实验目标

（详细论述所设计作品的功能）

作品名称：基于以太网口的文本显示器。

从一块 Digilent Nexys4 开发板上先通过 16 个开关设置一个 32 位的数据，将这 32 位的数据通过以太网口传输到另一块 Digilent Nexys4 开发板。当第二块 N4 开发板上收到以太网口传输来的数据后开始解析，当解析任务完毕后就向 CPU 发出中断请求。当 CPU 响应中断后，进入中断执行程序。判断这 32 位数据是否符合设定的规范，并且将数据做成一定的映射。在数据映射好之后，把 32 位数据送到 VGA 模块，在一个 20*15 的显示屏幕上的指定位置显示指定颜色的指定字符。

一、总体设计

1. 作品功能设计及原理说明（作品总体设计说明）

以太网口发射端：将数据从一块 N4 开发板传输到另一块 N4 板。

这里简化传输过程，每次传输 72 个字节的数据，8 个字节前导码，6 个字节 SRC.MAC，6 个字节 DST.MAC，2 个字节类型，46 个字节数据，4 个字节校验码。在 CLKIN 下降沿拉高 TX_EN 信号，在 CLKIN 的上升沿时，PHY 芯片自动把 TXD0 与 TXD1 上的数据发送出去。

以太网口接收端：接收从另一块 N4 开发板上传输的数据，解析完毕后向 CPU 发出中断请求。

当检测到 CRS_DV 信号拉高，同时 RXD0 与 RXD1 都为高电平的时候，开始接收数据。6 个字节 SRC.MAC，6 个字节 DST.MAC，2 个字节类型，46 个字节数据，4 个字节校验码。把 46 个字节数据的前 4 个字节取出，作为以太网口接收端模块的输出数据，同时在接收完毕后向 CPU 发出中断请求。

VGA 模块：接收 CPU 写入的数据，在 20*15 的屏幕上，显示指定的字符。

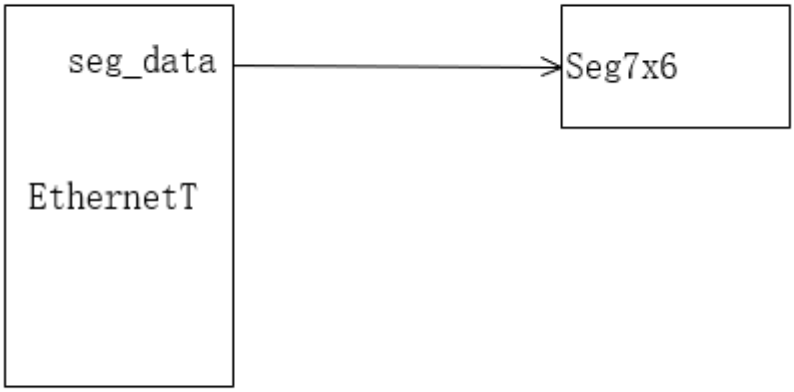
将屏幕划分为 300 个区域，每行 20 个字符，一共 15 行。用 600 个寄存器用来存储这 300 个区域的每个区域所对应的字符以及颜色。当 CPU 向 VGA 写入数据的时候，就修改指定区域所对应的寄存器，从而修改该区域的字符以及颜色。

