Quantifying Energy and Latency Improvements of FPGA-Based Sensors for Low-Cost Spectrum Monitoring

Arani Bhattacharya*, Han Chen†, Peter Milder‡ and Samir R. Das§
Stony Brook University
*arbhattachar@cs.stonybrook.edu, †han.chen.2@stonybrook.edu, ‡peter.milder@stonybrook.edu, §samir@cs.stonybrook.edu

***Abstract*—There is a recent interest in large-scale RF spectrum monitoring using low-cost crowdsourced spectrum sensors. A major challenge here is improving the latency and energy usage of signal processing algorithms on the sensor. This improves operational cost and effectiveness and also makes the sensors more responsive to the monitoring task. We specifically consider the case of signal detection using such sensors. Typical crowdsourced implementation using a low-cost software radio connected to a Raspberry Pi or a smartphone as host is not energy-efficient and incurs significant latencies. We propose use of field-programmable gate array (FPGA) to improve both metrics for the signal detection task. Our benchmarking shows significant improvements with FPGA platforms relative to using a Raspberry Pi or smartphone, upto a factor of 73 in terms of latency and a factor of 29 in terms of energy usage.**

I. Introduction

With the continual growth of RF spectrum occupancy, there is an increasing interest in effective spectrum monitoring. Spectrum monitoring can be used to understand use of spectrum, opportunistically allow the use of under-utilized portions of the spectrum, and/or to detect unauthorized usage of spectrum. To make such monitoring ubiquitous and cost-effective, recent work has focused on crowdsourcing spectrum monitoring using low-cost spectrum sensors. For example, FlightAware uses crowdsourced spectrum sensors to detect signals from airplanes flying overhead, and track them [1]. A number of academic studies on spectrum monitoring using crowdsourcing has been reported [2, 3, 4].

While crowdsourced spectrum monitoring is effective, its actual deployment faces a number of challenges. One of the main bottlenecks towards deployment of a spectrum monitoring system is its high energy cost. Each spectrum sensor needs to run continuously, thus consuming a lot of energy. Moreover, since the crowdsourced spectrum sensors can be mobile in nature [5], the energy efficiency is even more critical. A second challenge is to reduce the latency of signal detection. Many spectrum monitoring tasks directly or indirectly detect specific signals. This requires processors with more compute power. Thus, reducing both energy consumption and latency is an important problem for large-scale, crowdsourced spectrum monitoring.

Current crowdsourced spectrum monitoring proposals (e.g., [3, 5]) utilize a single board computer such as Raspberry Pi or smartphone connected to a software-defined radio like RTL-SDR [6] or LimeSDR [7]. These devices are often energy-constrained in nature, especially in the case of outdoor deployments. Current state-of-the-art techniques can optimize energy either by reducing the duty cycle or using only a subset of the available sensors, both of which hurt accuracy of detection. Moreover, it is not possible to reduce latency of detection without significantly improving the compute power of the processor used. Since Raspberry Pis and smartphones have limited compute power, reducing latency requires using a more expensive processor board. Thus, a spectrum sensor that can significantly reduce both energy consumption and latency can improve the overall performance of spectrum monitoring.

In this work, we explore the use of field-programmable gate arrays (FPGAs) in spectrum sensors. An FPGA is a digital chip based on programmable logic. FPGAs allow algorithms to be implemented directly in hardware, producing significant performance and energy improvements. At the same time, their reconfigurability allows FPGAs to be adapted for use in differing conditions or to solve new problems without manufacturing new hardware. FPGAs can be viewed as a midway point between processor-based systems and the use of application-specific integrated circuits (ASICs). ASICs exhibit even higher efficiency than FPGAs, and with a lower per-part cost. However, the use of ASICs requires an expensive and time-consuming design and manufacturing process. This combined with their inflexibility mean that, although ASICs likely represent the best solution for a stable large-scale deployment of spectrum sensors, FPGA-based sensors can be better suited for the purpose of research.

We first observe that, when using a processor, computation of the power spectral density (PSD) and detection algorithms consume around $40\,\%$ of total energy cost, and incur $95\,\%$ of total latency. (Details of these experiments are explained in Section II.) Thus, we implement these components in hardware on the FPGA to reduce energy and latency. By implementing a flexible system that allows easy use of several different FFT sizes and detection algorithms, we have produced a framework that enables a systematic evaluation of the real-world energy and latency costs of spectrum sensing on FPGA and in software on smartphones and the Raspberry Pi.

