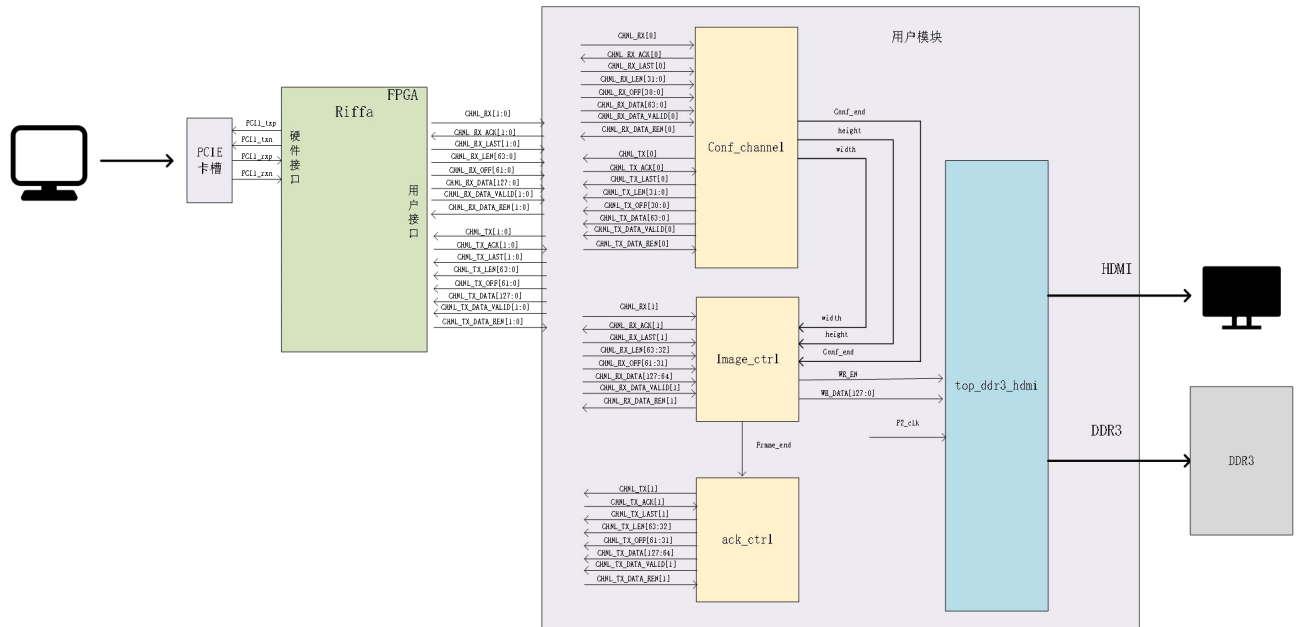


项目：基于 Riffa 框架的 PCIE 2.0 x2 的桌面共享

1.项目框架



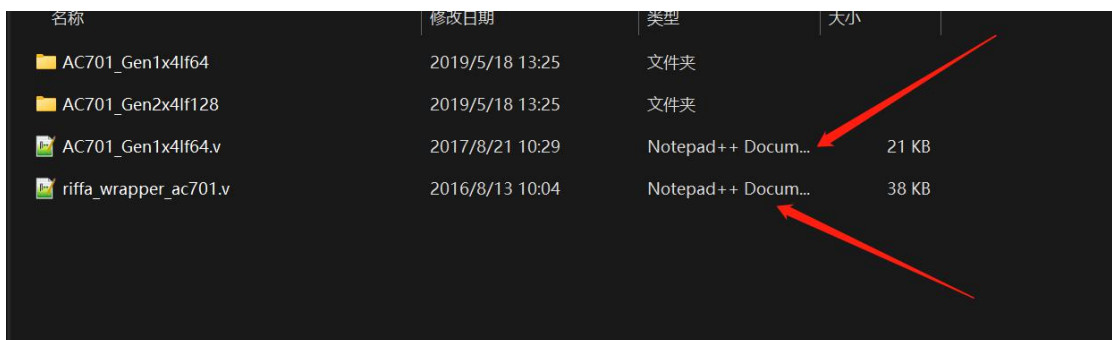
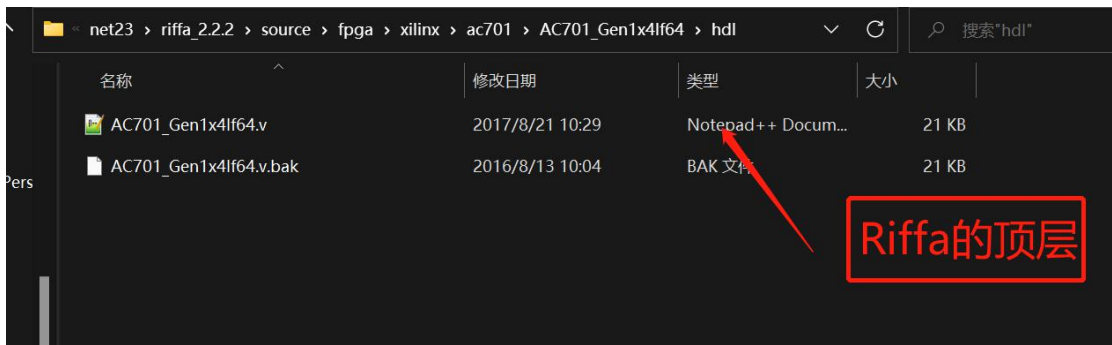
2.项目流程

