

外设的实现

计时器

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之一。

在今年的教程中，我们向同学们提供已实现的计时器 [Verilog 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](#)。

中断发生器

这是课程组抽象简化现实中外设后得到的一种外设，会在不确定的时刻产生一个中断信号（就好像电脑并不知道谁会在什么时候敲击键盘一样），并持续置高。直到微系统做出响应，才变回低位。

对中断发生器的响应是通过系统桥来实现的，通过 `store` 类指令访问地址 `0x7F20`，就可以达到响应中断的目的。

中断发生器的实现并不需要同学们来完成，不同的中断发生器（中断信号产生的规则不一样）都是在测试的 tb 上实现的。同学们只需要确保自己的 P7 微系统，具有以下两个能力，就可以满足这个方面的测试：

- 微系统可以通过外部端口接受外部中断信号（在计时器部分已经实现了）。
- 微系统可以通过访问地址 `0x7F20` 的 `store` 类指令，改变对应的微系统输出信号（`m_int_addr`，`m_int_byteen`），即系统桥实现正确。


系统桥

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，读写相应数据即可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，我们通过系统桥内部的转换代码，将请求转变为对相应寄存器的读写操作。


下表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，P7 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。

条目	地址或地址范围	备注
数据存储器	0x0000_0000 ~ 0x0000_2FFF	
指令存储器	0x0000_3000 ~ 0x0000_6FFF	
PC 初始值	0x0000_3000	
异常处理程序入口地址	0x0000_4180	
定时器 0 寄存器地址	0x0000_7F00 ~ 0x0000_7F0B	定时器 0 的 3 个寄存器
定时器 1 寄存器地址	0x0000_7F10 ~ 0x0000_7F1B	定时器 1 的 3 个寄存器
中断发生器响应地址	0x0000_7F20 ~ 0x0000_7F23	

注意实现系统桥时，其必须作为独立 module 来实现，不能包含在 CPU 内部。关于系统桥的具体编写，请大家参考该文件 [L15-支持 IO.pdf](#)。

 **思考题**

为何与外设通信需要 Bridge?

 **思考题**

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。