Lab3 实验报告

PB20000072 王铖潇

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

实验内容

实现替换策略

FIFO

FIFO每次替换出最早使用的块，对于每个SET，维护一个大小为`WAY_CNT`的数组，第`i`个数组元素表示“已经存在于内存中的时间为`i`的块”。每次需要换出存在内存中时间最长的块，即标号为`WAY_CNT - 1`的块。

```
if(strategy == `FIFO) begin
    way_addr = fifo[set_addr][WAY_CNT - 1];
end
```

换入新的块时，“已经存在于内存中的时间为`i - 1`的块”变成“已经存在于内存中的时间为`i`的块”，“已经存在于内存中的时间为`0`的块”是新换入的块。

```
for (integer i = WAY_CNT - 1; i > 0; i--) begin
    fifo[set_addr][i] = fifo[set_addr][i-1];
end
fifo[set_addr][0] = way_addr;
```

LRU

LRU每次替换出最长没有使用的块，对于每个SET，维护一个大小为WAY_CNT的数组标记每个块“距离换出的时间”。

换入或使用一个新的块时，其余的块中“距离换出的时间”比该块大的，说明是本来要在这个块之后换出，现在这个块被使用，换出的时间后移，所以要将那些块的“距离换出的时间”减一；并且将这个块的值设为`WAY_CNT`（最大值）。

```
for(integer i = 0; i < WAY_CNT; i++) begin
    if(way_addr != i) begin
        if(lru[set_addr][i] > lru[set_addr][way_addr])
        lru[set_addr][i] <= lru[set_addr][i] - 1;
    end
    else
        lru[set_addr][i] <= WAY_CNT - 1;
end
```

当一个块的"距离换出的时间"为0时，将它换出：

```
for (integer i = 0; i< WAY_CNT; i++) begin
    if(lru[set_addr][i] == 0) begin
        way_addr = i;
        break;
    end
end
```

Cache资源消耗评估

修改 Cache 的参数（组数、组相连度、line大小等）进行综合得到资源占用报告，其中 LUT 和 FF 两个参数的使用量代表 Cache 所占用电路的资源量，可用的资源分别是63400（LUT）和3061（FF），多次修改参数得到下表：

LRU

组数	组相联度	line大小	LUT	FF
8	1	8	1144	3061
8	2	8	1989	5228
8	4	8	3860	9561
4	4	8	3377	5270
16	4	8	7744	18114
8	4	4	2131	5109
8	4	16	7952	18495

按照行顺序的截图如下：

Resource	Utilization	Available	Utilization %
LUT	1144	63400	1.80
FF	3061	126800	2.41
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	1989	63400	3.14
FF	5228	126800	4.12
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	3860	63400	6.09
FF	9561	126800	7.54
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	3377	63400	5.33
FF	5270	126800	4.16
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	7744	63400	12.21
FF	18114	126800	14.29
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	2131	63400	3.36
FF	5109	126800	4.03
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	7952	63400	12.54
FF	18495	126800	14.59
BRAM	4	135	2.96
IO	81	210	38.57

FIFO

组数	组相联度	line大小	LUT	FF
8	1	8	1144	3058
8	2	8	2122	5183
8	4	8	3900	9455
4	4	8	4652	5218
16	4	8	7344	17914
8	4	4	2263	5011
8	4	16	8394	18404

按照行顺序的截图如下：

Resource	Utilization	Available	Utilization %
LUT	1144	63400	1.80
FF	3058	126800	2.41
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	2122	63400	3.35
FF	5183	126800	4.09
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	3900	63400	6.15
FF	9455	126800	7.46
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	4652	63400	7.34
FF	5218	126800	4.12
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	7344	63400	11.58
FF	17914	126800	14.13
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	2263	63400	3.57
FF	5011	126800	3.95
BRAM	4	135	2.96
IO	81	210	38.57

Resource	Utilization	Available	Utilization %
LUT	8394	63400	13.24
FF	18404	126800	14.51
BRAM	4	135	2.96
IO	81	210	38.57

Cache性能评估

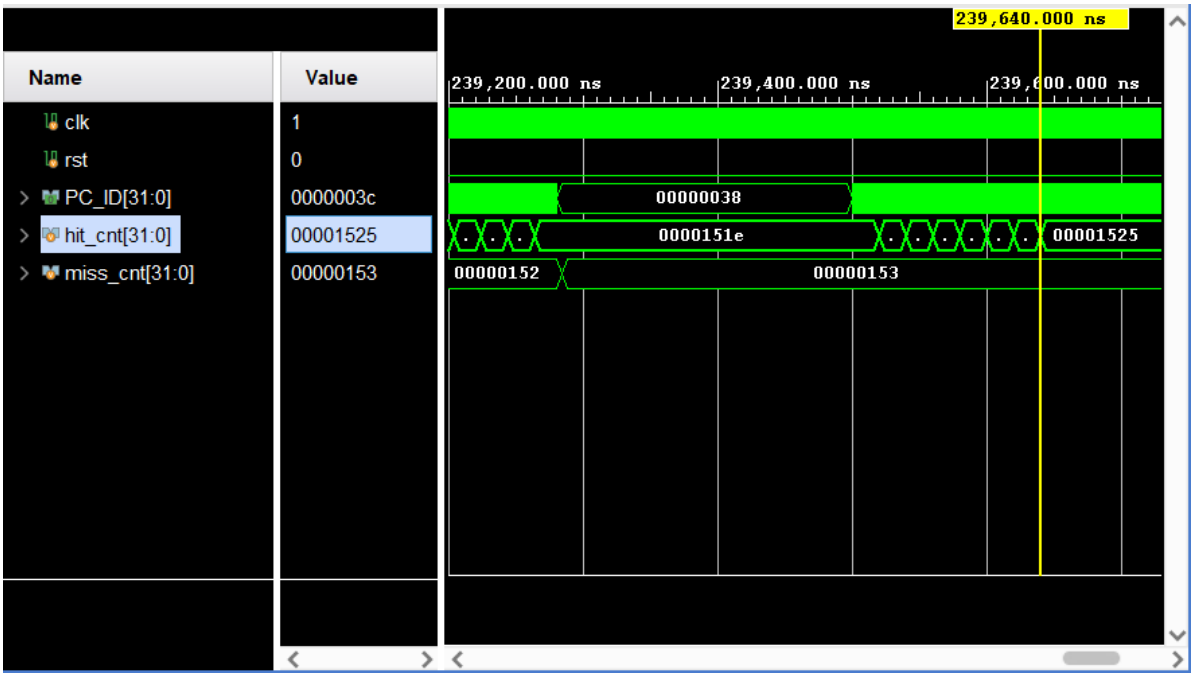
快速排序规模是256，矩阵乘法规模是16*16 ， 对LRU和FIFO分别进行测试。