1) 移植 Riffa 到自己的项目中

Riffa 源码有这么几个部分

名称	修改日期	类型	大小
documentation	2019/5/18 13:25	文件夹	
install	2019/5/18 13:25	文件夹	
source	2019/5/18 13:25	文件夹	
README.txt	2016/8/13 10:05	文本文档	7 KB

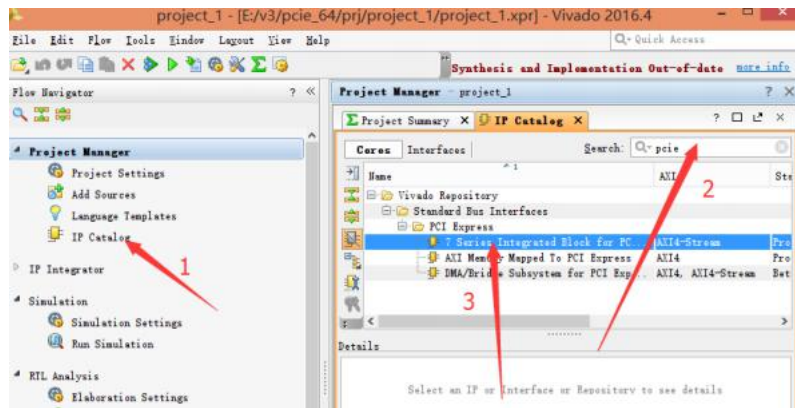
我们需要的是以下 2 个部分



将这两个部分的文件复制到自己的工程 design 中

2) 生成 PCIE 的 IP(重要!!!)

双击 IP Catalog, 搜索 PCIe, 双击 7 Series Integrated Block for PCI Express



首先在 model 里选择 ADVANCED 模式，这样用户就有更多选项可选择。

在定义选项中我们不用以下选项：

LinkRegisters, PowerMangement, Ext.Capabilities, Ext.Capabilities2, T L Settings and DL/PL Settings.

