2014 17th Euromicro Conference on Digital System Design

A Time Synchronization Circuit with an Average 4.6 ns One-Hop Skew for Wired Wearable Networks

Fardin Derogarian, João Canas Ferreira, Vítor M. Grade Tavares
INESC TEC, Faculdade de Engenharia, Universidade do Porto
{mpt09020, jcf, vgt}@fe.up.pt

***Abstract*—This paper describes and evaluates a fully digital circuit for one-way master-to-slave highly precise time synchronization in a low-power, wearable system equipped with a set of sensor nodes connected in a mesh network. Sensors are connected to each other with conductive yarns that are used as one-wire bidirectional communication links. The circuit is designed to perform synchronization in the Medium Access Control (MAC) layer. In each sensor node, the synchronization circuit provides a synchronized, programmable clock signal and a real-time counter for time stamping. Experimental results obtained with an implementation in 0.35 µm CMOS technology for a network of electromyography sensors show that the circuit keeps the one-hop average clock skew below 4.6 ns, a value small enough to satisfy many wearable application requirements.**

I. Introduction

Time synchronization is a very important aspect of many sensor networks. In fact, a synchronization method is required in any distributed data acquisition system. Since sensor networks are distributed data acquisition systems with limited resources, the design challenge is to preserve performance while ensuring precise synchronization between the nodes.

A time synchronization method with sufficient accuracy is a necessary part of many wired or wireless sensors applications. Such a method ensures that time keeping at all nodes of the network proceeds with a small, bounded difference to a reference clock. Some applications that depend critically on time synchronization are: environment monitoring, localization, coordinated sleep wake-up scheduling mechanisms, gait analysis for evaluation and diagnosis of mobility impairments, and data fusion [1], [2]. As an example, the application described in [2] requires that inertial and Electromyography (EMG) measurements made by the different nodes be time stamped with an error below 0.5 ms.

Usually each Sensor Node (SN) is equipped with an independent clock generator based on a quartz crystal. The frequency tolerance of this kind of oscillators is in the range of a few part per million (ppm). Because of the independence of these local oscillators, the frequency and phase difference between them results in the absence of synchronization between the nodes over the time of operation. A synchronization method, such as network protocol or dedicated hardware, is necessary to establish synchronization in an acceptable range [3]–[6]. The time synchronization problems has been studied in all areas of networking [1], [3], [7]. The main challenge is to overcome to non-deterministic delay effects on synchronization within the limited resources available. In general, the delay of transferring data between to nodes has four components:

- Send time: time for assembling a message and handing it over to the Medium Access Control (MAC) layer;
- Access time: time for accessing the communication channel;
- Propagation time: delay due to signal propagation between nodes;
- Receive time: time for receiving and processing message at the receiver.

Except for propagation time, the delay sources are usually non-deterministic in general.

Among the many synchronization methods, two-way message exchanges between pairs of nodes are widely used to estimate the time delay and offset between two nodes [8]–[10]. In this approach, delay and offset are calculated by sending and receiving messages with timing information and measuring the round trip delay. A second approach uses only one-way communication [11], [12]. In this case, the node that acts as a time reference sends timing information to client nodes, which must then estimate delay and offset.

Reducing clock skew due to delay and offset depends on the synchronization and estimation method. Eliminating any non-deterministic items of data delay will increase the estimation accuracy and decrease clock skew. This work, which is intended for application in wearable wired networks, uses a fixed timing message length, with the sender taking a sample of its time exactly at the moment of message transmission. The receiver node serves the message immediately, without buffering. In this way, the delay between sender and receiver will be mostly deterministic. The only non-deterministic delay is due to local clock frequency differences between sender and receiver, because the clock oscillators in each sensor are independent and their frequency may exhibit small, variable drifts.

This paper presents a fully digital circuit for one-way synchronization based on this approach implemented at the MAC layer. When used in a wearable mesh network for clinical applications with node connections made through conductive yarns embedded in textile fabrics [2], the Application-Specific Integrated Circuit (ASIC)-based implementation is capable of minimizing the clock skew between neighboring nodes to 4.6 ns on average.

The rest of the paper is organized as follows: Section II provides the background and describes related work. Section III gives an overview of the utilized synchronization protocol. The organization and operation of synchronization circuit

978-1-4799-5793-4/14 $31.00 © 2014 IEEE
DOI 10.1109/DSD.2014.63

is described in IV. Section V presents experimental results obtained from a network of sensors. The main conclusions are summed up in Sec. VI.

II. Background and Related Work

A classification of synchronization protocols based on synchronization issues and application-dependent features can be found in [7]. In a master-slave approach, the master node is the time reference and slave nodes synchronize their clock with the master node, while in peer-to-peer protocols each node can directly get timing information from the other nodes. The advantages of peer-to-peer protocols are failure elimination and flexibility, but their control flow is more complex, In many networks, specially when the number of sensors is large, communication uses a multi-hop connection via intermediate nodes. In this case, synchronization must be performed by sender-to-receiver or receiver-to-receiver approaches. In the former approaches, the receiver node synchronizes its clock with the sender and re-sends timing data to the next node after synchronization; in the latter approaches, the sender broadcasts timing messages and receivers in its coverage area exchange messages among themselves instead of interacting with the sender.

