CS-330 Project
A complete BPSK,QPSK telecommunication system
Phase B

December 11, 2023

1. Receiver

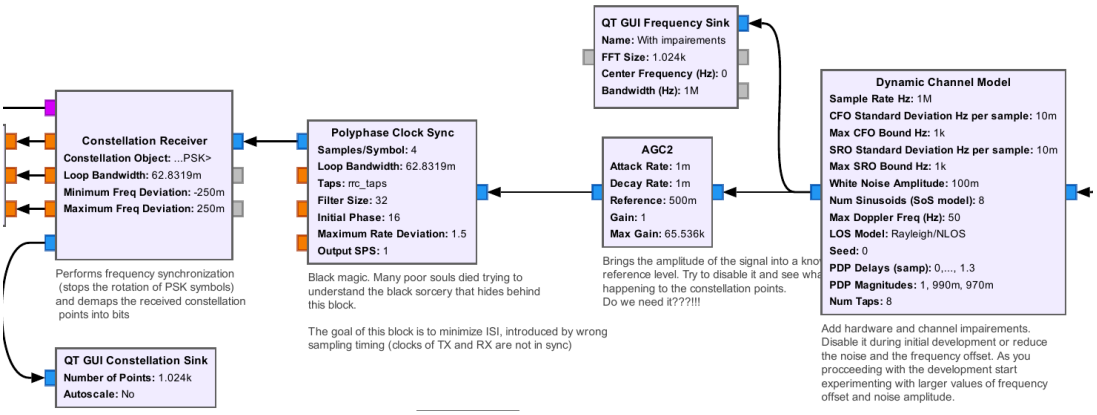

Figure 1: Receiver analog domain

Unfortunately, the receiver has to deal with all the impairments that a wireless channel introduces. Right from the flowgraph, you will observe that the complexity of a receiver is significantly greater than the transmitter. Hopefully, most of the required signal processing tasks are performed with existing GNU Radio blocks. Figure 1 shows the analog domain of the receiver. The *Dynamic Channel Model* is a channel emulator and tries to introduce a number of channel and hardware impairments, like frequency offset, multipath and noise. Using this block, makes the signal fed into your decoder more realistic. As you start implementing your receiver blocks, *it may be useful to disable this block*. As you continue, re-enable it and start to experiment with larger frequency offsets and noise amplitudes.

The AGC block is used to transform the input signal level into a reference amplitude, in our case $\pm\frac{\sqrt{2}}{2}$. This will enable to maximize the SNR at the *Constellation Receiver* block. Before retrieving the bit symbols, the *Polyphase Clock Sync* block performs decimation and clock synchronization, minimizing the ISI (Intersymbol Interference). The *Constellation Receiver* block tries to minimize the rotation of the constellation points due to possible frequency offsets and converts each complex symbols to the corresponding bits based on the constellation used. However, depending the constellation order, each output item of the *Constellation Receiver* block may carry a different number of bits. This could be 1 bit per item for the BPSK, 2 bits for the QPSK, etc. In order to simplify the logic of the receiver, items produced from the *Constellation Receiver* are passed through the *Unpack K bits* block. The parameter K is automatically retrieved from the **constellation** parameter. Therefore, after this block there is a stream of unpacked bytes containing one useful bit per byte for any given constellation.

Then the receiver operates in the digital domain, as shown in Figure 2. The frame synchronization block extracts the payload from a received frame. The resulting frame passes through the deinterleaver and the FEC decoder. Then the decoded payload passes through a CRC check. If the computed CRC matches the CRC carried by the frame, the PDU is propagated for printing through the *Debug Message* block. The CRC check is

already implemented and you do not have to do something for it.

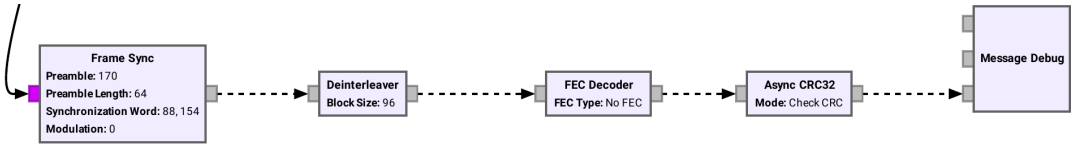

Figure 2: Receiver digital domain

1.1 Phase 2: Frame Synchronization (40%)

The frame synchronization is perhaps the most crucial task of the receiver. It is responsible to identify the reception of a frame, correct any possible constellation rotation issues, retrieve the frame length and then extract the payload.

