湖南大學

HUNAN UNIVERSITY

路由与交换机

实验报告

小组成员： 计科1802张继伟，谢正宇

目录

实验一

一、 实验目的

1、熟悉 NetMagic08 的硬件编程方式；
2、基于 NetMagic08 搭建实验环境，包括 NetMagic08 的安装、Quartus 与 NetMagic08 的对接
3、使用 Quartus 设计硬件逻辑
4、了解 FPGA 编程基础

二、 实验内容

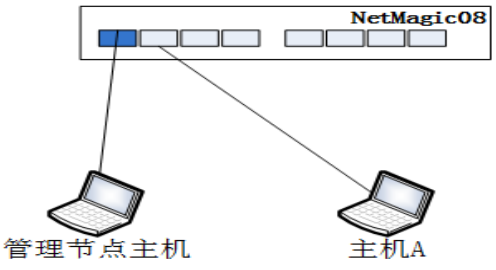
1、基础要求：

在 UM_my/UM.v 中使用 Verilog 语言编写一个模块让 NetMagic08 实现 2 口进 3 口出的基本功能。

2. 扩展要求：

在实现了实验内容 1 后，改写程序实现从 NetMagic08 的 2 端口进入的包转发至所有端口；所有端口进入的包都转发到 2 号端口。

三、 实验环境



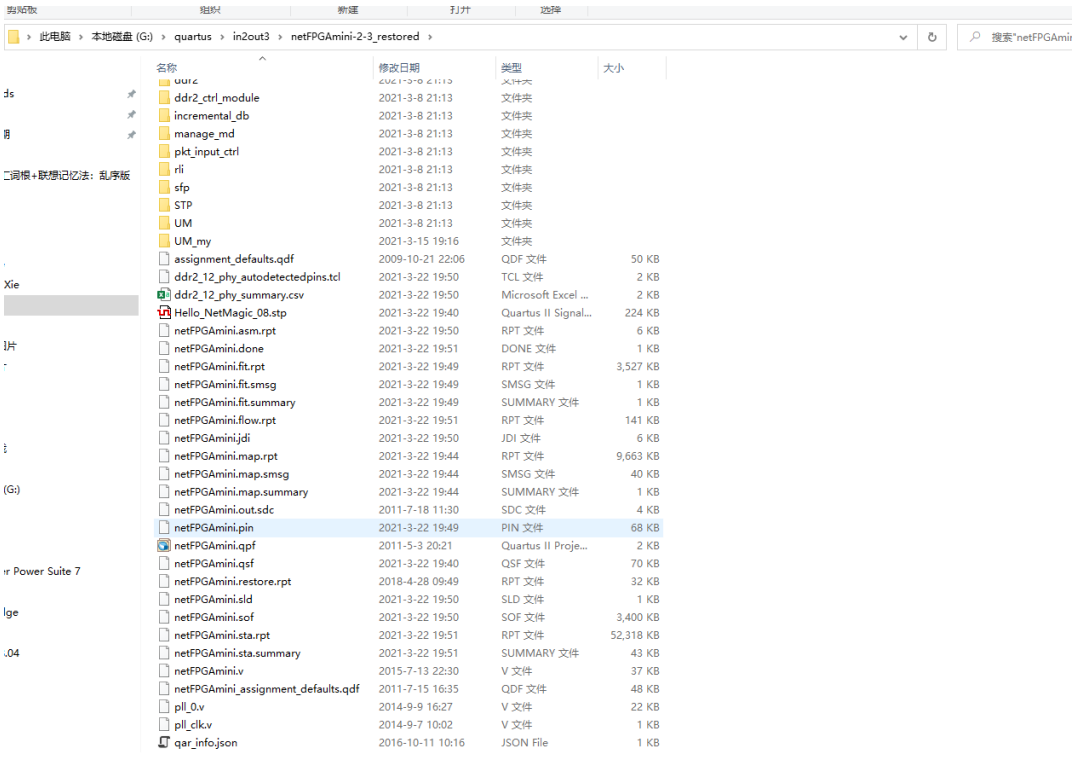
1. 1 台管理节点主机；1 台主机 A；（分别连接到 2 口和 3 口）；
2. 2 根网线；
3. NetMagic08 开发平台；
4. 软件 Quartus 16。主机及网络详细配置参照附带的实验环境拓扑及软件配置文档。

