## Cache 实验常见疑问：

**问：无论进行快速排序，还是矩阵乘法，最终的主存（mem.sv 模块里的ram\_cell变量）里的数据都与正确结果整体上是相同的，但会略有差异，是为什么？**

**答：因为是写回策略的cache，所以最终会有一些数据还在cache中未写入主存。属于正常现象。但如果你的cache写错了，那么快速排序和矩阵乘法的结果就会很离谱。在检查实验时，助教主要通过第一阶段cache\_tb.sv （即脱离CPU的cache检验）去判断你的cache的正确性。**

**问：Cache实验都检查什么内容**

**答：阶段1的cache\_tb必须正确通过，然后需要向助教展示你所写的cache代码，讲解你的思路，包括但不限于如何在原有基础上加并行TAG比较、如何实现LRU和FIFO换出策略。**

**阶段2的分值不高，只需要展示你所写的LRU和FIFO策略的cache能够跑通快速排序和矩阵乘法就行，对于快速排序来说，跑通的结果应该是主存中的数据变成有序的。对于矩阵乘法来说，跑通的结果应该是主存中的数据与注释中的数据相同。当然，因为有些数据在cache中还未写入主存，所以允许有一些不同。**

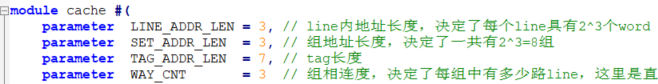
**问：如何写cache实验的报告**

**Cache实验报告在cache实验中占很大的分值比重。因为Cache实验是相对于CPU流水线设计而言是开放的，重点在权衡性能和面积的体会，而Cache本身的实现只是前提。阶段二所提供的快速排序和矩阵乘法的benchmark就是用来在实验报告中进行实验分析的，也鼓励自己编写更多的汇编benchmark进行测试。在做实验时，修改Cache的参数，体会cache size、组相连度、替换策略针对不同程序的优化效果，以及策略改变带来的电路面积的变化。针对不同程序，权衡性能和电路面积给出一个较优的cache参数和策略。注意，需要权衡性能和电路面积。**

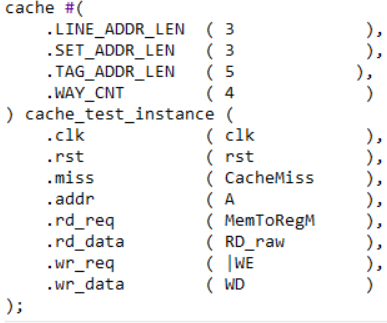
**其中“性能”参数使用运行仿真时的时钟周期数量进行评估。“资源占用”参数使用vivado给出的综合报告进行评估。在写报告时，不能仅仅进行理论分析，实验报告中需要给出实验结果（例如仿真波形的截图、vivado综合报告等）。**

**问：如何修改cache的参数**

**答：cache.sv的开头中有如下的parameter定义：**

****

**这些参数应该在编写实验报告时进行修改，用于权衡性能和电路面积。修改方法是在调用它的模块（即WBSegReg.v）中，修改这几个参数，如下：**

****

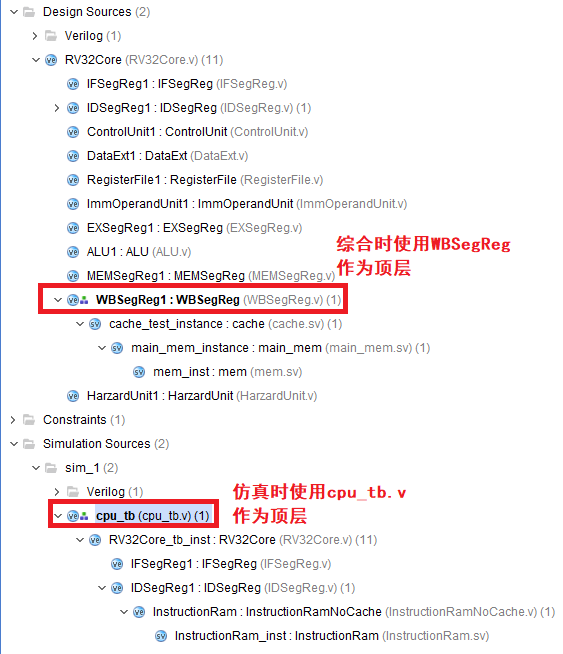
**Verilog/SystemVerilog 中的parameter类似C++中的默认参数，在被调模块中只是指定默认值。在主调模块中，如果指定，则使用该指定值；如果不指定，则使用默认值。**

**问：vivado的综合和仿真是否互相依赖？**

**答：没有，综合前不需要仿真，仿真前也不需要综合。**

**问：为了进行电路面积评估，如何进行综合？**

**答：因为我们只关心cache的电路面积，而CPU我们是不会修改的，所以建议在综合前，将WBSegReg.v 设置为顶层，可以提高综合速度。在vivado中应该如下设置仿真和综合的顶层：**

****

**因为仿真与综合互相独立，所以可以使用不同的顶层。设置cpu\_tb.v作为仿真的顶层，用于进行cache性能的评估。设置WBSegReg.v作为综合的顶层，综合后查看综合报告，用于进行电路面积的评估。**

**问：综合特别慢怎么办？**

**在5月16日之前提供的mem.sv里，ram\_cell的定义为：**

|  |
| --- |
| **reg [31:0] [0:MEM\_SIZE-1] ram\_cell;** |

**要改成**

|  |
| --- |
| **reg [31:0] ram\_cell [MEM\_SIZE];** |

**后者 vivado会综合成BRAM，所以很快，前者vivado会综合到LUT里，所以慢。5月16日后的github库修改了这个问题。修改后综合一般只需要不到1分钟。**