](#_Toc107389933)

[2 设计规格 2](#_Toc107389934)

[3 功能描述 2](#_Toc107389935)

[4 总体设计 2](#_Toc107389936)

[4.1 整体框图 2](#_Toc107389937)

[4.2 接口列表 3](#_Toc107389938)

[4.3 接口时序 3](#_Toc107389939)

[4.4 时钟复位 4](#_Toc107389940)

[4.5 寄存器配置 4](#_Toc107389941)

[4.6 补充说明 5](#_Toc107389942)

[5 模块设计 5](#_Toc107389943)

[5.1 二级模块A 6](#_Toc107389944)

[5.1.1 功能 6](#_Toc107389945)

[5.1.2 整体框图 6](#_Toc107389946)

[5.1.3 接口列表 6](#_Toc107389947)

[5.1.4 接口时序 6](#_Toc107389948)

[5.1.5 关键电路 6](#_Toc107389949)

[5.1.6 三级模块设计 6](#_Toc107389950)

[5.2 二级模块B 6](#_Toc107389951)

[5.3 关键电路 6](#_Toc107389952)

[5.3.1 时钟切换电路 6](#_Toc107389953)

[6 PPA优化设计 7](#_Toc107389954)

[7 验证关注点 7](#_Toc107389955)

[8 Floorplan 建议 7](#_Toc107389956)

[9 遗留问题 8](#_Toc107389957)

1. 简介
   1. 文档介绍

*概述本文档的目的、用途、适用人群、在整体项目中的位置等*

*例如：*

本文当是XXX的AS文档，描述XXX架构设计。

本文档主要用于指导芯片模块的详细设计及验证。

* 1. 参考文档

*列出相关的参考文档。*

1. XXXX
2. XXXX
   1. 术语说明

*列出本文档的关键术语说明。*

表1.1 术语说明

|  |  |  |
| --- | --- | --- |
| **缩写** | **全称** | **描述** |
| CRU | Clock Reset Unit | 时钟复位单元 |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. 技术背景

*可选项。简要介绍本模块的技术背景，比如协议说明、应用范围等。如果不需要则直接删除本小节。注意要根据模块特性进行提炼，不要大段的复制黏贴。*

1. 设计规格

*如果是模块的AS需要列出本模块支持的规格，如果是总体AS可忽略*

*规格中包含功能、性能、PPA规格。*

*例如：*

*1. 支持从内存取值*

*2. 支持压缩指令*

*3. 支持单条指令跨预测块*

*4. 支持内存指令预译码*

*5. 支持5种类型的内存指令分支预测错误检查*

*6. 支持分支预测错误后重定向预测器*

*7. 支持从MMIO取值*

*8. 支持从外部刷新IFU流水线*

*9. 支持一次输出一个预测块的码流以及附加信息*

*10. 支持debug辅助*

1. 功能描述
   1. 功能概述

ExuBlock 模块接收来自 Dispatch 模块的 uop，经内部 Scheduler 模块的调度后，送入内部 FUBlock 模块或 外部 MemBlock 执行。ExuBlock 模块中负责记录 CPU 的绝大部分体系结构相关的状态，包括整型寄存器、浮点寄存器及CSR寄存器。

ExuBlock 有两个特化的模块，即 IntBlock 和 FloatBlock，在 verilog 代码中分别对应 ExuBlock 和 ExuBlock\_1。IntBlock 负责所有整型运算指令的调度与执行，和所有访存指令的调度，访存指令在MemBlock中执行。此外CSR指令、fence指令、分支跳转指令、整数转浮点数指令也在 IntBlock 中调度并执行。FloatBlock 负责所有浮点运算指令的调度和执行，还有浮点转换整数指令的调度和执行。

* 1. 功能详述

ExuBlock 模块顶层除了对配置包含 fastUopOut 的写回信号、从 ROB 来的重定向延迟一拍外，没有其它寄存器。在此描述的功能详述是ExuBlock内部Scheduler和FUBlock功能总和。

