Configurable Hardware-Based Radio Interferometric Node Localization

Sándor Szilvási, János Sallai, Isaac Amundson, Péter Völgyesi, Ákos Lédeczi
Institute for Software Integrated Systems
Vanderbilt University, 2015 Terrace Place, Nashville, TN 37203, USA
{sandor.szilvasi, janos.sallai, isaac.amundson, peter.volgyesi, akos.ledeczi}@vanderbilt.edu

Abstract—Wireless ad-hoc sensor networks are typically deployed randomly in space to gather information about the physical world. In these spatially random sensor networks, one of the fundamental problems is precise node localization along with the key requirement to keep node cost at minimum. Existing radio interferometry-based node localization techniques achieve cm resolution using only low-cost radio chips readily available on sensor nodes. This paper proposes an alternative hardware-based solution for the signal processing part of such a technique. We show that our approach tolerates a much wider range of interference frequencies while freeing up precious computational and storage resources on the sensor node. The proposed ranging functionality can eventually be included in current COTS radio transceiver chips, thus accurate localization will be available on the nodes, without a significant additional cost in computation or price.

TABLE OF CONTENTS

1. INTRODUCTION 1
2. RELATED WORK 2
3. PROPOSED ALGORITHM DESCRIPTION 3
4. FPGA IMPLEMENTATION 4
5. RESULTS AND CONTRIBUTION 7
6. CONCLUSION AND FUTURE WORK 9
ACKNOWLEDGEMENTS 9
REFERENCES 9
BIOGRAPHY 10

1. INTRODUCTION

In the past decade, considerable research effort has been expended on wireless sensor network (WSN) applications. These WSNs usually consist of tens, hundreds or thousands of spatially distributed sensor nodes that cooperate to achieve a common goal. In order to do that, nodes must perform several tasks such as sensing, local and cooperative processing of the sensed data and establishing/maintaining the wireless network connection. In several applications the individual nodes in the sensor network are deployed randomly, but the precise location information of the nodes is crucial for the application. Such applications include transient event detection-based sniper localization [1] and tracking periodic signal sources via beamforming [2]. Both of these call for an accurate and reliable node localization method.

One way to classify node localization methods is based on the signal modality they rely on. A wide range of these modalities exist, such as acoustic sound and ultrasound, infrared and visible light, and radio waves. The particular choice of signal modality influences the attainable accuracy, power-efficiency, and the hardware requirements of the localization – factors that are of great significance in WSNs. The demand for small-size sensors with considerable resource constraints, built from cheap and disposable hardware components, inherently determine the method that can be used in a particular WSN application.

The Global Positioning System (GPS) is based on radio signals and is a readily available, widely used technology for localization, providing an adequate positioning solution for various applications. However, the hardware components required to compute the location estimate at each GPS receiver are expensive and computationally demanding, rendering GPS less desirable for WSNs. Furthermore, GPS requires line-of-sight to at least four satellites, thus this localization method is unavailable indoors, or in cluttered urban and forested environments.

Sensor node localization algorithms are usually a three-step process [3]. The first step is meant to determine the spatial relationship between the nodes, and is commonly referred to as *ranging*. This low-level step may utilize measurements based on the above mentioned signal modalities in order to derive relative distances and/or bearings between sensor nodes. The ranging accuracy is crucial regarding the overall localization performance. Therefore, a *calibration* step is performed to compensate for errors due to hardware manufacturing differences and varying operating conditions. Both the ranging and calibration processes are calculated on the sensor nodes, however, the *localization* step is often computed in a centralized manner. Localization resolves the relative or absolute positions of the nodes by fusing the ranging measurements, and is often formulated as an optimization problem. Such a multi-step process that achieves centimeter-resolution accuracy is the Radio Interferometric Positioning System (RIPS) [4].

The main idea of RIPS is to utilize the low-cost radio chips already available on the sensor nodes for localization. Earlier approaches required CPU-intensive signal processing algorithms – tying up resources of the typical low power 8-bit microcontroller architectures used on these sensor platforms. In this paper, we propose an alternative

[1] 978-1-4244-3888-4/10/$25.00 ©2010 IEEE
[2] IEEEAC paper#1397, Version 1, Updated 2009:12:09

hardware-based ranging solution for RIPS that can be included in future radio chips to support precise node localization in WSNs. Results show that our approach out-performs the pre-existing RIPS solution with regards to robustness, accuracy and speed.

The remainder of this paper is organized as follows. In Section 2, we briefly overview the ranging step of the RIPS algorithm. In Section 3, we describe our proposed method for enhanced ranging accuracy. In Section 4, we present a solution using a prototype sensor node based on reconfigurable hardware. Then, we evaluate the performance of our solution using real-world radio signal measurements and make a comparison with previous results in Section 5. Finally, we conclude in Section 6.

2. RELATED WORK

The Radio Interferometric Positioning System was developed as a means for accurately determining relative position of a set of sensor nodes over a wide area by only using the onboard radio hardware. The randomness in the initial phases of the local oscillators on the transmitter and receiver nodes prohibits using the phase of the RF carrier directly for distance measurements. Instead, RIPS employs transmitter pairs at close frequencies for generating an interference signal. The envelope phase of the interference signal is independent of the phase of the receivers. The ambiguity in the phase at the transmitter side is eliminated by using a pair of receivers for measuring the phase *offset* between the receiver pairs. The pairwise phase differences between a sufficient number of participating nodes constrains each node to a unique position in the sensing region. RIPS was shown to have an accuracy of 3 cm and a range of up to 160 meters.

RIPS was originally implemented on the COTS Mica2 mote platform [5]. These resource-constrained devices have a 7.4 MHz processor, 128 kB program memory (ROM), 4 kB RAM, ADC with 9 kSPS sampling rate, and a CC1000 tunable radio transceiver [6] that operates in the 433 MHz ISM band. Although the radio hardware is quite versatile for its size and cost, 433 MHz is too high to analyze the received signal directly.

