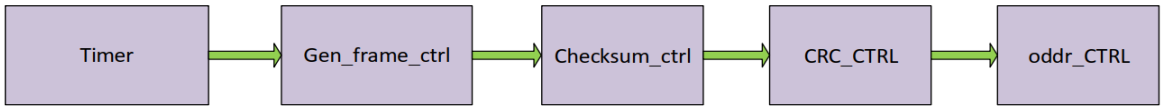
高级练习13 千兆通信中发送链路的 CRC 模块和 ODDR 模块实现

一、练习内容

通过学习以太网的 UDP 帧协议， 完成包的构建和使用网络抓包工具抓包分析数据包是否正确

二、设计分析



三、设计分析

参见练习步骤，没有其他的东西。

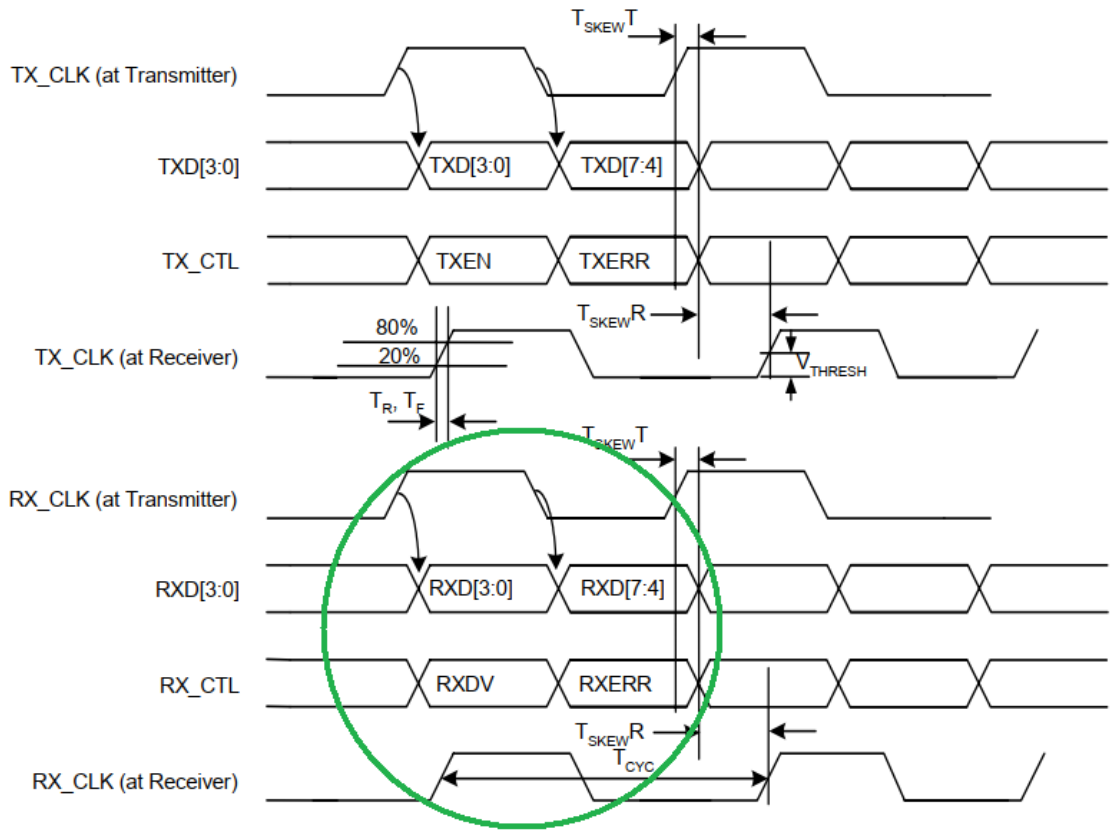
四、练习步骤

1.oddr_ctrl模块的实现

oddr_ctrl的实现与iddr_ctrl模块的实现相反，这里需要将其作为对比，从而寻求更好的解决办法。

查看输入的工作时序，如下图所示。

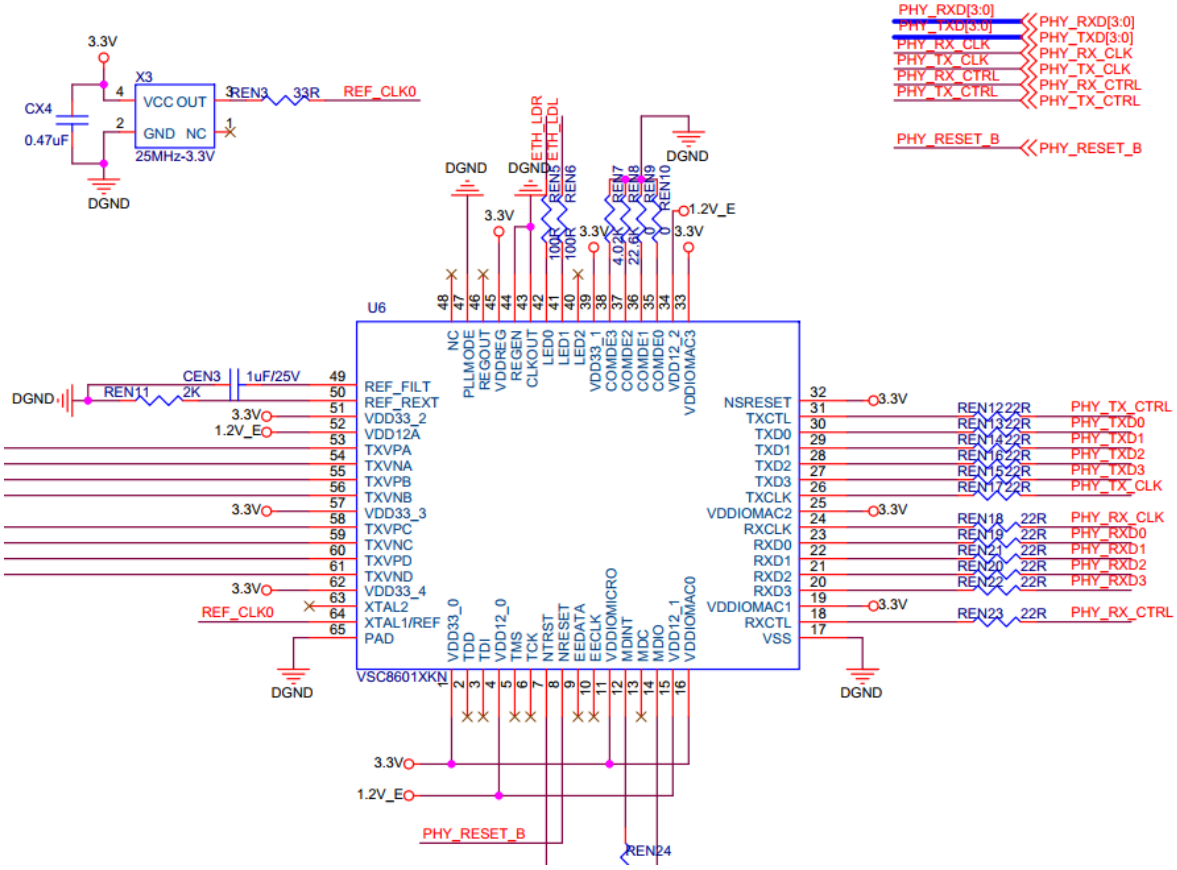
Figure 21. RGMII Uncompensated Timing



和接收相比，这里时序也一点区别，主要是输入输出上的差距，这里的tx_ctrl和tx_dat[3:0]都是输出信号，有tx_en和tx_data转换而来。

由于要用到phy芯片，这部分的使能程序也是必须要有的，要先完成初始化过程，才有后续的相关操作。这里直接将代码复制过来即可。

根据上述分析，这里主要有两部分的信号，分别是tx_en和tx_data[7:0]转换为tx_ctrl和tx_dat[3:0]。这样的话，实现上并不是很难。

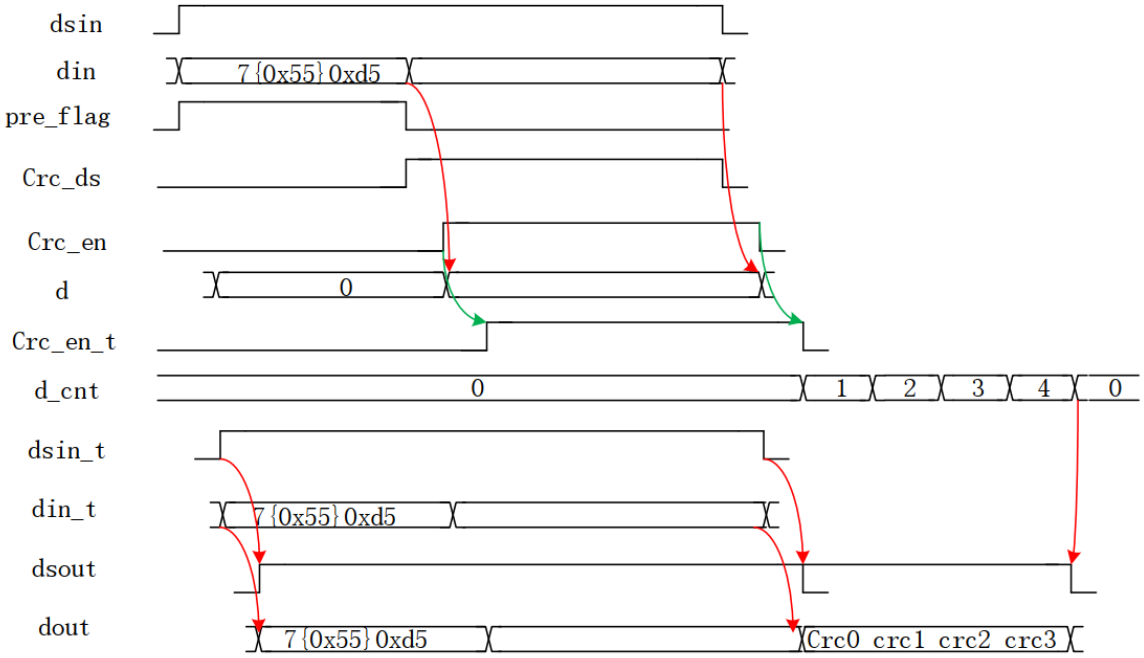


查看相关原理图，我们发现，这里主要控制的主要有两类信号，tx_ctrl和tx_dat, 其他就没有喽。

在代码中，也主要是实现上述相关引脚的电平信号的转变。

2.crc_ctrl模块的实现

这里暂时不做实现，按照下面的时序图实现即可。



3.查看结果

基本代码敲定以后，就需要将程序下入开发板，查看运行的结果。这里主要是观察是否有数据包往上添加，其他没有什么东西。

（1）大局观

查看整体的接收数据，发现所有接收到的数据都是udp的数据，为什么显示是NTP呢，这是因为NTP是UDP的一种，其固定的端口是123，与我们设置的目的端口一致，故由此显示。

