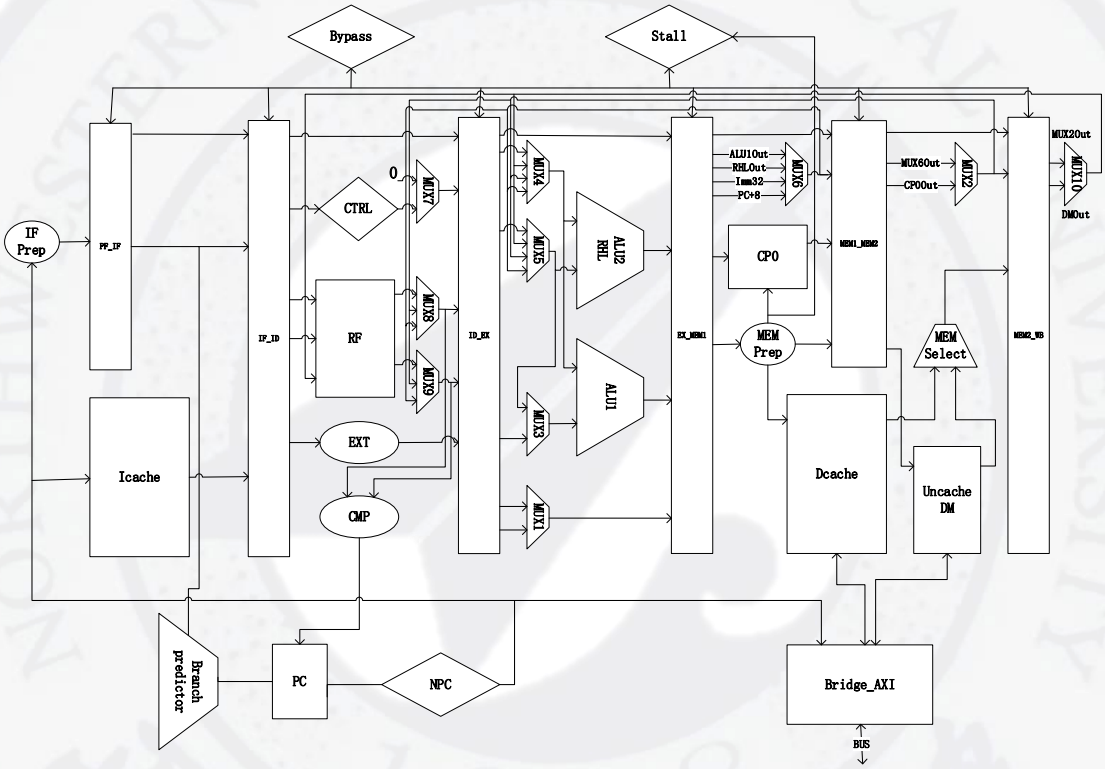
西北工业大学 1 队

# SuboltMIPS CPU 设计报告




王翰墨　王玉佳　吴奇　杨士欣

# 目录

第一章　项目概览 3

1.1 总述 3

1.2 性能 3

1.3 指令集 3

1.4 结构 3

第二章　CPU 设计 4

2.1 整体架构设计 4

2.2 CP0 5

2.3 异常处理 5

2.4　分支预测 5

2.5 高速缓存设计 7

2.6 AXI 总线桥 9

附录 A 参考文献 10

# 第一章 项目概览

## 1.1 总述

- 实现 MIPS I 基准指令集：57 条指令，部分中断例外
- 实现 Cache
- 支持复杂总线接口：类 SRAM 总线、AXI 总线
- 运行随机延迟功能测试通过
- 运行记忆游戏测试通过
- 运行性能测试通过
- 运行系统测试通过（带中断）

