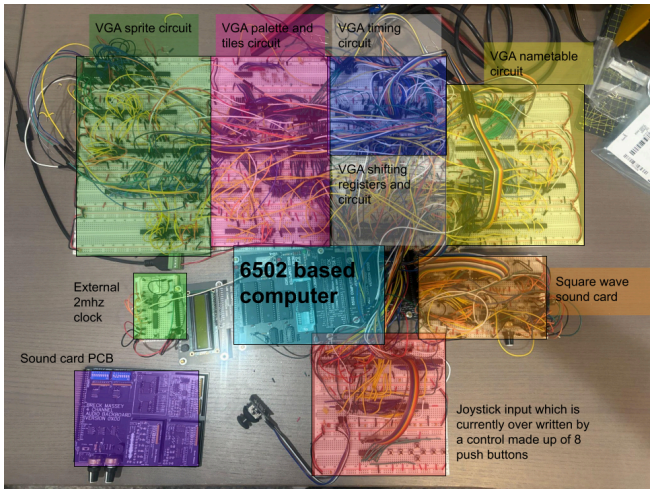
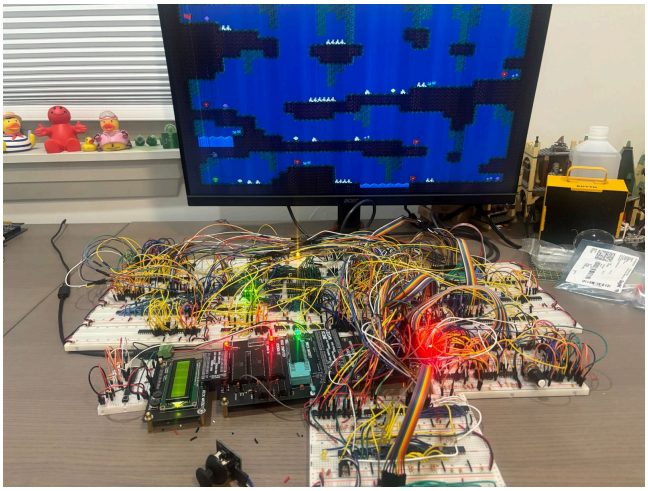
Building a Computer to Understand Their Inner Workings

Breck Massey
Advisors: Todd Massey, massey@scubareviews.com, Dr. Hunter Adams, vah3@cornell.edu
April 2, 2024





I. Abstract

While computers surround us, knowledge on how they work is limited. The purpose of this research project is to design an 8-bit computer to gain an understanding of how computers work. The first home computers with microprocessors started to appear in the 70s and 80s. One of the main chips that led to this revolution was the 6502 microprocessor. The computer design will rely on this chip. The goal of the project was to make a functioning computer similar in power to the computers from the 70s and 80s. The stretch goal was to add a video and sound output. The final project ended up being a working 6502-based computer with a VGA graphics card and a 4-channel sound card. The VGA graphics card has a 320 by 240 pixel 60 Hz output with 256 different output colors. It is similar to the NES where the graphics are made up of 8x8 pixel tiles (Parent, 2019). Unlike the NES, the graphics have 4 times the colors but it has support for only one sprite (NES, 2015). Similarly to the NES, my computer has an audio system with one

sound channel.

Keywords: 6502, 65c02, microprocessor, VGA graphics card,

II. Introduction

The purpose of this project is to gain an understanding of computers by designing my own 8-bit computer/console. Computers are used to control almost every aspect of our lives from our phones to air conditioning to our cars. Even though this technology surrounds us every day, almost no one knows how it works. Many programmers write code to communicate with computers as their job but have little knowledge of how the code gets executed. While many people may understand what basic components do such as the Central Processing Unit (CPU), Random Access Memory (RAM) and the Graphics Processing Unit (GPU), they do not understand how those components function. The goal was to pull back the veil hiding the inner workings of these complex machines by making a computer. This is especially important in the modern day as Moore's Law dies. Moore's Law predicts that every two years the number of transistors (the component that does the calculations) in computer chips will double. Our current method of shrinking transistors is helped back by the fact that electrons can jump through transistors causing chips to act sporadically. An overall goal would be to gain an understanding of electronics to find a solution to Moore's Law so technology can keep progressing at a rapid rate. To keep the project feasible it would be similar to computers from the 70s and 80s such as the Apple II, Commodore 64, and the Nintendo Entertainment System (NES). All of these computers use the 6502 microprocessor. A CPU is a type of microprocessor. The final goal was to build a computer with a stretch goal of having sound and video output so it could run a custom-programmed video game. The result is a computer that runs at 2 MHz, as well as a custom built Video Graphics Array (VGA) graphics card that can produce 256 colors on a 320 by 240 pixel screen at 60 Hz. In addition, it has one audio channel to produce a square wave sound.

