ECE 463 Lab 9: OFDM

1. Introduction

In orthogonal frequency division multiplexing (OFDM), the symbols are considered to start in the frequency domain. OFDM operates on groups of symbols, the resulting group of symbols being called an OFDM symbol (or packet). In this lab, we will implement the key features of the OFDM such as cyclic prefix, null tones, and OFDM channel estimation/correction. However, this lab will not cover OFDM CFO correction and OFDM phase tracking due to time and resource constraints.

1.1. Contents

1. Introduction
2. OFDM blocks
3. Putting Together in Simulation
4. Putting Together in USRP

1.2. Report

Submit the answers, figures and the discussions on all the questions. The report is due as a hard copy at the beginning of the next lab.

2. OFDM Blocks

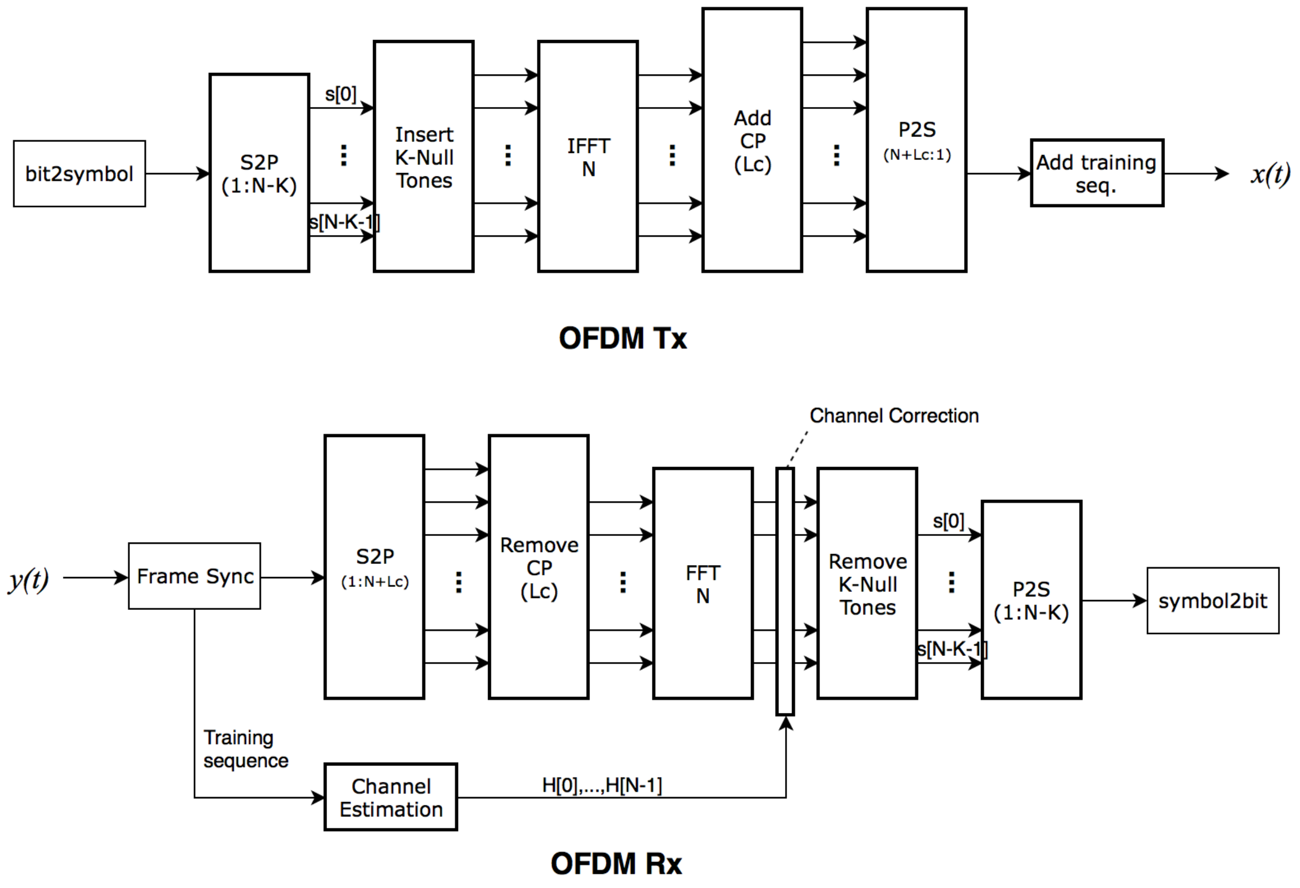


Figure 1. Overall OFDM block diagram.

