## 本历史

| 文档更新记录 | 文档名: | Lab04_利用阻塞解决相关引发的冲突 |
|---|---|---|
| | 版本号 | V0.1 |
| | 创建人: | 计算机体系结构研讨课教学组 |
| | 创建日期 : | 2019-09-18 |

**更新历史**

| 序号 | 更新日期 | 更新人 | 版本号 | 更新内容 |
|---|---|---|---|---|
| 1 | 2019/09/18 | 贾凡 | - | 草稿。 |
| 1 | 2019/09/18 | 邢金璋 | V0.1 | 初版。 |
| | | | | |
| | | | | |

文档信息反馈: xingjinzhang@loongson.cn

# 1 实验四 利用阻塞解决相关引发的冲突

在学习并尝试本章节前，你需要具有以下环境和能力：

(1) 装有 Vivado 的电脑一台。

(2) 熟悉 Vivado，并能初步使用。

如果对 Vivado 不熟悉，请参考课程讲义中的第一讲内容。

(3) 初步掌握 Verilog 的简单语法。

(4) 熟悉龙芯体系结构实验箱（Artix-7）。

(5) 了解 CPU 流水线结构。

通过本章节的学习，你将获得：

(1) 理解流水线冲突的产生原因。

(2) 掌握使用阻塞的方式解决相关引发的流水线冲突的方法。

## 1.1 实验目的

1. 加深对流水线结构的理解。
2. 学会使用阻塞的方式解决相关引发的流水线冲突的方法。

## 1.2 实验设备

1. 装有 Xilinx Vivado 的计算机一台。
2. 龙芯体系结构教学实验箱（Artix-7）一套。

## 1.3 实验任务

本次实验只有一个子任务：

1. 在实验三的 CPU 代码基础上，加入适当的逻辑处理寄存器写后读数据相关引发的流水线冲突（本次实验只要求使用阻塞的方式，但不限制使用更高级的前递方式），运行 func_lab4，要求成功通过仿真和上板验证。
2. 本次实验要求每个人独立提交实验报告和调试好的 RTL 代码，以报告评分和现场检查评分作为最后的实验得分：

   (1) 报告评分：描述自己的设计方案，记录调试过程。实验报告模板请使用 lab3 的模板。

   (2) 现场检查评分：检查包含仿真检查和上板检查。

## 1.4 实验检查

检查前需提交实验报告（每人一份）和调试好的 RTL 代码。本次实验在 2019 年 9 月 24 日进行检查。

现场检查，分仿真检查和上板检查：

1) 仿真检查：对照波形进行描述寄存器写后读数据相关引发的流水线冲突。

2) 上板检查：查看上板行为。

现场检查要求能正确应对检查者的提问，并根据要求进行正确的操作演示

## 1.5 实验提交

提交的作品包括纸质档和电子档。

**(1) 纸质档提交**

提交方式：课上现场提交，每人都必须要有。

截止时间：2019 年 9 月 24 日 18:10。

提交内容：纸质档 lab4 实验报告。

**(2) 电子档提交**

提交方式：打包上传到 Sep 课程网站 lab4 作业下，每人都必须要有。

截止时间：2019 年 9 月 24 日 18:10。

提交内容：电子档为一压缩包，文件名是“lab4_箱子号_学号.zip” ，目录层次如下（请将其中的“箱子号”替换为本组箱子号，“学号”替换为自己的学号）。

| | |
|---|---|
| \|-la4_箱子号_学号/ | 目录，lab4 作品。 |
| \| \|--lab4_箱子号_学号.pdf/ | Lab4 实验报告，实验报告模板参考“Lab03 实验报告模板_仅供参考.docx” |
| \| \|--**myCPU /** | **目录，myCPU 源码。目录请加一个 readme，简单描述下各文件。** |

## 1.6 实验环境

本次实验我们只发布软件环境（func_lab4.zip），和上次的软件相比，本次软件去掉了一些 NOP 指令，所以会有一些寄存器写后读数据相关。

本次实验的步骤是：

1) 准备好 lab3 的实验环境，UCAS_CDE，该环境会也会作为 lab4 的实验环境。
2) 将 lab4 发布的软件程序 func_lab4.zip，解压后，将 func_lab4/拷贝到 UCAS_CDE/soft/目录里，与 func_lab3 同层次。
3) 打开 cpu132_gettrace 工程（UCAS_CDE/cpu132_gettrace/run_vivado/cpu132_gettrace/cpu132_gettrace.xpr）。
4) 参考 1.6.1 节的说明，对 cpu132_gettrace 工程中的 inst_ram 重新定制，此时选择加载 func_lab4 的 coe（UCAS_CDE/soft/func_lab4/obj/inst_ram.coe）。
5) 运行 cpu132_gettrace 工程的仿真（进入仿真界面后，直接点击 run all 等待仿真运行完成），生成新的参考 trace 文件 golden_trace.txt（UCAS_CDE/cpu132_gettrace/golden_trace.txt）。要等仿真运行完成，golden_trace.txt 才有完整的内容。
6) 打开 myCPU 工程（UCAS_CDE/mycpu_verify/run_vivado/mycpu_prj1/mycpu_prj1.xpr）。
7) 参考 1.6.1 节的说明，对 myCPU 工程中的 inst_ram 重新定制，此时选择加载 func_lab4 的 coe（UCAS_CDE/soft/func_lab4/obj/inst_ram.coe）。
8) 运行 myCPU 工程的仿真（进入仿真界面后，直接点击 run all），开始 debug。

