Space Frequency Block Coding Design for the Neptune Digital Communications Project

Michelle Thompson W5NYV
w5nyv@arrl.net
3/21/24

Foreword

Space Frequency Block Coding (SFBC) is a modern digital communications technique used in mobile cellular systems to improve the resiliency of transmitted signals.

When we talk about signal diversity, we are discussing things that increase the resiliency of our signal. For example, simply repeating the data we send some number of times is a simple and effective way to increase the probability that a signal will be received. We use repetition when we repeat a call sign or exchange in an amateur radio contest. If one out of three repetitions of the call sign get through noisy conditions, or if the suffix of a call sign gets through on one repetition and the prefix in another, then we are further along the way towards a successful contact. When we send and receive digital signals, we have a wide variety of techniques to increase the useful redundancies in a signal. This article discusses a set of signal diversity techniques called Space Frequency Block Coding (SFBC).

SFBC provides spatial, frequency, and code diversity. In digital communications, the goal is to get as many bits of information as possible from the transmitters to the receiver, without errors. Transmitter diversity techniques, like SFBC, give significant advantage to our signals and are worth our time to learn about and implement in amateur radio digital communications.

Introduction

SFBC is a technique used in orthogonal frequency division multiplexing (OFDM) modulated transmissions. SFBC uses multiple antennas. For our case study, we have two transmit antennas and one receive antenna. The transmitted samples are transformed in a particular mathematical way before they are sent out over the air.

Having multiple antennas allows us to have diversity in space. Antennas that are physically apart from each other gives what we call a spatial advantage. This helps in a multi path environment.

Multi path signals are the result of transmissions bouncing off of the variety of surfaces that exist in the environment. Buildings, mountains, towers, and even airplanes can create reflections and refractions of the original signal. We now have multiple signals that are traveling paths that differ in length. Because some of the reflected or refracted signals are taking a longer path to get to the receive antenna, the signals are time-shifted copies of each other. Different echoes traveling at different time delays can end up, worst case, cancelling each other out. There could also be changes to the frequency and amplitude of the signal along some of these paths.

This means we may get a very damaged received signal. The amount of damage that we get from multi path depends on the physical environment and the frequency we're using. If you have ever heard the choppy sound of "picket fencing" on the 2 meter band, then you have heard a multi path signal. If you have ever corrected "picket fencing" by moving, you have solved destructive interference by moving your antenna to a different position. The design discussed in this article is intended to operate at 5 GHz, which has a much smaller wavelength than 2 meters. This means that a small distance between transmitting antennas can result in a big improvement against multi path at the receiver. If the receiver is at a place where one of the transmit signals is attenuated due to to physical blockage, or has destructive interference due to reflective multi path, then the receiver can use the second (or third, fourth, etc) path instead. By using transmit diversity, we are giving our receiver the best possible opportunity to recover the strongest possible signal. This is done before any error correction techniques are applied.

The mathematical transformations that we are using mitigate the damage that our signal might suffer. These mathematical techniques work for us even if the different transmit paths had a worst case relationship of being exactly 180 degrees out of phase due to multi path conditions. Without this extra coding technique, our signal would disappear.

The multiple subcarriers in OFDM give a diversity with respect to frequency. An OFDM signal is composed of many baseband subcarriers. The subcarriers are spaced out evenly across the bandwidth of our OFDM signal. The width of a subcarrier is called the subcarrier spacing. An OFDM baseband signal is converted from a frequency representation of a collection of subcarriers to a complex time series by using an Inverse Fast Fourier Transform (IFFT). This series of complex samples in time is what we send out over the air and is what we are dealing with in this article. After the samples are received, they are converted back to our baseband representation by using a Fast Fourier Transform (FFT).

OFDM signals may have hundreds or thousands of subcarriers. If one subcarrier has frequency-dependent interference, then another subcarrier just might be in better shape. SFBC is done in addition to error correction and other digital signal processing techniques. The encoding that we are doing in SFBC does not correct for errors and does not increase the maximum throughput. It increases the resiliency of our signal, making it more possible to achieve maximum throughput.

Additional spatial and frequency redundancy do come at a cost. Having two or more antennas instead of one, the extra calculations to produce the required transmit symbols, and the extra work required at the receiver, all result in a larger and more complicated system than a single transmitter and single receiver. It's up to us as designers to figure out when and if the extra complexity is worth the additional cost. Using the mathematical transformations in SFBC means we do not have to know the channel conditions at the transmitter to get better performance at the receiver, but it also reduces our SNR by half, or 3dB. Trade offs like this are found in any communications system. When an advantage is worth the cost, it makes sense to go ahead and do it. An important part of the implementation part of this project is to show the increase in resiliency over the air and publish the results.

On the microwave bands, with wide bandwidth signals and multi path interference, it is frequently the case that the gains with techniques like SFBC are well worth the additional circuitry. This article describes the design and implementation of SFBC for a particular amateur radio application called Neptune that comes from Open Research Institute.

Neptune is an OFDM based microwave band communications project intended for drone and aerospace deployments.

ORI is a non-profit research institute devoted to open source digital radio work. ORI primarily benefits amateur radio. Find out more about ORI at https://openresearch.institute

#

Design

The SFBC circuits on Neptune are located at the end of the transmitter chain. We have taken our baseband data signal and done the required channel coding, scrambling, and modulation. We now have a stream of transmit samples. Our understanding of what the samples stand for are organized and documented by using an OFDM Resource Grid. The Resource Grid is a convenient and useful representation for OFDM. The columns of the Resource Grid form OFDM symbols. One column is one entire symbol, which covers the entire bandwidth of our signal. Each symbol is composed of some number of transmit samples. Each individual square in the resource grid is a Resource Element. Each Resource Element (each square) holds one transmit sample.

Resource Elements contain either User Data, Reference Signals, or Control Data. The pattern of the different types of data is important. There are many different ways to group our data in the Resource Grid along the time and frequency axes. Different ways of grouping our data can change the way the higher layers of our communications system work. It’s common practice to represent different types of data with different colors, as can be seen in the example Resource Grid below.