## 1.2 性能

CPU 主频 150MHZ，性能分 78

## 1.3 指令集

实现了初赛要求的所有指令

## 1.4 结构

7 级单发射流水线

# 第二章 CPU 设计

## 2.1 整体架构设计

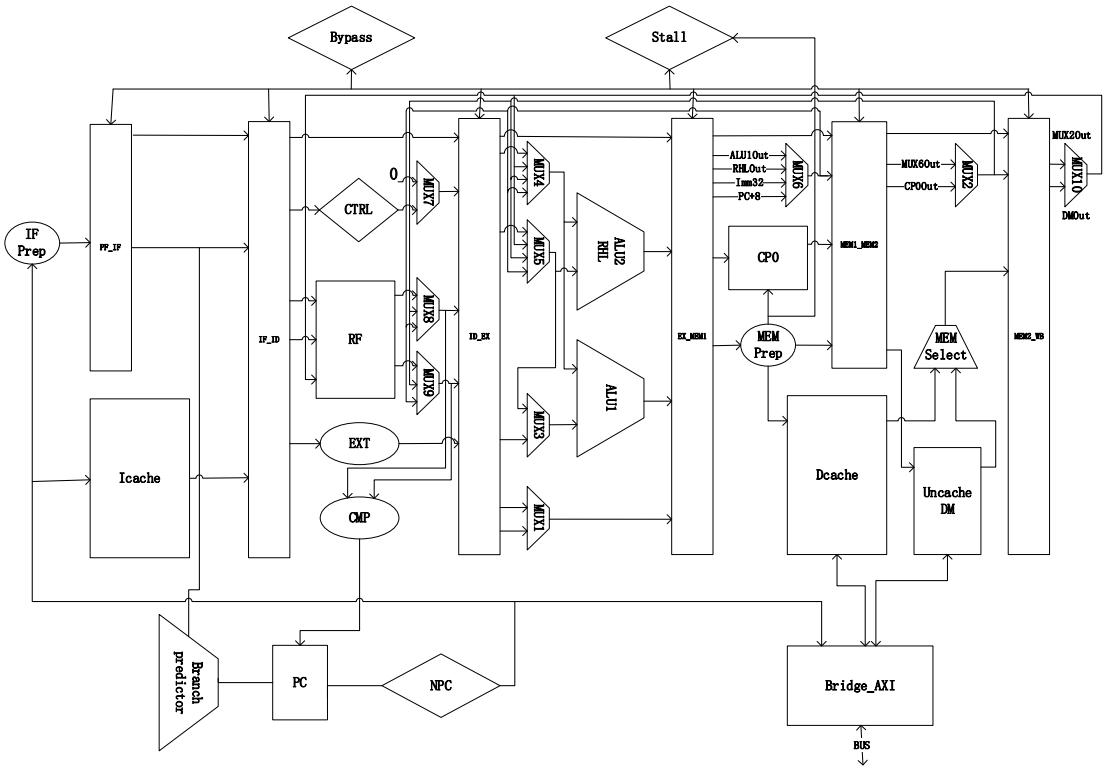



**图 2.1 CPU 数据通路图**

我们的 CPU 是一个 7 级流水线 CPU，分别是 PF（预取指）、IF（取指）、ID（译码）、EX（执行）、MEM1（访存第一阶段）、MEM2（访存第二阶段）、WB（写回阶段）。

- PF：计算出 NPC 的值，完成将取指地址送入 Icache 的操作。
- IF：根据 IF-Prep 送入的地址取出对应的指令（针对 Icache）；将 IF_PC 送入分支预测模块，得到分支信号以及分支地址并送入 PC。
- ID：完成对指令的译码，并生成各种控制信号往后传输；完成对分支指令的源操作数的比较；完成对 16 位立即数的扩展。
- EX：完成加减乘除移位等指令的运算。对分支模块进行更新
- MEM1：对 CP0 寄存器进行访存，报出例外；根据 ALU 算出的地址送入 Dcache
- MEM2：根据 MEM1 送入的地址取出对应的数据（针对 Dcache）；对访存地址进行判断，根据对应的结果决定访存是走 Dcache 模块还是 Uncache_DM 模块（访问外设）。
- WB：将要写回寄存器的数据写回寄存器堆。

2.2 CP0

表 2.2 实现的 CP0 寄存器

| *寄存器名称* | *功能定义* |
|---|---|
| *Count* | 处理器时钟计数器 |
| *Compare* | 计时器中断控制 |
| *Status* | 处理器状态与控制寄存器 |
| *Cause* | 存放上一次例外原因 |
| *EPC* | 存放上一次发生例外指令的 PC |

2.3 异常处理

表 2.3 实现的异常处理

| EXCODE | 助记符 | 描述 |
|---|---|---|
| 0x00 | Int | 中断 |
| 0x04 | AdEL | 地址错例外（读数据或者取指令） |
| 0x05 | AdES | 地址错例外（写数据） |
| 0x08 | Sys | 系统调用例外 |
| 0x09 | Bp | 断点例外 |
| 0x0a | RI | 保留指令例外 |
| 0x0c | Ov | 算术溢出例外 |