In [13] an external hardware-based clock circuit for use in sensor networks is presented. Synchronization is based on using ambient magnetic field emitted by 60 Hz power lines. The work of [14] describes another clock tuning circuit, but this one is based on the ambient electric field. The first approach is able to achieve an average synchronization between all nodes in a multi-hop network of less than 1 ms; the second approach is able to keep clock drift to within 0.864 µs after 24 hours. The main limitation of this approach is the necessity of having of an additional module; in addition, power line fields are not stable and available everywhere.

The Network Time Protocol (NTP) is a well-known time synchronization protocol based on two-way message exchange [9]. NTP is one the most used protocols, specially on the Internet, and is known as an effective, secure and robust protocol. NTP clients synchronize their local clocks to the NTP time servers by statistical analysis of the round-trip time. To achieve highly precise time synchronization, the time servers are equipped with or synchronized to atomic clocks or GPS signals. The significant complexity of NTP makes its implementation in sensor networks difficult. In addition, the non-determinism in transmission time of WSNs can introduce large delays. Therefore, NTP is suitable only for synchronization in Wireless Sensor Networks (WSNs) with low precision demands.

If a clock source broadcasts its time, adjacent nodes will receive the same timing message at approximately the same time. In this way, receiver nodes can obtain timing information that is subject to very little delay variability. This one-way method has been employed in the RBS protocol [11]. In this approach, which can be used in both wired and wireless networks, a node is selected as a time reference for all other nodes. Each node records its local time immediately after receiving the message and compares it with the received time. This protocol uses a sequence of synchronization messages from the clock source node to estimate both offset and skew of the local clocks relative to each other. Implementation of RBS protocol on IEEE 802.11 (wireless networks) produces one-hop clock synchronization with $1.85\,\mu\text{s} \pm 1.28\,\mu\text{s}$ accuracy.

The Precision Time Protocol (PTP), defined by IEEE standard 1588 [15], is a hierarchical master-slave architecture for clock distribution and time synchronization with highly precise synchronization [8]. In this protocol, the clock server periodically sends synchronization messages. A client replies to the server with a message including the reception time of the first message from the server and the replying time. The server then estimates delay and offset and sends this information with another message. The client uses information in the second message for adjusting the local clock. To avoid or minimize the effects of clock drift on time stamping, client nodes periodically set their time to server time. PTP is a highly precise synchronization protocol, which can ensure clock skew values in the sub-microsecond range.

Although the aforementioned protocols are widely used to provide synchronization in different kind of networks, they are not hardware-aware protocols. In other words, their performance depends on the implementation characteristics of different systems. The main challenge in synchronization is the estimation of the non-deterministic delay of data transmission due to the properties of the hardware. If it is possible to keep variations of the communication and processing delay in hardware level at a minimum, thus making it possible to achieve synchronization with a simpler and more resource-efficient method, as described next.

III. Synchronization Method

This section presents the proposed synchronization method that is implemented in the hardware module that it used as part of the sensor node of a wearable network. The idea is based on eliminating the sources for non-deterministic delays during the time information exchange and compensating the deterministic delays at the receiver. Periodic exchange of time information is used to compensate for the non-deterministic local clock drift. With an appropriately designed one-way method, it is possible to achieve the same accuracy of synchronization with two-way methods. Because this work is focused on the circuit implementation and corresponding experimental results, the detailed description of the synchronization protocol has been omitted. The validity of the equations shown next has been confirmed by the experimental results obtained with the implemented circuit. We assume throughout that all SNs are similar; in particular, their local clocks have the same nominal frequency.

A. Principle of Operation

The limited availability of resources in sensor networks imposes the need to reduce communication and processing as much as possible. However, this effort needs to take into account the accuracy of synchronization. To evaluate a mechanism for synchronization of a wired wearable network, consider the mesh network of SNs shown in Fig. 1. The BS is a server node for collecting data and also is the clock reference. A routing protocol is used to setup the network and route the packets from the sensors to the BS over paths with minimum cost [16]. During setup of the network, each sensor determines

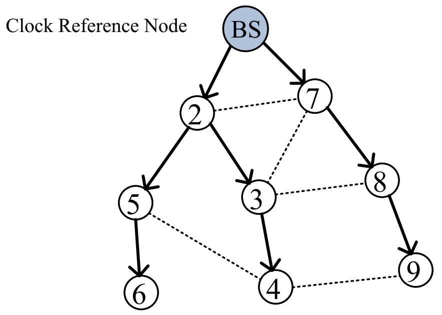

Figure 1. A mesh network of SNs with Base Station (BS) as a base node

which of its neighbour nodes on the path to BS has minimum cost and registers it as its *near-node*. The setup defines a tree of paths from BS to all SNs as shown by the bold lines in Fig. 1. Each SN keeps the real-time clock in synchronization with its *near-node* by sending a request message and receiving time information (e.g. SN3 synchronizes with SN2 and SN7 with BS).

To minimize the effects of non-deterministic delays due to the buffering of messages, the synchronization protocol ought to be implemented at a protocol layer below the network layer. In the approach discussed here, timing information is processed directly in the MAC layer without any buffering. By avoiding the buffering delay, the delay between SNs will be only due to signal propagation and sampling of data at the receiver node.