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
2	1.000192	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
3	2.000299	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
4	3.000248	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
5	3.999955	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
6	4.999801	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
7	6.000141	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
8	6.999967	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
9	7.999977	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved
10	8.999995	192.168.0.1	255.255.255.255	NTP	106	reserved, reserved

（2）局部观

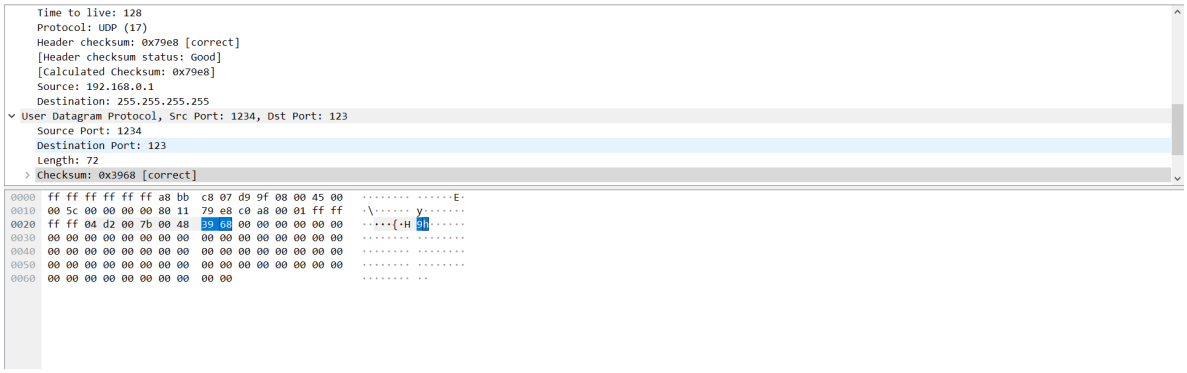
查看某一次接收的具体数据，同时查看相应的IP分析。

首先，我们解析到目标MAC地址和源MAC地址，然后是源端口，源地址和目的端口，目的地址等。

```
> Frame 1: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
v Ethernet II, Src: a8:bb:c8:07:d9:9f (a8:bb:c8:07:d9:9f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  v Destination: Broadcast (ff:ff:ff:ff:ff:ff)
      Address: Broadcast (ff:ff:ff:ff:ff:ff)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
    > Source: a8:bb:c8:07:d9:9f (a8:bb:c8:07:d9:9f)
      Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 1234, Dst Port: 123
> Network Time Protocol (reserved, reserved)
```

```
0000  ff ff ff ff ff ff a8 bb  c8 07 d9 9f 08 00 45 00   ······ ·· ·····E·
0010  00 5c 00 00 00 00 80 11  79 e8 c0 a8 00 01 ff ff   ·\····· y·····
0020  ff ff 04 d2 00 7b 00 48  39 68 00 00 00 00 00 00   ·····{·H 9h·····
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ···· ···· ····
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ···· ···· ····
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ···· ···· ····
0060  00 00 00 00 00 00 00 00  00 00                     ···· ··
```

