IEEE1588 精密时钟同步关键技术研究

孔令彬，文赫胜，陈向文
(中国地质大学 机电学院，湖北 武汉 430074)

摘要：随着网络控制技术的发展，分布式控制系统对时钟同步精度提出了更高要求；以 IEEE1588 精密时钟同步标准为背景，阐述了高精度时钟同步机制和时钟校正原理，分析了 IEEE1588 协议的核心算法——最佳主时钟（BMC）算法和本地时钟同步（LCS）算法，同时从技术开发的角度对系统中同步精度的影响因素如时间戳的生成方式，网络的对称性等作了分析，并提出了一些减少干扰，提高系统时钟同步精度的改进方法。

关键词：IEEE1588；时钟同步；主时钟；时间戳

Research on Key Technology of IEEE1588 Precision Clock Synchronization

Kong Lingbin, Wen Hesheng, Chen Xiangwen
(Department of Mechanical and Electronic Engineering, China University of Geosciences, Wuhan 430074, China)

Abstract: Distributed control system are set for higher requirements of clock synchronization with the development of network control technology. By IEEE1588 principles, high-accuracy time synchronization and the clock correction mechanism are elaborated. The Best Master Clock Algorithm and Local Clock Synchronization Algorithm in IEEE1588 are analyzed. Clock synchronous accuracy influencing factors are also discussed from technology development aspect. Methods for increasing the control system clock synchronization precision are proposed.

Key words: IEEE1588; clock synchronization; master clock; time stamp

0 引言

IEEE1588 是用于网络测量和控制系统的精密时钟同步协议标准，是新一代测控总线 LXI 标准的重要组成部分。为了满足测量仪器、工业控制等领域中亚微秒级的时间同步要求，IEEE1588 标准在这方面作出了重大贡献，目前的最新 V2 标准于 2008 年 7 月发布[1]。

IEEE1588 标准的全称是“网络测量和控制系统的精密时钟同步协议标准（IEEE1588 Precision Clock Synchronization Protocol）”，简称 PTP（Precision Timing Protocol），它的基本原理是通过一个同步信号周期性对网络中所有节点的时钟进行校正同步，可以使基于以太网的分布式系统达到亚纳秒级的精确同步。与已有的时间同步技术如 NTP/SNTP、GPS 等相比具有配置容易，快速收敛，精度高以及对网络带宽和资源消耗少等特点[2]。在对时间同步要求比较严格的应用场所，如工业实时以太网，电力自动化系统，移动通信网络中，逐渐引起人们的关注。

1 IEEE1588 时钟同步机制

1.1 PTP 时钟状态机

PTP 时钟同步系统是由主时钟和从时钟构成的一种主从关系的网络层次结构，它由一个或多个 PTP 子域组成，每个子域包含一个或多个彼此通信的时钟。在网络中的每一个 PTP 时钟都有可能处于从时钟（Slave Clock）或主时钟（Master Clock）这两种状态，时钟状态由最佳主时钟（BMC）算法决定。处于主时钟状态的设备被认为精确时钟，它将同步从时钟的时间，但在同一个通信子域内只能存在一个主时钟。

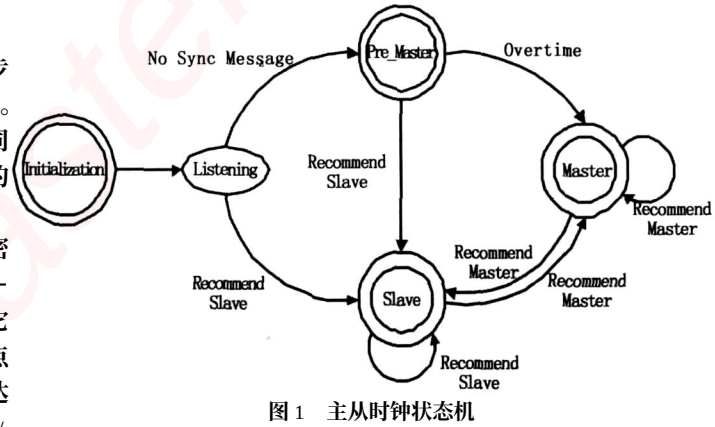

图 1 主从时钟状态机

PTP 时钟状态转换关系如下图 1 所示。

在 PTP 网络系统中各个时钟设备通过周期性交换带有时间信息的同步（sync）报文来进行主从时钟间偏差（Offset）和网络时延（Delay）的计算，进而纠正偏差，补偿时延，实现主从时钟之间亚纳秒级的同步[1]。当一个时钟上线，它在系统指定的时间内监听来自主时钟的 Sync 消息。