However, the phase and frequency of the envelope signal can be measured by making successive reads of the received signal strength indicator (RSSI). The phase of this low-frequency signal can be estimated with adequate accuracy, less than 5% error, using limited processing power and currently available time-synchronization methods (<5 us) [7].

Figure 1 illustrates the approach. Two nodes, A and B, transmit pure sinusoids at respective frequencies f_A and f_B, such that $f_B < f_A$. The two signals interfere, resulting in an interference signal with beat frequency $|f_A - f_B|$. The phase offset between receiver pairs is measured, which is a linear combination of the distances between the four participating nodes:

$$\Delta\varphi = \frac{2\pi}{\lambda} d_{AD} - d_{BD} + d_{BC} - d_{AC} \pmod{2\pi}$$

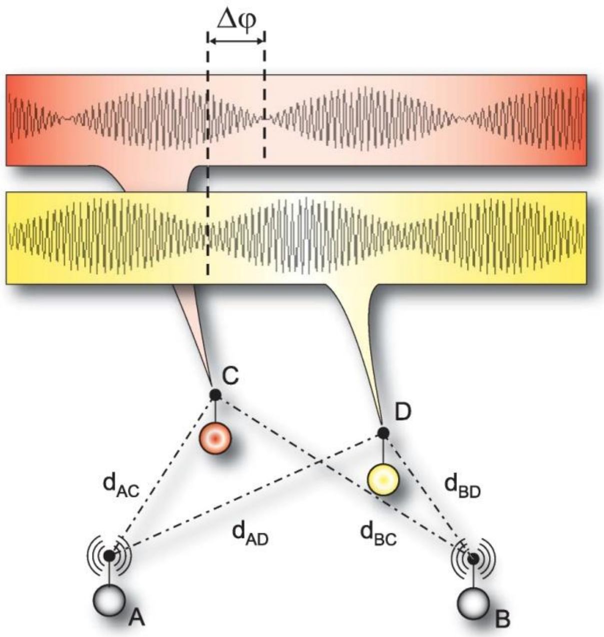

Figure 1: The Radio Interferometric Positioning System

Due to bandwidth limitations, the raw RSSI signal cannot be transmitted to the base station, and therefore must be processed on the mote. However, signal analysis on the resource-constrained mote hardware is non-trivial, involving both online and post-processing steps. During the online processing step, the RSSI signal is placed into a 256-byte buffer, one sample at a time upon each A/D converter interrupt. The clock speed and sampling rate limit the processing to approximately 820 CPU cycles per sample. Upon each interrupt the raw sample is incorporated into a moving average in order to filter out signal noise. Amplitude is determined by collecting the minimum and maximum peak values from the first 10% of the buffered samples, enough to cover at least one period. The amplitude is used as a signal quality indicator, as well as for determining a threshold for peak detection. The peaks are indexed, and are used to determine the frequency and phase of the signal in the post-processing step.

Post-processing is run after the buffer is full. The indexed peaks are used to determine the signal period, and outlying peaks are discarded. Signal frequency is determined by taking the reciprocal of the average period length. Similarly, signal phase is estimated by averaging the phases of the filtered peaks. Upon completion of the post processing step, the computed signal amplitude, frequency and phase are sent to the base station where the localization algorithm is run. The post-processing frequency and phase estimation of the RIPS algorithm takes less than 10,000 cycles per measurement and yields to a phase measurement error of approximately 0.09 radian (approximately 5 degrees).

3. Proposed Algorithm Description

The goal of our proposed FPGA-based RIPS algorithm is to improve the performance of the original RIPS by estimating the phase of the interference signal envelope in a faster, more robust and more accurate way. According to RIPS, two transmitter nodes transmit pure sine waves at frequencies f_1 and $f_2 = f_1 + f_0$ respectively, which produce a signal with an envelope frequency of f_0. The goal of the phase estimation algorithm is to determine the phase of the beat signal at a given time instant (t_1) after the start of reception (t_0). That is, the problem can be summarized as follows:

Given a starting time t_0 and an beat signal s(t) determine the phase of the envelope of s(t) at time t_1 ($t_1 > t_0$).

The current RF frontend in our hardware is integrated into the radio chip, which mixes the RF signal down to an intermediate frequency (IF). However, this IF signal contains not just the frequency components f_1 and f_2. Rather, it is distorted by noise, high frequency harmonic components and significant amount of DC-bias. Most of these originate from either the antenna or the radio chip, and therefore have to be removed to enable robust and accurate phase estimation. The high-level block diagram of the proposed phase estimation algorithm is shown in Figure 2. First, the IF signal is fed through several *signal shaping* stages, removing the carriers and the DC bias in order to obtain the fundamental component of the envelope signal. Then, this signal is fed into a Phase-Locked Loop (PLL) as a reference to lock-on. Once the pre-calculated $\Delta t = t_1 - t_0$ PLL lock-in time elapses, the actual phase of the PLL is captured. This phase value represents the output of the phase estimation algorithm.

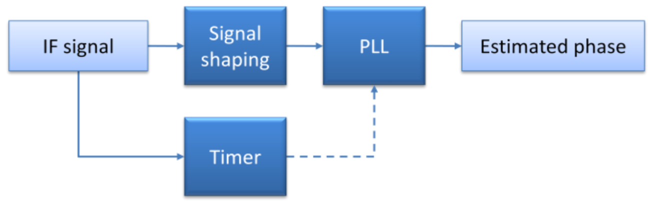

Figure 2: High level design of the Phase Estimation algorithm

IF signal shaping

The signal shaping part of the phase estimation process consists of five consecutive stages in order to gain a DC-unbiased smooth envelope from the IF signal, while exploiting parallelism and keeping resource demands low. Parallelism is exploited by running the stages in a pipeline, while resource usage is kept low by utilizing multi-rate signal processing techniques, that is, filtering the IF signal at a significantly lower rate. The detailed dataflow diagram of these stages is shown in Figure 3.

