11. 以太网传输实验例程

11.1 实验目的

MES22GP 开发板使用 Realtek RTL8211E PHY 实现了一个 10/100/1000 以太网端口，用于网络连接。该器件工作电压为支持 2.5V、3.3V。PHY 连接到 BANK R3，并通过 RGMII 接口连接到 PGL22G。RJ-45 连接器是 HFJ11-1G01E-L12RL，具有集成的自动缠绕磁性元件，可提高性能，质量和可靠性。RJ-45 有两个状态指示灯 LED，用于指示流量和有效链路状态（详情请查看“MES22GP 开发板硬件使用手册”）。

下图显示了 MES22GP 开发板上的网口连接框图。

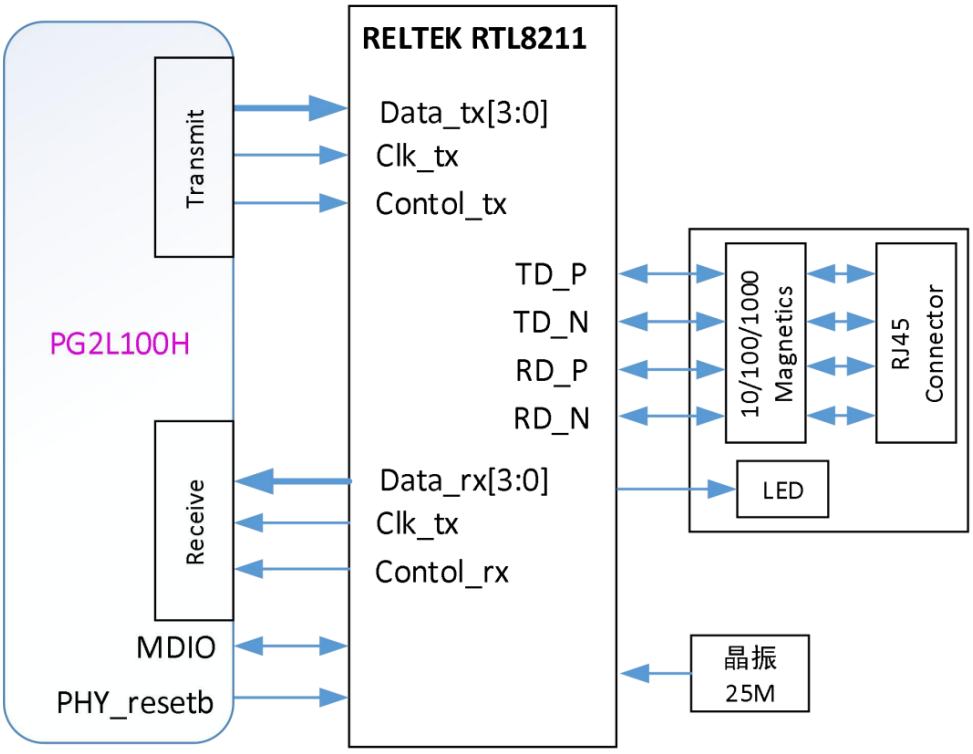
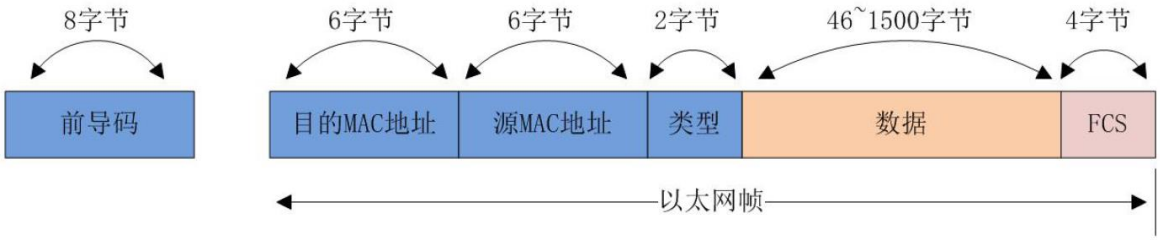