Figure 2.a depicts the timing diagram of two-way message exchange for synchronization in PTP [15]. Equation 1 and 2 show the delay and offset estimation for this case.

$$\text{delay} = \frac{(T_2 - T_1) + (T_4 - T_3)}{2} \tag{1}$$

$$\text{offset} = \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \tag{2}$$

For a Body Area Network (BAN), the nodes are so close that it can be assumed that all propagation delays are equal. In this case, in a synchronized network can be shown that equations 1 and 2 simplify to:

$$\text{delay} = s \times \tau \tag{3}$$

$$\text{offset} = 0, \tag{4}$$

where s is the number of clock cycle required to send a message and τ is system clock period. Equations 3 and 4 it show that for equal propagation times equations 1 and 2 do not provide extra information.

Now consider a one-way, master-to-slave message exchange as shown in Fig. 2.b. Synchronization is only based on the *Sync* message shown in Fig. 3. The master node sends the *Sync* message at time T_1 and the receiver gets it at a time T_2 given by

$$T_2 = T_1 + s \times \tau + d_\ell + S_r(t), \tag{5}$$

where d_ℓ is the signal propagation delay from sender to receiver, and $S_r(t)$ is the delay due to the phase difference

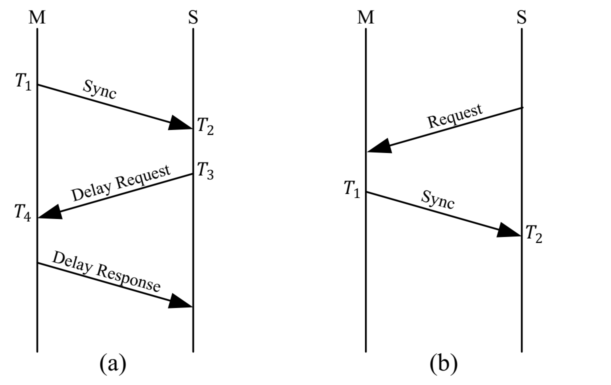

Figure 2. Timing diagram: a) Precision Time Protocol, b) One-way protocol

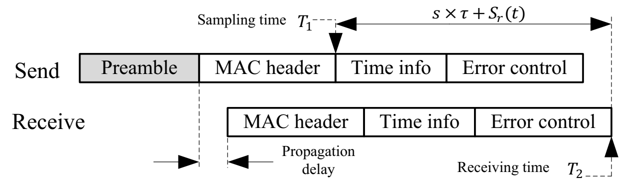

Figure 3. Important time points of the MAC-level message transmission

between sender and receiver clocks. The average value of $S_r(t)$ satisfies the condition

$$\langle S_r(t) \rangle = \frac{1}{2} \times \tau_d = \tau, \tag{6}$$

where τ_d is the data rate period and, for this system, $\tau_d = 2 \times \tau$. The propagation delay d_ℓ is almost constant and usually in the range of a few nanosecond in wearable systems with conductive yarns. The clock offset value is given by the term $s \times \tau$. In the approach used here, the master node adds the offset value to the *Sync* message, so that $C_S(t)$, the clock skew between sender and receiver, is

$$C_S(t) = T_2 - (T_1 + s \times \tau) = d_\ell + S_r(t). \tag{7}$$

The average value of $C_S(t)$ is then

$$\langle C_S(t) \rangle = d_\ell + \langle S_r(t) \rangle = d_\ell + \frac{1}{2}\tau_d. \tag{8}$$

To reduce the average clock skew, the term $\frac{1}{2}\tau_d$ in eq. 8 can be removed by using $s' = s + 1$ instead of s. In this case, the last two equations become

$$C_S(t) = T_2 - (T_1 + s' \times \tau - \tau) = d_\ell + S_r(t) - \tau \tag{9}$$

$$\langle C_S(t) \rangle = d_\ell + \langle S_r(t) \rangle - \tau = d_\ell \tag{10}$$

For a multi-hop connection with h hops between any arbitrary SN and the BS, equations 9 and 10 generalize to

$$C_S(t) = h\,(d_l + S_r(t) - \tau) \tag{11}$$

$$\langle C_S(t) \rangle = h \times d_\ell. \tag{12}$$

because of the accumulation of clock skews after each intermediate node.

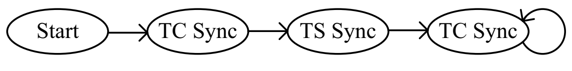

Figure 4. Timing MAC message

The average clock skews given by Eq. 10 and Eq. 12 are the smallest values that can be achieved without any data processing. Because d_ℓ, the delay due to electrical signal propagation, is smaller than the system clock period, it cannot be compensated by this approach.

The approach used in this work achieves the minimum clock skew using a one-way method, without calculations or data processing. The value of s is fixed and deterministic: it depends on the transmitter and receiver circuits, and on the length of the timing message.

B. Protocol Implementation

This subsection summarizes how the approach just described can be used to generate a synchronized local clock and a time stamp.

In this context, assume that all SNs should have a synchronized clock signal *Clk-sync* whose frequency is smaller than the system frequency. One method to generate such a clock signal is to use an auto-reload counter: when the value of a down counter *TC* reaches zero, an active transition of *Clk-sync* is generated and the counter is reloaded with a predefined value *TCPR*. To generate similar clocks, the value of *TCPR* in all SNs must be the same.

Since we want to synchronize the *Clk-sync* signals of all SNs, the values of all *TC* counters must be synchronized. The counter *TC* is driven by the local system clock, which exhibits some clock skew relatively to the system clock of other nodes. The procedure described in the previous subsection can be used to bound this skew. For this, each SN periodically updates its *TC* value with the *TC* value received from the timing message.

