第 36 卷 第 5 期 计 算 机 工 程 2010 年 3 月
Vol.36 No.5 Computer Engineering March 2010
·工程应用技术与实现· 文章编号：1000—3428(2010)05—0240—03 文献标识码：A 中图分类号：TP331

基于精确时间协议的 IP 核设计

谢昊飞，郑 鸣，王 平
(重庆邮电大学网络化控制与智能仪器仪表教育部重点实验室，重庆 400065)

摘 要：针对传统精确时间协议的软件设计中存在随机抖动、误差大的问题，提出一种 IP 核设计。该设计采用数字逻辑电路获取精确时间戳，并实现同步算法、晶振补偿算法和状态转移控制算法，利用晶振分频比微调减小晶振频率漂移对同步精度和稳定性的影响。仿真结果表明，该设计具有较高的同步精度，精度值可达 10 ns。

关键词：精确时间协议；IP 核；同步；状态转移控制

IP Core Design Based on Precision Time Protocol

XIE Hao-fei, ZHENG Ming, WANG Ping
(Key Laboratory of Network Control & Intelligent Instrument, Ministry of Education,
Chongqing University of Posts and Telecommunications, Chongqing 400065)

【Abstract】 This paper proposes an IP core design, aiming at the problem of randomized jitter and large error in pure software design of traditional Precision Time Protocol(PTP). It gives prominence to adopt digital logic circuit to achieve getting of precision time stamp, synchronization algorithm and quartz oscillator frequency-compensated algorithm and state-transfer controlled algorithm, makes use of oscillator's divided frequency adjusted to reduce influence of oscillator's frequency-excursion and circumstance factor to precision and stability of synchronization. Simulation result shows that this design has higher synchronization precision, and precision value can reach 10 ns.

【Key words】 Precision Time Protocol(PTP); IP core; synchronization; state transfer control

精确时间协议(Precision Time Protocol, PTP)由 IEEE1588 标准定义，用于对标准以太网或其他分布式总线系统设备进行微秒级和亚微秒级同步。纯嵌入式软件实现时，存在通信协议栈处理延迟、系统任务调用、中断响应、算法执行效率低、晶振频率漂移等问题，给时钟同步精确度造成了某种程度上的制约。本文重点研究基于 PTP 协议的 IP 核设计，并结合相应算法的数字逻辑实现，提升 PTP 协议系统的合理性和算法的时效性。

1 协议实现方式

PTP 协议同步过程如图 1 所示，同步报文历经 PTP 协议应用层(PTP Apl)、网卡驱动(Driver)、介质访问控制层(Media Access Control, MAC)、介质无关接口(Media Independent Interface, MII)和物理层(PHY)之后发送到网络上[1-2]。

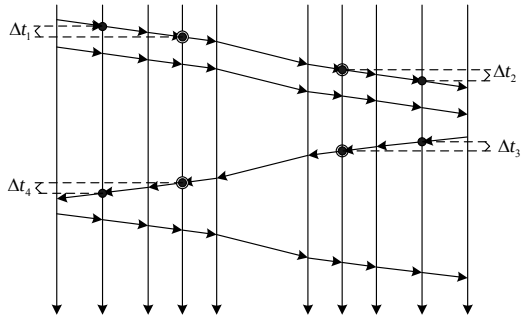

图 1 PTP 协议时钟同步过程

Driver 中网卡收发中断获取 PTP 报文收发时间戳($T1$, $T2$, $T3$, $T4$)，但在 MAC 层中要有载波侦听、冲突检测、地址识别、循环冗余校验、队列缓冲等处理，其延迟不确定性很大，尤其当网络负荷比较大时，MAC 中会形成较长的发送或接收等待队列，等待时延无法估计。因此，Drivers 中获取的时间戳准确性得不到保障。有些文献提出在 MII 获取时间戳，越过 MAC 延迟可以提高精确度，然而没有具体的实现方案。由于网卡芯片不提供 MII 软件接口，因此单靠软件无法解决问题。本文通过设计数字逻辑电路与 MII 连接方式，监测通过 MII 的数据帧以获取精确的时间戳($Tm1$, $Ts1$, $Ts2$, $Tm2$)，然后将其交由 IP 核内部的同步算法控制模块进行计算线路延时和时钟偏差：

$$\begin{cases} delay = (Ts1 - Tm1 + Tm2 - Ts2)/2 \\ offset = (Ts1 - Tm1) - delay \end{cases} \quad (1)$$

