

Exploiting Manufacturing Variations for Compensating Environment-induced Clock Drift in Time Synchronization *

Thomas Schmid, Zainul Charbiwala, Jonathan Friedman, Young H. Cho, Mani B. Srivastava
Dept. of Electrical Engineering
University of California, Los Angeles
{schmid, zainul, jf, young, mbs}@ee.ucla.edu

ABSTRACT

Time synchronization is an essential service in distributed computing and control systems. It is used to enable tasks such as synchronized data sampling and accurate time-of-flight estimation, which can be used to locate nodes. The deviation in nodes' knowledge of time and inter-node resynchronization rate are affected by three sources of time stamping errors: network wireless communication delays, platform hardware and software delays, and environment-dependent frequency drift characteristics of the clock source. The focus of this work is on the last source of error, the clock source, which becomes a bottleneck when either required time accuracy or available energy budget and bandwidth (and thus feasible resynchronization rate) are too stringent. Traditionally, this has required the use of expensive clock sources (such as temperature compensation using precise sensors and calibration models) that are not cost-effective in low-end wireless sensor nodes. Since the frequency of a crystal is a product of manufacturing and environmental parameters, we describe an approach that exploits the subtle manufacturing variation between a pair of inexpensive oscillators placed in close proximity to algorithmically compensate for the drift produced by the environment. The algorithm effectively uses the oscillators themselves as a sensor that can detect changes in frequency caused by a variety of environmental factors. We analyze the performance of our approach using behavioral models of crystal oscillators in our algorithm simulation. Then we apply the algorithm to an actual temperature dataset collected at the James

*This material is supported in part by the U.S. ARL and the U.K. MOD under Agreement Number W911NF-06-3-0001, by the NSF under award CNS-0614853, and by the Center for Embedded Networked Sensing at UCLA. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the listed funding agencies. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMETRICS'08, June 2-6, 2008, Annapolis, Maryland, USA
Copyright 2008 ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

Wildlife Reserve in Riverside County, California, and test the algorithms on a waveform generator based testbed. The result of our experiments show that the technique can effectively improve the frequency stability of an inexpensive uncompensated crystal 5 times with the potential for even higher gains in future implementations.

Categories and Subject Descriptors

B.m [Hardware]: Miscellaneous

General Terms

Experimentation, Measurement

Keywords

Time Synchronization, Oscillator, Clocks, Emulation

1. INTRODUCTION

Distributed computing systems use time synchronization to perform a variety of tasks. These time synchronization services can be generalized into three different modes of operation:

- Operation to determine the relative time of one node to another. E.g., for time-of-flight measurement
- Operation for an individual node to obtain the absolute global time. E.g., for coordinated data collection
- Operation to match the frequency and the phase of all the nodes. E.g., for improving communication channel capacity

One convenient way to obtain a very accurate notion of time is through the use of the global positioning system (GPS) infrastructure. The GPS satellite system maintains accurate time information by using on-board atomic clocks and tight time synchronization between the satellites. As the satellites orbit the earth, they consistently send out timing signals that allow the synchronization of GPS receivers on the surface of the Earth.

However, the timing signals from the GPS infrastructure are not always available. For instance, radio signals from the satellites may not be able to reach receivers within or between buildings. Applications with tight energy budgets may not afford the relatively high power requirement of GPS time synchronization. For these reasons, alternatives for distributed systems have emerged.

Each protocol has different goals and characteristics. One of the most widely used time synchronization protocols in the Internet is NTP [18]). The protocol uses the computational resources of a general purpose microprocessor to execute complex time synchronization algorithms over a network to achieve an accuracy on the order of milliseconds. IEEE 1588-2002 is another alternative which proffers down to sub-microsecond accuracy under controlled conditions [5].

Due to a unique set of performance and cost constraints that are associated with wireless sensor nodes, there has been strong interest in time synchronization research by the sensor network community. The Flooding Time Synchronization Protocol (FTSP, [16]), Reference Broadcast Synchronization (RBS, [6]), Timing-sync Protocol for Sensor Networks (TPSN, [9]), and Rate Adaptive Time Synchronization (RATS, [8]) are just a few of the many new time synchronization protocols for sensor networks.

All of these protocols have one thing in common. They use a local clock to timestamp messages and then employ statistics and latency estimations to improve the expected offsets and local clock drifts. While the platform design directly impacts the total accuracy that can be achieved by a time synchronization protocol [12, 16], the limiting factor in the precision of these protocols is the stability of the local clock source. This has far reaching implications. Dutta et al. [4] demonstrated that node lifetime is bounded by the local clock stability. Through experimentation they show that a radio duty cycle of only 0.002% is required for their 1-hop network. However, their ± 50 parts per million local clock stability compromised this minimum requiring their radio to maintain a 0.01% duty cycle (5 times longer).

The focus of this paper is on this key source of timing error – the local clock (in)stability. Traditionally, this has required the use of expensive clock sources that are not cost or energy effective in low-end wireless sensor nodes. However, since the frequency of a crystal is a product of manufacturing and environmental parameters, we propose an approach that exploits the subtle manufacturing variation between a pair of inexpensive oscillators placed in close proximity to algorithmically compensate for the drift produced by the environment. The algorithm effectively uses the oscillators themselves as a sensor that can detect changes in frequency caused by a variety of environmental factors – a concept that we call the software compensated crystal oscillator (SCXO).

In Section 2 we discuss the characteristics of clock sources. In Section 2.1 we introduce differential drift. In Section 3 we introduce the SCXO. In Section 3.4 we analyze the performance of our approach and demonstrate it on an experimental testbed in Section 4. In Section 5 we discuss the implications if better local clocks are used (5.1), future hardware design ideas for a crystal compensated crystal oscillator (XCXO) (5.3), and a cost versus performance estimation of our implementation (5.4).

2. CLOCK SOURCE CHARACTERISTICS

Timestamping is usually performed by reading a hardware counter that is periodically updated by some system clock. The system clock runs at a rate of, say, f Hz, incrementing the counter every $1/f$ sec. Therefore, at any time t since the n -bit counter was reset, it would read a count $c(t) = \lfloor f \cdot t \rfloor \bmod 2^n$. The floor operator $\lfloor \cdot \rfloor$ comes into play due to the digital nature of the hardware counter, which limits the precision with which one can measure time to $1/f$ sec. This

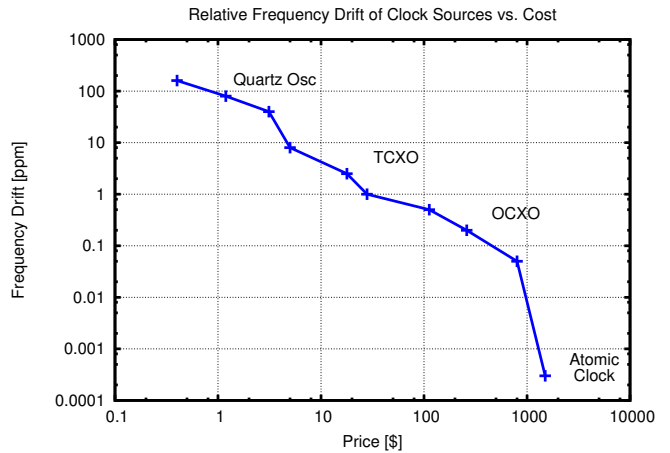


Figure 1: Relative Frequency Drift vs Cost of commercially available clock sources. Data gathered from different vendors in Fall 2007.

error is viewed as a temporal quantization error. The range of time that can be measured is, of course, bounded by the number of bits, n , of the counter. Most importantly though, the accuracy with which we can measure time depends on how accurately we know the rate f at which the counter is incremented. Unfortunately, all clock implementations have some deviation from the nominal clock frequency F_0 Hz that they are designed with.

