Demonstration of All-Spectrum Cognitive Channelization on GNU Radio and USRPN-210

George Sklivanitis,[†] Emrecan Demirors,[†] Dimitris A. Pados,[†] Stella N. Batalama,[†*] Tommaso Melodia,[†]
and John D. Matyjas[‡]
[†]Department of Electrical Engineering,
The State University of New York at Buffalo, Buffalo, NY 14260-2050 USA
[‡]Air Force Research Laboratory/RIGF, 525 Brooks Rd, Rome, NY 13441
{gsklivan, emrecand, pados, batalama, tmelodia}@buffalo.edu[†], John.Matyjas@us.af.mil[‡]

***Abstract*—We create new software signal processing blocks and provide transmitter and receiver designs in GNU Radio and MATLAB to experimentally demonstrate the theoretical concepts of all-spectrum cognitive channelization in a software-defined-radio (SDR)-based testbed. Three low-cost, SDR nodes (USRPN-210) are deployed in an indoor, multipath-fading, lab environment connected via Gigabit Ethernet (GigE) to three host-PCs for carrying out real-time signal processing.**

I. Experimental Testbed Setup

We focus on the software design, implementation in software-defined radio (SDR), and final deployment of a low-cost indoor testbed for evaluating the concepts of cognitive channelization [1]. For this reason three commercial SDR transceivers (Fig. 1) named Universal Software Radio Peripheral ($\mathrm{USRPN}-210$) [2] are used to provide a setup of a secondary link operating in the presence of primary user's interference.

A. USRPN-210, SDR platform

$\mathrm{USRPN}-210$ is a low-cost, flexible SDR platform that consists of a Field-Programmable-Gate-Array (FPGA)-based motherboard interfaced with a daughterboard that plays the RF front-end role. Digital baseband data processing takes place in a host-PC connected to the Universal-Software-Radio-Peripheral (USRP) via Gigabit Ethernet (GigE). Particularly, when our SDR acts as a transmitter the on-board FPGA is listening to digital samples coming from the PC which are then converted into analog waveforms (via digital-to-analog converter (DAC)) and transmitted. On the other hand, when USRP plays the receiver role, digital data (via analog-to-digital converter (ADC)) are fed back to the PC in the 16-bit in-phase and 16-bit quadrature format. When digital signal processing is carried out by the host PC, the true maximum sampling rate is upper bounded by the host's hardware specifications and therefore processing capabilities.

Physical layer processing at the host PC may be conducted via numerous software tools that include communications and signal processing toolboxes and libraries (e.g. GNU Radio, MATLAB, Simulink, LabVIEW).

*Corresponding author.
This work was supported by the U.S. Air Force Research Laboratory under Grant FA8750-11-1-0016. Approved for Public Release; Distribution Unlimited: 88ABW-2011-5073.

B. GNU Radio software

GNU Radio [3] is open-source software that offers a plethora of C++ digital signal processing blocks and libraries that provide primitive functionalities, i.e. filtering, different forms of modulation (BPSK, QPSK, GMSK, etc.), carrier phase/frequency and time synchronization, etc. However, due to the specific signal processing features of the proposed cognitive channelization technique, custom signal processing blocks were created for this demonstration and were wrapped along the existing interface.

II. Transmitter design in GNU Radio

Both transmitters (secondary, primary) are designed and implemented in GNU Radio. Both follow the block diagram of Fig. 2.

In the flowgraph depicted in Fig. 2, a random byte generator source repeatedly feeds the transmitting signal processing path with a pre-specified number of bytes. Then, the packet encoding block encapsulates the incoming bytes in a specific packet format (preamble, syncword, payload, cyclic-redundancy-check (CRC)). Each byte is afterwards converted to data symbols of 8 bits. In that way, we produce a sequence of J bits, where each is mapped to 1 or -1 according to the BPSK constellation to create bitstream $b_k(i) \in \{\pm 1\}$, $i = 0, \ldots, J-1$, $k = 1, 2$. The generated bitstream is then given as an input to our own custom signal processing block which modulates each bit with a digital waveform of code bits $s_k(l)$, $l = 0, \ldots, L-1$. The waveform-bit sequence and its length L are assumed to be known between every transmitter-

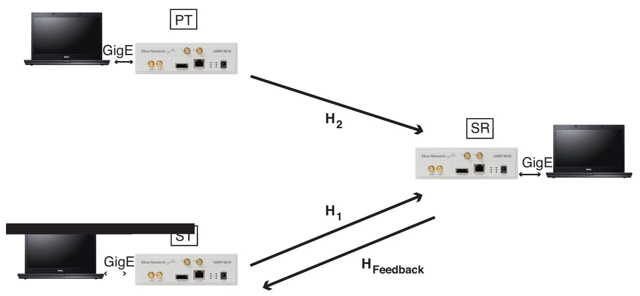

Fig. 1. Testbed schematic.

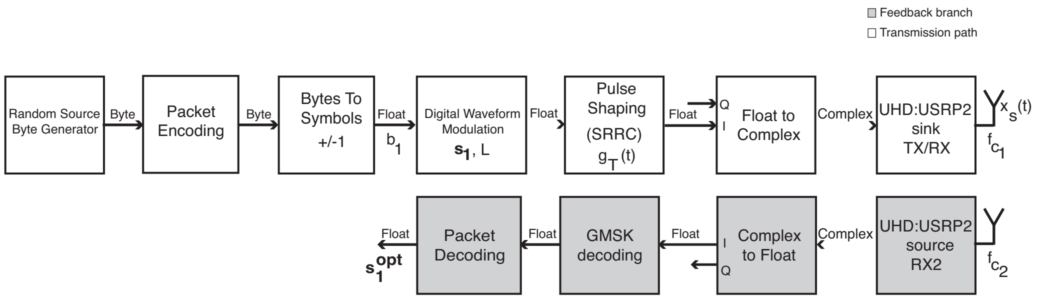

Fig. 2. Primary/Secondary transmitter schematic.