下面主要查看checksum是否正确。

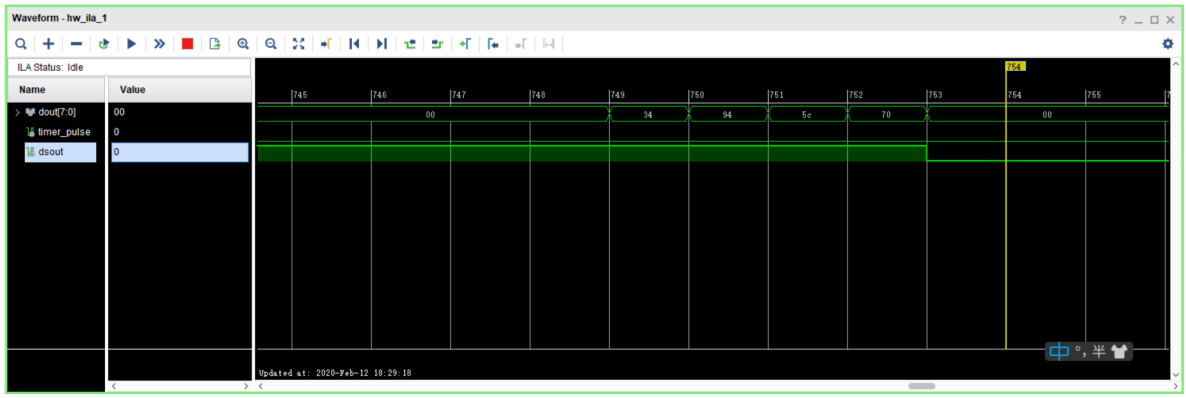


我们发现IP checksum和UDP checksum都是正确的，没有什么问题，基本上搞定啦。

五、实际波形仿真

这次好像是没有波形输出的，搞定就行啦。

查看校验和是否一致保持不变，这个是很重要的。



从结果中我们看到正确捕获到了crc校验，而且0x34945C70这个数值是完成不变的，这是很重要的一部分。

六、总结与讨论

1.ODDR原语的使用

参考Xilinx 7 Series FPGA Libraries Guide for HDL Designs 和 7 Series FPGAs SelectIO Resources文档，主要从基本引脚配置及其功能，工作模式和实例化模板三个方面进行介绍。

（1）主体结构及引脚功能

如下图所示，主要有6个引脚，分别是双边沿数据输出Q，单边沿数据输入D1和D2，时钟使能信号CE，时钟信号输入C, 置位信号S和复位信号R。其中S和R都是高电平有效，同时还能设置同步和异步复位和置位，以及初始化数值等。

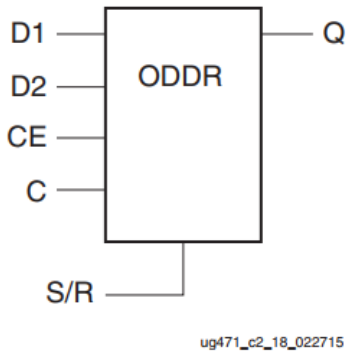


Figure 2-20: **ODDR Primitive Block Diagram**

