计算机体系结构实验报告

LAB03

题目: Cache 实现及性能评估

姓名: 魏钊

学号: PB18111699

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连度（相连度增大 cache size 也会增大，但是冲突 miss 会降低） 。

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。

实验平台

Vivado 2019.1

实验内容

1. 实现 N 路组相连的 Cache， 策略分别为 FIFO 和 LRU。

2. 将 Cache 组合到之前实现的 CPU 上，进行仿真（使用快速排序和矩阵乘法的 benchmark） 和综合。

3. 对不同 Cache 策略和参数进行性能和资源的测试评估， 其中“性能”参数使用运行仿真时的时钟周期数量进行评估， “资源占用”参数使用 vivado 给出的综合报告进行评估。

实验结果与分析

一、资源占用

保持主存大小为 2^13，TAG_ADDR_LEN+SET_ADDR_LEN+LINE_ADDR_LEN= 13

1. 直接映射

(1)不同参数下的使用情况

LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9

Resource	Estimation	Available	Utilization %
LUT	517	32600	1.59
FF	1123	65200	1.72

LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8

Resource	Estimation	Available	Utilization %
LUT	738	32600	2.26
FF	1671	65200	2.56

LINE_ADDR_LEN	2
SET_ADDR_LEN	4
TAG_ADDR_LEN	7

Resource	Estimation	Available	Utilization %
LUT	1233	32600	3.78
FF	2759	65200	4.23

LINE_ADDR_LEN	2
SET_ADDR_LEN	5
TAG_ADDR_LEN	6

Resource	Estimation	Available	Utilization %
LUT	2168	32600	6.65
FF	4919	65200	7.54

LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8

Resource	Estimation	Available	Utilization %
LUT	1154	32600	3.54
FF	2012	65200	3.09

LINE_ADDR_LEN	3
SET_ADDR_LEN	3
TAG_ADDR_LEN	7

Resource	Estimation	Available	Utilization %
LUT	1354	32600	4.15
FF	3068	65200	4.71

LINE_ADDR_LEN	3
SET_ADDR_LEN	4
TAG_ADDR_LEN	6

Resource	Estimation	Available	Utilization %
LUT	1860	32600	5.71
FF	5172	65200	7.93

LINE_ADDR_LEN	3
SET_ADDR_LEN	5
TAG_ADDR_LEN	5

Resource	Estimation	Available	Utilization %
LUT	3726	32600	11.43
FF	9364	65200	14.36

LINE_ADDR_LEN	4
SET_ADDR_LEN	2
TAG_ADDR_LEN	7

Resource	Estimation	Available	Utilization %
LUT	2151	32600	6.60
FF	3797	65200	5.82

LINE_ADDR_LEN	4
SET_ADDR_LEN	3
TAG_ADDR_LEN	6

Resource	Estimation	Available	Utilization %
LUT	2332	32600	7.15
FF	5873	65200	9.01

LINE_ADDR_LEN	4
SET_ADDR_LEN	4
TAG_ADDR_LEN	5

Resource	Estimation	Available	Utilization %
LUT	3477	32600	10.67
FF	10017	65200	15.36

LINE_ADDR_LEN	4
SET_ADDR_LEN	5
TAG_ADDR_LEN	4

Resource	Estimation	Available	Utilization %
LUT	7112	32600	21.82
FF	18289	65200	28.05

LINE_ADDR_LEN	5
SET_ADDR_LEN	2
TAG_ADDR_LEN	6

Resource	Estimation	Available	Utilization %
LUT	4593	32600	14.09
FF	7376	65200	11.31

LINE_ADDR_LEN	5
SET_ADDR_LEN	3
TAG_ADDR_LEN	5

Resource	Estimation	Available	Utilization %
LUT	4009	32600	12.30
FF	11496	65200	17.63

LINE_ADDR_LEN	5
SET_ADDR_LEN	4
TAG_ADDR_LEN	4

Resource	Estimation	Available	Utilization %
LUT	6772	32600	20.77
FF	19728	65200	30.26

LINE_ADDR_LEN	5
SET_ADDR_LEN	5
TAG_ADDR_LEN	3

Resource	Estimation	Available	Utilization %
LUT	12476	32600	38.27
FF	36176	65200	55.48

从上面的测试数据我们可以看到， LUT 和 FF 资源的占用和以下几个因素呈正相关：

1. 每个 LINE 中 WORD 的数量（2^LINE_ADDR_LEN）

2. SET 的数量（2^SET_ADDR_LEN）

2. 组相联之 FIFO 策略

(1)不同参数下的使用情况

LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9
WAYCNT	3, 4, 5, 6, 7, 8
Policy	FIFO

Resource	Estimation	Available	Utilization %
LUT	1668	32600	5.12
FF	2357	65200	3.62

Resource	Estimation	Available	Utilization %
LUT	2303	32600	7.06
FF	2913	65200	4.47

Resource	Estimation	Available	Utilization %
LUT	2615	32600	8.02
FF	3469	65200	5.32

Resource	Estimation	Available	Utilization %
LUT	3028	32600	9.29
FF	4025	65200	6.17

Resource	Estimation	Available	Utilization %
LUT	3417	32600	10.48
FF	4581	65200	7.03

Resource	Estimation	Available	Utilization %
LUT	3779	32600	11.59
FF	5137	65200	7.88

LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5, 6, 7, 8
Policy	FIFO

Resource	Estimation	Available	Utilization %
LUT	1987	32600	6.10
FF	4129	65200	6.33

Resource	Estimation	Available	Utilization %
LUT	2685	32600	8.24
FF	5233	65200	8.03

Resource	Estimation	Available	Utilization %
LUT	3052	32600	9.36
FF	6337	65200	9.72

Resource	Estimation	Available	Utilization %
LUT	3542	32600	10.87
FF	7441	65200	11.41

Resource	Estimation	Available	Utilization %
LUT	4070	32600	12.48
FF	8545	65200	13.11

Resource	Estimation	Available	Utilization %
LUT	4277	32600	13.12
FF	9649	65200	14.80

LINE_ADDR_LEN	2
SET_ADDR_LEN	4
TAG_ADDR_LEN	7
WAYCNT	3, 4, 5, 6, 7, 8
Policy	FIFO

Resource	Estimation	Available	Utilization %
LUT	3345	32600	10.26
FF	7652	65200	11.74

Resource	Estimation	Available	Utilization %
LUT	4337	32600	13.30
FF	9843	65200	15.10

Resource	Estimation	Available	Utilization %
LUT	5094	32600	15.63
FF	12035	65200	18.46

Resource	Estimation	Available	Utilization %
LUT	5980	32600	18.34
FF	14229	65200	21.82

Resource	Estimation	Available	Utilization %
LUT	6857	32600	21.03
FF	16421	65200	25.19

Resource	Estimation	Available	Utilization %
LUT	8355	32600	25.63
FF	18632	65200	28.58

LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5, 6, 7, 8
Policy	FIFO

Resource	Estimation	Available	Utilization %
LUT	3397	32600	10.42
FF	4262	65200	6.54

Resource	Estimation	Available	Utilization %
LUT	4217	32600	12.94
FF	5326	65200	8.17

Resource	Estimation	Available	Utilization %
LUT	4490	32600	13.77
FF	6390	65200	9.80

Resource	Estimation	Available	Utilization %
LUT	5669	32600	17.39
FF	7454	65200	11.43

Resource	Estimation	Available	Utilization %
LUT	6168	32600	18.92
FF	8518	65200	13.06

Resource	Estimation	Available	Utilization %
LUT	6893	32600	21.14
FF	9582	65200	14.70

从上面的数据我们可以看到， LUT 资源的占用和以下几个因素呈正相关:

1. 每个 LINE 中 WORD 的数量（2^LINE_ADDR_LEN）

2. SET 的数量（2^SET_ADDR_LEN）

3. 每个 SET 中的 WAY 的数量（WAY_CNT）

我们也可以看到，FF 资源的占用和以下几个因素呈正相关：

1. 每个 LINE 中 WORD 的数量（2^LINE_ADDR_LEN）

2. SET 的数量（2^SET_ADDR_LEN）

3. 每个 SET 中的 WAY 的数量（WAY_CNT）

值得注意的是， SET_ADDR_LEN 为 2 和 3 的时候， LUT 占用相差不大， 这很可能和 LUT 的实现方式有关。

查找表（Look-Up-Table）简称为 LUT， LUT 本质上就是一个 RAM。目前 FPGA 中多使用 4 输入的 LUT，所以每一个 LUT 可以看成一个有 4 位地址线的的 RAM。 由于一个 LUT 对应 4 个输入，如果电路的规模不是很理想，比如，并不是正好为 4 的倍数，可能带来一些不必要的浪费，上面的 SET_ADDR_LEN 为 2 和 3 时，资源使用差不多，说明 SET_ADDR_LEN 为 2 时，占用了不必要的资源。

3. 组相联之 LRU 策略

(1)不同参数下的使用情况

LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9
WAYCNT	3, 4, 5,
Policy	LRU

Resource	Estimation	Available	Utilization %
LUT	2460	47200	5.21
FF	2809	94400	2.98

Resource	Estimation	Available	Utilization %
LUT	3624	47200	7.68
FF	3489	94400	3.70

Resource	Estimation	Available	Utilization %
LUT	4049	47200	8.58
FF	4173	94400	4.42

LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5,

Resource	Estimation	Available	Utilization %
LUT	2968	47200	6.29
FF	4969	94400	5.26

Resource	Estimation	Available	Utilization %
LUT	3952	47200	8.37
FF	6329	94400	6.70

Resource	Estimation	Available	Utilization %
LUT	4648	47200	9.85
FF	7692	94400	8.15

LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5,
Policy	LRU

Resource	Estimation	Available	Utilization %
LUT	3897	47200	8.26
FF	4714	94400	4.99

Resource	Estimation	Available	Utilization %
LUT	5409	47200	11.46
FF	5902	94400	6.25

Resource	Estimation	Available	Utilization %
LUT	6156	47200	13.04
FF	7098	94400	7.52

从上面我们可以看到， LUT 资源的占用和以下几个因素呈正相关:

1. 每个 LINE 中 WORD 的数量（2^LINE_ADDR_LEN）

2. SET 的数量（2^SET_ADDR_LEN）

3. 每个 SET 中的 WAY 的数量（WAY_CNT）

SET_ADDR_LEN 为 2 和 3 的时候， LUT 占用相差不大，原因分析同 FIFO 策略

从图中我们可以看到， FF 资源的占用和以下几个因素呈正相关：

1. 每个 LINE 中 WORD 的数量（2^LINE_ADDR_LEN）

2. SET 的数量（2^SET_ADDR_LEN）

3. 每个 SET 中的 WAY 的数量（WAY_CNT）

4. 直接映射和组相联的 FIFO 策略/LRU 策略在资源占用方面的比较

从上面的图表中，我们可知， 直接映射及组相联的两种策略在 LUT 和 FF 的占用方面对于几个参数的变化是一致的（随着 LINE_ADDR_LEN，SET_ADDR_LEN，WAY_CNT 的增大而增大，因为这几个参数的增大意味着 Cache 的 size 变大，自然所需的资源变多） 。

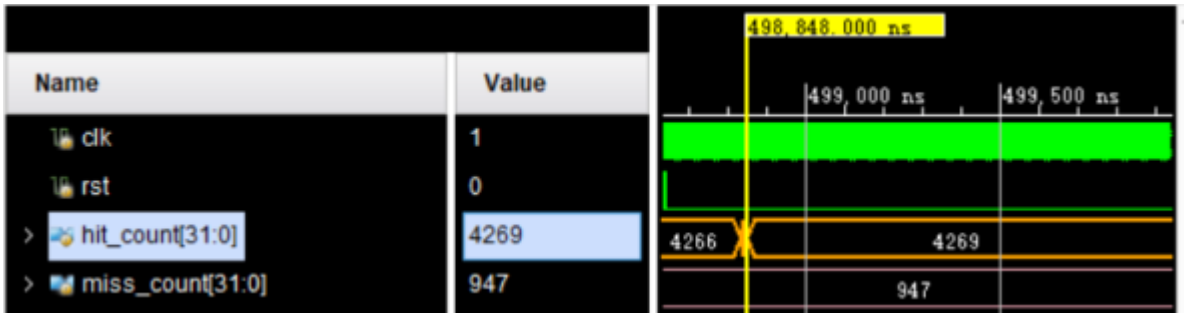
组相联的 FIFO 策略/LRU 策略资源占用数量上较为接近， 其他参数相同的情况下，使用 LRU 策略的资源占用比 FIFO 高出约 10%-20%。而直接映射的资源占用最低，因为直接映射相当于组相联的 WAY 数为 1。

因此，在实现 Cache 的时候， 资源占用方面主要考虑的是几个参数的大小，也就是 Cache 的大小， Cache 越大，资源占用越多。

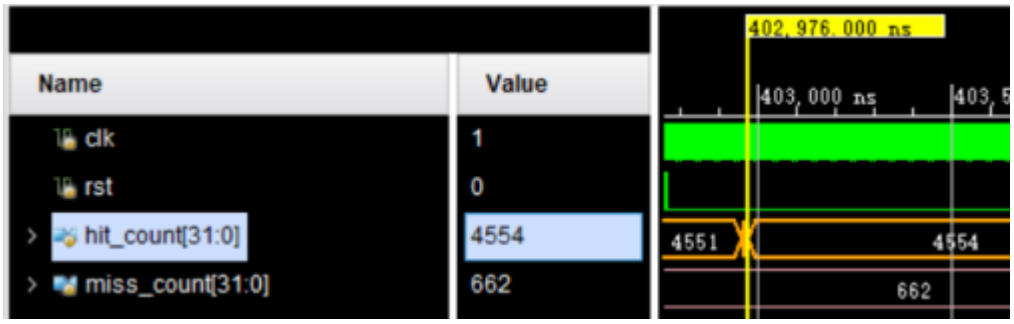
二、 QUICKSORT（256 个数） 性能

1. 直接映射

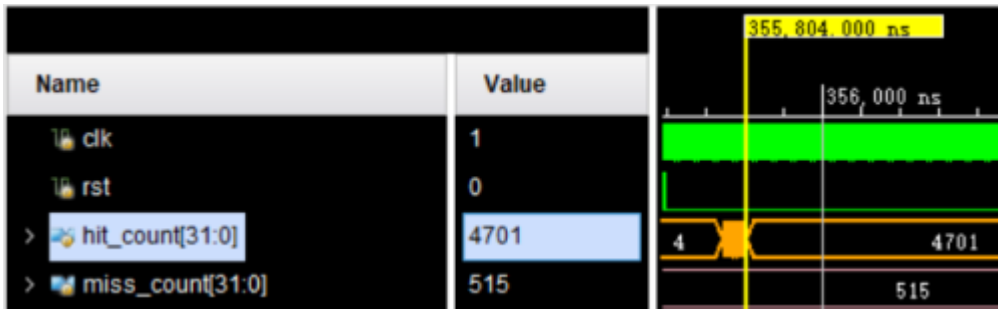
LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9
Test File	QUICKSORT



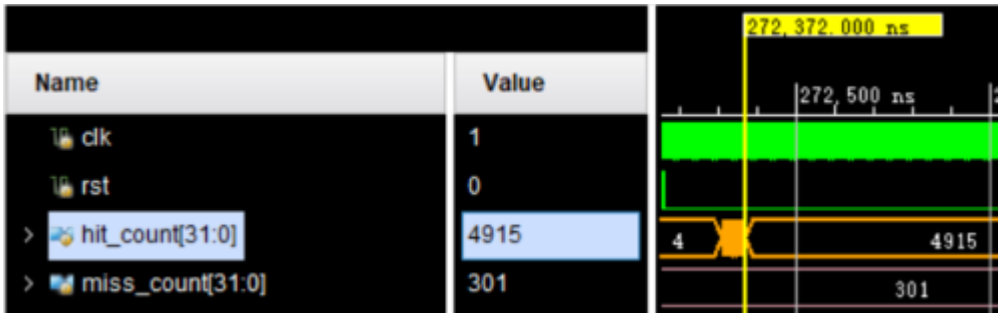
LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8
Test File	QUICKSORT



