Lab 8: Match filtering

Wadhwani Electronics Lab

Department of Electrical Engineering
Indian Institute of Technology, Bombay.

Aim of the experiment

1. Implementing a digital modulation scheme.
2. To understand working of an match filter.

Pre-lab Work

- Study how to determine a match filter and pulse shaping works.
- Make sure that you have read the supporting material uploaded along with this document.
- For additional information about equalizers, refer Adaptive Filter Theory, Simon Haykin.

Part 1: BPSK Transmitter

- Generate a BPSK constellation using the blocks available in GNU Radio.
  - For this use the "Random Source" (generating bytes with min=0, max=2). Send the output to "Chunks to Symbols" block with 2 constellation points (equally separated on an unit on x-axis). This output should goes to the pulse shaping filter implemented by raised root cosine filter i.e. expt 7.
  - Observe the output constellation .
- Transmit the the real modulated signal with the carrier of 100 Khz with sampling rate=2 Mhz.
- Add the Gaussian noise with the noise source and vary the amplitude from 0 to 1.

Part 2:BPSK Receiver

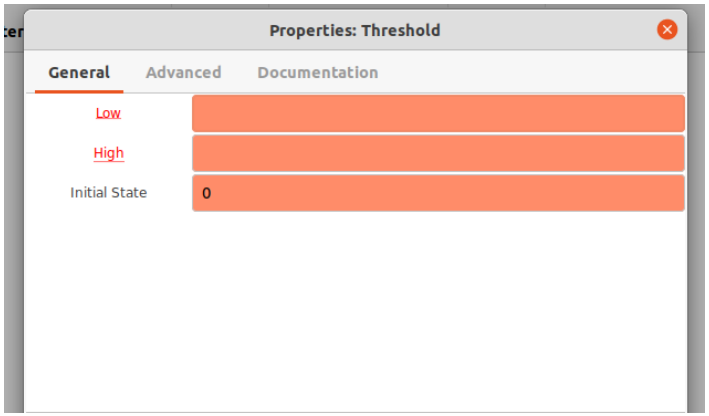
- Demodulate the received signal and observe the output signal in the presence of noise.
- Implementing the match filter by using the root raised cosine filter and observe the output and compare it with the above output with the variation of noise.
  - Extract the symbols from the waveform to the required symbols.
  - To get the symbols use the Threshold block and observe the received constellation.More info for threshold block is given in next slides.

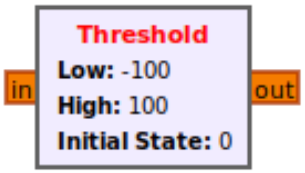
Part 3:Transmitting and receiving a text message.

- For this use the transmitter and receiver in part1 and part 2 to send and receive text file as message.
- Use the below given blocks to for this part.

Threshold

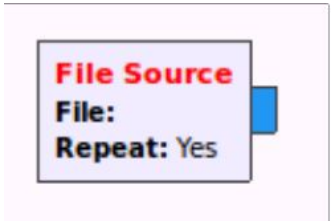
Threshold: Output a 1 or zero based on a threshold value.
Test the incoming signal against a threshold. If the signal exceeds the High value, it will change its output to 1, and if the signal falls below the Low value, it will change its output to 0.



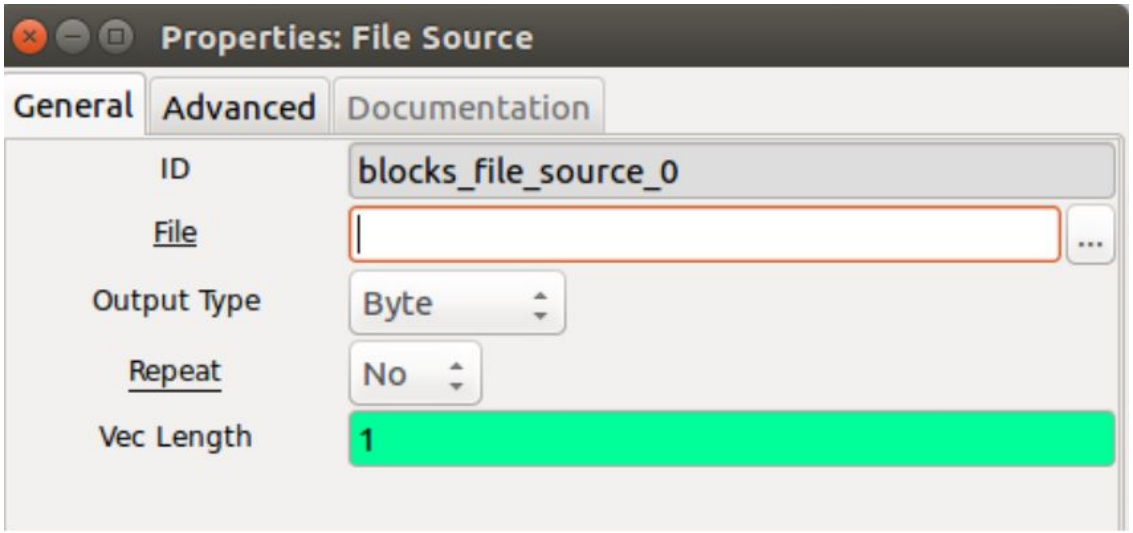


File Source

File Source - Reads raw data values in binary format from the specified file.



