

OpenMIPS 快速入门手册

——教学版(VHDL)

Lei

leishangwen@163.com

2013-12-5

v1.0

1 OPENMIPS 项目简介	3
2 OPENMIPS 支持的指令	4
3 OPENMIPS 的流水线	4
4 OPENMIPS 中的寄存器	4
5 OPENMIPS 的异常模型	5
6 OPENMIPS 教学版主要特点	5
7 OPENMIPS 教学版文件目录	7
8 测试例程	9
9 测试步骤	11
9.1 测试环境搭建	11
9.2 直接使用完整的 OPENMIPS 进行试验	15
9.2.1 新建 ModelSim 工程	16
9.2.2 编译测试程序	16
9.2.3 仿真	17
9.3 按照 OPENMIPS 实现步骤进行试验	19
9.2.1 新建 ModelSim 工程	19
9.2.2 编译测试程序	19
9.2.3 仿真	19

1 OpenMIPS 项目简介

OpenMIPS 开始于 2013 年 8 月，目的是开发一款 32 位、兼容 MIPS32 指令集的开源软核处理器，便于老师教学、学生体会理解计算机体系结构课程的相关知识，同时也可以做实际用途。OpenMIPS 将坚持自由软件的理念，保持开源的形式，同时采用商业友好的 LGPL 授权。并且分为两个版本：教学版、实践版，每个版本都使用 VHDL、Verilog HDL 两种语言编写，这样实际是有四个版本，分别命名为（以 1.0 版为例）

- | | |
|----------------------------------|-----------------|
| ✓ OpenMIPS_VHDL_study_v1.0 | 教学版（VHDL）1.0 |
| ✓ OpenMIPS_Verilog_study_v1.0 | 教学版（Verilog）1.0 |
| ✓ OpenMIPS_VHDL_practice_v1.0 | 实践版（VHDL）1.0 |
| ✓ OpenMIPS_Verilog_practice_v1.0 | 实践版（Verilog）1.0 |

教学版的主要设想是尽量简单，比如：在一个时钟周期内可以取到指令，完成存储、加载数据，这样处理器的运行情况（比如：流水线的运行）就比较理想化，与教科书相似，代码也很清晰简单，便于使用其进行教学、学术研究和讨论，也有助于各位同学理解课堂上讲授的知识。

实践版的主要设想是使 OpenMIPS 成为一个实际可用的处理器，能够下载到 FPGA 上，运行实际有用的程序，为此，添加了 wishbone 总线接口，使其可以挂接在 wb_conmax 互联矩阵上，这样就能方便的利用 OpenCores 上提供的 SDRAM、Flash、GPIO、UART、LCD 等模块控制器，组成一个 SOPC，完成特定功能，进一步还可为其移植操作系统。

OpenMIPS 是采用具有哈佛结构的 32 位标量处理器，兼容 MIPS32 体系结构，这样可以使使用现有的 MIPS 编译环境。具有以下特点：

- 五级整数流水线，分别是：取指、译码、执行、访存、回写
- 哈佛结构，分开的指令、数据接口
- 32 个 32 位整数寄存器
- 大端模式
- 向量化异常处理，支持精确异常处理
- 8 个外部中断
- 32bit 数据、地址总线宽度
- 单周期乘法
- 支持延迟转移
- 兼容 MIPS32 体系结构，支持 MIPS32 指令集中的所有整数指令
- 大多数指令可以在一个时钟周期内完成
- 可综合
- 兼容 wishbone b2 的指令、数据总线接口（只有实践版提供）
- LGPL 开源
- VHDL、Verilog HDL 两种语言版本

本手册主要介绍了 OpenMIPS 教学版（VHDL），包括其主要特点、文件目录结构、测试例程的作用、测试步骤等，通过本手册，读者可以快速了解 OpenMIPS 的基本情况。同时掌握利用 OpenMIPS 教学版进行测试的步骤。（与本教程同步发布的还有一个视频教程，用

友们可以下载)。

希望广大读者不吝赐教，针对设计中可能存在的问题及时指出。

让我们为了 OpenMIPS 的茁壮成长共同努力！

2 OpenMIPS 支持的指令

OpenMIPS 支持 MIPS32 指令集中的如下指令（包含全部整数指令），各指令的具体含义可以参考 doc 目录下的文档《MIPS32 指令集》。

- 逻辑操作指令 AND、ANDI、LUI、NOR、OR、ORI、XOR、XORI
- 移位操作指令 SLL、SLLV、SRA、SRAV、SRL、SRLV
- 算术操作指令 ADD、ADDI、ADDIU、ADDU、CLO、CLZ、SLT、SLTI、SLTIU、SLTU、SUB、SUBU、MADD、MADDU、MSUB、MSUBU、MUL、MULT、MULTU、DIV、DIVU
- 移动操作指令 MFHI、MFLO、MOVN、MOVZ、MTHI、MTLO
- 控制指令 NOP、SSNOP
- 跳转指令 J、JAL、JALR、JR
- 分支指令 B、BAL、BEQ、BGEZ、BGEZAL、BGTZ、BLEZ、BLTZ、BLTZAL、BNE
- 加载类指令 LB、LBU、LH、LHU、LL、LW、LWL、LWR
- 存储类指令 SB、SC、SH、SW、SWL、SWR
- 协处理器访问指令 MFC0、MTC0
- 自陷指令 SYSCALL、TEQ、TEQI、TGE、TGEI、TGEIU、TGEU、TLT、TLTI、TLTIU、TLTU、TNE、TNEI
- 异常返回指令 ERET

3 OpenMIPS 的流水线

OpenMIPS 具有五级流水线，在各个阶段完成的工作如下：

- 取指阶段：从指令存储器取得指令，修改 pc 的值
- 译码阶段：指令译码，根据译码结果取得指令执行需要的寄存器的值、立即数的值，并判断是否是多周期指令
- 执行阶段：判断并解决数据相关问题，执行指令操作，判断是否转移
- 访存阶段：如果是加载存储指令，那么读写数据存储器
- 回写阶段：写目的寄存器。判断是否有异常发生

