Computer Standards & Interfaces 49 (2016) 22–33

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Synchronization of multiple USB 3.0 devices using Isochronous Timestamp Packet

Chong Han Lim[a,*], Bakhtiar Affendi bin Rosdi[a], Chee Fai Yap[b]

[a] School of Electrical & Electronic Engineering, Universiti Sains Malaysia Engineering Campus, Seberang Perai Selatan, Nibong Tebal, Penang 14300, Malaysia
[b] NI Malaysia Sdn. Bhd. No. 8, Lebuh Batu Maung 1, 11960 Bayan Lepas, Penang, Malaysia

ARTICLE INFO

Article history:
Received 19 January 2016
Received in revised form 10 May 2016
Accepted 30 June 2016
Available online 4 July 2016

Keywords:
Synchronization
USB3.0
Triggering
Isochronous Timestamp Packet
Data acquisition

ABSTRACT

In test and measurement, synchronizing multiple devices to timely coordinate data acquisition is crucial. This is due to the limited number of channels on a single device and the need for mixed-signal channels. Currently, various techniques have been proposed for triggering and synchronization among multiple devices such as GPIB, VXI, PXI/PXIe and LXI. However, they are expensive and need large footprint and additional cable or hardware to set up. Due to that, with the low cost and easy-to-setup USB 2.0, it is enhanced to have synchronization ability. There are various synchronization methods in USB 2.0 such as software-timed trigger, external hardware trigger and USB-inSync. The software-timed trigger is purely implemented in software and has high trigger delay, which is up to microseconds. Besides that, for both external hardware trigger and USB-inSync, they can achieve up to nanoseconds of synchronization precision. However, both of those methods need extra hardware or setup to synchronize multiple devices. In this paper, to overcome the drawbacks from the existing techniques, a trigger algorithm is proposed based on the study of USB 3.0 technology. The Isochronous Timestamp Packet (ITP) in USB 3.0 is used as the main component in the trigger algorithm. Without any hardware implementation, the trigger algorithm is able to trigger multiple USB 3.0 devices with a trigger delay around 200 ns. In addition, the trigger algorithm can support up to maximum to three consecutive connected hubs without affecting the performance. In conclusion, the trigger algorithm increases the performance of synchronization of multiple USB device with only software intervention.

1. Introduction

In this modern age, many experiments require measurements of various physical quantities in one test [1]. There is a problem of limited channels in a single device to acquire multiple types of data. To solve this, multiple measurement devices are used together to acquire multiple data. Due to that, data from multiple measurement devices need to be synchronized so that the data will be timely coordinated and processed at the same time. With data synchronization, data acquisition becomes more accurate and precise. Therefore, the quality of products or researches can be enhanced.

There are numbers of test and measurement buses are introduced to synchronize multiple instruments such as General Purpose Interface Bus (GPIB), Versa Module Europa (VME) extension for Instrumentation (VXI), Peripheral Component Interface (PCI) extension for Instrumentation (PXI/PXIe) and Local Area Network (LAN) extension for Instrumentation (LXI). In 1972, GPIB is introduced, which was designed by Hewlett Packard. It was brought forward to realize the clock synchronization and triggering of multiple instruments [2]. In 1987, a new standard of instrument architectures - VXI is introduced. This new architecture is to address the limitation of GPIB and further promote a standard not only for test and measurement, but for other industries as well [3].

Besides that, with the purpose to increase the rate of data transfer and to be more compatible with personal computer (PC), the PXI based on the PCI bus was proposed in 1997 and was launched in 1998 [4]. Tight timing and synchronization is implemented into PXI to provide faster data transmission and higher precision of clock synchronization. Furthermore, followed by the ubiquity of internet technology, synchronization of multiple measurement devices through the network becomes possible. In 2004, the LXI bus based on LAN was proposed and was released in 2005 [5]. LXI is based on the well-established Ethernet technology to network measurement devices and can embed multiple devices seamlessly into the available LAN in the lab or office.

However, even though these buses provide high precision of synchronization, they are either expensive, take longer time to setup or require large footprint [6]. Due to that, the cheap and easy-to-use Universal Serial Bus (USB) becomes popular and is enhanced to

* Corresponding author.

http://dx.doi.org/10.1016/j.csi.2016.06.006
0920-5489/

support synchronization ability [7]. USB had been used in industrial application as a low cost data acquisition device [8]. USB is introduced in 1995 and is maintained and developed by a non-profit organisation - USB Implementers Forum (USB-IF). Currently, five versions of USB have been introduced which are USB 1.0, USB 1.1, USB 2.0, USB 3.0 and the latest USB 3.1 [9].

There are many trigger and synchronization methods introduced for USB such as USBinSync [7], software-timed trigger and external trigger [10]. USB-inSync provides up to picosecond levels of synchronization precision. However, extra circuits and hardware are needed to implement this method and this increases the cost and time to produce it. Besides that, software-timed trigger uses software application to send a trigger signal to the device to trigger an event. Trigger signal is sent to each connected device one by one. Therefore, it has higher delay and low synchronization precision. On the other hand, the external trigger uses external signal such as power source to trigger an event in a device. However, this method needs to set up an extra circuit to supply the external trigger source to device.

Due to the drawbacks of each method in USB triggering and synchronization, a new algorithm based on USB 3.0 technology is proposed in this paper to trigger multiple USB 3.0 devices at the same time. With this proposed algorithm, there is no need any implementation of circuit or hardware into USB 3.0 protocol and is able to trigger multiple USB 3.0 devices at the same time by only using software. It also improved the trigger delay performance from previous methods in USB. In this proposed algorithm, the new added Isochronous Timestamp Packet (ITP) in USB 3.0 is utilized to trigger multiple USB 3.0 devices at the same time.