CPU54 模块：执行 IMEM 中的指令以及响应中断。

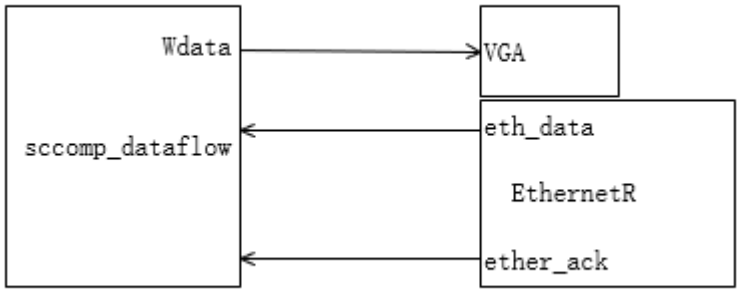
2. 硬件逻辑图

（除实验一外，均需给出硬件逻辑图，不得使用软件中自动生成的图，要求使用 visio 画，务必清晰明了）

发射端：



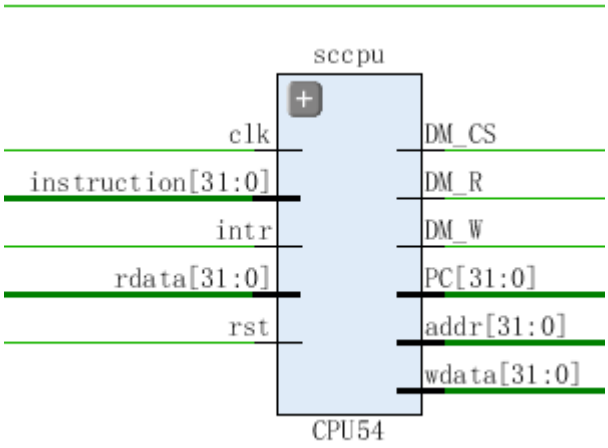
sccomp_dataflow：



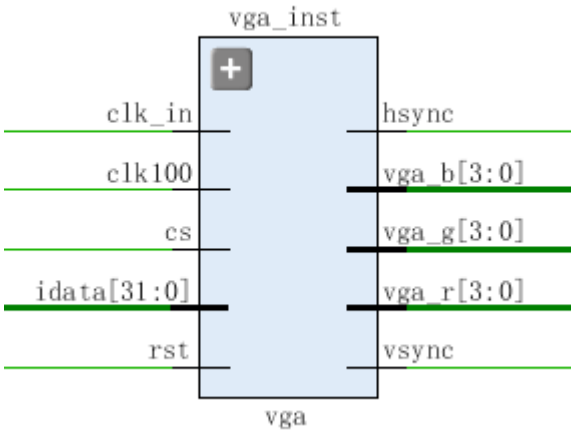
二、主要模块设计

（该部分要求分点描述，对总体设计原理图中的每个功能模块进行描述说明）

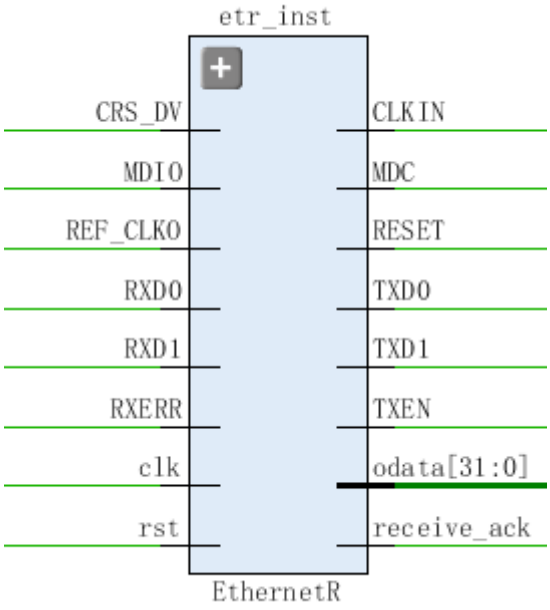
CPU: 执行 IMEM 中的指令以及响应中断。



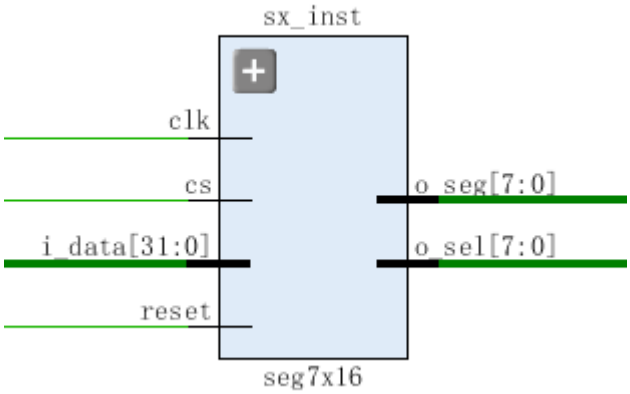
VGA: 接收 CPU 写入的数据，在 20*15 的屏幕上，显示指定的字符。



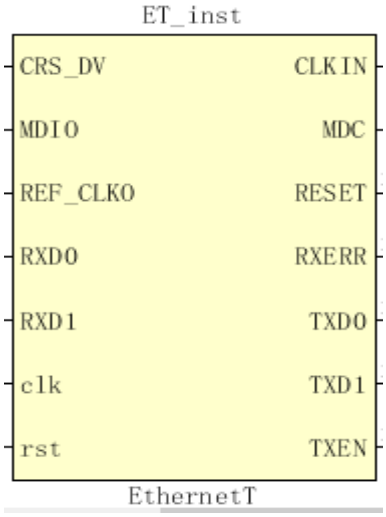
EthernetR: 以太网接收端，接收从另一块 N4 开发板上传输的数据，解析完毕后向 CPU 发出中断请求。