OFDM Subcarriers are the rows of the Resource Grid. Subcarriers are numbered in the vertical direction from bottom to top. The Subcarriers represent different frequencies at baseband, but because we have converted our frequency domain baseband signal representation to a time domain series of samples by using an IFFT, the samples contained in the Resource Elements are transmitted one after another in time. This can be confusing if you are coming from modulation schemes where you have a time domain series of data, and you convert it to the frequency domain with an FFT, and then send that signal out over the air. To make it even more confusing, some communities refer to the rows in the Resource Grid as Tones. Tones and Subcarriers are names for things in the frequency domain. Yet we are in the time domain. Hang in there, because things do get less confusing.

Symbol slots (time) are numbered in the horizontal direction. When we send a stream of samples, we transmit one complex sample per subcarrier entry, starting at the bottom of a column and going to the top of the column. When we have transmitted all the samples in one column in order, we have sent one OFDM symbol. We then immediately start over at the bottom of the next column. The number of items in the column of the Resource Grid equals the number of transmit samples per symbol that we are sending out over the air. For example, the version of Neptune discussed in this article has 1106 samples per transmitted OFDM symbol.

The goal remains the same no matter how we organize our data. We want to get as much data across the radio link as possible, using the link control information and reference signals to help us adapt to changing radio conditions.

A Resource Grid example can be found in the open source FlexLink specification at https://github.com/OpenResearchInstitute/Neptune/tree/main/FlexLink/docsAll

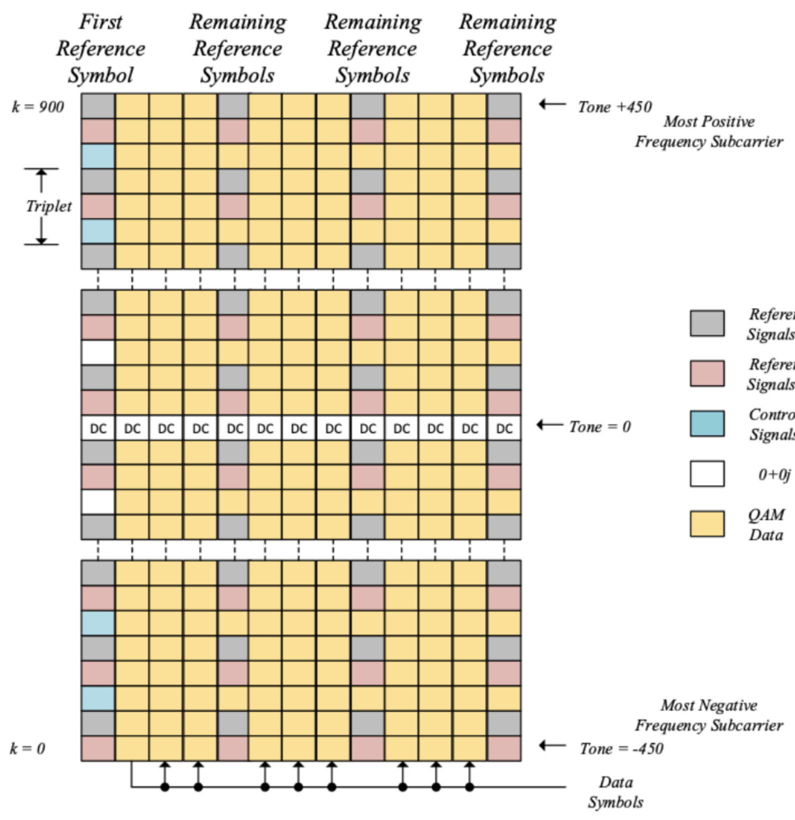


Figure 23: Reference Grid for the 20MHz Bandwidth Case and Subcarrier Spacing of 20KHz

Resource Grids are very useful and are the traditional way of representing the different types of content in an OFDM signal. However, in order to understand SFBC, all we need to know is that we have a time series of complex numbers that we must transmit. Let's assume that we're doing a good job of keeping track of where we are in our OFDM symbols by using our Resource Grid.

The series of complex numbers that we are transmitting represent the in-phase and quadrature components (I and Q) of quadrature modulation. I is the real component and Q is the imaginary component of a complex number. The values of I and Q are the amplitudes of two sinusoids that have the same frequency and are 90° out of phase. By convention, the I signal is a cosine waveform with amplitude I, and the Q signal is a sine waveform with amplitude Q. I and Q are all we need to reproduce any signal over the air.

For a refresher on quadrature modulation, please see W2AEW's excellent video at https://www.youtube.com/watch?v=h_7d-m1ehoY

Neptune is a digital communications system that is targeting a field programmable gate array (FPGA), so we're working in fixed point representation. The real component is 16 bits long. The imaginary component is 16 bits long. An example of the representation of a transmit sample would be:

Real + Imaginary √(-1)

0001101010101101 + 1110101000000001i

If we think of our transmit samples as a single stream of complex numbers going to an antenna, then we can start talking about how SFBC transforms this stream of samples.

Our transmit data stream is expressed as a series of complex numbers called x[n], where x1 is the first sample (from the bottom of a column), x2 is the second sample, and so on until we get to x1106, which is the index of the final sample in our OFDM symbol (at the top of a column). We then move on to the next symbol (the next column in the Resource Grid) of OFDM data, starting with x[1].

We have two antennas. To prepare our time series of transmit samples, we take them two at a time. We create a mathematical transformations of each of these two samples. We now have a total of four transmit samples. With two antennas, we can send them out two by two. This takes the same amount of time as sending the original two samples one after another.

When we assign the samples to be transmitted, we will use the labels s1 and s2. The "s" is to remind us we are sending samples and transformations of samples.

