논문 2017-12-27

CDR을 사용한 FPGA 기반 분산 임베디드 시스템의 클록 동기화 구현

(An Implementation of Clock Synchronization in FPGA Based Distributed Embedded Systems Using CDR)

송 재 민, 정 용 배, 박 영 석*
(Jae-Min Song, Yong-Bae Jung, Young-Seak Park)

Abstract : Time synchronization between distributed embedded systems in the Real Time Locating System (RTLS) based on Time Difference of Arrival (TDOA) is one of the most important factors to consider in system design. Clock jitter error between each system causes many difficulties in maintaining such a time synchronization. In this paper, we implemented a system to synchronize clocks between FPGA based distributed embedded systems using the recovery clock of CDR (clock data recovery) used in high speed serial communication to solve the clock jitter error problem. It is experimentally confirmed that the cumulative time error that occurs when the synchronization is not performed through the synchronization logic using the CDR recovery clock can be completely eliminated.

Keywords : Clock synchronization, High-speed serial communication, FPGA (Field Programmable Gate Array), Distributed embedded system, RTLS (Real-Time Locating System), CDR (Clock Data Recovery)

Ⅰ. 서 론

TDOA (time difference of arrival) 기반 RTLS (real-time locating system)에서 분산 시스템 간에 시각 (Time) 동기화는 시스템 설계에서 고려해야 할 매우 중요한 요소 중에 하나이다. 최근 네트워크 상에 적용 가능한 시각 동기화 기법에 대한 연구가 많이 진행되고 있고, 대표적인 시각 동기화 기법으로는 RBS (refernece broadcast synchronization) [1], TPSN(timing-sync protocol for sensor network) [2], FTSP (flooding time synchronization protocol) [3] 등이 있다.

시각 동기화에서 위와 같은 소프트웨어적인 동기화 알고리즘뿐만 아니라, 하드웨어 상의 클록 발진기 (Clock Oscillator) 동기가 시각 동기를 유지하는데 매우 중요한 역할을 수행한다. 각각의 클록 발진기는 서로 다른 지터 (jitter) 오차가 존재한다. 이러한 지터 오차는 소프트웨어적으로 시각 동기화를 수행했음에도 시각 오차가 계속 누적되며 이러한 시각 오차가 결국 측정하고자 하는 타겟 (target)의 위치 오차를 증가시키는 문제를 초래한다.

실제적인 사례를 들면 UWB기반의 Decawave사의 DW1000을 사용하여 TDOA 기반 실시간 위치 추적 시스템을 구현할 경우 각각의 분산 시스템은 DW1000에 38.4Mhz의 클록을 제공해야 한다. DW1000은 분산 시스템에서 제공받은 38.4Mhz를 내부 PLL를 사용하여 65GHz의 속도로 타임스템프를 증가시키고 이를 사용하여 송신 타임스템프와 수신 타임스템프의 오차를 사용하여 거리를 측정하게 된다. 하지만 분산 시스템의 특성상 독립되어 있는 노드간의 DW1000에 제공하는 38.4Mhz의 수정 발진자는 개별적으로 사용해야하고, 앞서 살펴 본 바와 같이 모든 수정 발진자는 서로 다른 지터 오차가 존재하므로 지터 오차로 인해 DW1000의 내부 타임스템프가 소프트웨어적인 동기화 알고리즘을 사용하여 동기화를 수행했음에도 시간에 따라

*Corresponding Author (yspark@kyungnam.ac.kr)
Received: Mar. 16 2017, Revised: May 19 2017, Accepted: July 8 2017.
J.M. Song, Y.B.Jung, Y.S. Park : Kyungnam University
ⓒ IEMEK J. Embed. Sys. Appl. 2017 Aug. 12(4) 239-246
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2017.12.4.239

