揭秘88E1518以太网芯片：比RTL8211更麻烦，配置多一步”

基于FPGA的以太网相关文章导航，点击查看。

本文通过讲解88e1518以太网　芯片，该芯片会比RTL8211之类的麻烦那么一点，具体体现在内部寄存器的配置，会多一个步骤。

1、芯片引脚

88E151X这个系列包含几种芯片，本文只讲解88E1518，其余芯片差不太多，存在细微区别。拿到一颗芯片，首先应该了解其管脚，因为我们只有给芯片的输入管脚提供正确的数据才能驱动芯片，同理输出管脚才能输出想要得到的数据。88E1510和88E1518这两颗芯片的封装相同，如图1所示，该芯片是包含48个引脚的QFN封装。

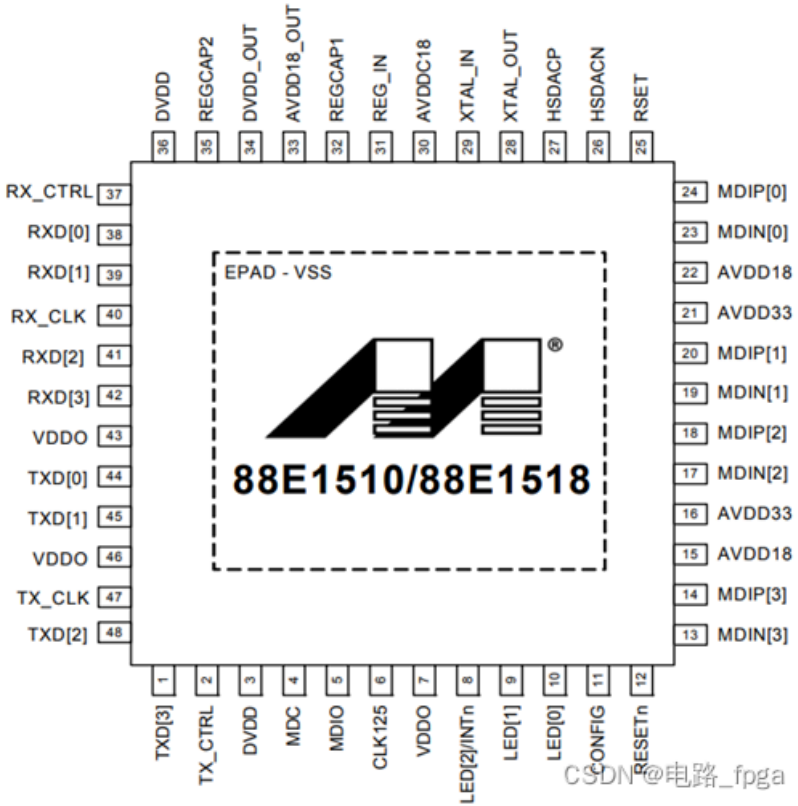

图1 88E1510或88E1518芯片

其中（MDIP[0]和MDIN[0]）、（MDIP[1]和MDIN[1]）、（MDIP[2]和MDIN[2]）、（MDIP[3]和MDIN[3]）是与RJ45座子连接的四组物理差分线，对应原理图上如下图所示：

