Demo Abstract: A Platform Enabling Local Oscillator Frequency Synchronization

Anh Luong[1], Thomas Schmid[1], Neal Patwari[1,2]
[1]University of Utah, [2]Xandem Technology
Salt Lake City, Utah 84112, USA
{anh.n.luong, thomas.schmid, neal.patwari}@utah.edu

ABSTRACT

We introduce an platform architecture and algorithm to frequency synchronize multiple devices. The platform allows clock unification among oscillator, microcontroller, and radio. The platform accesses complex baseband samples from the radio, estimates the carrier frequency offset, and iteratively drives the main local oscillator (LO) frequency difference between two devices to zero.

1. INTRODUCTION

Many applications require wireless devices to have frequency synchronized clocks. Low-power wireless sensors that sleep for long periods of time have a duty cycle that is fundamentally limited by their time and thus clock frequency synchronization accuracy [1]. The estimation of relative velocity from Doppler, useful for source localization, requires accurate knowledge of a signal's center frequency at zero velocity. Distributed MIMO communication systems or distributed phased array receivers [2] can improve throughput in ad hoc networks but require frequency synchronization across groups of transceivers.

While frequency synchronization is performed in communications receivers to enable demodulation, the measured frequency (or frequency offset) of the received signal is not typically available outside of the receiver. An oscillator's frequency can be transmitted as a pilot and used by multiple receivers [2], but this has not been used to synchronize other clock frequencies on the devices. Typically, the processor clock is independent of the radio clock. Thus even if the radio's clock can be synchronized to that of the transmitting radio, the processor runs at a rate independent of the transmitter's processor.

Standard platforms enable clock frequency correction, in which a receiver learns its relative clock offset through repeated time of reception measurements of packets transmitted at a known period on the transmitter's clock. This estimated clock frequency skew is then used to correct a timer derived from the receiving device's local clock [3]. Alternatively, the ambient electric grid frequency can train a low-power oscillator [1], but such methods have not been used to synchronize the main local oscillators on different devices.

In this demonstration abstract, we demonstrate the ability to frequency synchronize nodes' local oscillators by leveraging the IQ sample feature of an off-the-shelf low-power transceiver. Additionally, the platform provides clock unification, that is, the synchronization extends to the clocks used by the two devices' microcontrollers.

SenSys '16 November 14-16, 2016, Stanford, CA, USA

ACM ISBN 978-1-4503-4263-6/16/11.
DOI: http://dx.doi.org/10.1145/2994551.2996532

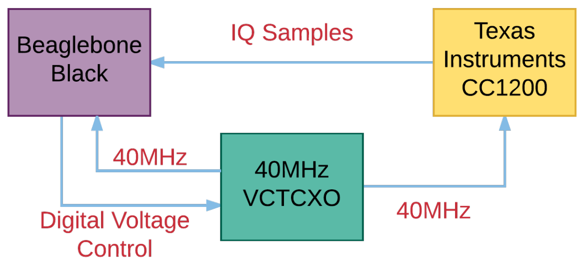

Figure 1: A Beaglebone Black PRU collects complex baseband samples from the CC1200, computes the frequency offset and corrects the shared VCTCXO. The algorithm iterates until the frequency difference between two devices is zero.

2. SYSTEM DESIGN

To explore clock frequency synchronization between two wireless devices, we build a platform that provides the system architecture shown in Fig. 1. The key components on the platform used in this demonstration are:

1. Voltage control temperature compensated crystal oscillator (VCTCXO) which produces a stable 40 MHz signal. The VCTCXO has the capability to adjust its frequency in a range of 40 MHz $\pm$ 400 Hz.

2. TI CC1200 transceiver. The CC1200 is a transceiver which can operate in any of the 169 MHz, 434 MHz, and 900 MHz frequency bands.

3. Beaglebone Black (BBB). The main processor runs a preemptive Linux OS and thus can't capture the IQ samples from the CC1200 at a precise regular interval. Instead, we use one of the two real-time co-processors

in the programmable real-time processing unit (PRU) sub-system to collect the timing-dependent samples.

The platform also has a Microsemi IGLOO FPGA which is used as a routing table for signal wiring and clock synthesis distribution. It also has a compatible header for the Beaglebone Black and serves as a cape.

The CC1200 provides a feature called *automatic frequency compensation*, which corrects the carrier frequency via internal phase-locked loop (PLL) control registers. The correction factor is calculated from the frequency offset between the received signal and CC1200-generated carrier. This however does not provide an option to alter the frequency of the local oscillator external to the CC1200.

