Verilog-lab3 实验报告

梁峻滔 PB19051175

实验目的

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联度(相联度增大cache size也会增大, 但是冲突miss会减少)
3. 体会使用复杂电路实现复杂替换策略所带来的收益和简单替换策略的优势(有时候简单策略效果不一定比复杂策略差)
4. 理解写回法的优劣

实验内容

阶段一

要求

理解所提供的直接映射策略的cache, 将其修改为N路组相联的cache, 并通过读写测试.

实现思路

1. 理解组相联

首先数据在memory和cache之间是按块传输, 一个块(line)包含多个字, 一个组包含多个块.

CPU访问数据Memory时会给出一个32位访存地址addr, 该地址划分成几部分: tag(用于判断cache中的块是否是想要访问的块), set_addr(指示块在哪个组), line_addr(块内字偏移), word_addr(字内字节偏移).

组相联cache和直接相联cache的区别就在于:

- 直接相联cache中一个组只有一个块, 所以访问cache判断是否命中时, 只需要根据set_addr比较valid位是否有效, 以及tag是否一样即可.
- 组相联cache中一个组内有多个块, 所以访问cache判断是否命中时, 需要比较set_addr对应的组内所有块的valid和tag(该过程可以串行也可以并行), verilog使用for循环是并行判断(因此不需要加break !!!).

因此需要将cache_mem, cache_tags, valid, dirty等原本对应set_addr的一维修改为二维. 在判断是否命中时记录命中时的组内偏移, 或者miss时要换出的块所在的组内偏移.

2. 两种替换策略的电路实现

主要思路: 维护cache中每个块的历史信息, 需要换出时根据各个块的历史信息选择换出的块. 对于FIFO策略, 该历史信息是每个块换入时的时刻到当前时刻的时间(时钟周期数); 对于LRU策略, 该历史信息是每个块自上一次被访问到当前的时间(时钟周期数). 对于这两种策略, 换出时选择历史信息值最大的块作为换出块.

对于FIFO, 实际上不需要每个时钟周期都更新历史信息, 因为只需要保持各个块历史信息的相对大小即可, 这样也可以减小历史信息记录溢出的机会. 只需要在某个块被换入时将其历史信息清零, 所有其他块的历史信息值加一.

对于LRU, 也不需要每个时钟周期都更新历史信息, 只需要在一个块被访问时将该块的历史信息清零, 将所有其他块的历史信息加一. 而一个块被访问只有两种情况: 命中, 以及缺失时换入, 这都可以通过命中判断和换入时的组内偏移来确定是要将哪个块的历史信息清零.

阶段二

要求

将阶段一实现的N路组相联cache接到Lab2所实现的CPU上(用本次实验的cache替换前面的DataCache), 并修改相应的数据通路使cache和CPU可以正常工作, 同时统计cache的缺失率.

实现思路

1. 修改CPU中的数据通路

需要修改的有三个模块: `WbData.v`, `Hazard.v` 和 `RV32ICore.v`.

- `WbData.v`: 用cache替换原来的DataCache, 需要设置一些中间变量来实现对接, 以及添加一些变量计算缺失率
- `Hazard.v`: 需要添加一个输入端口miss, 当cache发生缺失时需要停顿整个流水线
- `RV32ICore.v`: 添加一个miss中间变量, 修改以上两个模块的接口

2. 添加统计缺失率的电路

主要思路: 在`WbData.v`中统计miss次数miss_count和访存次数access_count, 缺失率就是miss_count / access_count.

由于miss信号和access信号(即读请求或写请求)一般会持续多个周期, 所以采用取边沿的方法, 统计对应的上升沿次数就是缺失次数和访存总次数.

数据分析

固定主存大小为4096个字(32位).

快速排序的cache性能和电路资源分析

由于我们分析的是cache的性能, cache的性能主要受缺失率影响, 我们可以用缺失率来表征性能. 电路资源主要关注LUT和FF的数量开销. 影响cache大小的参数有三个: 组的数量(2^SET_ADDR_LEN), 组相联度(WAY_CNT), 块的大小(2^LINE_ADDR_LEN). 暂不考虑块大小的影响, 保持块大小为8个字.