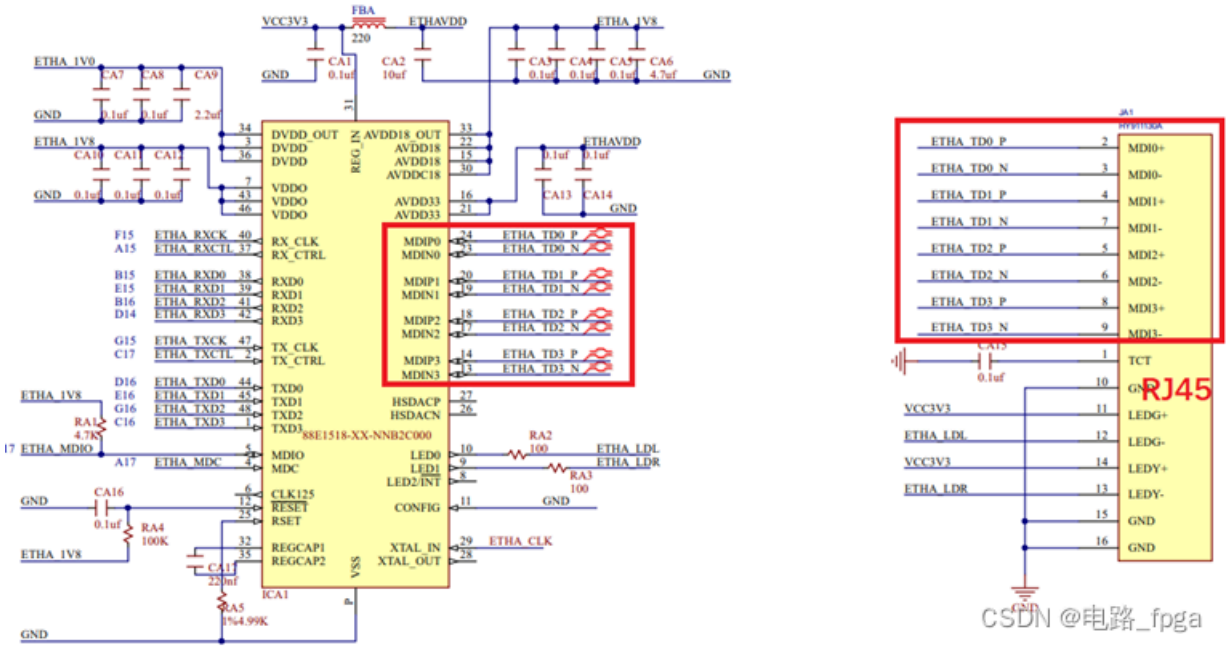

图2 物理差分线原理图

该芯片只有RGMII接口与MAC传输数据，根据前文，RGMII接口包含TX_CLK、TX_CTL、TX_DATA[3:0]、RX_CLK、RX_CTL、RX_DATA[3:0]等12个端口。

然后就是用于配置88E1518的MDIO端口，包含时钟信号MDC，和双向数据信号MDIO。**注意MDC最大不能超过12MHz，MDIO需要一个1.5KΩ~10KΩ的上拉电阻。**图2原理图中MDIO管脚使用4.7KΩ电阻上拉到1.8V电源电压。

有三个LED信号，LED[2]可以被用作中断信号，但是 FPGA 不需要中断信号，一般ARM才会使用，所以图2该引脚悬空。LED[1]和LED[0]与RJ45座子对应信号连接，可以通过配置芯片内部寄存器，让这两个灯对一些状态进行指示。

然后就是config引脚，这个引脚被用来设置PHY的地址，一个MDIO总线上可以挂载多个PHY芯片，通过PHY芯片的地址进行区分，PHY芯片的地址有5位，**但是88E1518芯片的高四位地址恒定为0，最低位由芯片硬复位后config引脚状态决定，所以一根MDIO总线最多挂载两颗88E1518芯片，PHY地址最低位与config状态一致。**

