计算机体系结构实验报告
LAB03

题目：Cache 实现及性能评估

姓名：林祥

学号：PB16020923

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

EDA 工具为 Vivado 2017.4

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

实验原始数据截图见同目录下附表文件，下面直接列出数据及分析

一、资源占用

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

1. 直接映射

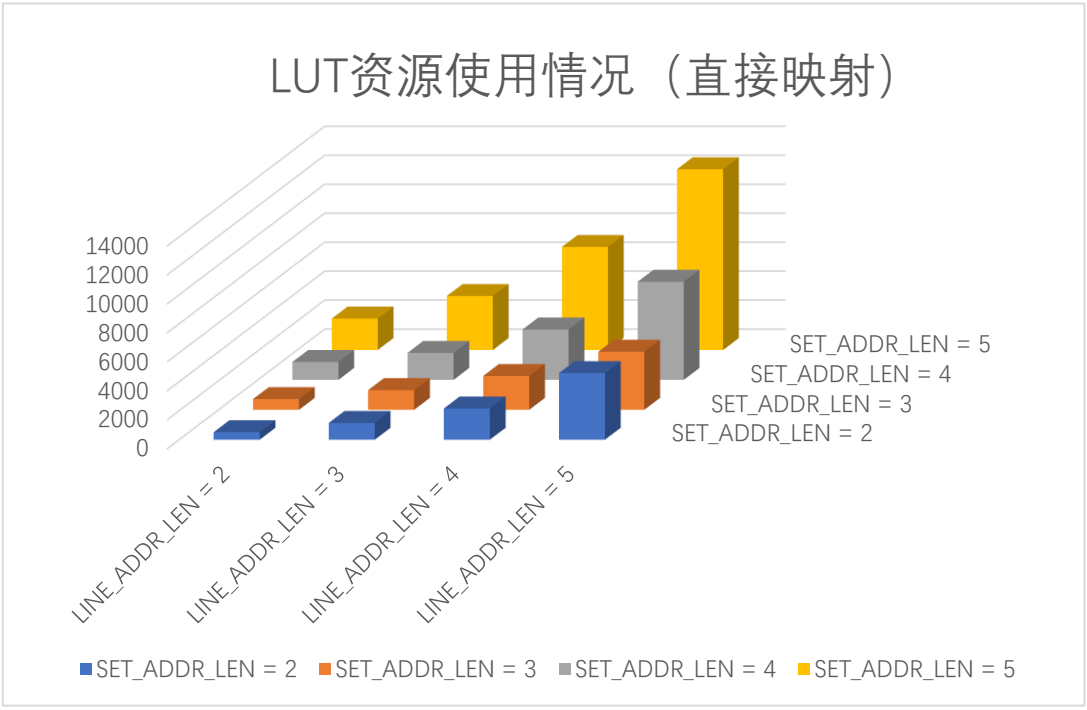
(1)不同参数下的 LUT 使用情况

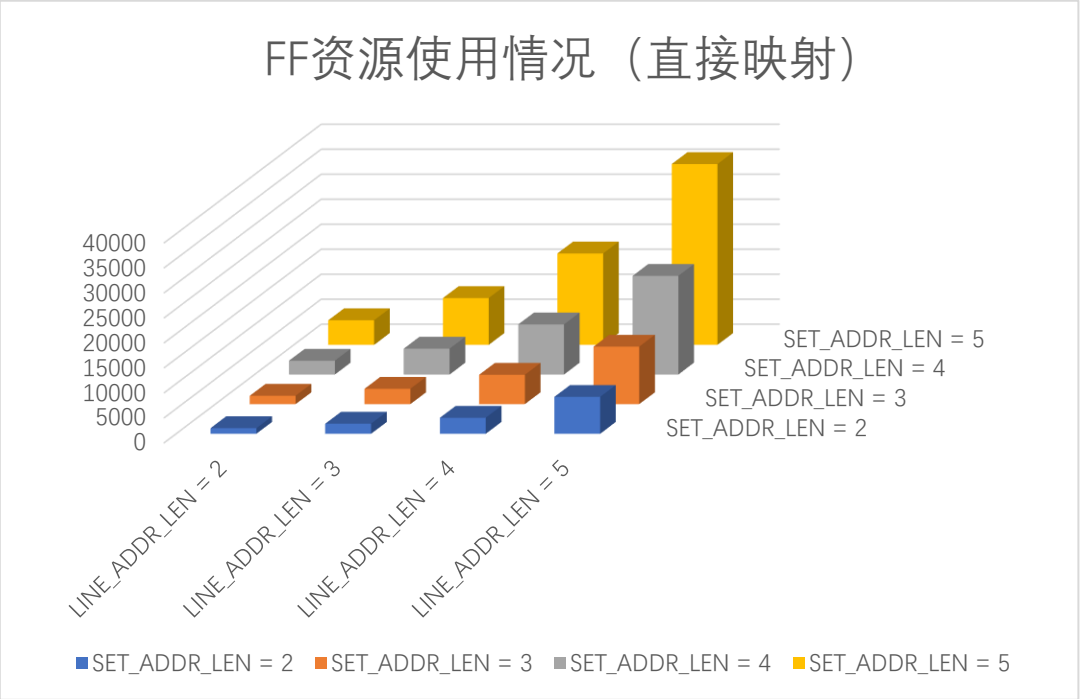
LUT（直接映射）				
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4	5
2	517	738	1233	2168
3	1154	1354	1860	3726
4	2151	2332	3477	7112
5	4593	4009	6772	12476

(2)不同参数下的 FF 使用情况

FF（直接映射）				
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4	5
2	1123	1671	2759	4919
3	2012	3068	5172	9364
4	3197	5873	10017	18289
5	7376	11496	19728	36176

可得图表如下





从图中我们可以看到，LUT 和 FF 资源的占用和以下几个因素呈正相关：

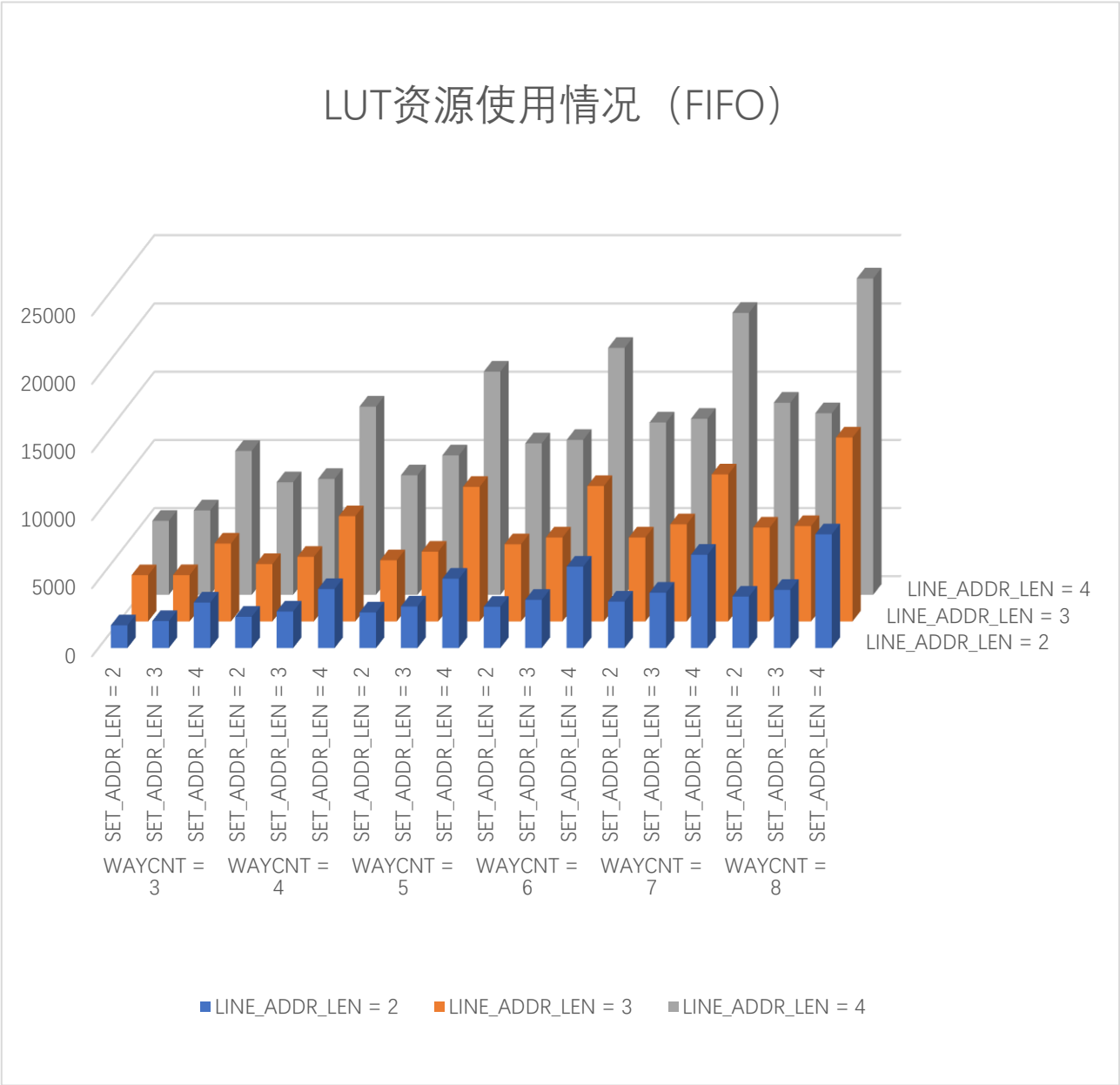
1. 每个 LINE 中 WORD 的数量（2^LINE_ADDR_LEN）
2. SET 的数量（2^SET_ADDR_LEN）

2. 组相联之 FIFO 策略

(1)不同参数下的 LUT 使用情况

LUT（FIFO）			
WAYCNT = 3			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1668	1987	3345
3	3397	3403	5720
4	5434	6202	10571
WAYCNT = 4			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	2303	2685	4337
3	4217	4747	7730
4	8264	8514	13815
WAYCNT = 5			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	2615	3052	5094
3	4490	5123	9888
4	8777	10241	16378
WAYCNT = 6			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3028	3542	5980
3	5669	6169	9948
4	11122	11374	18113
WAYCNT = 7			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3417	4070	6857
3	6168	7135	10796
4	12653	12922	20677
WAYCNT = 8			
SET_ADDR_LEN1 LINE_ADDR_LEN	2	3	4
2	3779	4277	8355
3	6893	7006	13507
4	14099	13327	23197

