INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS
Int. J. Circ. Theor. Appl. 2016; **44**:530–548
Published online 27 April 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cta.2092

A small fully digital open-loop clock and data recovery circuit for wired BANs

Fardin Derogarian*,†, João Canas Ferreira and Vítor Grade Tavares

INESC TEC, Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, Porto, 4200-465, Portugal

SUMMARY

This paper proposes a new open-loop and low complexity (small size) fast-lock synchronization circuit for clock and data recovery in wearable systems. The system includes sensors embedded in textile and connected by conductive yarns. Synchronization is based on the open-loop selection of the correct phase of the receiver clock synchronously with the incoming signal. The clock generator of the receiver is an autonomous oscillator set to operate at the same nominal frequency. The circuit lock time is at most one clock cycle, faster than all methods based on phase-locked loops or delay-locked loops. The circuit can be used for baseband communication independently of the signal coding method used in the physical layer, making it suitable for many applications. The fully digital circuit (including non-return-to-zero inverted decoder) occupies 0.0022 in a 0.35 complementary metal-oxide semiconductor (CMOS) process, a smaller implementation than many existing circuits, and supports a maximum system clock frequency of 70 for a 35-data rate. Experimental results demonstrate that the proposed circuit robustly generates a synchronous clock for data recovery. The circuit is suitable for systems that tolerate some jitter but requires fast lock time, small size, and low energy consumption. Copyright © 2015 John Wiley & Sons, Ltd.

Received 31 July 2014; Revised 15 December 2014; Accepted 16 March 2015

KEY WORDS: clock and data recovery (CDR); open-loop clock synchronization; fully digital CDR; wearable sensor network; body area network (BAN)

1. INTRODUCTION

In any digital communication system, synchronization between receiver and transmitter for data recovery plays a major role. Synchronization refers to the process of making events occur at the same time in different parts of a distributed system [1–3]. To perform a successful clock and data recovery (CDR), symbol or bit synchronization is performed in every digital receiver by extracting timing information from the received signals. A decider circuit at the receiver uses that timing information to determine when to sample the incoming data. The optimum sampling instant is in the middle of the bit period. However, sampling precisely at that instant is not guaranteed, because of channel noise and jitter. Nonetheless, receivers can still correctly detect data by sampling within a small interval of time around the ideal instant. The sampling period depends entirely on the method used for clock recovery. Usually, keeping the sampling period in small ranges demands an increase in bandwidth to include more information for synchronization and/or more complex receiver architectures.

Depending on the synchronization method, symbol synchronization is classified in two main groups: (1) feedback or closed-loop and (2) feedforward or open-loop [1,4]. In the closed-loop approach, the receiver attempts to lock a local oscillator to the incoming clock and generates a

*Correspondence to: Fardin Derogarian, INESC TEC, Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, Porto, 4200–465, Portugal.
†E-mail: mpt09020@fe.up.pt

synchronized clock to recover the data. Usually, a phase-locked loop (PLL) or a delay-locked loop (DLL) is used to track the incoming clock [5]. The early–late gate synchronizer is one of the must widely used closed-loop methods [6].

Like all feedback systems, closed-loop synchronizers need some time to lock and produce a stable signal at the output. Open-loop synchronizers avoid this delay. A popular approach is to generate the sampling clock by filtering the incoming signals with a band-pass filter and by feeding the resulting signal to a comparator, thereby creating a square clock signal. For this approach to work, the base-band modulation has to contain a spectral component at the data rate to enable the extraction of clock signal by filtering. Another method relies on oversampling the incoming data at a rate that is at least three times higher than the data rate [7, 8]. The recovered data value is chosen to be the value (‘1’ or ‘0’) that has been sampled more often in each data period. For the decision to be unambiguous, the number of samples must be odd. Oversampling is able to provide data recovery without delay.

Clock and data recovery circuits based on closed-loop systems are more complex than open-loop systems, but they are widely used in many systems because of their good performance with low signal-to-noise ratio (SNR) systems and signals. Nevertheless, in communication systems with relatively high SNR and limited resources like sensor networks in wearable systems, an appropriately designed open-loop system may be preferable. This paper describes a CDR circuit based on open-loop synchronization for use in wearable systems consisting of several sensors connected by conductive yarns [9, 10]. In this particular application, each sensor node (SN) has four bidirectional ports and acts both as a data collector and a router in a mesh network. Sensor nodes are able to handle simultaneous communication with other nodes. For that purpose, each port is equipped with an independent CDR circuit. Because there are several instances of the CDR in each node, the aim of this work is reducing the circuit size as much as possible. Its main contributions are as follows:

(1) the design of a very small, fully digital, fast-lock clock CDR circuit for use in resource-constrained SNs;
(2) the implementation of the design in two different technologies: Actel field-programmable gate arrays (FPGAs, Actel Corporation Mountain View, CA, USA) and a 0.35 CMOS integrated circuit (IC); and
(3) the experimental evaluation of the proposed design using fabricated IC prototypes.

The rest of the paper is organized as follows: Section 2 provides some background on synchronization methods and describes related work. Section 3 presents the motivation for the proposed circuit. Section 4 gives an overview of the adopted method, while section 5 discusses and analyzes the implementation. Section 6 presents and discusses the experimental results, followed by the main conclusions in section 7.

2. RELATED WORK

Many CDR methods have been discussed in the literature. Descriptions of the fundamental aspects of synchronization and of basic CDR techniques can be found in [3], [4], and [11]. As mentioned in the Introduction section, in many open-loop methods, the data signal carries spectral information about the clock signal. Filtering methods process the input signal and then use a band-pass filter to remove unwanted frequency components, followed by a high gain amplifier and a Schmitt trigger to generate a square clock signal. A simple processing step is to rectify the signal by using a square law device [12]. Another method multiplies the incoming signal with a 90-phase shift of the carrier signal, which is generated by filtering of the incoming signal [13]. The third method uses an edge detector at the input to detect the transitions of the signal and to generate spikes and recover clock from spikes [13].

Filtering methods are easy to implement and exhibit no-lock time problems. Their main disadvantages are unavoidable non-zero mean tracking, low performance in the presence of noise, and the requirement to carry clock spectral information in the signal.

DOI: 10.1002/cta

Oversampling CDR circuits implement another widely used feedforward recovery approach [7,14, 15]. In this method, each received data-bit is sampled in multiple points within its time interval. To properly recover the data, at least three (but often five or more for higher accuracy) samples per bit are necessary. All sampled data are saved in a (FIFO) memory. For sifting the data from the samples, a bit boundary detector finds the sampled bits at the data edge; then, a data selector determines the appropriate samples near the middle of incoming signal and recovers the data. Oversampling methods result in fast and stable CDR circuits, but they need a high sampling rate and large FIFO memories [7].

In more complex communication systems, feedback methods have been used to overcome the disadvantages of the filtering and oversampling approaches [3], [16–19]. In these systems, a local voltage-controlled oscillator (VCO), placed in a closed-loop configuration, generates the desired clock signal. In many cases, a PLL or DLL is used to generate the clock signal at the receiver, which is synchronized with the incoming data stream and is used for recovering the data bits [20–27]. Lock speed depends on the design, with DLLs being usually faster than PLLs. Several DLL-based designs are compared in [28], with the fastest method requiring at least five clock cycles to lock to the incoming signal.

Early–late gate synchronizer is another widely used closed-loop CDR circuit [29–31]. This method performs two integrations of the received signal over two different periods of the symbol interval. The difference of integrals is used as a feedback signal to control the local oscillator.

Another approach works by injecting data directly in a ring oscillator [32]. A VCO-based ring oscillator is used to generate four differential phases of the clock, and a selector circuit is used to select one of the four phases as the recovery clock. This method is more complex than the open-loop methods, but it achieves instantaneous CDR.

Although various CDR methods are available, specific systems may have requirements that do not match the characteristics of the available circuits. The next section discusses the motivation for the design of a CDR circuit for the use in a wearable sensor system that comprised SNs interconnected by conducting yarns embedded in textile fabrics.

3. MOTIVATION

The work described here was performed in the context of a wearable system designed to measure human lower limb activity. The system includes a number of SNs connected to each other by conductive yarns in a mesh topology, as shown in Figure 1.