The figure shows the overall block diagram of OFDM transmitter and receiver.
We will re-use *bit2symbol, frame sync, and symbol2bit* blocks. The new OFDM blocks to implement are the followings:

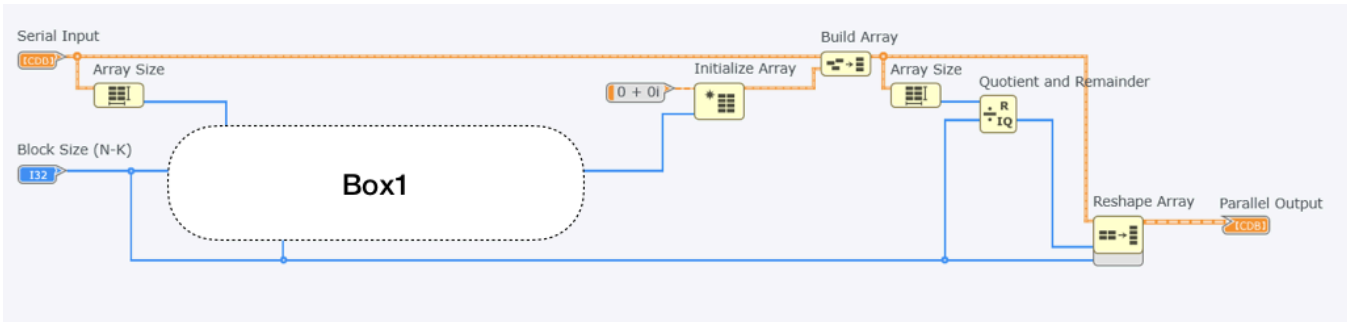
- **S2P**
- **P2S**
- **OFDM_insert_null_tones**
- **OFDM_remove_null_tones**
- **OFDM_add_cp**
- **OFDM_remove_cp**
- **OFDM_preambles**
- **OFDM_add_preambles**
- **OFDM_ChEst**
- **OFDM_ChCor**

2.1. S2P & P2S

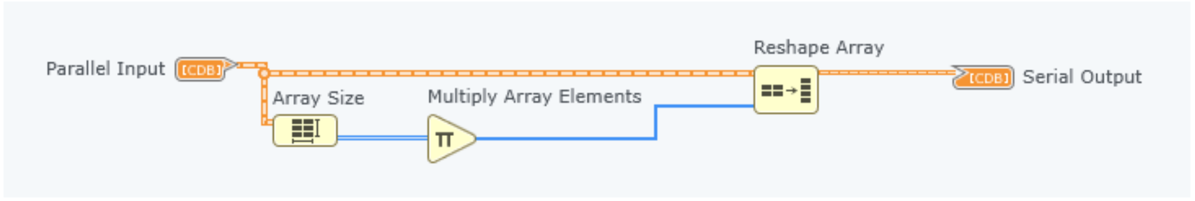
S2P block will convert the serial input stream (1D array) into the parallel output (2D array). The number of the columns of the output array equals to the block size (N-K). If the total length of the input stream is not the integer multiple of (N-K), we will pad zeros to match the size. For example,

$$\text{N-K=3, input stream=}\ [1,2,3,4] \longrightarrow \begin{bmatrix} 1 & 2 & 3 \\ 4 & 0 & 0 \end{bmatrix}$$

Complete Box1 in the following figure which calculates the number of zeros to pad.



P2S block simply serializes a 2D array into an 1D array. Implement the following figure for P2S block.



2.2. Insert Null Tones

We will implement a block that insert the null tones into the N-K subcarriers from S2P block. Note that the output of S2P block is a 2D array whose columns correspond to the subcarriers.

