Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2013-009 May 22, 2013

BigBand: GHz-Wide Sensing and Decoding on Commodity Radios

Haitham Hassanieh, Lixin Shi, Omid Abari, Ezzeldine Hamed, and Dina Katabi

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

BigBand: GHz-Wide Sensing and Decoding on Commodity Radios

Haitham Hassanieh Lixin Shi Omid Abari Ezzeldine Hamed Dina Katabi
Massachusetts Institute of Technology
{haitham, lixin, abari, ezz, dina}@csail.mit.edu

Abstract– The goal of this paper is to make sensing and decoding GHz of spectrum simple, cheap, and low power. Our thesis is simple: if we can build a technology that captures GHz of spectrum using commodity Wi-Fi radios, it will have the right cost and power budget to enable a variety of new applications such as GHz-wide dynamic access and concurrent decoding of diverse technologies. This vision will change today's situation where only expensive power-hungry spectrum analyzers can capture GHz-wide spectrum.

Towards this goal, the paper harnesses the sparse Fourier transform to compute the frequency representation of a *sparse* signal without sampling it at full bandwidth. The paper makes the following contributions. First, it presents BigBand, a receiver that can sense and decode a sparse spectrum wider than its own digital bandwidth. Second, it builds a prototype of its design using 3 USRPs that each samples the spectrum at 50 MHz, producing a device that captures 0.9 GHz — *i.e.*, 6× larger bandwidth than the three USRPs combined. Finally, it extends its algorithm to enable spectrum sensing in scenarios where the spectrum is not sparse.

Keywords Spectrum Sensing, Sparse Fourier Transform, Wireless, ADC, Software Radios

1. INTRODUCTION

The rising popularity of wireless communication and the potential of a spectrum shortage have motivated the FCC to take steps towards releasing multiple bands for dynamic spectrum sharing [9]. Last July, the President's Council of Advisors on Science and Technology (PCAST) recommended the immediate release of 100 MHz of spectrum for sharing, and advocated a plan for further releasing one GHz of government-held spectrum [26]. Within just a few months, the FCC began the process of opening up 100 MHz between 3.5-3.6 GHz [8]. Dynamic sharing is a key pillar of the FCC's vision for these new spectrum bands, and is motivated by the fact that actual utilization of the spectrum is sparse in practice. For instance, Fig. 1 from the Microsoft Spectrum Observatory [20] shows that, even in urban areas, large swaths of the spectrum remain underutilized. The 2012 PCAST report advocates dynamic sharing of much of the currently under-utilized spectrum, creating GHz-wide spectrum superhighways "that can be shared by many different types of wireless services, just as vehicles share a superhighway by moving from one lane to another."

Motivated by this vision, this paper explores the potential for GHz-wide spectrum sensing and reception on low-power inexpensive devices. Making GHz-wide sensing (*i.e.* the ability to detect occupancy) and reception (*i.e.* the ability to decode) available on commodity radios enables multiple applications:

- **Realtime Spectrum Monitoring:** Cheap GHz sensing enables spreading thousands of small devices in a metropolitan area for large-scale realtime spectrum monitoring. Today, the only way to monitor GHz of spectrum in realtime is to use expensive, power hungry spectrum analyzers. Commercial devices rely on sequential sensing, hopping from one channel to the next, acquiring only tens of MHz at a time [24, 23]. Sequentially scanning one GHz of spectrum means each band is monitored for only 1% to 2% of the time, and hence it is fairly easy to miss short lived signals (*e.g.*, radar).

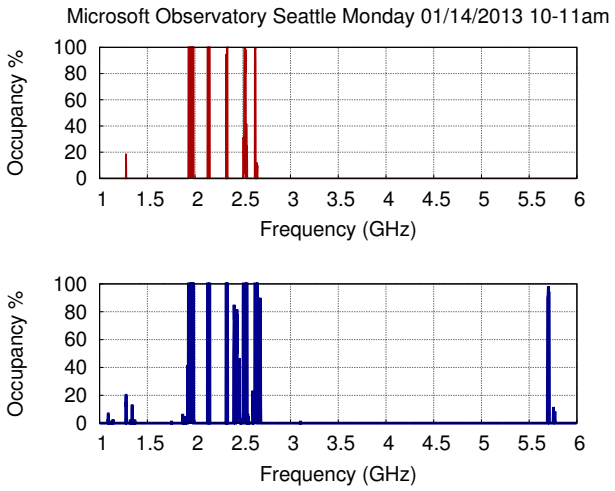


Figure 1—Spectrum Occupancy: The figure shows the average (top) and maximum (bottom) spectrum occupancy at the Microsoft spectrum observatory in Seattle on Monday January 14, 2013 during the hour between 10 am and 11 am. The figure shows that between 1 GHz and 6 GHz, the spectrum is sparsely occupied.

- **GHz-wide Dynamic Access:** Realtime GHz sensing enables truly dynamic spectrum access, where secondary users can detect short spectrum vacancies and leverage them, increasing overall spectrum efficiency [4].

- **Concurrent Decoding of Diverse Technologies:** Beyond sensing, the ability to decode signals in a GHz-wide spectrum on low-power cheap devices can enable new forms of communications. A single receiver may decode many concurrent transmissions occurring simultaneously in diverse parts of the spectrum. For example, a GHz receiver can concurrently receive Bluetooth at 2.4 GHz, GSM at 1.9 GHz, and CDMA at 1.7 GHz. Alternatively, it may receive Wi-Fi at 5 GHz and WiMax at 5.8 GHz. Ideally, it would do so with the same cost and power consumption of a narrowband Wi-Fi receiver.

But how hard is it to build a GHz receiver? The key difficulty in providing low-power cheap GHz sensing or receiving stems from the need for very high-speed ADCs, which are both costly and power hungry. To acquire GHz of bandwidth, the ADC needs a sampling rate higher than Giga sample per second (GS/s). An off-the-shelf 1 GS/s ADC costs 100's of dollars and consumes more than 2 watts [25, 6]. In contrast, a 50 MS/s ADC, like in Wi-Fi re-

ceivers, costs about $2, and consumes an order of magnitude less power [6].

Our goal is to build a technology that uses the same hardware as a Wi-Fi radio, which typically captures only tens of MHz of digital bandwidth, and adapt it to capture a GHz-wide bandwidth. Given the size, power, and cost of Wi-Fi hardware, such a technology can enable GHz sensing and reception capabilities for small embedded and mobile devices.

To achieve our goal, we harness recent advances in sparse recovery, which permit signals whose frequency domain representation is sparse to be recovered using only a small subset of their samples. One may use compressive sensing to acquire GHz of sparsely utilized spectrum without sampling at GS/s [16, 22, 27]. Compressive sensing however does not work with low-power commodity hardware because it requires custom hardware that can perform complex analog matrix multiplications and analog mixing at GHz speeds. As a result, compressive sensing may consume as much power as (and sometimes more than) high-speed ADCs [2, 1]. In contrast, we exploit the sparse FFT algorithm [13, 12, 11], which both provides sparse recovery and outputs the frequency domain signal, eliminating the need for additional processing.

Contributions: This paper makes contributions in the following two areas:

GHz-wide Sensing: The paper introduces BigBand, a technology that can sense GHz of spectrum, using a few (3 or 4) off-the-shelf low-speed ADCs. Furthermore, it can do so whether the spectrum is sparse or not. As such, the paper makes two contributions in the sensing domain. First, it introduces a new sparse FFT algorithm tailored for spectrum acquisition. Specifically, past sparse FFT algorithms use a sub-sampling pattern that picks samples that are spaced by the inverse of the signal bandwidth. Thus, applying those algorithms to spectrum sensing would still require a high-speed ADC that samples the signal at GS/s. Instead, BigBand introduces a new sparse FFT algorithm that uses only uniform samples obtained from a few low-speed ADCs. We analytically prove that by using low-speed ADCs whose sampling rates are co-prime, BigBand achieves the same running time as the original sparse FFT, and uses the same number of samples in expectation.

Our second sensing contribution extends BigBand to deal with scenarios in which the spectrum is not sparse. The basic idea is simple: instead of taking the FFT over the time signal, we take the FFT over changes in the time signal. Since only a small fraction of the spectrum is likely to change its occupancy over short intervals of a few milliseconds, the FFT of "changes" is sparse and we can apply our algorithm to it.[1]

GHz-wide Receiving: BigBand can do more than spectrum sensing – the action of detecting occupied bands. It can also decode the signal. BigBand presents the first receiver that decodes a *sparse* signal whose bandwidth is wider than its own digital bandwidth, using commodity low-rate ADCs, without using any high speed sampling or mixing. This is in contrast to recent attempts to build sparse recovery receivers using compressive sensing, which need custom ADCs with complex analog hardware and GHz analog mixing [27, 16].

Implementation and Results: We have built a working prototype of BigBand using USRP radios. Our prototype uses three USRPs, each of which can capture 50 MHz bandwidth to produce a device that captures 0.9 GHz –*i.e.*, 6× larger bandwidth than the digital bandwidth of the three USRPs combined. An empirical evaluation of this prototype provides the following results:

- BigBand can sense a sparse 0.9 GHz frequency band in real time. It can identify occupied frequencies with an error rate less than 2% for SNRs larger than 3 dB, and an error rate less than 0.5% for SNR larger than 10 dB. For sparsity of 5%, its false positive rate is 2% and its false negative rate is 0.2%.
- We use BigBand to sense the spectrum between 2 GHz and 2.9 GHz, a one-GHz stretch used by diverse technologies [20]. Our outdoor measurements reveal that, in our metropolitan area,[2] the above band has an occupancy of 2–5%. These results are in sync with similar measurements conducted at other locations [20].
- BigBand's extended version can identify changes in occupancy of non-sparse spectrum. For any spectrum occupancy up to 95%, BigBand can discover the changes in spectrum occupancy and find unoccupied frequencies with less than 1% false negatives and 2% false positives, as long as at most 1% of the spectrum changes its occupancy every millisecond.
- BigBand can correctly decode sparse signals in a 0.9 GHz band. Specifically, it can decode up to 30 transmitters that are simultaneously frequency hopping in a 900 MHz band with less than 3.5% packet loss.

2. RELATED WORK

Related work falls in the following areas.

Spectrum Sensing: Most of the earlier work on spectrum sensing focuses on narrowband sensing [30, 3, 21]. Narrowband sensing techniques include detecting the signal's energy [3], its waveform [30], its cyclostationality [14], or its power variation [21]. In contrast, our work focuses on wideband spectrum sensing, where the challenge is the need for high speed ADCs.

