California Polytechnic State University

Build-A-Board

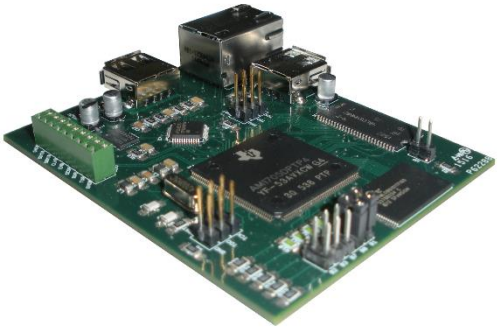

Figure 1: Build-A-Board Single-Board-Computer

By

Christopher Messerer

Brian Holland

Computer Engineering Department

College of Engineering

California Polytechnic State University

Date Submitted: June 7, 2016

Advisor: Dr. Bridget Benson

Abstract

The purpose of this project is to build a single board computer that will be able to run Linux and provide some expansion capabilities in the form of GPIO, USB, etc. This single board computer will have all the components that a fully functional computer has but with a much smaller form factor and overall capabilities.

Acknowledgements

First we would like to thank Dr. Bridget Benson for being our advisor for this process of completing our senior project. We would also thank her for all the she has given us in completing this project. Another person we would like to thank is Caleb Munsill for being a spring board for all the ideas we had about this project. To our friends and classmates that helped us get though all the classes to get to our senior project we thank you as well. Lastly, we would love to thank our family, without them we would not be here.

Table of Contents

List of Figures

List of Tables

Introduction

Circuit boards are at the heart of all modern day electronics. There has always been a need to connect different components to create a useful device, and it is foreseeable that this will remain the case even as more advanced system-on-chips are developed. This concept prompted this project idea – that is to create a single-board-computer that is capable of running a distribution of the Linux operating system in order to develop skills that will prove useful in industry and as hobbyists. This project not only reinforces many concepts that were learned through the computer engineering curriculum, but it also further advanced many of these skills.

There are already various affordable small form factor single-board-computers on the market today, many of which provide more capability than mainstream computers from less than two decades ago. Some popular examples of these boards come from Raspberry Pi, Beagleboard, and ODROID. Processors are a large part of what distinguishes boards. This project's board hosts an ARM9 series processor, and is by far the oldest architecture (ARMv5TE) when compared to even some older single-board-computers by the companies listed. For instance, the original Raspberry Pi uses an ARM11 device which is not only a newer architecture (ARMv6) but is also clocked much faster (700MHz base clock as opposed to 476MHz). The BeagleBone used a Cortex-M3 which is one of the earlier Cortex series microcontrollers, but is also still far more capable compared to Build-A-Board as it supports more 256MB DDR2 as opposed to 64MB SDRAM and is clocked at 720MHz. The BeagleBone is the most comparable in terms of connectivity, as both boards support one USB 2.0 On-The-Go ports and one 10/100 Ethernet port. Comparably, the Raspberry Pi 1 Model B+ and the ODROID-C2 have four USB 2.0 ports. This project starts getting left behind in capabilities when noting it has less GPIO and doesn't support video. Overall, this board would not ideally suit anyone looking for alternate single-board-computers as it was never designed to fulfill such a role. This project is merely a stepping stone so that its creators gain the skills to create even more capable designs in the future.

Requirements

The primary requirement of this project is to create a board that is capable of running any version of Linux. This requirement creates several sub-requirements that needed to be met for the board to be able to support Linux. Firstly, a distribution of Linux is required for the particular architecture of the processor selected. Secondly, as a majority of Linux versions require usage of external memory, a Memory Management Unit was required. These issues are further discussed in the hardware design section. **Table 1** shows all the requirements that this project needed to confirm that the project was a success. The table shows what category of requirement each requirement falls under. Other headers in the table are parameter description, the requirement itself, how the requirement will be met and the tolerance level that each requirement will be tested against. Also, the table shows what the total risk of each requirement failing. The risk levels are H – high, M – medium, and L – low.