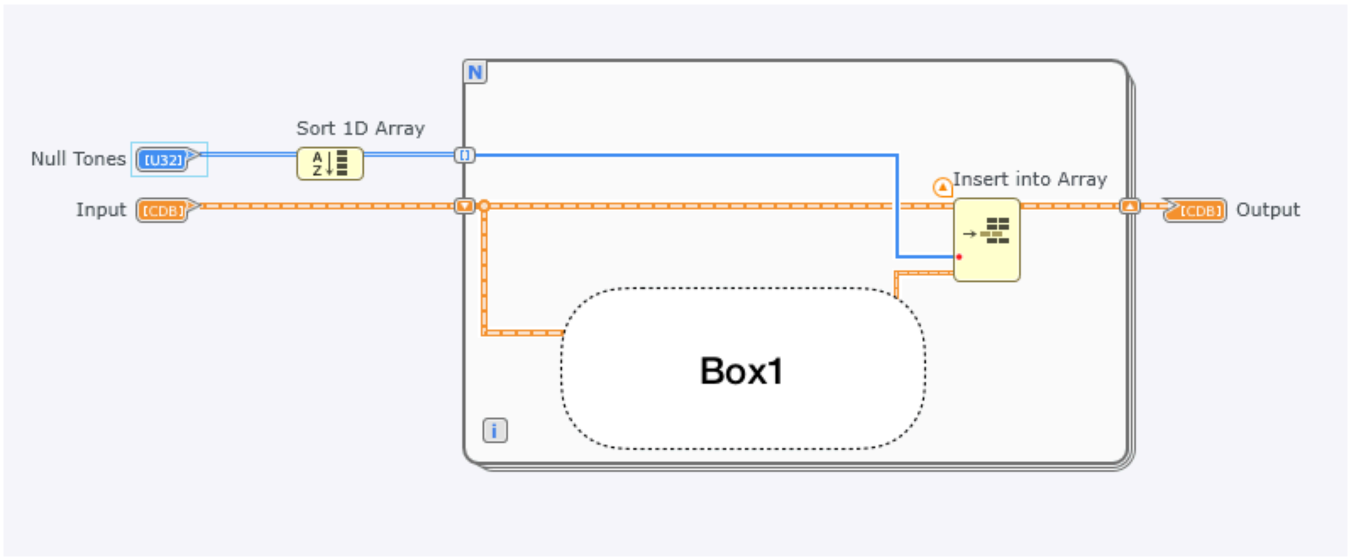
OFDM_insert_null_tones.gvi	Terminal name	Type	Description
Input	*Null Tones*	Unsigned integer array (1D)	K-location to insert the null tones
	Input	Complex double array (2D)	N-K columns
Output	*Output*	Complex double array (2D)	N columns

The following shows an example how this block works.

$$\text{Null tones}=[0,4,5,6]$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 & 4 & 5 & 6 \\ 0 & 7 & 8 & 9 & 0 & 0 & 0 & 10 & 11 & 12 \end{bmatrix}$$

Complete Box1 in the following figure. Box1 creates a column to be inserted at a given location. The number of entries of the column should be equal to the number or rows of the input 2D array. Use *Array Size* to get the number of rows. Use *Initialize Array* to set all the entries to zero.



2.3. Remove Null Tones

We will implement a block that removes the null tones. This block will undo *OFDM_insert_null_tones* block.

