Spread Spectrum Based Cooperative Communication Transceiver on FPGA Platform

Babak Azimi-Sadjadi, Satya Prakash Ponnaluri, Ali Namazi, Siddharth Gaddam
*†Intelligent Automation Inc., 15400 Calhoun Dr., Rockville, MD 20854
{babak, sponnaluri, anamazi, sgaddam}@i-a-i.com

Dan McCarthy, Paul J. Oleski
Air Force Research Laboratory, Rome, New York 13441
{daniel.mccarthy.9,Paul.Oleski@rl.af.mil}

ABSTRACT

In this paper we describe the implementation of a spread spectrum based cooperative relaying on an FPGA platform. We focus on a network comprising one source, one or more relays, and one destination. Once the source's message is received by the relays, the relays simultaneously transmit the information to the destination. In order to avoid interference between the various relay transmissions, each relay node spreads its information using a unique spreading code chosen from a set of sequences with good cross-correlation and autocorrelation properties. These properties allow separation of the signals in the presence of frequency and timing offsets between the various transmissions. This platform is tested through an audio application using a commercially available RF front end. The platform not only enables testing of cooperative communications, but also other related concepts relying on software-defined radio such as Cognitive Radio.

Categories and Subject Descriptors

B. Hardware B.4 INPUT/OUTPUT AND DATA COMMUNICATIONS B.4.1 Data Communications Devices Subjects: Transmitters**; Receivers (e.g., voice, data, image)

Keywords

Cooperative Communication, Spread Spectrum, RAKE receiver, FPGA, Cognitive radio

1. INTRODUCTION

Over the past decade, cooperative diversity or cooperative relaying has been a topic of great interest for researchers around the world due to its ability to realize the benefits of multiple antenna communications using multiple single antenna nodes. Cooperative communications exploit the broadcast nature of wireless media to provide increased protection against wireless phenomena such as fading. This increased protection against fading is obtained when one or more nodes called relays assist a source in transmitting its message to a destination, and is manifested in the form of faster decay of probability of error as a function of average link signal to noise ratio between the source and its intended destination.

When more than one relay can assist a source, the diversity protection is increased. However, the primary challenges are surrounding the information exchange between the various relays that have successfully decoded the source's message[1], and coordinating the transmissions from these relays so that they do not interfere. All these problems come under the umbrella of distributed multiple-input, multiple-output (MIMO) transmission, and significant research has been carried out to date in designing protocols and space-time codes that allow multiple terminals to cooperate with a single source. Examples range from classic orthogonal space-time block coding [2] to randomized space-time coding protocols [3]. We refer to [2–5] for a detailed account. An alternative to space-time coding that is suitable for cellular communications using code-division multiple access is the utilization of existing spreading codes with good autocorrelation and cross-correlation properties to resolve the relay transmissions in the presence of frequency and timing offsets. In the context of cognitive radios, cooperative communications adds another layer of flexibility to primary and secondary users to share spectrum. Cooperative schemes can use the cognitive capabilities of radios to be able to listen to signals from others and process it for future use.

Much of the work to date has been focused on the development of protocols and algorithms to demonstrate the benefits the cooperative diversity under various network and channel conditions. However, these have remained in the theoretical and simulation realm for the most part, with slower progress on the implementation front. In the past few years, a few cooperative communications test-beds have been developed that allow the demonstration of the benefits of cooperation. Most notable of these is the wireless open-access research platform developed at Rice university [6]. Other test-bed implementations are either on a digital sig-

*This work is supported by Air Force Research Lab under a SBIR contract FA8650–07–C–4519.
†Approved for Public Release: Distribution Unlimited: 88ABW-2010-1433

[1]We focus only on decode-and-forward based relaying [1].

nal processor (DSP) platform [7], or on a general purpose software-defined radio such as a GNU Radio [8]. Zhang et al provide a good overview of the existing test-beds in [8].