可得图表如下



从图中我们可以看到，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）

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。

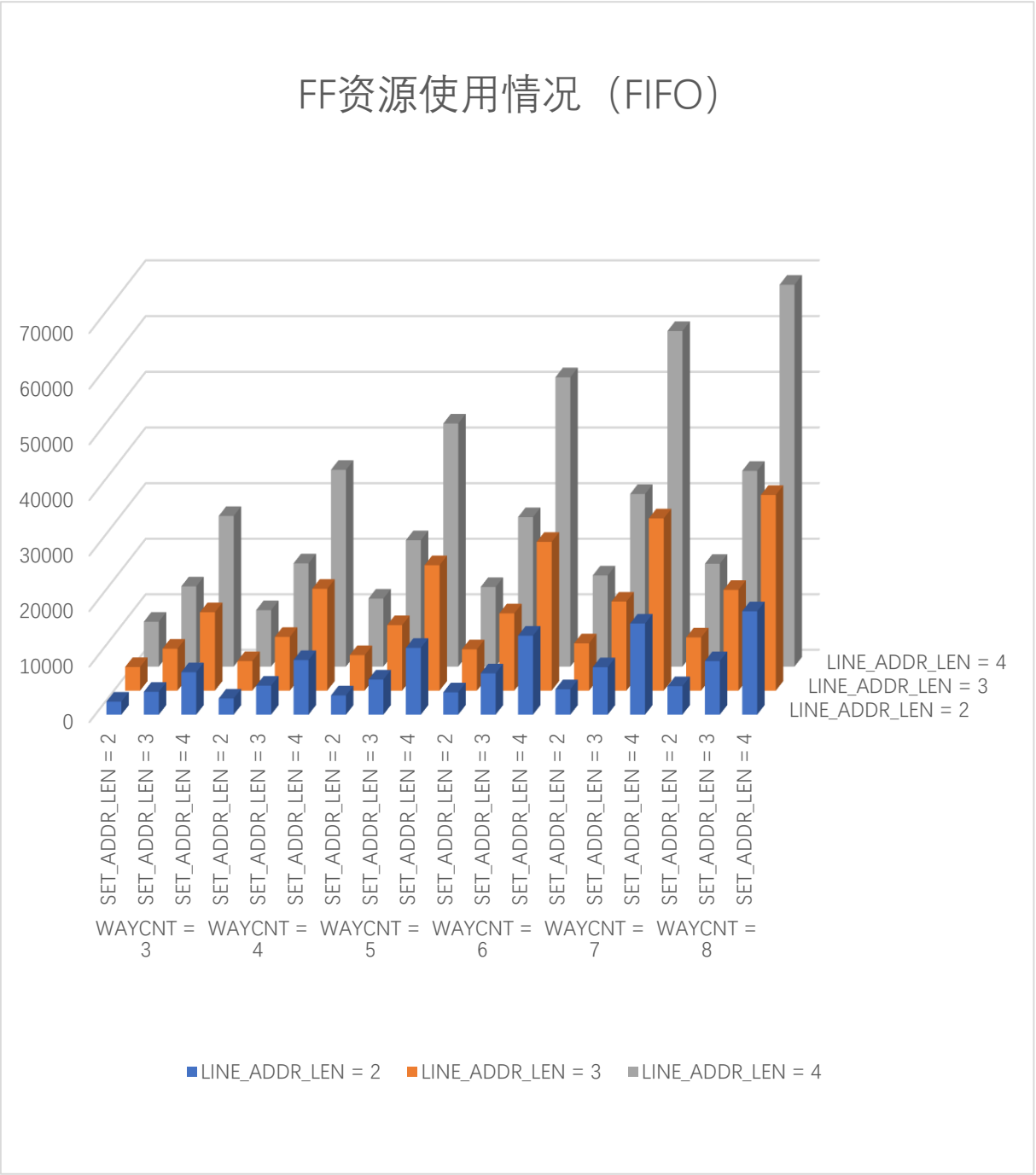
查找表（Look-Up-Table）简称为 LUT，LUT 本质上就是一个 RAM。目前 FPGA 中多使用 4 输入的 LUT，所以每一个 LUT 可以看成一个有 4 位地址线的的 RAM。当用户通过原理图或 HDL 语言描述了一个逻辑电路以后，PLD/FPGA 开发软件会自动计算逻辑电路的所有可能结果，并把真值表（即结果）事先写入 RAM，这样，每输入一个信号进行逻辑运算就等于输入一个地址进行查表，找出地址对应的内容，然后输出即可。

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

(2)不同参数下的 FF 使用情况

FF（FIFO）			
WAYCNT = 3			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	2357	4129	7652
3	4262	7558	14128
4	8087	14443	27133
WAYCNT = 4			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	2913	5233	9843
3	5326	9678	18352
4	10171	18603	35437
WAYCNT = 5			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3469	6337	12035
3	6390	11798	22595
4	12255	22768	43739
WAYCNT = 6			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4025	7441	14229
3	7454	13918	26803
4	14339	26923	52049
WAYCNT = 7			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4581	8545	16421
3	8518	16038	31024
4	16423	31089	60351
WAYCNT = 8			
SET_ADDR_LEN1 LINE_ADDR_LEN	2	3	4
2	5137	9649	18632
3	9582	18158	35252
4	18515	35243	68658

可得图表如下



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：

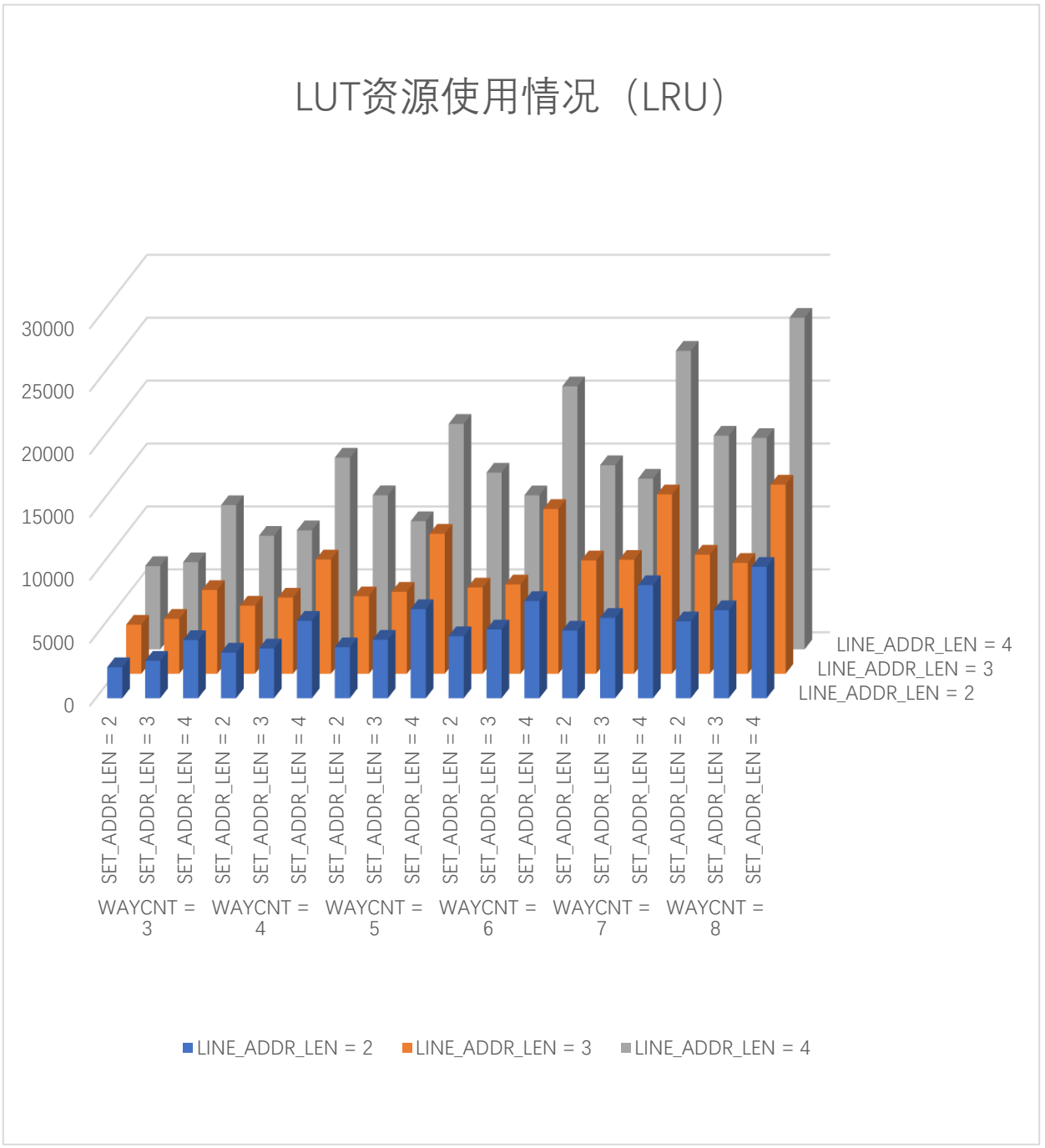
1. 每个 LINE 中 WORD 的数量（2^LINE_ADDR_LEN）
2. SET 的数量（2^SET_ADDR_LEN）
3. 每个 SET 中的 WAY 的数量（WAY_CNT）

3. 组相联之 LRU 策略

(1)不同参数下的 LUT 使用情况

LUT（LRU）			
WAYCNT = 3			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	2460	2968	4618
3	3897	4367	6659
4	6616	6924	11462
WAYCNT = 4			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3624	3952	6148
3	5409	6052	9083
4	9028	9459	15237
WAYCNT = 5			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4049	4648	7077
3	6156	6519	11130
4	12248	10187	17919
WAYCNT = 6			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4927	5461	7720
3	6854	7100	13096
4	14046	12244	20904
WAYCNT = 7			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	5370	6382	8985
3	9011	9055	14257
4	14639	13566	23728
WAYCNT = 8			
SET_ADDR_LEN1 LINE_ADDR_LEN	2	3	4
2	6110	6993	10436
3	9466	8813	15033
4	16985	16800	26353

