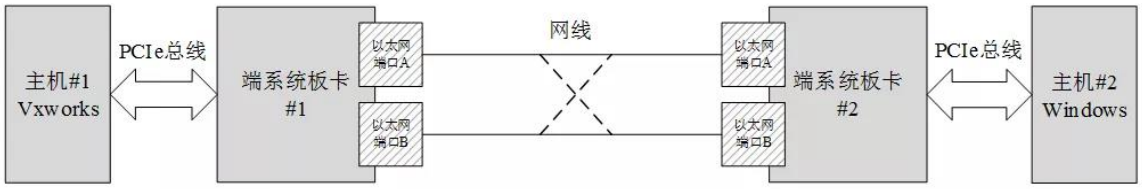
实测 VxWorks 响应 PCIe 中断的最小时间间隔

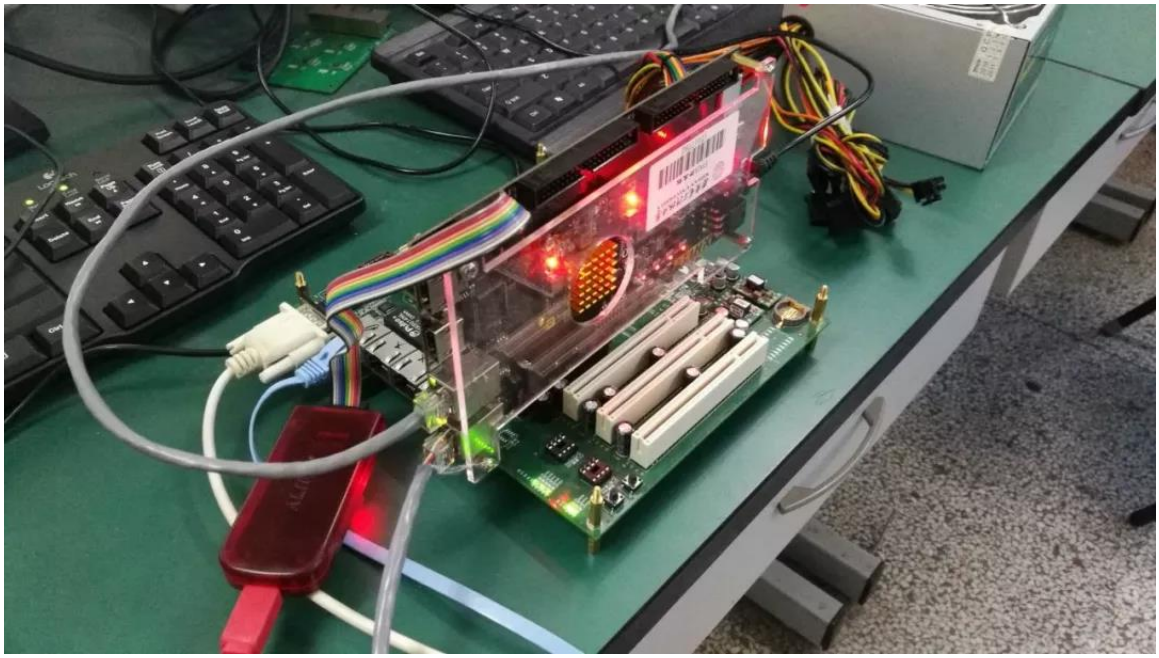
准备工作

硬件平台环境如下图所示，采用两台带有以太网口的设备相连，一端是 PC 机插有 PCIe 的 FPGA 开发板，运行 Windows 操作系统；另一端是嵌入式设备，运行 VxWorks 操作系统。