Time stamps are used to label the data acquired by all the SNs so that it can be correctly combined. The current time stamp at each SN is maintained by an up counter *TS*, which is driven by the synchronized clock signal *Clk-Sync*. The time resolution of *TS* depends on the frequency of *Clk-Sync*, which must be chosen according to the sensed phenomena: e.g., the EMG signals of [2] have frequencies below 500 Hz.

It should be noted that using *Clk-sync* only ensures that *TS*s counts synchronously. Therefore, it is necessary to synchronize the contents of *TS* with a timing message. Because each SN synchronize its *Clk-sync* periodically, it is not necessary to sync *TS* periodically: after a one-time *TS* synchronization, time stamping will stay synchronized while the system is running. The process is summarized in Fig. 4: after starting, a SN first synchronizes *TC* for synchronization of the local *Clk-sync* signal; next, *TS* is synchronized only once, while *TC* must be synchronized periodically to compensate for the drift of the local system clock.

IV. Synchronization Circuit

This section describes a synchronization circuit based on method of the previous section. The circuit has been implemented as a part of the sensor node communication subsystem

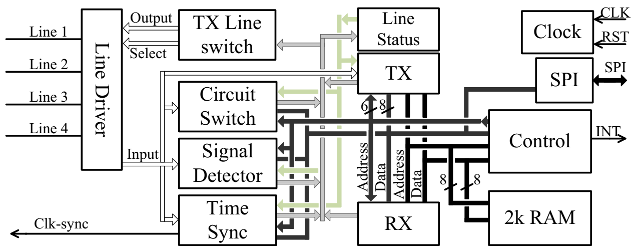

Figure 5. Block diagram of IC including time synchronization module(*Time Sync*).

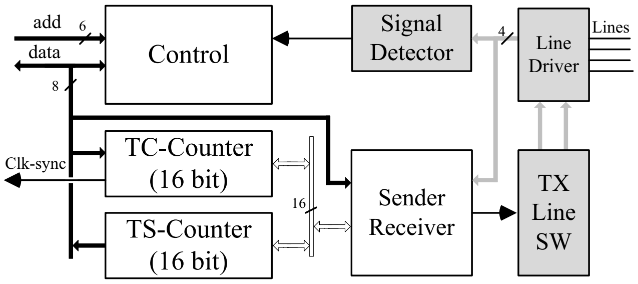

Figure 6. Block diagram of the circuit.

shown in Fig. 5 (*Time Sync* module) for use both in FPGA and in a CMOS ASIC.

Figure 6 shows the block diagram of the circuit including all necessary modules to send and receive timing information. The three modules in gray (*Signal Detector, TX Line SW and Line Driver*) do not belong to the synchronization circuit, but they are involved in the communication process. An 8-bit internal system bus connects the module to the system core, which controls the time synchronization circuit, and reads or writes to the registers (e.g., the real-time time stamp). A 16-bit bus is used to connect the internal modules and to access the two counters *TC* and *TS*. The signal *Clk-sync* is a synchronized clock signal that is available at one of the output pins. The synchronization circuit also manages a real-time time stamp that is available via internal bus. The operation of each module is described next.

A. Control Module

The control module manages all the activities related to time synchronization. It is connected to the system core by the internal 8-bit bus and also to the *Signal Detector*. All the register values can be read or written via this module.

The SN must send a *Request* message to its *near-node* every time it needs to update the values of the timing counters *TC* and *TS*. The procedure is started by a command from the microcontroller, which is sent via the SPI port to the *Control* module. The latter sets the synchronization module up for sending and receiving timing information, and starts the transmission.

The SNs employ several kinds of MAC messages, with timing synchronization being just one of them. The detection of a timing request message from an adjacent SN is carried

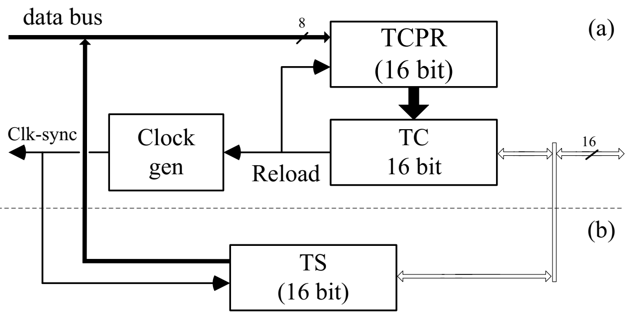

Figure 7. Counters: a) TC-Counter, b) TS-counter

out by the *Signal Detector* module. This module is responsible for determining the type of the MAC message by decapsulating and processing the MAC header, and forwarding the rest of incoming data to the destination module (which is the synchronization circuit for timing messages). For timing message the MAC header also determines whether the request is for updating the *TC* or the *TS* counter. The *Control* module receives the timing message from the *Signal Detector* module and updates the appropriate counter as described next.