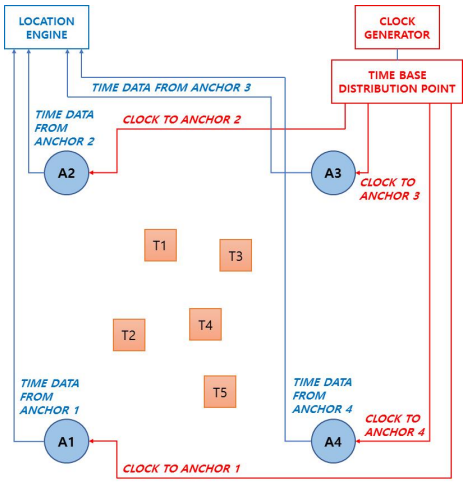


그림 1. Decawave의 클록 동기화
Fig. 1 Clock synchronization for decawave

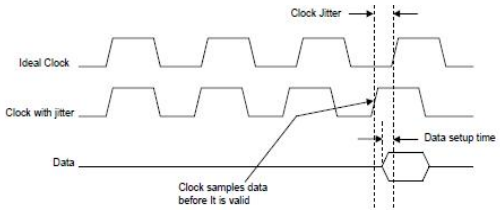


그림 2. 클럭 신호 내 지터
Fig. 2 Jitter of clock signals

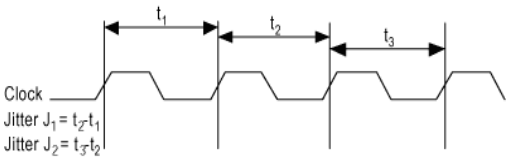


그림 3. 사이클 대 사이클 지터
Fig. 3 Cycle-to-Cycle jitter

오차가 누적되어 측정 거리의 오차가 증가하는 문제가 발생한다. 이러한 문제를 Decawave사는 클록 제공 시스템을 구성하여 각각의 분산 시스템 노드에 CAT5 케이블을 사용하여 클록을 공유하는 방식을 사용한다 [4, 5]. 그림 1은 Decawave 사에서 제공하는 클록 공유 방법을 보여준다.

하지만 Decawave 사에서 제시하는 클록 공유 방식은 클록 제공을 위해 새로운 시스템을 구성해야 하는 문제뿐만 아니라 분산 시스템의 모든 노드에 클록을 제공하기 위해 물리적인 비용 증가와 시스템의 구조가 복잡해지는 문제가 발생하게 된다.

따라서 본 논문에서는 이러한 하드웨어 상의 클록 발진기의 지터 오차로 인한 시각 오차 누적 문제를 해결하기 위해 고속의 데이터 송수신에 사용하는 고속 직렬 통신에 사용되는 CDR (clock data recovery)을 사용하여 주 (master) 시스템과 다수의 종 (slaver) 시스템 간에 클록 발진기의 신호를 동기화하는 방법에 대해 기술하고, FPGA 기반 분산 임베디드 시스템 상에서 구현하여 그 결과를 확인하고자 한다.

본 논문의 구성은 다음과 같다. 2절에서는 실험에 필요한 이론에 대한 사전 지식들을 기술하고, 3절에서는 실험을 수행할 FPGA 플랫폼의 구성에 관하여 설명하고, 4절에서는 CDR을 사용하여 클록을 동기화하는 로직에 대해 기술한다. 5절에서는 구현한 로직을 사용하여 실제 실험을 수행한 결과를 요약 정리한다.

II. 클록 동기화 이론

1. 지터 (Jitter)

지터는 그림 2에서 볼 수 있듯이 “이상적인 위치의 신호에 대해 단기간의 변화”를 말한다. 이상적인 위치에서 출력 전이에 이러한 편차는 데이터 전송 품질에 부정적인 영향을 미칠 수 있고, 대부분의 경우에는 신호의 스큐 (Skew) 및 잡음과 같은 다른 신호들의 편차가 결합되어 지터로 분류된다. 과도한 지터는 데이터 비트 스트림을 잘못 전송함으로써 통신 신호의 비트 오류율 (BER)을 증가시킬 수 있고, 디지털 시스템에서의 지터는 타이밍 오차에 의해 회로가 부적절하게 동작할 수 있다 [6, 7].

주기 지터 (period jitter)는 클록의 출력 전환이 연속적인 클록 에지를 통해서 이상적인 위치에서 변경된 것을 말한다.