Seg7x16: 七段数码管，用来显示需要传输到另一块 N4 开发板的 32bit 的数据。

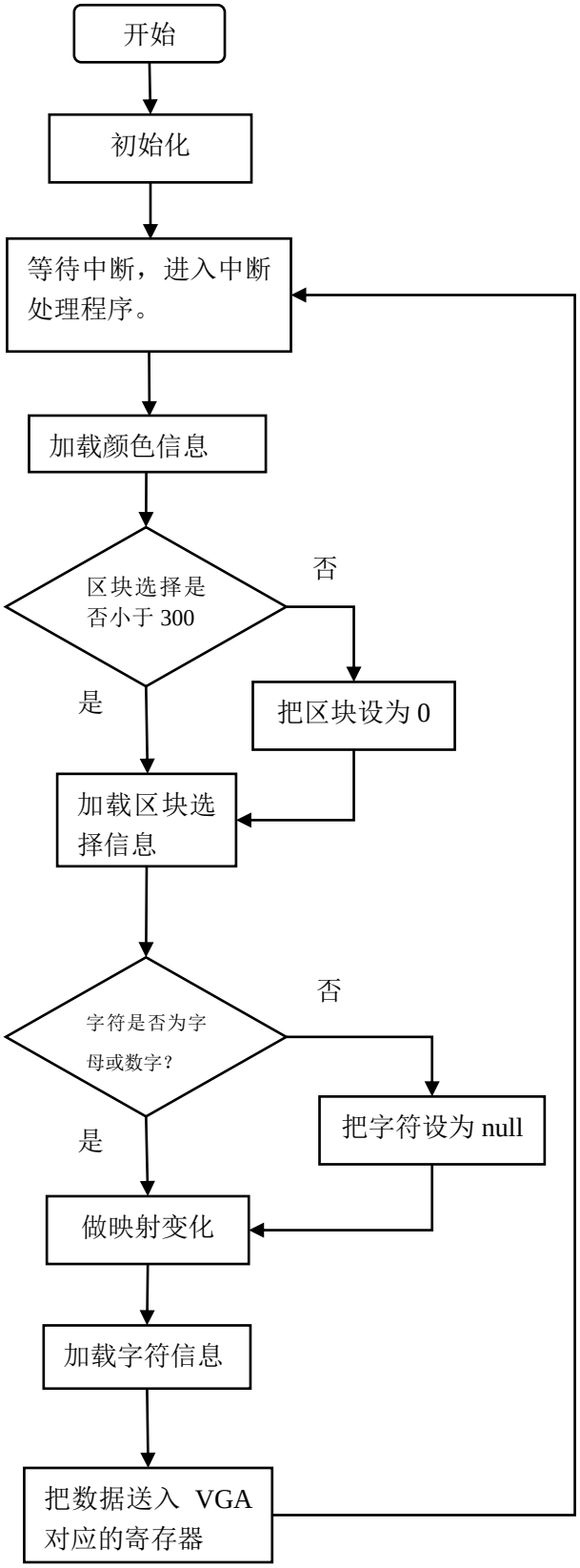


EthernetT: 以太网发送端，将数据从一块 N4 开发板传输到另一块 N4 板。



三、应用程序

1. 应用程序流程图



2. 程序模块（对每个模块进行功能的说明，程序必须加注释）

```
j _start
sll $0, $0, 0
j _exceptions
sll $0, $0, 0


_start:
lui $a0,0x1001 #a0 为 dmem 取数首地址
lui $a1,0x1081 #a1 为 vga in  地址

lui $a2,0x1081
addiu $a2,$a2,0x0004 #a2 为 ethernet  数据地址

lui $s1,0xfff0    #s1 为 0xfff0000  用来截取颜色信息

lui $s2,0x000f
addiu $s2,$s2,0xff00 #s2  为 0x000fff00  用来截取位置信息

addiu $s3,$0,0x00ff #s3  为  0x000000ff  用来截取字符信息

lui $t4,0xffff
addiu $t4,$t4,0xffff # t4  为 0xffff_ffff

xor $t1,$s1,$t4 # t1 为  0x000fffff
xor $t2,$s2,$t4 # t2 为  0xfff000ff
xor $t3,$s3,$t4 # t3 为  0xffffff00


cycle:
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
ori $v1,$0,0x0020
j cycle


_exceptions:
#进入中断，此时以太网传来数据
```

```
#先获取以太网传输到 cpu 的数据
lw $t0,($a2)#加载以太网传来的数据
and $s5,$t0,$s1    #把颜色装填进入 s5

and $t5,$t0,$s2 #取出位置信息
srl $t5,$t5,0x0008 #把位置信息整体向右移动 8 位

and $t6,$0,$0 #t6 清零
addiu $t6,$0,0x012c #t6=300
sltu $t7,$t5,$t6 #判读是否小于 300

and $t5,$t0,$s2 #再次取出位置信息
bne $t7,$0,position_normal
and $t5,$0,$0
position_normal:
or $s5,$s5,$t5 #把位置装填进入 s5

#考虑字符信息
and $t5,$t0,$s3 #取出字符信息

#大于 z
addiu $t6,$0,0x007b #比 z 大 1
sltu $t7,$t5,$t6 #判读是否小于 123
beq $t7,$0,output # ==0 说明大于等于 123，输出字符为 null

#大于 a-1
addiu $t6,$0,0x0061 #a=97
sltu $t7,$t5,$t6 #判读是否小于 97