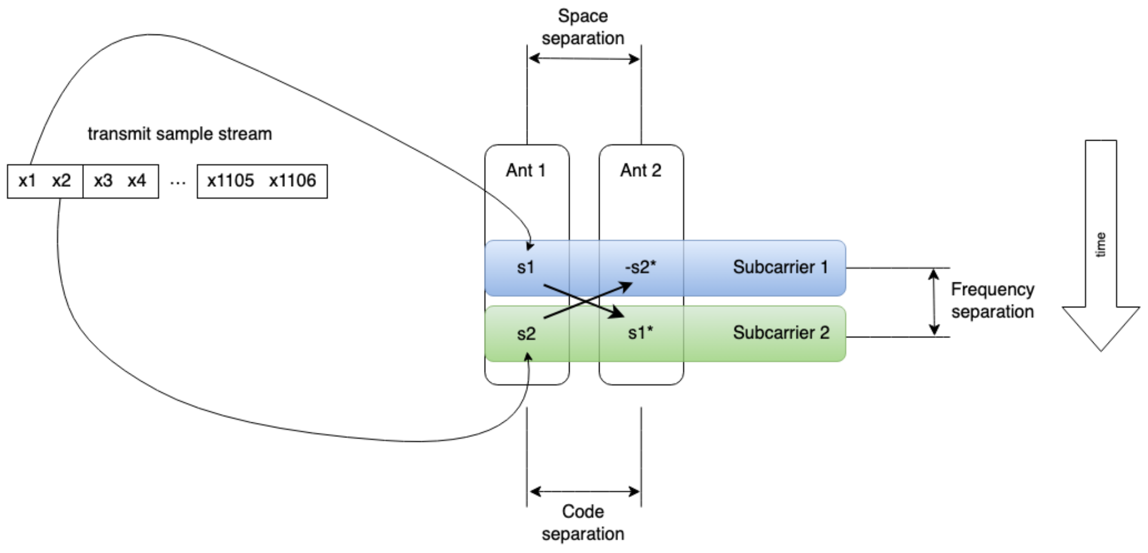
Using two antennas provides spatial diversity. The mathematical transformations of the samples we are transmitting provide coding diversity. The way we assign the samples and their mathematical transformations to subcarriers provides additional transmit diversity because of the frequency separation between subcarriers.

We prepare our data by picking up two samples at a time, starting with s1 and s2. We create the complex conjugate of s1 and the negative complex conjugate of s2. s1 is transmitted on Subcarrier 1 on Antenna 1, and the negative complex conjugate of s2 is transmitted on Subcarrier 1 on Antenna 1.

We transmit s2 on Subcarrier 2 on Antenna 1, and the complex conjugate value of s1 on Subcarrier 2 on Antenna 2. An asterisk represents the complex conjugate of a sample.

Below are diagrams describing SFBC. They show how we set up our stream of transmit data and what it looks like at the receiver. Our diagrams are inspired by the work at https://www.sharetechnote.com/html/Handbook_LTE_SFBC.html

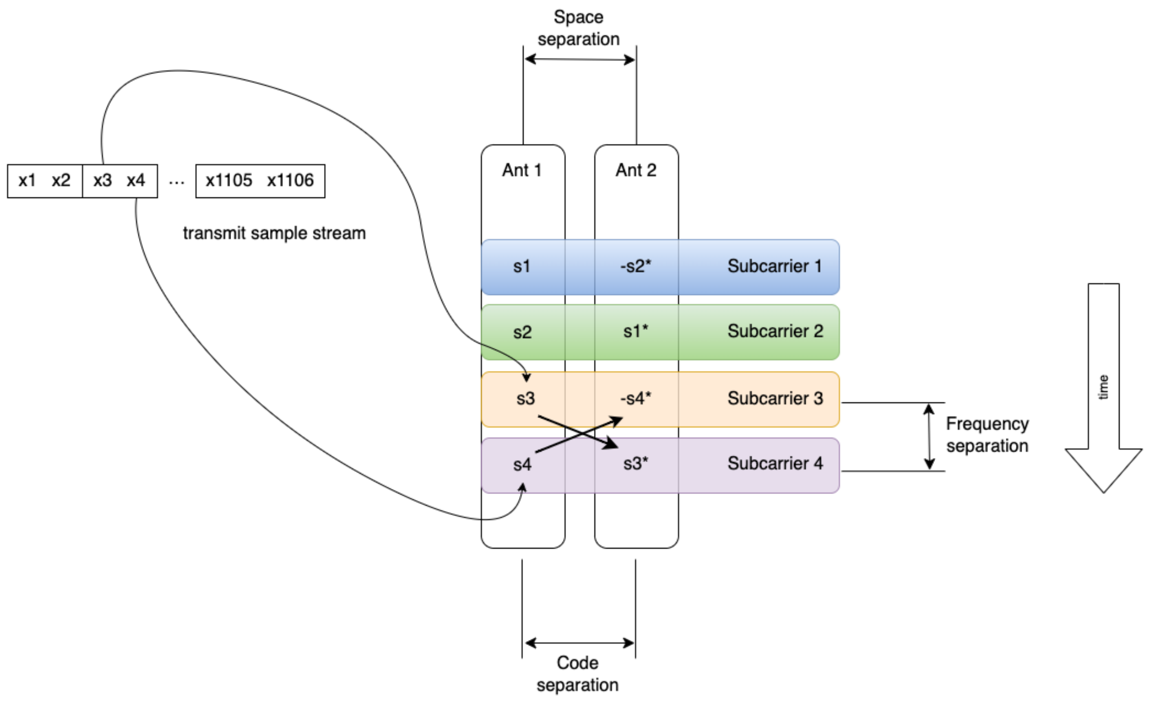
#



#

The diagram below shows how the next two samples are prepared and sent over the air.

#



#

Complex conjugation is when we change the sign of the imaginary component of a complex number. For example, the complex conjugate of 2 + 5i is 2 - 5i. The negative complex conjugate of 4 + 8i would be -(4 - 8i), or -4 + 8i.

If we take the complex conjugate of a complex conjugate, we get the original number back.

