Proceedings of 20th International Congress on Acoustics, ICA 2010

23-27 August 2010, Sydney, Australia

Precision Time Synchronization Using IEEE 1588 for Distributed Acoustic Measurement

Longhua Ma, Xiaoliang Chen, Bin Chen, Zhaoli Yan

Key Laboratory of Noise and Vibration Research, Institute of Acoustics, CAS, Beijing 100190, China

PACS: 43.58.Ta

ABSTRACT

Distributed measurement based on Local Area Network (LAN) is popular in measuring machines' vibration, of which the timing requirement is becoming increasingly stringent. Traditional synchronal methods, such as Network Time Protocol (NTP), Simple Network Time Protocol (SNTP), can achieve accuracy of microseconds, but it doesn't meet the requirements. At the same time, the sync cable is not competent for long dsitance transmission. A synchronal device based on Precision Time Protocol (PTP, IEEE1588) is designed. A Field Programmable Gate Array (FPGA) is emplaced between Medium Independent Interface (MII) and PHY. Normal data packets pass by without modification, and IEEE 1588 packets are unpacked and stamped according to the accurate time. The FPGA, which also has a management unit to process IEEE1588 events, compensates the drift of crystal oscillator that is notable during the sync interval by using discrete linear Kalman filter. Trigger can be programmed, of which the output is used to driver the sample unit. Acoustic measurement equipments base on the design. Rough experiment shows that it satisfies the need .

1 INTRODUCTION

With the development of network technology, human being has the opportunity to experience the surrounding environment. This will give people a tremendous upgrade the quality of life. Many country and regions has begun the study. Such as China is launching a project called Experience China. Distributed acoustic measurement device is a means to achieve the goals. In order to facilitate analysis, all collected must be synchronized. Synchronization has been investigated for many years，and there many methods to achieve synchronization, such as Global Positioning System (GPS), LORAN-C etc. But it has two disadvantages, one is it is expensive and the other is in some circumstance signal is blocked by shelter. At the same time, the sync cable is not competent for long distance transmission. For the reason above we need technology which is inexpensive, less change and suitable for long distance. Web-based synchronization is a good choice. The most common web-based synchronization method is Net Time Protocol (NTP, RFC-1305) devised by Mills [1] of University of Delaware and its simplified version Simplified Net Time Protocol (SNTP, RFC-2030). They have been used to keep synchronization of internet's clock. NTP and SNTP are not based on the principle of synchronizing machines with other. It is based on the principles of having all machines get as close as possible to the Universal Coordinated Time (UTC). A basic NTP network is composed of a time server and clients. The NTP protocol has hierarchical design in order to prevent large numbers of clients from accessing the same primary time sources. An NTP packet consists of 4 timestamps, the client uses these timestamps to determine the difference between its internal time and UTC time and adjust its local time to coincide with the reference. The client can also determine the network latency and apply a correction factor when it adjusts its internal time. The ability to remove network latency results in a more precise level of synchronization. However, not all the latency can be removed because the paths to and from the server are symmetrical. We develope symmetric star network test environment to analysis synchronization accuracy of SNTP with a Cisco4056, master clock is National Time Service Center (NTSC) time server. Experiment shows that it doesn't meet the requirement [2]. So we need more precise synchronization, IEEE 1588 gives accuracy in submicroseconds, and suitable for our application. Although IEEE 1588 is similar in concept with NTP and SNTP, but it is more complex and advanced implement, it typically provides two or three orders of magnitude better timing accuracy than NTP and SNTP.

2 INTRODUCTION OF IEEE 1588[3]

2.1 Overview of IEEE 1588

Due to the need for synchronization, IEEE 1588 was released as a standard of protocol in 2002 and the standard has also been approved by the IEC as IEC61588. The latest version IEEE 1588-2008 has been approved by IEEE Standards Board Review Committee as an IEEE standard at their 2008 meeting. IEEE 1588 provides a standard procotol for synchronizing clocks connected via a multicast capable network, and it is designed to provide fault tolerant synchronization among hetergenous network clocks requiring very little bandwidth consumption, processing power, and setup.