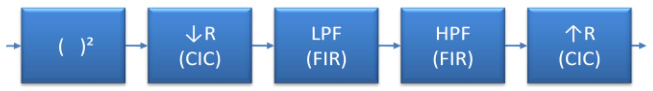

Figure 3: IF signal shaping stages with decimation and interpolation

Consider an ideal IF signal that consists of two spikes at f_1 and f_2, as shown in Figure 4 (a). In order to access the envelope, the IF signal is simply multiplied by itself. By squaring the signal, the linear combinations of the original frequencies appear in the spectrum raising periodic components at 0 Hz (DC), $f_2 - f_1$, $2f_1$, $f_1 + f_2$ and $2f_2$, as depicted in Figure 4 (b).

Four of these components have to be suppressed to access the smooth $f_2 - f_1$ component corresponding to the envelope. This can be achieved by applying appropriate filters to the squared signal. Constructing filters operating at the original 1 MSPS frequency is extremely resource-intensive, therefore, decimation is performed first. Decimation helps relax the filter specifications and allows more time for processing each sample. Therefore, the squared signal is decimated by a factor of *R* using a decimation filter. The sketch of the resulting spectrum is depicted in Figure 4 (c). Note that some high-frequency components transform into the spectrum due to the non-ideal low-pass characteristic of the anti-alias filter.

The down-sampled signal is then further smoothed by a low-pass filter with a cut-off frequency at slightly above $f_2 - f_1$. The resulting spectrum that is free of the three higher frequency components can be seen in Figure 4 (d).

The signal after the third stage is still highly DC-biased. This DC component is removed in the fourth stage by a high-pass filter. The spectrum of the filtered, hence DC-unbiased signal is depicted in Figure 4 (e).

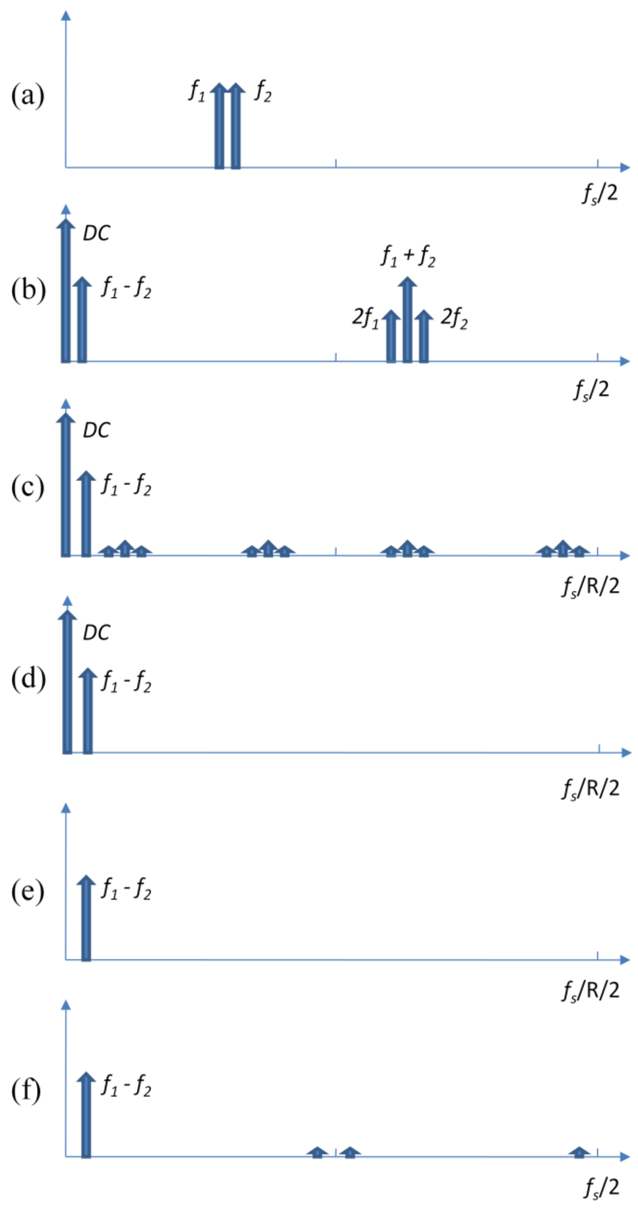

Figure 4: Schematic spectrum of the received IF signal (a), IF signal square (b), after decimation by a factor of R (c), low-pass filtering (d), high-pass filtering (e) and interpolation (f).

Finally, an interpolating filter is utilized in the fifth stage in order to up-sample the signal and to suppress resulting images. The corresponding spectrum is sketched in Figure 4 (f).

Phase-Locked Loop

The RIPS algorithm requires the transmitters to start transmission for phase measurements in a time-synchronized manner. This instantaneous switching produces transients in the received RF and the down-mixed IF signals. The transients in the IF signal also recur on the output of the filter structure that is on the input of the PLL, which can adversely affect the PLL lock-in time. Therefore, a timer logic is utilized to disable the PLL during the transient, as depicted in Figure 2. The PLL is enabled by the timer logic only after a fixed time delay, ensuring that the PLL receives a transient free, smooth beat signal from the *Signal shaping module* that can be used as a reference signal.

The PLL module is realized as an All-Digital PLL (ADPLL), which is a simplified version of the Jitter Bounded DPLL described in [8]. The Jitter Bounded DPLL is a hardware-friendly PLL implementation that can produce a phase aligned output signal with a frequency matching that of the input (reference) multiplied by a rational number from a given range, and a jitter error that is below a well defined and adjustable bound. It uses a logarithmic search approach to find the input frequency in the range of 0 Hz to the sampling frequency, and adjusts the output frequency accordingly. This search is carried out in a fixed number of steps determined by the internal phase representation bit-width and the range of frequencies to scan.