A recent work on wideband sensing called QuickSense [29] recognizes it is inefficient to sequentially scan a wideband. To speed up the scanning process, QuickSense moves the scanning to the analog domain using cheap analog filters and energy detectors. It then uses a hierarchical search algorithm to minimize the number of scans. BigBand differs from QuickSense in two main ways: First, BigBand can decode the signal (obtain the I and Q components) as opposed to only detecting spectrum occupancy. Second, for highly utilized spectrum (*i.e.* not sparse), QuickSense converges to sequentially scanning the spectrum whereas BigBand's differential algorithm provides a fast sensing mechanism for non-sparse spectrum.

BigBand also complements the geo-location database required by the FCC for identifying the bands occupied by primary users (e.g., the TV stations in the white spaces). The database, however, has no information about frequencies occupied by secondary and unlicensed users in the area. Also, due to the complexity of predicting propagation models, the database provides only long-term predictions, and can be inaccurate, particularly with dynamic access patterns [9, 4].

Also related to our work are past measurement studies of spectrum occupancy [20, 19, 5]. The resulting data reveals that apart from the bands below 1 GHz and few bands around 2.4 GHz, the spectrum is sparsely utilized. Bands proposed for re-purposing and spectrum sharing are typically highly under-utilized, like those around 3.5 GHz, above 4.2 GHz, and between 1675 MHz and 1850 MHz [8, 26]. Despite these attempts at measuring the spectrum, the data is fairly scarce. In an attempt to address the problem, a past proposal advocated that researchers in universities and research labs volunteer time-slots on their spectrum analyzers, which could be coordinated and used for real-time spectrum monitoring [15]. BigBand shares the objective of enabling large scale spec-

[1]The above gives the intuition. However, technically, we compute changes in the signal power, not the actual signal(see §5 for details).

[2]Place name is removed for anonymity.

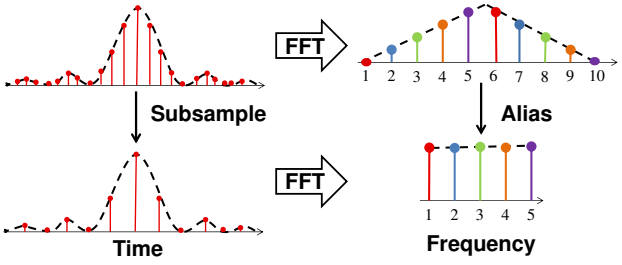


Figure 2—The correspondence of sub-sampling and aliasing: Sub-sampling the time domain signal in the top left to half the number of samples results in the signal in the bottom left. In the Fourier domain, the FFT of the sub-sampled signal is an aliased (folded) version of the FFT of the initial signal; namely, samples 1 and 6 in the top right signal add into sample 1 in the aliased signal in the bottom right, samples 2 and 7 into sample 2, etc.

trum measurements. However it addresses the issue by making GHz sensing cheaper and more accessible.

Sparse Recovery: The closest solutions to our work are wideband sparse recovery techniques based on compressive sensing [16, 22, 27, 28]. However, since compressive sensing requires random projections, these techniques end up using complex analog hardware to avoid using an ADC that samples at Nyquist rate. This includes custom hardware that can perform analog matrix multiplication and analog mixing at Nyquist rates [16, 27]. Further, some of these hardware implementations end up consuming as much power as an ADC that samples at Nyquist rate [2, 1].

Finally, our work builds on earlier theoretical work on sparse Fourier transform [13, 12, 11]. However, as explained in §1, the original sparse FFT algorithms require random sampling and are not suitable for cheap low power spectrum sensing and signal recovery.

3. ILLUSTRATIVE EXAMPLES

We start with two illustrative examples that give an intuition of how BigBand's sparse FFT algorithm works. In these examples and throughout the paper, we will refer to the *value* of a frequency by $\mathbf{X}_f$, and its *position* in the spectrum by f. Also, for clarity, in these examples we assume the value of unused frequencies is zero, *i.e.*, we ignore the noise (Our results in §8 naturally include signal noise). We can then refer to the used frequencies as the non-zero frequencies.

Before introducing our sampling algorithm, we remind the reader of a basic property of the Fourier transform that we rely on in our design: *Sub-sampling a signal in the time domain causes aliasing in the frequency domain.* Fig. 2 illustrates this property.

3.1 One Non-Zero Frequency

Let us consider a very simple case where we have a signal of size N but only one frequency f has a non-zero value $\mathbf{X}_f$, as shown in Fig. 3(a). For simplicity, we chose $N = 28$ and $f = 11$. In general, to compute the frequency representation of this signal, one would take an FFT over N time samples –*i.e.*, one needs an ADC that can take $N = 28$ samples per time unit.[3] Say however that we are allowed only a low-speed ADC that takes 4 samples every time unit. How can we correctly compute the full spectrum of size $N = 28$?

[3]Throughout this paper when we refer to a sample, we mean a complex sample that is both I and Q. Thus, the Nyquist criterion implies that a bandwidth of N Hz requires N complex samples per second (real and imaginary samples).

The low-speed ADC *sub-samples* the signal in the time domain. As described earlier, this causes aliasing in the frequency domain [18]. We will refer to aliasing as *Bucketization*, since taking the FFT over the 4 time samples returned by our low-speed ADC causes the 28 frequencies to hash into 4 buckets, such that the value of each bucket is the sum of the 7 frequencies that hash to that bucket, *i.e.*, frequency f hashes to bucket $i = f \bmod 4$, as shown in Fig. 3(b).

Now, lets try to reconstruct the 28-point spectrum from the 4 buckets. Non-zero frequency $f = 11$ hashes to bucket $i = 11 \bmod 4 = 3$, and hence only this bucket will have a non-zero value as shown in Fig. 3(b). Further, the value of this bucket $\mathbf{b}_i$ will be equal to the value of the non-zero frequency $\mathbf{X}_f$ since all other frequencies that hash to this bucket are zero. Thus, by computing the values of 4 buckets, we can find the value of the non-zero frequency.

Although bucketization allows us to find the value of the non-zero frequency, we still do not know its frequency position f, since there are multiple frequencies mapped to the same bucket. To compute f, we leverage the *phase-rotation property* of the Fourier transform, which states that a shift in time translates into phase rotation in the frequency domain [18]. Specifically, say that we repeat the whole process of bucketization, after shifting the input signal by τ samples. Then, the phase of $\mathbf{X}_f$, and consequently the phase of the bucket it hashes to, is going to change by:

$$\Delta\phi = \frac{2\pi \cdot f \cdot \tau}{N}, \text{ and hence } f = \frac{\Delta\phi \cdot N}{2\pi\tau}. \quad (1)$$

Thus, we can figure out the position f of the non-zero frequency by looking at how much its value rotates after a time shift as shown in Fig. 3(c). We refer to the process of finding the positions of non-zero frequencies as the *Estimation* step.

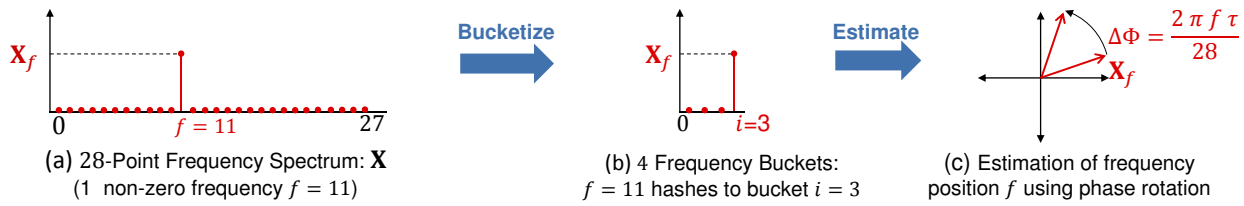
The example above outlines the basic ideas underlying BigBand's approach for computing a wideband sparse spectrum using low-speed ADCs. Namely, we alias the spectrum into a small number of buckets, ignore the empty buckets (buckets whose value is close to zero) and then estimate the frequencies in the non-empty buckets by exploiting the phase rotation rule in Eq. 1. The above approach works if we have no collisions, *i.e.*, no two non-zero frequencies fall into the same bucket. The next example provides the basic idea for resolving collisions.

3.2 Three Non-Zero Frequencies

Let us now consider a slightly more complex case, where we have three non-zero frequencies, $f_1 = 11$, $f_2 = 19$, and $f_3 = 25$, as shown in Fig. 4. In this case, if we perform the above bucketization, frequencies f_1 and f_2 will hash to the same bucket since $11 \bmod 4 = 19 \bmod 4 = 3$. We refer to this as a *collision* of non-zero frequencies. A collision prevents us from finding the value of each of the non-zero frequencies. It also prevents us from estimating the positions of the two frequencies f_1 and f_2 since the phase rotation of the bucket is no longer proportional to f_1 or f_2. Of course, we can still find the position and value of f_3 using the above method because this frequency does not suffer from a collision.[4] However to reconstruct the full spectrum, we need to resolve the collision. So, how can we resolve collisions?

To resolve the collision, we need to repeat the bucketization in a way that guarantees that the colliding frequencies do not collide again. Say that we are given a second low-speed ADC, which takes 7 samples per time unit. We can repeat the above bucketization but

[4]Note that we need to be able to detect which buckets have a collision and which don't so that we can estimate the frequencies that do not collide. In §4.3, we describe how to detect collisions.

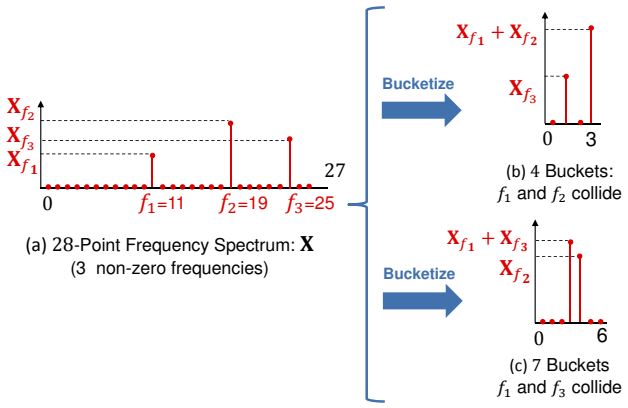


(a) 28-Point Frequency Spectrum: $\mathbf{X}$ (1 non-zero frequency $f = 11$)

(b) 4 Frequency Buckets: $f = 11$ hashes to bucket $i = 3$

(c) Estimation of frequency position f using phase rotation

Figure 3—Estimating one non-zero frequency: (a) Sub-sampling the time signal using a low rate ADC to get 4 samples and taking the 4-point FFT bucketizes the 28 frequencies to 4 buckets. (b) Non-zero frequency 11 is hashed to bucket $3 = 11 \bmod 4$ which allows us to estimate its value $\mathbf{X}_f$ (c) Repeating the bucketization with a time shift τ, rotates the phase of $\mathbf{X}_f$ by $2\pi f\tau/N$ which allows us to estimate f.



