

ELEC 391 - Electrical Engineering Design Studio II
Dr. Paul Lusina & Dr. Cristian Grecu
Section 941

Demo 1 Report: Communication Model & Design Simulation
Team #: 2 - Bryant Ariola, Akshat Ranjan, Jiahao Yan
Scenario: B

Table of Contents

Objectives, Requirements & Constraints.....	2
System Overview.....	3
Source/Sink.....	4
A/D & D/A Converters.....	6
Error Correction Encoder/Decoders.....	7
Channel.....	9
Modulators/Demodulators.....	11
Transmitter/Receiver.....	12
References.....	14

Objectives, Requirements & Constraints

Design Specifications: (Scenario B)

Performance Spec B

Table 6: Performance Scenarios

Scenario (Teams)	High Priority			Low Priority	
	Audio BW (kHz)	Target BER	Spectral Mask (kHz)	Target Channel	Delay (ms)
B	8	10^{-4}	100	γ	25

Target Channel (Gilbert Channel)

γ	$p_{GB}^{(\gamma)} = 0.05$	$p_{BG}^{(\gamma)} = 0.20$
----------	----------------------------	----------------------------

Objectives:

The objective of this project is to design and implement a digital communication system on a FPGA that meets the requirements that are listed above. Additionally, the digital communication system must be simulated and tested on MATLAB Simulink with satisfactory results complying to the requirements listed.

Requirements: Listed in the Design Specifications

Constraints:

1. Use of FPGA for Implementation: The design for the digital communication system must be implemented on a FPGA system, such as a DE1-SOC development board.
2. Use of Simulink for Simulation: The design for the digital communication system must be simulated and tested on MATLAB Simulink.

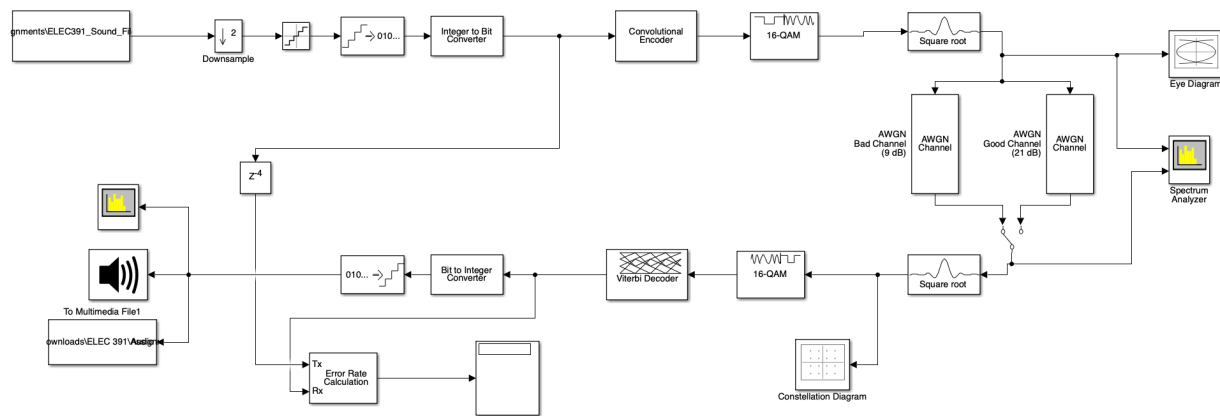
Systems Progress Report:

1. ADC & DAC: Satisfying design constraints and requirements
2. Encoding/Decoding: Satisfying design constraints and requirements
3. Modulation/Demodulation: Satisfying design constraints and requirements
4. Transmitter/Receiver: Satisfying design constraints and requirements
5. Channel: Partially satisfying design constraints and requirements (Gilbert implementation, error rate still ~10 times higher than required)

Significant Design Challenge:

The most significant design challenge has been to balance out the trade-offs to ensure a low Bit Error Rate (BER), and complying with the spectral mask, as well as ensuring that the output of that is received has high fidelity and low latency. To address this challenge, the team has ensured to test individual subsystems for the bit-error rate as part of verification tests, and also looked at trade-offs in quantization levels. Although the challenge does remain, significant progress has been accomplished towards achieving the requirements and the constraints set by the scenario assigned.

