YODA Project: Audio Steganography Encoder & Decoder

EEE4120F: High Performance Embedded Systems

Kudzai Chris Kateera‡
"R2-D2"
EBE
dept. Electrical Engineering
University of Cape Town
Cape Town, South Africa
KTRKUD001@myuct.ac.za‡

Shingirai Denver Maburutse†
"Anakin Skywalker"
EBE
dept. Electrical Engineering
University of Cape Town
Cape Town, South Africa
MBRSHI002@myuct.ac.za†

Gregg Finn§
"Obi-Wan Kenobi"
EBE
dept. Electrical Engineering
University of Cape Town
Cape Town, South Africa
FNNGRE002@myuct.ac.za§

***Abstract*—*Index Terms*—embedded-systems, modulation, time, performance, correlation, accelerator, python, Verilog, Steganography, Audio, Star-Wars, parallelism, FPGA**

The passing of secret messages across the internet is becoming more and more of an insecure endeavour. Interception and eavesdropping is becoming easier for hackers who are using tools to keep up with the current methods of message hiding. In a galaxy far-far away, it's a common issue where both fronts want to be sure that if their messages are intercepted, at least the main plans to either destroy/make the death-star to not leak to the opposition The report introduces the background, literature and methods of steganography that are currently under research. It then sheds light on the loopholes and gaps to what is not yet or not sufficiently looked at methods of steganography. There are several methods of conducting steganography but in this paper we focus on Audio Steganography - hiding an audio inside another audio in a manner that is inaudible to humans. It is then introduced and justified for its validity and reasoning to be under study. The methodology of how the experimentation is going to be conducted is discussed and justified on why certain decisions were made in regards to the method undertaken. The golden measure program is used for the methodology with a serial implementation. After ensuring it meets the usability requirements a Design for the Accelerated version is discussed. This includes how the parallelism is going to be done and the benefits of it are expressed mathematically. A proposed development strategy is then setup on how the product can be commercialized and be used by the general public through abstraction of technical details. A planned experimentation is forgone leading to the Results and Discussion of the implementation. The Report is finally concluded and the results compare to the goal of the study.

I. Introduction

Steganography is defined as the technique of hiding classified data within a non-secret file with the intention that the data remains undetected until its extraction. Steganography should not be confused with the art of cryptography, which rather involves the implementation of "crypto" keys to lock out unauthorized users from accessing information that they are not entitled to access.

At its core, the idea of steganography is covert communication in comparison to cryptography's data protection. Despite this, steganography has a plethora of other potential benefits and applications, including digital rights management.

The focus for this project is the development and prototyping of a steganography encoder and decoder. Since the objective is to hide data in a unique way, there are many formats of steganography, such as image, audio or video, as well as many methods of executing it, such as Least Significant Bit Encoding or Echo Hiding. In steganography there are two components that make up the output file. These two components are the cover object and the stego object, which are defined as the overt data being sent and the secret data embedded within it, respectively.

While the current methods of steganography are already well established and proficient, the objective of this paper is to investigate the concept of encoding the covert data into the high-frequency signals of the cover object such that they are imperceptible to the human ear.

II. Background

Sending a message from one location to another is not a fully secure endeavour for the sender and receiver because of the possibility of interception through eavesdropping. To ensure that the messages are not accessed(at least the parts where privacy is a priority), steganography is used. This is where the sender and receiver would be aware of how the message was encoded and therefore know how to decode it. The eavesdropper will not be aware of how the encoding was therefore will not be able to intercept the hidden message.[8]

With the emergence of Artificial Intelligence and Advanced Machine Learning it has become easier for eavesdroppers to easily intercept messages with encoded hidden messages due to the ubiquity of practises like least significant bit manipulation.

Upon exploring the avenue of Audio Steganography which is already a method not widely used, it was found that the

least significant bit manipulation method for .wav audio files is the most ubiquitous method.

A more robust mainly in the perspective of covertness is frequency manipulation. Frequency manipulation involves changing a hidden audio message's frequency to a higher band (> 20kHz) that it cannot be interpreted by the human ear.