可得图表如下



从图中我们可以看到，LUT 资源的占用和以下几个因素呈正相关：

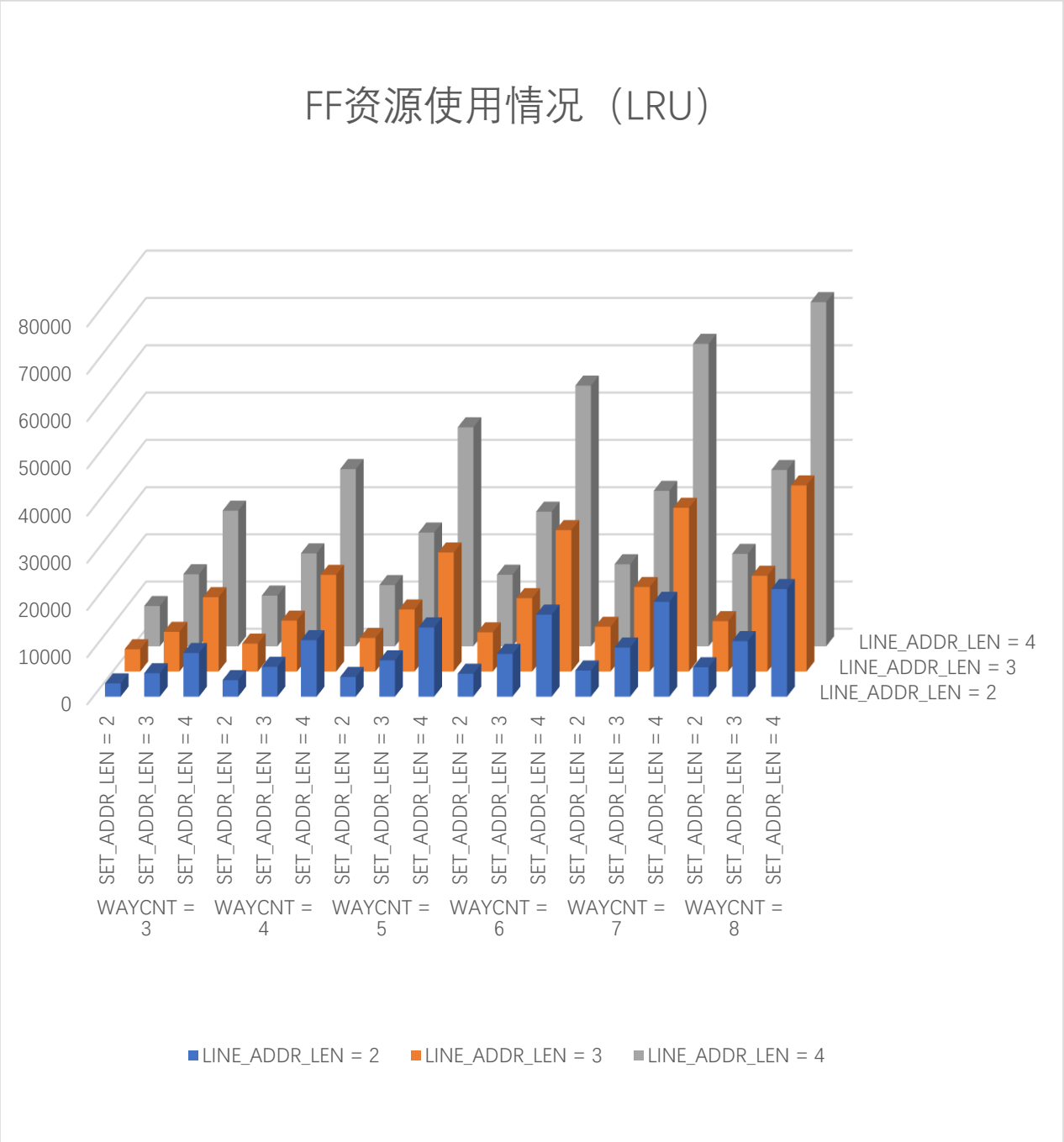
1. 每个 LINE 中 WORD 的数量（2^LINE_ADDR_LEN）
2. SET 的数量（2^SET_ADDR_LEN）
3. 每个 SET 中的 WAY 的数量（WAY_CNT）

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

(2)不同参数下的 FF 使用情况

FF（LRU）			
WAYCNT = 3			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	2809	4969	9264
3	4714	8398	15739
4	8541	15279	28744
WAYCNT = 4			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3489	6329	11973
3	5902	10774	20483
4	10747	19704	37565
WAYCNT = 5			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4173	7692	14676
3	7098	13151	25221
4	12973	24114	46386
WAYCNT = 6			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4861	9054	17384
3	8292	15530	29959
4	15175	28535	55203
WAYCNT = 7			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	5541	10417	20096
3	9482	17908	34696
4	17383	32956	64024
WAYCNT = 8			
SET_ADDR_LEN1 LINE_ADDR_LEN	2	3	4
2	6225	11777	22803
3	10672	20287	39437
4	19595	37369	72846

可得图表如下



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，不同参数下的命中数/缺失数

SET_ADDR_LEN LINE_ADDR_LEN	2	3	4	5
2	4269/947	4554/662	4767/449	4926/290
3	4701/515	4915/301	5030/186	5132/84
4	4191/275	5067/149	5152/64	5196/20
5	5030/186	5141/75	5206/10	5206/10

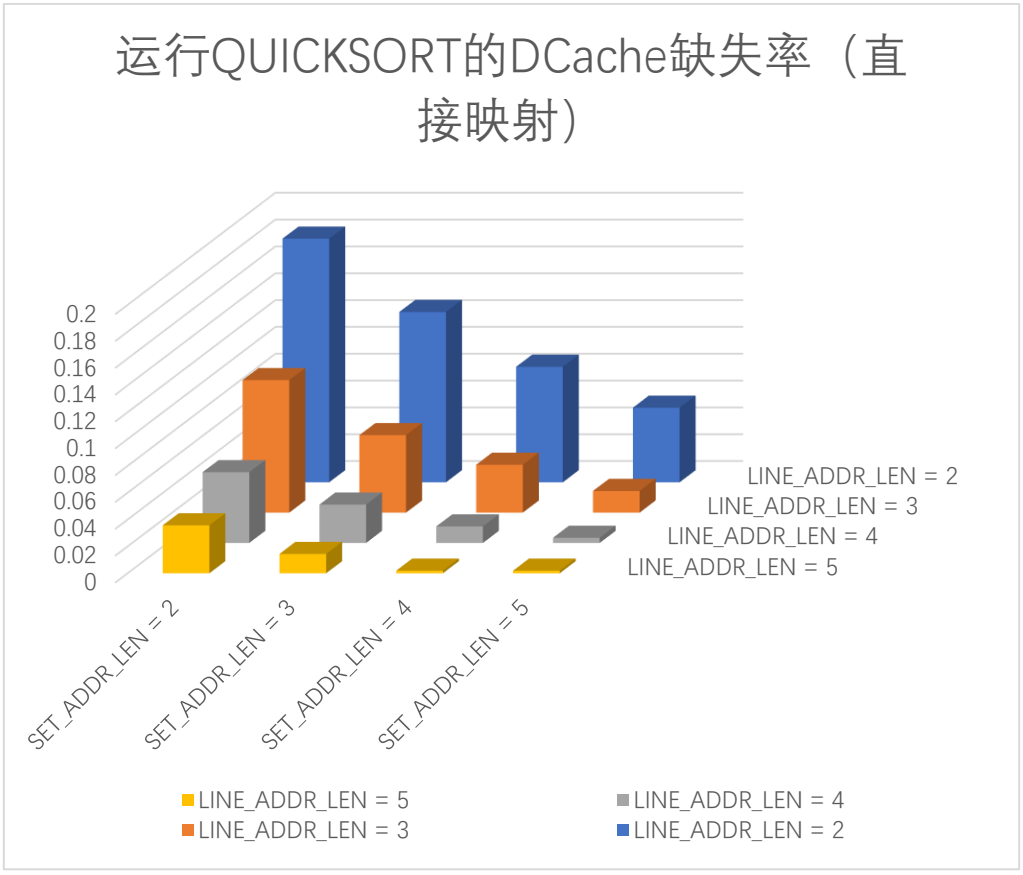
计算得 DCache 缺失率如下（总访存次数 5216 次）

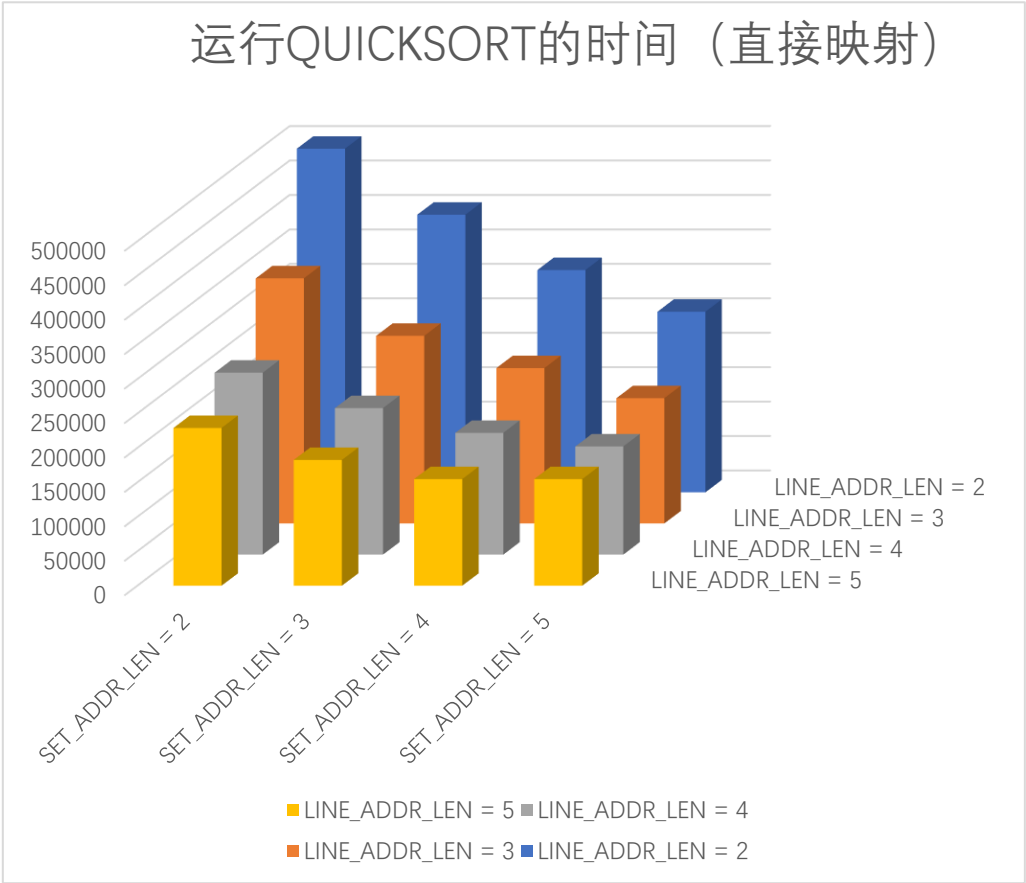
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4	5
2	0.181556748	0.126917178	0.086081288	0.05559816
3	0.098734663	0.057707055	0.035659509	0.016104294
4	0.052722393	0.028565951	0.012269939	0.003834356
5	0.035659509	0.014378834	0.001917178	0.001917178

