논문 19-44-10-03 The Journal of Korean Institute of Communications and Information Sciences '19-10 Vol.44 No.10
https://doi.org/10.7840/kics.2019.44.10.1812

정밀 시간 동기 시스템을 위한 자리올림 오류가 없는 새로운 주파수 보정 모듈의 구조와 성능분석

이 정 도◆, 박 부 식°, 손 명 환*, 윤 종 호**

A Novel Architecture of Frequency Compensation Module without Round-Over Errors for Precision Timing Protocol Systems

Jeong-do Lee◆, Pu-sik Park°, Myeong-hwan Son*, Chong-ho Yoon**

요 약

자동차, 항공 우주 및 군수 산업과 같이 정밀 제어 및 측정 자동화를 필요로 하는 산업은 이더넷 기반 IEEE 1588 시간 동기화 프로토콜(PTP)에 의해 제공되는 결정적 시간 기준을 필요로 한다. 시간 동기 시스템에서 모든 슬레이브는 마스터로부터 1초(또는 1/8초, 또는 2초)마다 주기적으로 PTP 메시지를 전송받아 마스터 시간과 동기화한다. 이를 위해 PTP 모듈 내부에는 보정 시스템이 내장되어 있다. 기존 보정 시스템은 클럭 주파수 보정에 대한 가산 값이 변경되는 오류가 발생한다. 이런 문제를 해결하기 위해 Addend 및 Increment 레지스터 값을 소수 대신 정수로 출력하여 Second 레지스터로 트리거 신호를 출력 할 때 누산기(Accumulator)의 나머지 값이 없게 보정 시스템을 수정하였다. 이 방식은 Subsecond 레지스터의 값이 지속적으로 변경되는 오차를 제거함으로써 시간 동기 시스템에서 마스터와 슬레이브 간에 시간 오프셋이 발생하지 않는다. 제안된 주파수 보정 모듈의 성능은 MATLAB을 사용하여 확인하였다.

Key Words : Frequency compensation, IEEE1588, Precision Time Protocol, PTP, Time synchronization

ABSTRACT

Industries requiring automation of measurement, such as automobiles, aerospace, and military industries, need a deterministic time reference provided by Ethernet-based IEEE 1588 time synchronization protocol. In a time synchronization system, all slaves receive a PTP message periodically every 1 second (or 1/8 second, or 2 seconds) from the master and synchronize with the master time. To this end, a correction system is built in the PTP module. In the existing correction system, an error occurs that the addition value for the clock frequency correction is changed. To solve this problem, the correction system was modified so that when the trigger signal is output to the Second register by outputting the Addend and Increment register values as integers instead of decimals, the remainder of the accumulator is not present. This scheme doesn't cause a time offset between the master and the slave in the time synchronization system by eliminating the constantly changing error of the value of the Subsecond register. It is evaluated by using MATLAB.

※ 본 연구는 국토교통부가 지원하는 철도기술연구사업의 일환으로 수행되었습니다.(19RTRP-B109166-05)
※ 본 연구는 과학기술정보통신부가 지원하는 정보통신방송연구개발사업의 일환으로 수행되었습니다.(2018-0-00846)
◆ First Author : Korea Electronics Technology Insititute, ljdhihi@keti.re.kr, 정회원
° Corresponding Author : Korea Electronics Technology Insititute, pusik@keti.re.kr, 종신회원
* Korea Electronics Technology Insititute, mhson@keti.re.kr
** Korea Aerospace University Aviation Electronics Information Engineering, yoonch@kau.ac.kr, 종신회원
논문번호 : 201907-131-A-RN, Received July 8, 2019; Revised August 7, 2019; Accepted August 7, 2019

Ⅰ. 서 론

최근 차량, 항공, 우주, 군사 산업 등과 같은 근거리 통신 분야에 디지털화가 진행되고 실시간 영상 및 음성 처리 시스템이 탑재되어 데이터량이 증가함에 따라, 하드웨어의 고성능, 고신뢰성이 보장되어야 하고 분산된 장비들에 대한 정밀한 시간 기준이 제공되어야 한다[1]. 이를 위하여 NTP (Network Time Protocol), IRIG (Inter-Range Instrumentation Group), PTP (Precision Time Protocol) 등의 다양한 시간동기 관련 프로토콜이 개발되었다. NTP는 국제 표준시계들(UTC나 GPS와 같은)부터 시작하여 계층 구조로 분산 네트워크를 이루며 동기화를 진행하는데, 네트워크상에서 주고받는 패킷의 정보를 통해 서로간의 전파지연시간(Delay)과 시간오차(offset)등을 계산하게 된다[2]. IRIG는 시간정보가 실린 코드 포맷 표준이며 포맷의 종류는 Bit Rate에 따라 A, B, D, E, G, H가 있다. PTP는 이더넷 대기시간과 지터 문제를 네트워크 물리계층에서 하드웨어 타임스탬핑을 통하여 해결함으로써 높은 정확도 가질 수 있다[3].

위의 시간동기 프로토콜들은 각 기술별로 시간오차 정밀도 차이가 있다. NTP는 클라이언트와 서버 사이의 홉수와 네트워크 지연시간에 따라 다른 정밀도를 갖는다. WAN에서는 10~100ms 정도이며 LAN에서는 2~10ms 정도로 매우 정확한 시간을 필요로 하는 분야에서 사용하기에는 정밀도가 다소 떨어진다. IRIG는 1~10μs의 정밀도를 제공하며 정밀도는 NTP에 비해 높지만 IRIG 신호를 전달하기 위한 전용 케이블을 설치하여야 하기 때문에 구축 비용이 증가하고 추가적인 관리 부담이 증가하게 되는 단점이 있다. PTP는 이더넷 LAN을 사용함으로 인해 NTP처럼 비용이 적게 소요되고 IRIG의 정밀도를 능가한다[3]. PTP를 지원하는 스위치를 사용하면 20~100ns의 동기 정밀도를 얻을 수 있다.

이 중에서 가장 정밀한 프로토콜인 PTP는 <그림 1>과 같이 주파수 보정 모듈의 Addend 및 Increment 레지스터의 값을 가산기(adder)를 통해 누산기(Accumulator)에 누적시켜 시간을 생성한다. Increment 레지스터는 사용자가 정의하는 Tick 값에 의해 정해지고 Addend 레지스터는 SystemClk, Increment 레지스터, 누산기 크기에 의해 초기 값을 계산한다. 그 후 슬레이브가 주기적으로 마스터와 메시지를 교환 하여 얻은 시간오차(Offset)을 통해 Addend 레지스터 값을 변경하여 마스터의 시간에 동기화한다. 하지만 지속적으로 보정하여 마스터의 시간