There are two methods proposed to achieve this:

- Frequency modulation of the hidden message
- Amplitude modulation of the hidden message
- Phase modulation of the hidden message

III. Methodology

The human audible spectrum exists between the range of 20Hz to 20kHz and thus are only capable of detecting sound within that range, which might not be the case for other mammals and species in the galaxy. The upper limit of the spectrum may extend itself slightly past 20kHz for infants, but with age and maturity, it is gradually decreased to anywhere between 15 and 17kHz for the average adult.

Considering the characteristics of both humans and audio files, it becomes apparent that embedding secret messages within the high frequencies of a cover object has a convincingly large amount of potential to become a unique and efficient form of steganography.

In order to obtain a golden measure, the program was be designed in python, as it is not only easily accessible but has incredibly vast library support. Despite originally attempting to find a solution using frequency modulation, it was discovered that the necessary frequency changes were substantially complicated and out of the projects available time period and overall scope. Thus, a decision was made to use amplitude modulation instead. This would ensure that when a carrier wave of sufficiently high frequency is used, there would be minimal interference with the main audio file

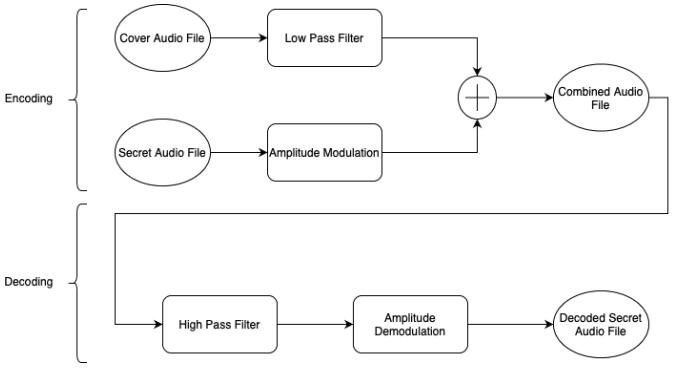


Fig. 1. Methodology Flow Diagram

A. Encoding

Essentially, two waveform audio files are loaded into the program, a cover object and a stego object. This is done using a simple line of code that stores both the wave files data and sample rate:

```
self.rate, self.data =
    wav.read('./audios/swthemesong.wav')
```

For any given wave file, the sampling rate is standardized at 44.1kHz, which means without aliasing, the highest frequency found within the file would be 22.05kHz. Considering the average adults audible range which was previously discussed, these leaves about 5khz of frequency that is capable of storing data but will most likely be unheard.

In order to clear this frequency band, our cover audio is passed through a low pass filter of 18kHz, leaving a 4kHz band for our secret audio to be placed in to. This is done using the function defined below:

```
def lpf(data, rate):
    sos = signal.butter(10, 10000, 'lp',
        fs=rate, output='sos')
    filtered = signal.sosfilt(sos, data)
    return filtered.astype(data.dtype)
```

Using audio modulation, the secret audio is then modulated with a carrier frequency of 20kHz, the mid point between 18kHz and 22kHz. Amplitude modulation is performed using the following equations with the intention of shifting the audio signal to a desirable frequency.:

$$c(t) = A_c \sin(\omega_c t) \tag{1}$$

$$m(t) = A_m \sin(\omega_m t + \phi) \tag{2}$$

$$A_c sin(W_c t) = \frac{A_c m_a}{2} \tag{3}$$

Amplitude modulation was implemented into the project using the Python function created and defined in the block of code below:

```
def am_modulator(data, modulation_index=0.5):
    t = np.linspace(0, 1, data.size)
    f_c = 20000
    A_c = 10
    carrier = A_c * np.cos(2 * np.pi * f_c * t)
    product = ((1 + modulation_index) * data)
        * carrier
    return product.astype(data.dtype), carrier
```

The now modulated secret audio is combined with the LP filtered cover audio to create the final intended output, an audio file containing a hidden message. The wave audio file is subsequently saved to the local path.

```
write(filename="./audios/asd1-swsteganograph.wav",
    rate=self.rate, data=self.final)
```

B. Decoding

While the encoding aspect of the project is sufficient in the field of digital watermarking, the intention was to develop a form of viable secret communication. Therefore, this section details how decoding and extracting the secret message from the combined audio file was implemented within the program.

