

** **HUNAN UNIVERSITY**

课程名

实验报告

|  |  |  |
| --- | --- | --- |
|  | 小组成员： |  |

目录

[实验一：2进3出端口配置 2](#_Toc10336)

[一、 实验目的 2](#_Toc7086)

[二、 实验内容 3](#_Toc2325)

[三、 实验环境 3](#_Toc26397)

[四、 实验步骤 4](#_Toc9665)

[项目1： 4](#_Toc15697)

[五、 实验思考 7](#_Toc8986)

[2进N出端口配置 7](#_Toc14210)

[一、 实验步骤 8](#_Toc1899)

[项目2： 8](#_Toc5372)

[二、 实验思考 13](#_Toc18007)

# 实验一：2进3出端口配置

1. 实验目的

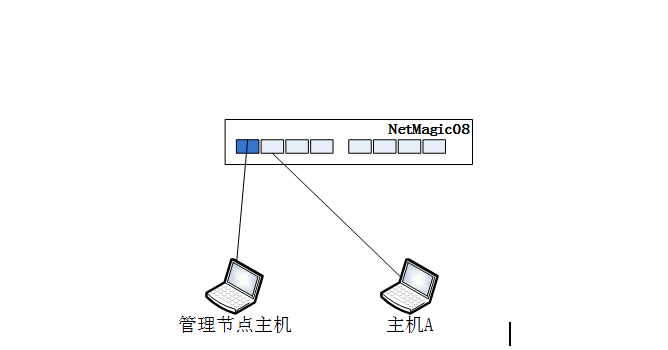
1、熟悉NetMagic08的硬件编程方式；

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

1. 实验内容
2. 基础要求：
   1. 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。
3. 扩展要求：
   1. 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。
4. 实验环境

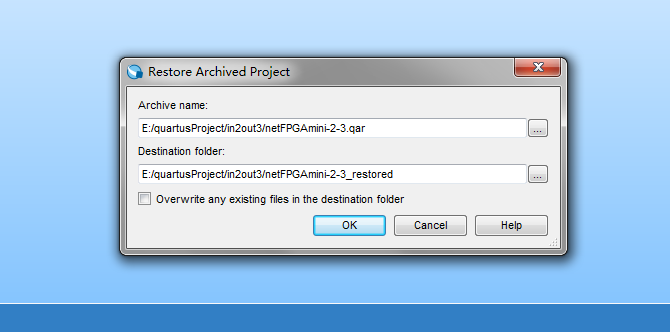


1. 1台管理节点主机；1台主机A；（分别连接到2口和3口）；
2. 2根网线；
3. NetMagic08开发平台；
4. 软件Quartus 16。主机及网络详细配置参照附带的实验环境拓扑及软件配置文档。
5. 实验步骤

