Demo: Observing wideband RF spectrum with low-cost, resource limited SDRs

Demo: Observing wideband RF spectrum with low-cost, resource limited SDRs

Raghav Subbaraman
UC San Diego
rsubbaraman@eng.ucsd.edu

Nishant Bhaskar
UC San Diego
nibhaska@eng.ucsd.edu

Sam Crow
UC San Diego
scrow@eng.ucsd.edu

Moein Khazraee
MIT
moein@mit.edu

Aaron Schulman
UC San Diego
schulman@eng.ucsd.edu

Dinesh Bharadia
UC San Diego
dineshb@eng.ucsd.edu

1 INTRODUCTION

Software Defined Radios (SDRs) combine a universal radio frontend with flexible processing. The radio frontend can be tuned to capture various wireless signals, while software processing allows quick and scalable deployment for diverse applications. SDRs seem like a good fit for the ever-evolving needs of today's spectrum usage: SDRs can be deployed today, then managed and upgraded with software to support the needs of tomorrow. However, the prevailing architecture of SDRs prevent real-time observation of wideband RF signals due to backhaul and processing resource constraints.

For example, low-cost SDRs like the Pluto-SDR [1] are increasingly adopted by the research and maker community. While these devices have a 61.44 MHz sample rate, the user cannot take advantage of it due to the low-speed USB-2.0 backhaul. Even if the backhaul was not limited, processing and long-term storage of tens of MHz of sampled data is intractable on low-compute laptops or embedded devices like raspberry-pi. In SparSDR [7], we proposed a new SDR architecture that allows selective backhaul of information, combined with selective software compute. SparSDR performs a frequency channelization on the radio samples, and backhauls only the channels where activity is detected. While conventional SDRs have a fixed backhaul rate and a fixed requirement on compute power, SparSDR's requirements scale with the activity in the spectrum or the intended use case.

SparSDR's ability to channelize and threshold the spectrum allows deployment of applications hitherto thought impossible. In this work, we deploy SparSDR on a Pluto-SDR, and demonstrate a BLE beacon scanning application. Monitoring two BLE channels (37 and 38) separated by 25 MHz is typically thought to require 25 MHz of bandwidth. But with SparSDR, we can selectively backhaul only those two channels, limiting the bandwidth backhauled to 4 MHz! In addition, SparSDR backhauls samples only when a signal is present, further reducing the data generated (Figure 1). In typical BLE beacon scanning scenarios, SparSDR could compress the spectrum by 288x, reducing the compute requirement for decoding as well as I/Q sample storage [6]. To improve system accessibility and adoption, we have integrated SparSDR with software suites used commonly with the Pluto-SDR: libiio and GNURadio. Users can use SparSDR to deploy their applications by using the open source code-base available at **github.com/ucsdsysnet/sparsdr**.

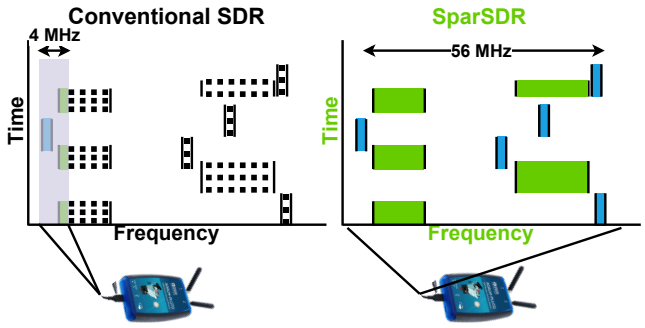

Figure 1: Conventional Full-capture SDRs (left) sample the entire spectrum to capture fleeting signals in limited bands. SparSDR (right) compresses the spectrum using STFT and thresholds, thus scaling backhaul and compute proportional to the true amount of activity in the spectrum.

2 SPARSDR ON PLUTO-SDR

The Pluto-SDR [1] is a device developed by Analog Devices Inc. It features an AD9363 RF frontend [2] IC, combined with a Xilinx Zynq-7 FPGA [8] featuring a single-core ARM Cortex-A9 at 667 MHz [3]. Additionally, it features 512 MB of DDR3L RAM, and a USB 2.0 backhaul. It is possible to modify the Pluto-SDR and unlock its 56 MHz of RF bandwidth with a tuning range from 70 MHz - 6 GHz [4]. The system level architecture of the Pluto-SDR is described in Figure 2.

FPGA Implementation: The SparSDR IP is deployed on the Zynq-7 FPGA on the Pluto-SDR. The IP is built on AXI-Stream interfaces and can be easily ported to other SDRs[1]. The SparSDR IP supports full-rate throughput, and can stream at maximum rate onto the on-board DRAM. The throughput out of the Pluto-SDR is bottlenecked by the USB-2 backhaul (Figure 2). The compression protocol has been upgraded recently to yield close to 2x gain in compression rate.

[1] SparSDR is also available on the USRP N210 [7]

MobiSys '22, June 25–July 1, 2022, Portland, OR, USA

ACM ISBN 978-1-4503-9185-6/22/06…$15.00
https://doi.org/10.1145/3498361.3538662

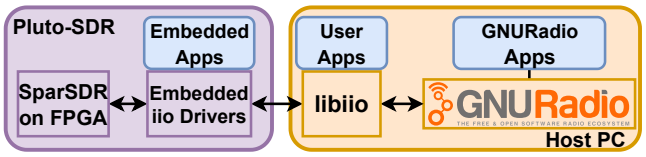

Figure 2: Architecture of SparSDR on Pluto-SDR. The software drivers for SparSDR are fully contained within the embedded linux. libiio or GNURadio may be used from the host PC to interact with SparSDR . User applications that use SparSDR can be flexibly deployed on the Pluto-SDR's embedded linux, or on the Host PC.