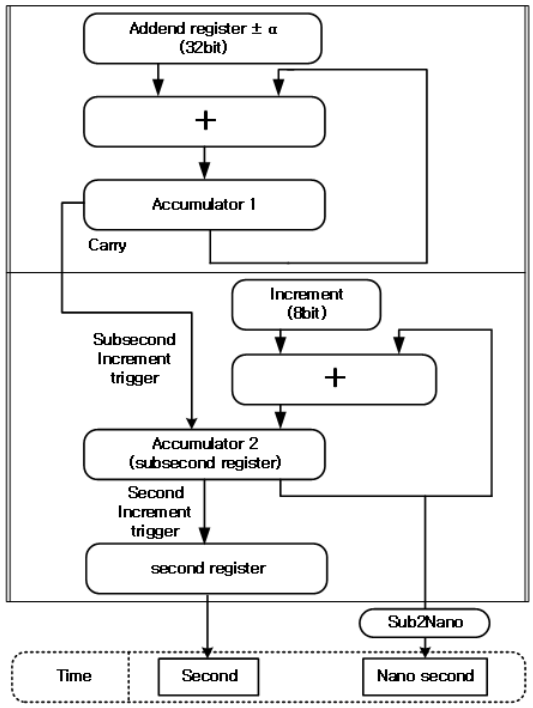


그림 1. 기존의 주파수 보정 모듈의 구조
Fig. 1. Structure of existing frequency correction module

에 동기화함에도 불구하고 Addend 및 Increment 레지스터 값들은 소수점 이하의 손실이 생기게 되고 가산기를 통해 누산기에 누적되어 다음 자리올림이 발생하는 시간에 지속적으로 영향을 미치게 되어 비주기적인 자리올림으로 인해 결국 Second와 Nanosecond를 생성하는 주파수 보정 모듈에 시간 오차가 발생하여 PTP의 정확도를 떨어뜨린다.

이러한 문제를 해결하기 위해 타임스탬프를 추가하거나 PTP 절차 후 추가 메시지를 보내는 등의 알고리즘을 만들어 PTP의 정확도를 향상시키는 연구들이 있었다[4,5].

본 논문에서는 이러한 문제를 해결하고 PTP의 정밀도를 더 향상시키기 위해 알고리즘은 변경하는 대신 기존의 주파수 보정 클럭 구조를 변경한 시간오차 저감형 주파수 보정 클럭을 제안하고 성능분석을 수행하였다.

제안된 시간오차 저감형 주파수 보정 클럭은 2n 크기의 n-bit크기 누산기와 가산기로 구성된 기존 주파수 보정 모듈을 개선하였다. 비교기(Comparator), 셀렉터(Selector) 그리고 감산기(Subtracter)를 추가하여 누산기를 정확한 시간에 자리올림 하도록 변경하였다. 또한 Increment 및 Addend 레지스터 계산식이 바뀌

어 소수점 버림이 발생하지 않기 때문에 정확한 계산이 된다. 이러한 방식은 Subsecond 레지스터의 값이 지속적으로 변경되는 오차를 제거함으로써 시간동기 시스템에서 마스터와 슬레이브 간의 시간오차가 발생하지 않도록 한다. 제안된 방식에 대하여 MATLAB을 통하여 기존 방식과 비교하였다.

논문의 구성은 다음과 같다. 본 서론에 이어, 제 2장에서는 시간동기 기술과 종류에 대해 다룬 후 제 3장에서는 PTPv2 프로토콜에 대하여 상세히 기술한다. 제 4장에서는 기존 주파수 보정 모듈의 문제점을 분석한 후, 자리올림 오류가 없는 새로운 주파수 보정 모듈을 제안하여 두 모듈의 성능을 비교분석 하고 마지막으로 제 5장에서 결론을 맺는다.

Ⅱ. 시간동기 기술 개요

이 장에서는 기존 시간동기 기술들을 소개하고 기존 기술들의 각각 어떤 특징이 있는지 소개한다.

2.1 시간동기

시간 동기는 다음과 같이 Syntonized 및 Synchronized 동기로 구분된다. Syntonized는 <그림 2>의 왼쪽 그림과 같이 주기는 동일하나 위상 차이를 알 수 없고 SDH/SONET 등에 사용된다. Sychronized는 <그림 2>의 오른쪽 그림과 같이 주기와 위상이 동일하고 PTP, NTP 등에 사용된다.

- Syntonized Clock: 마스터 클럭과 슬레이브 클럭이 존재할 때, 슬레이브 클럭이 마스터 클럭의 내부 클럭을 정확한 주기로 참조하여 클럭주기가 동기된 클럭
- Synchronized Clock: 마스터 클럭과 슬레이브 클럭이 존재할 때, 슬레이브 클럭의 장치가 마스터 클럭이 참조하는 시간정보(연,월,일,시,분,초등)에 동기되는 클럭

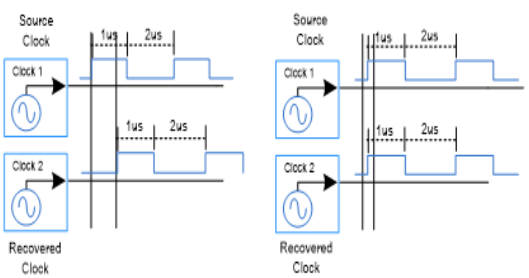


그림 2. Syntonized 및 Synchronized 클럭의 비교
Fig. 2. Comparing Syntonized and Synchronized Clocks

2.2 시간동기 프로토콜

서론에서 이미 언급했듯이, 시간동기 프로토콜은 NTP, IRIG-B, PTP 등이 있다. 이 프로토콜들은 정밀도가 가장 높은 클럭인 GMC(Grand Master Clock)가 생성한 현재의 시간정보를 메시지에 수납하여 RS-485, 이더넷 등의 전달 수단을 통해 전달함으로서 시스템 내의 다른 장치들이 동일한 시간정보를 유지하도록 한다.

- NTP: NTP는 라우터로 연결되어 가변적인 지연을 초래하는 인터넷을 통하여 GMC와의 ToD를 설정하는 절차이다. 특히 가변적인 지연을 갖는 인터넷의 특성을 고려하여 여러 개의 GMC에 대한 시간정보를 기반으로 정확한 시간을 설정할 수 있도록 Marzullo's 알고리즘을 사용한다[6].
- IRIG-B: IRIG time code B는 100 pps (10 milliseconds) 전송율로 BCD 형식의 time-of-year 값을 전달한다. 각 IRIG-B 펄스 신호형식은 pulse-width coded 신호형식의 디지털 신호 ("unmodulated IRIG-B") 또는 1KHz 정현파로 AM 변조한 아날로그 신호 ("modulated IRIG-B") 이다. 또는 Modified Manchester 방식도 있다.
- PTP : 1990 년대 말 계측기 전문회사인 Agient Technology 사에서 개발된 PTP 프로토콜은 분산된 노드들간의 측정한 데이터 사이에서 명확한 시간 순서를 정하는 문제를 해결한다. 이것은 2002 년 11 월 IEEE 1588v1(PTPv1) "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems"로 표준화되었다.

2008 년에는 PTP 가 적용되는 네트워크 요소중 브리지, 스위치 등의 장치에서 발생하는 가변적인 지연시간을 보정하기 위해 IEEE 1588v2(PTPv2)로 개정되었다. 현재 PTPv2 는 해당 네트워크의 고장감내(Fault Tolerance) 기능을 위해 BMC(Best Master Clock) 알고리즘을 바탕으로 네트워크에 참여하는 각 노드들이 자체적으로 최적의 GMC를 협상할 수 있도록 기능도 함께 제공하고 있다[7].