Table 2-10: **ODDR Port Signals**

Port Name	Function	Description
Q	Data output (DDR)	ODDR register output.
C	Clock input port	The CLK pin represents the clock input pin.
CE	Clock enable port	CE represents the clock enable pin. When asserted Low, this port disables the output clock on port Q.
D1 and D2	Data inputs	ODDR register inputs.
S/R(1)	Set/Reset	Synchronous/Asynchronous set/reset pin. Set/Reset is asserted High.

(2) 工作模式

主要有两种工作模式，即OPPOSITE_EDGE和SAME_EDGE模式。

首先是OPPOSITE_EDGE模式，数据输入不是在同一时刻进行采样，故不采用。

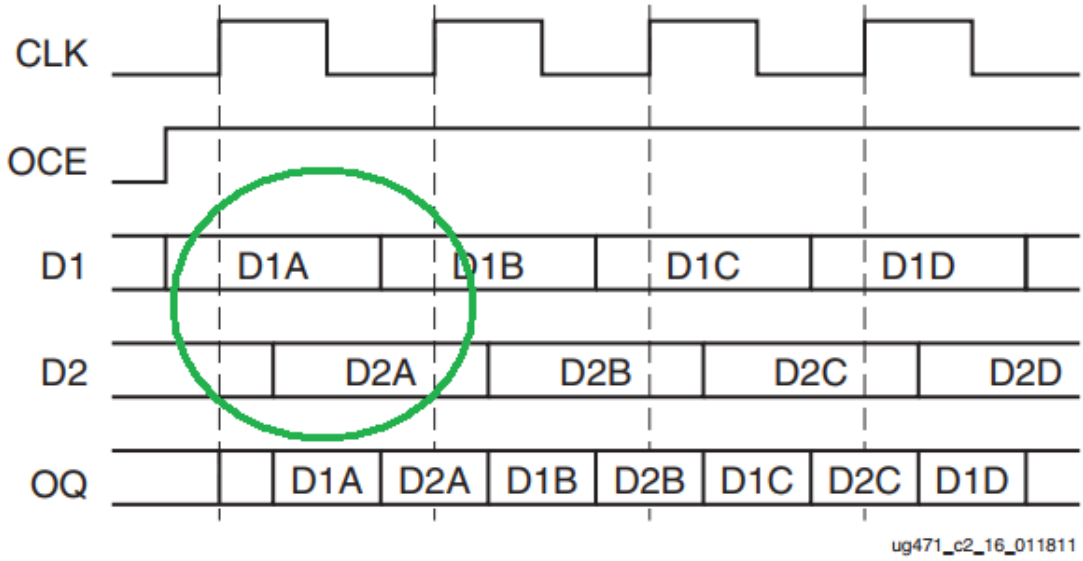
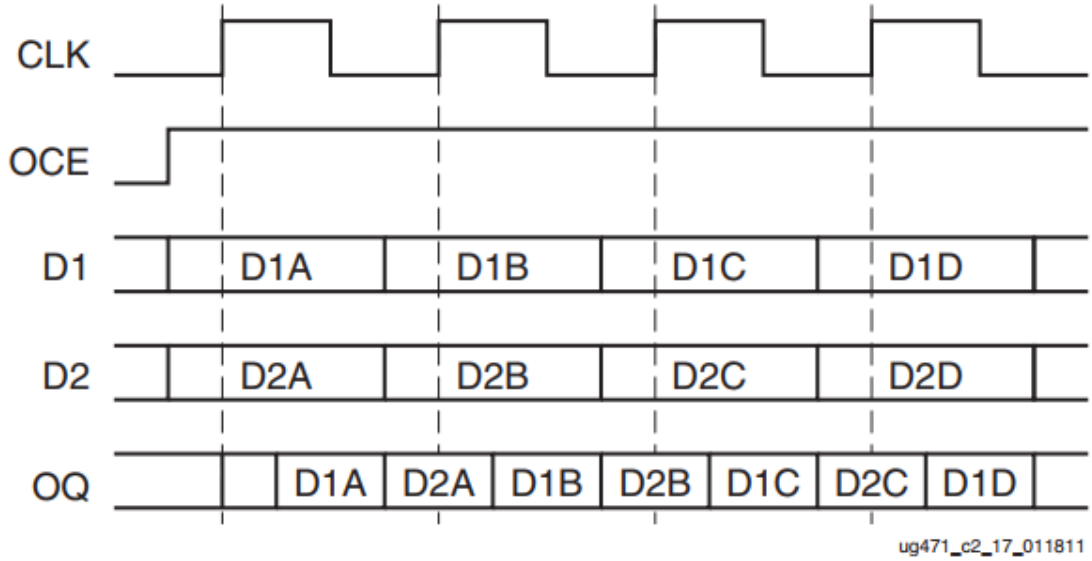