The *Framer Sync* block accepts three parameters:

1. **Preamble:** The pattern used by the transmitter
2. **Preamble Length:** The length of the preamble used by the transmitter
3. **Synchronization Word:** The synchronization word used by the transmitter
4. **Modulation:** The modulation used. 0 for BPSK, 1 for QPSK. The reason why the modulation should be known is described in next paragraph.

In our case, for simplicity the synchronizer is implemented on the digital domain and not on the complex signal domain. That means that the frame synchronization should be done right after the bit information retrieval from each constellation point. However, this introduces a significant problem. The *Constellation Receiver* block performs blind frequency offset estimation and rotation of the constellation points. Without any pilot symbols, there is a phase ambiguity of $[0, \pi]$ for the BPSK and $[0, \frac{\pi}{2}, \pi, 3 \times \frac{\pi}{2}]$ for the QPSK. In other words, preamble and synchronization word may have been rotated. If the preamble is repeated and alternating sequence it is not affected. However, this does not hold for the synchronization An easy way to solve this problem is to use two in case of BPSK, four for QPSK FSMs in parallel. One that tries to synchronize with the original bits and the others with their rotated values.

The default state of the FSM is where the frame synchronization block searches for a portion of the preamble. As the preamble is a repeated pattern, the synchronization does not need to match (and in most cases due to settling period, it cannot) the whole preamble sequence.

After finding this portion, the FSM proceeds to a state that searches for the synchronization word. If this word is not found after a number of bits (# of preamble bits + # of synchronization word bits), the FSM resets to the default state. Otherwise it continues by extracting the 2 byte frame length field (remember that this is Network Byte Order). Extract the payload and create a GNU Radio valid PDU (**pmt::cons(pmt::pmt_t meta,**

pmt::pmt_t data)). Note that only the payload should be placed into the PDU, without the preamble, synchronization word or the length field.

Provide the implementation of the frame synchronizer at the *lib/frame_sync_impl.cc* file.

- **Hint 1:** To increase the sensitivity of the synchronizer, you can allow some of the bits at the preamble and/or at the synchronization word to be wrong. Normally this is 2-10% of the their length. It is a quite common technique, due to the fact that most of the preambles and synchronization words are not protected by any error correction mechanisms.
- **Hint 2:** You may find the class **cs330_project2023::shift_reg** quite handy. It implements a shift register and overloads most of the bitwise operators.
- **Hint 3:** At this point do not test your syncronizer with the entire transceiver flowgraph. Use a simplified version as Figure 3 shows. Try to add some noise to check how your synchronizer performs.

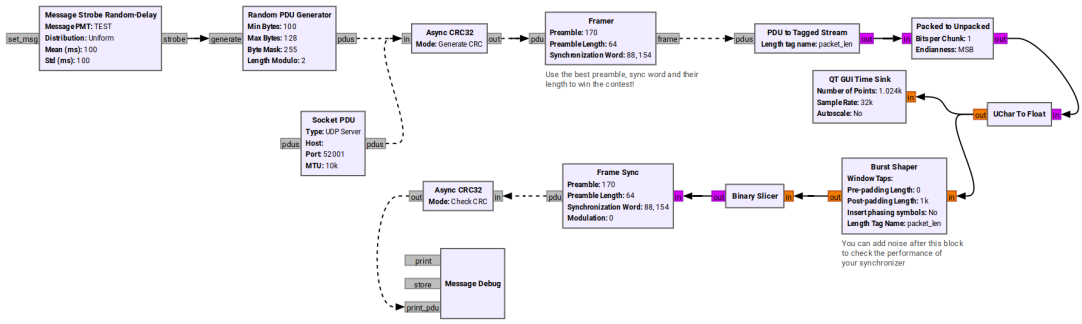

Figure 3: Receiver digital domain