System Overview



The system uses an audio input file known as the multimedia file. After that the input signal goes through the analog to digital converter, in which it is first down-sampled to an appropriate sampling rate, followed by quantization. After this step, the data type is changed to an integer using a uniform encoder. The integer to bit converter then converts the input signal into binary, and sends it to the convolution encoder. The encoder works on the basis of parity bits that help improve reliability of the data transmission. The next step is modulation, in which 16-QAM modulation is used, it makes use of a 4 by 4 matrix grid for modulation, and each symbol represents 4 bits of data. The signal then goes through the square root raised cosine transmit filter, to then be fed into the channel. The channel is divided into a good channel and bad channel, with the decision of which channel is being utilized dependent on the Gilbert Fading model, that makes use of the probabilities to transition between good and bad states. After the channel, the square root raised cosine receive filter receives the signal, and the opposite steps are carried out as the signal goes through demodulation and decoding. The signal is then fed back to a digital to analog converter, which consists of a bit to integer converter, that converts the binary into integers. The integers are then fed into a uniform decoder which outputs a data type that results in the formation of the audio file.

Power Transmitted - 1 Watt (W): 16-QAM modulation and the filtering help the power stay constant, ensuring reliable signal

Bandwidth - The bandwidth is maintained through the sampling rate, and the ADC/DAC system along with the modulation system in place.

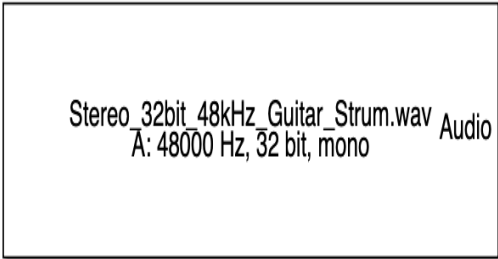
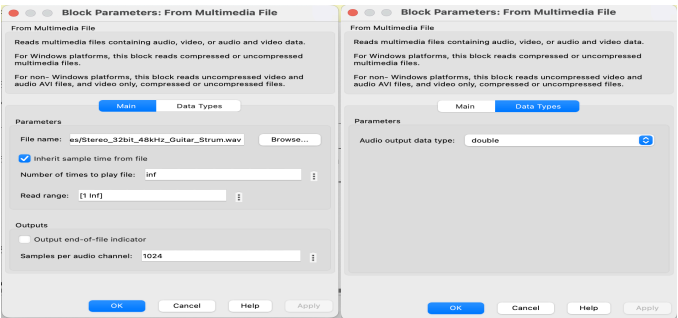
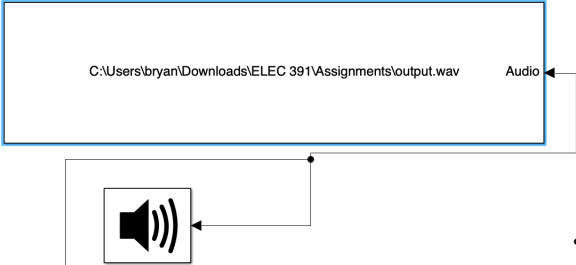
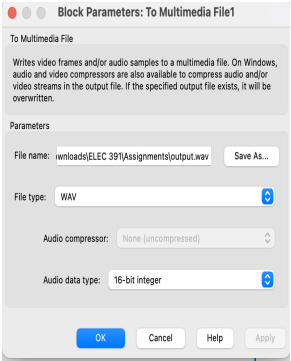
BPS (Bits per second) - This is determined through the sampling rate and the modulation. 16-QAM transmits 4 bits per symbol.

Error Rate - The error rate is maintained at the specified value through the encoder and the decoder in place, specifically the convolutional encoder and the Viterbi decoder.