Our implementation is simplified in the sense that it produces an output with exactly the same frequency as the input signal and that it tries to acquire a lock on the input signals from the range of approximately 0 Hz to a several times $f_2 - f_1$, that is, the nominal frequency of the received envelope signal.

4. FPGA IMPLEMENTATION

Throughout the design process, an FPGA-based sensor board (SB) was used as an experimental platform. The SB and its block diagram are shown in Figure 5 and Figure 6 respectively. The heart of the SB is a Xilinx Spartan-3L XC3S1000 FPGA. This low-power FPGA model from Xilinx is running at 20 MHz system clock and has 1 million gate equivalent programmable logic resources suitable for implementing fairly complex signal processing algorithms. The rest of the SB is built around the FPGA. The FPGA configuration data is stored in a Xilinx Platform FLASH memory and is read automatically upon power-on. An 8-Mbyte pseudo-SRAM (pSRAM) is utilized to store intermediate results from the application or to serve as a sample buffer for high sampling rate measurements. The 70 ns clock cycle pSRAM has 16-bit wide data lines which makes it suitable to store up to 4M 16-bit samples at a rate of 1 MSPS. High speed communication between a PC and the FPGA on the SB is supported by an FTDI USB controller providing a Virtual UART communication channel. This channel can be used for downloading samples from and for tuning parameters in the FPGA. The eight LEDs residing symmetrically around the FPGA are also directly attached to the FPGA pins rendering them a useful tool to visualize inner states. The purpose of the low-cost CC1000 radio chip included in the SB is twofold. On one hand it is connected to the FPGA via an SPI port and therefore it can be used for “regular” radio communication. On the other hand, either the power present in the received radio signal or the down-mixed intermediate frequency (IF) signal is accessible on the RSSI/IF pin of the CC1000. This analog signal is then sampled by a 12-bit ADC, connected to the FPGA as an SPI slave module, at a rate of up to 1 MSPS.

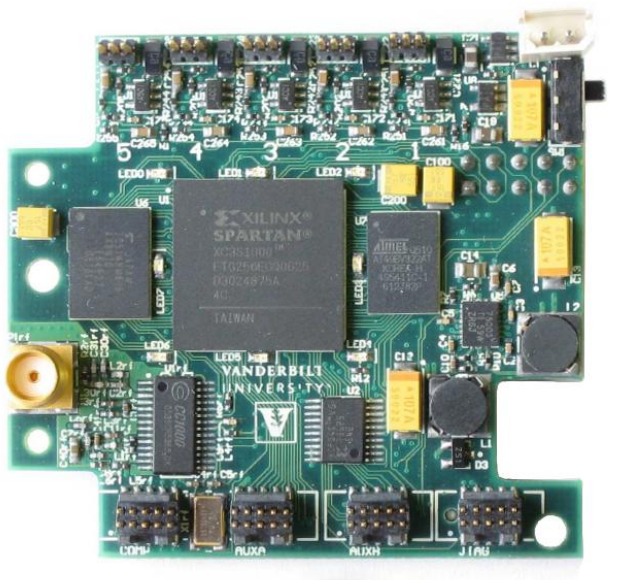

Figure 5: FPGA-based sensor board

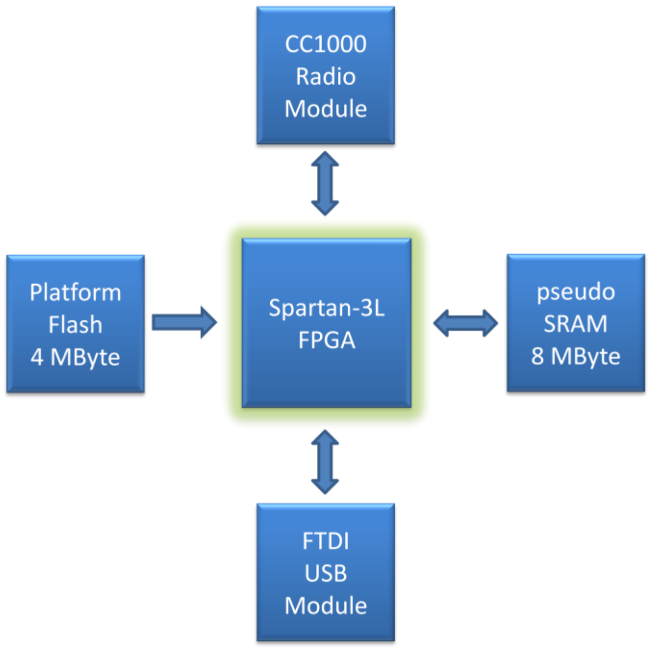

Figure 6: Block diagram of the FPGA sensor board

The FPGA design of the RIPS phase estimation algorithm consists of several modules implemented as register transfer level (RTL) VHDL modules. These modules are essentially signal processing blocks that were either generated with MATLAB toolboxes or were directly written in VHDL. There are, however, some controller functionalities of sequential nature that were implemented using a simple soft-core processor.

The high-level block diagram of the entire FPGA design is described in Figure 7. The IF signal received from the CC1000 radio chip is sampled with a 12-bit A/D converter at 1 MSPS (not shown). Based on the settings dictated by the on-chip *PicoBlaze controller*, the samples are then either routed to the recorder or to the signal shaping block of the RIPS phase estimator module. Raw recordings obtained through the upper branch were used to experiment with, and to help determine specifications for and to design the appropriate filters in MATLAB. On the other hand, the lower branch carries out the actual phase estimation. Once a phase estimation result is available, an interrupt is generated for the *PicoBlaze controller* which, in return, reads and forwards it to the FTDI USB controller. A detailed description of the individual modules is presented below.