For example,

s54 = 12 - 73i

s54* = 12 + 73i

s54** = 12 - 73i

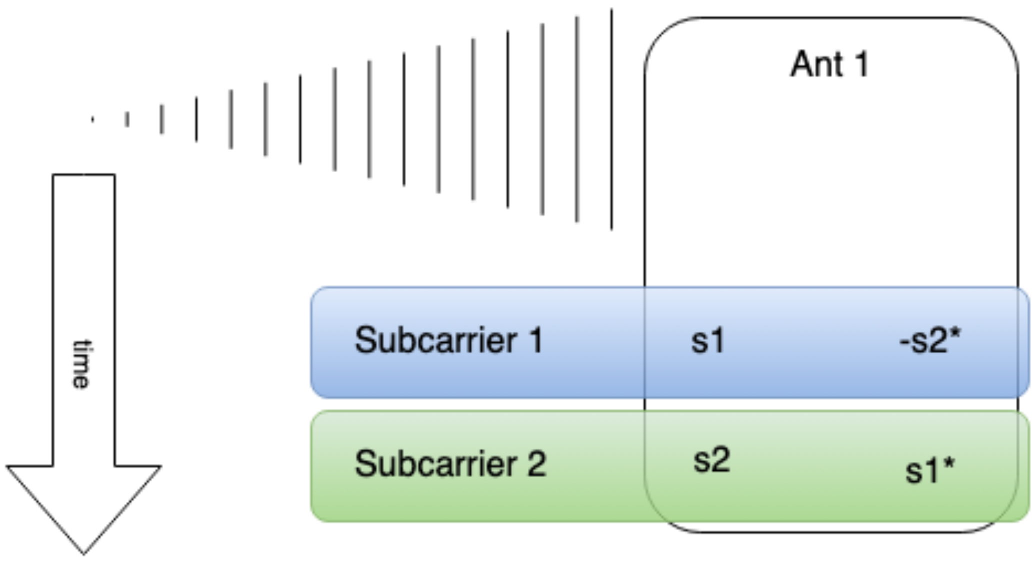
If we think of our series of transmit samples as train cars going down a track, we have come to a Y intersection. We send our original train down one track (Antenna 1), and a magical re-ordered ghost train down the other track (Antenna 2). Because we are using two tracks for twice the number of train cars, doubling the number of train cars doesn't make the train longer or go slower.

Think of this magical second ghost train as full of clipboard-carrying bean-counters that keep track of what is on our train cars. The bean counters watch our train as it goes through scary tunnels and up and down steep grades, and this makes it more likely all the cars will get through.

At the station at the end of the line, or at our receiver, we combine the two tracks back into one. We use our clipboard-carrying bean-counters to mathematically verify the quality of the original train cars. We aren't increasing the amount of cargo going down the track or making it go faster. We are making it more likely that the cargo we started out with will arrive at the station.

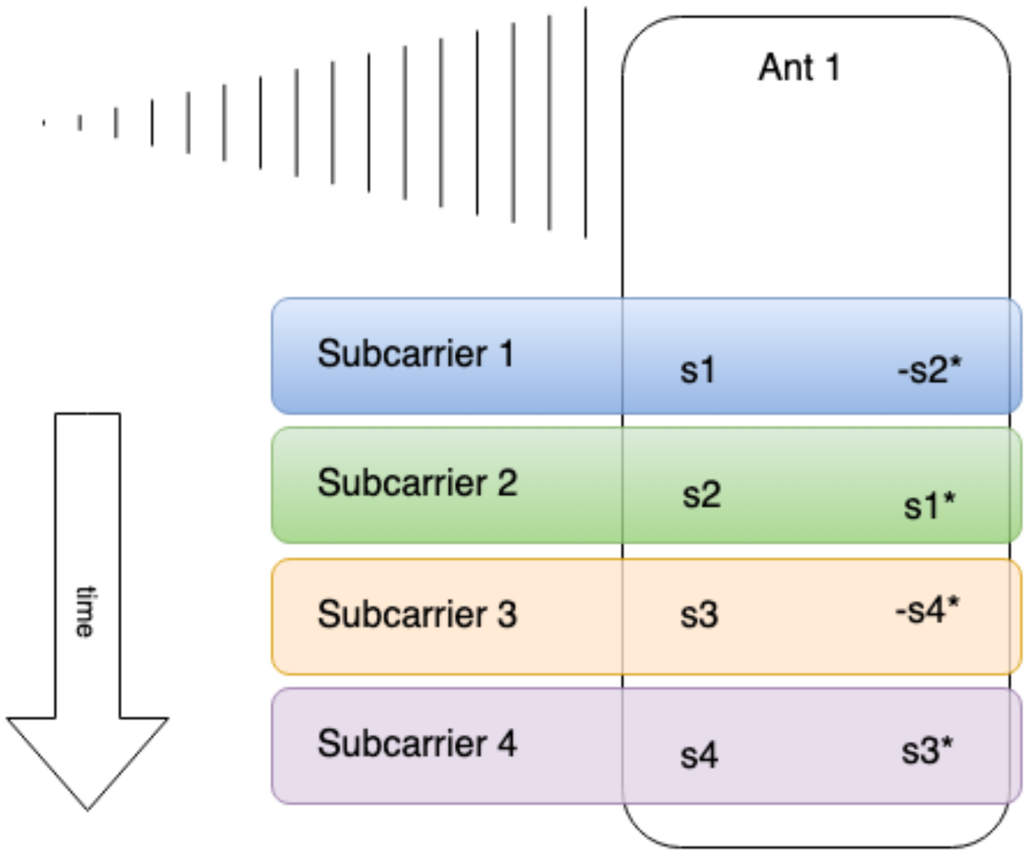
Here is what happens at the receiver. We receive s1 and -s2* combined at our single receive antenna in the first time slot. Both signals were modified by different path conditions. Plus, we picked up noise along the way. We receive s2 and s1* combined, modified by different path conditions, plus noise at the receive antenna in the second time slot.