사이클 대 사이클 지터 (cycle-to-cycle Jitter)는 한 사이클에서 다음 사이클까지의 클록 주기의 차이를 말한다. 그림 3에서와 같이 J1과 J2는 측정된 지터의 값으로 여러 사이클에 걸쳐 측정된 최대값은 최대 순환 주기 지터이고, 이러한 사이클 대 사이클 지터는 시스템 동작의 오류를 유발할 수 있다 [6].

반주기 지터 (half-period jitter)는 1/2 기간 동안의 이상적인 위치에서 클록 출력 변화의 최대 변화를 측정한 것으로 그림 4는 반주기 지터를 보여준다 [8].

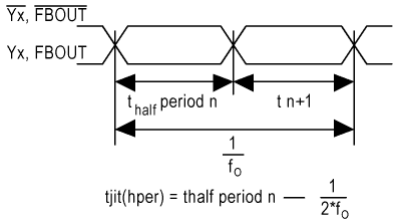


그림 4. 반주기 지터

Fig. 4 Half-period jitter

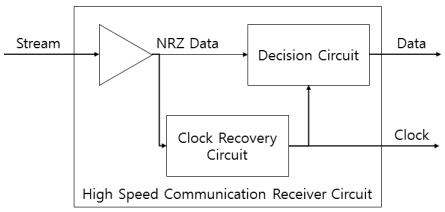


그림 5. CDR 블럭도

Fig. 5 CDR block diagram

2. 클록 데이터 복구 (Clock Data Recovery)

대부분의 고속 직렬 인터페이스에는 수반되는 클록이 없기 때문에 수신기는 데이터를 샘플링하기 위해 송신 측의 클록을 복구해야 한다. 수신기는 대략적인 기준 주파수에서 클록을 생성하고, 클록을 복구하기 위해 수신기는 입력되는 데이터 스트림에 기준 클록의 위상을 재정렬하며 이를 클록 복구(clock recovery)라고 한다.

이렇게 복구된 클록을 사용하여 비트 스트림 상에 들어오는 데이터 신호를 샘플링 하는 것을 데이터 복구라고 하고 이 모두를 클록 데이터 복구(CDR: clock data recovery)라고 한다. 그림 5는 CDR 동작에 대한 블록 다이어그램이다.

3. CDR을 사용한 클록 동기화 기법

분산 임베디드 시스템의 CPU 외부 클록으로는 수정 발진자 (cystal oscillator)를 많이 사용한다. 수정 발진자의 경우, 이상적으로는 해당 주파수의 클록을 발생시키지만 에이징 (aging) 또는 온도, 습도, 공급 전압의 변화 등에 의해 실제적으로는 해당 주파수에서 약간 어긋난 (skew) 주파수의 클록을 발생시킨다. 이러한 클록 스큐로 인해, 수신기 간에 특정 순간 동기가 맞춰져 있다 하더라도 시간이 지남에 따라 두 수신기 클록의 시각오차는 누적된다 [9, 10]