The normalized deviation in frequency of a clock source from its nominal is termed its relative frequency drift or just *drift*¹, denoted as δf , and is measured in a unitless quantity called *ppm* or Parts Per Million computed as follows:

$$\delta f = \frac{f - F_0}{F_0} \times 10^6 \quad (1)$$

The multiplication by 10^6 is a measure of convenience since the drift of most commercially available clock sources range from few tens of *ppm* to less than a few hundredths. Figure 1 shows how the stability of a clock source is related to its price. The x -axis could represent other cost parameters as well, such as power consumption and/or size without appreciable change. To give a perspective on the values of drift, a 10ppm clock source would introduce a measurement error of over 5 minutes over a year. A 0.1ppm clock source on the other hand would be off by only about 3 seconds over a year.

High accuracy clock sources mostly employ the native oscillatory properties of quartz crystals. Due to a variety of factors, however, the natural resonance frequency of the crystal deviates from the nominal frequency causing drift. The three major factors affecting drift are manufacturing imprecision, temperature and aging. Other factors [10] include effects of humidity and pressure, acceleration effects (shock and gravity), effects due to electric and magnetic fields and particle radiation. In this paper we focus only on effects due to the first two. The effects of humidity and pressure are effectively reduced by hermetically sealing the quartz crystal in the oscillator package. The other effects (excluding aging) are minor and do not play a significant

¹Some researchers refer to this quantity as skew. Our definition follows the IEEE recommended Mil Specs standard [17].

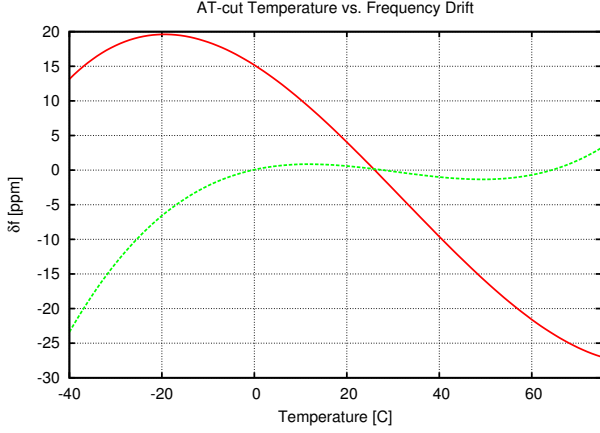


Figure 2: Frequency Drift vs Temperature for multiple AT-cut crystals. The crystals are cut at an angle of $35^{\circ}20' + \theta$

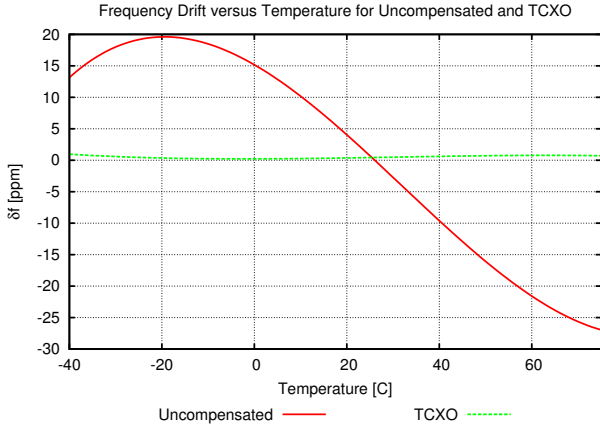


Figure 3: Frequency Drift vs Temperature for an uncompensated AT-cut quartz crystal oscillator and temperature compensated crystal oscillator (TCXO).

role at the drift levels we are concerned with. The effect of aging [7] is small compared to that of manufacturing imprecision and temperature, but becomes pronounced once these effects are compensated. Effects of aging is expected to be taken into account in subsequent research. The frequency drift of a quartz crystal based oscillator (due to the two factors) could be modeled as:

$$\delta f = \delta f_{tolerance} + \delta f_{stability}(T) \quad (2)$$

where $\delta f_{tolerance}$ is a constant drift in the range of 20-50ppm due to imprecision in the geometric cut of the crystal in the manufacturing process. $\delta f_{stability}$ is a non-linear temperature dependent factor that is usually in the range of 10-20ppm. Manufacturers of quartz crystal resonators and oscillators specify ranges for these values in datasheets as frequency tolerance and frequency stability respectively.

To produce a quartz resonator, manufacturers cut out a tiny sheet from a quartz crystal at a specific shear angle. Different angles of cut produce vastly different crystal characteristics. The most popular type is called AT-cut, which is cut at a nominal angle of $35^{\circ}20'$, and accounts for up to 75% of all quartz resonators made due to its excellent frequency-temperature ($f - T$) characteristics. The $f - T$ characteristics of AT-cut crystals is well studied in the liter-

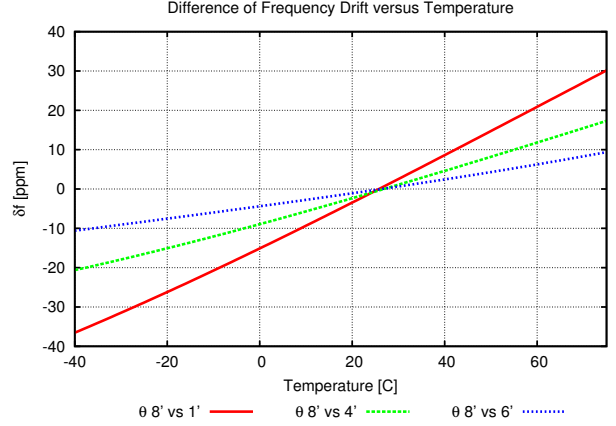


Figure 4: Differential Frequency Drift vs Temperature for multiple pairs of differently AT-cut oscillators. Note that the steeper the slope, the better it is for our compensation algorithm.

ature and is found to follow a third order polynomial using:

$$\delta f_{stability}(T) = A(T - T_0)^3 + B(T - T_0) + C \quad (3)$$

where A, B, C, T_0 are unique to each device. Interestingly, this is the key observation behind the compensation techniques detailed later, the value of B is extremely sensitive to any imprecision in the angle of the cut itself. Figure 2 shows how the $f - T$ characteristics vary for AT-cut crystals sheared with a slightly different angle of cut. This slight difference could be either deliberate or due to manufacturing variation. The key idea behind improving the stability of the clock source is to exploit this difference in $f - T$ characteristics for two sources to compensate one of them.

Before we describe our software based compensation technique, it is noteworthy to mention the state-of-the-art in oscillator design. For mid-performance applications, like GPS receivers, designers prefer to use a temperature compensated crystal oscillator or TCXO [20, 21, 24]. The approach followed by a TCXO manufacturer is to characterize each device in the factory to obtain its $f - T$ curve [23]. Then, by using a custom analog matching circuit [13] or a digital tuning circuit with a temperature sensor [3, 14, 15], the $f - T$ characteristics are corrected by minuscule frequency adjustments to negate the effects of drift. Figure 3 shows the improvement in the drift performance of a commercially available digital TCXO [1] over an uncompensated oscillator. For high-performance applications, like GSM or CDMA cell phone basestation towers, designers employ an oven controlled crystal oscillator or OCXO which has an active mechanism of maintaining the temperature of the crystal structure, allowing much higher frequency stability over external temperature variations at the cost of an active heating element.