#



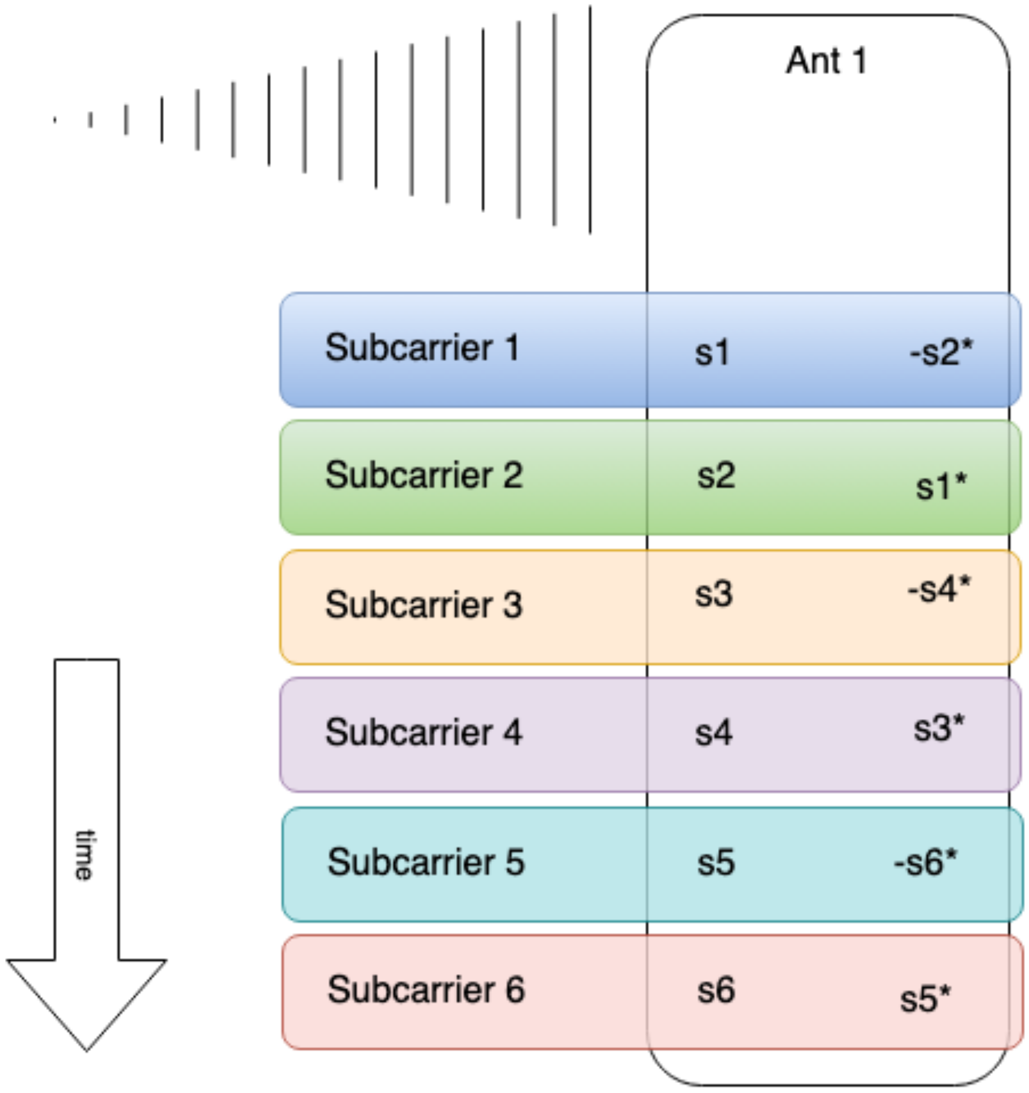
#

And, so on. As time passes, we receive the rest of our encoded transmit sample combinations. Below are the third and fourth time slots.



Here is what we’ve received by the time the fifth and sixth time slots arrive.

#



How do we untangle our original samples (for example, s1 and s2) from these combinations at the receiver? First let's consider a simplified situation where we have multi path, but no other effects. We're not doing any corrections or equalization. In the first time slot we transmit s1 from Antenna 1. At the same time we transmit -s2* from Antenna 2. In the second time slot we transmit s2 from Antenna 1. At the same time we transmit s1* from Antenna 2. The transmitted samples arrive at our one receive antenna.

#

Time Step 1: s1 - s2*

Time Step 2: s2 + s1*

#

Let’s simplify the situation even more to show how using two transmit antennas and one receive antenna helps us. We’re sending two samples, A and B. We send A from Antenna 1 and B from antenna 2. The next time step, we send B from Antenna 1 and A from Antenna 2.

	Antenna 1	Antenna 2	Receive Antenna
Time Step 1	A	B	A+B
Time Step 2	B	A	B+A

#

We use multiple observations to increase the chance we receive data even with destructive interference. Just one of the two possible channels needs to be good to recover the data. What happens if we can’t hear anything from Antenna 1?

	Antenna 1	Antenna 2	Receive Antenna
Time Step 1	A	B	0+B
Time Step 2	B	A	0+A

#

B and then A gets through from Antenna 2. We are on the right track with this redundancy, however, if the paths between Antenna 1 and Antenna two are perfectly out of phase, nothing is heard at the receiver at all. We indicate that the paths are perfectly out of phase by multiplying the signal from Antenna 2 by negative 1.

	Antenna 1	-Antenna 2	Receive Antenna
Time Step 1	A	-B	A-B
Time Step 2	B	-A	B-A

#

This result at the receiver could have been because A=B=0 or A=B=1. That's quite a lot of ambiguity. The root cause is because A-B and B-A are not linearly independent. They give the same information. You can get the second equation (B-A) by multiplying the first equation (A-B) by -1.

How can we fix this? We flip the sign of a sample sent on the second antenna in the second time step. Assume the worst case channel conditions exist between the two paths. The two samples are out of phase with each other in ways that result in no signal at the receiver during that time step. With the way we have changed the sign of a transmitted sample as above, we can recover from even the worst case.

	Antenna 1	-Antenna 2	Receive Antenna
Time Step 1	A	-B	A-B = 0
Time Step 2	B	-(-A)	B+A = 2

#

We now have independent observations captured in these equations. In the example above, we can now determine that A = B = 1.