运行时间如下，单位 ns

SET_ADDR_LEN LINE_ADDR_LEN	2	3	4	5
2	498848	402976	322656	262104
3	355804	272372	225496	181844
4	264032	212712	176692	156996
5	229008	182604	154836	154836

可得图表如下


运行QUICKSORT的DCache缺失率（直接映射）
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
SET_ADDR_LEN = 5
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 5
LINE_ADDR_LEN = 5
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2


运行QUICKSORT的时间（直接映射）
500000
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
SET_ADDR_LEN = 5
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 5
LINE_ADDR_LEN = 5
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

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。

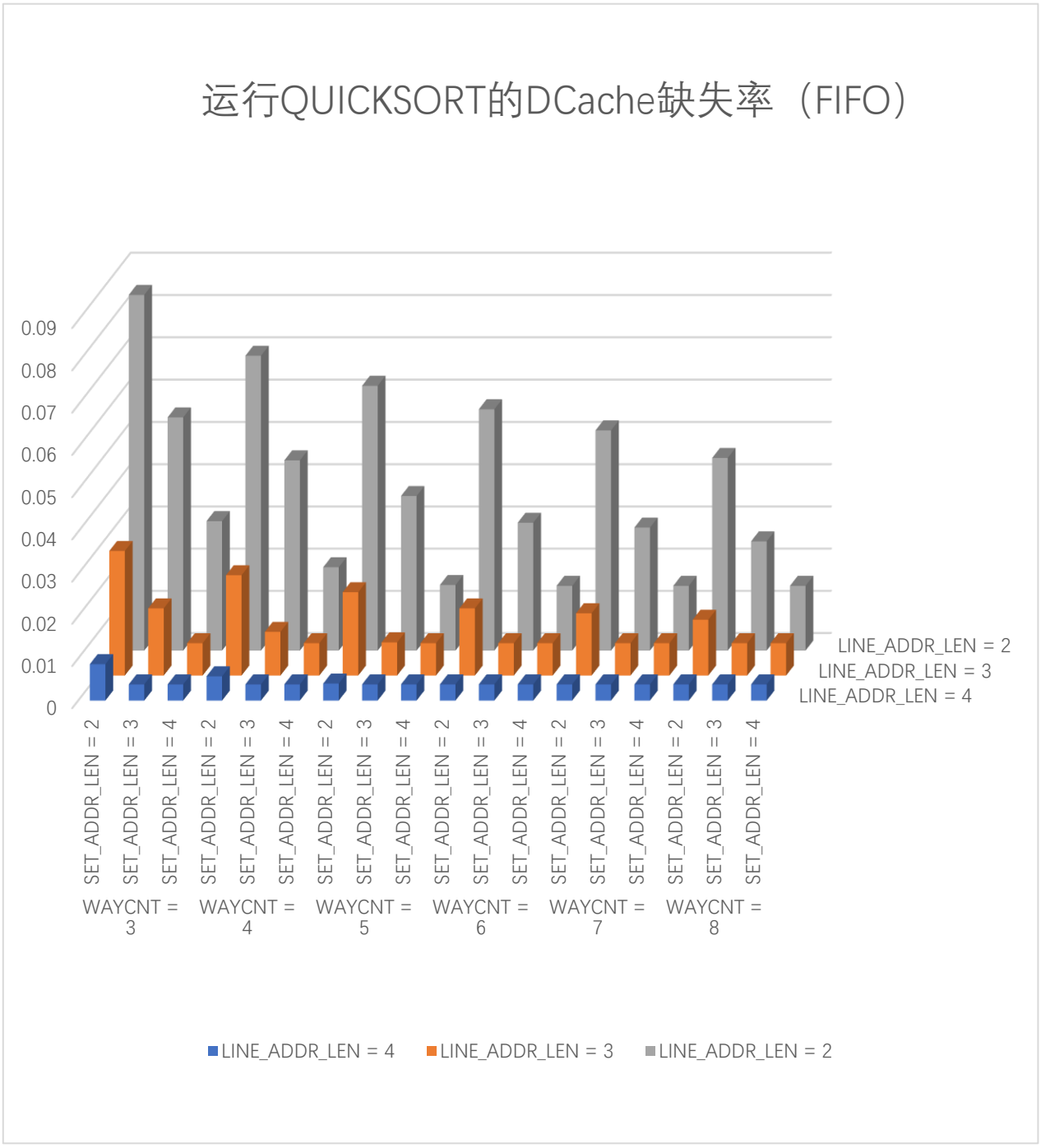
2. 组相联之 FIFO 策略，不同参数下的命中数/缺失数

WAYCNT = 3, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4777/439	4928/288	5056/160
3	5062/154	5133/83	5176/40
4	5171/45	5196/20	5196/20
WAYCNT = 4, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4852/364	4981/235	5113/103
3	5092/124	5162/54	5176/40
4	5186/30	5196/20	5196/20
WAYCNT = 5, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4889/327	5025/191	5135/81
3	5113/103	5175/41	5176/40
4	5195/21	5196/20	5196/20
WAYCNT = 6, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4918/298	5058/158	5136/80
3	5133/83	5176/40	5176/40
4	5196/20	5196/20	5196/20
WAYCNT = 7, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4944/272	5064/152	5136/80
3	5139/77	5176/40	5176/40
4	5196/20	5196/20	5196/20
WAYCNT = 8, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4978/238	5081/135	5136/80
3	5147/69	5176/40	5176/40
4	5196/20	5196/20	5196/20

计算得 DCache 缺失率如下（总访存次数 5216 次）

WAYCNT = 3, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.08416411	0.0552147	0.0306748
3	0.02952454	0.0159126	0.0076687
4	0.008627301	0.0038344	0.0038344
WAYCNT = 4, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.069785276	0.045053681	0.019746933
3	0.023773006	0.010352761	0.007668712
4	0.005751534	0.003834356	0.003834356
WAYCNT = 5, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.062691718	0.036618098	0.015529141
3	0.019746933	0.007860429	0.007668712
4	0.004026074	0.003834356	0.003834356
WAYCNT = 6, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.057131902	0.030291411	0.015337423
3	0.015912577	0.007668712	0.007668712
4	0.003834356	0.003834356	0.003834356
WAYCNT = 7, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.052147239	0.029141104	0.015337423
3	0.01476227	0.007668712	0.007668712
4	0.003834356	0.003834356	0.003834356
WAYCNT = 8, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.045628834	0.025881902	0.015337423
3	0.013228528	0.007668712	0.007668712
4	0.003834356	0.003834356	0.003834356

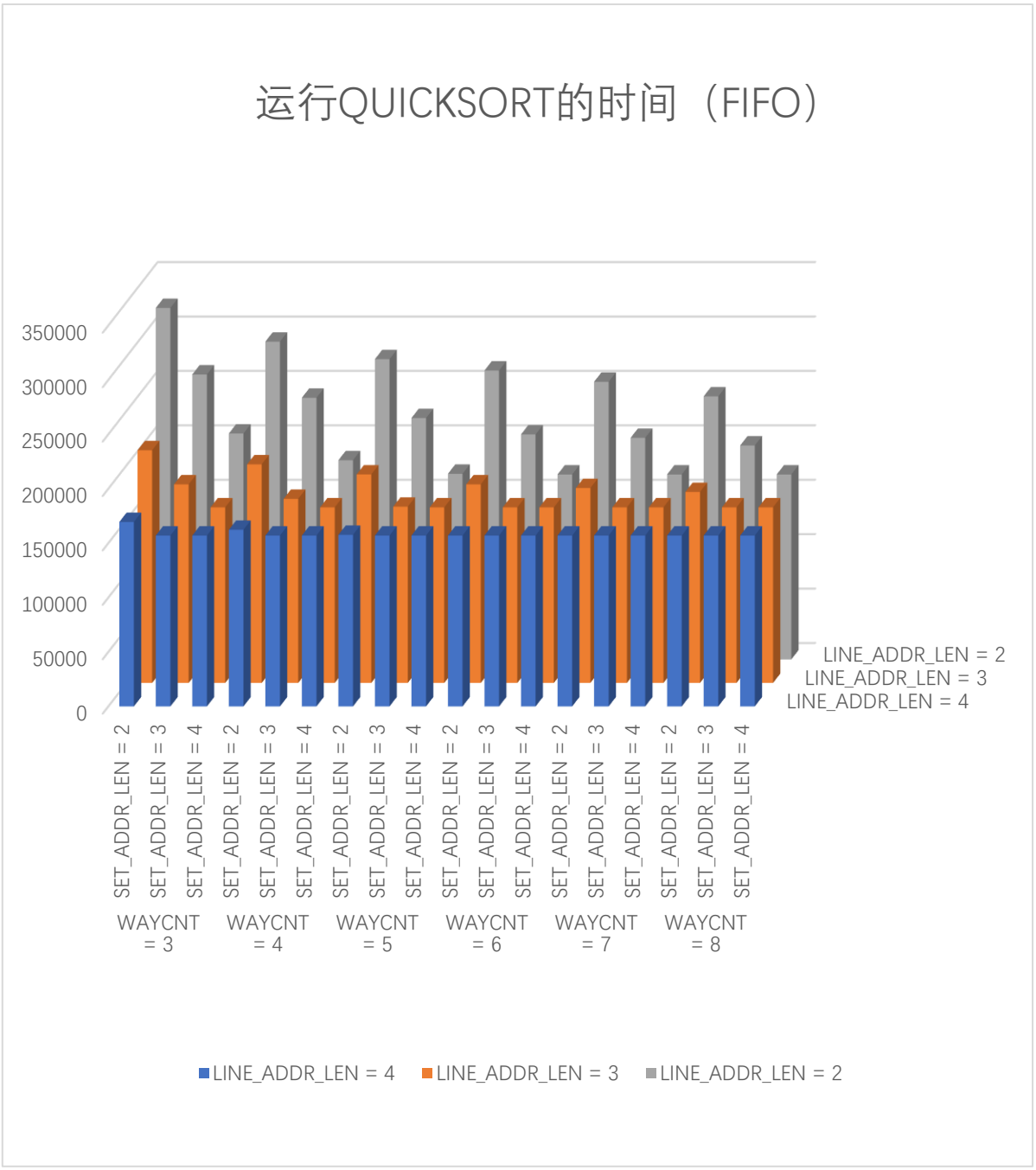
可得图表如下


运行QUICKSORT的DCache缺失率（FIFO）
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
WAYCNT = 3
WAYCNT = 4
WAYCNT = 5
WAYCNT = 6
WAYCNT = 7
WAYCNT = 8
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

