系统桥与计时器

系统桥 (bridge)

一个 MIPS 微系统不能仅由 CPU 构成，其还应该含有外部设备，这也就要求 CPU 有与不同外部设备进行沟通的能力。因此，我们需要实现与外部设备沟通的**系统桥 (bridge)**。在本次实验中，我们采用划分地址空间的方式来选择与哪个外设通信。下图是规定的地址空间设计，测试程序也将以此为根据编写。需要注意的是，本 project 与《See MIPS Run Linux》和 PPT 中给出的 MIPS 系统地址范围是不同的，而与 MARS 相同，这主要是为了能够让你能更好的验证设计。

	地址或地址范围	备注
数据存储器	0x0000_0000 至 0x0000_2FFF	
指令存储器	0x0000_3000 至 0x0000_6FFF	
PC 初始值	0x0000_3000	
Exception Handler 入口地址	0x0000_4180	
计时器 0 寄存器地址	0x0000_7F00 至 0x0000_7F0B	计时器 0 的 3 个寄存器
计时器 1 寄存器地址	0x0000_7F10 至 0x0000_7F1B	计时器 1 的 3 个寄存器

最后两行是为我们此次实验的外部设备之一，计时器 (timer) 准备的。本页面的第二小节对计时器进行了详细介绍。

在 P7 实验中有两类外部设备，一类是存储器，一类是计时器。两类外设都通过系统桥模块与 CPU 相连。我们可以把系统桥理解为一个用于区分地址的组合逻辑模块，当读写计时器相关地址时，CPU 通过系统桥与我们自己实例化的两个 timer 进行数据交互；当需要读写数据存储器时，CPU 通过系统桥将相关信号上传至顶层模块（`mips.v`），与官方 tb 中实现的存储器进行数据交互。

关于系统桥的具体编写，请大家参考该文件 L15-支持 IO.pdf。

> **独立的系统桥**
>
> 在实现系统桥时，其必须作为独立 module 来实现，不能包含在 CPU 内部。

计时器 (timer)

在 P7 这个简单的 MIPS 微系统中，计时器是一种外部设备，其主要功能就是**根据设定的时间来定时产生中断信号**，是我们系统的中断来源。

怎样使外设与 CPU 进行沟通呢？采用划分地址空间的办法后，与外设沟通和与 DM 沟通的方式类似，通过一个 CPU 视图下的内存地址，读写相应数据达到与外设沟通的目的。而这个所谓的内存，在外设中，实际上只是若干寄存器。系统桥传入对地址的访问请求后，我们通过计时器内部的转换代码，将请求转变为对相应寄存器的读写操作。

除了使用访存请求来沟通外，CPU 还需要响应外设和 Testbench 传入的中断信号，并执行对应的中断处理程序。

往届同学在完成 P7 时，教程通过文字叙述与转移图等尽可能解释计时器行为规范，但依旧存在许多争议点，因此最终实现的计时器在功能与操作上存在细微的差异。因此在今年的教程中，我们决定向同学们提供已实现的计时器 Verilog 源代码： 源代码。 timer 内部需要定义多个程序员可见寄存器，如 **CTRL**、**PRESET** 等，也需要定义若干用于完成功能的内部寄存器(程序员不可见)，详情请参考设计文档：CO 定时器设计规范-1.0.0.4.pdf

同学们需要在阅读源代码后，完成以下内容并以 PDF 文档的形式进行提交：

1. 请用**状态转移图**的方式分别描述两种模式下计时器状态和状态转移的条件。有关状态转移图的规范和绘制工具，可参考 ProcessOn 或 diagrams.net。
2. 编写**计时器使用说明**，分别阐述不同计时器模式中、不同计时器状态下，用户对计时器的操作规范(即针对计时器寄存器的操作)，规范中应包括可行的操作与对应功能、不可行操作与误操作后果等等。

> **思考题**
>
> 1. BE 部件对所有的外设都是必要的吗?
> 2. 请阅读官方提供的定时器源代码，阐述两种中断模式的异同，并分别针对每一种模式绘制状态移图。