2.2 Theory of IEEE 1588

A heterogeneous network of clocks is a network containing clocks of varying characteristics, such as the origin of a clock's time source, and the stability of the clock's frequency.

All partcipating clocks synchronize to the highest quality clock in the network. IEEE 1588 defines a standard set of clock characteristics and defines value ranges for each. By running a distributed algorithm, called the Best Master Clock (BMC) algorithm each clock in the network identifies the highest quality clock, which has best set of characteristics.

The best ranking clock is called the grandmaster clock, and all other slave clocks synchronize to the grandmaster. If the grandmaster clock is removed from the network, or if its characteristics change in a way such that it is no longer the best clock, the BMC algorithm provides a way for all the participating clocks to automatically determine which will be the current best clock and new grandmaster. The best master clock algorithm provides a fault tolerant, and administrative free way of determining the clock used as the time source for the entire network.

Slave clocks synchronize to the grandmaster by using bidirectional multicast communication. The packets used by grandmaster clock and slave clock for synchronization consists of four message types exchanged between a master clock and slave clock (see figure 1). These messages are Sync, Follow_Up, Delay_Req and Delay_Resp. Sync packet contains a timestamp of the time when the packet left the grandmaster clock. The grandmaster may also, optionally, issue a Follow_Up packet containing the timestamp for the Sync packet. The use of a separate Follow_Up packet allows the grandmaster to accurately timestamp the Sync packet on networks where the departure time of a packet cannot be known accurately beforehand. For example, the collision detection and random back off mechanism of Ethernet communication prevents the exact transmission time of a packet from being known until the packet is completely sent without a collision being detected, at which time it is impossible to alter the packet's content.

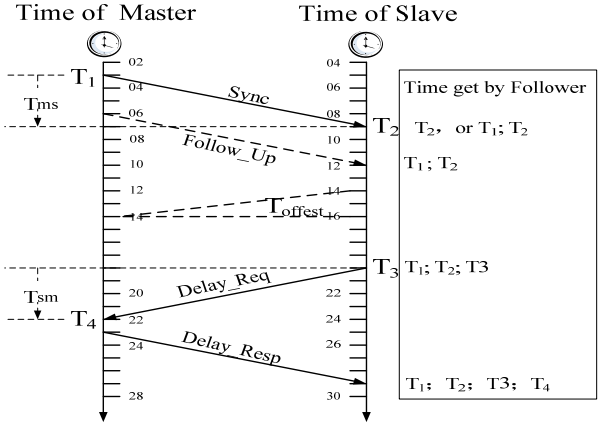

Figure 1. Basic synchronization message exchange

A slave clock receives the grandmaster's Sync packet and timestamps the packet's arrival time using its own clock. The salve clock waits a random time and send Delay_Req packet to master clock. The master clock records the time which the Delay_Req is accept and stamp the time in Delay_Resp packet. After Delay_Resp packet is accepted by slave clock, the slave gets all the need to calculate the offset between master and salve clock.

We assume that the offset between master clock and slave clock is T_{offset}. The master clock sends Sync message to slave and notes that at the time T_1, the Sync message was sent. The slave receives this message after a delay equal to transit time from master to slave (T_{ms}) at T_2. T_3 is the time Delay_Req is sent. The master receives the Delay_Req message after a delay of time (T_{sm}) at time T_4. Through a series of message exchanged, the slave gets all four time needed for synchronization. We can obtain the following equation from Figure 1.

$$T_2 - T_1 = T_{ms} + T_{offset} \quad (1)$$

$$T_4 - T_3 = T_{sm} - T_{offset} \quad (2)$$

For a symmetry network, we assume that T_{ms} equal to T_{ms} and get equation 3 and 4.

$$T_{ms} = T_{sm} = \frac{(T_2 - T_1) + (T_4 - T_3)}{2} \quad (3)$$

$$T_{offset} = \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \quad (4)$$

The accuracy of T_{offset} determines the synchronization accuracy. We can use the offset to adjust the clock of slave.