RIPS Phase Estimator module

The *Phase Estimator module*, found in the lower branch in Figure 7, logically encapsulates the *Signal shaping* and *ADPLL* modules and provides timing control logic, as shown in Figure 2. It receives the 12-bit IF sample values as its input at a rate of 1 MSPS and provides the estimated phase as its output. The received IF signal is expected to have dominant frequency components f_1 and f_2, providing an interference frequency of $f_2 - f_1 = 1$ kHz ± 500 Hz.

All the functionalities of the *RIPS Phase Estimator module* were first developed in MATLAB using "synthetic" pure sine wave input signals and double precision floating point numeric representation as a proof of concept. The input signal was replaced with actual recordings, and later all the internal variables were turned into fixed-point values of appropriate precision. Once the corresponding VHDL modules were created, they were functionally verified by bitwise matching the MATLAB and VHDL (ModelSim) simulation results.

Signal shaping module

The *Signal shaping module* contains a hardware multiplier and a set of multi-rate filters to extract the 1 kHz beat signal and to suppress all the other components. This module is always enabled and has the structure shown in Figure 3 with a decimation and interpolation rate of $R = 16$.

The first and last filter stages are responsible for the sample rate conversion. Cascaded Integrator Comb (CIC) decimator and interpolator filters are utilized that provide a resource-economic low-pass filter to overcome anti-aliasing and anti-imaging, respectively [9]. The *CIC decimator* in the first stage operates at 1 MSPS and down samples the IF signal by a factor of $R = 16$ to 62.5 kHz. The CIC filter has $M = 2$ stages, meaning that the output has to be compensated by a constant factor of $R^M = 256$.

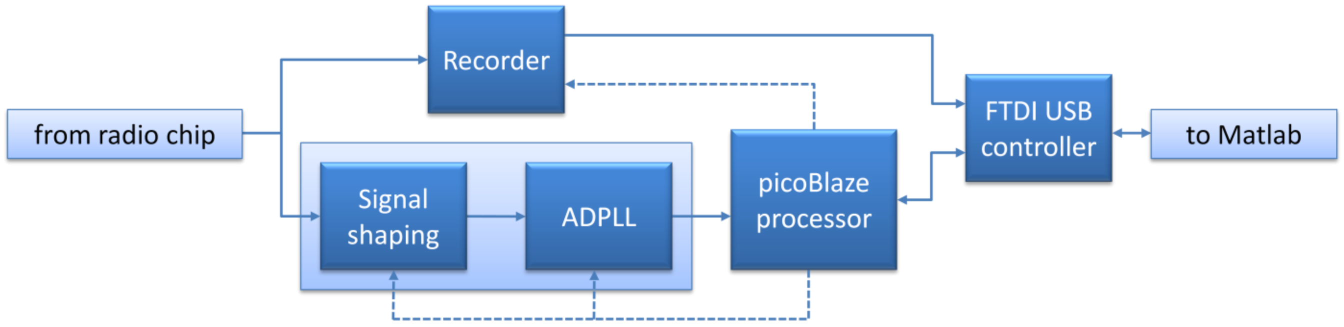

Figure 7: High-level block diagram of the phase estimator FPGA design

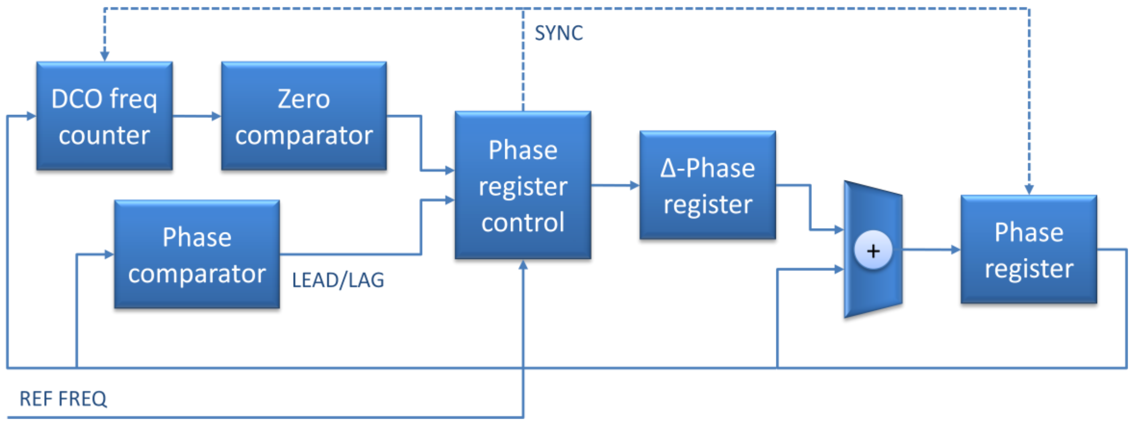

Figure 8: Block diagram of the ADPLL module

The *low-pass filter* at the second stage is intended to eliminate components at frequencies higher than 1 kHz by providing a stopband attenuation of 30 dB for frequencies higher than 2 kHz. The selected filter structure is a 72-tap symmetric FIR filter that has to operate at 1 MSPS / R = 62.5 kHz. Since the system clock frequency of 20 MHz is significantly higher, there is enough processing time to calculate the filter output serially, using a single hardware multiplier.

The *high-pass filter* at the third stage aims to remove the DC content of the signal. To achieve the steep transitions dictated by the 50 dB stopband attenuation below 1 Hz without affecting the frequency components around 1 kHz, a 114-tap symmetric FIR filter structure is used. This filter also operates at the reduced 1 MSPS / R = 62.5 kHz frequency.

The *CIC interpolator* at the last filter stage is a low hardware cost implementation of an interpolator and an anti-imaging filter. It uses 3 stages to upsample the signal by a factor of $R = 16$ and, therefore, to restore it to the original 1 MSPS sampling rate.