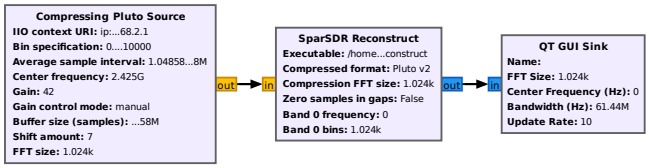

Figure 3: An example GNURadio application that streams data fron a SparSDR enabled Pluto-SDR and reconstructs spectrum activity in real-time for visualization.

The configuration registers of SparSDR control the bin-wise threshold, masks and average monitoring intervals. The user can also control the FFT size and scaling for fine grained control over channels and dynamic range. The Pluto-SDR's AD9361 IP and controls are unmodified, allowing them to be used as-is.

Software and drivers: The Pluto-SDR runs a custom Linux on the ARM core, and uses ADI's libiio [5] library to stream samples to and from the radio. In the same vein, we implemented IIO drivers for SparSDR, making it possible to use the entire stack with just IIO API. The Pluto-SDR can fully control and stream from SparSDR using only the on-board linux system, without the explicit need of additional compute like a PC or Raspberry-Pi.

2.1 Host-Side Software

Reconstruction signal processing: To decompress and reconstruct SparSDR's output, we use a software executable written in Rust. This executable works on Linux named pipes and can be fit inside any streaming application. By taking advantage of highly parallel FFT libraries, the overall footprint of this software scales with the sparsity in the channel. For more details, we refer the reader to [7].

GNURadio: For ease of use and a GUI, we integrated control of the SparSDR IIO modules into gr-iio and created a out-of-tree GNURadio module called gr-sparsdr. This module can be used to interact with SparSDR on the Pluto-SDR directly from GNURadio, allowing access to the large set of signal processing libraries.

Software utilities: To aid the SparSDR user in selecting per-bin thresholds, and correct FFT dynamic ranges, we have written automated calibration utilities and included them in the open-source release. These utilities use only libiio commands and can be run without GNURadio if required.

3 APPLICATION: BLE BEACON SENSING

Personal devices such as smartphones are continuously transmitting BLE beacons. These beacons enable various features such as COVID-19 contact tracing or finding a lost device. Ability to observe these beacons in real-time and over a long duration can help us understand not only spectral usage, but also smart device behaviour and related security/privacy implications [6]

Sensing all transmitted beacons in real-time however requires us to observe multiple BLE channels spread across a wide-band. For example, if the user intends to capture beacons on two channels (1 MHz wide) centered at 2.402 GHz and 2.426 GHz, they would need a receive bandwidth of 25 MHz. While the PlutoSDR can theoretically support this bandwidth, this translates to a continuous backhaul of 800 Mbps which cant be supported by the USB 2.0 interface. Furthermore, long duration temporal analyses will require us to store these raw BLE beacon samples, which will need an impractical amount of disk space (~350 GB per hour).

This application can benefit greatly from SparSDR. The masking feature acts as a channelizer, allowing us to selectively backhaul only the two 1 MHz bands corresponding to the two BLE channels. This reduces the effective backhaul by a factor of 12.5x. Furthermore by using thresholds, we can exploit the sparsity of the signals further reducing the backhaul. The authors of [6] were able to utilize these SparSDR features on a USRP N210 based setup and run a field experiment. They were able to capture BLE beacons over three BLE channels spread across 80 MHz in a public place. The experiment ran for over 2 days (10 hours per day), and the total space on disk to store all compressed captures was only 100 GB or approximately 5 GB for every hour.

4 DEMONSTRATION

We will demonstrate BLE beacon sensing across two BLE channels separated by 25 MHz using a single PlutoSDR and open source GNURadio based flowgraph (Figure 3) on a consumer laptop.

ACKNOWLEDGEMENTS

This work was supported in part by a grant from Amateur Radio Digital Communications (ARDC).

REFERENCES

[1] Analog Devices Inc. [n.d.]. ADALM PLUTO Overview. https://wiki.analog.com/university/tools/pluto.
[2] Analog Devices Inc., AD9363. [n.d.]. https://www.analog.com/en/products/ad9363.html.
[3] Analog Devices Inc. ADALM-PLUTO, Detailed Specifications. [n.d.]. https://wiki.analog.com/university/tools/pluto/devs/specs.
[4] Analog Devices Inc., Customizing the Pluto configuration. [n.d.]. https://wiki.analog.com/university/tools/pluto/users/customizing.
[5] Analog Devices Inc., What is LibIIO? [n.d.]. https://wiki.analog.com/resources/tools-software/linux-software/libiio.
[6] H. Givehchian, N. Bhaskar, E. Rodriguez Herrera, H. Lopez Soto, C. Dameff, D. Bharadia, and A. Schulman. 2022. Evaluating Physical-Layer BLE Location Tracking Attacks on Mobile Devices. In *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 507–521. https://doi.org/10.1109/SP46214.2022.00030
[7] Moein Khazraee, Yeswanth Guddeti, Sam Crow, Alex C. Snoeren, Kirill Levchenko, Dinesh Bharadia, and Aaron Schulman. 2019. SparSDR: Sparsity-Proportional Backhaul and Compute for SDRs. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 391–403. https://doi.org/10.1145/3307334.3326088
[8] Xilinx, Zynq 7000 overview. [n.d.]. https://www.xilinx.com//support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.