Using this system, we perform a set of measurements to understand the performance improvement and energy savings of our FPGA-based sensor. We benchmark both energy-based

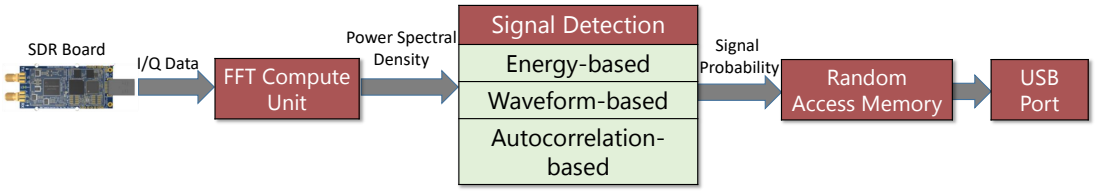
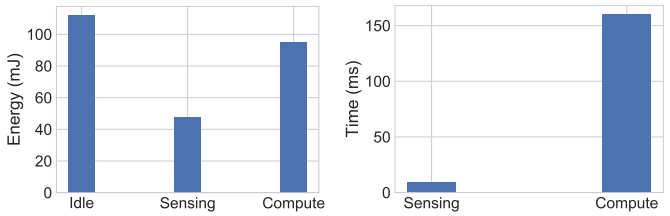


Figure 1: An overview of our FPGA-based spectrum sensor.



(a) Energy consumption **(b) Latency**

Figure 2: Energy consumption and latency of each unit of a Raspberry-Pi based spectrum sensor.

and autocorrelation-based detection using our FPGA. We find that the FPGA is able to achieve similar detection performance with 73 times lower latency than using a smartphone or a Raspberry Pi, while consuming up to 29 times less energy.

We summarize our contributions as follows:

- We demonstrate that computation speed is a performance bottleneck on existing Raspberry Pi and smartphone-based sensors.
- We implement an FPGA based spectrum sensing system.
- We run a set of benchmarks using multiple parameters and algorithms, showing that the FPGA system has 73 times lower latency and 29 times lower energy consumption compared to a Raspberry Pi based sensor.

The rest of this paper is organized as follows. In Section II, we explain the working of spectrum sensors and the motivation behind using an FPGA. Section III presents the design of our FPGA sensor. In Section IV, we describe the measurement results. Section V discusses related work, and we conclude in Section VI.

II. Background & Motivation

In this section, we describe the working of a spectrum sensor and explain the motivation behind using an FPGA.

A. Inside a Spectrum Sensor

We first explain the process of signal detection. The signal is captured by the radio front-end (sensing unit) as complex numbers, representing discrete amplitude and phase values, in terms of I and Q samples. A detection algorithm directly operates on these samples to detect the presence of a signal on an attached compute platform (compute unit). The Fast Fourier transform (FFT) on the I/Q samples is a common operation used by many such algorithms.

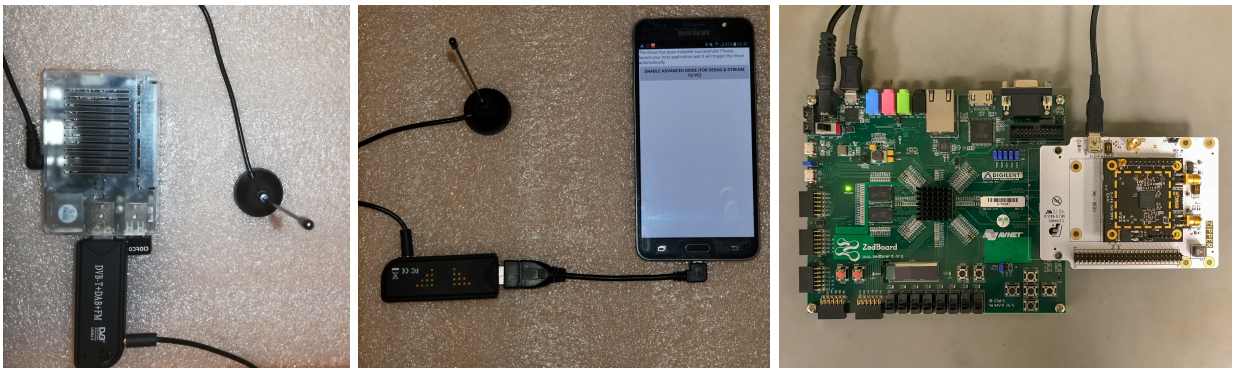
Many types of detection algorithms are possible. We will focus on two common ones – energy-based and autocorrelation-based detection, also used in prior benchmarking studies [5]. In energy-based detection, the power of the signal within a given channel is compared with a threshold. If it is greater than the threshold, then the sensor considers the signal to be present. The fundamental computation here is an FFT on the I/Q samples to compute the Power spectral density (PSD), followed by summing of the squared magnitudes of each frequency bin in the FFT. The accuracy of energy-based detection depends on the number of I/Q samples and the number of frequency bins in the FFT. Increasing these parameters improves the accuracy of detection by providing higher frequency resolution. However, this also costs time and energy, resulting in a trade-off between accuracy and cost.