bne $t7,$0,big_case # ==1 说明小于 97，去跳转判断是否为大写字母
subiu $t5,$t5,0x0056
or $s5,$s5,$t5
j output

big_case:

#大于 Z
addiu $t6,$0,0x005b #比 Z 大 1
sltu $t7,$t5,$t6 #判读是否小于 91
beq $t7,$0,output # ==0 说明大于等于 91，输出字符为 null

#大于 A-1
addiu $t6,$0,0x0041 #A=65
sltu $t7,$t5,$t6 #判读是否小于 65
bne $t7,$0,number # ==1 说明小于 65，去跳转判断是否为数字
```

```
subiu $t5,$t5,0x001c
or $s5,$s5,$t5
j output

number:

#大于‘9’
addiu $t6,$0,0x003a #比'9'大 1
sltu $t7,$t5,$t6 #判读是否小于 58
beq $t7,$0,output # ==0 说明大于等于 58，输出字符为 null

#大于‘0’
addiu $t6,$0,0x0030 #'0'=48
sltu $t7,$t5,$t6 #判读是否小于 48
bne $t7,$0,output # ==1 说明小于 48，只剩下 null
subiu $t5,$t5,0x002f
or $s5,$s5,$t5
j output


#下面这段可以直接删除掉
#char_out_null:
#j output
output:
sw $s5,($a1) #把数据送到 vga

#j _epc_plus4

#_epc_plus4:
#sll $0, $0, 0
mfc0 $k0, $14
addi $k0, $k0, 0x4
mtc0 $k0, $14
eret
```

四、测试/调试过程

（分别描述硬件模块的测试及调试实验程序的详细过程）

以太网口发射端：

通过 wireshark 进行以太网口数据抓包，在测试中发现在 N4 开发板上发送数据后并没有被 wireshark 所监测到，由此可判定发射端存在一定的问题。查阅 N4 开发板的 PHY 芯片 LAN8720A 的数据手册，发现需要在 CLKIN 信号的下降沿将 TX_EN 有效信设置为 1，在 CLKIN 信号的上升沿 PHY 芯片自动完成当前两

位数据的发送，因此将发送的触发沿从上升沿修改为下降沿。

再次使用 wireshark 进行抓包，发现依旧无法获取开发板以太网口传输来的数据。此时发现 N4 开发板上的以太网口的两个 LED 灯不符合要求，因此断定是 PHY 芯片的状态设置出现问题。再次查阅 LAN8720A 的数据手册，需要将 MDIO 初始化设置为 1，RESET 初始化设置为 1，为 MDC 提供一个 5M 的时钟。同时修改发送端的发送顺序。

这里举个例子：发送八个 bit 的数据 12345678，1 为最高位，8 为最低位。按照 12 ->{TX0, TX1}，34->{TX0, TX1}，56 ->{TX0, TX1}，78->{TX0, TX1}的顺序进行发送。