设备端口类型选择 PCI Express 端点设备

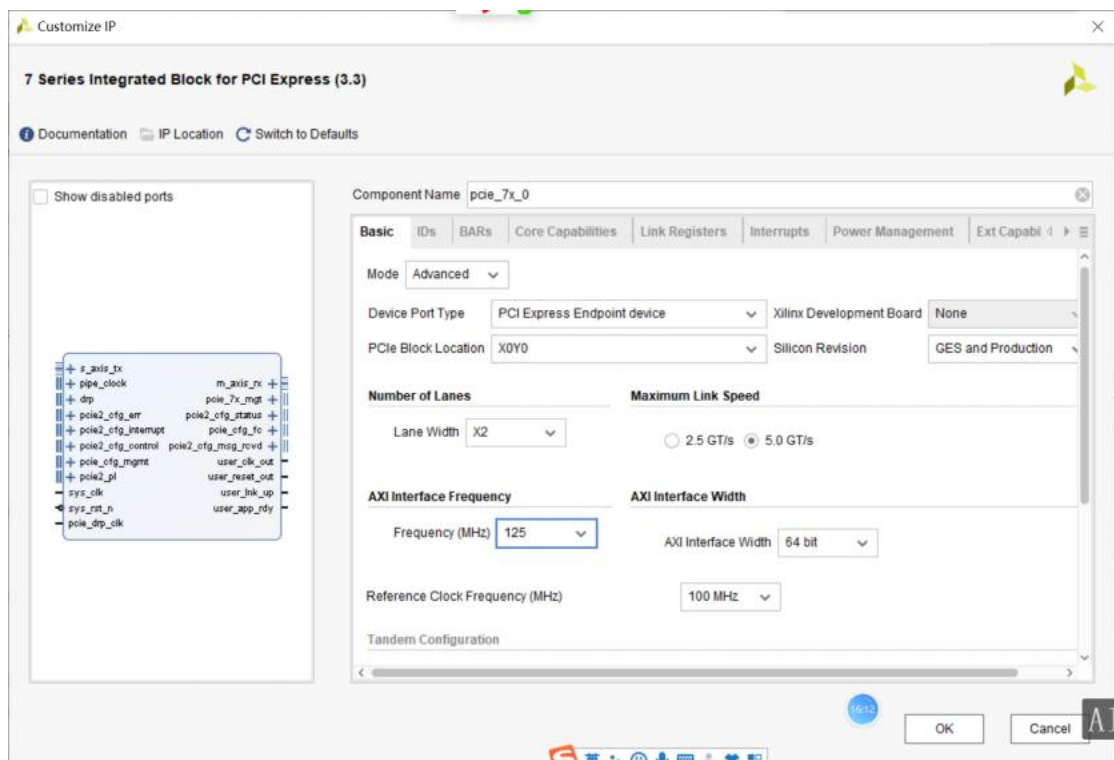
线宽选择 x2

速率选择 5.0Gt/s 即 PCIE2.0

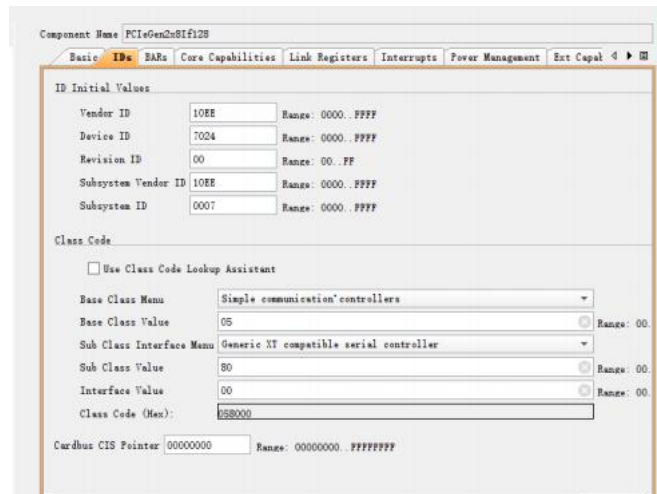
axi 时钟选择 62.5，也可以选择更高时钟

数据位宽选择 64

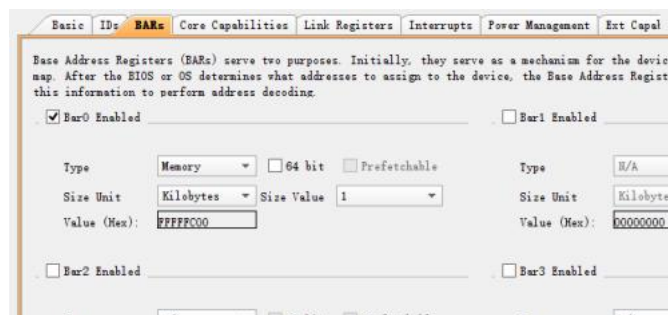
PIPE 模式仿真选择 none, 我们没有仿真模型，不能进行仿真



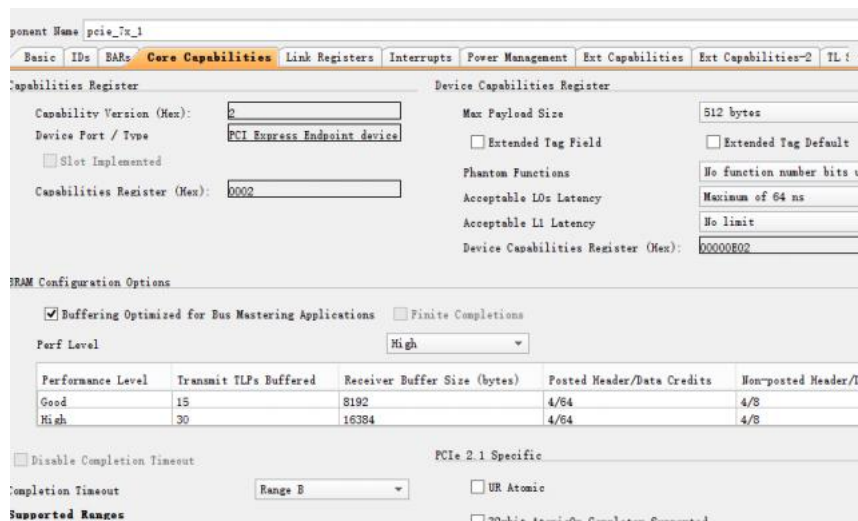
设置设备 ID 有助于识别不同的 FPGA 在多 FPGA 系统中，可以设置也可以默认不修改。其他选项，特别是供应商 ID，必须保持不变。



