实验 4 - L1 cache 设计

浙江大学计算机体系结构实验

DDL: 2022.11.30 23:59

1 实验目的

- 了解 cache 在 CPU 中的作用
- 了解 cache management unit(CMU) 与 cache 和 memory 之间的交互机制
- 在 CPU 中集成 cache

2 实验环境

- **HDL**：Verilog
- **IDE**：Vivado
- **开发板**：NEXYS A7 (XC7A100T-1CSG324C) 或 Sword 4.0 (XC7K325T-2FFG676)

3 实验原理

3.1 Cache

cache 作为 CPU 和内存之间的存储结构，能够利用其速度快、容量小的特点，在速度相差较大的两种硬件之间，起到协调两者数据传输速度差异的作用，是 CPU 存储层次中的重要组件。

本次实验要求 cache 实现 **2 路组相联**，采用 **write-back,write-allocate** 的数据更新策略，采用 **LRU** 的替换策略。这些内容相信在理论课上都已经有详细的描述，在此不再过多的展开。cache block 如图1所示。此外，本次实验实现的 Cache 大小为 1024B，每个块有 16B，因此合计有 1024/16/2=32 个 set。

LRU	V	D	Tag	Data

图 1: cache block

3.2 CMU

CMU 实质上是把 cache 中的状态机部分和与 CPU, Memory 的交互部分独立出来，作为一个控制单元，控制数据的处理，为了方便 CMU 的控制，本次实验又把数据处理部分单独列了出来。CMU 基本的架构与交互模式如图2所示，流水线如图5所示。

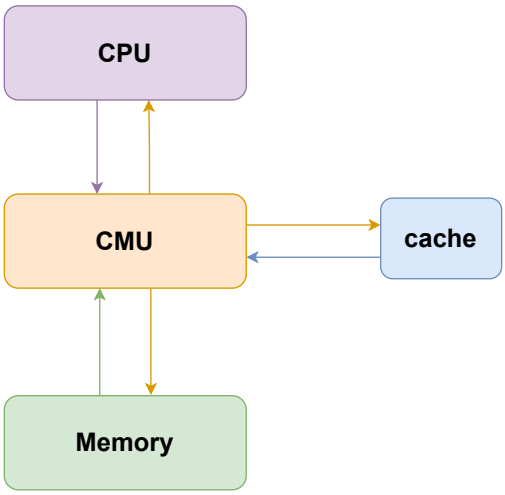


图 2: 架构

正如之前所述，CMU 是一个专门的控制模块，用于控制数据的读写，对于其控制逻辑我们一般采用状态机的模式来管理，当然也是可以像 CPU 的 datapath 一样用流水线来实现 CMU。下面图3给出了 CMU 的状态机。