(1) 如果收到来自主时钟的 Sync 消息，则本地时钟进入 Slave 状态；

(2) 如果一段时间内没有收到 Sync 消息，该时钟则假定自己为主时钟，此时的状态为 Pre_Master，时钟端口表现为主时钟的状态，但是不发送 Sync 消息；这个 Pre_Master 状态会保持一定的时间，如果在指定的时间内仍没有收到其它时钟的 Sync 消息，则该时钟状态为主状态 Master，并且发送 Sync 消息。

时钟各端口都执行最佳主时钟算法确定自己和网络内其它时钟的状态[2]。如果时钟的某个处于从状态 Slave 的端口确定它能比现在的主时钟质量高，则呈现主时钟的状态并发送 Sync 消息，主时钟的一个端口收到更好的 Sync 消息，则停止

宣布自己为主时钟并停止发送 Sync 消息。从时钟端口通过与主时钟交换报文信息获得正确时间来更新本地时钟的时间，以达到与主时钟同步。

1.2 PTP 时钟同步与本地时钟同步（LCS）算法

对于已经确定主从关系的时钟网路，将根据本地时钟同步(Local Clock Synchronization) 算法校准本地时钟节点与主时钟同步，其流程如图 2 所示[3]。主、从时钟之间存在 Sync，Delay Req，Follow up，Delay Resp 4 种类型的报文交换，报文交换期间产生 4 个发送或接收时刻值，此动作称之为打时间戳事件，这 4 个数值用来计算偏差和时延以同步从时钟。在测量过程中，假设传输介质是对称均匀的，从时钟先通过报文传输的往返迭代得出路径延时（delay），然后计算出主、从时钟的时间偏移（offset），最后对从时钟进行同步调节。

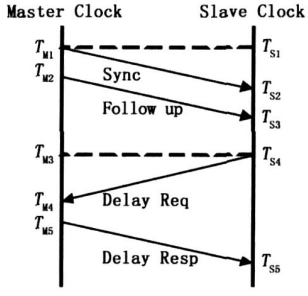

图 2　时钟同步示意图

在图 2 中：

(1) 从时钟在 T_{C2} 时刻收到主时钟发送的 Sync 广播报文。

(2) 在 T_{S3} 时刻，从时钟收到主时钟发送的携带同一回合 Sync 报文发送时间 T_{M1} 的 Follow UP 报文，从时钟与主时钟的时间偏移 Toffset 为：

$$T_{offset} = T_{S2} - T_{M1} - T_{delay} \quad (1)$$

(3) 从时钟在 T_{C4} 时刻向主时钟发送 Delay Req 报文。

(4) 在 T_{S5} 时刻，从时钟收到主时钟发送的与同一回合的 Delay Req 报文相对应的 Delay Resp 报文，其包含了主时钟收到 Delay Req 的时刻 t_{M4}，其延时 T_{delay} 为：

$$T_{delay} = T_{offset} + T_{M4} - T_{S4} \quad (2)$$

将延时 T_{delay} 带入式 (1) 可以得出 T_{offset}，进而可以对从时钟进行调节[1]。

2 最佳主时钟（BMC）算法

最佳主时钟算法（Best Master Clock Algorithm）独立运行于 PTP 系统中的每个时钟上，主要完成选举主时钟和生成时钟网络拓扑结构两个任务，它们分别由数据设置比较算法（Data Set Comparison Algorithm）和状态决定算法（State Decision Algorithm）来实现。

2.1 数据设置比较（DSC）算法

数据设置比较算法根据同步报文不同的数据集（包含时钟品质、时钟等级、时钟类型、时钟偏移等参数信息的变量集合），比较筛选出可用最佳报文，确定最佳主时钟，进而产生拓扑结构。它是动态运行于每个时钟上的，即在时钟同步系统运行中根据实时数据不断计算比较时钟选举数据集，动态调整各节点和端口的状态，也就会调整时间的传递路线。所以在当前主时钟故障或性能下降时，系统可能会选择其它更合适的节点替代它作为主时钟[1]。

表征时钟品质的变量值通过 Allan 均方差公式得出。Allan 方差公式原用于振荡器频率的统计误差计算，这里用于表示时间的统计误差：

$$\sigma^2_{ptp} = \frac{1}{3}\frac{1}{2(N-2)} \times \sum_{k}^{N-2}(X_{k+2} - 2X_{k+1} + X_k)^2 \quad (3)$$

