EE340 Communications Lab

Orthogonal Frequency Division Multiplexing

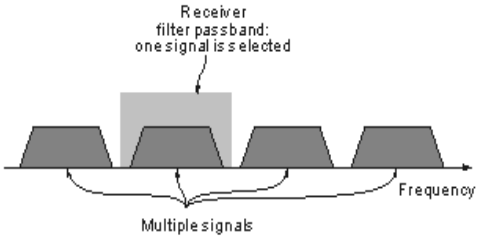
Kalpesh Patil (130040019)
Krishna Reddy (130020118)

Introduction

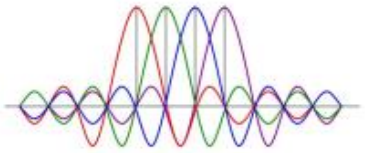
Orthogonal Frequency Division Multiplexing (OFDM) system is implemented using GNU Radio. A text file was encoded using flow graph in GNU Radio and resultant stream was wirelessly transmitted with the help of USRP. Another USRP was used at the receiver end. Decoding was done using GNU Radio and output was stored in a text file. Few custom made blocks were used in the flow graphs of transmission and reception part. Later bit error rate was calculated by comparing input and output files.

OFDM

In Frequency Multiplexing different data channels share the available bandwidth. Traditionally individual message signals were sent using different carriers in such way that they don't overlap with each other. This is not the case with OFDM. Although the sidebands from each carrier overlap, they can still be received without the interference that might be expected because they are orthogonal to each another. This is achieved by having the carrier spacing equal to the reciprocal of the symbol period.



Traditional view of receiving frequency modulation

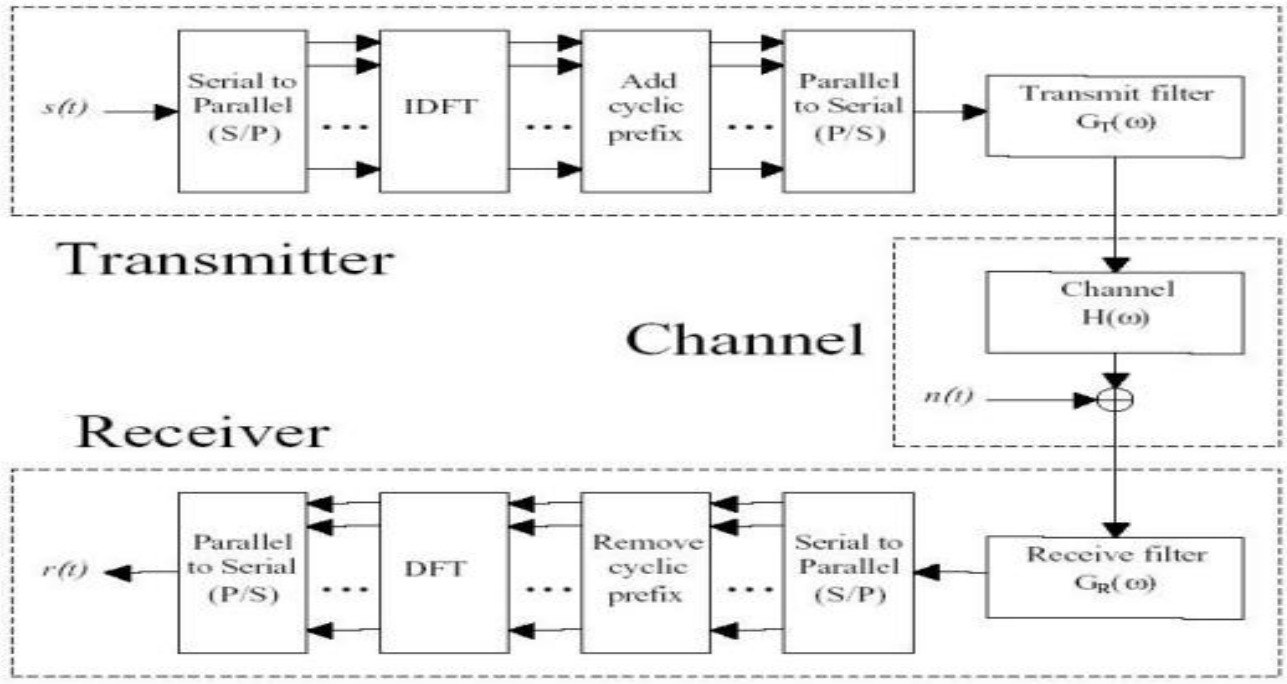


OFDM subcarriers

Each of the OFDM subcarriers has a range of frequencies assigned to it, and all of them together fill the spectrum used for the OFDM carrier, that is the bandwidth available.