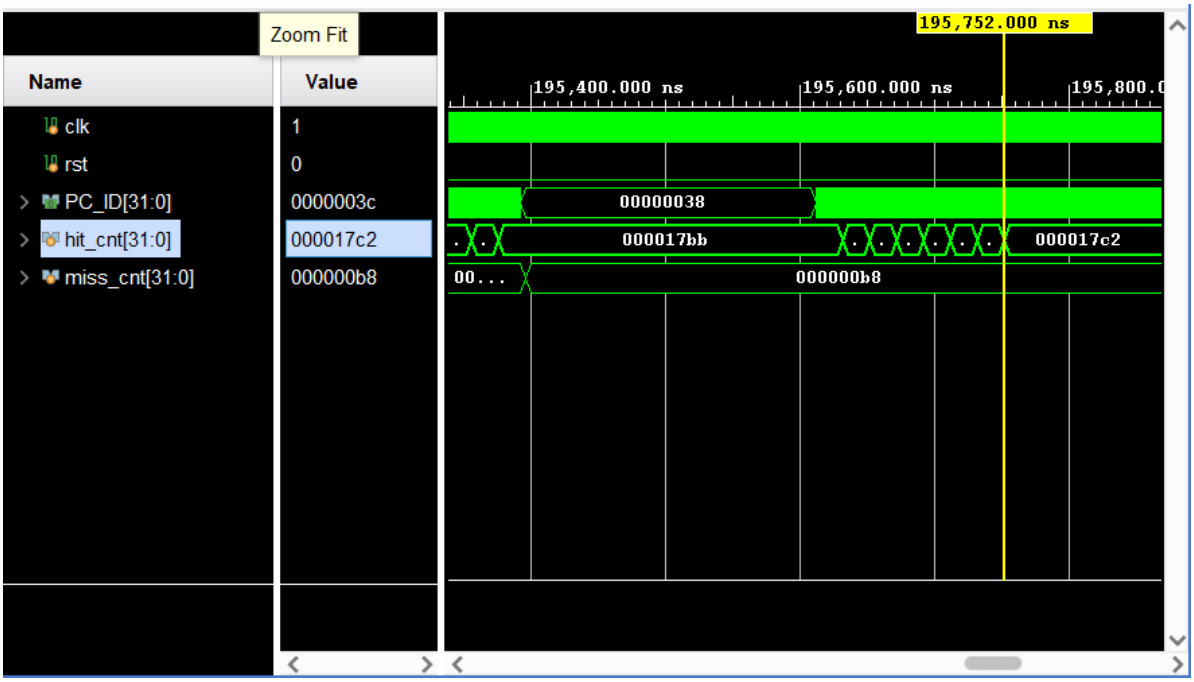
LRU

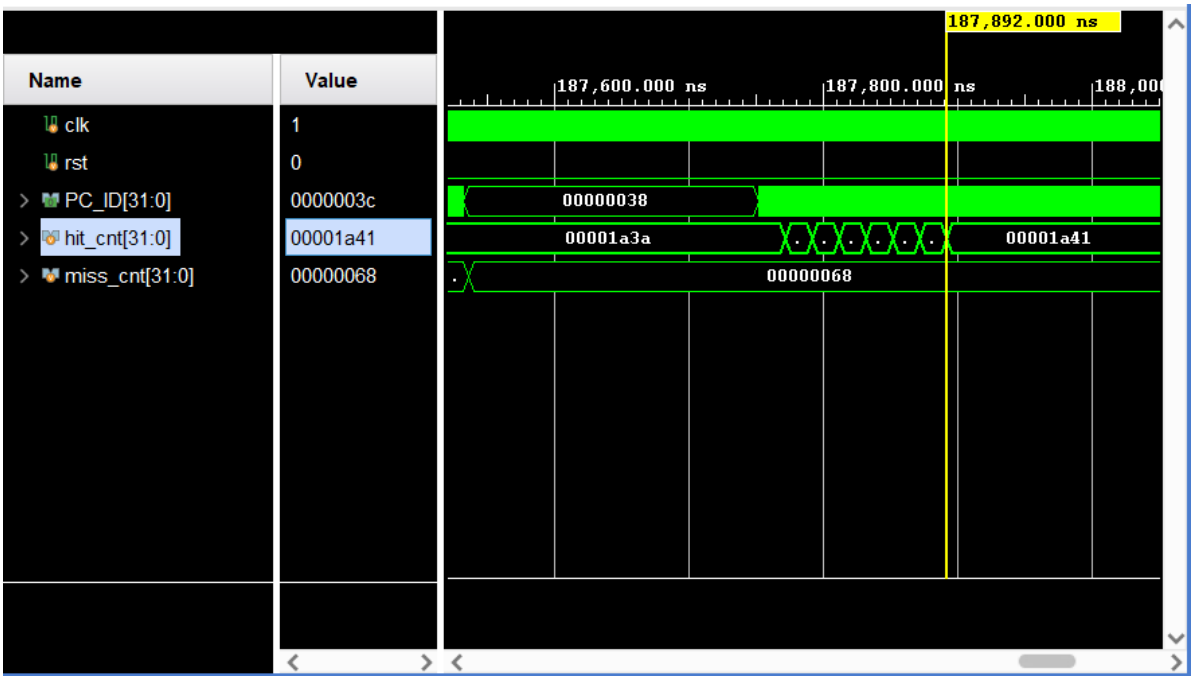
快速排序

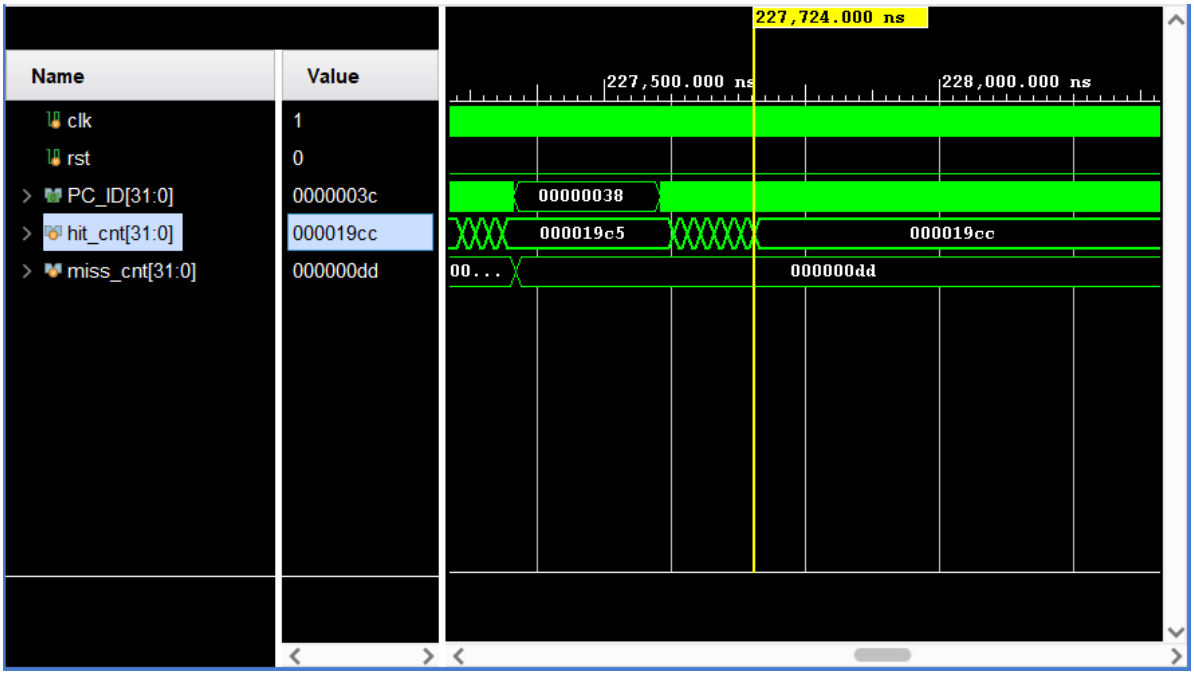
组数	组相联度	line大小	未命中	命中	命中率	周期数
8	1	8	339	5413	94.1%	59910
8	2	8	184	6082	97.1%	48938
8	4	8	104	6721	98.5%	46973
4	4	8	221	6604	96.8%	56931
16	4	8	42	6783	99.4%	40245
8	4	4	412	6413	94.0%	71561
8	4	16	21	6804	99.7%	39111

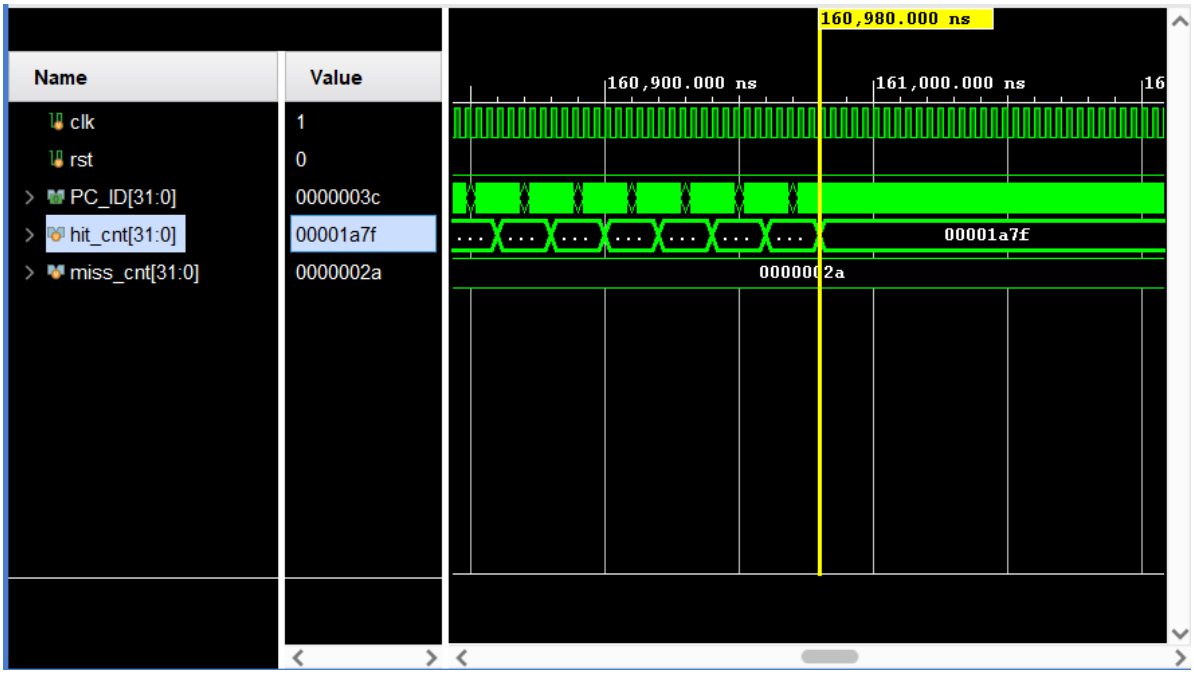
按照行顺序的截图如下：

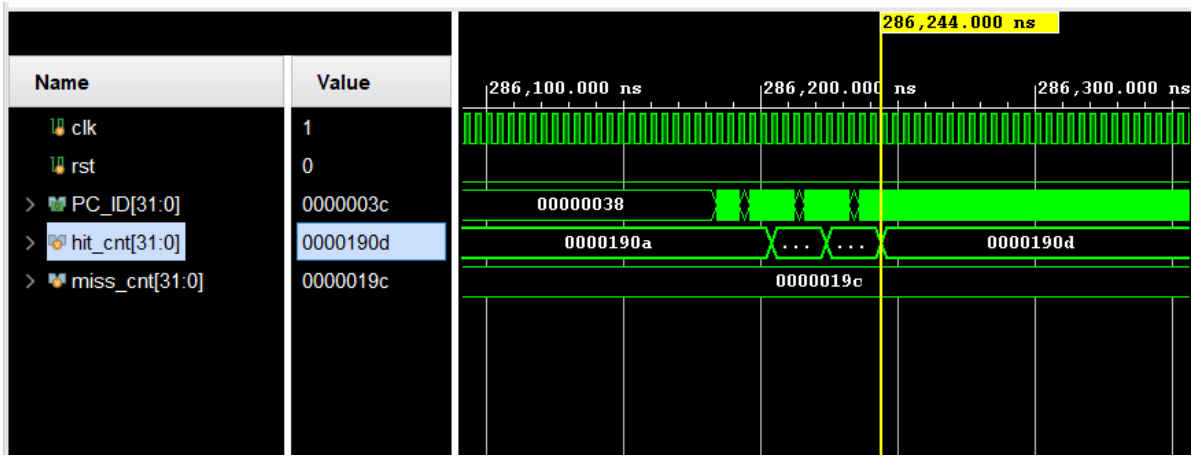

Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
00001525
miss_cnt[31:0]
00000153
239,640.000 ns
239,200.000 ns
239,400.000 ns
00000038
0000151e
00001525
00000152
00000153

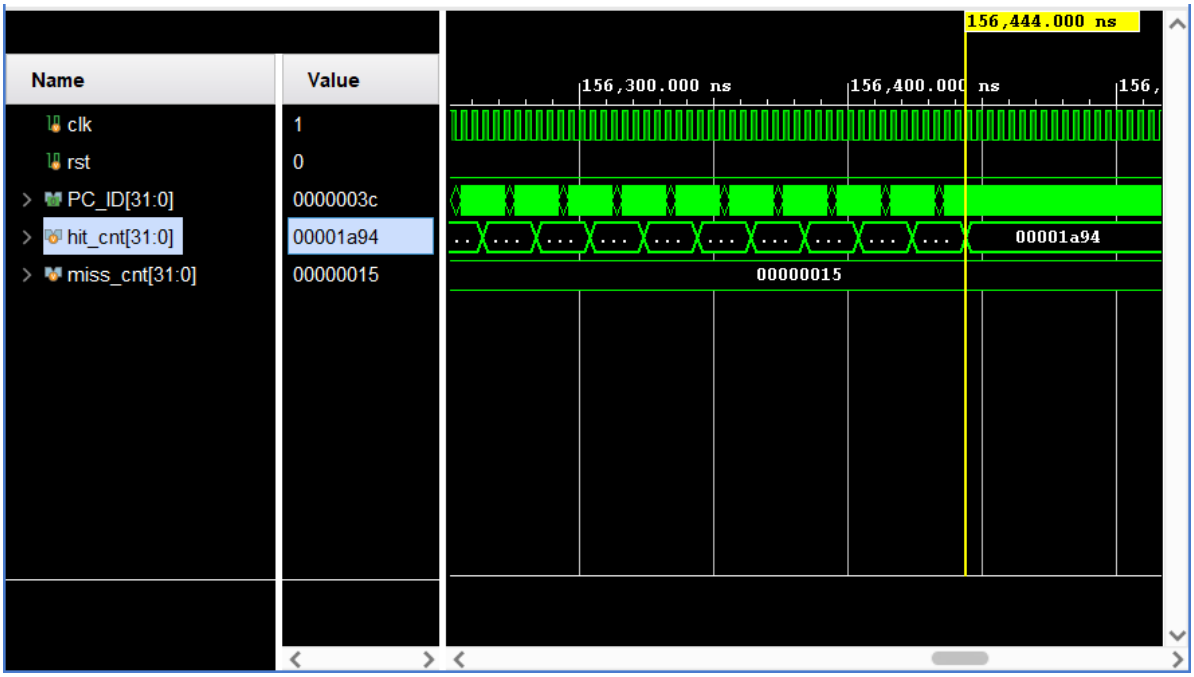

Zoom Fit
Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
000017c2
miss_cnt[31:0]
000000b8
195,752.000 ns
195,400.000 ns
195,600.000 ns
00000038
000017bb
000017c2
00...
000000b8


