# 本历史

| 文档更新记录 | 文档名: | Lab03_简单 CPU 参考设计调试 |
|---|---|---|
| | 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 : | 2019-09-10 |

更新历史

| 序号 | 更新日期 | 更新人 | 版本号 | 更新内容 |
|---|---|---|---|---|
| 1 | 2019/09/10 | 贾凡 | - | 草稿。 |
| 2 | 2019/09/11 | 邢金璋 | V0.1 | 初稿。 |
| | | | | |
| | | | | |

文档信息反馈: xingjinzhang@loongson.cn

# 1 实验三 简单 CPU 参考设计调试

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）。

通过本章节的学习，你将获得：

(1) 无阻塞五级流水线 CPU 设计的知识。

(2) 对 CPU 实验开发环境的熟悉。

(3) 对 CPU 设计出错类型的识别、调试的能力。

## 1.1 实验目的

1. 加深对指令集的理解。
2. 了解 CPU 设计中流水线切分的原理。
3. 学会识别常见波形异常，理解其产生原因，并能修正。
4. 熟悉并运用 verilog 语言进行电路设计。

## 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

本次实验分为两个子任务：

1. 子任务一：阅读并理解我们提供的代码。结合讲义，画出简单流水线 CPU 的结构设计框图，然后对照自己画的结构框图，对代码做简要阐述。
2. 子任务二：代码调试。我们提供的代码中加入了 7 处 bug，请大家通过仿真波形的调试修复这些 bug，使设计可以通过仿真和上板测试。
3. 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：子任务一要求结构设计框图和相应的阐述简洁、正确且完整，子任务二要求记录下所有 bug 的调试过程和机理分析。

   (2) 现场检查评分：子任务二的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 17 日进行检查。

现场以抽查的方式进行，分仿真检查和上板检查：

1) 仿真检查：子任务二对照波形进行描述各流水级的工作内容。

2) 上板检查：子任务二查看上板行为。

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 17 日 18:10。

提交内容：纸质档 lab3 实验报告。

**(2) 电子档提交**

提交方式：打包上传到 Sep 课程网站 lab2 作业下，每人都必须要有。

截止时间：2019 年 9 月 17 日 18:10（具体以课程通知为准）。

提交内容：电子档 lab3 实验报告，为 pdf 文档，文件名是“lab3_箱子号_学号”（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）。

## 1.6 实验环境

本次实验提供了 Vivada 工程文件和全部的 RTL 文件以及生成好的 IP，同学们需要调试的 CPU 设计源码位于 mycpu_verify/rtl/myCPU 中。

注意：源码中包含 7 处 bug，请先理解 CPU 设计代码，画出设计框图，再进行调试。调试过程中请多多交流、多多提问。

| | |
|---|---|
| \|-cpu132_gettrace/ | 用龙芯开源 GS132 处理器核生成参考 trace 的部分。 |
| \| \|--rtl/ | SoC_lite 设计代码目录。 |
| \| \| \|--soc_lite_top.v | SoC_lite 的顶层文件。 |
| \| \| \|--CPU_gs132/* | 龙芯开源 GS132 处理器核设计。 |
| \| \| \|--CONFREG/ | confreg 模块，用于访问 CPU 与开发板上数码管、拨码开关等外设。 |
| \| \| \|--BRIDGE/ | 1×2 的桥接模块，CPU 的 data sram 接口同时访问 confreg 和 data_ram。 |
| \| \| \|--xilinx_ip/ | 封装后的 Xilinx IP，包含 clk_pll、inst_ram、data_ram。 |
| \| \|--testbench/ | 功能仿真验证目录。 |
| \| \| \|--tb_top.v | 仿真顶层，该模块会抓取 debug 信息生成到 trace_ref.txt 中。 |
| \| \|--run_vivado/ | Vivado 工程的运行目录。 |
| \| \| \|--soc_lite.xdc | Vivado 工程设计的约束文件。 |
| \| \| \|--cpu132_gettrace/ | 创建的 Vivado 工程，名字就叫 cpu132_gettrace。 |
| \| \| \| \|--cpu132_gettrace.xpr | Vivado 创建的工程文件。 |
| \| \|--trace_ref.txt | 利用 GS132 处理器核运行 func 测试程序所生成的参考 trace。 |
| \| | |