3 FREQUENCY COMPENSATION

IEEE 1588 gives out the time offset measurement method, but the protocol doesn't discipline the clock, a simple and reliable frequency compensated algorithm in this section is introduced to resolve the question. In practice, all crystal oscillators are not running at same rate. A complete description of crystal oscillator behavior is difficult, but it is a good approximation in a short period of time as described in figure 2.The relation between real time and local node time can be described in equation 5:

$$T(t) = t + st + t_0 \quad (5)$$

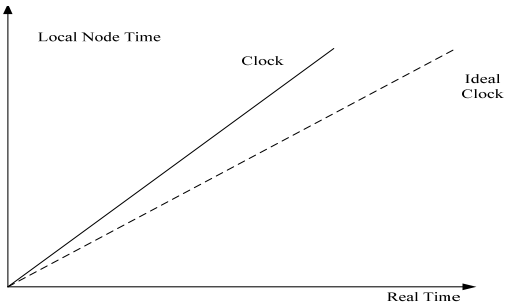

Figure 2. Drift of clock

Where T(t) is local time, s means clock skew, t_0 is offset between ideal clock and local clock at time t ,and t is ideal time(master clock). Then, the clock time error between local and local time is:

$$E(t) = -st - t_0 \quad (6)$$

The constant s can be calculated by Discrete Linear Kalman Filter (DLKF). In 1960, R.E.Kalman published his famous paper describing a recursive solution to the discrete data linear filtering problem. Since that time, the Kalman filter [4] has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. It is a recursive filter that minimizes the mean square error [5]. The Kalman filter is an optimal estimator when the variations in the tracked signal are Gaussian distributed. The Kalman gain effectively determines the bandwidth of the filter. This can be adjusted by choosing appropriate values of the process and measurement noise which doesn't change the filter delay. If timestamps are recorded close to PHY or PHY and there is less or no messages delay effected by traffic load. The skew distribution can therefore be assumed stationary. The Kalman gain converges to a constant value that can be pre-calculated. This reduces the filter computational overhead to one subtraction, one multiplication and one addition operation. The properties above make Kalman filter suitable for clock skew estimation.

We asumme that Sync message is sent by grandmaster clock every T seconds and the slave clock measures an offset of Δt_k in time step k. Let S_k represent the actual skew in time step k, we get:

$$S_k = S_{k-1} + V_k \tag{7}$$

Where V_k is process noise, i.e., deviations from model due to oscillator jitter. This is a zero mean, Gaussian distributed, random variable. Let S_k^* represent the measured skew at the slave clock in time step k. Then,

$$S_k^* = S_k + W_k \tag{8}$$

Where W_k represents the measurement noise. Equation (7) and Equation (8) represent the standard model for DLKF. We let:

Q and R represent the variance of V and W,

$\hat{S}_k$ represents the estimated skew in time step k,

$\hat{S}_k^-$ represents the predicted estimate of clock skew,

P_k^- represents predicted error variance,

K_k represents the Kalman gain,

P_k represents the corrected error variance.

The Kalman filter recursive equations are reported below.

Prediction:

$$\hat{S}_k^- = \hat{S}_{k-1} \tag{9}$$

$$P_k^- = P_{k-1} + Q \tag{10}$$

Correction:

$$K_k = P_k^-(P_k^- + R)^{-1} \tag{11}$$

$$\hat{S}_k = \hat{S}_k^- + K_k(S_k^* - \hat{S}_k^-) \tag{12}$$

$$P_k = (1 - K_k)P_k^- \tag{13}$$

The skew estimated by the filter is used as input to Frequency Compensation Real Time Clock (FCRTC) [6]. The FCRTC is comprised of a *p*-bit clock counter, a *q*-bit accumulator as high precision frequency divider and *r*-bit addend register holding frequency compensated value. All the constituents of module operate at the frequency of input oscillator *f* .The compensated value contained in register is added to accumulator once every 1/*f*. The FCRTC's diagram is shown in Figure 3.

