以太网模块问题及优化

基于FPGA的以太网相关文章导航，点击查看。

在使用以太网　发送摄像头数据到PC端时，发现了之前编写以太网发送模块的两个小问题，对实际使用有一定的影响，本文将问题指出来并进行修正。

问题一：为了方便设计，工程直接把以太网发送模块的忙闲指示信号输出作为用户端口的udp发送模块忙闲指示信号，存在的问题是用户把udp发送使能信号拉高后，udp发送模块忙闲指示信号会滞后一个时钟周期　才能拉低，并不能马上拉低。

如下图所示，udp_tx_en是提供给用户的udp发送使能信号，拉高该信号后，udp发送模块忙闲指示信号udp_tx_rdy会延迟一个时钟才会被拉低。这对于我这种喜欢通过计数rdy拉高时钟个数来记录发送报文个数的人来说，很不方便。

tx_rdy信号是以太网发送模块的忙闲指示信号，该模块最初的设计就是两个信号复用，eth_tx_start是以太网发送模块的发送使能信号。

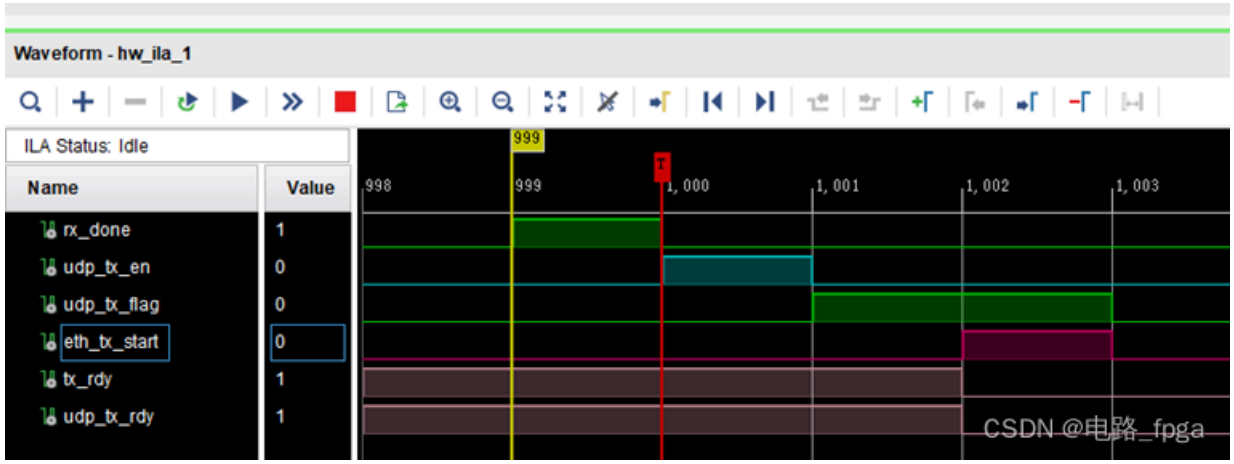

图1 rdy信号问题

造成上述延时的原因是因为该以太网模块在发送用户数据的同时，可能需要发送来自PC端的ARP请求和回显请求的应答数据报文，所以需要进行调度，eth_tx_start至少会滞后eth_tx_en两个时钟周期。


54 //把UDP发送使能信号暂存，可能当前发送模块处于工作状态;
55 always@(posedge clk)begin
56 if(rst_n==1'b0)begin//初始值为0;
57 udp_tx_flag <= 1'b0;
58 end
59 else if(udp_tx_en)begin
60 udp_tx_flag <= 1'b1;
61 end
62 else if(eth_tx_start && (&eth_tx_type))begin
63 udp_tx_flag <= 1'b0;
64 end
65 end
CSDN @电路_fpga

图2 信号联系

```
98      //开始发送以太网帧;
99      always@(posedge clk)begin
100         if(rst_n==1'b0)begin//初始值为0;
101             eth_tx_start <= 1'b0;
102             eth_tx_type <= 2'd0;
103             arp_tx_type <= 1'b0;
104             icmp_tx_type <= 8'd0;
105             icmp_tx_code <= 8'd0;
106             icmp_tx_id <= 16'd0;
107             icmp_tx_seq <= 16'd0;
108             iudp_tx_byte_num <= 16'd0;
109         end
110         //接收到ARP的请求数据报时，把开始发送信号拉高;
111         else if(arp_tx_flag && tx_rdy)begin
112             eth_tx_start <= 1'b1;
113             eth_tx_type <= 2'd1;
114             arp_tx_type <= arp_req_r ? 1'b0 : 1'b1;//发送ARP应答报文;
115         end//当接收到ICMP回显请求时，把开始发送信号拉高;
116         else if(icmp_tx_flag && tx_rdy)begin
117             eth_tx_start <= 1'b1;
118             eth_tx_type <= 2'd2;
119             icmp_tx_type <= 8'd0;//发送ICMP回显应答数据报文。
120             icmp_tx_code <= 8'd0;
121             icmp_tx_id  <= icmp_rx_id;//将回显请求的的ID传回去。
122             icmp_tx_seq <= icmp_rx_seq;
123             iudp_tx_byte_num <= iudp_rx_byte_num;
124         end//当需要发送udp数据时，把开始发送信号拉高;
125         else if(udp_tx_flag && tx_rdy)begin
126             eth_tx_start <= 1'b1;
127             eth_tx_type <= 2'd3;
128             iudp_tx_byte_num <= udp_tx_data_num;
129         end//如果检测到模块处于空闲状态，则将开始信号拉低。
130         else begin
131             eth_tx_start <= 1'b0;
132         end
133     end
```


