学 生 实 验 报 告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验三 构建基于SoCLib的单核SoC（硬件设计） | | | | |
| 班级 |  | 姓名 |  | 学号 |  |
| 实验时间 |  | 实验地点 |  | 指导教师 |  |

1. **实验目的**
2. 了解SoCLib平台工作原理。
3. 掌握如何利用SoCLib定义一个单核SoC硬件平台，如何在该SoC平台之上添加其他硬件设备。
4. 学习如何编写简单C程序以控制各种硬件设备。
5. 在所定义的单核SoC上运行串行的Motion-JPEG程序。。
6. **实验内容与步骤**
7. 首先，利用实验平台所提供的一个最基本的单核SoC熟悉SoCLib的工作原理。如图4所示，该单核SoC硬件平台由一个MIPS R3000处理器，一个存储器以及一个显示终端TTY组成。这些硬件设备通过片上网络——Generic Micronetwork（GMN）进行互连。



图4 基本单核SoC硬件平台

1. 该基本单核SoC基于SoCLib实现的顶层文件***top.cpp***位于***/projet\_soc/TP/TP0/HW***下。建议学生仔细阅读该顶层文件，了解如何通过该文件利用SoCLib所提供的各种ESL模型组件搭建SoC验证平台，如何定义声明各种设备模块以及各设备模块之间如何连接。
2. 在熟悉SoCLib工作原理和顶层文件***top.cpp***的组织形式后，在图5所示的单核SoC平台之上添加定时器VCI\_TIMER、文件系统VCI\_FDACCESS、帧缓存VCI\_FRAMEBUFFER以及同步锁VCI\_LOCKS等设备模块，修改后的单核SoC平台如图3所示。

图5 修改后的单核SoC硬件平台

基于SoCLib添加各个设备模块时，应重点检查以下几个方面：

* 1. 设备模块与各种信号的声明是否正确？
  2. 设备模块的初始化及各种参数的设置是否正确？
  3. 各设备模块与互连网络及模块间信号的连接是否正确？
  4. 各设备模块的内存映射（Memory Mapping）地址的设置是否正确？
  5. 与互连网络连接的主设备、从设备数目设置是否正确？

注意：使用SoCLib，每添加一个设备模块都会遇到上述问题，因此应该特别注意。

1. 编写简单的C程序，控制所添加的各种设备模块的功能，以验证SoC系统各设备工作的正确性。如何添加各设备模块以及如何使用请参见SoCLib主页中的相关信息：

VCI\_TIMER：http://www.soclib.fr/trac/dev/wiki/Component/VciMultiTimer

VCI\_FDACCESS：http://www.soclib.fr/trac/dev/wiki/Component/VciFdAccess

VCI\_FRAMEBUFFER：http://www.soclib.fr/trac/dev/wiki/Component/VciFrameBuffer

VCI\_LOCKS：http://www.soclib.fr/trac/dev/wiki/Component/VciLocks

1. 对位于***/projet\_soc/TP/TP0/SW/mjpeg\_seq***文件夹下的MJPEG串行程序使用MIPS交叉编译器进行编译，移植到图3所示的单核SoCLib平台之上，熟悉MJEPG的算法流程，思考对该算法进行并行化时，如何进行的软/硬件的改进。建议利用TIMER模块，统计MJPEG串行程序每部分在MIPS3000上运行所需要的时间。

**实验实现步骤**

// top.cpp添加头文件

#include "vci\_fd\_access.h"

#include "vci\_framebuffer.h"

#include "vci\_locks.h"

#include "vci\_timer.h"