III. Research

In 1975 MOS Technology released the 6502 microprocessor (Dudley, 2020). It led to a home computer revolution as it was used in many famous computers like the Commodore 64, Apple II, and the NES game console. As I wanted to design a computer and a console, most of the research focused around the NES. While the NES did not use a 6502, the microprocessor it did use, the Ricoh 2C02, was the same as the 6502 with added audio capabilities (Martin, 2022). The 6502 is an 8-bit microprocessor with a 16-bit address. This means the 6502 does calculations and transfers data 8 bits at a time. The 16 bit address space means it can access up to 65536 unique indices or RAM or ROM. RAM will hold any values the computer writes to it as long as it has power. This is useful and needed for keeping track of user inputs or the current state of the program. From this data the program can make certain decisions. For example, you could store the position of a video game character in RAM. The program could then look at this data and update it based on inputs from a controller. ROM cannot be written to by the computer, instead the computer can only read the data coming from it. ROM also saves its data even when the computer loses power. This means it is perfect for holding the program and data. Once the program is perfected the computer will never need to change it. Instead, a computer will need to see what steps it needs to take from the program.

Playing games on a game console is very difficult if there is no screen. The NES outputs a 256 by 224 pixel signal with up to 64 predefined colors (NES, 2015). The NES uses 8x8 tiles of pixels to make up the screen. Each tile can have up to 13 different colors. Furthermore, the NES can display up to 64 different sprites on the screen at once (Parent, 2019). A sprite is an 8x8 set of pixels that can be moved around the screen without having to snap to the 8x8 tile grid for the background. The NES uses the composite video standard to communicate with the TV. In an effort to keep my computer simpler, I went a different route than the NES. My graphics card uses the VGA standard. The reason the VGA standard was easier is because it uses digital signals instead of analog signals. Digital signals are either on or off but analog signals can be anywhere between on and off. Because most computers are digital like the one designed for this project, it was easier to stay digital instead of having to deal with analog signals. Modern video signals such as HDMI have switched to be fully digital.

IV. Materials and Methods

To design my computer I used a variety of different software and hardware tools. The first step of the design process was to research the basic functionality of a certain circuit. Then based on my research I came up with

the requirements for a certain circuit. Then I came up with a block diagram of the circuit on paper. I made changes to this design so that I could build it with the components I had or could easily buy. This sometimes changed the requirements. After I had the design I made it on breadboards. Breadboards make it easy to quickly test certain circuits to see if they work. After I built the circuit I had to debug it. I started off by writing a program that would use the circuit to produce a specific result. The program was used to test if the circuit works. If the circuit did not work I used tools such as an oscilloscope and a logic analyzer to look at the signals in the circuit. I compared these signals against the expected signals to figure out where the problem was. Once I finalized the working circuit design on breadboards, I designed a Printed Circuit Board (PCB). I used a free online editor, EasyEDA, to draw out the schematics and the layout of the board. Then I used JLCPCB to produce the PCBs. I used DigiKey to supply my parts. After two weeks I received the board and parts and soldered the components onto the boards. After finishing the boards I tested the capabilities similarly to how I tested the capabilities of the circuit on the breadboard. For the most part I used 74x00 series logic chips. These can vary from logic gates to counters and registers. For the main computer I used the 65c02 for the microprocessor. It is an upgraded version of the 6502 which was used in many computers in the 80s including the Apple II, the Commodore 64 and the NES. The 65c02 has extra instructions and can run faster compared to the original (The Western Design Center, Inc., 2018). Furthermore, I used a TL866 II Plus EEPROM Programmer to write my code into the read only memory (ROM) on my computer. To write the programs I used Visual Studios 2019 and 2022. The pixel art was made using Piskel App and Aseprite. To debug my computer I used an online simulator, tejotron.com, to make sure my code was executing as expected. Once I knew the code worked I used the Siglent SDS 1202X-E oscilloscope to check the quality of signals throughout the computer. Since the oscilloscope could only check two signals at once I needed to use a KingstVIS LA1010 logic analyzer to check up to 16 signals at once. This made it easy to track data traveling through the computer. The oscilloscope and log analyzers are very useful to track signals since my computer's graphics card can change signals up to 25 million times per second.