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
00001a41
miss_cnt[31:0]
00000068
187,892.000 ns
187,600.000 ns
187,800.000 ns
00000038
00001a3a
00001a41
00000068


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
000019cc
miss_cnt[31:0]
000000dd
227,724.000 ns
227,500.000 ns
228,000.000 ns
00000038
000019c5
000019cc
00...
000000dd


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
00001a7f
miss_cnt[31:0]
0000002a
160,980.000 ns
160,900.000 ns
161,000.000 ns
00001a7f
2a

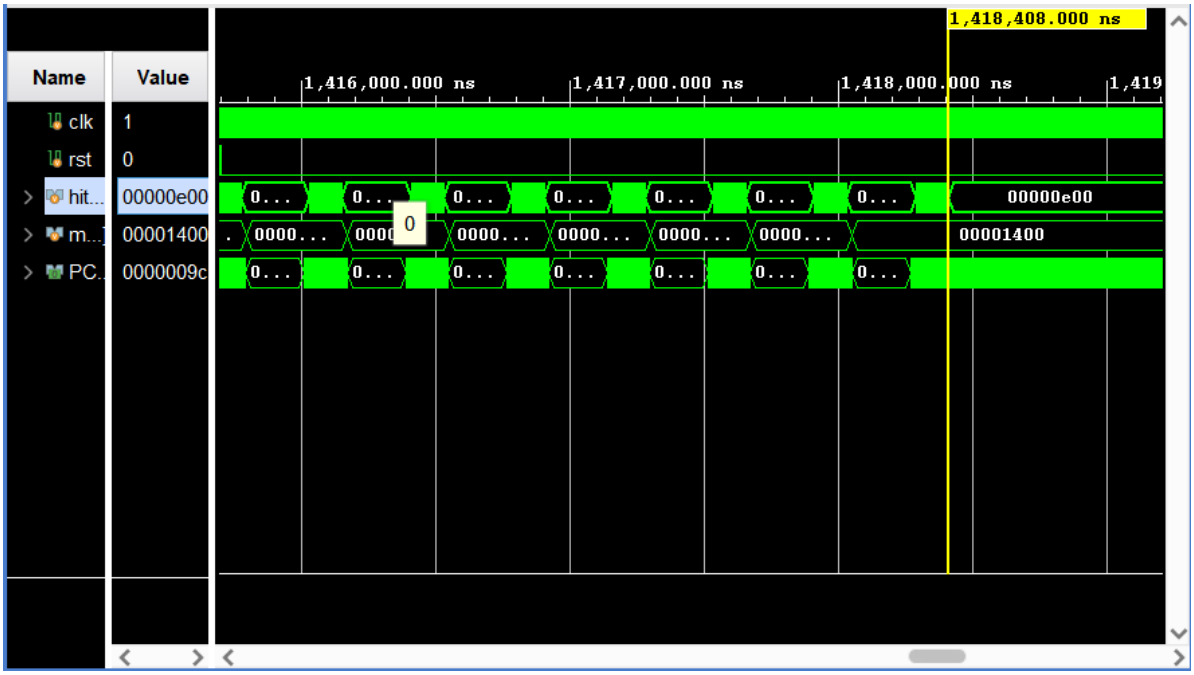

Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
0000190d
miss_cnt[31:0]
0000019c
286,244.000 ns
286,100.000 ns
286,200.000 ns
286,300.000 ns
00000038
0000190a
0000190d
0000019c

