Three Different Implementations for Quadrature Amplitude Modulation: Development of Teaching Materials from Research

Hadi Alasti

August 17, 2021

THREE DIFFERENT IMPLEMENTATIONS FOR QUADRATURE AMPLITUDE MODULATION: DEVELOPMENT OF TEACHING MATERIALS FROM RESEARCH

Hadi Alasti, Purdue University Fort Wayne

Abstract

This article presents three different implementation approaches for quadrature amplitude modulation (QAM) for teaching undergraduate technology students in digital communication courses. The approach of the article is to project a simplified version of the researchers' published work in development of the teaching materials for technology students. The important objectives of the approach are development of methods that are easy to understand for technology students and are low-cost to realize or implement, possibility to improve the quality of experience of students, increasing the student engagement with hands-on activities, and last but not the least improvement of the students' outcomes.

Introduction

The advancement of digital technology and the advent of modern software-defined technologies on microcontrollers and field programmable gate array (FPGA) have revolutionized the digital communication technology (Tioh, 2012). The application of wireless and wired communication is seen in most of the modern smart devices, such as home appliances, health monitoring instruments, industrial automated systems and tools, smart cars, etc. (Liscidini, 2015). However, in teaching of communication technologies to Engineering Technology students, the focus is mainly on fundamentals. Frequently, students return to the instructors that they have never seen any real-life industrial products as an instant of what they learned in the course. Moreover, the most up-to-date textbooks for analog and digital electronics communications are either based on basic conventional circuits, or mainly focus on theoretical concepts and fundamentals.

This article is on transforming a simplified outcome of research for teaching, and its objective is to present how algorithmic implementation of concept may help reduce the gaps between the course outcomes and the industry approaches in system implementation. In this article, the algorithmic implementation of quadrature amplitude modulations (QAM) using analog electronics circuits, analog-mixed signal ICs and software-defined components is presented.

Digital modulations are integral parts of digital communication systems. QAM is one of these modulations that have been used for years in variety of communication systems, such as digital television (DTV), digital subscriber line families (XDSL); almost all WiFi technologies; digital microwave radio links; 3G, 4G and 5G cellular mobile radio; fast optical communication over fiber optics; etc. This article reviews the realization or implementation of variants of QAM modulations using analog circuits, software-defined devices, and analog-mixed integrated circuits (ICs) for engineering technology students in electronics communications course. This implementation is based on simplification of the published research works (Alasti, 2015; Alasti, 2017; Alasti, 2018; Alasti 2020). The assumption for teaching this approach are familiarity of students with foundations of college algebra, familiarity with basic analog electronic circuits, and some basic background in programming.

Since hand-on learning approaches can help improve student-engagement (Chen, Luttuca & Hamilton, 2013; Lee, Godwin & Nave, 2018) and improve the quality of experience of students, and the student outcomes, it is expected that the proposed approach have the same result. For instance, by implementing 16-QAM using operational amplifiers, the students are not only engaged in a hands-on activity, but also can track the trace of mapping of binary data in output analog signals; or similarly in using reconfigurable computing or analog- mixed signal integrated approach. The algorithmic thinking feature of the approach allows the students to generalize, for instance implementing 64-QAM after learning what to do for 16-QAM.

The rest of this paper is organized as follows. In the next section the required algebraic foundation of the problem is reviewed. After that, the methods of implementation, based on analog circuits, analog-mixed signal ICs, and software-defined devices will be discussed.

Algebraic foundation:

Binary phase shift keying (BPSK) is one of the simplest binary digital modulations, with constant amplitude, where switching from a high to a low binary data or backward changes the radio frequency (RF) carrier phase, that can be done by multiplication of bipolar binary data and the RF carrier. Quadrature phase shift keying (QPSK) is made by superposition of two BPSK on 90-degree phase shifts of the same RF carrier. With orthogonality of Sine and Cosine, the simplest way to present this superposition is vectored implementation of QPSK using its two orthogonal components of in-phase and quadrature. Figure 1, illustrates this implementation by assigning one pair of binary data to a unique point in the signal space that is represented by one vector.

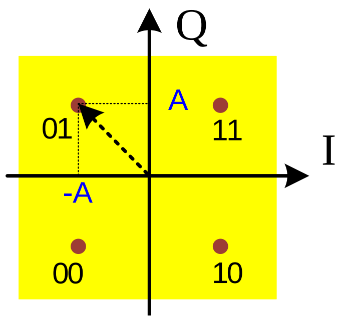

Figure 1: Vectored implementation of QPSK.