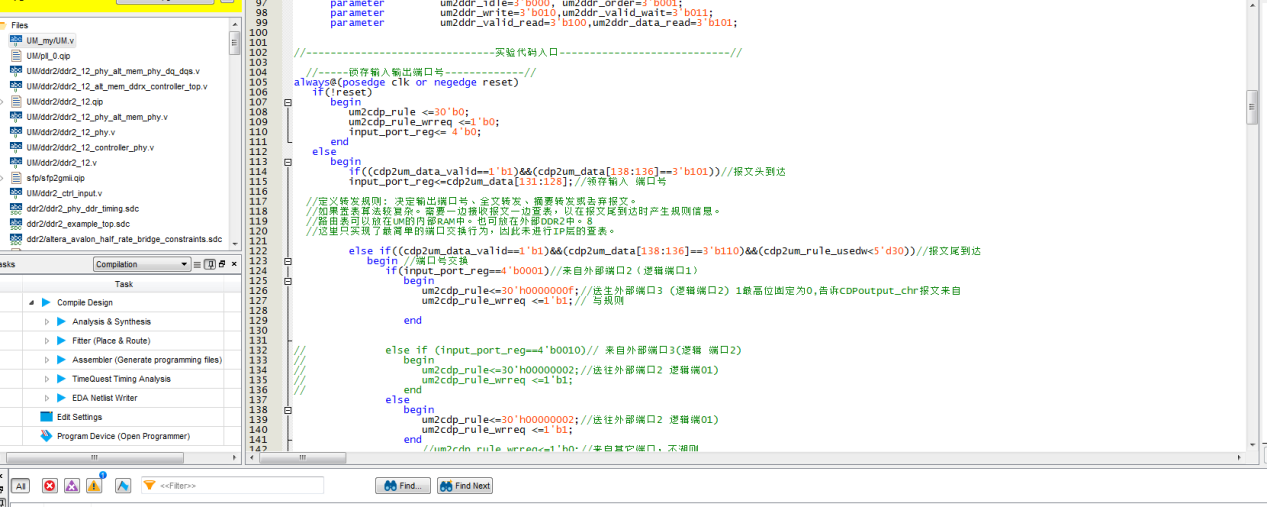
我们首先完成基础要求，实现2进3出的功能配置。

**项目1：**

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



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）

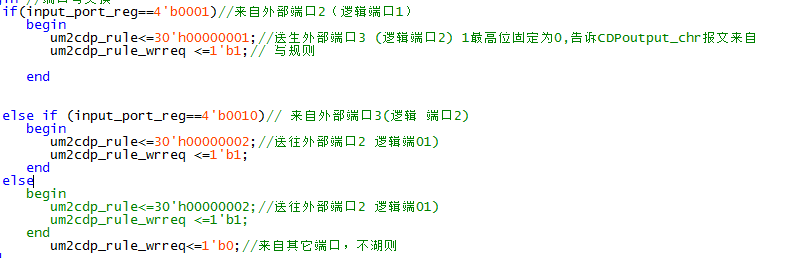


Step3． 在UM.v文件中增加一个新的处理逻辑，在一个always@(posedge clk or negedge reset)逻辑中完成端口2进端口3出的硬件设计。

相关代码如下图：



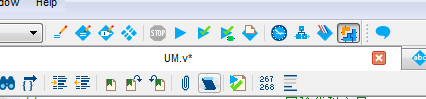
关键代码解释：



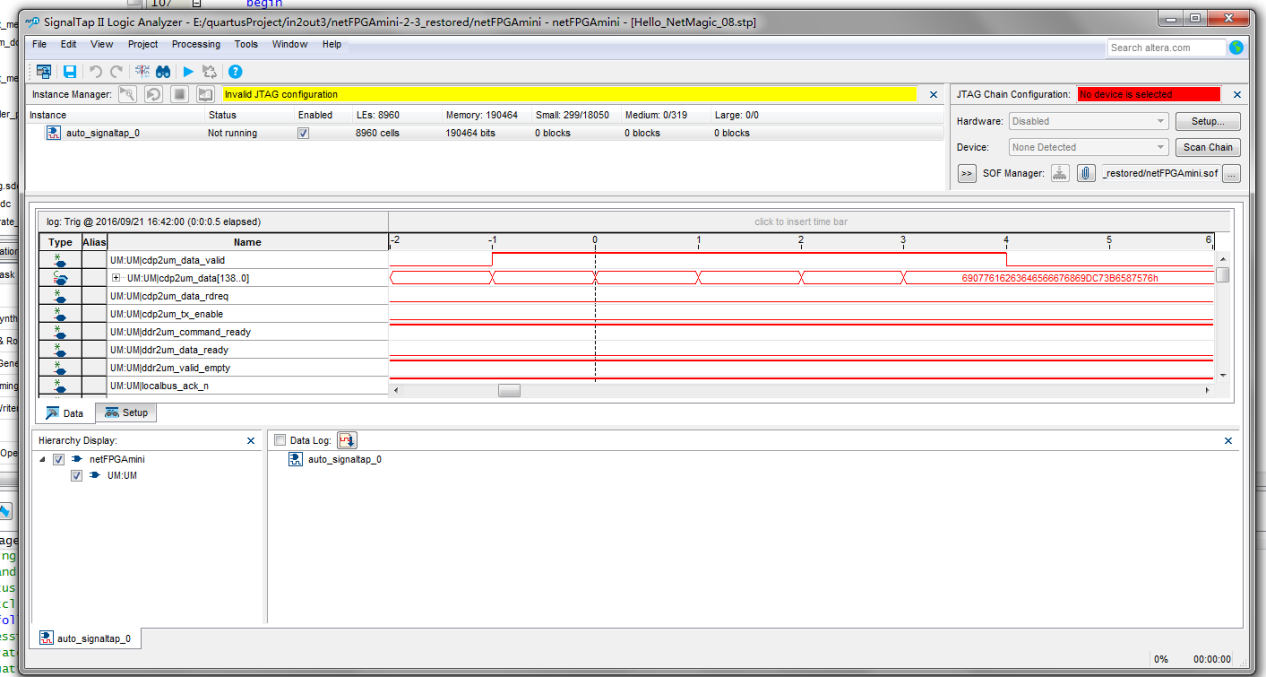
主要的就是内部的if判断语句首先判断输入信息是否是来自端口2的，如果是来自端口2则将信息转发到端口3；同理判断输入信息是否来自端口3，若果是来自端口3则将信息转发到端口2；最后，对于其他情况，我们不转发。