* + 1. 执行单元处理相应的指令

香山依据功能单元的源操作数类别将执行单元划分进两个ExuBlock中，具体如下表所示。在每个ExuBlock中，执行单元全部在FUBlock中。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 功能单元 | numSrc(Int/Fp) | write(Int/Fp) | 重定向 | 延迟 | fastUop | 可触发异常 | flushPipe | 写回不需要仲裁 | ExuBlock | RS | Exu/Fu |
| alu | 2/1 | 1/0 | 1 | 0 | 0 | - | 0 | 1 | Int | rs/rs\_0  rs/rs\_1 | exeUnits\_0/alu  exeUnits\_1/alu  exeUnits\_2/alu  exeUnits\_3/alu |
| mul | 2/0 | 1/0 | 0 | 2 | 1 | - | 0 | 1 | Int | rs\_1/rs\_0 | exeUnits\_4/mul  exeUnits\_5/mul |
| div | 2/0 | 1/0 | 0 | - | 1 | - | 0 | 1 | Int | rs\_1/rs\_0 | exeUnits\_4/div  exeUnits\_5/div |
| bku | 2/0 | 1/0 | 0 | 1 | 1 | - | 0 | 1 | Int | rs\_1/rs\_0 | exeUnits\_4/bku  exeUnits\_5/bku |
| jmp | 1/0 | 1/0 | 1 | 0 | 0 | - | 0 | 0 | Int | rs\_2/rs\_0 | exeUnits\_6/jmp |
| i2f | 1/0 | 0/1 | 0 | 2 | 1 | - | 0 | 0 | Int | rs\_2/rs\_0 | exeUnits\_6/i2f |
| csr | 1/0 | 1/0 | 0 | 0 | 0 | illegalInstr  breakPoint  ecallU  ecallS  ecallM | 1 | 0 | Int | rs\_2/rs\_0 | exeUnits\_6/csr |
| fence | 2/0 | 0/0 | 0 | - | 0 | illegalInstr | 0 | 0 | Int | rs\_2/rs\_0 | exeUnits\_6/fence |
| ldu | 1/0 | 1/1 | 0 | - | 0 | loadAddrMisaligned  loadAccessFault  loadPageFault | 1 | 0 | Mem | rs\_3/rs\_0 | - |
| sta | 1/0 | 0/0 | 0 | - | 0 | storeAddrMisaligned  storeAccessFault  storePageFault | 0 | 1 | Mem | rs\_4/rs\_0 | - |
| mou | 1/0 | 0/0 | 0 | - | 0 | loadAddrMisaligned  loadAccessFault  loadPageFault  storeAddrMisaligned  storeAccessFault  storePageFault | 0 | - | Mem | - | - |
| mouData | 1/0 | 0/0 | 0 | - | 0 | - | 0 | - | Mem | - | - |
| fmac | 0/3 | 0/1 | 0 | - | 1 | - | 0 | 1 | Fp | rs/rs\_0  rs/rs\_1 | exeUnits\_0/fmac  exeUnits\_1/fmac  exeUnits\_2/fmac  exeUnits\_3/fmac |
| f2i | 0/1 | 1/0 | 0 | 2 | 1 | - | 0 | 1 | Fp | rs\_1/rs\_0 | exeUnits\_4/f2i  exeUnits\_5/f2i |
| f2f | 0/1 | 0/1 | 0 | 2 | 1 | - | 0 | 1 | Fp | rs\_1/rs\_0 | exeUnits\_4/f2f  exeUnits\_5/f2f |
| fdivSqrt | 0/2 | 0/1 | 0 | - | 1 | - | 0 | 1 | Fp | rs\_1/rs\_0 | exeUnits\_4/fdivSqrt  exeUnits\_5/fdivSqrt |