ADPLL module

The detailed block diagram of the ADPLL module is shown in Figure 8. The ADPLL is continuously fed through the REF FREQ signal with the output of the Signal Shaping module at a rate of 1 MSPS. However, the samples of the $f_2 - f_1 = 1$ kHz input signal are ignored until the filter

transients decay and the ADPLL is enabled. Once enabled, it uses the input as a reference to adjust the frequency and phase of an internal digitally-controlled oscillator (DCO). The DCO uses the *Phase register* as an accumulator to store its actual phase with 20-bit precision and to increment it by the content of the Δ-*Phase register* in every clock cycle. The actual phase, the value of the *Phase register*, is continuously monitored by a *DCO freq counter* to compare the frequency of the internal DCO and the reference signal. Comparison is done by counting the number of DCO signal periods that occur under one period of the reference signal. The counter value is connected to the *Zero comparator*, which detects the occurrence of too many or too few DCO periods, indicating that the DCO frequency is respectively too high or too low. Based on the *Zero comparator* output, the *Phase register* control adjusts the content of the Δ-*Phase register*, which is done in a Successive Approximation Register (SAR) fashion. SAR calibration means that the appropriate Δ-*Phase register* value is searched by successively adding to it or subtracting from it a logarithmically decreasing value in every reference signal period. Therefore, in the case of our N = 20-bit wide Δ-*Phase register*, it takes N-1 = 19 steps to set the phase step value accurately. The final value of the Δ-*Phase register* corresponds to approximately 1 kHz. Hence, by using this a priori information, only a small 2^p-size ($p = 12$) subspace, that corresponds to the 0 Hz to 4 kHz range, has to be searched during the successive approximation procedure (compared to the original 2^N possible values). Synchronization of the DCO signal to the reference signal is achieved by resetting both the *DCO freq counter* and the *Phase register* on the rising edge of the reference signal.

Once the DCO frequency is set by the above method, a fine tuning mechanism provided by the *Phase comparator* takes place. Based on the actual value of the *Phase register* at the time of the rising edge of the reference signal, the *Phase comparator* tells if the DCO still leads or lags, (i.e., too fast or too slow, respectively). One LSB is added to the Δ-*Phase register* when lead is indicated and one is subtracted when lag is detected.

Recorder module

The *Recorder module* in the upper branch in Figure 7 provides an alternative data path for the received samples. First, IF input samples are buffered into the external pSRAM memory in a circular buffer. While continuously recording, a simple threshold-based state machine watches for a trigger event to occur. Once the trigger conditions are met, a number of additional samples are stored in the pSRAM. After that, the buffer content is accessible through the USB module.

PicoBlaze processor

The *PicoBlaze* soft-core processor shown in Figure 7 is a simple controller used to deal with low-level operations such as setting and reading register values of the different modules while presenting a relatively high-level interface through the USB controller. For that, the *PicoBlaze* core accepts text commands received through a (virtual) serial COM port. These include automated parameter setting for more complex operations (e.g. frequency setting of the CC1000 radio module) and direct read/write access of both the FPGA registers and the CC1000 registers.

USB module

The *USB controller* module is the driver for the FTDI USB chip found on the sensor board. It provides a convenient interface for other modules of the design including the *PicoBlaze processor* and the *Recorder module* to communicate through the USB port via a Virtual COM Port. The *Recorder module* is always granted access first in order to achieve high transfer rates while the *PicoBlaze processor* has a lower priority. The latter is connected to the *USB controller* module to send and receive parameter values and to report phase estimation results.

5. RESULTS AND CONTRIBUTION

In our experimental setup there were two transmitters and one receiver node. Transmitter SB nodes were replaced with a single software-defined radio (SDR) [10, 11] equipped with two radio front-ends in order to synthesize interfering radio signals at 433 MHz and 433 MHz + 1 kHz with precise frequency separation. Since the SDR operates from a single clock source (single oscillator) only, the frequency uncertainty originating from differences and instability in the internal oscillator of the separate nodes were eliminated. On the receiver side, we used the SB shown in Figure 5 to record the IF signal and evaluate the estimation accuracy offline.

Evaluating a single measurement, the spectra of the original IF signal and that at the output of the different filter stages are shown in Figure 9 (a). The spectrum of the IF signal show significant components around DC and somewhat below 160 kHz. The two dominant frequency components at 156 kHz and 157 kHz correspond to the received two interfering sinusoidal signals, whereas the side lobes ± 12 kHz apart from these are likely to occur due to the spurious harmonic content already present in the received RF signal and due to the non-linearity added by the radio front-end. When multiplying the signal by itself, the sum and difference frequencies appear in the spectra as reflected in Figure 9 (b). The sum frequencies are largely attenuated by the CIC decimation filter, while the components of the difference frequencies, around 1 kHz and DC, remain undamped. There are, however, components that transformed into the 0 Hz to 1/R MSPS (62.5 kHz) spectrum due to the decimation. The low-pass filter clearly suppresses these over 2 kHz and the high-pass filter removes the DC-content as shown in Figure 9 (d) and (e). As a result, the only remaining component is the 1 kHz envelope signal that is restored to the original 1 MSPS sampling rate by the CIC interpolator filter.