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

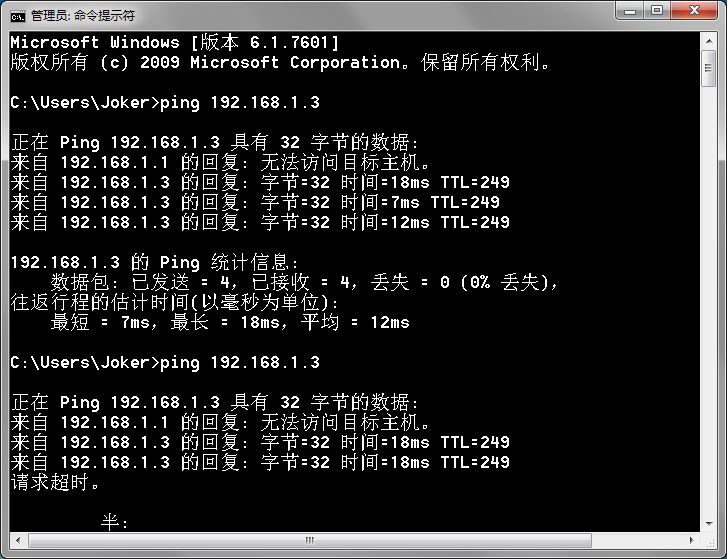


下载完毕后，我们使用网线将两台PC与NetMagic设备相连接，在设置了各自的静态IP地址后，使用命令行名ping命令，查看是否连通。

我们将两台计算机的一台IP设置为：192.168.1.1；另一台设置为：192.168.1.3



从192.168.1.3顺利ping通192.168.1.1（此时192.168.1.1连接2口，192.168.1.3连接3口）



再从192.168.1.1ping192.168.1.3，发现顺利ping通。

至此项目实现2进3出的代码顺利调试成功。

1. 实验思考

项目2完成后3号口和4号口能否联通？

答案是不能。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2（端口3）的，如果是则将信息转发到端口3（端口2），而对于其他情况，并不执行转发。

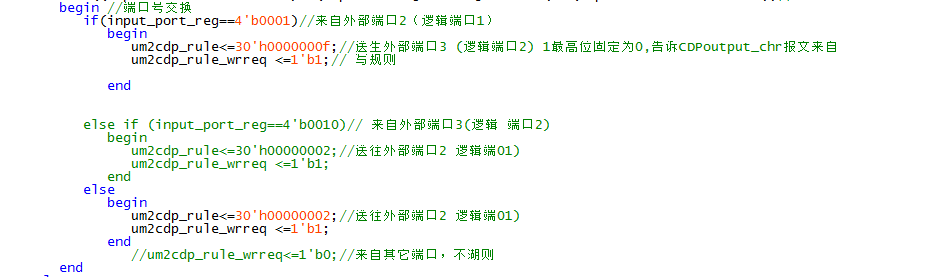
# N进N出端口配置

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）

1. 实验步骤

**项目2：**

我们修改原来实现2进3出的代码，首先要找到其中转发信息的关键部分：



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'h0000000f;**//送生外部端口1234，最高位固定为0,告诉CDPoutput\_chr报文来自