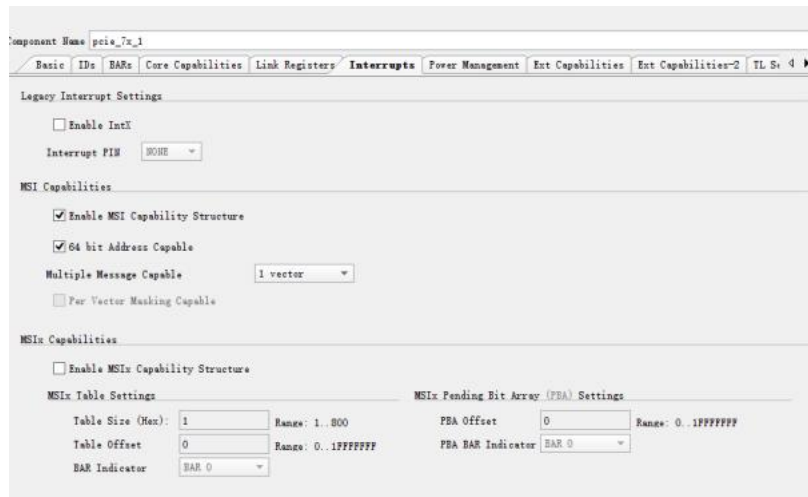
必须配置下图选项，让 BAR0 启用。将类型设置为 Memory 和单位 KiloBytes，并从下拉菜单中将“大小”值设置为 1。如果这些值没设置正确，RIFFA 的驱动将不能识别 FPGA 器件



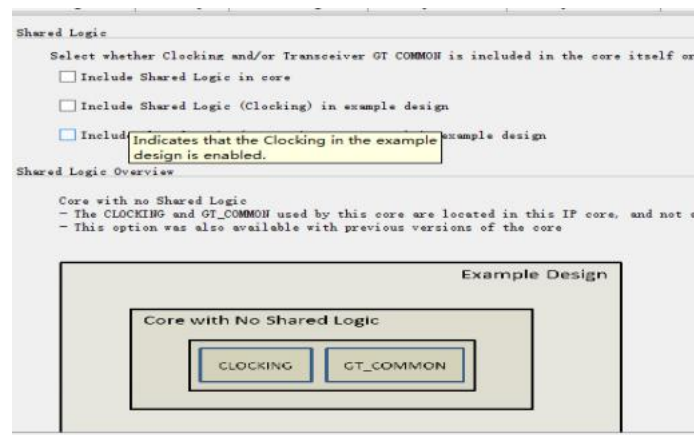
在此选项卡中选择“优化总线主控应用程序”和“扩展标签字段”。从下拉列表中选择最大有效载荷大小，用于设置 RIFFA C MAX PAYLOAD BYTES 参数。



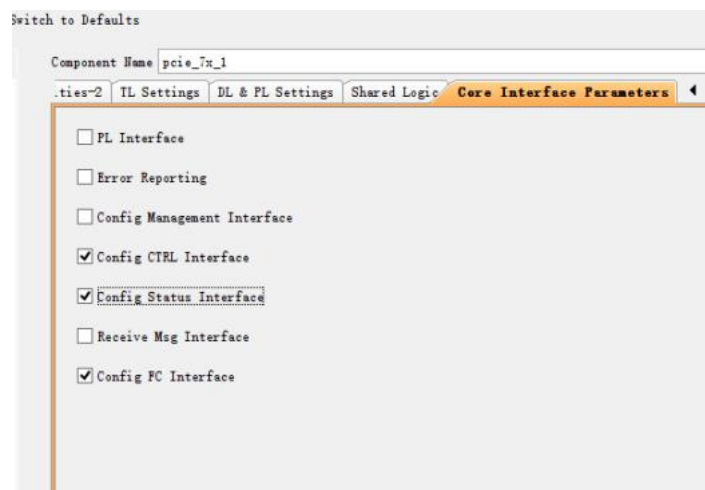
在下图所示的中断选项卡中，清除 Enable INTx（禁用）的复选框的 INTx，即不用传统带中断管脚的中断模式。选择消息中断，64 位地址信息。