图 2.9 RTL8211 连接示例

通过以太网端口实现 PC 端和开发板间通信，实现了 ARP，UDP 功能。

11.2 以太网协议简介

11.2.1 以太网帧格式



前导码（Preamble）：8 字节，连续 7 个 8’h55 加 1 个 8’hd5，表示一个帧的开始，用于双方；设备数据的同步。

目的 MAC 地址：6 字节，存放目的设备的物理地址，即 MAC 地址；

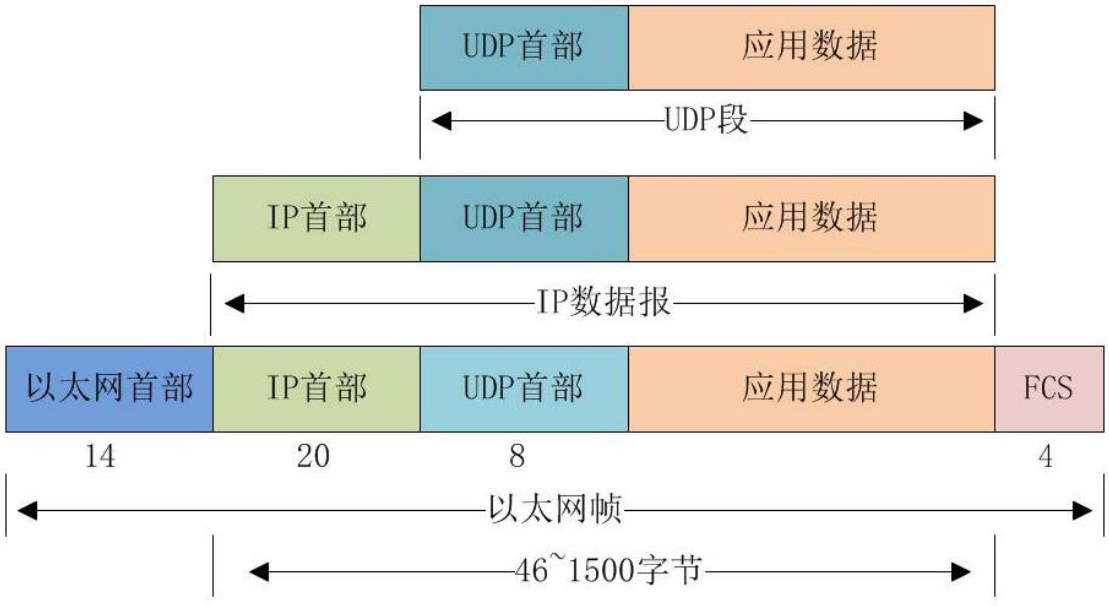
源 MAC 地址：6 字节，存放发送端设备的物理地址；

类型：2 字节，用于指定协议类型，常用的有 0800 表示 IP 协议，0806 表示 ARP 协议，8035 表示 RARP 协议；

数据：46 到 1500 字节，最少 46 字节，不足需要补全 46 字节，例如 IP 协议层就包含在数据部分，包括其 IP 头及数据。

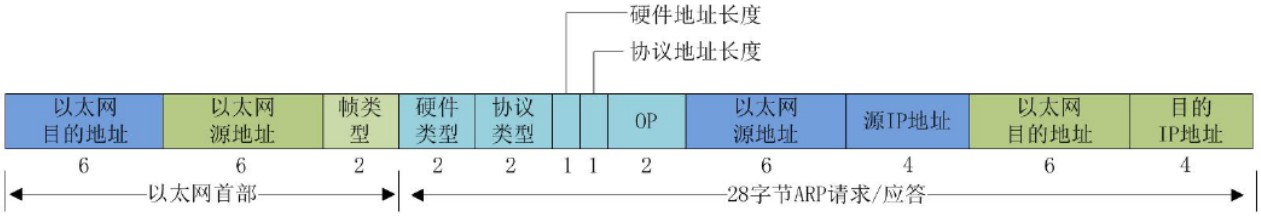
FCS：帧尾，4 字节，称为帧校验序列，采用 32 位 CRC 校验，对目的 MAC 地址字段到数据字段进行校验。

进一步扩展，以 UDP 协议为例，可以看到其结构如下，除了以太网首部的 14 字节，数据部分包含 IP 首部，UDP 首部，应用数据共 46~1500 字节。



11.2.2 ARP 数据报格式

ARP 地址解析协议，即 ARP（Address Resolution Protocol），根据 IP 地址获取物理地址。主机发送包含目的 IP 地址的 ARP 请求广播（MAC 地址为 48’hff_ff_ff_ff_ff_ff）到网络上的主机，并接收返回消息，以此确定目标的物理地址，收到返回消息后将 IP 地址和物理地址保存到缓存中，并保留一段时间，下次请求时直接查询 ARP 缓存以节约资源。下图为 ARP 数据报格式。



帧类型：ARP 帧类型为两字节 0806；

硬件类型：指链路层网络类型，1 为以太网；

协议类型：指要转换的地址类型，采用 0x0800 IP 类型，之后的硬件地址长度和协议地址长度分别对应 6 和 4；

OP 字段中 1 表示 ARP 请求，2 表示 ARP 应答

例如：|ff ff ff ff ff ff|00 0a 35 01 fe c0|08 06|00 01|08 00|06|04|00 01|00 0a 35 01 fe c0|c0 a8 00 02| ff ff ff ff ff ff|c0 a8 00 03|

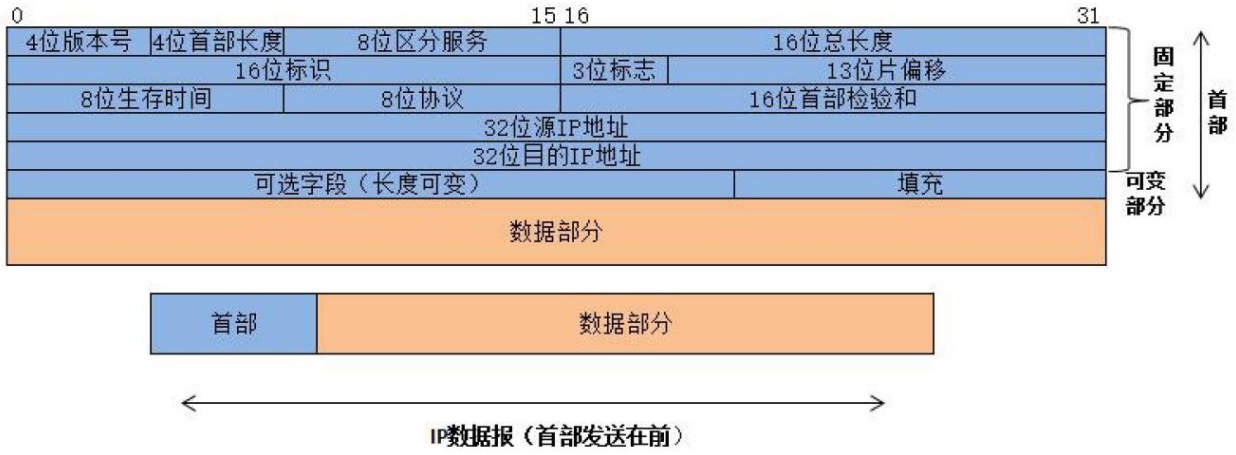
表示向 192.168.0.3 地址发送 ARP 请求。

|00 0a 35 01 fe c0 | 60 ab c1 a2 d5 15 |08 06|00 01|08 00|06|04|00 02| 60 ab c1 a2 d5 15|c0 a8 00 03|00 0a 35 01 fe c0|c0 a8 00 02|

表示向 192.168.0.2 地址发送 ARP 应答。

11.2.3 IP 数据包格式

因为 UDP 协议包只是 IP 包中的一种，所以我们来介绍一下 IP 包的数据格式。下图为 IP 分组的报文头格式，报文头的前 20 个字节是固定的，后面的可变



版本: 占 4 位,指 IP 协议的版本目前的 IP 协议版本号为 4 (即 IPv4);

首部长度: 占 4 位,可表示的最大数值是 15 个单位(一个单位为 4 字节)因此 IP 的首部长度的最大值是 60 字节;

区分服务: 占 8 位,用来获得更好的服务,在旧标准中叫做服务类型,但实际上一直未被使用过 1998 年这个字段改名为区分服务.只有在使用区分服务(DiffServ)时,这个字段才起作用.一般的情况下都不使用这个字段;

总长度: 占 16 位,指首部和数据之和的长度,单位为字节,因此数据报的最大长度为 65535 字节总长度必须不超过最大传送单元 MTU

标识: 占 16 位,它是一个计数器,用来产生数据报的标识

标志(flag):