2 进 3 出端口配置

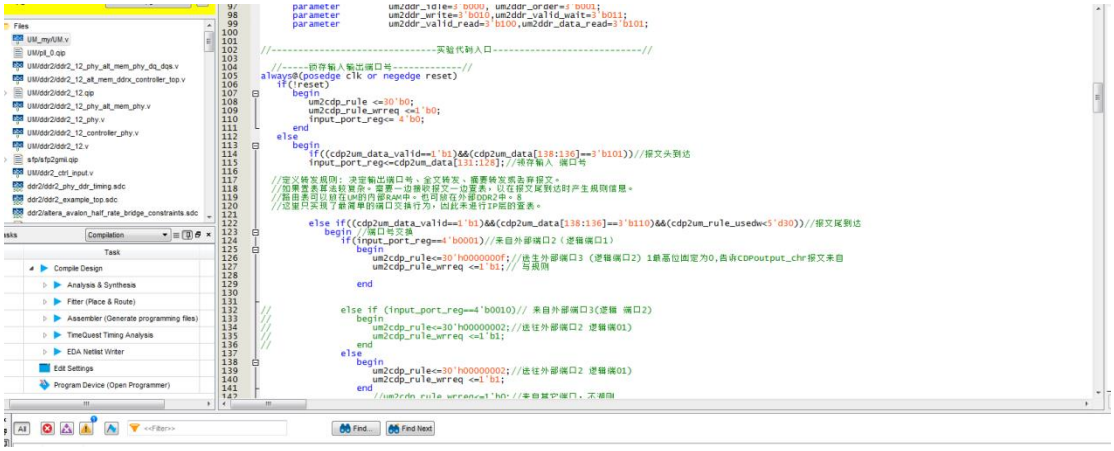
四、 项目 1 实验步骤

我们首先完成基础要求，实现 2 进 3 出的功能配置。

step1.使用 Quartus 打开 in2out3 目录下的硬件工程项目 netFPGAmini-2-3.qsf。



step2. 在工程项目中，打开文件列表对 UM.v 文件进行编写（用户模块文件）



Step3. 在 UM.v 文件中增加一个新的处理逻辑，在一个 always@(posedge clk or negedge reset)逻辑中完成端口 2 进端口 3 出的硬件设计。

相关代码如下图：

```
//------------------锁存输入输出端口号实验代码在此输入-----------//

always@(posedge clk or negedge reset)
   if(!reset)
      begin
         um2cdp_rule <=30'b0;
         um2cdp_rule_wrreq <= 1'b0;
         input_port_reg <= 4'b0;
      end
   else
      begin
         if((cdp2um_data_valid==1'b1)&&(cdp2um_data[138:136]==3'b101)) //报文头到达
            input_port_reg <= cdp2um_data[131:128]; //锁存输入端口号
//定义转发规则:决定输出端口号、全文转发、摘要转发或丢弃报文。
//如果查表算法较复杂，需要一边接收报文一边查表，以在报文尾到达时产生规
//测信息。路由表可以放在UM的内部RAM中，也可以放在外部DDR2中。
//这里只实现了最简单的端口交换行为，因此未进行IP层的查表。
         else if((cdp2um_data_valid==1'b1)&&(cdp2um_data[138:136]==3'b110)&&(cdp2um_rule_usedw<5'd30)) //报文尾到达
            begin //端口号交换
               if(input_port_reg==4'b0001) //来自外部端口2（逻辑端口1）
                  begin
                     um2cdp_rule <=30'h00000004; //送往外部端口3（逻辑端口2）1最高位固定为0，告诉CDP output_ ctr报文来自UM
                     um2cdp_rule_wrreq <=1'b1; //与规则
                  end
               else if(input_port_reg==4'b0010) //来自外部端口3（逻辑端口2）
                  begin
                     um2cdp_rule <=30'h00000002; //送往外部端口2（逻辑端口1）
                     um2cdp_rule_wrreq <=1'b1;
                  end
               else
                  um2cdp_rule_wrreq <=1'b0; //来自其它端口，不与规则
            end
         else
            um2cdp_rule_wrreq <= 1'b0;
      end
```