- 在 PF 级检测取指令的地址错例外，在 ID（译码）级检测系统调用例外、断点例外、保留指令例外，在 MEM1 级检测算术溢出例外、读数据和写数据的地址错例外。
- 将中断视作一种特殊的例外，检测到中断时将这种例外标记在 MEM1 级的指令上。
- 所有例外（包括中断）统一在 MEM1 级报出

2.4分支预测

2.4.1 设计思路

在没有分支预测的情况下，分支指令在 ID 级才能确定分支方向和分支目标地址，此时 IF 级是一条分支延迟槽指令，而 PF 级就很有可能是一条要被刷新的指令。最初我们是预测分支总是不发生，故若分支发生，那么将会损失一个周期的代价。理想情况是当分支指令在 IF 级时就可以知道它的分支方向和目标地址，因此我们将分支预测单元设置在了 IF 级，包括分支方向预测和分支目标地址预测。

### 2.4.2 分支方向预测

采用两位饱和计数器和 PHT 结合的方法进行分支预测

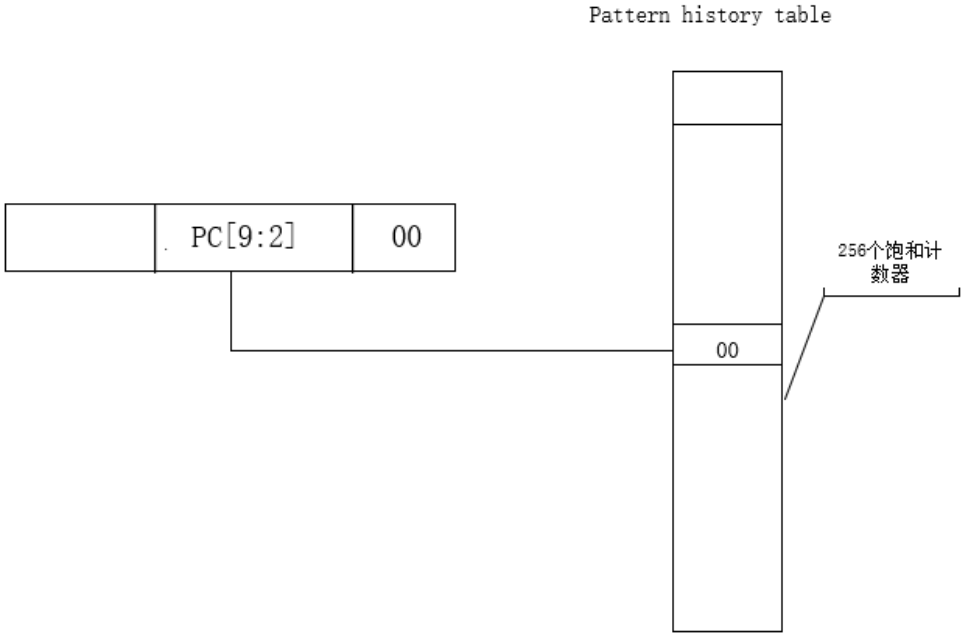



**图 2.4.1 PC 寻址 PHT 中的饱和计数器**

用 PC 中的 8 位对 PHT 进行寻址，因此 PHT 中一共有 256 个饱和计数器，每一个 PC 都有对应的饱和计数器

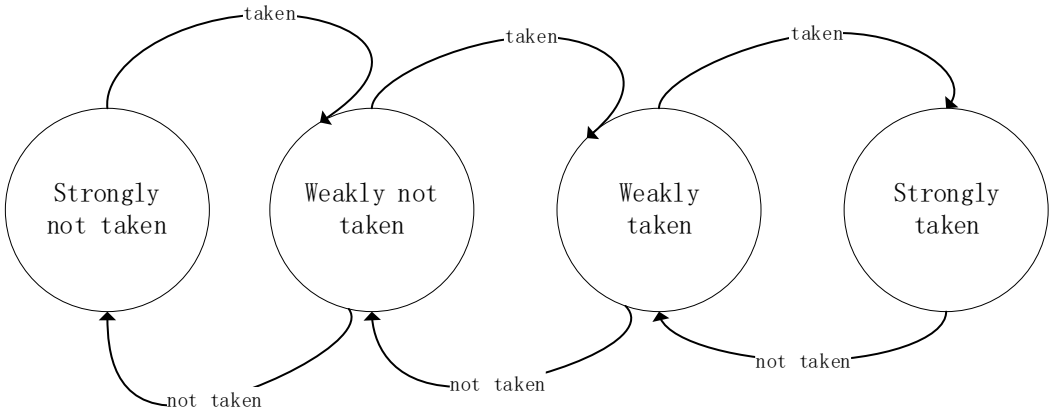



**图 2.4.2 方向预测单元状态机**

设置一个有限状态机，初始状态为 strongly not taken，由状态机示意图可以看出，当某条分支指令连续两次发生跳转，那么下一次必定预测它跳转，反之一样。