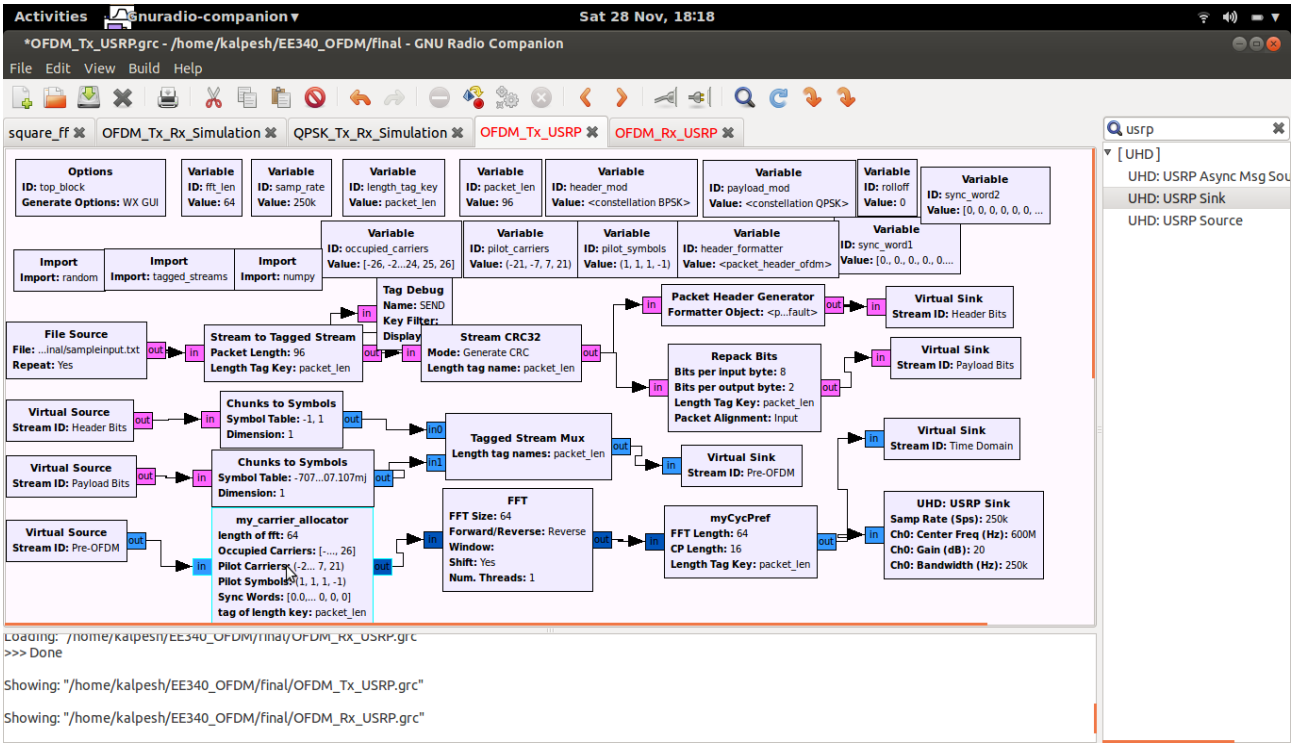
The data stream will be split for the subcarriers by using serial to parallel converter. For each subcarrier it will be individually modulated (QPSK in our case). The vector is passed through IFFT (inverse FFT) block which creates a time domain signal. This creates a spread spectrum broadband signal which contains data of all the subcarriers. Cyclic Prefix is added before transmitting it at RF frequency to compensate Multipath Effect. At the receiver end, first Cyclic Prefix is removed first and then FFT of the signal is taken. Final signal is obtained by converting it the stream.