因此，看到 MAC 到 MII 之间的不确定性延迟时间(Δt_1, Δt_2, Δt_3, Δt_4)被消除了。

PTP 协议作为一种应用层服务实体，与操作系统和处理器存在无关，可以根据用户的需求进行灵活设计。因此，本文提出一种新的 IP 核与软件协议引擎结合的架构设计思想。

在本设计中，由 IP 核来完成 64 位硬件定时器和同步算法、晶振补偿算法等控制，由软件实现 PTP 协议引擎功能，

基金项目：国家“863”计划基金资助项目(2007AA041301-04)
作者简介：谢昊飞(1978－)，男，博士研究生，主研方向：工业以太网，智能仪器仪表；郑 鸣，硕士研究生；王 平，教授、博士、博士生导师
收稿日期：2009-08-12 **E-mail**：tomxiefine@hotmail.com

负责应用层报文处理。协议引擎解析报文得到主时钟时间戳(*Tm*1, *Tm*2)通过总线交给同步算法控制模块，定时器由 MII 信号触发记录本地收发报文时间戳(*Ts*1, *Ts*2)，同步算法控制器利用得到的 4 个时间戳计算时钟偏差 *offset*，修正定时器。定时器通过总线接口为应用层提供时钟信号。这样软、硬件实现合理分工、协同工作，既能够发挥协议引擎应用层处理优势，又能够发挥 IP 核硬件获取精确时间戳和高效计算的优势，对于实现高精度时钟同步提供了一种新思路。

2 IP核的设计

本文研究的 IP 核基于 Altera 公司可编程片上系统(System On a Programmable Chip, SOPC)片内 Avalon 总线架构，设计实现 Avalon 总线接口，可作为可复用控制单元挂载在由 NiosII 软核构成的 SOPC 系统上，实现同步控制。

2.1 IP核总体结构

IP 核的总体结构采用模块化设计，各主要的功能模块独立，方便模块的功能验证，通过统一定义的内部信号端口进行连接，“例化”完成模块组合。IP 核结构如图 2 所示。

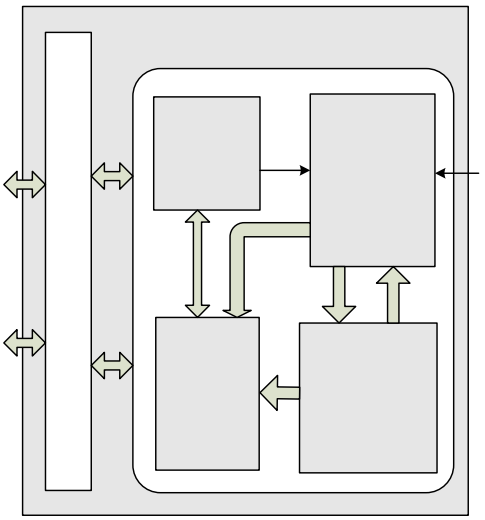

图 2 IP 核结构

在 IP 核算法控制部分(弧形方框内的部分)中，分频控制器(Divide Frequence Controller, DFC)基于晶振时钟 clk 进行分频，得到 div_clk 作为定时器计数脉冲，定时器作为本地时钟进行计时，同步算法控制器(Synchronization Algorithm Controller, SAC)从定时器获得本地时间戳(*Ts*1, *Ts*2)，从 PTP 协议引擎获得主时钟时间戳(*Tm*1, *Tm*2)，计算本地时钟偏差 *offset*，对定时器时间修正。晶振补偿控制器(Crystal Compensate Controller, CCC)利用从定时器和同步算法控制器获得的参数，进行晶振频率补偿值计算，然后调整分频周期来补偿晶振频率偏差。

