Implementation of Faster-than-Nyquist Signaling using Software Defined Radios

Gregory Dzhezyan[†], Michel Kulhandjian[‡], Hovannes Kulhandjian[†]

[†]Department of Electrical & Computer Engineering, California State University, Fresno, Fresno, CA 93740, U.S.A.
E-mail: {gregorydz,hkulhandjian}@mail.fresnostate.edu
[‡]Electrical and Computer Engineering Department, Rice University, Houston, TX, 77005, U.S.A.
E-mail: michel.kulhandjian@rice.edu

***Abstract*—In this paper, we consider the development and implementation of a faster-than-Nyquist (FTN) signaling system using a binary phase-shift keying modulation (BPSK). We present a review of the theory and pertinent historical works related to the FTN signaling scheme, as well as practical work carried out in a laboratory environment. The popular open-source, and radio-specific, signal processing application GNU Radio was used to develop transmitter and receiver architectures. These architectures were then deployed for execution on various software-defined radio (SDR) hardware. We have evaluated the minimum Euclidean distance (MED) and probabilistic data association (PDA) estimators and performance in terms of bit-error-rate (BER) are compared. We observed for higher noise variance a BER of 10^{-3} is achieved for $\tau = 0.8$ when using MED and $\tau = 0.9$ when using PDA algorithm.**

***Index Terms*—Faster-than-Nyquist (FTN) signaling, pulse shaping, intersymbol interference (ISI), Mazo limit, partial response, self-interference, sequence estimation.**

I. Introduction

Since the seminal works of [1]–[3] showed that it would be possible to exceed the Nyquist signaling rate while maintaining an equivalent bit-error-rate (BER) performance, researchers have proposed many novel and creative methods for the implementation of a faster-than-Nyquist (FTN) system. A large majority of these works have concentrated on the development of algorithms for mitigating or undoing the effects of intentionally introduced inter-symbol interference (ISI). Furthermore, many of the works have relied purely on results produced from simulated models of FTN signaling. The reason for this being that these methods have very high computational complexity and would introduce far to great of an overhead cost on hardware, for example the turbo encoded and trellis based transceivers proposed in [4], [5].

However, there have been some works that have discussed and even attempted implementations of both FTN transmitters and receivers. These implementations have predominately been concentrated in the areas of non-orthogonal frequency domain multiplexing (NOFDM) and multi-carrier systems.

With regards to an FTN transmitter architecture, [6] proposes a look up table (LUT) based transmitter architecture for an OFDM system. The LUT implements a mapping of offset-QAM symbols to set of pre-computed optimum FTN symbols for input into a traditional OFDM transmitter. In [7] the proposed FTN mapping system is actually implemented targeting a field-programmable gate array (FPGA) device.

In [8] yet another transmitter architecture is implemented on FPGA but for a so called spectrally efficient frequency division multiplexed (SEFDM) system, which intentionally overlaps carriers in OFDM. The SEFDM signal generated by hardware is compared to simulated models. The proposed FTN mapper system in [7] is combined with an FTN receiver for a full transceiver system on FPGA hardware in [9]. The transceiver has the ability to default to standard OFDM signaling depending on the quality of the channel. Follow up papers for this transceiver system [10], [11] make improvements in the very-large-scale integration (VLSI) parameters of memory, area and power consumption.

To the best of our knowledge, the works discussed above are the only attempts made at hardware based implementation of an FTN communication system. All these works use FPGAs and target more advanced communications schemes. In this work, we seek to utilize existing software defined radio (SDR) hardware and the digital communications toolbox GNU Radio to implement a FTN transceiver based on a low-complexity probabilistic data association algorithm (PDA) [12]. The transceiver intentionally employs a basic binary phase-shift keying (BPSK) scheme with root-raised cosine (RRC) pulse shaping filters. We have evaluated the minimum Euclidean distance (MED) and PDA estimators and performance in terms of bit-error-rate (BER) are compared. We observed for higher noise variance a BER of 10^{-3} is achieved for $\tau = 0.8$ when using MED and $\tau = 0.9$ when using PDA algorithm.

The rest of the paper is organized as follows. In Section II, we present the signal model. Then in Section III, we present our GNU radio implementation. In Section IV, we present our experimental and simulation results before presenting our conclusion in Section V.

II. Signal Model

The generic architecture for a pulse amplitude modulated (PAM) communication system is shown in Fig. 1. The upper half of Fig. 1 is the transmitter chain and the lower section represents the receiver chain.