From the standpoint of the network, each SN is a four-port router with bidirectional links to other SNs. The SNs are equipped with a local crystal oscillator. When sensors communicate, the receiver node must be able to synchronize its clock to the incoming data. Because there is no global synchronization, a local synchronization method is necessary. The following factors were taken into account in the design of the clock synchronization circuit:

- Size: In the architecture described here, SNs are multitasking network devices that can handle several transmitting or receiving connections simultaneously. In the prototype implementation, each SN has four independent CDR circuits. In these situations, it is important to have small circuits, because it is impractical to have several PLLs/DLLs and FIFO memories in resource-constrained SNs: for instance, smaller FPGAs like the IGLOO® nano family [33] have at most one PLL.
- Energy consumption: SNs are typically energy-limited systems. Traditional open-loop approaches need higher bandwidth to carry synchronization signals, increasing the energy consumption at the transmitting part. Nevertheless, they usually consume less energy than the closed-loop ones, because of their simpler structure and absence of modules like VCOs or frequency synthesizers.
- Bandwidth: Although conductive yarns are similar to regular copper wires, they exhibit a much higher electrical resistance, leading to increasingly significant signal attenuation with increasing frequency. This drawback leads to the used communication mechanisms or modulations that

DOI: 10.1002/cta

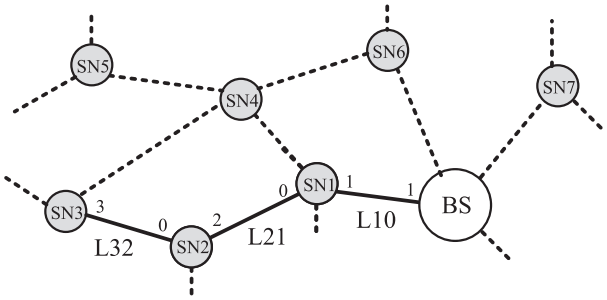

Figure 1. A mesh-based body area network. The base station (BS) gathers the data collected by the sensor nodes (SNs). All nodes are connected by conductive yarns embedded in textiles.

reduce the necessary bandwidth. However, open-loop approaches based on band-pass filtering require clock spectral information to be present in the signal, thereby increasing bandwidth usage, because clock frequency is usually two times the data rate of baseband communication.

- Internal node synchronization: Even if precise clock recovery is achieved, synchronization between the core of the system and the recovered clock signal should be taken into account. Each sensor is a multitasking device that must be able to communicate with all its neighbors simultaneously. The core of the system must be able to process data from different sensors simultaneously (using different recovered clocks), and therefore some kind of internal synchronization is required. FIFO memories or elastic buffers that can be used to do internal synchronization, but they increase circuit size.

The open-loop synchronization method introduced in this work is a trade-off between the aforementioned factors. Like open-loop approaches, it has a less stable sampling point than provided by closed-loop methods, but it is able to recover data with high accuracy and very low BER. Because of the small number of elements needed to implement the circuit, size and energy consumption are much smaller than of any oversampling or closed-loop circuit. Closed-loop circuits need some preamble signals to lock on the incoming signal and to generate a stable clock, but the circuit presented here only needs one transition in the incoming signal to start the clock synchronization. In contrast to some open-loop circuits, the proposed circuit does not use a filter, and there is no need for the incoming signal to include clock spectral information. Therefore, the new circuit architecture utilizes less bandwidth than open-loop systems based on band-pass filtering. Finally, the proposed circuit uses the system clock as reference, so the receiver circuitry is always synchronized with the core of SN without using elastic buffers or FIFO memories.

4. SYNCHRONIZATION METHOD

This section describes the proposed clock synchronization method for CDR without using feedback loops or embedded spectral information. We also determine the conditions for correct operation: the data rate must not exceed half the clock frequency.

4.1. Autonomous clock oscillators and synchronization

Crystal quartz resonators have very low variability, slight temperature dependency and present very high quality factors; they are the preferred devices for generation of stable waveforms [34]. Therefore, we assume that SNs have an autonomous clock generator that provides a signal with the same nominal frequency and drives the whole system including transmitter and receiver modules.

The output of an ideal oscillator is given by eq. (1)

$$C(t) = A\sum_{k\in Z} p(t - kT) \tag{1}$$

DOI: 10.1002/cta

Here, A denotes the signal amplitude, T is the clock period, and $p(t)$ is the unit square pulse with $T=2$ width. In addition to inter-device variations, a real clock waveform also suffers from noise, temperature and power supply variations, and other ambient effects. Equation (2) shows the real local clock waveform at SN n,

$$C_n(t) = A\sum_{k\in Z} p(t - kT_n - \tau_n(t)), \tag{2}$$

which includes phase noise $\tau(t)$ (with zero mean value) and clock period T_n. The term $\tau(t)$ causes the signal to shift randomly over time and generates jitter. The frequency of each oscillator ($f_n = 1/T_n$) is a random variable that is uniformly distributed around the nominal frequency with accuracy of a few parts per million.

Consider the case where a SN is communicating with its neighbor in base band (e.g., SN1 and SN2 in Figure 1). In general, recovery of clock timing depends on the line encoding. For example, in Manchester coding, there is at least one transaction for each transmitted bit [4]. In other words, spectral information of clock signal is transferred to the receiver, which can easily recover the timing information. The drawback of such an encoding method is the need for higher bandwidth. As discussed in section 3, the present work does not use this approach.

In contrast, encoding methods like non-return-to-zero (NRZ) or NRZI need half of the bandwidth for the same amount of data, but the receiver has to use a more complex method to extract timing information from the incoming signal [4]. Figure 2 shows a NRZI-encoded signal $S(t)$ produced by encoding the signal $m_s(t)$ using clock $C_s(t)$ with period T_s (subscript s denotes to the sender node). The dashed line indicates the optimum sampling time at the receiver, which is in the middle of the data interval. For sampling to occur at that time, the phase difference between receiver and transmitter clocks must be zero, but noise and jitter may cause the receiver not to take a sample at the optimum instant. However, as long as sampling occurs around the optimum point (the gray region in Figure 2), it is possible to recover data successfully. The boundaries of the sampling range depend on the signal jitter.

The autonomous system oscillator at the receiver has a fixed frequency, which is m times the data rate, and generates a recovered synchronized clock signal from the system clock by tracking the transitions of the incoming signal. The recovered clock is a set of pulses with frequency equal to the data rate (m times slower than the system clock); the CDR's task is to shift the pulses so that the active edges are always in the sampling range for correct data recovery. The general idea is to detect if an edge of the incoming signal occurs close to an active edge of the local clock signal. If the incoming edge occurs just before an active clock edge, the next clock edge is delayed by half the data period; if the incoming edge occurs just after the active edge, the following one is advanced by the same amount of time.

Considering the clock signal given by eq. (2), the transmitter's clock phase $\phi_s(t)$ is given by

$$\varphi_s(t) = 2\pi\frac{t_s(t)}{T_s}, \tag{3}$$

where $t_s(t)$ is

$$t_s(t) = t - kT_s - \tau_s(t). \tag{4}$$

The receiver's clock phase $\varphi_r(t)$ is given by a similar equation. Using the index s for quantities associated with the transmitter and index r for those associated with the receiver, the phase difference between sender and receiver is given by

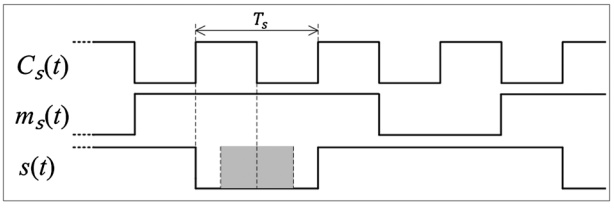

Figure 2. Non-return-to-zero inverted (NRZI) signal $S(t)$ generated from $m_s(t)$.

DOI: 10.1002/cta

$$\Delta\varphi(t) = \varphi_s(t) - \varphi_r(t) = 2\pi\left(\frac{t_s(t)}{T_s} - \frac{t_r(t)}{T_r}\right). \tag{5}$$

Assuming that the mean values of the phase noise $\tau_s(t)$ and $\tau_r(t)$ are zero and that their values in any given cycle k are negligible, then eq. (5) can be rewritten as

$$\Delta\varphi(t) \approx 2\pi t\left(\frac{1}{T_s} - \frac{1}{T_r}\right) = 2\pi t(f_s - f_r) = 2\ \pi t\ \Delta f. \tag{6}$$

Equation (6) shows that the $\Delta\varphi(t)$ changes linearly over time with a constant slope $2\pi\Delta$f.

As a preliminary step, we evaluate the effects of $\Delta\varphi(t)$ on synchronization when the receiver clock generator frequency is equal to data rate. Assume that the receiver takes samples of data at the negative edge of the clock, and that at $t=0$, the receiver clock is in synchronization for sampling at the optimum point ($\Delta\varphi=0$). Over time, $\Delta\varphi(t)$ changes until it is out of the sampling range at t_1 Therefore,

$$\Delta\varphi(t_1) = \Delta\varphi_{sp} \Rightarrow t_1 = \frac{\Delta\varphi_{sp}}{2\pi\Delta f}, \tag{7}$$

where $\Delta\varphi_{sp}$ is the maximum phase difference between $C_r(t)$ and $C_s(t)$ so that data is correctly detected (negative edge of receiver clock is in the gray area of Figure 2). For $t > t_1$, the clock receiver is out of synchronization and needs to shift the negative edge to be in the sampling interval again. As mentioned in section 3, synchronization between peripheral modules and system core must also be maintained without using FIFO memories. Any partial shifting of the clock (as performed in DLLs) will make it difficult to keep the synchronization with the system core and will require FIFO memories and a more complex circuit. To keep the circuit simple, shifting of the clock signal must be performed by a full clock period to the right or to the left. However, shifting the clock signal by one cycle keeps the relative sampling time unchanged. Therefore, under these constraints, the receiver cannot detect data correctly if its frequency is the same as the data rate.