분산시스템 상에서 보다 정밀한 시간동기를 위해 PTP를 사용하는 주된 이유는 소프트웨어로 구현되어 ms 이상의 지연시간이 발생하는 NTP와는 달리 PHY 칩과 MII(Media Independent Interface)의 타임스탬

프를 직접 참조하여 프로토콜의 동작을 수행하는 PTP의 시간오차가 10-9초 단위의 지연시간으로서 보다 정밀한 시간동기가 가능하기 때문이다.

Ⅲ. PTPv2 정밀 시간동기 프로토콜

이 장에서는 PTPv2 정밀 시간동기 프로토콜의 시간동기화 절차, 타임스탬핑 방법, 제약사항, 시간오차 보정방법, Clock servo, 주파수 보정 모듈에 대해 설명한다.

PTP 는 클럭 동기화 프로토콜의 표준으로서 하나 이상의 노드를 가지는 분산네트워크 시스템 상에서 각 노드들이 타임스탬핑(메시지 송신시의 시각을 기록하는 방법), 실시간 시스템의 관리와 같은 목적을 수행하기 위해 RTC(Real Time Clock) 을 기반으로 네트워크에 참여하는 노드간의 정밀한 클럭동기화 메커니즘을 제공한다.

PTP 시스템은 분산네트워크 시스템을 기반으로 보통클럭(Ordinary Clock), 경계클럭(Boundary Clock), 투명클럭(Transparent Clock) 등의 PTP 클럭 장치와 브리지와 같은 일반 이더넷 장치들로 구성된다.

상기 PTP 프로토콜에서 마스터클럭를 제외한 각 노드(슬레이브클럭)가 마스터클럭과의 클럭동기화를 위해 IEEE1588규격에 명시된 PTP 메시지를 통해 마스터클럭과의 시간정보를 교환함으로서 각 노드가 마스터클럭과 시간을 동기화할 수 있다. 이때 네트워크 상의 각 노드는 자신의 역할로써 마스터클럭 또는 슬레이브클럭의 관계를 설정하는 Best Master Selection 절차인 BMC 기능을 수행한다. 이를 위하여 각 노드는 자신의 로컬클럭(Local Clock)에 대한 정보를 Sync메시지 또는 Announce 메시지에 수납하여 주기적으로 송신하여 자신의 능력을 광고한다. 이 결과로 각 노드는 상대적인 클럭 우선순위(Clock Priority)를 판단하여 GMC(Grand Master Clock) 을 결정한다.

3.1 시간동기화 절차

클럭 동기 과정은 크게 'One-Step Clock' 과 'Two-Step Clock' 방식으로 구분된다. 먼저 One-Step 방식은 MAC-PHY 를 통해 송수신되는 PTP 의 'Sync' 메시지영역 중 <originTimestamp>를 하드웨어단의 타임스탬핑 장치가 상시 기록해야하는 별도의 기능이 존재해야 한다.

반면에 Two-Step 방식은 송수신 시점을 'Sync' 메시지의 송신 바로 직후 뒤따라 송신되는 'Follow_Up' 메시지를 통해 상기한 'Sync' 메시지의 송신시점을

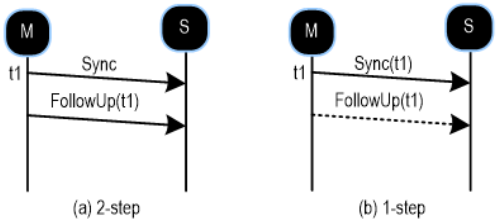


그림 3. 클럭 동기 방법
Fig. 3. Clock synchronization method

상대방 노드(슬레이브클럭)에게 알린다[8].

PTP에서 사용되는 기본적인 시간동기 절차에서 사용되는 시간은 크게 { offset, delay, drift }로 구분된다.

- Offset : 마스터클럭과 슬레이브클럭간의 시간차이다.
- Delay : 마스터클럭과 슬레이브클럭간의 전파지연시간이다. 여기에는 경로상의 브리지 내부에서의 residence time도 포함된다.
- Drift : 마스터클럭의 송신주기와 슬레이브클럭의 수신주기간의 오차율이다.
- Latency : 내부에서의 송신/수신지연시간이다.

<그림 4>와 같이 {Sync, Followup, Delay_Req, Delay_Resp} 메시지를 사용하여 마스터클럭과 슬레이브클럭 간의 시간오차(Offset)와 전파지연시간(Delay)을 동시에 얻어 동기화를 수행한다. <그림 4>에서 A는 마스터 노드에서 전송한 'Sync' 메시지를 슬레이브 노드에서 수신할 때까지 걸리는 시간으로 오프셋과 딜레이를 더한 값이고 B는 슬레이브 노드에서 전송한 'Delay_Req' 메시지를 마스터 노드에서 수신할 때까지 걸리는 시간으로 딜레이에서 오프셋을 뺀 값이다.

시간동기를 위해 마스터가 'Sync' 메시지를 슬레이브에게 전송한다. 이때, 'Sync' 메시지는 'Announce' 메시지의 전송과 마찬가지로 <originTimestamp> 에 0을 수납하여 전송하고, 실제 송신시각 t1 은 뒤따르는 'Follow_Up' 메시지에 수납되어 전달된다. 'Sync' 메시지를 수신한 슬레이브는 수신한 시각 t2를 버퍼에 저장한다. 슬레이브는 마스터에 'Delay_Req' 메시지의 전송 시점에 타임스탬프 t3 를 생성 후 이를 저장하고 'Delay_Req' 메시지를 전송한다. 슬레이브 노드가 송신한 'Delay_Req' 메시지를 수신한 마스터는 수신한 시각 t4를 버퍼에 저장하고 슬레이브에게

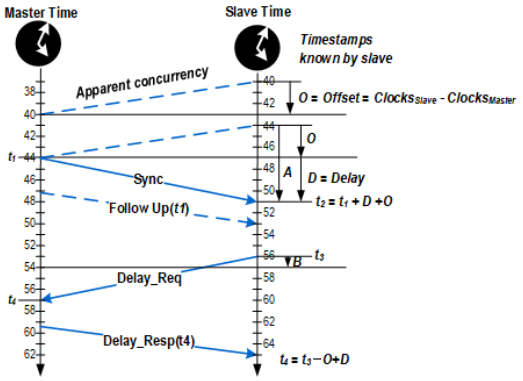


그림 4. PTPv2동기화 과정
Fig. 4. PTPv2 synchronization process

'Delay_Resp' 메시지에 t4를 수납하여 전송한다. 'Delay_Resp' 메시지를 받은 슬레이브는 t1, t2, t3, t4의 시간으로<Mean Path Delay>와 <Offset from Master>를 구한다.