运行时间如下，单位 ns

WAYCNT = 3, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	322984	261880	207612
3	213800	182452	161316
4	169460	156996	156996
WAYCNT = 4, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	292016	240448	183032
3	200872	169120	161316
4	162272	156996	156996
WAYCNT = 5, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	275916	221796	170588
3	191556	161948	161316
4	157628	156996	156996
WAYCNT = 6, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	265496	206764	169956
3	182452	161316	161316
4	156996	156996	156996
WAYCNT = 7, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	255092	203596	169956
3	179076	161316	161316
4	156996	156996	156996
WAYCNT = 8, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	241720	196596	169956
3	175476	161316	161316
4	156996	156996	156996

可得图表如下


运行QUICKSORT的时间（FIFO）
350000
300000
250000
200000
150000
100000
50000
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
WAYCNT = 3
WAYCNT = 4
WAYCNT = 5
WAYCNT = 6
WAYCNT = 7
WAYCNT = 8
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

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。

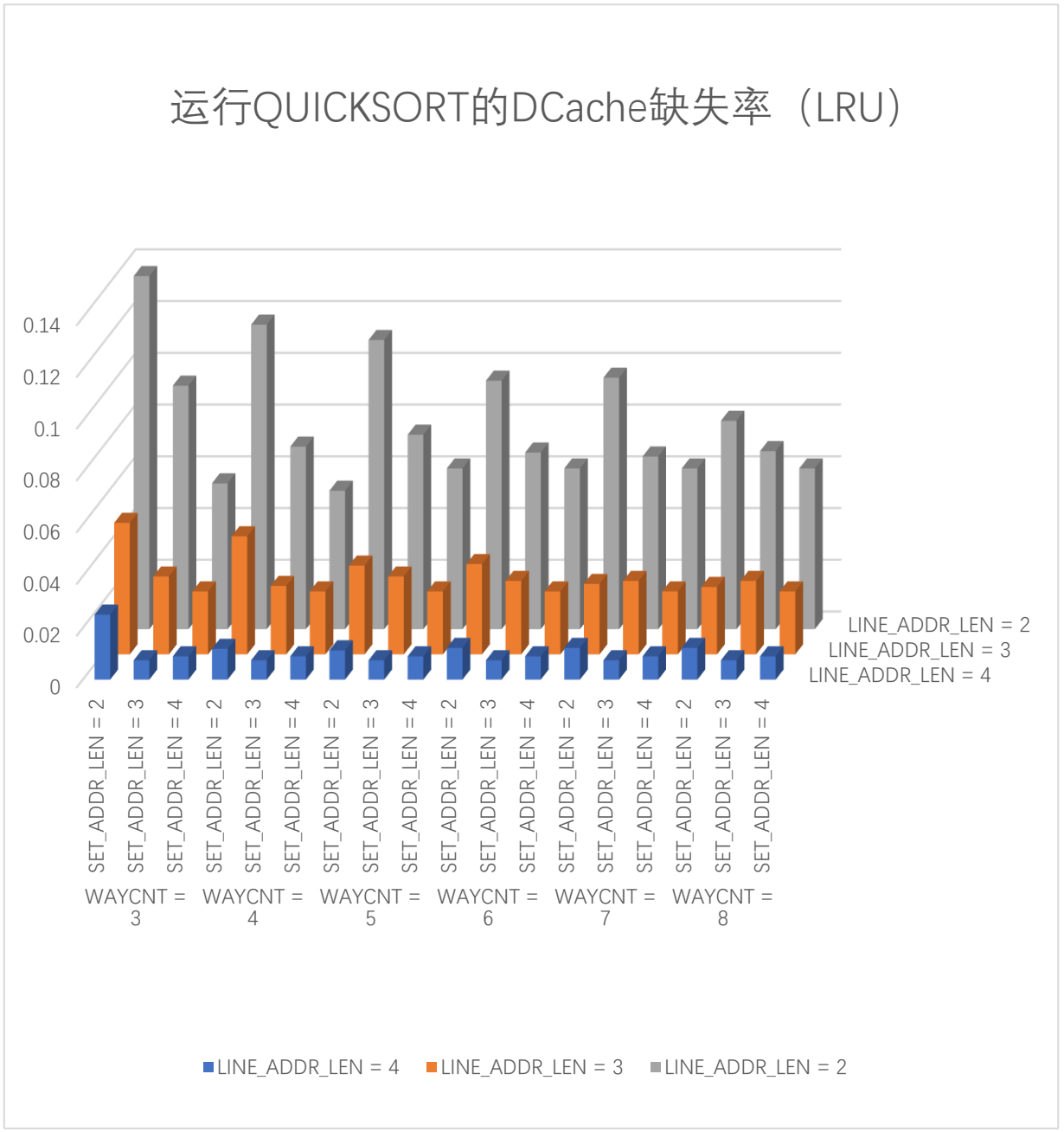
3. 组相联之 LRU 策略，不同参数下的命中数/缺失数

WAYCNT = 3, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4504/712	4725/491	4922/294
3	4951/265	5059/157	5089/127
4	5085/131	5177/39	5169/47
WAYCNT = 4, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4602/614	4848/368	4937/279
3	4978/238	5078/138	5089/127
4	5154/62	5177/39	5169/47
WAYCNT = 5, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4633/583	4824/392	4892/324
3	5037/179	5059/157	5089/127
4	5157/59	5177/39	5169/47
WAYCNT = 6, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4715/501	4860/356	4892/324
3	5034/182	5068/148	5089/127
4	5152/64	5177/39	5169/47
WAYCNT = 7, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4709/507	4868/348	4892/324
3	5074/142	5068/148	5089/127
4	5152/64	5177/39	5169/47
WAYCNT = 8, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4796/420	4857/359	4892/324
3	5080/136	5068/148	5089/127
4	5152/64	5177/39	5169/47

计算得 DCache 缺失率如下（总访存次数 5216 次）

WAYCNT = 3, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.136503067	0.094133436	0.056365031
3	0.050805215	0.030099693	0.02434816
4	0.025115031	0.007476994	0.009010736
WAYCNT = 4, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.117714724	0.070552147	0.053489264
3	0.045628834	0.026457055	0.02434816
4	0.011886503	0.007476994	0.009010736
WAYCNT = 5, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.111771472	0.075153374	0.062116564
3	0.034317485	0.030099693	0.02434816
4	0.01131135	0.007476994	0.009010736
WAYCNT = 6, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.096050613	0.068251534	0.062116564
3	0.034892638	0.028374233	0.02434816
4	0.012269939	0.007476994	0.009010736
WAYCNT = 7, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.09720092	0.066717791	0.062116564
3	0.027223926	0.028374233	0.02434816
4	0.012269939	0.007476994	0.009010736
WAYCNT = 8, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.080521472	0.068826687	0.062116564
3	0.02607362	0.028374233	0.02434816
4	0.012269939	0.007476994	0.009010736

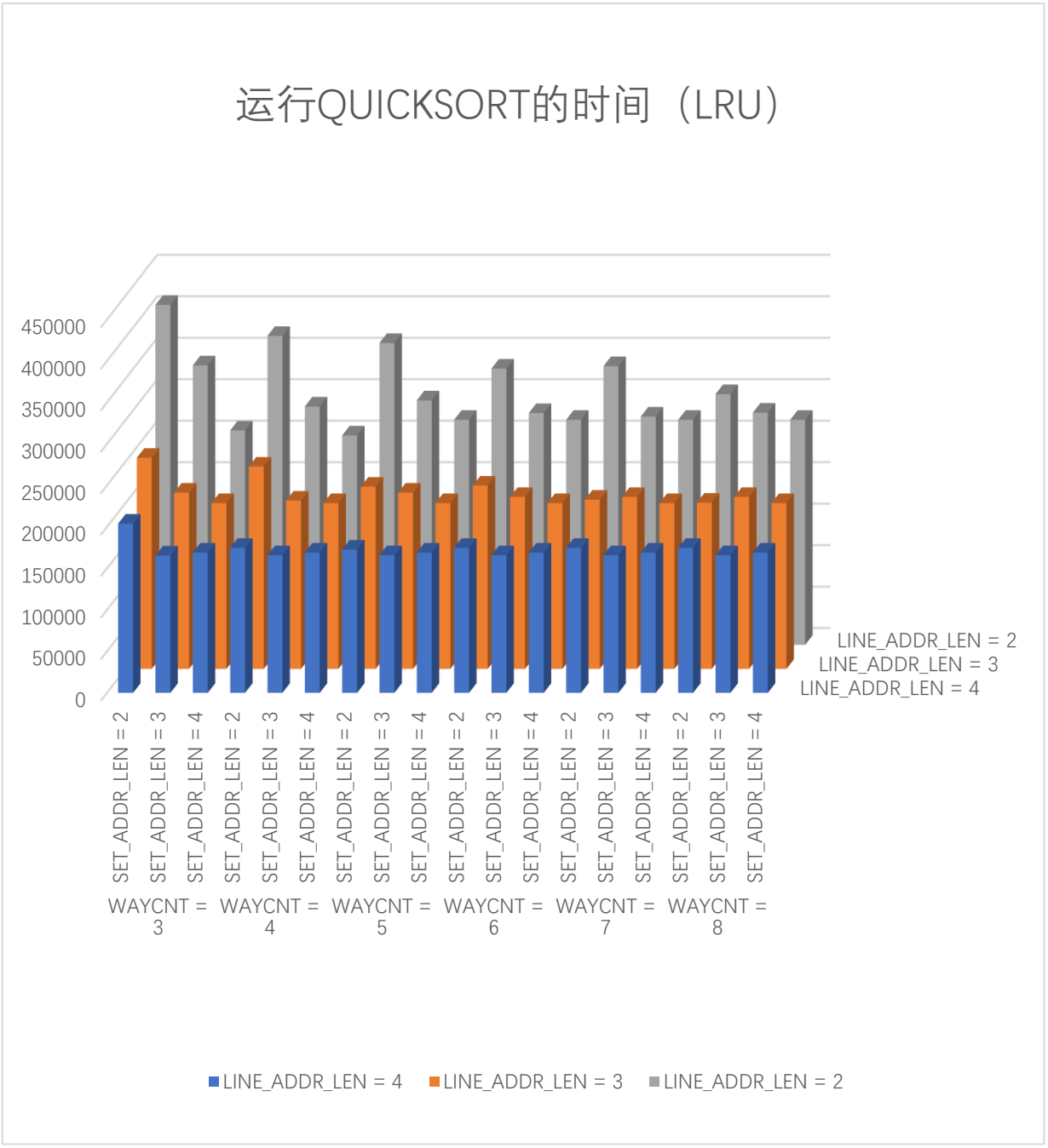
可得图表如下


运行QUICKSORT的DCache缺失率（LRU）
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
WAYCNT = 3
WAYCNT = 4
WAYCNT = 5
WAYCNT = 6
WAYCNT = 7
WAYCNT = 8
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