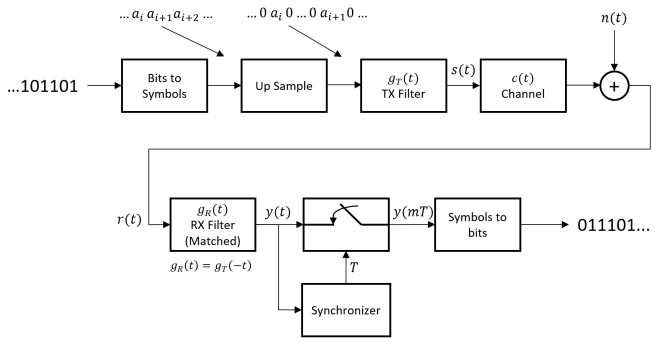

Fig. 1: Generic transmitter and receiver architecture for PAM signals.

In FTN signaling a new parameter τ is introduced into the PAM signal $s(t)$,

$$s(t) = \sum_n a_n g_T(t - n\tau T). \tag{1}$$

The parameter τ can be varied in the range $(0, 1]$, when $\tau = 1$ we obtain standard, ISI free, Nyquist signaling and when $\tau < 1$ faster-than-Nyquist signaling occurs. The parameter τ behaves as compression factor, forcing the pulse shaped symbols to come earlier than they normally would. Let the incoming data have a symbol rate F_d or symbol interval $T_d = \frac{1}{F_d}$ and let the sampling rate of the transmitter filter $g_T(t)$ be F_s with sample period $T_s = \frac{1}{F_s}$, where $F_s > F_d$. Now also define the samples-per-symbol (λ) to be,

$$\lambda = \frac{F_s}{F_d}. \tag{2}$$

Assuming N input symbols and using the information defined above, we can describe the input symbols to the system using the following discrete equation as

$$\sum_{n=0}^{N} a_n \delta(t - nT_d). \tag{3}$$

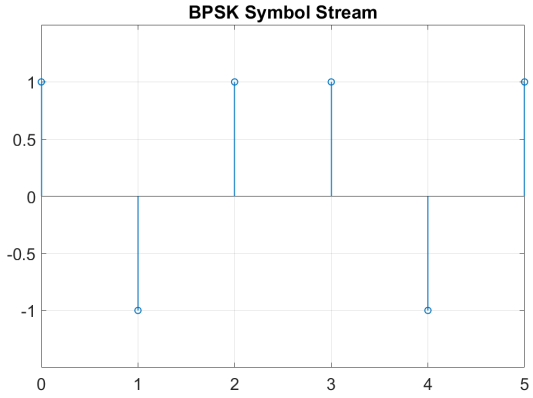

Fig. 2: A stream of BPSK symbols. The original data bits were $[1, 0, 1, 1, 0, 1]$, using the formula $2 * bit - 1$ the bits are mapped to BPSK symbols $[1, -1, 1, 1, -1, 1]$.

Fig. 2 shows an example of a symbol stream for $N = 6$ BPSK symbols that is described in (3). Note that in BPSK there is a one-to-one ratio of bits to symbols.

In addition, discretizing $g_T(t)$ using sampling rate F_s to produce M total filter samples of some pulse shape (e.g., $g_T(t) = p(t)$), describe the transmitter filter as

$$\sum_{m=0}^{M} g_m \delta(t - mT_s), \tag{4}$$

where g_m sequence are sampled values of the transmitter filter $g_T(t)$. Fig. 3 shows an example of the g_m sampled values for the the RRC pulse shape.

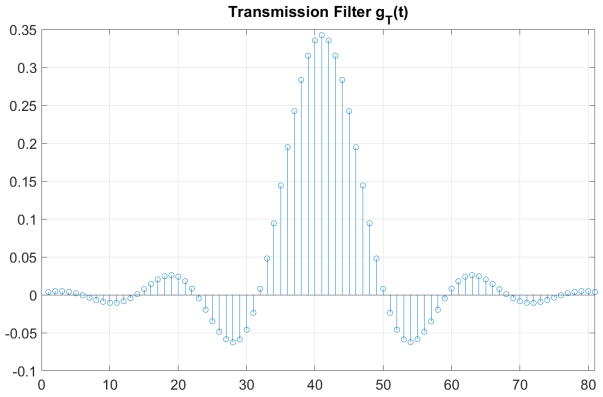

Fig. 3: Sampled values of a root-raised cosine (RRC) transmission filter with samples-per-symbols $\lambda = 10$, a filter delay of 4 symbols, and excess bandwidth $\alpha = 0.3$.