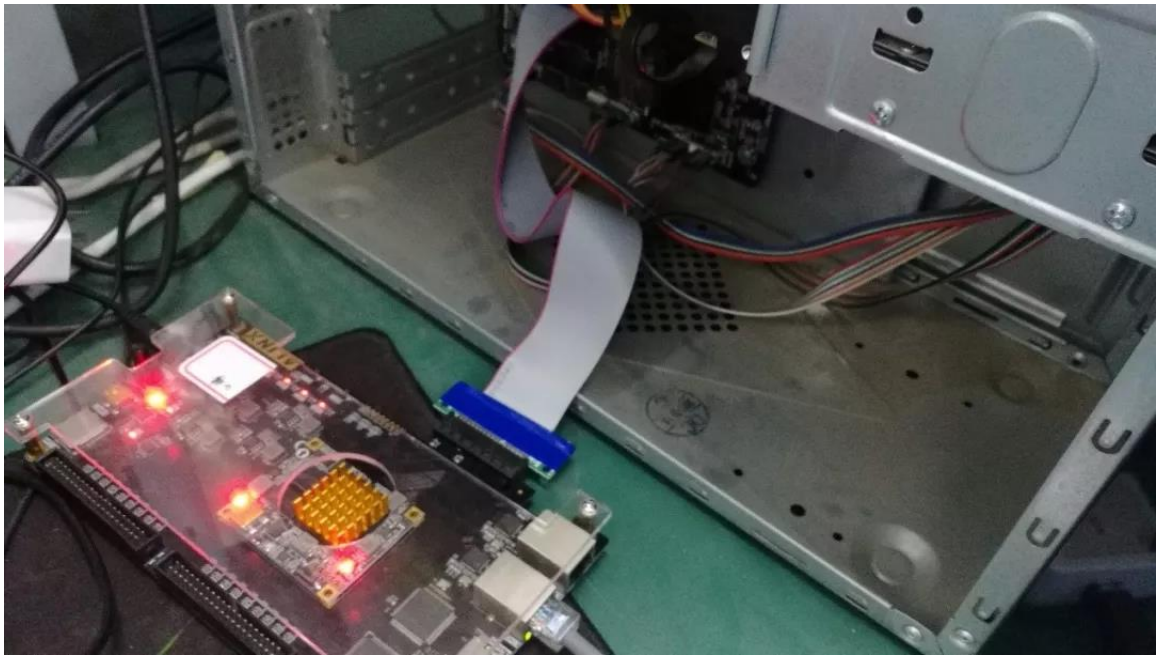
1、嵌入式设备

母板为 P2020 开发板，PCIe 板卡为黑金 Xilinx Artix-7 PCIE AX7103 FPGA 开发板，运行 VxWorks 操作系统。



2、PC 端

电脑主机一台，拆开（机箱比较脏，见谅），通过 PCIe 连线连到黑金 Xilinx Artix-7 PCIE AX7103 FPGA 开发板上，运行 Win7 操作系统。