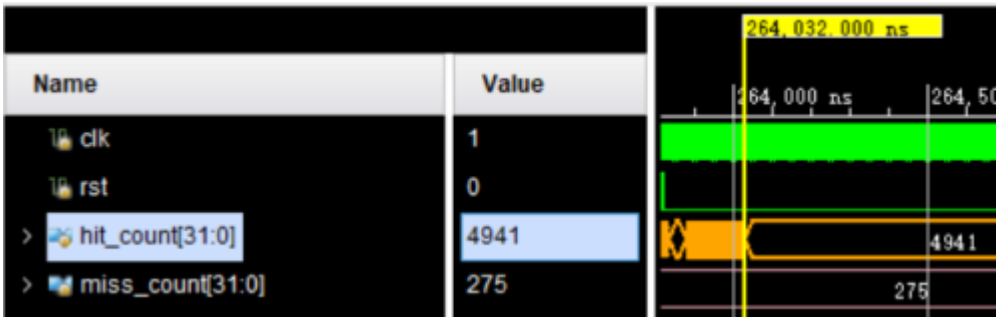
LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8
Test File	QUICKSORT



LINE_ADDR_LEN	3
SET_ADDR_LEN	3
TAG_ADDR_LEN	7
Test File	QUICKSORT



LINE_ADDR_LEN	4
SET_ADDR_LEN	2
TAG_ADDR_LEN	7
Test File	QUICKSORT



缺失率分析：

考虑 SET_ADDR_LEN 参数：这个参数决定 SET 的数量，因此随着这个参数的增大， Cache 容量变大，容量缺失会随之减少，由于直接映射每个 SET 只有一个 WAY，因此映射到同一个 SET 的块不可避免地要进行替换，如果 SET 数足够多，就能降低冲突缺失。上面两个原因使得更大的 SET_ADDR_LEN 带来更低的缺失率，从图中也可以看到这样的趋势。

考虑 LINE_ADDR_LEN 参数：这个参数决定了一个 LINE 中的 WORD 数量，也就是一个块的大小。块是 Cache 替换的基本单位，因此较大的块可以充分利用空间局域性，减少强制缺失的数量，从而降低缺失率，从图中也可以看到这样的趋势。

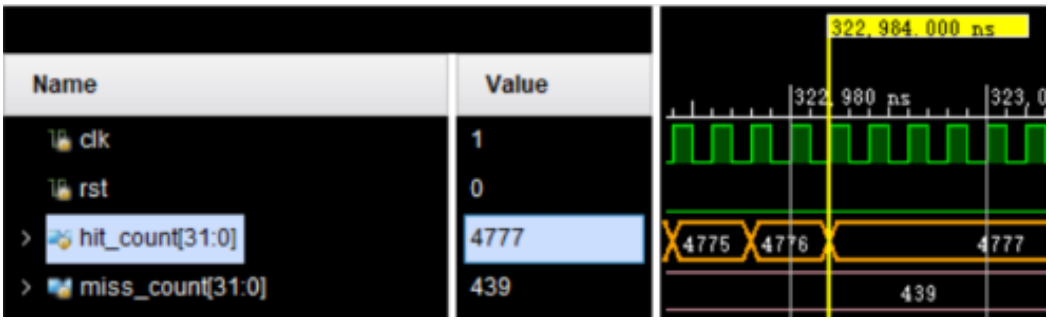
如果同时考虑 LINE_ADDR_LEN 参数和 SET_ADDR_LEN 参数，使它们的和不变，也就是 Cache 的大小保持不变。增大 LINE_ADDR_LEN 而减小 SET_ADDR_LEN，仍然能带来缺失率的降低，这说明，块的大小比相联度对缺失率的影响较大，这是因为，快速排序的空间局域性较好。

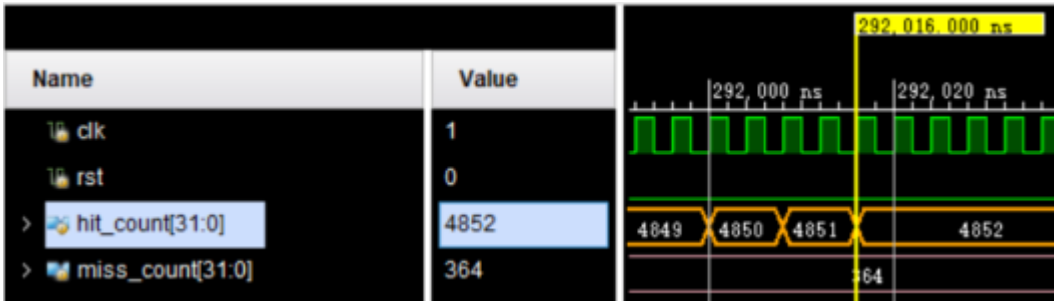
时间分析：

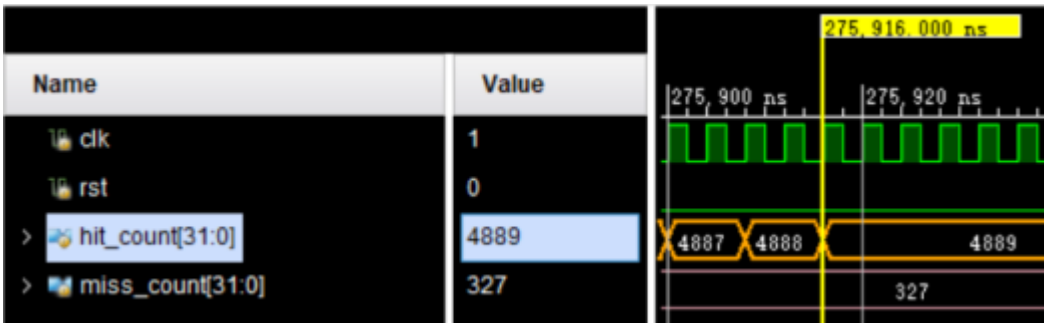
整体上， 访存时间和缺失率是大致成比例的（访存的大部分时间花在缺失上，一次缺失可能带来 50 周期或 100 周期开销， 而命中只需 1 个周期） ，但在缺失率较低时， 总时间并不是特别低，因为程序除了访存的时间，还有计算的时间，当缺失率较低因而访存时间较低时，计算的时间在总时间中所占比例就会比较高。

2. 组相联之 FIFO 策略

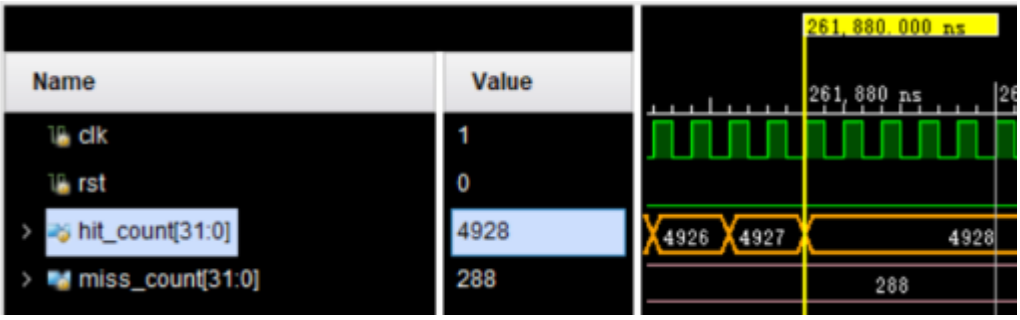
LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9
WAYCNT	3, 4, 5
Policy	FIFO

