Implementation of OFDM-based Superposition Coding on USRP using GNU Radio

Zhenhua Gong, Chia-han Lee, Sundaram Vanka, Radha Krishna Ganti, Sunil Srinivasa, David Tisza, Peter Vizi, and Martin Haenggi

Department of Electrical Engineering
University of Notre Dame
Sept. 2009

Abstract

This report summarizes our PHY layer implementation of a OFDM-based superposition coding system. In theory, multi-user techniques such as superposition coding (SPC) are known to improve throughput in wireless networks. However, in order to understand their practical limitations, it is imperative to actually implement and experiment with such techniques in a realistic setting. In this report, we describe the physical layer design of superposition coding system in software defined radio. Hardware issues in the implementation are also discussed.

1 Introduction

Superposition coding (SPC) [1] is one of the specific PHY techniques proposed in information theory society. Suppose two messages destined to two users are queued up for transmission at a base station. Instead of splitting transmit resources as space/time/frequency division multiplexing, a superposition encoder jointly encodes both the messages under the same total transmit power constraint as in multiplexing schemes. Therefore, both messages are sent simultaneously and share the transmitter resources. Message symbols of one user will appear as interference to the other, and from an information-theoretic
standpoint, both the receivers can decode their messages by successive interference cancellation. For broadcast channels, it has been shown that this scheme improves aggregate user throughput compared to an orthogonal multiplexing scheme [1].

Theoretically, superposition coding scheme provides a nice bound of extended capacity in the broadcast channels. However, such performance has not yet been evaluated in the experiment. It is because that such performance gain is under the strict assumptions such as perfect synchronization and error-free feedback, which may not be practical in some cases. Hence, it is the main reason we implement the system and evaluate the performance.

Hardware implementation was widely used in the past. However, one of the drawbacks is that it is hard to do reconfiguration in the implementation. Instead, software defined radio (SDR) can implement the physical layer architecture in software. SDR can be easily modified and it does not require sophisticated hardware programming knowledge. GNU Radio [2] is an open-source software toolkit for deploying SDR. It uses Python language to connect different signal processing blocks written in C++, and its SWIG library provides interface between the two languages. While not primarily a simulation tool, GNU Radio also supports development without using RF hardware. In our implementation, GNU Radio is used for physical layer design.

In addition to the software toolkit, Universal Software Radio Peripheral (USRP) is used as RF front-end. USRP, a product of Ettus Research LLC, is specifically designed for GNU Radio. It performs digital up/downconversion of the signal and communicates with the PC via USB port (for USRP1, the first generation USRP) or Ethernet (for USRP2, the second generation USRP). In our implementation, we use USRP2 since it has several advantages over USRP1, which will be discussed later.

The structure of this report is the following: we first introduce theoretical background in Section 2; physical layer implementation of the transceiver is discussed in Section 3; system performance will be shown in Section 4; the lessons learned from the transceiver implementation are addressed in Section 5; finally, we conclude the report in Section 6.
2 Theoretical Background

Consider a base station $B$ that wishes to independently communicate with two users $N$ (a “near” user) and $F$ (a “far” user) over AWGN channels. Assume that the (point-to-point) channel between $B$ and the near user $N$ has a higher capacity than from $B$ and $F$. Such a scenario can arise, for example, in a cellular system where one user is close to the base station and the other near the cell edge. One of the key questions is: What is the communication scheme that can achieve the maximum possible transmission rates to both users?

In information theory, this question is answered by formulating the problem as that of communication over *degraded* broadcast channels [1]. The capacity region, which is the set of all simultaneously achievable rates for both users, is known. All these rates can be achieved using *superposition coding*, i.e., the superposition (and simultaneous transmission) of the encoded messages of all the users. The idea is as follows: allocate most of the power to users with bad channels to the base station. Users with better channel capacities can always decode messages meant for those with poorer channels and can thus effectively cancel interference from those messages in their received signal(s). A brief description of the decoding strategy follows; details can be found in [1].