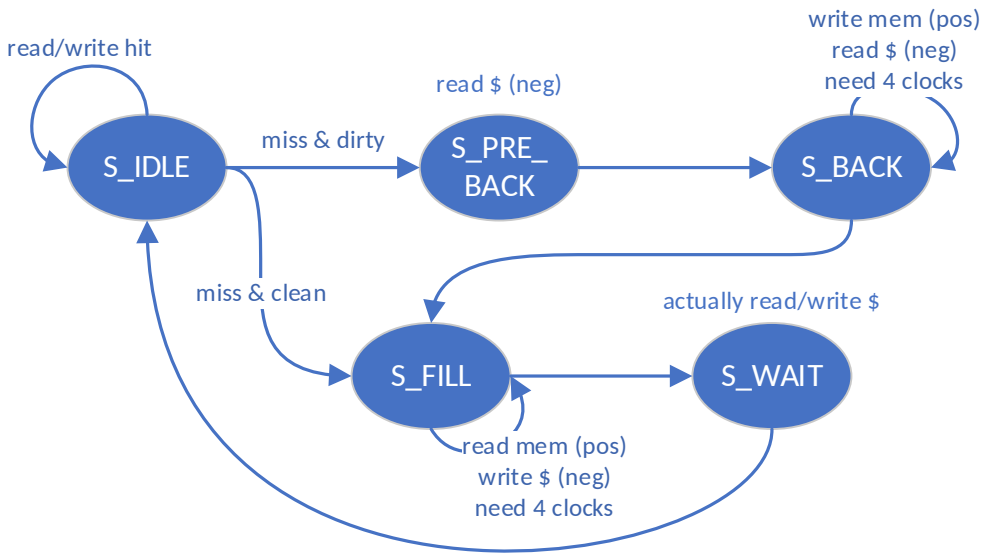


图 3: CMU 状态机

CMU 的状态机为 5 个状态，各个状态的功能为描述如下：

1. S_IDLE: cache 正常读写，即 load/store 命中或者未使用 cache。
2. S_PRE_BACK: 为了写回，先进行一次读 cache
3. S_BACK: 上升沿将上个状态的数据写回到 memory，下降沿从 cache 读下次需要写回的数据（因此最后一次读无意义），由计数器控制直到整个 cache block 全部写回。由于 memory 设置为 4 个周期完成读写操作，因此需要等待 memory 给出 ack 信号，才能进行状态的改变。
4. S_FILL: 上升沿从 memory 读取数据，下降沿向 cache 写入数据，由计数器控制直到整个 cache block 全部写入。与 S_BACK 类似，需要等待 ack 信号。
5. S_WAIT: 执行之前由于 miss 而不能进行的 cache 操作。