Table 1: Requirements List with Related Tolerance, Risk, and Compliance Determination

CATEGORY	SPEC. NUMBER	PARAMETER DESCRIPTION	REQUIREMENT	TOLERANCE	RISK
PERFORMANCE	1	Operating System	Linux	uClinux	H
USABILITY	2	Peripheral Expansion	10 Peripherals	5 Peripherals	M
PERORMANCE	3	External Volatile Memory	SDRAM		L
DESIGN	4	Fully Designed in Eagle Software	Schematic in Eagle Software		L
PERFORMANCE	5	Crosstalk of Wire Trace Lines	20mV	50mV	M
DESIGN	6	Tape Out of Designed PCB board	Send Schematic to PCB Board Manufacturer	4 Boards Made	L
PERFORMANCE	7	Speed of Memory			M
PERFORMANCE	8	Speed of CPU	456-Mhz	375-Mhz	M
PERFORMANCE	9	Crosstalk of Memory Wire traces			H
USABILITY	10	Ethernet Connection	Ethernet Female Type		L
PERFORMANCE	11	ARM926EJ Processor	Texas Instruments AM1705		L
USABILITY	11	USB Connection to Interface to CPU	USB Female Type A		L
USABILITY	12	USB Connection For Power	USB Female Type Micro-B		L
DESIGN	13	Surface-Mount IC	All IC will be SMT	SMT	L
USABILITY	14	SPI interface to Processor	SPI		L
PERFORMANCE	15	Power			H
USABILITY	16	UART interface to Processor	UART		L
PERFORMANCE	17	External Non-Volatile Memory	NOR/NAND flash		L
USABILITY	18	Boot Options	UART, SPI, NOR/NAND		L
USABILITY	19	USB On-The-Go	Texas Instruments AM1705		M

Design

Hardware – Component Selection

In order to create a board that can run Linux, there are several considerations to take into account. As an operating system, Linux needs to have the ability to manipulate a sizable sum of program and system memory. Microcontrollers offer onboard memory, but not enough to effectively run an operating system. The most common way for Linux to access memory is through the Memory Management Unit (MMU) on a processor. This makes an MMU an important requirement when selecting a processor. There are alternative versions of Linux that can work without an MMU, such as uCLinux, however it was arbitrarily decided that preferentially a more common version of Linux would be supported (e.g. Debian).

Another major consideration was the physical packaging of the components that the board would use. Ball-grid-array (BGA) packages would prove difficult to inspect the solder joints and ensure electrical connectivity between the chip and the board. It also offers difficulties when considering board layout because the trace density can become very high, especially because the board fabrication houses that were being considered for their competitive pricing did not support blind or buried vias. Surface-mount (SMT) parts offered the ability to use somewhat modern technology while offering the ability to inspect, test, and fix any problems by hand. Through-hole processors as well as peripherals are generally far more architecturally archaic and thus will not support a modern operating system.

An ARM926EJ processor was selected after considering these sub-requirements for the following reasons:

- Supported Linux builds offered by Debian, Arch Linux, Fedora, and many others.
- Offered in SMT packages, and also supported SMT memory (SDRAM).
- Most recent processor architecture that met requirements.

The specific chip chosen was the Texas Instruments AM1705. This chip was chosen over others because if offered a larger pinout, requiring less multiplexing of pins to use different peripherals, and because TI offers a large amount of documentation for the processor. It was decided that important interfaces that should be supported by the board would be SPI, USB, UART, as well as both external memory interfaces for volatile (SDRAM) and non-volatile memory (NOR/NAND flash).

The processor offers several boot options, including SPI, UART, and NOR/NAND memory. After the processor is powered it checks several boot pins to determine what device to boot from. The board supports header pins to be used with jumpers to allow the user to boot from any of the supported options. The intended boot method is from the non-volatile memory which will be programmed with a supported distribution of Linux. Only particular flash memories are supported by the processor, so the memory was carefully selected.

AM1705 supports USB On-The-Go (USB-OTG) which allows it to act as a host, as well as a peripheral. This means devices can be connected to the board and it will initiate communication with them, acting as the host like a computer would, but when attached to a computer the board will act as a client. Since this protocol is supported by the processor no other peripheral components besides a USB port are required for its operation. Ethernet is also supported, however, unlike USB-OTG, the AM1705 does not

have the protocol fully integrated on the chip. Instead, the physical layer is left up to an external Ethernet PHY device.

