ELEC 391 FINAL PROJECT

Team 7
Bobby Smith, Harnoor Saigal, Saksham Mahajan

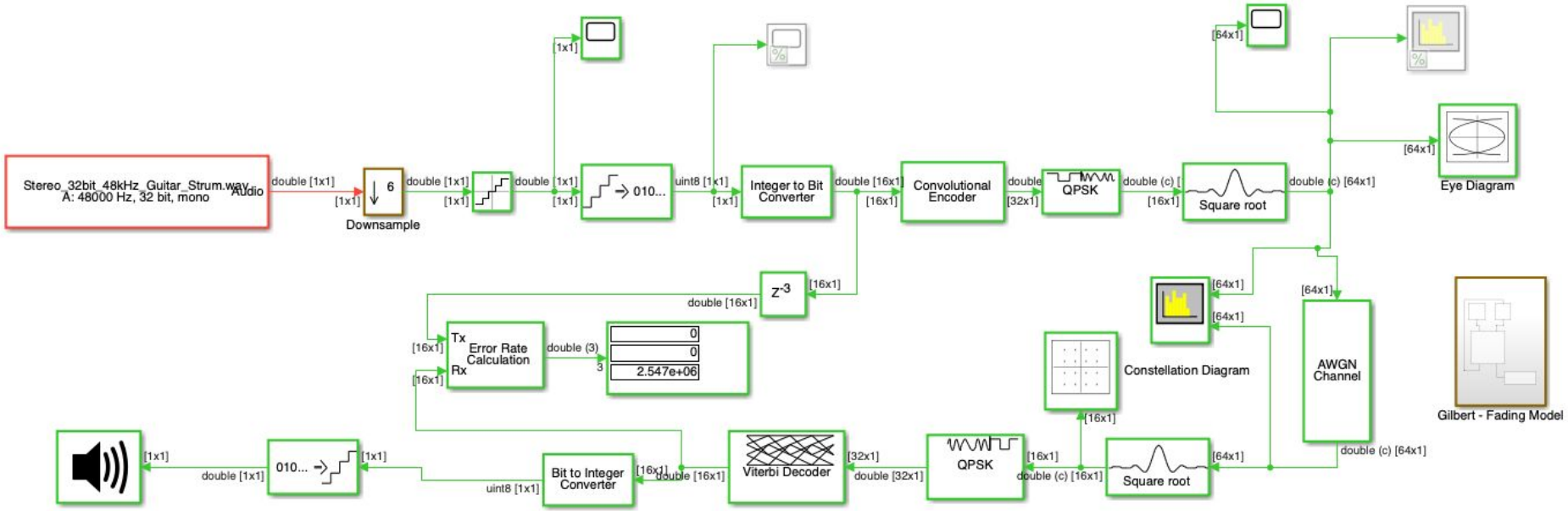
Introduction

- **Primary objective:** To design and implement a reliable digital communication system for audio and data communication.
- **Conditions, requirements, and constraints:**

Audio BW(kHz)	Target BER	Spectral Mask (kHz)	Target Channel	Delay (ms)
4	10^{-5}	100	δ	25

- **Main Focus:** on achieving high fidelity and low bit error rate (BER) while operating within the specified bandwidth and processing delay constraints.