4 OpenMIPS 中的寄存器

OpenMIPS 支持 32 个 32 位整数寄存器，还支持协处理器 CP0 中的如下寄存器：COUNT、COMPARE、STATUS、CAUSE、EPC、CONFIG、PrId、BadVAddr、ErrorEPC。这些寄存器的含义可以参考 doc 目录下的文档《MIPS32_3_Privileged_Resource》，或者《MIPS 体系

结构与编程》、《MIPS 体系结构透视（See MIPS Run）》等书籍，本手册不再罗列。

5 OpenMIPS 的异常模型

OpenMIPS 支持精确异常模型，当异常发生时，有关处理器的状态信息被存储到协处理器 CP0 的部分寄存器中，然后 OpenMIPS 处理器转移到事先定义好的一个地址，在那个地址中往往存储有异常处理例程，在其中进行异常处理，这个地址称为异常处理例程入口地址。

具体来说，当异常发生时，OpenMIPS 会进行如下操作：

（1）设置 EPC 寄存器：如果当前指令不在延迟槽中，那么当前指令（或下一条指令）地址被存储到 EPC 寄存器中，如果当前指令在延迟槽中，那么当前指令的上一条转移指令地址被存储到 EPC 寄存器中。

（2）设置 CASUE 寄存器：在其 ExeCode 字段存储异常原因代码，具体含义可以参考 doc 目录下的文档《MIPS32_3_Privileged_Resource》，或者《MIPS 体系结构与编程》、《MIPS 体系结构透视》等书籍。

（3）设置 STATUS 寄存器：设置其 EXL 位为 1，表示处理器处于异常处理状态。

（4）转移到相应的异常处理例程继续执行。

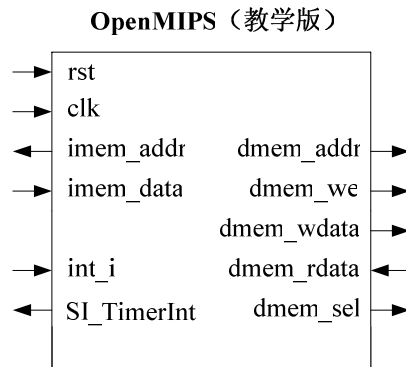
当异常处理结束后，需要使用指令 eret 从异常返回，eret 指令会将 EPC 的值恢复到 PC，同时设置 STATUS 的 EXL 位为 0，处理器回到异常发生前的状态继续执行。

OpenMIPS 支持的异常类型及对应处理例程的入口地址如下表所示，其中无效指令、系统调用、自陷的处理例程入口地址都是 0x40，用户也可以自己修改，使得这三个异常的处理例程入口地址不一样。异常的优先级也是可以调整的，朋友们只要理解了 OpenMIPS 的代码，这些都是可以灵活修改的。

异常类型	优先级	处理例程地址	引起异常的条件
复位（Reset）	1	0x0	由软件或硬件复位引起
外部中断	2	0x20	OpenMIPS 支持 8 个外部中断
系统调用（System Call）	3	0x40	使用指令 SYSCALL
无效指令	4	0x40	指令不是 OpenMIPS 支持的指令
自陷（Trap）	5	0x40	使用指令 TEQ、TEQI、TGE、TGEI、TGEIU、TGEU、TLT、TLTI、TLTIU、TLTU、TNE、TNEI 等

6 OpenMIPS 教学版主要特点

OpenMIPS 教学版没有提供 wishbone 总线接口，其接口如下图所示：



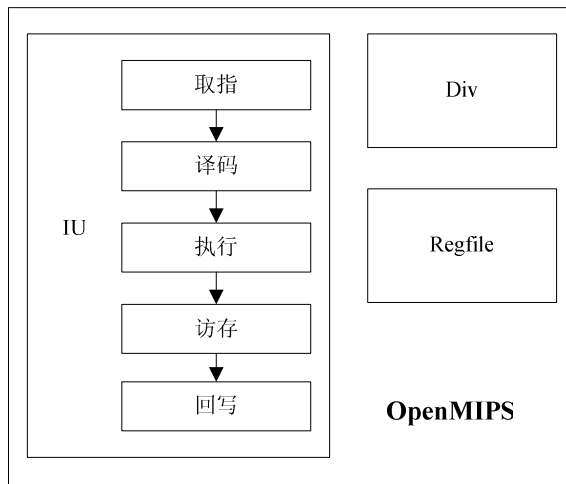
各接口描述如下表所示：

序号	接口名	宽度（bit）	输入/输出	作用
1	rst	1	输入	复位信号，高电平有效
2	clk	1	输入	时钟信号
3	imem_addr	32	输出	指令存储器地址输出
4	imem_data	32	输入	指令存储器数据输入
5	int_r	8	输入	中断信号
6	SI_TimerInt	1	输出	时钟中断信号
7	dmem_addr	32	输出	数据存储器地址输出
8	dmem_we	1	输出	数据存储器写使能
9	dmem_wdata	32	输出	数据存储器数据输出
10	dmem_rdata	32	输入	数据存储器数据输入
11	dmem_sel	4	输出	数据存储器字节有效信号

从上表可以知道 OpenMIPS 与外部指令存储器有两个信号连接，与外部数据存储器有五个信号连接。注意一点，OpenMIPS 教学版要求指令存储器是异步的，同时要求数据存储器的读操作也要是异步的，具体可以参考源文件 `imem.vhd`、`dmem.vhd`。