V. Results

The final product of my computer consists of the main computer, a VGA graphics card, and a 4-channel sound system.

A. The Main System

The actual computer part consists of 4 main chips along with 2 supporting chips. The design is a modified version of Ben Eaters 6502 breadboard computer that allows for memory mapped input and output (Eater, n.d.). The computer uses a 65c02 microprocessor along with a 6522 which is meant to be paired with the 65c02. The 6522 allows for additional functions such as 16 IO ports, timers, and interrupts (Eater, n.d.). The computer has 16384 bytes of RAM, and 32768 bytes of Read Only Memory (ROM). RAM is used to store temporary data like a player's position or score. The ROM is used to store the program or different data such as graphics. Sections of the RAM can be overwritten by different IO ports; this is known as memory mapped IO. The layout of my computer's address space is shown in Fig 1.

Address (in hex)	Components using Address
0x0000-0x00ff	RAM (Also used by IO)
0x0100-0x01ff	RAM - Where 65c02 store stack
0x0200-0x3FFF	RAM
0x4000-0x5FFF	Empty
0x6000-0x6FFF	6522 IO chip
0x7000-0x7FFF	Empty
0x8000-0xFFFF	ROM/program

Fig 1. Shows the address layout for my computer. Empty means no device uses that area of the address.

Memory-mapped IO makes it very simple to add additional components that need to communicate with the main circuit by checking to see if the address equals a predefined value. The computer has a 1 MHz crystal as the default clock but it also has the ability to switch to an external clock. This allows me to run the computer at faster speeds if needed or slow it down to make it easier to debug. The system can currently run at 2 Mhz before it starts causing errors. Without the VGA circuit attached, I

have been able to run it at 6 MHz. The 65c02 is rated for 14 MHz but my system has too much noise, especially with the VGA circuit attached, and the ROM used in the computer is extremely slow. The main computer is built upon a 2 layer PCB using only through-hole components shown in Fig 2.