1. 不同cache大小的缺失率和电路资源

- 改变组的数量, 保持组相联度(WAY_CNT=4)和块大小(8个字)不变

  缺失率:

替换策略	4组(16块)	8组(32块)	16组(64块)	32组(128块)
FIFO	0.0930	0.0558	0.0355	0.0173
LRU	0.0930	0.0558	0.0355	0.0173

  电路资源:

4组(16块)	8组(32块)	16组(64块)	32组(128块)
LUT: 8.94%, FF: 4.90%	LUT: 8.96%, FF: 9.03%	LUT: 14.49%, FF: 17.26%	LUT: 22.82%, FF: 33.59%

  分析:

  替换策略的不同几乎没有影响. 组数越多, 缺失率越低(性能越好), 但相应地电路资源开销也会增加, 而且随着组数增加, 电路资源开销增加的程度大于缺失率降低的程度. 在从4组增加到8组时最划算.

- 改变组相联度, 保持组的数量(8组)和块大小(8个字)不变

  缺失率:

替换策略	2路(16块)	3路(24块)	4路(32块)	5路(40块)
FIFO	0.0558	0.0558	0.0558	0.0558
LRU	0.0558	0.0558	0.0558	0.0558

  注: 实际测出来就是这几个都是一样的, 有可能是统计miss次数部分的逻辑出了问题.

  电路资源:

2路(16块)	3路(24块)	4路(32块)	5路(40块)
LUT: 3.72%, FF: 4.89%	LUT: 5.74%, FF: 6.96%	LUT: 8.96%, FF: 9.03%	LUT: 10.38%, FF: 11.10%

分析:

替换策略几乎没有影响. 保持组的数量为8组时, 改变组相联度对缺失率(性能)影响不大, 但对电路资源有影响. 组相联度越大, 电路资源开销越大, 电路资源开销的增长与组相联度近似成线性关系.

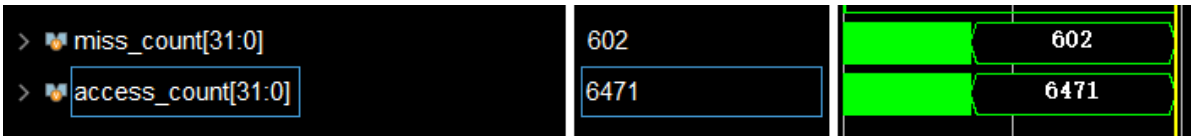
附图

1. SET_ADDR_LEN = 2, WAY_CNT = 4

   资源使用

Resource	Utilization	Available	Utilization %
LUT	5668	63400	8.94
FF	6219	126800	4.90
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count



2. SET_ADDR_LEN = 3, WAY_CNT = 4

   资源使用

Resource	Utilization	Available	Utilization %
LUT	5682	63400	8.96
FF	11452	126800	9.03
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count

3. SET_ADDR_LEN = 4, WAY_CNT = 4

   资源使用

Resource	Utilization	Available	Utilization %
LUT	9185	63400	14.49
FF	21883	126800	17.26
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

4. SET_ADDR_LEN = 5, WAY_CNT = 4

资源使用

Resource	Utilization	Available	Utilization %
LUT	14470	63400	22.82
FF	42594	126800	33.59
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

5. SET_ADDR_LEN = 3, WAY_CNT = 2

资源使用

Resource	Utilization	Available	Utilization %
LUT	2358	63400	3.72
FF	6199	126800	4.89
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

6. SET_ADDR_LEN = 3, WAY_CNT = 3

资源使用

Resource	Utilization	Available	Utilization %
LUT	3642	63400	5.74
FF	8827	126800	6.96
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

7. SET_ADDR_LEN = 3, WAY_CNT = 5

   资源使用

Resource	Utilization	Available	Utilization %
LUT	6578	63400	10.38
FF	14074	126800	11.10
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

2. 相同cache大小下, 不同组相联度的缺失率和电路资源

固定cache大小为32个块, 对组数量和组相联度一起调整.