um2cdp\_rule\_wrreq <=1'b1;// 写规则

end

**//这部分代码是全部注释掉我们直接用一个else判断将其他端口的包转到2号端口**

**// 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**

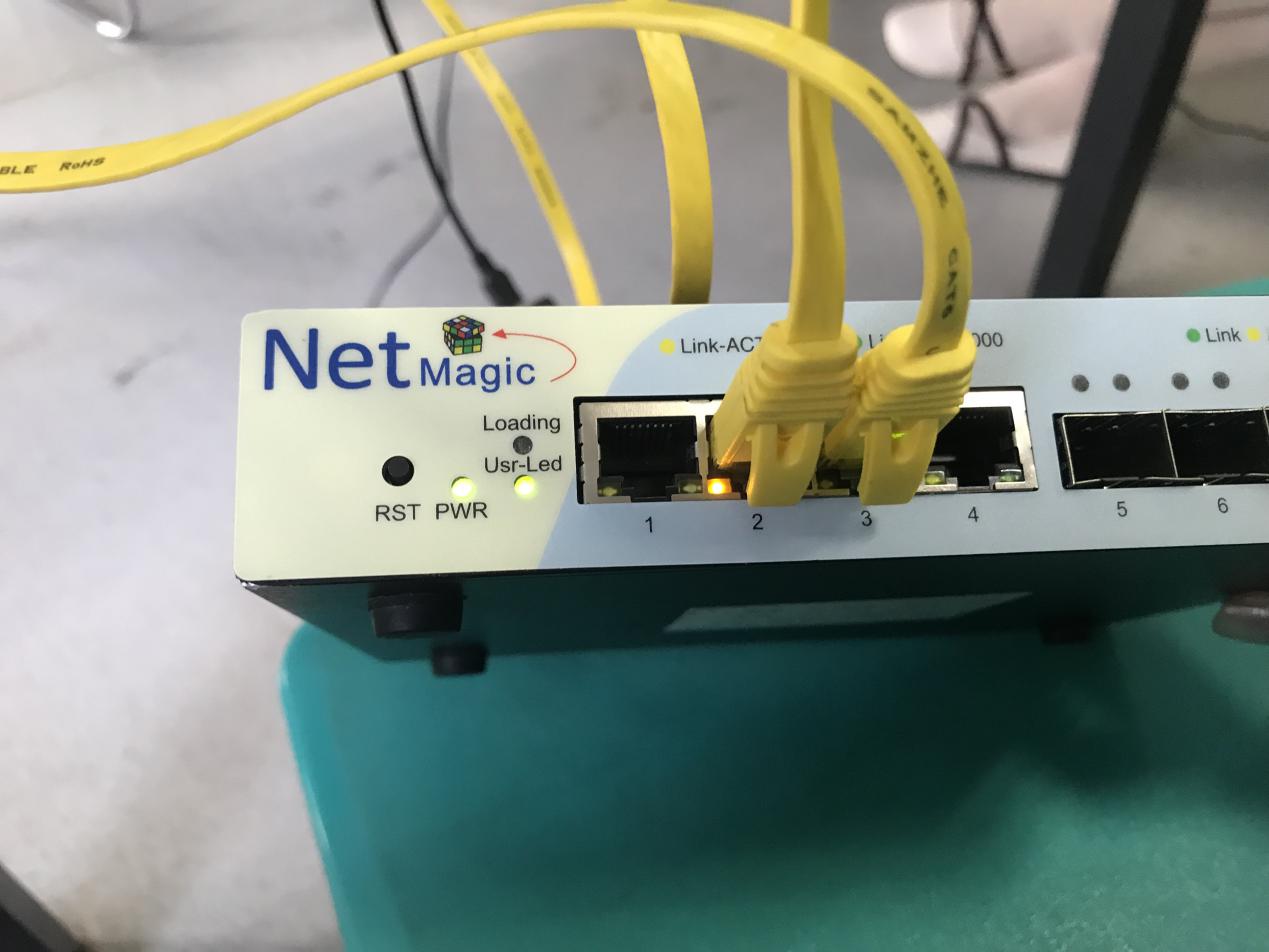
//um2cdp\_rule\_wrreq<=1'b0;//来自其它端口，不湖则