4.2. Synchronization using a faster system clock

One way to overcome that problem is to increase the system clock frequency and to generate the receiver clock by division. This situation is shown in Figure 3.

Signal $C_2(t)$ is the system clock and has twice the frequency of $C_r(t)$, which matches the data rate. The $C_r(t)$ signal is generated locally in the receiver by division of $C_r(t)$ at the raising edge and synchronized with the incoming signal. The arrows on the falling edges of $C_r(t)$ indicate the sampling times, and the gray interval marks one specific sampling range.

As in the previous case, whenever the falling edge of $C_r(t)$ would fall outside the sampling range, the CDR circuit has to detect the situation and shift the edge to the correct sampling range (cf. section 5). In this case, the shifting of $C_r(t)$ will be by one $C_2(t)$ clock period or $\frac{1}{2}T_r$. For example, if the frequency of $C_r(t)$ is a little lower than the data rate, then $C_r(t)$ will be moved to the right in relation to the incoming data. When the skew of the optimum sampling point reaches $\frac{1}{4}T_r$, the receiver shifts $C_r(t)$ from point t_r in Figure 3 to point t_l on the left. Therefore, the phase difference between transmitter and receiver clock will be kept in the range

$$\frac{-\pi}{2} < \Delta\varphi(t) < \frac{\pi}{2}. \tag{8}$$

4.2.1. Impact of jitter. After shifting, $C_r(t)$ still has to be in the sampling range. Therefore, the sampling range must be larger than the $\frac{1}{2}T_r$, resulting in the condition

$$R > \frac{T_r}{2}, \tag{9}$$

where R is the acceptable sampling range to detect data correctly, which should be at least 50% of the data period. This range may not be acceptable for some other communication systems, especially for those with small SNR or high BER. In systems with low BER like the wearable system mentioned before, the range is acceptable. The mechanism for detecting when to shift $C_r(t)$ so as to avoid loss of synchronization is described in section 5.

DOI: 10.1002/cta

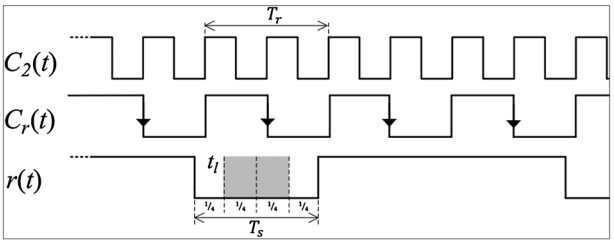

Figure 3. Receiver clocks and incoming signal.

In the absence of jitter, sampling can be performed at any instant inside the data period, and R is upper bounded by T_r. However, the random fluctuations of the incoming signal due to jitter may affect the generated clock $C_r(t)$ and may cause the sampling point to be limited as shown in Figure 4. Combining the upper bound condition and the impact of jitter, the sampling range of eq. (9) must satisfy

$$T_r - 2|J| > R > \frac{T_r}{2}, \tag{10}$$

where J denotes the total jitter of the incoming signal (including both random and deterministic jitter (DJ)). In this case, eq. (8) becomes

$$\frac{-\pi}{2} - \varphi_J < \Delta\varphi(t) < \frac{\pi}{2} + \varphi_J, \tag{11}$$

where φ_J is maximum phase noise caused by jitter. For data to be correctly detected, any jitter must occur outside the sampling range.

Because any shifting of the incoming signal by jitter should not alter the sampling range, jitter should be less than $\frac{1}{2}(T_r - R)$. However, jitter also shifts $C_r(t)$. To take account of the effect of jitter on $C_r(t)$, the overall effect of jitter on the sampling range must be multiplied by two. Therefore, correct operation implies that the jitter must satisfy

$$|J| < \frac{1}{2}\left(\frac{1}{2}(T_r - R)\right). \tag{12}$$

Combining eqns (9) and (12) gives

$$|J| < \frac{1}{8}T_r. \tag{13}$$

4.2.2. System clock frequency. In general, if the clock frequency of the oscillator at the receiver is m times higher than the data rate, then a shift by one clock cycle will correspond to $\frac{1}{m}T_r$. In this case, the phase difference between transmitter and receiver clock is given by

$$\frac{-\pi}{m} - \varphi_J < \Delta\varphi(t) < \frac{\pi}{m} + \varphi_J. \tag{14}$$

Again, the sampling range must obey the condition

$$T_r - 2|J| > R > \frac{T_r}{m}. \tag{15}$$

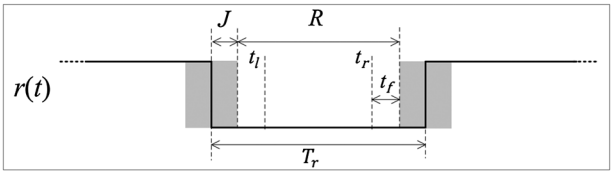

Figure 4. Boundaries of sampling range in the presence of jitter.

DOI: 10.1002/cta

As explained for $m=2$ and from eqns (12) and (15), the jitter must be bound according to

$$|J| < \left(\frac{m-1}{4m}\right) T_r. \tag{16}$$

Figure 5 shows the clock skew between sender and receiver for $m=2$ and $m=3$ for the situation when the sender's clock frequency is a little less than the receiver's ($\Delta f < 0$). The dotted lines show the jitter effect on $C_r(t)$. When $m=2$, the clock skew slowly increases until $\Delta\varphi$ reaches $\pi/2$. Then, the synchronizer shifts the receiver clock to the right, so as to have a phase difference of $-\pi/2$ to the incoming signal. For $m=3$, the range is smaller.

4.2.3. Average of $\Delta\varphi(t)$. For each period of $\Delta\varphi(t)$, the average value of $\Delta\varphi$ can be calculated as the middle point of each line as

$$E[\Delta\tilde{\varphi}] \quad = \frac{1}{2}\left(\frac{\pi}{m} - \frac{\pi}{m}\right) = 0. \tag{17}$$

For any arbitrary period starting from $\varphi 1$, we have

$$E[\Delta\tilde{\varphi}] = \sum_{i=\varphi 1}^{\varphi 1+2\frac{\pi}{m}} \Delta\varphi_i . p(\Delta\varphi_i), \tag{18}$$

where $p(\Delta\varphi_i)$ is the probability of $\Delta\varphi_i$. Considering that after each clock period, the value of $\Delta\varphi$ changes by $2\pi\Delta t\Delta f$ (eq. (5) with $\Delta t = T_s - T_r$), and the conditions from eq. (8), we have

$$p\left(\Delta\tilde{}_{\varphi i}\right) = \frac{\Delta\varphi_i}{2\frac{\pi}{m}} = \mathrm{m}\ \Delta\mathrm{t}\ \Delta\mathrm{f}. \tag{19}$$

Regardless of the receiver and transmitter oscillator frequencies, $\Delta t\Delta f$ is constant, so that $f(\Delta\varphi_i)$ is the same for all points and

$$E[\Delta\tilde{\varphi}] \quad = p(\Delta\varphi)\sum_{i=\varphi 1}^{\varphi 1+2\frac{\pi}{m}} \Delta\varphi_i = 0. \tag{20}$$

Equation (19) shows that the receiver always tries to keep the clock edge around the ideal sampling point. With jitter, the $E\left[\Delta\tilde{\varphi}\right]$ value will be the same, because jitter is due to random processes with zero mean value.

Increasing the receiver system clock frequency decreases the sampling range and increases the accuracy of sampling. However, this also increases energy consumption. For the wearable system where the proposed circuit is used, 50% sampling range is acceptable, so the minimum value of $m=2$ is used for the prototype circuit presented in the next section.

4.2.4. Number of input signal transitions. The CDR mechanism is based on tracking the transitions of the received signal. A long string of 1s, or 0s (with a NRZ line code), or just a long string of 0s (with a NRZI line code) may cause loss of synchronization. To evaluate how long the CDR can be allowed to stay free running for $m=2$, it is necessary to consider the worst case situation after the last transition of the incoming signal: the recovered clock is exactly at the edge of the sampling range (t_r in Figure 4) and will leave that range in the next period. The data are still correctly acquired if the sampling point is inside the safe zone shown in Figure 4. This means that the sampling point may shift right by up to

$$t_f = \frac{T_r}{2} - \frac{R}{2} - |J| < \frac{1}{8}\tau. \tag{21}$$

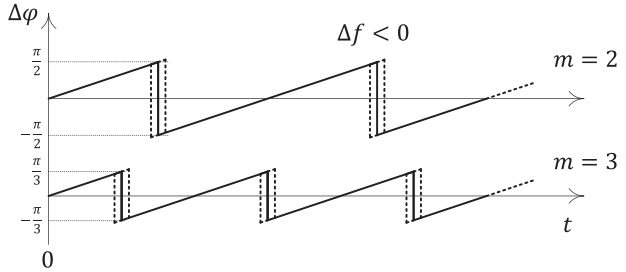

Figure 5. $\Delta\varphi(t)$ for $m=2$ and $m=3$ with $\Delta f < 0$.