### 2.4.3 分支目标地址预测

采用 BHT 和 RAS 结合对目标地址进行预测。对于直接跳转指令和除了 return 指令外的间接跳转指令，将他们的跳转地址存放在 BHT 中，对于 return 指令（JR $31），每当遇到一条 call 指令，就将 call 指令所在地址的 PC+8 存放到 RAS 中，并将栈指针加 1，每当遇到一条 return 指令，就从 RAS 的栈顶取出地址，并将栈指针减 1，从而完成对 return 指令的预测。

## 2.5 高速缓存设计

在流水线中我们设计了两个一级高速缓存，分别是指令 CACHE 和数据 CACHE，大小均为 8KB，采用二路组相联，组大小 64，Cache Line 大小为 64B。这两个模块分别为 ICACHE 和 DCACHE，其中 ICACHE 是 DCACHE 的缩减版，删去了写的相关功能。下面主要介绍 DCACHE 的完整功能。

DCACHE 需要实现四个方向的数据交互：①从流水线到 cache；②从 cache 到流水线；③从总线到 cache；④从 cache 到总线。由于 cache 是使用 LUTRAM 和寄存器结合来实现的，以求得良好的时序，因而一条访存指令需要两拍才能完成（在不缺失的情况下），因而可以将第一拍作为发起请求，在 MEM1 级执行；第二拍作为返回结果/完成操作，在 MEM2 级执行。在一开始，cache 可以处于 IDLE 空闲状态，表示不在工作，而当有连续的访存指令时，前一拍指令在 MEM2 级，对应 LOOKUP 状态，如果不缺失，那么下一拍继续就保持在 LOOKUP，下一拍会将当前 MEM1 级的后序请求送过来，实现流起水来的效果。

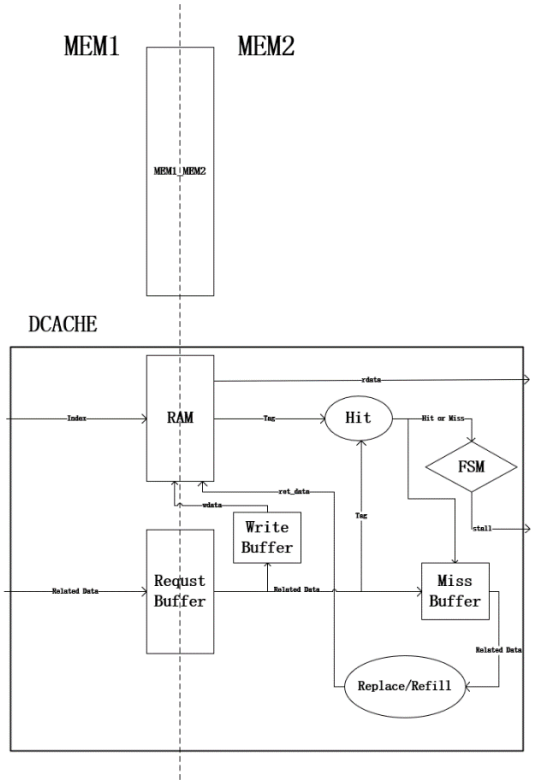



**图 2.5.1 DCache 内部数据通路示意图**

从上图中可以看出，Request Buffer 起到了和 MEM1_MEM2 相同的作用，将两个流水级清晰地分隔开，方便进行访存第一步和第二步的相应操作，而 Miss Buffer 则是在 cache 未命中时才启用的，专门用于脏块替换和处理缺失的。如此，若发生 cache 缺失，则将对应的访存指令阻塞在 MEM2 级，等到走完 MISS->REPLACE->REFILL 后重新回到 LOOKUP，此时 cache 应当命中，并进行相应的操作。下图是 DCACHE 内部的状态机。