Figure 2-18: **Output DDR Timing in OPPOSITE_EDGE Mode**

然后是SAME_EDGE模式，高低位的数据输入同在上升沿进行采样，最后进行整合。需要注意的是，数据输入时要有高低位的区分，否则容易出错。



（3）实例化模板

实例化模板最好是使用vivado template中的原语模板，排版较为合理。将其列出如下所示。

```
ODDR #(
   .DDR_CLK_EDGE("SAME_EDGE"), // "OPPOSITE_EDGE" or "SAME_EDGE"
   .INIT(1'b0),    // Initial value of Q: 1'b0 or 1'b1
   .SRTYPE("SYNC") // Set/Reset type: "SYNC" or "ASYNC"
) ODDR_inst (
    .Q(tx_dat[0]),   // 1-bit DDR output
    .C(clk),   // 1-bit clock input
    .CE(1'b1), // 1-bit clock enable input
    .D1(tx_data[4]), // 1-bit data input (positive edge)
    .D2(tx_data[0]), // 1-bit data input (negative edge)
    .R(rst),   // 1-bit reset
    .S(1'b0)    // 1-bit set
);
```

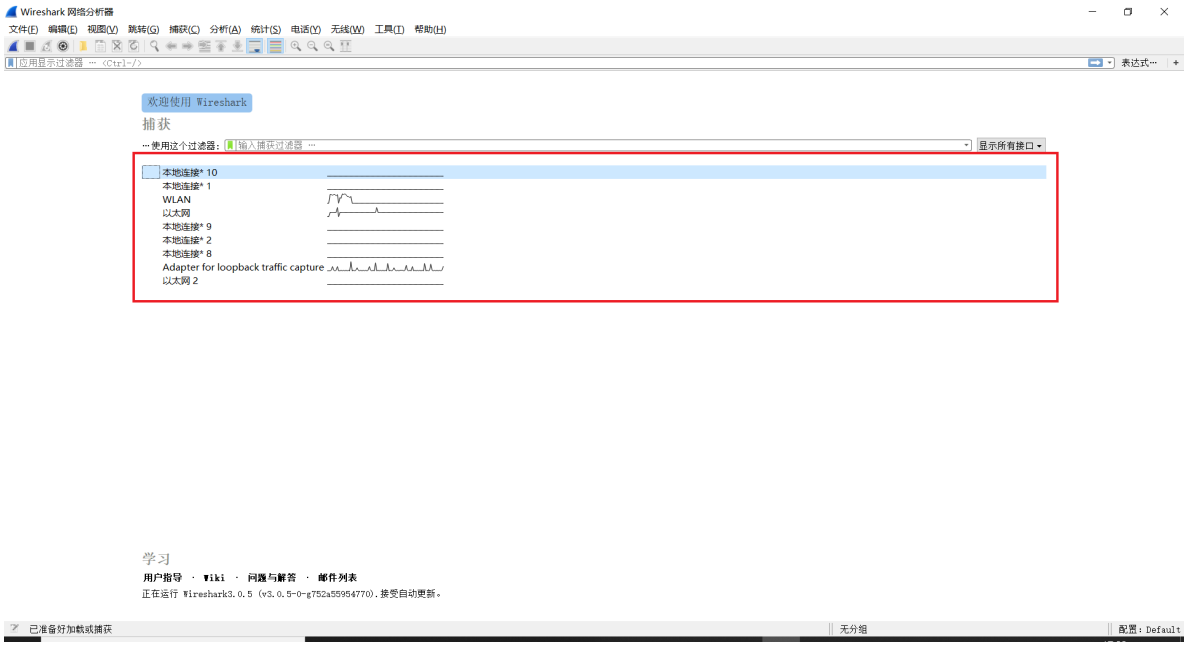
2.Wireshark的使用

这里根据老师的讲解进行总结，自己的话还是不太了解哈。

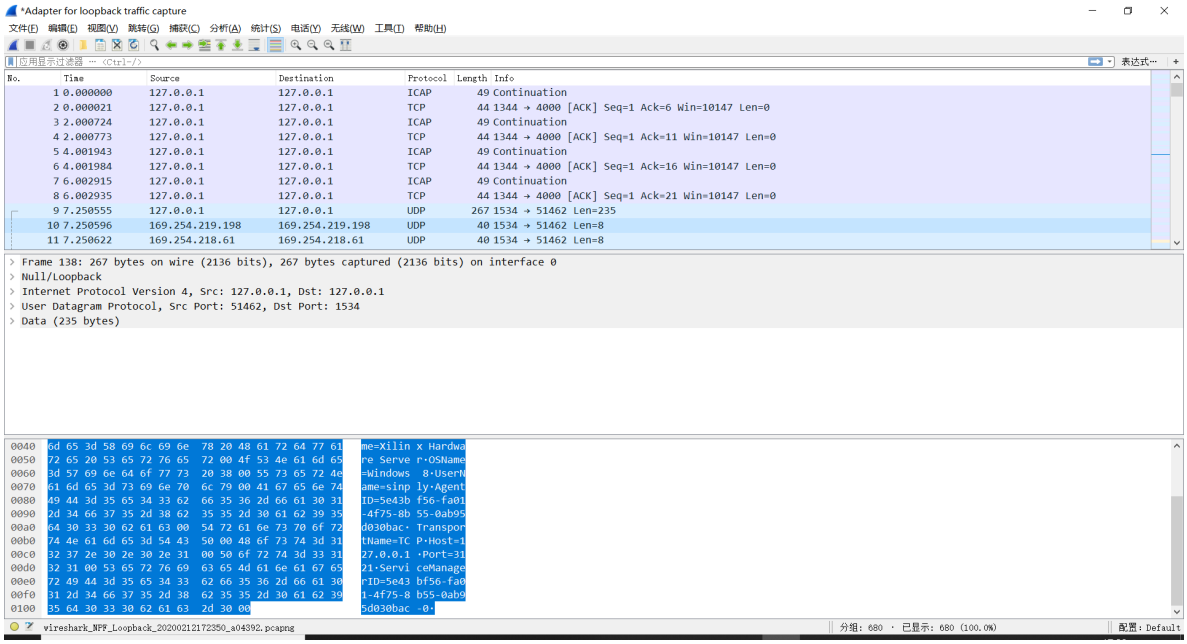
必须要安装Npcap，不然检测不到网络接口，这是必须的，千万要注意，不然无法检测本地网络接口。