这张底板的MDIO挂载两路88E1518芯片，一片config引脚接地，一片接电源电压，如图3所示。

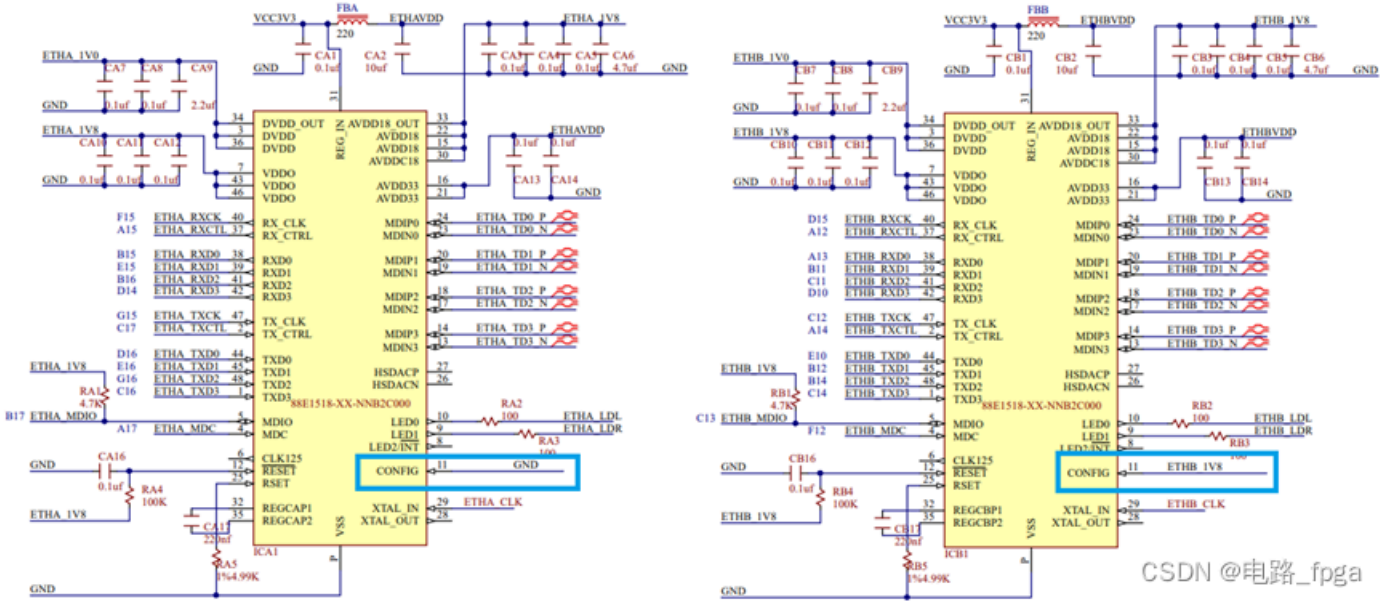

图3 两片PHY芯片地址引脚

Config引脚还有个设置VDDO支持电平的功能，如果该引脚硬复位(RESETn无效)后，出现上升沿或者下降沿，则VDDO支持1.8V或2.5V电平，如果一直保持不变，则VDDO支持1.8V和3.3V电平。

CLK125是芯片内部输出的一个参考时钟，一般会使用外部的晶振时钟作为参考时钟信号，所以该引脚一般不会被使用，悬空即可。

XTAL_IN和XTAL_OUT是与外部晶振连接，使用外部参考时钟时使用，接无源晶振时，两个引脚都要使用，**如果连接有源晶振，则从XTAL_IN输入25MHz时钟信号，XTAL_OUT必须悬空，不能接地或者接电源。**

RESETn是硬件复位信号，低电平有效，正常工作时位高电平。

RSET是电压源基准管脚，通过外部4.99 kΩ 1%电阻连接到地。

HSDACN和HSDACP是测试引脚，正常工作时不会使用，所以一般悬空处理即可。

至此就已经介绍了33个端口信号了，剩下的管脚均与供电电源相关，根据手册上的说明连接即可，是不是也比较简单。

最后要注意，**该芯片底部还有一个大焊盘，这个大焊盘必须接地，** 在设计PCB电路是需要特别注意。

该芯片的内部功能如图4所示，左侧是与RJ45连接的4对差分信号，右侧是与MAC进行通信的RGMII接口，下面是时钟、复位、测试、配置、LED、参考电压等模块。因为左侧RGMII接口传输的是数字信号，而右侧与RJ45传输的是模拟信号，所以内部存在ADC和DAC的结构。外部提供25MHz参考时钟，该芯片能够实现10Mbps、100Mbps、1000Mbps速率，

