Data Encoding and Compression

By Matthew Trivigno, ECE '22

Introduction

Modern digital devices are expected amongst consumers to have the ability to be plugged into something and be used immediately with ease. Many of us can remember a time when plugging in a new external drive or keyboard into a computer required installing an extensive number of drivers. These drivers are essential in facilitating communication between the operating system and your device. What many people never think about is how this transfer of data occurs. Often times, this requires a standardized encoding of data and compression to increase the speed of data transfer

Background Information

Fundamental to encoding and compression is an understanding of how data is stored in the first place. Mainly it is important to understand the distinction between analog and digital data. Analog data is any data that is stored physically, be it information written on a piece of paper, or sound etched into a vinyl record. This data is limited by the precision with which you can record and read the data. Think of a ruler that is precise only to a $1/16^{th}$ of an inch and trying to mark a line that's a $1/32^{nd}$ inch long. Digital data is data that is recorded using discrete values, usually represented in binary. It is limited simply by the amount of storage you have. If you want to represent a binary number with a certain number of digits, you need only the storage to record all those digits.

To help better frame the two in the context of audio devices, think of how a sound wave might be recorded on a vinyl record versus your computer. The analog signal can move the needle that scratches the record with extensive precision, whereas the digital microphone would need to sample the wave at a set rate and record its best guess as to the shape of the wave.

Encoding Techniques

Digital data is represented in binary, with the prefix bi meaning two. That's why in binary there are just 1's and 0's. If you want to encode binary/digital information into an analog signal, you can either make set changes to the amplitude, frequency, or phase of a wave to represent bits.

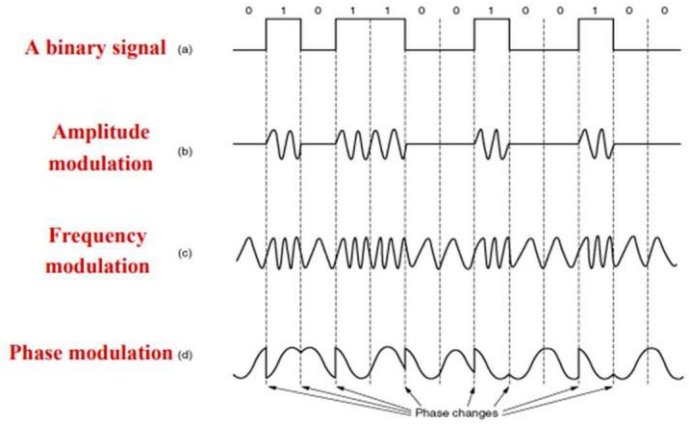

Figure 1: Visualization of Digital Data Encoded as Analog Signals

This type of data encoding is very useful in modems, where your home network is connected to the internet, and information is carried along coaxial cables. However, when most people thing of digital to analog converters, they are thinking of transforming data on their computers into audio recordings. This is achieved through digital to analog converters (DAC), and this can be best illustrated through an R-2R Ladder DAC.

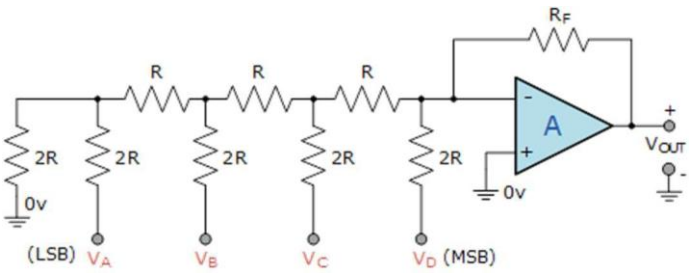

Figure 2: R-2R Ladder DAC Circuit

Shown in Figure 2 is a 4-bit R-2R DAC. All bits being low results in no voltage, and all bits high allows voltage to flow through each junction resulting in the max voltage. Enabling some bits allows variations in the voltage on the output. By switching these bits extremely fast, one could replicate signals such as sine waves, triangle waves, or a multitude of different sounds. Including more DACs of course allows for the layering of multiple audio channels to replicate recordings.

Digital Data to Digital Signal

The technique of converting digital data to a signal is very useful in transferring said data over a wired serial connection. This is how USB enabled devices communicate, as well as how computers on wired ethernet connect to a local area network. The main types of digital-to-digital processing are NRZ and Manchester encoding.

Non-Return-to-Zero Codes

Non-Return-to-Zero-Level (NRZ-L) transfers data by encoding a voltage high for a bit of 1 and a voltage low for a bit of 0. Non-Return-to-Zero-Invert (NRZ-I) encodes a 1 as signal transition and a 0 as no signal transition. It does not matter if this transition is from low to high or high to low voltage, any transition at the beginning of the bit time encodes a 1.

Manchester Encoding

Manchester encoding differs from NRZ encoding in that it always has a mid-bit transition. This is an advantage over NRZ encoding, as it can be used as a clocking mechanism to synchronize with the device receiving the data. Standard Manchester encoding encodes a 1 with a low-to-high transition, and a 0 with a high-to-low transition. Differential Manchester encoding is similar to NRZ-I encoding, in that the absence and presence of a transition at the bit interval indicates a bit. The absence of a transition is a 1, and the presence a 0.

