A Simple Wi-Fi PHY: an Educational Tool for LabView

Nathan Schimpf
Department of Electrical and Computer Engineering
Student of University of Louisville
Louisville, United States
ntschi01@louisville.edu

***Abstract*—One of the predominant schemes for wireless communication, Wi-Fi enables the majority of all internet traffic. As internet use increases, the need for Wi-Fi systems to provide higher throughput rates has persisted since its inception, leading to its use of Quadrature-Amplitude Modulation (QAM) and Orthogonal Frequency Division Multiplexing (OFDM) first seen in the 802.11a standard. To gain a better understanding of the core Wi-Fi physical layer (PHY), these core modulation and demodulation techniques were developed in LabView**

Keywords—Wi-Fi, OFDM, QAM, Physical Layer, PHY, LabView

I. Introduction

Modern physical-layer Wi-Fi implementations incorporate a number of modulation and coding techniques to maximize throughput and robustness. Figure 3 illustrates the overall architecture.

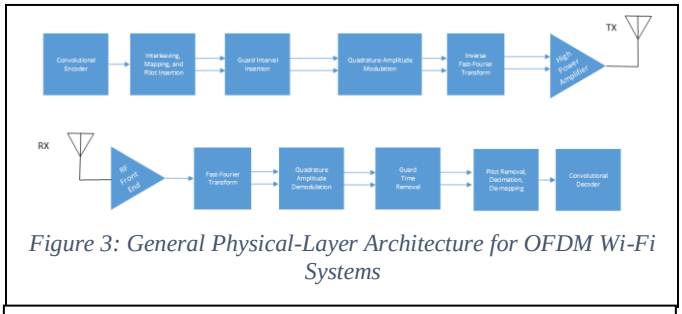

Figure 3: General Physical-Layer Architecture for OFDM Wi-Fi Systems

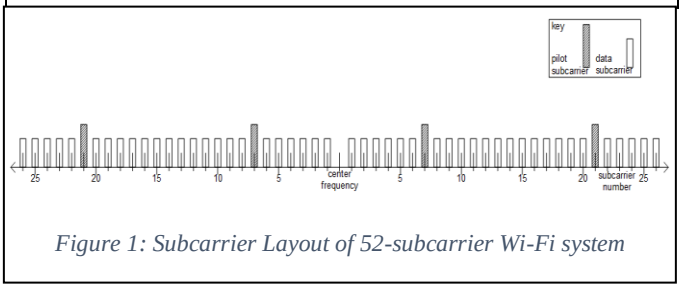

Figure 1: Subcarrier Layout of 52-subcarrier Wi-Fi system

At its core, OFDM is a specialized technique of grouping multiple subcarriers (52, for this project's scope) into a single channel. Each subcarrier sends individual data streams on a specified carrier frequency, in such a way that they are both tightly-packed and non-interfering. While most of the subcarriers present are data-bearing, there are four pilot subcarriers which provide a fixed, repeated signal for estimation processes. The layout of these subcarriers can be seen in figure 1.2.

It should be noted that the orthogonality of these subcarriers can be affected by changes in the physical space of transmission, whether that be the medium of the channel or movement of the devices; in both cases, the interference caused by this loss of orthogonality is known as fading, and is addressed via Forward-Error Correction coding and estimations of phase and frequency offsets of each subcarrier.

The transmission of bits of data by each subcarrier leverages the broad modulation scheme of Quadrature-Amplitude Modulation (QAM). In order to utilize this modulation technique, the transmitter and receiver must be synchronized, expecting a character to be transmitted at a fixed interval of time based on the period of the lowest-frequency subcarrier – the symbol rate of the channel. Synchronization is accomplished using a guard-time prefix, extending the first symbol prior to transmission.