In autocorrelation-based detection, we leverage the fact that many signals can have a periodic component and thus could be correlated over time, but noise is always uncorrelated. In this technique, we calculate the correlation of the signal with delayed copies of itself at specific periodic intervals. As the signal is correlated, the same patterns should get repeated at some lags, and thus a high degree of correlation should be found at these lags. In this way, it is possible to detect very weak signals, even in cases where energy-based detection fails.

B. Motivation

We motivate the need for FPGA-based spectrum sensors using the following observations:

1) **High resource cost of computation**: To understand the proportion of energy consumed in computation, we plot the energy consumed by a Raspberry Pi in its different stages. We use a Monsoon Power Monitor [8] to measure the energy consumed when performing 1024-point FFTs on 100K samples of data. (Details of the experimental setup are described in Section IV.) Figure 2a shows the energy consumed individually by: (i) a Raspberry Pi if it is lying idle, (ii) by the sensing unit (SDR), and (iii) by the compute unit (when it runs autocorrelation-based detection) for the same amount of time taken by the compute unit. We obtain the idle energy by measuring the energy consumed during the entire amount of time taken to compute the power spectral density if no computation is performed. We compute the energy consumed by the sensing unit and the compute unit by subtracting the energy value obtained from the power monitor by the idle energy. We note that the energy cost of the compute unit is overall 37 % of the



(a) Raspberry-Pi based sensor (b) Smartphone based sensor (c) FPGA based sensor

Figure 3: Our experimental testbed consisting of three different types of sensors.

total cost. Thus, a significant amount of energy is spent on computation.

We now look at the compute latency. We measure the latency of different portions of the algorithm by printing the timestamps at different stages of our software. We plot the latency of sensing and computing incurred while running an energy-based detector on 100K samples (Figure 2b). The compute latency includes the time to perform the FFT and to run the detection algorithm. The sensing latency refers to the time to read data from the SDR front-end and send it to the buffer, assuming the SDR is already on. We again find that around $95\,\%$ of the latency is caused by the compute time. This demonstrates that faster computation can significantly speed up signal detection.

2) **Repetitive execution**: A second key observation is that the computations performed (running the FFT and the signal detection algorithm) are repetitive in nature. In other words, the same process is repeated many times during execution. This represents an ideal situation for hardware acceleration, where a carefully-crafted hardware design can speed up the specific "hot spots" of the algorithm.

III. Our FPGA-based Spectrum Sensor

This section presents the design of our FPGA-based spectrum sensor, as shown in Figure 1. Our system is implemented using a Xilinx ZedBoard FPGA development board connected to a Myriad-RF1 transceiver board [9]. The ZedBoard uses a Xilinx Zynq-7000 All Programmable SoC (XC7Z020-CLG484-1); this chip combines a standard FPGA reconfigurable fabric with an embedded ARM processor, which we are using only for debugging. The Myriad-RF1 board includes a Lime Microsystems LMS6002D transceiver, which covers a frequency range of 300 MHz to 3.8 GHz, and uses a 12-bit ADC. The RF1 connects to the ZedBoard FPGA system via a Myriad-RF Zipper board, which provides a convenient physical interface to the FPGA.