Source/Sink

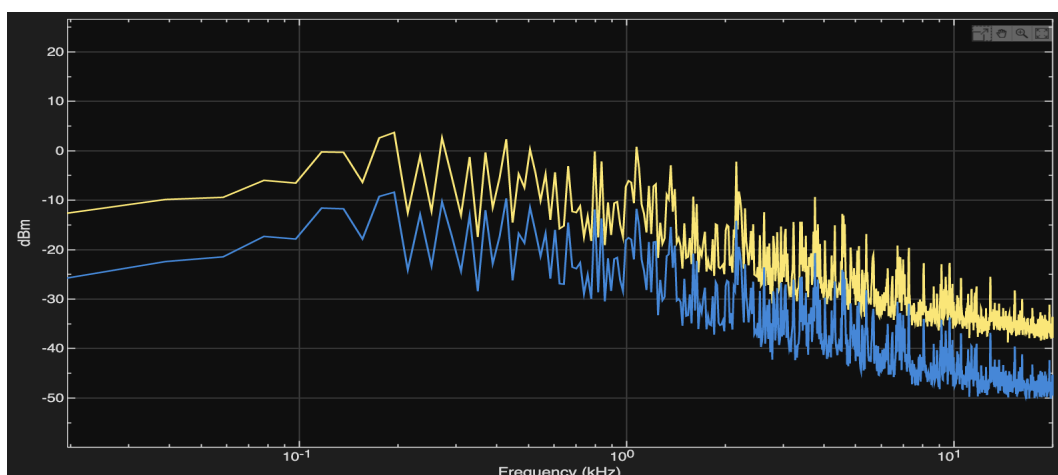
Description of Design: Using WAV file as input to the simulation.

The multi-media simulink block is used, in which the WAV file location on the computer is inserted onto the parameters, with an infinite number of plays, and samples per audio channel being 1024. Additionally the data type used on the output is a double. On the side of the sink, a block is used to save the audio received at the sink into a WAV file. Below shows the two blocks that are being used.

Block	Parameters
	
	

Verification Test:

The spectrum analyzer for the source and sink is used to compare the frequency spectrums observed at the source and at the sink of the system. Below shows the image of the frequency spectrums.



Justifications:

Microphone & Speaker I/O	WAV File
<p>Real-time processing - The system can be tested and displayed with results in real-time, highlighting the performance of the system.</p> <p>Applicable to Real-World - The system is a better use case when considering mobile applications and communications in general.</p>	<p>Controlled & Repeatable Testing Since the files have consistency in terms of what the file consists of, it is better for evaluating a system's functionality. It also allows for a greater quantity of tests to be performed with a high level of accuracy in terms of results.</p> <p>Simplistic Design - Simulink blocks enable simple implementation of the WAV files.</p>
<p>Variability - The environment around the microphone which contributes to the noise is inconsistent, and therefore an accurate test of the system would be difficult to perform.</p> <p>Complexity - When considering application to FPGA, it is difficult to implement especially when utilizing a stronger filtration system for the noise.</p>	<p>Limited Real-time ability and applications It is difficult to show the real-time capability of the system. The simulation would also not capture the real-world challenges of communication.</p>

A/D & D/A Converters

Description of Design:

ADC: The analog to digital converter module makes use of a downsampler block set at a factor of 2, followed by a quantizer which follows a quantization interval of 0.0625. After the quantizer, the uniform encoder is used to convert the double format data input into unsigned integers. Lastly, the integer to bit converter is used to convert the integers into binary bits which would further be used in processing.

DAC: The digital to analog converter module makes use of a bit to integer converter, that converts the binary bits into integers for sink to output an audio. The integer is then inputted into a uniform decoder, which is used to convert the integer format back to a double data type, which is then outputted through a speaker block within simulink, and also stored into a file on the computer.

Justifications:

Sampling Rate: Based on Nyquist Theorem, the sampling rate must at least be twice of the system bandwidth. Given scenario B, the bandwidth is given at 8kHz. Therefore the sampling rate value must at least be 16kHz. To achieve this, a down-sampler should be used with a factor of 2, to optimally perform downsampling for inputs with 48kHz frequency.