DOI: 10.1002/cta

Over N successive clock cycles, the accumulated shift $N|\tau_s - \tau_r|$ must stay below t_f:

$$N|\tau_r - \tau_s| < t_f < \frac{1}{8}\tau \Rightarrow N < \frac{\tau}{8|\tau_r - \tau_s|}. \tag{22}$$

The frequency of oscillators is distributed around the nominal frequency and can be in the range of few parts per million for a crystal quartz oscillator. To obtain the minimum value of N, consider the worst case: $f_r = f(1 + p_c)$ and $f_s = f(1 - p_c)$, where p_c denotes to the frequency tolerance range. For the CDR circuit to work correctly, N must be

$$N < \frac{\frac{1}{f}}{8\left|\frac{1}{f_r} - \frac{1}{f_s}\right|} \Rightarrow N < \frac{1 - p_c^2}{16 p_c}. \tag{23}$$

A typical tolerance value for a quartz oscillator is ± 30 ppm, which leads to $N < 2083$. A well-stabilized resistor–capacitor (RC) oscillator may have a tolerance range of ± 1000 ppm, leading to $N < 6.25$. If the conditions on N are not satisfied by a specific implementation, a line code such as 8b/9b [4] must be used to overcome the effect of long strings of identical symbols. The prototype IC used to obtain the experimental results can be configured to use any of the line codes 8b/9b, 16b/17b, and 64b/65b.

5. THE SYNCHRONIZATION CIRCUIT

This section presents the implementation of a digital circuit that operates on the principles described in the last section. The goal is to have a small size and fully digital circuit that can be completely implemented in an FPGA or digital application-specific integrated circuit (ASIC). In order to keep the frequency of the local clock generator as low as possible, a solution corresponding to $m = 2$ is presented.

To generate a synchronized clock, the circuit has to meet the conditions of eqns (8) and (9). Figure 6 shows the block diagram of the circuit. At the receiver, a stable crystal quartz oscillator generates the system clock $C_2(t)$, whose frequency is twice the data rate and nominally equal to the clock frequency of the transmitter.

An edge detector determines both falling and raising edges of the incoming data signal $r(t)$ and generates the $e(t)$ signal shown in Figure 7. Signal $e(t)$ goes high on any transition of $r(t)$ and goes low at the negative edge of $C_2(t)$. The width of the $e(t)$ pulses can vary from 0 to $\frac{1}{2}T_r$, depending on the phase difference between $r(t)$ and $C_2(t)$.

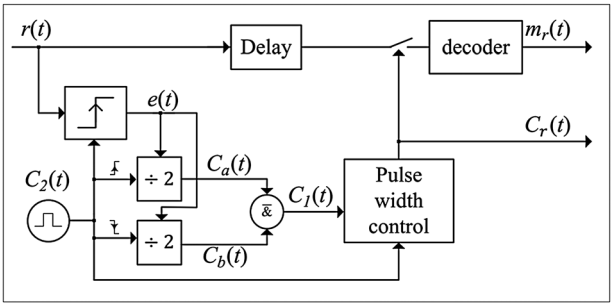

Figure 6. Block diagram of the circuit.

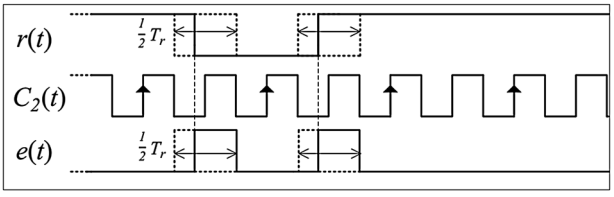

Figure 7. Edge detection signal $e(t)$ generated from $r(t)$ and $C_2(t)$ signals.

DOI: 10.1002/cta

Two clock signals $C_a(t)$ and $C_b(t)$ are generated from $C_2(t)$ by division at the rising and falling edges, respectively. These have the same clock period T_r (equal to the unit interval of the system), but a phase difference of 90. They are generated by D-type flip-flops triggered respectively by the rising and falling edges of $C_2(t)$. The generation of both signals is also controlled by $e(t)$ in order to be synchronized with the incoming $r(t)$: the $e(t)$ pulses are used to asynchronously reset the flip-flop that generates $C_a(t)$ and presets the flip-flop that generates $C_b(t)$. Because the negative edges of $e(t)$ are synchronized with the negative edges of $C_2(t)$, both $C_a(t)$ and $C_b(t)$ will always be synchronized with $e(t)$ as well. This is indicated by the arrows in Figure 8.

Signals $C_a(t)$ and $C_b(t)$ are combined by a negative-and (NAND) gate to produce $C_1(t)$, which is also synchronized with $e(t)$. Given the width of $e(t)$ and the time order of $e(t)$ and $C_1(t)$ in synchronization mode, $e(t)$ always goes high when $C_1(t)$ is high. However, this is not guaranteed at the start of the communication. A rising edge of $e(t)$ when $C_1(t)$ is low would cause $C_1(t)$ to have a low pulse whose width may be too small for correct operation of the system. The pulse width control module in Figure 6 ensures that a low value of $C_1(t)$ is never influenced by $e(t)$, and the duty cycle of the synchronized clock signal never exceeds 75%.

Figure 8 shows that the time difference between the falling edges of $e(t)$ and $C_1(t)$ is always $\frac{1}{4}T_r$. On the other hand, the width of the high pulses of $e(t)$ is a variable between 0 and $\frac{1}{2}T_r$. Therefore, the time difference between $r(t)$ and $C_1(t)$ obeys to the condition

$$r(t)\updownarrow + \frac{T_r}{4} < C_1(t)\downarrow < r(t)\updownarrow + \frac{T_r}{4} + \frac{T_r}{2}, \tag{24}$$

where $r(t)\updownarrow$ is any instant at which $r(t)$ makes a transition and $C_1(t)\downarrow$ is the time for the next falling edge of $C_1(t)$. Assuming that communication starts at $t=0$, then $r(t)\updownarrow = kT_s$, where k is an integer. Considering that $T_s = T_r$, that is, that the oscillator frequencies at sender and receiver are nominally equal. With these conditions, eq. (24) will be

$$kT_r + \frac{T_r}{4} < C_1(t)\downarrow < kT_r + \frac{3}{4}T_r. \tag{25}$$

Signal $C_1(t)$ is periodic, and its falling edges occur in the range defined by eq. (25), from which it follows that the phase difference between $C_1(t)$ and $r(t)$ obeys to the condition

$$\frac{-\pi}{2} < \Delta\varphi(t) < \frac{\pi}{2}, \tag{26}$$

which is the same as the one given by eq. (8). Therefore, the circuit satisfies the synchronization conditions, and $C_1(t)$ (and $C_r(t)$) is a synchronized receiver clock.

Because of the internal delays of the synchronization circuit, changes in $r(t)$ or $C_2(t)$ do not affect $C_r(t)$ immediately, that is, the delays shift the sampling range of the receiver. This effect is small, but in high speed systems (in which the delay may be comparable to the data rate period) it should be compensated by delaying $r(t)$. This is the purpose of module delay in Figure 6. It should be noted that the propagation delay is independent from the data rate or clock frequency. Therefore, the

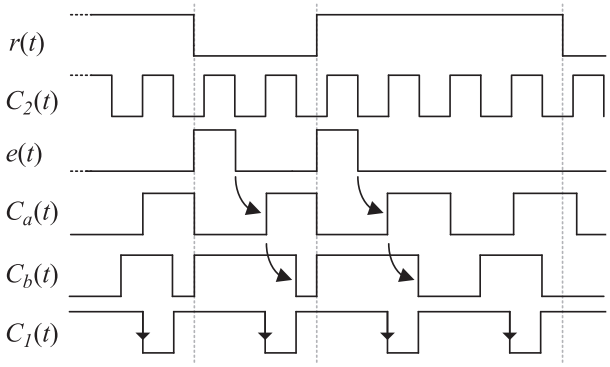

Figure 8. Signals $C_a(t)$, $C_b(t)$, and $C_1(t)$ depend on the phase difference between incoming signal $r(t)$ and local clock $C_2(t)$.

DOI: 10.1002/cta

delay introduced by delay is fixed and only depends on the details of the specific physical implementation. The impact of this delay and the effect of jitter are evaluated experimentally in the next section.

The phase difference between $r(t)$ and $C_2(t)$ is non-deterministic, so three different situations may arise, as analyzed next.