As shown in Figure 1, I/Q samples flow out of the Myriad-RF1 (labeled "SDR Board") and into the FPGA. First, we designed logic to buffer the raw input data and synchronize it with the FPGA's internal clock rate. Then, our system feeds the data into an FFT core. We create the FFT modules using the Spiral FFT generator [10], a system that produces customized hardware for FFTs, parameterized by the FFT size, data precision, and other options. We used Spiral to produce different configurations, optimized for different FFT sizes. The FFT core feeds its result (the frequency domain representation of the input signal) into the signal detection unit.

The signal detection unit can be pre-configured to run a detection algorithm based on energy detection, waveform detection, or autocorrelation. For the energy detection algorithm, we compute the energy over a specified (and configurable) range of the frequency spectrum. For the autocorrelation-based algorithm, we designed the hardware module to compute the autocorrelation in the frequency domain. Each of these algorithms outputs a stream of data that denotes the probability that a signal of interest is present.

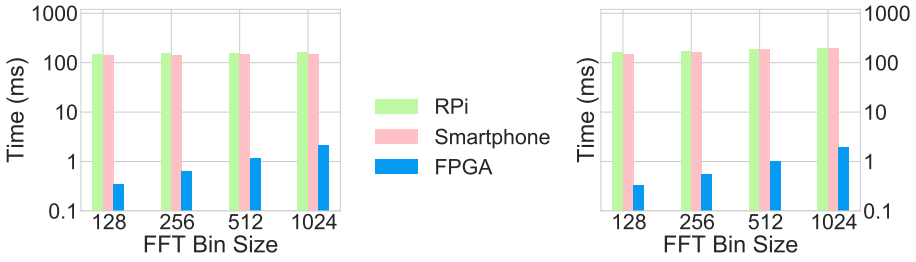
For convenience, we can store the stream of result data in RAM using a Direct Memory Access (DMA) module or output it directly to pins. From memory, the data is then sent out to a computer or other device using either USB or UART ports.

IV. Measuring Resource Usage

We focus on two distinct metrics of resource usage—(i) compute latency and (ii) energy consumption. We perform separate experiments on our Raspberry Pi, smartphone and FPGA-based sensors. Figure 3 shows photos of our testbeds: a Raspberry Pi 3 (model B) [11], a Samsung Galaxy S4 smartphone with an RTL-SDR based sensor, and the Xilinx ZedBoard FPGA with Myriad-RF. The FPGA system is constructed as described in Section III; for the Raspberry Pi and smartphone, we have implemented the algorithms in C.

A. Computation Latency

We first compare the computation latency of the three types of sensors. For our FPGA, the latency is the time incurred from the first I/Q sample arriving in the FPGA board, to the output of the signal detection units. Because the FPGA system uses our custom hardware design, its latency is a deterministic value dependent only on the number of clock cycles required by the computation hardware and the FPGA's



(a) Energy-based Detection (b) Autocorrelation-based Detection

Figure 4: Comparison of latency of the compute unit of our FPGA, Raspberry-Pi and smartphone based sensors.

internal clock frequency. For the Raspberry Pi and smartphone implementations, we measure the average latency by timing the C code operating on a file of previously-stored I/Q data.

Figure 4 shows the latency of all three sensors executing the energy-based and autocorrelation-based detection algorithms. We find that the FPGA results in significant latency improvements– for energy-based detection, FPGA has around $73\times$ lower latency than the Raspberry Pi and $69\times$ less than our smartphone. This is expected, since the FPGA is configured to use specialized parallel computational structures that can execute the detection algorithms more efficiently than the general-purpose processors used in the other systems. We also note that the latency increases by a similar absolute value with an increase in the FFT bin size. However, as the number of samples is constant, this increase in latency is not significant because the number of FFT computations reduces with an increase in the FFT bin size.

B. Energy Consumption

To measure the power and energy consumption of the Raspberry Pi and smartphone, we use a Monsoon Power Monitor [8]. We subtract the power consumption reported by the power monitor by the idle power to get the power consumed in computation. For all systems, we find the energy and power consumed by the system while running the energy-based detection algorithm with FFT bin sizes ranging from 128 to 1024. For both smartphone and Raspberry Pi, we avoid running other compute-intensive processes to avoid interference. For the Raspberry Pi, we run the algorithms remotely over a console by connecting it over a wired network.