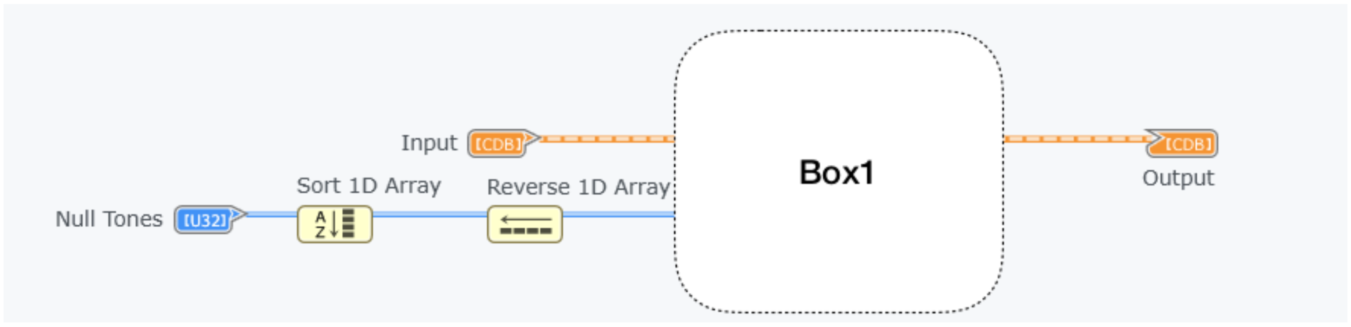
OFDM_remove_null_tones.gvi	Terminal name	Type	Description
Input	*Null Tones*	Unsigned integer array (1D)	K-location to remove the null tones
	Input	Complex double array (2D)	N columns
Output	*Output*	Complex double array (2D)	N-K columns

The following shows an example how this block works.

$$\text{Null tones}=[0,4,5,6]$$

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 & 4 & 5 & 6 \\ 0 & 7 & 8 & 9 & 0 & 0 & 0 & 10 & 11 & 12 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix}$$

Complete Box1 in the following figure. Notice that the null tones are sorted in descending order because the current removal should not affect the index for the next removal. Use a for loop, shift registers, and *Delete from Array block.*



2.4. Add Cyclic Prefix

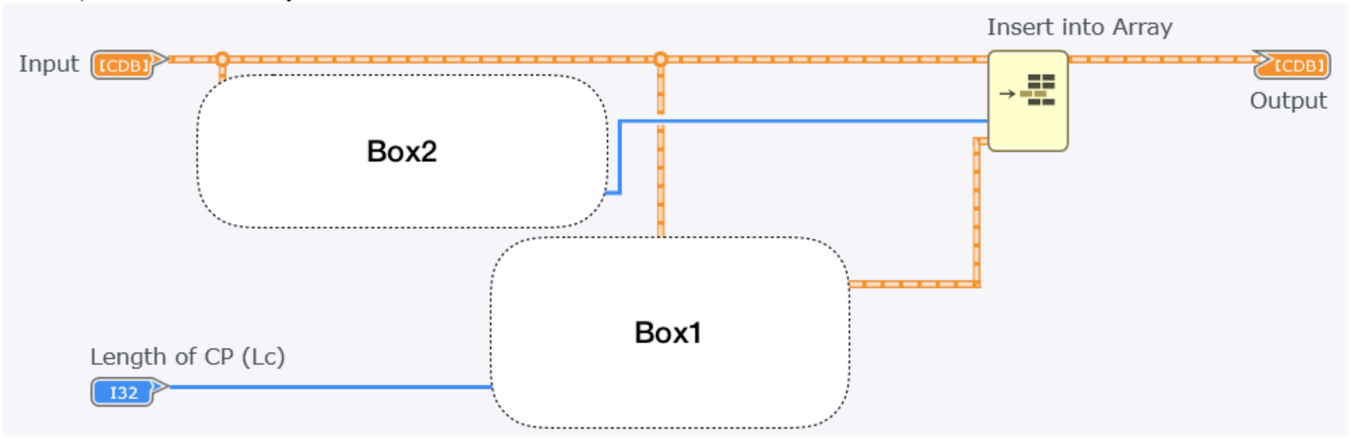
We will implement a block that adds the cyclic prefix. This block will take first few samples (Lc) and append them to the end of the samples.

OFDM_add_cp.gvi	Terminal name	Type	Description
Input	*Length of CP (Lc)*	Unsigned integer	
	Input	Complex double array (2D)	N columns
Output	*Output*	Complex double array (2D)	N+Lc columns

The following shows an example how this block works.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix} \xrightarrow{\text{Length of CP} = 2} \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 1 & 2 \\ 7 & 8 & 9 & 10 & 11 & 12 & 7 & 8 \end{bmatrix}$$

Complete Box1/Box2 in the following figure. Box1 returns the first Lc columns of the input array (Use *Array Subset*). *Insert into Array block* takes the column index from Box2.