Before demodulation of the secret message can occur, thus bringing it back to its original frequency, the cover audio must first be filtered from the combined audio. A high pass filter is implemented to filter out any frequencies below 18kHz and is done so using the following code:

```
def hpf(data, rate):
    sos = signal.butter(10, 11020, 'hp',
        fs=rate, output='sos')
    filtered = signal.sosfilt(sos, data)
    return filtered.astype(data.dtype)
```

The filtered signal is then demodulated using the same 20kHz carrier frequency as before:

```
def am_demodulator(modulated_data,
     modulation_index=0.5):
    t = np.linspace(0, 1, modulated_data.size)
    f_c = 20000
    A_c = 10
    carrier = A_c * np.cos(2 * np.pi * f_c * t)
    q = ((modulated_data / carrier) - 1) /
        modulation_index
    return q.astype(modulated_data.dtype)
```

The remaining signal will be one that resembles the initial signal of the secret audio, although it may suffer from some distortion due to the additive noise from both addition and operating at high frequencies due to modulation.

Source Code: Github

IV. Design

Due to the need for a high-speed decoding to match the firing input audio, a high-speed Parallel Algorithm will be used. The audio will be batched into multiple sections using time as the determining parameter.

Assuming the input signal of the digital filter is x(n), the output y(n). The mathematical expression of the FIR filter will be expressed as: [5]

$$[H]y(n) = \sum_{i=0}^{N-1} h_i x_{n-i} \tag{4}$$

where:
h(i) is the impulse response coefficients of the FIR filter
N is the order of the filter
Frequency Response Filter:

$$H(z) = \sum_{i=0}^{N-1} h_i z^{-k} \tag{5}$$

A 4-tap transposed FIR filter is derived from the following Signal Flow Diagram:

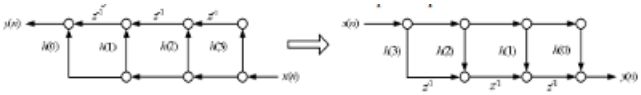


Fig. 2. Serial Signal Flow Diagram

The sampling period of the structure is:

$$T_s >= T_{multi} + T_{add} \tag{6}$$

Sampling rate:

$$f_S = \frac{1}{T_{multi} + T_{Add}} \tag{7}$$

The Serial Transposed FIR filter is a single input single output system (SISO). It is also Single Data Single Program system (SDSP).

For the parallelized transposed filter (Fig. 2) it will be an Unfold Transform[REF:6] version of the serial FIR filter.

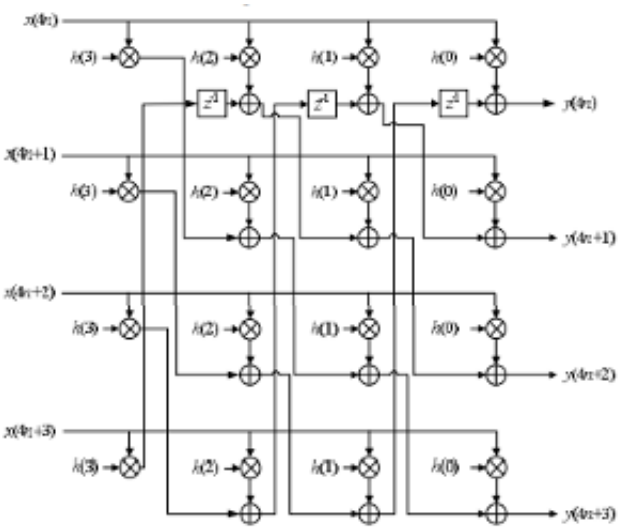


Fig. 3. Parallel Signal Flow Diagram

The parallel transposed FIR filter uses a Multiple-Input Multiple-Output system. In each clock cycle, there are 4 parallel input signal to be processed and four sampling points to be generated simultaneously. The critical path is kept constant and the clock period should be maintained:

$$T_{clk} = T_{multi} + T_{Add} \tag{8}$$

And because 4 sampling points are processed in a single clock cycle, the sampling cycle will be a quarter of the original:

$$T_s = \frac{1}{4} T_{clk} >= \frac{1}{4}(T_{multi} + T_{Add}) \tag{9}$$

Thereby producing a system with 4 times the throughput of a serial system.