Step4.关键代码解释：

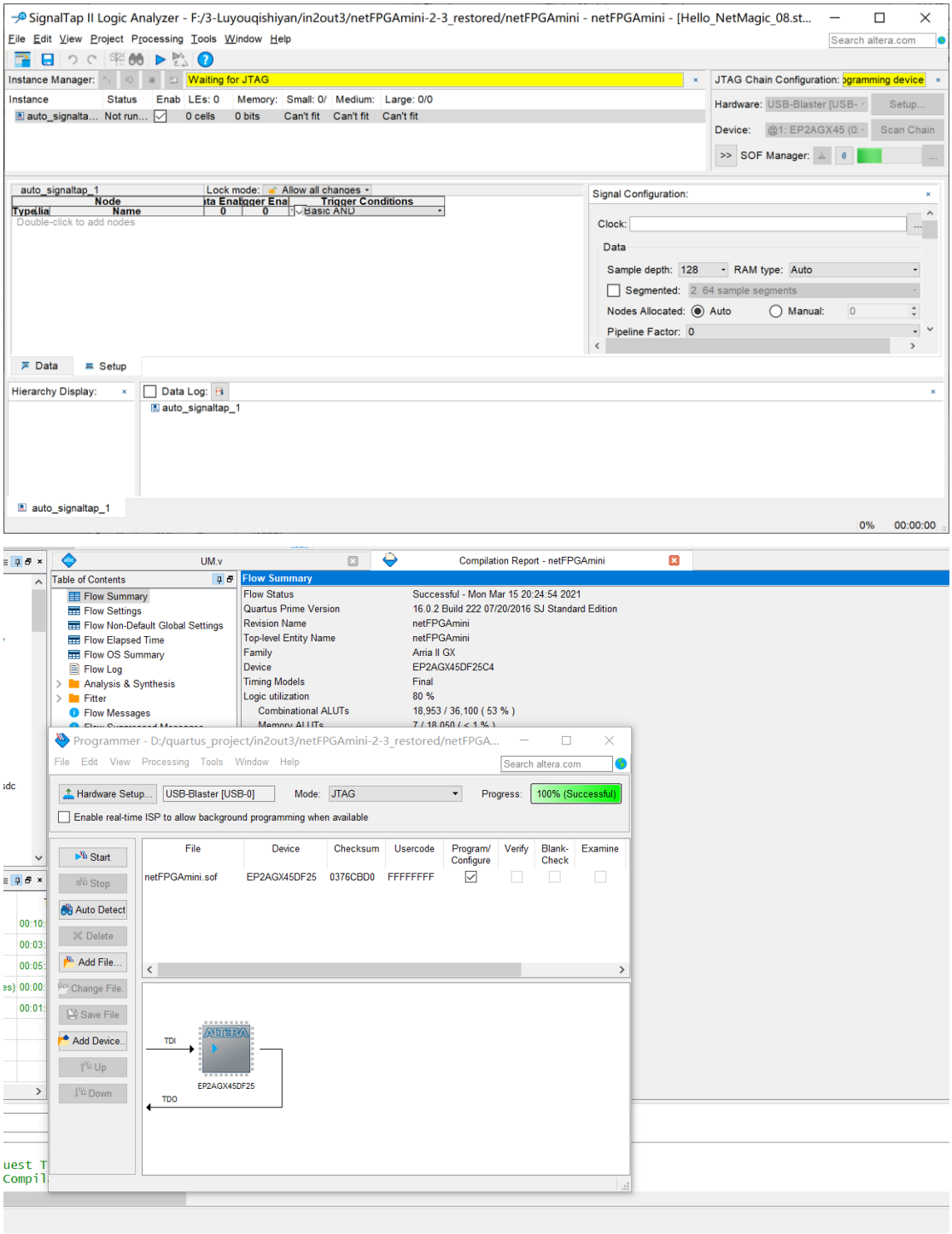
```
if(input_port_reg==4'b0001)//来自外部端口2（逻辑端口1）
   begin
      um2cdp_rule<=30'h00000001;//送生外部端口3（逻辑端口2）1最高位固定为0,告诉CDPoutput_chr报文来自
      um2cdp_rule_wrreq <=1'b1;// 与规则

   end


else if (input_port_reg==4'b0010)// 来自外部端口3(逻辑 端口2)
   begin
      um2cdp_rule<=30'h00000002;//送往外部端口2 逻辑端01)
      um2cdp_rule_wrreq <=1'b1;
   end
else
   begin
      um2cdp_rule<=30'h00000002;//送往外部端口2 逻辑端01)
      um2cdp_rule_wrreq <=1'b1;
   end
      um2cdp_rule_wrreq<=1'b0;//来自其它端口，不湖则
```