이러한 시각 오차 누적 문제를 해결하기 위해 본 논문에서는 고속 직렬 통신에서 사용되는 CDR

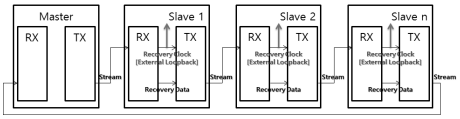


그림 6. CDR을 사용한 클록 동기화

Fig. 6 Clock synchronization using CDR

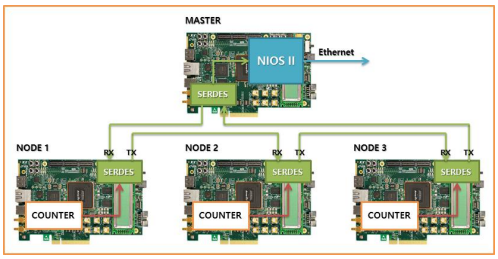


그림 7. 실험 플랫폼 구성

Fig. 7 Configure the experimental platform

의 복원 클록을 사용한다. 클록 복원은 고속 직렬 통신 시 수반되지 않은 클록을 수신측에서 송신측의 클록을 복원 하는 것으로 수신측은 송신측의 클록 신호를 재생성 하게 된다. 이렇게 생성된 송신측의 클록신호를 데이터 송신 클록으로 재사용할 경우 다음 노드에서도 같은 클록을 재생성하게 되고 결과 적으로 분산 시스템의 모든 노드의 클록은 처음 송신을 시작한 클록에 동기화 되게 된다. 그림 6은 주 노드 (master node)의 송신 클록으로 모든 종 노드 (slave node)의 클록을 동기화 하는 것을 도식적으로 나타낸다.

Ⅲ. FPGA 플랫폼 네트워크 동기화 논리

본 논문에서 구현할 FPGA 기반 실험 플랫폼의 구성은 1개의 주노드와 3개의 종노드로 구성되며 이 4개의 노드는 SFP+(Small form-factor pluggable transceiver) 모듈과 광케이블을 사용하여 서로 링 (Ring)형으로 연결되어 있다. 그림 7은 실험 플랫폼의 구성을 보여 준다.

1. 주 (Master) 노드

주 노드는 고속 직렬 통신으로 전달되는 카운터 값을 수집하고 Intel (구 Altera)사의 소프트코어 (Softcore) Nios II Processor [11]를 사용하여 이더넷 소켓 서버 (ethernet socket server)를 통해 네트워크로 데이터를 전송한다. MFC 기반의 소켓 클라이언트 (socket client)는 수신 받은 데이터를

그림 8. 디스플레이 응용프로그램

Fig. 8 Display application

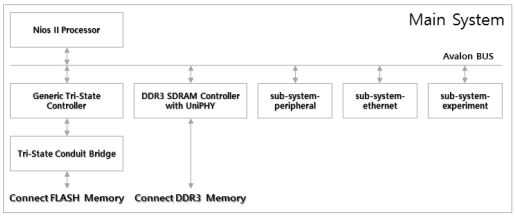


그림 9. 주 시스템 구성

Fig. 9 Main system configuration

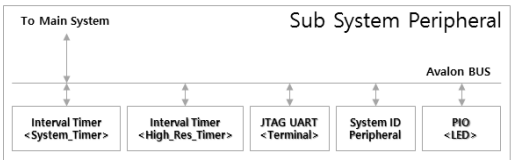


그림 10. 주변장치 부 시스템 구성

Fig. 10 Peripheral sub system configuration

디스플레이 하고 로그 (log)로 데이터를 수집한다. 그림 8은 MFC 기반의 데이터 수집 프로그램의 화면을 보여준다.

Intel FPGA 개발툴인 Quartus II에서는 소프트 코어 프로세서인 Nios II Processor를 제공하고 이는 Qsys Builder를 사용하여 구성할 수 있다. 그림 9는 주 노드의 주 시스템 (main system) 구성을 보여준다.

주 노드는 1개의 주 시스템과 3개의 부 (Sub) 시스템으로 구성되어 있다. 주 시스템은 CPU와 Flash, DDR3 메모리 IP로 구성되고, 부 시스템으로는 주변장치 부 시스템 (peripheral sub system), 이더넷 부시스템 (ethernet sub system), 실험 부 시스템 (experiment sub system)으로 구성된다.

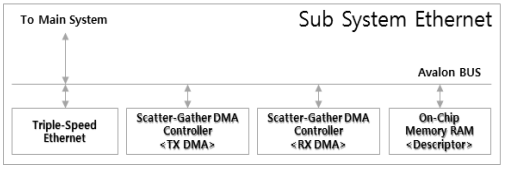


그림 11. 이더넷 부 시스템 구성

Fig. 11 Ethernet sub system configuration

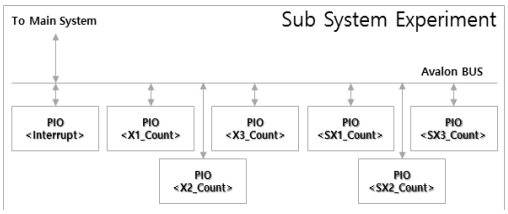


그림 12. 실험 부 시스템 구성

Fig. 12 Experiment sub system configuration

주변장치 부 시스템은 Nios II Processor에 필요한 각종 장치를 구성하는 IP들로 되어 있고, 여기에는 디버그에 필요한 JTAG-UART, TIMER, LED 등이 있다. 그림 10은 주변장치 부 시스템의 구성을 보여준다.