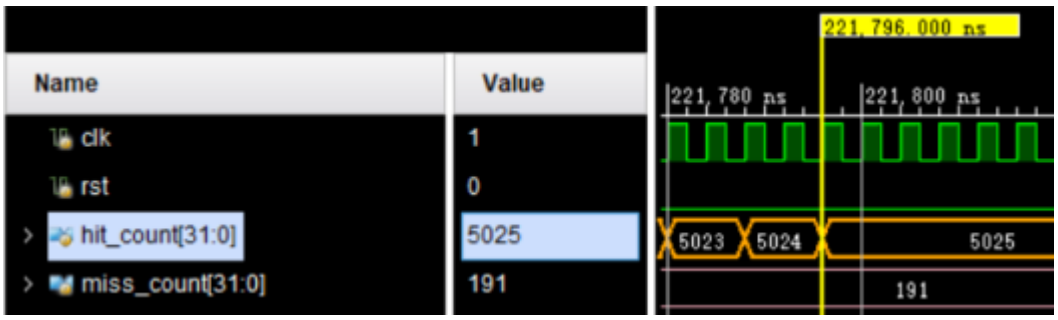




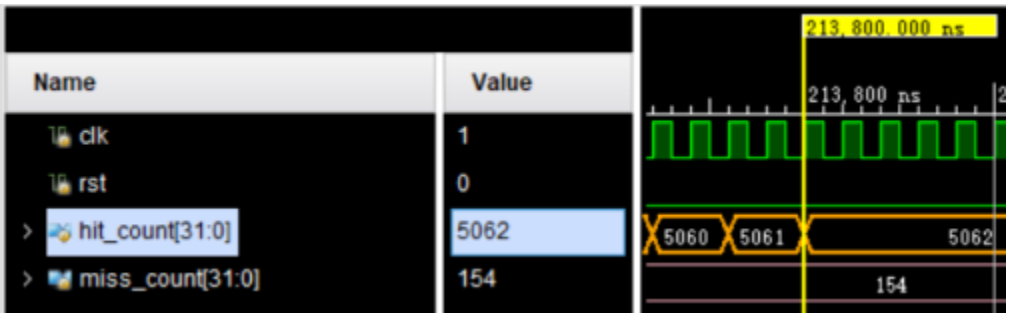


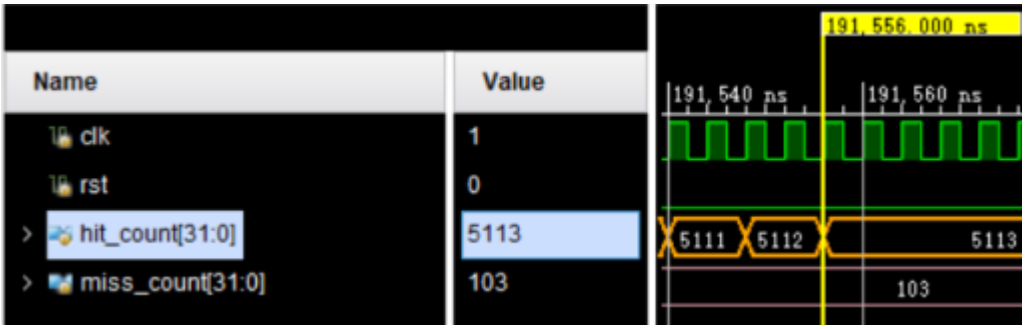
LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5
Policy	FIFO





LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5,
Policy	FIFO





缺失率分析:

考虑 WAY_CNT 和 SET_ADDR_LEN 参数： 这两个参数分别是一个 SET 中的 LINE 数和 SET 的数量，它们决定了 Cache 的大小。 因此随着这两个参数的增大，Cache 容量变大， 容量缺失会随之减少， WAY_CNT 的增大还会使得冲突缺失的减少， 从而降低缺失率， 从图中也可以看到这样的趋势。 事实上，如果 WAY_CNT 过大，是会增加命中时间的，因为需要对更多的路进行比较，但这在 Verilog 的周期数中没有体现出来。

考虑 LINE_ADDR_LEN 参数： 这个参数决定了一个 LINE 中的 WORD 数量，也就是一个块的大小。 块是 Cache 替换的基本单位， 因此较大的块可以充分利用空间局域性，减少强制缺失的数量， 从而降低缺失率， 从图中也可以看到这样的趋势。

如果同时考虑 LINE_ADDR_LEN 参数和 SET_ADDR_LEN 参数，使它们的和不变，也就是 Cache 的大小保持不变。增大 LINE_ADDR_LEN 而减小 SET_ADDR_LEN，仍然能带来缺失率的明显降低，这说明，块的大小比相联度对缺失率的影响较大，这是因为，快速排序的空间局域性较好。

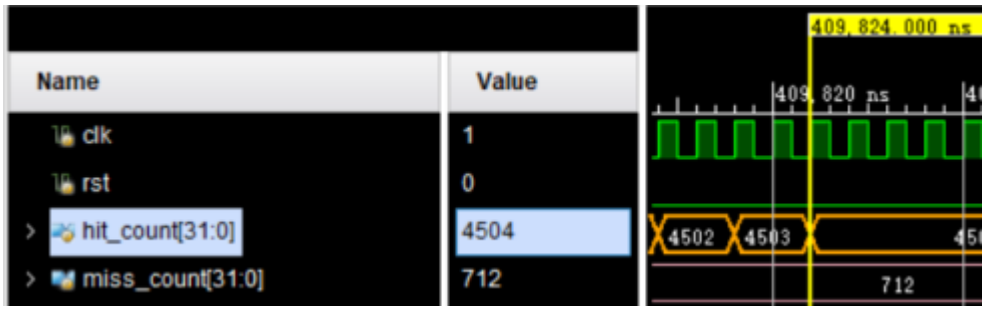
从图中我们还可以发现，当 LINE_ADDR_LEN 较大时， 改变其他参数对缺失率的影响不大， 这是因为强制缺失是不可避免的。

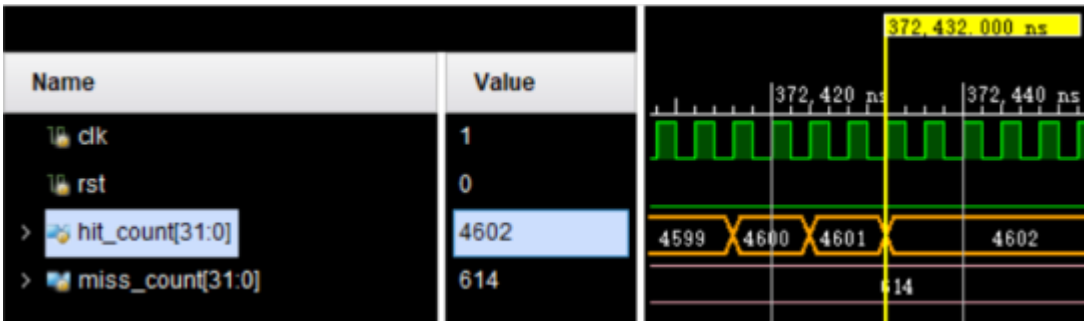
时间分析:

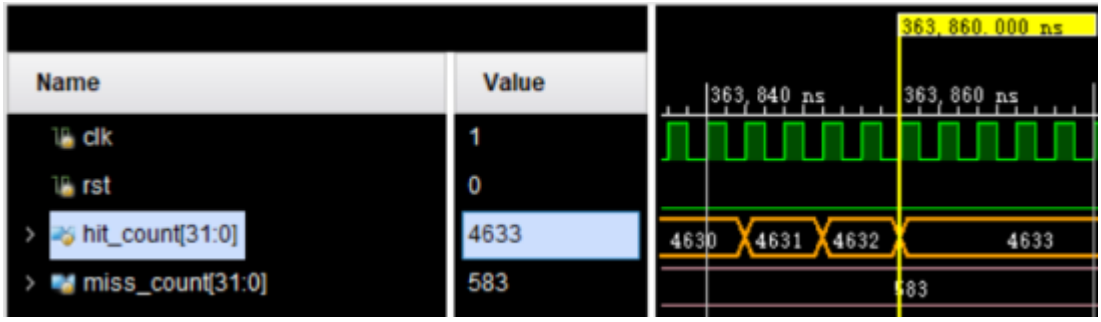
整体上，访存时间和缺失率是大致成比例的（访存的大部分时间花在缺失上， 一次缺失可能带来 50 周期或 100 周期开销， 而命中只需 1 个周期） ，但在缺失率较低时，总时间并不是特别低，因为程序除了访存的时间，还有计算的时间，当缺失率较低因而访存时间较低时，计算的时间在总时间中所占比例就会比较高。

