32 位 CPU 计软联合实验组

测试文档

Totoro 组

杨乐 琚锡廷 董原良 李林涵

2019 年 1 月

1 文档说明

本文档是 Totoro 组 32 位 MIPS CPU 的测试文档，由于开发此项目与普通的软件项目有很大的不同，每一个开发周期要分别设计不同的测试方法，故在项目初需要大致了解定下测试计划，以免前一阶段的错误影响到后续的测试。

以下为我们组的开发周期，加粗字段为需要测试的阶段：

A. Sprint1 需求分析：分析 ucore 操作系统及需求（MIPS、流水线、操统）

B. Sprint2 流水线搭建、异常处理、访存实现、仿真测试：流水线搭建，异常与中断的支持，MMU 与 TLB，SRAM 控制器

C. Sprint3 CPU 联合测试、上板子：流水线、异常处理、访存联合调试，并上板子通过测试测例

D. Sprint4 操作系统测试、外设支持：根据操作系统进行调试，添加对 Flash、VGA 的支持

E. Sprint5 额外功能：改写 CPU 与操作系统使其支持额外功能

另外除 CPU 的测试外，我们为每个外设也单独设立了测试环节。

2 流水线指令测试

针对流水线尚未搭建好时。通过手动编写机器指令，将指令放在固定的空间让 CPU 顺序执行，观察输出波形，与预期输出波形进行比对。本阶段的测试参考的是《自己动手造 CPU》上的测试方法。

3 功能测例

3.1 简介

针对流水线已搭建好后。功能测例为助教提供的一组系统的测试样例，可以测试若干条操作系统所需要指令的正确性。在运行功能测例时，序号为 4 的寄存器储存的是此时正在测试的指令序号，序号为 19 的寄存器则存储了已经通过的指令条数，我们可以通过观察 4、19 号寄存器的数值来确定是否通过测试，以及具体是哪一条指令出现错误；以下是几种测试的方式，适用于不同的情况。

3.2 测试方法

a. 行为仿真，访存使用数组模型

此时流水线部分已经搭建完成，但访存 SRAM 部分还有待单独测试。故这里使用数组模型来代替 SRAM 进行读写.

b. 行为仿真，访存使用 SRAM 仿真模型

在助教提供的 Thinkpadtop 工程中提供了 SRAM 仿真模型，里面通过约束时序关系对 SRAM 进行仿真。在 SRAM 模块完成后，可以连接 SRAM 仿真模型，并进行仿真。

c. 上板测试，用实际的板子上进行测试

这时需要把序号为 19 的寄存器的值连接到板子的 LED 上，通过 LED 上的数字来确认通过指令的条数。需要注意，在生成测例的二进制时，可以指定目标是仿真还是真实板子；若是板子，则在测试指令间有约为 1s 的间隔，方便大家可以肉眼观察到 LED 的跳动情况；此开关可以通过 Makefile 来查看。

4 操作系统测试

4.1 简介

针对功能测试通过后。此时板子上功能测例已经通过，我们可以通过操作系统来测试 CPU 的正确性。

首先在这个阶段我们无法通过行为仿真来测试，因为操作系统的执行条数过多，仿真时间很长；故我们只能通过板子来进行测试。在进行此测试前，需把外设的串口通信调试好，因为操作系统的反馈信息只能通过串口给到。

在此过程中，我们运用了两种调试方式，如下所示。

4.2 测试方法

a. 操作系统输出

当 CPU 有问题时，操作系统会在有错误的地方停下 ;这是我们可以通过修改操作系统，往其中加入输出信息，来确定错误的位置，从而推测出出错的原因。

不过此方法对分析在内核态的错误比较有用（操作系统初始化时）。此时我们可以得到错误函数的位置，定位出反汇编出的 MIPS 指令，从而推断出错误原因。但对于处于用户态的错误，由于其指令无法定位，故推测时只能通过异常帧来确定，错误较难被发现。

b. JTAG 调试

JTAG 调试的原理是，在硬件中插入探针(probe)，并通过硬件直接监听其中的波形。如果合理的设定触发条件（即当探针侦测到某波形时），即可返回附近的波形；与操作系统只能在软件端的调试相比，JTAG 可以在硬件端进行信号抓取并回送，只要合理设置触发条件，就能大大方便调试的进度。但由于 JTAG 需要改写硬件的结构，所以监听大量信号会增加综合的时间，这方面需要作出一定的平衡。

5 串口测试

5.1 状态机设计

由于串口通信无法进行行为仿真，所以我们直接进行上板测试

串口模块的测试电路可以用一个状态机描述，共有四个状态：

* **IDLE**
    * 状态机的初始状态
    * 读取 0xC 地址，判断当前串口是否收到了消息
        * 如果读出数据的次低位为 1，则下一周期跳到 RECEIVE 状态
        * 否则保留在当前状态
* **RECEIVE**
    * 读取串口收到的消息
    * 读取 0x8 地址，解析收到的消息
        * 如果收到了消息为 FF，则下一周期跳到 SEND_PREPARE 状态
        * 否则跳回 IDLE 状态
* **SEND_PREPARE**
    * 准备发送消息，保证串口发送的消息都能够顺利发出
    * 读取 0xC 地址，判断当前串口是否可写
        * 如果读出数据的次低位为 1，说明收到了来自 PC 的停止信号，下一周期跳到 IDLE 状态
        * 如果读出数据的最低位为 1，则下一周期跳到 SEND 状态，准备发送的数据自增 1
        * 否则保留在本状态
* **SEND**
    * 串口的发送状态
    * 将准备好的数据写入 0x8 地址
    * 下一周期跳到 SEND_PREPARE 状态