功能单元编码及所在的ExeUnits如下。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FuType编码** | **类别** | **ExeUnits**  **Int** | **ExeUnits**  **Float** | **MemUnits** | **描述** |
| b0000 | jmp | 6 | - | - | 来自ROB的重定向信号，详见《重要Bundle描述》的Redirect。 |
| b0001 | i2f | 6 | - | - | 来自dispatch的uop入队信号。详见《重要Bundle描述》的MicroOp。  X的取指范围为[0, numEnq)。 |
| b0010 | csr | 6 | - | - | 第 X 个入队uop的第 Y 个源操作数。由Dispatch2Rs从物理寄存器堆中读取。 |
| b0110 | alu | 0，1，2，3 | - | - |  |
| b0100 | mul | 4，5 | - | - |  |
| b0101 | div | 4，5 | - | - |  |
| b0011 | fence | 6 | - | - |  |
| b0111 | bku | 4，5 | - | - |  |
| b1000 | fmac | - | 0，1，2，3 | - |  |
| b1011 | fmisc | - | 4，5 | - |  |
| b1010 | fDivSqrt | - | 4，5 | - |  |
| b1100 | ldu | - | - | LoadUnit\_0,  LoadUnit\_1 |  |
| b1101 | stu | - | - | StoreUnit\_0  StoreUnit\_1 |  |
| b1111 | mou | - | - | AtomicsUnit |  |

* + 1. 为执行单元准备数据

在每个ExuBlock中，包含一个Scheduler模块，用于对dispatch发送来的指令进行调度，目的是等待指令需要的源操作数准备好，再将指令送入FUBlock或MemBlock中执行。

* + - 1. 从寄存器堆读取源操作数

指令的源操作数一般取自 Scheduler 中的寄存器堆。南湖设计中，仅在指令从 DispatchQueue 发出，经 Dispatch2Rs 模块将指令送入保留站的过程中，读取寄存器堆的数据。整型寄存器堆与浮点寄存器堆各有14个读端口，读端口分配情况如下所示。

|  |  |  |
| --- | --- | --- |
| **Dispatch2Rs** | **Out port** | **读端口编号** |
| intDispatch | 0 | 0,1 |
| intDispatch | 1 | 2,3 |
| intDispatch | 2 | 4,5 |
| intDispatch | 3 | 6,7 |
| lsDispatch | 0 | 8 |
| lsDispatch | 1 | 9 |
| lsDispatch | 2 | 10 |
| lsDispatch | 3 | 11 |
| lsDispatch | 4 | 12 |
| lsDispatch | 5 | 13 |

浮点寄存器读端口分配表

|  |  |  |
| --- | --- | --- |
| **Dispatch2Rs** | **Out port** | **读端口编号** |
| fpDispatch | 0 | 0,1,2 |
| fpDispatch | 1 | 3,4,5 |
| fpDispatch | 2 | 6,7,8 |
| fpDispatch | 3 | 9,10,11 |
| lsDispatch | 4 | 12 |
| lsDispatch | 5 | 13 |

StoreUnit既可以读整型寄存器写到内存，也可以读取浮点寄存器写到内存，由于访存指令在IntBlock中调度，需要从FloatBlock中的浮点寄存器堆读取数据。当store指令的源操作数来自浮点寄存器时，lsDispatch的4和5号端口将从浮点寄存器堆读取数据。

* + - 1. 从保留站读取源操作数
    1. 将执行结果写回

除了store等没有目的寄存器的指令外，其余指令都会将执行结果写回寄存器。而寄存器的写端口是有限的，因此需要对写回信号进行仲裁，让优先级高的写回信号先写回寄存器。没有目的寄存器的执行单元，也需要写回这个步骤来表明指令执行结束，写回的目标是CtrlBlock中的ROB。不产生异常的执行单元，如alu、mdu等，不需要额外写回到控制单元的信号。浮点运算执行单元，除了将执行结果写回寄存器外，也需要将浮点状态位写回到CtrlBlock。Load执行单元除了写回访存数据外，也需要将可能出现的异常信息更新到控制单元。alu和jump执行单元会生成重定向信号，重定向信号会写回到CtrlBlock中的RedirectGenerator，最终生成发送到frontend的重定向信号。

ExeUnit配置的写回优先级如下。

|  |  |  |
| --- | --- | --- |
| **ExeUnit** | **整型写回优先级** | **浮点写回优先级** |
| AluExeUnit | 0 | IntMax |
| MulDivExeUnit | 1 | IntMax |
| JumpCSRExeUnit | 2 | IntMax |
| FmacExeUnit | IntMax | 0 |
| FmiscExeUnit | IntMax | 1 |
| LdExeUnit | 0 | 0 |