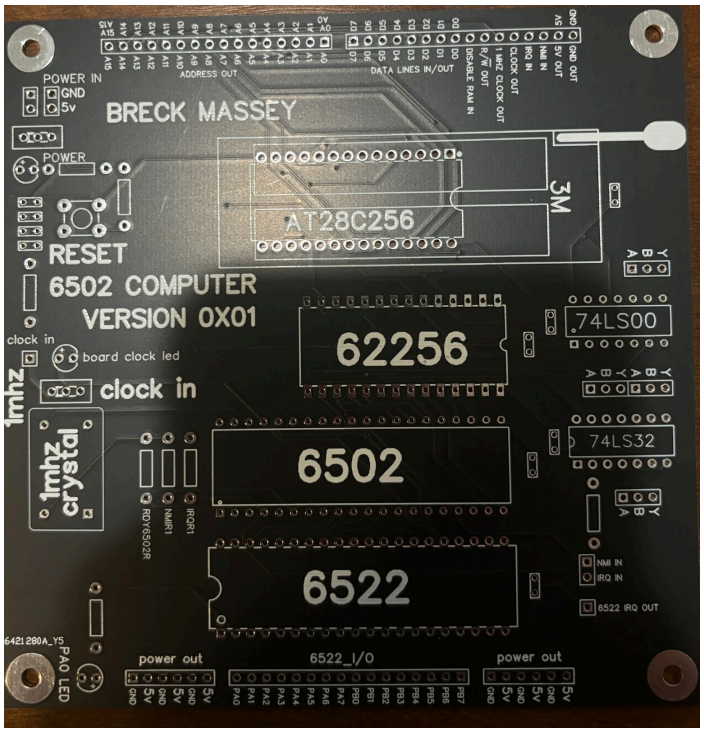


Fig 2. Shows the main PCB

B. VGA Graphics Card

The VGA graphics card was heavily inspired by the NES's graphics capabilities. The main components of a graphics card similar to the NES is timing, shifting, the pallet, the name table, the tiles, and sprites.

1. Timing

The NES has a screen resolution of 256 × 240 pixels (Parent, 2019). While this resolution works, I decided I wanted to have a higher screen resolution of 640 by 480. Unfortunately, my VGA graphics card had a final resolution of 320 by 240 pixels. While my graphics card has the same Y resolution it has a slightly different X resolution. The reason for the difference in X resolution was because of the different video standards. The NES uses composite video while my circuit uses VGA. One of the VGA standards is 640 by 480 pixels. I wanted to go with this high resolution because I could add a lot of detail to my video output but my circuitry was a bit noisy and made it hard to see fine details as demonstrated in Figure 3.

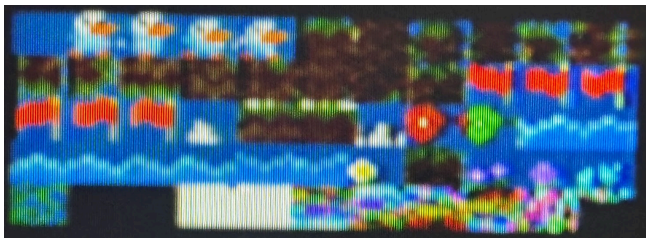


Fig 3. Shows a tileset on the 640 by 480 pixel mode. Detail is lost in the Duck Character and some of the block tiles.

This meant it was hard to get a clean signal for every pixel; instead, I divided the resolution down by 2 causing it to go from 640 x 480 to 320 x 240. By chance, I had the same Y resolution of 240 pixels. Since the NES used composite video and my computer used VGA, the two graphics cards have different timing. My card uses the 640 by 480 VGA standard for timing. This means my graphics card outputs a resolution up to 640 by 480 pixels at 60 frames per second. For the TV to understand what the graphics card is trying to output, the card has to feed certain signals to the TV. The two main signals needed to synchronize with the TV are Horizontal Sync (h sync) and Vertical Sync (v sync) (ProjectFpga.com, 2023). The TV draws from left to right and up to down. The h sync and v sync signals tell the TV where in the drawing cycle the graphics card is. Both signals start off at 5 volts (high) during the visible area and frog porch. Then at the sync pulse, the signals will drop to 0 volts (ground/low) (ProjectFpga.com, 2023). This is the signal that the TV will synchronize off of. Finally, they will go back to high for the Back porch. Then the cycle repeats. Figures 4 and 5 specify the timing needed for each section of the signal.

Horizontal timing (line)

Polarity of horizontal sync pulse is negative.

Scanline part	Pixels	Time [µs]
Visible area	640	25.422045680238
Front porch	16	0.63555114200596
Sync pulse	96	3.8133068520357
Back porch	48	1.9066534260179
Whole line	800	31.777557100298

Fig 4. Shows the timing in microseconds for each section of the h Sync signal for a 640x480 60 Hz VGA signal. The signal should be low during the Sync pulse and High otherwise (VGA Signal, n.d.).

Vertical timing (frame)

Polarity of vertical sync pulse is negative.

Frame part	Lines	Time [ms]
Visible area	480	15.253227408143
Front porch	10	0.31777557100298
Sync pulse	2	0.063555114200596
Back porch	33	1.0486593843098
Whole frame	525	16.683217477656

Fig 5. Shows the timing in milliseconds for each section of the v Sync signal for a 640x480 60 Hz VGA signal. The signal should be low during the Sync pulse and High otherwise. (VGA Signal, n.d.).