Using (3) and (4) the output of the transmitter chain can be described in discrete form using convolution as follows,

$$s(t) = \sum_{n=0}^{N} a_n \delta(t - nT_d) * \sum_{m=0}^{M} g_m \delta(t - mT_s) \tag{5}$$

$$= \sum_{n=0}^{N} \sum_{m=0}^{M} a_n g_m \delta(t - nT_d - mT_s). \tag{6}$$

Where (6) follows from the sifting property of the dirac delta function $\delta(t)$. From (2) we can re-write T_d in terms of T_s as

$$F_s = \lambda F_d \tag{7}$$

$$\frac{1}{T_s} = \lambda \frac{1}{T_d} \tag{8}$$

$$T_d = \lambda T_s \tag{9}$$

This allows us to write $s(t)$ as

$$s(t) = \sum_{n=0}^{N} \sum_{m=0}^{M} a_n g_m \delta(t - (n\lambda - m)T_s). \tag{10}$$

Expression in (10) describes the output of the transmitter filter in Fig. 1 for normal Nyquist signaling, we can see an up-sampling of the input symbols occurs due to the $n\lambda$ term, where n indexes the symbol stream. As an example, Fig. 4

shows the BPSK stream from Fig. 2 up-sampled by $\lambda = 10$ along with RRC shaped symbols placed at these locations. These RRC waveforms will eventually be summed together to produce the final waveform for transmission, which is shown in Fig. 5.

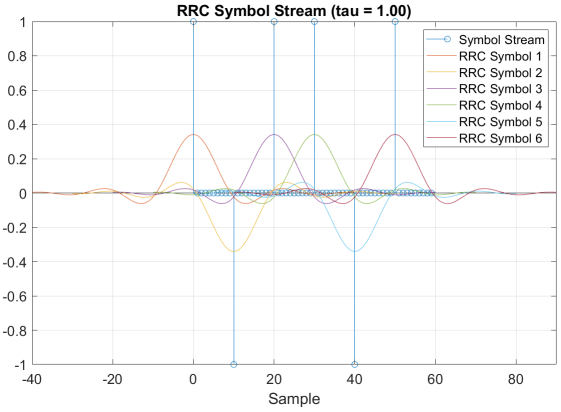

Fig. 4: RRC shaped symbol stream. The original BPSK symbol stream can be seen here but has been up-sampled by $\lambda = 10$. Up-sampling introduces $\lambda - 1$ zeros in between symbols.

To alter (10) to be reflective of FTN signaling, simply realize that (3) should be changed to include τ, forcing each symbol to come at an earlier time as follows,

$$\sum_{n=0}^{N} a_n \delta(t - n\tau T_d). \tag{11}$$

Therefore, the FTN transmission signal $s_{FTN}(t)$ can now be expressed as

$$s_{FTN}(t) = \sum_{n=0}^{N} \sum_{m=0}^{M} a_n g_m \delta(t - (n\tau\lambda - m)T_s). \tag{12}$$

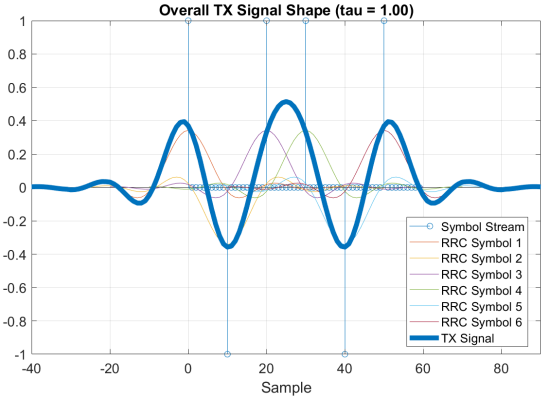

Fig. 5: Cumulative sum of the RRC shaped symbols, produces the final waveform, which will be transmitted across the channel.

We can see in (12) how τ impacts the original Nyquist signal (10) by effectively reducing the up-sampling rate, λ, of the symbol stream. An example of this can be seen in Fig. 6 for the extreme case when $\tau = 0.2$, causing the RRC shaped symbols to be packed closer together. The ultimate effect of this on the transmission waveform, as shown in Fig. 7. Comparing Fig. 7 to the one in Fig. 5, we can see that the final transmission waveform has become compressed, representing more data in a shorter period of time. Also note that in Fig. 6, it is obvious how ISI is introduced with lower values of τ since multiple RRC symbols overlap with each other at non-null regions.

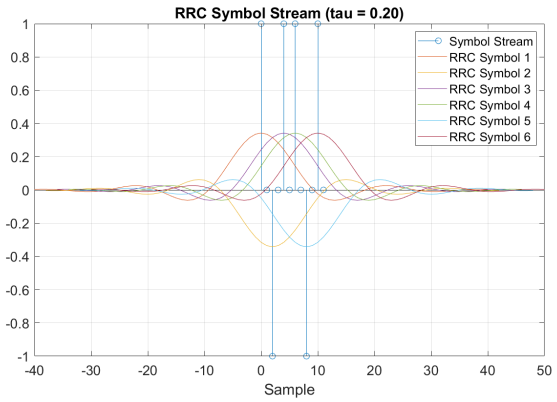

Fig. 6: RRC shaped symbol stream. The original symbol stream has been up-sampled by $\tau\lambda = 2$.