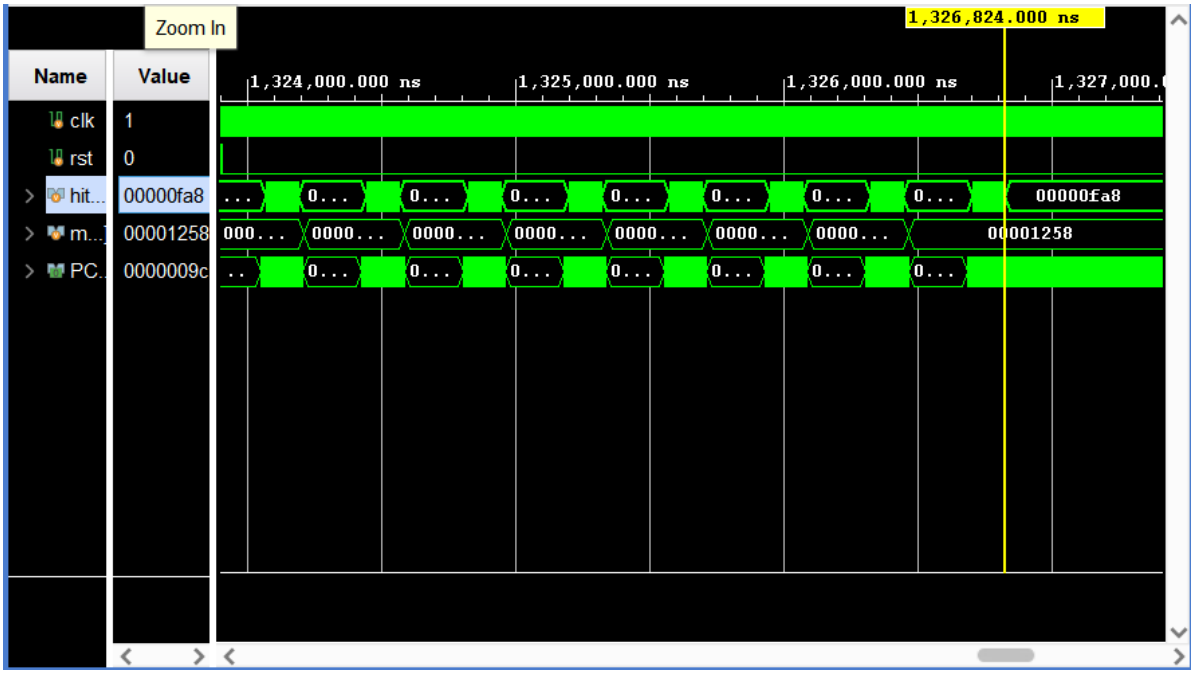


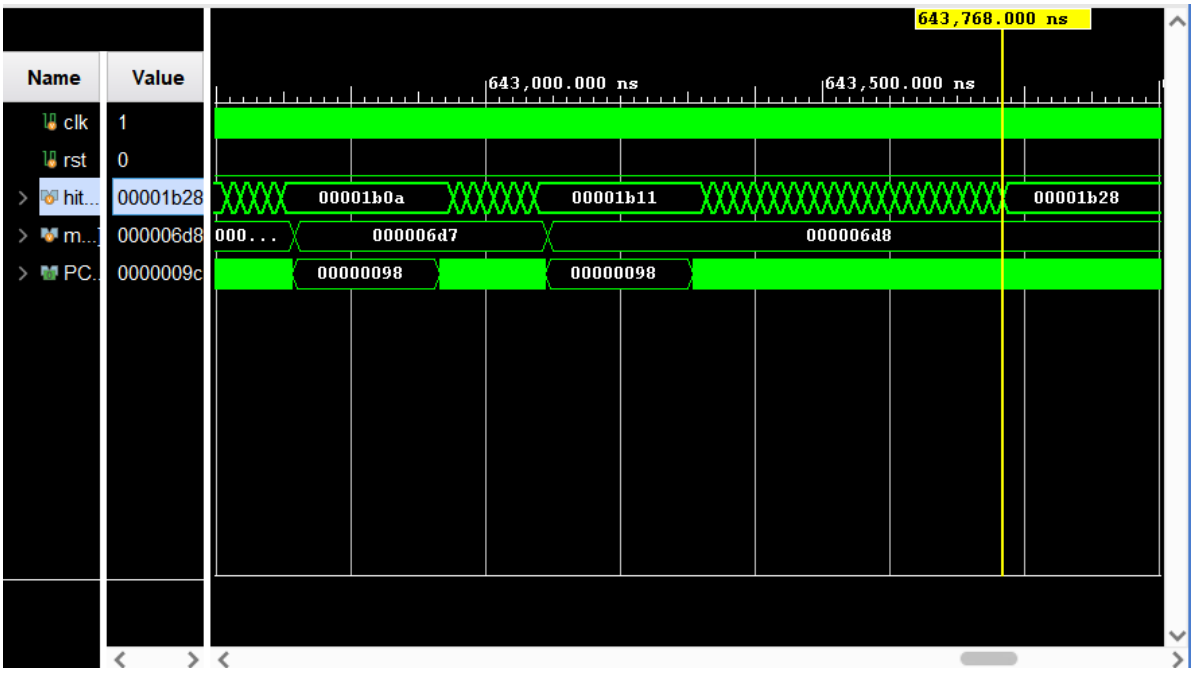
矩阵乘法

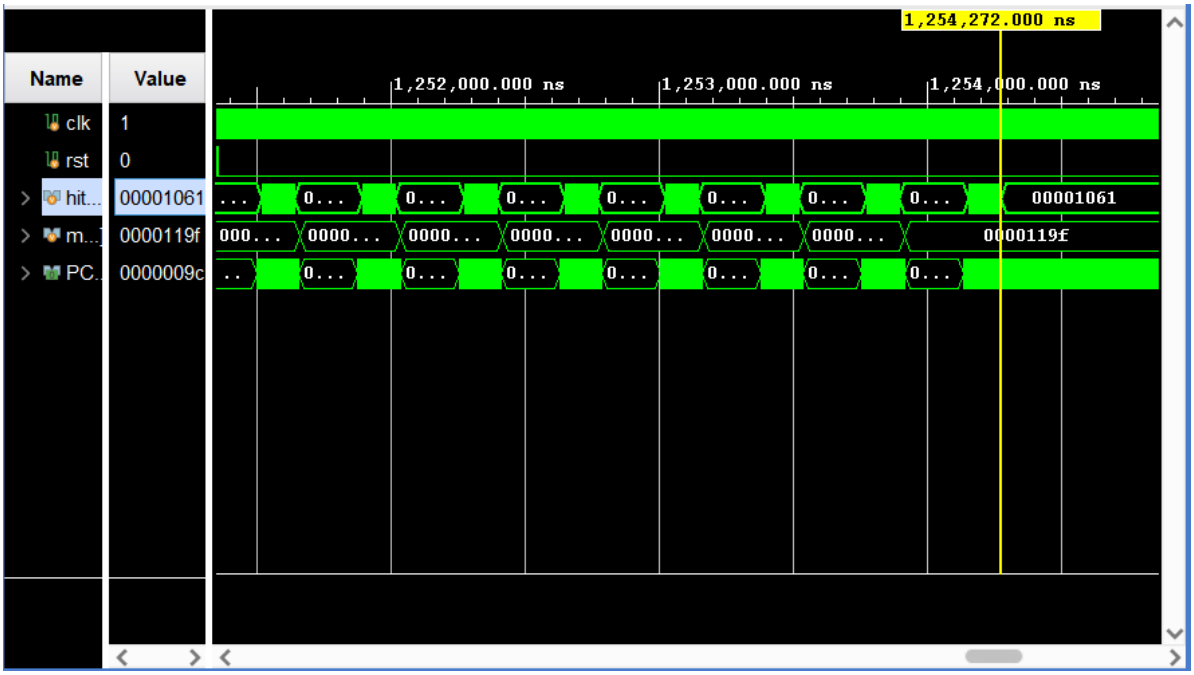
组数	组相联度	line大小	未命中	命中	命中率	周期数
8	1	8	5120	3584	41.2%	354602
8	2	8	4696	4008	46.0%	331706
8	4	8	1752	6952	79.9%	160942
4	4	8	4511	4193	48.2%	313568
16	4	8	153	8551	98.2%	73868
8	4	4	4668	4036	46.4%	326310
8	4	16	90	8614	99.0%	70102

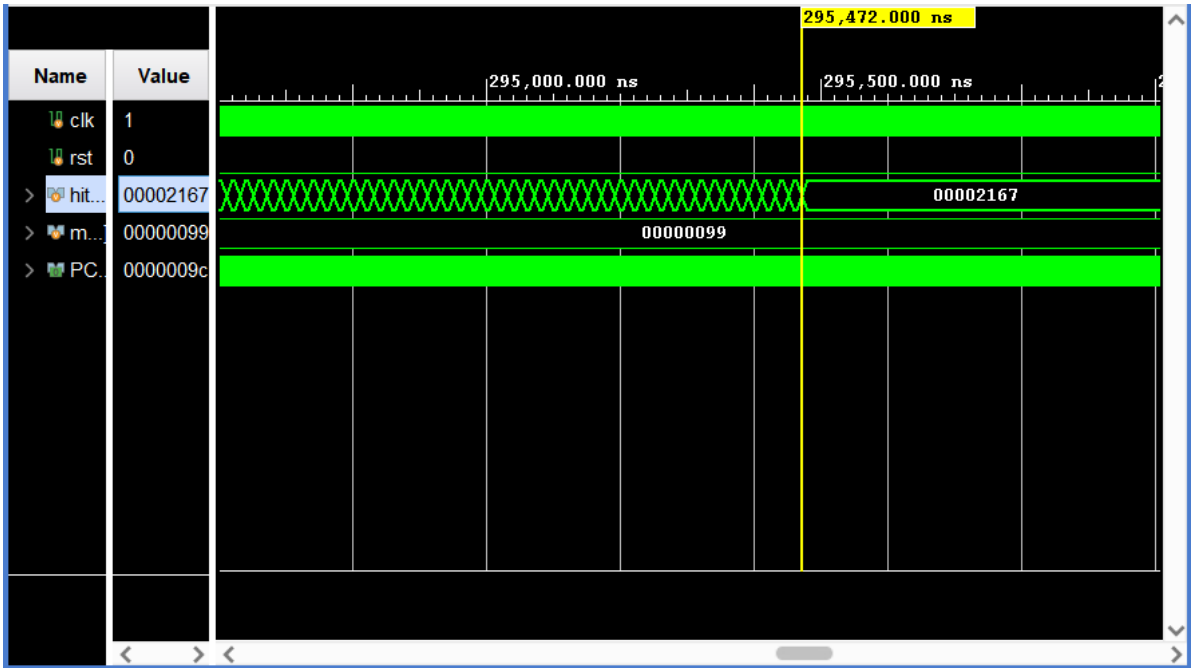
按照行的顺序截图如下：

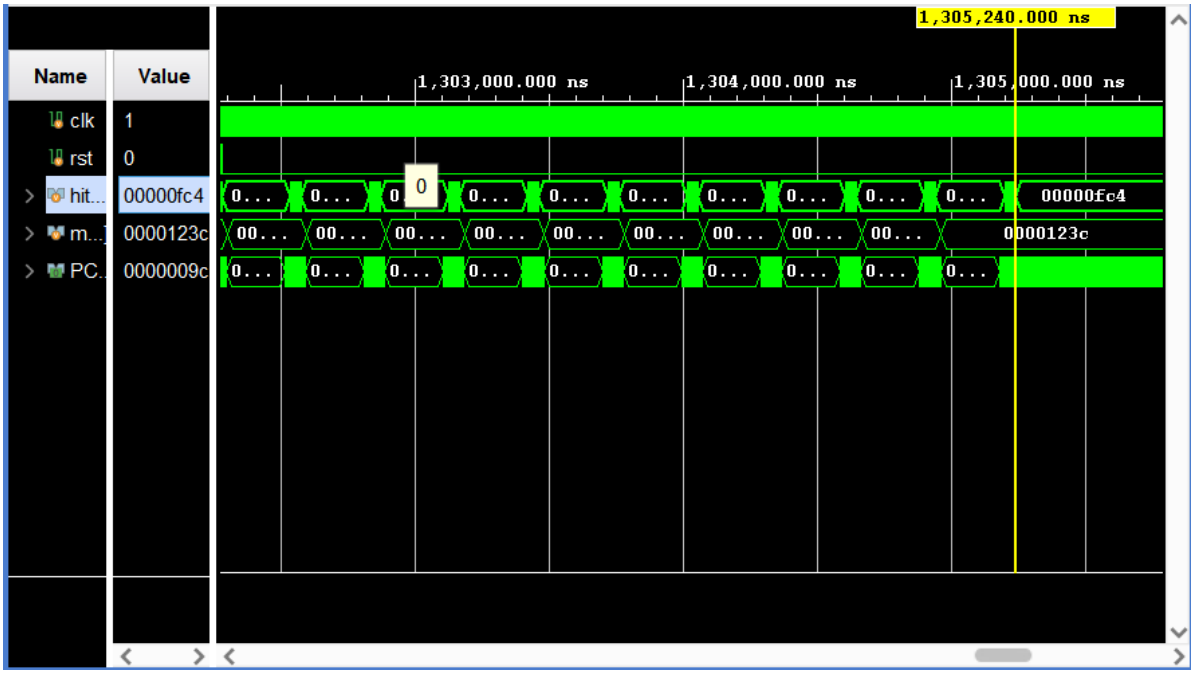

Name
Value
clk
1
rst
0
hit...
00000e00
m...
00001400
PC...
0000009c
1,418,408.000 ns
1,416,000.000 ns
1,417,000.000 ns
1,418,000.000 ns
1,419
00000e00
00001400

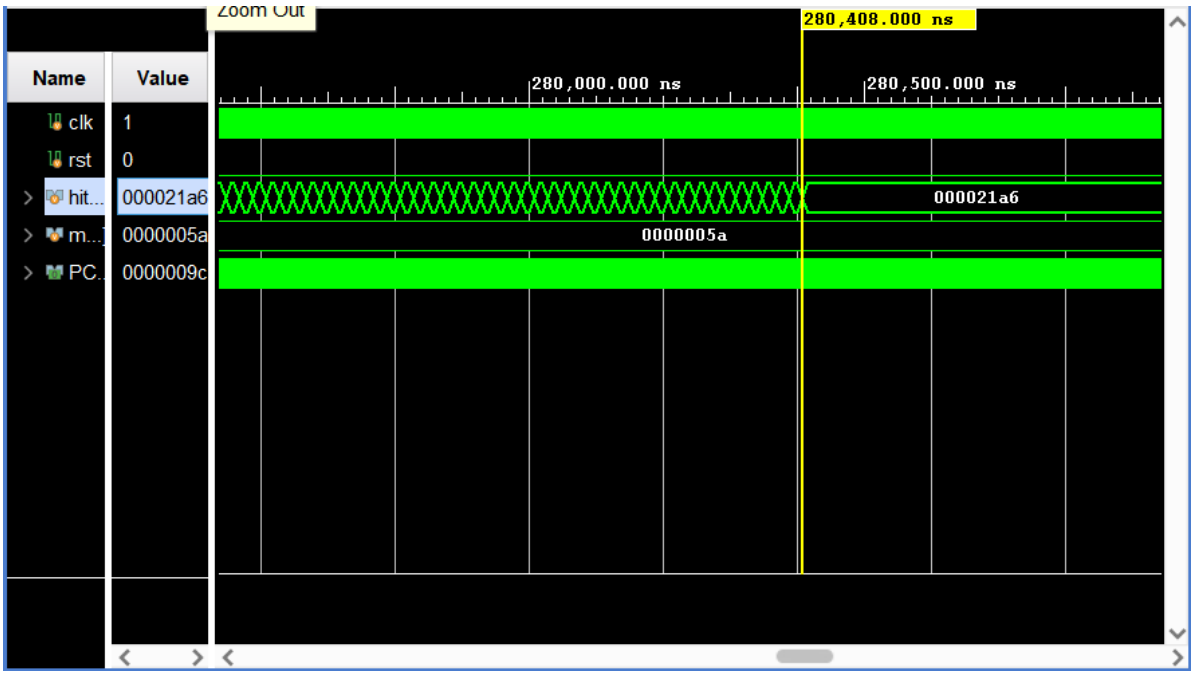

Zoom In
Name
Value
clk
1
rst
0
hit...
00000fa8
m...
00001258
PC...
0000009c
1,326,824.000 ns
1,324,000.000 ns
1,325,000.000 ns
1,326,000.000 ns
1,327,000.
00000fa8
00001258