占 3 位,目前只有前两位有意义

MF

标志字段的最低位是 MF (More Fragment)

MF=1 表示后面“还有分片”。MF=0 表示最后一个分片

DF

标志字段中间的一位是 DF (Don't Fragment)

只有当 DF=0 时才允许分片

片偏移: 占 12 位,指较长的分组在分片后某片在原分组中的相对位置.片偏移以 8 个字节为偏移单位;

生存时间: 占 8 位,记为 TTL (Time To Live) 数据报在网络中可通过的路由器数的最大

值,TTL 字段是由发送端初始设置一个 8 bit 字段.推荐的初始值由分配数字 RFC 指定,当前值为 64.发送 ICMP 回显应答时经常把 TTL 设为最大值 255;

协议:占 8 位,指出此数据报携带的数据使用何种协议以便目的主机的 IP 层将数据部分上交给哪个处理过程, 1 表示为 ICMP 协议, 2 表示为 IGMP 协议, 6 表示为 TCP 协议, 17 表示为 UDP 协议;

首部检验和:占 16 位,只检验数据报的首部不检验数据部分，采用二进制反码求和，即将 16 位数据相加后，再将进位与低 16 位相加，直到进位为 0，最后将 16 位取反;

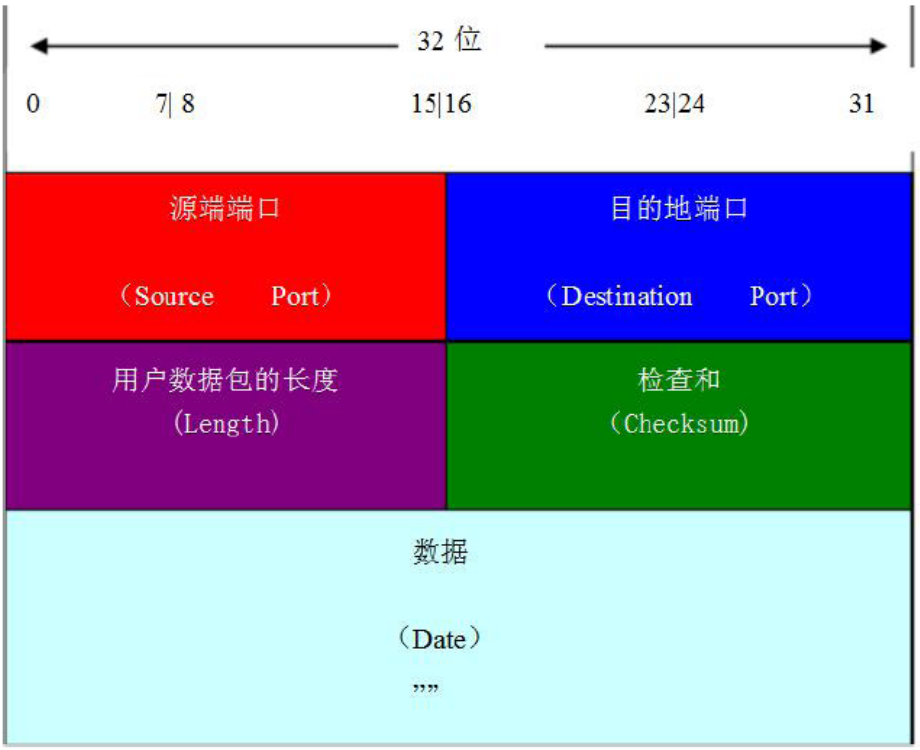
源地址和目的地址:都各占 4 字节,分别记录源地址和目的地址;

11.2.4 UDP 协议

UDP 是 User Datagram Protocol（用户数据报协议）的英文缩写。UDP 只提供一种基本的、低延迟的被称为数据报的通讯。所谓数据报，就是一种自带寻址信息，从发送端走到接收端的数据包。UDP 协议经常用于图像传输、网络监控数据交换等数据传输速度要求比较高的场合。

UDP 协议的报头格式:

UDP 报头由 4 个域组成，其中每个域各占用 2 个字节，具体如下：



① UDP 源端口号

② 目标端口号

③ 数据报长度

④ 校验和

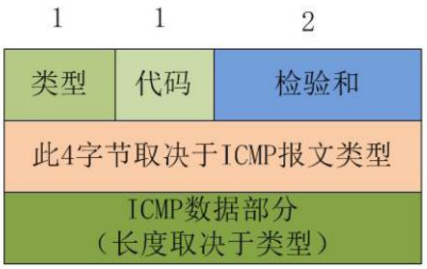
UDP 协议使用端口号为不同的应用保留其各自的数据传输通道。数据发送一方将 UDP 数据报通过源端口发送出去，而数据接收一方则通过目标端口接收数据。

数据报的长度是指包括报头和数据部分在内的总字节数。因为报头的长度是固定的，所以该域主要被用来计算可变长度的数据部分（又称为数据负载）。数据报的最大长度根据操作环境的不同而各异。从理论上说，包含报头在内的数据报的最大长度为 65535 字节。不过，一些实际应用往往会限制数据报的大小，有时会降低到 8192 字节。

UDP 协议使用报头中的校验值来保证数据的安全。校验值首先在数据发送方通过特殊的算法计算得出，在传递到接收方之后，还需要再重新计算。如果某个数据报在传输过程中被第三方篡改或者由于线路噪音等原因受到损坏，发送和接收方的校验计算值将不会相符，由此 UDP 协议可以检测是否出错。虽然 UDP 提供有错误检测，但检测到错误时，错误校正，只是简单地把损坏的消息段扔掉，或者给应用程序提供警告信息。

11.2.5 Ping 功能

UDP 协议使用报头中的校验值来保证数据的安全。校验值首先在数据发送方通过特殊的算法计算得出，在传递到接收方之后，还需要再重新计算。如果某个数据报在传输过程中被第三方篡改或者由于线路噪音等原因受到损坏，发送和接收方的校验计算值将不会相符，由此 UDP 协议可以检测是否出错。虽然 UDP 提供有错误检测，但检测到错误时，错误校正，只是简单地把损坏的消息段扔掉，或者给应用程序提供警告信息。