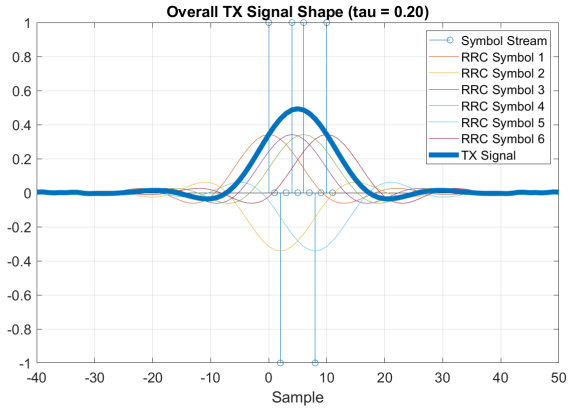

Fig. 7: Cumulative sum of the RRC shaped symbols produces the final waveform which will be transmitted across the channel. Note how compressed the waveform has become for $\tau = 0.2$.

Finally, to showcase a better idea of the effect of τ on the final transmission waveform, Fig. 8 shows the transmission waveform for a random sequence of $N = 50$ BPSK symbols for several values of τ. It can be observed how τ behaves as a compression factor from these waveforms. For example, refering to the third plot in Fig. 8, when $\tau = 0.6$ the waveform's extent in time has nearly been halved compared to when $\tau = 1.0$. Additionally, note how severely distorted the waveforms become as τ decreases compared to when $\tau = 1.0$, this distortion is the result of ISI.

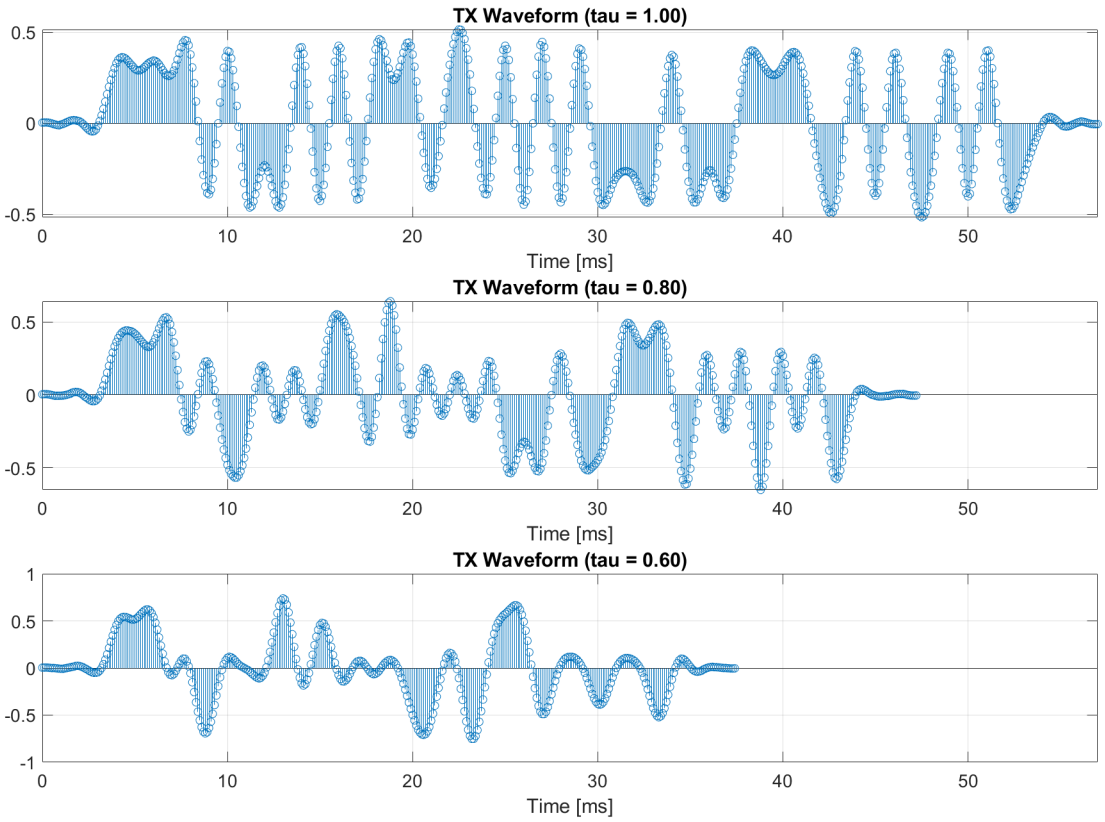

Fig. 8: Samples of the final transmission waveform for $N = 50$ random BPSK symbols and several values of τ. RRC pulse shaping filter with $\lambda = 10$, delay of 4 symbols, and $\alpha = 0.3$ was used as the transmission filter.

III. GNU Radio Implementation

The implementation of the FTN BPSK transceiver was carried out partially in GNU Radio (version 3.8.2.0) and in MATLAB. The SDR used was National Instruments (NI) USRP 2920. The down-sampling of the received waveform and the PDA and MED demodulation algorithms are implemented in MATLAB. This corresponds to the very last steps of the receiver chain in Fig. 1 (i.e., the synchronize/sample and symbols-to-bits blocks.)