2.5. Remove Cyclic Prefix

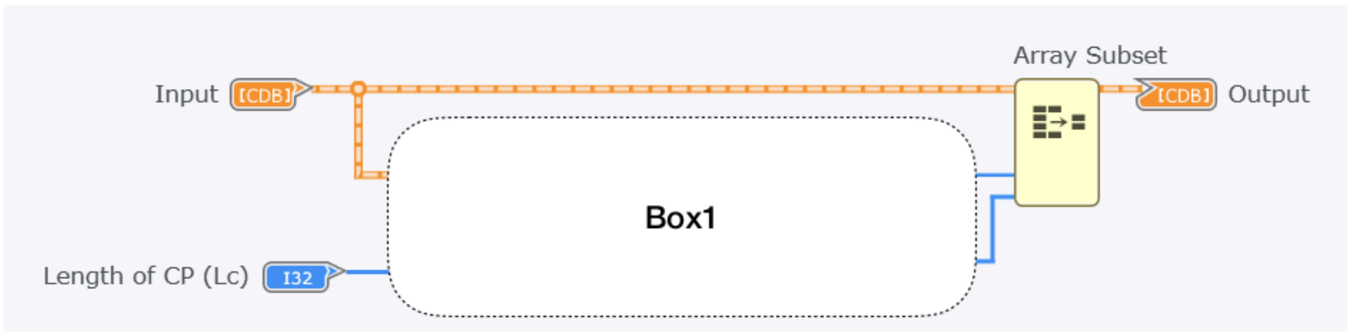
We will implement a block that removes the cyclic prefix. This block will remove the last Lc columns from the array.

OFDM_remove_cp.gvi	Terminal name	Type	Description
Input	*Length of CP (Lc)*	Unsigned integer	
	Input	Complex double array (2D)	N+Lc columns
Output	*Output*	Complex double array (2D)	N columns

The following shows an example how this block works.

$$\text{Length of CP} = 2$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 1 & 2 \\ 7 & 8 & 9 & 10 & 11 & 12 & 7 & 8 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{bmatrix}$$



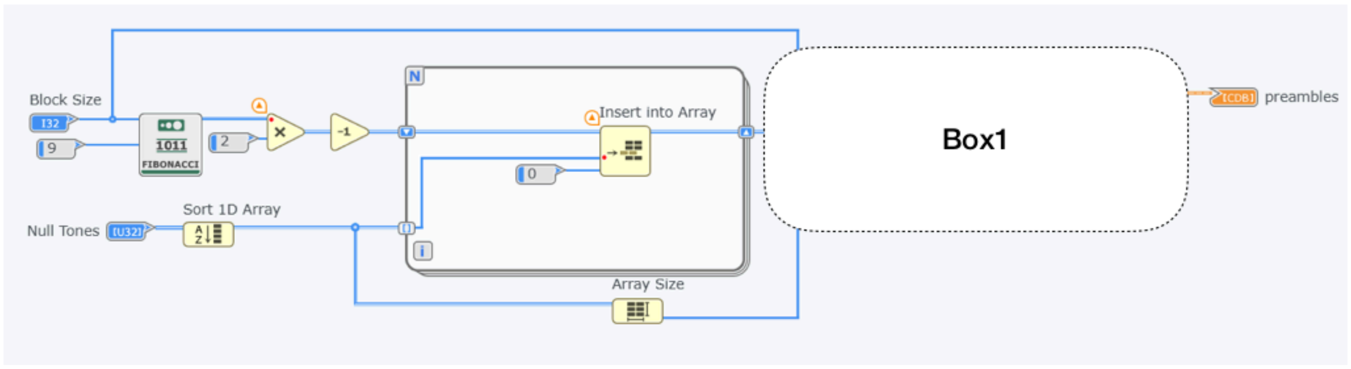
Complete Box1 in the following figure to remove the last Lc columns.

2.6. Generate OFDM Preamble

We will implement a block that generates two OFDM preambles. The block first creates (N-K) random BPSK symbols (in the frequency domain) and inserts K-null tones. After the insertion, it performs IFFT to transform the symbols in the time domain.