End

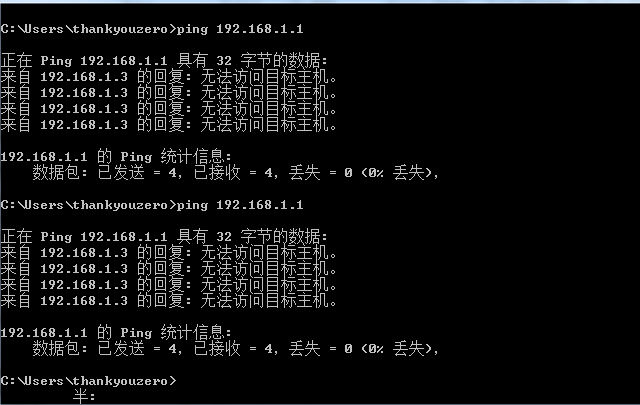
两处红字代码为关键部分，我们首先判断如果信息来自端口2那么我们用f表示将信息转发至所有端口（当然也可以用D表示将信息转发至除2号端口以外的几个端口）；之后再用else判断将从其他几个端口接收到的信息统一转发至2号端口。

我们首先和之前一样，编译程序然后将程序下载到我们的机器上。

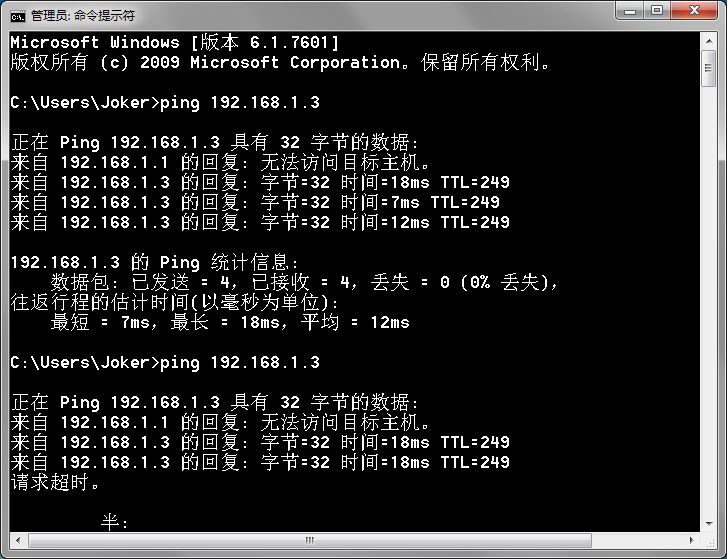
现在我们先连接2号口和3号口：



将相连的两台PC设置好静态IPv4地址后，使用ping命令进行判断是否连通。

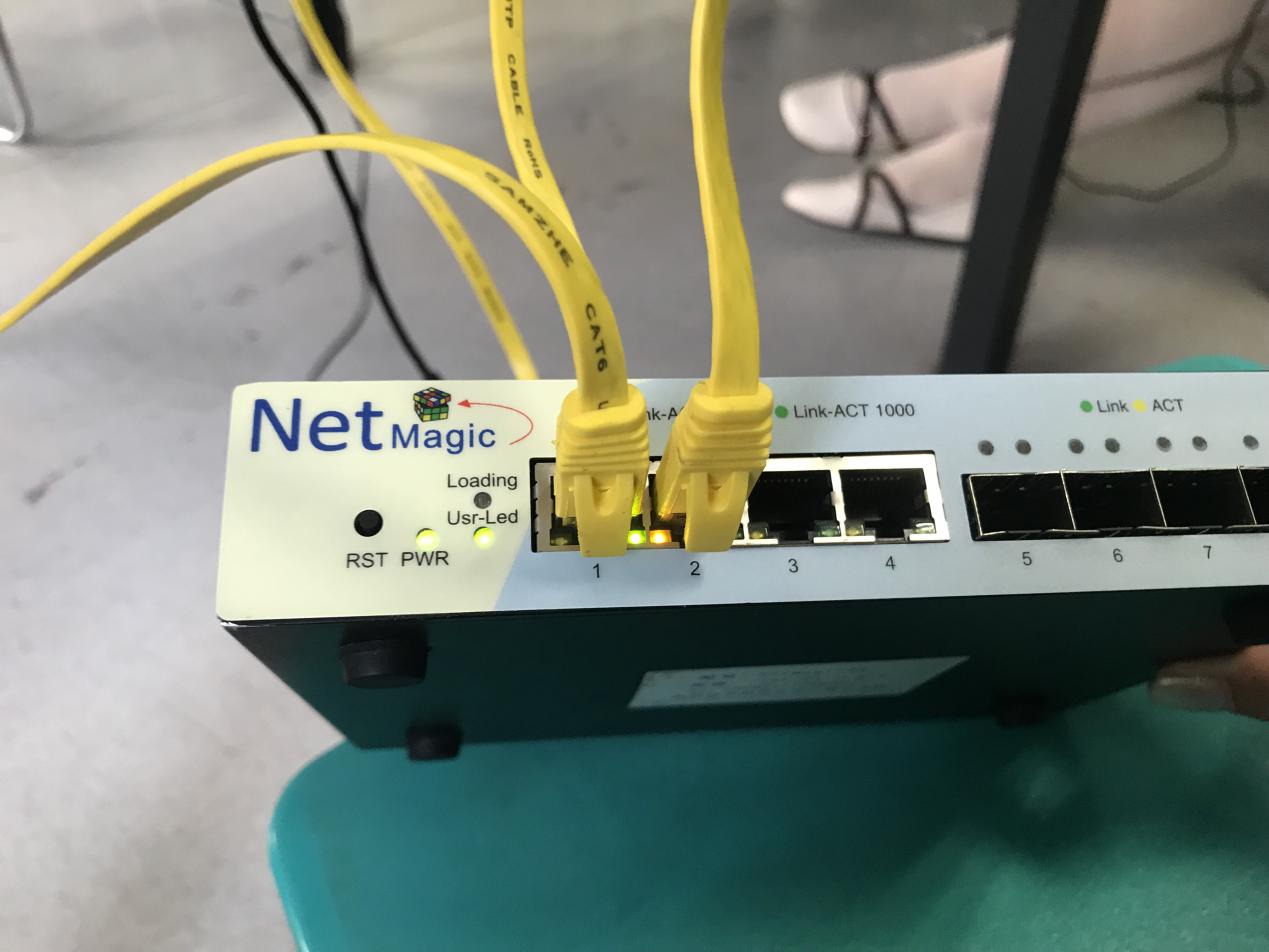


上面的一次ping命令为此时使用2号和3号端口进行通信的演示。同时在另外一台PC上使用ping命令：



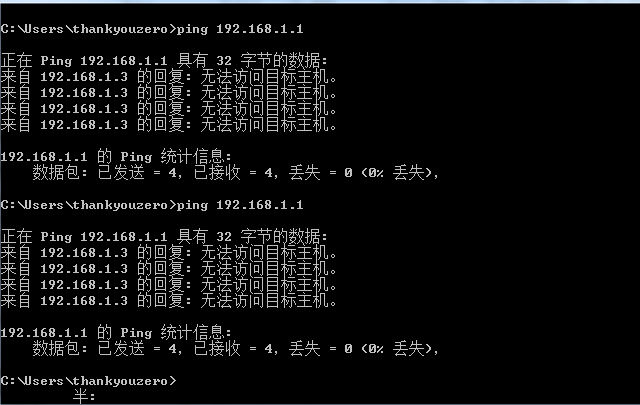
实验成功，证明此时2号和3号端口能成功连通。

现在我们将线连接在1号端口和2号端口：



再在两台PC机上使用ping命令进行连通：

![4PEJR$$UBE)(Y3%1](@(UN3](data:image/png;base64,)



这个时候出现一点小问题，因为换了一个网线的问题，此时从2号到1号端口是能成功连通的，但1号端口到2号端口存在丢包，查看后发现确实是由于网线的问题，不过至此实验已经可以证明成功。

成功实现2进n出的进阶实验。

1. 实验思考（个人部分单独完成）

我们在2进3出的硬件逻辑设计基础上更进一步，做出了2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2号端口和其他三个端口的互通，而其他端口之间不能通信的效果。

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，如上图：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（不管你是否需要），由于其不用路径选择，所以其网络成本可以很低廉。