2.1 Introducing Differential Drift

In order to explain how our software based compensation technique works, we first describe the mechanism intuitively. Assume that the system under consideration has two AT-cut quartz crystal oscillators with slightly different shearing angles. For now, we pick the crystals with the top and bottom curves from Figure 2 representing the $35^{\circ}21'$ cut and the $35^{\circ}28'$ cut. We first measure the “difference of drift”

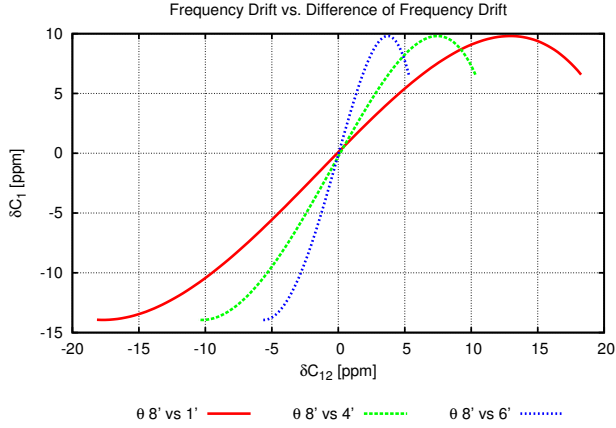


Figure 5: Frequency Drift versus Difference of Frequency Drift for several pairs of AT-cut oscillators. The larger the full span of δf_{12} the better our algorithm can compensate.

between the two oscillators (or differential drift) over the entire temperature range and plot it against temperature, to obtain Figure 4. The reason that Figure 4 is almost a straight line is due to the fact that between the two crystals, it is the B parameter from Equation 3 that dominates. Now, if instead we plot the frequency drift of one of the oscillators against the differential drift, we obtain Figure 5, which is similar to the $f - T$ curve of that oscillator. This leads us to believe that if the system measures the differential drift at run time, it can estimate the relative drift of one of its oscillators. Using this information, the system can make a correction to its oscillator when it deviates and thus gain higher frequency stability.

This approach is analogous to traditional temperature compensation techniques except that there are two significant advantages to compensating using differential drift. On the one hand, temperature sensing is itself error-prone, requiring its own calibration and compensation system to provide an appreciable accuracy in the reading. Further, temperature sensing displays non-linear dynamic behavior causing hysteresis effects during temperature variations. On the other hand, the measurement of differential drift is completely done in digital logic and software, and the accuracy of measurement can be set arbitrarily high. There are no dynamic behaviors that affect the reading and the speed of acquiring a reading scales with the speeds of deep sub-micron process technology. Additionally, the potential saving in hardware and thus production cost is tremendous since the logic circuitry could directly be integrated into microcontrollers or systems on a chip (SoC) at virtually no additional cost. To put it briefly, using simpler hardware with smarter algorithms can benefit in lower production cost and performance advantages, as we will show later on.

3. SOFTWARE COMPENSATED CRYSTAL OSCILLATOR (SCXO)

The software compensated crystal oscillator uses measurements of differential drift introduced in Section 2.1 to compensate for the frequency deviation of the clock source. The following illustrates how this is done. Let the frequency of the two oscillators in the system be denoted as f_1 and f_2 . These frequencies deviate due to various factors from the

nominal oscillation frequency of $F_0 Hz$. The relative frequency drift for each oscillator is given by Equation 1. The differential drift, defined in Section 2.1 and denoted as δf_{12} , is given by $\delta f_{12} = \delta f_1 - \delta f_2$. To measure the differential drift, observe that δf_{12} can be simplified using Equation 1 to:

$$\delta f_{12} = \frac{f_1 - F_0}{F_0} - \frac{f_2 - F_0}{F_0} = \frac{f_1 - f_2}{F_0} \quad (4)$$

Thus, measuring differential drift could be implemented by measuring the difference in frequencies between the oscillators. Measuring frequencies in software is achieved by counting the number of clock pulses within some fixed time interval and dividing by that interval. The interval must be large enough so that the $1/f$ temporal quantization error does not affect the accuracy of the measurement. On the other hand, the interval must remain small to ensure short measurement acquisition times and quick response to dynamic environments. The sampling interval could be derived from a faster clock source in the system by generating a low frequency sampling signal F_s .

The frequency correction algorithm is executed in two parts, a one time calibration phase and a run time compensation phase. The simplest implementation of the algorithm requires the software to have low level access to a hardware timer and two hardware counters. The authors believe it will be feasible to relax this requirement to one counter in future implementations.

3.1 Calibration Phase

In the calibration phase, the system develops an equivalent of the δf_{12} vs. δf_1 characteristics by measurement against a known reference of the sampling clock, F_s . The calibration phase is performed for a pair of oscillators at the factory. This is analogous to the calibration phase performed for TCXOs described earlier, except that the calibration reference required is simply a stable externally applied sampling clock, F_s . The requirement to accurately measure temperatures is eliminated. The calibration phase requires the use of two free running hardware counters C_1 and C_2 that are being fed from the two oscillators with frequencies f_1 and f_2 respectively. The system then captures the values of both counters at the positive edge of F_s via an interrupt service routine and resets them to zero for the next sample. The collection of counter samples is performed over the largest temperature variation possible, ideally over the entire range from -40 to $+85^\circ C$. Let the collection of counter samples, sampled at F_s , be denoted as C_1 and C_2 .

Define two new quantities δC_i and δC_{ij} as follows:

$$\delta C_i = \frac{C_i}{F_0/F_s} - 1 \quad (5)$$

$$\delta C_{ij} = \delta C_i - \delta C_j \quad (6)$$

Before the sampling routine exits, the routine computes δC_1 , δC_2 and δC_{12} and publishes it for subsequent storage. This procedure is shown in the pseudo-code in Procedure 1.

It will now be shown that $\delta f_1 = \delta C_1$, $\delta f_2 = \delta C_2$ and $\delta f_{12} = \delta C_{12}$, so that a model of δC_{12} vs. δC_1 is equivalent to a model of differential drift δf_{12} versus frequency drift δf_1 . The value of the counter in each interval can be given by:

$$C_i = \frac{f_i}{F_s} \quad (7)$$

Procedure 1 Calibration:On_Fs_Interrupt

```
C1 ← Counter1.value
C2 ← Counter2.value
Counter1.reset()
Counter2.reset()
dC1 ← [C1 / (F0/Fs)] - 1
dC2 ← [C2 / (F0/Fs)] - 1
dC12 ← dC1 - dC2
print dC1, dC2, dC12
```

where f_i is the *mean* frequency of the oscillator over the sampling interval. δf_i is given by Equation 1 as:

$$\delta f_i = \frac{f_i - F_0}{F_0} = \frac{f_i}{F_0} - 1 \quad (8)$$

$$= \frac{C_i \cdot F_s}{F_0} - 1 = \frac{C_i}{F_0/F_s} - 1 \quad (9)$$

$$= \delta C_i \quad (10)$$

A similar relationship can be shown for δf_{12} and δC_{12} .

At the end of the calibration phase, the system has collected a set of $\langle \delta C_{12}, \delta C_1 \rangle$ and $\langle \delta C_{12}, \delta C_2 \rangle$ tuples. We found that these tuples fit well to a third order polynomial function because C_{12} is linear proportional to the temperature (see Figure 4). Thus, only the values of A , B , C , and D in the following equation are required to be stored:

$$\delta C_1 = A \cdot (\delta C_{12})^3 + B \cdot (\delta C_{12})^2 + C \cdot \delta C_{12} + D \quad (11)$$

The entire set of tuples can also be stored as a lookup table, subject to memory availability. We show in Section 3.5 how this choice affects the performance of the compensation algorithm.

3.2 Compensation Phase

The compensation phase performs a continuous frequency correction at run time to provide a higher stability clock. Instead of stabilizing f_1 or f_2 using hardware based tuning circuits, we focus on regenerating a replica of the stable sampling clock, F_s from the unstable clock sources. It can be shown that the fact that F_s is a fairly low frequency does not affect the accuracy of the clock as long as it captures dynamic variations in the environment adequately.