Name
Value
clk
1
rst
0
hit...
00001b28
m...
000006d8
PC...
0000009c
643,768.000 ns
643,000.000 ns
643,500.000 ns
00001b0a
00001b11
00001b28
000006d7
000006d8
00000098
00000098


Name
Value
clk
1
rst
0
hit...
00001061
m...
0000119f
PC...
0000009c
1,254,272.000 ns
1,252,000.000 ns
1,253,000.000 ns
1,254,000.000 ns
00001061
0000119f


Name
Value
clk
1
rst
0
hit...
00002167
m...
00000099
PC...
0000009c
295,472.000 ns
295,000.000 ns
295,500.000 ns
00002167
00000099


Name
Value
clk
1
rst
0
hit...
00000fc4
m...
0000123c
PC...
0000009c
1,305,240.000 ns
1,303,000.000 ns
1,304,000.000 ns
1,305,000.000 ns
00000fc4
0000123c

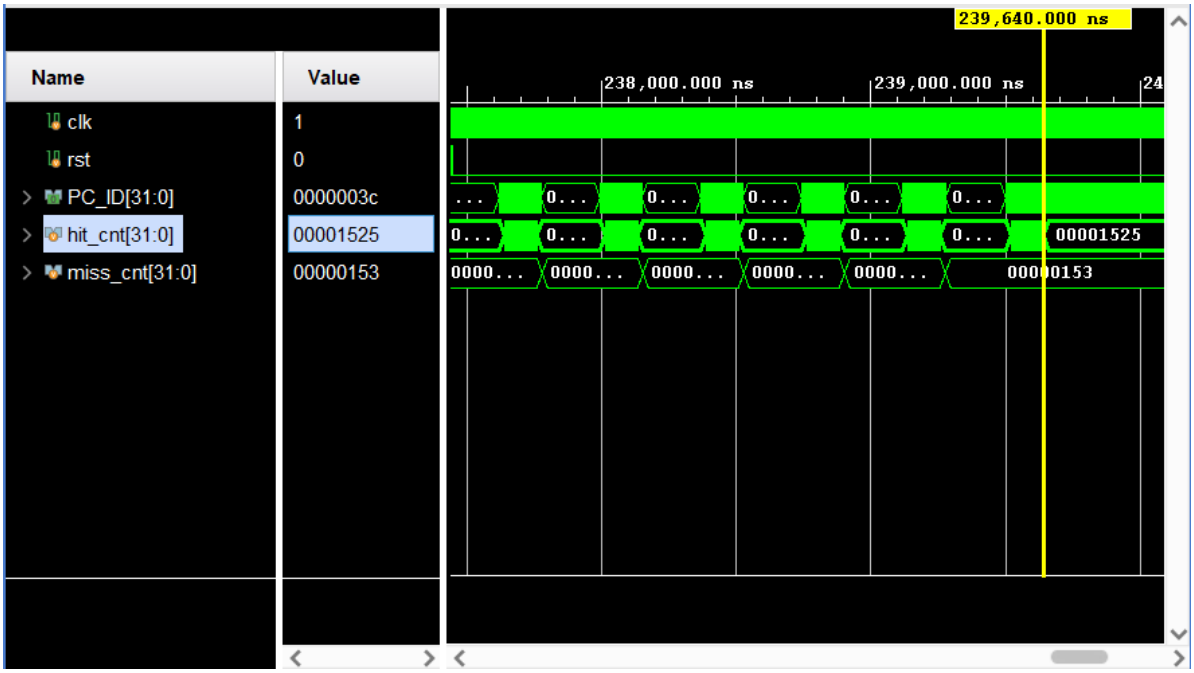


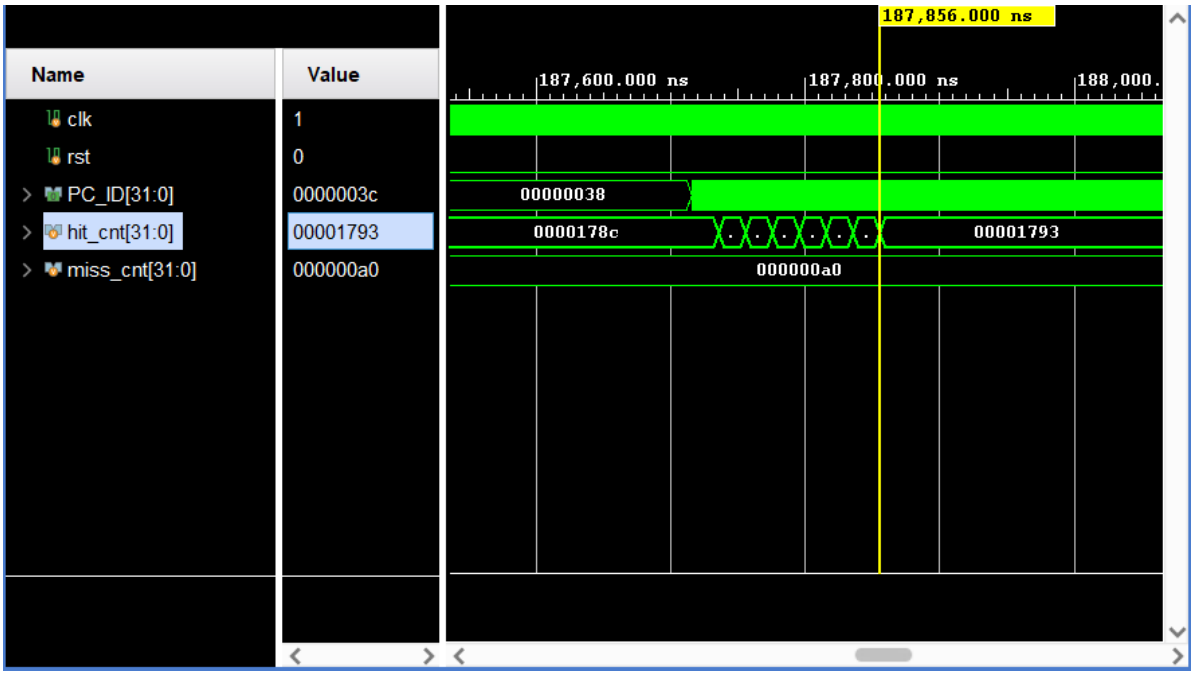
FIFO

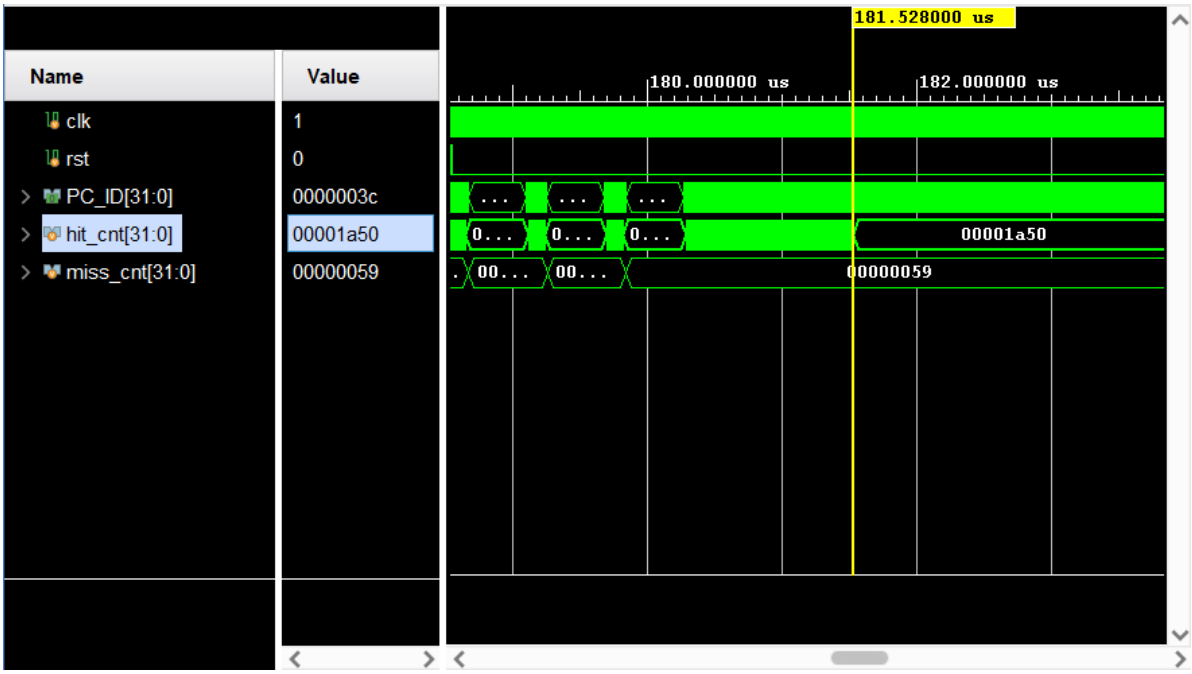
快速排序

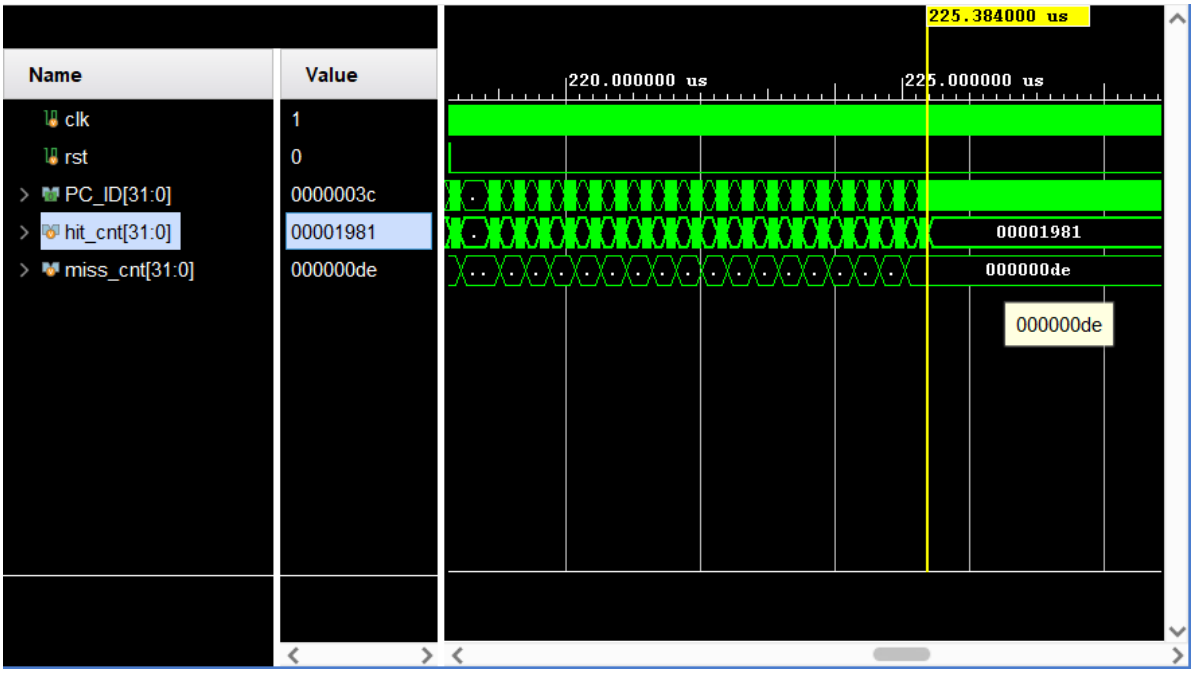
组数	组相联度	line大小	未命中	命中	命中率	周期数
8	1	8	339	5413	94.1%	59910
8	2	8	160	6035	97.4%	46964
8	4	8	89	6736	98.6%	42363
4	4	8	222	6529	96.7%	56346
16	4	8	42	6783	99.3%	40245
8	4	4	409	6356	94.0%	71080
8	4	16	21	6804	99.7%	39111