B. TC-Counter and TS-Counter Modules

The *TC-Counter* module shown in Fig. 7.a generates the clock signal *Clk-sync* for time stamping and use as a synchronized clock by other part of sensors. Whenever the down counter 16-bit *TC*, which is connected to the global system clock, reaches zero, it is reloaded with the value of the 16-bit *TCPR* register. The periodic *Reload* signal is used to drive the clock generation circuit *Clock_gen*, which produces the reference clock signal *Clk-sync*. As mentioned before, this clock signal is used as input clock of the *TS-Counter* module, and is also available for use as clock reference in other parts of the sensor node. The output of this module is a pulse signal with duration of 1 or 32 system clock cycles. The wider pulses are necessary for some circuits. For example, when *Clk-sync* is used as an external interrupt of a microcontroller, a short pulse width may not be sufficient to generate a valid interrupt.

The value of *TCPR* must be in range 1–65535 (33–65535 for the wide pulse mode). The frequency f_{sync} of *Clk-sync* is given by

$$f_{\text{sync}} = \frac{f_s}{1 + \text{TCPR}}, \tag{13}$$

where f_s is the frequency of system clock. f_{sync} with $f_s = 20\,\text{MHz}$ will be in range 300 Hz to 10 MHz.

Figure 7.b shows the *TS-Counter* module. This module includes *TS* and generates time stamping that can be read by the system, or even written, via 16 bit internal bus. Like *TC*, other SNs are able to read the *TS* for updates of their own *TS* counter. The time interval between *TS* overflows is:

$$T_{\text{ov}} = \frac{65536}{f_{\text{sync}}}. \tag{14}$$

The wearable system of [2] is designed to capture EMG signals using a 1 kHz sampling frequency. To generate the *Clk-sync* reference for the EMG signal sensing part from a 20 MHz system clock , the *TC* counter must have at least a 16-bit length.

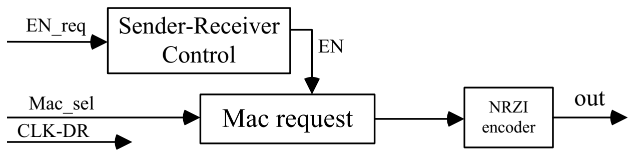

Figure 8. Configuration of *Sender Receiver* to send a request.

C. Sender-Receiver Module

In our context, synchronizing two SNs is equivalent to ensuring that their *TC-Counters* have identical values at the same time. This ideal situation cannot be ensured at all times, but a small bound on the difference between the counters may be enough for practical purposes. This is achieved by an exchange of special messages between the nodes. All processing of timing messages is done by the *Sender-Receiver* module. It may operate in one of three different modes:

- *Request for timing information (Figure 8):* start the synchronization process by sending a request message;
- *Reply to a Request (Fig. 9):* reply to a timing request message;
- *Reception of timing message (Fig. 10):* receive timing information in response to a request message.

1) Timing Request Message: In order to illustrate the operation of *Sender-Receiver* module, we will consider the situation where node SN3 in Fig. 1 wants to synchronize its *TC* value with its *near-node* SN2.

The process is started by the upper layers of the control software of node SN3 by sending a command (via the internal bus) instructing the *Control* module to activate request mode. The *Control* modules instructs the *Sender-Receiver* module to start communication with SN2. The configuration of the module in this mode is shown in Fig. 8.

The *Control* module generates both of the *EN_req* and *Mac_sel* signals: the first enables the request mode and the second specifies that the request concerns the value of *TC*. Based on the request type (which could be *TC* or *TS*), the *Mac_ Request* module generates the appropriate MAC message, which must then be encoded for transmission. The line encoding scheme used in our implementation is Non-Return-to-Zero Inverted (NRZI). Therefore, the output of *mac request* is encoded before driving the line. Module *TX SW* (Fig. 6) forwards the MAC message to the line connected to SN2. At the end of the process, the *Control* module changes the configuration of the *Sender-Receiver* module to reception mode and waits for a reply from SN2.

2) Timing reply message: When the *Signal Detector* module of SN2 detects the timing message request, it forwards the incoming data to the synchronization circuit. The *Control* module enables reply mode by asserting the *EN_rep* signal. In this mode, node SN2 must send the value of its *TC* counter to SN3, after performing an adjustmento to account for the

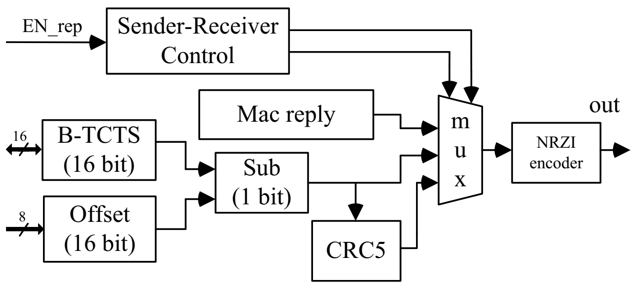

Figure 9. Configuration of *Sender-Receiver* for reply to a timing information request.

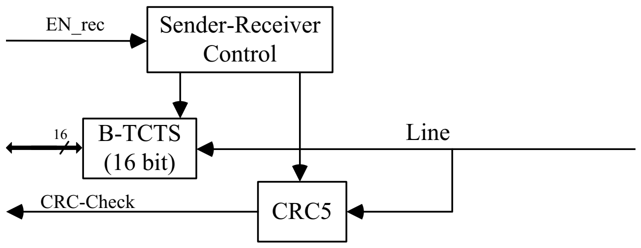

Figure 10. Configuration of *Sender Receiver* to receive timing message.

offset between the nodes. The configuration of the *Sender-Receiver* module is the one shown in Fig. 9. In that figure *B-TCTS* is a 16-bit register for buffering the value of *TC* is in node SN2 at the start of reply processing. The 16-bit register *Offset* is used to memorize the offset value, which will be used to compensate for the skew introduced by the transmission delay. The value of *Offset* can be managed by the upper software layers through the *Control* module. Module *Sub* is a serial subtractor that is used to subtract *Offset* from the *TC* value buffered in *B-TCTS*. According to equation 5, the *Offset* value must be added to the *TC* value that is sent back to SN3. However, *TC* is a down-counter, so the *Offset* value must be subtracted instead. It should be noted that a 1-bit serial subtractor can be used to perform the subtraction of two 16 bit values, since communication is serial and the subtraction can be performed during reception.

To protect and check the validity of the data at the receiver node, the *CRC5* module generates a 5-bit cyclic redundancy check, which is concatened to the outgoing data. A multiplexor controlled by *Send-Receive Control* selects, in order, the *Mac reply*, *Sub* and *CRC5* modules to build the full reply message at the output, which goes through the NRZI encoder before being sent to node SN3.

3) Reception of Timing Message: As mentioned previously, node SN3 changes the configuration of the *Sender-Receiver* module after sending the timing request message. The configuration for reception is shown in Fig. 10. In this case, register *B-TCTS* acts as a Serial-In-Parallel-Out buffer to receive the *TC* value. After the *CRC5* module confirms the validity of the received data, the value buffered in *B-TCTS* is loaded to the *TC*.

The processor for updating *TS* is the same as the one used for *TC*, with the only difference that, in the reply step, the value of *TS* will be sent without any adjustment. As mentioned in Sec. III, *TS* counters inherit synchronization from *Clk-sync* signals; exchanging *TS* values is necessary only for time stamp alignment.

Table I. MAIN CHARACTERISTICS OF THE EXPERIMENTAL TEST SETUP

Parameter	Value
Technology	0.35 µm
# Logic cells	971
Total area	0.138 mm^2
Supply voltage	3.3 V
Clock frequency	20 MHz
Data rate	10 Mbps
Clk-sync frequency	1 KHz

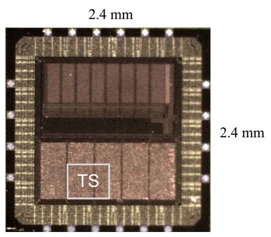

Figure 11. CMOS ASIC (0.35 µm) microphotograph with indication of the synchronization circuit (TS).

V. Experimental Results

This section presents experimental results obtained with the IC version of the circuit, which has been fabricated in 0.35 µm CMOS technology and is shown in Fig. 11. In this figure, TS is the synchronization circuit. The main parameters of the setup used for validation are summarized in Tab. I. The network of SNs includes the nodes BS, SN2, SN3 and SN4 shown in Fig. 1. Since this circuit is part of a wearable system that is used to acquire EMG signals at 1 kHz, this frequency was also chosen for *Clk-sync*.

A. Physical Layer Signaling

Signals generated in the physical layer during message exchange (request and reply) for synchronization of the *TC* are shown in Fig. 12. As mentioned before, the line between the nodes is bidirectional, so the reply message appears on the same line immediately after the request message. The Least Significant Bit (LSB) of the adjusted *TC* value is sent first; the message trailer consistes of the CRC check bits.

For the evaluated setup, the entire process, from sending the request message to the end of the reply message, takes 5.2 µs. To keep network nodes synchronized, it is necessary to exchange timing messages periodically, which requires that utilization of system resources, such as channel time should be take into account. For an interval between timing messages of 1 ms only 0.52 % of the channel time will be used for synchronization, a very small percentage of the total traffic on the network.

B. One-Hop and Multi-Hop Clock Skew

The one-hop clock skew between SN2 and BS can be seen in Fig. 13. The oscilloscope signals shown in the figure are

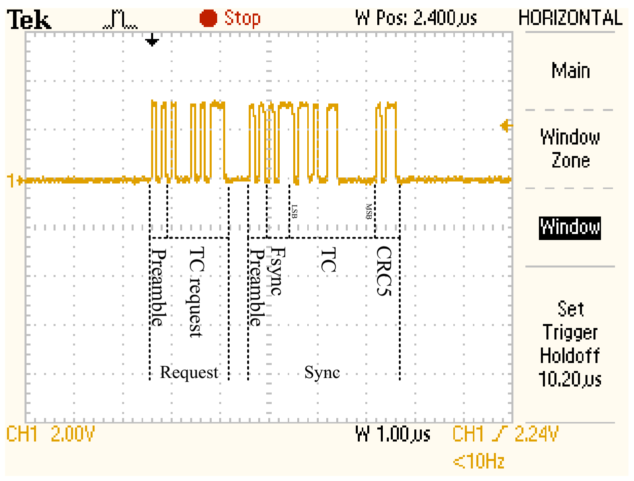

Figure 12. Physical layer signals during timing message exchange between SN2 and BS.