在下图所示的共享逻辑选项卡中，清除显示的所有复选框。 这些设置不会影响生成的核心，但会影响 Vivado 生成的示例设计。生成的示例设计将为 RIFFA 提供一个镜像设计。

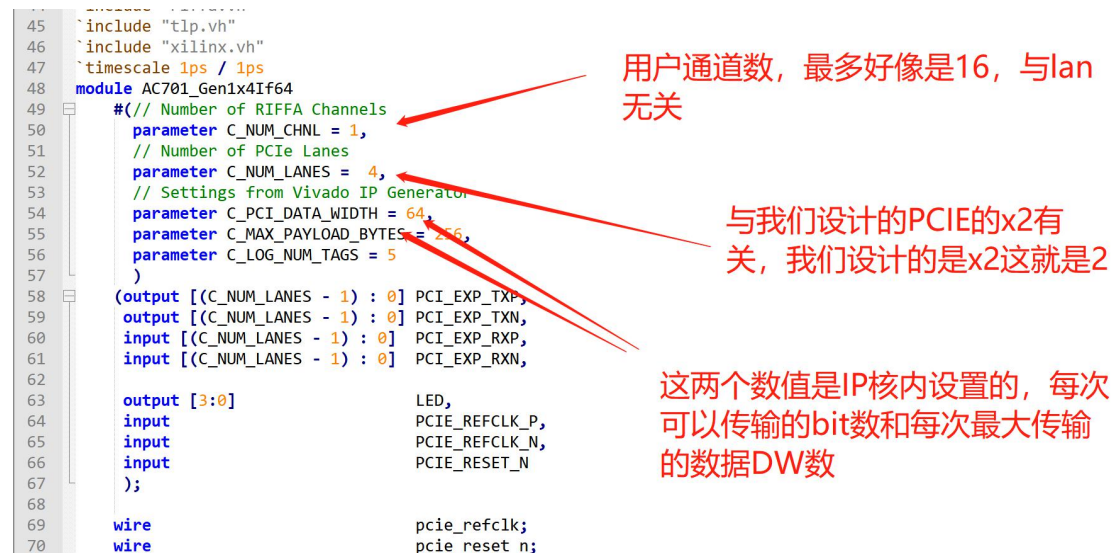


在“接口参数”选项卡中，选择与下图一样的选项，简化生成的核心接口



3) 将 RIFFA 顶层中的 IP 改成自己 IP 的 name, 并删除无效文件

4) 更改顶层中的参数



The screenshot shows a Verilog module definition for AC701_Gen1x4If64. The code includes parameters for the number of channels, lanes, data width, payload bytes, and log number of tags. It also defines inputs and outputs for PCI Express signals and control signals like LED, PCIe reference clock, and reset. Red arrows point from Chinese annotations to specific code lines.

```
45 `include "t1p.vh"
46 `include "xilinx.vh"
47 timescale 1ps / 1ps
48 module AC701_Gen1x4If64
49 #((// Number of RIFFA Channels
50 parameter C_NUM_CHNL = 1,
51 // Number of PCIe Lanes
52 parameter C_NUM_LANES = 4,
53 // Settings from Vivado IP Generator
54 parameter C_PCI_DATA_WIDTH = 64,
55 parameter C_MAX_PAYLOAD_BYTES = 256,
56 parameter C_LOG_NUM_TAGS = 5
57 )
58 (output [(C_NUM_LANES - 1) : 0] PCI_EXP_TXP,
59 output [(C_NUM_LANES - 1) : 0] PCI_EXP_TXN,
60 input [(C_NUM_LANES - 1) : 0] PCI_EXP_RXP,
61 input [(C_NUM_LANES - 1) : 0] PCI_EXP_RXN,
62
63 output [3:0] LED,
64 input PCIe_REFCLK_P,
65 input PCIe_REFCLK_N,
66 input PCIe_RESET_N
67 );
68
69 wire pcie_refclk;
70 wire pcie_reset_n;
```