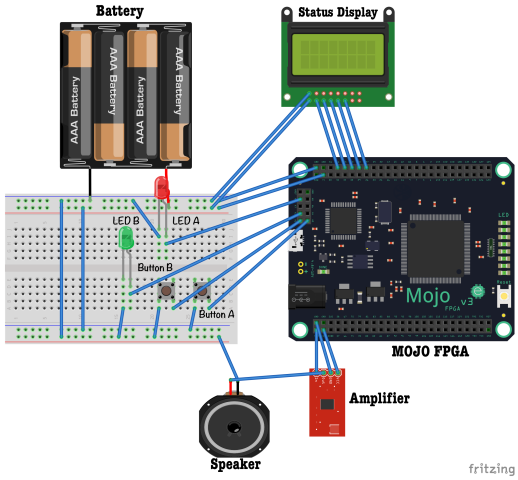


Fig. 4. Bread-Board Schematic

Fig 4 Labels:

- *Battery*: Power Supply to the Device Status Display: Displays the encoding/decoding process (eg. Decoding..., Decoding a Success! Press Button B)
- *Amplifier*: Elevates the output sound for clarity and high value sound
- *Speaker*: Outputs the decoded audio
- LED_A : Flashes to signify malfunction
- LED_B : Flashes to signify successive decoding (ready to play)
- Button_A: Start/Reset
- Button_B: Play Audio

V. Proposed Development Strategy

The product is intended to be stand-alone, allowing for external device connection using a Serial bus interface, Disk Drive and an audio jack interface to connect to a speaker as its output.

The audio with the encoded message will be read for the external drive where the message will be filtered, shifted in the frequency band and then played.

Universal Serial Bus Interface[7]: The USB can allow direct decoding or connecting to a host computer containing the audio file to be decoded. The USB assures asynchronous serial data transmission with approximate speed of 10Mbits/s, allowing the audio file to be fully buffered in a short period of time. The interface allows for a Plug and Play functionality standard based on the communication layer as shown on Fig3

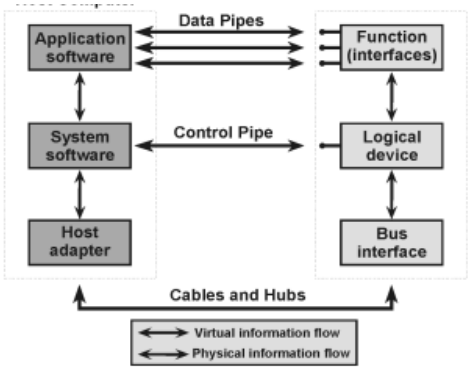


Fig. 5. USB Communication Layers[7]

Audio-jack Interface

To make it compatible for broadcasting, the audio jack can be useful to output to high power speakers. It can also be used as a portable device thereby allowing use of earphones and headphones is a possibility.

With the schematic representation on Fig6 it is possible to make a minimalistic PCB version Fig7 allowing it to be fully mobile and be used as a private communication Device

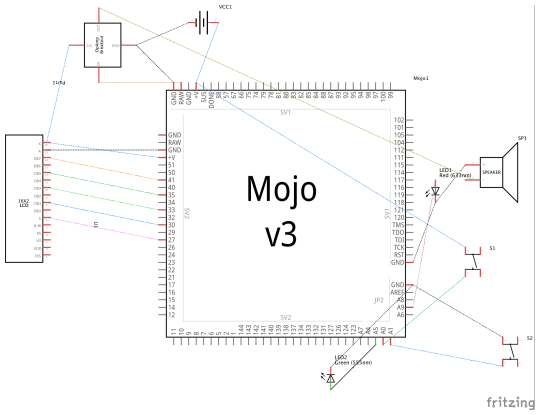
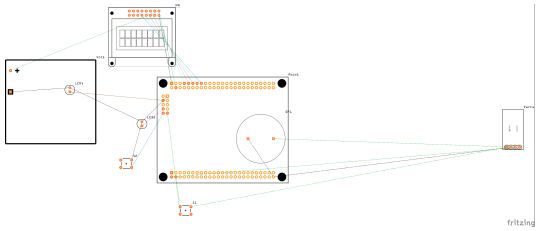