The key idea of the compensation algorithm is to estimate the frequency of the sampling clock as closely as possible. Thus, at every pulse of this generated clock, we know that (close to) $1/F_s$ *seconds of real time* has elapsed. This value is accumulated in a register that then provides the real time at the pulse edge. To timestamp an event or read time between two pulses, an intermediary correction is required as described in Section 3.3.

If the timer is fed the clock source f_1 and loaded with a value γ , the timer expires after a time interval γ/f_1 *secs* asserting an interrupt. If the timer reloads itself on expiry with the same value γ , it generates a periodic sampling signal with frequency denoted as f_γ . Again, two counters C_1 and C_2 are used, fed by the two oscillators running at frequencies f_1 and f_2 respectively.

As shown in the initialization routine in Procedure 2, the timer is loaded with a value $\gamma = F_0/F_s$ since this provides the best initial estimate of the sampling signal. On every timer interrupt, a sampling and compensation routine is executed as shown in Procedure 3.

Procedure 2 Compensation:Initialization

```
gamma ← F0/Fs
RealTime ← 0
Timer1.value ← gamma
Timer1.reset()
Counter1.reset()
Counter2.reset()
```

Procedure 3 Compensation:On_Timer_Interrupt

```
//Read and reset counters
C1' ← Counter1.value
Counter1.reset()
C2' ← Counter2.value
Counter2.reset()
// Calculate the normalized difference
dC12' ← (gamma - C2') / (F0/Fs)
//Calculate the correction term
dC1' ← A · (dC12')3 + B · (dC12')2 + C · dC12' + D
gamma ← (F0/Fs) · (1+dC1')
//Increment the corrected counter
RealTime ← RealTime + 1/Fs
//Update the timer
Timer1.value ← gamma
```

First, the values of the two counters are captured and the counters reset to zero for the next sample. Note that since *Counter1* is incremented with the same clock source as the timer, when the interrupt is asserted, *Counter1* would have incremented γ times, i.e., $C_1' = \gamma$. Then, an estimate of δC_{12} is computed using Equation 6 as follows:

$$\delta C_{12}' = \frac{\gamma - C_2'}{F_0/F_s} \quad (12)$$

This is only an estimate of the original δC_{12} since f_γ is not necessarily equal to F_s . Note that since f_γ is generated using f_1 :

$$f_\gamma = \frac{f_1}{\gamma} \quad (13)$$

and $C_1' = \gamma$. Using the values of A , B , C and D from the calibration phase, $\delta C_1'$ is computed as:

$$\delta C_1' = A \cdot (\delta C_{12}')^3 + B \cdot (\delta C_{12}')^2 + C \cdot \delta C_{12}' + D \quad (14)$$

$\delta C_1'$ represents an estimate of the mean relative drift of f_1 with respect to F_0 over the sampling period. Compensating for this drift would require retuning the oscillator such that f_1 is reduced by an amount $-F_0 \cdot \delta C_1'$. Instead, we compensate by continuously adjusting the value of γ such that $f_\gamma \approx F_s$ as follows:

$$\gamma = (1 + \delta C_1') \frac{F_0}{F_s} \quad (15)$$

γ is then loaded into the timer for the next sampling period. By adjusting the value of γ in this way, we create as close a replica of the reference sampling signal as possible, accumulating *real time* epochs in the *RealTime* register as described below.

3.3 Corrected Timestamping

In order to use the above technique to estimate real time, a memory register *RealTime* is used to keep track of the accumulated counts. At every sampling interrupt, this register

is incremented by a fixed value $1/F_s$ as shown in Procedure 3. An accurate estimate of the real-time t is then given by reading $RealTime$ and C_1 and applying an inter-sampling period correction.

$$t = RealTime + \frac{C_1(t)}{\gamma \cdot F_s}$$

where $C_1(t)$ is the captured value of the counter at the instant the timestamp is required. This process is described in Procedure 4.

Procedure 4 Compensation:On_Get_Time

```
// Return RealTime's value and add the corrected
// elapsed time since the last increment
Correction = Counter1.value * (1/Fs) * (1/gamma)
return RealTime + Correction
```

3.4 Performance Evaluation

To evaluate the effects of implementation choices on performance, four oscillator architectures are considered: an uncompensated crystal, a commercial TCXO, our SCXO implementing the cubic fit Equation 11, and our SCXO implemented via a lookup table. Due to the intrinsically low level nature of the compensation technique, we found it illustrative to evaluate its performance through simulation. The simulator models the drift characteristics of the oscillators and estimates their performance based on the implementation of the algorithms described previously. Figure 6 illustrates the $f - T$ characteristics for an oscillator with various schemes of compensation. The $f - T$ characteristics of the TCXO [1] is also plotted to serve as a reference. It is observed that the stability of the SCXO is significantly better than the uncompensated version but shows large variations around the $0ppm$ line – an effect examined below. Note that these characteristics correspond to $F_0 = 1 MHz$ and $F_s = 2 Hz$.

The performance of a particular compensation technique is quantified in *compensation gain*, defined as the ratio of the frequency drift of the oscillator without compensation to the drift with compensation at a specific temperature. A log plot of this compensation gain over temperature for the TCXO and SCXO is shown in Figure 7. The plot reveals that even though there is large variation between the compensation schemes, the mean value of gain over all temperatures for both the TCXO and SCXO is about the same, between 14 and 14.5 dB. The dip in the TCXO curve is a surprising artifact. We attribute this to an uncharacteristically high stability in the uncompensated oscillator at 25°C at which the TCXO seems to have a slightly higher frequency drift.

3.5 Model Fitting Effects

During the calibration phase described in Section 3.1, the system collects a set of tuples given by $\langle \delta C_{12}, \delta C_1 \rangle$ and $\langle \delta C_{12}, \delta C_2 \rangle$. The cubic fit SCXO employs a third order polynomial fit to the tuples to form the relation given by Equation 11. It is found that the cubic fit fares quite well with root mean square error consistently below $10^{-1} ppm$.

Alternatively, the SCXO lookup table implementation compiles the calibration data in a one-to-one mapping between δC_{12} and δC_i . Since the values of δC_{12} are quantized and

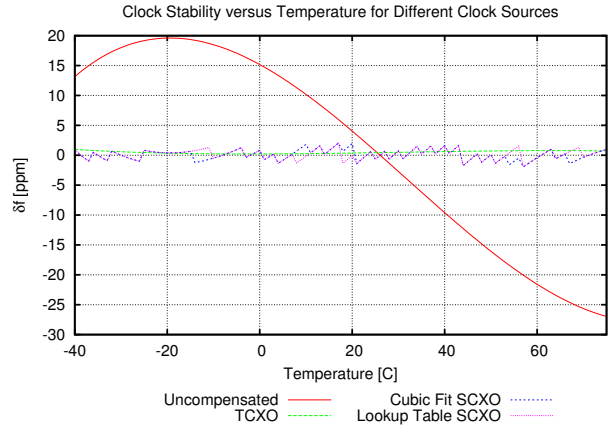


Figure 6: Frequency Drift versus Temperature characteristics for Uncompensated, TCXO, Cubic Fit SCXO and Lookup Table SCXO

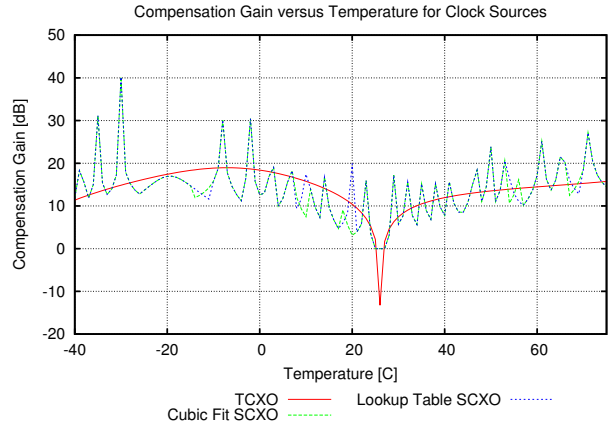


Figure 7: Frequency Drift versus Temperature characteristics for Uncompensated, Temperature Compensated, Cubic Fit based Software Compensation and Lookup Table based Software Compensation

we have a calibration data point for each possible value of δC_{12} stored in the table, no interpolation is required.