Hardware – Board Layout

After all of the components were selected and connected in the schematic it was time to perform board layout. There are many considerations that had to be taken into account when performing board layout and this process allotted for most of the board's design time. Many concepts were researched before going about the board layout process, which is one reason for the layout process having taken so long.

While performing board design there are several considerations that need to be met depending on the components being used. One of the first questions that was asked was what should the correct trace width be in order to support the proper amount of current and handle the high frequency signals without creating antennas and causing crosstalk. The overall power requirements of each component need to be considered when creating supply and ground traces. Traces will inherently create resistance, and the less cross-sectional area in a trace then the higher resistance it will create. This becomes a problem when supplying power because large amounts of current can be flowing through the trace which can cause the trace to overheat and fail. Equations have already been derived to formulate the correct trace width which were used[1]: **Equation 1** finds the overall cross-sectional area (A) based on the current flowing through the trace (I), the temperature rise (t) and if the board is an internal or external layer (k, which is 0.024 for internal and 0.048 for external); and **Equation 2** finds the trace width (W) based on this area and the trace thickness (T) which is usually listed by the PCB fabrication house selected.

$$A = \left(\frac{I[\text{amps}]}{k \cdot t[^{\circ}\text{C}]^{0.44}} \right)^{1/0.725} \quad \text{Eq. 1}$$

$$W = \frac{A[\text{mils}^2]}{T[\text{mils/oz}] \cdot 1.378} \quad \text{Eq. 2}$$

Using these equations the supply traces can be designed to operate properly for any current. These equations also need to be considered when creating pads and through-holes because they have a fixed contact size. Making the exposed copper pad slightly larger to accommodate more solder is a solution to having a possible constraint with contact size. All supply traces were designed on this board for 1 amp which is more than any of the individual supplies might require except the 3.3V supply. This supply and ground have their own layers to handle larger currents. The AM1705's datasheet recommended it be designed to operate with a total supply current of 1 amp, this primarily shared between the 1.3V and 1.8V supplies, while the 3.3V supply is primarily for I/O and to operate peripheral components such as the Ethernet PHY chip.

When considering trace widths, their operating frequency also needs to be considered. The wider a trace the larger of an antenna it becomes, the more EMI coupling with neighboring traces. The solution is, where applicable, keep trace widths small and at least maintain one trace width distance between it and any neighboring traces. The farther traces are spread apart then the less interference is induced. For this reason it is important to avoid running traces parallel to one another that operate on different clocks. Signals that operate on the same clock will have interference at the moment the signal changes,

however, digital circuits have set rise and fall times that are not instantaneous so by the time the voltage is sampled it will have recovered. This is important because bus lines are typically run together in parallel; because their destination is the same and therefore it would be impossible to have no interference. Traces are still separated as much as possible to still eliminate as much coupling as possible. When communication lines that operate on different clocks or frequencies are nearby, it is even more important to reduce interference because it is now possible for the noise to become an issue. The best solution is for a perpendicular intersection between these traces on completely opposite layers, and any parallel sections that exist should be distanced as much as possible and on opposite layers as well. Perpendicular traces will not couple, and is therefore the best method to reduce crosstalk. The layer stack is also designed to reduce crosstalk between the signal layers. By having supply and ground layers between the signal layers (microstrip), the supply and ground planes act as shields for any crosstalk between the two signal layers. With the signal layers adjacent (stripline) there is greater coupling between the two layers. The design chosen was microstrip for this reason; it allows for different signals to be routed on different layers and thus reduces any crosstalk[2]. The use of vias on high speed signals was reduced as much as possible because they can potentially cause reflections and other unwanted interference[3]. Certain parts of the board needed to be as noise free as possible, such as the clock input to the AM1705 and the PLL power which is used to create the higher internal clock speeds for the ARM9 processor. To accommodate for this traces were routed around these components and any passing signals crossed perpendicular to prevent having a noisy internal clock.