To get this timing my VGA graphics card has two similar circuits for both the h sync and v sync signals. The h sync signal uses a 25.175 MHz clock signal to count up 2 8 bit counters. These counters are then fed into the address of an EEPROM. The EEPROM is programmed to output the h sync signal based off of the current address of the counters. It also serves as the clock for the 2 v sync counters. Similarly these counters feed into the address of another EEPROM which outputs the v sync signal. The two EEPROMs also output a variety of other signals to synchronize other parts of the VGA circuit such as the pixel counters. The pixel counters serve to keep track of the X and Y positions of the image being drawn. When the frame starts both the X and Y pixel counters are set to zero. Then X will count up until the end of the screen and reset back to zero. This will then trigger the Y counter to count once. This happens many times until the Y counter reaches the bottom of the screen and resets itself. These two counters are then fed into other parts of the circuit so the graphics card knows where to grab data and make the correct image appear.

2. Shifting

While these pixel counters can simply be used as described above they can also be used to shift the image on the screen around. While I said the counters reset back to zero, the counters reset to a value stored into the VGA shift registers which are controlled by the program. Fig 6. shows the address corresponding to the VGA shift registers.

Address (in hex)	Register
0x0010	X shift register low byte
0x0011	X shift register high byte
0x0012	Y shift register low byte
0x0013	Y shift register high byte

Fig 6. Shows 4 addresses corresponding to the VGA's shift registers.

One common use for shifting in the NES was travel through level. This can be seen in many 2d platformers such as the original Super Mario Bros. To demonstrate the capabilities of the shift registers I took a static image of the Statue of Liberty (Fig 6) then as the image was being drawn I shifted the individual rows using a sine wave look up table. This creates a wavy effect, demonstrated in Fig 7.

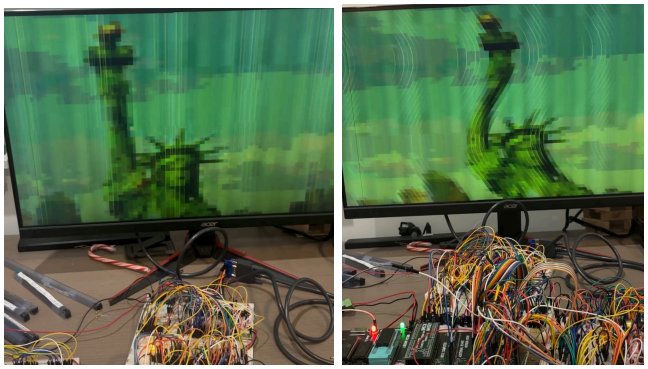


Fig 7 shows the VGA circuit outputting an image of the Statue of LIberty. Fig 7. Shows the same image of the Statue of Liberty with each row of pixels shifted based off of a sine wave.

3. Pallet

While my graphics card is designed after the NES's graphics, I felt that the Pallet circuit could be vastly improved. The NES had the ability to display 64 predefined colors shown in Fig 8.

Fig 8. Shows the 64 colors that the NES is able to produce (Parent, 2019).

I improved the circuit by giving my computer the ability to produce 256 colors, four times more than the NES shown in Figure 9.

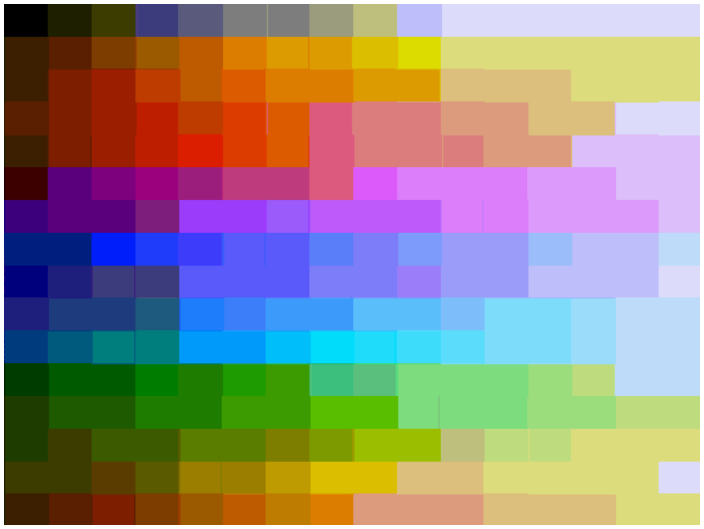


Fig 9. Shows the 256 colors that the my computer can produce is able to produce.