此时再使用 wireshark 进行抓包，成功获取一个以太网帧的包数据，取到 preamble 和 FSC 之后共计 60 字节的数据。

以太网口接收端：

通过 N4 开发板上的七段数码管来观测从以太网口传输进来的数据。在发射端发出数据后，七段数码管没有明显的改动。于是继续查阅 LAN8720A 的数据手册，发现需要在 CRS_DV 有效的情况下，RX0 与 RX1 都是 1 的时候开始接收帧数据，即前导码不接收入寄存器中，只是用来协调接收时序。

修改过后再次发生数据，发现七段数码管上的数据和发射端传输来的数据之间并不一致，但是可以做到每次发送都会改动七段数码管上的数据。由此可以断定，时序问题已经成功解决，问题是出现在数据解析上。通过仔细观察发送数据和接收数据，发现每个发送和接收数据的字节是一种镜像对称的关系，即发送 1234_5678(这里的数字只是指代数据位置)接收到的数据是 8765_4321。于是修改接收时候的存储顺序，把先传来的两个 bit 位存在[7:6]，以此类推。

经过再次修改后，七段数码管上的数据成功显示发射端上的传输数据。

VGA 显示模块：

对硬件模块进行下板测试，发现通过拨动开关来修改指定位置的输出符号并不能达到预期的效果，屏幕上一片漆黑。查阅硬件的源代码，发现里面关于显示当前扫描到的坐标所属区块的计算出现问题，verilogHDL 的计算与 C 语言的计算不同，需要考虑运算变量的位数。

修改计算表达式之后，显示屏上显示出来一片乱码，但是乱码与乱码之间规律性很强，于是查看对应字符的字模，发现字模与乱码一模一样，因此断定是预习存入的字符的字模 COE 出现问题。用 PCtoLCD2002 重新生成共计有 63 个字符的字模，重新导入到 COE 文件进行加载。

修改 COE 文件过后，屏幕上显示出对应字符。但是修改输入数据，发现每一次变动都会导致好几个位置的字符都发生变化，而且变化很有规律，每 16 个区块都对应相同的字符和颜色，因此可以断定是 verilogHDL 语法中某些变量的位数设置出现问题，查看源代码，发生是用来存储 300 个区块的 mem 大小设置出现问题应该设置为

reg[5:0]content[299:0]; reg[11:0]color[299:0];

但是我都设置为

reg[5:0]content[8:0]; reg[11:0]color[8:0];

再次修改过后，再次下板运行，发现实现预期目标。

MIPS 程序：

先在 MARS 上进行 DEBUG，初始化一个需要传输的 32 位数据，把原本需要等待中断才能进入中断执行的程序修改成直接进入终端的程序，直接模拟中断的过程。发现得到的输出结果是 0x0000_0000。一步步退回发现是一些初始的参数设置存在问题，于是进行修改。

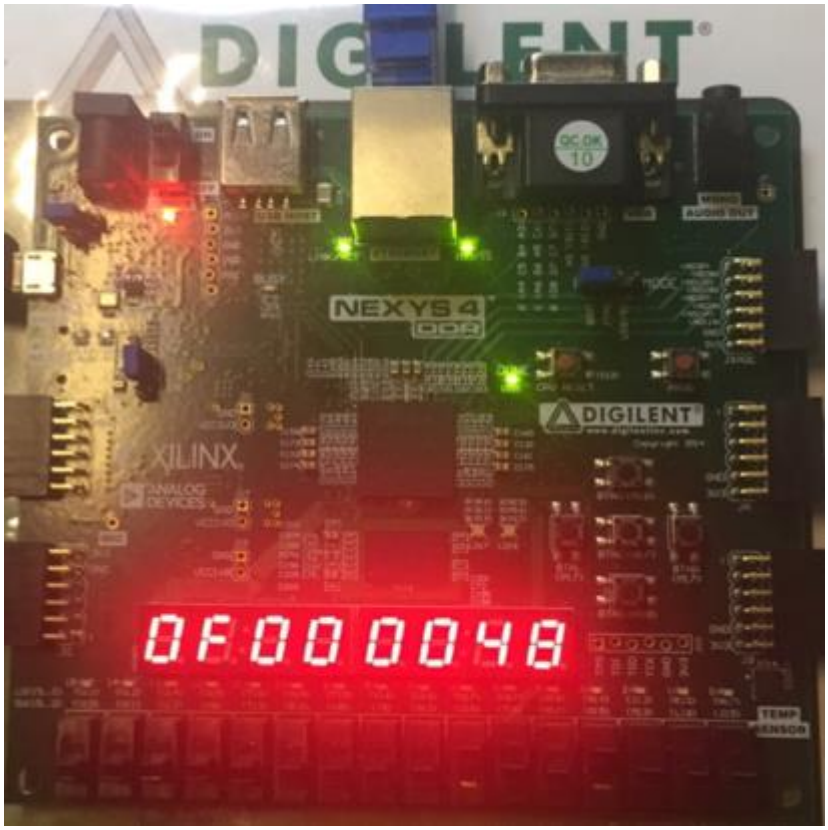
再次在 MARS 上运行测试 asm 文件，发现得到的数据不再是 0，但是也不是预期的数据，发现忘记对输入数据进行映射转换，于是在 MIPS 汇编程序的一些跳转分支中加入数据映射，把输入的 ASCII 码映射成对应字符在 VGA 缓存中相应字符的存储顺序。