Nearly all variables and parameters for blocks in the GNU Radio flow graphs of both the transmitter and receiver are controlled through a single Python script. Hence, all of the variables, the generated random data sequences, preambles, and file paths are all defined in the script and the GNU Radio blocks reference those values from the script. The script is imported into the GNU Radio environment using the Python module block.

A. Transmitter

The transmitter flowchart in Fig. 9 begins with a vector source block set to produce 10 packets of $10,064$ bits each. The first 64 bits of each packet is a known sequence called the preamble. The preamble sequence in hexadecimal is defined as $[\texttt{0xAC}, \texttt{0xDD}, \texttt{0xA4}, \texttt{0xE2}, \texttt{0xF2}, \texttt{0x8C}, \texttt{0x20}, \texttt{0xFC}]$.

The data produced by the vector source is in a byte format, and so the packed to unpacked block splits every byte it receives into 8 bits. The chunks to symbols block then maps every bit received into the appropriate BPSK symbol $0 \rightarrow -1$ and $1 \rightarrow 1$. The symbols then are convolved with the root raised cosine transmission filter. The RRC filter is setup such that it is an interpolating filter with interpolation factor set to $\texttt{truncate}(\tau\lambda)$ (e.g., if $\tau = 0.8$ and $\lambda = 20$ then interpolate/up-sample by 16). Note also that $\lambda = 20$, $\alpha = 0.3$, and the span of the filter (which is equal to two times the filter's symbol delay) is 8 (i.e., a 4 symbol delay).

Since BPSK symbols have zero imaginary component, the samples coming out of the RRC block are all real floating-point numbers, but in general radios will require complex input. Hence, a float to complex block is used. A stream mux block inserts a radio silence period in between the transmission of each packet. The stream mux block will take the appropriate amount of samples from the RRC symbol stream for one full packet and then it will take $20,000$ samples from a null source block (zero valued samples) to simulate the radio silence in between packets. The rational

resampler block up-samples the final signal to match an appropriate sampling rate required by the hardware (UHD: USRP block). Note that the BPSK data sequence generated from the chunks to symbols block is saved to a file.

B. Receiver

The receiver processing chain in Fig. 10 begins with the UHD hardware block in GNU Radio. The complex samples received from the radio are first down-sampled by the rational resampler block at the same rate as they were up-sampled in the transmitter. The sampled data is then passed through a low-pass filter with a cutoff frequency set to the expected symbol rate. A separate processing chain the splits off from the rational resampler block that calculates the squared average magnitude of the samples from the receiver, this chain is meant to calculate the energy of the received signal. This energy data is written to a file for further analysis.

Once the samples have made their way through the low-pass filter, they are fed into another RRC filter. The RRC filter here at the receiver contains the same exact filter taps as in the transmitter, hence it is a matched filter. After matched filtering, the energy is re-measured (matched filtering should theoretically have a significant effect on SNR.) The samples output from the matched filter are written to a file. This matched filtered file along with several other files recorded throughout the Tx and Rx flowcharts are read into MATLAB. Within MATLAB the known preamble is used to discover the starting locations of each transmitted packet using cross-correlation. The packets are segmented and then down-sampled at rate determined by $\tau\lambda$. This down-sampled data should represent the original symbol sequence transmitted, hence it is handed off to both the PDA and MED demodulation algorithms to produce data bits.

As a final note on the receiver, if it is desired such that two different radios are used, one for transmission and another for reception, then a modification must be made. A frequency lock loop (FLL) band edge block must be inserted at the very front end of receiver chain. The FLL band edge block will perform coarse frequency adjustment to account for the difference in oscillators due to the use of two separate radios.

IV. Experimental Results

In this section, we discuss the experimental setup and present the results. The testbed of the experimental setup is shown in Fig. 11. The SDRs used for testing were the NI USRP 2920 and the NI USRP X310. Additionally a simulation was conducted within GNU Radio where the transmitter and receiver were connected directly to each other using a channel model block. The channel model block was set to behave as an ideal channel (i.e., zero noise, no mulitpath, etc.) The parameter τ was varied from 1.0 to 0.5 in 0.1 increments in all tests. Ten packets, each containing 10000 random bits, were transmitted for each value of τ. Outcomes produced from these experiments were not necessarily as expected. It was anticipated the PDA algorithm would have superior performance over MED since it accounts for and corrects issues related to ISI.