Quantifying the exact power and energy consumption of the FPGA is non-trivial. The ZedBoard FPGA system is a development board that contains a variety of components (such as DRAM, flash memory, and an ARM processor) which are unused in our spectrum sensor. For this reason, physically measuring the power of the entire development board does not give a realistic measure of the power consumed by the necessary components of the sensor. Instead, to obtain the power consumption of only the FPGA (without considering the unnecessary components of the development board) we use the FPGA vendor's power simulation tool (part of Xilinx Vivado) to produce estimates of the power consumed by the FPGA, which are labeled *Zynq FPGA* in our results.

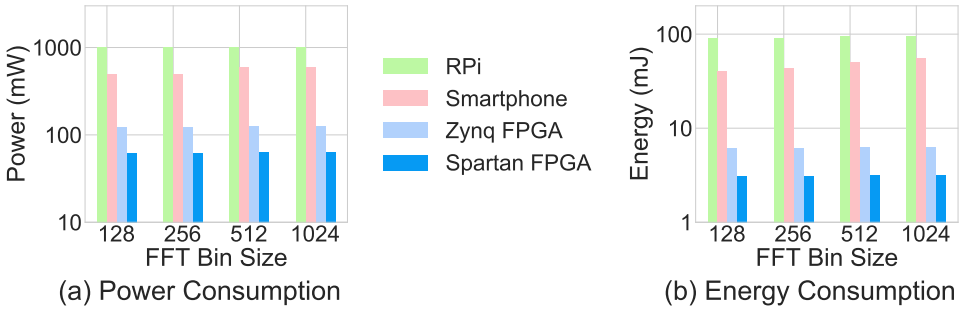
Lastly, we note that the ZedBoard's Zynq FPGA is greatly over-provisioned for this application; our largest design uses $< 20\%$ of the chip's reconfigurable logic and only two-thirds of its arithmetic units. To quantify the further improvement available from using a smaller FPGA, we re-implemented the exact same functionality (running at an identical speed) on a smaller and lower-power Xilinx Spartan 7 FPGA (XC7S50FTGB196). Vivado's power simulation tool shows that the same design on the Spartan 7 FPGA requires approximately one half of the power as the Zynq FPGA. This design is labeled *Spartan FPGA* in our results.

Figure 5(a) shows the power consumption of these systems. We observe that the Zynq FPGA consumes about 8 times less power than the Raspberry Pi. Moreover, the power consumption can be further reduced by another $50\,\%$ by using the smaller Spartan FPGA, whose size and logic capacity are more appropriate for the application. Our results show that it is possible to reduce power consumption using an FPGA, but a careful design of the logic as well as the overall FPGA system and its board is necessary.

To measure the energy consumption, we run PSD on 100K samples with the sensor sampling rate set at 2 million samples per second. We then measure the energy consumed in computing the PSD of these 100K samples by integrating the result given by the Monsoon Power Monitor over the entire period. Figure 5(b) shows the energy consumption of the three sensors. Note that because of the effect of pipelining, the energy consumption is not necessarily equal to the product of power and latency. In general, we find a similar trend as seen in power consumption. The energy consumption of the Zynq FPGA and the Spartan FPGA are 14 and 29 times smaller respectively than the Raspberry Pi at an FFT bin size of 256. This confirms that a carefully designed FPGA can lead to significantly lower energy consumption.

V. Related Work

Recently, there has been a lot of interest in spectrum monitoring using low-cost spectrum sensors, e.g., Specsense [2], Electrosense [3] and Radiohound [12]. All utilize a large number of RTL-SDRs, with each connected via USB to a Raspberry Pi. Snoopy [13] proposes attaching a frequency converter to smartphones and utilizing individual smartphones as a spectral analyzer. In [5] authors benchmark latency and resource usage in similar Raspberry Pi and smartphone-



(a) Power Consumption (b) Energy Consumption

Figure 5: Comparison of power and energy consumption of our FPGA, Raspberry-Pi and smartphone based sensors.

based spectrum sensors. In [4] authors investigate algorithmic issues in optimizing overall energy consumption by intelligent selection of the most relevant sensors.

The use of FPGAs for spectrum sensing is attractive due to FPGAs' flexibility, power efficiency, and computational abilities [14, 15]. However, the relative difficulty of FPGA implementation poses a significant barrier to implementation and to understanding important tradeoffs. Prior work has made a case for using FPGAs for spectrum sensing or related problems, but these are limited in the scope of their hardware considerations. For example, the FPGA system described by [16] considers only one algorithm (energy detection) with a single FFT size. In contrast to these studies, we focus on performing a systematic performance comparison of an FPGA-based sensor with the more widely available smartphone and Raspberry Pi-based sensors.