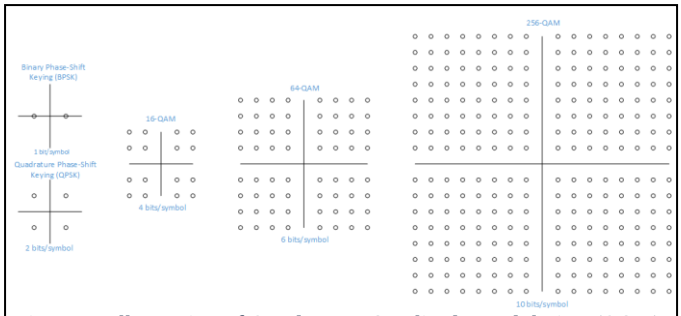

Figure 2: Illustration of Quadrature-Amplitude Modulation (QAM) techniques in Wi-Fi systems

Given this synchronization, in-phase and quadrature (90-degree lagging) components of the subcarrier are used to determine bit values. A transmitter may increase the complexity and bitrate of its subcarriers by using a larger range of component values. A graphical representation (1.3) of the supported modulation techniques is provided to better illustrate this.

By developing the core QAM and OFDM modulation and demodulation techniques, this project aims to provide a minimal-feature simulation environment for students to examine

and locally experiment with typical wireless systems. The exact features this project aimed to incorporate are described below.

The project will provide:

- Capabilities for all QAM modulation schemes in figure 1.3. This will be implemented using the RF Communications Toolbox, as well as only LabView's base components.
- Selection of the QAM implementation from a front-panel input.
- Visibility of the QAM implementation via a front-panel constellation diagram.
- OFDM on a 52-subcarrier system.
- OFDM visibility implementation via FFT waveform chart.
- A guard time for the beginning of each transmission.

II. Detailed Examples and Discussion

At the time of publication, the implementation includes a *Top VI* which allows for signal simulated noise and channel fading between the transmitter and receiver; *Top VI* implements the high-level QAM modulator (*QAM Mod*), OFDM transmitter (*OFDM TX*), OFDM receiver (*OFDM RX*), QAM demodulator (*QAM Demod*), and various helper functions, with VI hierarchy illustrated in Figure 5.

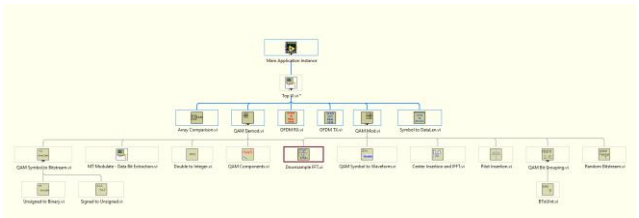

Figure 5: Project VI Hierarchy

At a cursory view, *Top VI* allows the user to specify QAM parameters (such as samples per symbol and complexity), while also providing basic analysis tools such as a frequency-domain waveform chart and constellation graph of the receiver; validation indicators are provided as well, referencing the number of correctly-received subcarriers and correctly-decoded bits. Finally, *Top VI* provides a control for selecting which QAM de/modulator is used: the Labview-provided de/modulation tools in the *RF Communications Toolbox (MT Modulate)*, or the custom-written de/modulation tools. It should be noted that Labview does not provide an OFDM implementation, which therefore must use the custom Vis.

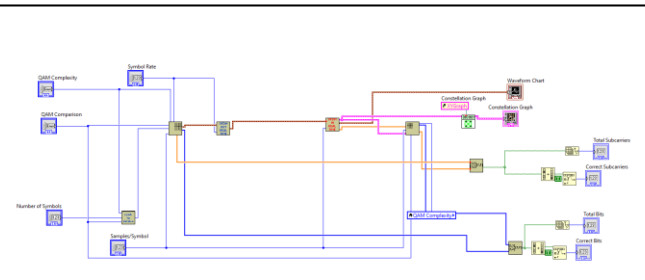

Figure 7: Top VI Code

Identify applicable funding agency here. If none, delete this text box.

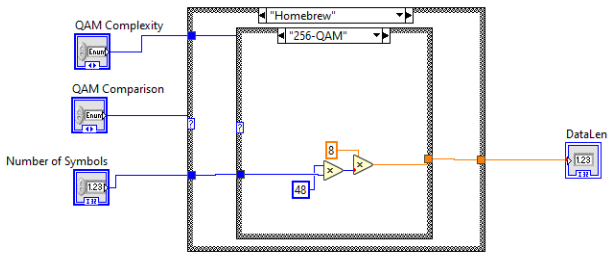

Figure 4: Symbol to Datalen VI Code

Supporting the core transmitter/receiver pair is *Symbol to DataLen*, determining the number of bits need to provide 48 data-bearing subcarriers during the QAM modulation. Though the number of bits per QAM symbol should only vary by QAM complexity in accordance with Table 1, some control was altered due to *MT Modulate* requiring a minimum of 4 symbols (2 bits) in it's symbol map. As a result, the dual-case structure in Figure 4 was implemented to scale the number of bits per symbol.