5.2 测试流程

将 bit 文件烧到 thinpad 上，打开直联串口，在 PC 上打开串口调试精灵，在电脑上发送一个“FF”，观察 PC 收到的结果，再发送一个“EE”，观察 PC 是否不再收到信息。

5.3 预期结果

第一次发送"FF"后，PC 收到大量数据（公差为 1 的循环等差数列），再次发送"EE"后，PC 停止收到信息。

6 SRAM 测试

6.1 行为仿真

Level1: sram 采用数组模型, 循环进行 :连续写两个相邻地址, (回退到最开始的地址) 连续读两个相邻地址，(回退到最开始的地址) 先读后写再读同一地址，sram 控制器的频率设置为状态机频率的两倍，再设置适当相位差，观察仿真后的波形是否正确，此阶段的目的是为了测试 sram 控制器的组合逻辑部分是否有误；

Level2: sram 采用模板工程中提供的仿真模型，仍然使用上述状态机，观察仿真后的波形是否正确，此阶段是在前一阶段基础上，测试 sram 控制器的时序逻辑在行为仿真（不加数据通路延迟）的情况下是否正确。

6.2 与串口联合测试

6.2.1 状态机设计

*** IDLE**

- 循环判断串口是否产生中断信号
  - 若产生了中断，则下一周期跳到 WRITE_SRAM 状态

*** WRITE_SRAM**

- 将当前数据写入 sram 当前地址，数据自增 1
- 下个周期跳到 READ_SRAM 状态

*** READ_SRAM**

- 从 sram 读取当前地址的内容，地址自增 1
- 将读到的数据缓存到寄存器里
- 下个周期跳到 SEND_PREPARE 状态

*** SEND_PREPARE**

- 准备发送消息，保证串口发送的消息都能够顺利发出
- 读取 0xC 地址，判断当前串口是否可写
  - 如果读出数据的最低位为 1，则下一周期跳到 SEND 状态
  - 否则保留在本状态

*** SEND**

- 串口的发送状态
- 将 READ_SRAM 中读到的数据写入 0x8 地址
- 下一周期跳到 WRITE_SRAM 状态

6.2.2 测试流程

将 bit 文件烧到 thinpad 上，打开直联串口，在 PC 上打开串口调试精灵，在电脑上发送一个“FF”，观察 PC 收到的结果。

6.2.3 预期结果

发送"FF"后，PC 收到大量数据（公差为 1 的循环等差数列）。

7 Flash 测试

7.1 行为仿真

利用模板工程提供的 flash 模型，在 flash 文件中事先写入数据，在一个小型状态机做连续读取 flash 的行为仿真（用 flashStall 信号控制状态机的跳转），观察输出波形和 flash 文件中的数据是否一致。

7.2 与串口联合测试

7.2.1 状态机设计

* **IDLE**
    * 循环判断串口是否产生中断信号
        * 若产生了中断，则下一周期跳到 READ_FLASH 状态
* **READ_FLASH**
    * 循环判断 flash 是否读取完毕（flashStall 信号是否为 0）
    * 读取完毕时
        * 将读到的数据缓存到寄存器里
        * 下个周期跳到 SEND_PREPARE 状态
* **SEND_PREPARE**
    * 准备发送消息，保证串口发送的消息都能够顺利发出
    * 读取 0xC 地址，判断当前串口是否可写
        * 如果读出数据的最低位为 1，则下一周期跳到 SEND 状态
        * 否则保留在本状态
* **SEND**
    * 串口的发送状态
    * 将 READ_SRAM 中读到的数据写入 0x8 地址
    * 下一周期跳到 READ_FLASH 状态

7.2.2 测试流程

先在板子上烧一个 bin 文件到 flash 上，将 bit 文件烧到 thinpad 上，打开直联串口，在 PC 上打开串口调试精灵，在电脑上发送一个“FF”，观察 PC 收到的结果与之前在 flash 上写入的结果是否一致。

8 BootRom 测试

8.1 行为仿真

将 PC 的初始地址改为 0xbfc00000，接入 BootRom 和 Flash，利用模板工程提供的 flash 模型，在 flash 文件中事先写入操作系统的 elf 文件，观察一段时间后，PC 能否顺利跳转到 0x80000000。

8.2 上板测试

8.2.1 测试流程

将 PC 的初始地址改为 0xbfc00000，在之前的 bootloader 汇编代码的最后加入一条写串口的语句，再加上几条陷入 nop 循环的语句（去掉之前的跳转语句），重新生成 BootRom，在板上 flash 芯片上事先写入操作系统的 elf 文件，观察串口是否有输出，若有输出，再利用 thinpad 提供的工具将 sram 中的全部内容复制出来，与用 elf 生成的 bin 文件进行对比。

8.2.2 预期结果

串口输出了正确的调试信息，对比两个 bin 文件完全相同，说明 bootloader 正确地将 flash 中的操作系统转移到了主存中。