OFDM Block Diagram



GNU Radio Flow Chart

Transmitter

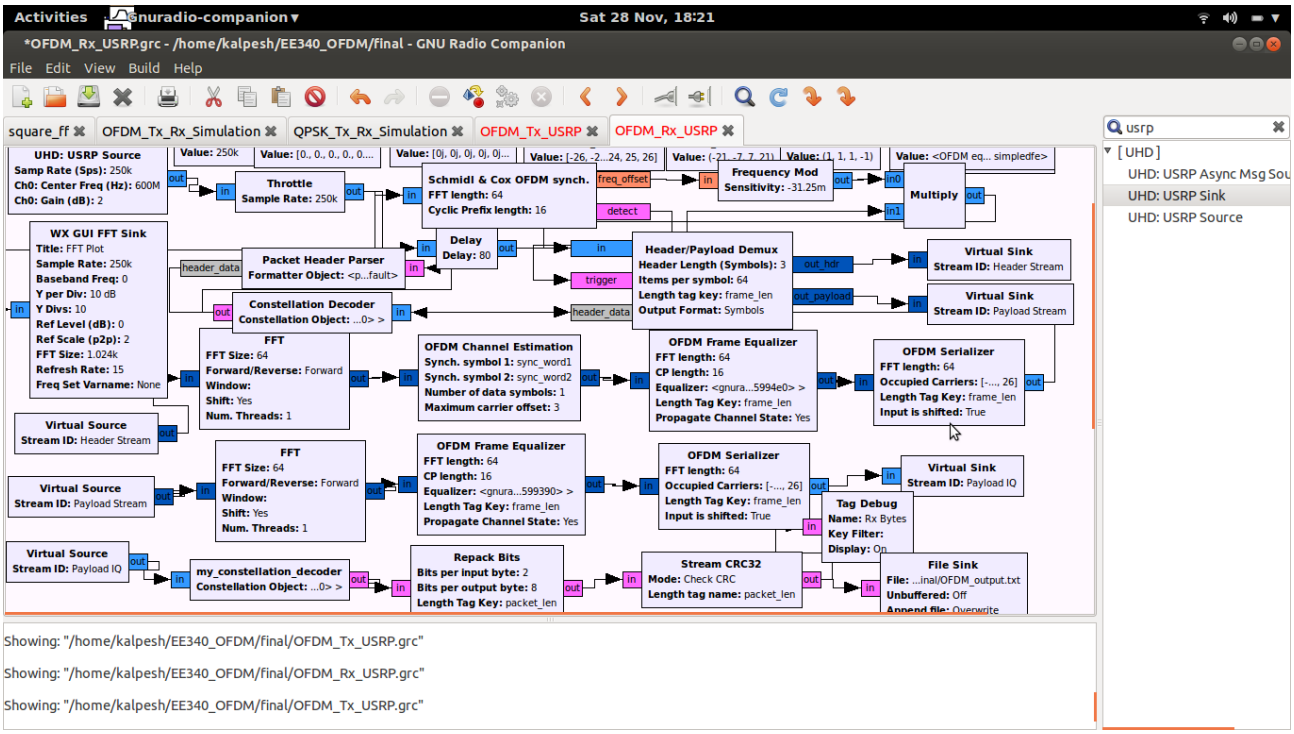


Description of the major blocks used in transmitter

- **Stream To Tagged Stream**
  It converts the continuous stream of bytes into tagged streams. The tags are inserted at the packet boundaries. Knowing Packet boundaries is necessary for some blocks which are used further.

- **CRC32**
  Cyclic Redundancy Check is a method which appends few bits (in our case 32) at the end of the packer. These 32 bits are calculated by successive division (XOR) on the input data and final remainder left is the CRC32 value that is going to appended in the output
- **Packet Header Generator**
  Packet header contains information related to the payload, which includes length of the payload, type of data etc.
- **OFDM Carrier Allocator**
  This block allocates available carriers to data symbols, pilot symbols and sync words. Few carriers at the border are left unoccupied (zero is allocated) to maintain the smoothness of the spectrum. Pilot symbols are sent at particular pilot frequencies, so as to detect frequency offset at the receiver end.
- **FFT**
  This block calculate reverse FFT of the input data symbol. Also the FFT is shifted so as to avoid jamming due to high value at DC
- **OFDM Cyclic Prefixer**
  This blocks majorly performs 2 tasks
  1. Adding Cyclic Prefix: The signal is extended by adding certain values from the end in the front. This allows some transmission in guard band and avoids jamming of the receiver
  2. Pulse Shaping: It removes the discontinuities in the digital signal by pulse shaping with the help of root raised cosine filter.
- **UHD USRP Sink**
  RF frequency of 600MHz was used to transmit the data through USRP

Receiver



Description of the major blocks used in receiver

- **UHD USRP Source**
  This block is used to receive the wireless signal
- **Schmidl and Cox OFDM Synch**
  This block calculates frequency offset added due to channel. It also detects if header has arrived and generates a trigger signal for Header/Payload Mux on the arrival of header.
- **Header/Payload Demux**
  This block separates header and payload bits. Header which is generated on the trigger is passed through different blocks and finally header data is decoded. The header data is fed back to this block which tells information about payload and thus the block is able to separate heaer and payload bits
- **OFDM Channel Estimation**
  Based on the sync words, this blocks estimates the channel response and appends a tag containing information about frequency offset etc to the output stream.
- **OFDM Equalizer**
  Depending upon the estimation from Channel Estimator, equalizer tries to remove the channel effect and generates a stable output, from which original data can be recovered. It also removes the multipath effect and ISI
- **OFDM Serializer**
  It works opposite to the carrier allocator block. It separates occupied and pilot carriers. Finally the symbols are decoded after constellation decoder and output is fed to a text file after passing through CRC32.

Custom Made Blocks

1. Carrier allocator
2. Cyclic prefixer
3. Constellation decoder

Bit error calculation

Output file was compared with the input file. No. of lines which were received erroneously were found out and divided by the total no. of bits in one line; which gave us the bit error rate (not exact bit error; but a rough estimate). C++ code was written for this purpose

Challenges faced

1. Interfacing USRP with PC
   USRP was connected using LAN to the PC. Few drivers were required to be installed for this purpose.
2. Installing custom made blocks in GNU
   New blocks were created with help of tutorial about out-of-tree module available on official GNU radio. Also we had to refer GNU documentation time to time for working on various GNU blocks