就会存在锁相环DPLL生成对应的频率的时钟信号。

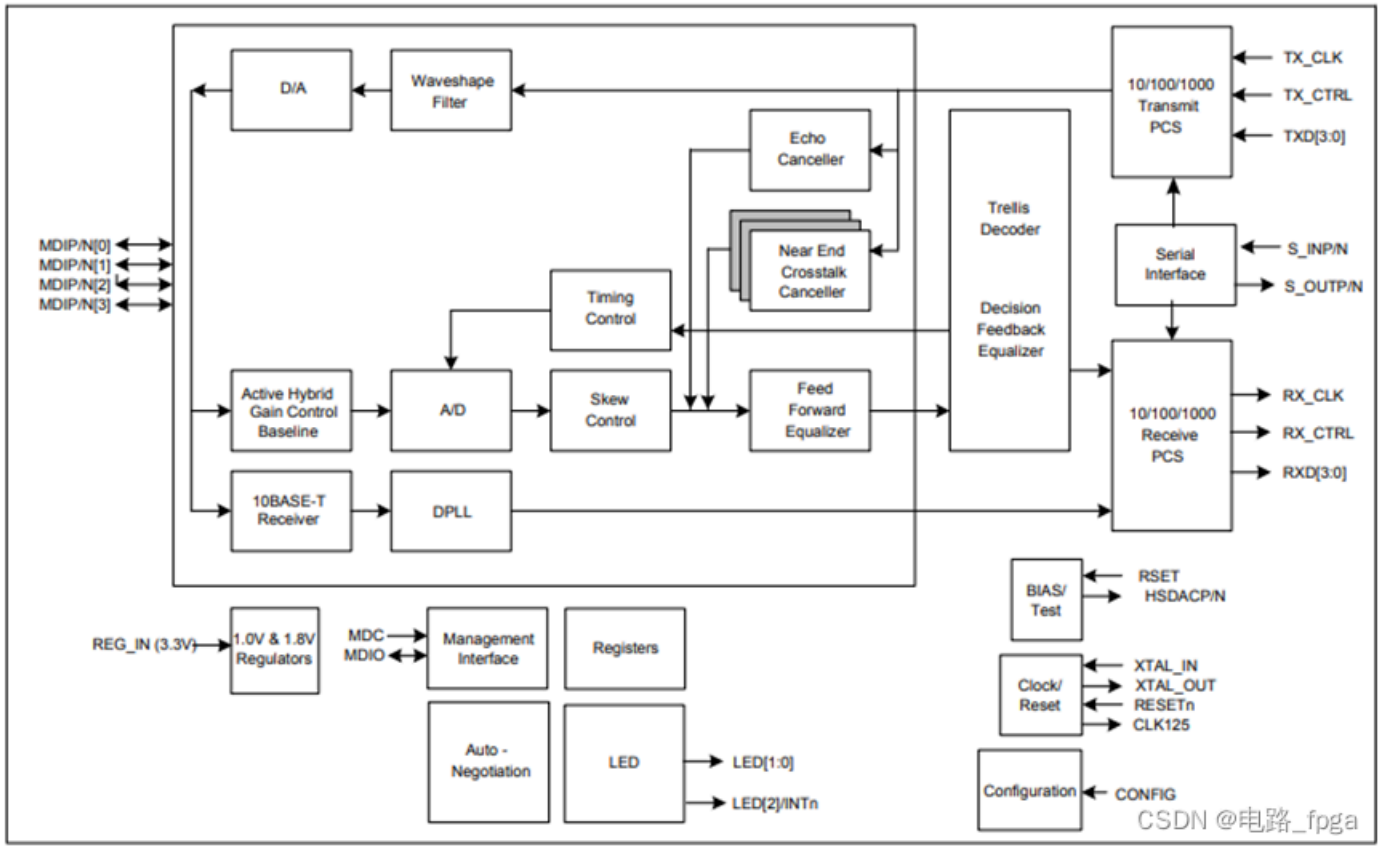

图4 88E1518内部结构图

2、内部寄存器

上文熟悉了88E1518芯片的管脚功能以及原理图上对应管脚的连接，对于这颗芯片使用来说，更重要的是内部寄存器的功能以及配置，88E1518相对其余以太网PHY芯片来说，内部寄存器很多，并且实现的功能也比较多，本文只选几个重要的寄存器讲解配置方式，需要了解更多的请查看数据手册。

88E1518芯片内往往会出现下面这种描述“**寄存器0_18.12”之类的字样，其实表示的意思是第18页第0个寄存器的第12位数据。**88E1518内部寄存器分了不同的页，每页存在多个寄存器，每个寄存器16位。**但是每页的22号寄存器都只有一个功能，那就是切换页**，如图5所示。

Register Name	Register Address	Table and Page
1000BASE-T Status Register	Page 0, Register 10	Table 74, p. 82
Extended Status Register	Page 0, Register 15	Table 75, p. 83
Copper Specific Control Register 1	Page 0, Register 16	Table 76, p. 83
Copper Specific Status Register 1	Page 0, Register 17	Table 77, p. 85
Copper Specific Interrupt Enable Register	Page 0, Register 18	Table 78, p. 86
Copper Interrupt Status Register	Page 0, Register 19	Table 79, p. 87
Copper Specific Control Register 2	Page 0, Register 20	Table 80, p. 88
Copper Specific Receive Error Counter Register	Page 0, Register 21	Table 81, p. 88
Page Address	Page Any, Register 22	Table 82, p. 89
Global Interrupt Status	Page 0, Register 23	Table 83, p. 89
Fiber Control Register	Page 1, Register 0	Table 84, p. 89
Fiber Status Register	Page 1, Register 1	Table 85, p. 91

图5 切换页寄存器

芯片上电或复位后，默认是在0页的，如果想要读写其他页的寄存器，就必须先向22号寄存器写入想要访问寄存器所在页，然后芯片就会切换到该页，滞后才能对该页的寄存器进行读写访问。其余以太网芯片好像是没有这么多寄存器的，所以不要这么操作。

比如上电后想要访问第6页的16号寄存器的5位到9位的数据，即16_6.9:5，那么首先需要往22号寄存器写入6，此时切换到第6页，然后向16号寄存器读写相应数据即可。

但是上电或复位后，如果是对第0页的寄存器进行读写，那么可以不用切换页，直接访问该寄存器即可。

在查看寄存器之前，首先需要了解一些寄存器类型简写，如下表1所示：

表1 寄存器类型

类型	含义
LH	该位数据为高电平时锁存高电平，直到寄存器被读取或寄存器复位时拉低。
LL	该位数据为低电平时锁存低电平，直到寄存器被读取或寄存器复位时拉高。
RES	保留位。
Retain	执行软件复位后，寄存器值将保留。
RO	只读寄存器。
ROC	**只读寄存器，并且数据被读取后自动清除。**