OFDM_preambles.gvi	Terminal name	Type	Description
Input	*Block size (N-K)*	Unsigned integer	N-K length
	Null Tones	Unsigned integer array (1D)	K-location to remove the null tones
Output	*Preambles*	Complex double array (1D)	N length

Complete Box1 in the following figure to create two OFDM preambles. Normalize the IFFT outputs by multiplying $\sqrt{\boldsymbol{N}}$**.**



2.7. Add OFDM Preambles

We will implement a block that adds two OFDM preambles at the beginning of the OFDM packet. Note that the packet is serialized by P2S block, and therefore it is 1D array. Before adding the preambles, add the cyclic prefix at the end of them. Use *OFDM_preambles* to generate the preambles.

OFDM_add_preambles.gvi	Terminal name	Type	Description
Input	*Packet in*	Complex double array (1D)	N-K length
	Block size (N-K)	Unsigned integer	N-K length
	Null Tones	Unsigned integer array (1D)	K-location to remove the null tones
	Length of CP (Lc)	Unsigned integer	
Output	*Packet out*	Complex double array (1D)	2N length

2.8. Frame Sync

In this lab, we will re-use our old cross-correlation frame sync block since we already have it. We encourage you to implement the sliding window method discussed in the lecture, if you are interested in.
You may need some modifications in your old frame sync block. First, make sure it takes CDB-type inputs. Second, normalize the correlation to make the detection independent with the input signal level. Finally, replace the old Barker sequence with the OFDM preambles.

2.9.OFDM Channel Estimation

We will implement a block that estimates the channel in the frequency domain.

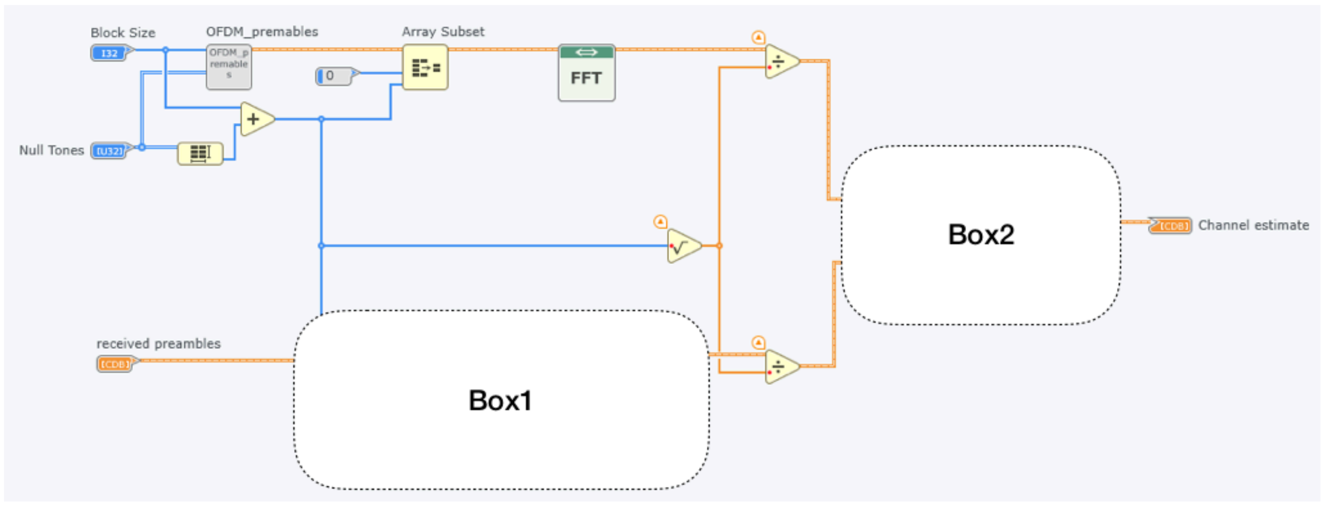
OFDM_ChEst.gvi	Terminal name	Type	Description
Input	*Block size (N-K)*	Unsigned integer	N-K length
	Null Tones	Unsigned integer array (1D)	K-location to remove the null tones
	Received preambles	Complex double array (1D)	Two preambles
Output	*Channel estimate*	Complex double array (1D)	

Recall that the channel can be estimated by using the preambles as following:

- Use two preambles to average noise: $\tilde{H}(f) = \dfrac{Y_1(f) + Y_2(f)}{2\,X(f)}$

where $\boldsymbol{X}(\boldsymbol{f})$ is the original transmitted preamble and $\boldsymbol{Y_1}(\boldsymbol{f}), \boldsymbol{Y_2}(\boldsymbol{f})$ are the received two preambles.