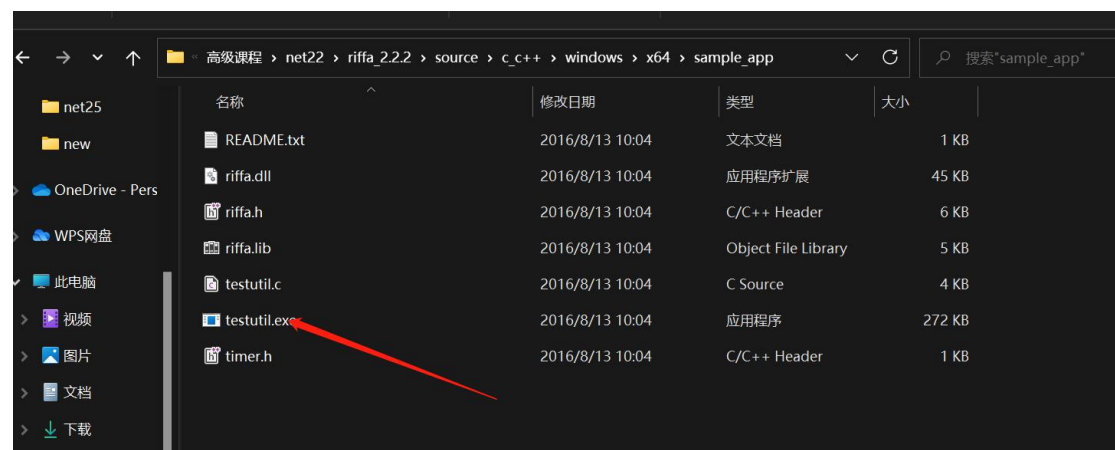
用户通道数, 最多好像是16, 与lan无关

与我们设计的PCIe的x2有关, 我们设计的是x2这就是2

这两个数值是IP核内设置的, 每次可以传输的bit数和每次最大传输的数据DW数

5) 绑定管脚, 生成 bit, 下载板卡

6) Riffa 官方提供了 c 测试文件



The screenshot shows a file explorer window displaying the contents of the 'sample_app' directory. The files listed are README.txt, riffa.dll, riffa.h, riffa.lib, testutil.c, testutil.exe, and timer.h. A red arrow points to the testutil.exe file.

名称	修改日期	类型	大小
README.txt	2016/8/13 10:04	文本文档	1 KB
riffa.dll	2016/8/13 10:04	应用程序扩展	45 KB
riffa.h	2016/8/13 10:04	C/C++ Header	6 KB
riffa.lib	2016/8/13 10:04	Object File Library	5 KB
testutil.c	2016/8/13 10:04	C Source	4 KB
testutil.exe	2016/8/13 10:04	应用程序	272 KB
timer.h	2016/8/13 10:04	C/C++ Header	1 KB

为了使用这个测试, 我们需要安装驱动, 官方驱动没有数字签名所以必须先要关闭

Win10 关闭数字签名的方法:

管理员运行 cmd, 输入 bcdedit.exe /set nointegritychecks on 重启电脑

准备安装驱动

打开设备管理器,查看是否有 PCI 位置设备,如果没有请重复检查是否 FPGA 程序下载然后重启,重启后查看是否有 PCI 未知设备,如果没有需要检查 FPGA 程序是否正确了。