System Overview - Simulink

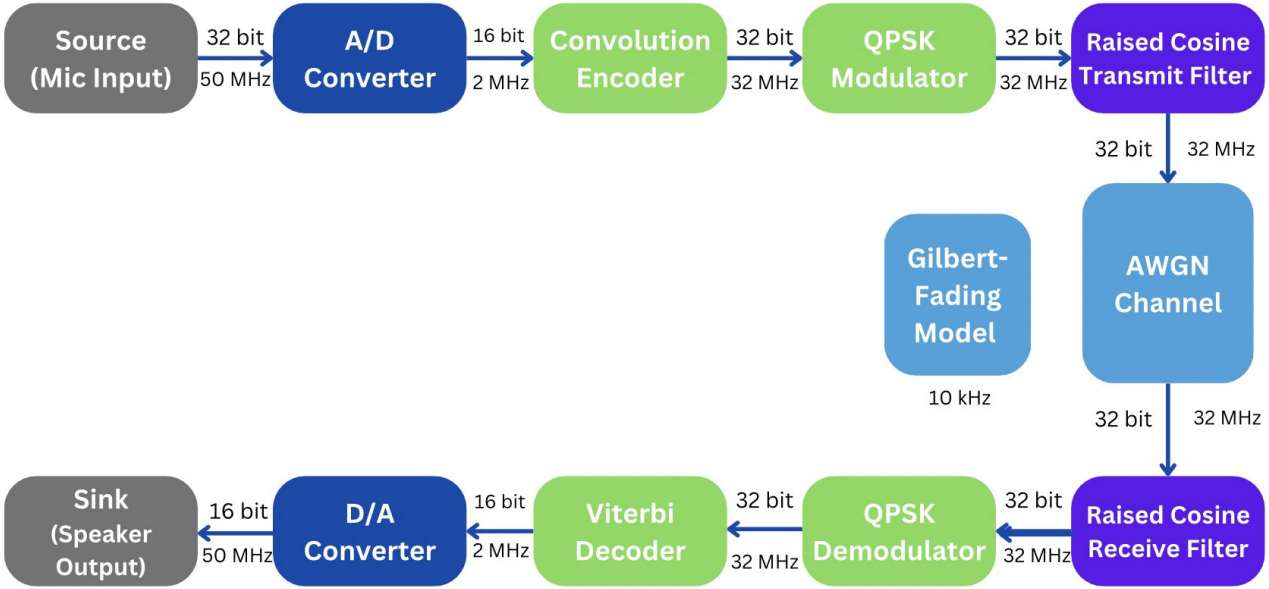


Simulink Model

System Overview - Simulink

- **Source/Sink:** System involved an audio input file of 32 bits, 48 kHz.
- **ADC:** The input is then changed from analog to digital signal using the Audio CODEC, and is downsampled and then quantized.
- **Encoding and Modulation:** The output of the ADC goes into the convolutional encoder that doubles the number of bits and is modulated using QPSK.
- **The symbols are shaped and normalized through the raised cosine filter and go through the AWGN channel.**
- The Gilbert fading model, the channel switches between good and bad using δ target probabilities through a state flow chart.
- **The receiver raised cosine filter converts the waveform to symbols that get demodulated, and the Viterbi decoder performs error correction and produces binary outputs.**
- The **DAC** subsystem finally produces integer values that are played on the speaker.

System Overview - FPGA



Block Diagram for overall system

Performance Table

Criterion	Simulink Perf.	FPGA Perf.
Message Transmission (bit rate)	4kHz	3.81kHz
Transmission Reliability (bit error probability)	0	0
Processing Delay	129 ms	7.3ms
Channel Bandwidth	74.25kHz	88.54kHz

Table 6: Performance Scenarios

Scenario (Teams)	High Priority				Low Priority
	Audio BW (kHz)	Target BER	Spectral Mask (kHz)	Target Channel	Delay (ms)
A	4	10^{-5}	100	δ	25
B	8	10^{-4}	100	γ	25
C	20	10^{-3}	150	β	35
D	4	10^{-5}	150	δ	15
E	8	10^{-4}	200	γ	15
F	20	10^{-3}	200	β	25

