The George Washington University

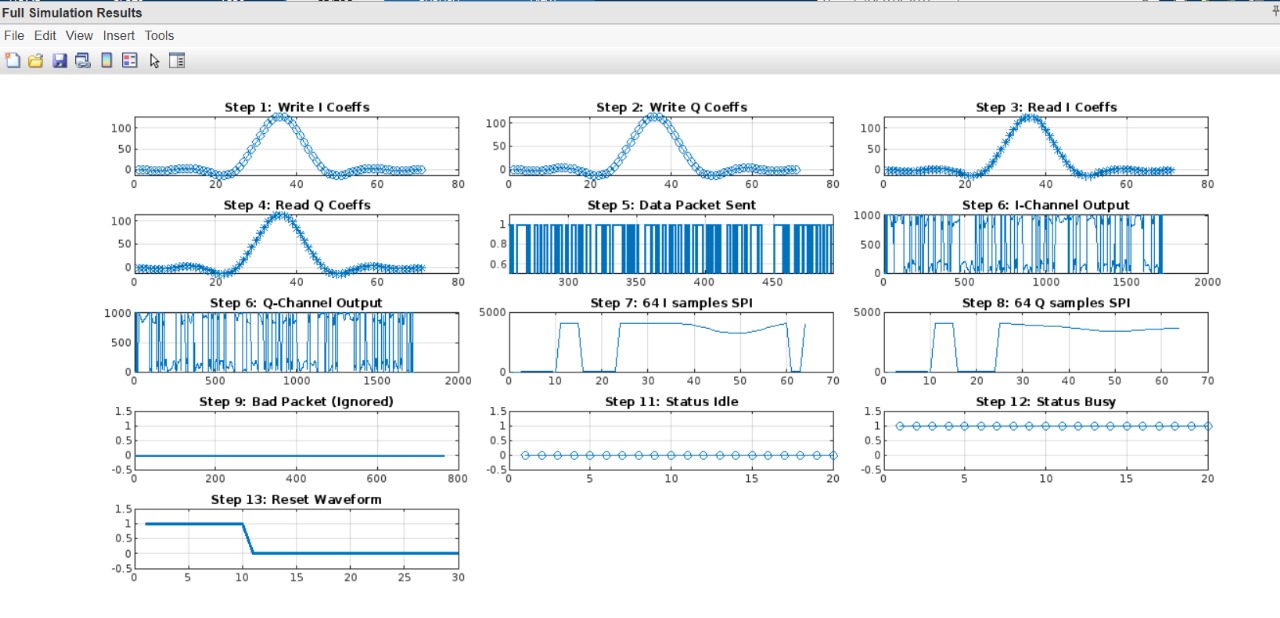
Department of Electrical and Computer Engineering

ECE 6214

Project 9: 64QAM Modulator

**Step 1:** Implement the design described in the provided 64QAM modulator datasheet.

**Step 2:** Simulate the design using the MATLAB files to help generate test stimulus and response.



1. Write all 71 I-channel filter coefficients (graph step 1)

The graph(step1) depicts the 71 I-channel filter taps that is being written.  
It exactly matches with the expected RRC shape, confirming proper tap generation.

1. Write all 71 Q-channel filter coefficients (graph step 2)

The Q-channel coefficients are like the I-channel but slightly lower in amplitude, as required.  
Graph 2 confirms correct 0.9 scaling applied during filter creation.

1. Read back and check all 71 I-channel coefficients (graph step 3)

This graph displays that all I-channel filter taps were correctly stored and retrieved without any mismatch.

1. Read back and check all 71 Q-channel coefficients (graph step 4)

This confirms that Q-channel filter taps were being correctly written to and read back from memory.

1. Send data packet (graph step 5)

The bitstream for the packet is completely visualized, displaying a correctly randomized 64-QAM data packet transmission.

1. Check all I/Q outputs to verify they are correct for the entire data packet (graph step 6)

The graph(step6) represents the filtered modulator outputs (after pulse shaping and down-scaling).  
Values are within expected range, confirming correct I/Q processing.

1. Read back and check all 64 stored I-channels values via SPI interface (graph step 7)

The reconstructed I samples are matching expected output samples correctly from SPI addresses 512–639.

1. Read back and check all 64 stored Q-channels values via SPI interface (graph step 8)

Like above, the reconstructed Q samples match expected output samples correctly from SPI addresses 768–895.

1. Send data packet with incorrect header, demonstrate baseband data packet is ignored (graph step 9)

The module ignores the invalid packet successfully, as no processing or storage occurs for the bad packet.

1. Disable mapping via SPI interface, demonstrate baseband data packet is ignored (graph step 10)

After disabling the mapping, the valid packets are also ignored, demonstrating proper control via mapping enable.

1. Read back baseband mapping status address (register address x0) when no baseband data packed is being sent, check that status is correct (graph step 11)

Mapping status reads 0 continuously, this confirms no active packet transmission and correct IDLE detection.

1. Read back baseband mapping status address (register address x0) when baseband data packed is being sent, check that status is correct (graph step 12)

Mapping status reads 1 during transmission, showing the baseband is BUSY, confirming proper packet activity detection.

1. Reset waveforms showing proper synchronization and sequencing (graph step 13)

The reset signal goes from high to low. This confirms the correct system reset behavior for initialization or error recovery.

**Step 3: Synthesize the design, fix any synthesis errors that occur. Make sure you update the constraints for clock domain crossing with 3 clock domains.**

What is the maximum SPI clock frequency?

What is the maximum data clock frequency?

What is the maximum DSP clock frequency?

What is the area?

**Step 4: Conduct back-annotated simulations**

**Symbol mapping**