여기서 <Mean Path Delay>는 식 (1)과 같이 마스터 노드와 슬레이브 노드 간 전송시간의 평균값을 의미하고, <Offset from Master>는 식 (2)와 같이 슬레이브와 마스터의 시간차이를 의미한다. 시간오차(Offset)가 0보다 크면 슬레이브가 마스터보다 시간이 빠른 것이고 오프셋이 0보다 작으면 슬레이브가 마스터보다 시간이 느린 것이다.

$$Mean\,Path\,delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \quad (1)$$

$$Offset\,from\,Master = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \quad (2)$$

3.2 타임스탬핑

마스터클럭이 유지하고 있는 시간정보를 수납한 메시지를 슬레이브클럭에게 전달할 때 이 시간정보가 PTP응용계층에서의 취득한 정보라면 하위 계층을 따라 전달되면서 스케줄링 또는 버퍼링 등에 의해 실제 물리계층을 통해 전송되는 시점에 비하여 과거 시점일 뿐만 아니라 가변적인 지연에 의해 그 시간 값 역시 변동되는 문제점이 있다.

만약 네트워크 카드 드라이버 레벨에서 타임스탬핑을 한다면 프로토콜 스택을 경유하는 과정에서의 가변적인 지연시간은 해결되지만 카드내부 FIFO에서의 버퍼링 등에 의해 이것 역시 부정확하다.

특히 수신관점에서는 실제 물리적으로 수신된 시점이 드라이버 레벨 또는 응용계층 레벨에서의 시간 값과 큰 오차를 초래하게 된다. 따라서 PTP에서는 이더넷 PHY와 MAC간의 인터페이스인 MII에서의 실제 송신된 시점과 수신된 시점을 물리적으로 스탬핑하는 방법을 권고한다[9].

- 소프트웨어 타임스탬핑: 소프트웨어 타임스탬핑은 <그림 5>의 (a)와 같이 PTP 계층에서의 시간을 기준으로 시간동기 절차를 수행하는 방식이다. 응용계층에 위치한 PTP가 획득한 현재 시간 값을 수납한 패킷을 물리계층을 통해 송신한다. 수신된 프레임이 최종적으로 PTP계층까지 전달되는 시점에서야 PTP가 이 메시지의 수신시점을 인식 할 수 있으므로 실제 수신개시 시점과는 큰 오차를 초래 하게 된다. 정밀도가 20μs 정도로 낮은편이다.
- 하드웨어 타임스탬핑: 하드웨어 타임스탬핑은 <그림 5>의 (b)와 같이 이더넷 PHY와 MAC간의 인터페이스인 MII에서의 실제 송신된 시점과 수신된 시점을 별도의 타임스탬퍼로 추출하는 방식이다. 이더넷의 경우, 프리앰블의 마지막, 즉 SFD(Start Frame Delimiter) 마지막 비트의 송신시점 또는 수신시점이 바로 스탬핑 기준점이 된다.

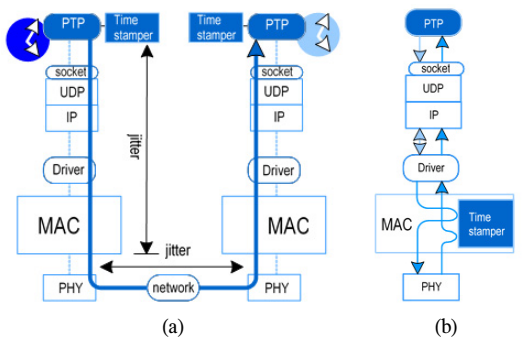


그림 5. (a) 소프트웨어 타임스탬핑 (b) 하드웨어 타임스탬핑
Fig. 5. (a) Software timestamping (b) Hardware timestamping

3.3 시간동기기술의 제약사항

정밀한 시간동기가 제공되려면 당연히 시스템 내부 클럭의 정밀도가 높아야 한다. 이것은 보통 1초 간격으로 수신되는 PTP 프레임에 수납된 시간정보로부터 다음 프레임이 수신될 때까지는 자신의 PTP 클럭 기반에서 10^{-9}초 단위 시간을 생성하기 때문이다.

이 PTP 클럭의 정밀도가 낮거나 안정성이 낮은 경우, PHY에서 SFD를 감지한 시점에서 수신용 PLL Clock Recovery부에서의 클럭이 변동되면서 이후 FIFO, SERDES, CODEC 등에서의 실제 처리시점이

변동될 수 있다. 또한 시스템 클럭율에 따라 clock quantization 오차도 있다. 예를 들어 125MHz 클럭을 사용한다면 클럭주기마다 +/- 4ns단위의 정밀도(Granulity)를 제공할 수 있지만 25MHz 클럭을 사용시에는 +/-20ns 단위의 정밀도만 제공할 수 있다. 그리고 직/병렬화과정, 내부 처리시간 등에 의한 내부지연도 있다.

3.4 시간동기 오차 보정 방법

시간오차는 크게 clock frequency오차, 즉 drift에 의한 미세오차와 초 단위의 offset오차가 있다. <그림 5>는 이러한 시간오차를 보정하는 과정을 도시한 것이다.

슬레이브의 초기화시, 자신은 마스터에 비하여 큰 시간오차가 있다. 마스터와 슬레이브간 시간동기 절차가 개시되면서 마스터클럭과의 시간을 일치시키는 offset 보정절차가 수행된다. 하지만, 이후 마스터와 상이한 슬레이브의 PTP 클럭에 의한 미세한 drift가 발생한다. 참고로 <그림 6> 의 예는 슬레이브의 클럭 주파수가 마스터에 비해 빠른 경우이다. 슬레이브는 자신의 PTP 클럭 주파수 증감을 통해 PLL을 제어하거나 system tick 카운터를 조정함으로써 시간오차 증가율을 감소시킨다[10].

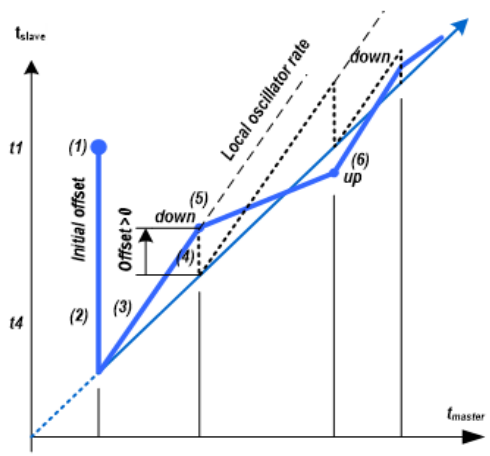


그림 6. 시간 오차에 대한 보정
Fig. 6. Compensation for time error

3.5 Clock Servo

Clock Servo는 <그림 7>과 같이 전파지연시간(Delay) 및 offset과 one-way delay의 계산, offset 및 LPF, servo출력을 조정하는 PI controller 로 구성되어 있다.

동작은 Sync interval 입력 시마다, PTPd의 동작에

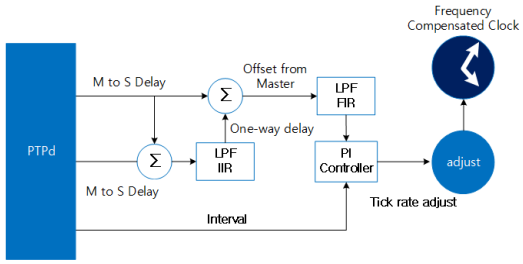


그림 7. Clock Servo 시스템
Fig. 7. Clock Servo System

따라 주기적으로 master-to-slave delay값과 slave-to-master delay를 얻는다. 이 값을 평균하여 one-way delay를 얻는다. 이 값은 수시로 변동하므로 LPF IIR을 거치도록 한다.