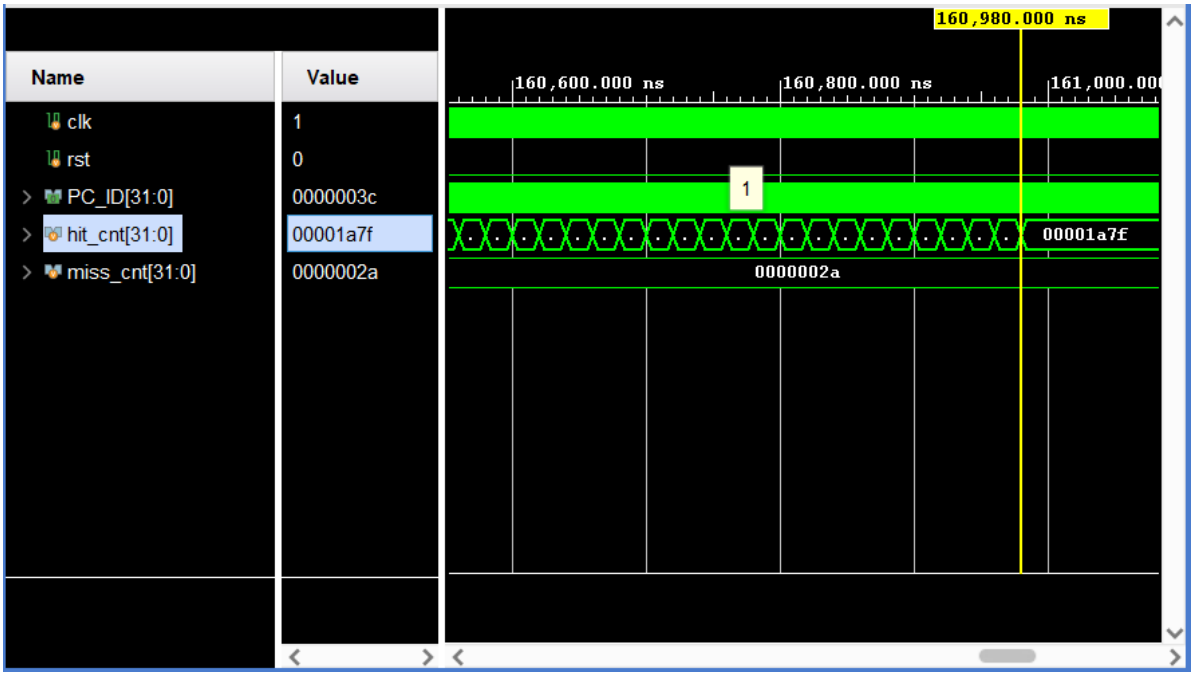
按顺序的仿真截图如下：

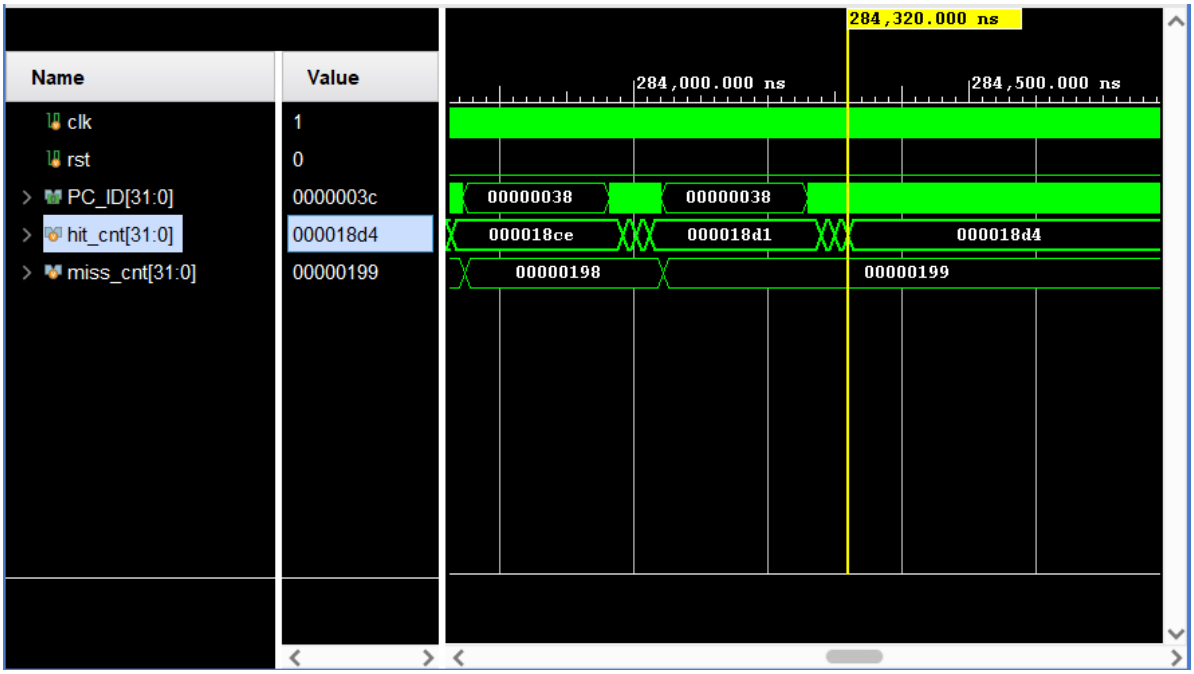

Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
00001525
miss_cnt[31:0]
00000153
239,640.000 ns
238,000.000 ns
239,000.000 ns
00001525
00000153

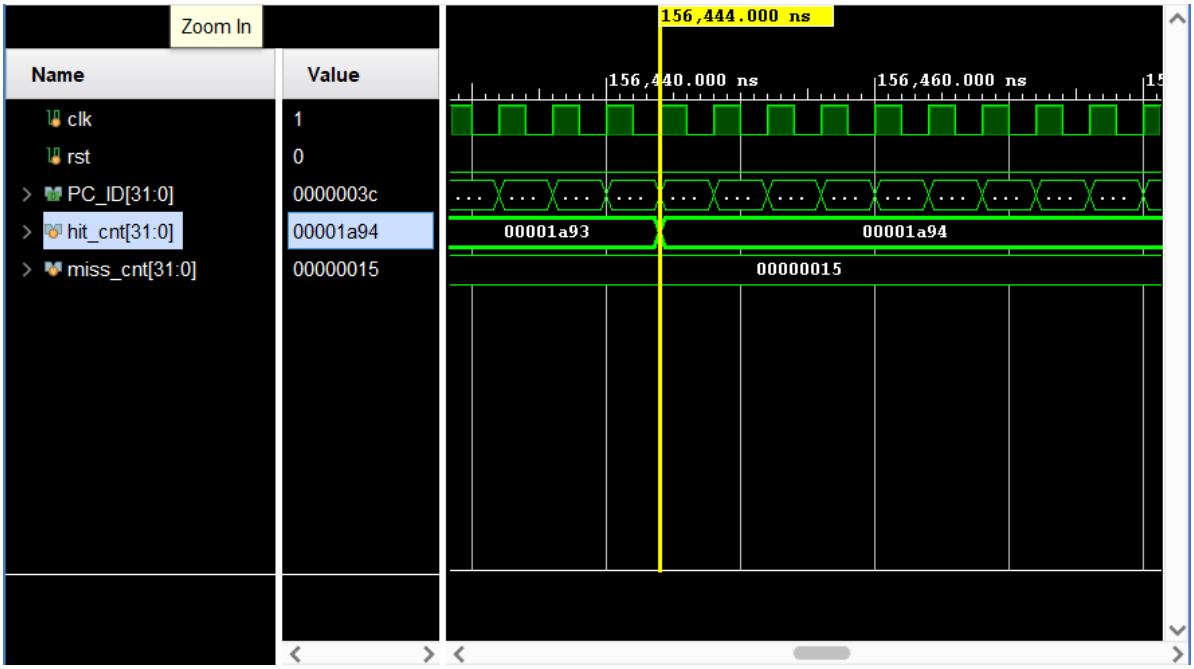

Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
00001793
miss_cnt[31:0]
000000a0
187,856.000 ns
187,600.000 ns
187,800.000 ns
188,000.
00000038
0000178c
00001793
000000a0


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
00001a50
miss_cnt[31:0]
00000059
181.528000 us
180.000000 us
182.000000 us
00001a50
00000059


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
00001981
miss_cnt[31:0]
000000de
225.384000 us
220.000000 us
225.000000 us
00001981
000000de
000000de


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
00001a7f
miss_cnt[31:0]
0000002a
160,980.000 ns
160,600.000 ns
160,800.000 ns
1
00001a7f
0000002a

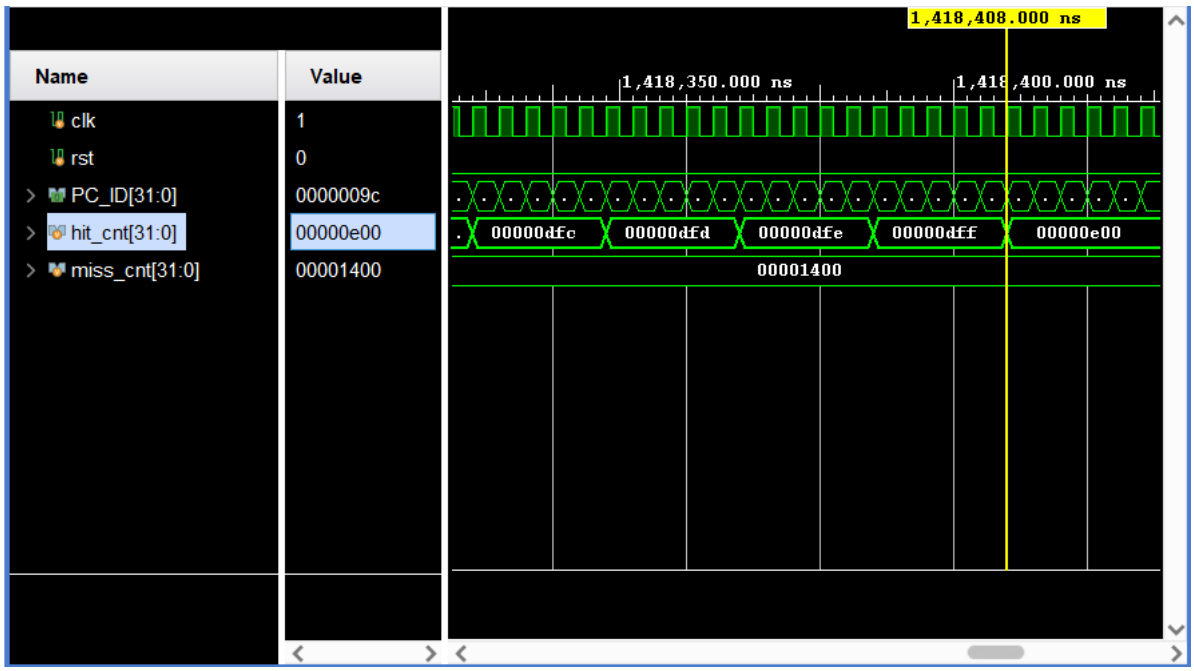

Name
Value
clk
1
rst
0
PC_ID[31:0]
0000003c
hit_cnt[31:0]
000018d4
miss_cnt[31:0]
00000199
284,320.000 ns
284,000.000 ns
284,500.000 ns
00000038
00000038
000018ce
000018d1
000018d4
00000198
00000199

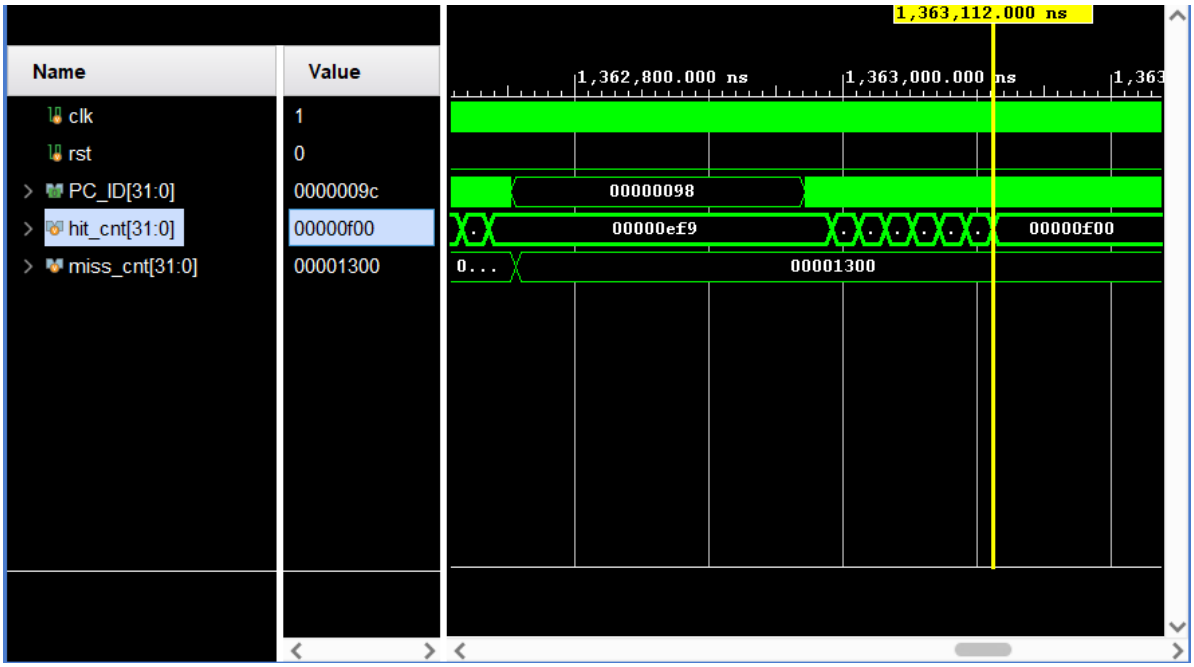


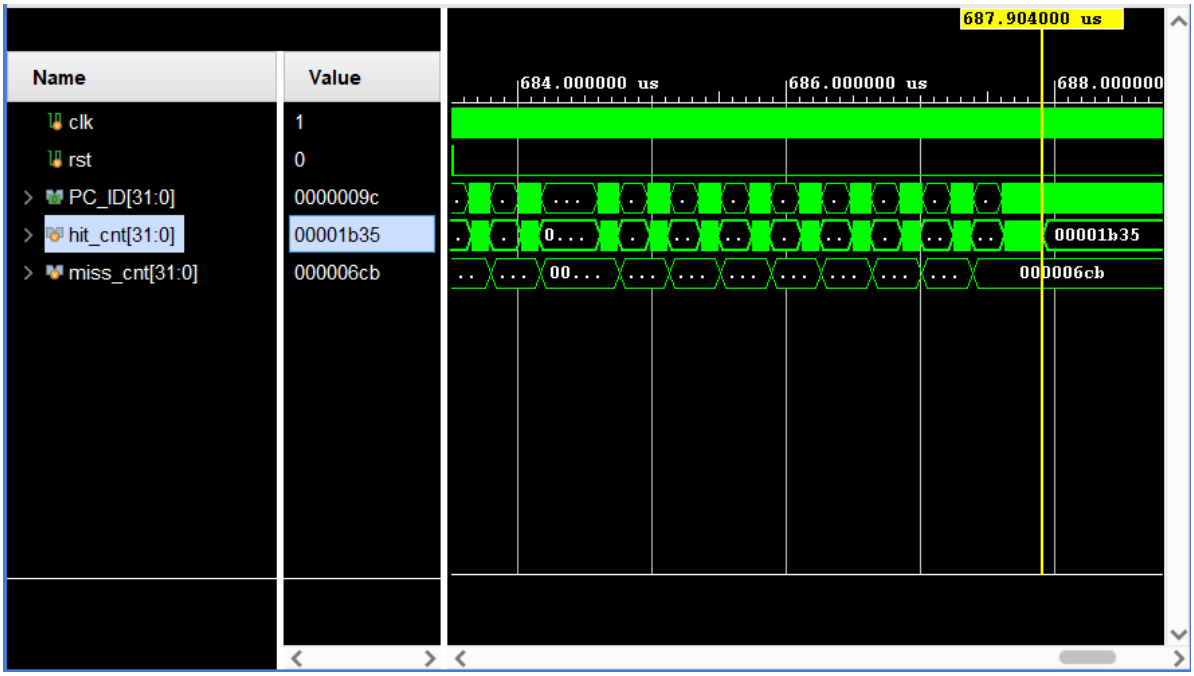
矩阵乘法

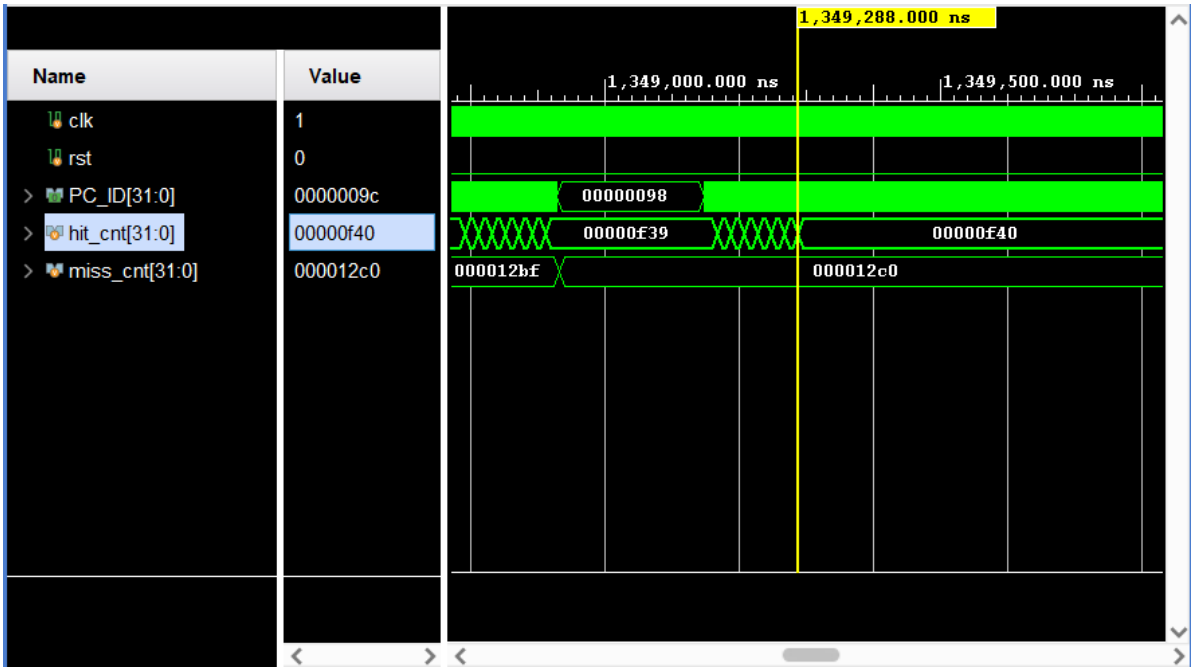
组数	组相联度	line大小	未命中	命中	命中率	周期数
8	1	8	5120	3584	41.2%	354602
8	2	8	4864	3840	44.1%	340778
8	4	8	1739	6965	80.0%	171976
4	4	8	4800	3,904	44.9%	337322
16	4	8	144	8560	98.3%	73430
8	4	4	4760	3944	45.3%	335170
8	4	16	72	8632	99.2%	69128