运行时间如下，单位 ns

WAYCNT = 3, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	409824	336920	258392
3	254404	212368	199868
4	204032	165468	169068
WAYCNT = 4, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	372432	287264	252452
3	243800	202860	199868
4	174788	165884	169068
WAYCNT = 5, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	363860	294740	271324
3	219612	212368	199868
4	172684	165884	169068
WAYCNT = 6, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	333040	279476	271324
3	220884	207308	199868
4	174804	165884	169068
WAYCNT = 7, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	336000	275256	271324
3	203924	207308	199868
4	174804	165884	169068
WAYCNT = 8, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	302240	279920	271324
3	200344	207308	199868
4	174804	165884	169068

可得图表如下


运行QUICKSORT的时间（LRU）
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
WAYCNT = 3
WAYCNT = 4
WAYCNT = 5
WAYCNT = 6
WAYCNT = 7
WAYCNT = 8
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

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，不同参数下的命中数/缺失数

SET_ADDR_LEN LINE_ADDR_LEN	2	3	4	5
2	2976/5472	3360/5088	3552/4896	3656/4792
3	2912/5536	3360/5088	3584/4864	7484/964
4	2880/5568	3360/5088	7410/1038	7095/543
5	4416/4032	6930/1518	7921/527	8424/24

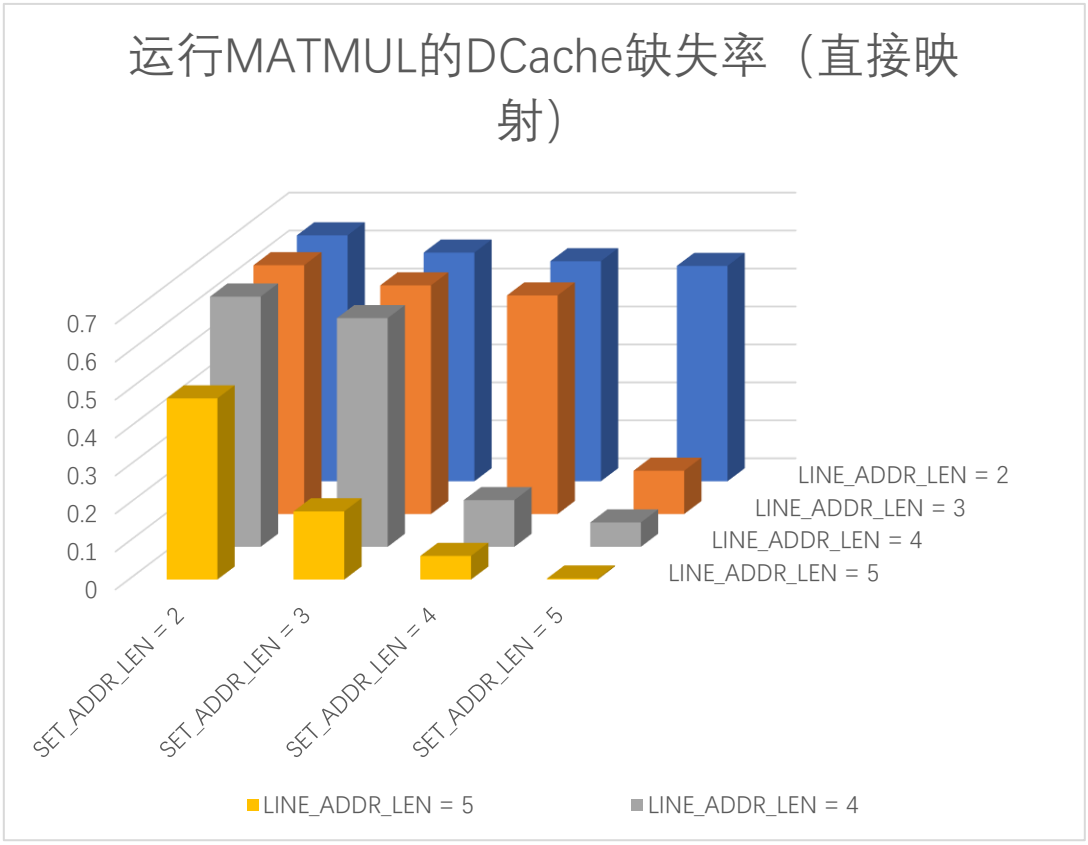
计算得 DCache 缺失率如下（总访存次数 8448 次）

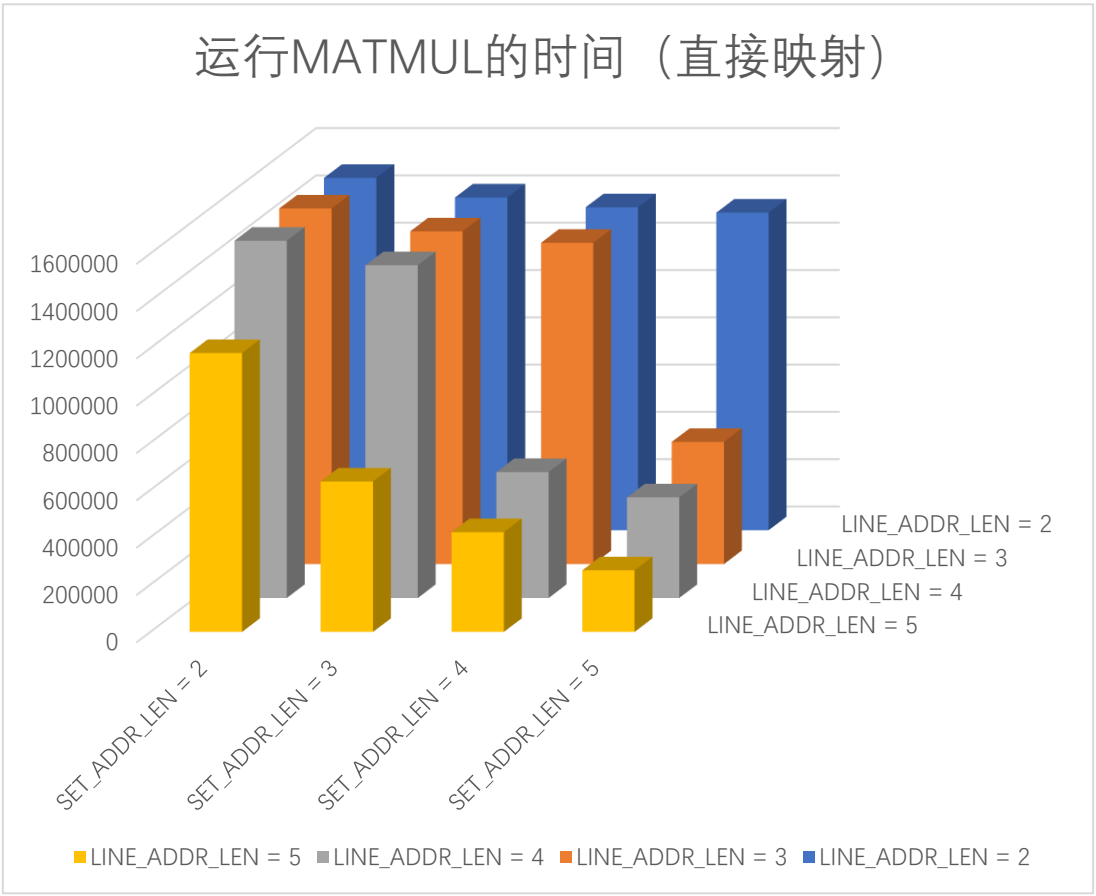
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4	5
2	0.647727273	0.602272727	0.579545455	0.567234848
3	0.65530303	0.602272727	0.575757576	0.114109848
4	0.659090909	0.602272727	0.122869318	0.064275568
5	0.477272727	0.1796875	0.062381629	0.002840909

运行时间如下，单位 ns

SET_ADDR_LEN LINE_ADDR_LEN	2	3	4	5
2	1489992	1407048	1365576	1343112
3	1503784	1407048	1358632	516232
4	1510632	1406952	532152	425232
5	1178728	635704	421648	260428

可得图表如下


运行MATMUL的DCache缺失率（直接映射）
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
SET_ADDR_LEN = 5
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 5
LINE_ADDR_LEN = 5
LINE_ADDR_LEN = 4


运行MATMUL的时间（直接映射）
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
SET_ADDR_LEN = 5
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 5
LINE_ADDR_LEN = 5
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