Denote by $X_N$ ($X_F$) the encoded near (far) user’s signal, each with unit power. Let the total power of the base station $B$ be unity, of which a fraction $\alpha$ (resp. $1 - \alpha$) is given to the far (resp. near) user. Since the channel from $B$ to $N$ is AWGN, the near user observes $Y_N = \alpha X_F + (1 - \alpha) X_N + W_N$ where $W_N$ denotes WGN with variance $\sigma^2$. Since the channel from $B$ to $N$ has a higher capacity than that from $B$ to $F$, the near user $N$ can decode the far user’s message. After canceling the interference $\alpha X_F$, the near user observes $Y_N - \alpha X_F = (1 - \alpha) X_N + W_N$ which can be used to decode the near user’s message. The far user, on the other hand, can only decode its own message but not the near user’s message.

3 Physical Layer Implementation

Figure 1 shows the physical layer design of OFDM-based superposition coding system. In the transmitter, the payload pair (near and far user data in bits) is provided to the physical layer by the higher layers. Data bits of the near and far users are encoded separately by passing through a scrambler, a channel encoder, and an interleaver. The SPC modulator then multiplexes these two streams of
bits and maps to SPC constellation. Pilot tones are added afterward, followed by the OFDM modulator performing IFFT to each OFDM symbol. Preamble sequence and channel estimation symbols are then added at the beginning of the packet before sending to the USRP for transmission.

At the receiver, the timing recovery block utilizes the cross-correlation to find the beginning of the packet, and then the frequency recovery, including coarse and fine frequency tracking, is employed. Channel estimation is also performed as this stage. The OFDM demodulator simply performs the FFT operation and then the SPC demodulator de-maps the near and the far user data, which are later decoded using a Viterbi decoder.

For the rest of this section, we discuss each block in detail, starting with the transmitter.
3.1 Transmitter

3.1.1 Channel coding

The standard convolutional code in IEEE 802.11a is chosen as the channel codec [3]. This encoder includes three stages: data scrambling, convolution coding, and data interleaving. The data scrambler uses generator polynomial \( S(x) = x^7 + x^4 + 1 \) with “all ones” (1111111) as the initial state. The 127-bit binary sequence is used repeatedly to be XORed with the data bit sequence. The output of the scrambler is sent to a rate 1/2, \( K = 7 \) convolutional encoder with generator polynomials \( g_0 = 133_8 \) (1011011) and \( g_1 = 171_8 \) (1111001). The encoded data bits are then passed to an interleaver with the block size corresponding to the number of bits in a single OFDM symbol. The interleaver is defined by a two-step permutation. The first permutation ensures that the adjacent coded bits are mapped onto nonadjacent subcarriers, while the second permutation ensures that the adjacent coded bits are mapped alternately onto less and more significant bits of the constellation and, thereby, long runs of low reliability (LSB) bits are avoided. Detailed index mapping algorithms for the interleaver and the deinterleaver are available in the IEEE 802.11a standard [3].

3.1.2 SPC modulator

The near and the far users individually choose standard modulation schemes such as PSK or QAM to map bits into symbols. Superposition coding scheme multiplexes these two modulations with different ratio of power. Figure 2 shows the case of a BPSK over BPSK superposition coding scheme. For a QPSK over QPSK, the SPC constellation looks similar to a 16-QAM.
3.1.3 Pilot tones

Our system uses 16 subcarriers for OFDM. However, not all of these subcarriers are used for data transmission. For example, fine frequency tracking requires a few subcarriers to be reserved as pilot tones. In addition, experimental results showed that some of the subcarriers always have bad channel response, so we leave those subcarriers unused. At the end, we use eight subcarriers for data for the sake of easy representation of data bits in one OFDM symbol using only one byte. The usage map of the 16 subcarriers is thus

\[
\text{Usage Map} = -1 \ 1 \ 1 \ -1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ -1 \ 1 \ 1 \ -1 ,
\]

(1)

where -1, 1, and 0 stands for the subcarrier being reserved for pilot tones, data, and unused, respectively.

3.1.4 Preamble

The packet structure is depicted in Fig. 3. The preamble sequence is used for frequency and timing recovery and is designed by repeating a pseudo-random training sequence of length 24 symbols (TS1) twice. The channel estimation symbols are used for performing equalization and are generated by repeating a 16-symbol pseudo-random sequence three times. The length of the cyclic prefix is 4 symbols (4 \(\mu s\)). A 32-bit CRC is added at the end of the each packet for checking packet errors.
3.2 Receiver

3.2.1 Timing recovery

As shown in Fig. 3, the preamble consisting of \( L \)-bit (\( L = 24 \) in our design) training symbol is repeated twice at the beginning of the packet. At the receiver, the cross-correlation between the first half and the second half of these \( L \) samples is given by [4]

\[
P(d) = \sum_{m=0}^{L-1} (r_{d+m}^* r_{d+m+L}),
\]

(2)

where \( d \) is a time index corresponding to the first sample in the total length of samples \( L \). (2) needs to be further normalized by the auto-correlation of the second half of the samples, which is defined by

\[
R(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2.
\]

(3)

Now we can define a timing metric by

\[
M(d) = \frac{|P(d)|^2}{(R(d))^2}.
\]

(4)

When \( M(d) \) exceeds a certain threshold, we announce that the packet starts from the position \( d \). The metric threshold is set to 0.2.

3.2.2 Coarse frequency recovery

After finding the starting point of the packet, frequency recovery is achieved using the Schmidt-Cox algorithm [4]. The main difference of the first half and the second half of the \( L \) samples is the phase difference \( \phi = \pi T \Delta f \), which can be estimated by

\[
\hat{\phi} = \text{angle} (P(d)).
\]

(5)

If \( |\hat{\phi}| \) is less than \( \pi \), then the frequency offset estimation is given by

\[
\hat{\Delta f} = \frac{\hat{\phi}}{\pi T},
\]

(6)

where \( \pi T \) is the correlation length of the training symbol.
3.2.3 Equalization

Equalization is performed using the channel estimation symbols. The received channel estimation symbols are compared with the known values of the transmitted symbols to obtain the channel gain of each subcarrier. Compensation is then applied to each subcarrier.

3.2.4 Fine frequency tracking

As we found in the experiment, USRP systems suffer from additional fine frequency offset. A small frequency offset will be accumulated over long packets and cause the bits near the end of the packet to suffer from severe synchronization problem, harming the bit error rate (BER) badly.

In order to overcome that offset, the fine frequency tracking algorithm uses four of the sixteen subcarriers in each OFDM symbol as pilot tones to correct the frequency offset. The pilot sequence is a pseudo-random sequence of 1’s and -1’s. After the coarse frequency tracking, OFDM demodulation is performed in order to extract the pilot tones. By cross-correlating the received pilot tone sequence with the known pilot sequence, the residual frequency offset can be estimated [5, 6]. After compensating the frequency offset, channel estimation is performed again before doing the OFDM demodulation for data.

Figure 4 shows an example of the error distribution for far and near users for the first 512 bits of a packet. The x-axis represents the bit index in a packet and the y-axis represents the average number of errors at the corresponding bit index. Without fine frequency tracking, the residual frequency offset makes the BER higher toward the end of the packet. After applying the fine frequency tracking, the BER is distributed uniformly. It shows that the fine frequency offset needs to be compensated in order to achieve good BER performance.

3.2.5 Decoding

The decoding of the convolutional code is the reversed process of the encoding: deinterleaving, convolutional decoding, and followed by the descrambling. The deinterleaving algorithm is documented in detail in the IEEE 802.11a standard [3]. The Viterbi algorithm is used in the decoder and the traceback length is the data length. Finally, the same scrambling sequence is used for the descrambling and the output is obtained by XORing the decoded bits with the scrambling sequence.
Figure 4: Bit error distribution in the packet for (a) far user and (b) near user.
<table>
<thead>
<tr>
<th>System bandwidth</th>
<th>1 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center frequency</td>
<td>903 MHz</td>
</tr>
<tr>
<td>Modulation</td>
<td>BPSK</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.8</td>
</tr>
<tr>
<td>Data rate</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>Payload</td>
<td>508 bytes</td>
</tr>
</tbody>
</table>

Table 1: Parameters used in the experiments.

Figure 5: Depiction of the relative locations of the radios in the implementation set-up. These distances were chosen so that both the receivers lie in the far-field of the transmitter and observe different SNRs.

4 System Performance

The experimental setup for testing the performance of the implemented system consists of a transmitter and two (near and far) receivers. Each transceiver is a software radio along with a USRP2 and a PC. Fig. 5 shows the experimental setup with the relative locations of the three USRP2’s. Since the gain control in the USRP must be adjusted manually, we sometimes encountered the saturation problem during the experiments. As a result, we need to adjust the transmitter power and choose the USRP locations carefully. Table 1 shows the parameters we used in the experiment. Since the 32-bit CRC is added at the end of each packet, the coded packet length is 1024 bytes.

Fig. 6 plots the PER versus SNR performance of the SPC system for both the near and far users when $\alpha = \frac{4}{5}$. The experiment was conducted indoors in our laboratory. Since more power is allocated to the far user, the far user has better PER. The figure shows that our SPC system gives very good performance (PER achieves $10^{-2}$).
Figure 6: PER versus SNR for the near and far users for $\alpha = \frac{4}{5}$. Note that the SNR is defined as the measured preamble power divided by the noise power. Since the far user is allocated more power than the near user, its PER is lower. Both the far and the near user achieve a good performance (PER $< 10^{-2}$) at moderate SNR.

5 Lessons Learned

We now briefly overview some issues involved in the implementation of the SPC system using GNU Radio.

5.1 Reuseability versus Efficiency

The GNU Radio scheduler was designed for a flow-based framework, i.e., each block operates on a stream of data rather than packets of data. This makes the design of a frame-based system quite challenging. Moreover, implementing feedback loops between signal processing blocks is cumbersome. One solution is to implement all the functions in one big GNU Radio block, but this sacrifices the reusability of the functional blocks. From our experience, implementing all the functions in one GNU Radio block significantly simplifies the design.
5.2 Software Interpolation Filter

The effective bandwidth of the USRP1 is much smaller than that set by the user. The cause of this problem lies in the highly non-ideal transmit path implementation of the USRP1. The DAC’s on the transmit path are designed to operate at a fixed frequency of 128 MHz [2]. Therefore, any digitally synthesized signal at a lower bandwidth to be input to the DAC’s must be interpolated to 128 MHz. The baseband signals generated by the GNU Radio must be interpolated in the USRP in order to achieve the target bandwidth and then be upconverted to the desired RF band. For example, the interpolation rate is set to 128 for 1 MHz bandwidth, 256 for 0.5 MHz bandwidth, 64 for 2 MHz bandwidth, and etc. However, we observe that the USRP1 uses a rather simplistic scheme to implement this interpolation, so it shows a poor passband response (see Fig. 7). Such a frequency response causes significant degradation of subcarrier SNR as one moves away from the DC subcarrier. Similar problems were reported recently elsewhere [7]. For OFDM systems, that means the frequency response of the subcarriers is not flat, and it causes huge difference in bit error rate among different subcarriers.

The solution for this problem is to introduce a software-based interpolation filter at the GNU Radio which has the interpolation rate 2 or 4, and then the USRP1 filter does interpolation of 64 or 32 to achieve total interpolation rate of 128 for 1 MHz bandwidth. The software interpolation filter is realized using the GNU Radio class “gr_rational_resampler_base”. With the help of this software filter, we are able to get a much flatter frequency response.

Figure 8 compares the bit error rates of the eight data subcarriers with and without software interpolation filter. In the experiment, the software interpolation rate is set to 2 and the USRP1 interpolation rate is 64. The measured SNR is 16 dB. Without the software interpolation filter, the error rate varies significantly from subcarrier to subcarrier, with the worst one having bit error rate ten times larger than the best one. The error rates are almost the same for every subcarrier when the software interpolation filter is applied.

In USRP2, this problem seems to have been alleviated. The frequency response of RF signals transmitted by USRP2 is much flatter than by USRP1. Hence, software interpolation is not necessary in USRP2, and this is why we use USRP2 for experiments. However, the performance of the middle four tones of the OFDM symbols is still worse than others.
Figure 7: Frequency response of the USRP TX path for different interpolation factors. When using USRP1, we used an interpolation factor of 128, which corresponds to a bandwidth of 1 MHz. We observe that the 3-dB bandwidth is about 450 kHz for an interpolation factor of 128. We also observe that this frequency selectivity is consistent over different interpolation factors.

Figure 8: Comparison of error rate for each data subcarrier with and without software interpolation filter.
6 Conclusions

Superposition coding has theoretically excellent performance, but it is rarely evaluated by experiments. In this report, we have introduced the physical layer implementation of an OFDM-based superposition coding system in software defined radio. Detailed discussion of the functional blocks, packet structure, and receiver algorithms has been presented. Several hardware issues such as nonflatness of the RF bandwidth and the fine frequency offset also have been addressed, and feasible solutions have been provided to deal with such issues. We plan to use this platform to conduct further experiments involving superposition coding.

References