(a) 28-Point Frequency Spectrum: $\mathbf{X}$ (3 non-zero frequencies)

(b) 4 Buckets: f_1 and f_2 collide

(c) 7 Buckets f_1 and f_3 collide

Figure 4—Estimating 3 non-zero frequencies: (a) Frequencies f_1, f_2, f_3 are occupied. (b) Hashing into 4 buckets results in f_1 and f_2 colliding the same bucket which prevents us from estimating their values and positions. We can estimate f_3. (c) Hashing into 7 buckets, f_1 and f_3 collide but not f_2. We can estimate f_2 and subtract it from the bucket where it collided with f_1 which allows us to estimate f_1.

this time we bucketize into 7 buckets and a frequency f is hashed into the bucket $f \bmod 7$. In this case, non-zero frequency $f_1 = 11$ will hash to bucket 4, $f_2 = 19$ to bucket 5, and $f_3 = 25$ to bucket 4. This time, f_1 and f_3 collide, but f_2 does not collide, as shown in Fig. 4(c).

Now, we have two sets of buckets (shown in the second column of Fig. 4), which are the 4 buckets generated by taking an FFT over the output of the first low-speed ADC, and the 7 buckets generated by taking an FFT over the output of the second low-speed ADC. Each set of buckets has a collision. Yet together the two sets of buckets can be used to resolve both collisions. Specifically, we compute the value and position of frequency f_3 from the first bucketization, where it does not collide (using the same approach we used above when we had only one non-zero frequency). Similarly, we compute the value and position of frequency f_2 from the second bucketization, where it does not collide. After resolving f_2, we go back to the first bucketization and subtract its value $\mathbf{X}_{f_2}$ from the bucket 3 where it collides.[5] This leaves only frequency f_1 in bucket 3, which can now be resolved. Thus, the combination of the two bucketizations using two low-speed ADCs allows us to reconstruct the full spectrum.

But how do we guarantee that the same pair of frequencies that collided in the first bucketization does not collide again in the second bucketization? We can do so because the numbers of buckets across bucketizations (4 and 7) are co-prime. We know from modular arithmetic that for any two integers f_1 and f_2, we have, $f_1 \bmod 7 = f_2 \bmod 7$ and $f_1 \bmod 4 = f_2 \bmod 4$ if and only if $f_1 \bmod 28 = f_2 \bmod 28$. Hence, using these co-prime bucketizations, two distinct frequencies in an N-wide spectrum will never collide twice.

These examples give an intuition of how we can find the values and positions of non-zero frequencies. In the next section, we generalize these ideas to any number of non-zero frequencies and show how these ideas can be implemented efficiently on off-the-shelf hardware.

[5]Note that we subtract $\mathbf{X}_{f_2}$ from the bucket and $\mathbf{X}_{f_2} e^{i2\pi f_2 \tau/N}$ from the time shifted version of the bucket.

4. BIGBAND

BigBand is a receiver that can capture a sparse spectrum wider than its own bandwidth, *i.e.*, it can recover a sparse signal with a significantly lower sampling rate than the Nyquist criterion. Thus, BigBand can do more than spectrum sensing – the action of detecting occupied bands. BigBand provides the details of the signals in those bands (I's and Q's of wireless symbols), which enables decoding those signals.

BigBand presents a new sparse FFT algorithm tailored for spectrum acquisition using low speed ADCs. In this section, we describe in details BigBand's sparse FFT algorithm and in §6 we outline its similarities and differences to the original sparse FFT algorithm.

At a high-level, BigBand operates in two key steps: *bucketization* and *estimation*. In the *bucketization* step, BigBand hashes the frequencies in the spectrum into buckets. Since the spectrum is sparse, many buckets will be empty and can be discarded. BigBand then focuses on the non-empty buckets, and computes the values of the frequencies in those buckets in what we call the *estimation* step. Below we describe both steps in detail.

4.1 Frequency Bucketization with Co-prime Aliasing

Bucketization has to satisfy the following requirements:

1. It needs to hash the frequencies into buckets, *i.e.*, every bucket has the same number of frequencies, every frequency falls in a unique bucket, and the value of the bucket is the sum of the values of frequencies that hash to it.
2. It should admit sub-Nyquist sampling, *i.e.*, it should operate on a small number of time samples, such that the number of samples per second is proportional to the number of occupied frequencies not the total bandwidth.
3. It should be possible to implement sub-sampling with purely low-rate ADCs.
4. It should be possible to repeat the bucketization but with different frequencies sharing the same bucket so that we can resolve collisions.

BigBand uses a bucketization scheme based on co-prime aliasing filters which satisfy the above requirements. Below we explain how aliasing filters satisfy requirements 1, 2, 3 and how making the filters co-prime satisfies requirement 4.

So what are aliasing filters? Recall the following basic property of the Fourier transform: *sub-sampling in the time domain causes*

aliasing in the frequency domain. Formally, let $\mathbf{x}$ be a discrete time signal of length N, and $\mathbf{X}$ its frequency representation. Let $\mathbf{x}'$ be a subsampled version of $\mathbf{x}$, where $\mathbf{x}'_i = \mathbf{x}_{i \times N/B}$ and B divides N. Then, $\mathbf{X}'$, the FFT of $\mathbf{x}'$ is an aliased version of $\mathbf{X}$, *i.e.*:

$$\mathbf{X}'_f = \sum_{j=0}^{N/B-1} \mathbf{X}_{f+jB}. \quad (2)$$

Thus, aliasing is a form of bucketization in which frequencies equally spaced by an interval B end up in the same bucket, *i.e.*, frequency f will hash to bucket $i = f \mod B$. Further, the value in each bucket is the sum of the values of the frequencies that hash to the bucket as shown in Eq. 2.

Aliasing directly satisfies requirements 1, 2, and 3. The only tricky part is to satisfy requirement 4, which translates to identifying different aliasing filters that randomize how frequencies hash into buckets. To do so, we use aliasing filters with different sampling intervals. In this case, each bucketization requires subsampling at a different rate, which can be accomplished with multiple low-rate ADCs.

So how should we choose the different sampling intervals of the aliasing filters? As we have seen in the example in §3.2, choosing sampling intervals which are co-prime (4 and 7) randomizes the bucketization and prevents the same frequencies from colliding in both filters. Therefore, the best choice is co-prime aliasing filters. Said differently, the filters have $B_1 = N/p_1$ and $B_2 = N/p_2$ buckets where p_1 and p_2 are co-prime. In the Appendix, we prove the following lemma:

LEMMA 4.1. *Given two aliasing filters with $B_1 = N/p_1$ and $B_2 = N/p_2$ buckets such that p_1 and p_2 are co-prime integers that divide N, then for any two frequencies $f \neq f'$, we have: $f' = f \mod B_1 \rightarrow f' \neq f \mod B_2$.*

The lemma states that given the above aliasing filters, any two frequencies that collide in the first bucketization will not collide in the second bucketization and hence this choice of bucketization satisfies requirement number 4. Hence, co-prime aliasing filters satisfy our four requirements.

Two important points are worth clarifying:

- The number of frequencies that hash to each bucket needs to be co-prime and not the total number of buckets, *i.e.* p_1 and p_2 must be co-prime but not necessarily B_1 and B_2. In the example in §3.2, it happens that $B_1 = 4$ and $B_2 = 7$ are co-prime and $p_1 = 7$ and $p_2 = 4$ are also co-prime since $N = 28$.
- How does this translate into ADC sampling rates? The best choice of aliasing filters suggests that for a bandwidth BW, we should use two ADCs that sample at rates BW/p_1 and BW/p_2 where p_1 and p_2 are co-prime. Of course, ADCs might be not readily available at any sampling rate. However, one can always find a variety of off-the-shelf ADCs that can recover a bandwidth slightly higher but close enough to the desired bandwidth. For example, to recover a 1 GHz bandwidth, we can use a 42 MHz ADC [6] along with a 50 MHz ADC. The combination of these two ADCs can capture a bandwidth of 1.05 GHz. This is because 42 MHz $=$ 1.05 GHz$/25$ and 50 MHz $=$ 1.05 GHz$/21$ where 21 and 25 are co-prime.

4.2 Frequency Estimation with Phase Rotation

The bucketization step allows us to separate the occupied frequencies into their own buckets with the potential of some buckets having frequency collisions. In the next section, we will present a mechanism to detect buckets with collisions. For the time being, let us focus on the buckets that do not have collisions and estimate the value and the position of the occupied frequency, *i.e.*, $\mathbf{X}_f$ and the corresponding f.

Term	Definition
BW	total GHz bandwidth we wish to reconstruct
T	total sampling time of the signal, FFT window time
N	size of the FFT, $N = T \times BW$
K	sparsity : number of non-zero frequency coefficients
B	number of buckets
p	number of frequencies that hash to a bucket, $p = N/B$
f	frequency index ($0 \leq f < N$)
τ	time shift of the signal in number of samples
$\mathbf{x}$	time signal of length N sampled at a rate of BW
$\mathbf{X}$	frequency domain signal, $\mathbf{X}$ = FFT($\mathbf{x}$)

Table 1—Terms used in the description of BigBand.

In the absence of a collision, the value of the occupied frequency is the value of the bucket. Since many frequencies fall into the bucket, it is not clear which frequency f is associated with this value. However, as explained in the example in §3.1, we can estimate the position of the frequency using phase rotation. Specifically, we repeat the bucketization after a time shift τ. Since a shift in time translates into phase rotation in the frequency domain, the value of the bucket of interest changes from $\mathbf{X}_f$ to $\mathbf{X}_f \cdot e^{\mathrm{i}2\pi \cdot f \cdot \tau / N}$. Hence, using the change in the phase of the bucket, we can estimate our frequency of interest and we can do this for all buckets that do not have collisions. This implies that for each of the two co-prime sampling rates, the system needs to use two ADCs one of which is sampling after a time shift of τ, *i.e.* BigBand uses 4 ADCs in total. Note however that the two co-prime ADCs and their shifted versions need not have the same shift τ.