In this research we address some of the practical problems of implementing the basic physical layer of cooperative communication system [2]. The relay nodes that have decoded the received packets correctly (using a cyclic redundancy check code, for example) cooperate and transmit uncoded packets at the same time. We address the following practical problems in this setting: a) different cooperative transmitters have different frequency offsets and clock drifts, b) synchronizing the transmitters to start transmission at the same time cannot be done at the chip level (for spread spectrum based communication) and any practical technique can synchronize concurrent reception to within a few bits. With these constraints in mind we design and implement a cooperative communication system on a field-programmable gate array (FPGA) based software defined radio in 915 MHz frequency band for an audio application.

The rest of the paper is organized as follows: In Sec. 2, we describe the architecture of the cooperative communication system and provide the simulation performance, in Sec. 3 we present the detailed baseband implementation of the spread spectrum based cooperative on the FPGA platform. In sections 4-6 we describe the radio frequency (RF) front-end, audio application component, and the test methodology. Finally, some concluding remarks are provided in Sec. 7.

2. DESIGN ARCHITECTURE

We focus on a cooperative relaying network consisting of one source, one or more relays, and one destination. Once the source's message is received by the relays, they simultaneously transmit the information to the destination. In order to avoid interference between the various relay transmissions, each relay node spreads its information using a unique spreading code chosen from a set of sequences with good cross-correlation and autocorrelation properties. These properties allow separation of the signals in the presence of frequency and timing offsets between the various transmissions. In what follows, we provide a description of the various components at each relay transmitter and the destination receiver.

2.1 Transmitter

Each relay utilizes a unique spreading code to transmit its data. We adopt the spreading code used in IS-95 code-division multiple access (CDMA) standard, with each relay utilizing a different portion of the long spreading code. The various subblocks in the transmitter are as follows:

- Framing: The incoming data is collected to form frames with the first 6 bytes used as a preamble with known data useful for synchronization.
- Spreading: The data is then spread using the spreading code assigned to the particular relay. Each bit of the data is spread using a spreading sequence of length 16. Each data bit is modulated using binary phase shift keying.
- Pulse-shaping: In order to ensure the pulse is band-limited, we utilize a half-sine pulse to represent each data bit, i.e., a '+1' data bit is represented using the positive half of a sinusoid, and a '-1' is represented using the negative half of a sinusoid.

[2]The implementation of medium access control is beyond the scope of this paper.

2.2 Receiver

Fig. 1 shows the sequence of actions that the receiver takes in order to detect and decode a received packet. As shown in Fig. 1, the receiver calculates the received signal strength based on received signal strength indicator (RSSI) periodically until it detects arrival of a packet. As soon as the receiver detects the arrival of a packet it searches for all cooperating packets for the synchronizing bytes which then are used for coarse carrier recovery and code acquisition (CCRCA). Then it uses a cooperative Rake receiver (cf. Sec. 2.2.2) for demodulating and detecting the received packets. In the receiver design we have assumed

- The channel between a transmitter and the receiver is non-dispersive.
- We assume that transmitters are synchronized to within 8 bit intervals, corresponding to a differential delay of 40 us.
- No *a priori* carrier synchronization between the transmitters and the receiver, so that any carrier offsets and/or Doppler shifts due to mobility result in an overall frequency offset at the receiver.
- The noise at the receiver is assumed to be additive white Gaussian.
- We assume that the maximum frequency offset is 20 KHz. For example, an oscillator stability of 5 ppm (max) at a carrier frequency of 2.4 GHz results in a maximum carrier offset of 12 KHz.

The two major building blocks of the designed receiver are the CCRCA block and cooperative Rake block which are described below.

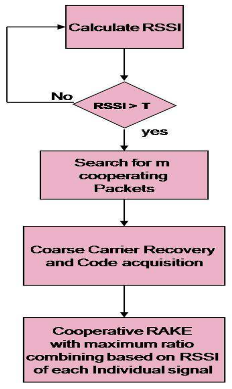

Calculate RSSI
No
RSSI > T
yes
Search for m cooperating Packets
Coarse Carrier Recovery and Code acquisition
Cooperative RAKE with maximum ratio combining based on RSSI of each Individual signal

Figure 1: Receiver flow chart.