2.2 IP核核心算法

2.2.1 同步过程控制算法

根据 PTP 同步原理和 MII 获取时间戳的设计思想，同步过程控制算法如图 3 所示。

由于 MII 接口的数据流是以半位元码(nibble)形式传输，即每个时钟周期传输 4 位数据[3]。设计中实现一个特殊的 nibble 计数器，用于记录通过 MII 接口的 4 位半码的个数，以此可以定位 PTP 帧的不同协议字段，产生相应控制信号。当 MII 接收有效(RX_DV)信号为高电平时，开始侦听数据信号端口，一旦收到帧起始定界符“10101011”，立即产生时间戳触发信号，触发定时器保存当前时刻值。同时启动 nibble 计数器开始记录收到的半位元码数，以此找到 PTP 帧中关键字段进行如图 3 所示算法控制，最终完成一次同步计算和时钟修正过程。

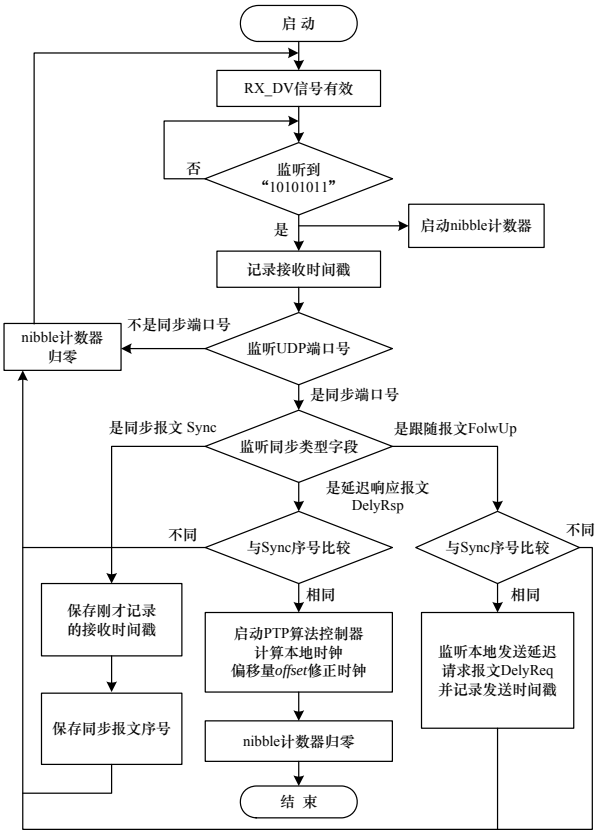

图 3 同步控制算法流程

2.2.2 状态转移控制算法

IP 核设计上采用 Mealy 有限状态机为核心的逻辑控制，如图 4 所示。该设计利用 Mealy 状态机直接完成同步状态转移控制算法，使 IP 核设计更加紧凑。且使用 Mealy 状态机对整个 IP 核进行逻辑控制，有利于满足综合条件并生成门级网表。

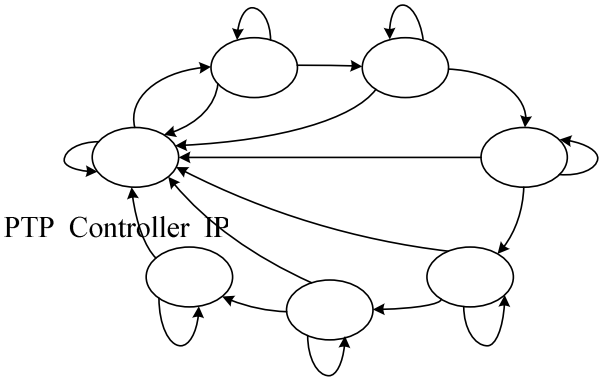

图 4 状态转移算法 Mealy 状态机

系统启动后处于空闲状态(*idle*)，等待同步报文，当收到同步报文(Sync)，由 MII 检测帧定界符来触发定时器记录接收时间戳(*Ts*1)，同时启动超时定时器并转入 *Ts*1 状态。然后等待接收跟随报文(FolwUp)，当未等到跟随报文，超时定时器溢出，状态返回 *idle* 状态。当收到跟随报文，PTP 协议引擎解析报文得到主时钟时间戳(*Tm*1)并交给 IP 核，同时超时

定时器清零并转入 *Tm*1 状态。接着 PTP 协议引擎随机延迟一段时间后发送延迟请求报文(DelyReq)，在 MII 接口同样触发记录发送时间戳(*Ts*2)，同时转入 *Ts*2 状态，在下一个时钟周期再次启动超时定时器，等待延迟请求响应报文(DelyRsp)。当收到延迟请求响应报文，协议引擎解析时间戳 *Tm*2 并交给 IP 核，同时超时定时器清零并转入 *Tm*2 状态。如果 IP 核一直没有检测到 DelyRsp 报文，超时定时器溢出，并返回 *idle* 状态。当得到 4 个时间戳后，进入 *delay* 状态计算线路延时 *delay*，并判断 *delay* 是否有效，如果不满足直接返回 *idle* 状态，清除寄存器中的 4 个时间戳。如果满足就进入 *offset* 状态计算主时钟、从时钟偏差 *offset* 并修正本地时钟定时器，完成本次时钟校准。

2.2.3　传输延时抖动修正算法

在 MII 获取时间戳，能够大大提高精度，但在计算 *delay* 时，是假设 Sync 报文和 DelyReq 报文在传输路径上延时相等。然而传输媒质和网卡的收、发路径一般采用不对称设计，以减小远程末端串扰[3]，当网络负荷大时，这种不对称性表现的更加明显，计算 *delay* 相应也不准确。为了减小这种误差，采用统计求平均值方法先对 *delay* 加以修正：

$$delay = \frac{\sum_{i=0}^{k} delay_i}{k+1} \tag{2}$$

然后再计算 *offset*。式(2)中，*k* 是同步次数，是将本次同步和前面同步得到的所有 *delay* 进行统计求平均值，这样在经过一段时间多次同步后，得到的 *delay* 趋于收敛。

2.2.4　IP 核晶振补偿算法

由以往纯嵌入式软件实现 PTP 协议和测试结果可知，同步间隔内主时钟、从时钟速率存在偏差。这是由于电子系统中常用晶振频率精度长期稳定性差，受温度、压力等影响大[4]。假设系统中主时钟、从时钟都采用±20 p/m 精度、50 MHz 标称频率的晶振，实际晶振频率为 50 MHz±1 000 Hz(±20 p/m×50 MHz = ±1 000 Hz)，则每个同步间隔(如 2 s)主时钟、从时钟之间产生的最大时间偏差为 $2\times2\times1\,000/50\,\text{MHz} = 80\ \mu\text{s}$，已经不能忽略。为了减小这种偏差，采用晶振补偿算法。算法过程为：设从时钟收到第 *n* 次 Sync 报文计时值为 S_n，主时钟计时值为 M_{n+1}，从时钟收到第(*n*+1)次 sync 报文计时值为 S_{n+1}，主时钟计时值为 M_{n+1}；从时钟收到第(*n*+2)次 Sync 报文计时值为 S_{n+2}，主时钟计时值为 M_{n+2}。因为主时钟一般都以固定的同步间隔发送 sync 报文，所以：

$$\begin{cases} M_{n+1} = (M_{n+1} - M_n) + M_{n+1} \\ S_{n+1} = (S_{n+1} - S_n) + S_{n+1} \end{cases} \tag{3}$$