To be able to implement the above frequency estimation, we need to answer the following two questions:

1. How can we sample the signal with a shift? This is fairly simple as we can connect the antenna to the two ADCs using different delay lines (which is what we do in our implementation). Alternatively, we can use different delay lines to connect the clock to the two versions of the ADC.

2. What values of τ are suitable? It is important to note that not all values of τ will allow us to uniquely estimate multiple frequency positions. This is because the phase wraps around every 2π. For example, say that we use a shift of $\tau = 2$ samples out of N where N is the size of the sparse FFT, and consider two frequencies f and $f' = f + N/2$. After a shift by τ, the phase rotation of f is $\Delta\phi(f) = 2\pi \cdot f \cdot 2/N$. The phase rotation of f' is $\Delta\phi(f') = 2\pi \cdot (f + N/2) \cdot 2/N \mod 2\pi = 2\pi \cdot f \cdot 2/N$. Thus, with a time shift of 2 samples, the phase shifts observed for two frequencies f and f' separated by $N/2$ are the same, and BigBand will be unable to disambiguate between them. BigBand can use a shift of $\tau = 3$ to disambiguate between f and $f + N/2$, but this does not address the situation completely since a shift of $\tau = 3$ will be unable to disambiguate frequencies separated by $N/3$. In general, we need to pick a τ that gives a unique mapping between the phase rotation and the frequencies, independent of the separation between the frequencies. Formally, for all separations s between the frequencies ($1 \leq s \leq N$-1), we need to ensure that $s\tau/N$ is not an integer. We can ensure this property for either τ =1, or any τ invertible modulo N.

4.3 Detecting Frequency Collisions

So far we have assumed that we know which buckets have a single occupied frequency and which buckets have a collision. However, we need to be able to detect collisions in order to avoid estimation errors.

1 Pseudocode for BigBand

```
PROCEDURE: BIGBAND(x)
X ← 0
B_1 ← N/p_1
B_2 ← N/p_2
▷ Bucketization: FFT of sub-sampled and shifted signal
b^1 ← FFT(x[0], x[p_1], ··· , x[N − p_1])
b^2 ← FFT(x[0], x[p_2], ··· , x[N − p_2])
b̃^1 ← FFT(x[τ_1], x[τ_1 + p_1], ··· , x[τ_1 + N − p_1])
b̃^2 ← FFT(x[τ_2], x[τ_2 + p_2], ··· , x[τ_2 + N − p_2])
▷ Estimation: Iterate between filters
repeat
  for u ∈ {1, 2} do
    for non-empty b^u_i do
      if no collision then
        f ← (∠b̃^u_i − ∠b^u_i) · N/(2π · τ_u)
        X_f ← b^u_i
        Subtract X_f from b^1, b̃^1, b^2, b̃^2
until all buckets are empty or log(K) iterations
return X
```

BigBand uses the *phase rotation* property of the Fourier transform to determine if a collision has occurred. Specifically, if there is no collision and the only occupied frequency is f, then the values $\mathbf{b}$ and $\mathbf{b}^{(\tau)}$ of a bucket in the two time-shifted bucketizations are related as $\mathbf{b}^{(\tau)} = \mathbf{b}e^{\mathrm{i}2\pi\cdot f\tau/N}$. In particular, these values only differ by a phase shift, and their magnitudes are equal. On the other hand, consider the case where there is a collision between, say, two frequencies f and f'. Then the value, $\mathbf{b}$, in the bucket before time-shifting can be written as $\mathbf{X}_f + \mathbf{X}_{f'}$. After time-shifting by τ, the value of the bucket, $\mathbf{b}^{(\tau)} = \mathbf{X}_f \cdot e^{\mathrm{i}2\pi\cdot f\tau/N} + \mathbf{X}_{f'} \cdot e^{\mathrm{i}2\pi\cdot f'\tau/N}$. As described in §4.2, the two phase shifts for f and f' are different by choice of τ, and hence the magnitudes of $\mathbf{b}$ and $\mathbf{b}^{(\tau)}$ are different. Thus, we can determine whether there is a collision or not by comparing the magnitudes of the buckets before and after time-shifting, and verifying whether they are the same or not.

4.4 BigBand's Sparse FFT Algorithm

To put the pieces together, Algorithm 1 provides a pseudocode for BigBand's sparse FFT algorithm. BigBand proceeds as follows: after bucketization, it estimates the occupied frequencies that did not collide in the first bucketization. It then subtracts the values of these frequencies from the buckets they hashed to in all 4 bucketizations and estimates the remaining frequencies from the second bucketization. BigBand iterates between the two bucketizations until all frequencies have been recovered. In the Appendix, we prove the following theorem about the algorithm.

THEOREM 4.2. *For sparsity $K = c\sqrt{N}$ and $p_1, p_2 = \Theta(\sqrt{N})$, BIGBAND runs in time $O(K \log N)$, uses $O(K)$ samples and returns the correct result with probability at least $1 - O(\alpha)$ for some small enough constants α and c.*

4.5 Sparsity Range

Since BigBand is a sparse FFT algorithm, it is natural to ask what sparsity range it works for. BigBand uses only two aliasing filters, there is a small probability that it fails to resolve collisions, and this limits the sparsity that it can handle.

BigBand will fail to resolve a collision when there is a deadlock, *i.e.*, during the estimation step in Algorithm 1, it fails to find any non-empty bucket without a collision. For example, say we have four frequencies (f_1, f_2, f_3, f_4) such that in the first aliasing filter f_1 collides with f_2 and f_3 collides with f_4 whereas in the second aliasing filter f_1 collides with f_3 and f_2 collides with f_4. Then, we will not be able to resolve these collisions. The probability that a deadlock occurs depends on how sparse the spectrum is.

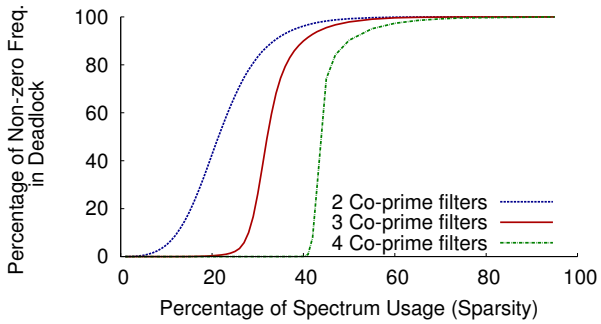


Figure 5—BigBand's Sparsity Range: We ran a simulation to check the percentage of frequencies which are in a deadlock and hence will not be recovered by BigBand versus the sparsity. The figure shows that with each additional co-prime filter we can significantly reduce the frequencies in deadlock and increase the sparsity range for which BigBand can recover all frequencies.

In order to support a denser spectrum, we need to add a third aliasing filter that is co-prime with the first two. This will allow us to resolve deadlocks of size 4. However, with 3 aliasing filters, one can have deadlocks of size 8 or larger, and more generally, with m aliasing filters, one can have deadlocks of size 2^m or greater. Thus, intuitively, the likelihood of a deadlock reduces with the number of co-prime filters, as the deadlock needs to involve exponentially more frequencies.

Fig. 5 shows the results of a simulation that reports the fraction of occupied frequencies in a deadlock as a function of the sparsity of the spectrum for two, three or four co-prime aliasing filters. As the figure shows, for a fixed number of aliasing filters, increasing the sparsity reduces the likelihood that the occupied frequencies are in a deadlock. The figure also shows that each additional co-prime aliasing filter can significantly reduce the number of frequencies in a deadlock and allow BigBand to support higher spectrum usage.

5. SENSING NON-SPARSE SPECTRUM

In this section, we extend BigBand's algorithm to deal with sensing a non-sparse spectrum. The key idea is that although the spectrum might not be sparse, the changes in spectrum usage are sparse i.e. over short intervals, only few frequencies are freed up or become occupied. We refer to this as differential sparsity. To see how differential sparsity allows D-BigBand to sense a non-sparse spectrum we will start with an example.

5.1 Illustrative Example

In this example, we are going to assume that the state of any frequency can either be occupied or empty. However, if a frequency is occupied, its value does not change over time. We will later explain how to deal with the fact that values of occupied frequencies change over time. Let us consider the case where one frequency $f = 12$ which was occupied becomes unoccupied after time TW as shown in Fig. 6(a,b). Now if we bucketize the spectrum, all buckets will be non-empty and will have collisions. Hence, we cannot use the previous algorithm. However, since frequency $f = 12$ became empty after time TW, the power in the bucket it hashes to will become lower after time TW. Further, since it is the only frequency that changed state, only the power of that bucket changes. Hence if we subtract the bucketization at time TW from that at time 0,

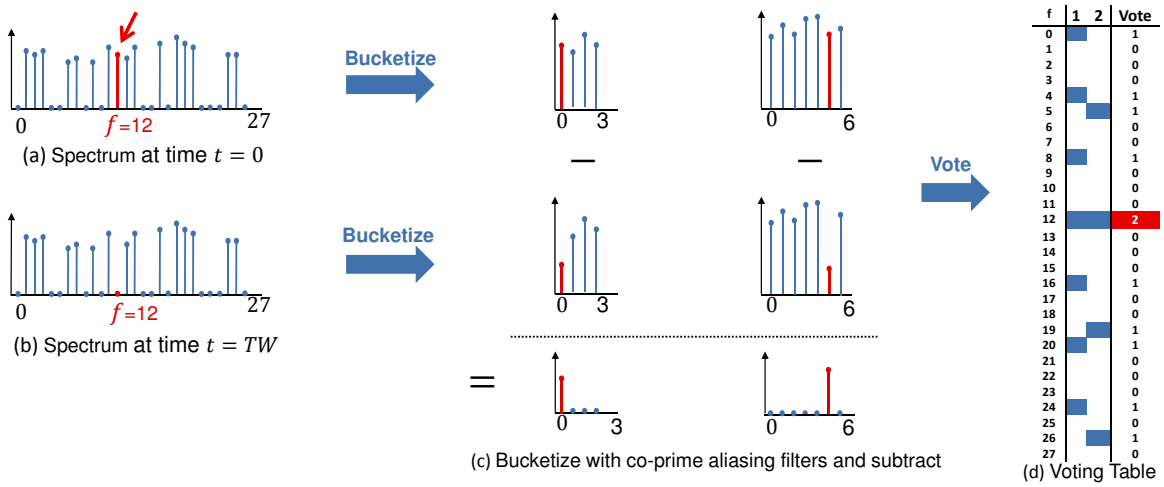


Figure 6—Sensing one change in non-sparse spectrum: (a) f=12 is occupied at $t = 0$. (b) f=12 is empty at $t = TW$. (c) Bucketize the spectrum at $t = 0$ and $t = TW$ using co-prime aliasing filters and subtract the two bucketizations to discover changing buckets. Changes are sparse. (d) Each co-prime filter votes for the frequencies that hash to a changing bucket. Only f=12 gets two votes.

we can find which buckets have frequencies that changed state as shown in Fig. 6(c).