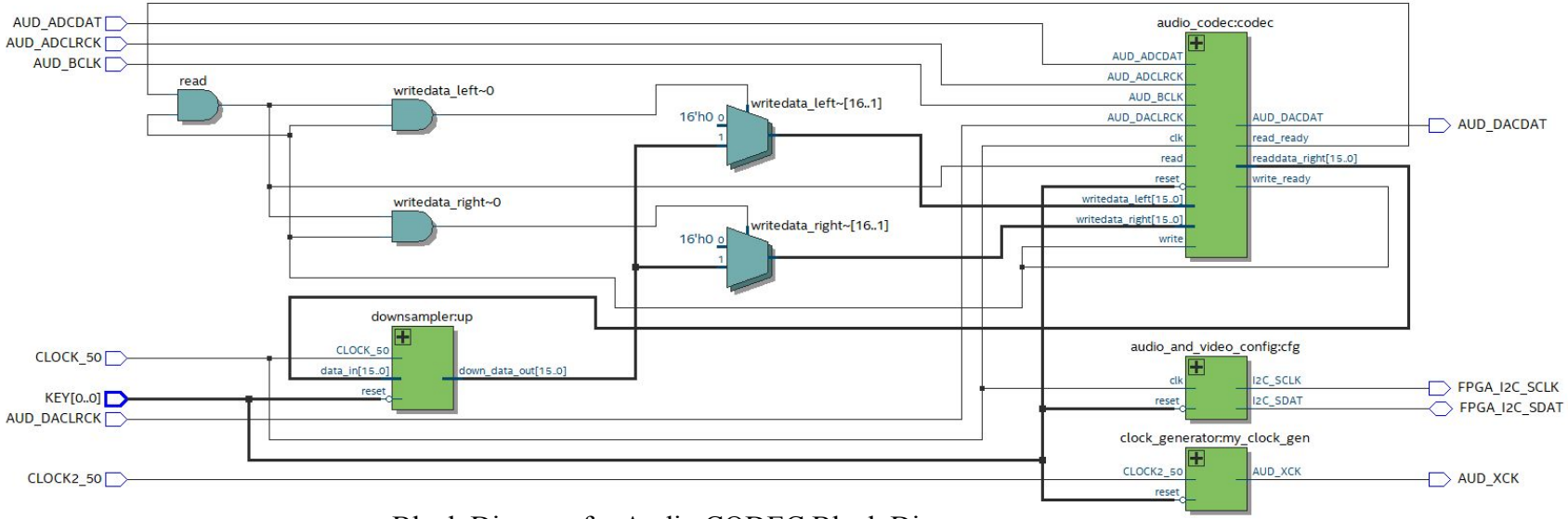
Table 1: Project Conditions

Message transmission	a) 4 kHz bandwidth b) 8 kHz bandwidth c) 20 kHz bandwidth
Reliable transmission	a) Bit error rate of 10^{-3} b) Bit error rate of 10^{-4} c) Bit error rate of 10^{-5}
Processing Delay [1] (mic to speaker)	a) 15 ms b) 25 ms c) 35 ms
Simulation tool	• Matlab / Simulink / ModelSim
Prototype	• Altera DE1-SoC FPGA board
Average transmit power	• 1 W
Channel bandwidth (See Figure 1)	a) 100 kHz spectral mask b) 150 kHz spectral mask c) 200 kHz spectral mask
Audio source	• Audio file in a WAV format with 32 bits per sample • DE1 microphone input • Duration 5 sec – 10 sec
Audio sink	• Audio file in WAV format based on team requirements • DE1 speaker output

Performance Scenario Tables

Validation of Subsystems

ADC and DAC



Block Diagram for Audio CODEC Block Diagram

- **Heavily Relied on Wolfson Audio CODEC:** For Analog to Digital and Digital to Analog
- The **Two Main Parameters** were: Sampling Frequency and Quantization

ADC and DAC - Sampling Frequency & Quantization

- Our Audio CODEC, does not downsample the input.
- **Sampling Frequency:** Created a Downsampler Module to sub-sample our input from 48kHz to 8kHz, we had to choose a downsampling factor of 6
- For Quantization, we just changed the parameters in the Audio CODEC.
- **Trade-off:** We could have used 32 bits for higher resolution, but to avoid our system to increase our latency and bandwidth, we chose to go ahead with 16-bit.

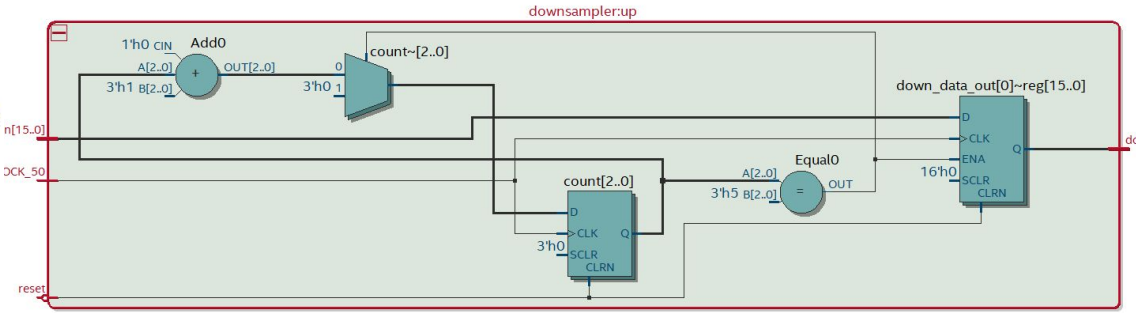
Parameter	Old Value	New Value
AUDIO_DATA_WIDTH	24	16
BIT_COUNTER_INIT	5’d23	4’d15

Quantization Parameters

ADC and DAC - Downsampling

```
module downsampler(
input CLOCK_50,
input wire reset,
input wire [15:0] data_in,
output reg [15:0] down_data_out
);

reg [2:0] count;

always@(posedge CLOCK_50 or posedge reset) begin
if(reset) begin
count<=3'b0;
down_data_out<=16'b0;

end else if (count == 3'b101) begin
down_data_out <= data_in;
count <=3'b0;

end else begin
count <= count + 3'b1;
down_data_out <= down_data_out;
end
end
endmodule
```

Figure : Verilog Code for downsampler



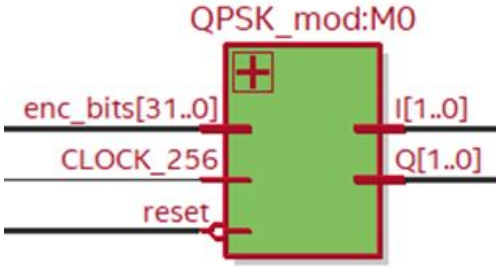
Block Diagram for downsampler

Modulation - QPSK

- It has the lowest error probability compared to other higher modulation techniques so it would reliably guarantee BER of 10^-5
- Gray coding is chosen for constellation ordering to minimize bit errors, as adjacent symbols differ by only one bit.

Trade off:

- Unlike the Simulink, for the FPGA implementation the phase offset was chosen as pi/2 to prevent phase ambiguity and not needing to use floating point values.
- Outputting 2 bits each of I and Q at every clock cycle of 32 MHz clock frequency instead of frame based to make it easier to process the bits but increases the bit rate for transmission.