### 1.6.1 重新定制 inst_ram

本次软件环境较上次实验有变化，但 Vivado 工程中的 inst_ram 里面仍然装载着 lab3 的软件，所以我们需要重新定制 inst_ram 以进行后续的实验。用于重新定制 inst_ram 的 coe 文件位于新发布的 func_lab4/obj/inst_coe。重新

定制 inst_ram 的流程如下:

1) 在 Sources 窗口中找到 inst_ram IP，双击或者右键后点击”Re-customize IP…”，进入定制界面；

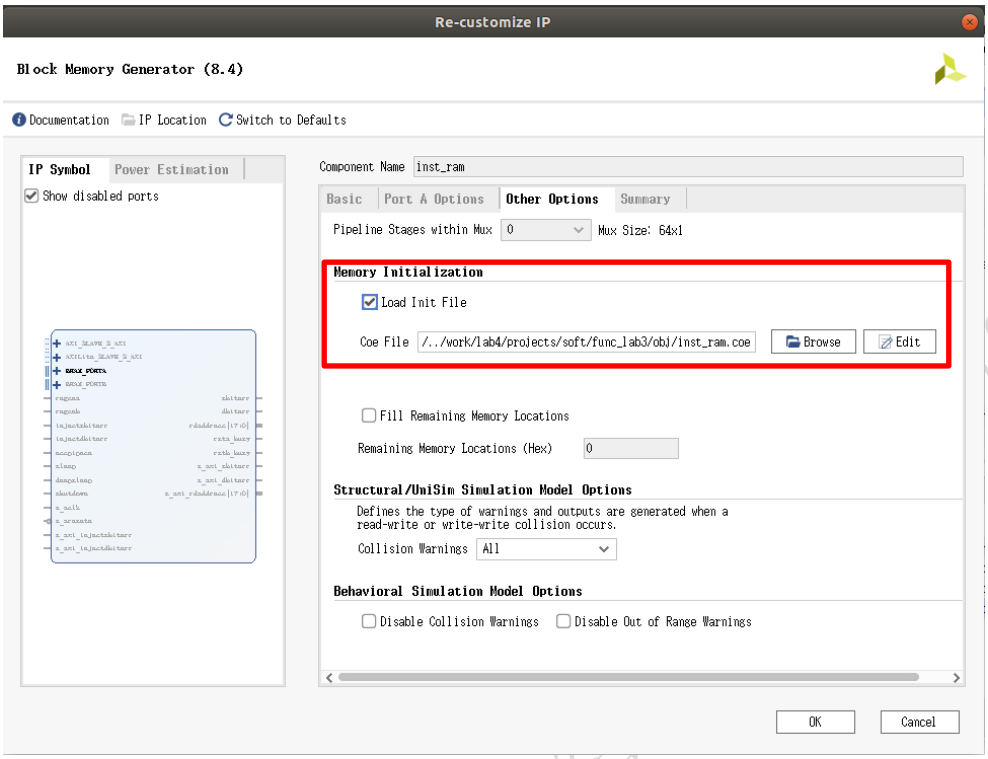


**图 1-1 重新定制 inst ram IP 界面**

2) 在弹出的重新定制 IP 界面中选择 Other Options 选项卡，勾选 Load Init File，并且在 Coe File 中通过点击“Browse”选择新生成的 inst_ram.coe，点击 OK。

3) 在弹出的 Generate Output Product 窗口中点击 Generate（根据需要选择 global 或 ooc 模式），完成 inst_ram 的重新定制。

### 1.6.2 重新生成 golden_trace.txt

golden_trace.txt 可以由 cpu132_gettrace 工程生成。此工程内部集成了一个 GS132CPU 核，阅读 testbench 文件可以知道，它在仿真的时候会把每一次发生寄存器写回时的 PC、写回寄存器号以及写回内容写入 trace 文件中。重新生成 golden_trace.txt 的步骤如下：

1) 打开 cpu132_gettrace 工程，按照 1.6.1 步骤重新定制其中的 inst_ram。[1]