再次在 MARS 上运行，成功得到预测数据，更换数据进行测试依旧得到预期结果。于是生成 CPU 指令，初始化 IMEM，直接下板观察结果。

五、实验结果分析

（该部分可截图说明）

发射端：



VGA 显示：

六、结论

本次课程设计是在 CPU54 的基础上，开发以太网口的运用。通过以太网口，用一块 DIGILENT NEXY4 开发板来控制另一块 N4 开发板上连接的 VGA 显示器上的输出文本。

七、心得体会及建议

由于计算机网络课程是下个学期的课程，因此在开发以太网口之前对于网络的知识是处于一种一无所知的状态。在明确了要开发以太网口的运用之后，开始收集资料。首先是查看 N4 开发板上关于 Ethernet PHY 的资料，在这份资料中得知 N4 开发板的网卡是 MICROCHIP 公司生产的 LAN8720A 芯片，LAN8720A 芯片对应的数据手册也附带在 01.Nexys4_DataSheet 中。N4 板上的 Ethernet 资料中提及了一些 MAC 层的 IP 核，但是 MAC 的控制器都是 MII，而 LAN8720A 芯片的控制器是 RMII，两者之间并不对应，前者是一次发送 4 个 bit 的信息，后者是一次发送 2 个 bit 位的信息，因此还需要一个 MII_TO_RMII 的 IP 核。在 vivado 中大概搜索了一下对应的 IP 核，发现里面的控制信号过于复杂，于是决定不使用 MAC 层的 IP 核，自己写一个 MAC 层来封装一个以太网数据帧。

在了解了这些信息后，开始浏览 LAN8720A 芯片手册。由于这份手册是全英文的，里面关于 PHY 芯片的初始化的一些信号设置看起来是一头雾水，于是连蒙带猜的给 N4 板上的 PHY 上电进行调试，直到两个 LED 全部亮起。

此时对于以太网数据格式没有任何概念，于是查阅《AC620 以太网设计与应用教程 V1.0》这个 pdf，从中得知以太网帧的封装格式。8 个字节的前导码，6 个字节的 SRC.MAC，6 个字节的 DST.MAC，2 个字节的报文类型，至少 46 个字节数据，以及最后 4 个字节的 FSC 校验码。因为只到达链路层，所以不涉及 IP 层，46 个字节中的数据不需要进一步细化为具体哪一种类型的报文。此时回过头来重新查阅 LAN8720A 的数据手册，发现芯片是符合 802.3 协议规范。

在有了对于以太网口比较全面的了解过后，开始设计状态机来发送和接收数据。中间又来来回回的查阅 LAN8720A 数据手册，下降沿的时候拉高 TX_EN 信号等等。

为了检测发送端的正确性，下载 wireshark 软件进行抓包。在这样的基础上 DEBUG 发送端的代码，直至可以被 wireshark 成功抓包。发送端完成过后开始设计接收端，为了让接受过程可视化，把接受到的数据通过七段数码管输出，直至收发两端能够一致，整个以太网硬

件设计完成。

以太网口的开发很好的锻炼了获取信息的能力，在短时间内完成一个没有预备知识的任务。

建议：

关于以太网口的资料过少，因此一开始着手的时候很容易让人摸不着头脑，没有前进的方向。因此，我把本次所查阅的所有资料作为附录一并提交，仅供参考。