// 设置帧缓存模块的相关宏定义，即framebuffer的高度和宽度

#define FBUFFER WIDTH 256

#define FBUFFER HEIGHT 144

//定义存储器RAM 相关区段及各种设备的寻址空间并加入mapping table中，可通过maptab.add（Segment（参数1，参数 2，参数3，参数4，参数5））语句进行添加。其中，参数1定义所添加区段或外设的唯一ID；参数2定义寻址空间的基地址；参数3定义寻址空间的大小；参数4定义设备索引；参数5定义Cache标志（该参数表示相关区段是否可缓存，有两个参数值可以配置，分别为 false 和 true。若设置为 true，则表示可以被Cache 缓存，将提高访问和运算速度。需要注意的是只有存储器RAM的相关区段才可以配置为true，其他设备只能配置为false。）本实验中，所需修改的mapping table如下：

maptab.add(Segment("reset", RESET\_BASE, RESET\_SIZE, IntTab(1), true));

maptab.add(Segment("excep", EXCEP\_BASE, EXCEP\_SIZE, IntTab(1), true));

maptab.add(Segment("text", TEXT\_BASE, TEXT\_SIZE, IntTab(1), true));

maptab.add(Segment("data", DATA\_BASE, DATA\_SIZE, IntTab(1), true));

maptab.add(Segment("semlocks\_seg", SEMLOCKS\_BASE, SEMLOCKS\_SIZE, IntTab(3),

false));

maptab.add(Segment("timer", TIMER BASE, TIMER SIZE, IntTab(4), false));

maptab.add(Segment("fd\_access", FD\_ACCESS BASE, FD ACCESs\_SIZE, IntTab(5),

false));

maptab.add(Segment("frame buffer", FBUFFER BASE, FBUFFERl SIZE, IntTab(6),

false));

// 修改互连结构参数

soclib::caba::VciVgmn<vci\_param> vgmn("vgmn", maptab, 3, 7, 2, 8);

// 添加设备模块

soclib::caba::VciLocks<vci\_param> semlocks("semlocks", IntTab(3), maptab)

soclib::caba::VciTimer<vci\_param> timer("timer", IntTab(4), maptab, 1);

soclib::caba::VciFdAccess<vci\_param> fd\_access("fdaccess", maptab, IntTab(2),

IntTab(5));

soclib::caba::VciFrameBuffer<vci\_param> fbuffer("fbuffer", IntTab(6), maptab,

FBUFFER MIDTH, FBUFFER HEITOHr);

// 添加相关singals

soclib::caba::VciSignals<vci\_param> signal\_vci\_semlocks("signal\_vci\_semlocks");

soclib::caba::VciSignals<vci\_param>

signal\_vci\_timer("signal\_vci\_timer");

sc\_signal<bool> signal\_timer\_it("signal\_timer\_it");

soclib::caba::VciSignals<vci\_param>

signal\_vci\_fd\_access("signal\_vci\_fd\_ access");

soclib::caba::VciSignals<vci\_param>

signal\_vci\_inv\_fd\_access("signal\_vci\_inv\_fd\_access");

sc signal<bool> signal\_fd \_access\_it("signal\_fd\_access\_it");

soclib::caba::VciSignals<vci\_param> signal\_vci\_fbuffer("signal\_vci\_fbuffer");

// 添加总线

vgmn.p\_to\_initiator[2](signal\_vci\_inv\_fd\_access);

vgmn.p\_to\_target[3](signal\_vci\_semlocks);

vgmn.p\_to\_target[4](signal\_vci\_timer);

vgmn.p\_to\_target[5](signal\_vci\_fd\_access);

vgmn.p\_to\_target[6](signal\_vci\_fbuffer);

semlocks.p\_clk(signal\_clk);

semlocks.p\_resetn(signal\_resetn);

semlocks.p\_vci(signal\_vci\_semlocks);

timer.p\_clk(signal\_clk);

timer.p\_resetn(signal\_resetn);

timer.p\_vci(signal\_vci\_timer);

timer.p\_irq[0](signal\_timer\_it);

fd\_access.p\_clk(signal\_clk);

fd\_access.p\_resetn(signal\_resetn);

fd\_access.p\_vci\_target(signal\_vci\_fd\_access);

fd\_access.p\_vci initiator(signal\_vci\_inv\_fd\_access);

fd\_access.p\_irq(signal\_fd\_access\_it);

fbuffer.p\_clk(signal\_clk);

fbuffer.p\_resetn(signal\_resetn);

fbuffer.p\_vci(signal\_vci\_fbuffer);