	Antenna 1	-Antenna 2	Receive Antenna
Time Step 1	A	-B	A+B = 0
Time Step 2	B	-(-A)	B+A = 0

#

Let's look at another case. In the example above, the only correct result is A = B = 0. The ambiguity we had before is now gone, by sending one of our samples from the opposite antenna with the opposite sign. This is called the Alamouti Technique (S. M. Alamouti, "A simple transmit diversity technique for wireless communications," in IEEE Journal on Selected Areas in Communications, vol. 16, no. 8, pp. 1451-1458, Oct. 1998).

A friendly and accessible video walkthrough of these concepts can be found at https://www.youtube.com/watch?v=cbD4NsZQKYw&ab_channel=ArtoftheProblem

But what about multi path and noise? In order to recover our transmitted samples from real channels with real multi path effects plus noise there is some additional information that we need to know. We have looked at the worst case signal conditions, where the two paths from the two transmit antennas are completely out of phase, but what about everything between the worst case and the best case? We need to know more detail about the conditions on the paths from our two transmit antennas to our one receive antenna. Remember the Reference Signals in the Resource Grid? We get the information we need to equalize our signals from the reference or pilot signals that are embedded in the OFDM transmit signal. We use these reference signals like beacons. If we can barely hear our beacon and it’s got a frequency shift, then we can calculate the corrections we must make for the subcarriers covered by that beacon. If we can clearly hear our beacon and it arrived as expected, then conditions are quite clear and fewer corrections need to be made.

For this article, we will assume that this part is working well, and we have information about our channel. The values we need are h1, the channel conditions between Antenna 1 and the receive antenna, and h2, the channel conditions between Antenna 2 and the receive antenna. When we send a sample through a channel, and the channel changes the signal, we represent this by multiplying the channel value (h1 or h2) times the signal. The path between the transmitter and receiver may distort our signal quite a lot or not much at all. The channel condition values are complex numbers.

We are going to determine what s1 and s2 were, based on our received signal y and our knowledge of h1 and h2. The received signal from the first time slot is called y1. The received signal from the second time slot is called y2, and so on. We assume that our receiver has successfully synchronized to the transmitted signal and we are tracking any changes. This is a very important job for the receiver to do and there is a lot of engineering that goes into it. We are going to proceed as if synchronization is working correctly.

The received signal is the sum of the modified transmit sample s1 and the modified transmit sample -s2*. We know that s1 was sent from Antenna 1, therefore s1 is transformed by the channel value h1. We know that -s2* was sent from Antenna 2, therefore -s2* is transformed by the channel value h2. Therefore, we know that y1 = (h1 x s1) + (h2 x -s2*).

From this point, we are going to use vectors to represent the sets of samples and channel values. A row vector is written as [a b], and a column vector is written as [a b]’. The ‘ mark stands for vector transpose. A brief explanation of vector arithmetic can be found at https://www.khanacademy.org/math/precalculus/x9e81a4f98389efdf:matrices/x9e81a4f98389efdf:mat-intro/a/intro-to-matrices

#

We start with our first received sample.

y1 = (h1 x s1) + (h2 x -s2*)

#

Convert to vector notation.

y1 = [h1 h2] x [s1   -s2*]’

#

We continue by taking our second received sample and re-writing it.

y2 = (h1 x s2) + (h2 x s1*)

#

Change the order of the addends. We want to end up with an [s1 -s2*] term on the right when we convert to vector notation. This will let us line things up with y1.

y2 = (h2 x s1*) + (h1 x s2)

#

Take the complex conjugate of both sides. We want to have s1 and -s2*, not s1* and s2. We are using the property that (a x b)* = a* x b* and a** = a.

y2* = (h2* x s1) + (h1* x s2*)

#

Multiplying by two negative numbers is a positive number. Use this fact to set up -s2* in the second vector. This makes h1 negative in the first vector.

y2* = (h2* x s1) + (-h1* x -s2*)

#

Rewrite in vector notation.

y2* = [h2* -h1] x [s1   -s2*]’

#

There are two things that we’ve done here. The first vector of y1 and the first vector of y2 have an important characteristic that we are going to use. And, the second vector for both y1 and y2 is now the same, namely [s1 -s2*]. In order to take advantage of both of these facts, we combine y1 and y2.

$$\begin{bmatrix} y1 \\ y2* \end{bmatrix} = \begin{bmatrix} h1 & h2 \\ h2* & -h1* \end{bmatrix} \begin{bmatrix} s1 \\ -s2* \end{bmatrix}$$

We are going to re-arrange the equation in order to reveal that first important characteristic mentioned above, which has to do with h1, h2*, h2, and -h1*. We now have a set of equations, y1 and y2. The column vectors act upon the samples received in the first time slot. We take the columns of the h matrix and name them.

$$c1 = \begin{bmatrix} h1 \\ h2* \end{bmatrix} \quad c2 = \begin{bmatrix} h2 \\ -h1* \end{bmatrix}$$

y = [h1   h2*]’ x s1 + [h2   -h1*]’ x -s2* y = c1 x s1 + c2 x -s2*

#

c1 and c2 are related in a special way. If we take the transpose conjugate of c1 and multiply it by c2, then we get zero. The conjugate transpose of a vector is indicated by a superscript H.

#

$c1^H$ x c2

[h1*   h2] x [h2 -h1*]’

(h1* x h2) - (h2 x h1*)

(h1* x h2) - (h1* x h2)

0

#

This is the definition of orthogonal vectors. c1 and c2 are orthogonal. In these two time steps we’ve received both s1 and s2 along with s1* and -s2*. We can separate s1 and s2 using the orthogonality of c1 and c2. Here’s how that’s done.

#

Here’s what we received in the first two time slots.

y = c1 x s1 + c2 x -s2* + noise

#

First we isolate s1.

Multiply both sides of this equation by conjugate transform of c1 divided by the norm of c1. (This could be explained more. This could be the definition of a projection)

We know that that conjugate transform of a vector multiplied by that vector is the norm squared. We know that c1 and c2 are orthogonal. After we isolate s1 from our received signal, we use the conjugate transform of c2 over the norm of c2 to isolate -s2*. Since we know the values of c1 and c2, the norm of the vectors is a scaling term for s1 and -s2*.