缺失率:

替换策略	1路, 32组	2路, 16组	4路, 8组	8路, 4组	16路, 2组
FIFO	0.0173	0.0355	0.0558	0.0930	0.1445
LRU	0.0173	0.0355	0.0558	0.0930	0.1445

电路资源:

1路, 32组	2路, 16组	4路, 8组	8路, 4组	16路, 2组
LUT: 5.18%, FF: 7.36%	LUT: 7.49%, FF: 9.01%	LUT: 8.96%, FF: 9.03%	LUT: 13.21%, FF: 9.05%	LUT: 17.88%, FF: 9.06%

分析:

替换策略几乎没有影响. cache大小保持不变的情况下, 组相联度越大(对应组数越少), 缺失率越大(性能越差), FF资源开销几乎不增加, LUT资源开销增加, 但增加得不多. 说明在cache大小不变的情况下, 增加组数比增加组相联度更有利于提高性能和减少电路资源开销.

附图

1. 1路, 32组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	3287	63400	5.18
FF	9334	126800	7.36
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count

2. 2路, 16组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	4749	63400	7.49
FF	11420	126800	9.01
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count

3. 4路, 8组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	5682	63400	8.96
FF	11452	126800	9.03
BRAM	4	135	2.96
IO	81	210	38.57

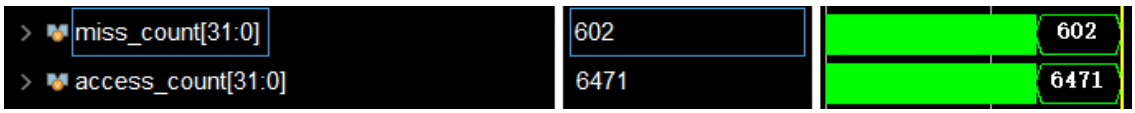
   FIFO/LRU下的miss_count和access_count

4. 8路, 4组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	8374	63400	13.21
FF	11478	126800	9.05
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

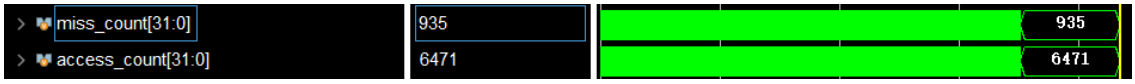


5. 16路, 2组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	11336	63400	17.88
FF	11491	126800	9.06
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count



说明: 从运行结果上来看, 两种策略的miss_count总是一样的, 至于为什么会这样我也不知道...我猜可能是这两种替换策略在实际运行时替换的块就是一样的, 虽然代码写出来不同, 但可能逻辑等价了...

3. 结论

根据上面的测试结果, 采用不同替换策略对cache的性能影响很小. 增加组数和增大组相联度都可以提高性能, 但相应地也会增加电路资源开销, 对比发现, 增加组数比增大组相联度更划算(增加相同的电路资源的情况下, 增加组数带来的性能提高效果更加明显). 对于这个例子而言, cache大小为32个块的情况下, 最好的方案是设置32个组, 每组只有一个块.

矩阵乘法的cache性能和电路资源分析

1. 不同cache大小的缺失率和电路资源

- 改变组的数量, 保持组相联度(WAY_CNT=4)和块大小(8个字)不变

  缺失率:

替换策略	4组(16块)	8组(32块)	16组(64块)	32组(128块)
FIFO	0.6397	0.5882	0.5625	0.1143
LRU	0.6397	0.5882	0.5625	0.1143

  电路资源:

4组(16块)	8组(32块)	16组(64块)	32组(128块)
LUT: 8.94%, FF: 4.90%	LUT: 8.96%, FF: 9.03%	LUT: 14.49%, FF: 17.26%	LUT: 22.82%, FF: 33.59%

分析:

替换策略的不同几乎没有影响. 组数越多, 缺失率越低(性能越好), 但相应地电路资源开销也会增加, 在从16组增加到32组时最性能提升最明显, 但电路资源开销也快速增大.