VI. Conclusion

In this work, we systematically compare the performance of spectrum sensors for signal detection tasks, where the compute part of the sensors is based on embedded platforms such as Raspberry Pi, smartphone, vs FPGAs. We first made the observation that computation is the most energy-intensive process in spectrum sensing. We then described our implementation of FPGA-based sensor, which efficiently runs the computation entirely in hardware. We then compared its power consumption and latency with the Raspberry Pi-based and smartphone-based sensors. Our measurements show that the FPGA-based sensor consumes up to 29 times lower energy, and has around 73 times lower latency than a Raspberry Pi-based sensor.

Acknowledgement

This work is partially supported by NSF grants AST-1443951, CNS-1642965 and by the MSIT, Korea, under the ICT Consilience Creative Program (IITP-2017-R0346-16-1007). The authors also thank Ayon Chakraborty for implementing some of the algorithms and insightful discussions.

References

[1] "FlightAware – Flight Tracker/Flight Status/Flight Tracking," accessed Aug 15, 2018. [Online]. Available: https://flightaware.com/

[2] A. Chakraborty, M. S. Rahman, H. Gupta, and S. R. Das, "Specsense: Crowdsensing for efficient querying of spectrum occupancy," in *Proc. IEEE INFOCOM Conference*, 2017.

[3] R. Calvo-Palomino, D. Giustiniano, V. Lenders, and A. Fakhreddine, "Crowdsourcing spectrum data decoding," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[4] A. Chakraborty, A. Bhattacharya, S. Kamal, S. R. Das, H. Gupta, and P. M. Djuric, "Spectrum monitoring using crowdsourced spectrum sensors," in *Proc. IEEE INFOCOM Conference*, 2018.

[5] A. Chakraborty, U. Gupta, and S. R. Das, "Benchmarking resource usage for spectrum sensing on commodity mobile devices," in *Proc. 3rd ACM HotWireless Workshop*, 2016, pp. 7–11.

[6] RTL-SDR, accessed August 13, 2018. [Online]. Available: https://www.rtl-sdr.com/

[7] L. Microsystems, accessed May 26, 2018. [Online]. Available: https://www.crowdsupply.com/lime-micro/limesdr-mini

[8] "Monsoon solutions." [Online]. Available: https://www.msoon.com/

[9] MyriadRF, "Myriad-RF 1," accessed May 31, 2018. [Online]. Available: https://wiki.myriadrf.org/Myriad-RF_1

[10] P. Milder, F. Franchetti, J. C. Hoe, and M. Püschel, "Computer generation of hardware for linear digital signal processing transforms," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 17, no. 2, p. 15, 2012.

[11] "Raspberry Pi 3 Model-B," accessed May 31, 2018. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[12] N. Kleber, A. Termos, G. Martinez, J. Merritt, B. Hochwald, J. Chisum, A. Striegel, and J. N. Laneman, "RadioHound: A pervasive sensing platform for sub-6 GHz dynamic spectrum monitoring," in *Proc. 2017 IEEE DySpan Symp.*

[13] T. Zhang, A. Patro, N. Leng, and S. Banerjee, "A wireless spectrum analyzer in your pocket," in *HotMobile*, 2015.

[14] S. Srinu, S. L. Sabat, and S. K. Udgata, "FPGA implementation of cooperative spectrum sensing for cognitive radio networks," in *Cognitive Wireless Systems (UKIWCWS), 2010 Second UK-India-IDRC International Workshop on*. IEEE, 2010, pp. 1–5.

[15] G. Chaitanya, P. Rajalakshmi, and U. Desai, "Real time hardware implementable spectrum sensor for cognitive radio applications," in *Signal Processing and Communications (SPCOM), 2012 International Conference on*. IEEE, 2012, pp. 1–5.

[16] D. Cabric, A. Tkachenko, and R. W. Brodersen, "Experimental study of spectrum sensing based on energy detection and network cooperation," in *Proceedings of the first international workshop on Technology and policy for accessing spectrum*. ACM, 2006, p. 12.