优先级及其含义如下。

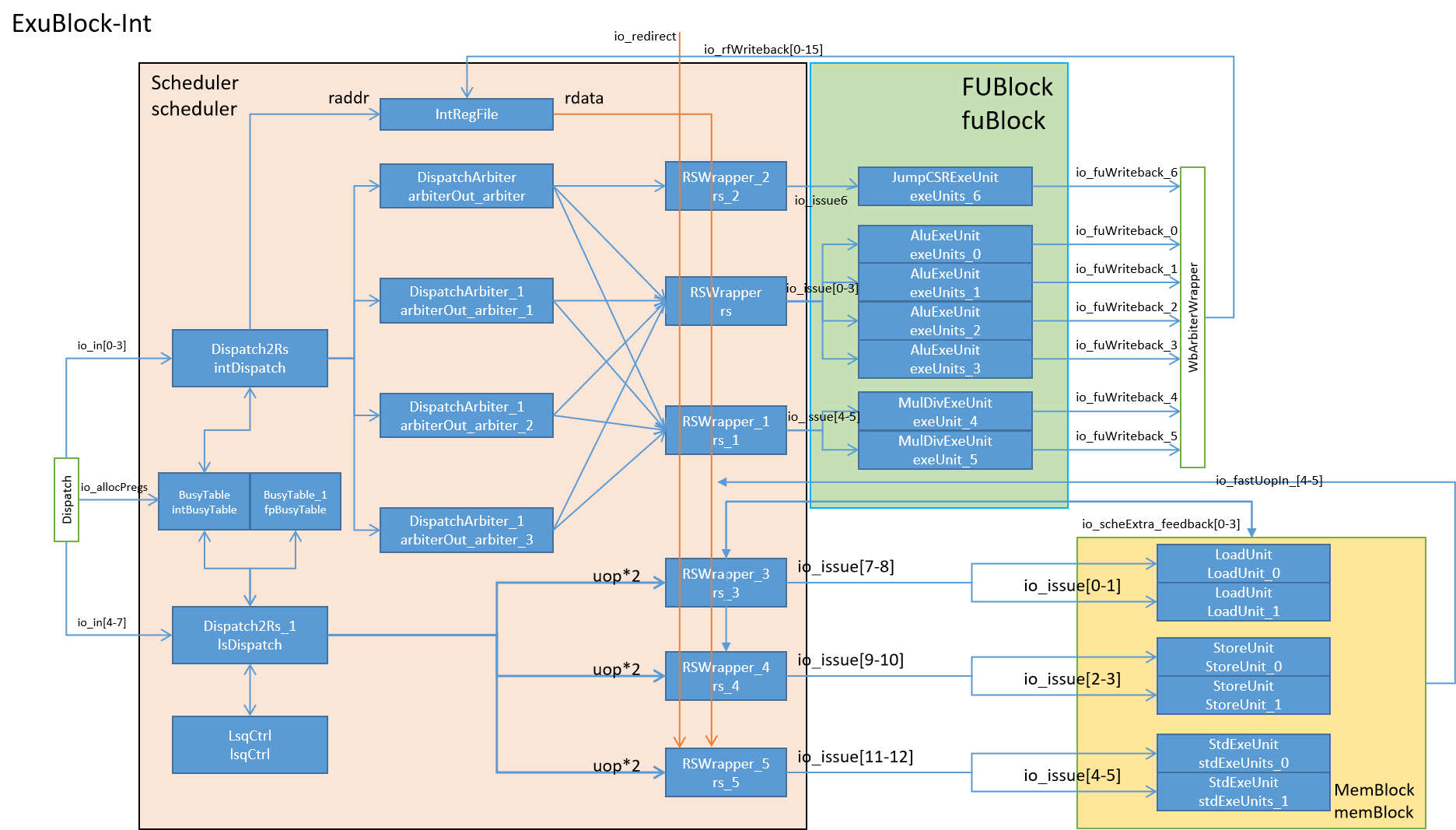
|  |  |
| --- | --- |
| **优先级** | **描述** |
| 0 | 独占写回端口 |
| 1 | 参与写回仲裁，仲裁优先级最高，执行完毕的下一拍即可写回 |
| >1 | 参与写回仲裁，值越大，优先级越低 |