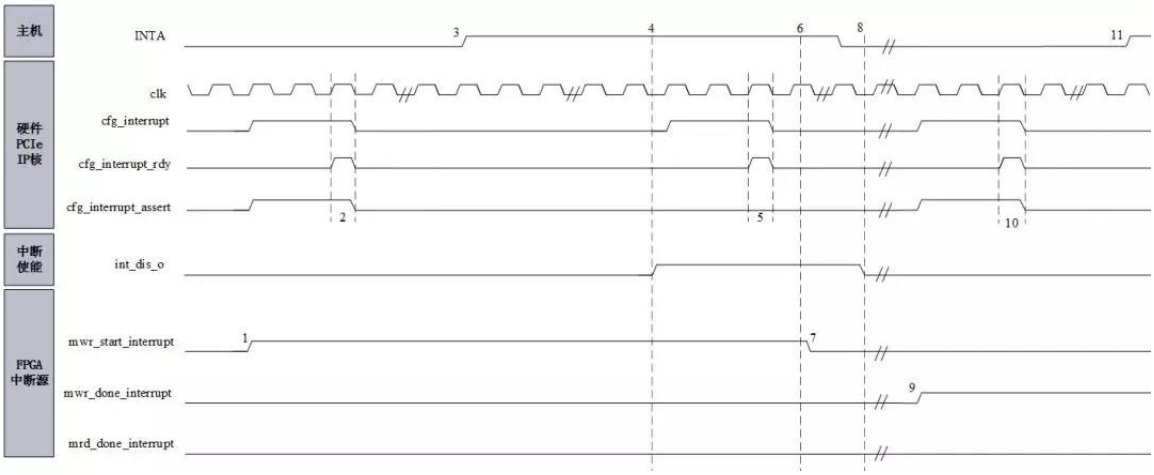


两台设备之间通过双冗余的网线连接。

中断处理流程

在上面的环境中，按照以太网帧传递过程中的需求，任何一端的中断处理都包含三个不同的主体，首先是 CPU 内核的中断响应机制，然后是加上操作系统之后对中断响应的处理又有操作系统的要求，之后是 PCIe 硬件设备也有一套向 CPU 操作系统发送中断的规范。任何一方的中断处理机制都可以写很长很长的文字去描述，本文在此不再赘述。

PCIe 总线支持两种中断方式，传统的 INTx 中断和基于存储器写请求的中断请求机制即消息中断。本文的设计方案中使用的是传统的 INTx 中断。为了叙述上的方便，我们从 FPGA 的时序图的角度去描述中断的处理流程，具体分为主机（PCIe 发给主机的中断信号）、PCIe 硬核、驱动来配置的中断使能信号、FPGA 侧的中断源。下图是具体的主机操作系统为 VxWorks 时 FPGA 开发板与主机的中断交互流程。



1）FPGA 侧有三个中断源可以触发中断，分别是 DMA 写开始、DMA 写完成和 DMA 读完成中断，其中，写开始中断源是 FPGA 告知主机此时有数据要通过 DMA 写操作进行上传；写完成中断是 FPGA 将所有的数据封装成 DMA 写请求包；读完成中断是 FPGA 收齐了所有来自主机的 DMA 读完成包。上图中“1”处是中断源 mwr_start_interrupt 拉高了。

2）任意一个中断源拉高，FPGA 侧给 PCIe IP 核配置“置中断”时序，在 cfg_interrupt 和 cfg_interrupt_rdy 握手成功后，cfg_interrupt_assert 为高则为置中断。（cfg_interrupt 为 PCIe 硬核发给主机的中断请求，cfg_interrupt_rdy 为主机接收到中断请求后的回应，此时需要看 cfg_interrupt_assert 的状态，cfg_interrupt_assert 为高，则为置中断，如上图中“2”处所示；cfg_interrupt_assert 为低电平，则为清中断请求，如上图中“5”处所示。）

3）“置中断”后一段时间（此处约为 17 个时钟），主机侧硬中断电平 INTA 拉高，**此时才是 FPGA 板卡真正的向主机发出了一个中断。**如上图中“3”。

4）驱动检测到中断电平拉高后，以 PIO 写操作的方式往 PCIe 的 BAR 空间中控制状态寄存器 04H 的第[31]位写 1，关闭接收中断功能，此时中断使能信号线 int_dis_o 拉高，如上图中“4”位置。**int_dis_o 为高电平期间，CPU 不再响应 FPGA 板卡的中断请求，此处非常重要**。之后 CPU 则以 PIO 读的形式读 FPGA 的中断状态寄存器。