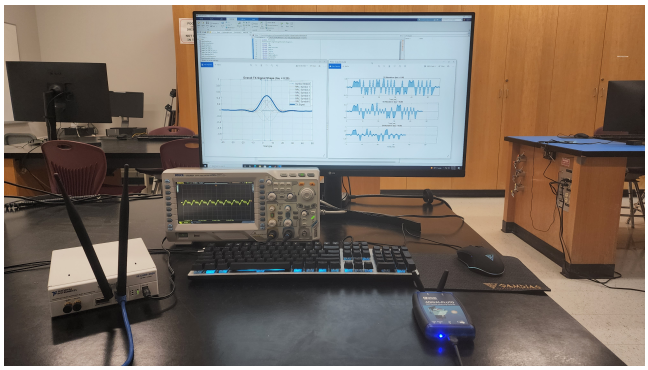

Fig. 11: Experimental setup of the NI USRP 2920.

PDA is very sensitive to noise. So two different simulations were conducted using the ideal channel, one where the noise variance given to the PDA algorithm was fixed to $\sigma^2 = 0.0158$ and another where the noise variance was set to the extremely low value of $\sigma^2 = 1.0 \times 10^{-10}$. This extremely low value was chosen since theoretically in an ideal channel the noise variance would be zero.

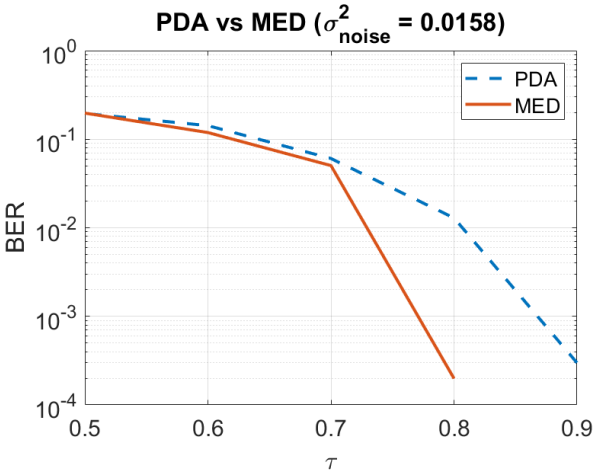

Fig. 12: Demodulation results for both PDA and MED with fixed noise variance of $\sigma^2 = 0.0158$. These results were produced by transmitting through a simulated ideal channel.

The results for both these tests are shown in Figs. 12 and 13. In both these simulations it can be seen that no errors are made by either algorithm when $\tau = 1.0$. In Fig. 13 both algorithms seem to be performing similarly with no errors until $\tau = 0.8$.

In Fig. 12 MED outperforms PDA detection scheme in terms of packing factor τ. MED achieves a BER of 10^{-3} with only $\tau = 0.8$ whereas PDA $\tau = 0.9$.

After the simulations, a test was performed using the USRP 2920 in a wireless loop-back configuration to transmit the FTN signals for the same values of the τ given above, as shown in Fig. 14. Here PDA makes no errors until $\tau = 0.8$, where it begins to significantly under perform compared to MED. A final test was conducted using the more advanced

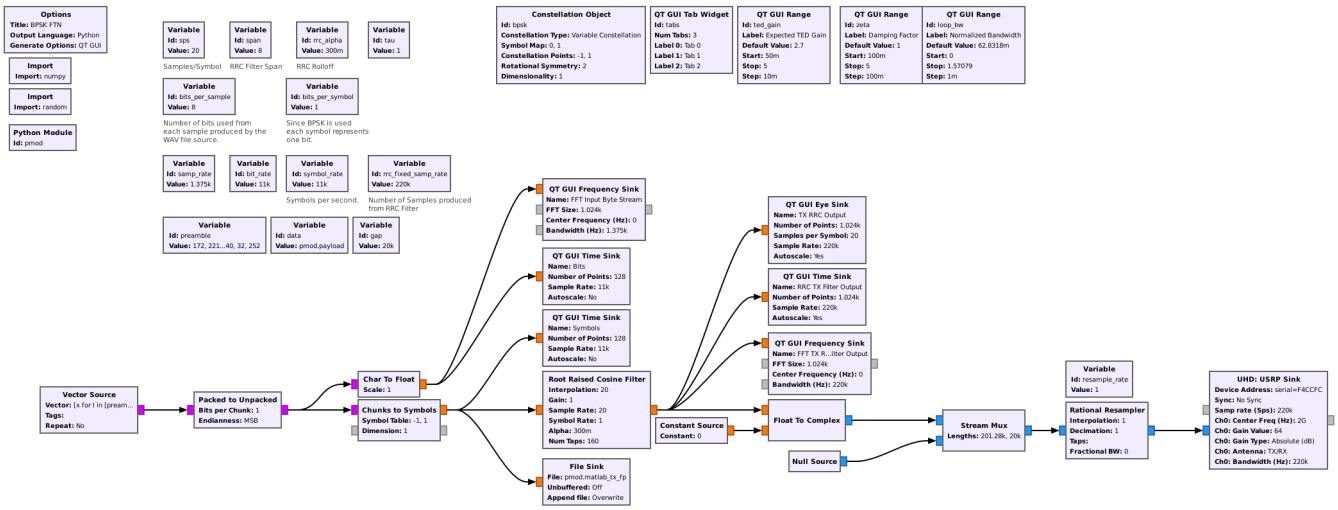

Fig. 9: GNU Radio Transmitter flowchart.