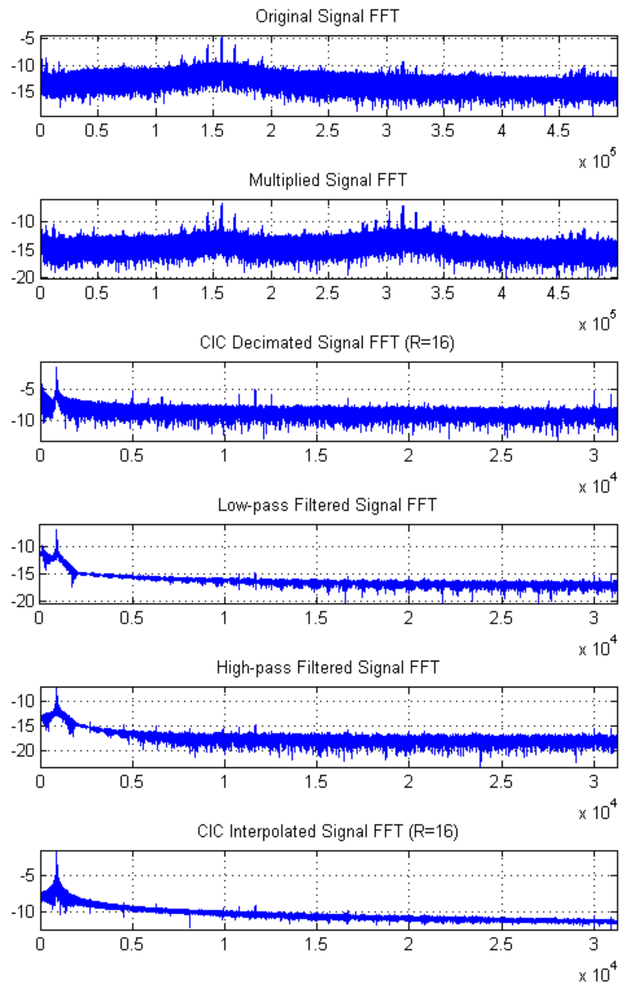

Figure 9: Offline evaluated spectrum of the received IF signal (a), IF signal square (b), after decimation by a factor R = 16 (c), low-pass filtering (d), high-pass filtering (e) and interpolation (f)

Phase estimation accuracy

Accounting for imprecision in the frequency of the transmitted signals, and hence in the interference frequency, the performance of the *RIPS Phase Estimator* algorithm was evaluated over a large range of frequencies around 1 kHz. First, the phase estimate output of the *RIPS Phase Estimator* was calculated for a given recording using fixed-point simulations. This result was then compared to a reference phase, defined as the phase corresponding to the interference frequency in the high resolution DFT of the filtered IF signal. Repeating the experiment 50 times at the same interference frequency, we determined the average phase estimation error over the 100 Hz – 2100 Hz frequency range. The results summarized in Figure 10 indicate a robust operation in the 400 Hz – 1500 Hz range with an average error of less than 0.03 radian (approximately 2 degrees) on each tested frequency. That corresponds to an accuracy that outperforms the former RSSI-based RIPS algorithm by more than 60%.

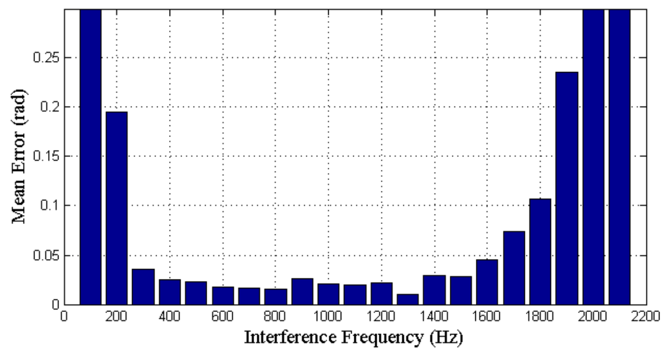

Figure 10: Mean deviation of phase measurements at different interference frequencies

Phase estimation speed

When measured from the turn-on time of the IF signal, the processing time of the phase estimation algorithm consists of two dominant parts. The first one is the signal shaping transient length that mainly originates from the delays introduced by the filter structure. Since only CIC and FIR filters are used in the design, a finite upper bound can easily be obtained by summing the individual filter impulse-responses. The latencies caused by the multiplier and the CIC filters are on the order of microseconds and are therefore negligible compared to the FIR filters operating at the relatively low 62.5 kHz sampling rate. The 72-tap low-pass filter and the 114-tap high-pass filter add 1.152 ms and 1.824 ms latencies, respectively, resulting in a total latency of approximately 3 ms for the signal shaping module.

The second part is the time associated with the ADPLL lock-in time. The modified architecture of the ADPLL searches for the correct Δ-*Phase register* value only in the range of 0 to 2^p, where p = 12. Setting each bit in the Δ-*Phase register* requires one period of the filtered interference signal, thus resulting in an upper bound of 12 times the lowest interference frequency expected. Assuming that the two received RF signals are at least 500 Hz apart, the maximum ADPLL lock-in time can be estimated to take 24 ms. That is, the overall processing time is 27 ms when measured from the start of a transient (e.g. start of reception) and neither the filter variables are properly filled nor the ADPLL has acquired the lock yet. However, after the ADPLL locked in, the phase information of the DCO, and therefore, that of the input signal, is updated and available in every 1 MSPS clock cycle. Also, measurements on several adjacent channels, needed by RIPS for phase disambiguation, can be carried out much faster if the radio signals are not turned off completely between channel hopping. We have yet to investigate the exact gain in these situations.

Hardware resource requirement

Timing reports showed that the current design is capable of running at the targeted 20 MHz system clock frequency. A summary of resource utilization is shown in Table 1, and indicates moderate usage of the resources available in the XC3S1000 Spartan-3L FPGA. The overall design uses one quarter of the Flip Flops and 4-input LUTs available in the FPGA. Clearly, the majority of the Flip Flops are utilized by FIR filters as storage elements. Two thirds of the LUTs in the design are also harnessed to the FIR filters found in the phase estimator module, typically as distributed memory for FIR filter coefficients. Since coefficients are stored in LUTs, only one block-RAM is used in the design, namely to store the program code of the PicoBlaze processor. Table 1 also shows that three hardware multipliers were utilized. One of the three is used to multiply the input signal by itself, while the other two are exploited by the FIR filters, respectively.