主要的就是内部的 if 判断语句首先判断输入信息是否是来自端口 2 的，如果是来自端口 2 则将信息转发到端口 3；同理判断输入信息是否来自端口 3，若果是来自端口 3 则将信息转发到端口 2；最后，对于其他情况，我们不转发。

Step5．点击编译命令，对代码进行编译，软件会完成硬件电路的设计和优化，生成和工程名同名的 sof 文件。

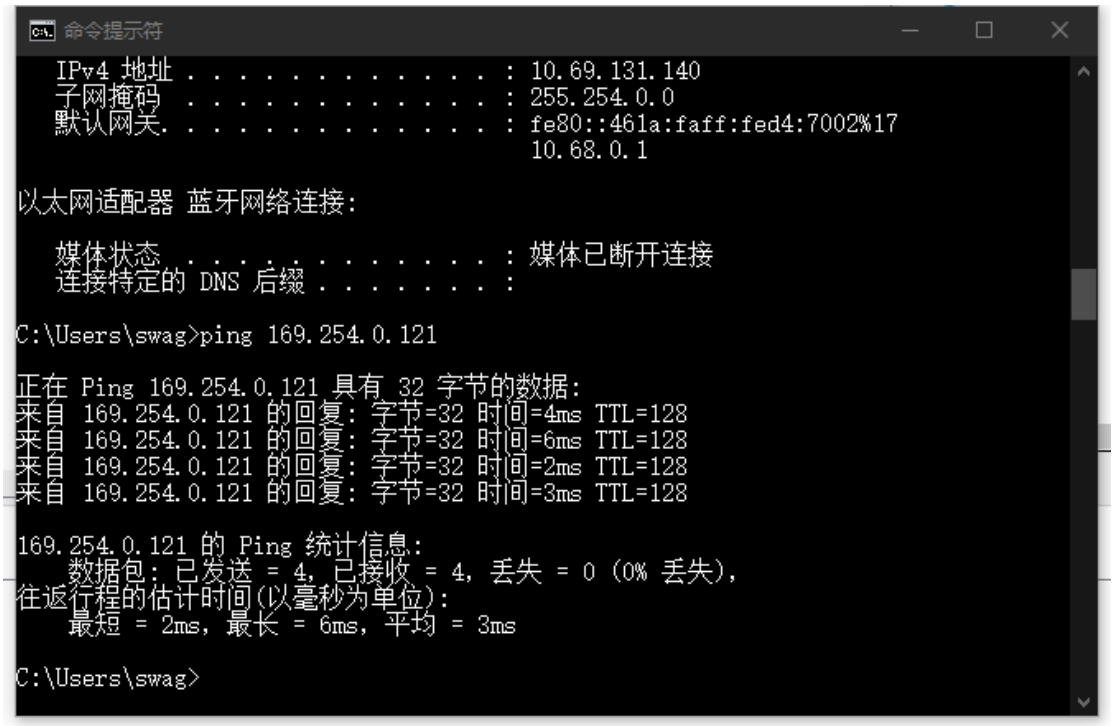
点击图中的蓝色箭头即可进行编译。编译时间可能会较长。

Step6．将 sof 文件下载到 NetMagic08 中进行硬件调试