A comparison of the cubic fit SCXO and the lookup table SCXO is shown in Figure 6, which illustrates the $f - T$ characteristics for $F_0 = 1 MHz$ and $F_s = 2 Hz$. It is observed that the lookup table characteristics generally follow that of the cubic fit. For this set of oscillator characteristics and choice of F_0/F_s , the lookup table size is only 37 entries which covers all possible values of δC_{12} .

Figure 7 depicts the difference in compensation gain, as defined in Section 3.4. As the δC_{12} and δC_i values are themselves quantized and the calibration data points cover all values, the mean compensation gain for the lookup table implementation is slightly higher than that of the cubic fit.

3.6 Quantization Effects

Due to the discrete-time nature of the compensation algorithm, quantization errors are introduced in the measurement of δC_{12} and in the adjustment of γ . Recall from Equation 6 that δC_{12} is computed from the captured values of the counters. In each sampling period, each counter increments by a nominal value of F_0/F_s . Therefore, the maximum error

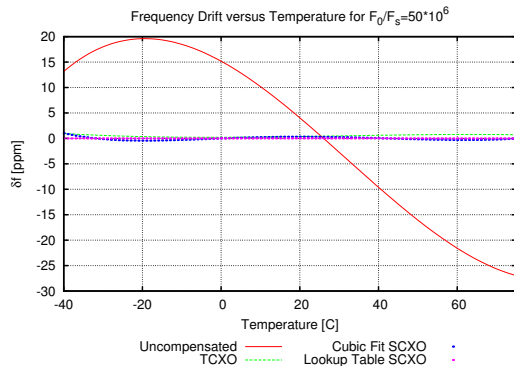


Figure 8: Frequency Drift versus Temperature characteristics for Clock Source 2 showing minimal effects due to quantization when $F_0/F_s = 50 \times 10^6$

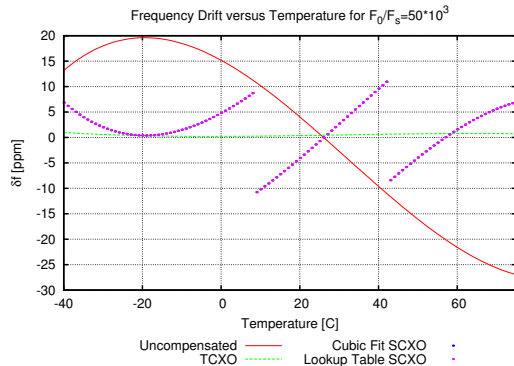


Figure 9: Frequency Drift versus Temperature characteristics for Clock Source 2 showing detrimental effects of quantization when $F_0/F_s = 50 \times 10^3$

in the measurement of δC_{12} due to both counters is $2F_s/F_0$. This relation shows that if the F_0 is made higher or F_s is lowered, the performance of the compensation scheme would improve.

Further, during the compensation phase the value of γ that is computed is approximated to the closest integer value in order to adjust the counter that generates f_γ . Due to this quantization, an error is introduced in the value of f_γ . This in turn affects the accuracy with which the measurement of δC_{12} is taken, since the sampling period in the compensation phase should match $1/F_s$ as closely as possible. It can be shown that the error in f_γ due to this is approximated by F_s^2/F_0 . This relation further supports the intuition that if the nominal frequency was increased or the sampling interval was increased, it would lead to better stability. This is verified through simulation. Figure 8 illustrates the performance when F_0/F_s is set to 50×10^6 and Figure 9 depicts the case when F_0/F_s is 50×10^3 . Note that the dynamics of temperature variation pose a lower limit on the value F_s as it must be ensured that F_s is high enough that temperature variations are not missed. Consequently, the choice of $F_s = 2 \text{ Hz}$ in our evaluation (described in Section 4) is conservative from a performance perspective.

3.7 Other Effects

Apart from the effect of choices made within the compensation technique described above, numerous other areas have been identified that could lead to reduced performance if not

F_0/F_s	Cubic Fit Gain (dB)		Lookup Table Gain (dB)	
	μ	σ	μ	σ
5K	0	0	0	0
50K	6.1	5.4	6.1	5.4
500K	14.3	6.3	14.5	6.1
5M	18.5	6.1	25.8	6.6
500M	18.5	6.8	45.2	6.0

Table 1: Quantization Effects on Compensation Gain

handled properly. With the exception of certain latency issues covered in Section 4.3, the detailed effects of these issues will be considered in subsequent research. Firstly, an error is introduced in the measurement of δC_{12} when $f_\gamma \neq F_s$. This effects to which part of the $\delta f_{12} - \delta f_1$ curve δC_{12} is being mapped. Secondly, when the span of δf_{12} is small, as in $\theta = 6', 8'$ curve in Figure 5, the effects of quantization error are more pronounced.

The third set of errors accrue from the fact that executing the compensation algorithm in software results in latencies and some indeterministic jitter due to interrupt handling. The following latencies all affect the accuracy of the compensated clock:

1. Computational latency in executing the algorithm, which sets an upper bound on the value of F_0/F_s
2. Latency in capturing counters C_1, C_2 and also latency in adjusting the value of γ in the timer
3. Latency in resetting counters for the next sample
4. Time difference in capturing and resetting counters (i.e., time difference between first four steps in Procedure 3) since they cannot be done concurrently in software (effect is especially large if the processor clock is lower than F_0)

Finally, the algorithm assumes floating point computation, and only considers temporal quantization errors. However, if a floating point unit is unavailable and fixed point computation is required due to resource constraints, additional quantization error will be introduced.

4. EVALUATION

We created a testbed in order to emulate the oscillator behavior under different environmental impacts, and to evaluate our algorithm on real hardware. The goal of the testbed is to emulate different oscillator characteristics without the need of rewiring different types of oscillators for every test, and to remove the need of a temperature chamber, that would be needed in order to change the oscillators drifts. Furthermore, the testbed allows us to simulate other impacts on clock drift, such as change in voltage source, change of oscillator speed itself, etc.

4.1 Environment Emulator

We use two Agilent 33220A arbitrary waveform generators to emulate any environment temperature by remotely programming them with a predefined frequency. Figure 2 depicts the frequency drift for different AT-cut crystals. Using these graphs and choosing two different cuts, one can set the two waveform generators to two individual frequencies, and thus successfully emulate the impact of a specific environment temperature on the crystals themselves.

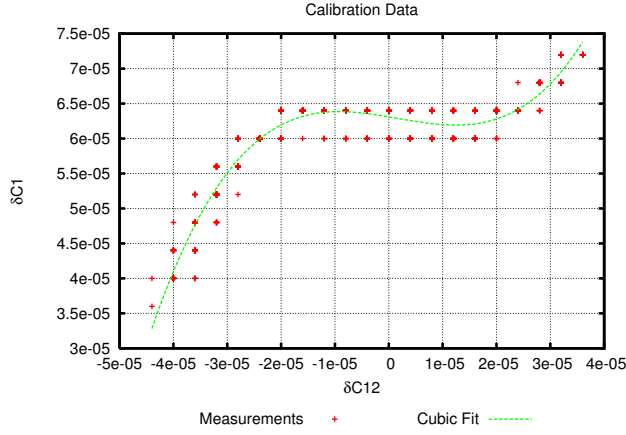


Figure 12: Measured calibration and its cubic curve fit. Note the quantization effect which comes from measuring frequency difference with digital counters. The effect on error of this is discussed in Section 3.6.