Table 1: QAM Symbol and Bit Requirements

QAM System	QAM Symbol Range	Bits required per symbol
BPSK	[-1+j0, 1+j0]	1 (homebrew) 2 (MT Modulate)
QPSK	[-1-j1, 1+j1]	2
16-QAM	[-2-j2, 2+j2]	4
64-QAM	[-4-j4, 4+j4]	6
256-QAM	[-8-j8, 8+j8]	8

A. QAM Modulation

Handling the QAM modulation is *QAM Mod*, which utilizes several subVIs to craft the individual QAM weights from a randomly-generated bitstream; insert pilot and center frequency weights according to Figure 1; and generate a waveform from weights. The custom implementation illustrates this process

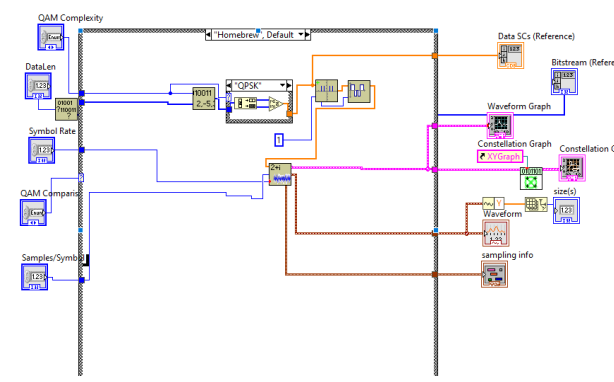

Figure 6: QAM Mod VI Code, Custom Implementation

rather symbolically in Figure 6.

More complicated is the QAM process using *MT Modulate – QAM*, which requires two key parameters: filtering coefficients, which must be generated using *MT Generate*

Filter Coefficients; and QAM parameters, including the samples per symbol and symbol listing. Providing the symbol listing was easily done with constant arrays at first. However, for 256-QAM, it became more practical to design and read from a CSV file than to generate the array in LabView; Figure 11 illustrates the *MT Modulate* implementation, including reading the symbol listing from a local CSV file.

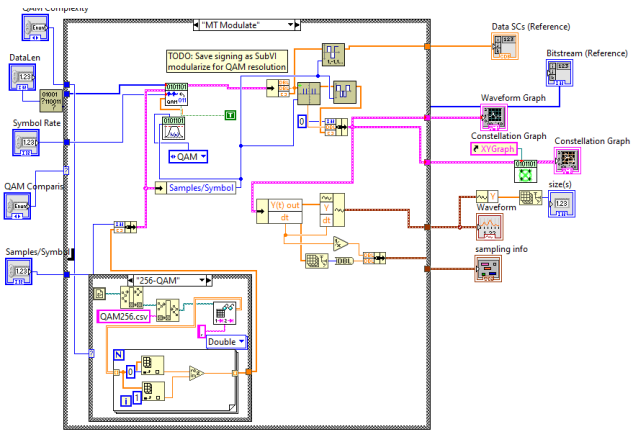

Figure 11: QAM Mod VI Code, MT - Modulate Implementation

Examining *QAM Bit Grouping* illustrates how symbol weights are generated. The VI iterates through all bits in the received bitstream and – using case structures and shift registers, converts grouped bits into unsigned integers, which are then scaled to span the appropriate QAM constellations in Figure 2. This process is exemplified in Figure 10.