5）FPGA 将中断状态寄存器的值以 PIO 读完成包形式发送给 CPU，告知 CPU 该中断具体为何种中断，同时配置“清中断”时序。如上图中“5”处所示。

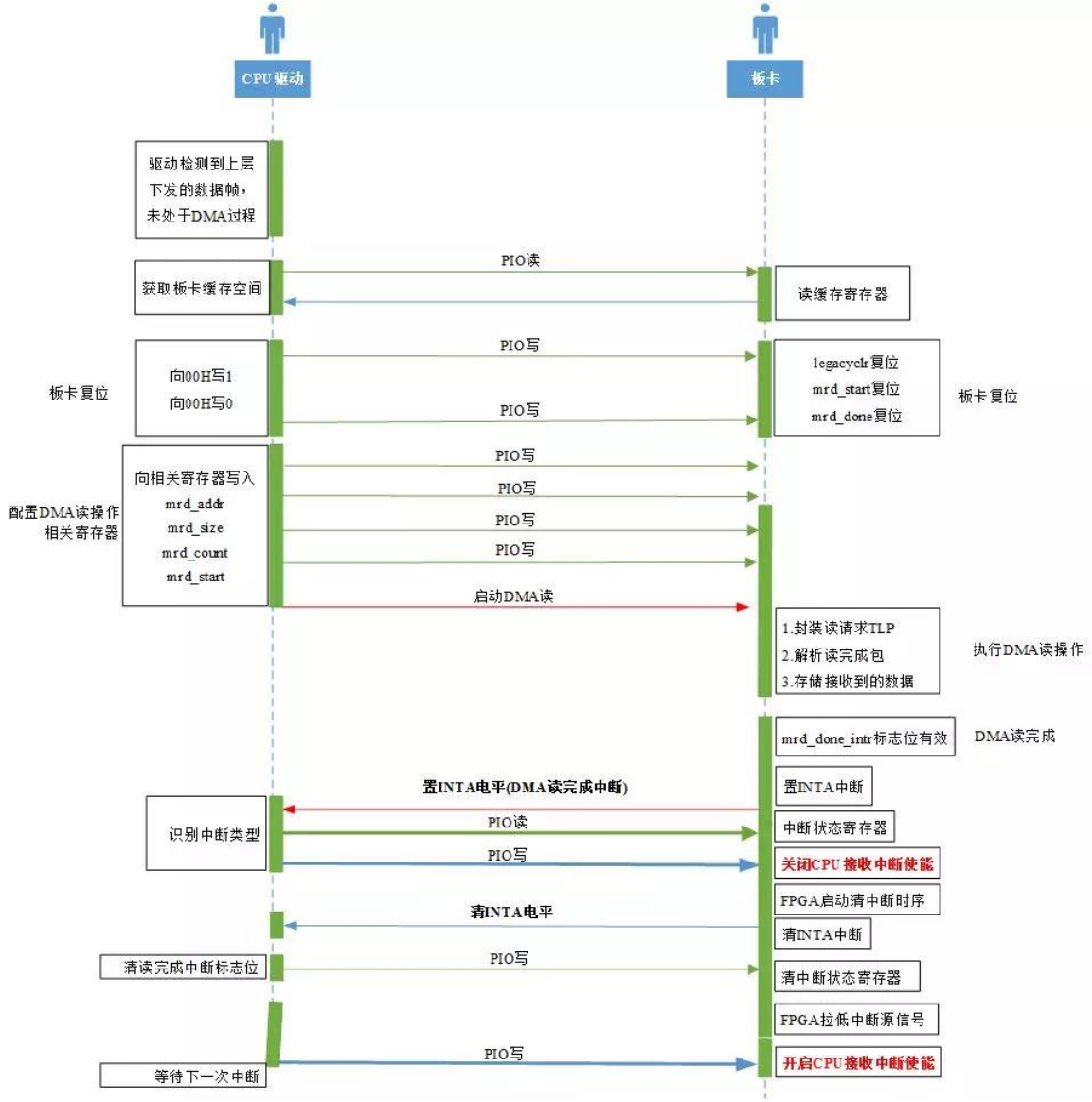
6）CPU 驱动记录中断源后复位相应中断标志位，如上图中“6”处所示。（此处也可由 FPGA 自己完成）