이더넷 부 시스템은 소켓 서버를 사용하기 위한 이더넷 관련 IP로 구성되며, Quartus II 프로그램에서는 3배속 이더넷 (Triple-Speed Ethernet) IP를 사용하여 네트워크를 구성한다. 그림 11은 이더넷 부 시스템 구성이다.

실험 부 시스템은 종 노드에서 전송되는 카운터 데이터를 Nios II 프로세서에서 읽기 위한 PIO로 구성되어 있고, 1Hz의 신호의 인터럽트를 통해 카운터 값을 샘플링 받아 서버를 통해 클라이언트로 데이터를 전송한다. 그림 12는 실험 부 시스템 구성이다.

2. 종 (Slave) 노드

종 노드는 고속 직렬 통신에 필요한 Altgx IP와, 50Mhz 클록을 사용하는 카운터, 그리고 Altgx에서 제공되는 CDR 클록을 사용하는 카운터로 구성된다. 종 노드의 동작은 주 노드에서 전송되는 데이터 요청 패킷에 50Mhz의 카운터 값, CDR의 카운터 값을 패킷 하위에 붙여 다음 노드에 전달한다. 그림 13, 14는 50Mhz 클록의 카운터 동작과 CDR 클록의 카운터 동작을 도식적으로 보여준다.

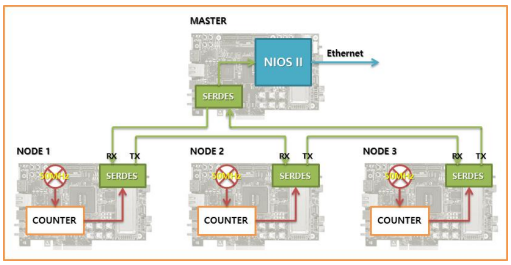


그림 13. 50Mhz 실험 구성

Fig. 13 50Mhz experimental configuration

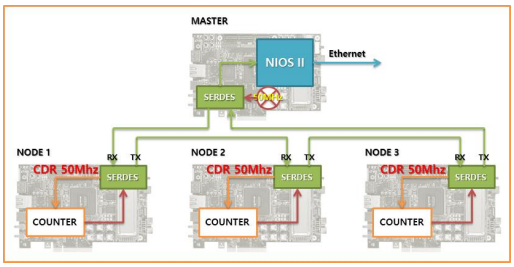


그림 14. CDR 실험 구성

Fig. 14 CDR experimental configuration

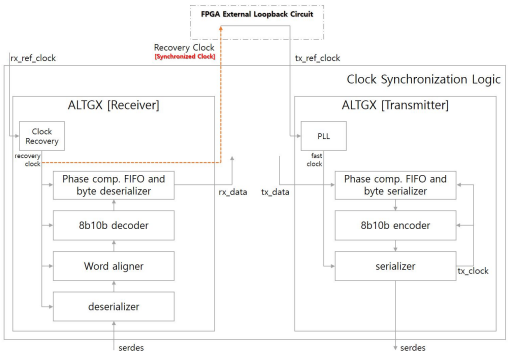


그림 15. CDR 클록 동기화

Fig. 15 CDR clock synchronization

Ⅳ. 동기화 로직 구성

본 논문에서는 이러한 클록 동기화를 데이터 송수신에 사용되는 고속 직렬 통신의 CDR 모듈을 사용하여 클록을 동기화 하는 방법을 제시하였다. 실험에서 사용하는 고속 직렬 통신은 Intel사에서 제공 하는 ALTGX IP [12]를 사용한다. 표 1은 실험을 위해 사용한 ALTGX의 설정을 요약해서 정리한 것이다.