Fig. 6. Circuit Sketch



A 3D printed case would be made with to encapsulate and abstract the circuitry and allows for a great user experience for the general public. The Orthographic View with 3rd angle projection of the 3D printed case is as shown in Fig 8

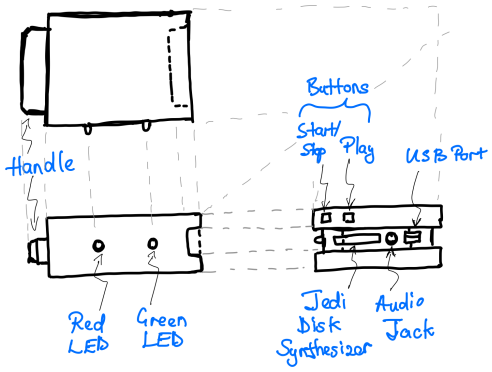


Fig. 8. Printed Case 3rd Angle Projection View

VI. Planned Experimentation

The golden measure was setup using Python and a working program is the first step that is taken. Each implementation involving a set of data structure is noted and analysed.

The problem at hand is analysed and each data-structure implementation is checked to see its Big O time complexity. It is then considered whether parallelization should be considered based on how complex the section is to parallelize and if it yields a considerable speedup. For the sections that are considered beneficial to parallelize , a partitioning scheme is agreed upon. A partitioning scheme that minimizes overhead, at the same time maximizes parallel performance. Functional decomposition is used for functions in the program and domain decomposition is implemented on wave manipulation sections. The granularity of the problem is considered and compared to the granularity achieved when parallelism is executed. After data dependencies are ensured to be considered synchronization spots are selected. To make sure all threads have optimal work they are doing, load balancing is implemented. Finally performance analysis is done and tuning to maximize this performance boost and compared to the golden measure to show evidence of high acceleration.

VII. Results and Discussion

This section presents and discusses the results of the projects implementation.

Performance				
exp no	modulation	superposition	filter	demodulation
1	0.183	0.017	0.138	0.381
2	0.292	0.017	0.293	0.37
3	0.192	0.021	0.185	0.479
4	0.163	0.013	0.175	0.3
5	0.182	0.018	0.138	0.335
6	0.214	0.026	0.138	0.302
7	0.198	0.018	0.147	0.33
8	0.184	0.015	0.19	0.612
9	0.149	0.013	0.228	0.564
10	0.176	0.015	0.19	0.29
avg	0.1933	0.0173	0.1822	0.3903

TABLE I
TABLE TO COMPARE FUNCTION TO FOR LOOP

Table 1 shows the timings for the modulation of the secret message signal, superposition of the modulated signal and the carrier audio in the encoding process.In the decoding process, the high pass filtering process is timed as well as the demodulation of the secret message. The timings show that the encoding superposition occurs fastest followed by the decoding filtering then encoding modulation and finally decoding demodulation. The accelerator can be utilized to speed up these processes which include the same set of instructions on different data in the arrays.

Using the formula from the design section (9). We can perform a speedup of magnitude 4 or the filtering (low-pass and high-pass) process.

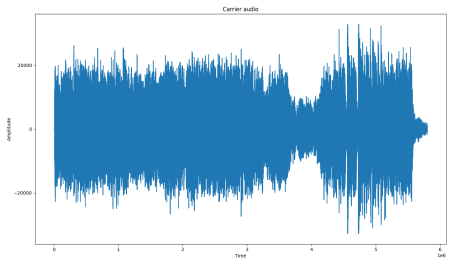


Fig. 9. Carrier Audio Amplitude vs Time

Figure 9 above shows the carrier audio before it has been modified to include the modulated secret wave. The wave audio is 4:23 long.

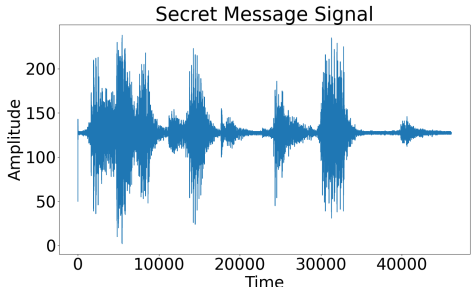


Fig. 10. Secret Message Signal

Figure 10 above shows the 2-second secret message signal. That is its time domain representation before modulation.