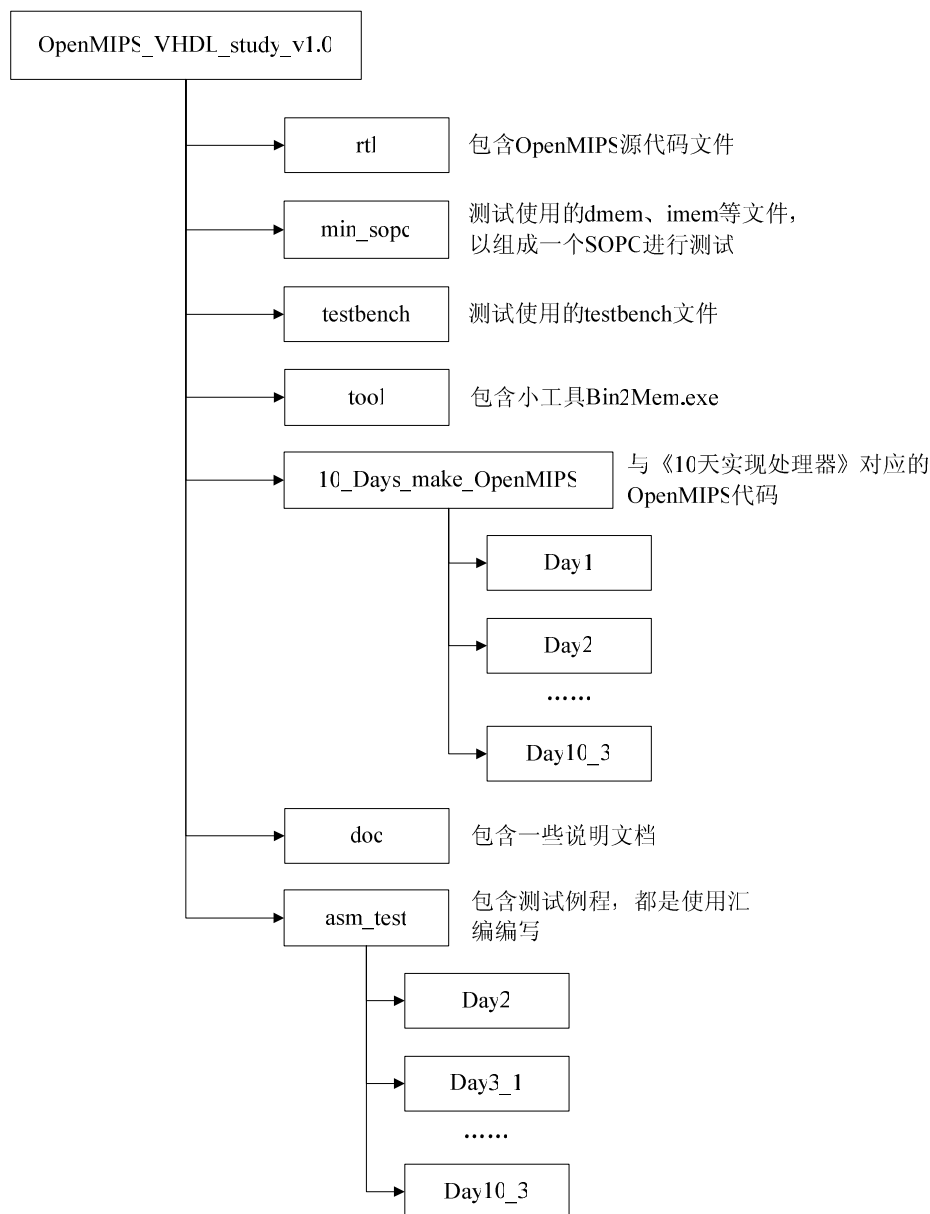
OpenMIPS 支持 8 个外部输入中断，通过 `int_i` 接口输入。

OpenMIPS 内部结构如下图所示，可见内部包含一个整数单元 `IU`、一个除法单元 `Div`、一个寄存器文件单元 `Regfile`，这三个单元分别对应 `iu.vhd`、`div.vhd`、`regfile.vhd` 三个源文件，其中 `IU` 内部实现了五级整数流水线，`Div` 模块实现了 32 位整数除法，采用的是试商法，`Regfile` 实现了 32 个 32 位整数寄存器。此外还有 `OpenMIPS.vhd` 文件实现了顶层模块 OpenMIPS。



7 OpenMIPS 教学版文件目录

OpenMIPS 教学版的文件目录如下：



各个文件夹具体说明如下：

- **rtl**

所有 OpenMIPS 的源代码文件在该文件夹下，包括流水线文件 `iu.vhd`、除法模块 `div.vhd`、寄存器文件 `Regfile.vhd`、顶层文件 `OpenMIPS.vhd`、宏定义文件 `stdlib.vhd`

- **min_sopc**

包括指令存储器 `imem.vhd`，数据存储器 `dmem.vhd`，以及一个用来测试 OpenMIPS 的最小 SOPC 的顶层文件 `OpenMIPS_min_sopc.vhd`，在其中例化 `dmem`、`imem`、`OpenMIPS`，形成一个很小的 SOPC

- **testbench**

包括 testbench 测试文件 `OpenMIPS_min_sopc_tb.vhd`，在其中例化了 `OpenMIPS_min_sopc`

- **tool**

包括一个小工具 `Bin2Mem.exe`，该工具用来将 GCC 编译得到的二进制文件进行格式变化得到 `inst_rom.data` 文件，使用后者初始化 `OpenMIPS_min_sopc` 中的指令存储器 `imem`，以

便进行测试

- **10_Days_make_OpenMIPS**

在之前发布过一些文档介绍如何使用 10 天时间实现处理器 OpenMIPS，本文件夹下的内容是相关资料的整理，包含一个文档《10 天实现处理器——OpenMIPS 成长记》，以及与之对应的每一天的 OpenMIPS 代码，比如：第二天对应的 OpenMIPS 代码位于 Day2 目录下，朋友们通过逐步增长的 OpenMIPS 代码，可以更好的理解 OpenMIPS 的设计