그림 15는 동기화를 도식적으로 표현하고 있다. 분산 시스템의 노드간 CDR 클록을 동기화하기

표 1. 고속 시리얼 통신 IP 설정

Table 1. High-speed serial communication IP setting

ALTGX Transmitter	
Which device variation will you be using?	GX
Which protocol will you be using?	Basic
Which subprotocol will you be using?	None
What is the operation mode?	Transmitter only
What is the number of channels?	1
What is the deserializer block width?	Single
What is the channel width?	16bits
What would you like to base the setting on?	Data Rate
What is the input clock frequency?	50MHz
What is the effective data rate?	1000.0 Mbps
8b10b Decoder/Encoder	Enable
ALTGX Receiver	
Which device variation will you be using?	GX
Which protocol will you be using?	Basic
Which subprotocol will you be using?	None
What is the operation mode?	Receiver only
What is the number of channels?	1
What is the deserializer block width?	Single
What is the channel width?	16bits
What would you like to base the setting on?	Data Rate
What is the input clock frequency?	50MHz
What is the effective data rate?	1000.0 Mbps
8b10b Decoder/Encoder	Enable

위해서는 ALTGX를 전송 (transmitter) 모드와 수신 (receiver) 모드로 구분하여 생성한다. 구분되어진 수신 모드의 CDR 클록을 FPGA 외부 루프백(loopback)을 통해 다시 입력 받아 전송 모드의 클록으로 사용한다. 따라서 처음 전송을 수행한 주 노드의 주 클록이 첫 번째 노드의 수신단의 CDR을

```
altgx_basic_transceiver     altgx_basic_transceiver_inst  (
   .cal_blk_clk              (clock),

   .pll_inclk                (pll_inclk),
   .pll_locked               (pll_locked),
   .tx_clkout                (serdes_clk_tx),
   .tx_digitalreset          (tx_d_reset),
   .tx_ctrlenable            (tx_ctrlenable),
   .tx_datain                (tx_datainQ),
   .tx_dataout               (serdes_tx)
);

altgx_basic_receiver     altgx_basic_receiver_inst  (
   .cal_blk_clk              (clock),

   .rx_cruclk                (pll_cruclk),
   .rx_clkout                (serdes_clk_rx),
   .rx_analogreset           (rx_a_reset),
   .rx_digitalreset          (rx_d_reset),
   .rx_freqlocked            (rx_freqlocked),
   .rx_ctrldetect            (rx_ctrldetect),
   .rx_dataout               (rx_dataoutQ),
   .rx_datain                (serdes_rx)
);
```

그림 16. ALTGX IP 신호

Fig. 16 ALTGX IP signal

통해 동기화되어 복원되고, 이를 사용하여 다시 전송단으로 전송을 수행하기 때문에 모든 노드의 CDR 클록은 주 노드의 주 클록으로 동기화 되어진다. 결과적으로 이 CDR 클록을 사용하여 카운터를 증가시킬 경우 모든 카운터의 오차는 없어지게 된다.

그림 16은 앞서 설정한 Intel사의 ALTGX IP들의 입력 신호들을 보여준다. 여기서 cal_blk_clk은 고속 직렬 통신에서 사용되는 Calibration 클록을 뜻하고, 전송모듈의 pll_inclk는 전송단의 전송 클록을 tx_clkout은 데이터 전송 주기, tx_datain은 전송을 위한 16bit 데이터, tx_dataout는 고속 트랜시버 출력단을 의미한다. 수신모듈의 rx_cruclk는 수신단의 CDR 복원을 위한 기준 클록, rx_clkout은 CDR을 사용하여 복원된 클록, rx_dataout은 수신된 16bit 데이터, rx_datain은 고속 트랜시버 입력단을 의미한다. 따라서 본 논문에 사용하는 동기화를 위해서는 수신단의 rx_clkout을 FPGA 외부로 출력하여 디바이스 회로상의 Loopback을 통해 다시 FPGA로 입력받고 이를 사용하여 송신단의 pll_inclk로 사용하면 현재 노드의 rx_clkout은 이전 노드의 pll_inclk과 동기화 되고, 다음 노드의 rx_clkout은 현재 노드의 pll_inclk과 동기화 되므로, 분산 임베디드 시스템의 모든 노드의 rx_clkout은 동기가 맞춰진 클록이 된다. 이를 사용하여 카운터를 증가 시킬 경우 모든 카운터의 오차는 없어지게 된다.