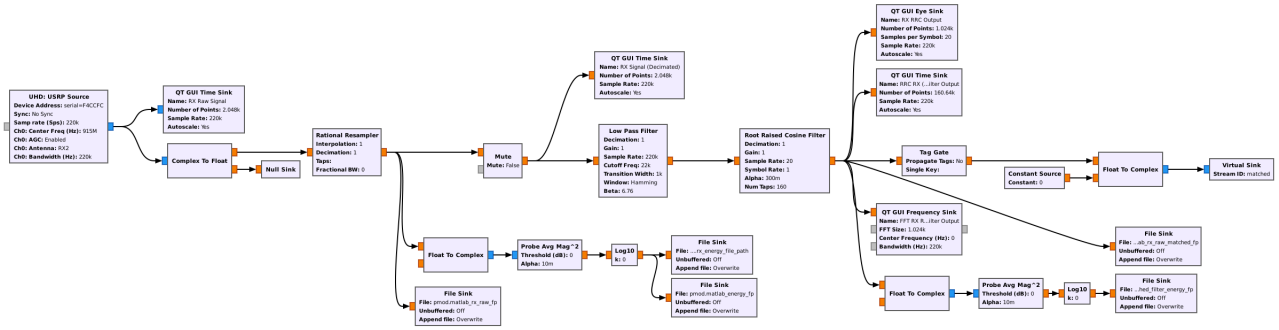

Fig. 10: GNU Radio Receiver flowchart.

SDR the USRP X310. The results of which are shown in Fig. 15. As we can see the performance of PDA and MED are fairly comparable. PDA achieves a BER of 10^{-3} with $\tau = 0.86$ while MED $\tau = 0.83$. For $\tau = 0.7$ and $\tau = 0.9$ they have near identical performance in terms of BER.

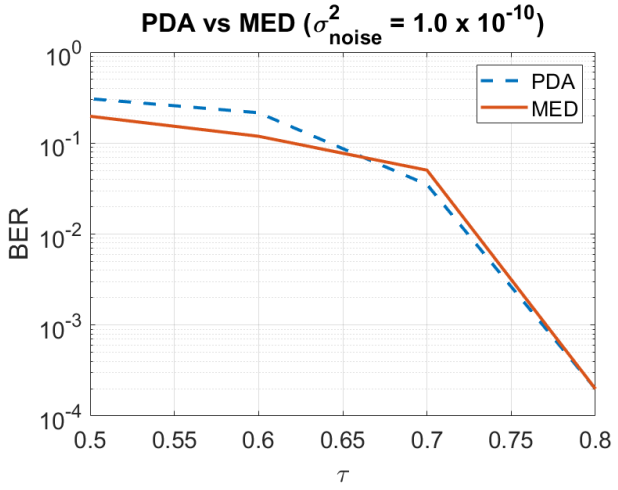

Fig. 13: Demodulation results for both PDA and MED with fixed noise variance of $\sigma^2 = 1.0 \times 10^{-10}$.

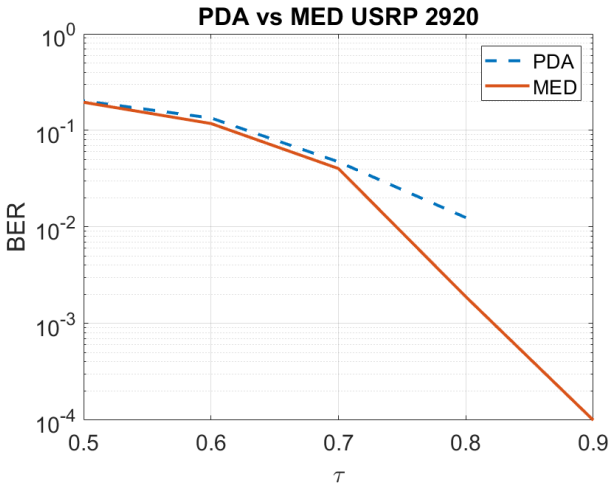

Fig. 14: Results for PDA vs MED using the USRP 2920.

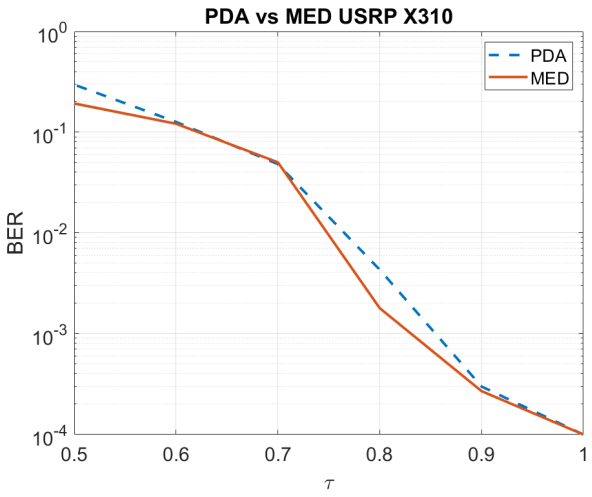

Fig. 15: Results for PDA vs. MED using the USRP X310.