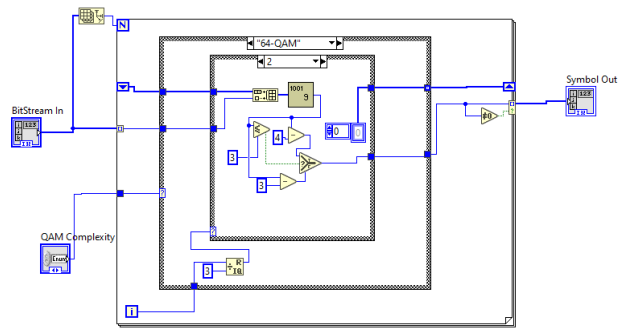

Figure 10: QAM Bit Grouping VI Code

Inserting pilot subcarriers and the center frequency are accomplished in similar manner, though with separate VIs. Each symbol fragment is iterated through (for pilot insertion, 48 subcarriers; for center insertion, 52 subcarriers), and each insertion is accomplished based on Equation 1 and Equation 2.

This operation is performed 4 times per OFDM frame for pilot insertion (Figure 9), and once per OFDM frame for center insertion (Figure 8). Because insertion may happen either before or after the QAM waveform has been generated, the equations may scale by s_{SC}: setting this as the system Samples-per-Symbol allows for insertion after waveform generation, or setting this as 1 allows for use before generation.

$$i_{array} = \left(\left\lfloor \frac{n_{SC}}{2} \right\rfloor + i_{SC}\right) * s_{SC} \qquad \textit{Equation 1}$$

$$w_i = w_{SC} * s_{SC} \qquad \textit{Equation 2}$$

I_{array}: insertaion index of array
N_{SC}: current total of subcarriers
S_{SC}: samples per subcarrier
W_i: width of inserted subarray
W_{SC}: width in terms of subcarriers

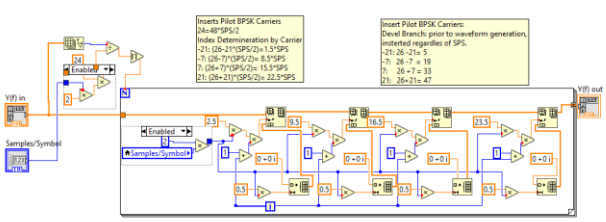

Figure 9: Pilot Insertion VI Code

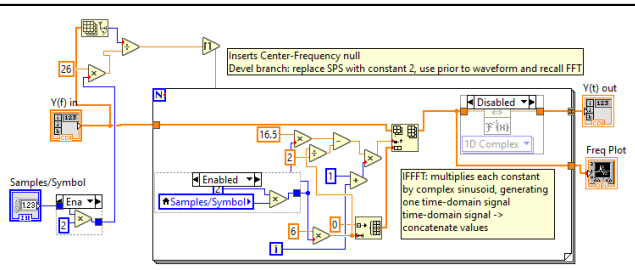

Figure 8: Center Insertion VI Code

For the custom QAM implementation, *QAM Waveform Generation* is a necessary component. In effect, it elongates the upsamples and duplicates each value from the symbol listing according to the number of samples per symbol; this is verified via modulo operation and illustrated in Figure 12.

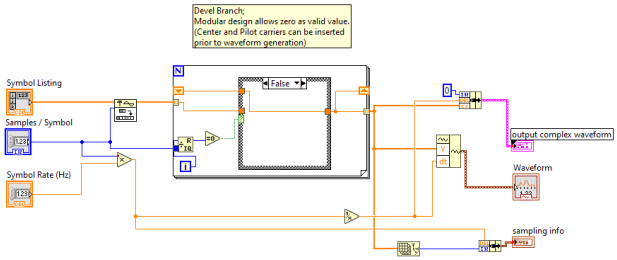

Figure 12: QAM Waveform Generation VI Code

When running *QAM Mod*, the custom implementation performs as expected, providing a series of square pulses representing each frequency-domain subcarrier at each complexity. As seen in Figure 13, the pilot and center insertions appear to work properly as well. By all appearances, the data produced is identical to that with *MT Modulate – QAM* (Figure 16).