上述状态要遵循的基本逻辑流程即为图4中展示的。总体来说，CMU 处理的事务主要包括三件：读、写、替换。

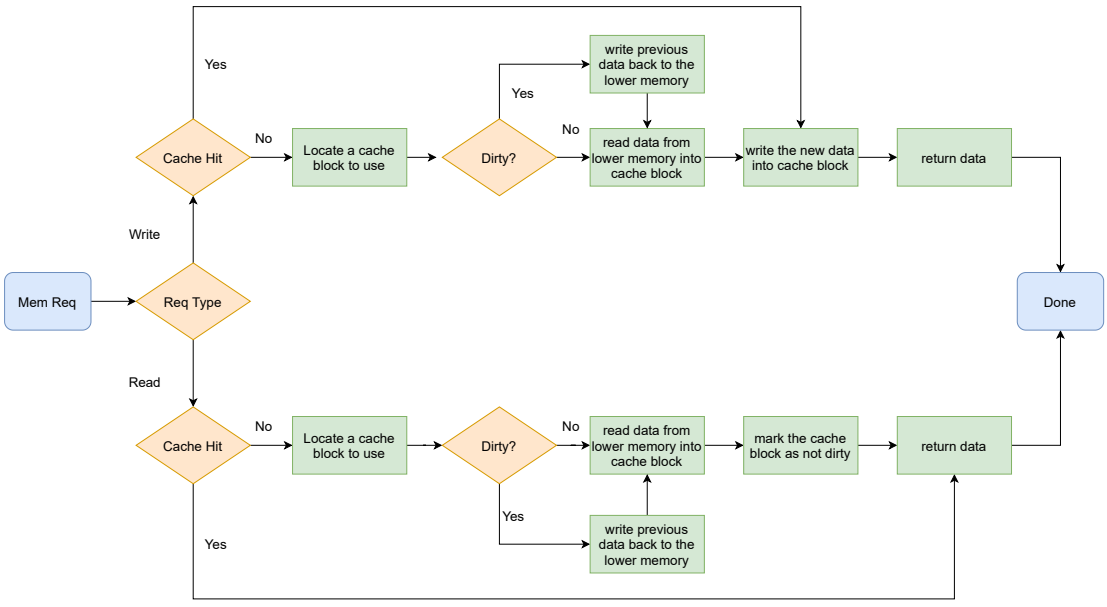


图 4: CMU 读写事务流程

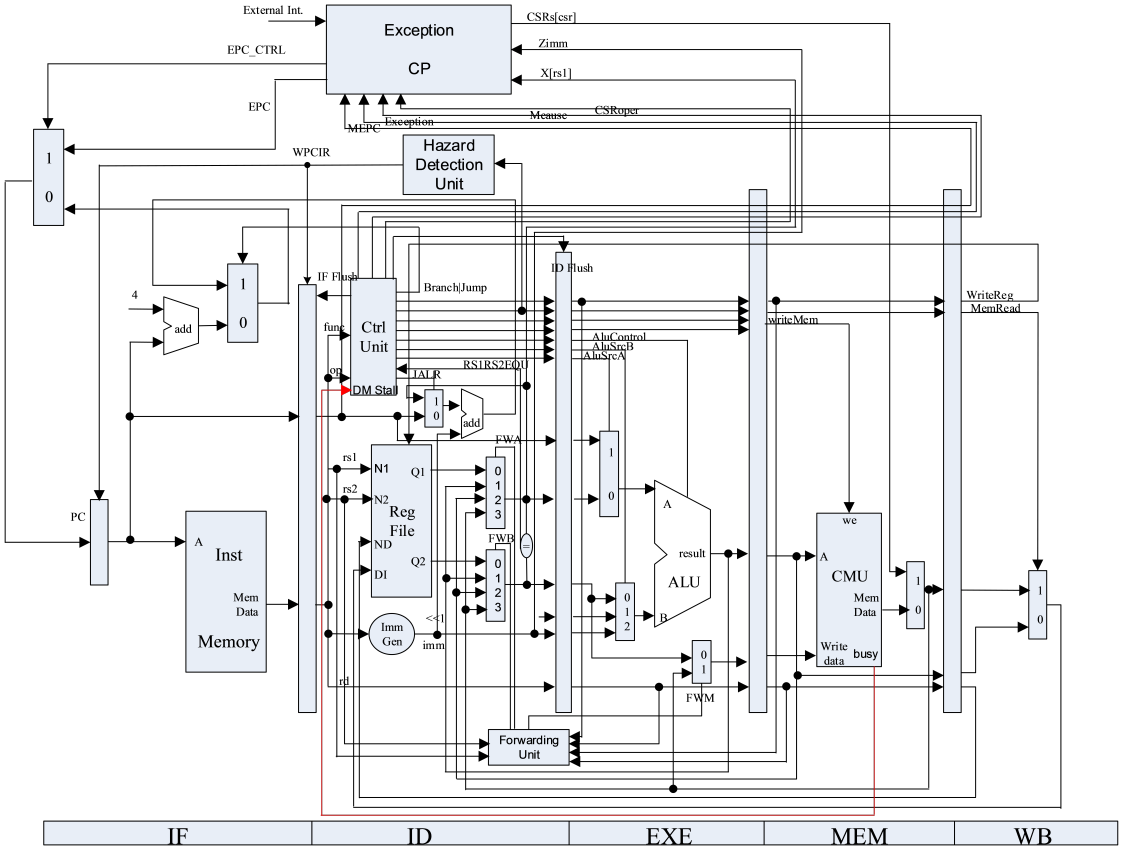


图 5: datapath

4 实验要求

1. cache 和 CMU 要求采取 write-back, write-allocate,LRU 的策略
2. 通过仿真测试和上板验证

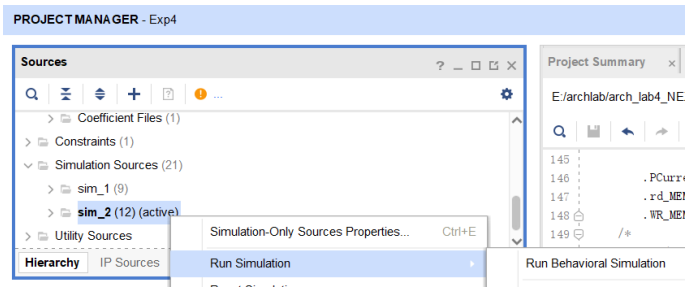
5 实验步骤

5.1 总体步骤

1. 理解所给代码框架的 cache 和 CMU 模块（不懂的地方问助教）
2. 补全 cache 和 CMU 模块的代码
3. 在给定的 SoC 中，加入自己的 CPU，通过仿真测试和上板验证

5.2 仿真验证

1. 本次实验有两个仿真，一个仅用于仿真 CMU 和 Cache 模块，对应于 code/cache/sim/sim_top.v；另一个用于仿真完整的 CPU，对应于 code/sim/core_sim.v。
2. 前者会仿真一些访存操作，后者则和以往一样，运行简单的测试程序。
3. CPU 的测试仅有 18 条指令，所以请仔细核对以确保结果正确。
4. 推荐先对模块仿真，然后再对整体仿真。
5. 若仿真 CMU 和 Cache 模块，在 Sources 中的 Simulation Sources 目录中右键 sim_1 选择运行仿真；若仿真完整 CPU，则在 sim_2 上右键运行仿真。