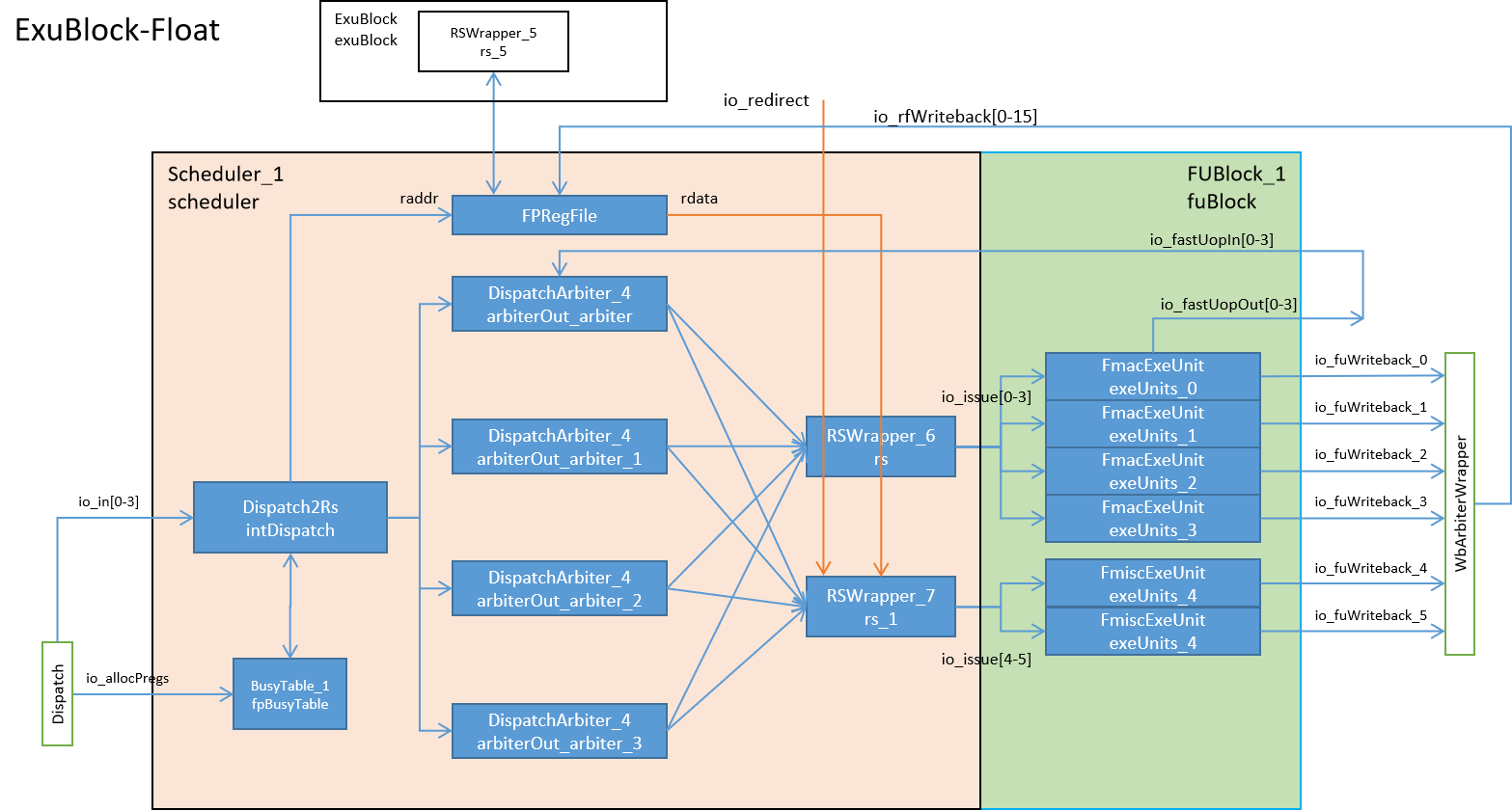
整型寄存器堆支持8写，浮点寄存器支持8写，WbAriber 中按以上配置生成写回仲裁逻辑，下面是功能单元写回仲裁表。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Exu** | **wbArbiter** | **subArbiterin** | **优先写回** | **subArbiterout** | **wb2Ctrl** | **备注** |
| int/exeUnits\_0 | 0 | int\_0 | 1 | int\_0 | - |  |
| int/exeUnits\_1 | 1 | int\_1 | 1 | int\_1 | - |  |
| int/exeUnits\_2 | 2 | int\_2 | 1 | int\_2 | - |  |
| int/exeUnits\_3 | 3 | int\_3 | 1 | int\_3 | - |  |
| int/exeUnits\_4 | 4 | int\_4 | 1 | int\_6 | - | 经过仲裁器，但是优先级最高 |
| int/exeUnits\_5 | 5 | int\_5 | 1 | int\_7 | - | 经过仲裁器，但是优先级最高 |
| int/exeUnits\_6 | 6 | int\_6|fp\_0 | 0 | int\_6|fp\_6 | 6 |  |
| fp/exeUnits\_0 | 7 | fp\_1 | 1 | fp\_0 | 7 |  |
| fp/exeUnits\_1 | 8 | fp\_2 | 1 | fp\_1 | 8 |  |
| fp/exeUnits\_2 | 9 | fp\_3 | 1 | fp\_2 | 9 |  |
| fp/exeUnits\_3 | 10 | fp\_4 | 1 | fp\_3 | 10 |  |
| fp/exeUnits\_4 | 11 | int\_7|fp\_5 | 0|1 | int\_7|fp\_6 | 11 | f2i写回整型寄存器|浮点结果写回经过仲裁器，但是优先级最高 |
| fp/exeUnits\_5 | 12 | int\_8|fp\_6 | 0|1 | int\_6|fp\_7 | 12 | f2i写回整型寄存器|浮点结果写回经过仲裁器，但是优先级最高 |
| loadUnit\_0 | 13 | int\_9|fp\_7 | 1|1 | int\_4|fp\_4 | 13 | load指令可以写回整型或浮点寄存器 |
| loadUnit\_1 | 14 | int\_10|fp\_8 | 1|1 | int\_5|fp\_5 | 14 | load指令可以写回整型或浮点寄存器 |
| storeUnit\_0 | 15 | - | - | - | 15 |  |
| storeUnit\_1 | 16 | - | - | - | 16 |  |
| atomicsUnit | 13|15 | int\_9|fp\_7 | 1|1 | int\_4|fp\_4 | 13|15 | MemBlock中协调loadUnit\_0和atomicsUnit |
| stdExeUnits\_0 | 17 | - | - | - | - |  |
| stdExeUnits\_1 | 18 | - | - | - | - |  |