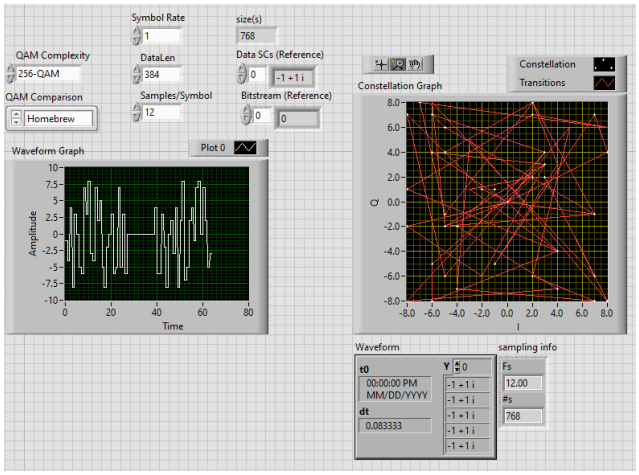

Figure 13: QAM Mod Execution, Custom Implementation

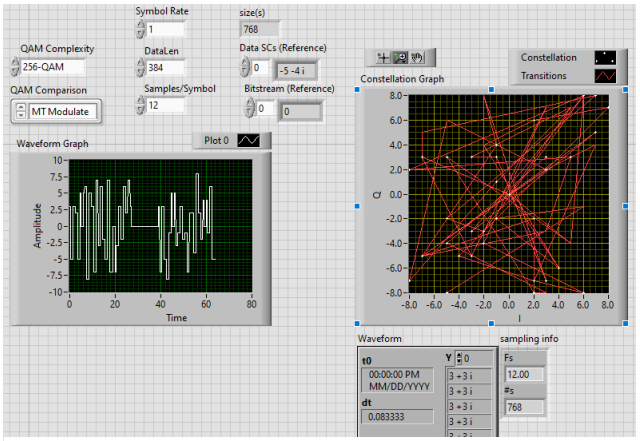

Figure 16: QAM Mod, MT - Modulate Implementation

B. *OFDM Transmit*

Though initially more complicated, *OFDM TX* effectively became an Inverse Fast-Fourier Transform (IFFT) with some scaling and documentation in the waveform cluster output. The orthogonality principle of OFDM is achieved by aligning subcarriers at integer-multiple frequencies of one another; as seen in Figure 15, the only other operation is redefining waveform increments (dt). The increment is redefined by Equation 3, a ratio of the period of the symbol rate R_S (the slowest frequency in contained by the OFDM frame, and therefore duration of the frame) and total samples in the IFFT of the signal.

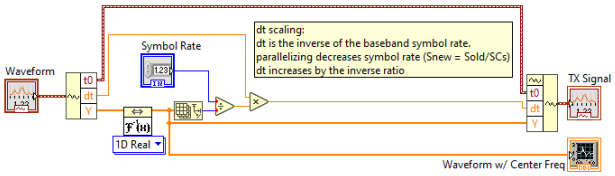

Figure 15: OFDM TX VI Code

$$dt = \frac{1/R_S}{n_{samples}} \qquad \textit{Equation 3}$$

R_s: symbol rate
$n_{samples}$: total number of samples

Initial implementation efforts achieved this by manually generating each subcarrier based on independent QAM waveforms; this approach was quickly abandoned due to the decentralization of subcarrier data (providing no central graph to observe each OFDM frame), as well as runtime inefficiency (generating 48 times the number of intended OFDM frames, which required several minutes and approximately 4 GB of program memory).

After further discussion and research, the formula for an IFFT was recalled as

$$x(t) = \frac{1}{N}\sum_{k=0}^{N-1} x(f)\, e^{\frac{j2\pi kt}{N}} \qquad \textit{Equation 4}$$

N: total frequency samples
x(f): frequency-domain signal

which effectively sums the received frequency-domain signal as an average of complex sinusoids with frequency k/N, where k is the present sample of a subcarrier, and N is the total number of samples. For simplicity's sake, this implementation was used; the initial implementation has been preserved under "A-Simple-WiFi-PHY\OFDM\DO NOT USE\," though will not be further analyzed.