7）FPGA 拉低相应中断源信号，如上图中“7”处所示。

8）CPU 驱动通过 PIO 写操作往控制状态寄存器 04H 第[31]位写 0，**重新开启接收中断功能。**如上图中“8”处所示。

9）重复步骤 1）启动下一次中断；10）下一次置中断时序；11）硬中断电平再次拉高。

下图为一次完整的 DMA 读操作时 CPU 与 FPGA 板卡之间的交互流程，最后会涉及到 DMA 读完成中断，详细过程的描述略。

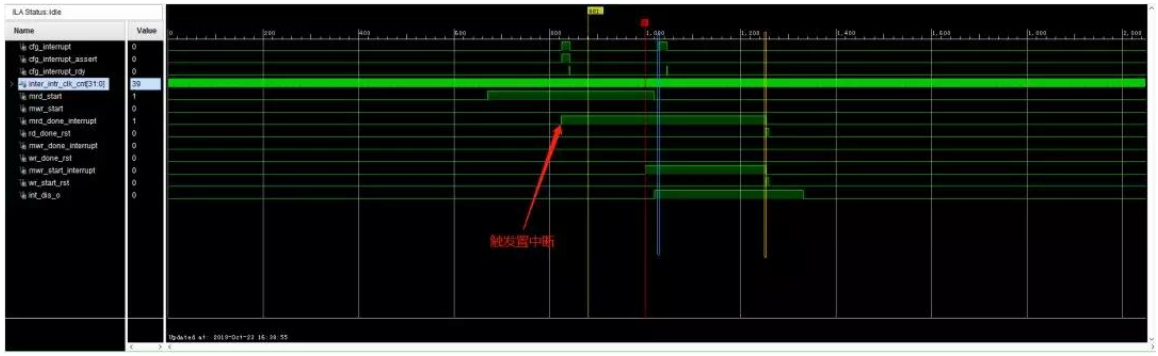


VxWork 响应 PCIe 中断的最小间隔

为了得到 VxWorks 响应 PCIe 中断的最小间隔，我们在 FPGA 侧对两次“置中断”间隔，即上图步骤 2）与步骤 10）进行了时钟计数，在“置中断”时序(cfg_interrupt_rdy & cfg_interrupt_assert)下将间隔时间寄存器 inter_intr_clk_cnt[31:0]计数复位，否则计数加一，直到下一次“置中断”进行计数复位，这样就能计算出中断信号两次拉高的时间间隔。