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。

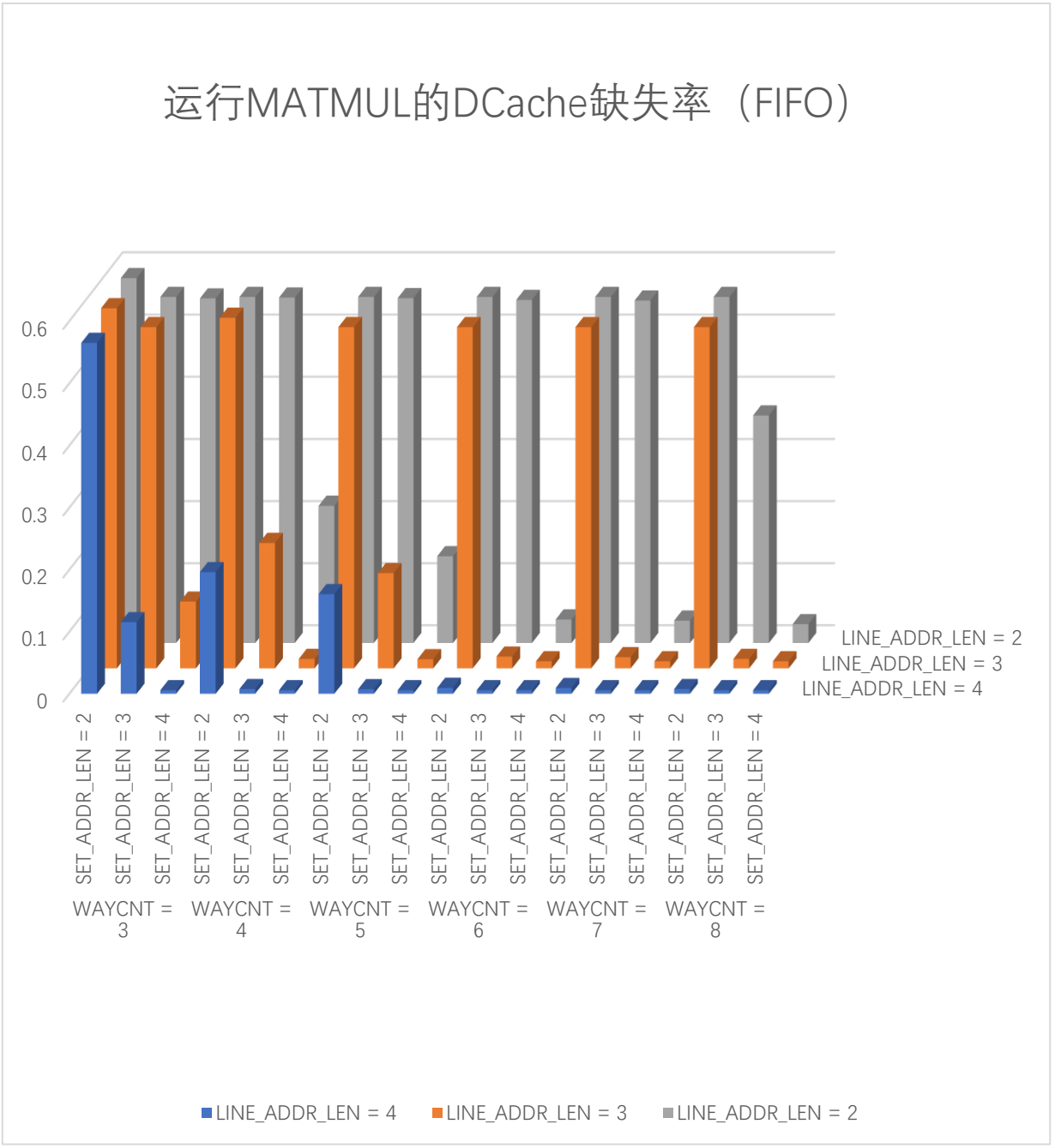
2. 组相联之 FIFO 策略，不同参数下的命中数/缺失数

WAYCNT = 3, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3488/4960	3744/4704	3764/4684
3	3552/4896	3808/4640	7539/909
4	3676/4772	7474/974	8400/48
WAYCNT = 4, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3744/4704	3752/4696	6560/1868
3	3680/4768	6740/1708	8320/128
4	6792/1656	8384/64	8400/48
WAYCNT = 5, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3744/4704	3760/4688	7267/1181
3	3808/4640	7151/1297	8324/124
4	7092/1356	8386/62	8400/48
WAYCNT = 6, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3744/4704	3784/4664	8128/320
3	3808/4640	8288/160	8352/96
4	8368/80	8400/48	8400/48
WAYCNT = 7, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3744/4704	3792/4656	8144/304
3	3808/4640	8296/152	8352/96
4	8372/78	8400/48	8400/48
WAYCNT = 8, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3744/4704	5352/3096	8192/256
3	3808/4640	8320/128	8352/96
4	8384/64	8400/48	8400/48

计算得 DCache 缺失率如下（总访存次数 8448 次）

WAYCNT = 3, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.587121212	0.556818182	0.55445076
3	0.579545455	0.549242424	0.10759943
4	0.564867424	0.115293561	0.00568182
WAYCNT = 4, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.556818182	0.555871212	0.22111742
3	0.564393939	0.20217803	0.01515152
4	0.196022727	0.007575758	0.00568182
WAYCNT = 5, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.556818182	0.554924242	0.1397964
3	0.549242424	0.153527462	0.01467803
4	0.160511364	0.007339015	0.00568182
WAYCNT = 6, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.556818182	0.552083333	0.03787879
3	0.549242424	0.018939394	0.01136364
4	0.009469697	0.005681818	0.00568182
WAYCNT = 7, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.556818182	0.551136364	0.03598485
3	0.549242424	0.017992424	0.01136364
4	0.009232955	0.005681818	0.00568182
WAYCNT = 8, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.556818182	0.366477273	0.03030303
3	0.549242424	0.015151515	0.01136364
4	0.007575758	0.005681818	0.00568182

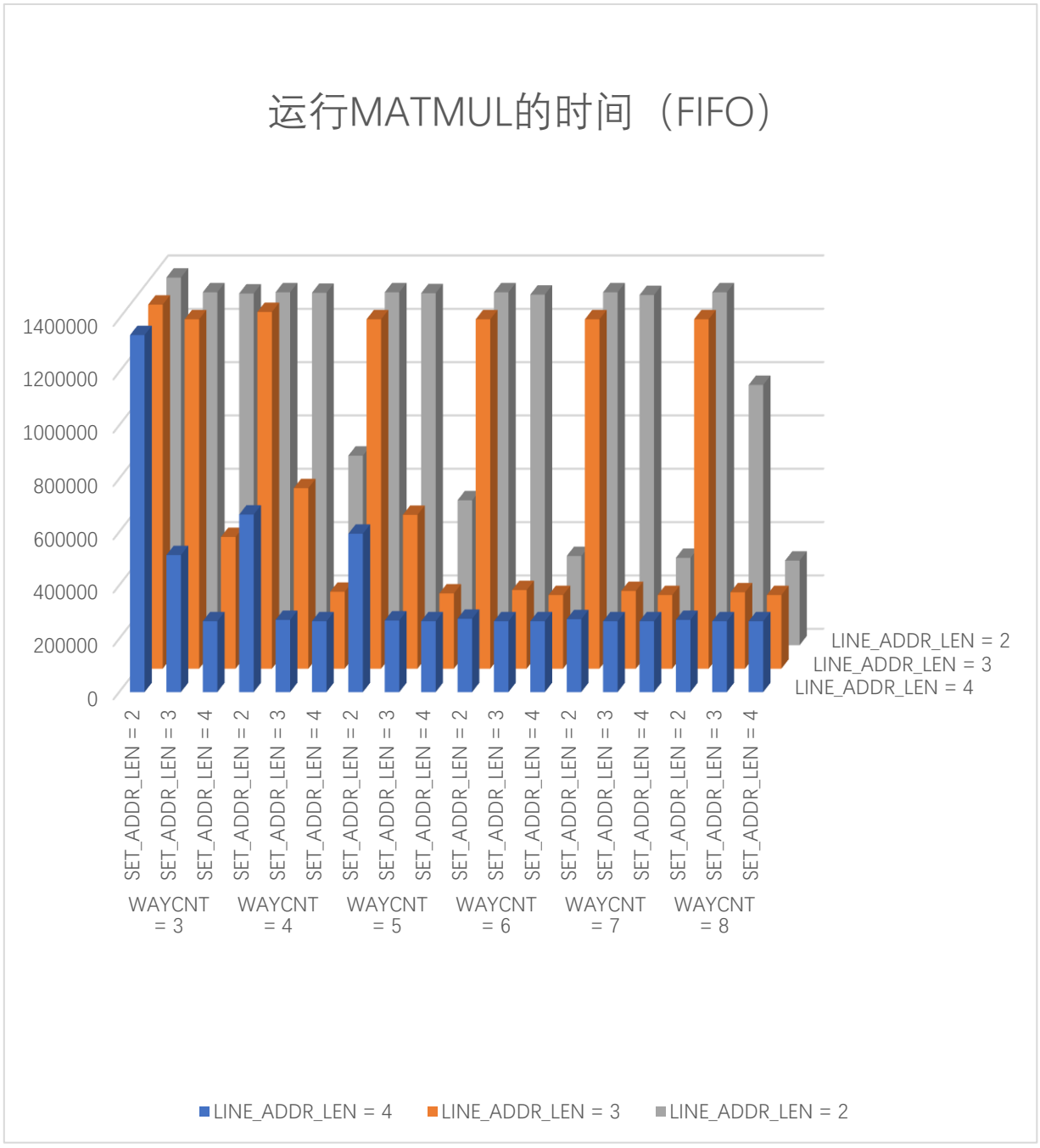
可得图表如下


运行MATMUL的DCache缺失率（FIFO）
0.6
0.5
0.4
0.3
0.2
0.1
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
WAYCNT = 3
WAYCNT = 4
WAYCNT = 5
WAYCNT = 6
WAYCNT = 7
WAYCNT = 8
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

运行时间如下，单位 ns

WAYCNT = 3, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1378776	1323480	1319160
3	1365336	1310040	493752
4	1338696	513760	265612
WAYCNT = 4, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1323480	1321752	710904
3	1337688	676728	288216
4	665640	270728	265612
WAYCNT = 5, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1323480	1320024	543384
3	1310040	576728	282028
4	593776	268636	265612
WAYCNT = 6, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1323480	1314840	334344
3	1310040	294792	275980
4	275016	265612	265612
WAYCNT = 7, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1323480	1313112	327560
3	1310040	291400	275980
4	273320	265612	265612
WAYCNT = 8, FIFO			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1323480	976152	317192
3	1310040	286216	275980
4	270728	265612	265612

可得图表如下


运行MATMUL的时间（FIFO）
1400000
1200000
1000000
800000
600000
400000
200000
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
WAYCNT = 3
WAYCNT = 4
WAYCNT = 5
WAYCNT = 6
WAYCNT = 7
WAYCNT = 8
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