In both my computer and the NES there is the pallet circuit which has an array of colors that are chosen from by the program. In the NES this array is predefined and does not change. This means the first color will always be gray (as seen in fig 9). In my computer any color can be assigned to any spot in the array. This means the first spot could be gray, red, blue, green, or anything between. The user can specify the color they want by writing to address 0x17 for which index they want to change, then write to 0x18 to change the color in that index. This allows users to quickly change the color of many things across the screen with two write functions. Comparatively the NES would have to change every pixel in tile/sprite to change the color of everything at once.

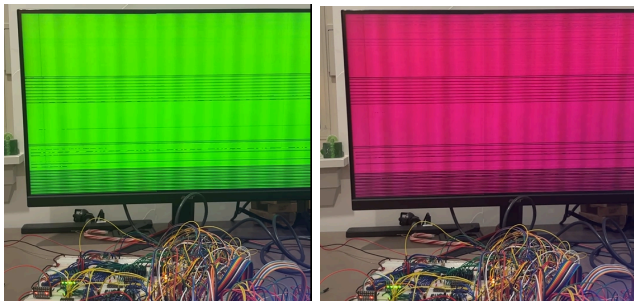


Fig 10. Shows the whole screen turning from green to pink in one frame after changing one index in the Pallet.

4. Name Tables and Tiles

To produce an image on the screen the graphics card has to come up with the colors of the pixels for all 320 by 240 pixels or 76800 bytes of data per screen. This is more RAM than my graphics card could have. Another solution to split the screen into 8x8 sections of pixels. These 8x8 sets of pixels are tiles. This would lower the screen resolution of 40 by 30 tiles or 1200 bytes per screen. This is a significant decrease in RAM usage. The name table RAM holds the index to each tile that makes up the screen as demonstrated in figure 11.

```
17,13,13,13,13,13,13,13,13,13,13,13,13,13,13,16
10,00,00,00,00,00,00,00,00,00,44,00,00,00,00,08
10,00,00,00,42,00,00,41,00,00,44,00,00,19,00,08
10,00,00,07,40,40,40,40,09,00,44,00,07,18,05,14
10,00,00,11,13,13,13,13,12,00,44,00,11,06,06,06
15,09,00,00,00,00,00,44,00,00,00,00,00,11,06,06
06,10,00,39,00,00,00,44,00,00,00,00,00,00,11,16
06,15,05,40,05,09,00,00,00,00,00,29,00,00,00,08
17,13,13,13,16,15,31,31,31,05,05,05,09,00,00,08
10,00,44,00,11,13,13,13,13,13,13,13,12,00,07,14
10,00,44,00,00,00,00,00,44,44,00,00,00,00,11,16
10,00,44,00,00,00,00,00,44,00,00,00,25,25,00,08
10,00,44,00,00,00,00,00,00,00,07,40,26,26,05,14
10,00,00,00,00,00,00,00,00,00,07,14,01,02,03,04,06
10,00,30,00,28,28,00,43,07,14,06,02,06,06,06,06
15,05,05,05,27,27,05,40,14,06,06,06,06,06,06,06
```

Fig 11. Shows an example of a name table.

The 7 bit value in the name table feeds into the tile RAM. The tile ram holds the data for what each tile looks like as seen in figure 12.

Fig 12. Shows an example a set of tiles

When the tiles are placed onto the screen based off of the name space then an image is formed, shown in figure 13.

Fig 13. Shows an example of a name table.

The computer allows for 128 unique tiles. Unlike the NES, the tiles are not limited to a certain number of colors.