For our demonstration system, we require frequency synchronization of the local oscillator of the device, which is shared across the device. Thus we rely instead on the *IQ sample* feature of CC1200. The CC1200 exports two registers for the angle of each sample, immediately after its coordinate rotation digital computer (CORDIC) algorithm, with 10 bits resolution. The sampling period is determined by the channel bandwidth setting. In our setup, a new IQ sample comes through the 7 MHz SPI bus every 22.2 μs, a sample rate of 45.044 kHz. The data is meaningless if the sample is read out while the buffer is being written with another sample. Thus, we check the rising edge of CC1200 `MAGN_VALID` signal, which indicates the availability of the new measurement.

Our application on the BBB first configures the radio, starts angle data collection on the PRU, and writes the data into shared memory. The application then unwraps the phase and accumulates the phase change that occurs over a period of N samples, and from it estimates the carrier frequency offset. The phase is measured with 10 bits resolution, which translates to a frequency resolution of $\frac{45044}{2^{10}(N-1)} = 43.99/N$ Hz. For example, with $N = 1000$, we can expect to resolve carrier frequency offset to within 0.044 Hz.

3. EVALUATION

In order to validate our system measurements, we use a National Instruments vector signal generator to generate a carrier frequency at 434 MHz. With a Keysight 53230A 12-bit frequency counter, we measure the generated frequency to be 434 MHz -62.27 Hz. For CC1200, the closest achievable carrier frequency with the ideal 40 MHz is 434 MHz -61 Hz.Thus, the ideal internal PLL's multiplier and divisor registers were setup with the factor of 10.849998475. We validated this factor by transmitting a pure carrier wave with the CC1200 after setting the correct factor and various local oscillator frequencies.

With the aforementioned PLL factor, the target local oscillator frequency should be 40 MHz -0.117 Hz. The CC1200 has various intermediate frequency (IF) frequency settings for the user to choose from. For this particular setup, we selected 138.88 kHz as our IF frequency. We ran the preliminary frequency synchronize algorithm, as the crystal oscillator settle at $40MHz + 0.404Hz$, just 0.52 Hz from the ideal frequency. As shown in Fig. 2, we were able to approach zero frequency difference after the sixth iteration of the sync algorithm (w/ $N = 1000$ samples each iteration). In order to achieve higher accuracy, we would have to increase the sample size per group to see a difference of less than ± 0.26 Hz.

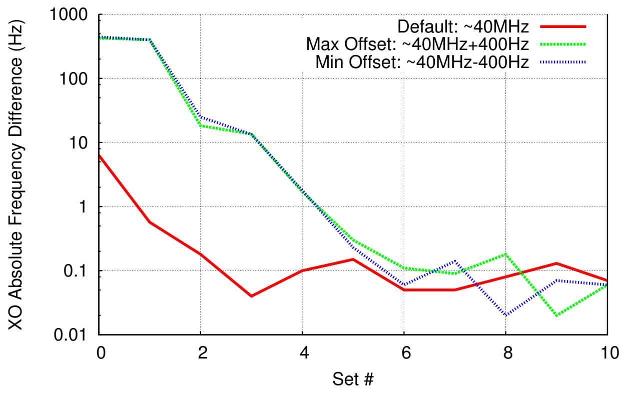

Figure 2: Absolute frequency difference vs. iteration and LO starting frequency. Default: LO starts within 10 Hz of 40 MHz; Max and Min Offset: LO starts at approximately 40 MHz plus or minus 400 Hz, respectively. The receiver synchronizes after at most six iterations (each w/ $N = 1000$ samples) to within about 0.1 Hz of the TX 40 MHz LO.

4. DEMONSTRATION

We demonstrate the frequency synchronization mentioned above with two nodes. One node transmits the carrier frequency at 434 MHz, while the other node stays in RX mode. We start the demo with the RX node having a frequency offset from the TX node. The clocks of the two devices will be observed on an oscilloscope. When the frequency difference is high (100s of Hz) the two clock signals move rapidly on the scope with respect to each other. We will then run our algorithm to synchronize the receiver's 40 MHz clock frequency to the transmitter's 40 MHz clock. After the synchronization, the two clock signals are on the order of 0.1 Hz apart. The observer can watch as the two clock signals stay aligned and move slowly with respect to the other.

5. ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. National Science Foundation under Grant Nos. #1329755 and #1407949.

6. REFERENCES

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

[2] D. Jenn, Y. Loke, T. C. Hong Matthew, Y. E. Choon, O. C. Siang, and Y. S. Yam. Distributed phased arrays and wireless beamforming networks. *International Journal of Distributed Sensor Networks*, 5(4):283–302, 2009.

[3] A. Vallat and D. Schneuwly. Clock synchronization in telecommunications via ptp (ieee 1588). In *2007 IEEE International Frequency Control Symposium Joint with the 21st European Frequency and Time Forum*, pages 334–341, May 2007.