3）在弹出的 Generate Output Product 窗口中，根据需要选择 Global 或 Out of context per IP 模式，点击 Generate 完成 inst_ram 的重新定制。

3. 重新生成 golden_trace.txt

当在 cpu132_gettrace 里重新定制 inst_ram IP 后，还要重新生成 golden_trace.txt，其步骤如下：

1）打开 cpu132_gettrace 工程，确保 inst_ram IP 已重新定制。

2）运行仿真，仿真结束后，cpu132_gettrace 目录下的 golden_trace.txt 会被更新。

4. 替换 mif 文件快速仿真

本节内容仅供参考，如果未掌握本节内容也不会影响完成实践任务。

本节将提供一种方法：在不重新定制 inst ram 的情况下，直接使用新的 func 进行仿真。

前面介绍的重新定制 inst_ram 比较耗时，如果只进行仿真的话，可以选择修改、替换 mif 文件的方法来达到更改 inst_ram 初始值的目的。

以 cpu132_gettrace 工程为例，在启动仿真界面后，会发现工程文件中出现目录“ cpu132_gettrace/run_vivado/cpu132_gettrace/cpu132_gettrace.sim/sim_1/behave/xsim/”，将该目录中的 inst_ram.mif 换成软件环境 obj 目录下的 inst_ram.mif（CPU_CDE/soft/func_lab4/obj/inst_ram.mif），再点击“Restart”（注意：不是“Relaunch Simulation”）回到零时刻，然后点击“run all”开始仿真，此时 inst_ram 的初始值会变成新的 inst_ram.mif 中所写的数据。

注意：每当新打开仿真或者点击“Relaunch Simulation”时，inst_ram.mif 的文件会根据先前定制好的 inst_ram 来重新生成。

4.4　CPU 设计的功能仿真调试技术

上一章我们已经介绍了数据逻辑电路设计在功能仿真过程中常用的调试方法和技术。在这一节中，我们将结合 CPU 这种具体的数字逻辑电路以及我们所提供的开发环境，详细介绍一些有关的功能仿真调试技术。

这里我们仍然以观察仿真波形作为 CPU 功能仿真验证调试的主要手段。在上一章中，我们介绍过观察数字逻辑电路功能仿真波形的基本思路，其核心就是“找到一个能明确的错误点，然后沿着设计的逻辑链条逆向逐级查看信号，直至找到错误的源头。”CPU 也是数字逻辑电路，所以它的功能仿真验证调试同样可以遵循这个思路。因此，我们主要结合 CPU 功能仿真验证的特殊性，总结具体的调试方法。

4.4.1　为什么要用基于 Trace 比对的调试辅助手段

通过前面 CPU 设计环境的介绍，读者已经知道在对 CPU 进行功能验证时，会采用一段测试指令序列（或称为测试程序）作为 CPU 功能验证的激励，验证通过与否是通过观察测试