No sampling point adjustment. Figure 8 shows the signals for the case when $C_r(t)$ is in the correct range before a transition occurs in the incoming signal: the edges of $r(t)$ are more than $\frac{1}{4}T_r$ apart from the sampling edges, so there is no need to shift the pulses of $C_r(t)$. Signal $e(t)$ influences the shape of $C_a(t)$ and $C_b(t)$ as shown with dashed lines, but the resulting $C_r(t)$ signal is not affected.

Faster clock at sender. Figure 9 shows the case when $C_r(t)$ is adjusted because an edge of $r(t)$ occurs closely after the sampling edge of $C_r(t)$ (indicated by dashed lines). This case occurs when the receiver clock frequency is less than the transmitter's ($T_s < T_r$). The pulses of $C_r(t)$ are shifted to the left (anticipated in time) by $\frac{1}{2}T_r$, so that the next sampling edge is in the synchronization range.

Shifting $C_r(t)$ to the left is only possible if the duty cycle is bigger than 50. Otherwise, shifting would cause the pulses to overlap, and the receiver would lose one data bit. By using a sampling clock with 75 duty cycles, overlapping never occurs.

Slower clock at sender. The third case is depicted in Figure 10. This case, the receiver clock frequency is higher than the transmitter's ($T_s > T_r$), so the circuit shifts the negative pulses of $C_r(t)$ by $\frac{1}{2}T_r$ to the right, delaying the next pulse.

When communication starts, there is no input signal for a while, so there is no synchronization between transmitter and receiver. Therefore, data must be preceded by a preamble (start bits) to synchronize the receiver. For this architecture, just one transition in the input is enough to synchronize the clock $C_r(t)$. Such a fast settling time is possible because an open-loop architecture is used.

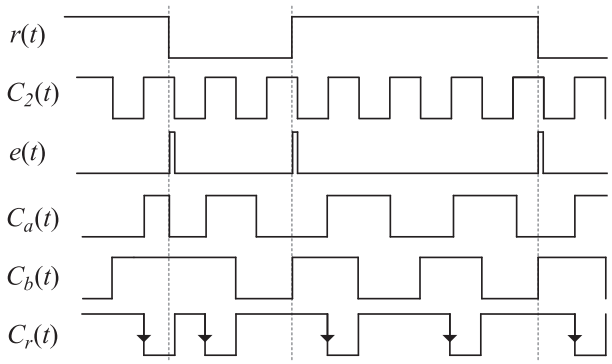

Figure 9. Shifting the negative pulses of $C_r(t)$ to the left (clock at sender is faster than clock at the receiver).

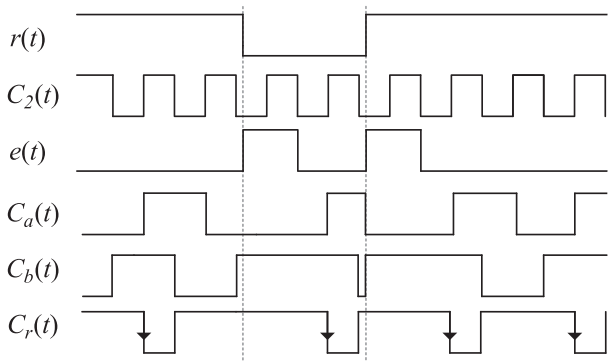

Figure 10. Shifting the negative pulses $C_r(t)$ to the right (clock at sender is slower than clock at the receiver).

DOI: 10.1002/cta

The fact that $C_r(t)$ is synchronized with both the incoming signal and the system clock provides the ability for simultaneous communication with several SNs, a basic need for multitasking operation. For instance, each SN of the prototype IC (section 6) has four independent CDR circuits. Each recovered clock is in synchronization with the received signal (to be in the correct sampling range), and its edges are also inherently in synchronization with the system clock of the receiver. This enables each SN to handle communication with several independent SN without buffering.

The gate-level circuit of Figure 11 implements the block diagram of Figure 6. Flip-flop FF1 and gate exclusive-or (XOR)1 implement the edge detector and create $e(t)$. The clock dividers that produce $C_a(t)$ and $C_b(t)$ are implemented by FF2 (with inverter (INV)1) and by FF3, respectively. An reset/set (RS) flip-flop (active low) is used for pulse width control with $C_2(t)$ and $C_1(t)$ connected to the set and reset inputs of the flip-flop, respectively. The output of the flip-flop ($C_r(t)$ signal) will be low when $C_1(t)$ is low and $C_2(t)$ is high. Regardless of the value of $C_1(t)$, $C_r(t)$ will remain low until $C_2(t)$ changes to low. The gray area is a NRZI decoder and is not a part of the synchronization circuit. For NRZ line encoding, the output of FF5 is used directly, and gates XOR2 and FF6 are not necessary.

The synchronization circuit consists of eight logic gates in total, so it is small enough to have four instances included in the communication ASIC for the aforementioned wearable system. The circuit occupies 0.0022 in a 0.35 CMOS process, which is much less than the area of the circuits listed in [25]; the smallest of which occupies 0.1326 in a 0.18 technology.

6. EXPERIMENTAL RESULTS

This section describes the experimental results obtained with the circuit shown in Figure 11, which corresponds the block diagram of Figure 6. The circuit was implemented both on a low power IGLOO FPGA [33] and as part of a communications IC fabricated in 0.35 CMOS technology for use in a wearable sensor system. The results reported here were obtained with the IC version as is shown in Figure 12 together with the sensor printed circuit board (PCB). For the measurements, two sensors, SN1 and SN2, were connected to each other and set to communicate normally over a one-wire bidirectional link. The clock frequency of both SNs is 16.383. All results shown in Figures 13–18 were obtained with a digital oscilloscope. In all cases discussed next, SN1 is the sender and SN2 is the receiver.

In our prototype, each SN has a microcontroller with a crystal quartz oscillator. The clock pin of the IC is connected to the clock output of the microcontroller (without using any additional clock generator). Many microcontrollers also have their own accurate internal RC oscillator. The accuracy of these RC oscillators is less than the accuracy of crystal oscillators, but the prototype IC is able to

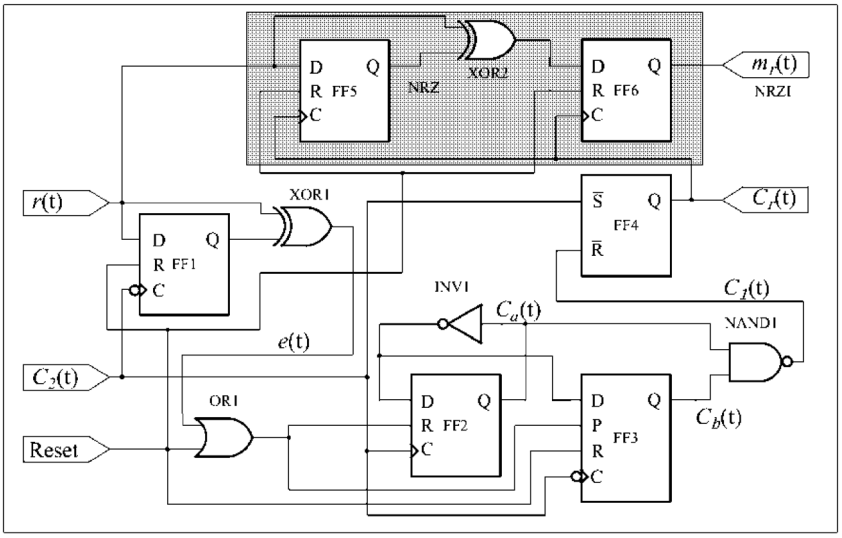

Figure 11. Combined circuit for synchronization and non-return-to-zero inverted (NRZI) decoding. NRZ, non-return-to-zero; FF, flip-flop.

DOI: 10.1002/cta

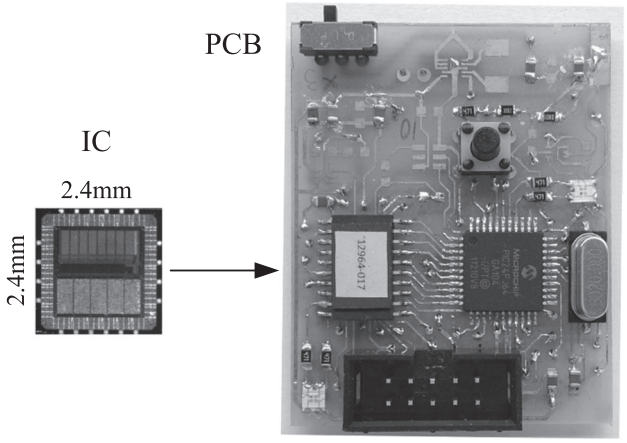

Figure 12. Sensor board with an integrated circuit (IC) including the clock and data recovery (CDR) circuit shown in fig:circuit1.

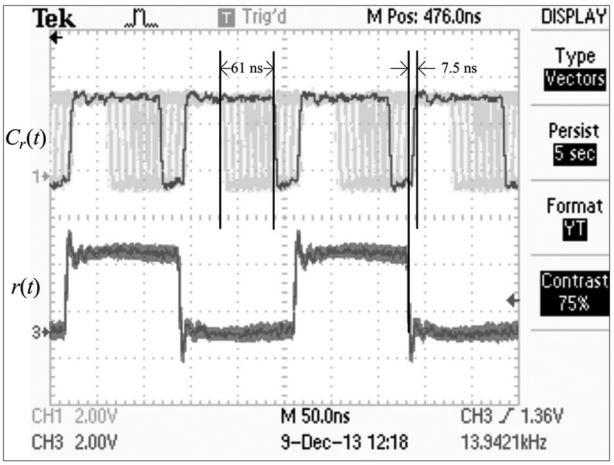

Figure 13. Signals captured using five persistence: L_{12} is the incoming signal on the wire and $C_r(t)$ is the local clock signal. (The edges shown directly to the right of the sampling range are raising edges.)

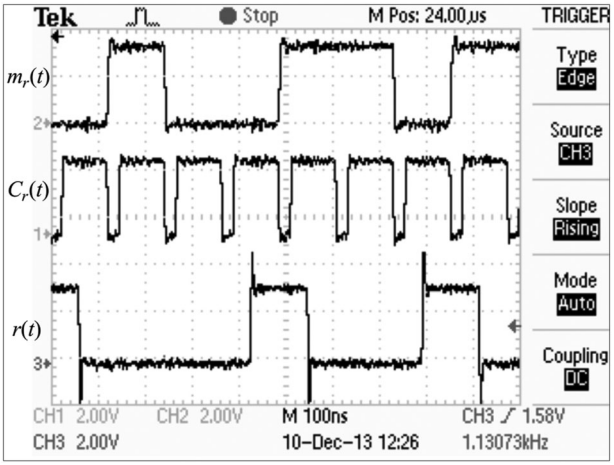

Figure 14. Signals at the receiver when no local clock adjustment is needed: data input $r(t)$, locally synchronized clock $C_r(t)$, and non-return-to-zero inverted (NRZI)-decoded data signal $m_r(t)$.

DOI: 10.1002/cta

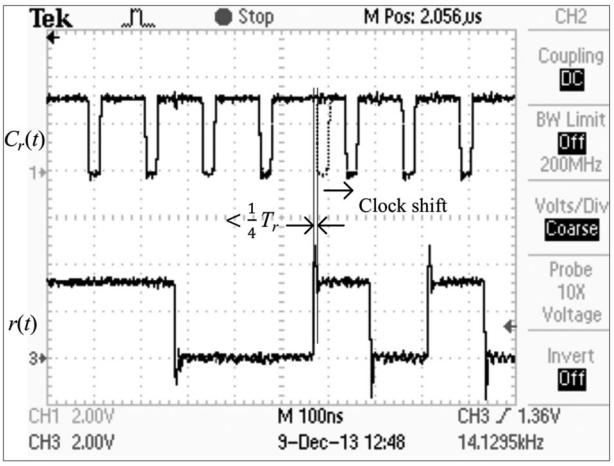

Figure 15. Correction of receiver clock $C_r(t)$ when $T_s > T_r$.

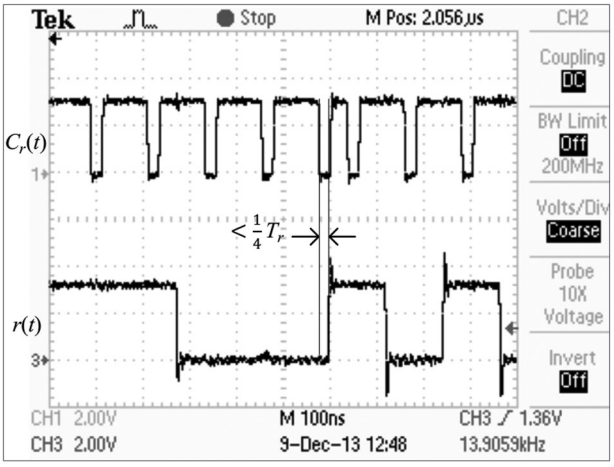

Figure 16. Correction of receiver clock $C_r(t)$ when $T_s < T_r$.

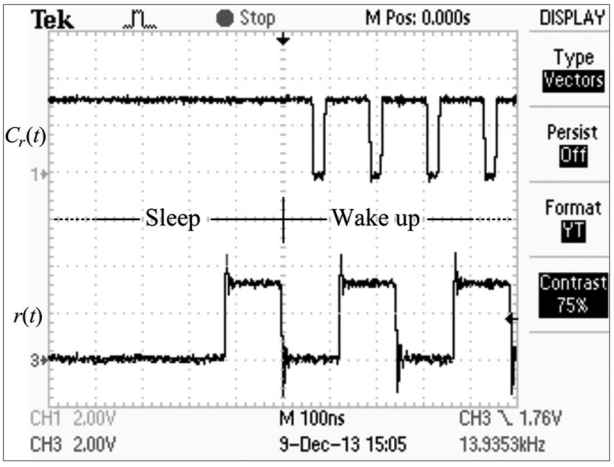

Figure 17. Initiating generation of $C_r(t)$ after exiting from sleep mode.

DOI: 10.1002/cta

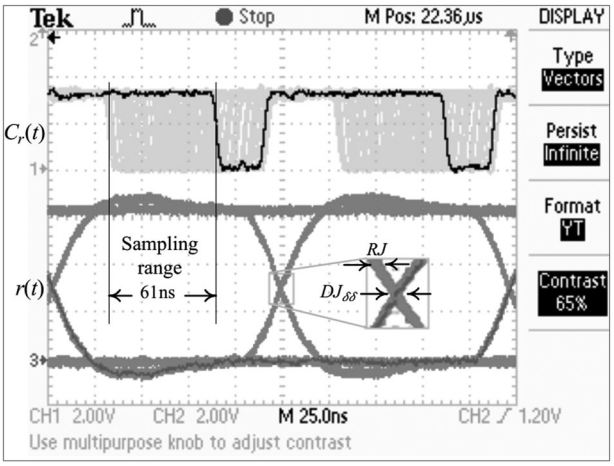

Figure 18. Eye diagram and synchronized clock signal $C_r(t)$ when using four parallel conductive yarns.

work with both of them. In this case, no quartz crystal oscillator would be needed, as long as the total jitter stays below the threshold given by eq. (12). The stability of the oscillator also affects the choice of line coding.

6.1. Normal operation

The oscillators of the SN are running free and generate clock signals with a frequency that is randomly distributed around the nominal value. The difference between sender and receiver clocks causes the phase between the clocks to change over the time as shown in Figure 13. This figure was captured using five-time persistence; the last sample is indicated by a darker line. Figure 13 confirms that the synchronization circuit is keeping the negative edge of $C_r(t)$ in the sampling range, which is 61 for this setup.

This figure also shows that the delay between the incoming signal and $C_r(t)$ is about 7.5. Signal $r(t)$ is captured at the input of the IC and $C_r(t)$ at the output (both in SN2). Therefore, the measured delay d includes the latency inside the IC, which is mainly due to the input and output pads:

$$d \approx d_{ipad} + d_{circuit} + d_{opad}, \tag{27}$$

where d_{ipad} is the delay of the input pad to the core, d_{opad} is the delay from the core to the output pad, and $d_{circuit}$ is the delay due to signal propagation in the synchronization circuit. Datasheet information for the pad cells indicates that $d_{ipad} + d_{opad} \approx 5$, so that $d_{circuit} \approx 2.5$.

The delay $d_{circuit}$ causes a time skew of sampling point, because it decreases the safe zone where jitter can occur without problems and increase the BER. However, in comparison with the data rate period (61), $d_{circuit}$ is small and its effect on synchronization is typically negligible. If necessary, the effect of $d_{circuit}$ is deterministic and can be compensated by delaying $r(t)$ (using module delay of Figure 6).

The following experimental results show the operation of the synchronization circuit according to the phase difference between the $r(t)$ and $C_2(t)$ signals, as discussed in section 5.

No sampling point adjustment. Normal operation of the circuit is depicted in Figure 14, which shows signals $r(t)$, $C_r(t)$, and $m_r(t)$ during data transmission over a copper wire with a data rate of 8.1 Mbps. Signal $C_r(t)$ is the synchronized clock signal in SN2, and $m_r(t)$ is the received message after NRZI decoding. In this case, $C_r(t)$ is a periodic signal, because it is always in the correct sampling range. Samples of $r(t)$ are taken at the negative edges of $C_r(t)$, and the decoded signal appears at the output almost immediately. Other parts of the receiver would typically use $m_r(t)$ at the positive edge of $C_r(t)$ for further processing.