#

$$c1^H\, c1 \;=\; \|c1\|^2$$
$$c1^H c2 \;=\; 0$$

$$\left(c1^H / \|c1\|\right)(y) \;=\; \left(c1^H / \|c1\|\right)(\,c1\, s1 \;-\; c2\, s2* \;+\; noise\,)$$
$$\left(c1^H / \|c1\|\right)(y) \;=\; \left(c1^H c1 / \|c1\|\right)(\,s1) \;+\; \left(c1^H c2 / \|c1\|\right)(\,-s2*) \;+\; \left(c1^H / \|c1\|\right)(noise)$$
$$\left(c1^H / \|c1\|\right)(y) \;=\; \left(\|c1\|^2 \,/\, \|c1\|\right)(\,s1) \;+\; (0 / \|c1\|)(\,-s2*) \;+\; \left(c1^H / \|c1\|\right)(noise)$$
$$\left(c1^H / \|c1\|\right)(y) \;=\; \|c1\|\, s1 \;+\; \left(c1^H / \|c1\|\right)(noise)$$

$$s1\; component\; of\; y \;=\; \|c1\|\, s1$$

$$\left(c2^H / \|c2\|\right)(y) \;=\; \|c2\|\,(-s2*) \;+\; \left(c2^H / \|c2\|\right)(noise)$$
$$s2\; component\; of\; y \;=\; \|c2\|\,(\,-s2*)$$

#

We now have properly scaled (equalized) versions of s1 and -s2*. We take the negative conjugate of -s2* to recover s2.

See the Appendix for a walk through of why we multiply through by the transpose conjugate divided by the norm, first using c1 and then using c2. We are using transmit beamformers to isolate the parts of y that were sent from Antenna 1 and Antenna 2.

We dropped the noise term to focus on the recovered sample, but let’s look at what happens to the noise in our receiver, and see what signal to noise ratio we might expect.

$$\textit{Consider the noise term of } \left(c1^H / \|c1\| \quad\right)(noise)$$

$$\left(c1^H / \|c1\|\right)(y) = \|c1\|\, s1 + \textit{gaussian noise with variance equal to } \sigma^2$$

$$\textit{From the definition of signal to noise ratio, we get}$$

$$SNR = \|c1\|^2\, E\left\{|s1|^2\right\} / \sigma^2$$

$$E\left\{|s1|^2\right\} \textit{ is the expected value of a sample, such as } s1.$$

$$E\left\{|s1|^2\right\} = E\left\{|s2|^2\right\} = (0.5\, Ptx)$$

$$\textit{We transmitted } \begin{bmatrix} s1 \\ s2 \end{bmatrix} \textit{ in the first time step.}$$

$$\textit{Therefore each transmitted sample gets half the transmit power } (Ptx).$$

$$\textit{From the definition of norm, we get}$$

$$\|c1\| = \sqrt{\left(|h1|^2 + |h2|^2\right)} = \|\bar{h}\|$$

#

$$\textit{Substitute and rearrange the equation.}$$

$$SNR = \|\bar{h}\|^2\, (0.5)\, Ptx / \sigma^2 = (0.5)\, \|\bar{h}\|^2\, Ptx / \sigma^2$$

$$\textit{The term } \|\bar{h}\|^2\, Ptx / \sigma^2 \textit{ is the SNR of max ratio combining.}$$

$$\textit{We see that we have half the SNR of max ratio combining.}$$

#

We lose 3dB of SNR compared to a spatial diversity technique called maximum ratio combining, which is the optimal linear combining technique in digital signal processing. This is a good example of the sort of trade offs that we have to make in digital communications. Comparing our particular solution against optimal methods helps justify decisions and clarifies the pros and cons of a design. An advantage of the way we are doing things is that we do not need to know channel information at the transmitter, which is a difficult to implement requirement in maximum ratio combining. But, we lose 3dB of SNR.

Implementation

MATLAB and Simulink are software simulation packages from Mathworks Incorporated. The SFBC encoding operation is implemented as a custom function in MATLAB. The function was tested as a stand-alone module.

#

```
function [tx1, tx2] = neptune_sfbc(in)
%Neptune space frequency block code
%   for two transmit antennas, space frequency block coding is
implemented.
%   "in" is arriving as a 2 by 1 column vector.
persistent hTDEnc;
if isempty(hTDEnc)
    %use same object for both STBC and SFBC - clever coding from
    % "Understanding LTE with MATLAB" by Houman Zarinnkoub.
    hTDEnc = comm.OSTBCEncoder('NumTransmitAntennas', 2);
end
new_imaginary = -1*imag(in(2));
new_real = -1*real(in(2));
neg_conj_in2 = complex(new_real, new_imaginary);

new_imaginary = -1*imag(in(1));
new_real = real(in(1));
conj_in1 = complex(new_real, new_imaginary);

tx1 = [in(1); in(2)];
tx2 = [neg_conj_in2; conj_in1];
```

```
    end
                                        #
```

The custom function was then enclosed in a Simulink custom function block. The block was then inserted into the Neptune Simulink model flowgraph.

The input to the custom function Simulink block is a stream of transmit samples. The output of the block is a stream of samples to be sent to Antenna 1 and a stream of samples to be sent to Antenna 2. The stream of samples sent to Antenna 1 is unchanged from the input stream. The stream of samples sent to Antenna 2 are modified and re-ordered samples.

After the Neptune design is fully implemented as a Simulink model, it will be tested using environmental models from the mobile phone industry. Extensive channel models exist for 5 GHz, which is the band of interest for Neptune. Value added by SFBC will be measured and published.

Simulink models will be converted into hardware descriptive language code with a MATLAB toolbox called HDL Coder. The first step towards an FPGA or ASIC implementation of SFBC is simulation in MATLAB/Simulink. FPGA implementations of Neptune will be used on a wide variety of software-defined radio platforms. These implementations, in HDL source code form, will be provided for free to the general public under open source licensing. One does not need MATLAB or Simulink to use the HDL source code.