如果要使 $S_{n+2} = M_{n+2}$，那么必须从接收到第(*n*+1)次 Sync 报文到接收到第(*n*+2)次 Sync 报文的这段时间内，从时钟计数器值的增量为 $M_{n+2} - S_{n+1}$，该增量比原来应有的增量 $S_{n+1} - S_n$ 增加了：

$$(M_{n+2}) - (S_{n+1}) = 2(M_{n+1}) - (S_{n+1}) \tag{4}$$

即 $2\times offset_{n+1} - offset_n$，设分频控制器按照分频周期 DivPeriod 进行分频，这样每个分频周期需要补偿的值为

$$CompValue = \frac{2\times offset_{n+1} - offset_n}{S_{n+1} - S_n} \times DivPeriod \tag{5}$$

上述算法是每 2 次同步后进行一次晶振补偿，这样补偿实际不可取。实际实现时是在上述算法思想基础上做相应调整：每 2*N* 次同步后进行一次晶振补偿，*N* 值可以根据系统具体应用情况进行设定。

3　逻辑验证与性能测试

通过设计测试平台 Testbench，建立验证模型，对 IP 核进行逻辑功能验证。由 MII 接口测试激励向量模拟 MII 数据和控制信号对主时钟、从时钟在同步过程的相应时刻产生激励，记录时间戳，Avalon 总线接口测试向量对 IP 核读写控制。输出最终通过 ModelSim 以波形显示测试结果，如图 5 所示。

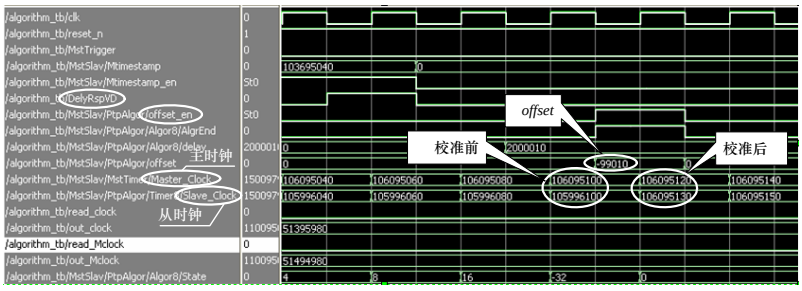

图 5　IP 核功能测试时序图

可以看到，当 DelyRspVD 信号为高电平时表明收到延迟请求响应报文，再经过 2 个时钟周期计算得到修正值 *offset* 为-99 010，同时修正值有效信号 *offset*_en 高电平，驱动从时钟校准。校准前主时钟 Master_Clock 值为 106 095 100，从时钟 Slave_Clock 值为 105 996 100，相差-99 000；校准后主时钟、从时钟值分别为 106 095 120 和 106 095 130，在 clk 频率为 50 MHz 时误差为 10 ns。从时序图看出，从时钟基本跟随主时钟频率变化，达到了晶振频率补偿目的。IP 核测试在实际网络系统中的同步精度需要进行进一步板级验证。

4　结束语

本文设计基于精确时间协议，充分发挥了数字逻辑电路的优势，提高了时钟同步精度。为进一步基于 FPGA 芯片的板级实现和网络控制协议 SoC/SoPC 系统设计打下了基础。

参考文献

[1] Weibel H. High Precision Clock Synchronization According to IEEE 1588 Implementation and Performance Issues[C]//Proceedings of 2005 Embedded World Conference. Nuremberg, Germany: [s. n.], 2005.

[2] IEEE Standard. IEC 61588-2004 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems[S]. 2004.

[3] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Std 802.3™-2005 Telecom and Information Exchange Between Local and Metropolitan Area Networks Specific Requirements[S]. 2005.

[4] Eidson J C. Measurement, Control, and Communication Using IEEE1588[M]. New York, USA: Springer-Verlag, 2006.

编辑　陆燕菲