2.2.1 *Coarse Carrier Recovery and Code Acquisition*

After detecting the arrival of a packet the receiver uses a sliding window to estimate the start of M cooperating packets (corresponding to the M relays). As mentioned earlier, we assume that the frequency mismatch between the local oscillator in the receiver and the combined effect of the Doppler frequency and frequency drift at the transmitter is between -20 KHz to 20 KHz, and therefore the receiver divides the 6 synchronization bytes into 6 streams of 8 bits. This is to avoid the error due to the phase rolling. The output of the sliding window correlator for the m-th cooperating signal is given as follows

$$C_m(\tau) = \sum_{B=1}^{6} \left| \left(\sum_{b=1}^{8} x^{(m)}(\tau + b + 8(B-1))^* y(b + 8(B-1)) \right) \right| \quad (1)$$

Where $x^{(m)}(\tau + b + 8(B-1))$ and $y(b + 8(B-1))$ are the complex baseband transmitted signal by the m-th relay delayed by τ samples and the received signal sample during $(b + 8(B-1))$-th bit in the packet, respectively. We should point out that the inner summation is done over 8 bits to limit the phase drift to 0.8 radian for the maximum frequency mismatch of 20 KHz. We then find τ^m_{max} that maximizes $C_m(\tau)$. τ^m_{max} is the optimum time shift for the m-th received packet. The range of τ in (1) depends on the memory available at the receiver and the maximum delay allowed in the communication link. In our design we have set the search space for τ so that 8 bits miss-synchronization between the cooperating transmitters is tolerated. After coarse synchronization, the carrier frequency is estimated by searching the frequency space from -20 KHz to 20 KHz with an increment of 1 KHz. Therefore the resolution of the coarse frequency offset estimation is about 1 KHz. The timing and frequency estimate from CCRCA unit is used in cooperative Rake where the timing and frequency of the received signal is tracked for each finger of the Rake.

2.2.2 *Cooperative Rake*

In cooperative Rake (Fig. 2) the coarse synchronization information from the Coarse Carrier Recovery and Code Acquisition unit is used to set the delays for each finger of the Rake. Each finger of Rake tracks the signal transmitted from one of the cooperative transmitters, therefore it should also track the frequency drift of that transmitter. For this purpose for each finger we need an independent carrier tracking circuit that is initialized using the coarse frequency estimate. The carrier recovery circuit we implemented uses decision feedback. The feedback for the carrier recovery is the result of maximum ratio combining of the output signals from all fingers. The carrier recovery circuit for each finger is depicted in Fig. 3.

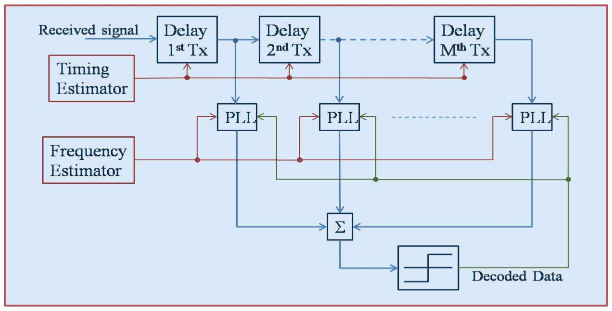


Figure 2: Cooperative Rake.

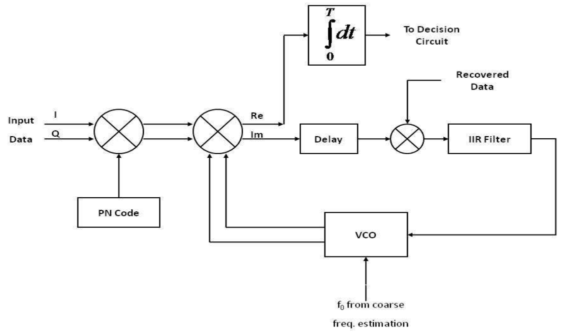


Figure 3: PLL.