类型	含义
RW	可读可写寄存器。
RWC	可读可写，在复位或读取寄存器后被清除为零。
SC	**将1写入此寄存器将立即执行其功能，然后在功能完成时将该寄存器字段自动清除为零。**
Update	**写入寄存器的值在软复位后才能执行。**
WO	只写寄存器。
NR	与保留的位基本相同，不能使用。

特别注意只读、只写、软复位后才能生效的寄存器了，还有SC及ROC这类寄存器，下文讲解几个常用的寄存器。

2.1 Copper Control Register(0_0)

88E1518每个寄存器的数据位宽时16位，首先查看第0页第0个寄存器，这个寄存器可以对RGMII的接口的传输速率、双工模式、软复位、回环等进行设置，如下表2所示：

表2 Copper Control Register(0_0)寄存器

Bit	类型	含义
15	RW，SC	**软复位，高电平有效，复位成功后自动清零。**
14	RW	PHY芯片把来自MAC侧的数据回环给MAC，高电平启用回环。
13	RW，Update	**与bit6共同决定RGMII接口速率。**
12	RW，Update	**高电平使能自动协商，低电平关闭自动协商。**
11	RW，Retain	该位和16_0.2控制断电。在PHY从掉电过渡到正常工作之前，这两位必须设置为0。
10	RW	正常工作时设置为0，设置为1时起隔离作用。
9	RW，SC	**高电平时重启自动协商。**
8	RW，Update	**高电平时PHY采用全双工模式，低电平时半双工模式。**
7	RO	碰撞测试，正常工作时不会被使用。
6	RW，Update	**与bit13共同决定RGMII接口速率。**

Bit	类型	含义
5：0	Reserved	保留位，一直为0。

下面对上表中一些位进行详细说明：

首先第15位复位，该位为高电平可以对第0、2、3、5、7的寄存器进行软复位。0_1_15可以将第1页的寄存器进行软复位，20_18.15可以把第6、18页寄存器进行软复位，复位成功后，这些位自动清零。

第14位用于回环测试，PHY芯片的回环测试由几种，其中0_0_14位控制的是PHY中回环MAC侧数据，用于测试MAC侧电路和代码是否正确，如下图所示：

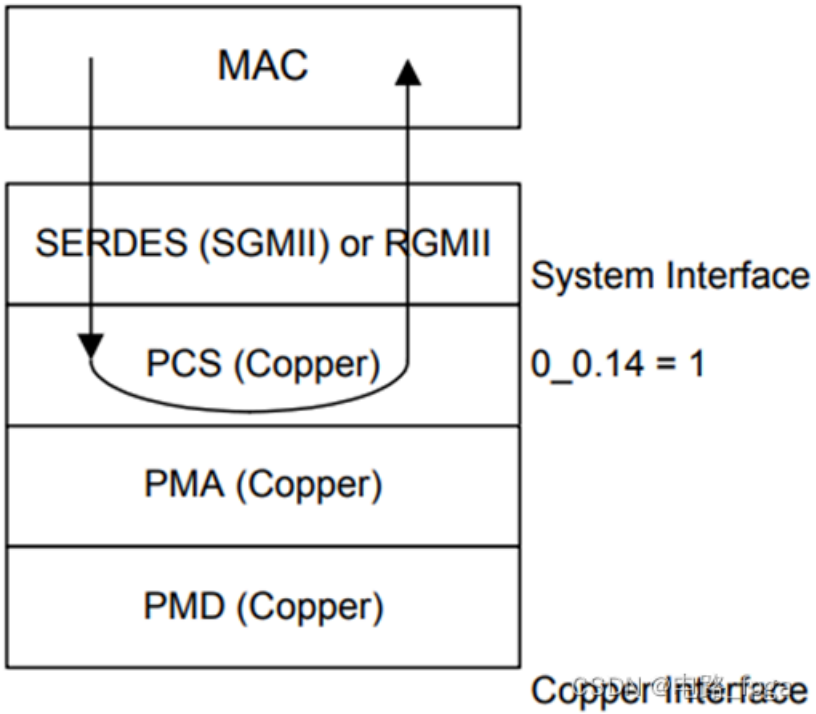

图6 系统接口环回图

其余回环可以在手册中进行查取，此处不做过多介绍。

PHY芯片的真实工作速率和模式，与0_0_12自动协商是否开启，以及0_0.6和0_0.13和0_0.8有关。如果自动协商使能，那么PHY芯片就会与PC端的PHY自己协商使用何种速率和模式进行通信，比如MAC侧的PHY芯片最大支持1000Mbps全双工模式，而PC端有2.5Gbps的网卡，那么双方协商的结果就是使用1000Mbps全双工模式进行通信。