3. 组相联之 LRU 策略

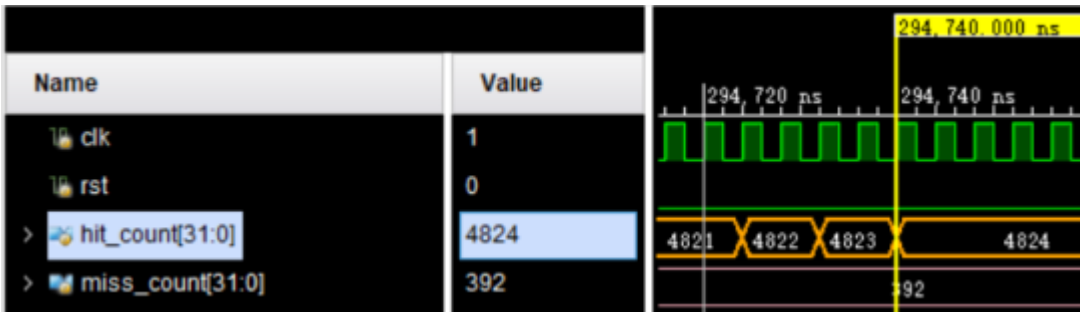
LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9
WAYCNT	3, 4, 5
Policy	LRU



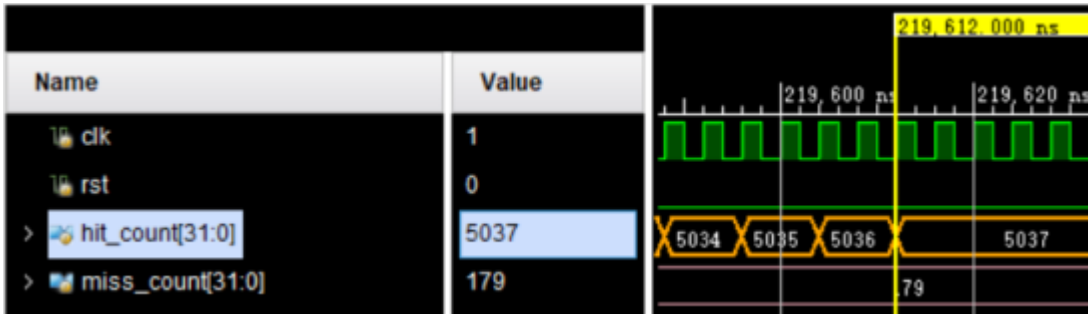




LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5
Policy	LRU



LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5,
Policy	LRU



缺失率分析:

考虑 WAY_CNT 和 SET_ADDR_LEN 参数：这两个参数分别是一个 SET 中的 LINE 数和 SET 的数量，它们决定了 Cache 的大小。因此随着这两个参数的增大，Cache 容量变大，容量缺失会随之减少， WAY_CNT 的增大还会使得冲突缺失的减少，从而降低缺失率，从图中也可以看到这样的趋势。但这两个参数增大到一定程度后，缺失率就不再降低。

考虑 LINE_ADDR_LEN 参数：这个参数决定了一个 LINE 中的 WORD 数量，也就是一个块的大小。块是 Cache 替换的基本单位，因此较大的块可以充分利用空间局域性，减少强制缺失的数量，从而降低缺失率，从图中也可以看到这样的趋势。并且当 LINE_ADDR_LEN 较大时，改变其他参数对缺失率的影响不大，这是因为强制缺失是不可避免的。

如果同时考虑 LINE_ADDR_LEN 参数和 SET_ADDR_LEN 参数，使它们的和不变，也就是 Cache 的大小保持不变。增大 LINE_ADDR_LEN 而减小 SET_ADDR_LEN，仍然能带来缺失率的明显降低，这说明，块的大小比相联度对缺失率的影响较大，这是因为，快速排序的空间局域性较好。

时间分析:

整体上，访存时间和缺失率是大致成比例的（访存的大部分时间花在缺失上， 一次缺失可能带来 50 周期或 100 周期开销， 而命中只需 1 个周期） ，但在缺失率较低时，总时间并不是特别低，因为程序除了访存的时间，还有计算的时间，当缺失率较低因而访存时间较低时，计算的时间在总时间中所占比例就会比较高。

4. 直接映射和组相联的 FIFO 策略/LRU 策略在 QUICKSORT 性能方面的比较

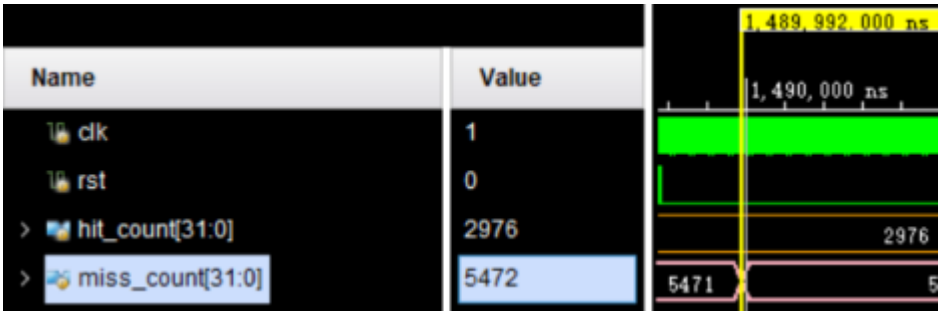
对比直接映射和组相联的 FIFO 策略/LRU 策略，在 SET_ADDR_LEN 和 LINE_ADDR_LEN 参数不变时， 直接映射的缺失率最高，直接映射相当于组相联的 WAY_CNT = 1，因此， 冲突缺失最多， 进行替换是最频繁的。

组相联的 FIFO 策略的缺失率明显低于 LRU 策略（FIFO 约为 LRU 的 50%-70%） ， 在这里分析一下可能的原因，快速排序是一个在连续内存上进行的分治算法，在对整个数组进行划分后，顺序递归调用两次自身，在一个更小的规模上进行同样的操作， 而此时对数据的访问只局限在当前范围内的数组，对于其他数据是不关心的，进行一部分访问后，再进行另一部分的访问，这样的程序执行特点（在访存上跳跃性不大） 和 FIFO 策略（将最早进入的块替换出去） 能较好地配合。