Symbol Mapping	Phase shift (rad)
00	0
01	pi/2
11	pi
10	3pi/2

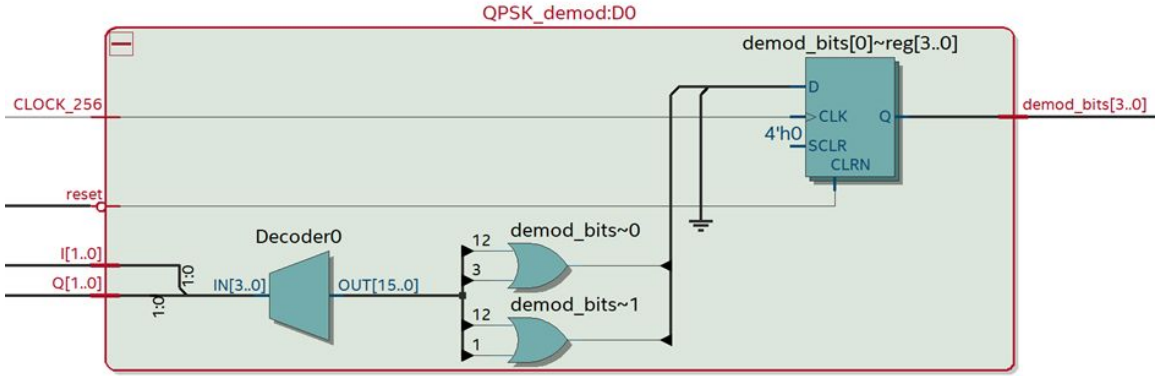
Table for Modulation parameters

Demodulation - QPSK

- At 32 MHz clock, for every clock cycle the receiver module sends I and Q as inputs with noise added from the channel when they were transmitted. The module processes 4 bits from I and Q in concatenation and maps the symbols to audio bits.

Trade off:

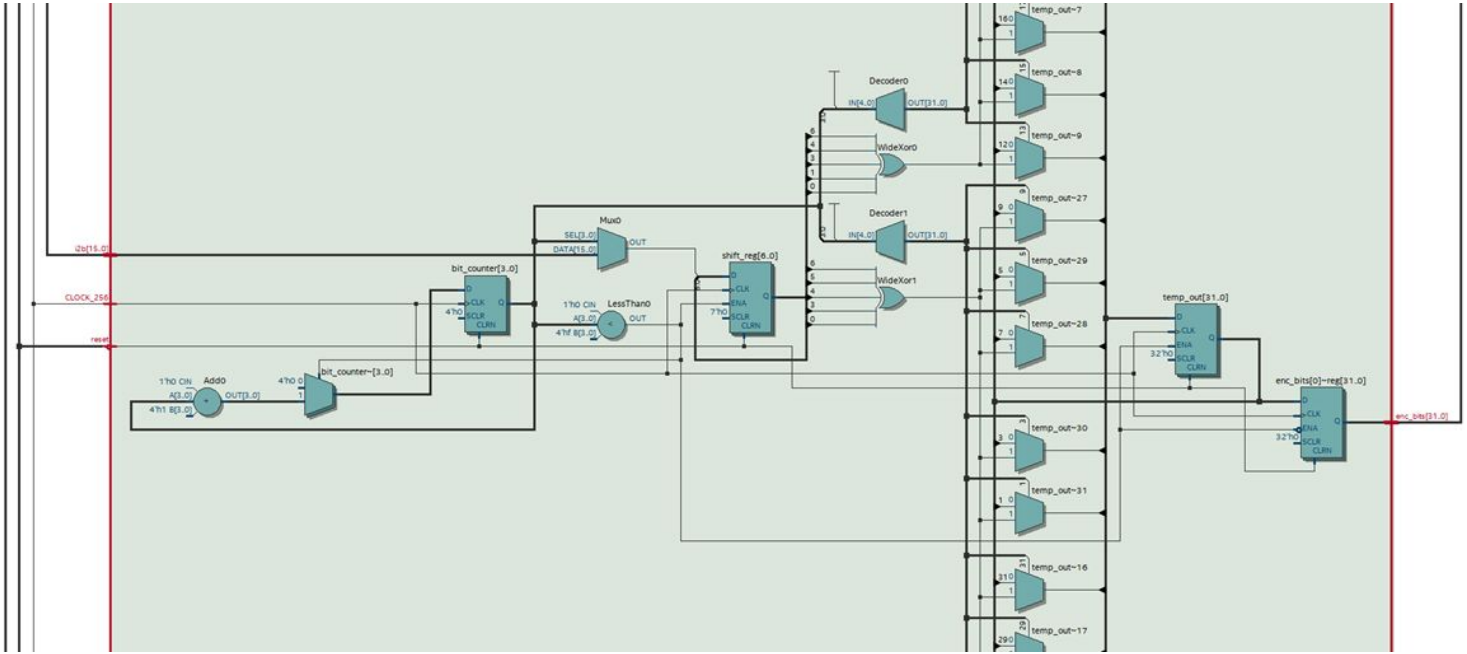
- The output demod_bits is sized to 4 because the Viterbi decoder from IP catalog takes 4 bits as input at once and outputs 1 bit because of which I had to add a buffer module that uses a counter to shift the output to a 16-bit register and then send to the DAC module.