- **doc**

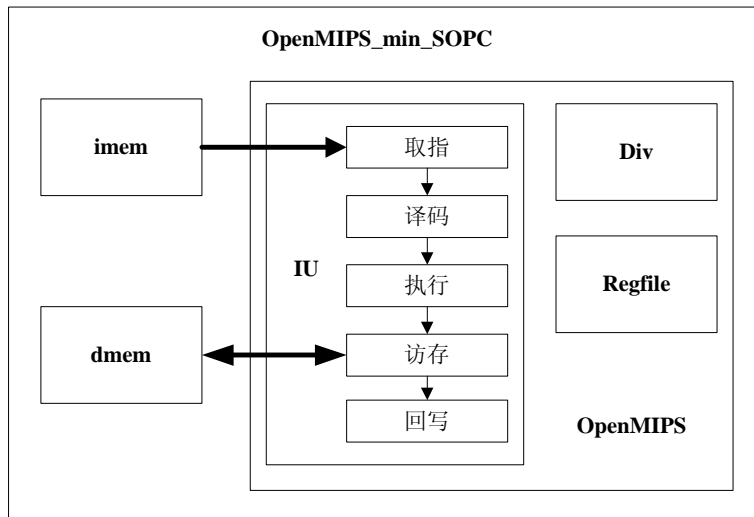
包含一些文档，具体有：《MIPS32 指令集》、《MIPS Architecture for Programmers Volume III》、《OpenMIPS 快速入门手册》

- **asm_test**

包括所有的测试例程，其组织方式是参照《10 天实现处理器——OpenMIPS 成长记》一文，按照“天”来组织，比如 Day2 文件夹中存放的是《10 天实现处理器——OpenMIPS 成长记》一文中第二天对应的测试例程，当然所有的测试例程都可以在最终的 OpenMIPS 中进行测试

8 OpenMIPS_min_sopc

OpenMIPS 只是一个处理器核，为了验证其实现是否正确，还需要提供必要的组件，包括指令存储器，OpenMIPS 从中取出指令执行，还包括数据存储器，用以验证加载、存储指令是否正确实现，为此，添加了指令存储器 imem、数据存储器 dmem，与 OpenMIPS 一起形成一个小小的 SOPC，结构如下图所示：



在教学版中，为了使情况简单，指令存储器 imem 设计是异步的，这样可以在给出地址后立即能得到对应的指令。数据存储器 dmem 的读操作也设计是异步的，这样可以在给出地址后立即能得到对应的数据，dmem 的写操作设计为同步的。具体可以参考 min_sopc 目录下的 imem.vhd、dmem.vhd 文件。

9 测试例程

asm_test 文件夹下的测试程序都是使用汇编语言写的，组织方式与《10 天实现处理器——OpenMIPS 成长记》教程中一致，每一天对应一个或几个文件夹，其中存放的是当天使用的测试程序。具体如下表所示：

文件夹名	测试目的
Day2	验证 ORI 指令是否正确实现
Day3_1	验证是否正确处理数据相关问题
Day3_2	验证是否正确实现其余的逻辑操作指令——AND、ANDI、LUI、NOR、OR、XOR、XORI
Day4_1	验证是否正确实现移位操作指令——SLL、SLLV、SRA、SRAV、SRL、SRLV
Day4_2	验证是否正确实现乘法、除法之外的所有算术操作指令——ADD、ADDI、ADDIU、ADDU、CLO、CLZ、SLT、SLTI、SLTIU、SLTU、SUB、SUBU
Day5_1	验证是否正确实现简单乘法指令——MUL、MULT、MULTU
Day5_2	验证是否正确实现乘法与加法、减法复合——MADD、MADDU、MSUB、MSUBU
Day5_3	验证是否正确实现除法指令——DIV、DIVU
Day6	验证是否正确实现移动操作指令——MFHI、MFLO、MOVN、MOVZ、MTHI、MTLO
Day7_1	验证是否正确实现跳转指令——J、JAL、JALR、JR
Day7_2	验证是否正确实现分支指令——B、BAL、BEQ、BGEZ、BGEZAL、BGTZ、BLEZ、BLTZ、BLTZAL、BNE
Day8_1	验证是否正确实现加载存储指令——LB、LBU、LH、LHU、LL、LW、LWL、LWR、SB、SC、SH、SW、SWL、SWR
Day8_2	验证将加载存储指令放在延迟槽中时，OpenMIPS 是否工作正确
Day9	验证是否正确实现协处理器 0 中的部分寄存器，以及是否正确实现协处理器访问指令——MFC0、MTC0
Day10_1	验证是否正确实现 SYSCALL 指令
Day10_2	验证是否正确实现自陷指令——TEQ、TEQI、TGE、TGEI、TGEIU、TGEU、TLT、TLTI、TLTIU、TLTU、TNE、TNEI
Day10_3	验证时钟中断及处理是否正确实现

程序的预期执行效果在程序的注释中都有说明，也可以参考《10 天实现处理器——OpenMIPS 成长记》一文中的对应章节，其中就有相应测试程序的执行效果说明，以及 ModelSim 仿真结果。

每一天测试例程对应的文件夹下包含如下主要文件：

- inst_rom.S：汇编源程序
- ram.ld：链接文件
- Makefile：编译指导文件
- Bin2Mem.exe：就是 tool 文件夹下的工具

编译后得到如下文件：

- inst_rom.data: 用来初始化指令存储器 imem 的文件
- inst_rom.bin: 编译后得到的二进制文件
- inst_rom.asm: 反汇编文件