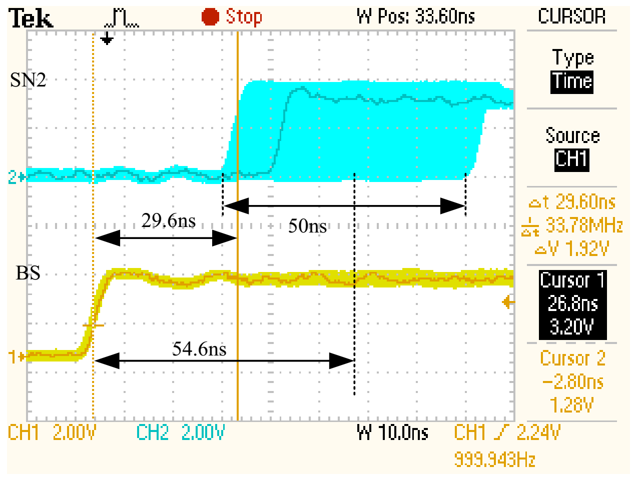

Figure 13. One-hop clock skew.

generated by using the "infinite time persist" display mode. The system clock of the BS is used as reference in all skew measurements. The offset value for compensating the constant delay value is measured to be s = 0x0030. The observed clock skew variation range is 50 ns, that is one cycle of the system clock, as described by Eqs. 7 or 11. The average one-hop clock skew is 54.6 ns, that is, the sum of signal propagation delay and S_r.

By increasing the number of hops, the clock skew increases as well. Figure 14 depicts the clock signal at different nodes (the offset value, 0x0030, is the same for all the nodes). As can be seen and expected, the average clock skew increases by 54.6 ns as the number of hops increases. In agreement with to Eq. 11, the clock skew variation range also increases: 50 ns at node SN2, 100 ns at node SN3, and 150 ns at node SN4.

The effect of changing the offset value is shown in Fig. 15. All the signals and settings, except for the offset values, are the same as shown in Fig. 14. In this case, the offset is $s' = s-1 =$ 0x002F. As can be seen, all clock signals have been shifted 50 ns to the left and the average clock skew is much better

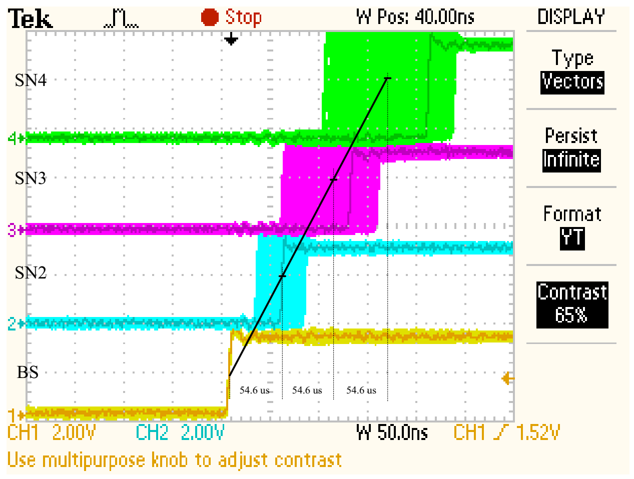

Figure 14. Multi-hop clock skew with offset = 0x0030.

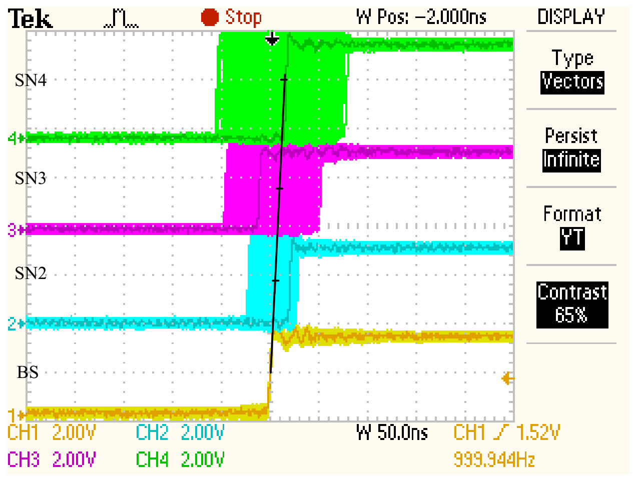

Figure 15. Multi-hop clock skew with offset = 0x002F.

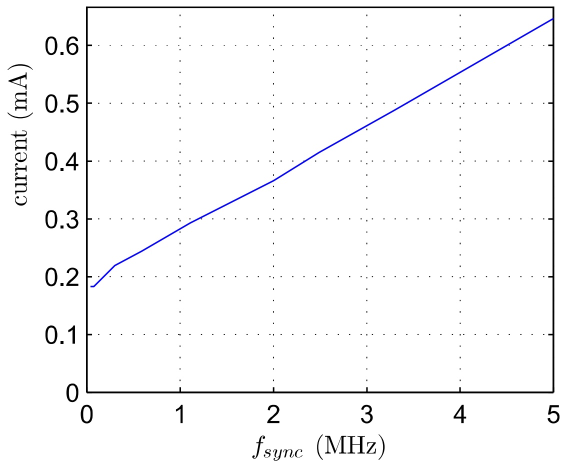

Figure 16. The circuit current consumption for f_{sync} up to 5 MHz

than before; clock skew variation ranges are exactly the same as previously. For this setup, the average clock skew is 4.6 ns , which is much smaller than the previous value of 54.6 ns. This shows the feasability of minimizing average clock skew by selecting appropriate values for the offset.

Figure 16 depicts the measured current consumption of the circuit as a function of the *Clk-sync* frequency from 305 Hz to 5 MHz. The current increase linearly from 0.18 mA to 0.64 mA as the frequency increases. With *Clk-sync* = 1 kHz current is almost 0.18 mA. The main of current is used to drive the *Clk-sync* pin that is available as an output of the ASIC.