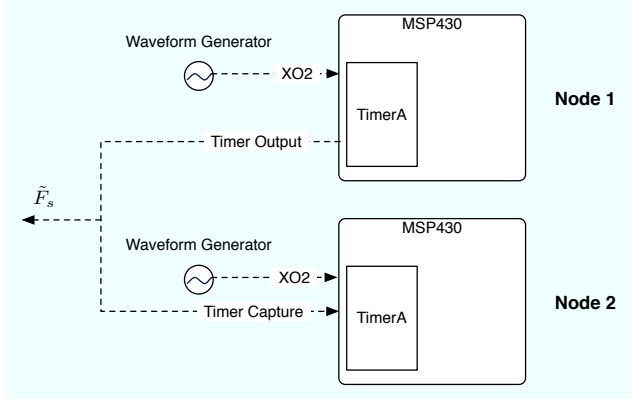


Figure 13: Block diagram of our compensation experiment.

While correcting for the new γ , Node 1 also keeps track of the global time estimate by incrementing a third counter by F_0/F_s every time it estimates a new γ .

4.7 Experimental Results

We tested our implementation of the compensation phase by subjecting the two nodes to the frequency drifts happening over the full scale of temperature from -40° to 75° Celsius. We measured how much Node 1's estimate γ diverges from F_0/F_s . Figure 14 illustrates this further. We can see that our implementation of the compensated clock lies within $\pm 7\text{ppm}$ of the real frequency F_0 , 95% of the time. This is expected since the Agilent frequency generator sources we use for the clocks themselves have an inaccuracy of about 10ppm.

Given our algorithm, we expected our software compensated oscillator to have a poor short term (order of seconds) stability. This happens because γ is quantized and thus will jump between two values back and forth. If γ is slightly below the value it should be after the fitted compensation curve, δ_{12} will be smaller in the next iteration, and thus the next γ will be a little bit bigger again. Nevertheless, this should even itself out for medium term (order of tens of seconds) stability. A good measure for clock stability is the

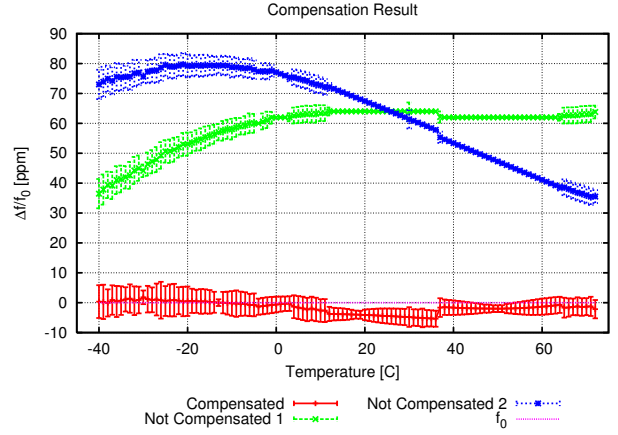


Figure 14: Compensated clock drift over the full temperature range of -40° to 75° Celsius. We also show the drift of the two un-compensated crystals used in the compensation algorithm.

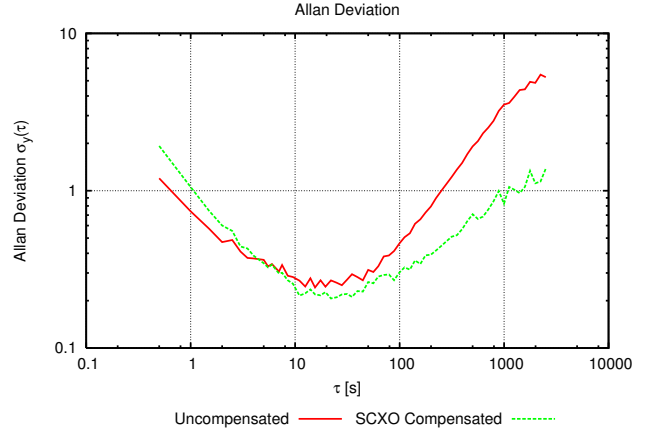


Figure 15: Frequency stability expressed by the Allan Variance for the compensated and un-compensated clock. We can see that we trade short term vs. long term stability by employing our compensation algorithm.

Allan Variance [2] defined as

$$\sigma_y^2(\tau) = \frac{1}{2} \mathbb{E} [(y_{n+1} - y_n)^2] \quad (16)$$

where y_n is the normalized average frequency over the sample interval, and τ is the length of each sample interval. Figure 15 depicts the Allan Variance for two different runs of our experiment. From this graph we can see that the compensated clock source indeed does have a worse short term than mid term stability. Over longer terms (order of minutes to hours) the clock stability gets worse again which indicates that there is still some drift over these time ranges. This was to be expected since we went through a huge temperature range. Future long term tests under realistic temperature ranges obtained from real measurements will show how the compensation techniques works under these conditions. At the same time, Figure 15 shows that the SCXO trades short term versus long term stability compared to the un-compensated clock source.

A more interesting measurement for application developers is the drift of the local clock in relation to global time. Figure 16 shows this difference for two runs of our emulator.

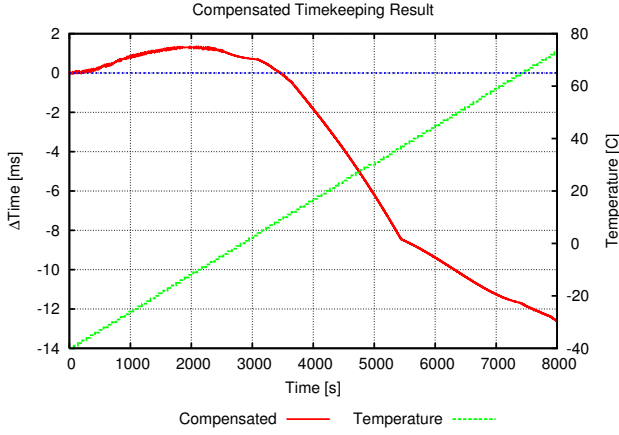


Figure 16: Difference between global time and the estimated time at node 1 over 2.5 hour and a change in temperature from -40° to 75° Celsius. The different phases come from the inaccuracy introduced by the frequency generators, which not always set the output frequency to the desired frequency (see Figure 14).

Again, we changed the temperature from -40° to 75° Celsius and queried Node 1 in regular intervals for its estimate of the global time. The graph shows the difference between Node 1's estimate and the real global time. After 2 hours, our compensated clock was 11.5 milliseconds too slow. This represents a mean effective clock stability of 1.6 ppm.

5. DISCUSSION

In this section we will compare how the SCXO compares to a TCXO for a couple of different applications. As we showed so far, the SCXO achieves a similar accuracy as the TCXO, but as we will show in Section 5.4 for a fraction of its price.

5.1 Applications

We want to show the impact on time keeping for a real-life scenario and compare our SCXO to a commercially available TCXO [1] and the uncompensated clock. We use the temperature data collected by the Mossam project [11] from James Reserve. The time we look at is from October 1st, 2004 to November 10th, 2007. The top plot in Figure 17 shows the temperature over the full time span. The temperature ranges from -12° to 34° Celsius, and one can clearly see the daily variations, as well as the seasonal changes in mean temperature. We can now use these numbers and calculate the effect on the frequency drift of an uncompensated AT-cut crystal, our lookup table (LUT) compensated clock, as well as the TCXO (subplots 2, 3, and 4 in Figure 17).