类型值（十进制）	ICMP报文类型
0	回送应答
3	终点不可达
4	源点抑制
5	改变路由
8	回送请求
11	时间超时

11.3 SMI(MDC/MDIO)总线接口

串行管理接口（Serial Management Interface），也被称作 MII 管理接口（MII ManagementInterface），包括 MDC 和 MDIO 两条信号线。MDIO 是一个 PHY 的管理接口，用来读/写 PHY 的寄存器，以控制 PHY 的行为或获取 PHY 的状态，MDC 为 MDIO 提供时钟，由 MAC 端提供，在本实验中也就是 FPGA 端。在 RTL8211EG 文档里可以看到 MDC 的周期最小为 400ns，也就是最大时钟为 2.5MHz。

Table 61. MDC/MDIO Management Timing Parameters

Symbol	Description	Minimum	Maximum	Unit
t_1	MDC High Pulse Width	160	-	ns
t_2	MDC Low Pulse Width	160	-	ns
t_3	MDC Period	400	-	ns
t_4	MDIO Setup to MDC Rising Edge	10	-	ns
t_5	MDIO Hold Time from MDC Rising Edge	10	-	ns
t_6	MDIO Valid from MDC Rising Edge	0	300	ns

11.3.1 SMI 帧格式

如下图，为 SMI 的读写帧格式：

	Management Frame Fields							
	Preamble	ST	OP	PHYAD	REGAD	TA	DATA	IDLE
Read	1...1	01	10	AAAAA	RRRRR	Z0	DDDDDDDDDDDDDDDD	Z
Write	1...1	01	01	AAAAA	RRRRR	10	DDDDDDDDDDDDDDDD	Z

名称	说明
Preamble	由 MAC 发送 32 个连续的逻辑“1”，同步于 MDC 信号，用于 MAC 与 PHY 之间的同步；
ST	帧开始位，固定为 01

OP	操作码，10 表示读，01 表示写
PHYAD	PHY 的地址，5 bits
REGAD	寄存器地址，5 bits
TA	Turn Around，MDIO 方向转换，在写状态下，不需要转换方向，值为 10，在读状态下，MAC 输出端为高阻态，在第二个周期，PHY 将 MDIO 拉低
DATA	共 16bits 数据
IDLE	空闲状态，此状态下 MDIO 为高阻态，由外部上拉电阻拉高

11.3.2 读时序

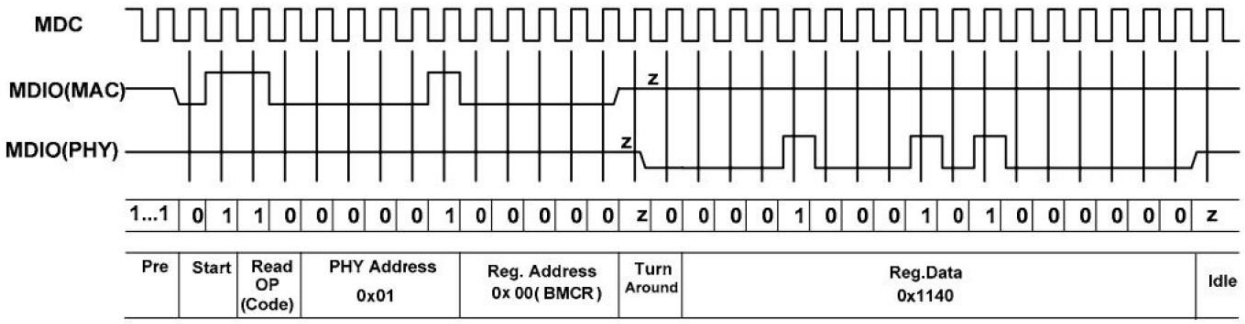


Figure 8. MDC/MDIO Read Timing

可以看到在 Turn Around 状态下，第一个周期 MDIO 为高阻态，第二个周期由 PHY 端拉低。

11.3.3 写时序

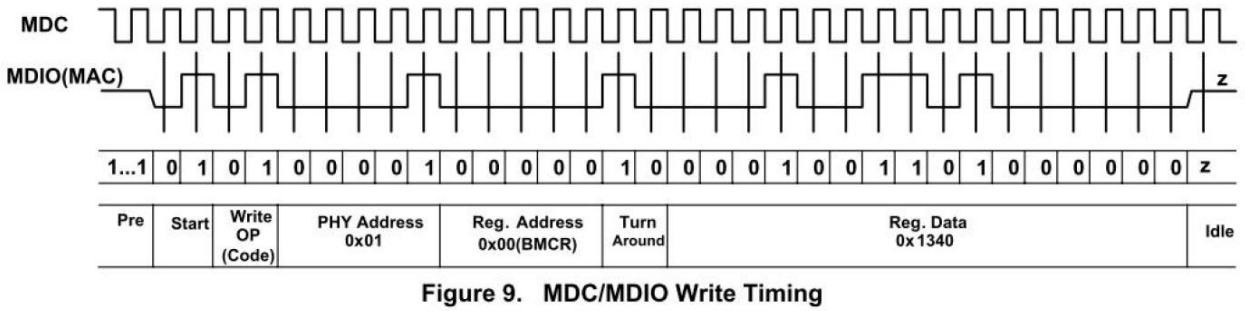


Figure 9. MDC/MDIO Write Timing

为了保证能够正确采集到数据，在 MDC 上升沿之前就把数据准备好，在本实验中为下降沿发送数据，上升沿接收数据。

11.4 实验设计

本实验以千兆以太网 RGMII 通信为例来设计 verilog 程序，会先发送预设的 UDP 数据到

网络，每秒钟发送一次.程序分为两部分，分别为发送和接收，实现了 ARP，UDP 功能。

11.4.1发送部分

11.4.1.1 MAC 层发送

发送部分中，mac_tx.v 为MAC 层发送模块,首先在SEND_START 状态,等待mac_tx_ready 信号，如果有效，表明 IP 或 ARP 的数据已经准备好，可以开始发送。再进入发送前导码状态，结束时发送 mac_data_req，请求 IP 或 ARP 的数据，之后进入发送数据状态，最后进入发送 CRC 状态。在发送数据过程中，需要同时进行 CRC 校验。前导码完成后就将上层协议数据发送出去，这个时候同样把这些上层数据放到 CRC32 模块中做序列生成，上层协议会给一个数据输出完成标志信号,这个时候 mac_tx 知道数据发送完成了,需要结束 CRC32 的序列生成，这个时候就开始提取 FCS，衔接数据之后发送出去。这样就连接了前导码---数据（Mac 帧）-----FCS。之后跳转到结束状态，再回到 IDLE 状态，等待下一次的发送请求。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位
mac_frame_data	input	8	从 IP 或 ARP 来的数据
mac_tx_req	input	1	MAC 的发送请求
mac_tx_ready	input	1	IP 或 ARP 数据已准备好
mac_tx_end	input	1	IP 或 ARP 数据已经传输完毕
mac_tx_data	output	8	向 PHY 发送数据
mac_send_end	output	1	MAC 数据发送结束
mac_data_valid	output	1	MAC 数据有效信号，即 gmii_tx_en
mac_data_req	output	1	MAC 层向 IP 或 ARP 请求数据
mac_tx_ack	output	1	MAC 层发送数据的对于 UPPER 请求的应答