2.3 Simulation Performance

2.3.1 *Floating Point*

The performance of the system for binary phase-shift keying (BPSK) modulation is given in Fig. 4(the non-smoothness in the graph is the result of Monte Carlo method). In all three cases the total power is kept the same, i.e. power of each individual transmitter for 2 (3) cooperative nodes is 1/2 (1/3) of the total transmit power. As it can be seen when the $\frac{E_b}{N_0}$ is low the performance of the cooperative communication is worse than the conventional communication system. This is because each individual transmitter uses lower power so the carrier recovery and synchronization does not work as well as the single transmitter case but as the $\frac{E_b}{N_0}$ increases the performance of cooperative communications is better than the conventional system. Also we should expect an error floor due to self interference. In fact, we are interested in the cases where bit error rate (BER) is around 10^{-3}-10^{-4} range where the cooperative communications shows a huge performance increase.

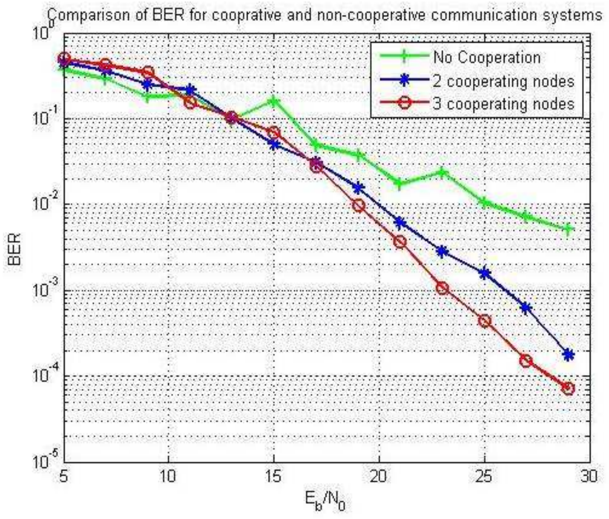


Figure 4: This plots the probability of a bit error (BER) with respect to the $\frac{E_b}{N_0}$ for different number of cooperating nodes.

3. HARDWARE IMPLEMENTATION

Prior to implementation of the cooperative transceiver on an FPGA-based software-defined radio, we need to determine the resolution of each adder and multiplier in every block. The process of determining the required resolution is an iterative process that can be implemented in Matlab. Once the resolution of each block is determined, the Matlab code is transferred to VHDL.

3.1 Transmitter

Based on the Matlab simulation model, the gate level architecture of the transmitter has been developed and implemented on the Virtex 5 FPGA development board. Multiple transmitters transmit the same data to a destination receiver, using a different PN spreading sequence. Different blocks of the transmitters are

- **Data Input**: This unit allows an external agent (e.g., Microcontroller, DSP) to read data from external memory and send data to the FPGA data pin.
- **Clock Input**: The clock signal is generated using the ICS 8442 clock synthesizer (available on the FPGA development board). We clock the system at 25 MHz frequency. The generated clock signal is a 2.5 V low-voltage differential signal (LVDS).
- **Packet Assembly** The main function of the packet assembly is to prepare the data packet for transmission. In this module, the preamble is appended to the transmit data to create the complete data packet. The preamble and transmit data info is read from the read-only memory (ROM) IP core modules using address generators. Once read, the data is stored in the FIFO.
- **Spreader Module** Once the packet is ready to be transmitted, one data bit is sent to the Spreader module at an interval of 64 clock cycles. In our test design, the spreading factor is 16 chips. For each chip, there are four quantized sample points. Hence, in order to properly synchronize the design, each data bit is transmitted in 64 clock cycles.
- **Pulse Shaper**: Once the spreading is done, the sequence is BPSK modulated signal. Each BPSK modulated sample is pulse shaped with the positive half of the sine wave. The 14 bit quantized half-sine wave values are hard-coded into the algorithm as a pulse template.