Slower clock at sender. Figure 15 illustrates circuit operation when the clock frequency of SN2 is higher than the frequency of SN1. The circuit modifies $C_2(t)$ to be in the correct sampling range by shifting the signal one $\frac{1}{2}T_r$ to the right. The gray pulse of signal $C_r(t)$ in Figure 15 was inserted to highlight the place where a pulse would occur if no correction had been performed. As can be seen in the figure, the time difference between edge of $r(t)$ and the falling edge of $C_r(t)$ would be less

DOI: 10.1002/cta

than $\frac{1}{4}T_r$. So, the circuit shifts $C_r(t)$ to the right to be in correct sampling range. Afterwards, $C_r(t)$ repeats periodically until another correction is necessary.

Faster clock at sender. Figure 16 corresponds to the situation in which the clock frequency of SN2 is less than the frequency of SN1. As shown in the figure, the circuit shifts $C_r(t)$ one $\frac{1}{2}T_r$ to the left to be in the correct sampling range, whenever the time difference between the falling edge of $C_r(t)$ and the next edge of $r(t)$ is less than $\frac{1}{4}T_r$.

6.2. From sleep mode to active mode

This synchronization circuit is a part of a system that includes a mechanism to monitor communication activity and put the system in sleep mode when possible, so as to reduce energy consumption. In this situation, $C_r(t)$ remains high, as shown in Figure 17. When a signal appears at the input, the system clock control module enables the clock and puts the system in active mode. The synchronization circuit correctly starts generating $C_r(t)$ as shown in Figure 17: $C_r(t)$ falls at the correct time to sample the second incoming bit. The first bit is defined to be a start bit that is used to determine the start of communication; the following data bits are correctly received even immediately after waking up the system. Both in active or sleep mode, one preamble bit is enough for synchronization circuit to work properly.

6.3. Bit error rate

The BER was measured for three different types of interconnection: (1) copper wire; (2) single conductive yarn in relaxed mode; and (3) four parallel conductive yarns. All connections are one long. A good equivalent circuit for a conductive yarn is composed by a resistor and an inductor in series [10], whose values change as the yarn is stretched. For the conductive yarns used in the measurements, the parameters in relaxed mode are $R = 1.56$ and $L = 8$.

Figure 18 shows the eye diagram of the signals obtained when using four parallel conductive yarns for connecting the nodes, which is the normal situation in the target wearable system, because a single yarn is often too fragile to withstand the wear. The diagram was generated by sending randomly varying data ($2^{64} - 1$) and using the oscilloscope's infinite persistence feature. The diagram shows that the receiver clock is always in the right range of sampling.

Figure 19 shows the measured BER as a function of the unit interval. The BER was calculated by using the dual-Dirac method that determines BER as a function of random jitter (RJ) due to noise from different sources (with a zero-mean Gaussian distribution) and DJ due to signal losses, duty-cycle distortion, crosstalk, and other causes [35]. Table I lists the measured peak-to-peak value of RJ and $\mathrm{DJ}_{\delta\delta}$. As Figure 19 shows the copper wire has lower BER than conductive yarns, the probability of error increases with increasing the resistance and inductance of the yarn, because of the increasing signal attenuation. This figure also shows that if the clock synchronization circuit keeps the receiver clock in the sampling range, then the data will be detected correctly with high probability.

Random jitter is mainly due to additive white noise superposed on the received signal. In systems with low SNR such as wireless communication, the random part may be significant and may increase the total jitter noticeably. Therefore, in those systems, the total jitter may be larger than the limit imposed by eq. (12). The relation between BER and SNR depends on the communication method. In the absence of DJ, the conventional analysis of baseband binary signals shows that BER due to AWGN with variance 2 at the input is given by

$$BER = Q\left(\sqrt{\frac{S}{2N}}\right), \qquad S = \frac{A_0^2 + A_1^2}{2}, \qquad N = \sigma^2, \tag{28}$$

where Q denotes the error probability density function, A_0 and A_1 the signal levels, S the signal power, and N the noise power [1]. However, DJ can be more significant than RJ even if the SNR is high. The total BER is the result of both SNR (appears as RJ) and DJ. Regardless of the communication method, the BER can be calculated by measuring the jitter values and using the Dual–Dirac method [35].

6.4. Comparison with other synchronization circuits

Many CDR circuits have been introduced in the last years. Table II summarily compares the present implementation to some other recent circuits. The maximum measured data rate for this

DOI: 10.1002/cta

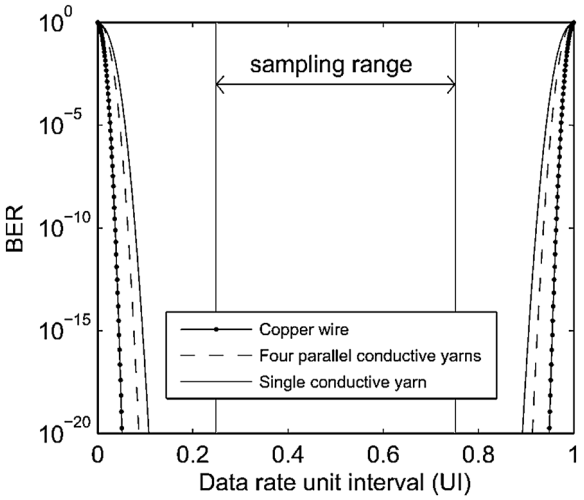

Figure 19. Measured BER and safe sampling range for three different interconnections.

Table I. The values of jitters.

Conductor	Random jitter (RJ) (ns)	$DJ_{\delta\delta}$ (ns)
Copper wire	1	1.5
Four parallel conductive yarns	1.6	2.5
Single conductive yarn	2	3

Table II. Comparison with other works.

Clock and data recovery (CDR) circuit –	Open-loop –	Process (μm)	Frequency (Mbps)	Clock jitter (pk-pk) (ps)	Settling time (bits/time (μs))	Power (mW)	Supply voltage (V)	Core area (mm^2)
This work	*y*	0.35	35	7.14×10^{3}	$1/28.56\times10^{-3}$	0.051	3.3	0.0022
[32]	*y*	0.065	100	242	N/A	0.36	1.2	0.07
[8]	*y*	0.18	180–720	N/A	$1/<5.56\times10^{-3}$	8.2	1.8	0.185
[25]	*n*	0.18	100	199.66	400/4	6.649	1.8	0.16
[26]	*n*	0.18	5120–6400	2.12	N/A	136	1.8	0.8
[16]	*n*	0.18	662–3125	62.2	>331000/>500	60	1.8	0.1326
[27]	*n*	0.18	2500	88	4862/3.89	N/A	1.8	0.133
[17]	*n*	0.18	155.52–3125	467	>15552/100	95	1.8	0.88
[18]	*n*	0.35	200–2000	120	>60000/>30	170	3.3	0.4

N/A, not available.

work is 35 ($T_r = \frac{1}{35\times10^{6}\text{Hz}=28.57\ \text{ns}}$). The settling time is one data period. The total jitter must be less than $\frac{1}{8}T_r$. Therefore, the peak-to-peak jitter must be less than $\frac{2}{8}T_r = 7.14$ ns.

Considering that the aim of the proposed CDR circuit is to minimize power consumption and area, a quick look at the table leads to the conclusion that this was readily achieved. Even though a 0.35 CMOS technology, with 3.3 power supply, is used, the present work attains the smallest area and power consumption of all the works listed in the table. It is noticeable that the low value of power consumption is the consequence of a small-size circuit (fewer active components), but also due to a lower clock frequency operation, a factor that in general sets the average consumption during logic switching, which is the biggest slice of consumed energy. However, if a simple relative frequency

DOI: 10.1002/cta

scaling is applied to the power consumption in order to predict how much power the circuit would consume at a higher clock rate, the estimated power is still substantially smaller than any of the other circuit counterparts. Although this is a weak estimate to be utilized for a direct comparison, it does suggest that in fact the proposed CDR architecture is very power efficient. In addition, the settling time just takes a single period of the data rate, which is also faster than any of the other closed-loop circuits in the table.

These characteristics are obtained at the cost of clock jitter, which is the highest present in the table. This is mainly due to the technique used for synchronization, which always adjusts the sample clock by $\pm\frac{1}{2}m$ of the data rate. Increasing m would increase the accuracy of sampling and the acceptable jitter range as well, but, for the same data rate, would require increasing the system clock frequency, which would also increase the power consumption. Nevertheless, as expected from analysis of the circuit and the results presented in this section, as long as signal jitter is less than 25% of the data rate as happens with our target application, the circuit with $m=2$ is able to correctly recover the data.

7. CONCLUSION

A very small, fully digital open-loop CDR method for use in wearable systems has been described. The receiver clock comes from a local, autonomous oscillator that is running at a frequency of twice the data rate and synchronization is based on open-loop selection of the correct phase of the clock in receiver synchronously with incoming signal.