Quantization: The quantization interval value is 0.0625, since it is a 32 bit input, and 2 to the power of 5 represents 32. The quantization value or the quantization interval value is computed through the formula shown below:

$$\text{Quantization Interval} = \frac{1}{2^{(N-1)}}$$

Block	Parameters	Block	Parameters
Downsampler		Integer to Bit Converter	
Quantizer		Bit to Integer Converter	
Uniform Encoder		Uniform Decoder	

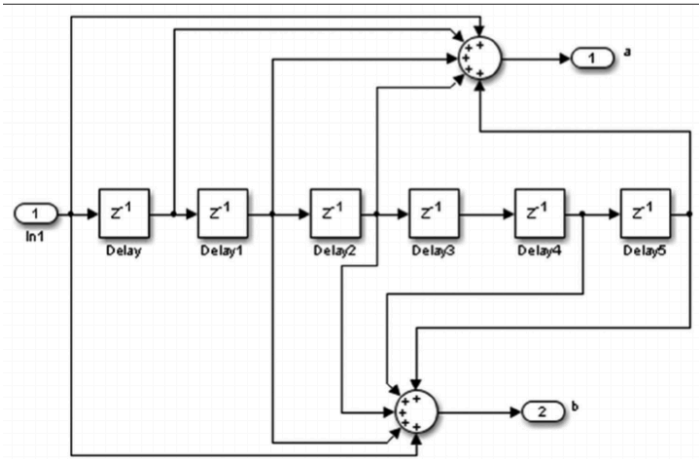
Verification Tests:

In the case of the ADC and the DAC, the Bit Error Rate (BER) is used as a way of testing whether the sub-system works. The transmitted signal is taken from the input, whilst the received signal is the output of the sink. The results are shown below. The lower the SNR, the higher the BER, because the noise is more prominent in the case of low SNR. In the case of the higher SNR value, the noise is less prominent, and therefore it has a significantly lower BER.

9 dB	100 dB
0.08117	0
4239	0
5.222e+04	1.441e+06

Error Correction Encoder/Decoders

Description of Design: Convolutional encoding-decoding for signal error correction. The convolutional encoder on the sender side takes an unsigned 16-bit input from the output of the ADC, and is configured as an encoder following the trellis structure: `poly2trellis(7,[171 133])`. This encoder is paired with a Viterbi decoder on the receiver side, following the same trellis structure and outputting an unsigned 16-bit output to the DAC module.

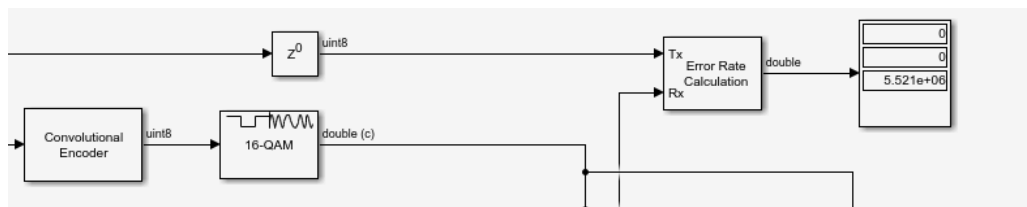


Block	Parameters
	<div>Convolutional Encoder (mask) (link)</div> <div>Convolutionally encode binary data. Use the <code>poly2trellis</code> function to create a trellis using the constraint length, code generator (octal) and feedback connection (octal).</div> <div>Select the "Terminate trellis by appending bits" operation mode to terminate the trellis at the all-zero state by appending tail bits at the end of each input frame. Check the Puncture code checkbox to puncture the encoded data for all other operation modes.</div> <div>Use the <code>istrellis</code> function in MATLAB to check if a structure is a valid trellis structure.</div> <div>Parameters</div> <div>Trellis structure: <code>poly2trellis(7, [171 133])</code> struct</div> <div>Operation mode: Continuous</div> <div>Viterbi Decoder (mask) (link)</div> <div>Use the Viterbi algorithm to decode convolutionally encoded input data. Use the <code>poly2trellis</code> function to create a trellis using the constraint length, code generator (octal) and feedback connection (octal).</div> <div>Main Data Types</div> <div>Encoded data parameters</div> <div>Trellis structure: <code>poly2trellis(7, [171 133])</code> struct</div> <div><input type="checkbox"/> Punctured code</div> <div><input type="checkbox"/> Enable erasures input port</div> <div>Branch metric computation parameters</div> <div>Decision type: Hard decision</div> <div><input type="checkbox"/> Error if quantized input values are out of range</div> <div>Traceback decoding parameters</div> <div>Traceback depth: 34</div> <div>Operation mode: Continuous</div>