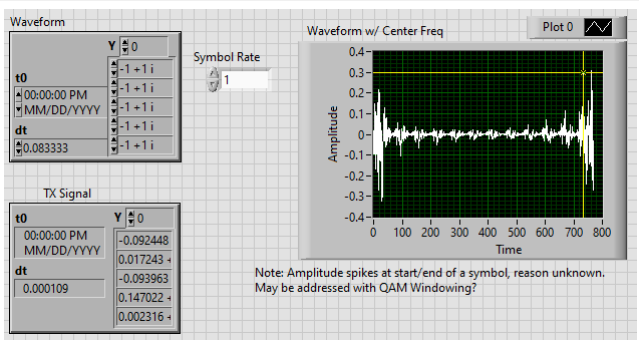

Figure 14: OFDM TX Execution

When executed, *OFDM TX* provides an accurate time-domain signal that combines each symbol within the OFDM frame, pictured in Figure 14.

C. OFDM Receive

Given the realizations from the OFDM Transmit VI, *OFDM RX* was implemented as a Fast-Fourier Transform (FFT) with time scaling defined by Equation 5. This implementation is seen in Figure 18.

$$df = dt * n_{samples} \qquad \text{Equation 5}$$

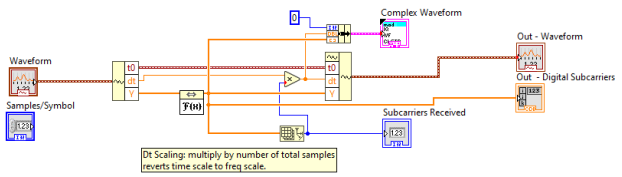

Figure 18: OFDM RX VI Code

When executed in the *Top VI* system, the frequency-domain waveform is accurately reproduced with marginal offsets; these offsets are more visible from the output constellation chart. When analyzed on a BPSK transmission (Figure 19), the imaginary and real component are seen to drift by ±1 fV, likely due to differences in execution time.

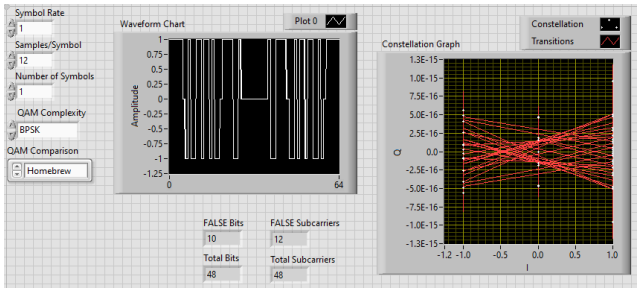

Figure 19: OFDM RX Execution on BPSK Signal

D. QAM Demodulation

QAM Demod handles all signal processing to take the frequency-based QAM waveform, extract symbols from each subcarrier in the waveform, and convert these (data-bearing) symbols to a bitstream; the VI provides this final bitstream, as well as the extracted subcarrier symbols for comparison.

When executing *QAM Demod* based on the *RF Communications* Toolbox Vis, the entire process is integrated into *MT Demodulate – QAM*. As in *QAM Mod*, the VI requires a list of filter coefficients, samples per symbol, and symbol listing be provided; this is accomplished by the same methods, depicted in Figure 22. To output only data-bearing bits, an additional VI, *MT Modulate – Data Bit Extraction*, was developed to selectively remove pilot and center-frequency bits from the bitstream; this is implemented as in Figure 17.

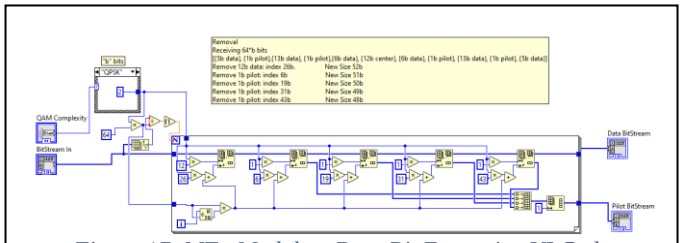

Figure 17: MT - Modulate Data Bit Extraction VI Code

The signal chain for providing data-bearing subcarriers for the *RF Communications* Toolbox implementation is almost identical to the custom implementation for bitstream recovery. As seen in Figure 23, the waveform is first decimated (*FFT Downsample*, Figure 21), which is then split into pilot and data subcarriers (*QAM Components*, Figure 24) before being iteratively rounded (*Double to Integer*, Figure 27) and converted to unsigned integers (*Singed to Unsigned*, Figure 26), then to the output bitstream (*QAM Symbol to BitStream*, Figure 20; *Unsigned to Binary*, Figure 28).