Block Diagram for QPSK Demodulator

Encoder - Convolutional

Trade off: Computational delay

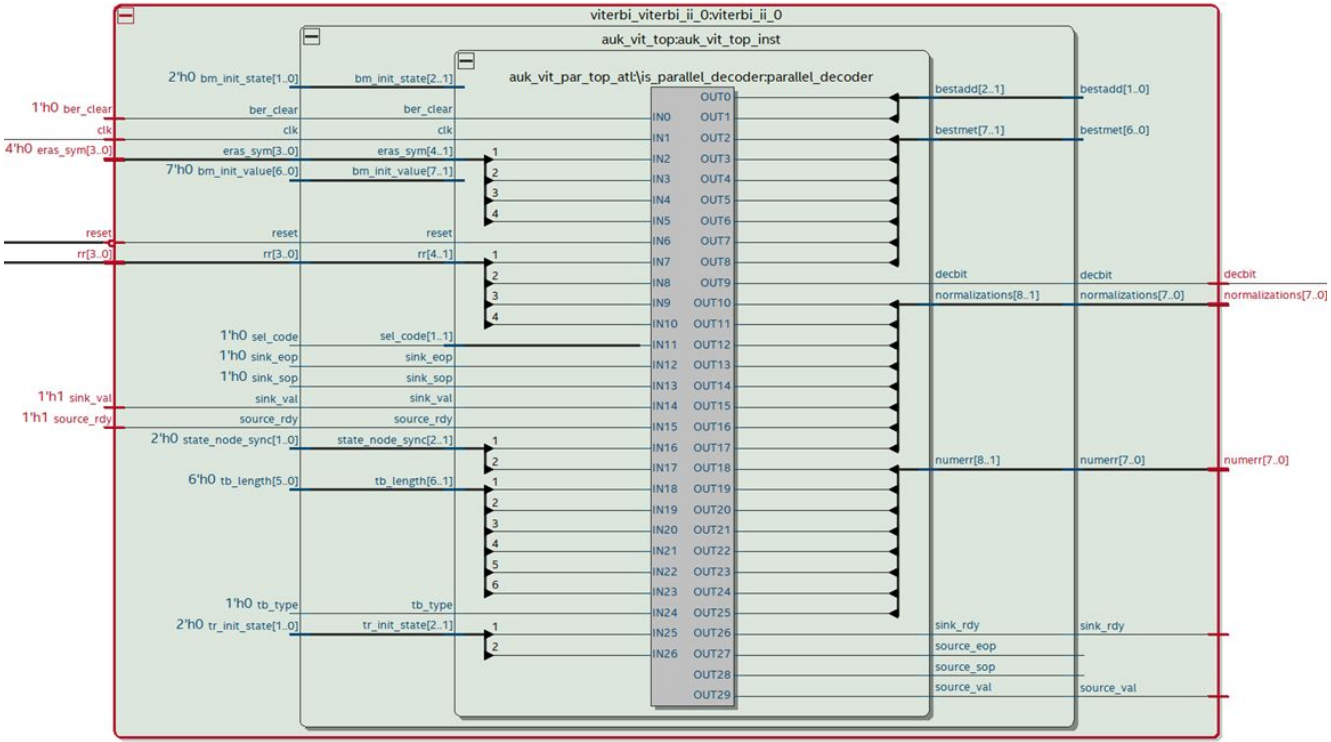


Block Diagram Convolutional Encoder

Decoder - Viterbi

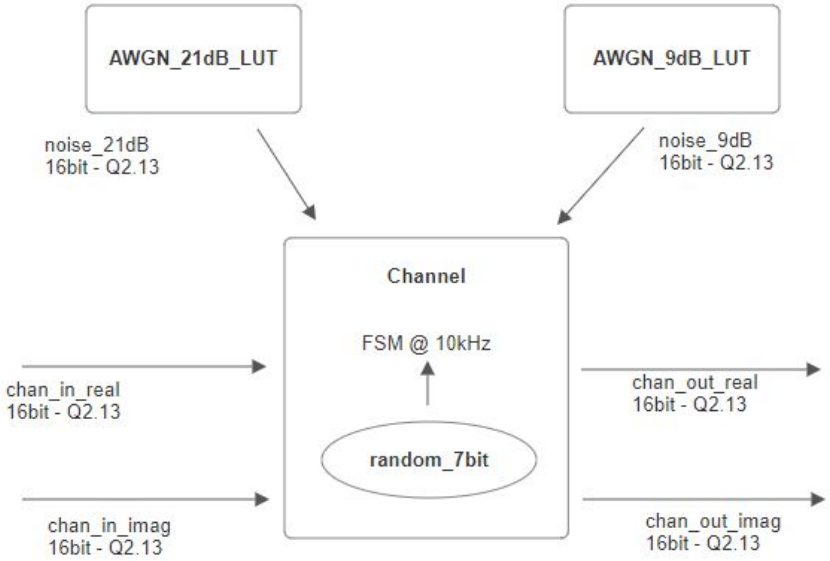
Trade off:

- Black box
- Signal description in IP core guide limited
- Every cycle only processes 4 bits



Block Diagram Viterbi

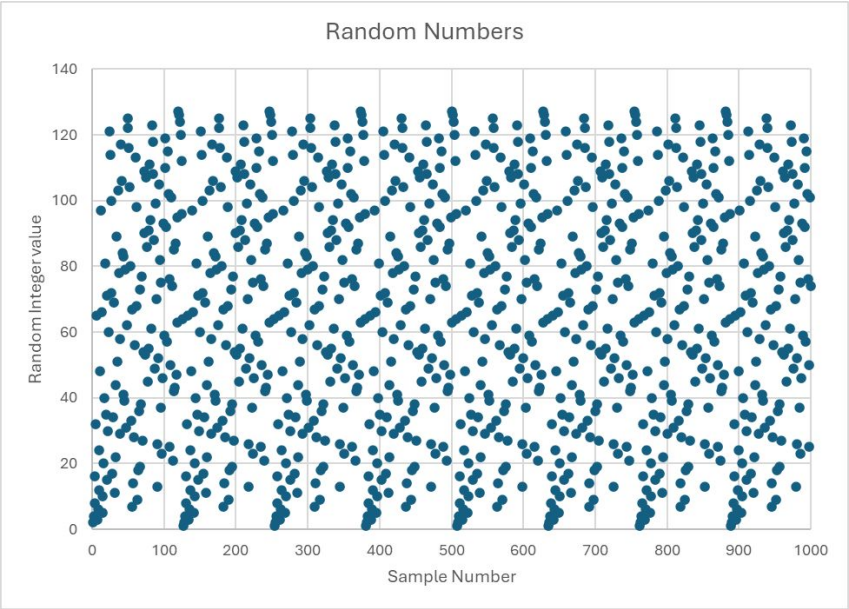
Gilbert-Fading Block Diagram

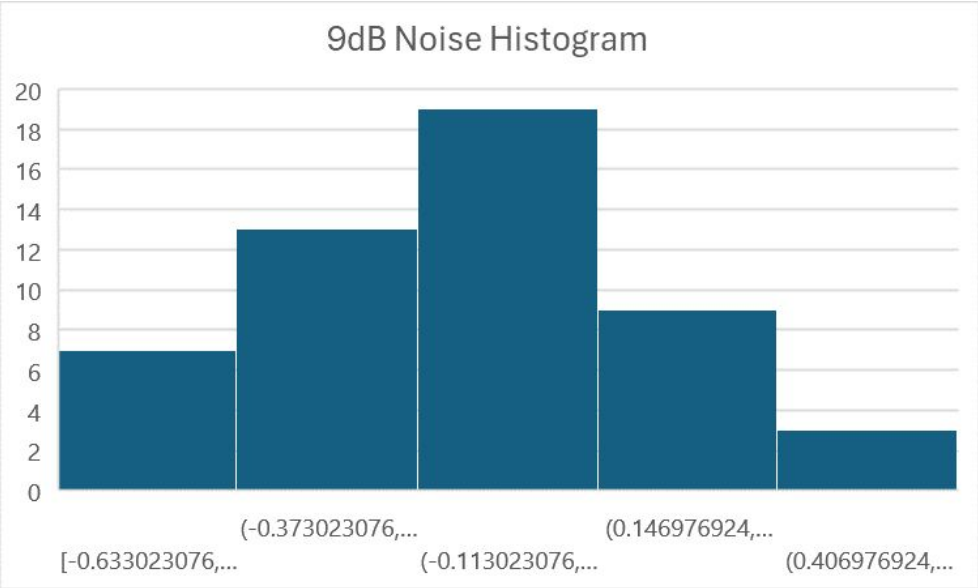


Block Diagram for Channel

Gilbert-Fading Design Decisions

- Decision to go with Q2.13 as my representation allowed me to meet the 16 bit requirement for channel quantization and also made it impossible to go outside of the +-4 range for channel amplitude. Some drawbacks of this were increased complexity in making sure my numbers (such as symbols, coefficients, noise lookup table values) were all being converted perfectly.
- Decision to go with a lookup table instead of HDL generated was based on simplicity, numbers were easy to generate by hooking up a constant 0 signal in simulink to an awgn generator then exporting results from 9 and 21dB, then converting to Q2.13 and store as signed decimals in verilog. A drawback of this is being limited to a finite number of samples, 50, reducing randomness.
- 7 bit linear shift feedback register to generate random numbers made it easy to translate the probabilities to verilog (ie. max random integer is 127, P(rand<19) = 0.15). It was relatively easy to validate as numbers could be exported and graphed in excel with minimal effort. The tradeoff here is linear shift feedback registers are only pseudo random so you do get some clustering around ranges of values before it moves into another range.