式 (3) 中，σ^2_{ptp} 是多次测量的均方差值。其中 X_k，X_{k+1}，X_{k+2} 是在 t_k，$t_{k+\tau}$，$t_{k+2\tau}$ 时刻所作的时间残差测量，τ 是测量的时间间隔，N 是测量的次数。统计方差公式已排除任何稳定的对称误差，时钟的漂移并不会影响方差 σ^2_{ptp}，但时钟的不规则跳动直接影响 σ^2_{ptp} 值。σ^2_{ptp} 值再经过取对数，乘以时间常数和滞环处理才成为数据集中使用的时钟变量参数。

2.2 状态决定（SD）算法

状态决定算法是在确定所属 PTP 子域主时钟后，根据不同数据集的信息进一步计算出每个时钟各个 PTP 端口的具体推荐状态。每个 PTP 端口都可能处于以下 9 种状态，包括 INITIALIZING、FAULTY、DISABLED、LISTENING、PRE_MASTER、MASTER、PASSIVE、UNCALIBRATED、SLAVE，本地时钟根据这些状态去做出对应的调整。为避免网络生成回路，状态决定算法会生成树形拓扑结构，将一些竞争失败的节点端口定义为禁用（disabled）状态和被动（passive）状态[1]。

设某一典型的时钟节点 C_0 由默认数据设置 Do 来描述，且该时钟有 M 个 PTP 端口。首先比较同一路径上所有端口 D_0 数据集的时钟等级 clock stratum，若相同则通过数据设置比较算法比较各端口发出的有效同步报文，决定这些报文中的 Erbest（接收信息的最好值），最后比较各 Erbest 的路径长度，最终从剩下的 N 个 Erbest 中选出 Ebest（所有信息最好值）。状态决定算法通过 D_0，Erbest 和 Ebest 之间的相互关系，判断 PTP 子域内时钟端口的状态，并根据判断结果对端口状态进行更新，其流程如图 3 所示。

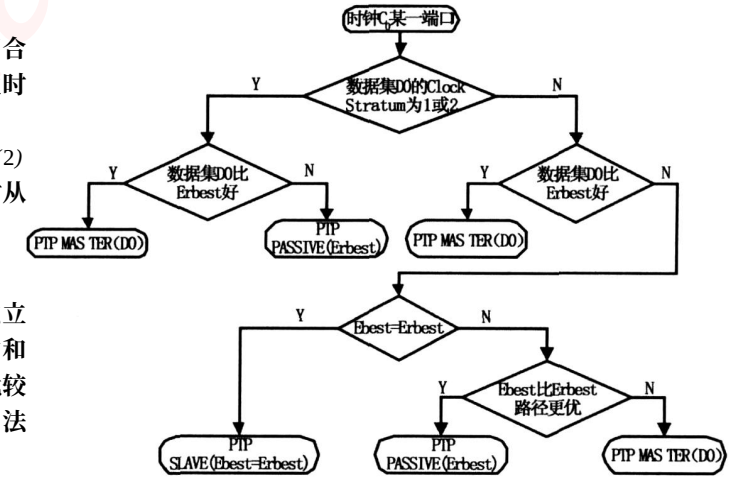

图 3　状态决定算法流程图

该算法运行在 PTP 子域中时钟的每个 PTP 端口，根据运算分析结果，动态调整各个时钟和端口的状态，所以在当前主时钟故障或性能下降时，本地时钟端口数据设置会改变，时钟状态会被更新，系统能够自动调用前面的数据设置比较算法选择其它更合适的时钟来替换当前主时钟，从而保证分布式控制系统的精密同步。

FIFO1 进行缓存[5]。

3.4 Flash 操控与 USB 通信模块

由于 Flash 芯片出厂时即存在一部分坏块，且在使用中也随时会产生新的坏块。因此，Flash 写程序设计采用先判断好坏块，再进行块擦除，最后完成块编程写入数据的操作顺序进行。其中，当判断本次要操作的为坏块时立即跳过，进入下一块的判决操作。Flash 读程序设计与此相同。USB 通信模块主要完成 FPGA 与 CY7C68013 之间的通信与数据交换，即负责判断 CY7C68013 所传递的上位机的不同控制命令并执行相应操作，实现 FPGA 中 FIFO 与 CY7C68013 中 FIFO 的数据传递。

3.5 USB 应用软件

固件程序指运行在 CY7C68013 内部 CPU 中的程序，采用 Keil C51 编译器开发，实现接收处理 USB 驱动程序请求等功能。Windows 驱动程序调用 CYPRESS 公司的 ezusb. sys 驱动程序[6]。主机应用程序包括动态链接库与操作界面。动态链接库采用 VC++编写，实现与 USB 驱动程序的通信。操作界面采用 VB 编写，实现数据回读、绘制曲线和设备擦除等一系列操控界面。

4 结束语

介绍了一种基于 FPGA 的多通道数模信号实时采编存储系统，文中对系统的功能及硬件、软件的模块化设计进行了详细介绍。采用 USB2.0 与上位机通信，经优化改进，在实际测试中的数据回传速率可达到每秒钟 5M 字节。本系统具有精度高、实时性强、功耗低、体积小及运行可靠等优点，并且已成功应用于某型号弹载实时动态测试试验中。

参考文献：

[1] 谭保华，王 堃，熊健民. 基于 USB 总线技术的高速数据采集卡研究 [J]. 计算机测量与控制，2008，(12).