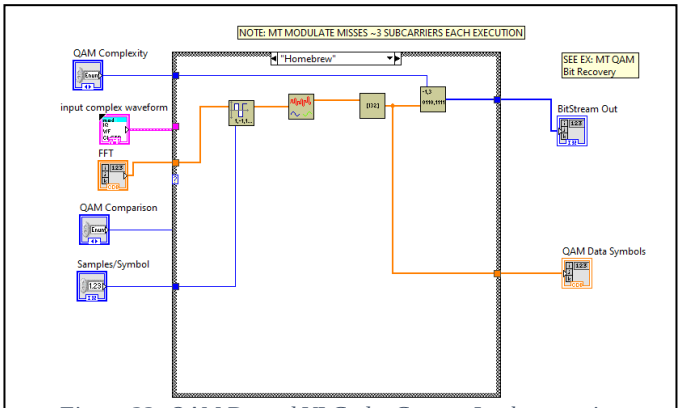

Figure 23: QAM Demod VI Code, Custom Implementation

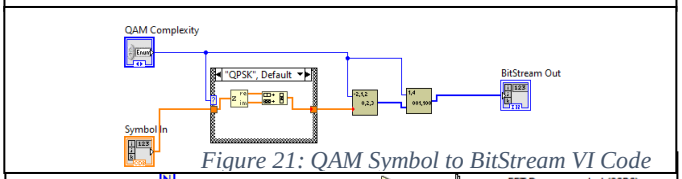

Figure 21: QAM Symbol to BitStream VI Code

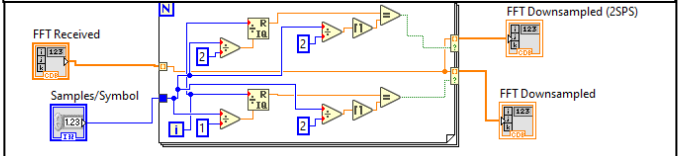

Figure 20: FFT Downsample VI Code

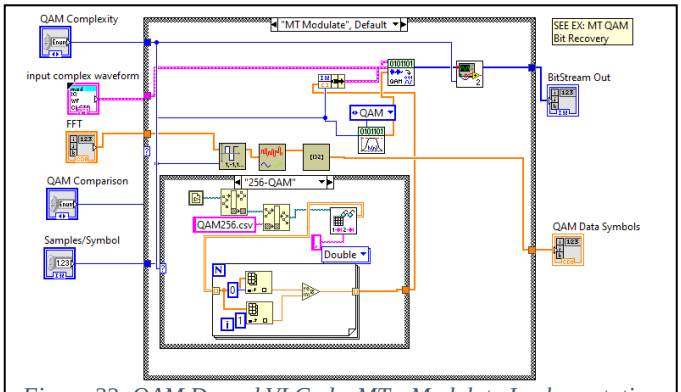

Figure 22: QAM Demod VI Code, MT - Modulate Implementation

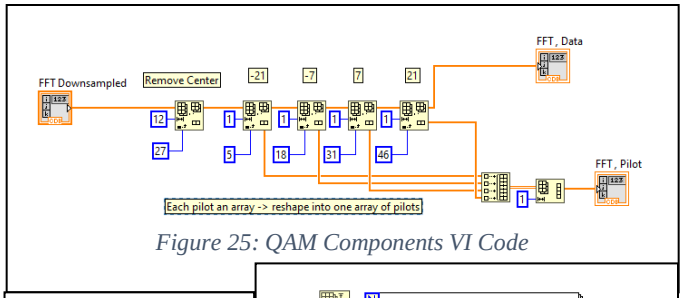

Figure 25: QAM Components VI Code

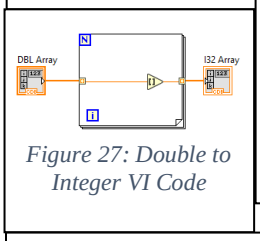

Figure 27: Double to Integer VI Code

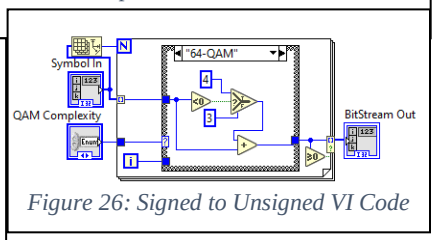

Figure 26: Signed to Unsigned VI Code

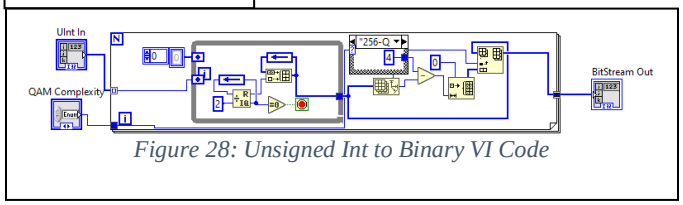

Figure 28: Unsigned Int to Binary VI Code

When executing the VI in the context of *Top VI*, some flaws become apparent in the overall operation. Notably, the *MT Modulate* implementation is consistently missing 3 subcarriers each execution. Comparison from the front panel indicates that the subcarrier weights, in both implementations, are not correctly received in comparison to the transmitted weights; this inaccuracy is widened with more complex QAM systems, where up to 22 subcarriers have been incorrect in a 256-QAM execution (Figure 25).

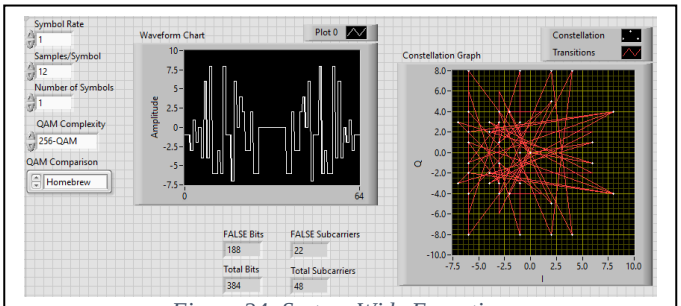

Figure 24: System-Wide Execution

III. Conclusions

While the operating principles are well-illustrated by the project, there are significant bugs to still be addressed to be useable as a simulation environment. Significant effort has been made, however, to make sure this project code is readable and well-structured; as such, it still provides a value as an educational tool for understanding how these signal processing techniques function and are implemented.