11.4.1.2 MAC 发送模式

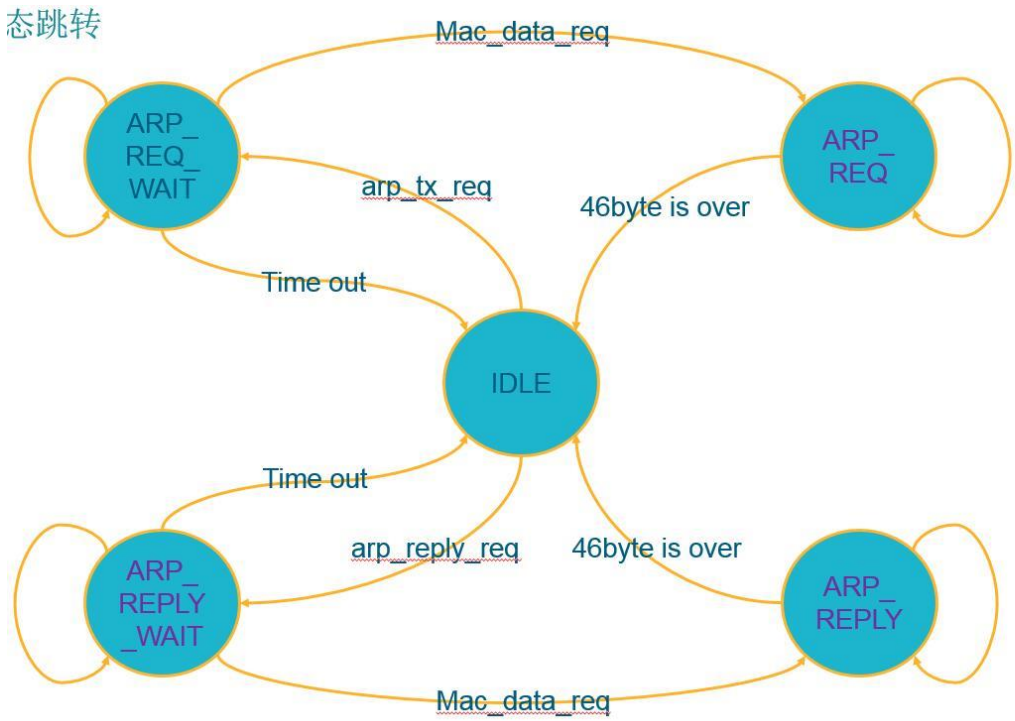
工程中的 mac_tx_mode.v 为发送模式选择，根据发送模式是 IP 或 ARP 选择相应的信号与数据。

信号名称	方向	位宽	说明

clk	input	1	系统时钟
rstn	input	1	低电平复位
mac_send_end	input	1	MAC 发送结束
arp_tx_req	input	1	ARP 发送请求
arp_tx_ready	input	1	ARP 数据已准备好
arp_tx_data	input	8	ARP 数据
arp_tx_end	input	1	ARP 数据发送到 MAC 层结束
arp_tx_ack	input	1	ARP 发送响应信号
ip_tx_req	input	1	IP 发送请求
ip_tx_ready	input	1	IP 数据已准备好
ip_tx_data	input	8	IP 数据
ip_tx_end	input	1	IP 数据发送到 MAC 层结束
mac_tx_ready	output	1	MAC 数据已准备好信号
ip_tx_ack	output	1	IP 发送响应信号
mac_tx_ack	input	1	MAC 发送响应信号
mac_tx_req	output	1	MAC 发送请求
mac_tx_data	output	8	MAC 发送数据
mac_tx_end	output	1	MAC 数据发送结束

11.4.1.3 ARP 发送

发送部分中，arp_tx.v 为 ARP 发送模块， 在 IDLE 状态下，等待 ARP 发送请求或 ARP 应答请求信号，之后进入请求或应答等待状态，并通知 MAC 层，数据已经准备好，等待 mac_data_req 信号，之后进入请求或应答数据发送状态。由于数据不足 46 字节，需要补全 46 字节发送。



信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位
dest_mac_addr	input	48	发送的目的 MAC 地址
sour_mac_addr	input	48	发送的源 MAC 地址
sour_ip_addr	input	32	发送的源 IP 地址
dest_ip_addr	input	32	发送的目的 IP 地址
mac_data_req	input	1	MAC 层请求数据信号
arp_request_req	input	1	ARP 请求的请求信号
arp_reply_ack	output	1	ARP 回复的应答信号
arp_reply_req	input	1	ARP 回复的请求信号
arp_rec_sour_ip_addr	input	32	ARP 接收的源 IP 地址，回复时放到目的 IP 地址
arp_rec_sour_mac_addr	input	48	ARP 接收的源 MAC 地址，回复时放到目的 MAC 地址

mac_send_end	input	1	MAC 发送结束
mac_tx_ack	input	1	MAC 发送应答
arp_tx_ready	output	1	ARP 数据准备好
arp_tx_data	output	8	ARP 发送数据
arp_tx_end	output	1	ARP 数据发送结束
arp_tx_req	output	1	ARP 发送请求信号

11.4.1.4 IP 层发送

在发送部分，ip_tx.v 为 IP 层发送模块，在 IDLE 状态下，如果 ip_tx_req 有效，也就是 UDP 或 ICMP 发送请求信号，进入等待发送数据长度状态，之后进入产生校验和状态，校验和是将 IP 首部所有数据以 16 位相加，最后将进位再与低 16 位相加，直到进入为 0，再将低 16 位取反，得出校验和结果。