// 修改segmentation.h文件，添加4个设备模块的寻址空间基地址和大小：

#define SEMLOCKS BASE 0xc1000000

#define SEMLOCKS SIZE 0x00000400

#define TIMER BASE 0xC2000000

#define TIMER SIZE 0x00000100

#define FD ACCESS BASE 0xC3000000

#define FD ACCESS SIZE 0x00001000

#define FBUFFER BASE 0xC4000000

#define FBUFFER SIZE 0x01000000

// 修改platform\_desc文件，注册所添加的4个设备模块，添加如下4行：

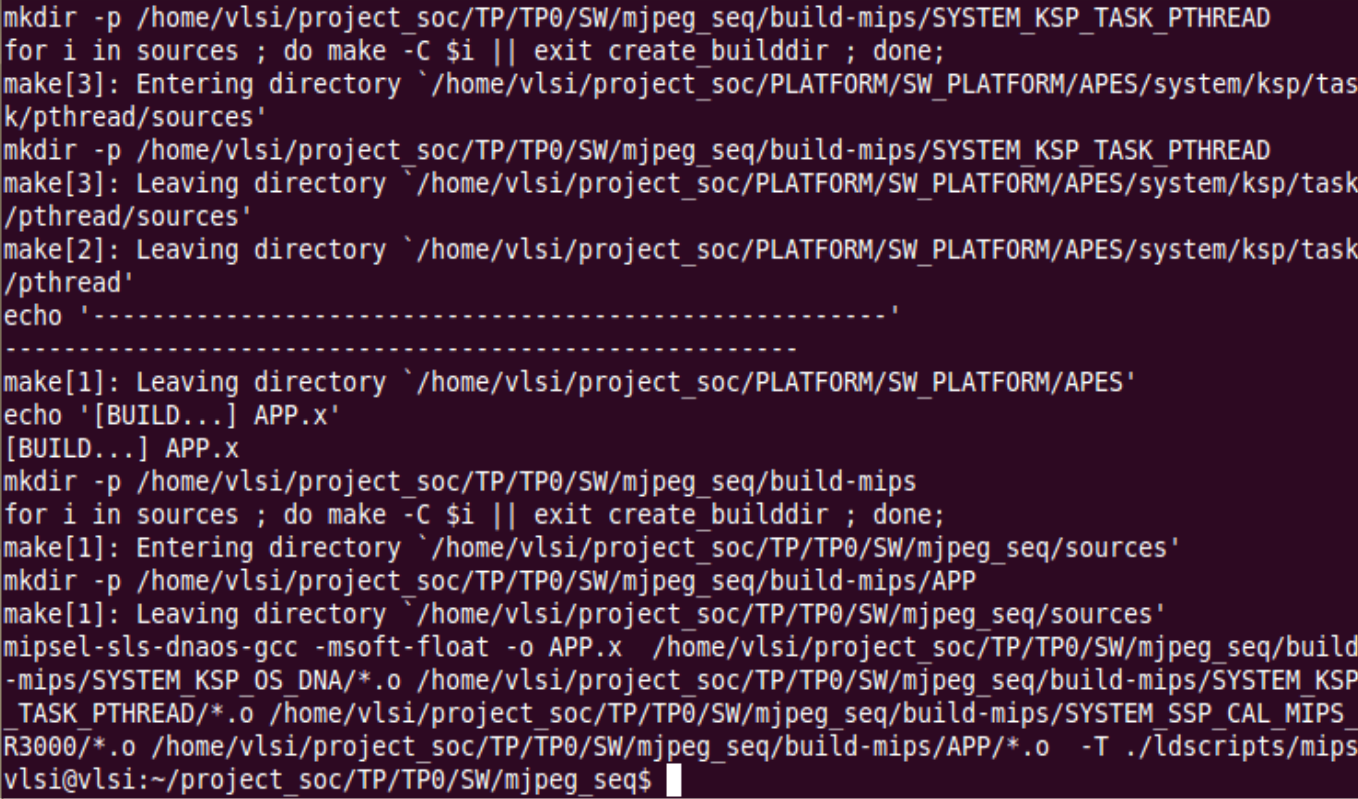
Uses('vci\_timer'),

Uses('vci\_fd access'),

Uses('vci\_framebuffer'),

Uses('vci\_locks'),

构建结果



**4、实验三评分及评语**

**#注 实验三与实验四，整体评分**