Another important design consideration was to ensure differential pairs were length and impedance matched. A method of communicating with devices over long distances is to use a differential pair which has the same signal going through both wires, only one of them is inverted. Because differential pairs are tightly coupled, any interference will be picked up by both wires, and can then be subtracted to create the original signal without any noise. This method relies on the signals being the same length so they reach their destination at the same time, and not out of phase with each other[3]. The USB and Ethernet signals, both of which use differential pair signals, must therefore be carefully laid out so each signal pair has the same signal length. Pairs are routed together to ensure they have the same number of vias if required, and to help in matching their signal lengths. Eagle provides a tool to measure trace lengths, and also can meander a signal to obtain a particular length. This is only match's lengths geometrically and might not provide the exact signal length, but it effectively makes the signals close enough for the distances they are routed on this board. Meandering can change the impedance of the signal, so instead the traces are length matched by hand.

Buses that operate on a given clock also needed to be maintained within certain distances. Common PCB design will have the clock as long as the longest signal to ensure the signals have all reached their destination within the access time from the clock edge. The method used while designing this board was to calculate the maximum difference in lengths between clock and the signals. **Equation 3** shows the simple ballpark calculation performed, where propagation speed of the signal (S) is the speed of light, and the access time (t) is 5ns from the memory's datasheet.

$$D = \frac{S}{t} = \frac{3 \cdot 10^{8}[\mathrm{m/s}]}{5 \cdot 10^{-9}[\mathrm{s}]} = 1.5\mathrm{m} \qquad \text{Eq. 3}$$

Since the actual routed difference in length on the memory bus line is only about 15mm, all the signals will be latched corrected. This calculation becomes more important for larger boards with higher speed communication where signals can be routed farther and the access time after the rising edge of the clock can be much smaller.

Cost was another important consideration when designing the board. For example, it was originally going to have mounting holes on the corners, however, they were omitted to try and minimize board area as it is factored into the cost by many PCB manufacturers. As this was a prototype it was felt the cost tradeoff was more important. The board could have also been given more functionality, for example more GPIO pins could have been added as well as a USB hub to allow for more USB ports on the board. These contrasted with the goal of keeping costs down because they would have increased the size of the board layout and added parts that weren't necessary to reach the end goal.

The complete parts list is shown in the table **Appendix A**.

Hardware – Assembly

The PCB needed to be assembled with components after it was fabricated. The easiest method for soldering the fine pitch of the surface-mount components was to use a reflow process. This involved applying a stencil over the PCB that only exposed the surface-mount pads and then spreading solder paste across the stencil. Once the stencil was removed, the once bare pads were now covered with solder paste. Components were then placed on the board, which was a simple process as all components were labeled with an indicator allowing easy reference to the schematic. Once the surface-mount parts were all placed, the board was placed in a reflow oven that had its reflow profile set to match that of the solder paste being used. The next procedure was to solder through-hole components which was done by hand.

Software Installation

To install the Linux operating system on the board many steps are need to accomplish this. The board and CPU do not come with any software installed in them. The CPU doses not even come with a boot loader installed and that is why it needs to be one of the first items to be done when installing an operating system.

There are a few tools that are needed to complete the install of the O.S... First is the SDK for the AM1705 processer. This has all the images and User Boot Loader (UBL) needed to make the AM1705 work. The second tool needed is CCStudio. This is used to install the UBL and run the NAND flash writer. The compiler tool set used to build the U-Boot is in CodeSourcery (if needed).

Requirements

1. Building and running the O.S. for the computer requires both a Windows and Linux machine.
2. The Windows computer must be able to install and run CCStudio 3.3 or 4.1. The program will build the User Boot Loader (UBL) and Flash writers. It will also be used to burn the boot images (UBL, U-Boot) into the flash using the flash writers found in the Linux EZ Software Development Kit.
3. Debian Linux system will be used to compile the U-Boot and Linux Kernel.
4. It will also be used to host the TFTP server required for downloading the kernel and file system images from the U-Boot using Ethernet.
5. CodeSourcery tools for the Linux host to help build the U-Boot (if needed).

Step 1: Download Texas Instruments Linux EZ Software Development Kit for AM1705 Processor

Texas Instruments has a development kit for their AM1705 processors. It is downloaded from their website for free.