在生成校验和之后，等待 MAC 层数据请求，开始发送数据，并在即将结束发送 IP 首部后请求 UDP 或 ICMP 数据。等发送完，进入 IDLE 状态。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位
dest_mac_addr	input	48	发送的目的 MAC 地址
sour_mac_addr	input	48	发送的源 MAC 地址
sour_ip_addr	input	32	发送的源 IP 地址
dest_ip_addr	input	32	发送的目的 IP 地址
ttl	input	8	生存时间
ip_send_type	input	8	上层协议号，如 UDP,ICMP
upper_layer_data	output	8	从 UDP 或 ICMP 过来的数据
upper_data_req	input	1	向上层请求数据
mac_tx_ack	input	1	MAC 发送应答
mac_send_end	input	1	MAC 发送结束信号
mac_data_req	input	1	MAC 层请求数据信号
upper_tx_ready	input	1	上层 UDP 或 ICMP 数据准备好

ip_tx_req	input	1	发送请求，从上层过来
ip_send_data_length	input	16	发送数据总长度
ip_tx_ack	output		产生 IP 发送应答
ip_tx_ready	output	1	IP 数据已准备好
ip_tx_data	output	8	IP 数据
ip_tx_end	output	1	IP 数据发送到 MAC 层结束

11.4.1.5 IP 发送模式

工程中的 ip_tx_mode.v 为发送模式选择，根据发送模式是 UDP 或 ICMP 选择相应的信号与数据。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位
mac_send_end	input		MAC 数据发送结束
udp_tx_req	input	1	UDP 发送请求
udp_tx_ready	input	1	UDP 数据准备好
udp_tx_data	input	8	UDP 发送数据
udp_send_data_length	input	16	UDP 发送数据长度
udp_tx_ack	output	1	输出 UDP 发送应答
icmp_tx_req	input	1	ICMP 发送请求
icmp_tx_ready	input	1	ICMP 数据准备好
icmp_tx_data	input	8	ICMP 发送数据
icmp_send_data_length	input	16	ICMP 发送数据长度
icmp_tx_ack	output	1	ICMP 发送应答
ip_tx_ack	input	1	IP 发送应答
ip_tx_req	input	1	IP 发送请求
ip_tx_ready	output	1	IP 数据已准备好
ip_tx_data	output	8	IP 数据

ip_send_type	output	8	上层协议号，如 UDP,ICMP
ip_send_data_length	output	16	发送数据总长度

11.4.1.6 UDP 发送

发送部分中，udp_tx.v 为 UDP 发送模块。

信号名称	方向	位宽	说明
udp_send_clk	input	1	系统时钟
rstn	input	1	低电平复位
app_data_in_valid	input	1	从外部所接收的数据输出有效信号
app_data_in	input	8	外部所接收的数据
app_data_length	input	16	从外部所接收的当前数据包的长度（不含 udp、ip、mac 首部）
udp_dest_port	input	16	从外部所接收的数据包的源端口号
app_data_request	input	1	用户接口数据发送请求
udp_send_ready	output	1	UDP 数据发送准备
udp_send_ack	output	1	UDP 数据发送应当
ip_send_ready	input	1	IP 数据发送准备
ip_send_ack	input	1	IP 数据发送应当
udp_send_request	output	1	用户接口数据发送请求
udp_data_out_valid	output	1	发送的数据输出有效信号
udp_data_out	output	8	发送的数据输出
udp_packet_length	output	16	当前数据包的长度（不含 udp、ip、mac 首部）

11.4.2 接收部分

11.4.2.1 MAC 层接收

在接收部分，其中 mac_rx.v 为 mac 层接收文件，首先在 IDLE 状态下当 rx_en 信号为高，进入 REC_PREAMBLE 前导码状态，接收前导码。之后进入接收 MAC 头部状态，即目的 MAC 地址，源 MAC 地址，类型，将它们缓存起来，并在此状态判断前导码是否正确，错误则进入

REC_ERROR 错误状态，在 REC_IDENTIFY 状态判断类型是 IP（8’h0800）或 ARP(8’h0806)。然后进入接收数据状态，将数据传送到 IP 或 ARP 模块，等待 IP 或 ARP 数据接收完毕，再接收 CRC 数据。并在接收数据的过程中对接收的数据进行 CRC 处理，将结果与接收到的 CRC 数据进行对比，判断数据是否接收正确，正确则结束，错误则进入 ERROR 状态。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位
rx_en	input	1	开始接受使能
mac_rx_datain	input	8	接受的数据
checksum_err	input	1	IP 层校验错误信号
ip_rx_end	input	1	IP 接受结束
arp_rx_end	input	1	ARP 接受结束
ip_rx_req	output	1	IP 接受请求
arp_rx_req	input	1	请求 ARP 接收
mac_rx_dataout	output	8	MAC 层接收数据输出给 IP 或 ARP
mac_rec_error	output	1	MAC 层接收错误
mac_rx_dest_mac_addr	output	48	MAC 接收的目的 IP 地址
mac_rx_sour_mac_addr	output	48	MAC 接收的源 IP 地址

11.4.2.2 ARP 接收

工程中的 arp_rx.v 为 ARP 接收模块，实现 ARP 数据接收，在 IDLE 状态下，接收到从 MAC 层发来的 arp_rx_req 信号，进入 ARP 接收状态，在此状态下，提取出目的 MAC 地址，源 MAC 地址，目的 IP 地址，源 IP 地址，并判断操作码 OP 是请求还是应答。如果是请求，则判断接收到的目的 IP 地址是否为本机地址，如果是，发送应答请求信号 arp_reply_req，如果不是，则忽略。如果 OP 是应答，则判断接收到的目的 IP 地址及目的 MAC 地址是否与本机一致，如果是，则拉高 arp_found 信号，表明接收到了对方的地址。并将对方的 MAC 地址及 IP 地址存入 ARP 缓存中。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位
local_ip_addr	input	32	本地 IP 地址
local_mac_addr	input	48	本地 MAC 地址
arp_rx_data	input	8	ARP 接收数据
arp_rx_req	input	1	ARP 接收请求
arp_rx_end	output	1	ARP 接收完成
arp_reply_ack	input	1	ARP 回复应答
arp_reply_req	output	1	ARP 回复请求
arp_rec_sour_ip_addr	input	32	ARP 接收的源 IP 地址
arp_rec_sour_mac_addr	input	48	ARP 接收的源 MAC 地址
arp_found	output	1	ARP 接收到请求应答正确