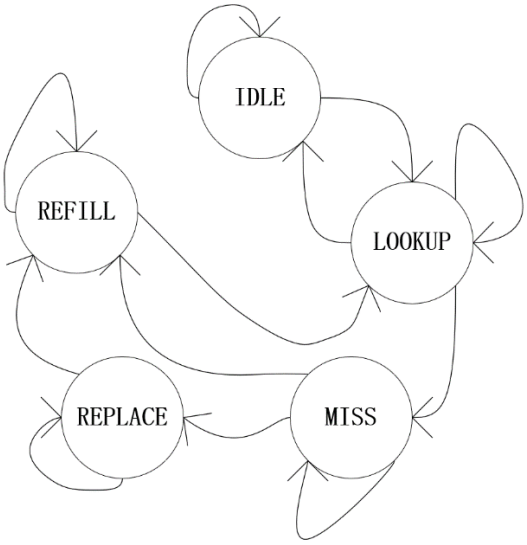



**图 2.5.2 Cache 有限状态机示意图**

对于 store 指令设置了一个 Write Buffer 作为缓冲使其不成为关键路径，存储 Tag 和 Data 的 LUTRAM 都采用读地址和写地址分开的方法，且采用写优先策略，针对 SW/SH/SB 指令，先将该地址对应的字读出来，再在 Write Buffer 后进行拼接，最终以字为单位写入 RAM 中。这样是为了更好地旁路，针对 SB 指令后紧接 LW 指令时避免了阻塞，此外也可以避免读写地址冲突带来的额外阻塞。

## 2.6 AXI 总线桥

该功能主要由 axi_sram_bridge 模块实现。此模块负责 CPU 与外部设备按照 AXI 接口协议进行数据交换，同时承担起了仲裁、对不同地址段地址按照读写要求设置不同信号的功能。此模块主要由两个状态机控制。

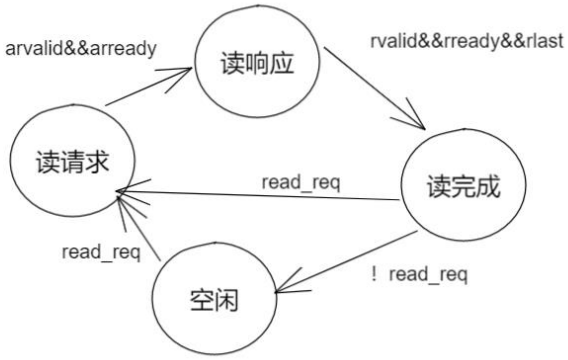



**图 2.6.1 读状态机**

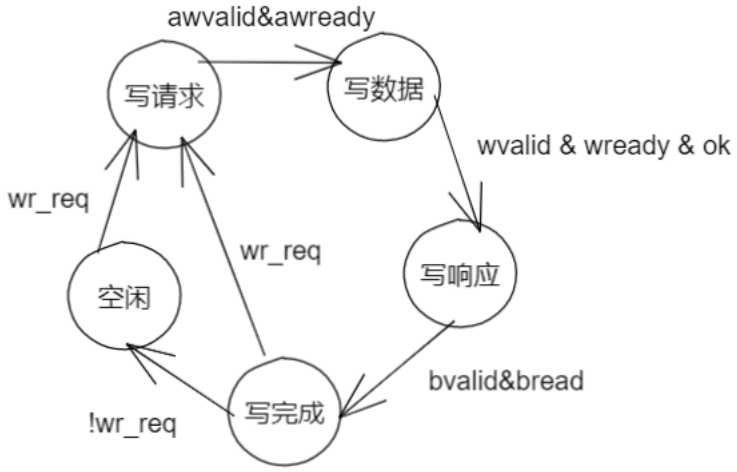



**图 2.6。2 写状态机**

该模块接受来自 CPU 的类 sram 读写请求，通过译码生成相应的 axi 总线请求。

- 对 uncache 段的读写请求，len 设置为 4‘b0，即数据仅仅传输 1 拍。burst 突发模式设置为 fixed，一次读写的数据大小 size 按照 CPU 发送来的对应信号进行译码得到。
- 对 cache 段的读请求，len 设置为 4’b1111，即数据传输 16 拍，burst 设置为 increase 模式，size 设置为 4 个字节。
- 对于 cache 段的写请求，cache 将 16 个字一起交给 axi_sram_bridge 模块中的 buf，然后由写状态机控制，依次向总线中写入这 16 个字。

# 附录 A 参考文献


- David A. Patterson, John L. Hennessy. 计算机组成与设计：硬件/软件接口（第 5 版）. 王党辉等 译. 机械工业出版社
- 姚永斌. 超标量处理器设计. 清华大学出版社
- 汪文祥，邢金璋. CPU 设计实战. 机械工业出版社