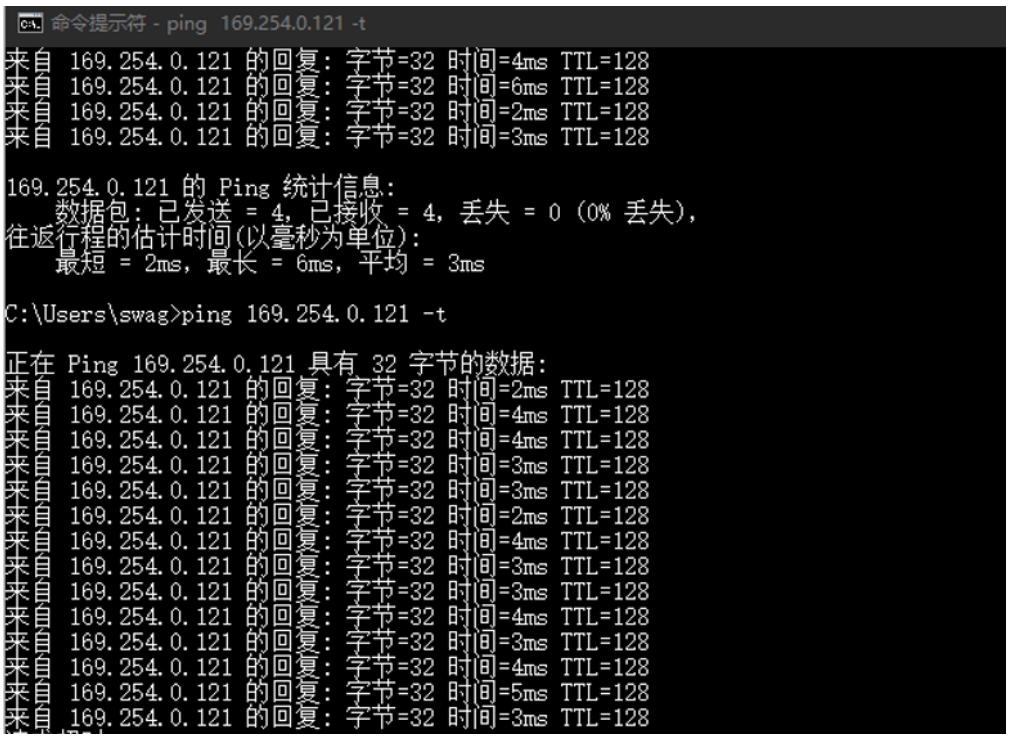


下载完毕后，我们使用网线将两台 PC 与 NetMagic 设备相连接，在设置了各自的静态 IP 地址后，使用命令行名 ping 命令，查看是否连通。

我们将两台计算机的一台 IP 设置为：169.254.0.121；另一台设置为：169.254.141.235



从 169.254.141.235 顺利 ping 通 169.254.0.121（此时 169.254.141.235 连接 2 口，169.254.0.121 连接 3 口）