Step 2: Download and install CCStudio

Download and install Code Composer Studio 4.1 from Texas Instruments on the Windows computer.

Step 3: Download and install CodeSourcery

1. Download and install CodeSourcery tools to the Linux computer.
2. Go the CodeSourcery web site and download the Sourcery G++ Lite 2009q1-203 for ARM tools
3. Download the IA32 GNU/Linux Installer package to your Linux workstation.
4. Make the downloaded file executable
    a. chmod +x arm-2009q1-203-arm-none-linux-gnueabi.bin
5. Run the installer
    a. For Debian use: sudo dpkg-reconfigure -plow dash
    b. ./arm-2009q1-203-arm-none-linux-gnueabi.bin
    c. Install ia32-libs:
        i. $ sudo apt-get install ia32-libs
    d. Make sure the Code Sourcery tools are in the path by:
        i. $ export
            1. PATH=/home/<user>/CodeSourcery/Sourcery_G++_Lite/bin:$PATH
    e. $ source ~/.bashrc

Step 4: Booting the User Boot Loader (UBL) and U-Boot

1. Set the pin position to NAND on the board.
2. Start CCStudio and connect the board with the USB.
3. Run the GEL function Setup_EMIFA_PinMux() in the Scripts menu in CCSv4.
4. Load the NAND flasher tool on the ARM. Use the pre-built binary in the images/utils/omap11x7/ directory of the PSP installation called nand_writer.out.
5. Run the NAND flasher. When the prompt for the input file type, enter armais and when prompted for the file path, enter images/boot-strap/am17xx/ directory of the PSP installation.
6. Run the NAND flasher again. This time enter uboot for the input file type and the path to the u-boot.bin file in the PSP installation.

Step 5: Installing the SDK Software

1. This is done the Linux computer
2. Copy the following files to a temp folder called /tmp.
    a. OMAP_L138_arm_#_#_#_#.tar.gz
    b. Bios_setuplinux_#_#_#.bin
    c. Xdctools_setuplinux_#_#_#_#.bin
    d. Ti_cgt_c6000_#.#.#_setup_linux_x86.bin
    e. Cs1omapl138_1_00_00-a_setup_linux.bin
3. Log into Linux and switch to “root”.
    a. $ su root
4. Change directory to the install folder from the download.
    a. $ cd /tmp
5. Change permission on the bin files
    a. $ chmod +x *.bin
    b. $ exit
6. Execute the ARM-side SDK

    a. $ mv OMAP_L138_arm_#_#_#_#.tar.gz /bhom/<useraccct>/
    b. $ cd /home/<useract>/
    c. $ tar zxf OMAP_L138_arm_#_#_#_#.tar.gz
7. Execute the XDCtools installer
    a. $ cd /tmp
    b. $ ./xdctools_setuplinux_#_#_#_#.bin
8. Execute the DSP/BIOS installer
    a. $ ./bios_setuplinux_#_#_#.bin
9. Execute the Code Generation Tools installer
    a. $ ./ti-cgt-C6000_#.#.#_setup_linux_x86.bin
    b. Path Information - When the installer prompts for an installation location, do not use the default location. Instead, use the entire path to the OMAPL138_arm_#_#_#_# codegen directory. You will need to manually create the folder "cg6x_#_#_#" (or equivalent folder name for the CGtool version).
    c. $ source ~/.bashrc
10. Execute the codec server installer
    a. $ ./cs1omapl138_1_##_##-a_setup_linux.bin.bin

Step 6: Setting Up the TFTP Server

TFTP server is need to be set up on the Linux computer so the Linux kernel images can be downloaded to the board.