The measured value of 4.6 ns is due to the signal propagation from output of sender node to the input port of receiver node, an unavoidable inherent characteristic of the communication path (I/O pads and connecting yarn). In resource-limited systems such as sensor networks, achieving a smaller clock skew is difficult in practice, even if more sophisticated two-way methods are used, as it would require more processing and a more complex implementation, Therefore, the low-overhead approach described here makes it practical for many wearable systems to achieve a very-low average one-hop skew (4.6 ns for the wired, yarn-based application used for evaluation) with a one-way method and without any further processing.

VI. Conclusion

The circuit described in this paper was designed for establishing time synchronization between sensor nodes of a wearable system. The synchronization is based on one-way master-to-slave message exchange implemented in the MAC layer, in order to avoid the non-deterministic delays caused by data processing and buffering in the higher levels of the protocol stack. By directly sending and processing the timing information without buffering, the proposed approach leads to an average clock skew of a few nanoseconds. The circuit generates two synchronized values: a programmable clock signal and a real-time counter for time stamping purposes.

Experimental evaluation with an ASIC implementation obtained an average one-hop clock skew of 4.6 ns that is the time required for signal propagation from sender output to the receiver input. Based on theoretical calculations, in a multi-hop network, the global average time skew grows linearly with hop count; this is supported by the experimental results. The low skew values provided by this approach satisfy the requirements of many BAN applications. Even for networks whose nodes are 10 hops away from the time reference node, the average global skew will typically be under 50 ns. A value of 10 hops exceeds the largest inter-node distance of many, if not all, existing wearable systems. The proposed circuit achieves the maximum synchronization performance that could be achieved by PTP, but with fewer timing messages and calculations, less complexity and better energy efficiency.

References

[1] P. Ranganathan and K. Nygard, “Time synchronization in wireless sensor networks: A survey,” *Intl. J. UbiComp*, vol. 1, no. 2, pp. 92–102, 2010.

[2] A. Zambrano, F. Derogarian, R. Dias, M. Abreu, A. Catarino, A. Rocha, J. da Silva, J. Ferreira, V. Tavares, and M. Correia, “A wearable sensor network for human locomotion data capture,” in *9th Intl. Conf. on Wearable micro and nano technologies for personalized health; pHealth2012*, 2012, pp. 216–223.

[3] F. Sivrikaya and B. Yener, “Time synchronization in sensor networks: a survey,” *IEEE J. Network*, vol. 18, no. 4, pp. 45–50, 2004.

[4] S. Lasassmeh and J. Conrad, “Time synchronization in wireless sensor networks: A survey,” in *Proc. IEEE SoutheastCon 2010 (SoutheastCon)*, 2010, pp. 242–245.

[5] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, “Clock synchronization of wireless sensor networks,” *IEEE Magazine, Signal Processing*, vol. 28, no. 1, pp. 124–138, 2011.

[6] I.-K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y.-C. Wu, “Clock synchronization in wireless sensor networks: An overview,” *J. Sensors*, vol. 9, no. 1, pp. 56–85, 2009. [Online]. Available: http://www.mdpi.com/1424-8220/9/1/56

[7] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, “Clock synchronization for wireless sensor networks: A survey,” *Elsevier J. Ad Hoc Networks*, vol. 3, pp. 281–323, 2005.

[8] K. Lee and J. Eidson, “IEEE-1588 standard for a precision clock synchronization protocol for networked measurement and control systems,” in *34th Annual Precise Time and Time Interval (PTTI) Meeting*, 2002, pp. 98–105.

[9] D. Mills, “Internet time synchronization: the network time protocol,” *IEEE J. Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[10] S. Ganeriwal, R. Kumar, and M. B. Srivastava, “Timing-sync protocol for sensor networks,” in *Proc. 1st Intl. Conf. on Embedded Networked Sensor Systems*, 2003, pp. 138–149. [Online]. Available: http://doi.acm.org/10.1145/958491.958508

[11] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” in *Proc. 5th symposium on Operating systems design and implementation, SIGOPS*, 2002, pp. 147–163. [Online]. Available: http://doi.acm.org/10.1145/1060289.1060304

[12] C. Lenzen, P. Sommer, and R. Wattenhofer, “Optimal clock synchronization in networks,” in *Proc. 7th ACM Conf. on Embedded Networked Sensor Systems*, ser. SenSys ’09, 2009, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/1644038.1644061

[13] A. Rowe, V. Gupta, and R. R. Rajkumar, “Low-power clock synchronization using electromagnetic energy radiating from ac power lines,” in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys ’09. New York, NY, USA: ACM, 2009, pp. 211–224. [Online]. Available: http://doi.acm.org/10.1145/1644038.1644060

[14] M. Buevich, N. Rajagopal, and A. Rowe, “Hardware assisted clock synchronization for real-time sensor networks,” in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, Dec 2013, pp. 268–277.

[15] IEEE 1588-2008 standard, *IEEE 1588-2008 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, 2008.

[16] F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “A routing protocol for WSN based on the implemention of source routing for minumum cost forwarding method,” in *Proc. 5th Intl. Conf. on Sensor Tech. Appl. (SENSORCOMM 2011)*, Aug. 2010, pp. 85–90.