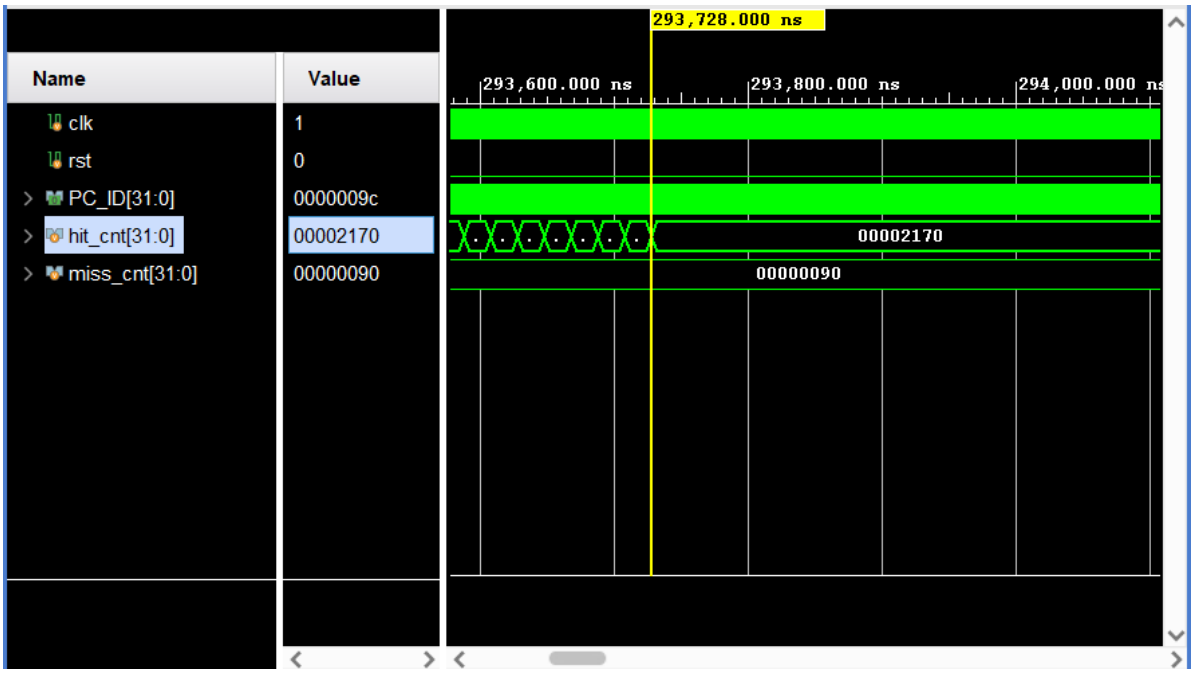
按照行的顺序截图如下：

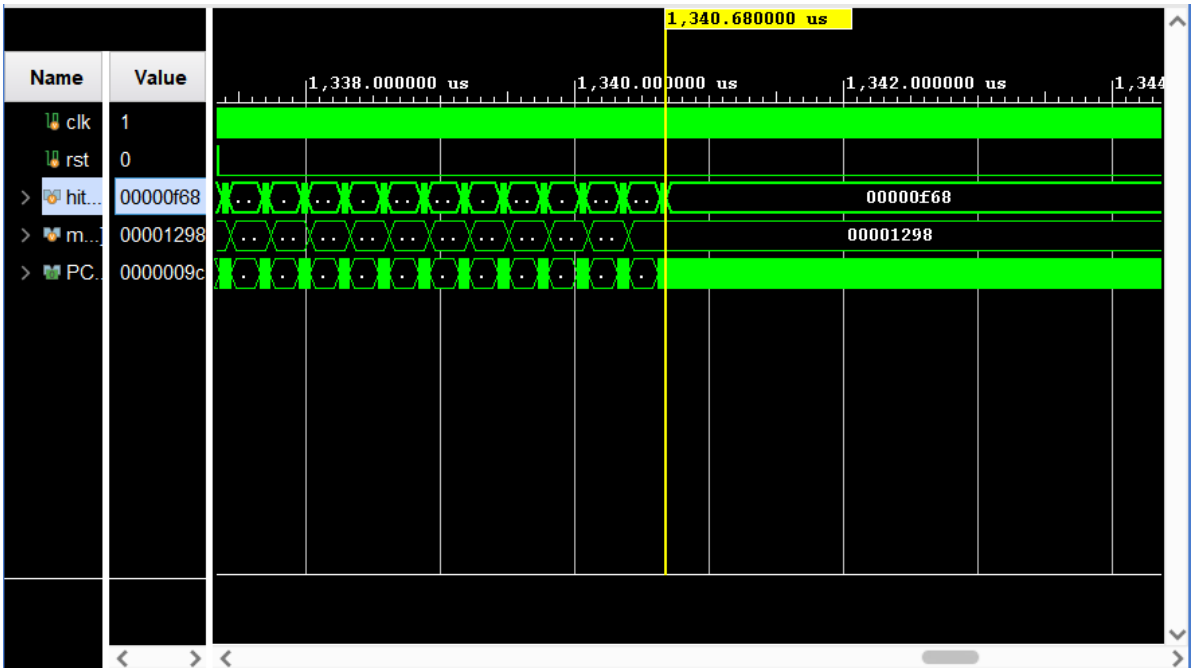

Name
Value
clk
1
rst
0
PC_ID[31:0]
0000009c
hit_cnt[31:0]
00000e00
miss_cnt[31:0]
00001400
1,418,408.000 ns
1,418,350.000 ns
1,418,400.000 ns
00000dfc
00000dfd
00000dfe
00000dff
00000e00
00001400

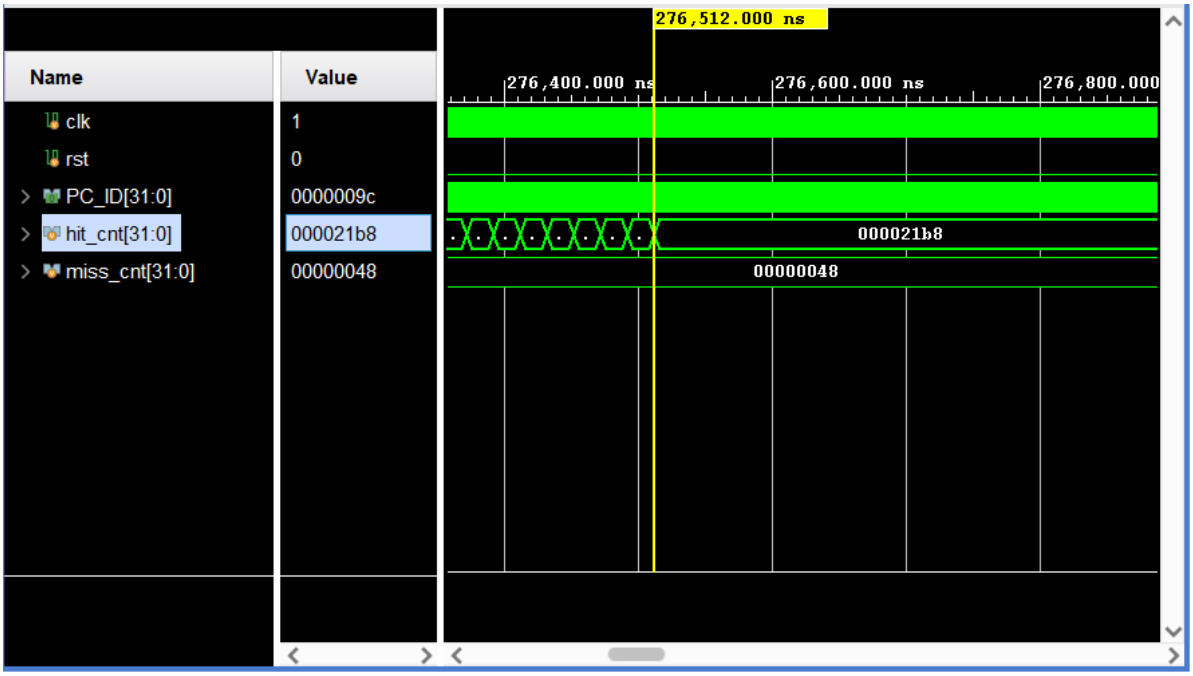

Name
Value
clk
1
rst
0
PC_ID[31:0]
0000009c
hit_cnt[31:0]
00000f00
miss_cnt[31:0]
00001300
1,363,112.000 ns
1,362,800.000 ns
1,363,000.000 ns
00000098
00000ef9
00000f00
0...
00001300


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000009c
hit_cnt[31:0]
00001b35
miss_cnt[31:0]
000006cb
687.904000 us
684.000000 us
686.000000 us
688.000000
0...
00001b35
00...
000006cb


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000009c
hit_cnt[31:0]
00000f40
miss_cnt[31:0]
000012c0
1,349,288.000 ns
1,349,000.000 ns
1,349,500.000 ns
00000098
00000f39
00000f40
000012bf
000012c0


Name
Value
clk
1
rst
0
PC_ID[31:0]
0000009c
hit_cnt[31:0]
00002170
miss_cnt[31:0]
00000090
293,728.000 ns
293,600.000 ns
293,800.000 ns
294,000.000 ns
00002170
00000090


Name
Value
clk
1
rst
0
hit...
00000f68
m...
00001298
PC...
0000009c
1,340.680000 us
1,338.000000 us
1,340.000000 us
1,342.000000 us
00000f68
00001298



结果总结

1. 对于矩阵乘法：

   组数和line大小相同时， LRU 优于 FIFO；

   组相连度相同时，组数/line大小较小时LRU更优，较大时FIFO更优。

2. 对于快速排序：

   组数和line大小相同时， FIFO 优于 LRU；

   组相连度相同时， LRU 与 FIFO 的性能几乎相同。

3. 增加组相联度能显著提升 Cache 的命中率，提高运行性能。增加组数和line大小也可以有同样的效果，但是需要较大的Cache空间代价。

4. 同时考虑性能和资源消耗，对于矩阵乘法，应选择组数为8，line大小为8，块数为，组相联度为 4 的 LRU 策略 Cache；对于快速排序，应选择组数为8，line大小为8，组相联度为 4 的 FIFO 策略 Cache 。

实验难点

1. 模块实例化名称出错

   在`WbData`中实例化`cache`时，写成了`Cache`，导致报错很奇怪（并不是“没有找到相应模块”之类的报错），找了很久才发现这里出了问题。

2. FIFO策略的实现

   我的想法是维护一个队列，每次需要换出的时候就将队头的块换出，队尾加入新的块。出错的点在于我只维护了一个队列，实际上应该给每个组维护一个队列。并且队列中每个块应该用WAY的编号，而不是SET的编号来标识，这样方便查询。

3. 对Cache性能的测试

   在测试资源消耗的时候，记得将Simulation文件中的cache_tb.sv设为顶层模块，但是忘了把Source文件中的cache.sv设为顶层模块。导致测出来的LUT和FF利用率超过了百分之百，找了很久才发现这里有问题。

实验总结及收获

本次实验实现了Cache，通过测试大量样例测试，比较了不同替换策略以及Cache参数大小不同的性能差异，对缓存机制有了更加深入的了解。

另外，本次实验文档非常详细！照着上面的提示来做实验并不难完成。