얻어진 one-way delay값을 mastet-to-slave delay값에서 빼면 마스터와의 시간오차(offset)이 추출되므로 이것을 LPF FIR을 거치도록 한다. 여기서의 LPF FIR은 EWMA 평균값을 구하는 일종의 smoothing 기능을 수행한다. 이것의 출력, 즉 마스터와의 시간차 정보를 {P, I}값으로 동작하는 PI Controller에 입력시켜 시스템 클럭의 fractional tick-rate를 조정하도록 한다. 이 과정을 클럭 discipline이라고 한다. 즉 현재 tracking되고 있는 시간오차(offset)값을 반영하여 오실레이터의 주파수

또는 소프트웨어 클럭의 파라미터를 수정한다[11].

마스터에서 슬레이브로의 지연시간 M2S Delay 와 슬레이브에서 마스터로의 지연시간 S2M Delay 의 평균값은 곧 두 노드사이의 Path Delay 를 지칭한다. 슬레이브는 상기한 Path Delay 를 참조하여 두 노드 사이의 시간오차를 보정하기 때문에, 실제 Clock Drift 는 Offset = M2S Delay - Path Delay를 만족한다. 통상적으로 임베디드 된 PTP 하드웨어 모듈은 Idle Low Pass Filter 를 이용하여 Impulse Delay 와 같은 고주파 성분을 제거한 Path Delay 를 출력하고 이를 바탕으로 시간오차(Offset)을 산출한다. 산출된 Offset(초 단위 미만의 Clock Drift)은 Low Pass Filter와 PI 시스템을 차례로 거쳐 주파수 보정 모듈의 Addend 레지스터 값을 변동시키며 미세오차를 조정한다[12].

3.6 주파수 보정 모듈

기존의 주파수 보정 모듈은 위의 <그림 1>과 같이 2종류의 레지스터(Addend 및 Increment 레지스터)와 누산기 그리고 가산기로 구성되어 있다. Clock Servo를 통해 전달되는 시간정보 간격 동안에 초단위의 신

호(Pulse per second, PPS)를 생성하기 위하여 내부에 일종의 반가산기인 31비트 크기의 Subsecond 레지스터와 이것의 자리올림(Carry)신호를 사용한다. 이를 위하여 기존 방법에서는 Subsecond 레지스터의 캐리가 발생될 수 있도록 Increment 레지스터 값을 트리거 신호가 생성되는 만큼 증가시키는 방법으로 구현된다.

이러한 트리거 신호 발생간격은 시스템 클럭(SysClk)에 독립적이어야 하므로, 전단부에 시스템 클럭(SysClk)마다 Addend 레지스터 값을 누산하면서 이때 발생하는 자리올림 신호(즉 트리거신호)를 발생시킬 수 있는 32비트 길이의 누산기와 가산기를 사용한다.

예를 들면 tick이 20ns이고, 시스템 클럭(SysClk)이 50Mhz일 경우 Increment 레지스터 값은 식 (3)처럼 tick 값을 대입해 Subsecond 레지스터의 크기인 2^{31}와 tick의 단위인 10^{-9}을 곱하여 계산하고 Addend 레지스터 값은 식 (4)와 같이 Accumulator 1과 Accumulator2의 크기를 더한 2^{63}을 계산한 Increment 레지스터와 SysClk 값을 곱한 값과 나누어 {43, 0xFFB34C02} 값이 나오게 된다.

$$Increment = \frac{tick \cdot 2^{31}}{10^9} = \frac{20 \cdot 2^{31}}{10^9} = 42.94967296 \cong 43 \quad (3)$$

$$addend = \frac{2^{63(32+31)}}{SysClk \cdot Increment} = \frac{2^{63}}{50M \cdot 43} = 4289940482.26 \cong 4289940482 \quad (4)$$

표 1과 같이 tick 값에 따라 계산한 레지스터를 사용해야 한다.

표 1. 기존시스템에서 시스템 클럭이 50Mhz인 경우 레지스터 값
Table 1. In the existing frequency module, when the system clock is 50Mhz, the register value

Tick	Increment	Addend
100 ns	215	0x3323DC00
50 ns	107	0x66C21295
40 ns	86	0x7FD9A601
20 ns	43	0xFFB34C02

Ⅳ. 새로운 주파수 보정 모듈의 구조 및 성능분석

4.1 기존 주파수 보정 모듈의 문제점

기존 주파수 보정 모듈은 2n 크기인 n-bit 레지스터 특성상 시스템 클럭(오실레이터 주파수)과 Tick 값에 의해 계산된 Addend 및 Increment의 실제 값이 식 (3)과 식 (4)처럼 소수가 되며 레지스터에는 반올림하여 정수로 들어간다. 그로 인해 Addend 및 Increment 레지스터 값들은 소수점 이하의 손실이 생긴다.

또한, 이 레지스터 값들이 가산기를 통해 누산기에 누적되어 자리올림 비트 값이 '1'이 될 때 자리올림 비트 값을 제외한 나머지 누산기 레지스터 값이 다음 자리올림이 발생하는 시간에 지속적으로 영향을 미치게 된다. 이로 인해 비주기적인 자리올림이 발생하고 결국 Second와 Nanosecond를 생성하는 주파수 보정 모듈에 시간 오차가 발생하게 된다.

또한 자리올림 신호가 발생하였을 때 사용자는 이 값을 10^{-9} 초 단위로 표현하기 위하여 Subsecond 레지스터에 있는 값을 2진 카운터의 최대 자릿수로 나누는 데 Increment 레지스터 값이 위의 식처럼 소수 값을 반올림하여 정수로 사용하기 때문에 Subsecond 레지스터에 남은 값이 지속적으로 바뀌게 되고 클럭 주파수 보정을 위하여 Addend 레지스터 값을 변경함에도 불구하고 시간오차(offset)가 발생하게 된다.

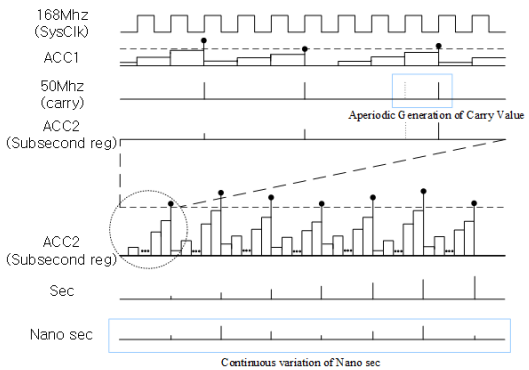


그림 8. 기존의 주파수 보정 모듈의 문제점
Fig. 8. Problem of existing frequency correction module

4.2 새로운 주파수 보정 모듈

본 논문에서 제안하는 주파수 보정 모듈은 위와 같은 문제점을 해결하기 위한 것으로 주파수 보정 모듈 내부의 구조를 변경하여 n-bit 크기의 누산기와 가산기로 구성된 기존 주파수 보정 모듈을 개선하였다. 비교기, 셀렉터 그리고 감산기를 추가하여 누산기를 정

확한 시간에 자리올림 하도록 변경하였다. 또한 Increment 및 Addend 레지스터 계산식이 바뀌어 소수점 버림이 발생하지 않기 때문에 정확한 계산이 된다. 이러한 방식을 제안하여 Subsecond 레지스터 값을 10^{-9}초 단위로 변환 시 오차를 제거함으로써 시간 동기 시스템에서 마스터와 슬레이브 간의 시간오차(offset)가 발생하지 않도록 한다.