Subtracting the bucketizations, allowed us to bucketize the "changes" in the spectrum. However, we still need to estimate which frequency is the one that changed state out of the frequencies that hash to the bucket. To do this, we introduce a new estimation procedure based on voting and co-prime aliasing filters. Both at time 0 and time TW, we perform two bucketizations; one using an aliasing filter with four buckets and another using an aliasing filter with seven buckets as shown in Fig. 6(c). Now every frequency that is hashed into a bucket that changed gets a vote. However, since the filters are co-prime, frequencies that hash to the same bucket as f in the first filter and get a vote, will hash to a different bucket in the second filter and will not get a second vote. Hence, only frequency $f = 12$ will get two votes which allows us to estimate its position as shown in Fig. 6(d).

The above example gives an intuition of how we can leverage the sparsity of changes in the spectrum to discover which frequencies become occupied and which become empty. However, to be able to generalize the above approach, we need to first address the following issues:

- Since the values of the occupied frequencies change after a time TW, the values of the buckets will change even if the state of the frequencies that hash to them did not change. Hence, we cannot simply subtract the two bucketizations. However, since FCC typically requires wireless transmissions to be whitened over time, the average power of a bucket will not change if the state of frequencies that hash to it does not change. To estimate the average power over a time window TW, D-BigBand performs the bucketization multiple times and averages the power of the buckets. The longer the time window TW, the better the estimate of the average power of each bucket. However, the longer the time window, the more frequencies change their state. In §8, we show that a time window $TW = 1$ ms allows us to properly detect changes in the buckets.

- If there is more than one change in the spectrum, we will need to use more than two co-prime aliasing filters. For example, 4 filters allow D-BigBand to support a differential sparsity of $K_d = o(\sqrt{N})$ where K_d is the number of frequencies whose state has changed.[6]

[6]This is because, given four aliasing filters with number of buckets $N/p_1, N/p_2, N/p_3, N/p_4$ where p_1, p_2, p_3, p_4 are co-prime, the probability that voting makes a mistake is bounded by K_d^4/N^2.

5.2 D-BigBand

D-BigBand's algorithm works as follows. Over a time window TW, D-BigBand bucketizes the signal multiple times[7] for each of the four co-prime aliasing filters and calculates the average power in the bucket over this time window. It then repeats these bucketizations over the next time window and subtracts the average power of the buckets in the first time window from that in the second time window. After that each filter votes for frequencies that hash to buckets where the power changed. Frequencies that get four votes are picked as the frequencies whose state has changed. Hence, based on our knowledge of the spectrum occupancy during the first time window, we can discover the spectrum occupancy during the second time window.

As with any differential system, we need to initialize the state of spectrum occupancy. However, an interesting property of D-BigBand is that we can initialize the occupancy of each frequency in the spectrum to unknown. This is because, when we take the difference in power we can tell whether the frequency became occupied or it became empty. Hence, once the occupancy of a frequency changes, we can tell its current state irrespective of its previous state. This avoids the need for initialization and prevents error propagation.

6. BIGBAND VS SFFT

In this section, we describe the differences between BigBand and the sparse FFT algorithm (sFFT) in [12, 13].

BigBand is designed and proved for the typical case of spectrum usage where the occupied frequencies are randomly distributed (with sparsity $K = O(\sqrt{N})$), whereas sFFT is proved for a worst case distribution of occupied frequencies (with sparsity $K = o(N)$).

Since BigBand is designed under fewer constraints than sFFT, it can be implemented much more efficiently than sFFT. Most importantly, BigBand works with off-the-shelf low speed ADCs. In contrast, sFFT, similar to compressed sensing, requires custom ADCs that can randomly sub-sample the signal with inter-sample spacing as small as the inverse of the signal bandwidth. Additionally, BigBand performs the bucketization step only 4 times, whereas sFFT needs to perform $O(\log K)$ bucketizations. Finally, BigBand's dif-

[7]The number of times D-BigBand can average is = TW/T where T is the FFT window time.

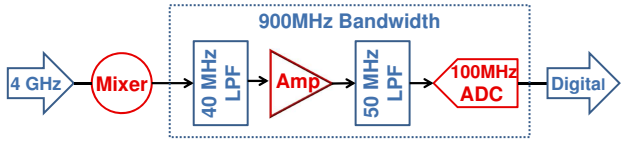


Figure 7—SBX Daughterboard Schematic: The board can tune to any frequency between 0.4 GHz to 4.4 GHz. After down-conversion, the signal passes through a 40 MHz low pass filter (LPF), an amplifier, and a 50 MHz LPF before the 100 MHz ADC. The baseband circuit bandwidth is about 900 MHz. BigBand bypasses both the 40 and 50 MHz LPFs to allow the baseband circuitry to receive 900 MHz.

ferential scheme, D-BigBand, enables the detection of occupied and empty frequencies for any level of spectrum usage, whereas sFFT is designed for a sparse spectrum.

7. A USRP-Based Implementation

We build a prototype of BigBand using USRP software radios [7]. We use three USRP N210 radios with the SBX daughterboards, which can operate in the 400 MHz to 4.4 GHz range. The clocks of the three USRPs are synchronized using an external GPSDO clock [17]. In order to sample the same signal using the the three USRPs, we connect the USRPs to the same antenna using a power splitter.

To be able to implement BigBand however, we had to address the following USRP limitations:

- **RF Frontend:** The RF frontend of the SBX daughterboard is designed to provide 40 MHz of bandwidth to a low rate ADC. However, the goal of BigBand is to use RF frontends that can pass a much larger bandwidth to the low rate ADC. We achieve this by modifying the SBX RF receive chain, whose architecture is shown in the schematic in Fig. 7. In particular, we bypass the 40 MHz and 50 MHz filters shown in the schematic. This allows the USRP's ADC to receive the entire bandwidth that its analog front-end circuitry is designed for. The ADC circuitry can receive at most 0.9 GHz. Once we bypass the filters, BigBand can use the SBX to sense 900 MHz, which will be aliased to the 50 MHz bandwidth of the USRPs.
- **Sampling rate:** The USRP ADC has a sampling rate of 100 MHz. However, the USRP digital processing chain cannot support 100 MS/s; the highest sampling rate it can support is 50 MS/s.[8] Further, the USRP has digital filters but these can only produce sampling rates which are integer dividers of 100 MS/s (*i.e.* 100/2, 100/3, 100/4, *etc.*). Hence, for 0.9 GHz bandwidth, it is not possible with USRPs to get two aliasing filters that sample at $0.9/p_1$ and $0.9/p_2$ where p_1 and p_2 are co-prime. We can implement the co-prime aliasing filters using commodity ADCs [6] as explained in §4.1. However, this would require building a new receiver that uses these ADCs. Instead, we implement BigBand using three USRPs, all of which use 50 MS/s aliasing filters. Our implementation of BigBand is more constrained than our description in §4 since it does not incorporate co-prime aliasing filters. However, we can still use it to resolve some collisions as we describe below.

7.1 Resolving Collisions without Co-prime Filters

Ideally, co-prime filters will allow us to resolve collisions. However, three aliasing filters sampling at the same rate with different time shifts allows us to solve collisions of two frequencies. To see how, notice that in the 50 MHz aliasing filters in our implementation, 18 frequencies (900/50) will hash together in one bucket since we are sensing a 900 MHz spectrum. Thus, two occupied frequencies f and f' that collide in the same bucket can be one of $\binom{18}{2} = 153$ possibilities. For each of these possibilities, we have two unknowns which are the values ($\mathbf{X}_f$ and $\mathbf{X}_{f'}$) of the two frequencies. However, these values combine with a different phase rotation in each of the three filters to give us three different values of the same bucket:

$$\begin{aligned} \mathbf{b}_j^1 &= \mathbf{X}_f + \mathbf{X}_{f'} \\ \mathbf{b}_j^2 &= \mathbf{X}_f \cdot e^{\mathrm{i}2\pi f \tau_1/N} + \mathbf{X}_{f'} \cdot e^{\mathrm{i}2\pi f' \tau_1/N} \\ \mathbf{b}_j^3 &= \mathbf{X}_f \cdot e^{\mathrm{i}2\pi f \tau_2/N} + \mathbf{X}_{f'} \cdot e^{\mathrm{i}2\pi f' \tau_2/N} \end{aligned} \tag{3}$$

where τ_1 and τ_2 are the time shifts of the second and third filters relative to the first filter. Hence, for each possible pair (f, f'), we get an over-determined system with three linear equations and two unknowns ($\mathbf{X}_f$, $\mathbf{X}_{f'}$). This system will have a solution only for the correct pair. Hence, by testing all possibilities we can find the correct positions of f and f'.

The previous discussion assumes that only two frequencies collide in the two buckets. If more than 2 frequencies collide, the equations above are extremely unlikely to have any pairs (f, f') that will satisfy them. Thus, this system also allows us to check for collisions, similar to the BigBand scheme described in §4.3.[9]

Once we discover a collision of more than two frequencies which we cannot solve, we set all frequencies that hash to the bucket as occupied. This increases the number of false positive errors (*i.e.*, unoccupied frequencies which are reported as occupied) by at most 15 for each of these collisions. However, this avoids false negative errors (*i.e.*, occupied frequencies which are reported as unoccupied). In the context of spectrum sensing, false negatives are more problematic since they can result in interfering with ongoing transmissions.