10 测试步骤

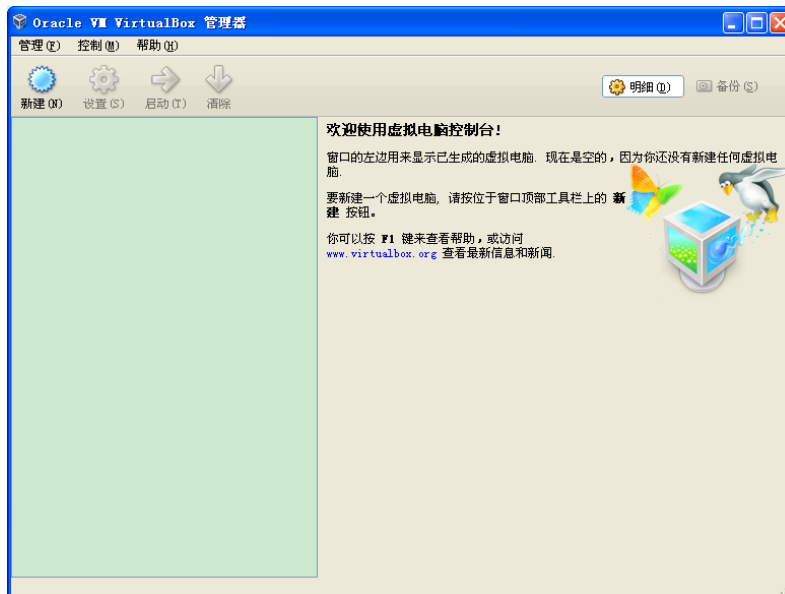
10.1 测试环境搭建

测试需使用如下工具：ModelSim10.1a、GCC、Ubuntu 虚拟机。

GCC 用来进行编译汇编文件，可以在 <http://bbs.eetop.cn/thread-429161-1-1.html> 下载，安装到 Ubuntu11.10 上，朋友们可以使用 OR1200 提供的 Ubuntu 虚拟机镜像，能够在 <ftp://openrisc.opencores.org/virtualbox-image/> 地址下载到，FTP 的用户名和密码都是 openrisc。登录后会出现如下图所示界面。



下载最新的那个文件就可以了，本手册使用的是 2011-12-15 版。下载完成后解压该文件，大约 4GB 左右。此时还需要下载 VisualBox 以打开该文件。VisualBox 是一款开源的虚拟机软件，本手册使用的是 4.1.22 版。下载完成后安装 VisualBox，安装完成后打开 VisualBox，界面如下图所示。



点击“新建”出现“新建虚拟机”向导，点击“下一步”，出现如下界面：



此处操作系统选择 Linux，版本选择 Ubuntu，点击下一步，设置内存大小，如下图所示。



图 1.6 新建虚拟机设置二

内存大小依据个人情况设置，本手册设置的是 512M，已经够用了，毕竟我们需要编译的程序都是十分简单的，点击下一步，选择“使用现有的虚拟硬盘”，然后选择解压后的虚拟机文件。



图 1.7 新建虚拟机设置三

点击“下一步”，VisualBox 会将用户刚才的设置都列出来，确认无误后，点击“创建”，这样虚拟机就创建好了。启动虚拟机，显示如下图所示。

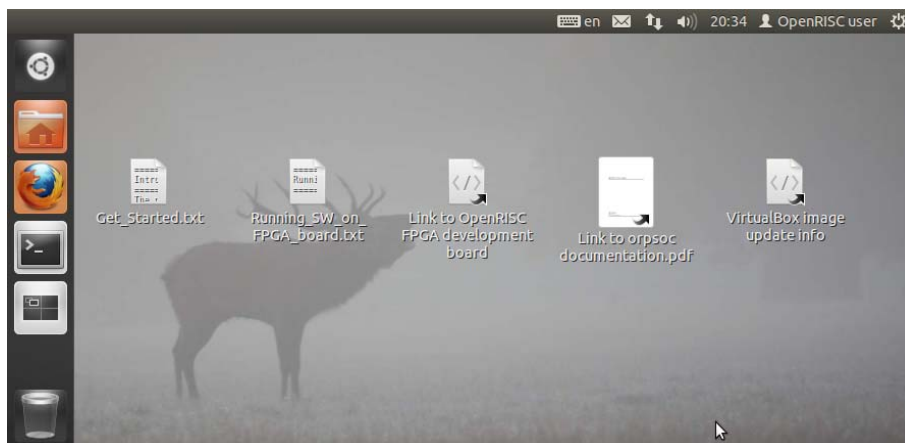
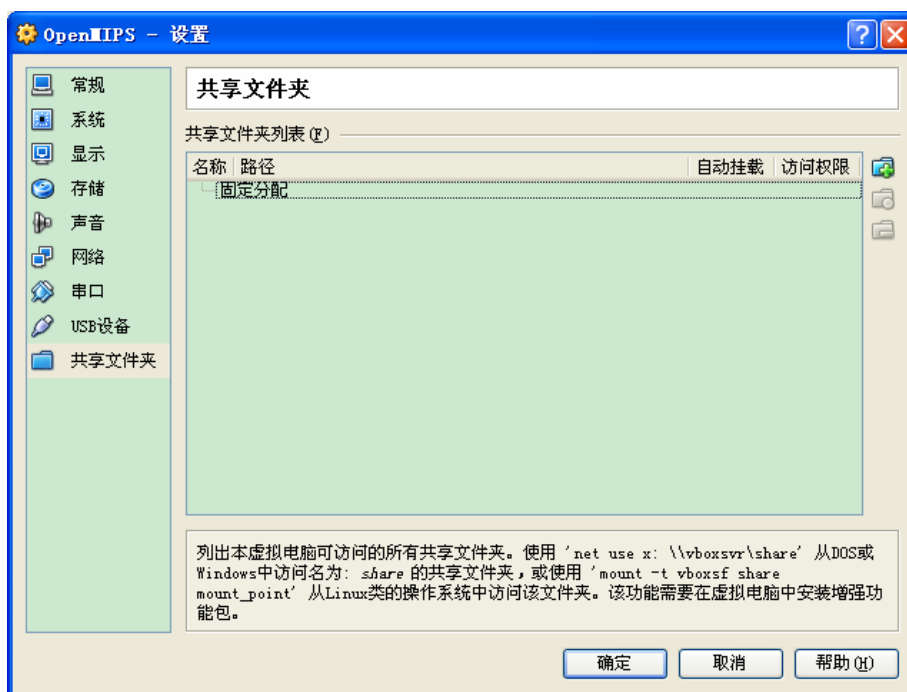
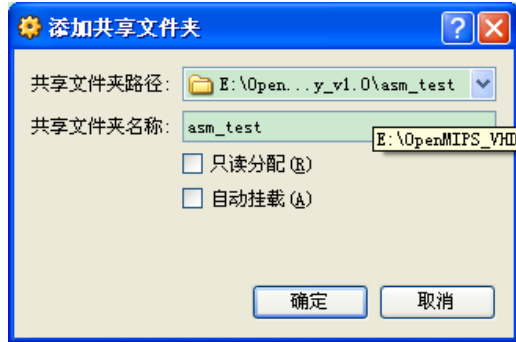