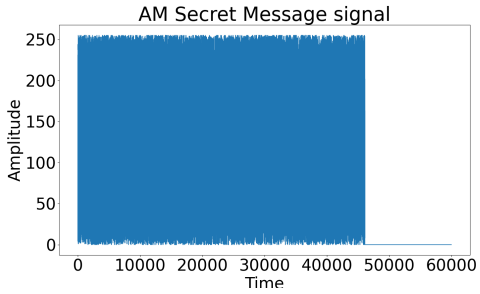


Fig. 11. Modulated Secret message

Figure 11 above show the secret message's signal after it has been amplitude modulated with a high frequency sinusoid of 20kHz.The information in the secret message is now carried on the envelope formed by the modulated signal.

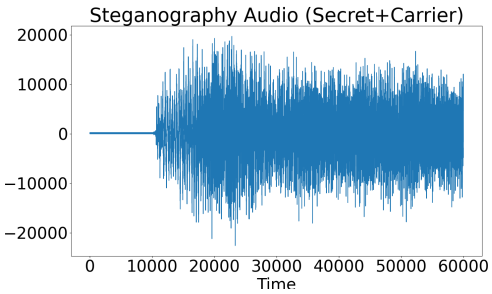


Fig. 12. Steganography Audio

Figure 12 above shows the combined signals: the modulated secret and the original carrier audio. The audio was exported to a wave file in the audio folder of the repository and plays the original audio as well as some high pitched buzz in the first two seconds of the audio. The buzz can be attributed to speakers being unable to process the sound beyond the 20kHz audible range.

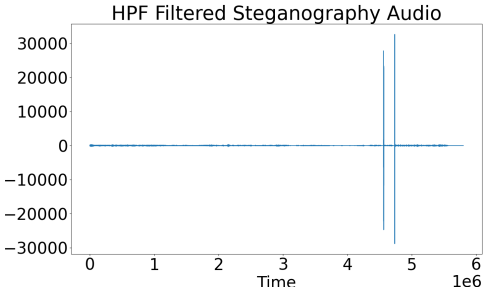


Fig. 13. Filtered Steganography Audio

The figure above shows the first step in the decoding process where the steganography audio is passed through a high pass filter to capture the modulated secret audio that is above 20kHz.Most low frequency signal components are removed at this stage.

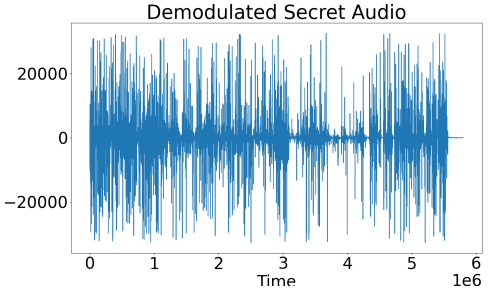


Fig. 14. Demodulated secret message signal

Above figure shows the demodulated secret message after it has been extracted from the filtered signal. An envelope detector could have been used but since we knew the carrier wave's characteristics, it was used to demodulate the signal.

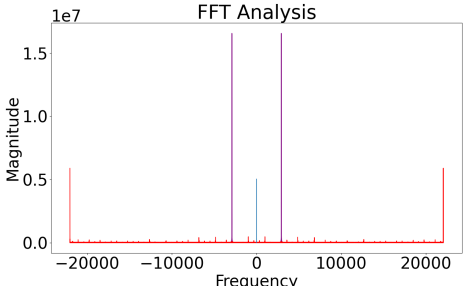


Fig. 15. Frequency Domain analysis

Above is the frequency domain analysis of the various signals. It shows the secret audio in green, the carrier wave in purple and finally the carrier audio in red.