Using these numbers, we can calculate the accumulated time error over time. Figure 18 illustrates the timekeeping error for the uncompensated clock, the TCXO, the SCXO with a cubic fit, and the SCXO with the lookup table implementation. The uncompensated clock performs the worst and accumulates a time shift of 960.8 seconds over the 3 year period. The SCXO with cubic fit performs much better and accumulates only 30.1 seconds. Even better is TCXO which accumulates a total of 26.6 seconds, and the best is the LUT compensated SCXO which accumulates only a 3.3 second time shift over the whole 3 years. These numbers correspond to a mean effective stability over the full 3 year

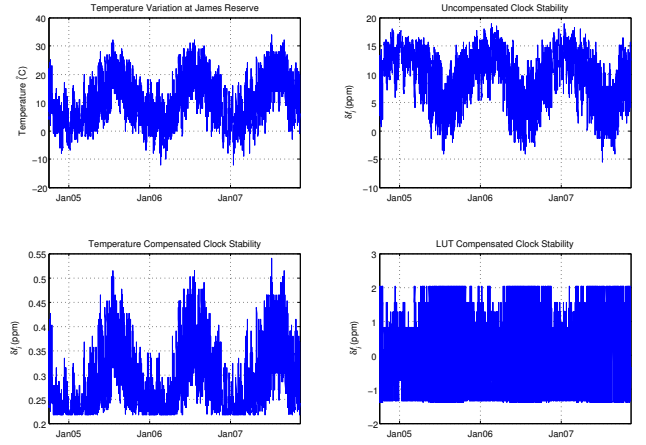


Figure 17: Clock stability using a real-life temperature data set. One can clearly see the daily temperature swings, as well as the seasonal changes in temperature, and how it affects the different compensation schemes.

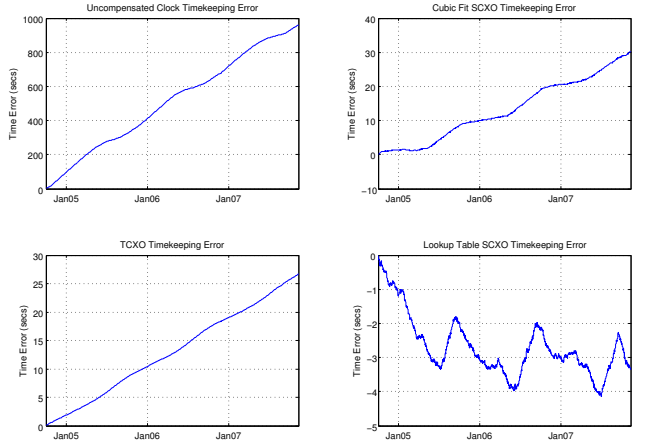


Figure 18: Illustration of the timekeeping error for the uncompensated crystal, and 3 different compensation schemes.

period of 10.13ppm for the uncompensated, 0.32ppm for the cubic fit SCXO, 0.28ppm for the TCXO, and 0.04ppm for the LUT SCXO respectively. This shows that there is a tremendous potential for timekeeping by employing more accurate clocks, and our LUT compensated clock even outperforms a commercially available TCXO by a factor of 8 over temperature ranges occurring in a real deployment. Furthermore, the LUT compensated clock doesn't require costly and power hungry TCXO hardware.

In wireless sensor networks, power is the most scarce resource and thus duty cycling is key to long network lifetime. In [4] P. Dutta argues that with a clock of $\pm 50ppm$ drift (or rather drift between two clocks in a network), the lower bound on duty cycling is 0.01%. Using the same formula, $DC = 2 * \delta f + \frac{t_{pkt}}{T_{pkt}}$, where t_{pkt} is the time it takes to send a packet, and T_{pkt} the packet interval, we can calculate that with our LUT compensated clock we push the lower bound of duty cycling to $8 \cdot 10^{-6}\%$. This shows that in that regime, radio speed and radio wakeup (impacts t_{pkt}), and the rate at which packets are sent clearly dominate the possible duty

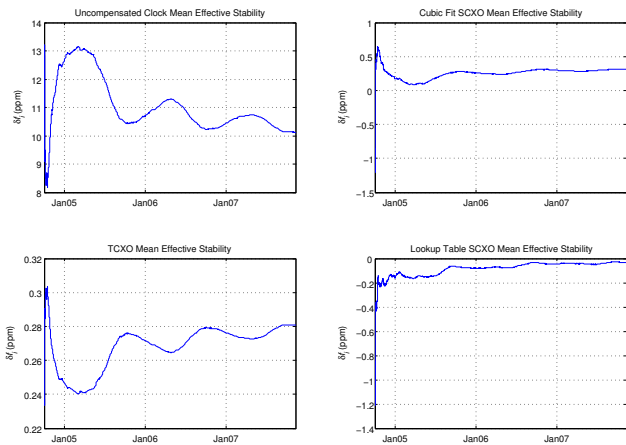


Figure 19: Mean effective clock stability for the uncompensated oscillator, and 3 different compensation schemes. One can see the clocks short, and long term stability over the period of 3 years.

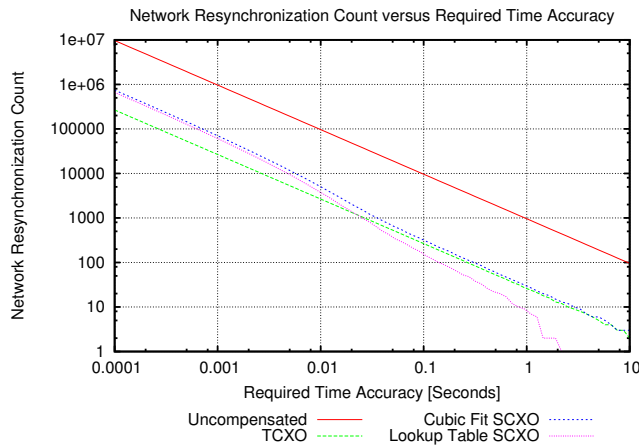


Figure 20: The number of times a node has to resynchronize for a given time accuracy threshold

cycle of a node again.

5.2 Resynchronization Rate

An other important measure in wireless sensor networks is the overhead introduced by a time synchronization protocol, given the application’s need for a specific time precision. We analyzed the James Reserve temperature data by using an oracle that could tell the node when it drifted more than the allowed threshold from the real time. Note that this is a lower bound on the number of necessary resynchronizations. If the nodes would have to estimate this time, they had to use a more pessimistic estimate in order to guarantee the time synchronization accuracy. Nevertheless, the oracle approach gives us a first idea on how more precise clocks impact the resynchronization rate.

Figure 20 shows the number of times each protocol would have to resynchronize in order to achieve a given time accuracy. We can see that the TCXO performs better for time accuracies below 20ms. We assume that this is an artifact of the bad short term frequency stability, and we have to inves-

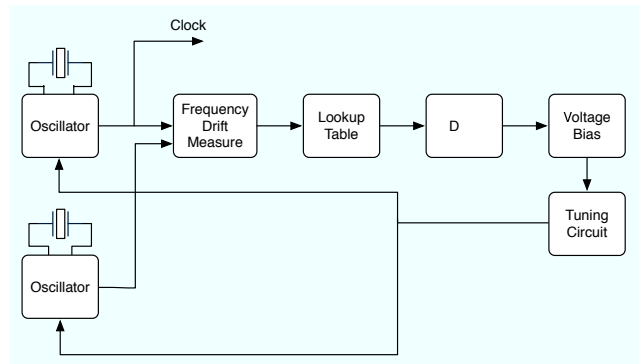


Figure 21: Block diagram of an ideal hardware base Crystal Compensated Crystal Oscillator (XCXO).

tigate further what one can do against this. Nevertheless, the cubic fit and the LUT SCXO still perform an order better than the uncompensated crystal oscillator. This directly impacts the benefit of fewer synchronization which results in tremendous power savings. Future analysis will show how the SCXO performs while using a prediction model as used by RATS [8].