6. 如果想先熟悉下正确的测试结果，可以先把 Cache 去掉，使用旧版内存，具体操作为：将 RV32Core 的 CMU 和 RAM 注释掉，同时把下方 “RAM_B data_ram ...” 的注释打开，最后把 cmu_stall 置为 0（**重新使用 Cache 时记得恢复**）。

5.3 上板验证

1. 使用 Sword 4.0 的同学可以将 SW13 拉高（对应 RV32Core 的 vga_sw 信号），这样会显示访存的地址和结果。
2. 现在 NEXYS A7 终于支持打印调试信息了，以下为使用说明：
   - 串口的连接和通信，以及在电脑显示输出的方法上次实验已经介绍，不再赘述
   - 使用时需要**把 SW8 拉高**并开启**单步调试模式**（SW0）
   - 每执行一步会输出一次调试信息，包括寄存器值、WB 阶段的 PC 和指令，以及访存的地址和结果，如图6所示
   - 如果想用以前的方法，即通过数码管查看其他信号值，将 SW8 拉低即可
   - 如果想添加别的信号，可查看 code/auxillary/debug_ctrl.v 并根据注释添加
   - 如果发现有 bug，欢迎指出:)

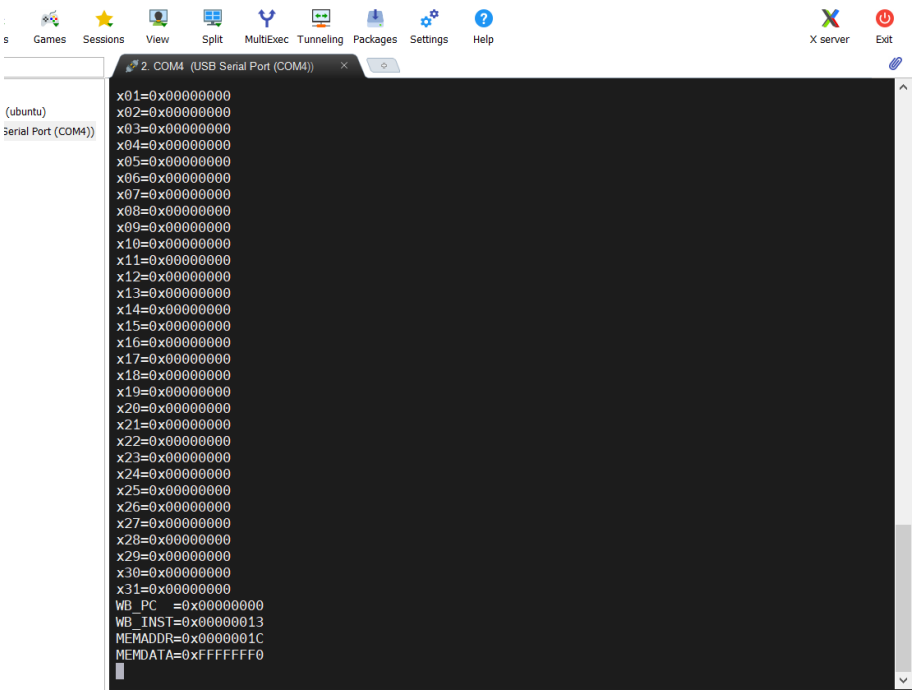


图 6: debug using UART

6 注意事项

1. NEXYS A7 已经支持通过 UART 在电脑输出调试信息，具体用法见“上板验证”章节
2. CMU&Cache 模块仿真参考结果见目录 ref/sim_ref
3. CPU 测试程序及内存数据的介绍见目录 ref/

7 思考题

1. 在实验报告分别展示缓存命中、不命中的波形，分析时延差异。
2. 在本次实验中，cache 采取的是 2 路组相联，在实现 LRU 替换的时候，每一个 set 需要用多少 bit 来用于真正的 LRU 替换实现?
3. 如果要实现 cache 的 4 路组相联，请描述一种真 LRU 和一种 Pseudo-LRU 的实现方式，并给出实现过程中每一个 set 需要用到多少 bit 来实现 LRU。关于 Pseudo-LRU，实现方式可以在网上查阅。

注：思考题写入实验报告内