图 1.8 Ubuntu 虚拟机桌面

因为宿主机是 Windows 平台，而且在后面仿真时使用的 ModelSim 也是 Windows 平台的，为了方便文件的传递，这里需要设置虚拟机与宿主机的文件共享。先关闭 Ubuntu 虚拟机，然后打开 VisualBox 中虚拟机的设置界面，选择“共享文件夹”，如下图所示。



点击界面右边的添加文件夹按钮，出现如下图所示界面：



在其中选择共享文件夹的路径，设置名称，参考如上设置，然后启动虚拟机，打开终端，输入命令：

```
sudo mount -t vboxsf asm_test /mnt/
```

该命令的作用是将共享文件夹挂载在/mnt/目录下，sudo 表示以 Root 用户身份执行该命令，终端会提示输入密码，Ubuntu 虚拟机默认 Root 用户的密码是 openrisc。这样就实现了虚拟机与宿主机的文件共享，对虚拟机而言共享文件放在/mnt/路径下，对宿主机而言共享文件放在上图所示的 E 盘 OpenMIPS_VHDL_study_v1.0/asm_test 文件夹下。

接下来要安装 GCC，将下载的 GCC 压缩包拷贝到 Ubuntu 的/opt 目录下，打开 Ubuntu 的终端，使用如下命令解压缩：

```
cd /opt
tar vfxj mips_linux_toolchain_bin-1.1.tar.bz2
```

在用户主目录下有一个 .bashrc 文件（注意：该文件是隐藏的），在此文件中加入 PATH 的设置如下：

```
export PATH="$PATH:/opt/mips-4.3/bin"
```

然后注销。重新登录，在终端中输入 mips-sde-elf，然后按两次 Tab 键，会列出安装的针对 MIPS 平台的所有编译工具，如下图所示，表示 MIPS 编译环境安装成功。

```
openmips@openmips-VM: /opt
openmips@openmips-VM:/opt$ mips-sde-elf-
mips-sde-elf-addr2line  mips-sde-elf-gcc          mips-sde-elf-objcopy
mips-sde-elf-ar         mips-sde-elf-gcc-4.3.2  mips-sde-elf-objdump
mips-sde-elf-as         mips-sde-elf-gcov      mips-sde-elf-ranlib
mips-sde-elf-c++        mips-sde-elf-gdb       mips-sde-elf-readelf
mips-sde-elf-c++filt    mips-sde-elf-gdbtui    mips-sde-elf-run
mips-sde-elf-conv       mips-sde-elf-gprof     mips-sde-elf-size
mips-sde-elf-cpp        mips-sde-elf-ld        mips-sde-elf-strings
mips-sde-elf-g++        mips-sde-elf-nm        mips-sde-elf-strip
openmips@openmips-VM:/opt$ mips-sde-elf-
```

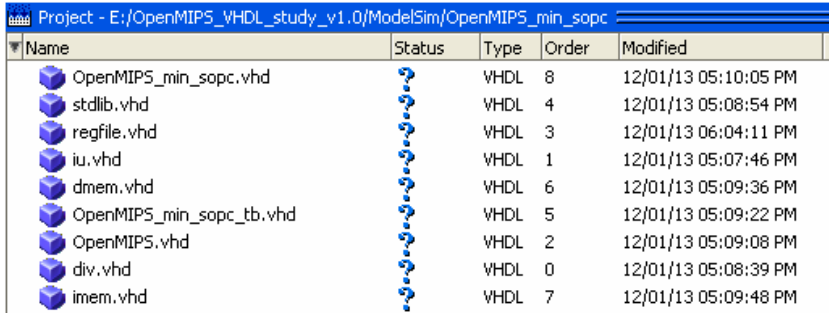
OpenMIPS 有两种测试方法：一种是直接使用完整的 OpenMIPS 进行测试，一种是使用《10 天实现处理器——OpenMIPS 成长记》提供的逐步完整的 OpenMIPS 进行测试，后者主要用在学习 OpenMIPS 设计时使用。下面分别介绍。