	Available	Total Design	RIPS Phase Estimator
Flip Flops	15,360	4,055 (26%)	3,637 (23%)
4-input LUTs	15,360	3,079 (22%)	2,262 (15%)
Block RAMs	24	1 (4%)	0 (0%)
Multipliers	24	3 (12%)	3 (12%)

Table 1: Device Utilization Summary

6. Conclusion and Future Work

The characteristics of ad-hoc WSNs require us to look at more than the pure signal processing problem itself when it comes to distributed real world applications. Additional steps include the configuration and maintenance of the wireless network topology, time-synchronization and localization of the individual nodes. We contributed to the node localization problem by providing a hardware-based solution that reduces the average error of the phase estimation step of the RIPS algorithm by 60%. We expect a speedup and/or accuracy improvement in the overall RIPS localization process; however, these metrics still have to be evaluated.

The FPGA-based prototype realization and the corresponding utilization data suggest a feasible hardware-oriented implementation. We recommend the inclusion of such phase estimator logic in low-cost radio chips (e.g. the CC1000) to support RIPS-based precise node localization on WSN nodes, without causing significant overhead in the processing unit. Even though the solution presented could already be implemented in radio chips, there is still room for improvement. Analog filter solutions should also be examined, while in the case of the same digital filter stage approach, the impact on the accuracy of filter word length reductions should be investigated. The processing time is a parameter our solution did not improve significantly. However, since the algorithm spends the bulk of its time waiting for the ADPLL to lock, replacing the SAR approach with a faster converging one could speed up the phase estimation process drastically.

Taking a look at the big picture, the RIPS localization algorithm heavily relies on the presence of an accurate time-synchronization algorithm in the WSN. Hardware support for clock synchronization would also considerably reduce the workload on the processing unit and increase the synchronization accuracy.

Acknowledgements

This work was supported in part by NSF grant CNS-0721604, ARO MURI Grant W911NF-06-1-0076 and US Army SBIR Grant W15QKN-08-C-0440.

References

[1] Ledeczi, A., Nadas, A., Volgyesi, P., Balogh, G., Kusy, B., Sallai, J., Pap, G., Dora, S., Molnar, K., Maroti, M., et al., "Countersniper System for Urban Warfare", *ACM Transactions on Sensor Networks*, vol. 1, pp. 153--177, November, 2005.

[2] Kushwaha, M., Amundson, I., Volgyesi, P., Ahammad, P., Simon, G., Koutsoukos, X., Ledeczi, A., and Sastry, S., "Multi-modal Target Tracking using Heterogeneous Sensor Networks", *17th International Conference on Computer Communications and Networks (ICCCN 08)*, St. Thomas, Virgin Islands (USA), 2008.

[3] Kusy B. "Spatiotemporal Coordination in Wireless Sensor Networks", Nashville, TN, USA, August 2007.

[4] Maroti, M., Kusy, B, Balogh, G., Volgyesi, P., Molnar, K., Nadas, A., Dora, S., and Ledeczi, A., "Radio Interferometric Geolocation", *ACM Third International Conference on Embedded Networked Sensor Systems (SenSys 05)*, San Diego, CA, pp. 1--12, November, 2005.

[5] Crossbow, "MICA, Wireless Measurement System Datasheet", http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf

[6] Texas Instruments, "CC1000: Single Chip Very Low Power RF Transceiver", http://focus.ti.com/lit/ds/symlink/cc1000.pdf

[7] Maroti, M., Kusy, B., Simon, G., and Ledeczi, A. "The flooding time synchronization protocol". In *Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems* (Baltimore, MD, USA, November 03 - 05, 2004). SenSys '04. ACM, New York, NY, 39-49.

[8] Walters, S.M.; Troudet, T., "Digital phase-locked loop with jitter bounded," *IEEE Transactions on Circuits and System*, vol.36, no.7, pp.980-987, Jul 1989

[9] Hogenauer, E., "An economical class of digital filters for decimation and interpolation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol.29, no.2, pp. 155-162, Apr 1981

[10] Ettus Research LLC, "USRP motherboard datasheet", http://www.ettus.com/downloads/er_ds_usrp_v5b.pdf

[11] "GNU Radio", http://gnuradio.org

BIOGRAPHY

Sándor Szilvási is currently a Ph.D. student at Vanderbilt University and a Research Assistant at the Institute for Software Integrated Systems at Vanderbilt University. His current research interests include wireless sensor networks and low-power FPGA design. He received his M.Sc. in Electrical Engineering from the Budapest University of technology and Economics, Budapest, Hungary, in 2008. Contact him at sandor.szilvasi@vanderbilt.edu.

János Sallai is an Academic Researcher at Institute for Software Integrated Systems, Vanderbilt University. His research interests include execution environments and languages for wireless sensor networks, and distributed acoustic beamforming. He received his Ph.D. in Computer Science from Vanderbilt University and his M.Sc. in Computer Science from the Budapest University of Technology and Economics. Contact him at janos.sallai@vanderbilt.edu.

Isaac Amundson is a Ph.D. student at Vanderbilt University and a Research Assistant at the Institute for Software Integrated Systems. His research focuses on methods related to spatio-temporal coordination in mobile wireless sensor networks. He received MS degrees in Mechanical Engineering and Computer Science, both from Vanderbilt University. Contact him at isaac.amundson@vanderbilt.edu.

Péter Völgyesi is a Research Scientist at the Institute for Software Integrated Systems, Vanderbilt University. His current research interests include wireless sensor networks and domain specific modeling environments. He received an M.Sc. in Computer Science from the Budapest University of Technology and Economics. Contact him at peter.volgyesi@vanderbilt.edu.

Ákos Lédeczi is an Associate Professor in Vanderbilt University's Department of Electrical Engineering and Computer Science and a Senior Research Scientist at its Institute for Software Integrated Systems. His research interests include model-integrated system development and wireless sensor networks. Lédeczi has a PhD in electrical engineering from Vanderbilt University. Contact him at akos.ledeczi@vanderbilt.edu.