The baseband presentation of QPSK in basic algebraic form is according to (1), where S is the QPSK signal and b_i and b_q are the binary data on in-phase (I) and quadrature (Q), respectively. Also, $\hat{x}$ and $\hat{y}$ are the unit vectors of the I and Q axis.

$$S = A\, b_i\, \hat{x} + A\, b_q\, \hat{y} \tag{1}$$

Vectored implementation of hierarchical QAM modulation (Alasti & Gazor, 2015) details how 16-QAM is generated by superposition of two QPSK signals at two different amplitudes levels of A_1 and A_2, as illustrated in Figure 2. In this implementation, the related modulated binary data for in-phase (I) and quadrature (Q) channels are separable, as it is illustrated in figure 2. The advantages of this approach are its low complexity and that it does not need to Gray coding if proper decoding is used at received.

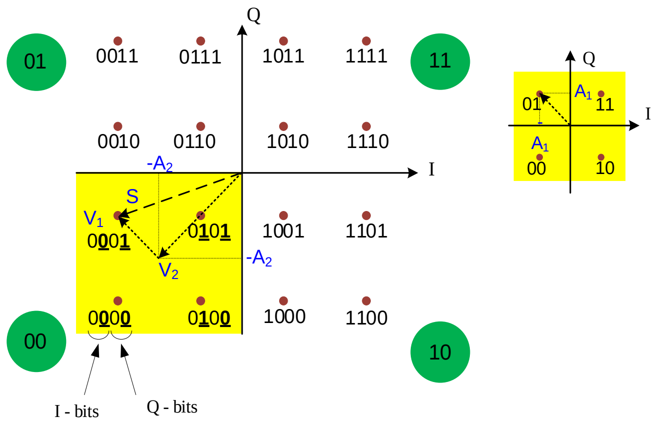

Figure 2: Vectored implementation of 16-QAM

Similarly, based on vectored approach it is possible to implement 2^{2n}-QAM using "n" vectors with equal I and Q components (Alasti & Gazor, 2015). The algebraic presentation of baseband 2^{2n}-QAM is according to (2).

$$S = \sum_{j=1}^{n} A_j\, b_{ij}\, \hat{x} + A_j\, b_{qj}\, \hat{y} \tag{2}$$

Methods of implementation:

Before starting this section, it is worthwhile to mention that, even though this article introduces three different implementation approaches, however their effectiveness and the student success depends on student readiness in related background of these three approaches, and also to the available hardware or software tools. The students experimentally learn by following the instructor. In the first experiment the students follow the instructor, step by step to learn the principle of the work. For instant, they follow the instructor in how to implement 16-QAM using one of these three approaches. In the next step, they will challenge with the problem, help each other and they may need to review the principles of the approach for implementation of 64-QAM.

Implementation of 2^{2n}-QAM using analog circuit:

According to (2), the algebraic implementation of baseband 2^{2n}-QAM consists of "n" comparators to map the binary data to the x and y components of vectors, that are "A_j" factors in (2) and two adders, for in-phase and quadrature channels.

Analog electronic circuits are usually among the courses that students take in their early years of college. Operational amplifiers (OpAmp) are the most common way of realization of adders and comparators. Taking this into account, the proposed vectored approach needs to "n" comparators and 2 adders, and the required resistors. Figure 3, illustrates the analog implementation of 16-QAM using IC 741. For this implementation, NI Multisim was used, which is a common software in Engineering and Technology schools.

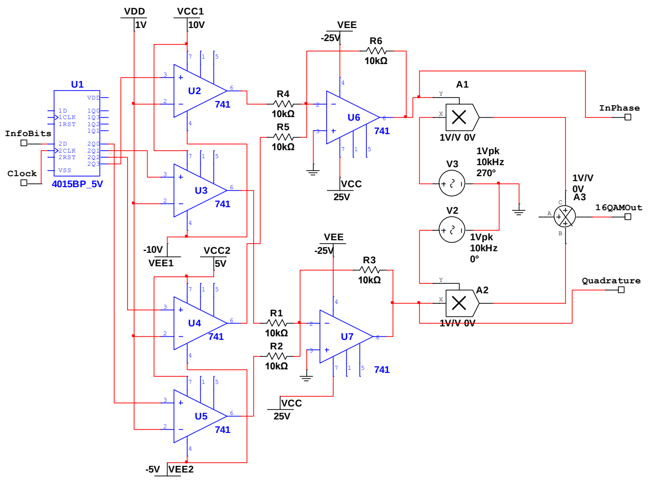

Figure 3: Analog implementation of 16-QAM using several IC 741 OpAmp

The RF output of this modulator is illustrated in figure 4. The sudden variation of the carrier's amplitude and its phase and also the number of possible carrier amplitudes are visible in this figure, where it allows the students to make a connection between signal space and waveform. The introduced analog implementation of 2^{2n}-QAM is one example for application of algebra and analog circuits in one real-life implementation. The approach shows the importance of these materials in program's curriculum. QAM is a necessary part of modern communication technology. Once the students see that what they learn in their courses and their program has application in modern and real-life projects, they get more engaged in what they get instructed.