3.1.1 Register Transfer Logic (RTL) Simulation

In the RTL simulation (cf. Fig. 5) we can observe all the important outputs i.e. the output of the packet assembly (FIFO output), output of the spreader module (spread output) and the final output which is the BPSK modulated. We can see that there is a 64 clock cycle interval between each FIFO output bit. Within that time span, it can be observed that the spreader and the pulse shaper outputs are generated before the next data packet bit is transmitted. The waveform outputs of the Matlab simulation and the RTL simulation are comparable to each other. No performance degradation was observed.

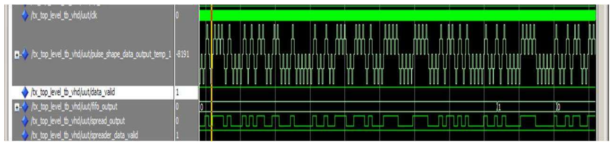

/tx_top_level_tb_vhd/uut/clk 0
/tx_top_level_tb_vhd/uut/pulse_shape_data_output_temp_1 -8191
/tx_top_level_tb_vhd/uut/data_valid 1
/tx_top_level_tb_vhd/uut/fifo_output 0
/tx_top_level_tb_vhd/uut/spread_output 0
/tx_top_level_tb_vhd/uut/spreader_data_valid 1

Figure 5: RTL Simulation.

3.2 Receiver

3.2.1 Coarse Synchronization

At the receiver the first step is to synchronize the receiver to the incoming signal, in both frequency and time. A correlator block called 'Sync Correlator' estimates the timing delay of each cooperating node in order to synchronize the subsequent decoding process. Fig. 6 shows the output of the correlator block for two cooperative transmitter blocks after summing. The maximum correlation value and its index point also can be seen. A six-sample delay is artificially added to the second transmitting node relative to the received signal from the first node.

Once the timing offsets are calculated for each node in the Sync Correlator block, the receiver attempts to remove the associated frequency offsets. This is done by the coarse frequency estimator that estimates the frequency offset of each relay transmission up to a resolution of 1 KHz.

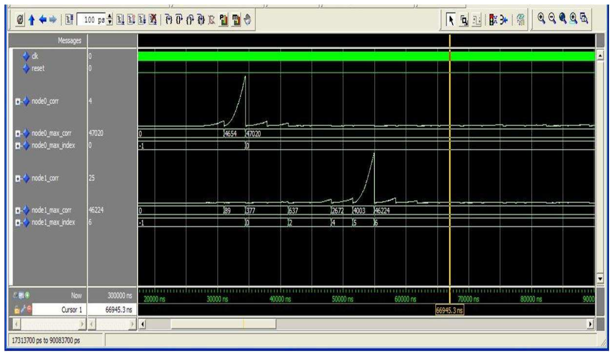


Figure 6: RTL simulation of Sync Correlator Output in a Two-node Coop Comm Operation.

3.2.2 Cooperative Rake

The coarse synchronization information from the coarse carrier recovery and Sync Correlator is used to set the delays for each finger of the Rake. Each finger of Rake tracks the signal transmitted from one of the cooperative transmitters, therefore it should also track the frequency drift of the corresponding transmitter. For this purpose for each finger we need an independent carrier tracking circuit that is initialized using the coarse frequency estimate. The carrier recovery circuit we implemented uses decision feedback. The feedback for the carrier recovery is the result of combining of the output signals from all fingers.

We simulated the VHDL implementation using Modelsim. Fig. 7 shows the simulation result for two transmitter nodes with 10 KHz and 18 KHz carrier offset. In this simulation we can see that the two PLLs finely tune to the offset frequencies and correctly recover the transmitted data (the recovered data bits matches with the original transmitted packet).

