

# RV32 Processor Design Specification

## Revision History

Remark	Author	Date
初始化 Repo，设置好 git 环境	刘恒雨	2025/03/26
...	...	...

## Abbreviations

- **LE**: Load Elimination – 合并不同核间 load 请求的技术
- **SRS**: Self-Redefined-Streaming – 使得 MMA 指令能够复用其 src 物理寄存器的技术

## Detailed Revision Outline

1. Teminology
2. Overview
3. Parameters
4. Interface
5. Micro-architecture
  - 5.1. baseline 重命名
    - 5.1.1. FreeList
    - 5.1.2. RAT
    - 5.1.3. RST
    - 5.1.4. Top
  - 5.2. Self-Redefined-Streaming Register Rename (SRS)
    - 5.2.1. RST
    - 5.2.2. Top
  - 5.3. Load Elimination (LE)
    - 5.3.1. LRT
    - 5.3.2. Top

## Terminology

(在此处定义相关术语)

## Overview

重命名模块负责将逻辑寄存器号翻译为物理寄存器号，以消除 WAW 与 WAR 冲突。对于多核系统，多个核的重命名模块将合并为一个。通过参数可配置是否启用 SRS、LE 等技术，目前的实现采用 Verilog（也可考虑使用 Chisel）。在本设计中，所有核的物理寄存器号将统一编号，例如 core0 使用 0-7 号，core1 使用 8-15 号，依此类推。

## Parameters

---

- **LE**: 当其置为 1 时，启用 LE 技术
- **SRS**: 当其置为 1 时，启用 SRS 技术
- **CORE\_NUM**: HOST CPU 的数量
- **PREG\_PER\_CORE**: 每个 HOST CPU 配备的物理 matrix 寄存器的数量
- **LREG\_PER\_CORE**: 每个 HOST CPU 配备的逻辑 matrix 寄存器的数量
- **USE\_STATIC\_RENAME**: 如果启用，将直接 bypass 掉 rename 级，并在指令槽处检查 WAW 与 WAR 冲突
- **releaseWidth**: 每个周期 RST 能向 FreeList 释放多少个 preg
- **LRTPAWidth**: 在 LRT 中每个 PA 使用的位宽，可探索面积效率比

## Interface

---

Rename 核心部件的输入 Interface 定义为 Flipped(Decoupled(Vec(CORE\_NUM, LRegInfo))) 输出 Interface 定义为 Flipped(Decoupled(Vec(CORE\_NUM, PRegInfo))).

(下表展示 **LRegInfo** 与 **PRegInfo** 的详细定义，具体内容请参照设计文档)

## Micro-architecture

---

每个核配备一个 FreeList，用于存储可分配的物理寄存器号，对 dest reg 进行分配；同时配备一个 Register Alias Table (RAT) 保存 lreg 与 preg 的映射；全局配备一个 Register Status Table (RST) 管理各核物理寄存器的状态。启用 LE 时，还会配备一个全局 Load Register Table (LRT) 用于记录 load 指令的相关信息。

### baseline 重命名

---

描述 baseline 重命名的架构和信号流程。

### FreeList

---

FreeList 是一个多输入多输出的队列，其关键参数包括：

- **inputWidth**: 单周期内从队列中 pop 出表项的数量
- **outputWidth**: 单周期内向队列中 push 进表项的数量
- **entryType**: 队列中存放数据的类型
- **depth**: 队列深度
- **bypassI2O**: 当同时输出和输入时，是否采用 advance 后的指针判断 push 能否进行
- **initData**: 队列初始化数据
- **initHead**: 队列 head 的初始位置
- **initTail**: 队列 tail 的初始位置

其 IO 定义示例如下：

// 示例：FreeList IO 定义

```
class MultiIOQueueIO[T <: Data](entryType: T, inputWidth: Int, outputWidth: Int)
```

```
extends Bundle {  
    val in = Flipped(Vec(inputWidth, Decoupled(entryType.cloneType)))  
    val out = Vec(outputWidth, Decoupled(entryType.cloneType))  
}
```

## RAT

---

RAT 是一个包含 LREG\_PER\_CORE 个 entry 的全连接表，用于记录 lreg 到 preg 的映射。要求：在同一周期内完成数据读取；若在单周期内对同一表项进行读写，写入数据不 bypass 到读取端。

## RST

---

RST 包含一个大小为 PREG\_PER\_CORE \* CORE\_NUM 的表，用于管理所有核的物理寄存器状态。主要任务包括：

- 更新 lastConsumerId
- 释放完成的寄存器到 FreeList
- 管理信号：allocated、been\_redefined、haveConsumer、lastConsumerCommitted 等

## Top

---

Top 模块整合 FreeList、RAT、RST 及其他模块接口，处理重命名请求。对于某一重命名请求，只有满足如下条件时才视为处理成功：

- 不需要重新分配物理寄存器；或
- (物理寄存器分配成功 且 RST 接受了其更新请求)

## Self-Redefined-Streaming Register Rename (SRS)

---

在 baseline 重命名基础上，SRS 模块主要修改顶层连接及 RST 行为，使得在满足条件时能够复用 preg。SRS 的条件包括：

- 指令的 src lreg 与 dest lreg 相同，且两者均为 valid
- 对应的 src lreg 所映射的 preg 的 haveConsumer 为 0
- 该 preg 属于当前核
- 输入请求为 valid

当 SRS 生效时，Top 模块会调整输出的 dest preg，并 bypass FreeList 分配。

## RST (SRS 模式)

---

RST 新增 IO 用于判断 SRS 条件，且在 SRS 生效时重置对应 preg 表项（除了 allocated 信号）。

## Top (SRS 模式)

---

当收到 SRS 有效信号时，Top 模块：

- 将相应 FreeList 的 allocate 端口 ready 置为 0
- 修改输出响应中的 dest preg 为 RST 提供的 preg 标号

- 无论分配是否成功，都输出 valid 响应

## Load Elimination (LE)

---

LE 模块通过增加 Load Register Table (LRT) 来记录每个 preg 是否来自合法 load 指令，以及 load 指令是否已 commit。

## LRT

---

LRT 表项包含以下信息：

- canMerge: 标识该表项是否来自合法 load 指令
- dataReady: 表示 load 指令是否已 commit
- VA: load 指令的 base 虚拟地址
- stride: load 指令的 stride
- configReg: load 指令的 size 信息
- PA: 对应的物理地址，每行一个
- PA\_updt\_cnt: PA 更新计数器

更新流程包括：

- 根据比对结果更新 stride、VA，并调整 canMerge 与 dataReady 信号
- 接收到 commit 信息时，如对应表项 canMerge 为真，则拉高 dataReady
- 处理 write\_snoop 信号，进行无效化
- 利用 translation\_done 信号更新 PA 信息
- 接收到 RST 的 release 信号时，拉低 canMerge

## Top (LE 模式)

---

在 LE 模块接入后，Top 模块：

- 从 LRT 和 FreeList 获取更新信息以更新 RAT
- 修改 FreeList 的 allocate ready 信号（当 srs\_grant 或 lrt\_grant 有效时）
- 使用最终重命名结果（final\_result）更新 RST

## Conclusion

---

以上 Typst 示例代码及章节结构概括了 Matrix CA Rename Specification 的主要内容，包括各模块的接口定义、参数说明以及关键代码框架。请根据项目的实际需求进一步细化各部分设计与实现。

利用 **Typst** 轻松实现图纸生成功能

---