Complete Box1/Box2 in the following figure. Box1 performs averaging two received preambles and transforms FFT to convert into the frequency domain. Box2 divides the FFT of the received preambles by the original preamble. Note that $\mathbf{1}/\sqrt{\boldsymbol{N}}$ is used for the normalization factor for FFT.



2.10. OFDM Channel Correction

We will implement a block that corrects the channel. This block simply divides the received symbols $\boldsymbol{Y(f)}$ by the channel response $\boldsymbol{H(f)}$**.**

OFDM_ChCor.gvi	Terminal name	Type	Description
Input	*Received symbols*	Complex double array (2D)	
	Channel estimate	Complex double array (1D)	N length
Output	*Channel corrected symbols*	Complex double array (2D)	

3. Putting Together in Simulation

Before proceeding further, make sure all the blocks work in correct way. In this section, we will simulate OFDM Tx/Rx to verify all the blocks we created work together.

1. Use any modulation scheme (e.g. 4QAM) and do bit-to-symbol mapping.
2. Connect the blocks as shown in Figure1. Make sure all the dimensions are correct each step.
3. FFT/IFFT blocks in Figure1 should process 2D array in row-by-row. Use for loop and auto-indexing to get a row from 2D array. Use a proper normalization factor for FFT/IFFT.
4. Try a simple case for the block size, CP length, and Null tones. Verify the final output symbols at the Rx is same as the original symbols at the Tx.

3.1. Question

3.1.1. (Null Tones) We want FFT size N as 128 and K as 8. What is the right index values for Null tones to avoid the DC bin and the guard bins? Explain why.

3.1.2. (Cyclic Prefix) One advantage of CP is that it preserves orthogonality by allowing some misalignment. Add an alignment error at the detected peak index from the frame sync block (e.g. if you detected 100 as a peak, add 1 to make it 101). Use the following configuration.

- Mod type = 4QAM
- Message length = 1000bits
- Block size = 60
- Length of CP = 8
- Null tones = [0,31,32,33]
- Misalignment error = +1

1) Compare the demodulated constellation at the receiver with CP and without CP (length 0). 2) Increase the misalignment error until the demodulation starts to fail. What value is the misalignment error?

3.1.3. (Channel Correction) To verify the channel estimation/correction work, convolve the transmitted symbols with the channel coefficients (time-domain). Use the following configuration.

- Mod type = 4QAM
- Message length = 1000bits
- Block size = 60
- Length of CP = 8
- Null tones = [0,31,32,33]
- Channel coeff. = [0.5+0.3i, 0.01+0.1i, -0.01+0.2i, 0.02-0.05i]

4. Putting Together in USRP

In this section, we will implement OFDM in USRP.

1. Remove upsampling/pulse shaping in Tx and downsampling/match filter in Rx. In Tx, connect the output of OFDM_add_preambles to USRP write Tx data block. In Rx, the received samples connects to frame sync block.
2. In Rx, make sure you capture two OFDM packets (before it was two frames). You need to count all the overhead (CP, Null tones, preambles).

4.1. Question

Use the following configuration.

- Mod type = 4QAM
- Message length = 1000bits
- Block size = 60
- Length of CP = 8
- Null tones = [0,31,32,33]
- IQ rate = 400kHz
- Carrier = 1GHz

4.1.1. What is the total overhead in one OFDM packet?

4.1.2. What is the data rate?

4.1.3. What is the subcarrier width?

4.1.4. Report the following figures.

- Demodulated constellation on data without channel correction
- Demodulated constellation on data with channel correction
- Demodulated constellation on preamble without channel correction
- Demodulated constellation on preamble with channel correction

4.1.5. Repeat Q.4.1.4 with CP length 30.