Justification

Convolutional Code	Linear Block Code
<p>Better for correcting random errors: Due to convolution coders' memory and output dependent on current and previous values, convolution code is better at correcting random bit errors, such as those that occur in the AWGN channel</p>	<p>Better for correcting burst errors: Block coders are better at correcting burst errors of consecutive bits</p> <p>Lower implementation complexity: Block encoders are combinational circuits, so they are easier to construct and do not have any timing delays as they are memoryless</p>
<p>Higher implementation complexity: Due to the state machine implementations of both the convolution encoder and Viterbi decoder, a delay is introduced in the hardware implementation.</p>	<p>Worse at correcting random errors: Due to its lack of memory, it is unable to 'smooth out' random bit errors that would occur in the AWGN channel. A higher rate block encoder would be needed.</p> <p>Implementation complexity increases quickly with higher-rate encoders Many more gates would be required in a higher-rate block encoder to be able to correct larger numbers of errors over time</p>

Verification Test:

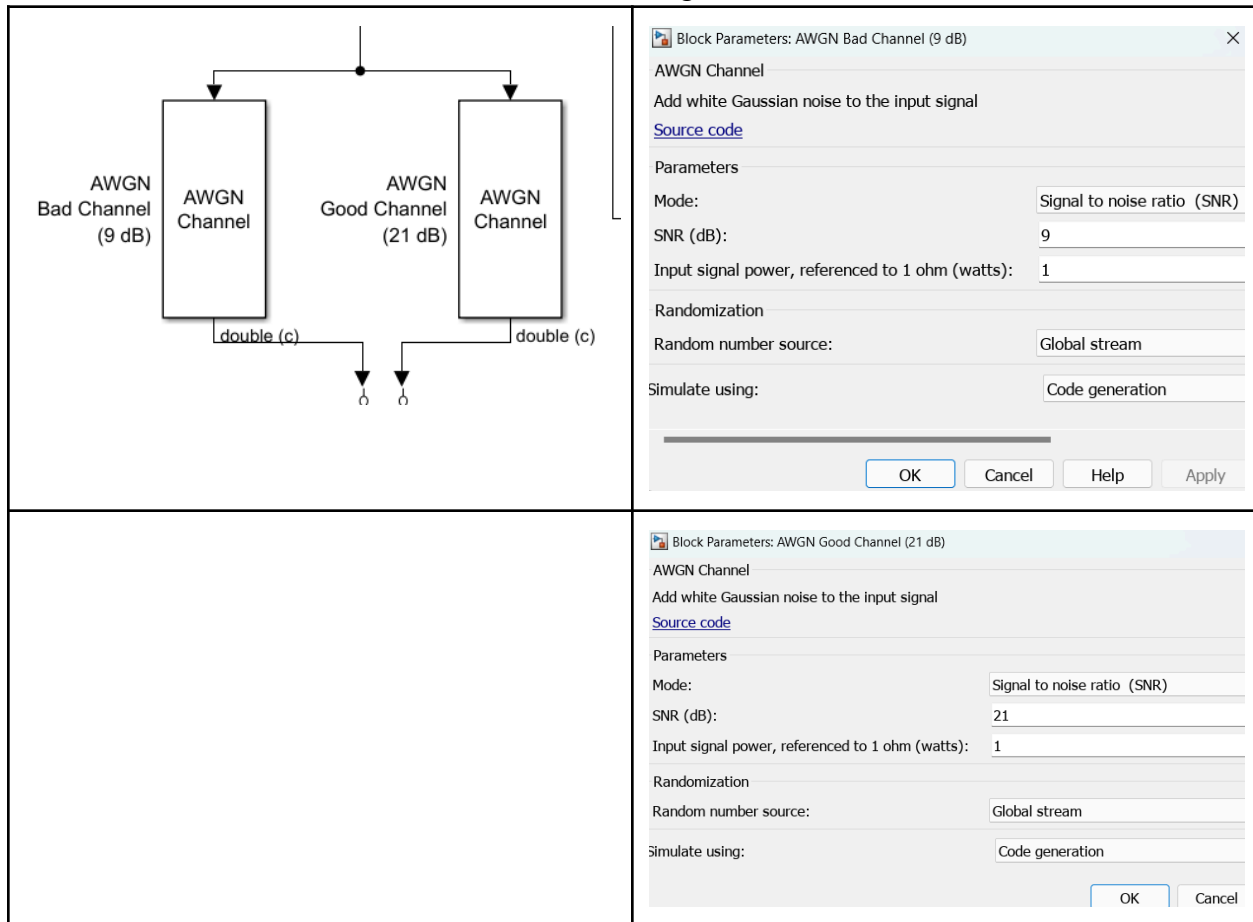


As shown in the above image, the convolutional encoder (connected to modulator setup) on its own achieves a BER of 0, when the modulation components are connected directly to the AWGN channel without the use of filters. In this case, it achieved 0 BER in both the 9dB (bad channel) and 21dB (good channel) cases, showing that it is an effective method of encoding/decoding.