7.2 Estimating the Channel and the Time Shifts

The earlier description of BigBand assumes that the values of the frequencies are scaled similarly on all three USRPs. Although the signals received at the three USRPs experience the same wireless channel since they come from the same antenna, they experience different channels on the hardware and hence they are scaled differently. Specifically, if an occupied frequency f whose value is $\mathbf{X}_f$ hashes to bucket j and does not collide, then the value of bucket j at each of the USRPs can be written as:

$$\begin{aligned} \mathbf{b}_j^1 &= h_w(f) \cdot h_1(f) \cdot \mathbf{X}_f \\ \mathbf{b}_j^2 &= h_w(f) \cdot h_2(f) \cdot \mathbf{X}_f \cdot e^{\mathrm{i}2\pi f \tau_1/N} \\ \mathbf{b}_j^3 &= h_w(f) \cdot h_3(f) \cdot \mathbf{X}_f \cdot e^{\mathrm{i}2\pi f \tau_2/N} \end{aligned} \tag{4}$$

where $h_w(f)$ is the wireless channel coefficient, $h_1(f), h_2(f), h_3(f)$ are the hardware channels on each of the USRPs, and $\cdot(f)$ indicates that these parameters are frequency dependent. $h_w(f)$ cancels out once we take the ratios, $\mathbf{b}_j^2/\mathbf{b}_j^1$ and $\mathbf{b}_j^3/\mathbf{b}_j^1$ of the buckets. However, the hardware channels are different and if we do not estimate and compensate for them, we cannot perform frequency estimation or detect collisions and solve them. In addition, we also need to estimate the time shifts τ_1, τ_2 in order to perform frequency estimation based on phase rotation.

[8] We use UHD to configure the USRP to transmit 16-bit ADC samples (8 bits each for I and Q) to the host so that we can receive 50 MS/s without saturating the Gigabit Ethernet.

[9] Again, similar to the scheme in §4.3, we can solve for collisions of three frequencies by adding a fourth filter. We will then have a system of four equations and three unknowns, and so on.

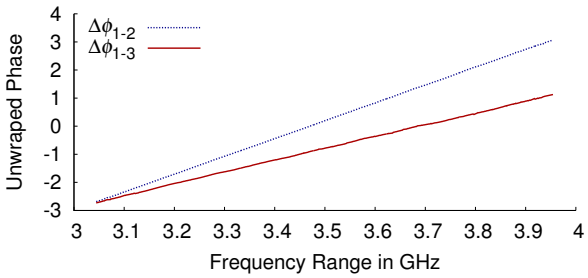


Figure 8—Phase rotation vs frequency: The figure shows that the phase rotation between the 3 USRPs is linear across the 900 MHz frequency spectrum and can be used to estimate the time shifts.

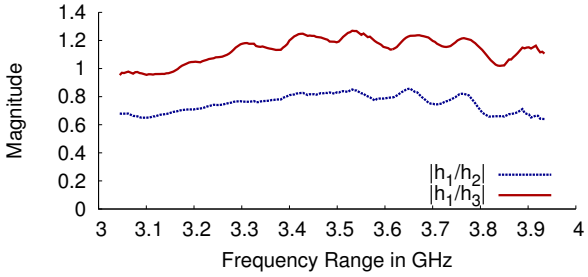


Figure 9—Hardware channel magnitude: The relative channel magnitudes $|h_1(f)/h_2(f)|$ and $|h_1(f)/h_3(f)|$ are not equal to 1 and are not flat across the frequency spectrum. Hence, we need to compensate for these estimates to be able to detect and solve collisions.

To estimate the channels and the time shifts, we divide the 900 MHz spectrum into 18 consecutive chunks of size 50 MHz. We then transmit a known signal in each chunk, one by one. Since we only transmit in one chunk at a time, there are no collisions at the receiver after aliasing. We then use Eq. 4 to estimate the ratios $h_2(f) \cdot e^{i2\pi f\tau_1/N}/h_1(f)$ and $h_3(f) \cdot e^{i2\pi f\tau_2/N}/h_1(f)$ for each f in the 900 MHz spectrum.

Now that we have the ratios, we need to compute $h_2(f)/h_1(f)$ for each frequency f, and the delay τ. We can estimate this as follows: Both the magnitude and phase of the hardware channel ratio will be different for different frequencies. The magnitude differs with frequency because different frequencies experience different attenuation in the hardware. The phase varies linearly with frequency because all frequencies experience the same delay τ, and the phase rotation of a frequency f is simply $2\pi f\tau/N$. We can therefore plot the phase of the ratio as a function of frequency, and compute the delay τ from the slope of the resulting line.

Fig. 8 shows the phase result of this estimation. As expected, the phase is linear across the entire 900 MHz. Hence, by fitting the points in Fig. 8 to a line we can estimate the shifts τ_1, τ_2 and the relative phases of the hardware channels. Fig. 9 also shows the relative magnitudes of the hardware channels on the USRPs (*i.e.* $|h_1(f)/h_2(f)|$ and $|h_1(f)/h_3(f)|$) over the 900 MHz between 3.05 GHz and 3.95 GHz. These hardware channels and time shifts are stable. We estimated them only once at the set up time of the implementation.

7.3 Implementing D-BigBand

D-BigBand's frequency estimation relies on different co-prime filters to vote on which frequency positions have changed occupancy and hence we cannot implement D-BigBand without co-prime ADCs. To verify that D-BigBand can sense a non-sparse spectrum, we use multiple USRPs sampling adjacent narrowband chunks to capture a full 1 GHz of spectrum. However, since our testbed has only 20 USRPs, we divide them into 10 receivers and 10 transmitters and capture 250 MHz at a time. We repeat this 4 times at center frequencies that are 250 MHz apart and stitch them together in the frequency domain to capture the full 1 GHz spectrum. We then perform the inverse FFT to obtain a time signal sampled at 1 GHz. We now subsample this time domain signal using co-prime aliasing filters with the following sampling rates: 1/21, 1/20, 1/23 GHz, and run D-BigBand on these subsampled versions of the signal.

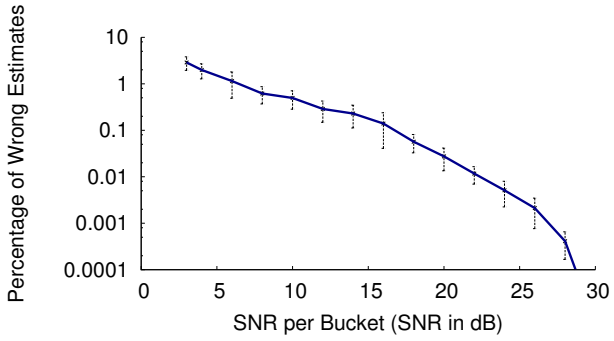


Figure 10—The accuracy of BigBand's frequency estimation: The error is less than than 2% for signals 3dB above the noise floor of each bucket. The error decreases to smaller than 0.5% if the SNR per bucket is larger than 10dB.

8. Empirical Results

In this section, we will evaluate the performance of BigBand and show that it can be used both for sensing and receiving (*i.e.*, decoding) sparse wideband signals. We also evaluate D-BigBand and show that it can be used for sensing even if the spectrum is not sparse.

8.1 Frequency Estimation as a Function of SNR

BigBand's basic primitive is the estimation of the frequency corresponding to a non-zero bucket by using the *phase rotation* property. Such an estimate of the phase is susceptible to sample noise. However, BigBand has two mechanisms that enhance its robustness to noise: averaging across samples obtained from multiple ADCs, and rounding the obtained frequency estimate to the nearest integer.[10] In this experiment, we verify the robustness of BigBand's frequency estimation as a function of SNR.

Method: We transmit signals in random frequency bins in the range 3.05-3.95 GHz. We set the sparsity to 1% and the FFT window to 1 ms. We vary the location of the receiver to get a range of SNR per bucket between 3 dB and 30 dB.

Results: Fig. 10 shows the percentage of frequencies that are estimated incorrectly as a function of the SNR in each bucket. The figure shows that the error is less than 2% for SNR larger than 3 dB and less than 0.5% for SNR larger than 10 dB. This shows that frequency estimation using phase rotation works across a large SNR range with little errors.

[10]Frequency f estimated in bucket i must satisfy $f = i \mod B$ where B is the number of buckets.

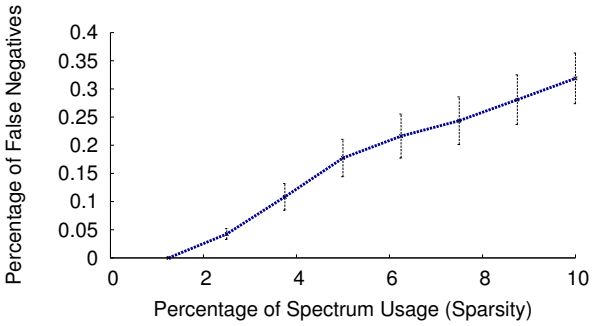


Figure 11—False negatives as a function of spectrum sparsity: BigBand has an extremely low rate of false negatives. Its false negative rate is less than 0.2% with less than 6% spectrum occupancy, and stays around 0.3% even when the spectrum occupancy grows as large as 10%.

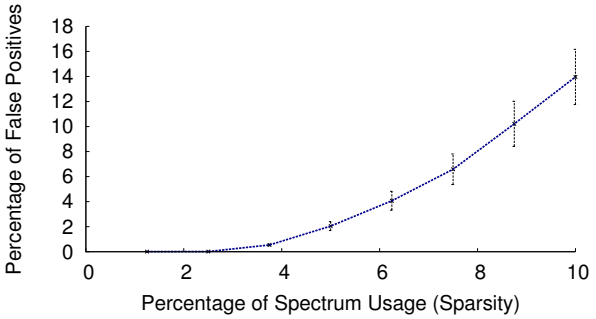


Figure 12—False positives as a function of spectrum sparsity: BigBand has a false positive rate of around 2% with 5% spectrum occupancy, and stays below 14% even when spectrum occupancy grows as large as 10%.

8.2 Evaluation of BigBand Spectrum Sensing

The primary motivation of BigBand is to be able to sense sparse spectrum. In this section, we verify the range of sparsity for which BigBand works.

Method: We vary the sparsity in the 3.05 GHz to 3.95 GHz range between 1% and 10% by transmitting from 5 different USRPs. Each USRP transmits a signal whose bandwidth is at least 1 MHz and at most 20 MHz. We randomize the bandwidth and the center frequencies of the signals transmitted by the USRPs. For each sparsity level, we repeat the experiment 100 times with different random choices of bandwidth and center frequencies. We run BigBand over a 1 ms FFT window. We use two metrics:

- **False Negatives**: The fraction of occupied frequencies that BigBand incorrectly reports as empty.
- **False Positives**: The fraction of empty frequencies that BigBand incorrectly reports as occupied.

Results: Fig. 11 shows that BigBand has an extremely low rate of false negatives, below 0.2% when the spectrum occupancy is less than 5%; it stays below 0.3% even when the spectrum occupancy goes up to 10%. The false negatives increase with spectrum occupancy since collision increases and it becomes more probable that BigBand fails to detect a collision. Compare this with today's sequential scanning techniques (e.g., RFeye [23]) which sense any particular frequency for only 2% of the time and hence do not measure that frequency for 98% of the time. As a result, they can miss a significant percent of occupied frequencies.