因此**自动协商使能时，通过设置0_0.6、0_0.13、0_0.8 寄存器来改变PHY芯片通信速率和工作模式是无效的。**

想要修改MAC侧PHY芯片通信速率就必须关闭自动协商，然后通过0_0.6、0_0.13设置通信速率，起真值表如下。通过设置0_0.8位高电平使用全双工模式，通过设置低电平使用半双工模式。

表3 PHY芯片工作速率

Bit6	Bit13	速率
1	1	保留
1	0	1000Mbps
0	1	100Mbps
0	0	10Mbps

只对寄存器0_0.12、0_0.13、0_0.6和0_0.8的值进行修改是不会生效的，除非发生以下情况之一：**软件复位(寄存器0_0.15为高电平)，重新启动自协商(寄存器0_0.9为高电平)，从断电到上电的转换（寄存器0_0.11为高电平）。**

为了启用或禁用自动协商，寄存器0_0.12应该与寄存器0_0.15或0_0.9同时更改。

例如，要禁用自动协商并使用10Mbps半双工模式，寄存器0_0应该写入0x8000。

2.2、Copper Status Register（1_0）

前文讲解了控制寄存器Copper Control Register，那么接下来讲解其对应的状态寄存器Copper Status Register，相应位如下表4所示。

表4 Copper Status Register（1_0）

Bit	类型	含义
15	RO	此协议不可用。
14	RO	1：PHY能够执行全双工100BASE-X
13	RO	1：PHY能够执行半双工100BASE-X
12	RO	1：PHY能够执行全双工10BASE-X
11	RO	1：PHY能够执行半双工10BASE-X
10	RO	此协议不可用。
9	RO	此协议不可用。

Bit	类型	含义
8	RO	1：寄存器15中的扩展状态信息。
7	RO	总是0。
6	RO	1：PHY接受带有前导抑制的管理帧。
5	RO	1：**自协商进程完成。0：自协商进程未完成。**
4	RO，LH	1：检测到远端故障状态。0：未检测到远端故障状态。
3	RO	1：PHY能够进行自动协商。
2	RO，LL	**从上次读取该位后链路就断开了。当前的链路状态，1：链路已连通，0：链路未连通。**
1	RO，LH	1：检测到Jabber条，0：未检测到Jabber条件
0	RO	1：扩展寄存器功能。

上述就是一个只读寄存器，与通常的状态寄存器类似，就是可以查看一些控制寄存器的状态，比较关注的是bit5和bit2，一个用来判断自协商是否完成，能否传输数据了，另一个用来检测数据传输链路是否出错。

2.3、Copper Specific Status Register 1(17_0)

Copper Control Register（0_0）寄存器能够对PHY的通信速率，工作模式进行设置，但是是否其效果，还与自动协商是否被使能有关，所以并不能通过读取Copper Control Register（0_0）的值获取当前PHY芯片给通信速率和工作模式。只能通过Copper Specific Status Register 1(17_0)来获取这些信息，该寄存器对应位的含义如图7所示（寄存器内容比较多，但是关注的也就几位，所以直接截图，一般读取黄色标记的几位数据判断即可。）：

Bits	Field	Mode	HW Rst	SW Rst	Description
15:14	Speed	RO	0x2	Retain	These status bits are valid only after resolved bit 17_0.11 = 1. The resolved bit is set when Auto-Negotiation is completed or Auto-Negotiation is disabled. 11 = Reserved 10 = 1000 Mbps 01 = 100 Mbps 00 = 10 Mbps
13	Duplex	RO	0x0	Retain	This status bit is valid only after resolved bit 17_0.11 = 1. The resolved bit is set when Auto-Negotiation is completed or Auto-Negotiation is disabled. 1 = Full-duplex 0 = Half-duplex
12	Page Received	RO, LH	0x0	0x0	1 = Page received 0 = Page not received
11	Speed and Duplex Resolved	RO	0x0	0x0	When Auto-Negotiation is not enabled 17_0.11 = 1. 1 = Resolved 0 = Not resolved
10	Copper Link (real time)	RO	0x0	0x0	1 = Link up 0 = Link down
9	Transmit Pause Enabled	RO	0x0	0x0	This is a reflection of the MAC pause resolution. This bit is for information purposes and is not used by the device. This status bit is valid only after resolved bit 17_0.11 = 1. The resolved bit is set when Auto-Negotiation is completed or Auto-Negotiation is disabled. 1 = Transmit pause enabled 0 = Transmit pause disable
8	Receive Pause Enabled	RO	0x0	0x0	This is a reflection of the MAC pause resolution. This bit is for information purposes and is not used by the device. This status bit is valid only after resolved bit 17_0.11 = 1. The resolved bit is set when Auto-Negotiation is completed or Auto-Negotiation is disabled. 1 = Receive pause enabled 0 = Receive pause disabled
7	Reserved	RO	0x0	0x0	0
6	MDI Crossover Status	RO	0x1	Retain	This status bit is valid only after resolved bit 17_0.11 = 1. The resolved bit is set when Auto-Negotiation is completed or Auto-Negotiation is disabled. This bit is 0 or 1 depending on what is written to 16.6:5 in manual configuration mode. Register 16.6:5 are updated with software reset. 1 = MDIX 0 = MDI
5	Reserved	RO	0x0	0x0	Reserved
4	Copper Energy Detect Status	RO	0x0	0x0	1 = Sleep 0 = Active
3	Global Link Status	RO	0x0	0x0	1 = Copper link is up 0 = Copper link is down