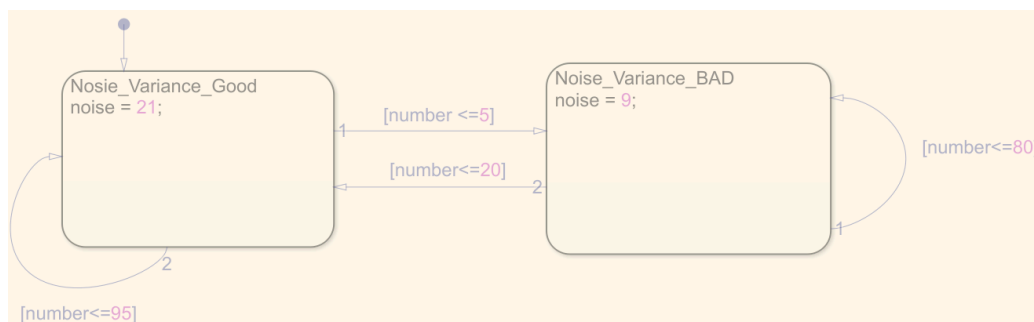
Channel

Description:

Given the Gilbert Fading Model which has γ parameters. We made two AWGN channels, one stands for bad channel with 9dB noise ratio, and the good channel is 21dB noise ratio.



FPGA schematic:



In HDL, it is possible to create a random integer generator, the output of a random integer could be connected to the state machine shown above, the states will represent which channel to use under these conditional probabilities.

Justification:

The Gilbert fading model is adept at simulating bursty error patterns, which are characteristic of wireless communication channels. In many real-world scenarios, errors do not occur independently but rather in bursts due to factors such as multipath propagation and shadowing. The Gilbert model's ability to switch between 'good' and 'bad' states captures this behavior accurately.

Verification:

$$\text{SNR}_{\text{linear}} = 10^{\left(\frac{\text{SNR}_{\text{dB}}}{10}\right)}$$

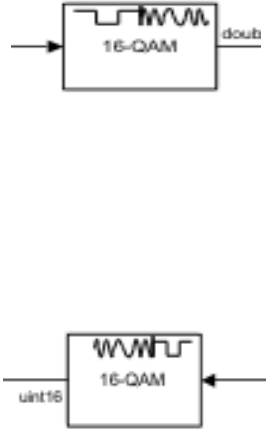
$$\sigma^2 = \frac{P_s}{\text{SNR}_{\text{linear}}}$$

$$\text{Variance}(\text{Good}) = 10^{-(21/10)} = 0.00794$$

$$\text{Variance}(\text{Bad}) = 10^{-(9/10)} = 0.1259$$

Modulators/Demodulators

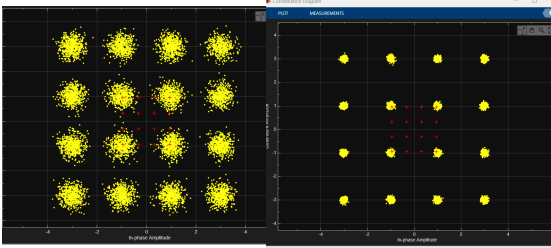
Description of Design: 16-QAM (quadrature amplitude modulation) modulator/demodulator components, the former’s input connected to the output of the sender’s convolution encoder, and the latter’s output connected to the input of the receiver’s Viterbi decoder. This maps the coded signal to one of 16 states, allowing for both amplitude and phase modulation.