제안된 주파수 보정 모듈은 <그림 9>와 같이 첫번째 32비트 폭의 Accumulator1이 시스템 클럭마다 32비트 크기의 Addend 레지스터의 값을 자신의 값에 더한다. Accumulator1과 Register1의 값을 32-bit 비교기를 이용하여 비교하여 Accumulator1의 값이 레지스터의 값보다 크거나 같으면 Output 값을 1을 출력하고 작으면 0을 출력하여 셀렉터로 전달한다. 감산기에서는 Accumulator1값에서 Register1값을 뺀 후 셀렉터로 전달하고 셀렉터에서는 비교기에서 전달 된 Select값을 토대로 Select값이 1일 때는 감산기에서 전달된 값을 가산기에 전달하고 0일 때는 Accumulator1값을 선택하여 Addend 레지스터 값과 더한다. 이 과정에서 32-bit 비교기에서 Output 값이 1이 출력되는 시점마다 다음 Subsecond 레지스터에 트리거 신호를 생성하여 전달하게 된다.

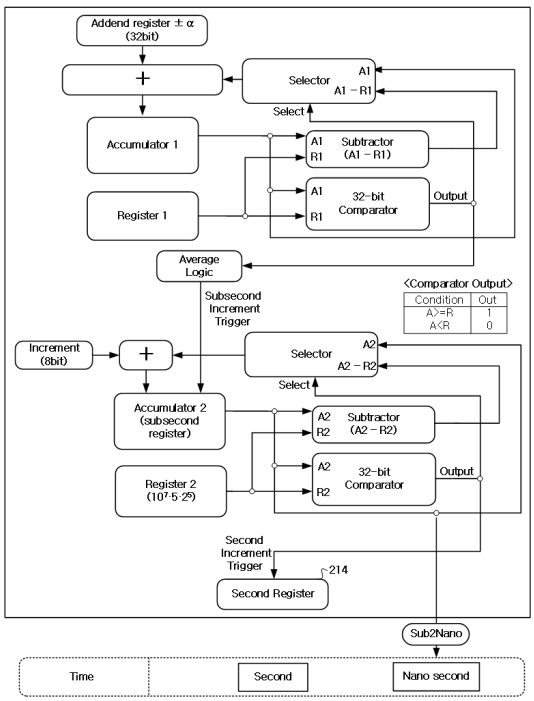


그림 9. 제안된 주파수 보정 모듈의 구조
Fig. 9. Structure of the proposed frequency correction module

두번째 Subsecond 레지스터는 첫 번째 Accumulator1으로부터의 트리거 신호에 맞추어 8비트 길이의 Increment 레지스터의 값을 자신의 값에 더한다. Subsecond 레지스터와 Register2의 값을 32-bit 비교기를 이용해 비교하여 Subsecond 레지스터의 값이 Register2의 값보다 크거나 같으면 Output 값을 1을 출력하고 작으면 0을 출력하여 셀렉터로 전달한다. 감산기에서는 Subsecond 레지스터 값에서 Register2 값을 뺀 후 셀렉터로 전달하고 셀렉터에서는 비교기에서 전달 된 Select 값을 토대로 Select 값이 1일 때는 감산기에서 전달된 값을 그대로 가산기에 전달하고 0일 때는 Subsecond 레지스터 값을 선택하여 Increment 레지스터 값과 더한다. 이 과정에서 32-bit 비교기에서 Output 값이 1이 출력될 때 트리거 신호를 생성한다. 이 트리거 신호가 바로 PTP 시간의 1초이다. 그리고 Subsecond 레지스터의 남은 값은 10^{-9}초 단위가 된다. 이 때 Subsecond 레지스터에 남은 값을 없게 함으로써 오차가 제거되고 마스터와 슬레이브 간의 시간오차(Offset)가 발생하지 않도록 한다.

주파수 보정 모듈의 구조를 변경하였기 때문에 레지스터들을 구하는 식도 변경되었다. 기존 식 (3), (4)는 식 (5), (6)으로 변경되었다.

식 (5)에서 Increment 레지스터는 Subsecond 레지스터에서 주기적으로 증가하는 10^{-9}초 단위의 증분 값을 의미한다. Register2 값인 $10^7 \cdot 5 \cdot 2^5$는 Register2의 크기이며 Increment 레지스터의 값이 정수가 나오게 한다. 이 값은 사용자가 정의하는 값이며 기존 주파수 보정 모듈의 Subsecond 레지스터 크기인 2^{31}과 비슷한 값으로 정하였다. 제약 사항은 Increment 레지스터가 정수가 나오게 하기 위한 tick의 값은 최소 5 이상이어야 한다.

$$Increment = \frac{tick \bullet (10^7 \bullet 5 \bullet 2^5)}{10^9} \tag{5}$$

주파수 보정 모듈을 운용하기 위해 사용되는 시스템 클럭에 대해 초기 Addend 레지스터를 설정하기 위한 연산식은 식 (6)과 같으며 기존 식에서 Accumulator1의 크기인 2^{32} 대신 Register1값을 시스템 클럭에 따라 사용자가 정의함으로써 Addend 레지스터의 값이 정수가 나오게 한다.

$$addend = \frac{(10^7 \bullet 5 \bullet 2^5) \bullet Register1}{SysClk \bullet Increment} \tag{6}$$

시스템 클럭(SysClk) = 50Mhz인 경우, Register1 값을 2^{31}을 사용함으로써 Addend 레지스터의 값이 정수가 나오게 한다.

따라서 Tick = 20ns로 설정할 경우(즉 20ns 정밀도를 갖도록 하려면), Increment 레지스터 값은 식 (7)처럼 '32'가 나오고 Addend 레지스터는 식 (8)과 같이 Increment 레지스터와 SysClk 값을 대입해 '0x80000000(2147483648)'이 나오게 된다. 표 1과 비교하였을 때 Addend 레지스터 값은 더 작지만 정수 값이 나오게 된다. 이로 인해 식 (3)과 (4)에서 소수점으로 인해 생기는 문제를 해결할 수 있다.

$$Increment = \frac{20 \cdot (10^7 \cdot 5 \cdot 2^5)}{10^9} = 32 \qquad (7)$$

$$Addend = \frac{(10^7 \cdot 5 \cdot 2^5) \cdot 2^{31}}{50M \cdot 32} = 2147483648 \qquad (8)$$

표 2와 같이 tick 값에 따라 계산한 레지스터를 사용해야 한다.