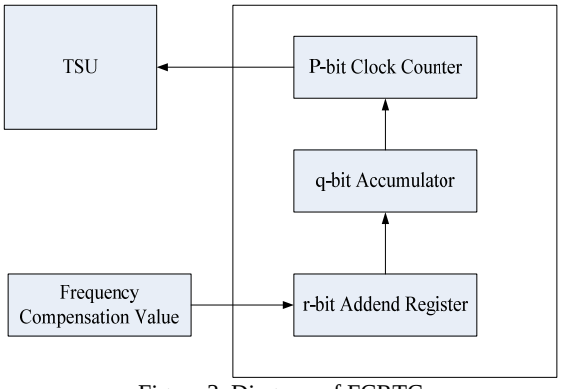

Figure 3. Diagram of FCRTC

The value of *p*, *q* and *r* can calculate in following equations:

$$C_f \le \frac{1}{f \times S_i} \tag{14}$$

$$2^p \ge \frac{1}{C_f} \tag{15}$$

$$2^r \ge 2^q \tag{16}$$

$$2^p \ge 2^q \tag{17}$$

Where C_f is accuracy of compensated value, S_i is a number representing interval between sync messages in seconds. Then the compensated value can obtain compensated value C_c by equation 18:

$$C_c = \frac{\hat{S}_k^-}{f \times S_i} \tag{18}$$

By default, the sync interval is 1 or 2 seconds in most system. For example, we assume that sync interval is 1 second, and the frequency of crystal oscillator is 44MHz, the accuracy compensated value is 1×10^{-9}, width of accumulator is 30. In order to facilitate the design, the width of accumulator is 32. Width of addend register *r* is 32. Width of clock counter is 64 to provide 22.73 nanoseconds.

4 HARDWARE IMPLEMENTATION

At present, there two IEEE 1588 implementations, one is software-only implementation and the other is hardware assisted software implementation [7]. In software implementation, the whole protocol is executed at application level. Therefore, it yields the least accuracy as the most errors are introduced into the timestamp, especially because of the fluctuations in the protocol network. Typically, errors range in the order of hundreds of microseconds to milliseconds, depending on the operating system in a software implementation. In the hardware assisted software implementation[8], the high precision time is generated while reading the time stamp on the synchronization packets in both transmit and receive directions，which leads to greater accuracy between tens of nanoseconds to several hundreds nanoseconds.

In order to achieve the required accuracy, we adopt hardware assisted implementation as our implementation. The Diagram is shown in figure 4.

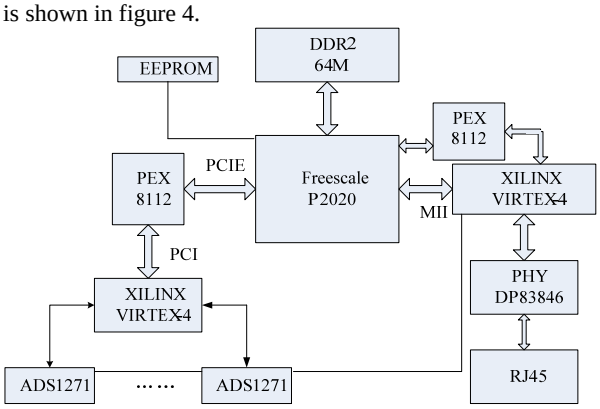

Figure 4. Block Diagram of Implementation

Figure 3 shows the approximate hardware implementation. System adopts Freescale P2020 as CPU, it has two cores. One core is used for application programme, and another is used for IEEE 1588 applocation. The system has 64M DDR2 memory; it is enough for our application. Freescale P2020 has PCI-Express bus (Pedpherd Component Interconnect bus); it is translated PCI. One of Xilinx Virtex-4(referred as FPGA) achieves control of ADS1271, packages sample data and uploads data. Another FPGA is inserted between MII (Me-