The achieved performance represents a trade-off between closed-loop and open-loop methods. The relatively higher jitter of the recovered clock is compensated by several favorable characteristics. The circuit size is much smaller than previously reported implementations, because it only consists of eight logic elements and occupies $0.0022\,\text{mm}^2$ in a 0.35-μm CMOS process. The circuit requires only one transition in the input for the alignment of the receiver clock in the correct sampling range. Therefore, it is faster than any closed-loop method. By increasing the system clock frequency, the synchronization performance of the circuit would improve, but the power consumption would also increase. The proposed circuit works with the minimum system clock frequency of twice of data rate. The experimental results obtained with a prototype ASIC show that the circuit can be successfully used in a wearable sensor system, as in this case, the peak-to-peak value of the jitter does not affect the sampling range and the circuit recovers the data with high BER.

ACKNOWLEDGEMENTS

This work is financed by the ERDF – European Regional Development Fund –through the COMPETE Programme (operational program for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) – within project PROLIMB PTDC/EEA-ELC/103683/2008 and through PhD grant SFRH/BD/75324/201.

REFERENCES

1. Gibson JD. *The Communications Handbook* (2nd edn). CRC Press LLC: Southern Methodist University Dallas, Texas, 2002.
2. Hamkins J, Simon MK. *Autonomous Software-Defined Radio Receivers for Deep Space Applications*. Wiley-Interscience: California Institute of Technology, 2006.
3. Meyr H, Moeneclaey M, Fechtel SA. *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. John Wiley and Sons, Inc: New York, 1998.
4. Madhow U. *Fundamentals of Digital Communication*. Cambridge University Press: Cambridge, 2008.
5. Tripathy MC, Mondal D, Biswas K, Sen S. Design and performance study of phase-locked loop using fractional-order loop filter. *Int. J. Circ. Theor. Appl.* 2014; 1–17.
6. Zicari P, Corsonello P, Perri S. A high flexible early-late gate bit synchronizer in FPGA-based software defined radios. In *4th European Conf. on Circuits and Systems for Communications* (*ECCSC2008*), 2008; 252–255.
7. Hsieh M, Sobelman G. Architectures for multi-gigabit wire-linked clock and data recovery. *IEEE Journal on Circuits and Systems Magazine* 2008; **8**(4):45–57.
8. Park SH, Choi KH, Shin JB, Sim JY, Park HJ. A single-data-bit blind oversampling data-recovery circuit with an add-drop FIFO for USB 2.0 high-speed interface. *IEEE Journal on Circuits and Systems II* 2008; **55**(2):156–160.

DOI: 10.1002/cta

9. Fardin Derogarian, Ruben Dias, Ferreira JC, Vítor M.Tavares G. Using a wired body area network for locomotion data acquisition. In *27th Conf. on Design of Circuits and Integrated Systems* (*DCIS2012*), 2012.
10. Zambrano A, Derogarian F, Dias R, Abreu MJ, Catarino A, Rocha AM, da Silva JM, Ferreira JC, Vítor M, Tavares G, Correia MV. A wearable sensor network for human locomotion data capture. In *9th Intl. Conf. on Wearable Micro and Nano Technologies for Personalized Health*, 2012.
11. Mengali U, D'Andrea AN. *Synchronization Techniques for Digital Receivers*. Plenum Press: New York, 1997.
12. Barletta L, Disaro R, Magarini M, Spalvieri A. Post-filter optimization in timing recovery based on square-law detection. *IEEE Journal on Photonics Technology Letters* 2013; **25**(9):821–824.
13. Bregni S. *Synchronization of Digital Telecommunications Networks*. John Wiley and Sons, Ltd: Chichester, West Sussex, UK, 2002.
14. Kim S, Woo JK, Shin WY, Hong GM, Lee H, Lee H, Kim S. A low-power referenceless clock and data recovery circuit with clock-edge modulation for biomedical sensor applications. In *Intl. Symposium on Low Power Electronics and Design* (*ISLPED*) 2011; 347–350.
15. Bushnaq S, Nakura T, Ikeda M, Asada K. All digital baseband 50Mbps data recovery using 5x oversampling with 0.9 data unit interval clock jitter tolerance. In *12th Intl. Symposium on Design and Diagnostics of Electronic Circuits Systems* (*DDECS*) 2009; 206–209.
16. Lin SH, Liu SI. Full-rate bang-bang phase/frequency detectors for unilateral continuous-rate CDRs. *IEEE Journal on Circuits and Systems II: Express Briefs* 2008; **55**(12):1214–1218.
17. Yang RJ, Chao KH, Hwu SC, Liang CK, Liu SI. A 155.52 Mbps–3.125 Gbps continuous-rate clock and data recovery circuit. *IEEE Journal on Solid-State Circuits* 2006; **41**(6):1380–1390.
18. Yang RJ, Chao KH, Liu SI. A 200 Mbps–2 Gbps continuous-rate clock and data recovery circuit. *IEEE Journal on Circuits and Systems I: Regular Papers* 2006; **53**(4):842–847.
19. Kao SK, Hsueh SH. *A Fast-Corrected All-Digital DCC with Synchronous Input Clock*. Int. J. Circ. Theor. Appl., 2014; 1–16.
20. Huang K, Wang Z, Zheng X, Ma X, Kunzhi Y, Z Chun, Wang Z. A novel clock and data recovery scheme for 10 Gbps source synchronous receiver in 65 nm CMOS. In *55*th IEEE Midwest Symposium on Circuits and Systems (MWSCAS) 2012; 932–935.
21. Chen Z, Zhang Y. A modified synchronization scheme for impulse-based UWB. In *6th Intl. Conf. on Information, Communications Signal Processing*, 2007; 1–5.
22. Ashari ZM, Nordin AN, Ibrahimy MI. Design of a 5 GHz phase-locked loop. In *IEEE Regional Symposium on Micro and Nanoelectronics* (*RSM*), 2011; 167–171.
23. Kim YS, Lee SK, Park HJ, Sim JY. A 110 MHz to 1.4 GHz locking 40-phase all-digital DLL. *IEEE Journal of Solid-State Circuits* 2011; **46**(2):435–444.
24. Cheng-chang Z, Dan-gui Y, Li-sheng Y, Qi HL, Li Cy. DLL-based multi-FPGA systems clock synchronization. In *5th IEEE Conf. on Industrial Electronics and Applications* (*ICIEA*), 2010; 1420–1423.
25. Chen CL, Wang CC, Juan CY. A fast-locking clock and data recovery circuit with a lock detector loop. In *13th Intl. Symposium on Integrated Circuits* (*ISIC*), 2011; 332–335.
26. Chen FT, Wu JM. An extended phase detector 2.56/3.2 Gb/s clock and data recovery design with digitally assisted lock detector. In *IEEE Intl. Symposium on Circuits and Systems* (*ISCAS*), 2009; 1831–1834.
27. Woo JK, Lee H, Shin WY, Song H, Jeong DK, Kim S. A fast-locking CDR circuit with an autonomously reconfigurable charge pump and loop filter. In *IEEE Asian Conf. on Solid-State Circuits* (*ASSCC*), 2006; 411–414.
28. Gholami M, Rahimpour H, Ardeshir G, Miar-Naimi H. A new fast-lock, low-jitter, and all-digital frequency synthesizer for DVB-T receivers. *Int. J. Circ. Theor. Appl.* 2013:13.
29. Zicari P, Corsonello P, Perri S. A high flexible early-late gate bit synchronizer in FPGA-based software defined radios. In *4th European Conf. on Circuits and Systems for Communications* (*ECCSC*), 2008; 252–255.
30. Shachi P, Mishra R, Jatoth RK. Coherent BPSK demodulator using Costas loop and early-late gate synchronizer. In *4th Intl. Conf. on Computing, Communications and Networking Technologies* (*ICCCNT*), 2013; 1–6.
31. Jayasimha S, Jyothendar P, Satish Bhargav B. Low-complexity DFT-pair carrier acquisition. In *National Conf. on Communications* (*NCC*), 2013; 1–4.
32. Kulkarni VV, Lee Jung Hyup, Liu X, Je M. A 100Mbps 0.36 mW injection locked clock and data recovery circuit for WBAN transceivers. In *IEEE Intl. Conf. on Microwave Workshop Series on RF and Wireless Technologies for Biomedical and Healthcare Applications* (*IMWS-BIO*), 2013; 1–3.
33. IGLOO nano low power flash FPGAs, 2013; Revision 18.
34. Sullivan DB, Allan DW, Howe DA, Walls FL. *Characterization of Clocks and Oscillators* (1st edn). US Department of Commerce, National Institute of Standards and Technology (NIST): Washington DC, 1990.
35. Stephens R. *Jitter Analysis: The Dual-Dirac Model, RJ/DJ, and Q-Scale*. Agilent Technologies, Whitepaper, 2004; 1–16.

DOI: 10.1002/cta