제안된 주파수 보정 모듈은 <그림 10>과 같이 주기적인 캐리 신호를 발생시킨다. 그리고 Second 레지

표 2. 제안된 모듈에서 시스템클럭이 50Mhz인 경우 레지스터 값
Table 2. In the proposed frequency module, when the system clock is 50Mhz, the register value

Tick	Increment	Addend
160 ns	256	0x10000000
50 ns	128	0x20000000
40 ns	64	0x40000000
20 ns	32	0x80000000

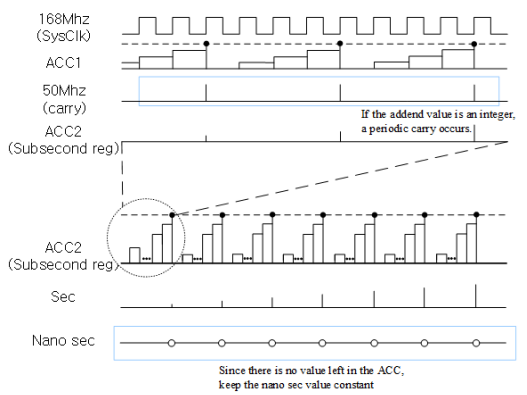


그림 10. 제안된 주파수 보정 모듈의 상태
Fig. 10. The state of the proposed frequency correction module

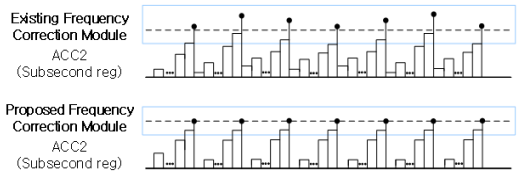


그림 11. 기존 및 제안된 주파수 보정 모듈의 Subsecond레지스터 비교
Fig. 11. Compare with Subsecond register of existing and proposed frequency correction module

스터에서 캐리가 발생할 때마다 Subsecond 레지스터의 값는 0이 된다.

<그림 11>은 기존 주파수 보정 모듈과 제안된 주파수 보정 모듈의 Subsecond 레지스터를 비교하였다. 기존 주파수 보정 모듈은 자리올림이 발생할 때 누산기에 자리올림을 하고 남은 값이 자리올림을 할 때 지속적으로 영향을 미치지만 제안된 주파수 보정 모듈은 자리올림이 발생할 때 정확한 시간에 자리올림하고 누산기에 남은 값이 없어 주기적으로 자리올림을 하여 시간오차가 발생하지 않는다.

4.3 결과 및 성능 분석

시뮬레이션을 위해 설계한 주파수 보정 모듈은 기존 모듈과 제안된 모듈을 비교하기 위하여 1장의 그림 1과 4장 3절의 그림 9와 동일하게 하였다. 외부에 지연이 없다고 가정하였고 마스터가 일정한 시간 값을 슬레이브에 전달하였을 때 슬레이브 기준으로 슬레이브의 시간 값을 확인 한 경우와 시스템 클럭이 50Mhz일 때 마스터를 기준으로 슬레이브의 시간 값을 확인하는 방식으로 진행하였다. 거기에 추가로 4장 4절의 알고리즘을 추가하여 레지스터를 구할 때 적용된 시스템 클럭과 실제 시스템 클럭이 다를 때도 오차를 제거 할 수 있는지 시뮬레이션을 통해 확인해 보았다.

<그림 12>과 <그림 13>은 마스터와 슬레이브의 내부 클럭이 이상적인 경우(즉, 마스터와 슬레이브가 1초마다 외부 지연 없이 시간동기 하였을 때)의 마스터 시간과 슬레이브의 시간을 도시화하였다.

<그림 12>는 슬레이브가 마스터와의 시간오차에 대한 보정은 하고 있으나 지속적으로 시간이 흔들리는 반면 <그림 13>의 슬레이브는 시간오차 없이 마스터와 시간이 동일하다는 것을 확인하였다.

<표 3>은 기존 모듈과 제안된 모듈의 OFM(Offset From Master)에 대한 평균오차와 표준편차를 표시하였다. 기존 모듈에 비해 제안된 모듈은 외부의 지연이 없는 이상적인 경우이기 때문에 평균오차와 표준편차

가 0이 나왔고 실제로 구현하였을 때 제안된 모듈보다 값이 적을 것으로 기대된다.

<그림 14>는 기존 및 제안된 주파수 보정 모듈에서 마스터의 시간을 기준으로 시간오차를 확인하기 위하여 마스터와 슬레이브의 시스템 클럭을 50Mhz이라 가정하고 슬레이브의 레지스터 값을 설정하였으며

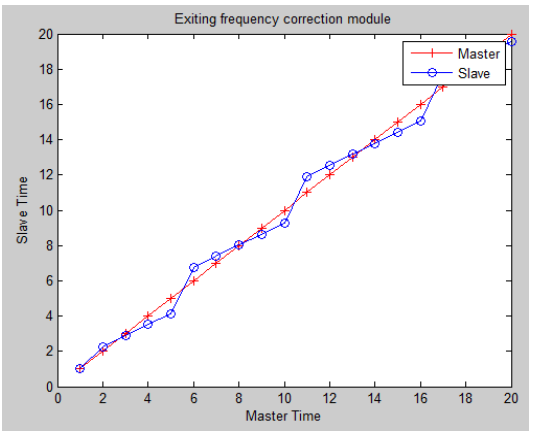


그림 12. 기존의 주파수 보정 모듈이 적용된 마스터와 슬레이브의 시간
Fig. 12. Time of master and slave with existing frequency correction module applied

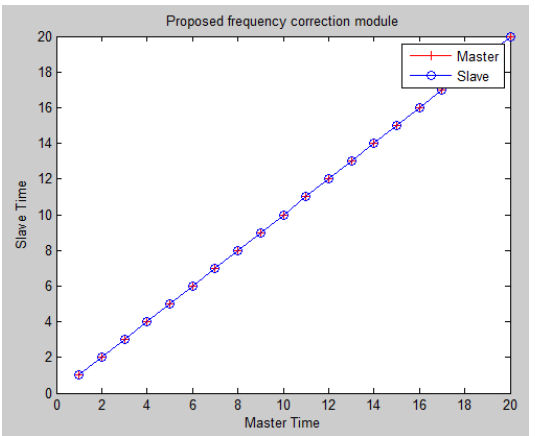


Fig. 13. 제안된 주파수 보정 모듈이 적용된 마스터와 슬레이브의 시간
Fig. 13. Time of master and slave applied with proposed frequency correction module

표 3. 기존 모듈과 제안된 모듈의 OFM에 대한 평균오차와 표준편차
Table 3. Average error and standard deviation for the OFM of the existing module and the proposed module

	평균오차	표준편차
기존 모듈	0.121408	0.542955
제안된 모듈	0	0

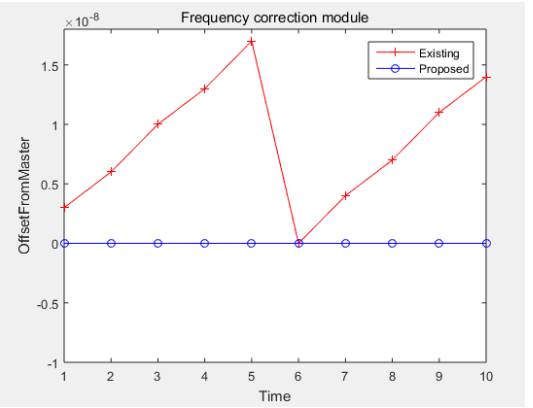


그림 14. 기존 및 제안된 주파수 보정 모듈이 적용된 마스터와 슬레이브간 시간오차
Fig. 14. Time error of master and slave with existing and proposed frequency correction module applied

5000만번 Addend 레지스터 값을 더할 때마다 슬레이브의 시간을 스탬핑하여 마스터 시간과 슬레이브의 시간을 도시화하였다.