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

### 1.6.3 升级工程和 IP

**本小节作为 lab3 实验任务书的补充。如果已经知道如何升级 IP，请略过本节。**

在低版本 Vivado 中创建的工程如果在高版本 Vivado 中打开，需要进行工程的升级和 IP 核升级。此过程与重新定制 IP 的过程相似：

1) 高版本 Vivado 打开低版本的工程，会弹出如下界面，选择第一个选项“Automatically Upgrade…”，点击“OK”进行升级。

[1]也可以略过此步骤，使用 1.6.4 提供的方法进行仿真

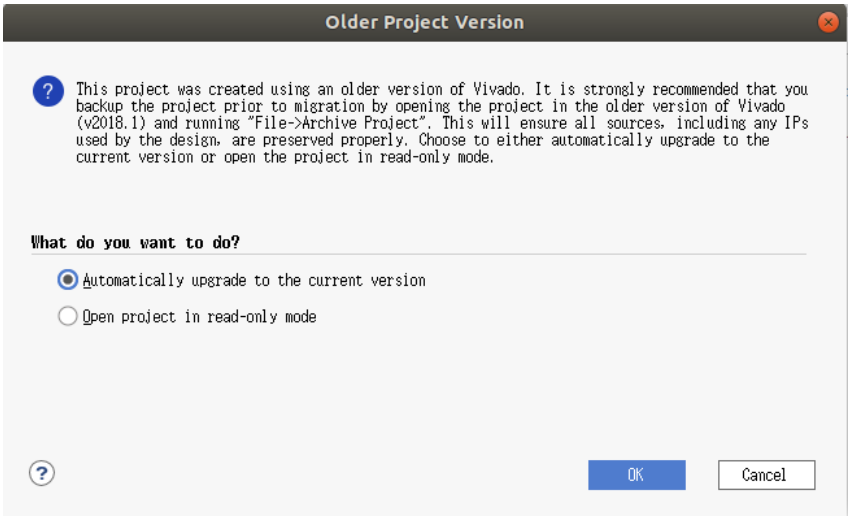


**图 1-2 提升将工程升级**

2) 在 Sources 窗口中找到要升级的 IP，右键后点击"Upgrade IP…"，会弹出如下界面，选择第二个选项"Continue with Core …"，点击"OK"即可。

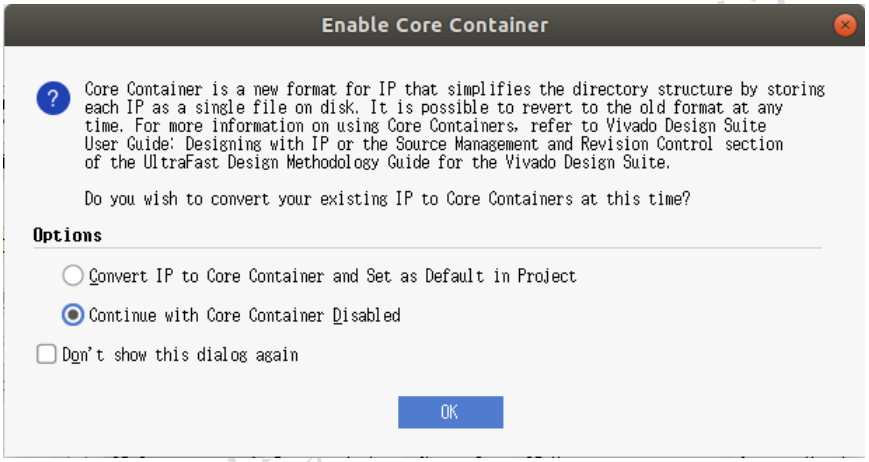


**图 1-3 提示 IP 升级的选项**

3) 在弹出的 Generate Output Product 窗口中点击 Generate（根据需要选择 global 或 ooc 模式），完成升级。

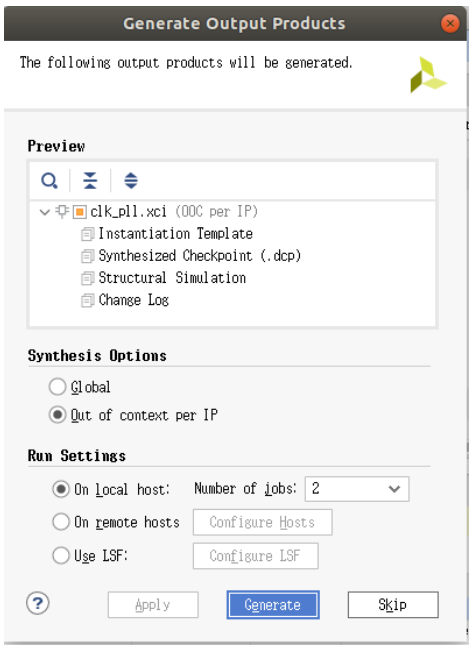


**图 1-4 完成 IP 升级**

### 1.6.4 替换 mif 文件快速仿真

**本小节内容仅供参考，如果未掌握本节内容，完全不影响本次实验。**

本小节提供一种方法：不重新定制 inst ram，直接使用新的 func 进行仿真。

前面讲到的重新定制 inst_ram 比较耗时，如果只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下面的目录”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（UCAS_CDE/soft/func_lab4/obj/inst_ram.mif），再点击”Restart”回到零时刻，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。