V. Conclusion

In this paper, we have implemented a faster-than-Nyquist (FTN) communication system utilizing software defined radios (SDR). Moreover, we have applied two estimation schemes minimum Euclidean distance (MED) and probabilistic data association (PDA). Specifically, we have implemented the binary phase-shift keying (BPSK) FTN signaling over the additive white Gaussian noise (AWGN). The FTN communication system was implemented with GNU Radio and allows a user to modify the FTN compression factor τ. It was shown that in order to realize a FTN system the minimum modifications needed were to change the upsampling factor into the RRC filter on the transmitter side and to switch out the standard minimum distance demodulator at the end of the receiver chain for a FTN specific demodulator (e.g., PDA). The estimation performance of the Euclidean distance and PDA based schemes are evaluated in terms of bit-error-rate (BER). We observed for higher noise variance to achieve a BER of 10^{-3} with $\tau = 0.8$ when using MED and $\tau = 0.9$ when using PDA.

For future work, we are investigating implementations of FTN signaling for other higher constellations such as quadrature phase-shift keying (QPSK) and quadrature amplitude modulation (QAM). Additional investigations may include development of GNU Radio flow graphs for NOFDM signaling, extending the core idea of non-orthogonal signaling in FTN to the frequency domain by allowing inter-carrier interference (ICI) in OFDM. Yet another potential flow graph implementation may even combine time domain FTN and frequency domain FTN.

Acknowledgement

Special thanks to the Air Force Research Lab and Wright Brothers Institute for the student travel grant support. In addition to that this work was partially supported by the DoD Instrumentation Grant Number W911NF2110210.

References

[1] D. Tufts, "Nyquist's problem—the joint optimization of transmitter and receiver in pulse amplitude modulation," *Proceedings of the IEEE*, vol. 53, no. 3, pp. 248–259, 1965.

[2] B. Saltzberg, "Intersymbol interference error bounds with application to ideal bandlimited signaling," *IEEE Transactions on Information Theory*, vol. 14, no. 4, pp. 563–568, 1968.

[3] J. E. Mazo, "Faster-than-nyquist signaling," *The Bell System Technical Journal*, vol. 54, no. 8, pp. 1451–1462, 1975.

[4] A. Liveris and C. Georghiades, "Exploiting faster-than-nyquist signaling," *IEEE Transactions on Communications*, vol. 51, no. 9, pp. 1502–1511, 2003.

[5] F. Rusek and J. B. Anderson, "Serial and parallel concatenations based on faster than nyquist signaling," in *2006 IEEE International Symposium on Information Theory*, 2006, pp. 1993–1997.

[6] D. Dasalukunte, F. Rusek, J. B. Anderson, and V. Owall, "Transmitter architecture for faster-than-nyquist signaling systems," in *2009 IEEE International Symposium on Circuits and Systems*, 2009, pp. 1028–1031.

[7] D. Dasalukunte, F. Rusek, V. Öwall, K. Ananthanarayanan, and M. Kandasamy, "Hardware implementation of mapper for faster-than-nyquist signaling transmitter," in *2009 NORCHIP*, 2009, pp. 1–5.

[8] M. R. Perrett and I. Darwazeh, "Flexible hardware architecture of sefdm transmitters with real-time non-orthogonal adjustment," in *2011 18th International Conference on Telecommunications*, 2011, pp. 369–374.

[9] D. Dasalukunte, F. Rusek, and V. Owall, "Multicarrier faster-than-nyquist transceivers: Hardware architecture and performance analysis," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 4, pp. 827–838, 2011.

[10] D. Dasalukunte, F. Rusek, and V. Öwall, "Improved memory architecture for multicarrier faster-than-nyquist iterative decoder," in *2011 IEEE Computer Society Annual Symposium on VLSI*, 2011, pp. 296–300.

[11] D. Dasalukunte, F. Rusek, and V. Öwall, "A 0.8 mm2 9.6 mw implementation of a multicarrier faster-than-nyquist signaling iterative decoder in 65nm cmos," *2012 Proceedings of the ESSCIRC (ESSCIRC)*, pp. 173–176, 2012.

[12] M. Kulhandjian, E. Bedeer, H. Kulhandjian, C. D'Amours, and H. Yanikomeroglu, "Low-Complexity Detection for Faster-than-Nyquist Signaling based on Probabilistic Data Association," *IEEE Communications Letters*, pp. 1–5, Dec. 2019.