| | |
|---|---|
| \|-soft/ | 功能验证所用测试程序代码及其编译环境所在目录。 |
| \| \|--lab*XX*_func/ | 第 *XX* 次实验任务所用的功能验证测试程序。 |
| \| \| \|--include/ | 功能验证测试程序共享的头文件所在目录。 |
| \| \| \| \|--asm.h | MIPS 汇编需用到的一些宏定义的头文件，比如 LEAF(x)。 |
| \| \| \| \|--regdef.h | MIPS o32 ABI 下，32 个通用寄存器的汇编助记定义。 |
| \| \| \| \|--ucas_cde.h | SoC_Lite 相关参数的宏定义，如访问数码管的 confreg 的基址。 |
| \| \| \| \|--inst_test.h | 各功能测试点的验证程序使用的宏定义头文件。 |
| \| \| \|--inst/ | 各功能测试点的汇编程序文件。 |
| \| \| \| \|--Makefile | 子目录里的 Makefile，会被上一级目录中的 Makefile 调用。 |
| \| \| \| \|--n*.S | 各功能测试点的验证程序，汇编语言编写。 |
| \| \| \|--obj/ | 功能验证测试程序编译结果存放目录 |
| \| \| \| \|--* | 详见讲义第 3.2.5 节说明。 |
| \| \| \|--start.S | 功能验证测试的引导代码及主函数。 |
| \| \| \|--Makefile | 编译功能验证测试程序的 Makefile 脚本 |
| \| \| \|--bin.lds.S | 链接脚本源码 |
| \| \| \|--bin.lds | 编译 bin.lds.S 得到的结果，可被 make reset 命令清除 |
| \| \| \|--convert.c | 生成 coe 和 mif 文件的处理工具的 C 程序源码 |
| \| \| \|--convert | convert.c 的本地编译后生成的可执行文件，可被 make reset 命令清除 |
| \| \| \|--rules.make | 子编译脚本，被 Makefile 调用 |
| \| \|--* | 其他功能或性能测试程序。 |
| \| | |
| \|-mycpu_verify/ | 学生实现的 CPU 的验证开发环境 |
| \| \|--rtl/ | SoC_lite 设计代码目录。 |
| \| \| \|--soc_lite_top.v | SoC_lite 的顶层文件。 |
| \| \| \|--myCPU/* | **同学们需要调试的 CPU 的 RTL 代码。** |
| \| \| \| \|--mycpu.h | myCPU 中需要使用的宏定义 |
| \| \| \| \|--mycpu_top.v | CPU 的顶层文件 |
| \| \| \| \|--IF_stage.v | 取指级代码 |
| \| \| \| \|--ID_stage.v | 译码级代码 |
| \| \| \| \|--EXE_stage.v | 执行级代码 |
| \| \| \| \|--MEM_stage.v | 访存级代码 |
| \| \| \| \|--WB_stage.v | 写回级代码 |
| \| \| \| \|--alu.v | 算术逻辑单元 |
| \| \| \| \|--regfile.v | 寄存器模块 |
| \| \| \| \|--tools.v | 其它小模块，只要包含一些译码器 |
| \| \| \| \|--mycpu_top.v | CPU 的顶层文件 |
| \| \| \| \|--IF_stage.v | 取指级代码 |
| \| \| \|--CONFREG/ | confreg 模块，用于访问 CPU 与开发板上数码管、拨码开关等外设。 |
| \| \| \|--BRIDGE/ | 1×2 的桥接模块， CPU 的 data sram 接口同时访问 confreg 和 data_ram。 |
| \| \| \|--xilinx_ip/ | 封装后的 Xilinx IP，包含 clk_pll、inst_ram、data_ram。 |
| \| \|--testbench/ | 功能仿真验证平台。 |
| \| \| \|--mycpu_tb.v | 功能仿真顶层，该模块会抓取 debug 信息与 trace_ref.txt 进行比对。 |
| \| \|--run_vivado/ | Vivado 工程的运行目录。 |

| | |
|---|---|
| \|  \|  \|--soc_lite.xdc | Vivado 工程设计的约束文件。 |
| \|  \|  \|--mycpu_prj1/ | 创建的第一个 Vivado 工程，名字为 mycpu_prj1。 |
| \|  \|  \|  \|--*.xpr | Vivado 创建的工程文件，可直接打开。 |
| \|  \|  \|--* | 创建的其他 Vivado 工程。 |