10.2 直接使用完整的 OpenMIPS 进行试验

共有三步：首先新建 ModelSim 工程，然后编译测试程序得到指令存储器初始化文件，最后使用该文件初始化指令存储器，在 ModelSim 中进行仿真。

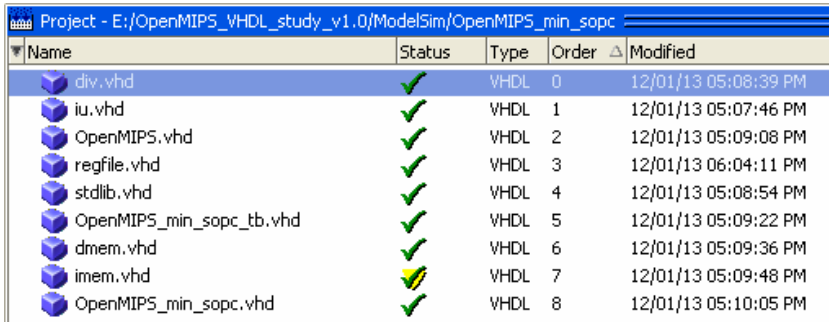
10.2.1 新建 ModelSim 工程

新建一个 ModelSim 工程，将 rtl、min_sopc、testbench 文件夹下的文件都添加到该工程中，如下图所示：



Name	Status	Type	Order	Modified
OpenMIPS_min_sopc.vhd		VHDL	8	12/01/13 05:10:05 PM
stdlib.vhd		VHDL	4	12/01/13 05:08:54 PM
regfile.vhd		VHDL	3	12/01/13 06:04:11 PM
iu.vhd		VHDL	1	12/01/13 05:07:46 PM
dmem.vhd		VHDL	6	12/01/13 05:09:36 PM
OpenMIPS_min_sopc_tb.vhd		VHDL	5	12/01/13 05:09:22 PM
OpenMIPS.vhd		VHDL	2	12/01/13 05:09:08 PM
div.vhd		VHDL	0	12/01/13 05:08:39 PM
imem.vhd		VHDL	7	12/01/13 05:09:48 PM

此时编译所有文件会提示出错，这是由于编译 Order 的原因，因为程序要求宏定义文件 stdlib.vhd 首先编译，而从上图可以知道 stdlib.vhd 的 Order 是 4，所以会出问题，有两种解决方法，第一种是再次编译所有文件，第二种是改变编译顺序，使得 stdlib.vhd 的 Order 为 0。编译成功后，所有文件的 status 一栏都是绿色的勾。



Name	Status	Type	Order	Modified
div.vhd	✓	VHDL	0	12/01/13 05:08:39 PM
iu.vhd	✓	VHDL	1	12/01/13 05:07:46 PM
OpenMIPS.vhd	✓	VHDL	2	12/01/13 05:09:08 PM
regfile.vhd	✓	VHDL	3	12/01/13 06:04:11 PM
stdlib.vhd	✓	VHDL	4	12/01/13 05:08:54 PM
OpenMIPS_min_sopc_tb.vhd	✓	VHDL	5	12/01/13 05:09:22 PM
dmem.vhd	✓	VHDL	6	12/01/13 05:09:36 PM
imem.vhd	✓	VHDL	7	12/01/13 05:09:48 PM
OpenMIPS_min_sopc.vhd	✓	VHDL	8	12/01/13 05:10:05 PM

10.2.2 编译测试程序

在 10.1 节测试环境中已经将 asm_test 文件与 Ubuntu 虚拟机共享了，挂载在/mnt 目录下。此时，在 Ubuntu 虚拟机中打开终端，进入 mnt 目录，可以使用 ls 命令列出所有的测试文件夹，如下：

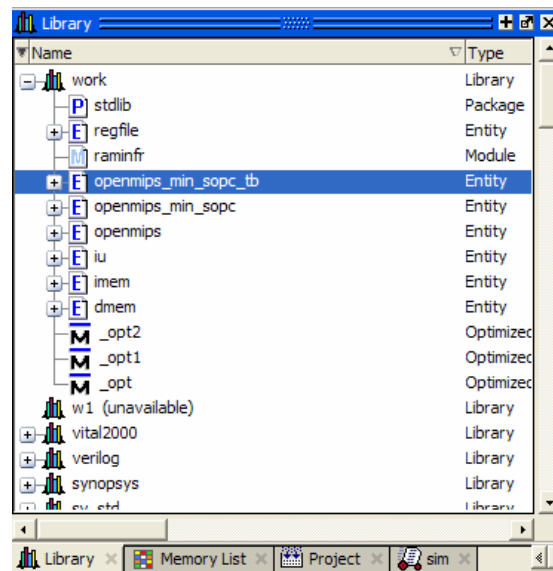
```
openrisc@openrisc-VirtualBox:/mnt$ ls
Day10_1 Day10_2 Day3_1 Day3_2 Day4_1 Day4_2 Day5_1 Day5_2 Day6_1 Day6_2 Day7_1 Day7_2 Day8_1 Day8_2 Day9_1 Day9_2
Day10_1 Day10_2 Day3_1 Day3_2 Day4_1 Day4_2 Day5_1 Day5_2 Day6_1 Day6_2 Day7_1 Day7_2 Day8_1 Day8_2 Day9_1 Day9_2
openrisc@openrisc-VirtualBox:/mnt$
```