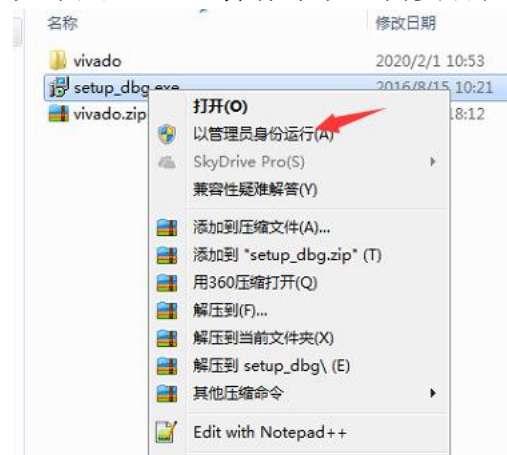
找到 riffa_2.2.2\install\windows\win7 下的

带有 dbg 结尾的是带有调试薪资打印功能的, 请安装此版本。

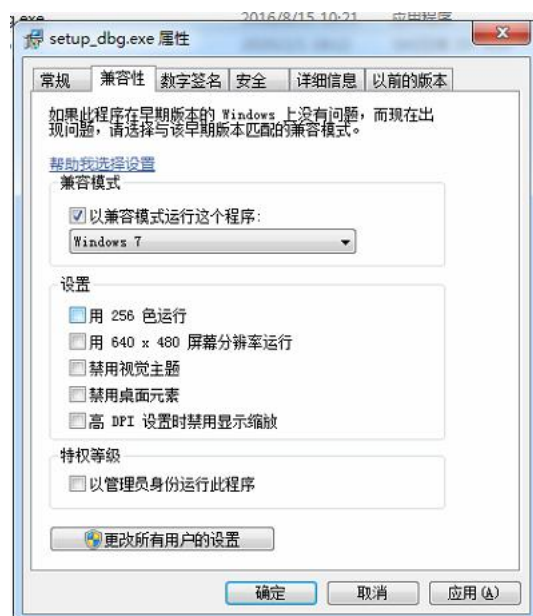
把此文件拷贝到目标需要安装的计算机中

Win7 直接右键管理员安装即可(注意必须已经关闭数字签名,也就之前的 CMD 的操作)

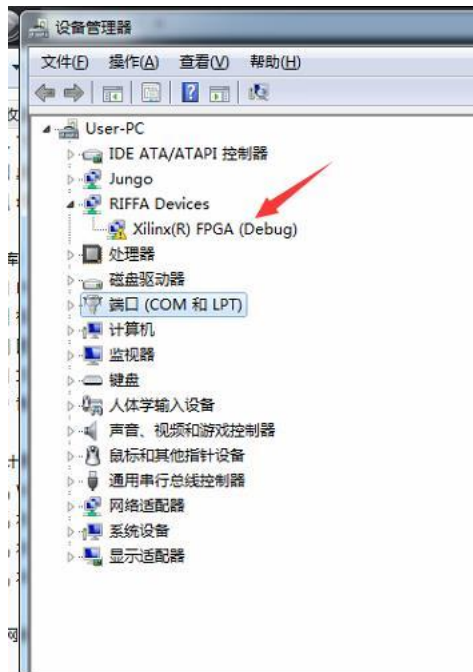
如果是 WIN10 操作系统,需要右键更改兼容模式为 win7 模式,再进行安装。



如果是 win10 需要设置兼容模式,在安装前设置注意注意!



再进入设备管理器查看已经有个 Riffa device, 然后需要重启计算机, 这步必须做



输入如下指令发送测试数据，其实这个指令是 `sample_app` 文件下的 `testutil.exe`

的应用程序，也就是 PCIE 测试程序

命令如下：

```
testutil 2 0 0 10240000
```

命令含义，`testutil` 是命令名称 2 表示测试收发数据，第一个 0 表示 FPGA ID 默

认只有一个 PCIE 板卡所以为 0，第二个 0 表示 `chnl` 为 0，这个再 PCIE 开

发中我们设置 1 个 `chnl` 所以 `id=0`；最后一个 0 是发送多少 word 用于测试数据。

运行后打印带宽信息如下证明板卡和程序驱动运行成功。

**7) 编写自己的用户模块了，在本项目中编写了一个 `conf_channel` ,
`image_ctrl` , `ack_ctrl`**

3.模块介绍

conf_channel	上位机发送截屏的大小，长宽
image_ctrl	上位机实时发送截屏
ack_ctrl	回复上位机收到
Top_ddr3_hdmi	Ddr3_hdmi 的封装