5.3 Crystal Compensated Crystal Oscillator (XCXO)

One major difference between the SCXO and a normal TCXO is that it doesn’t provide a corrected crystal, but a corrected counter. A future idea is to use a similar technique as applied in the SCXO and tune the crystals to the correct frequency. This would have the advantage that the compensated crystal could be used directly in digital circuits without counters. Figure 21 illustrates a block diagram of such a crystal. It still exploits two crystals and their difference, but instead of adapting the value of a counter, it directly tunes the crystal through a tuning circuit. However, the formulas and exact working of such a crystal compensated crystal oscillator (XCXO) have to be worked out and we leave it for future study.

5.4 Cost – Performance Comparison

In this section, we will discuss the expected bill of material (BOM) for an implementation of a SCXO, XCXO, and compare it to similar devices, mainly TCXO’s. First, we need to estimate the BOM of a TCXO. [22] mentions that 40-50 % of the cost of a TCXO go into calibration of the device. Thus, the pure BOM of a TCXO, given the data from Figure 1 lies between \$2.50-\$14 for an accuracy ranging between 8-1ppm. For an SCXO, we need two crystal oscillators (\$0.40/unit) and an additional unit that can do the compensation. For a prototype, we estimate that the whole algorithm could fit in hardware onto less than 100,000 gates. An appropriate FPGA would be the Xilinx Spartan-3E XC3S100E FPGA that cost \$2/unit². Thus, a prototype costs around \$3 which is well in the lower range of comparable TCXO’s cost. Nevertheless, one could imagine that the whole circuit would be an additional peripheral on a microcontroller or system on chips (SoC), where the increase in number of gates, and thus cost and complexity, would be negligible. In such a scenario, a SCXO with a precision of 5ppm comes at the cost of \$1, since it is just one additional crystal.

²Price for 500,000 units November 2007

6. CONCLUSION

We presented an initial algorithm and concept of a software compensated crystal oscillator. We demonstrated in simulation that we can achieve a $10\times$ improvement in accuracy over an uncompensated crystal by exploiting the manufacturing differences in crystal oscillators and employing a simple algorithm written in software. We show that our method can compete with the accuracy of modern temperature compensated crystal oscillators, and that even the estimated BOM of a prototype costs about the price of a TCXO with worse accuracy. Additionally, we discussed how higher accuracy in local clocks can benefit the timekeeping in real deployments. We showed that a standard oscillator will accumulate a time difference of 960.8 seconds over a period of 3 years, whereas a software compensated crystal oscillator with lookup table accumulates only 3.3 seconds. This corresponds to a mean effective stability of 10.13ppm for the uncompensated, and 0.04ppm for our software compensated crystal oscillator. This increase in clock stability allows the duty cycles of sensor network nodes to be lowered, and thus results in longer network and deployment life.

The possibilities and flexibility of a software compensated crystal oscillator have to be explored further. For example, we imagine that with the interaction of time synchronization protocols and the software compensated crystal, we may be able to learn the compensation parameters on the fly, which would decrease the cost of the crystal significantly by 40%-50%. Additionally, the software could learn a crystal's aging factors over time and compensate for them. To our knowledge, there is no such crystal currently on the market, even though aging is one of the major problems in current high precision crystal oscillators.

7. REFERENCES

- [1] DS4026-10MHz to 51.84MHz TCXO. *Maxim Integrated Products, Dallas Semiconductor*.
- [2] ALLAN, D., ASHBY, N., AND HODGE, C. *The Science of Timekeeping*. Hewlett-Packard, 1997.
- [3] CANDELIER, V., CARET, G., AND DEBAISIEUX, A. Low profile high stability digital tcxo: ultra low power consumption tcxo. *Frequency Control, 1989., Proceedings of the 43rd Annual Symposium on* (31 May-2 Jun 1989), 51-54.
- [4] DUTTA, P., CULLER, D., AND SHENKER, S. Procrastination might lead to a longer and more useful life. *HotNets-VI* (2007).
- [5] EIDSON, J. K. L. Ieee 1588 standard for a precision clock synchronization protocol for networked measurement and control systems. *Sensors for Industry Conference, 2002. 2nd ISA/IEEE* (19-21 Nov. 2002), 98-105.
- [6] ELSON, J., GIROD, L., AND ESTRIN, D. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review* 36 (2002), 147-163.
- [7] FILLER, R., AND VIG, J. Long-term aging of oscillators. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on* 40, 4 (1993), 387-394.
- [8] GANERIWAL, S., GANESAN, D., SHIM, H., TSIATSI, V., AND SRIVASTAVA, M. B. Estimating clock uncertainty for efficient duty-cycling in sensor networks. In *SenSys* (New York, NY, USA, 2005), ACM Press, pp. 130-141.
- [9] GANERIWAL, S., KUMAR, R., AND SRIVASTAVA, M. Timing-sync protocol for sensor networks. *SenSys* (2003).
- [10] IEEE STANDARDS BOARD. IEEE guide for measurement of environmental sensitivities of standard frequency generators. *IEEE Std 1193-1994* (27 Feb 1995).
- [11] JAMES RESERVE. MossCam. <http://www.jamesreserve.edu/mossCam/>, 2007.
- [12] JOHANNESSEN, S. Time synchronization in a local area network. *IEEE Control Systems Magazine* 24, 2 (2004), 61-69.
- [13] KUBO, K., AND SHIBUYA, S. Analog tcxo using one chip lsi for mobile communication. *Frequency Control Symposium, 1996. 50th., Proceedings of the 1996 IEEE International*. (5-7 Jun 1996), 728-734.
- [14] LEE, S.-J., HAN, J.-H., HANK, S.-H., LEE, J.-H., KIM, J.-S., JE, M.-K., AND YOO, H.-J. One chip-low power digital-tcxo with sub-ppm accuracy. *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on* 3 (2000), 17-20 vol.3.
- [15] LI, M., HUANG, X., TAN, F., FAN, Y., AND LIANG, X. A novel microcomputer temperature-compensating method for an overtone crystal oscillator. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on* 52, 11 (Nov. 2005), 1919-1922.
- [16] MARÓTI, M., KUSY, B., SIMON, G., AND LÉDECZI, Á. The flooding time synchronization protocol. *SenSys '08* (2004), 39-49.
- [17] MILITARY SPECIFICATIONS AND STANDARDS,. *MIL-O-55310, Military Specification, Oscillators, Crystal, General Specification for*. 700 Robbins Ave., Bldg. 4D, Philadelphia, PA 19111-5094, 2007.
- [18] MILLS, D. L. Internet time synchronization: the network time protocol. *IEEE Trans. on Communications* 39, 10 (1991), 1482-1493.
- [19] MOTEIV. Tmote sky datasheet.
- [20] NEWELL, D., AND BANGERT, R. Temperature compensation of quartz crystal oscillators. *17th Annual Symposium on Frequency Control. 1963* (1963), 491-507.
- [21] NEWELL, D., AND HINNAH, H. Automatic compensation equipment for tcxo's. *22nd Annual Symposium on Frequency Control. 1968* (1968), 298-310.
- [22] NEWELL, D. E., AND HINNAH, H. Automatic compensation equipment for tcxo's. *22nd Annual Symposium on Frequency Control* (1968), 298-310.
- [23] SHIN, K.-S., AND CHUNG, W.-Y. Automatic tcxo frequency-temperature test chamber using thermoelectric device array [temperature compensated crystal oscillator]. *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on* 3 (2001), 1624-1627 vol.3.
- [24] ZHOU, W., ZHOU, H., XUAN, Z., AND ZHANG, W. Comparison among precision temperature compensated crystal oscillators. *Frequency Control Symposium and Exposition, 2005. Proceedings of the 2005 IEEE International* (29-31 Aug. 2005), 5 pp.-.