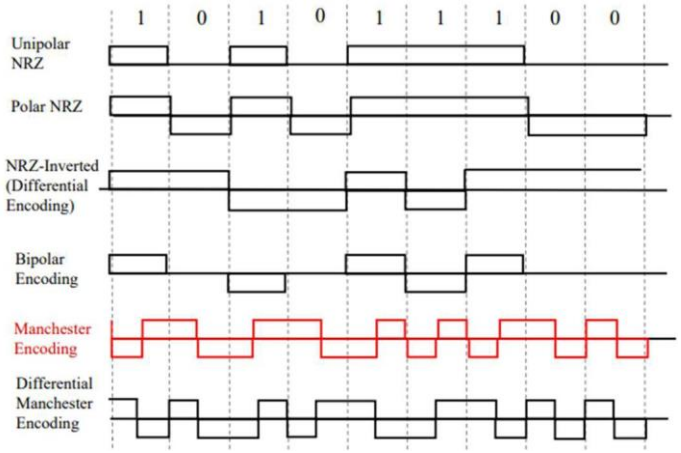

Figure 3: Different Types of Digital Data to Digital Signal Encoding Compared

Data Compression

The method of reducing the amount of data to transmit is known as compression. This is useful in increasing speeds of transmission and reducing file size. Compression comes in two primary forms: lossless and lossy compression. Lossless compression aims to exploit statistical redundancies to reduce the amount of data to be stored. Lossy compression makes logical and deliberate choices to remove some data that is not redundant, and thus some quality is lost. However, often times the data removed in lossy compression is imperceptible or unnecessary. An examples of lossy compression would be an MP3 file, and lossless compression would be a FLAC file, which is not used by many people other than enthusiasts and musicians.

Lossless Compression Methods

One of the most widely utilized methods of lossless compression is Huffman Coding, and this method will help demonstrate how to compress data without losing any information.

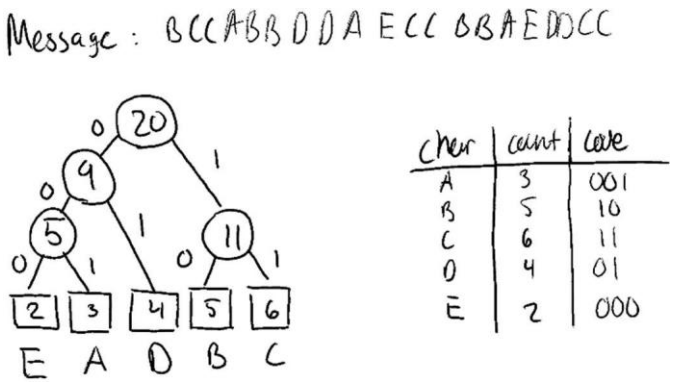

Figure 4: Example of a Hamming Code for an ASCII Message

Let's say you wanted to send a message with ASCII characters, but you were only using letters A through E. It would be redundant to send all 8 bits for each letter. A Hamming Code outlines a method for you to basically encode each character using fewer bits in a way that optimizes for space

This algorithm is performed by first constructing a table of each character in the message and counting how many times each character appears. Write those letters out in order from least often to most often. Create new nodes by adding the two smallest numbers available and writing that number in that node. You are finished once you have created a node of maximum size. Write the bit 0 on a left edge and the bit 1 on a right edge. Follow from the top of the tree to the bottom to get your new code for each letter. In the example given in Figure 4, this reduces our message from a size of 120 bits (8 bits per character) to 45 bits, and no information was lost. In this case, you would have to provide a conversion from ASCII to the new code that increases the size of the file, but ultimately still succeeds in compressing the message.

Other compression methods exist that work in much the same way of removing redundant information either through statistical analysis or algorithmically.

Lossy Compression Methods

Lossy compression methods attempt to remove some information from the data being stored, usually through the removal of data deemed unnecessary by the compression algorithm. Though each algorithm differs greatly in each context (audio, video, etc.), the main idea is to remove stored data that is either beyond human perception or otherwise superfluous. For example, audio compression will often remove frequencies that around outside the range of human hearing. Photo compression will blend similar shades of color together, which our eyes are often bad at distinguishing. Although some compression takes it further to the point where the differences are perceptible, there is always a tradeoff between file size and quality to be considered. In some artistic mediums, compression can be utilized intentionally and aggressively to convey emotions. Compressors are often used for effect on guitars and other instruments even during live performances, and analog circuits can even be constructed to give the same effect.

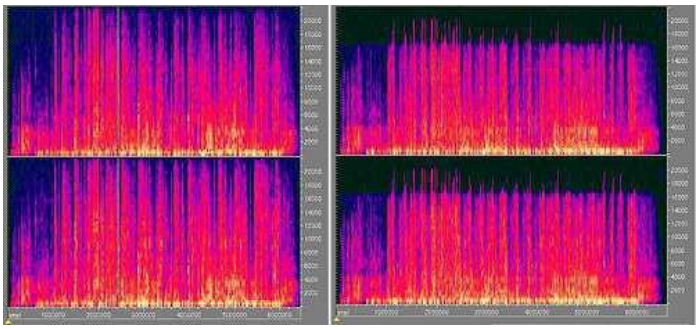

Figure 5: Comparison Between Lossless (left) and Lossy (right) Compression

References

1. GreatScott! (2014, December 5). *Electronic basics #10: Digital to analog converter (DAC)*. YouTube. Retrieved January 23, 2022, from https://www.youtube.com/watch?v=Y2OPnrgb0pY
2. Wikimedia Foundation. (2022, January 17). *Data compression*. Wikipedia. Retrieved January 23, 2022, from https://en.wikipedia.org/wiki/Data_compression
3. *Data Encoding Techniques*. WPI Computer Science. (n.d.). Retrieved January 23, 2022, from https://web.cs.wpi.edu/~rek/Undergrad_Nets/B06/Data_Encoding.pdf
4. Bari, A. (2018, February 8). *Huffman Coding - Greedy Method*. YouTube. Retrieved January 23, 2022, from https://www.youtube.com/watch?v=co4_ahEDCho
5. *Different techniques of encoding data for transmission*. Section.io. (n.d.). Retrieved January 23, 2022, from https://www.section.io/engineering-education/different-techniques-of-encoding-data-for-transmission/