（1）基本使用

首先，运行wireshark，软件会自动地检测本地接口，并且显示出来。



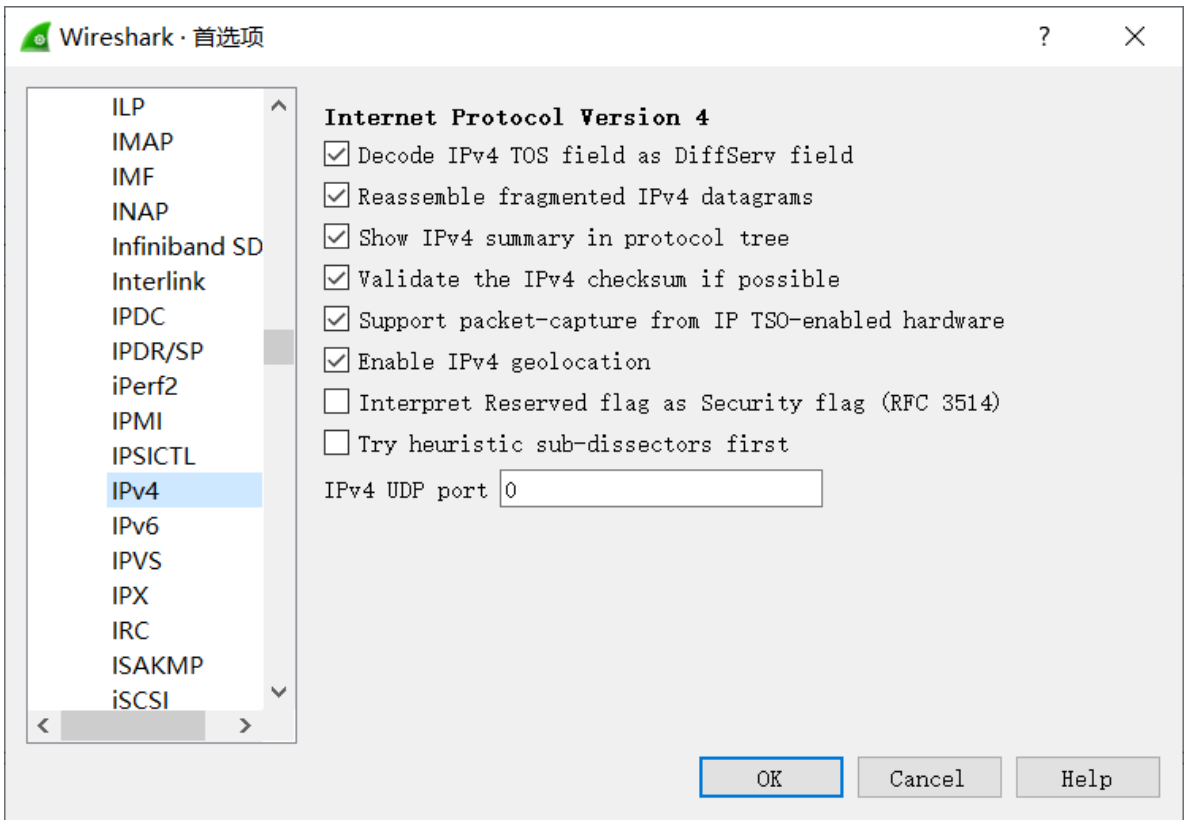
我们查看以太网的连接，故将点击网络端口，以太网进行查看数据。



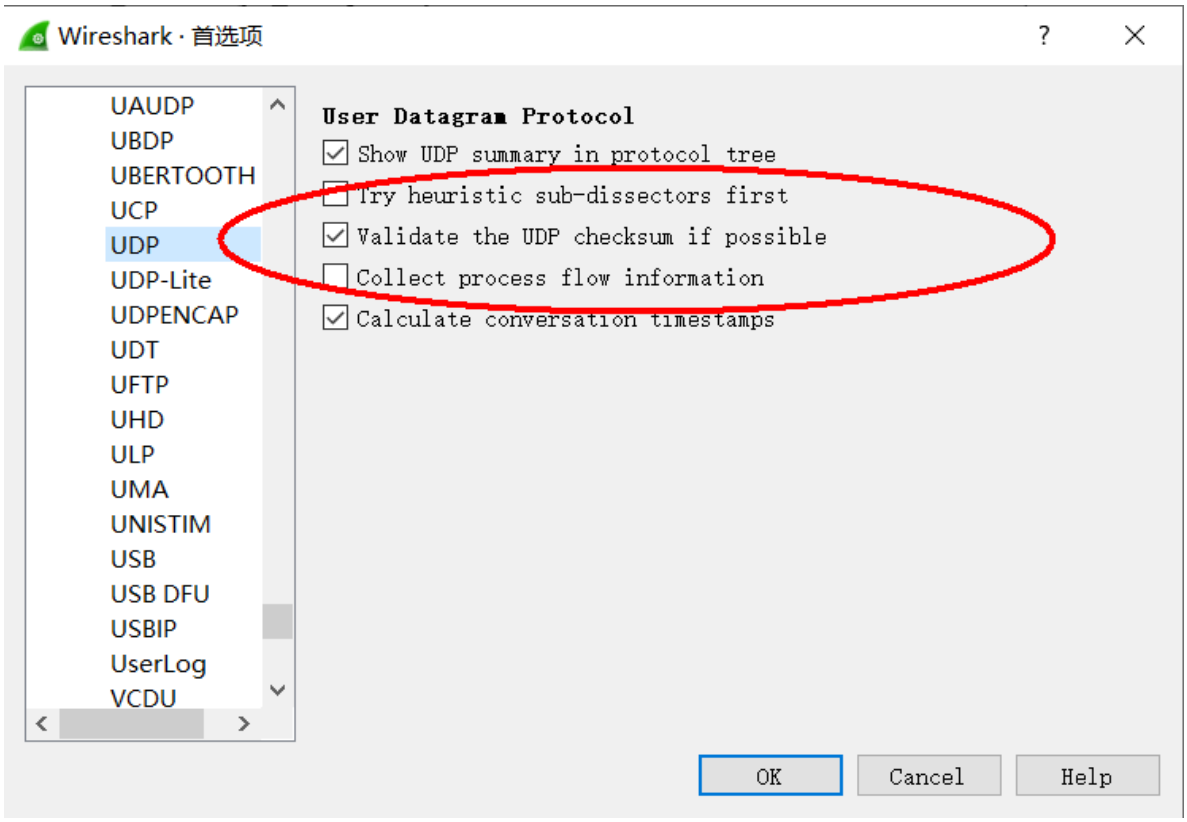
可以看到此网络端口出现的大量数据，最后我们需要找到相应的IP数据，查看数据是否正确。