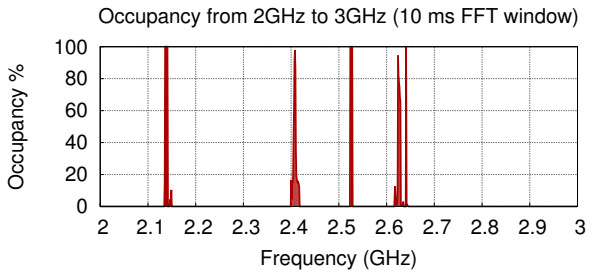


Figure 13—Spectrum Occupancy: The figure shows the average spectrum occupancy at our geographical location on Friday 01/15/2013 between 1-2pm:, as viewed at a 10 ms granularity. It shows that the spectrum is sparsely occupied.

Fig. 12 shows that the percentage of false positives of BigBand is less than 2% when the spectrum usage is below 5%. The number of false positives increases as the spectrum usage increases, but stays below 14% even for spectrum usage as large as 10%. BigBand's false positives increase as spectrum usage increases because it takes a conservative approach that errs in favor of false positives rather than false negatives. In particular, for each collision of 3 or more which BigBand cannot decode, it sets all 18 frequencies that hash to the bucket as occupied, which results in 15 additional false positives.

We note a few points: First, real-world spectrum measurements, for instance, in the Microsoft observatory, and in this paper, reveal that actual spectrum usage is 2–5%, in which regime BigBand's false positives would be less than 2%. Second, if the occupancy is high, causing the false positives to exceed the desired threshold, one may use D-BigBand, which deals with high occupancies (see results in §8.7.)

8.3 Outdoor Spectrum Measurements

This experiment shows that BigBand works in a real setting, in particular, measuring outdoor spectrum usage.

Method: We collect outdoor measurements in a metropolitan area from the roof top of a 24 floor building. We collect measurements between 2 GHz and 2.9 GHz. Measurements are collected using BigBand every 10 ms over a 30 min period, *i.e.*, we reconstruct the spectrum over an FFT window of 10 ms. We then calculate the percentage of 10 ms windows during which each frequency was occupied.

Results: Our results show that in our geographical area, between 2 GHz and 2.9 GHz, the spectrum usage is around 5%. These results were confirmed using a spectrum analyzer. Fig. 13 shows the fraction of time that each chunk of spectrum between 2 GHz and 2.9 GHz is occupied, as recovered by BigBand. The figure shows that the spectrum is sparsely occupied and that most of the occupied frequencies have 100% occupancy over the 30 min period, when viewed at a 10 ms granularity.

However, if we zoom in and perform the sparse FFT every 100 μs (or more frequently) instead of every 10 ms over the same period of 30 min, the spectrum occupancy changes. Table 2 examines this phenomenon further by showing the occupancy of some frequency bands for various FFT measurement windows. The occupancy of most frequencies drops, as compared to the 10 ms window. This shows that while these frequencies are occupied for some fraction of every 10 ms interval, there is a large number of shorter windows during these larger intervals where these frequencies are not occupied. This implies that the spectrum is sparser at finer time intervals,

FFT Window	2635-2640 MHz	2520-2530 MHz	2130-2140 MHz
10 μs	20%	64%	89%
100 μs	72%	78%	98%
1 ms	98%	87%	99%
10 ms	100%	100%	100%

Table 2—Occupancy vs FFT Measurement Window: Even frequencies that seem always occupied over longer measurement windows, are often likely to be detected as unoccupied when viewed over shorter windows. This motivates the need for real-time spectrum sensing to take advantage of short term vacancies.

FFT Window	BigBand (900 MHz)	3 USRP Seq. Scan (150 MHz)	RFeye Scan (20 MHz)
1 μs	1 μs	48 ms	22.5 ms
10 μs	10 μs	48 ms	22.5 ms
100 μs	100 μs	48 ms	—
1 ms	1 ms	54 ms	—
10 ms	10 ms	114 ms	—

Table 3—Scanning time: BigBand is multiple orders of magnitude faster than other technologies. This allows it to perform real-time sensing to take advantage of even short term spectrum vacancies.

and provides more opportunities for fine-grained spectrum reuse. Further, it motivates the need for fast spectrum sensing schemes to exploit these short-term vacancies.

8.4 BigBand vs. Spectrum Scanning

A key advantage of BigBand's ability to use low-speed ADCs for a wide band is that it can recover the band in one shot, and does not have to sequentially scan it in narrowband chunks. Hence, it reports spectrum occupancy in real time and does not miss spectrum chunks that are occupied only briefly. In this experiment, we compare the times taken by different techniques to capture a 0.9 GHz wide spectrum.

Method: Most of today's spectrum sensing equipment relies on scanning. Even expensive, power hungry spectrum analyzers typically capture a 100 MHz bandwidth in one shot, and end up scanning to capture a larger spectrum [24]. The performance of sequentially scanning the spectrum relies mainly on how fast the device can scan a GHz of bandwidth. Here, we compare how fast it would take to scan the 900 MHz bandwidth using three techniques: state-of-the-art spectrum monitors like the RFeye [23], which is used in the Microsoft spectrum observatory, 3 USRPs sequentially scanning the 900 MHz, or 3 USRPs using BigBand.

Results: Table 3 shows the results for different FFT window sizes. In all cases, BigBand takes exactly the time of the FFT window to acquire the 900 MHz spectrum. The 3 USRPs combined can scan 150 MHz at a time and hence need to scan 6 times to acquire the full 900 MHz. For FFT window sizes lower than 10 ms, the scanning time is about 48 ms. Hence, the USRPs spend very little time actually sensing the spectrum, which will lead to a lot of missed signals. Of course, state of the art spectrum monitors can do much better. The RFeye Node has a fast scanning mode of 40 GHz/second [23]. It scans in chunks of 20 MHz and therefore will take 22.5 ms to scan 900 MHz. Note that the RFeye has a maximum resolution bandwidth of 20 KHz, and hence cannot support any FFT windows larger than 50 μs.

Thus, in all cases, BigBand, which uses off-the-shelf components, is several orders of magnitude faster than even expensive scanning based solutions, allowing it to detect short-term spectrum vacancies.

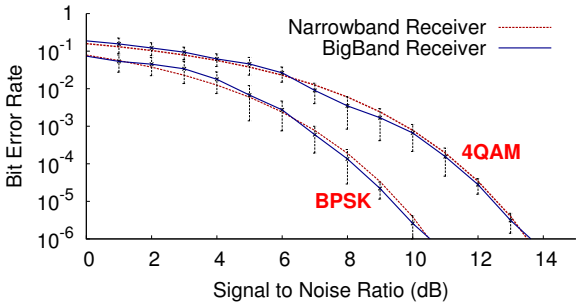


Figure 14—BigBand's Decoding Performance: BigBand's wideband receiver can decode sparse signals as efficiently as a narrowband receiver tuned to the transmitted signal across.

8.5 Decoding Performance as a Function of SNR

The key metric of a receiver is its decoding efficiency as a function of SNR. In this section, we compare the performance of BigBand's wideband receiver with a narrowband receiver that is tuned to the transmitter.

Method: We use our wideband receiver consisting of 3 USRPs that are all centered at 3.5 GHz and receive 50 MS/s. We transmit a sparse wideband signal by using 4 USRPs to transmit 4 20 MHz OFDM signals. The 4 transmitter USRPs are centered at the following frequencies: 3.215, 3.715, 3.44, and 3.84 GHz. Note that the total occupied bandwidth of the combined transmitted signal from all USRPs is 645 MHz. Similar to Wi-Fi, the transmitted OFDM symbols use 64 sub-carriers and a cyclic prefix of 16 samples.

Since each receiver USRP can sample a maximum of 50 MHz, the three receiver USRPs together cannot sense or decode the complete received signal in the absence of BigBand. With the BigBand receiver, the 645 MHz is aliased into the 50 MHz. We vary the location of the BigBand receiver to obtain different SNRs and in each location we transmit and decode 25×10^6 OFDM symbols. We compare the performance of BigBand with a traditional narrowband receiver that can decode the signals from a single narrowband transmitter.

Results: Fig. 14 shows the BER vs. SNR curve that BigBand achieves for both BPSK and 4-QAM modulation. The figure also shows the curve for a standard narrowband receiver with one transmitter. The BER vs SNR curve for BigBand matches that of the narrowband receiver. This shows that BigBand can decode wideband sparse signals at comparable performance to a traditional narrowband receiver.

8.6 Decoding Multiple Transmitters using BigBand

In this section, we verify that BigBand can decode a large number of transmitters. All the transmitters in our implementation use the same technology, but the result naturally generalizes to transmitters using different technologies.

Method: We use 10 USRPs to emulate up to 30 devices hopping in a spectrum of 0.9 GHz. At any given time instant, each device uses 1 MHz of spectrum to transmit a BPSK signal. Similar to the Bluetooth frequency hopping standard, we assume that there is a master that assigns a hopping sequence to each device that ensures that no two devices hop to the same frequency at the same time instant. Note however, that the hopping sequence for different devices might allow them to hop to frequencies that get aliased to the same bucket at a particular time instant, and hence collide in BigBand's

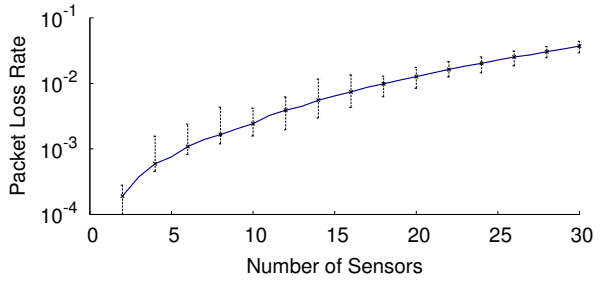


Figure 15—BigBand's Packet Loss as a function of the number of simultaneous transmitters: BigBand can decode as many as 30 simultaneous transmitters spread across a 900 MHz wide band, while keeping the packet loss less than 3.5%.