Block	Parameters
	<div><div>MainData TypesParameters</div><div>Parameters</div><div>M-ary number: 16</div><div>Input type: Bit</div><div>Constellation ordering: Gray</div><div>Normalization method: Min. distance between symbols</div><div>Minimum distance: 2</div><div>Phase offset (rad): 0</div><div>View Constellation</div></div> <div><div>MainData TypesParameters</div><div>Parameters</div><div>M-ary number: 16</div><div>Output type: Bit</div><div>Decision type: Hard decision</div><div>Constellation ordering: Gray</div><div>Normalization method: Min. distance between symbols</div><div>Minimum distance: 2</div><div>Phase offset (rad): 0</div></div>

Justification

16-QAM modulation can support high information throughput in a given period of time (faster information transfer, more efficient use of signal bandwidth). Currently, this is the only form of modulation we have managed to get working with convolution coding in Simulink, and since modulation is a task that can be done with a combinational circuit, implementation on the FPGA should be relatively straightforward if not more complex than lower-order modulation methods like QPSK.

Verification Test:



9 dB channel

21 dB channel

As with the above verification for convolution coding, the setup with 16-QAM modulation and no other components achieved 0 BER at both 9dB and 21dB channels, showing that this form of modulation paired with convolution coding is effective. Seen to the left, the constellation map shows that at a lower (9dB) SNR, the mapped points are more spread out than at a higher (21dB) SNR, due to greater influence of noise in the former and less influence in the latter. However, the 16-QAM pattern is easily observable in both cases, showing that the error rate is still low.

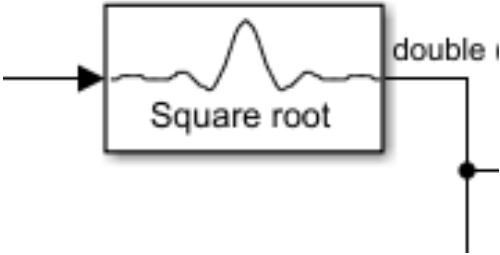
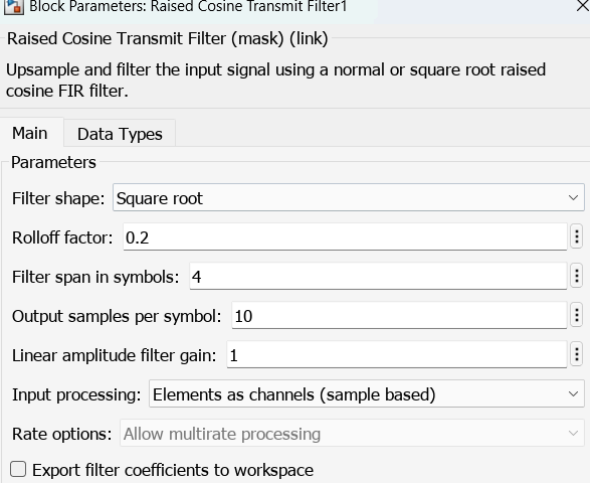
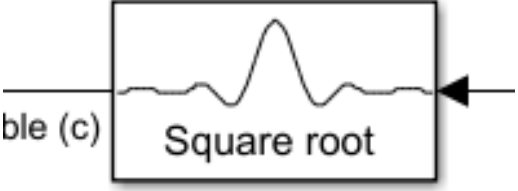
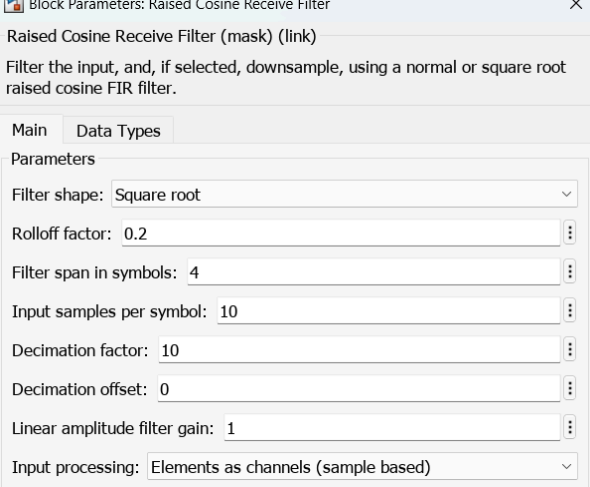
Transmitter/Receiver