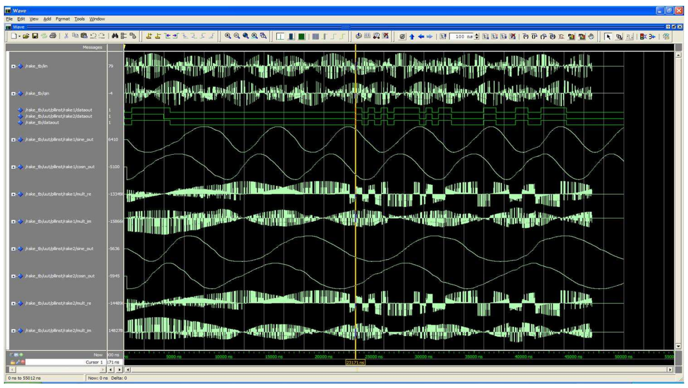


Figure 7: Modelsim simulation result for two cooperative nodes.

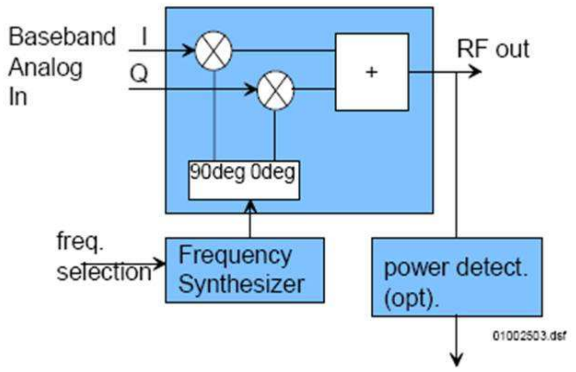
In Fig. 7, from top first we see in-phase ('I') and quadrature-phase ('Q') inputs with the pulse-shaped chips and Doppler shift. Below them we see the extracted data outputs of each finger followed by the combined decision output (i.e., final receiver output). The area on the left of the cursor shows the preamble section, therefore extracted data is mainly zeros (our preamble) except at the beginning where the PLL is locking. The next two waveforms are the outputs of the voltage controlled oscillator of node 1. Then, we see the real and imaginary parts of the complex multiplier output of node 1, where from the real part we can see the actual embedded data bits. Next four waves are for node 2.

4. RF BLOCK

In addition to the baseband test, we have tested the cooperative transceiver with an RF front-end. For transmit RF front end, we have used the ComBlock COM-4001 Dual Band 915 MHz/2.4 GHz Quadrature RF Modulator. On the receiver side, the ComBlock COM-3001 dual band 915 MHz/2.4 GHz receiver is used at the receive side where the 'I' and 'Q' test points are connected to the receiver FPGA's analog-to-digital converter board.

The ComBlock Control Center SW application is used to define the register settings of the two ComBlock RF modules. Using this graphical user interface, the RF frequency, gain control, local/external OSC are selected. Outlined below are the transmitter and receiver front-end settings along with their top level view.

- ComBlock COM-4001 Dual Band 915 MHz/ 2.4 GHz Quadrature RF Modulator:
  - RF Frequency is set at 915 MHz.
  - Gain control is set to 25 dB.
  - Local OSC is used.
- ComBlock COM-3001 Dual Band 915 MHz/ 2.4 GHz Receiver:
  - RF Frequency set at 915 MHz.
  - Local OSC is used.
  - 1 KHz pulse-width modulation signal is used for automatic gain control.