The paper is structured as follows: in Section 2 the proposed algorithm is presented; in Section 3 the implementation of proposed algorithm in both host PC and USB 3.0 devices is explained: Section 4 discusses the experimental setup for the performance assessment of the proposed algorithm, presents the results from the experiments and discusses the proposed algorithm's performance by comparing it to previous synchronization methods in USB and also the performance in different USB hub depths; Section 5 concludes the work in this paper.

2. Proposed algorithm

In 2007, USB 3.0 was introduced with improvements from previous version such as 5 Gb/s of transfer rate and better power management. In USB 3.0, there is a new added packet named Isochronous Timestamp Packet (ITP) which is used to carry host timing information to all active devices. This ITP is used as a main component in our proposed algorithm.

ITP is periodically multicast by host to all active connected devices at each bus interval boundary. Therefore, each active connected device will receive the same ITP at the same time. In ITP, there is a bus interval counter which counts from zero to 16,383 and then roll over to zero. Due to the same bus interval counter is received at the same time at each device, therefore, we can use it as a trigger source to trigger multiple USB 3.0 devices.

The proposed algorithm involves both USB 3.0 host and USB 3.0 devices. On the USB 3.0 host side, a user interface application is created to apply the proposed algorithm to calculate which bus interval counter the device(s) is to be triggered. On the other hand, the USB 3.0 device's firmware is programmed so that the device will be able to handle the commands from the host.

Fig. 1 shows the flow chart of the algorithm. The concept of the algorithm is the host calculates a specific bus interval counter value (SBIV) to set when of each connected device to start their event. The SBIV is calculated by the host based on the current bus interval counter value (CBIV) and the total time taken for a USB packet to travel from the host to each connected device (T). The calculated SBIV is sent to each connected device and each device will always compare the SBIV to the bus interval counter value in every received ITP. If both values are the identical, the device will start its event.

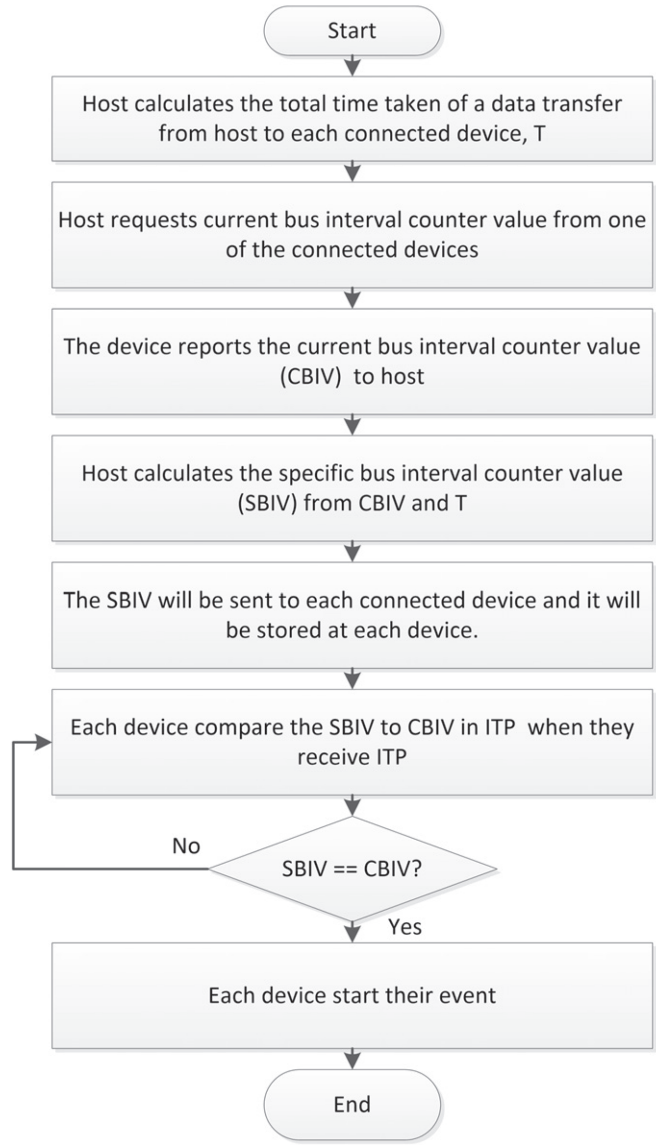

Fig. 1. Flow chart of the algorithm.

From the flow chart of the proposed algorithm in Fig. 1, host will first calculate the T which is used to compensate the packet transmission delay to each device. As USB services each device one at a time, the same data packet that is to be sent to multiple devices is sent at different times to the first device and to the last device. This may cause the last device to receive the same information later. As an example, if the host sends the same SBIV with a value of 1000 to all connected devices, the first device receives it before the CBIV of 1000, then second device receives it after the CBIV of 1000, therefore, first device will trigger first while second device has to wait until the next roll to trigger. Fig. 2 shows the example of algorithm without considering T.

To solve this, the calculation of total time taken for a packet to travel from the host to each device (T) is included in the algorithm. Fig. 3 shows the time taken of a packet to travel to each device from the host. Eq. (1) shows the formula to calculate T.

$$T = T_1 + T_2 + \ldots + T_n \quad (1)$$