图3 信号联系

修改方式其实很简单，只需要在以太网控制模块中重新生成udp_tx_rdy信号即可，如下图所示。

```
228     //当有数据需要发送时，用户接口不能传输数据;
229     always@(*)begin
230         if(arp_tx_flag || icmp_tx_flag || udp_tx_flag || (~tx_rdy) || udp_tx_en)
231             udp_tx_rdy = 1'b0;
232         else
233             udp_tx_rdy = 1'b1;
234     end
```

图4 修改代码

修改之后，使用ila抓取信号如下所示，当udp_tx_en拉高时，udp_tx_rdy会立即拉低，不会有延时，用户就可以通过udp_tx_rdy的状态判断能不能发送udp使能信号了。

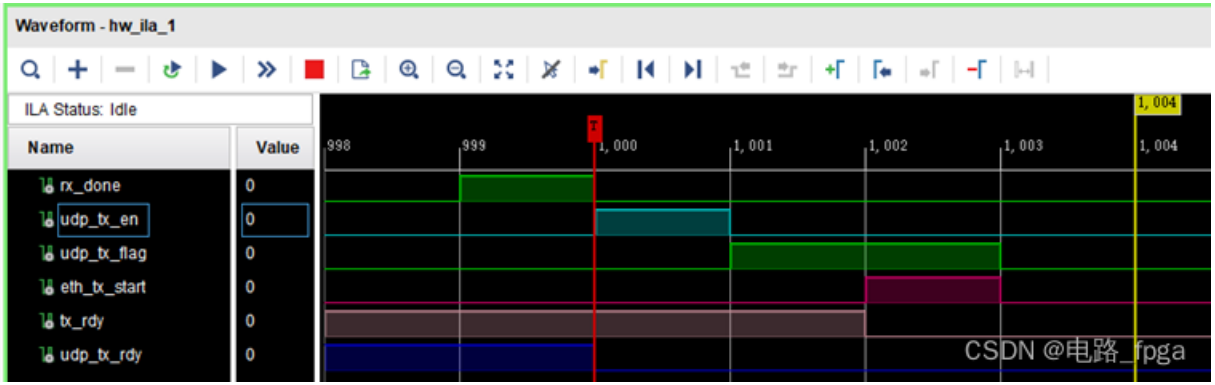

图5 忙闲指示信号修改后时序

问题二：前文可知，在udp_tx_en拉高后，需要至少延时两个时钟eth_tx_start才会被拉高，但是有一个信号udp_tx_data_num没有进行寄存，会出现数据丢失。造成用户发送数据包的长度错乱，因此在控制模块中增加对需要发送报文长度的寄存器，对应代码如下所示：

```
68    //把UDP发送使能信号暂存，可能当前发送模块处于工作状态;
69    always@(posedge clk)begin
70        if(rst_n==1'b0)begin//初始值为0;
71            udp_tx_flag <= 1'b0;
72            udp_tx_data_num_r <= 16'd0;
73        end
74        else if(udp_tx_en)begin
75            udp_tx_flag <= 1'b1;
76            udp_tx_data_num_r <= udp_tx_data_num;
77        end
78        else if(eth_tx_start && (&eth_tx_type))begin
79            udp_tx_flag <= 1'b0;
80        end
81    end
```