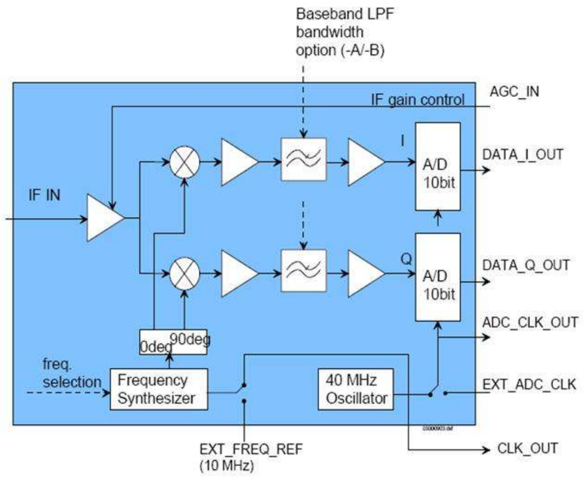
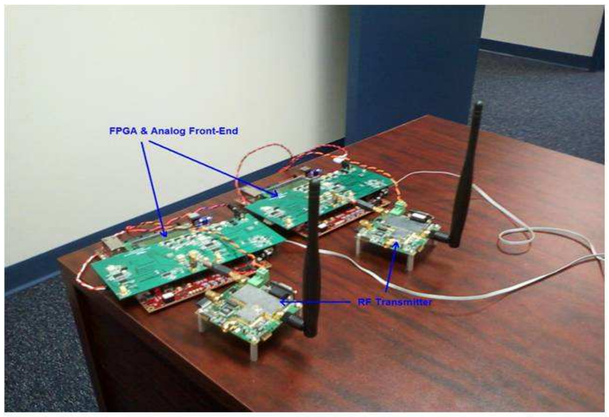


Figure 8: RF front end.

5. AUDIO APPLICATION

The goal of the audio application is to test a real application on the cooperative communication platform. In our case, the cooperating nodes transmit live audio data encoded using a Matlab script. The universal asynchronous receiver/transmitter (UART) interface is used to communicate data from the personal computer (PC) to the FPGA. This data is assembled in the packet format and then sent to the spreader and pulse shaping module. On the receiver side, after the threshold detector captures the valid transmitter data, the sync correlation and frequency estimation operations kick in. Once these modules are done, they provide the preamble and frequency offset values to the rake receiver. The rake receiver proceeds to recover the transmitted data and sends it back to the PC over the UART interface. On the PC, another Matlab script is put into action that reads the incoming serial port data, reformats it and then plays it back. This test allows us to demonstrate the various functionalities of the cooperative communication platform. Prior to implementation of live audio application, we tested the complete cooperative transmitter/receiver hardware with predefined data. In this test, some predefined data were generated in Matlab and then transferred to our two cooperative transmitter nodes through RS232 link at 115200 Kbps. The two transmitters sent the data to the receiver after spreading them with their individual codes.



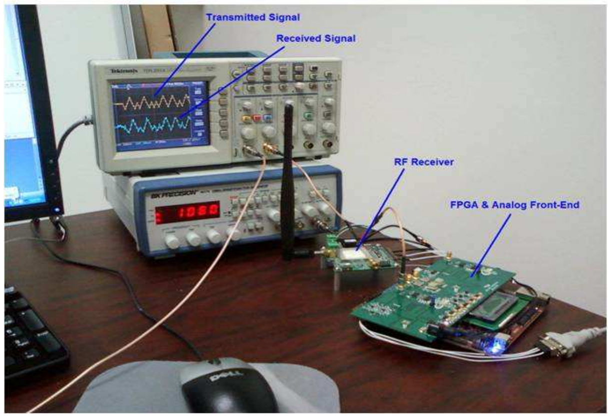


Figure 9: Up: Cooperative transmitters. Down: Receiver module with baseband signals..

At the receiver side, data were cooperatively detected and sent to a Matlab script via RS232 (at the same 115200 Kbps rate). We verified the received data which matched the original transmitted one. In our case, voice is sampled at 8000 bytes per second and our packet payload size is 80 bytes. So 100 packets are sent per second and there is a 10 ms delay between packets, which is required to ensure the receiver has enough time to run all of its coarse and fine computations.

6. RF TESTS

To be able to successfully test the end to end transmission we develop a novel RSSI measurement technique that determines if there is an incoming signal to be decoded, as opposed to using a technique that relies on received signal power. We developed a circuit to enhance the receiver trigger mechanism based on a one-bit correlator. Using the first 256 bits of the preamble as a reference, this correlator compares the received signal with the transmitted signal pattern that is quantized to 2 bits. A signal is deemed present if 3/4-th of the incoming bits match with that of the reference pattern. We tested the transmitter (2 nodes) and receiver hardware with RF front end in wireless mode, and this triggering mechanism greatly improved the system performance by reducing the packet drops or losses.