- 改变组相联度, 保持组的数量(8组)和块大小(8个字)不变

  缺失率:

替换策略	2路(16块)	3路(24块)	4路(32块)	5路(40块)
FIFO	0.5882	0.5882	0.5882	0.5882
LRU	0.5882	0.5882	0.5882	0.5882

注: 实际测出来就是这几个都是一样的, 有可能是统计miss次数部分的逻辑出了问题.

电路资源:

2路(16块)	3路(24块)	4路(32块)	5路(40块)
LUT: 3.72%, FF: 4.89%	LUT: 5.74%, FF: 6.96%	LUT: 8.96%, FF: 9.03%	LUT: 10.38%, FF: 11.10%

分析:

替换策略几乎没有影响. 保持组的数量为8组时, 改变组相联度对缺失率(性能)影响不大, 但对电路资源有影响. 组相联度越大, 电路资源开销越大, 电路资源开销的增长与组相联度近似成线性关系.

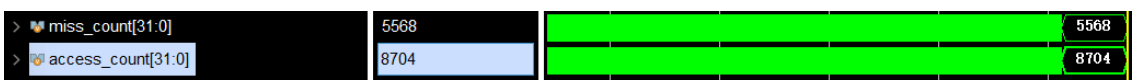
附图

1. SET_ADDR_LEN = 2, WAY_CNT = 4

   资源使用

Resource	Utilization	Available	Utilization %
LUT	5668	63400	8.94
FF	6219	126800	4.90
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count

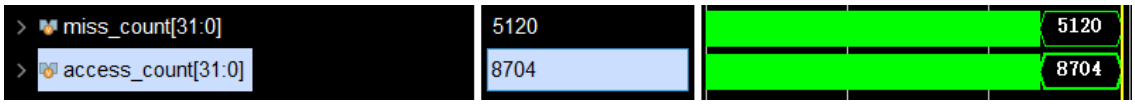


2. SET_ADDR_LEN = 3, WAY_CNT = 4

   资源使用

Resource	Utilization	Available	Utilization %
LUT	5682	63400	8.96
FF	11452	126800	9.03
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

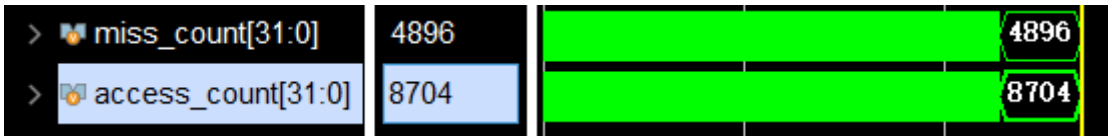


3. SET_ADDR_LEN = 4, WAY_CNT = 4

   资源使用

Resource	Utilization	Available	Utilization %
LUT	9185	63400	14.49
FF	21883	126800	17.26
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

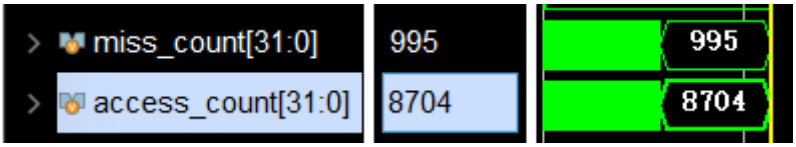


4. SET_ADDR_LEN = 5, WAY_CNT = 4

   资源使用

Resource	Utilization	Available	Utilization %
LUT	14470	63400	22.82
FF	42594	126800	33.59
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count



5. SET_ADDR_LEN = 3, WAY_CNT = 2

   资源使用

Resource	Utilization	Available	Utilization %
LUT	2358	63400	3.72
FF	6199	126800	4.89
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count

6. SET_ADDR_LEN = 3, WAY_CNT = 3

   资源使用

Resource	Utilization	Available	Utilization %
LUT	3642	63400	5.74
FF	8827	126800	6.96
BRAM	4	135	2.96
IO	81	210	38.57

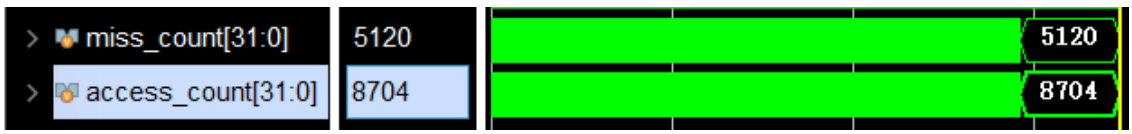
   FIFO/LRU下的miss_count和access_count

7. SET_ADDR_LEN = 3, WAY_CNT = 5

   资源使用

Resource	Utilization	Available	Utilization %
LUT	6578	63400	10.38
FF	14074	126800	11.10
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count



2. 相同cache大小下, 不同组相联度的缺失率和电路资源

固定cache大小为32个块, 对组数量和组相联度一起调整.

缺失率:

替换策略	1路, 32组	2路, 16组	4路, 8组	8路, 4组	16路, 2组
FIFO	0.1143	0.5625	0.5882	0.6397	0.7426
LRU	0.1143	0.5625	0.5882	0.6397	0.7426

电路资源:

1路, 32组	2路, 16组	4路, 8组	8路, 4组	16路, 2组
LUT: 5.18%, FF: 7.36%	LUT: 7.49%, FF: 9.01	LUT: 9.20%, FF: 9.04%	LUT: 13.21%, FF: 9.05%	LUT: 17.88%, FF: 9.06%

分析:

现象规律与快速排序中的一致. 不同的就是缺失率非常高!

附图

1. 1路, 32组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	3287	63400	5.18
FF	9334	126800	7.36
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count

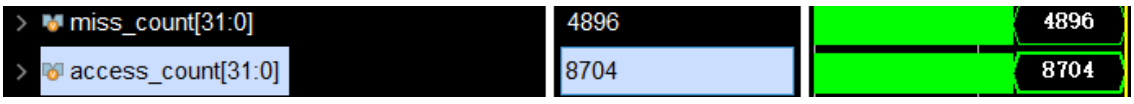
   

2. 2路, 16组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	4749	63400	7.49
FF	11420	126800	9.01
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count

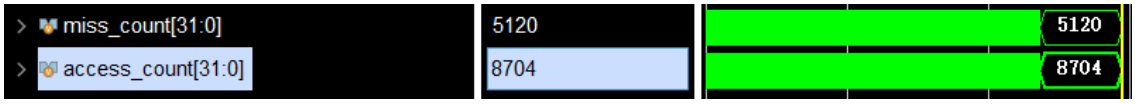
   

3. 4路, 8组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	5832	63400	9.20
FF	11458	126800	9.04
BRAM	4	135	2.96
IO	81	210	38.57

FIFO/LRU下的miss_count和access_count



4. 8路, 4组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	8374	63400	13.21
FF	11478	126800	9.05
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count

5. 16路, 2组

   资源使用

Resource	Utilization	Available	Utilization %
LUT	11336	63400	17.88
FF	11491	126800	9.06
BRAM	4	135	2.96
IO	81	210	38.57

   FIFO/LRU下的miss_count和access_count

3. 结论

结论与快速排序中的基本相同.

总结

对于不同的程序, cache的大小、组数、组相联度、替换策略等产生的效果大同小异. 基本规律是:

1. 采用不同替换策略对cache的性能影响不一定很大.
2. 增加组数和增大组相联度都可以提高性能, 但相应地也会增加电路资源开销, 对比发现, 增加组数比增大组相联度更划算(增加相同的电路资源的情况下, 增加组数带来的性能提高效果更加明显).