dium Independent Interface) and PHY. The FPGA detecting preamble of all incoming and outgoing peckets to determine whather a packet contains timestamp. If a packet does not contain timestamp, the packet is transmitted directly without any modification. If a pecket contains timestamp, the timestamp is recorded or modified by the demand of the protocol. The FPGA also achieves Time Stamp Uint (TSU) and FCRTC. The input of FCRTC is skew estimation of local clock. The FPGA diagram is shown in figure 5.

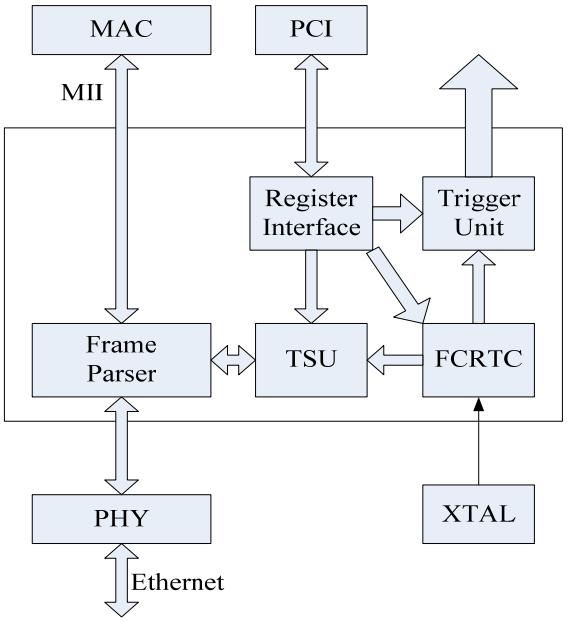

Figure 5. Diagram of FPGA

The Exteral Crystal Oscillator (XTAL) drivers FCRTC, and frequency compensation value is add to addend register to adjut local clock. The local clock's output can be program divided to the frequency which we need and drives ADS1271 to sample data.

5 CONCLUSION AND FUTUER WORK

The synchronization implementation with IEEE 1588 in this paper is under testing. It can provide a simple and accurate way to distributed acoustic measurement. We let one device as grandmaster clock and another as slave clock. Two device's PPS (Pluse per Second) are conneted to an oscilloscope. Test show the accuracy ranges from 20ns to 100ns. In the future synchronization error's statistical properties will show the final result.

REFERENCES

1 D. L.Mills, Internet Time Synchronization: The Network Time Protocol. IEEE Transactions on Communications COM 39 no. 10, p. 1482–1493, October 1991.

2 CHEN Xiaoliang. Precise time synchronization for networked acoustic measurement, Technical Acoustics.

3 IEEE Std 1588-2008, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems [S], 2008.

4 R. E. Kalman, A New Approach to Linear Filtering and Prediction Problems, Transactions of the ASME–Journal of Basic Engineering, D, vol. 82, pp. 35–45, 1960.

5 G. Welch and G. Bishop, An introduction to the Kalman filter. Department of Computer Science, University of North Carolina, Chapel Hill, NC, Tech. Rep. 95-041, July 2006.

6 S. Balasubramanian, K. R. Harris, and A. Moldovansky, A frequency compensated clock for precision synchronization using IEEE 1588 protocol and its application to Ethernet, presented at Proc. of the Workshop on IEEE 1588, Gaithersburg, U.S., 2003.

7 J. Guilford, Design of an FPGA-BasedHardware IEEE-1588 Implementation, IEEE-1588 Conference, September 2005.

8 P. Sharma, Hardware Assisted IEEE 1588 Implementation in a Next Generation Intel® Network Processor, IEEE-1588 Conference, September 2004.