IV. Licensing and Availability

In order to provide this project to receive feedback and share the project for academic and personal reference, the LabView code has been published to GitHub under a General Public License. It can be accessed at: https://github.com/schimpfen/A-Simple-WiFi-PHY

Acknowledgment

The author would like to thank Dr. Huacheng Zeng and Hossein Pirayesh for conceptual guidance, and Dr. Dan Popa for project coaching.

V. References

Bloessl, B., Segata, M., Sommer, C., & Dressler, F. (n.d.). An IEEE 802.11a/g/p OFDM Receiver for GNU Radio. *Proceedings of the Second Workshop on Software Radio Implementations Forum.* Hong Kong.

Chen, X. (2012). In *Coding in 802.11 WLANs* (pp. 2-10). Maynooth, Maynooth, Republic of Ireland: National University of Ireland Maynooth.

Cisco. (2017). Cisco Viual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022. San Jose: Cisco.

Firdose, S. e. (2018). An OFDM Transmitter and Receiver using NI USRP with LabVIEW. *International Journal of Engineering Research and Technology, 6*(13).

Gast, M. (2013). 13. 802.11a and 802.11j: 5-GHz OFDM PHY. In *802.11 Wireless Networks: The Definitive Guide* (pp. 283-307). Cambridge: O'Reilly Media.

National Instruments. (2019). Lab 15: Principles of OFDM. In *Lab Manual: Communications Principles Using the EMONA Communications board for NI ELVIS III.* National Instruments.

Wireless Village. (2018). Wireless Village Hacking Lab: B-Sides DC 2018 Edition. *B-Sides DC*, (pp. 52-70). Washington D.C. Retrieved from https://wirelessvillage.ninja/docs/Wireless_BsidesDC_Training.pdf