$$CDR\,Clock\,Rate = \frac{Transfer\,Speed}{Transfer\,Width} \qquad (1)$$

여기서 CDR 클록의 출력 rx_clkout은 인코딩 방식과 ALTGX IP의 설정에 따라 변화하게 된다. 이번 실험에서 ALTGX의 설정은 기본 프로토콜(basic protocol)을 사용하여 단일 모드 (single mode)의 16 비트 폭 (width)으로 1000Mbps 속도로 통신을 수행하고 8b10b 디코더/엔코더(decoder/encoder)를 사용한다. 따라서 복원된 클록은 수식 1에 따라 1000Mbps / 20bit = 50MHz가 된다.

그림 17. 카운터 누적오차 그래프

Fig. 17 Cumulative error graph of counter

표 2. 카운터 누적오차 표
Table 2. Cumulative error table of counter

Time (Hour)	Non Synchronization		Synchronization	
	X1-X2	X1-X3	SX1-SX2	SX1-SX3
0	0	0	0	0
2	3160826	-1159858	0	0
4	6388559	-2276790	0	0
6	9607478	-3391559	0	0
8	12828814	-4500693	0	0
10	16088389	-5597877	0	0
12	19341184	-6689086	0	0
14	22573232	-7786133	0	0
16	25825920	-8878291	0	0
18	29080600	-9967121	0	0
20	32352305	-11060667	0	0
22	35598713	-12164077	0	0
24	38416514	-13137972	0	0
Time Error (Sec)	0.768330 (760ms)	-0.262759 (-260ms)	0	0

V. 실험 및 결과

그림 17은 24시간 동안 분산 임베디드 시스템의 각각의 노드별 개별 클록을 사용한 경우의 클록 오차와, 4장에서 제시한 동기화 이론을 사용하여 클록 동기화를 수행한 경우의 클록 오차를 그래프로 표현한 것으로 각각의 노드별 개별 클록을 사용한 경우 시간의 흐름에 따라 일정하게 증가하는 것을 확인 할 수 있다. 하지만 클록 동기화를 수행한 클록 오차의 경우 시간의 흐름에도 오차가 발생하지 않는 것을 확인 할 수 있다. 그래프의 X축의 경우 시간 (t)를 Y축은 기준 노드 대비 카운터 오차 누적 값을 의미한다. 또한 X1, X2, X3는 분산 시스템 각각의 노드의 개별 클록을 사용하였을 경우 수집된 카운터의 값을 의미하고, SX1, SX2, SX3는 클록 동기화를 수행 한 경우 수집된 카운터의 값을 의미한다.

다음 수식 2는 시간의 흐름에 따른 오차 누적을 수식적으로 표현한 것으로 기준 시각과 노드 시각의 오차를 합산함으로써 오차 누적율을 구할 수 있다.

$$Time\,Error\,Rate = \sum (X_{basistime}(t) - X_{nodetime}(t)) \quad (2)$$

표 2의 경우 2시간 간격으로 결과를 정리 해놓은 것으로 X1-X2와 X1-X3는 X1 노드 대비 X2와 X3의 카운터 누적 오차를 말하고, SX1-SX2와 SX1-SX3는 SX1 노드 대비 SX2와 SX3의 카운터 누적오차를 말한다. 그 결과 분산 임베디드 시스템의 각 노드별 개별 클록을 사용했을 경우, 첫 번째 노드 대비 두 번째 노드는 2시간 간격으로 약 60ms의 오차가 증가하는 것을 확인 할 수 있고, 세 번째 노드는 약 -20ms의 오차가 증가하는 것을 알 수 있다. 결과 적으로 24시간 후 각각의 시간 오차는 약 760ms, -260ms가 된다. 이는 발진기 지터 오차로 인한 차이로 발진기의 지터에 따라 결과는 변화 할 수 있다. 하지만 4장에서 제시한 동기화 이론을 수행한 경우 첫 번째 노드 대비 모든 시간에서 오차가 0이 되는 것을 확인 할 수 있고, 24시간 후 시간 오차도 발생하지 않는 것을 알 수 있다.

