计算机体系结构 Lab3

PB18111787 林焕承

实验目标

1. 权衡cache size增大带来的命中率提升收益和存储资源电路面积的开销
2. 权衡选择合适的组相连度（相连度增大cache size也会增大，但是冲突miss会减低）
3. 体会使用复杂电路实现复杂替换策略带来的收益和简单替换策略的优势（有时候简单策略比复杂策略效果不差很多甚至可能更好）
4. 理解写回法的优劣

实验环境和工具

操作系统：Windows 10

实验工具：Vivado 2019.2

实验内容和过程

阶段一

FIFO

对于 FIFO，换出时只要按 `0,1,2,...,WAY_CNT-1,0,1...` 顺序循即可。

换出 WAY 下标选择

```
swap_way_index <= way_index[set_addr];
way_index[set_addr] <= (way_index[set_addr] + 1) % WAY_CNT;
```

换出

```
mem_wr_addr <= {cache_tags[set_addr][swap_way_index], set_addr};
mem_wr_line <= cache_mem[set_addr][swap_way_index];
```

LRU

换出时，要将最长时间未使用的 WAY 换出。

可以维护一个队列，队列按访问时间最近排序，队尾即是需要被换出的 WAY，每次 hit_cache 时，将该 WAY 提到最前面。

队伍维护：

```
if(wr_req | rd_req)begin
    for(integer i = 0; i < queue_pt; i++)begin
        queue[set_addr][i+1] <= queue[set_addr][i];
    end
end
queue[set_addr][0] <= hit_way_index;
```

两种方法都再最终生成的 256 组测试数据通过。

阶段二

256 位数据的快排。

策略	WAY_CNT	miss_count	total_count	miss_rate(%)	性能(ns)
FIFO	4	91	6256	1.455	165460
FIFO	5	44	6256	0.703	145744
FIFO	6	41	6256	0.655	144264
LRU	4	86	6256	1.374	168064
LRU	5	42	6256	0.671	144896
LRU	6	41	6256	0.655	144264

16 阶矩阵乘法

策略	WAY_CNT	miss_count	total_count	miss_rate(%)	性能(ns)
FIFO	4	1739	8704	19.979	687906
FIFO	5	1328	8704	15.257	588341
FIFO	6	184	8704	2.113	304034
LRU	4	1712	8704	19.669	646434
LRU	5	700	8704	8.042	606482
LRU	6	126	8704	1.448	292758

阶段三

策略	WAY_CNT	LUT	FF	BRAM	IO
FIFO	3	3282	7337	4	81
FIFO	4	4579	9466	4	81
FIFO	5	5123	11491	4	81
FIFO	6	6231	13026	4	81
LRU	3	3301	7950	4	81
LRU	4	5173	10165	4	81
LRU	5	5783	12757	4	81
LRU	6	6980	15234	4	81

硬件资源随着组数增加而增加。

FIFO 策略对于上述两个算法性价比都比较高。对于快排算法可以设置 WAY_CNT 为 5 性价比较高，矩阵乘法设置成 6 性价比高。

实验总结

一些踩坑

- 原 CPU 实现有误。
- 统计数量时没有考虑 miss 持续很多周期，结果错误。

收获

熟悉了 cache 的实现，学到了各种 cache 换出策略的实现，

时间

阶段一，二各花了约 4 小时，不停修改各个模块的 bug。

阶段三都花了 2 小时左右，不断仿真、综合花费很多时间。