Description of Transmitter/Receiver:

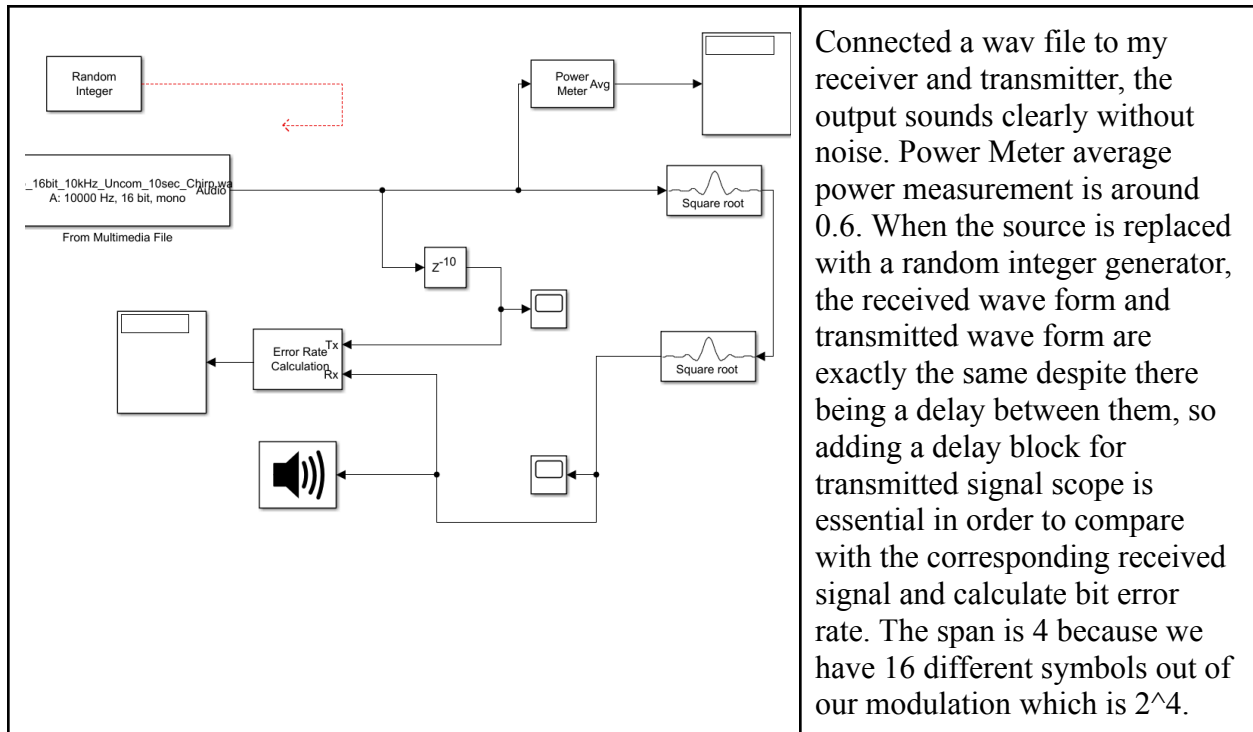
We have chosen to use the root raised cosine (RRC) filter for both our transmitter and receiver in our communication system. This decision was made due to the numerous advantages that the RRC filter provides, ensuring efficient and reliable data transmission.

1. Bandwidth Efficiency
2. ISI Mitigation
3. Flexibility and Control

Square wave in frequency domain and Sinc function in time domain which leads to less inner symbol interfaces. Parameters such as rolloff factor and span in symbols can be used to manipulate impulse shape and appropriate symbol span representation.

Justifications:



References

1. ELEC 391 Project Document - Found through Canvas
2. ELEC 391 Assessment Demo 1 Document - Found through Canvas
3. MATLAB Simulink Online Resources