11.4.2.3 IP 层接收模块

在工程中，ip_rx 为 IP 层接收模块，实现 IP 层的数据接收，信息提取，并进行校验和检查。首先在 IDLE 状态下，判断从 MAC 层发过来的 ip_rx_req 信号，进入接收 IP 首部状态，先在 REC_HEADER0 提取出首部长度及 IP 总长度，进入 REC_HEADER1 状态，在此状态提取出目的 IP 地址，源 IP 地址，协议类型，根据协议类型发送 udp_rx_req 或 icmp_rx_req。在接收首部的同时进行校验和的检查，将首部接收的所有数据相加，存入 32 位寄存器，再将高 16 位与低 16 位相加，直到高 16 位为 0 ，再将低 16 位取反，判断其是否为 0，如果是 0，则检验正确，否则错误，进入 IDLE 状态，丢弃此帧数据，等待下次接收。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位
local_ip_addr	input	32	本地 IP 地址
local_mac_addr	input	48	本地 MAC 地址
ip_rx_data	input	8	从 MAC 层接收的数据

ip_rx_req	input	1	MAC 层发送的 IP 接收请求信号
mac_rx_dest_mac_addr	input	48	MAC 层接收的目的 MAC 地址
udp_rx_req	output	1	UDP 接收请求信号
icmp_rx_req	output	1	ICMP 接收请求信号
ip_addr_check_error	output	1	地址检查错误信号
upper_layer_data_length	output	16	上层协议的数据长度
ip_total_data_length	output	16	数据总长度
net_protocol	output	8	网络协议号
ip_rec_source_addr	output	32	IP 层接收的源 IP 地址
ip_rec_dest_addr	output	32	IP 层接收的目的 IP 地址
ip_rx_end	output	1	IP 层接收结束
ip_checksum_error	output	1	IP 层校验和检查错误信号

11.4.2.4 UDP 接收

在工程中，udp_rx.v 为 UDP 接收模块，在此模块首先接收 UDP 首部，再接收数据部分，在接收的同时进行 UDP 校验和检查，如果 UDP 数据是奇数个字节，在计算校验和时，在最后一个字节后加上 8'h00，并进行校验和计算。校验方法与 IP 校验和一样，如果校验正确，将拉高 udp_rec_data_valid 信号，表明接收的 UDP 数据有效，否则无效，等待下次接收。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rst_n	input	1	低电平复位
udp_rx_data	input	8	UDP 接收数据
udp_rx_req	input	1	UDP 接收请求
ip_checksum_error	input	1	IP 层校验和检查错误信号
ip_addr_check_error	input	1	地址检查错误信号
udp_rec_rdata	output	8	UDP 接收读数据
udp_rec_data_length	output	16	UDP 接收数据长度
udp_rec_data_valid	output	1	UDP 接收数据有效

11.4.3 其他部分

11.4.3.1 ICMP 应答

在工程中，icmp_reply.v 实现 ping 功能，首先接收其他设备发过来的 icmp 数据，判断类型是否是回送请求（ECHO REQUEST），如果是，将数据存入 RAM，并计算校验和，判断校验和是否正确，如果正确则进入发送状态，将数据发送出去。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位
mac_send_end	input	1	Mac 发送结束信号
ip_tx_ack	input	1	IP 发送应答
icmp_rx_data	input	8	ICMP 接收数据
icmp_rx_req	input	1	ICMP 接收请求
icmp_rev_error	input	1	接收错误信号
upper_layer_data_length	input	16	上层协议长度
icmp_data_req	input	1	请求 ICMP 数据
icmp_tx_ready	output	1	ICMP 发送准备好
icmp_tx_data	output	8	ICMP 发送数据
icmp_tx_end	output	1	ICMP 发送结束
icmp_tx_req	output	1	ICMP 发送请求

11.4.3.2 ARP 缓存

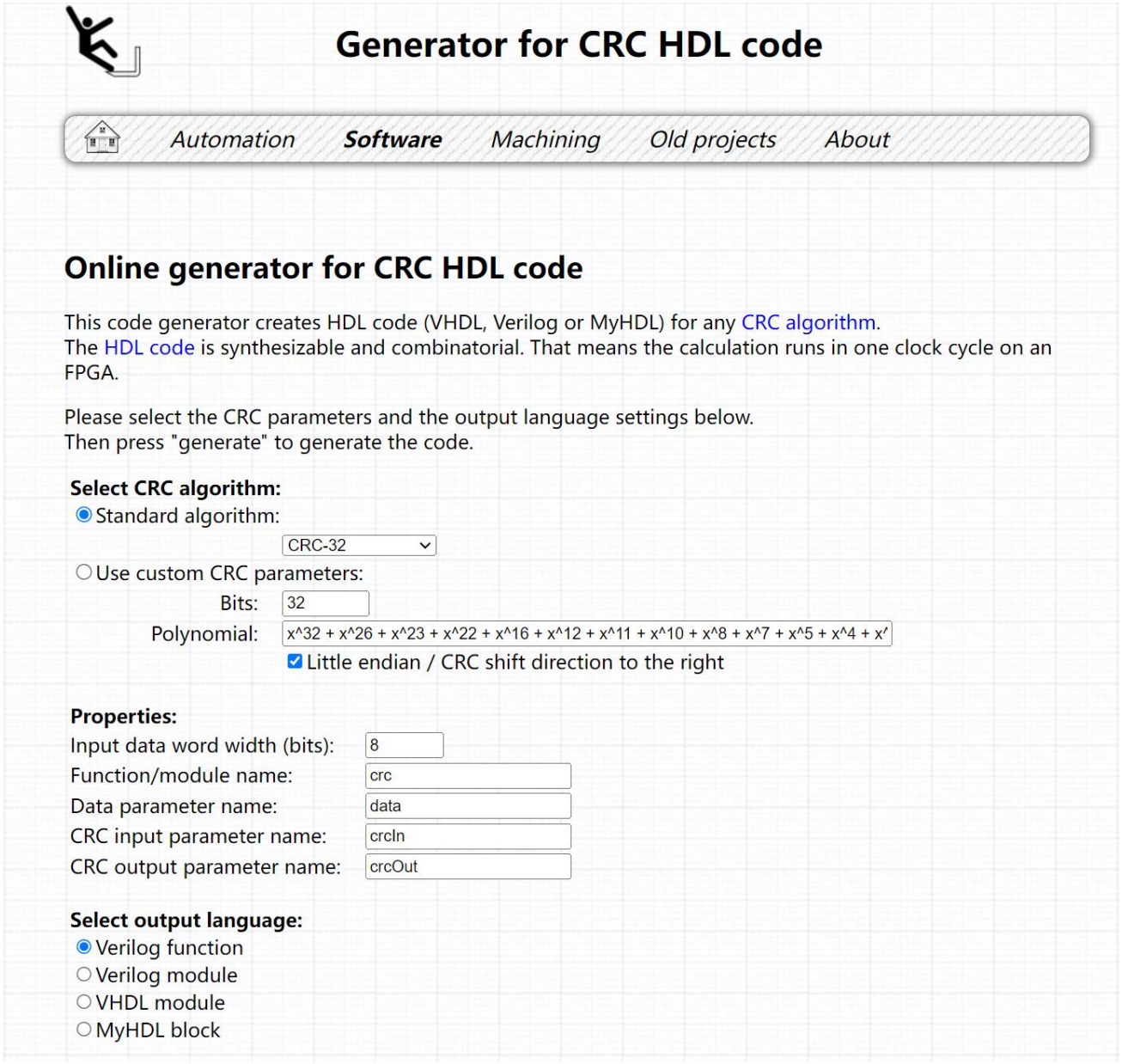
在工程中，arp_cache.v 为 arp 缓存模块，将接收到的其他设备 IP 地址和 MAC 地址缓存，在发送数据之前，查询目的地址是否存在，如果不存在，则向目的地址发送 ARP 请求，等待应答。在设计文件中，只做了一个缓存空间，如果有需要，可扩展。