<그림 14>에서 기존 주파수 보정 모듈은 시간오차에 대한 보정을 하고 있지 않기 때문에 마스터의 시간(0)을 기준으로 지속적으로 감소하고 있는 반면 제안된 주파수 보정 모듈은 마스터의 시간(0)과 동일하다는 것을 확인하였다.

명시한 결과로부터 본 논문에서 제안하는 자리올림 오류가 없는 주파수 보정 모듈은 기존 주파수 보정 모듈과 비교하여 시간오차가 개선되었음을 알 수 있다.

V. 결 론

최근 차량, 항공, 우주, 군사 산업 등 여러 분야에 디지털화가 진행되고 실시간 영상 및 음성 처리 시스템이 탑재되어 데이터량이 증가함에 따라, 하드웨어의 고성능, 고신뢰성이 보장되어야 하고 분산된 장비들에 대한 정밀한 시간 기준이 제공되어야 한다. 그러나 기존의 시간동기 시스템에서 마스터 클럭 기준, 슬레이브의 클럭 주파수의 변동으로 인한 시간오차 보정과정에서 마스터로부터 보통 1초마다 주기적으로 전달되는 시간 값을 기준으로 지속적으로 시간을 보정함에도 불구하고 내부 모듈의 구조로 인하여 시간오차가 발생하는 문제점이 있다.

이에 본 논문에서는 발생하는 시간오차를 줄여 PTP 모듈의 동기화 성능을 향상시키기 위하여 기존의 주파수 보정 모듈의 구조를 변경한 자리올림 오류가 없는 새로운 주파수 보정 모듈을 제안하였다.

본 논문에서 제안된 자리올림 오류가 없는 주파수 보정 모듈은 고정밀 시간동기 기능을 제공할 수 있으므로 컨베이어 시스템을 이용하는 공장자동화, GPS 수신이 불가능한 음영 지역에 필요한 망동기 신호, 정확한 계약 순서가 필요한 금융시스템, 군사시스템, 차량 멀티미디어 시스템 등과 같은 실시간 전송 기능이 필수적인 시스템에 활용 될 수 있을 것이다.

References

[1] Y. Kwon, J. Eom, J. Ahn, and S. Kim, "Timing and synchronization method for devices using a simple procedure in Bridged Local Area Network" in *Proc. Symp. KICS*, pp. 208-211, Nov. 2011.

[2] J. Hwang, D. Wi, and J. Lee, "RANSAC-based time synchronization over wireless networks," *J. KIISE*, vol. 45, no. 7, pp. 626-634, Jul. 2018.

[3] Y.-S. Han and Y.-T. Choi, "Technical trend of time synchronization equipment in naro space center," *Current Ind. and Technol. Trends in Aerospace*, vol. 6, no. 1, pp. 114-121, Jul. 2008.

[4] D. K. LAM, K. Yamaguchi, Y. Nagao, M. Kurosaki, and H. Ochi, "An improved precision time protocol for industrial WLAN communication systems," in *2016 IEEE Int. Conf. Ind. Technol.*, pp. 824-829, Mar. 2016.

[5] M. Lévesque and D. Tipper, "Improving the PTP synchronization accuracy under asymmetric delay conditions," in *2015 IEEE ISPCS*, pp. 88-93, Oct. 2015.

[6] K. Marzullo and S. Owicki, "Maintaining the time in a distributed system," in *Proc. Second Annu. ACM Symp. Principles of Distrib. Comput.*, ACM, 1983.

[7] IEEE Std 1588-2008 – *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IM/ST - TC9 – Sensor Committee Technology, Jul. 2008.

[8] J. C. Eidson, *Measurement, Control, and Communication Using IEEE 1588* (Advances in Industrial Control). London, U.K.: Springer, 2006.

[9] H. Weibel and D. Béchaz, "IEEE 1588 implementation and performance of time stamping techniques," in *Proc. 2004 Conf. IEEE 1588*, 2004.

[10] L. Benetazzo, C. Narduzzi, and M. Stellini, "Analysis of clock tracking performances for a software-only IEEE 1588 implementation," *2007 IEEE Instrumentation & Measurement Technol. Conf. IMTC 2007*, pp. 1-6, May 2007.

[11] S. Balasubramanian, K. Harris, and A. Moldovansky, "A frequency compensated clock for precision synchronization ing IEEE 1588 protocol and its application to Ethernet," in *Proc. IEEE Wksp.*, pp. 91-94, May 2003.

[12] R. Exel and F. Ring, "Improved clock synchronization accuracy through optimized servo parametrization," in *Proc. Int. IEEE Symp. Precision Clock Synchronization Meas. Control Commun. (ISPCS)*, pp. 65-70, Sep. 2013.

이 정 도 (Jeong-do Lee)

2016년 2월 : 한국항공대학교 항공정보통신공학과 학사 졸업
2018년 2월 : 한국항공대학교 항공전자정보공학과 석사 졸업
2018년 1월~현재 : 전자부품연구원 임베디드SW연구센터 연구원

<관심분야> 시간동기, TSN(Time Senstive Network), 임베디드 소프트웨어, TCN (Train Communication Network)

박 부 식 (Pu-sik Park)

1999년 2월 : 한국항공대학교 전자 공학 학사 졸업
2001년 8월 : 한국항공대학교 공학석사 졸업
2002년 1월~현재 : 전자부품연구원 임베디드SW연구센터 책임연구원

<관심분야> TSN (Time Sensitive Network), AFDX (Avionics Full-Duplex Switched Ethernet), AeroRing, TCN (Train Communication Network), Seamless Redundancy, 항공IT융합, 철도IT융합 등

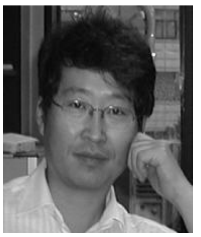
[ORCID:0000-0002-4308-5708]

손 명 환 (Myeong-hwan Son)

2017년 2월 : 한국항공대학교 항공전자 및 전자공학과 학사 졸업
2019년 2월 : 한국항공대학교 항공전자 및 전자공학과 석사 졸업
2019년 2월~현재 : 전자부품연구원 임베디드SW연구센터 연구원

<관심분야> TSN (Time Sensitive Network), 항공 및 차량 네트워크

윤 종 호 (Chong-ho Yoon)



1984년 : 한양대학교 전자공학과 학사 졸업
1986년 : 한국과학기술원 전기 및 전자공학과 석사 졸업
1990년 : 한국과학기술원 전기 및 전자공학과 박사 졸업

<관심분야> 차량용 이더넷, TSN(Time Sensitive Network), 시간동기