注：在MemBlock中协调loadUnit\_0和atomicsUnit、storeUnit\_0和atomicsUnit的写回信号，二者从同一个端口输入wbArbiter。

1. 总体设计
   1. 整体框图





* 1. 接口列表
     1. 接口定义使用的参数列表

|  |  |  |
| --- | --- | --- |
| **参数** | **含义** | **取值** |
| XLEN | 香山处理器支持数据运算最大位宽 | 恒等于64 |
| RenameWidth | 重命名阶段一拍输出的uop个数 | 6 |
| IntDqDeqWidth | 整型运算队列一拍出队的uop个数 | 4 |
| LsDqDeqWidth | 访存发射队列一拍出队的uop个数 | 4 |
| JmpCnt | JumpUnit数量 | 1 |
| AluCnt | Alu数量 | 4 |
| MduCnt | Mdu数量，乘除法器个数 | 2 |
| LduCnt | Load Unit数量 | 2 |
| StuCnt | Store Unit数量 | 2 |
| FmacCnt | FMAC Unit数量 | 4 |
| FmiscCnt | FMISC Unit数量 | 2 |
| FpExuCnt | 浮点运算器数量 | FmacCnt + FpExuCnt = 6 |
| numOutFu | RS 在ExuBlock中，但执行单元不在ExuBlock中的执行单元个数 | LduCnt + StuCnt \* 2=6 |
| numIssuePorts | 发射端口数量 | JmpCnt + AluCnt + MduCnt + LduCnt + StuCnt \* 2 = 13 |
| numIntRfWritePorts | 整型寄存器写端口数量 | AluCnt + MduCnt + LduCnt = 8 |
| numFpRfWritePorts | 浮点寄存器写端口数量 | FpExuCnt + LduCnt = 8 |