信号名称	方向	位宽	说明
clk	input	1	系统时钟
rstn	input	1	低电平复位

arp_found	input	1	ARP 接收到回复正确
arp_rec_source_ip_addr	input	32	ARP 接收的源 IP 地址
arp_rec_source_mac_addr	input	48	ARP 接收的源 MAC 地址
dest_ip_addr	input	32	目的 IP 地址
dest_mac_addr	output	48	目的 MAC 地址
mac_not_exist	output	1	目的地址对应的 MAC 地址不存在

11.4.3.3 CRC 校验模块(crc.v)

CRC32 校验是在目标 MAC 地址开始计算的，一直计算到一个包的最后一个数据为止。一些网站可以自动生成 CRC 算法的 verilog 文件：https://bues.ch/cms/hacking/crcgen.html



11.5 实验现象

用网线连接 MES22GP 开发板网口和 PC 端网口；

设置接收端(PC 端)IP 地址为 192.168.0.136，设置发送端(开发板)IP 地址为 192.168.0.5 如下图：

```
10  // Target Devices:
11  // Tool Versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module ethernet_test#(
24      parameter       LOCAL_MAC = 48'h11_11_11_11_11_11,
25      parameter       LOCAL_IP  = 32'hC0_A8_00_05,//  192.168.0.5
26      parameter       LOCL_PORT = 16'h1F90,
27
28      parameter       DEST_IP   = 32'hC0_A8_00_88, // 192.168.0.137
29      parameter       DEST_PORT = 16'h1F90
30  )(
31      input         clk_50m,
32      output reg    led,
33      output        phy_rstn,
```

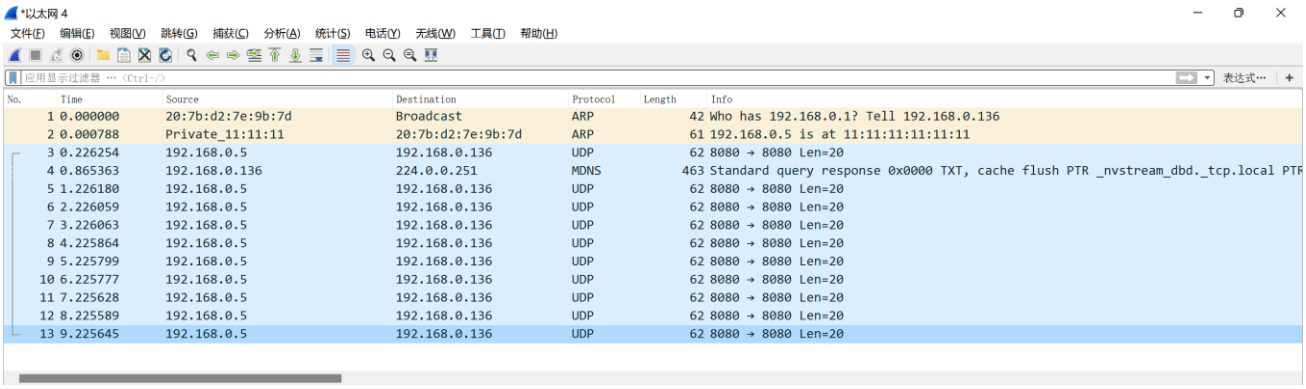
将程序下载到开发板后，便可以看到已连接网线的网口对应 LED 灯规律性闪烁或常亮：

信号名称	LED 编号	参考说明
led	1	闪烁
phy_rstn	2	rgmii_clk 模块 pll lock 锁定指示

本次实验通过 Wireshark 软件抓包验证数据链路是否正常连接以及数据传输是否正常。资料包中 Wireshark 安装包目录如下：

MES22GP_v1\5_Software\网络调试助手\Wireshark-win32-2.4.1.0.exe

PC 端打开 Wireshark 软件，烧录重新后进行捕获数据报可以看到如下所示的交互过程。



成功建立连接后会持续发送数据报“www.meyesemi.com”。

```
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.0.5, Dst: 192.168.0.136
v User Datagram Protocol, Src Port: 8080, Dst Port: 8080
    Source Port: 8080
    Destination Port: 8080
    Length: 28
    [Checksum: [missing]]
    [Checksum Status: Not present]
    [Stream index: 0]
v Data (20 bytes)
    Data: 7777772e6d65796573656d692e636f6d2020200a
```

```
0000  20 7b d2 7e 9b 7d 11 11  11 11 11 11 08 00 45 00    {.~.}.. ......E.
0010  00 30 00 0b 40 00 80 11  78 d4 c0 a8 00 05 c0 a8   .0..@... x.......
0020  00 88 1f 90 1f 90 00 1c  00 00 77 77 77 2e 6d 65   ........ ..www.me
0030  79 65 73 65 6d 69 2e 63  6f 6d 20 20 20 0a         yesemi.c om   .
```