进入其中任何一个文件夹，输入 make all，即可编译相应的测试例程，得到 inst_rom.data 文件。

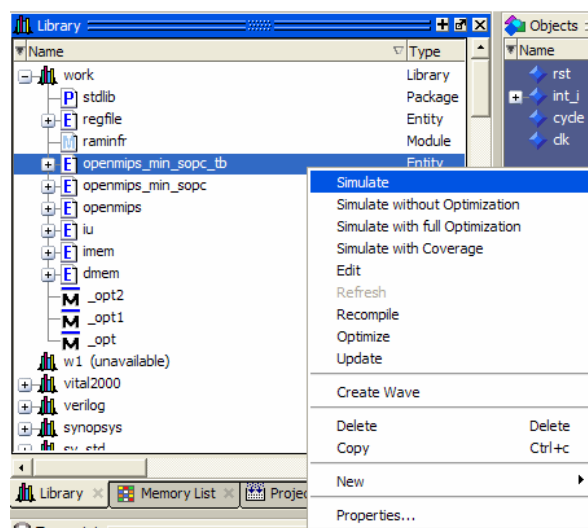
10.2.3 仿真

将上一步得到 inst_rom.data 文件拷贝到 10.2.1 节新建的 ModelSim 工程的目录下，该文件用来初始化指令存储器。

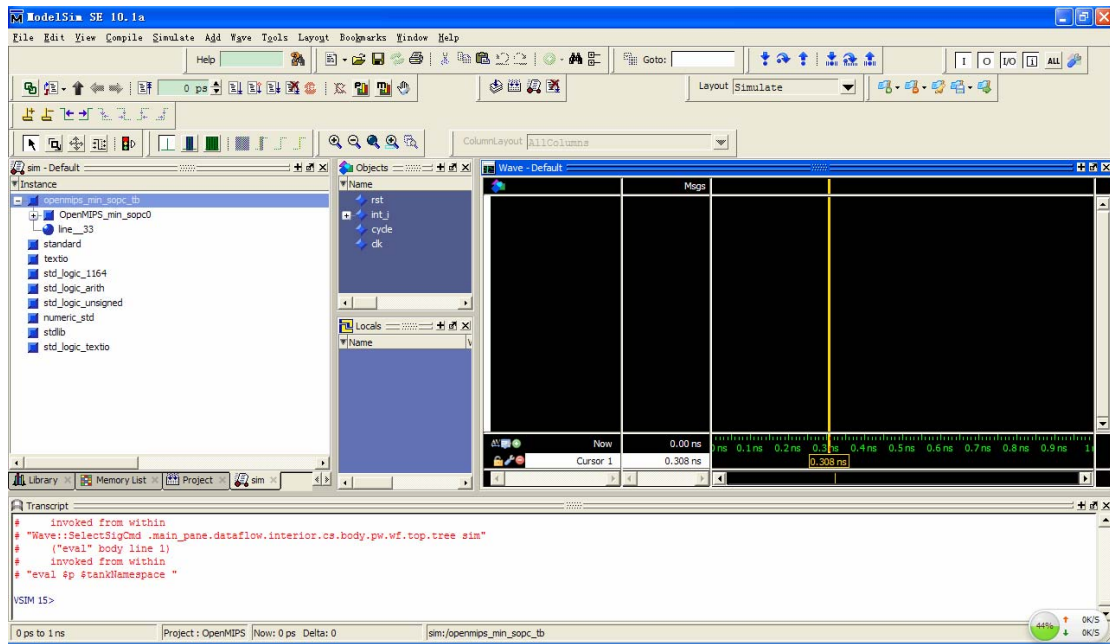
在 ModelSim 中，切换到 Library 这个 Tab 窗体，展开 work 如下：



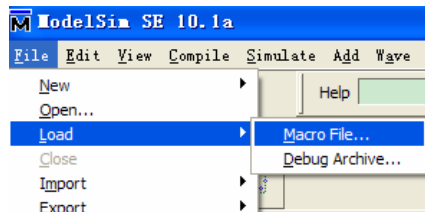
选择 openmips_min_sopc_tb，单击鼠标右键，选择 Simulate



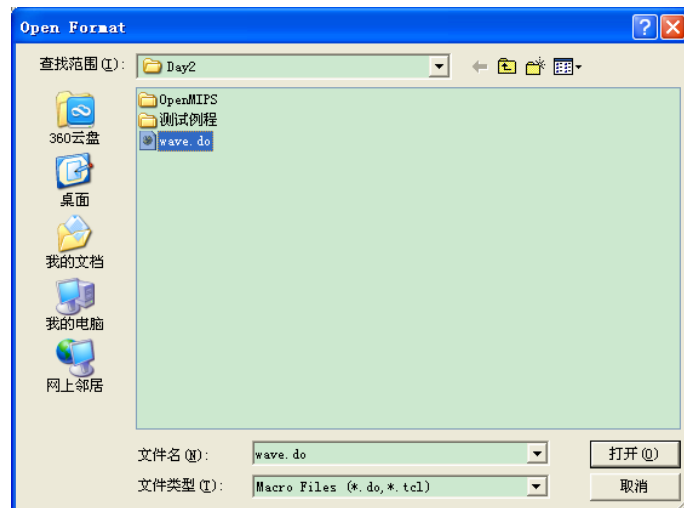
此时会出现波形显示窗口，如下：



鼠标单击黑色的波形显示窗口，然后选择 file -> load -> Macro File，如下：



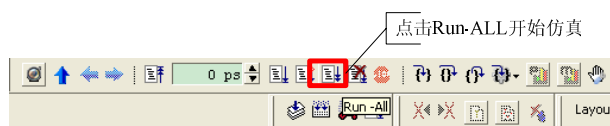
如果在 10.2.2 节中编译的程序所在的文件夹是 Day2，那么此时可以选择 10_Days_make_OpenMIPS 目录下的 Day2 文件夹下的 wave.do 文件，如下：



单击打开按钮，此时波形显示窗口就会出现一些要观察的信号，如下：



单击工具栏上的 run-all 按钮即可运行仿真，得到仿真结果。



wave.do 文件是一个事先保存的要观察信号的集合，通过观察这些信号就可以判断程序是否执行正确，当然，朋友们也可以添加自己感兴趣的信号进行观察，使用 wave.do 只是为了方便。

仿真的预期结果在程序的注释中都有说明，也可以参考文档《10 天实现处理器——OpenMIPS 成长记》，其中也有说明，并且有 ModelSim 仿真结果的截图。

10.3 按照 OpenMIPS 实现步骤进行试验

以《10 天实现处理器——OpenMIPS 成长记》中的第 6 天为例进行介绍。

10.3.1 新建 ModelSim 工程

与 10.2.1 节大致相同，唯一的区别是将 10_Days_make_OpenMIPS 目录下的 Day6 文件夹下的文件都添加到该工程中，而不是 rtl、testbench、min_socp 文件夹下的文件。

10.3.2 编译测试程序

编译步骤与 10.2.2 节相同，唯一区别是，在 10.2.2 节，可以进入任一目录进行编译，得到指令存储器初始化文件 inst_tom.data，而此时只能编译 Day6 目录中的测试例程。

10.3.3 仿真

与 10.2.3 节相同。