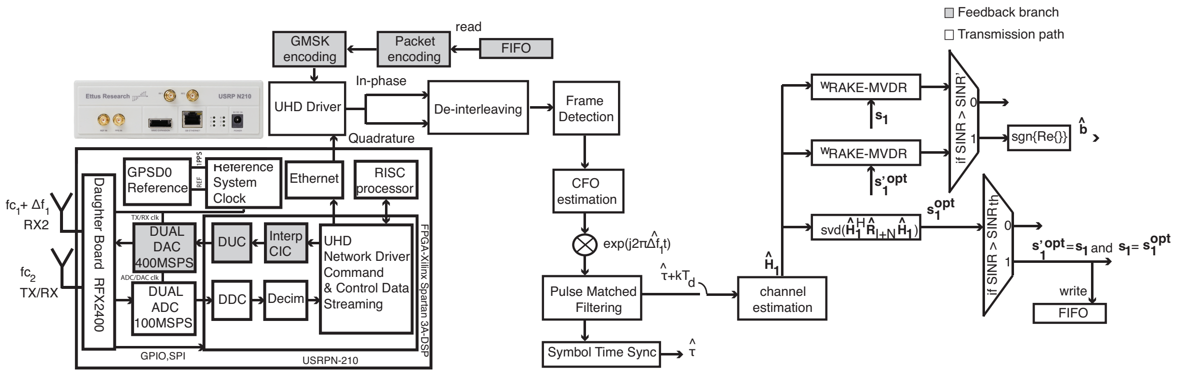

Fig. 3. Secondary receiver schematic.

receiver pair. Waveforms at the secondary transmitter are dynamically changing according to the feedback information it receives from the secondary receiver. Finally, pulse shaping is applied to the waveform-modulated bitstream. For this task, an already available GNU Radio block of a square-root-raised-cosine (SRRC) filter is used. The generated pulse train at the output of the last block provides the in-phase component of our IQ transmitter. The quadrature component is given by a null source. The feedback branch in Fig. 2 drives waveform adaptation at the transmitter side. Regarding the implementation of the feedback channel we preferred to apply a well-established GMSK modulation scheme, well tested with the specific hardware ($\mathrm{USRPN}-210$) and GNU Radio software. Thus, we managed to build a quite reliable wireless feedback channel that is responsible for communicating the optimal waveform derived by the receiver.

III. Receiver design in GNU Radio and MATLAB

At the receiver side of the secondary link (Fig. 3) we use both GNU Radio and MATLAB for implementing the receiver's digital signal processing blocks. GNU Radio along with the UHD driver offers us the RF front-end functionality for controlling the USRP. The selected daughterboards (i.e. RFX2400) allow us to operate in full-duplex mode. Two RF-chains operate simultaneously. One is receiving at $f_{c_1} = 2.480$ GHz, while the second is responsible for communicating the optimal waveform via the feedback channel at $f_{c_2} = 2.410$ GHz. Both transmitting and receiving frequencies were selected after conducting measurements with an RF-spectrum analyzer to ensure absence of unaccounted laboratory environment interference.

The secondary receiver always listens to its channel and stores the resulting vector in a FIFO-type file. The FIFO file is continuously read by a MATLAB thread that contains implementations of the frame detection, CFO estimation, time synchronization, etc., blocks. The design of the receiver has two parallel, almost identical chains, yet different in the waveform-bit sequence they use. These receive chains operate simultaneously and demodulate the incoming frame with both s_1 (preset) and $\mathrm{s}_1'^{\mathrm{opt}}$. When $\mathrm{s}_1'^{\mathrm{opt}}$ is not yet defined, it is initialized at a randomly selected, length-L waveform bit sequence. We keep the bit decisions $\hat{\mathbf{b}}$ that are calculated by the chain with the highest post-filtering pre-detection signal-to-interference plus noise ratio (SINR).

Optimal signature design takes place after having acquired both disturbance autocorrelation matrix $\hat{\mathbf{R}}_{I+N}$ and channel $\hat{\mathbf{H}}_1$ estimates. By eigendecomposing $\hat{\mathbf{H}}_1^H \hat{\mathbf{R}}_{I+N} \hat{\mathbf{H}}_1$ we derive the optimal waveform $\mathrm{s}_1^{\mathrm{opt}}$. If SINR is greater than a user-defined threshold (i.e. $\mathrm{SINR}_{\mathrm{th}}$), then $\mathrm{s}_1^{\mathrm{opt}}$ is written in another FIFO file and is transmitted back to the secondary transmitter via the GMSK-based feedback channel. If the above SINR condition holds the receiver's processing chains experience the following changes: $\mathrm{s}_1'^{\mathrm{opt}} = \mathrm{s}_1$ and $\mathrm{s}_1 = \mathrm{s}_1^{\mathrm{opt}}$. In that way, we manage to "protect" the receiver from the extreme case where the transmitter did not immediately update his waveform-bit sequence or the wireless feedback channel was unreliable.

References

[1] K. Gao, S. N. Batalama, D. A. Pados, and J. D. Matyjas, "Cognitive code-division channelization," in *IEEE Transactions on Wireless Communications,* vol. 10, no. 4, April 2011.

[2] (2014) USRP products. Matt Ettus. [Online]. Available: http://www.ettus.com

[3] (2014) GNU radio. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki