Mitagation against MAI in a space time spreading software defined radio

Shinhan Wee

Montse Ros  
*University of Wollongong*, montse@uow.edu.au

Peter James Vial  
*University of Wollongong*, peterv@uow.edu.au

Follow this and additional works at: https://ro.uow.edu.au/infopapers

Part of the Physical Sciences and Mathematics Commons

**Recommended Citation**

Wee, Shinhan; Ros, Montse; and Vial, Peter James: Mitagation against MAI in a space time spreading software defined radio 2009, 534-540.  
https://ro.uow.edu.au/infopapers/795

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
Mitigation against MAI in a space time spreading software defined radio

Abstract
A Software Defined Radio test bed using the Gnu Radio project was installed on Unix boxes and modified so that estimates of the channel state coefficients were taken for a Multiple Input Multiple output (MIMO) system to take advantage of space time transmission at a frequency of 2.4GHz. This system was modified to provide a Space Time Spreading test bed. The test bed was modified so that multiple access interference was experienced by offsetting different users data streams. The Walsh-Hadamard and Wysocki (low correlation) spreading sequences were deployed in the test bed to compare their Bit Error Rate performance. We confirmed that the low cross correlation spreading sequences experienced an improved Bit Error Rate compared to that obtained when using the Walsh-Hadamard Spreading sequence for high signal to noise ratios.

Keywords
software, defined, radio, space, mai, time, against, spreading, mitagation

Disciplines
Physical Sciences and Mathematics

Publication Details

This conference paper is available at Research Online: https://ro.uow.edu.au/infopapers/795
Mitigation against MAI in a Space Time Spreading Software Defined Radio Test Bed

Shinhan Wee, Montse Ros, Peter James Vial
University of Wollongong

Abstract
A Software Defined Radio test bed using the Gnu Radio project was installed on Unix boxes and modified so that estimates of the channel state coefficients were taken for a Multiple Input Multiple output (MIMO) system to take advantage of space time transmission at a frequency of 2.4GHz. This system was modified to provide a Space Time Spreading test bed. The test bed was modified so that multiple access interference was experienced by offsetting different users data streams. The Walsh-Hadamard and Wysocki (low correlation) spreading sequences were deployed in the test bed to compare their Bit Error Rate performance. We confirmed that the low cross correlation spreading sequences experienced an improved Bit Error Rate compared to that obtained when using the Walsh-Hadamard Spreading sequence for high signal to noise ratios.

Index Terms—Gnu Radio, Space Time Spreading, MIMO, Software Defined Radio

1. Introduction

The phenomenon which makes reliable wireless transmission difficult is time-varying multi-path fading. In most environments, antenna diversity is a practical, effective and a widely applied technique for reducing the effect of multi-path fading [1]. With the antenna diversity technique, there is no need to increase the transmitted power or sacrifice bandwidth to obtain better performances [2].

The conventional approach is to use a multiple antennas at the receiver, known as maximal-ratio receiver combining (MRRC) to improve the quality of the received signal. This technique is acknowledged as a powerful technique to mitigate against the effect of fading and shadowing for data transmission over wireless fading channels [3].

However, a major problem with the receive diversity approach (MRRC) is the cost, size and power of the receiver units. The use of multiple antennas and radio frequency (RF) switching circuits makes remote units larger and more expensive, therefore unfavourable for end users [1].

As a result, transmit diversity techniques have been applied to base stations where it has been proven to improve the reception quality at the receiver [1]. A base station often serves hundreds to thousands of remote units (mobile stations, typically) and it is therefore more economical to add equipment to base stations rather than the remote units. For this reason, transmit diversity schemes are very attractive [2].

The Space Time Spreading (STS) technique was initially proposed by [4] as providing a diversity gain using multiple orthogonal spreading codes. A STS test bed was constructed using the Gnu Radio equipment from www.ettus.com [5] to study the effect of Multiple Access Interference (MAI) on such a system as simulation studies had shown that improvement in BER performance was possible when using low correlation spreading codes [6,7]. This paper outlines the results measured from the STS test bed in the presence of MAI.

Section 2 briefly describes the Space Time Spreading (STS) technique proposed by [4]. Section 3 describes the Software Defined Radio equipment used in the test bed. Section 4 describes the experimental setup to induce MAI in the MIMO STS SDR test bed and the measured results. Section 5 provides conclusions and future work.

2. Space Time Spreading

STS is classified as an open loop transmit-diversity system, where there is no knowledge at the transmitter about the state of the channel between the transmitter and the receiver. This information is only needed at the receiver, and estimates of the complex channel coefficients can be found from associated pilot signals sent by the transmitter to the receiver [6, 8]. In [4], STS systems were shown to be efficient through the use of a limited number of orthogonal spreading sequences to provide a diversity gain. In the case of two transmitter antennas and one receiving antenna, the diversity gain is of order two. A STS receiver is also able to reconstruct
data bits every symbol period. Therefore, theoretically a STS system is twice as fast as an Alamouti system. In addition, if there is a deep fade on one of the two transmit antennas to receive antenna channels both transmitted bits can be recovered at the cost of a increased BER performance. This contrasts with other space time systems such as the Alamouti schema[1] where one or more bits would be lost depending on the length in time of the fade. In such circumstances forward error correction coding would be required to recover the data or retransmission of the data will be required.

2.1. The STS Transmitter

Before transmission, the STS scheme [8, 9] separates the intended binary data stream into odd and even symbols, identified as \(b_1\) and \(b_2\). The data would then be encoded for two transmit antennas as follows:

Data to be transmitted from antenna0:

\[
t_1 = \left(\frac{1}{\sqrt{2}}\right) (b_1c_1 + b_2c_2)
\]

(1)

Data to be transmitted from antenna1:

\[
t_2 = \left(\frac{1}{\sqrt{2}}\right) (b_1c_1 - b_2c_2)
\]

(2)

Where \(c_1\) and \(c_2\) are the orthogonal spreading codes used and \(b_1\) and \(b_2\) are the odd and even bit streams respectively. The multiplicative coefficient of \(1/\sqrt{2}\) is used to normalize the total transmission power of the transmit antennas to that of using one transmit antenna.

2.2. The STS Receiver

At the receiver, a superposition of the signals of \(t_1\) and \(t_2\) will be received in each symbol period. The following notations are used in [7]:

\[
\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} d_1 & d_2 \end{bmatrix}^T
\]

(3)

And,

\[
\mathbf{H} = \begin{bmatrix} h_1 & h_2 \\ -h_2 & h_1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad \mathbf{\nu} = \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}
\]

(4)

Where, \((\cdot)^H\) stands for the Hermitian or Conjugate transpose, \(\mathbf{\nu}\) is a \(N \times 1\) vector representing the channel noise, and \(h_0\) and \(h_1\) are the channel complex coefficients of antenna 0 and antenna 1. With reference to equations 3 and 4, the received signal vector \(\mathbf{d}\) can be represented at the receiver by:

\[
\mathbf{d} = \frac{1}{\sqrt{2}} \mathbf{H} \mathbf{b} + \mathbf{\nu}
\]

(5)

If this matrix equation is expanded out we get:

\[
d_1 = \frac{1}{\sqrt{2}} (h_1b_1 + h_2b_2) + \zeta_1^H \mathbf{\nu}
\]

(6)

\[
d_2 = \frac{1}{\sqrt{2}} (-h_2b_1 + h_1b_2) + \zeta_2^H \mathbf{\nu}
\]

(7)

Assuming a good signal to noise ratio (SNR), (e.g. by transmitting at a higher power) the channel noise can be ignored:

\[
d_1 = \frac{1}{\sqrt{2}} (h_1b_1 + h_2b_2)
\]

(8)

\[
d_2 = \frac{1}{\sqrt{2}} (-h_2b_1 + h_1b_2)
\]

(9)

From equations 8 and 9 we then get \(\tilde{b}_1\) and \(\tilde{b}_2\) (in complex form):

\[
\tilde{b}_1 = \frac{1}{\sqrt{2}} \left( \frac{h_1d_1 - h_2d_2}{|h_1|^2 - |h_2|^2} \right)
\]

(10)

\[
\tilde{b}_2 = \frac{1}{\sqrt{2}} \left( \frac{h_1d_2 - h_2d_1}{|h_1|^2 + |h_2|^2} \right)
\]

(11)

Finally, \(b_1\) and \(b_2\) can be reconstructed at every symbol period using the decision rule:

If \(\Re \tilde{b}_1\) \(\geq 0\) then \(b_1 = \text{bit 1}\)

If \(\Re \tilde{b}_1\) \(\leq 0\) then \(b_1 = \text{bit 0}\)

(12)

3. The Software Defined Radio

Universal Software Radio Peripheral (USRP) is an FPGA-based hardware component that creates the possibility of developing software defined radio (SDR) by acting as an RF frontend for a computer running GNU Radio. The USRP converts radio waves picked up by an antenna into a digital form suitable for processing on the host computer [9].

The USRP is produced by ETTUS Research LLC especially for the use of GNU Radio software and contains four high-speed, 64 mega samples-per-second (MS/s), 12-bit analog-to-digital (ADC) converters and four high-speed, 128 MS/s, 14-bit digital-to-analog (DAC) converters. A support circuitry including a high-speed USB 2.0 interface was also included in a USRP. [5, 10]
The Field Programmable Gate Array (FPGA) built into the USRP is the Altera Cyclone FPGA. The FPGA plays an important role on the USRP: it performs high bandwidth mathematics and reduces the data rates of incoming signals, such that they can be transmitted to the host PC via the USB2.0 port [5, 10].

As shown in Figure 2, all the ADCs and DACs are connected to the FPGA. The standard FPGA configuration includes digital down converters (DDC) implemented with cascaded integrator-comb (CIC) filters. CIC filters are high-performance filters using only adds and delays [10].

The receiving (RX) path consists of four ADCs and four DDCs. Each DDC has two inputs, I (inphase, real part of complex signal) and Q (quadrature, imaginary part of complex signal). Through the use of the FPGA multiplexer (MUX), each of the 4 ADCs can be routed to either I or Q input of any of the 4 DDCs. This allows a maximum of four multiple channels to be selected out of the same ADC sample stream. At the transmitting (TX) path, there are four DACs. The digital up converters (DUC) on the transmit side of the USRP are contained in the AD9862 CODEX chips and not in the FPGA. The only transmit signal processing block in the FPGA are the interpolators. By using the FPGA MUX, the interpolator outputs can be routed to any of the four AD9862 inputs[10]. This would allow up to four multiple transmit channels to be selected out of the same DAC sample stream.

The RFX2400 Transceiver daughter boards are specialized boards used to hold the RF transmitter and receiver interface of the USRP. These boards could be used for any software defined radio operating in the frequency range of 2.3-2.9 GHz. We used a frequency range of 2.4 to 2.483 GHz.

The antennas used by the SDR are the PATCH2400. These PATCH2400 antennas are manufactured specifically for use with the RFX2400 transceivers. The PATCH2400 is a vertically polarized antenna; rated to have a gain of 7dBi for ISM band frequencies between the frequency ranges of 2400-2480 MHz [11,12].

Table 1 lists the hardware components required for a MIMO system using the SDR hardware.

<table>
<thead>
<tr>
<th>Hardware components</th>
<th>Quantity</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers (Ubuntu OS)</td>
<td>2</td>
<td>Both computers must have GNU Radio installed One PC used as transmitter Another PC used as receiver</td>
</tr>
<tr>
<td>USRP main boards</td>
<td>2</td>
<td>One transmitting USRP and One receiving USRP</td>
</tr>
<tr>
<td>RFX2400 boards</td>
<td>3</td>
<td>Two TX slots for two transmit antennas One RX slots for one receive antennas</td>
</tr>
<tr>
<td>PATCH2400 antennas</td>
<td>3</td>
<td>Two transmit antennas One receive antennas</td>
</tr>
</tbody>
</table>

Table 1: MIMO systems hardware requirements
Figure 4 shows a block diagram of the hardware as it was used in the STS test bed. Figure 5 shows the physical environment in which all tests were conducted.

Figure 4: Block diagram on SDR MIMO hardware setup

The software used in the developed SDR MIMO STS system was developed from the GNU Open Source radio project. GNU Radio is a free software toolkit for building and deploying Software Defined Radios. The GNU Radio project has a large community of developers and users that have contributed to a substantial code base and provided many practical applications for the hardware and software. Reconfigurability is a key feature with all software-defined radio systems. Therefore, instead of purchasing multiple expensive radios, a single more generic radio is purchased, which provides a usable signal for powerful signal processing software [13, 14].

GNU Radio is a very flexible system and utilises two programming languages, C++ and Python [15]. Three layers are defined in GNU Radio (see Figure 6), they are:

1) Application layer: The Python script where the user runs his/her instructions to the SDR;
2) Python Layer: This is a collection of Python scripts that connect different C++ signal processing blocks together. The Python layer passes the digitised signal to and from the C++ blocks. Most of the software coding developed for this project is done at this layer; and
3) C++ layer: A collection of signal processing blocks which comes with the installation of GNU radio. An expert user can attempt to write a new C++ block for different application needs. Each block is a class implemented in C++. These blocks are the basic elements of any SDR developed using GNU radio, upon receiving a digital signal from the Python layer, a C++ block will perform the necessary processing and pass the signal to another block through the Python layer.

The MIMO transmitter was modelled from a GNU Radio SISO example code (benchmark_tx.py). The various signalling blocks used in the transmitter are classes implemented in C++ which comes with the installation of GNU Radio [14]. For the developed MIMO STS SDR system, all programming was performed at the python layer.

GNU Radio has libraries for many common software radio needs, including various modulations (BPSK used for this project), signal processing constructs (optimized filters, FFTs, equalizers, timing recovery), and scheduling.

Figure 5: Physical location of SDR

Figure 6: GNU radio software layers

The block diagram shown in Figure 7 illustrates the complete data flow path between the different signal processing blocks used in the transmitter SDR. The arrows representing the signalling links connect each block using the fg.connect flow chart function in the Python layer.

Figure 7: Block diagram of the MIMO transmitter
When the user runs the transmitter script, the USRP sink will first identify and enable both RFX2400 sub-devices through the function setup_usrp_sink. This command is performed once outside the main transmit loop and only every time the script is run. In the transmit loop, any data streams to be transmitted will be stored in two variables named payload0 (antenna0) and payload1 (antenna1). These two variables are then relayed to the Modulator block. The Modulator module is a large library of different modulator blocks. It processes the payload according to the instructions or options defined by the user in the main script. For this project, the modulation scheme used is binary phase shift keying (BPSK).

After modulation, the encoded bits are channelled to the two packet transmitter blocks where packet transmitter0 and packet transmitter1 will be responsible for the signal processing of payload0 and payload1 respectively. This arrangement is necessary in order to define two unique transmit paths. At each of the two packet transmitter, there is a Python level function called mod_packet. This function will perform two tasks:

1) mod_packet will attempt to create a packet with a length of 512 bytes (4096 bits). Therefore, the length of any data the user intends to send must not be greater than 4096 bits. If the data is smaller than 4096 bits, mod_packet will perform zero padding to be appended to the end of the intended data.

2) mod_packet will represent each bit of the data with 16 bytes of I-Q samples, which effectively means 16 complex numbers.

After the packet transmitter block, the modulated samples from both paths will be transferred to an amplifier. Here the samples are digitally multiplied by a constant number. This constant can be influenced by the user through the option (-tx-amplitude constant). The default value is 12000, which is normally sufficient for reliable communication. The amplified samples from signal amplifier 0 and 1 will next be channelled to port 0 and 1 of the Time-Interleaver block where samples from both paths are interleaved. After interleaving, the two streams of interleaved samples will be transferred to the USRP sink. In the USRP sink, two functions (send_pkt0 and send_pkt1) were defined to force payload0 to be transmitted through antenna0 and payload1 to be transmitted using antenna1. Finally, the MUX module in the USRP sink will ensure that both payloads are transmitted from their respective antennas simultaneously.

The SDR receiver is expected to receive a superpositioned payload (payload0 and payload1) and will attempt to reverse the signal processing that was done in the transmitter SDR. Figure 8 shows the block diagram of the connections between the different signal processing blocks.

![Figure 8: Block diagram of the MIMO receiver](image)

A function USRP_setup will be called when the user runs the receiver script. Similar to the transmitter, this function identifies and enables a RFX2400 device. Since the MIMO SDR STS system used a single receive antenna, automatic enabling is employed as there is no need to attain control over which antenna to receive the data. At the USRP source, a complex vector sink was implemented to collect all complex samples passing through the USRP source to an array called fs_usrp.data. These samples are used to estimate the channel coefficients in the Rx_callback function. When the USRP receives the superpositioned payload, the data will proceed to the Time Deinterleaver block. After being de-interleaved, the data is transferred to the channel filter and an FIR filter where inter symbol interference (ISI) is eliminated. The received payload at this point in the receiver is still in the form of complex samples. At the packet receiver, the function demod_pkt s called; this function reverses any signal processing that was performed by mod_pkt in the transmitter by converting the complex samples back to its binary representation. If zero-padding was used in the transmitter, demod_pkt will remove all the zeros that were appended to the payload at the transmitter. Lastly, the demodulator decodes the modulation coding and calls up a function Rx_callback, where the desired receive algorithm scheme is implemented.

4. The Use Of Low Correlation Spreading Codes In STS Test Bed

It is known that STS can suffer from MAI from simulation studies [6]. In [7] we showed through simulation studies that low correlation spreading sequences showed improved BER performance compared to using Walsh-Hadamard spreading codes. We setup the MIMO STS SDR system so that it emulated the effects of MAI (unsynchronised and different energy level interfering sources). The possible scenarios for such a system have been described in [6]. Here we outline the experimental setup used to emulate MAI and then present the results found when we used the low correlation codes (proposed in [16]) and the Walsh Hadamard spreading sequences.
The technique used to emulate a MAI environment in the developed MIMO SDR STS system was developed assuming that the targeted user of the receiver is user A while another multiple access user, user B, is transmitting within the radiation range of the receiving antenna as shown in Figure 9.

Figure 9: MAI experiment diagram

In the test bed, User B will not be another physically distinct STS transmitter. Instead, User B and User A data will be superimposed in the transmit algorithm before the actual transmission. This arrangement will simulate MAI from user B on user A. In order to better simulate an actual MAI environment, the MAI transmit algorithm was coded to apply an element of randomness to the interfering pair of spreading sequences. Before every transmission, the transmit algorithm creates two random integer numbers, H1 and H2. H1 is a random number in the range 1 to 3, where each number represents a specific pair of spreading sequence for User B. H2 is another random number in the range 1 to 16; this number will determine the amount of chip delay present in the interfering spreading sequence pair.

Five hundred and twenty eight bits of data were transmitted at an amplitude level of 500 across five relative distances of 2, 1, 0, -1 and -2. This was done for the Wysocki low correlation spreading sequences and the Walsh Hadamard. The results are plotted in Figure 10.

Looking at the values for BER performance provided by Figure 10 we note that at the lower range of relative distances, where the interferer (User B) is further away from receiver than the target user (User A), the Walsh Hadamard system outperformed the Wysocki codes system. However, as the interferer (User B) moves closer to the receiver (assuming User A remains stationary), the Wysocki code system demonstrates a lower BER. This superior performance at higher signal levels of the Wysocki code can be attributed to the superior low cross correlation properties of the Wysocki code [16] and confirms similar results found in the simulation studies provided in [7].

5. Conclusion

A MIMO SDR STS system was constructed using the GNU Radio platform. A set of MAI experiments were provided to confirm in a real physical test bed the effects of MAI on STS. It was shown that mitigation against MAI is possible, as shown previously in simulation studies [7], by using Wysocki spreading sequences over Walsh hadamard sequences in a STS based system. Future work may include investigating the use of two transmit and two receive antennas on the described system.

6. References