C. Sound

Originally the goal of this computer was to have a sound system similar to the NES. The NES has 5 audio channels which include 2 pulse wave channels, a triangle wave, a noise channel, and a Delta Modulation Channel (DCM) channel (APU, 2023) (Copetti, 2019). The sound card was built to support up to 4 channels at once. On the board, different channels could be swapped out. My design allowed the user to choose from 3 different channel types, square wave, sawtooth, and analog output. Unfortunately, when the audio card is plugged in, one of the chips (74als273) on the audio card's backboard heats up to extreme temperatures making the whole board inoperable.

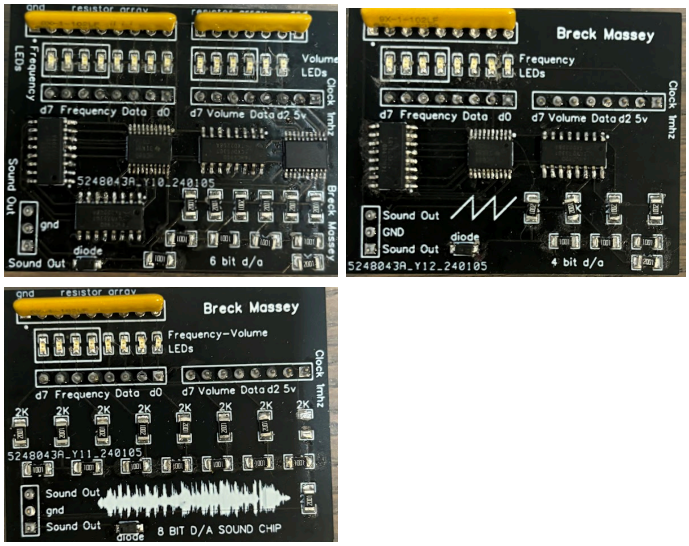


Fig 14. Shows the three different audio card types. Square wave in the top left, sawtooth in the top right, and analog in the bottom left.

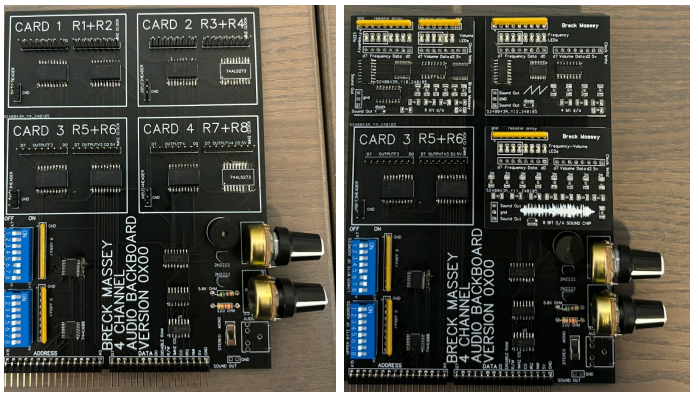


Fig 14. Shows the audio card backboard (left) and the audio card backboard populated with 3 channels (right).

Instead of using this whole sound card, the computer currently has a separate sound card that produces one square wave of any specified frequency.

D. Programming

Designing a computer is the process of crafting the hardware and software. The hardware and software have to work together to create any meaningful result. This project ran 6502 assembly code. Since the 6502 is an old microprocessor the only programming language available was 6502 assembly. Now there are assemblers that will convert C code into 6502 assembly code. I decided not to use these assemblers so that I would gain a better understanding of the hardware. The 65c02 has 70 different instructions (source datasheet). The different instructions are meant to transfer data throughout the system, do arithmetic operations, or jump to different parts of the code. The 6502 can do many calculations such as addition, subtraction, rotating data left and right, and logical operations. The data from these calculations can be stored to different registers in the 6502 or to external devices such as RAM. The results of these calculations could also be used to make decisions on whether the computer should jump to another part or not. For example, this can be useful if a player just lost all of their lives and now you need to call the death function. The program could use the BEQ instruction which will only jump to a function if the last arithmetic operation resulted in a zero. Over 2000 lines of code were written for the 2d platformer game which runs on this system.