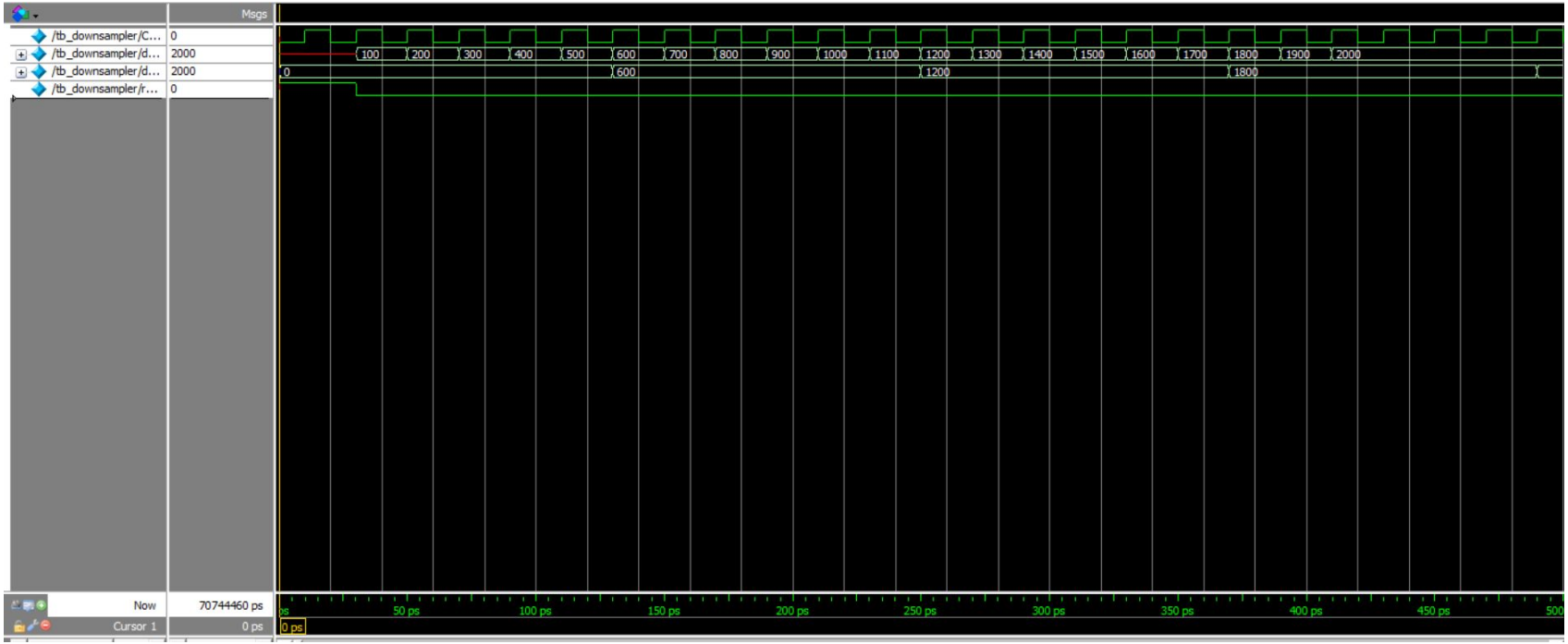


Left plot: Random numbers generated using LFSR for state switching

Right plot: Histogram of n=50 Lookup table noise generated for 9dB SNR

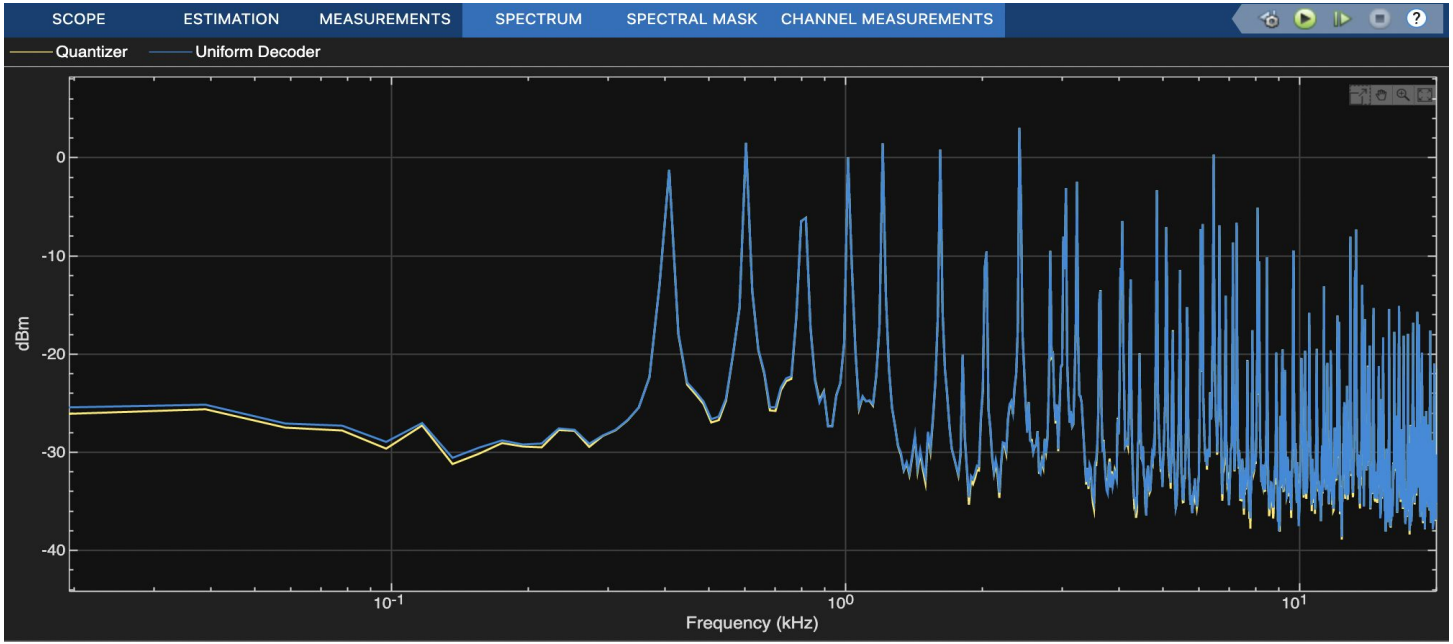
Supporting Slides