References

[1] J. Elson, L. Girod, D. Estrin, “Fine-grained Network Time Synchronization Using Reference Broad-casts,” ACM Operating Systems Review, Vol. 36, No. SI, pp. 147-163, 2002.

[2] S. Ganeriwal, R. Kumar, M.B. Srivastava, “Timing-sync Protocol for Sensor Networks,” Proceedings of the International Conference on Embedded Networked Sensor Systems, pp. 138-149, 2003.

[3] M. Marooti, B. Simon, A. Leedeczi, “The Flooding Time Synchronization Protocol,” Proceedings of the International Conference on Embedded Networked Sensor Systems, pp. 39-49, 2004.

[4] DECAWAVE, “aps007_wired_sync_rtls_with_the_dw1000,” www.decawave.com

[5] DECAWAVE, “aps003_dw1000_rtls_introduction,” www.decawave.com

[6] S.H. Park, “A Study on Phase-Noise and Jitter due to the Power Supply Noise of the CMOS Ring Oscillator,” The Journal of the Korea Institute of Maritime Information & Communication Sciences., Vol. 10, No. 2, pp. 298-302, 2006 (in Korean).

[7] PERICOM, “Jitter Measurement Techniques,” www.pericom.co.uk

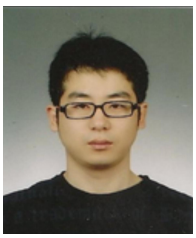
[8] SiTime, “Clock Jitter Definitions and Measurement Methods,” www.sitime.com

[9] J.W. Kim, D.S. Eom, "TDoA-Based Practical Localization Using Precision Time-Synchronization," Journal of Korea Information And Communications Society, Vol. 38 No. 2, 2013 (in Korean).
[10] J.M. Jeon, D.G. Kim, E.R. Kim, D.I. Lee, "Implementing IEEE1588 Based Clock Synchronization for Networked Embedded System," IEMEK J. Embed. Sys. Appl., Vol. 9, No. 1, pp. 33-41, 2014 (in Korean).
[11] ALTERA, "Nios II Gen2 Processor Reference Guide," www.altera.com
[12] ALTERA, "ALTGX Transceiver Setup Guide for Statix IV Devices," www.altera.com

Jae-Min Song (송 재 민)

In 2012 and 2014, respectively B.S. and M.S. degrees in Dept. of Information and Communication Engineering from Kyungnam University.
Email : lesofs306@naver.com

Yong-Bae Jung (정 용 배)



In 2003 and 2006, respectively B.S. and M.S. degrees in Dept. of Information and Communication Engineering from Kyungnam University. In 2010, Ph.D. degree in Advanced Eng. from Kyungnam University.
Email : wjddydqo1@naver.com

Young-Seak Park (박 영 석)

In 1979, B.S. degree in Electronic Engineering from Youngnam University. In 1981 and 1985, respectively M.S. and Ph.D. degrees in Electronic Engineering from Hanyang University. From 1990 to 1991, Invited Scientist of Communications Research Laboratory(or KARC) in Ministry of Posts and Telecommunications (Japan). From 2001 to 2002, Visiting Professor of Det. of Computer Eng. in North Carolaina State Univ. (United State). From 2010 to 2011, Dean of Engineering College in Kyungnam University. From 1985, currently Professor of Dept. of Information and Communication Eng, in Kyungnam University. His Research Interest : Software Engineering, Web-based Software Design & Development, Pattern Recongnition, Image Process, Computer Network & Network Computing, Embedded Processor System HW/SW
Email : yspark@kyungnam.ac.kr