* + 1. 接口列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **信号** | **方向** | **位宽** | **源** | **目的** | **描述** |
| clock | I | 1 |  |  |  |
| reset | I | 1 |  |  |  |
| io\_hartId | I | 1 | 核外 | difftest | 用于difftest的核心编号 |
| io\_redirect\_valid | I | 1 | ROB | ExuBlock | 来自ROB的重定向信号有效 |
| io\_redirect\_bits | I | Redirect | ROB | ExuBlock | 详见《重要Bundle描述》的Redirect |
| io\_allocPregs\_X | I | ResetPregStateReq |  |  | X 的取值范围 [0, RenameWidth)  详见《重要Bundle描述》的ResetPregStateReq |
| io\_in\_X\_valid | I | 1 | DispatchQueue | Dispatch2Rs | 来自 X 端口的 uop 有效。  X 的取值范围 [0, IntDqDeqWidth+LsDqDeqWidth)。下同。 |
| io\_in\_X\_ready | O | 1 | Dispatch2Rs | DispatchQueue | scheduler 可接收 X 端口的 uop |
| io\_in\_X\_bits | I | MicroOp | DispatchQueue | Dispatch2Rs | 详见《重要Bundle描述》的MicroOp |
| io\_issue\_X\_valid | O | 1 | ExuBlock | MemBlock | 第 X 个发射端口的访存指令发射信号有效。  X 的取值范围 [0, numOutFu) |
| io\_issue\_X\_ready | I | 1 | MemBlock | ExuBlock | 相应访存单元可以接收新的指令 |
| io\_issue\_X | O | ExuInput | ExuBlock | MemBlock | 指令执行相关信息  详见《重要Bundle描述》的ExuInput |
| io\_fastUopOut\_X\_valid | O | 1 | ExuBlock | XSCore | X 的取值范围 [0, numIssuePorts)。  用于PerfCounter统计。 |
| io\_rfWriteback\_X\_valid | I | 1 | WbArbiter | ExuBlock | 第 X 个ExuBlock外模块的写回信号有效。  X 的取值范围 [0, numIntRfWritePorts + numFpRfWritePorts)。  下同。 |
| io\_rfWriteback\_X\_bits | I | ExuOutput | WbArbiter | ExuBlock | 第 X 个ExuBlock外模块的写回信号。  详见《重要Bundle描述》的ExuOutput |
| io\_fastUopIn\_X\_valid | I | 1 | MemBlock | RS | 第 X 个 fastUopIn 信号有效。此信号来自 MemBlock，表明该 uop 执行完毕可以写回 pdest。 |
| io\_fastUopIn\_X\_bits | I | MicroOp | MemBlock | RS | 第 X 个 fastUopIn信号。  详见《重要Bundle描述》的MicroOp |
| io\_scheExtra | I/O | SchedulerExtraIO | - | - | 详见《重要Bundle描述》的SchedulerExtraIO |
| io\_scheExtra\_rsReady\_X | O | 1 | ExuBlock | CtrlBlock | 对端未使用。 |
| io\_scheExtra\_feedback\_X\_feedbackSlow | I | Valid[RSFeedBack] | MemBlock | ExuBlock | X 的取值范围{0, 1, 2, 3}  {0, 1}来自LoadUnit，{2, 3}来自StoreUnit。  详见《重要Bundle描述》的MemRSFeedbackIO。 |
| io\_scheExtra\_feedback\_X\_feedbackFast | I | Valid[RSFeedBack] | LoadUnit | ExuBlock | X 的取值范围{0, 1}。详见《重要Bundle描述》的MemRSFeedbackIO。 |
| io\_scheExtra\_fpRfReadIn\_X\_addr | O | 8 | ExuBlock | ExuBlock\_1 | X 的取值范围{0, 1}。读取浮点寄存器的地址。 |
| io\_scheExtra\_fpRfReadIn\_X\_data | I | XLEN | ExuBlock\_1 | ExuBlock | X 的取值范围{0, 1}。读取浮点寄存器的数据。 |
| io\_scheExtra\_loadFastMatch\_X | O | 2 | ExuBlock | LoadUnit | X 的取值范围{0, 1}。详见《重要Bundle描述》的SchedulerExtraIO。 |
| io\_scheExtra\_loadFastImm\_X | O | 12 | ExuBlock | LoadUnit | X 的取值范围{0, 1}。详见《重要Bundle描述》的SchedulerExtraIO。 |
| io\_scheExtra\_jumpPc | I | VAddrBits | CtrlBlock | ExuBlock | 详见《重要Bundle描述》的SchedulerExtraIO。 |