VI. Discussion

The final result of this computer is that it is capable of running programs at up to 2 MHz, has a video output, and has 4 sound channels. In comparison to the NES my computer has a higher clock speed, 2 MHz vs 1.789773 MHz. Furthermore, in comparison to the NES my graphics card is capable of displaying 4 times more colors at a slightly higher resolution. Moreover, the NES is limited to a certain number of colors per tile and sprite while my computer has no limit. The NES does have circuitry for sprites while the only way to display moving sprites on my computer is through software. This means it is a lot slower to have moving sprites on my screen. This would be a major improvement to help facilitate the creation of games on my computer. When comparing sound the NES has one extra channel in comparison to my computer, 5 versus 4. Both the NES and my computer have an audio channel for playing audio clips. My D/A audio channel is a simpler system that can produce higher-quality samples than the NES. In the future or if I had more time, the first improvement to the system would be converting the VGA circuit to PCB. This will reduce noise in the system because the long wires and slightly unstable connections do not provide the best signals. This will also reduce the overall size of the graphics card. Because it reduces noise in the system, the system might be able to run at a higher frequency. To further improve the clock speed, the ROM needs to be replaced with a faster FRAM chip. They are functionally the same but the FRAM chip is faster and newer.

VII. Conclusion

In conclusion, the development of my 8-bit system has taught me the fundamentals of computers. I have learned about computer systems on a large scale such as how different components communicate. In addition, this project has taught me the smaller details like the flow of electricity and how basic circuit components can be built up into something more complex. I have also learned the important skills of designing my own circuits from scratch to fulfill different requirements while using only the available materials. The project has developed my skills in designing printed circuit boards with industry standards. This project has taught me how data structures work in modern computers. Now that the project is complete I plan to continue development on my 8-bit system and continue research on modern computers. Over all this project has given me a solid foundation for my future indevours in electrical engineering so in the future I might be the one to find a solution to Moore's Law and let technology progress past anything we ever thought possible.

VIII. Resources

"APU." NESdev Wiki, 10 June 2023, www.nesdev.org/wiki/APU.

Copetti, R. (2019, January 25). NES Architecture | A Practical Analysis. Retrieved from Rodrigo's Stuff website: https://www.copetti.org/writings/consoles/nes/

Dudley, Rory, "MOS Technology 6502 CPU Emulation" (2020). Computer Science: Student Scholarship & Creative Works. 2. https://jayscholar.etown.edu/comscistu/2

Eater, B. (n.d.). Build a 6502 computer. Retrieved from eater.net website: https://eater.net/6502 https://eater.net/6502

Martin, Michael. “Digital Sound Playback on the NES.” Bumbershoot Software, 3 Dec. 2022, bumbershootsoft.wordpress.com/2022/12/03/digital-sound-playback-on-the-nes/. Accessed 10 Nov. 2023. https://bumbershootsoft.wordpress.com/2022/12/03/digital-sound-playback-on-the-nes/

“NES Graphics – Part 1 | Dustmop.io Blog.” Dustmop.io, 28 Apr. 2015, www.dustmop.io/blog/2015/04/28/nes-graphics-part-1/.

Parent, Micheal. “NES Graphical Specs.” BitBeamCannon, 10 Oct. 2019, bitbeamcannon.com/nes-graphical-specs/#:~:text=The%20NES%20could%20store%20in. Accessed 10 Nov. 2023. https://bitbeamcannon.com/nes-graphical-specs/#:~:text=The%20NES%20could%20store%20in,some%20sprites%20missing%20from%20frames

“ProjectFpga.com.” Projectfpga.com, projectfpga.com/vga/. Accessed 10 Nov. 2023. https://projectfpga.com/vga/

The Western Design Center, Inc. (2018). W65C02S 8-bit Microprocessor. The Western Design Center, Inc. Retrieved from The Western Design Center, Inc. website: https://eater.net/datasheets/w65c02s.pdf

VGA Signal 640 x 480 @ 60 Hz Industry standard timing. (n.d.). Retrieved February 10, 2024, from tinyvga.com website: http://tinyvga.com/vga-timing/640x480@60Hz