图7 Copper Specific Status Register 1(17_0)

上述十几个简单但比较常用的寄存器，均位于第0页，如果只需要对这几个寄存器进行配置，那么就不需要进行换页操作，直接读写即可，实测88E1518上电后默认停留在第0页。

3、MDIO接口时序

既然可以通过配置PHY芯片内部寄存器对其功能进行调整，那如何读写内部寄存器呢？

本文只简单讲解MDIO接口时序，后文将使用FPGA通过MDIO对内部寄存器进行配置，图8是MDIO的写时序。

MDC是MAC侧输出给PHY芯片的串行时钟信号，最大不能超过12MHz，如果MDIO总线上挂载的PHY芯片过多，MDC可能达不到12MHz。

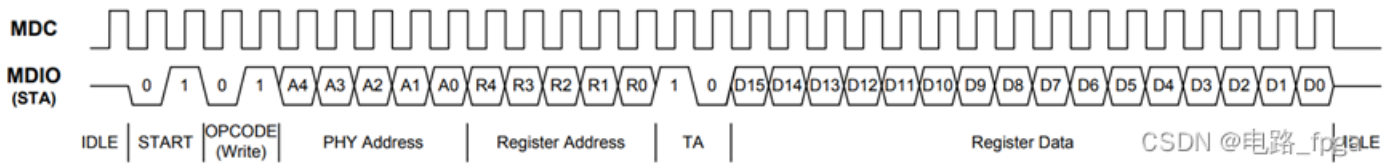

图8 MDIO写时序

MDIO是MAC侧读写PHY芯片的双向数据线，写时序各阶段的数据含义如图9所示：

32-Bit Preamble	Start of Frame	OpCode Read = 10 Write = 01	5-Bit PHY Device Address	5-Bit PHY Register Address (MSB)	2-Bit Turn around Read = z0 Write = 10	16-Bit Data Field	Idle
11111111	01	10	00000	00000	z0	0001001100000000	11111111

图9 串行管理接口协议

首先需要发送32位的前导码1，然后发送2位起始码，接着如果是写操作，则依次发送01，然后就是5位PHY芯片的地址（MDIO总线可以挂载多个PHY芯片，PHY芯片就是通过MDIO发送的PHY地址来判断是不是发送给自己的数据，88E1518高4位PHY地址恒为0，最低位与config引脚复位后的初始状态），然后就是需要写入寄存器的地址，如果是写操作，接下来发送10，然后跟需要写入寄存器的16即可，写入数据完成后，释放总线，由于MDIO数据线外部会接上拉电阻，此时总线被拉高。

其实对于写驱动程序来说，不管是什么协议，需要关注的点还有发送数据时先发高位还是先发低位，数据在时钟上升沿还是下降沿变化。由图8可知MDIO发送数据是先发高位，且数据在时钟下降沿发生变化。

图10是MDIO读取寄存器数据时的时序图，前导码、起始位、PHY地址、寄存器地址与写时序均一致。但是起始位后的操作码变为了10，**在发送寄存器地址后，MAC需要释放总线，输出高阻态**，此时总线被外部上拉电阻拉高，PHY芯片会在下个时钟周期将总线拉低作为应答，然后从高到低输出16位寄存器数据。