Fig. 9 shows the cooperative transmission test setup. We used the audio module to generate the transmit signal (Fig. 10). To verify the correct reception of the signal (in addition to a natural hearing test) we compared the transmitted and received signals in baseband on an oscilloscope to verify the RF up-conversion and down-conversion operation. Fig. 10 is a sample of these signals which shows a perfect match.

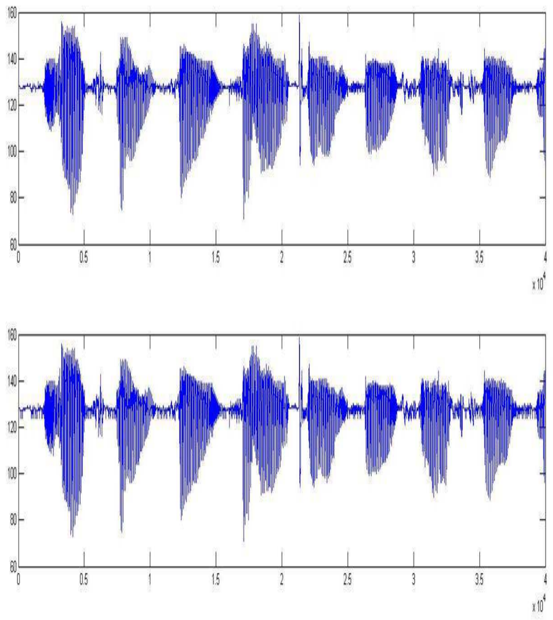


Figure 10: Transmitted and Received audio signals.

7. CONCLUSIONS

In this project we designed and implemented a spread spectrum based cooperative communication system. We implemented the base-band transceiver on an FPGA-based software defined radio. From a practical standpoint the frequency offset and time synchronization are two major roadblocks for hardware implementation. In order to overcome these roadblocks we designed a system that not only tracks the transmitted signals (frequency and timing) individually, but also combines the received signals to achieve cooperative diversity. This system was implemented on an FPGA with an RF front-end operating in 915 MHz frequency band, and tested using an audio application. Other potential applications include the use of the testbed in Cognitive Radios.

8. REFERENCES

[1] J. N. Laneman, D. N. C. Tse, and G. W. Wornell, "Cooperative diversity in wireless networks: Efficient protocols and outage behaviour," *IEEE Trans. Inf. Theory*, vol. 50, pp. 3062–3080, Dec 2004.

[2] J. N. Laneman and G. W. Wornell, "Distributed space-time-coded protocols for exploiting cooperative diversity in wireless networks," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2415–2425, Oct 2003.

[3] B. S. Mergen and A. Scaglione, "Randomized space-time coding for distributed cooperative communication," in *IEEE International Conference On Comm., ICC 2006.*, June 2006, pp. 4501–4506.

[4] S. Yiu, R. Schober, and L. Lampe, “Distributed space-time block coding,” *IEEE Trans. Commun.*, vol. 54, pp. 1195–1206, July 2006.

[5] L. Dai, B. Gui, and L. J. Cimini, “Selective relaying in OFDM multihop cooperative networks,” in *IEEE Wireless Networking and Communications Conference, WCNC 2007*, Mar. 11–15 , 2007, pp. 963–968.

[6] WARP Project, “http://warpproject.org”

[7] Z. Per, M. Christos, L. S. Aris, and M. Emmanouil, “Experimental investigation of cooperative schemes on a real-time dsp-based testbed,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2009, pp. 1–15, 2009.

[8] J. Zhang, J. Jia, Q. Zhang, and E. M. K. Lo, “Implementation and evaluation of cooperative communication schemes in software-defined radio testbed,” in *Proc. IEEE INFOCOM 2010*, Mar.15–19, 2010.