Conf_channel 信号介绍

output [31:0] WIDTH	接受上位机传送来的宽
output [31:0] HEIGHT	接受上位机传送来的高
output CONF_END	接受完成之后的 flag 信号

image_ctrl 信号介绍

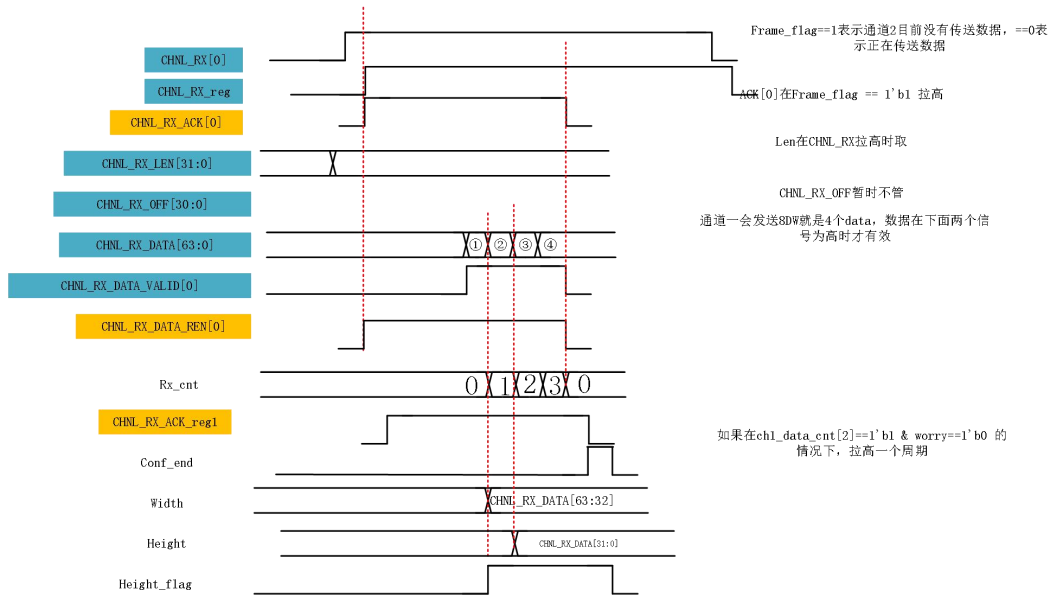
input CONF_END	通过这三个信号来判断是否是需要的帧，如果是则交给 ddr3_hdmi 模块存储显示，不需要就丢弃
input [31:0] HEIGHT	
Input [31:0] WIDTH	
output WR_EN,	若为有效帧，此两个信号为交给下一模块存储显示的数据信号
output [127:0] WR_DATA	
output FRAME_END,	帧结束信号

Top_ddr3_hdmi 模块介绍

Input	WR_EN	需要连续的 64 个 128bit 的数据
Input [127:0]	WR_DATA	
其他信号		其他信号都需要接到顶层，绑定管脚

4. conf_channel 模块时序

配置通道1：传输一幅图像的w和h，所有信号的低位
配置包长度为 8 个 32 位的 DW。
包头为第 0 个 DW 32'h01010101
配置数据宽度 第 1 个 DW :按照宽度进行下发。
配置数据高度 第 2 个 DW：按照高度进行下发。 第 3~7 个 DW：为保留填充 0；
发送配置包要求：需要在两个帧之间发送，不能在发送图像期间发送。
CHNL_RX_LEN:传输的是DW的数量，但是我们传输过来的DATA是64bit



① 低32位应该为32'h01010101，高32位为w

② 低32位应该为h，高32位为0

③④全为0

5. Image_ctrl 模块时序

图像数据通道 2

此通道用于接收上位机发送来的图像，每次发送一帧图像数据，图像数据长度等于图像 DW 的数量。不包含其他数据。

Frame ack 要求在每接收完成一帧图像给上位机发送一次 frame ack 帧数据。

Frame_ack 帧数据是 8 个 DW 长度的 32'h55555555, 用于通知上位机可以发送下一帧数据保证有序的发送图像帧数据。

通道 2 接收的数据可以通过 DDR3 控制器写入到内存中, 并通过读 DDR3 显示到显示器端。

1. PCIE 项目中每个像素都是由一个DW组成 {fr, red[7:0], green[7:0], blue[7:0]}，每次传输2个像素，我们需要在fpga中将其转换成两个16bit的像素 {red[4:0], green[5:0], blue[4:0]}。
2. 要将每次传送的64位像素拼接成32位两个像素，给FIFO存储，等到fifo中由64个128位的数的时候在给user_ctrl。
3. 一帧数据 1920*1080 一个DW传输完成之后产生 frame_end 信号给 ack_ctrl 用于产生给上位机 ACK

我们fifo有可能会到达满值,如果到达满再停止就晚了,我们提前拉高一个fifo_almost_full,当fifo_almost_full == 1的时候我们就不给PCIE再传数据就将CHNL_RX_DATA_REN拉低