Note : Give the path to the text file.
Output Type - Byte Repeat - No



Unpack K Bits

Unpack K Bits : Converts a byte with K relevant bits to K output bytes with 1 bit in the LSB.

Since a text file consists of ASCII characters (Each character is represented in 8 bits), we use this block to extract the bit stream corresponding to the text file.

If input to Unpack K Bits is 10000001 then output will be 00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000001. So every output of Unpack K Bits will in byte format but will represent a single bit.

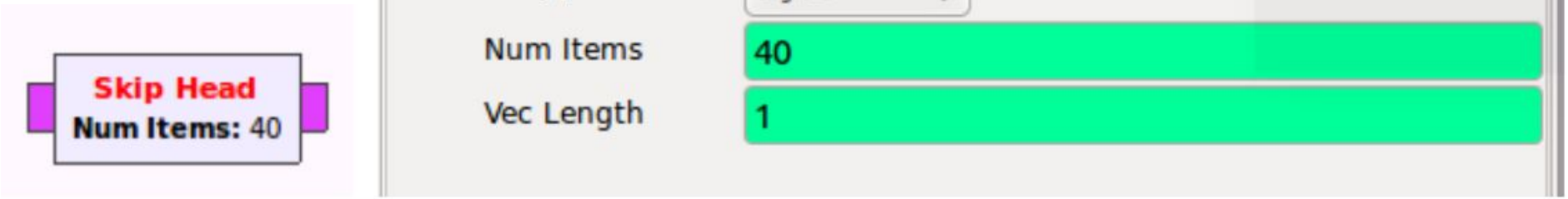
Note : K = 8

Float To UChar

To Change the format of data stream from float to UChar (Byte Format)

Skip Head

Skip the first N items, from then on copies items to the output. Useful when there are metadata or junk at the start. The processing in the GNU Radio adds some junk values in the start. If we don't use this then packing bits in byte format will cause wrong packing of ASCII code.



Note : Since the no. of junk bits added depends on your flow and the modulation scheme, so you need to do iterative trial to get the correct value. (Take help from TA's)

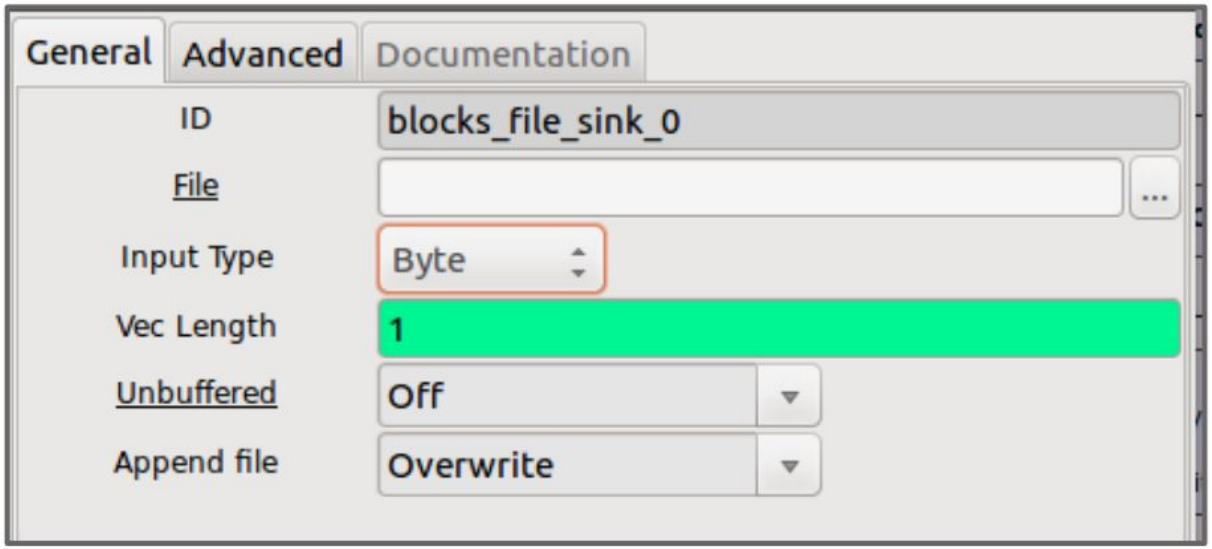
Pack K bits

Pack K Bits : Converts a vector of bytes with 1 bit in the LSB to a byte with K relevant bits.

After processing the data in bit format, to read whether we performed modulation and demodulation correctly we need to pack this bits into a byte. (K = 8)

File Sink

File Sink : Outputs raw data values in binary format to the specified file.



Note : Save file with .txt format