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。

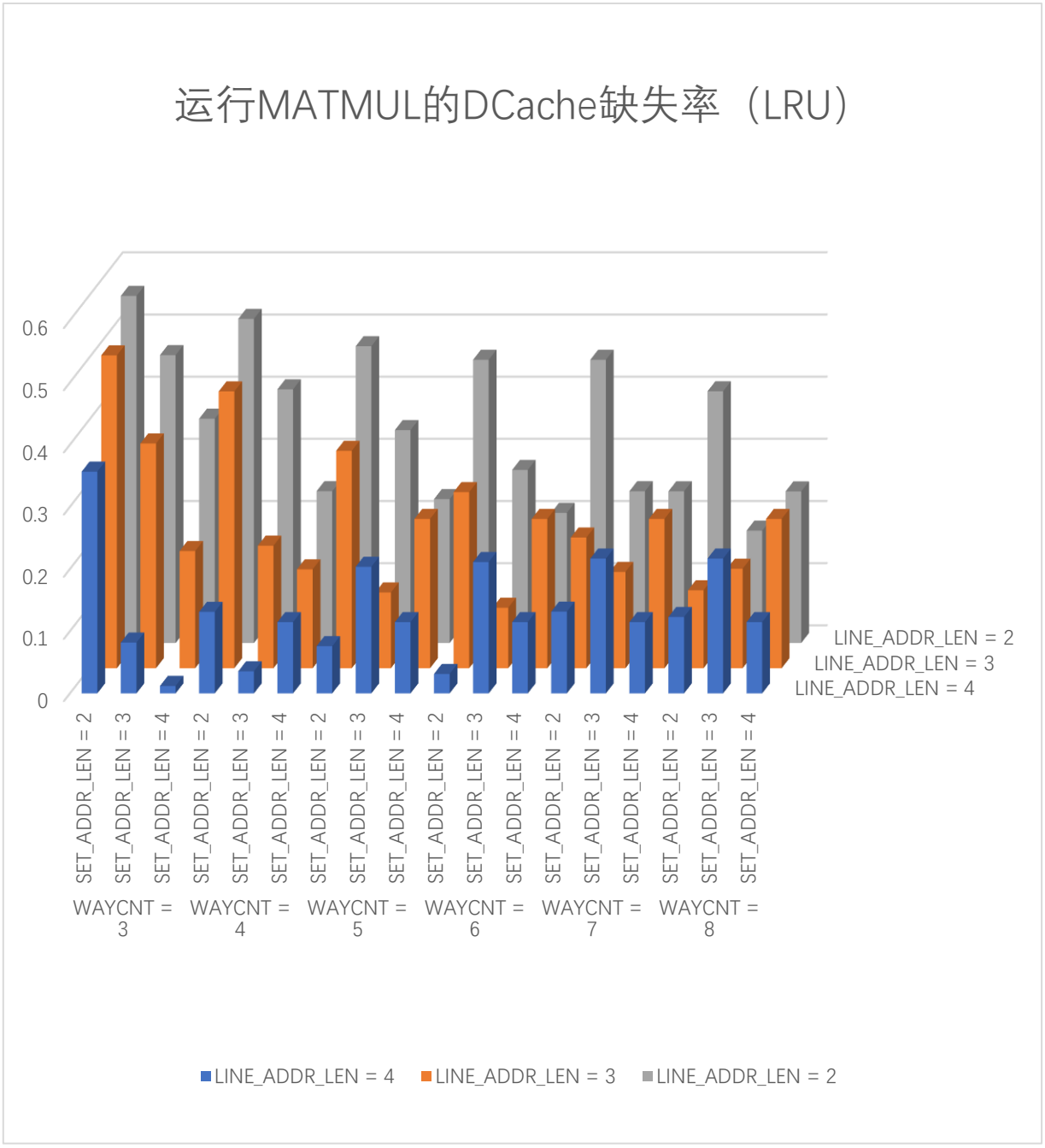
3. 组相联之 LRU 策略，不同参数下的命中数/缺失数

WAYCNT = 3, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	3732/4716	4537/3911	5401/3047
3	4198/4250	5395/3053	6857/1591
4	5436/3012	7758/690	8347/101
WAYCNT = 4, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4044/4404	5002/3446	6386/2062
3	4688/3760	6784/1664	7105/1343
4	7341/1107	8148/300	7480/968
WAYCNT = 5, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4416/4032	5556/2892	6494/1954
3	5495/2953	7419/1029	6420/2028
4	7807/641	6728/1720	7480/968
WAYCNT = 6, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4600/3848	6097/2351	6680/1768
3	6057/2391	7627/821	6420/2028
4	8184/264	6664/1784	7480/968
WAYCNT = 7, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	4600/3848	6387/2061	6388/2060
3	6673/1775	7140/1308	6420/2028
4	7338/1110	6617/1831	7480/968
WAYCNT = 8, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	5029/3419	6922/1526	6389/2059
3	7390/1058	7097/1351	6420/2028
4	7410/1038	6617/1831	7480/968

计算得 DCache 缺失率如下（总访存次数 8448 次）

WAYCNT = 3, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.558238636	0.46295	0.36068
3	0.503077652	0.36139	0.18833
4	0.356534091	0.08168	0.01196
WAYCNT = 4, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.521306818	0.40791	0.24408
3	0.445075758	0.19697	0.15897
4	0.131036932	0.03551	0.11458
WAYCNT = 5, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.477272727	0.34233	0.2313
3	0.349550189	0.1218	0.24006
4	0.075875947	0.2036	0.11458
WAYCNT = 6, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.455492424	0.27829	0.20928
3	0.283025568	0.09718	0.24006
4	0.03125	0.21117	0.11458
WAYCNT = 7, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.455492424	0.24396	0.24384
3	0.210108902	0.15483	0.24006
4	0.131392045	0.21674	0.11458
WAYCNT = 8, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	0.404711174	0.18063	0.24373
3	0.125236742	0.15992	0.24006
4	0.122869318	0.21674	0.11458

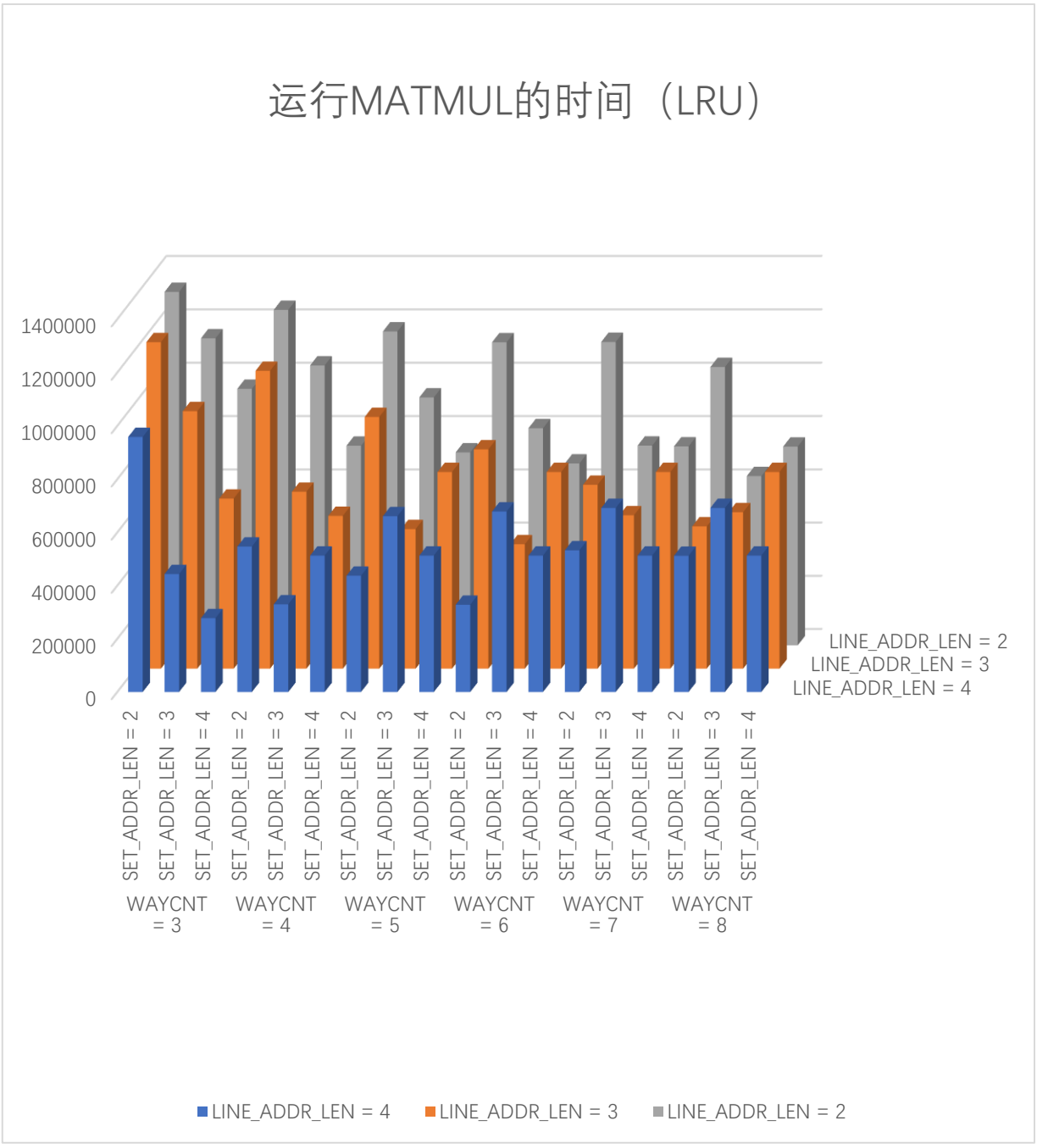
可得图表如下


运行MATMUL的DCache缺失率（LRU）
0.6
0.5
0.4
0.3
0.2
0.1
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
WAYCNT = 3
WAYCNT = 4
WAYCNT = 5
WAYCNT = 6
WAYCNT = 7
WAYCNT = 8
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

运行时间如下，单位 ns

WAYCNT = 3, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1326696	1152816	963696
3	1226008	967456	639656
4	958536	443672	277684
WAYCNT = 4, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1259304	1051128	750336
3	1118768	665352	574648
4	547056	329816	513204
WAYCNT = 5, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1178552	931256	725136
3	945856	525128	739468
4	437248	661292	513204
WAYCNT = 6, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1137960	815856	683716
3	824464	468136	739468
4	328072	678860	513204
WAYCNT = 7, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1139208	750744	747628
3	691408	577076	739468
4	533056	692124	513204
WAYCNT = 8, LRU			
SET_ADDR_LEN LINE_ADDR_LEN	2	3	4
2	1045296	636432	746996
3	535136	588452	739468
4	512308	692124	513204

可得图表如下


运行MATMUL的时间（LRU）
1400000
1200000
1000000
800000
600000
400000
200000
0
SET_ADDR_LEN = 2
SET_ADDR_LEN = 3
SET_ADDR_LEN = 4
WAYCNT = 3
WAYCNT = 4
WAYCNT = 5
WAYCNT = 6
WAYCNT = 7
WAYCNT = 8
LINE_ADDR_LEN = 2
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 4
LINE_ADDR_LEN = 3
LINE_ADDR_LEN = 2

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**