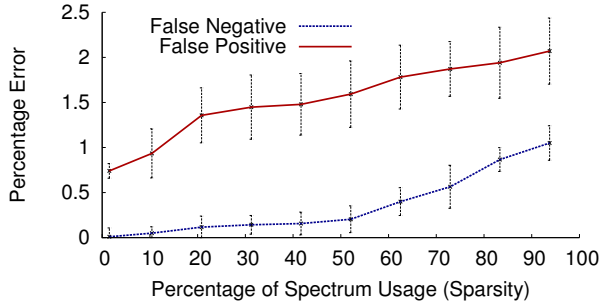


Figure 16—D-BigBand's effectiveness as a function of Spectrum Sparsity: Over a band of 1 GHz, D-BigBandcan reliably detect changes in spectrum occupancy even when the spectrum is 95% occupied, as long as the change in spectrum occupancy is less than 1% every ms.

50 MHz filter. Like in Bluetooth, each device hops 1, 3, or 5 times per packet, depending on the length of the packet.

Result: Fig. 15 shows the packet loss rate versus the number of devices hopping in the spectrum. The figure shows that BigBand can decode the packets from 30 devices spanning a bandwidth of 900 MHz with a packet loss rate less than 3.5%. Decoding all these sensors without using BigBand would either require a wideband 0.9 GHz receiver, or a receiver with 30 RF frontends both of which would be significantly more costly and power-hungry.

8.7 Evaluation of D-BigBand

In this section, we evaluate D-BigBand's ability to sense changes in spectrum occupancy independent of sparsity.

Method: We implement D-BigBandas described in §7.3. We vary the percentage of total occupied frequencies in the spectrum between 1% (sparse) to 95% (almost fully occupied). We then change the number of frequencies that change occupancy every 1 ms by up to 1% (*i.e.*, 10 MHz), and evaluate D-BigBand's accuracy in identifying the frequencies that change occupancy.

Results: As a function of spectrum occupancy, Fig. 16 shows the percentage of false positives (*i.e.*, frequencies whose occupancy has not changed, but BigBand erroneously declared as changed) and false negatives (*i.e.*, frequencies whose occupancy has changed, but BigBand erroneously declares as unchanged). We see that BigBand can robustly identify changes correctly even in a densely occupied network, with both false positives and false negatives remaining under 2% even at 95% occupancy.

9. Conclusion

This paper presents BigBand, a cheap system that enables GHz-wide sensing and decoding using off-the-shelf hardware. Empirical evaluation demonstrated that BigBand is able to sense the spectrum stably and dynamically under different sparsity levels; we also demonstrate BigBand's effectiveness as a receiver to decode GHz-wide sparse signals. We believe that BigBand enables multiple applications that would otherwise require expensive and power hungry devices, *e.g.* realtime spectrum monitoring, dynamic spectrum access, concurrent decoding of diverse techniques.

10. References

[1] O. Abari, F. Lim, F. Chen, and V. Stojanovic. Why analog-to-information converters suffer in high-bandwidth sparse signal applications. *IEEE Transactions on Circuits and Systems I*, 2013.

[2] O. Abari et al. Performance trade-offs and design limitations of analog-to-information converter front-ends. In *ICASSP*, 2012.

[3] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh. White space networking with wi-fi like connectivity. In *ACM SIGCOMM*, 2009.

[4] T. Baykas et al. Developing a standard for TV white space coexistence. *Wireless Comm, IEEE*, 19(1), 2012.

[5] D. Chen, S. Yin, Q. Zhang, M. Liu, and S. Li. Mining spectrum usage data: a large-scale spectrum measurement study. In *Mobicom*, 2009.

[6] DigiKey, ADCs. http://www.digikey.com/.

[7] Ettus. Inc. USRP. http://ettus.com.

[8] FCC: NPRM (FCC 12-148). http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-12-148A1.pdf.

[9] FCC, Second Memorandum Opinion & Order 10-174.

[10] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi. Sample-Optimal Average-Case Sparse Fourier Transform in Two Dimensions. *arXiv:1303.1209*, 2013.

[11] A. Gilbert, M. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal space fourier representations. In *SPIE*, 2005.

[12] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly optimal sparse fourier transform. In *STOC*, 2012.

[13] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Simple and practical algorithm for sparse FFT. In *SODA*, 2012.

[14] S. S. Hong and S. R. Katti. DOF: a local wireless information plane. In *ACM SIGCOMM*, 2011.

[15] A. P. Iyer, K. Chintalapudi, V. Navda, R. Ramjee, V. N. Padmanabhan, and C. R. Murthy. SpecNet: spectrum sensing sans frontières. In *NSDI*, 2011.

[16] J. Laska, W. Bradley, T. Rondeau, K. Nolan, B. Vigoda. Compressive sensing for dynamic spectrum access networks: Techniques tradeoffs. *DySPAN*, 2011.

[17] Jackson Labs, Fury GPSDO. http://jackson-labs.com/.

[18] R. Lyons. *Digital Signal Processing*. 1996.

[19] M. A. McHenry. NSF spectrum occupancy measurement project summary, 2005.

[20] Microsoft Spectrum Observatory. http://spectrum-observatory.cloudapp.net.

[21] H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat. Learning to Share: Narrowband-Friendly Wideband Networks. In *ACM SIGCOMM*, 2008.

[22] M. Rashidi, K. Haghighi, A. Panahi, and M. Viberg. A NLLS based sub-nyquist rate spectrum sensing for wideband cognitive radio. In *DySPAN*, 2011.

[23] RFeye Node. http://media.crfs.com/uploads/files/2/crfs-md00011-c00-rfeye-node.pdf.

[24] Tektronix Spectrum Analyzer. http://tek.com.
[25] Texas Instruments, "12-bit, 1000 MSPS ADC with analog input buffer.". http://www.ti.com/.
[26] PCAST: Realizing the full potential of government held spectrum to spur economic growth, 2012.
[27] J. Yoo, S. Becker, M. Loh, M. Monge, E. Candès, and A. E-Neyestanak. A 100MHz–2GHz 12.5x subNyquist rate receiver in 90nm CMOS. In *IEEE RFIC*, 2012.
[28] J. Yoo et al. A compressed sensing parameter extraction platform for radar pulse signal acquisition. *JETCAS*, 2012.
[29] S. Yoon, L. E. Li, S. Liew, R. R. Choudhury, K. Tan, and I. Rhee. Quicksense: Fast and energy-efficient channel sensing for dynamic spectrum access wireless networks. In *IEEE INFOCOM*, 2013.
[30] T. Yucek and H. Arslan. A survey of spectrum sensing algorithms for cognitive radio applications. *Communications Surveys Tutorials, IEEE*, 11(1), 2009.

APPENDIX

A. PROOFS

Preliminaries: We use $\mathbf{x}$ and $\mathbf{X}$ to denote the time signal and its frequency domain. In particular, we assume $\mathbf{X}$ has i.i.d. Bernoulli distribution where for each $i \in \{0,\ldots,N-1\}$, $\mathbf{X}_i \in \{0,a_i\}$ such that the sparsity is $\mathbb{E}[|\|\mathbf{X}\|_0)|] = K$. We also assume there exists two co-prime integers p_1 and p_2 that divide N such that $p_1 = \Theta(p_2)$ and $N/p_1 = O(K)$.

The proofs follow immediately from the proofs for the 2 dimensional sparse Fourier transform presented in [10]. Here, we will provide a proof for the noiseless case with $K = c\sqrt{N}$. The proofs for the noisy case and for $K < c\sqrt{N}$ can be found in [10].

LEMMA A.1. *Given two aliasing filters with $B_1 = N/p_1$ and $B_2 = N/p_2$ buckets such that p_1 and p_2 are co-prime integers that divide N, then for any two frequencies $f \neq f'$, we have: $f' = f \mod B_1 \rightarrow f' \neq f \mod B_2$.*

PROOF. Assume $f' \neq f \mod N$, but they are equal both modulo B_1 and B_2. Consequently, they are equal modulo the least common multiple: $\text{lcm}(B_1,B_2)$. Note that $\text{lcm}(B_1,B_2) = \text{lcm}(N/p_1,N/p_2) = N$ since p_1,p_2 are co-prime, which is a contradiction. □

LEMMA A.2. *The probability that any of the collision detection tests invoked by BigBand is incorrect is at most $O(1/N^{(a-5)/2})$ for some constant $a > 5$.*

PROOF. The proof of this lemma is found in [10] (proof of lemma 3.2). The idea is to show that the probability that a collision is mistaken for a single frequency is very small and then take the union bound over $\sqrt{N}\log N$ collision tests. □

LEMMA A.3. *For, any constant α, assuming that all collision detection tests are correct, the algorithm reports the correct output with probability at least $1 - O(\alpha)$.*

PROOF. Given Lemma A.1, the algorithm fails if there is a sequence of occupied frequencies $f_1, g_1, f_2, g_2 \ldots f_t$ that forms a "deadlock" *i.e.* for each $i \geq 1$, f_i and g_i collide in the first bucketization (*i.e.* $f_i = g_i \mod B_1$), while g_i and f_{i+1} collide in the second bucketization (*i.e.* $g_i = f_{i+1} \mod B_2$). Moreover, it must be the case that either the sequence "loops around", *i.e.*, $f_1 = f_t$, or $t > t_{\text{max}}$, where $t_{\text{max}} = C\log N$ is the number of iterations. We focus on the first case, the second one is similar. Define E_t as the event that such sequence exists. The number of such sequences is at most $\sqrt{N}^{2(t-1)}$, while the probability that the entries corresponding to a specific sequence are non-zero is at most $(K/N)^{2(t-1)} = (c/\sqrt{N})^{2(t-1)}$. Thus the probability of E_t is at most $c^{2(t-1)}$. Therefore, the probability that one of the events $E_1 \ldots E_{t_{\text{max}}}$ holds is at most $\sum_{t=3}^{\infty} c^{2(t-1)} = c^4/(1-c^2)$, which is smaller than α for c small enough. □

THEOREM A.4. *For $K = c\sqrt{N}$ and $p_1, p_2 = \Theta(\sqrt{N})$, the algorithm* BIGBAND *runs in time $O(K\log N)$, uses $O(K)$ samples and returns the correct result with probablility at least $1 - O(\alpha)$ as long as c is a small enough constant.*

PROOF. From Lemma A.3 and Lemma A.2, the algorithm returns the correct vector $\mathbf{X}$ with probability at least $1 - O(\alpha) - O(N^{-(a-5)/2}) = 1 - O(\alpha)$ for $a > 5$. The algorithm uses only $O(B_1 + B_2) = O(K)$ samples of x. The running time is bounded by the time needed to perform $O(1)$ bucketizations (*i.e.* FFT of sizes B_1 and B_2), and $O(\log N)$ invocations of the estimation. Both components take time $O(K\log N)$. □