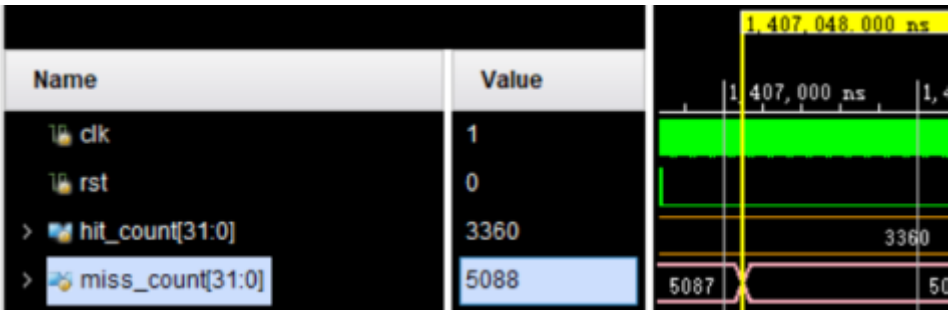
三、 MATMUL（16x16） 性能

1. 直接映射

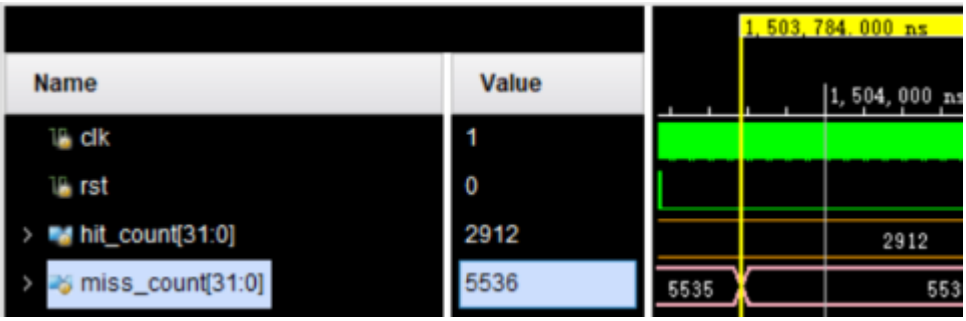
LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9
Test File	MATMUL



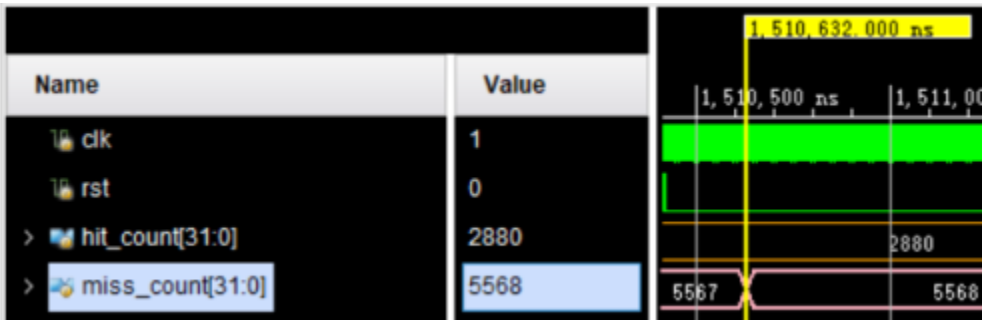
LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8
Test File	MATMUL



LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8
Test File	MATMUL



LINE_ADDR_LEN	4
SET_ADDR_LEN	2
TAG_ADDR_LEN	7
Test File	MATMUL



缺失率分析:

考虑 SET_ADDR_LEN 参数：这个参数决定 SET 的数量，因此随着这个参数的增大， Cache 容量变大，容量缺失会随之减少，由于直接映射每个 SET 只有一个 WAY，因此映射到同一个 SET 的块不可避免地要进行替换，如果 SET 数足够多，就能降低冲突缺失。上面两个原因使得更大的 SET_ADDR_LEN 带来更低的缺失率，从图中也可以看到这样的趋势。

考虑 LINE_ADDR_LEN 参数：这个参数决定了一个 LINE 中的 WORD 数量，也就是一个块的大小。块是 Cache 替换的基本单位，因此较大的块可以充分利用空间局域性，减少强制缺失的数量，从而降低低缺失率，从图中也可以看到这样的趋势。

值得注意的是，虽然上面两个参数的增大使得缺失率降低，但实际上效果并不明显，特别是两个参数比较小时，这是因为矩阵乘法的空间局域性比较差，而且直接映射的 Cache 的每个 SET 只有一个 WAY， 局域性不好的代码将带来频繁的换入换出。 另外，这两个参数增大到一定程度时，缺失率急剧降低，几乎为 0，这并非局域性变好了，而是因为 Cache 已经大到足够容纳整个矩阵，此时只剩下强制缺失

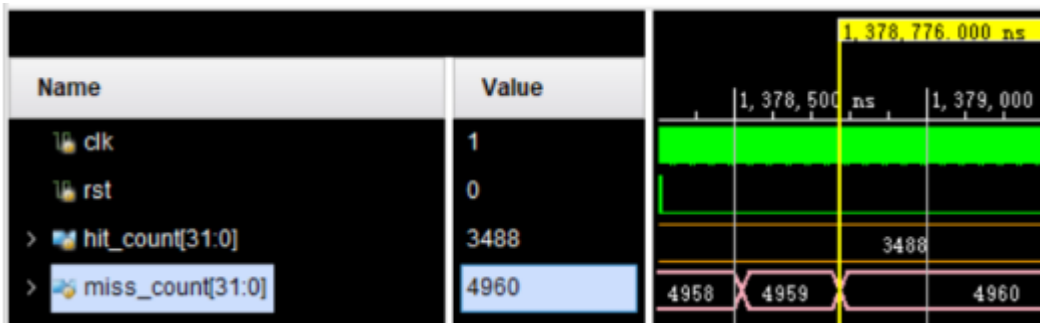
（第一次访问矩阵元素时不在 Cache 内），因此缺失率很低。

时间分析：

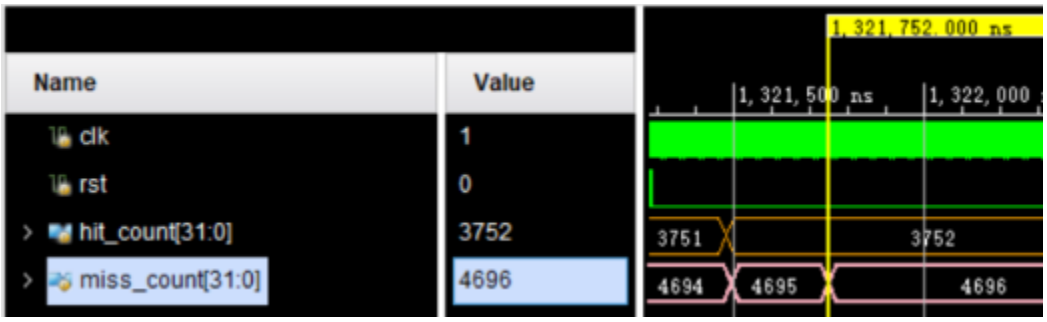
整体上，访存时间和缺失率是大致成比例的，但在缺失率较低时，总时间并不是特别低，因为程序除了访存的时间，还有计算的时间，当缺失率较低因而访存时间较低时，计算的时间在总时间中所占比例就会比较高。

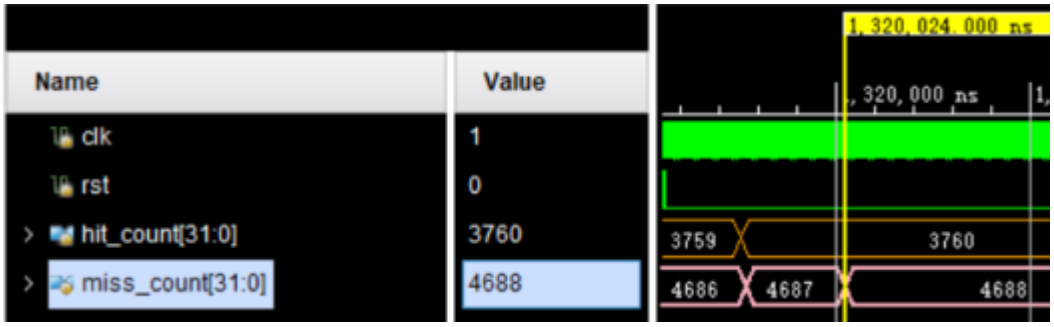
2. 组相联之 FIFO 策略

LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9
WAYCNT	3, 4, 5
Policy	FIFO



LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5
Policy	FIFO





LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5,
Policy	FIFO

缺失率分析：

考虑 WAY_CNT 和 SET_ADDR_LEN 参数：这两个参数分别是一个 SET 中的 LINE 数和 SET 的数量，它们决定了 Cache 的大小。 理论上， 随着这两个参数的增大， Cache 容量变大，容量缺失会随之减少， WAY_CNT 的增大还会使得冲突缺失的减少，从而降低缺失率。但我们从图中发现这样的趋势并不明显，特别是 LINE_ADDR_LEN 很小时， WAY_CNT 的增加几乎不能降低缺失率，一个可能原因是，矩阵乘法每次访问一行一列（并且计算结果矩阵同一行的连续一段时间内需要频繁访问源矩阵的同一行） ，假设为行优先，则同一行的相邻几个元素应该在同一个 LINE 中， 并且同一行的元素也可能对应到同一个（或相邻几个）SET 中， 如果 LINE 和 SET 过小， 就要求 SET 中要有更多的 WAY，否则可能有频繁的换入换出（FIFO 策略使得访问行尾部时将行首部的元素替换出去，而同一行元素需要多次顺序访问，这样一行的每个元素每次访问都需要换入换出） ，那么上图中在小范围内增大 WAY 并不会带来缺失率降低。

考虑 LINE_ADDR_LEN 参数：这个参数决定了一个 LINE 中的 WORD 数量，也就是一个块的大小。块是 Cache 替换的基本单位，因此较大的块可以充分利用空间局域性，减少强制缺失的数量，从而降低缺失率， 这样对应了上面的分析， 从图中我们也可以看到， LINE_ADDR_LEN 增大到一定程度（LINE_ADDR_LEN= 4， LINE 大小 2 ^ 4 = 16，正好为矩阵的行和列的长度） 时， 缺失率急剧降低，此时访问同一行的元素应该是没有发生替换的。 我们可以预测，LINE_ADDR_LEN 继续增大，缺失率会保持该低水平，但不会再明显降低，因为强制缺失不可避免。如果同时考虑

LINE_ADDR_LEN 参数和 SET_ADDR_LEN 参数（WAY_CNT 不变），使它们的和不变，也就是 Cache 的大小保持不变。增大 LINE_ADDR_LEN 而减小 SET_ADDR_LEN，缺失率没有明显变化， 这需要综合上面两种分析，一方面 LINE 的增大使得行内元素可以更多地同时在 Cache 的一个 LINE 中，减少首次调入的访存次数，一方面，SET 的减小使得多行对应到同一个 SET 中，增加了替换次数， 这样，强制缺失虽然降低但容量和冲突缺失会增加，总的缺失率不会有明显降低。

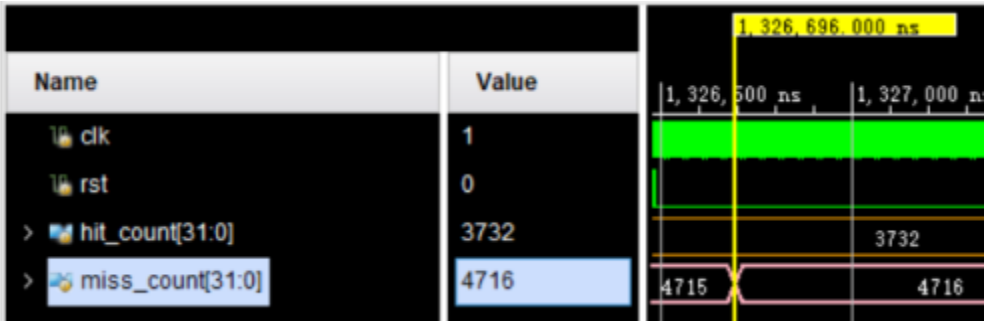
总体而言，矩阵乘法的空间局域性较差，因为每次计算一个元素需要矩阵的一行和一列，无论存储器采用行优先顺序还是列优先顺序，都有其一访问非连续存储空间，如果 Cache 的大小不足以容纳一次运算所需的全部元素， 那么频繁的换入换出将不可避免。

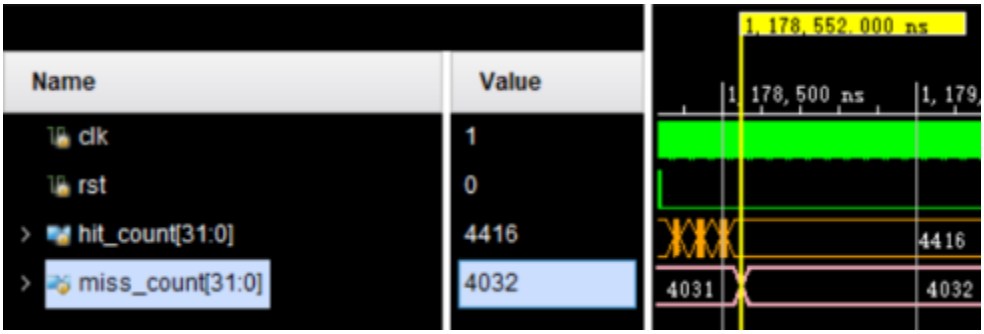
时间分析:

整体上，访存时间和缺失率是大致成比例的（访存的大部分时间花在缺失上， 一次缺失可能带来 50 周期或 100 周期开销， 而命中只需 1 个周期），但在缺失率较低时，总时间并不是特别低，因为程序除了访存的时间，还有计算的时间，当缺失率较低因而访存时间较低时，计算的时间在总时间中所占比例就会比较高。