图6 寄存udp数据长度

由于udp_tx_en与udp_tx_data_num对齐，而eth_tx_start滞后eth_tx_en至少两个时钟，所以不需要修改其余代码。

然后综合工程，进行arp应答和回显请求测试，结果如下所示，证明以太网数据链路没有问题。

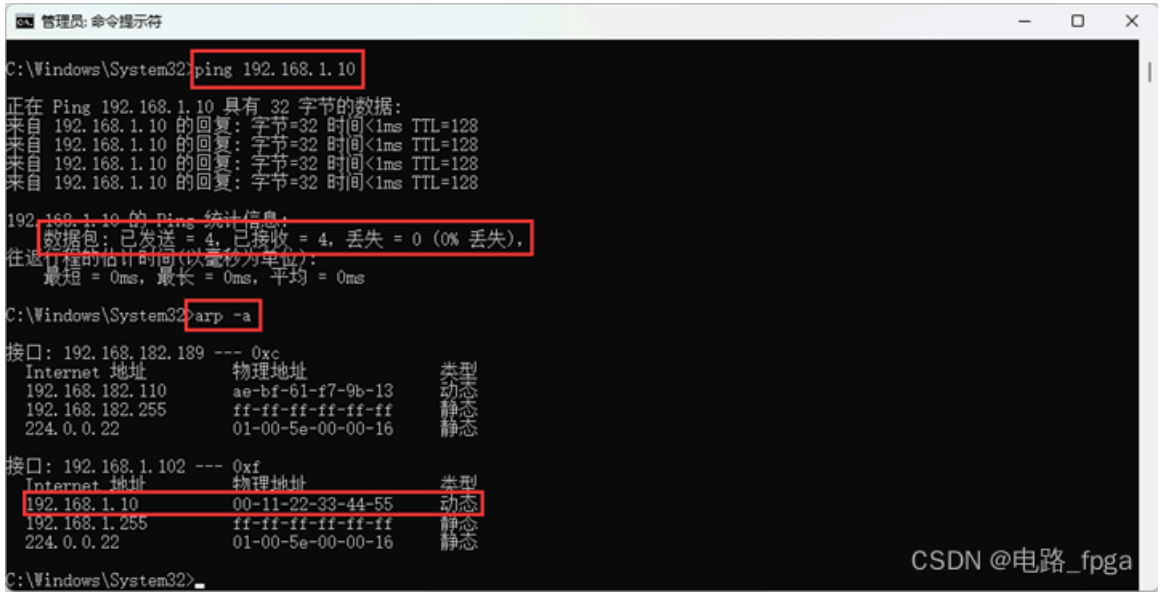

图7 回显应答

然后通过网络调试助手进行回环测试，进行循环发送测试，最后网络调试助手接收和发送的数据报文保持一致，由于这上位机发送报文间隔最小为1ms，就只能测试这么多了。

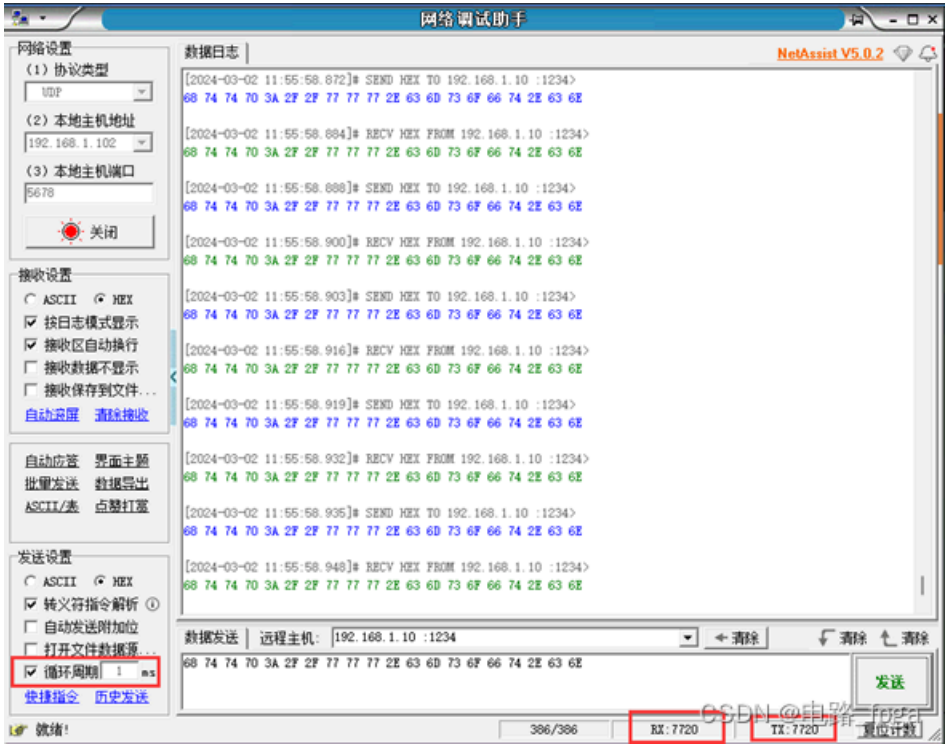

图8 UDP回环测试

上述两个问题会在特定的时候影响发送数据，所以做出了修改，在移植该工程的时候，还是需要把时钟简单约束一下，不然可能会有一些奇怪的问题。

修改后的工程依旧可以在公众号后台回复“**基于FPGA实用UDP设计**”（不包括引号）获取，只不过版本变为1。如果后续出现问题依旧会在该文件夹下更新工程。

如果对文章内容理解有疑惑或者对代码不理解，可以在评论区或者后台留言，看到后均会回复！

如果本文对您有帮助，还请多多点赞👍、评论💬和收藏⭐！您的支持是我更新的最大动力！将持续更新工程！