[2] 杨林楠，李红刚，张丽莲. 基于 FPGA 的高速多路数据采集系统的设计 [J]. 计算机工程，2007，33 (7)：246-248.

[3] 苏晓龙，王香婷. 基于 RS-485 总线的数据采集系统 [J]. 仪表技术与传感器，2007 (12)：52-54.

[4] CY7C68013data sheet [Z]. EZ-USB FX2TM USB Microcontroller.

[5] 黄智伟. FPGA 系统设计与实践 [M]. 北京：电子工业出版社，2005.

[6] 安 荣，任勇峰，李圣坤. 基于 FPGA 和 USB2.0 的数据采集系统 [J]. 仪表技术与传感器，2009，(3)：49-51.

(上接第 1586 页)

3 时钟同步精度影响因素

处于网络节点的 PTP 时钟同步精度受到多方面因素的影响。

(1) 时间戳的位置：IEEE1588 协议中时钟同步的精确度直接取决于时间戳的精确性。在 PTP 系统的实现过程当中，可以根据实际情况，选择不同的时间戳点。如图 4 所示，有 3 个位置可以作为时间戳的选择点[4]。

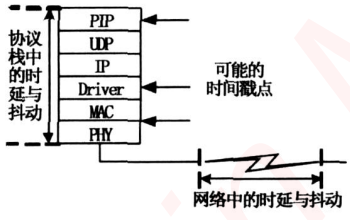

图 4 时钟时间戳点

时间戳越接近以太网底层，即真正意义上的发送和接收位置，时间误差越小，在 PTP 协议实现时，时间戳应做到尽量标记点接近底层，即在 MAC 层和 PHY 层之间的位置，具体实现需要考虑到时间戳如何传递到应用层等细节。

(2) 网络的对称性：若主从时钟之间的消息往返延迟是对称的，则根据本地时钟同步算法，延迟可以相互抵消。但在实际的 PTP 系统实现过程中，主从时钟之间的网络延迟在两个方向上是不对称的，即主时钟到从时钟的网络延迟与从时钟到主时钟的网络延迟往往不是相等的，因此，在系统的实现过程中就必须考虑这些因素[4]。必要时需测量出报文在网络中的延迟时间及延迟在较长一段时间内是否是一个常量。

(3) 同步间隔：通常情况下 PTP 在一定时间内同步的次数越多，网络内时钟的精度越高，但过高密度的同步间隔会影响网络性能，反而降低时钟同步精度，具体同步间隔在设计时可以根据需要进行设置。

(4) 其它影响因素：时钟的稳定性、精确度，网络传输过程中的抖动 (滤波、统计方法来消除)，以及网络中节点报文转发带来的时延 (使其支持 PTP，成为边界时钟可以解决此问题)，网络传输带来的抖动等[5]，这些因素直接决定了 PTP 实现的精度级别。

4 结束语

IEEE1588 提供的分布式控制系统精确时间同步机制，将时钟同步精度提高到亚纳秒级精度，并以其高度的可靠性和开放性，逐渐作为新的时间同步标准，取代标准化网络上的专用时间同步解决方案[5]。本文的研究工作对于 IEEE1588 协议在分布式控制网络中的实际应用具有一定指导意义。

参考文献：

[1] IEEE. Precision clock synchronization protocol for networked measurement and control system [S]. IEC 61588-2009.

[2] John Eidson. Measurement, Control, and Communication Using IEEE 1588 [M]. London: Springer-Vertag New York Inc, 2006.

[3] 袁振华，董秀军，刘朝英. 基于 IEEE1588 的时钟同步技术及其应用 [J]. 计算机测量与控制，2006，(12).

[4] Hans Weibel. IEEE 1588 Implementation and Performance of Time Stamping Techniques [A]. Gaithersbarg: Procedings of NIST Conference on IEEE 1588 [C]. 2004.

[5] Thomas Müller, Alexander Ockert, Hans Weibel. PHYs and Symmetrical Propagation Delay [A]. 2004 Conference on IEEE 1588 [C], Sept. 27-29, Gaithersburg.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统

7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析

Linux:

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究

PowerPC:

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发

ARM:

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计