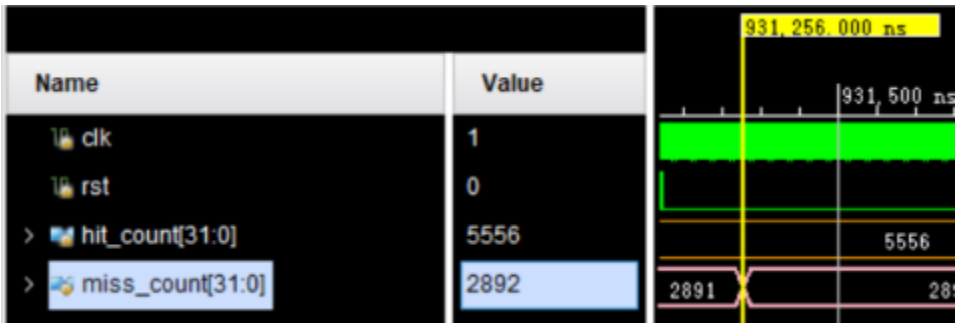
3. 组相联之 LRU 策略

LINE_ADDR_LEN	2
SET_ADDR_LEN	2
TAG_ADDR_LEN	9
WAYCNT	3, 4, 5
Policy	LRU

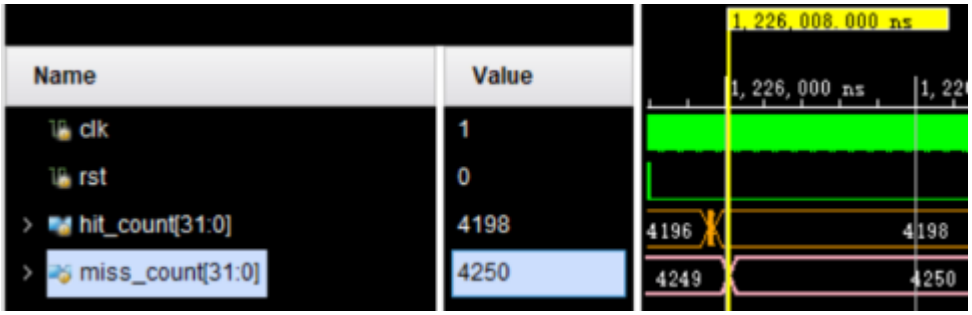


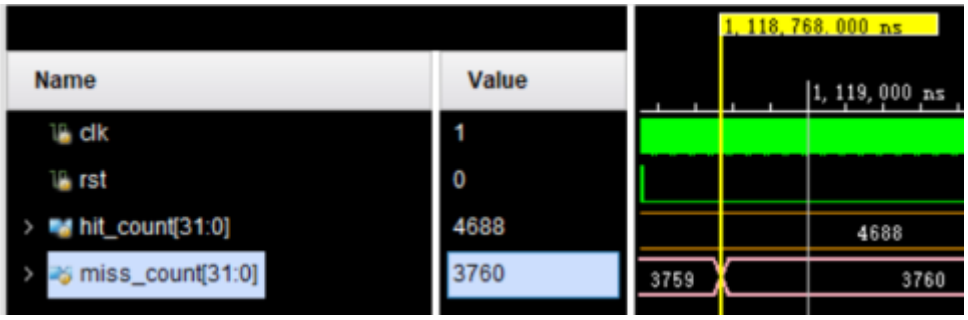


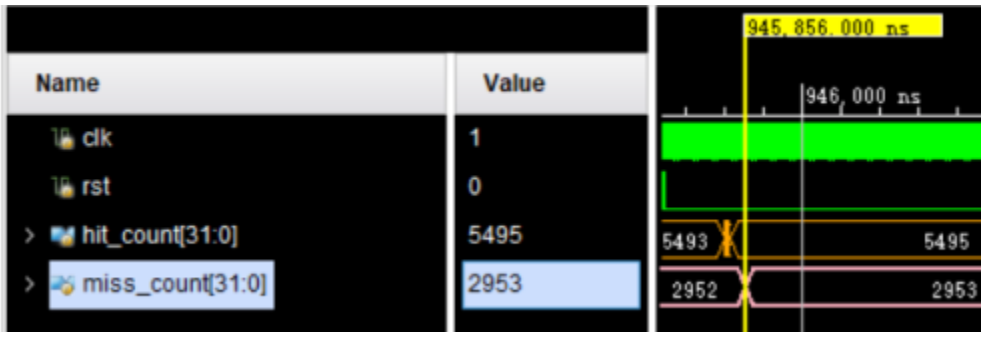
LINE_ADDR_LEN	2
SET_ADDR_LEN	3
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5,
Policy	LRU



LINE_ADDR_LEN	3
SET_ADDR_LEN	2
TAG_ADDR_LEN	8
WAYCNT	3, 4, 5
Policy	LRU







缺失率分析:

考虑 WAY_CNT 和 SET_ADDR_LEN 参数：这两个参数分别是一个 SET 中的 LINE 数和 SET 的数量，它们决定了 Cache 的大小。 随着这两个参数的增大，Cache 容量变大，容量缺失会随之减少， WAY_CNT 的增大还会使得冲突缺失的减少，从而降低缺失率。 另外， 我们从图中发现 WAY_CNT 和 SET_ADDR_LEN 比较大时，缺失率不再降低， 甚至更高， 这可能是矩阵乘法的访存跳跃性不能很好地适应 LRU 策略。

考虑 LINE_ADDR_LEN 参数：这个参数决定了一个 LINE 中的 WORD 数量，也就是一个块的大小。块是 Cache 替换的基本单位，因此较大的块可以充分利用空间局域性，减少强制缺失的数量，从而降低缺失率。如果同时考虑 LINE_ADDR_LEN 参数和 SET_ADDR_LEN 参数（WAY_CNT 不变），使它们的和不变，也就是 Cache 的大小保持不变。增大 LINE_ADDR_LEN 而减小 SET_ADDR_LEN，缺失率没有明显变

化，这需要综合上面两种分析，一方面 LINE 的增大使得行内元素可以更多地同时在 Cache 的一个 LINE 中，减少首次调入的访存次数，一方面， SET 的减小使得多行对应到同一个 SET 中，增加了替换次数，这样，强制缺失虽然降低但容量和冲突缺失会增加，总的缺失率不会有明显降低。

总体而言，矩阵乘法的空间局域性较差，因为每次计算一个元素需要矩阵的一行和一列，无论存储器采用行优先顺序还是列优先顺序，都有其一访问非连续存储空间，如果 Cache 的大小不足以容纳一次运算所需的全部元素，那么频繁的换入换出将不可避免。

时间分析：整体上，访存时间和缺失率是大致成比例的（访存的大部分时间花在缺失上， 一次缺失可能带来 50 周期或 100 周期开销， 而命中只需 1 个周期） ，但在缺失率较低时，总时间并不是特别低，因为程序除了访存的时间，还有计算的时间，当缺失率较低因而访存时间较低时，计算的时间在总时间中所占比例就会比较高。

4. 直接映射和组相联的 FIFO 策略/LRU 策略在 MATMUL 性能方面的比较

对比直接映射和组相联的 FIFO 策略/LRU 策略，在 SET_ADDR_LEN 和 LINE_ADDR_LEN 参数不变时，直接映射的缺失率最高，直接映射相当于组相联的 WAY_CNT = 1，因此，冲突缺失最多，进行替换是最频繁的。考虑组相联， FIFO 策略的缺失率和 LRU 策略的缺失率不相上， FIFO 的缺失率在 Cache 大小变大后急剧下降， 而 LRU 的缺失率变化比较平缓， 但在 Cache 大小较大时， 缺失率没有降到很低。 因此， 对于比较小的 Cache， LRU 比 FIFO 效果更好， 而对于比较大的 Cache， FIFO 比较好。

四、 结合资源占用和性能决定 Cache 参数和策略

（以下所提到的数据都是在本次测试范围内的结果）

1. QUICKSORT

在之前的性能分析中，我们知道，对于 QUICKSORT，直接映射的缺失率（最高达到 0.2）和时间（最高达到 500000ns）都是最高的，但如果 Cache 足够大，缺失率也能降到组相联的两种策略的水平（均在 0.15 以下，普遍在 0.08 以下） 。而直接映射的资源占用最小（LUT 最高 14000 左右， FF 最高 40000 左右，普遍在 20000 以下） ，组相联（LUT 最高达 25000~30000， FF 最高达到 70000)。因此，一种可选的方案是： 采用直接映射，但 Cache 大小需要足够大，比如 SET_ADDR_LEN = LINE_ADDR_LEN = 5（或 4）

另外，考虑组相联，之前已经比较， FIFO 的缺失率和时间明显低于 LRU，因此， 策略选择 FIFO，而 LINE_ADDR_LEN 至少为 3 才能有较低的缺失率/运行时间，再权衡资源占用， LINE_ADDR_LEN 为 4 时，资源占用已经很高，但运行时间不会有明显下降（因为有计算时间不可减少） 。同样地分析 SET_ADDR_LEN 和 WAY_CNT。因此，一种可选的方案是： 采用组相联， FIFO 策略，SET_ADDR_LEN = LINE_ADDR_LEN = 3， WAY_CNT = 5 至 8

2. MATMUL

在之前的性能分析中，我们知道，对于 MATMUL，直接映射的缺失率（最高达到 0.7）和时间（最高达到 1600000ns）都是最高的，但如果 Cache 足够大， 缺失率也能降到组相联的两种策略的水平（最低可降至 0.1 以下)。而直接映射的资源占用最小（LUT 最高 14000 左右， FF 最高 40000 左右，普遍在 20000 以下），组相联（LUT 最高达 25000~30000， FF 最高达到 70000)。因此， 一种可选的方案是: 采用直接映射，但 Cache 大小需要足够大，比如 SET_ADDR_LEN = LINE_ADDR_LEN = 5（或 4）

另外，考虑组相联，之前已经比较， LRU 的缺失率和 FIFO 的缺失率不相上下，对于比较小的 Cache， LRU 比 FIFO 效果更好， 这里策略选择 LRU，而 LINE_ADDR_LEN 至少为 3 才能有较低的缺失率/运行时间，再权衡资源占用，LINE_ADDR_LEN 为 4 时，资源占用已经很高，但运行时间不会有明显下降（因为有计算时间不可减少） 。 对于 SET_ADDR_LEN 和 WAY_CNT，从运行时间上看，变化规律不是特别明朗，似乎无规律可寻， WAY_CNT 比较大时候，SET_ADDR_LEN 反而越低越好，这里选择 SET_ADDR_LEN = 2。因此，一种可选的方案是： 采用组相联， LRU 策略， SET_ADDR_LEN = 2， LINE_ADDR_LEN =3， WAY_CNT = 5 或 6