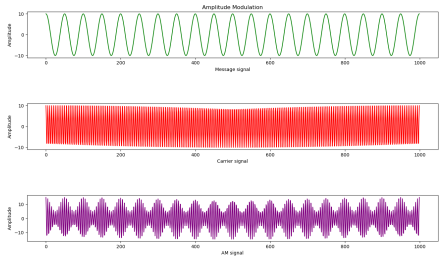


Fig. 16. Amplitude modulation

Above is a slight digression from the steganography but shows the amplitude modulation technique we used t shift the frequency of our audio wave to the desired frequency range. The modulating signal's information will be carried on the amplitude of the resultant wave though at a different frequency.

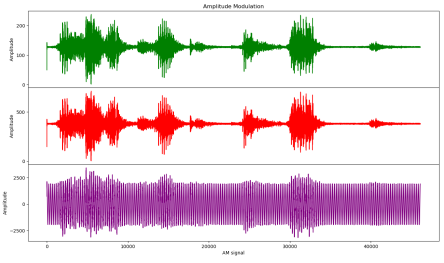


Fig. 17. Secret Message AM

The figure above shows the secret message in green, then the amplitude modulated secret message in purple. The purple signal is then demodulated and the red signal is recovered. This is the essence of the section of code that takes on from

the filtered steganography audio and results in the original secret audio being recovered. Regarding modulation, the purple signal's amplitude corresponds to the secret message audio's magnitude. By visual inspection, the two audio waved can be seen to be identical.

VIII. Conclusion

The objective of this project was to develop a method for transmitting secret messages within an audio file using Audio Steganography. While many of the common methods were heavily researched and explored, it was decided to tackle a lesser known method involving hiding messages within the high frequency bands of a cover audio file using amplitude modulation. A python script was successfully built in order to obtain a golden measure prior to the program being potentially improved and parallelized. While adequate results were obtained in terms of both timing and signal output, the secret audio was not fully and clearly recovered.

It is possible that the implementation of amplitude modulation instead of frequency modulation might have introduced some errors, but further and ample development time would be required to explore this as a potential solution. Another solution might involve passing the demodulated signal through a low pass filter in order to remove the additive noise from the encoding process, thus clarifying the audio and helping it more closely resemble the signal of original secret audio.

To make it a stand-alone device that utilizes parallelism discussed in the design section, the MOJO FPGA is used to decode a received signal. The python golden measure implementation is converted to Verilog.

Further Improvements and Recommendation:

- Addition of a wireless link to receive encoded audio from the internet
- Addition of a microphone to include sending of an encoded audio
- Using an additional audio channel in the wave file to include other encoding formats eg.images, text
- Encoding audios in real-time such that they can be streamed and decoded on the receiving end concurrently

IX. References

[1] https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1305/context=other

[2] Purves D, Augustine GJ, Fitzpatrick D, et al., editors. Neuroscience. 2nd edition. Sunderland (MA): Sinauer Associates; 2001. The Audible Spectrum. Available from: https://www.ncbi.nlm.nih.gov/books/NBK10924

[3] https://flylib.com/books/en/2.729.1/highpassfirfilterdesign.html

[4] Rosa, V.S.; Costa, E.; Bampi, S. A High Performance Parallel FIR Filters Generation Tool[J]. IEEE International Rapid System Prototyping, 2006, 6: 14-16

[5] Proakis. J.G., Manolakis .D.G, 'Introduction to Digital Signal Processing', McMillan Publishing, USA, 2003.

[6] Khan, Shoab. (2011). Unfolding and Folding of Architectures. 10.1002/9780470974681.ch8.

[7] Podgórski, Andrzej / Nedwidek, R. / Pochmara, M.. (2003). Implementation of the USB interface in the instrumentation for sound and vibration measurements and analysis. 159 - 163. 10.1109/IDAACS.2003.1249539.

[8] Jayaram, & Ranganatha, & Anupama,. (2011). Information Hiding Using Audio Steganography - A Survey. The International journal of Multimedia Its Applications. 3. 86-96. 10.5121/ijma.2011.3308.

[9] Djebbar, Fatiha & Ayad, Beghdad & Hamam, Habib & abed-meraim, Karim. (2011). A view on latest audio steganography techniques. 2011 International Conference on Innovations in Information Technology, IIT 2011. 10.1109/INNOVATIONS.2011.5893859.

[10] https://www.clear.rice.edu/elec301/Projects01/smokey_steg/freq.htm Accessed: 23/04/2021