ADC DAC - Downsampler

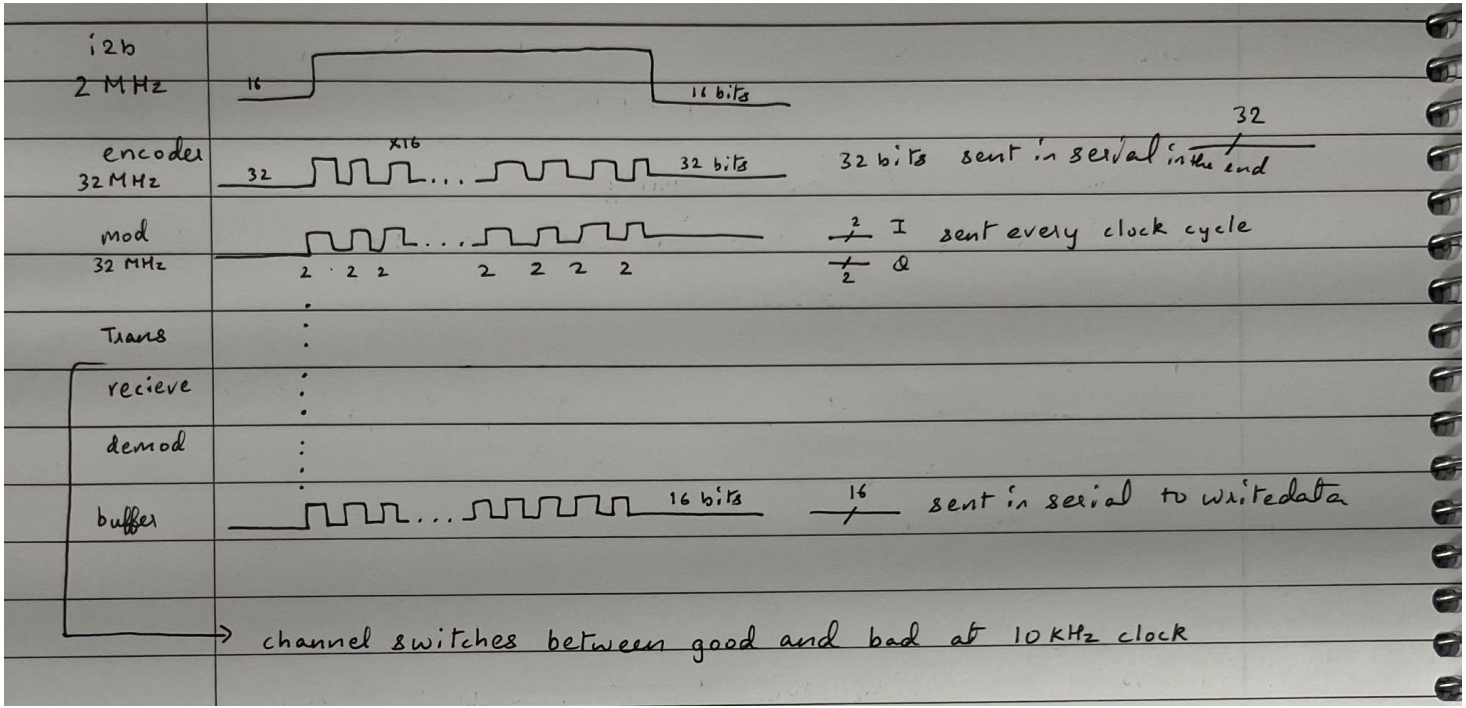


Simulated Waveform for the Downsampler Module

ADC DAC - Simulink Verrifaction



Clocks



Brief description about the three clocks in play in our system

Thank You