再从 169.254.141.235ping169.254.0.121，发现顺利 ping 通。

断开网口，发现请求超时

至此项目实现 2 进 3 出的代码顺利调试成功。

N 进 N 出端口配置

在完成了项目 1 中的 2 进 3 出的代码后，我们追求更进一步的配置功能。实现 N 进 N 出，即实现 N 号端口连通到其他所有 3 个端口（当然也可以是其他某几个端口）

五、 项目 2 实验步骤

修改代码如下：

```
//------------------锁存输入输出端口号实验代码在此输入N进N出-----------//
always@(posedge clk or negedge reset)
   if(!reset)
      begin
         um2cdp_rule<=30'b0;
         um2cdp_rule_wrreq<=1'b0;
         input_port_reg<=4'b0;
      end
   else
      begin
         if((cdp2um_data_valid==1'b1)&&(cdp2um_data[138:136]==3'b101))
            input_port_reg<=cdp2um_data[131:128];
         else if((cdp2um_data_valid==1'b1)&&(cdp2um_data[138:136]==3'b110)&&(cdp2um_rule_usedw<5'd30))
            begin
               if(input_port_reg==4'b0001)
                  begin
                     um2cdp_rule<=30'h0000000d;
                     um2cdp_rule_wrreq<=1'b1;
                  end
               else if(input_port_reg==4'b0010) //来自外部端口3 (逻辑端口2)
                  begin
                     um2cdp_rule <=30'h0000000b; //送往外部端口2 (逻辑端口1)
                     um2cdp_rule_wrreq <=1'b1;
                  end
               else if(input_port_reg==4'b0011) //来自外部端口4 (逻辑端口3)
                  begin
                     um2cdp_rule <=30'h00000007;
                     um2cdp_rule_wrreq <=1'b1;
                  end
               else if(input_port_reg==4'b0000) //来自外部端口1 (逻辑端口0)
                  begin
                     um2cdp_rule <=30'h0000000e;
                     um2cdp_rule_wrreq <=1'b1;
                  end
               else
                  um2cdp_rule_wrreq <=1'b0; //来自其它端口,不写规则
            end
         else
            um2cdp_rule_wrreq <= 1'b0;
      end
```

可以看到，主要是对端口号 if 判断分支进行了修改。

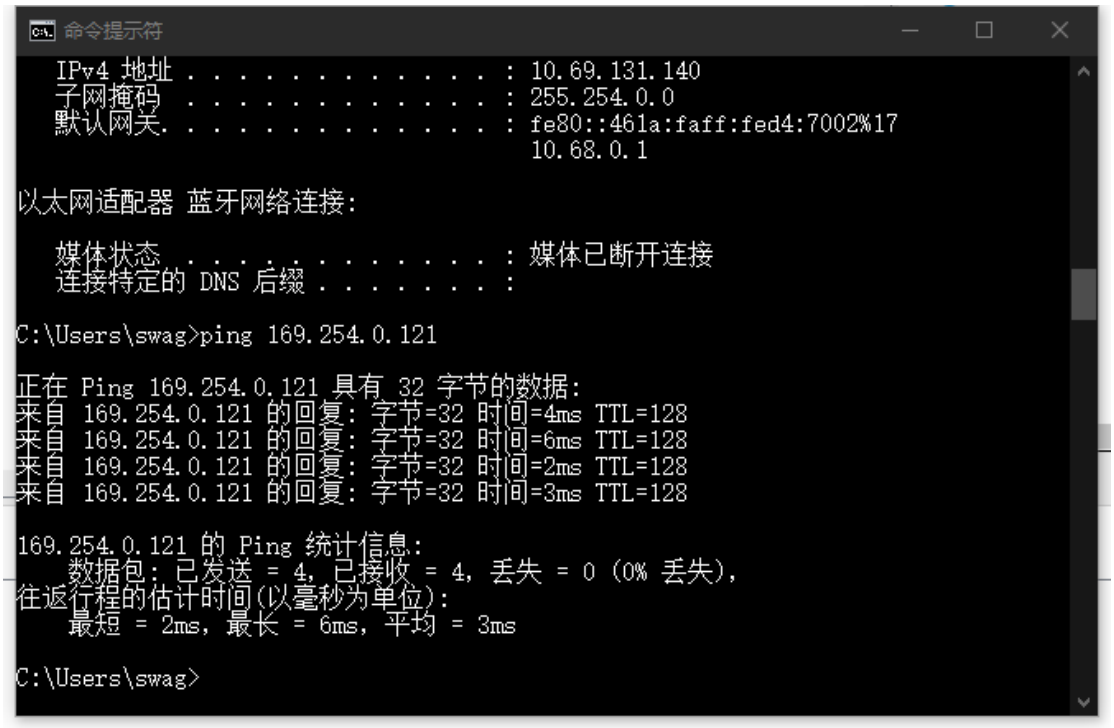
首先，从 2 号端口进来的数据（b0001，二进制的 1）将会被发送到 1、3、4 三个端口（h0000000d，最后一位相当于二进制的 1101），这样，只要能够连接到相应的目的主机，就能实现连接。