The repository for Neptune work can be found at https://github.com/ OpenResearchInstitute/Neptune

Future Work

Future work includes implementing SFBC decoding in Simulink and converting it to HDL source code. There is more complexity in the receiver compared to the transmitter, so this is quite a bit of work.

Testing the implementation in simulation against multi path channel models is followed by testing over the air. These channel models are available to us from academic literature and are in our version of MATLAB and Simulink.

Another area of future work is modifying the SFBC custom function so it can do an additional transmit diversity technique called Space Time Block Coding (STBC). This calculation is already in the function. SFBC is created from STBC by taking the negative conjugate of all even numbered input samples. If you do not do this transformation before splitting up the samples and reordering them, then you get STBC. If you do carry out this transformation, which we do, then you get SFBC. Being able to have a dynamically switched block that produces both STBC and SFBC would let us show the two different techniques with a live over the air signal. This could let us quickly demonstrate the performance differences that are described in the technical literature about these techniques. Additionally, the entire block could be switched in and out of use in order to show the difference between no transmit diversity techniques at all and either STBC or SFBC. In the case of no transmit diversity techniques at all, the input transmit samples would be sent to Antenna 1, and nothing would be sent to Antenna 2.

The repository for Neptune work can be found at https://github.com/OpenResearchInstitute/Neptune

#

How to Participate

Neptune exists entirely because of volunteers, students, and enthusiasts. Comment and critique of any of the designs is welcome and encouraged. Work is done not only to implement modern digital communications designs for amateur radio, but also to provide personal and professional development opportunities through working on high quality open source digital radio projects.

If you would like to learn more about this and other types of digital communications, then you are invited to join or follow any Open Research Institute project.

Please see https://openresearch.institute/getting-started to get involved. Participation is free. All work is published to the general public at no cost.

Appendix

We start with a received signal from a two transmitter and one receiver system.

$$y = \begin{bmatrix} h1 & h2 \end{bmatrix} \begin{bmatrix} s1 \\ s2 \end{bmatrix} = h1\,s1 + h2\,s2$$

$$\bar{\mathrm{S}} = \begin{bmatrix} s1 \\ s2 \end{bmatrix} = 1 / \|\bar{\mathrm{h}}\| \begin{bmatrix} h1* \\ h2* \end{bmatrix} s$$

where s is a single transmitted sample.

$$1 / ||\bar{\mathrm{h}}|| \begin{bmatrix} h1* \\ h2* \end{bmatrix}$$

is known as the transmit beamformer.
It's a unit vector pointing in the direction of the transmit path.

We rewrite y, substituting in $\bar{\mathrm{S}}$

$$y \; = \; \begin{bmatrix} h1 & h2 \end{bmatrix} \; 1 / ||\bar{\mathrm{h}}|| \begin{bmatrix} h1* \\ h2* \end{bmatrix} s$$

Rearrange y.

$$y \; = \; \begin{bmatrix} h1 & h2 \end{bmatrix} \; \begin{bmatrix} h1* \\ h2* \end{bmatrix} \; 1 / ||\bar{\mathrm{h}}|| \;\; s$$

We know that a vector times its conjugate transpose is the norm squared.

$$\begin{bmatrix} h1 & h2 \end{bmatrix} \; \begin{bmatrix} h1* \\ h2* \end{bmatrix} \; = \; ||\bar{\mathrm{h}}||^2$$

Therefore

$$y \; = \; \left(\, ||\bar{\mathrm{h}}||^2 / ||\bar{\mathrm{h}}|| \, \right) \, s$$

$$y \; = \; ||\bar{\mathrm{h}}|| \;\; s$$

so when we see something like this

$$\left(c2^{H} / \|c2\|\right)\ (y)$$

we are using a transmit beamformer that projects along c2
and (in our case) isolates $-\,s2*$

$$c2 \;=\; \begin{bmatrix} h2 \\ -h1* \end{bmatrix}$$

$$c2^{H} \;=\; \left[\, h2* \;\; -h1 \,\right]$$

$$y \;=\; c1\, s1 \;-\; c2\, s2*$$

$$\left(c2^{H} / \|c2\|\right)\ (y) \;=\; \left(\left[\, h2* \;\; -h1 \,\right] \;/\; \|c2\|\right) (c1\, s1 \;-\; c2\, s2*)$$

$$\left(c2^{H} / \|c2\|\right)\ (y) \;=\; \left(\left[\, h2* \;\; -h1 \,\right] \;/\; \|c2\|\right) \left(\begin{bmatrix} h1 \\ h2* \end{bmatrix} s1 \;-\; \begin{bmatrix} h2 \\ -h1* \end{bmatrix} s2*\right)$$

$$\left(c2^{H} / \|c2\|\right)\ (y) \;=\; (-1 / \|c2\|)\ \left[\, h2* \;\; -h1 \,\right] \begin{bmatrix} h2 \\ -h1* \end{bmatrix} s2* \;\;+\;\; (1 / \|c2\|) \left[\, h2* \;\; -h1 \,\right] \begin{bmatrix} h1 \\ h2* \end{bmatrix} s1$$

$$\overbrace{\left[\, h2* \;\; -h1 \,\right] \begin{bmatrix} h2 \\ -h1* \end{bmatrix}}^{c2^{H} c2 \,=\, \|c2\|^{2}} \qquad\qquad \overbrace{\left[\, h2* \;\; -h1 \,\right] \begin{bmatrix} h1 \\ h2* \end{bmatrix}}^{\textit{result is } 0}$$

$$\left(c2^{H} / \|c2\|\right)(y) = \left(\|c2\|^{2} / \|c2\|\right) - s2*$$

$$\left(c2^{H} / \|c2\|\right)(y) = \|c2\|\,(-s2*)$$

This is why we can multiply both sides of our received signal equation by the appropriate version of $\left(c^{H} / \|c\|\right)$.
Projecting along $c1$ *or* $c2$ *by using the transmit beamformer isolates the samples corresponding to* $c1$ *and* $c2$ *because* $c1$ *and* $c2$ *are orthogonal.*

#