| io\_scheExtra\_jalr\_target | I | VAddrBits | CtrlBlock | ExuBlock |  |
| io\_scheExtra\_stIssuePtr\_\* | I | SqPtr | StoreQueue | ExuBlock |  |
| io\_scheExtra\_lcommit | I | 3 | ROB | ExuBlock |  |
| io\_scheExtra\_scommit | I | 3 | ROB | ExuBlock |  |
| io\_scheExtra\_lqCancelCnt | I | 7 | LoadQueue | ExuBlock/LsqEnqCtrl |  |
| io\_scheExtra\_SqCancelCnt | I | 7 | StoreQueue | ExuBlock/LsqEnqCtrl |  |
| io\_scheExtra\_memWaitUpdateReq\_\* | I | MemWaitUpdateReq | ExuBlock  MemBlock | ExuBlock | 详见《重要Bundle描述》的MemWaitUpdateReq。 |
| io\_fuExtra | I/O | FUBlockExtraIO | - | - | 详见《重要Bundle描述》的FUBlockExtraIO |
| io\_fuExtra\_exuRedirect\_\* | O | Valid[ExuOutput] | FUBlock | CtrlBlock |  |
| io\_fuExtra\_csrio\_hartId | I | - | 核外 | ExuBlock/CSR | 详见《重要Bundle描述》的CSRFileIO |
| io\_fuExtra\_csrio\_perf\_perfEventsFrontend\_\* | I | - | Frontend | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_perf\_perfEventsCtrl\_\* | I | - | CtrlBlock | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_perf\_perfEventsLsu\_\* | I | - | MemBlock | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_perf\_perfEventsHc\_\* | I | - | 核外 | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_perf\_retiredInstr | I | - | CtrlBlock | ExuBlock/CSR |  |
| exuBlocks\_io\_fuExtra\_csrio\_fpu\_\* | I | - | ROB | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_fpu\_frm | O | - | ExuBlock/CSR | ExuBlock\_1 |  |
| io\_fuExtra\_csrio\_exception\_\* | I | - | ROB | CSR |  |
| io\_fuExtra\_csrio\_isXRet | O | - | ExuBlock/CSR | CtrlBlock/ROB |  |
| io\_fuExtra\_csrio\_trapTarget | O | - | ExuBlock/CSR | CtrlBlock/ROB |  |
| io\_fuExtra\_csrio\_interrupt | O | - | ExuBlock/CSR | CtrlBlock/ROB |  |
| io\_fuExtra\_csrio\_wfi\_event | O | - | ExuBlock/CSR | CtrlBlock/ROB |  |
| io\_fuExtra\_csrio\_memExceptionVAddr | I | - | MemBlock | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_externalInterrupt\_\* | I | - | DebugModule  CLINT  PLIC | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_distributedUpdate\_0\_w\_\* | I | - | MemBlock | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_distributedUpdate\_0\_w\_\* | I | - | Frontend | ExuBlock/CSR |  |
| io\_fuExtra\_csrio\_tlb\_\* | O | - | ExuBlock/CSR | TLB |  |
| io\_fuExtra\_csrio\_customCtrl\_l1I\_\* | O | - | ExuBlock/CSR | Frontend  MemBlock |  |
| io\_fuExtra\_csrio\_customCtrl\_l2\_\* | O | - | ExuBlock/CSR | Frontend  MemBlock |  |
| io\_fuExtra\_csrio\_customCtrl\_l1D\_\* | O | - | ExuBlock/CSR | Frontend  MemBlock |  |
| io\_perf\_X\_value | O | 6 |  |  |  |

1. 模块设计

NA

1. PPA

NA

1. 验证关注点

NA

1. Floorplan 建议

NA

1. 遗留问题

NA