(2) 功能设置

- 使能checksum校验

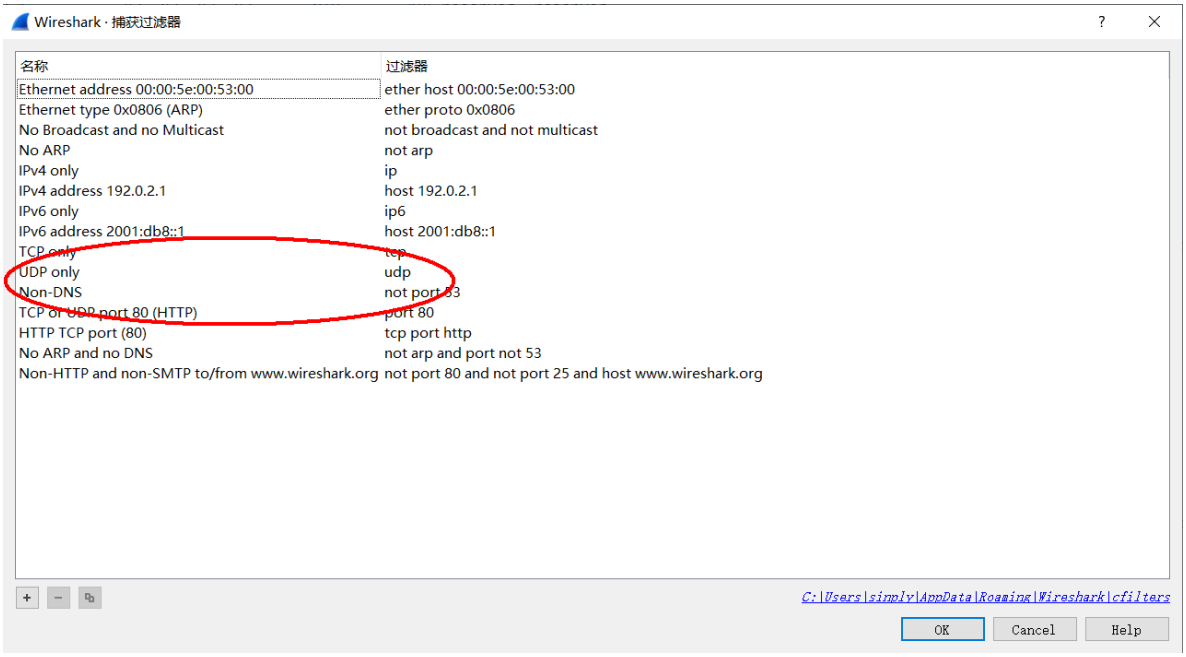


在首选项中的protocol选项中找到IPv4选项，然后使能IP checksum。



在UDP处找到UDP checksum使能的相关选项。

- 捕获滤波设置



可以设置只捕获udp，其他的则会标黑。

基本的使用就这样啦，应该会使用这个软件啦。

3.ODDR原语部分信号的理解

主要是tx_clk和tx_en两路信号为什么要经过ODDR模块理解的不是很透彻，经过老师的说明后，有了比较清楚的认识，下面分别进行说明。

```
    ODDR #(
      .DDR_CLK_EDGE("SAME_EDGE"), // "OPPOSITE_EDGE" or "SAME_EDGE"
      .INIT(1'b0),    // Initial value of Q: 1'b0 or 1'b1
      .SRTYPE("SYNC") // Set/Reset type: "SYNC" or "ASYNC"
    ) ODDR_inst_clk (
       .Q(tx_clk),   // 1-bit DDR output
       .C(tx_c),   // 1-bit clock input
       .CE(1'b1), // 1-bit clock enable input
       .D1(1'b1), // 1-bit data input (positive edge)
       .D2(1'b0), // 1-bit data input (negative edge)
       .R(rst),   // 1-bit reset
       .S(1'b0)    // 1-bit set
        );

    ODDR #(
      .DDR_CLK_EDGE("SAME_EDGE"), // "OPPOSITE_EDGE" or "SAME_EDGE"
      .INIT(1'b0),    // Initial value of Q: 1'b0 or 1'b1
      .SRTYPE("SYNC") // Set/Reset type: "SYNC" or "ASYNC"
    ) ODDR_inst_ctrl (
       .Q(tx_ctrl),   // 1-bit DDR output
       .C(tx_c),   // 1-bit clock input
       .CE(1'b1), // 1-bit clock enable input
       .D1(tx_en), // 1-bit data input (positive edge)
       .D2(tx_en), // 1-bit data input (negative edge)
       .R(rst),   // 1-bit reset
       .S(1'b0)    // 1-bit set
        );
```

（1）tx_en上升沿和下降沿都作为输入，两者起到不同的作用

要把tx_en传递出去，其中上升沿的tx_en输入表示数据有效，下降沿的tx_en则表示数据没有错误，因为这里通过高电平是表示没有错误的，故在tx_en有效的范围内，在下降沿输入tx_en拉高完全是可行的

（2）tx_clk经过ODDR模块

是为了数据同步，而不是为了相移，相移只能是PLL之类的IP模块进行实现。