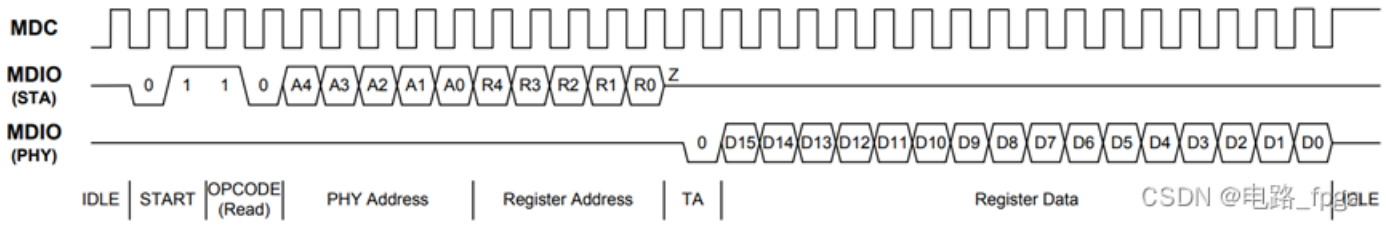

图10 MDIO读时序

了解上述内容就可以通过Verilog HDL实现驱动代码了，对于FPGA还需要知道该芯片端口寄存器的一些时间参数，便于时序约束。如图11所示。

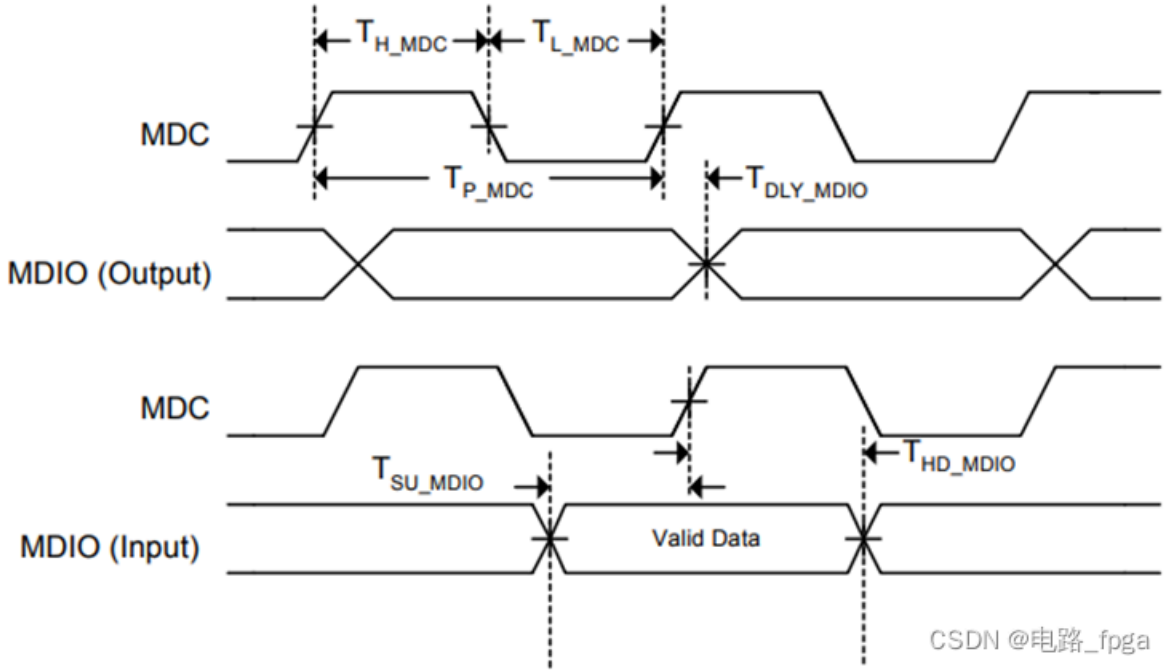

图11 MDC/MDIO时序

通过阅读数据手册可知，**MDIO端口寄存器的建立时间TSU_MDIO为10ns，保持时间THD_MDIO同样为10ns。而寄存器输出延迟TDLY_MDIO最大值为20ns，最小值为0ns**，如图12所示：

Symbol	Parameter	Min	Typ	Max	Units
T_{DLY_MDIO}	MDC to MDIO (Output) Delay Time	0		20	ns
T_{SU_MDIO}	MDIO (Input) to MDC Setup Time	10			ns
T_{HD_MDIO}	MDIO (Input) to MDC Hold Time	10			ns
T_{P_MDC}	MDC Period	83.3			ns[1]
T_{H_MDC}	MDC High	30			ns
T_{L_MDC}	MDC Low	30			ns

图12 MDC/MDIO时序参数

本文对88E1518芯片管脚，硬件原理图，内部寄存器，MDIO做了详细讲解，需要了解更多，在公众号后台回复”**以太网PHY芯片**”（不包括引号）即可。

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