同理，从 3 号端口进来的数据（b0010，二进制的 2）将会被发送到 1、2、4 三个端口（h0000000b，最后一位相当于二进制的 1011），这样，只要能够连接到相应的目的主机，就能实现连接。

同理，从 4 号端口进来的数据（b0011，二进制的 3）将会被发送到 1、2、3 三个端口（h00000007，最后一位相当于二进制的 0111），这样，只要能够连接到相应的目的主机，就能实现连接。

同理，从 1 号端口进来的数据（b0000，二进制的 0）将会被发送到 2、3、4 三个端口（h0000000e，最后一位相当于二进制的 1110），这样，只要能够连接到相应的目的主机，就能实现连接。

相当于我这里实现的是一个广播，任何两个口之间都可以实现互联（项目 2 验收要求）。后四位从左到右（从高到低）依次代表 4、3、2、1 号端口。
将网线分别插入 2，3 号端口：



实验成功，证明此时 2 号和 3 号端口能成功连通。
现在我们将线连接在 1 号端口和 4 号端口：
再在两台 PC 机上使用 ping 命令进行连通：

中间有丢包是以为中途拔掉了网线，然后又插上了。
至此实验已经可以证明成功。
成功实现 n 进 n 出的进阶实验。

六、 实验思考

项目 2 完成后 3 号口和 4 号口能否联通?
答案是能。因为代码内部的 if 判断语句判断的是输入信息是否是来自端口 N 的，如果是则将信息转发到其他端口，也就是说只要是两个不同端口就可以进行联通。

七、 实验思考（个人部分单独完成）

2 进 3 出的设计特别简单，它只能从 2 号端口进入，3 号端口传出，而对其他端口的使用并不会产生传输结果，我们在 2 进 3 出的基础上做出了 n 进 n 出的逻辑设计，改动了判断信息来源和转发这一部分的关键代码，能够做到任意一个端口和其他三个端口的互通。单播的信息的接收和传递只在两个节点之间进行，如互相 ping 时抓取的包，包的 source 和 destination 分别是两台 pc 机。单播在网络中得到了广泛的应用，网络上绝大部分的数据都是以单播的形式传输的，只是一般网络用户不知道而已。例如，你在收发电子邮件、浏览网页时，必须与邮件服务器、Web 服务器建立连接，此时使用的就是单播数据传输方式。而广播，发出点是 R1，目的地址是 ff:ff:ff:ff:ff:ff，也就是所有的地址。广播是主机之间“一对所有”的通讯模式，网络对其中每一台主机发出的信号都进行无条件复制并转发，所有主机都可以接收到所有信息，由于其不用路径选择，所以其网络成本可以很低廉。