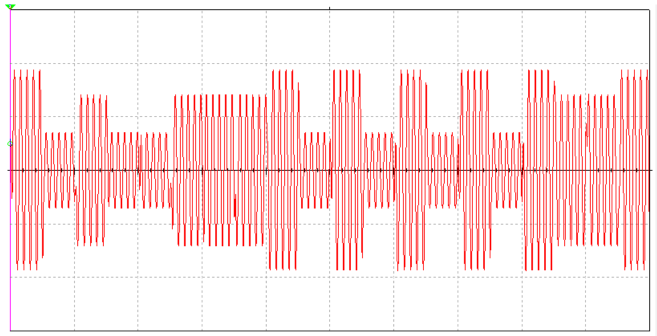

Figure 4: 16-QAM RF-waveform in time domain on oscilloscope screen.

Software-defined implementation of 2^{2n}-QAM:

Software defined applications are the most common market demands that college graduates need to know about. Luckily, the non-expensive microcontroller boards such as Arduino family boards easily allow the students and educators to conduct related experiments. The simple algebraic implementation of 2^{2n}-QAM according to (2) is easily possible on microcontroller using a few lines of code. The students may practice the first implementation as their instructor shows them how to do the work. However, the algorithmic thinking approach put them in challenge to implement other orders of the problem on their own. Figure 5, illustrates the signal space of 64-QAM. This oscilloscope snapshot represents the implementation of 2^6-QAM on ARM Cortex-M in Arduino-DUE board, that has two analog-outputs for the in-phase and quadrature channels. C programming language was used for this purpose. A simple implementation code for duplication of this constellation is provided in Appendix of this article.

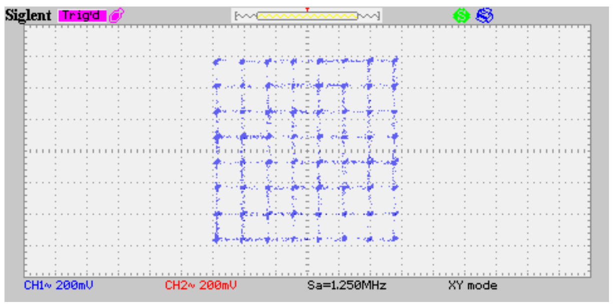

Figure 5. The signal space of 64-QAM, generated by implementation on ARM Cortex M

Besides microcontrollers, using other forms of software-defined devices such as DSPs and FPGAs is also possible. The simplicity of work, the hardware capability, expenses and compatibility are among the factors that are usually considered for selection of the device.

Implementation based on using analog-mixed ICs:

Besides using analog electronics and software-defined devices for this application, it is also possible to use analog-mixed signal ICs. Analog to digital converters (ADCs) and digital to analog converters (DACs) are among these ICs. It has been proved that the baseband QAM with rectangular signal space can be generated using two DAC units as it is illustrated in Figure 6 (Alasti, 2017). For this purpose, it is required to use DAC ICs, or software-defined equivalent for constellation mapping. Apparently, two ADC ICs with the same codeword size are required for binary data decoding. Figure 7, represents the binary data decoding using two ADCs. The input to the Figure 6 are the binary data and its outputs in I and Q branches are discrete analog levels. These discrete levels in Figure 7 are returend to the same original binary data.

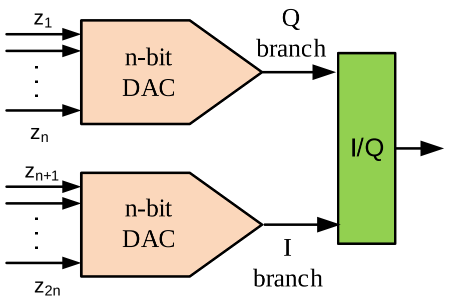

Figure 6. Constellation mapping for 22n-QAM using two DAC IC units

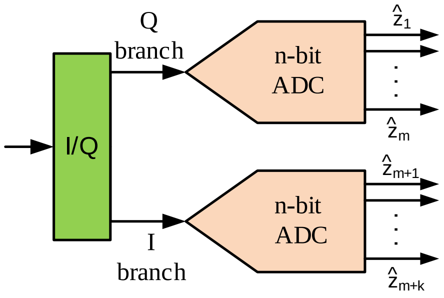

Figure 7. Binary data decoding using two ADC IC units at receiver side

Conclusion:

Three different ways of implementation are presented for teaching quadrature amplitude modulation (QAM) in technology program. The proposed approaches are the simplified versions of the discussed materials in research works. Analog electronic circuits, software-defined approach and analog-mixed signal integrated circuits are used for this implementation. The objectives of the proposed approach are to improve the quality of experience of students, to teach students how to think algorithmically to implement relatively complex systems, to improve the students' outcome, and to increase the student engagement with hands-on activities. With the proposed approaches, the students can trace the conversion of the binary data to the analog waveform, that will be ready to be transmitted over the channel. The proposed appoach provides one more step beyond pure theoretical learning, where it shows them exactly how the binary data in digital system will be combined before transmission, which bit is more vulnarable and which one is stronger. The improved student outcomes are algorithmically thinking with hands-on experience and applying the foundation and prerequisite courses in real-life implementation. The students' feedback showed that they enjoyed the correlation that these implementations provided between their foundation courses and the market needs. Among the market needs are "correct understanding of vulnerability of different combined bits in QAM", "experience in correlating the complexity of the waveform and the modulation contetnt", "work experience with implementation of a basic system using reconfigurable computing devices", "work experience with basic analog-mixed circuits", and last but not the least, "algorithmically thinking". In this study, no data was collected for statistical analysis purpose. However, in general, the students learned enough from these experiments. The success to implement a different order of the QAM from a previously instructed one can be a prove of the students' learning. The hands-on nature of the labs helped improve the student engagement and their performance. Overall, more than 70% of the students showed correct understanding of the problem and ability to think algorithmically to implement another order of the system.

References

Alasti, Hadi and Gazor, Saeed. 2015. "Vectored-implementation of hierarchical 2^{2n} QAM." IEEE Trans. Circuits Syst. II, 62(11): 1103-1107.

Alasti, Hadi. 2017. "Multi-resolution multiplexing for 5G wireless spectral efficiency enhancement in strong noise.", in Proceedings of IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), Montreal, QC.

Alasti, Hadi. 2018. "A Low Complexity and Flexible Implementation of 2^n-QAM for Software Defined Radio Applications," in proceedings of IEEE annual Consumer Communications & Networking conference (CCNC), Las Vegas, 1-6.

Alasti, Hadi. 2020. "A Generalized Vectored-Implementation of 2n-QAM for Software Defined Applications." In proceedings of IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), Virtual conference, 786-789.

Chen, Helen L., et al. 2013. "Conceptualizing Engagement: Contributions of Faculty to Student Engagement in Engineering." Journal of Engineering Education, 97(3).

Lee, Walter C., et al. 2018. "Development of the Engineering Student Integration Instrument: Rethinking Measures of Integration." Journal of Engineering Education 107(1).

Liscidini A. 2015. "Fundamentals of Modern RF Wireless Receivers: A Short Tutorial" IEEE Solid-State Circuits Magazine, 7(2): 39-48.

Tioh, J., et al. 2012 "Reconfigurable high-speed platform: shifting the paradigm in education, research and engineering." IEEE Communications Magazine, 50(1): 153-159.

Appendix

The following code is a simple implementation for baseband 64-QAM over Arduino Due with ARM Cortex M microcontroller. To view the constellation on oscilloscope, setup your screen in XY mode.

```
int B[10000];                     \\ random binary data
float Level[3]={1,2,4};           \\ Ai values
void setup() {
  pinMode(DAC0,OUTPUT);
  pinMode(DAC1,OUTPUT);
  randomSeed(analogRead(0));
  for(int k = 0; k < 10000; k++)
     B[k] = random(2);
}
void loop()
{
  for (int m = 0; m < 2500; m++)
     {
      float S_i = 0;               \\ in-phase term of signal S
      float S_q = 0;               \\ quadrature term of signal S

      S_i = S_i+Level[0]*Sgn((float)B[4*m+0]-0.5);
      S_i = S_i+Level[1]*Sgn((float)B[4*m+1]-0.5);
      S_i = S_i+Level[2]*Sgn((float)B[4*m+2]-0.5);

      S_q = S_q+Level[0]*Sgn((float)B[4*m+3]-0.5);
      S_q = S_q+Level[1]*Sgn((float)B[4*m+4]-0.5);
      S_q = S_q+Level[2]*Sgn((float)B[4*m+5]-0.5);

      S_i = S_i * 150.0 + 2047.0;
      S_q = S_q * 150.0 + 2047.0;
      analogWriteResolution(12);
      analogWrite(DAC0,S_i);
      analogWrite(DAC1,S_q);
     }
}
int Sgn(float x)
{
 if (x > 0)
   return 1;
 else
   return -1;
}
```

s