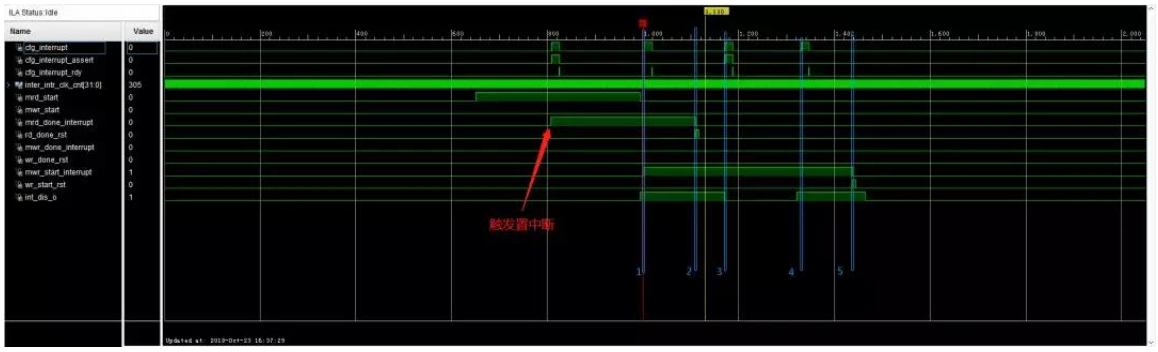
在测试的过程中，我们用 Vivado 抓取了实际数据传输时两种不同的中断场景。

1、场景 1：写开始中断和读完成中断一起处理



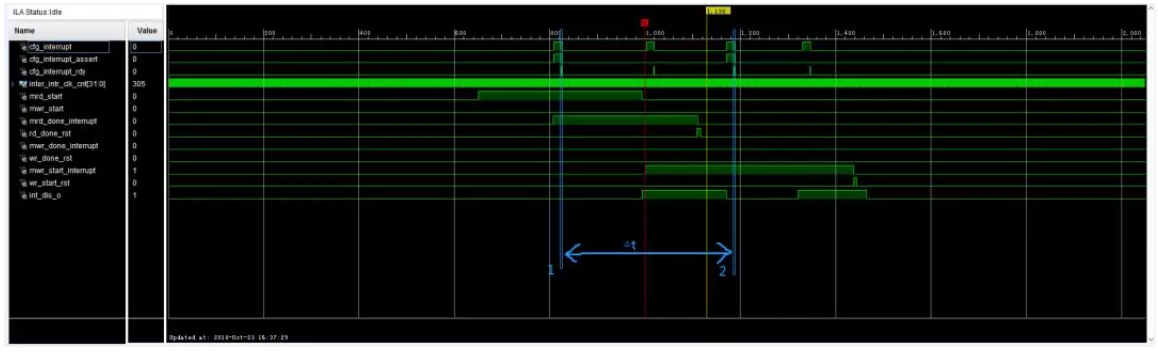
有了上面中断处理流程的介绍，就可以很方便的分析具体工作状态下的波形图。从上图可以看到，读完成中断 mrd_done_interrupt 触发置中断时序，主机的硬中断电平拉高，驱动往控制与状态寄存器 04H 的最高位（图示 int_dis_o 信号） PIO 操作写“1”，关闭中断功能，此时硬件这边不再产生置中断时序，直到驱动跳出中断复位程序，往 04H 的 int_dis_o 写“0”，使能中断；驱动 PIO 读中断状态寄存器（图示蓝线）“采样”到读完成（图示 mrd_done_interrupt 信号）和写开始（图示 mwr_start_interrupt 信号）两个中断标志位为高，此时，驱动会记录下来并同时对这两个中断标志位进行复位操作，然后驱动分别执行读完成中断和写开始中断状态机。

场景 2：写开始中断和读完成中断先后处理

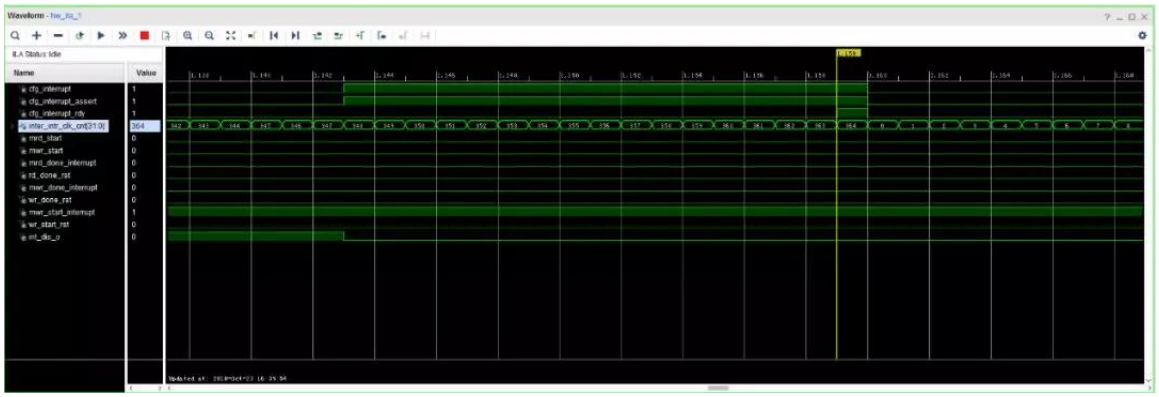


从上图可以看到，读完成中断 mrd_done_interrupt 触发置中断时序，主机的硬中断电平拉高，驱动往控制与状态寄存器 04H 的最高位（图示 int_dis_o 信号） PIO 操作写“1”，关闭中断功能，此时硬件这边不在产生置中断时序，直到驱动跳出中断复位程序，往 04H 的 int_dis_o 写“0”，使能中断；驱动 PIO 读中断状态寄存器（图示蓝线 1）采样到读完成（图示 mrd_done_interrupt 信号）中断标志位为 1，硬件产生清中断时序，将主机侧的硬中断电平拉低，注意，此刻写开始中断（图示 mwr_start_interrupt 信号）刚好拉高，驱动只记录读完成中断并对读完成中断标志位进行复位操作（图示蓝线 2），然后驱动执行读完成中断状态机，驱动跳出读完成中断状态机后重新使能中断（图示蓝线 3），此时硬件侧因为写开始中断才被允许产生置中断时序，驱动再次检测到硬中断电平信号为高，驱动 PIO 读中断状态寄存器（图示蓝线 4）采样到写开始中断标志位为 1，硬件产生清中断时序，将主机侧的硬中断电平拉低，驱动记录写开始中断并对写开始中断标志位进行复位操作（图示蓝线 5），然后驱动执行写开始中断状态机。

在第二个测试场景中，我们可以通过计数得知两个相邻中断的最小时间间隔，，硬件侧产生第一次中断段时序（图示蓝线 1），在执行完第一次中断后，驱动侧将 int_dis_o 拉低，重新使能中断，硬件侧立即产生置中断时序进行第二次中断操作（图示蓝线 2），如下图所示：



我们将图示蓝线 2 处进行放大得到下图：



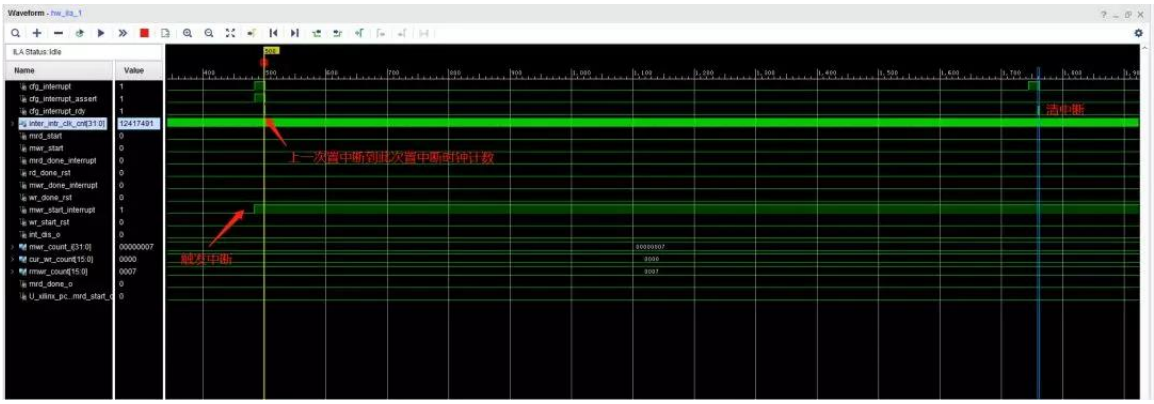
通过相邻中断时钟计数信号 inter_intr_clk_cnt[31:0]可以知道相邻两中断的最小间隔是 365 个钟，后边测试过多次，测试结果有 368，364，，我们取 365，时钟周期为 16ns，**由此可以计算得知 VxWorks 下最小中断间隔是 365*16=5.84us**。

结论：VxWorks 操作系统中断处理的最小时间间隔确实是传说中的微秒级！

Windows 操作系统下 PCIe 中断响应间隔测试

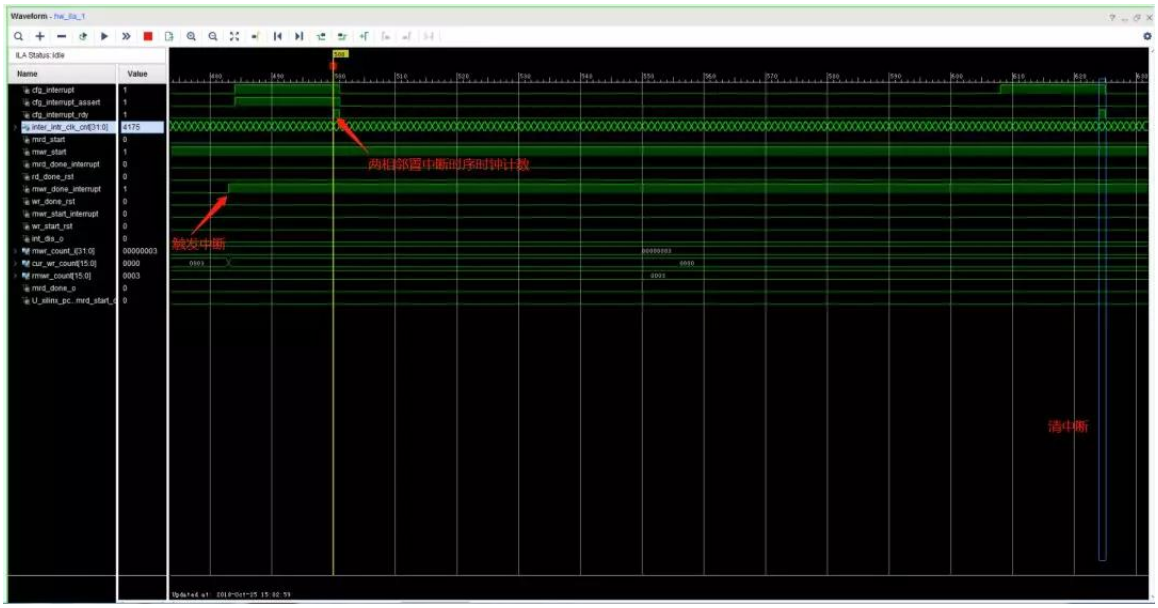
出于好奇，我们也尝试测了一下 Windows 操作系统下 PCIe 中断响应的时间间隔。在 Windows 平台下的驱动暂未使用开/关中断使能的功能，所以只是测试在点播视频以及拷贝视频文件两种场景下的中断间隔。

1、场景 1：点播视频，速率为 10Mbps 左右



从上图可以看到，上一次置中断时序复位后计数 12417491 个 clk（16ns）再次产生置中断时序，此时中断间隔约为 198.7ms，后面统计到一些计数值：19026416（304.4ms），6486433（103.8ms），9981793（159.7ms）。在点播视频时，带宽并未达到上限，驱动处理两个相邻中断的时间间隔＞100ms。为了在高带宽情况下测试，我们进行了场景 2 的测试。

场景 2：拷贝视频，速率为几百兆 bps



从上图可以看到，上一次置中断时序复位后计数 4175 个 clk（16ns）再次产生置中断时序，此时中断间隔约为 66.8us，后面统计到一些计数值：3595（57.5us）、7456（119.3us）、3582（57.3us）、4159（66.5us）。在带宽提升后，win32 驱动处理中断的频率有了显著地提高。

遇到的问题

在刚开始的时候，中断处理流程中 CPU 操作时并没有开启或关闭接收中断的操作，结果，在 Windows 平台下，没有任何的问题，生成 PCIe IP 核时，设置传输带宽上限为 2Gbps，在 1G 以太网口测试各种业务时都很稳定，没有出现操作系统崩溃的情况；但在 VxWorks 系统测试时，由于 VxWorks 系统实时性非常好，响应中断也比较及时，就会出现操作系统正在执行一个中断服务程序时，硬件又来了一个中断，直接导致 VxWorks 系统死掉，如下图所示。当然，这也是中断处理流程不规范造成的。后续会在 Windows 驱动中也添加上开关中断使能的步骤，测试一下 Windows 相应 PCIe 中断的最小间隔。不过从目前测试数据看，Windows 相应 PCIe 中断的速度肯定会比 VxWorks 慢很多。