1. Install atftp server on Linux
    a. $ sudo aptitude install atftpd
2. The atftpd needs to run as a server directly. So the /etc/default/atftpd file needs to be edited.
    a. $ sudo gedit /etc/default/atftpd
    b. Change the following line from:
        i. USE_INETD=true
    c. To
        i. USE_INETD=false
    d. Also change the directory at the end of the file to /tftpboot
    e. Save and Exit
3. Now run this command
    a. $ sudo invoke-rc.d atftpd start
4. Configure atftpd
    a. Create a directory
        i. $ sudo mkdir /tftpboot
        ii. $ sudo chmod –R 777 /tftpboot
        iii. $ sudo chown –R nobody /tftpboot
        iv. $ sudo /etc/init.d/atftpd restart
5. Install tftpd-hpa
    a. $ sudo apt-get install tftpd-hpa
6. Confiuge tftpd-hpa
    a. This will run TFTP as daemon by editing the /etc/default/tftpd-hpa file and replacing RUN-DAEMON=”no” with RUN_DAEMON=”yes”. Note: This will also define where the TFTP server is serving from.
        i. $ gksudo gedit /etc/default/tftpd-hpa
7. Start the TFTP service
    a. $ sudo service tftpd-hpa start

Step 7: Setting Up Linux Kernel Using U-Boot

Booting the kernel needs a target filesystem and a kernel image.

The first step to do is to set up TFTP download for use.

1. Connect the Ethernet cable to the board and computer.
2. Copy the kernel image to the /tftpboot directory on the Linux computer.
    a. $ cp /DaVinci-PSP-SDK-#.#.#.#/images/boot-strap/kernel/omap11x7/uImage /tftpboot
    b. $cp -a /home/<useracct>/OMAPL138_arm_#_#_#_#/DaVinci-PSP-SDK-03.#.#.#/images/fs/nfs.tar.gz /tftpboot
    c. $ tar xvfz nfs.tar.gz
    d. $ chmod 777 –R /tftpboot
3. Setup the EVM IP address
    a. $ U-Boot> setenv autoload no
    b. $ U-Boot> dhcp
4. Get IP address of Linux Computer
    a. $ /sbin/ifconfig
5. Set the IP address of the server
    a. $ U-Boot> setenv serverip <ip address of server>
6. Set the name of the image to be downloaded
    a. $ U-Boot> setenv bootfile <Linux kernel image file name>

Step 8: Booting from NAND Flash (ramdisk as rootfs)

1. Download the uImage and copy it to the NAND partition
    a. $ U-Boot> tftp 0xc0700000 uImage
    b. $ U-Boot> nand erase 0x200000 0x200000
    c. $ U-Boot> nand write.e 0xc0700000 0x20000 0x200000
2. Download ramdisk, and copy it to the NAND flash
    a. $ U-Boot> tftp 0xc1180000 /home/<useracct>/OMAPL138_arm_#_#_#_#/DaVinci-PSP-SDK-03.#.#.#/images/fs/nfs/
    b. $ U-Boot> nand erase 0x400000 0x400000
    c. $ U-Boot> nand write.e 0xc1180000 0x400000 0x400000
3. Set up the bootargs and bootcmd
    a. $ U-Boot> setenv bootcmd ‘nand read.e 0xc1180000 0x400000 0x400000; nboot.e 0xc0700000 0 0x200000; bootm’
    b. $ U-Boot> setenv bootargs mem=32M console=ttySc, 115200n8 root=/dev/ram0 rw initrd=0xc1180000, 4m ip=dhcp eth=${ethaddr}

Step 9: Turning on Board

Restart board and make sure the boot pins are set to NAND.

Reload into the board using the Ethernet cable.

Run first program.

Testing

Continuity

The PCBs needed to undergo some basic continuity testing in order to eliminate it as a possible problem in case the board did not work. The first part of this test was to use the boards as they were received prior to assembly where the components would be soldered on. By checking the board before mounting all of the components ensured that parts would not be wasted being soldered onto a defective board. A multimeter was used to detect if two points on the board were connected and the board layout was used to find traces that could have been potential problems. The smallest trace on the board was not the smallest trace the manufacturer could make, but it was worth checking these traces anyway. Another concern was shorts between close traces. Again, the minimum trace width was still wider than the board house was capable of. This test wasn't entirely necessary, especially because the next step was to perform the same test only after assembly. Ideally, when performing a test like this, a specially made pogo-board with simple LEDs to serve as indicators would prove extremely useful as probing each trace with the multimeter was exhausting. The final step was to ensure continuity after assembling the board with all of the components. The assembly step could have potentially shorts or pins that simply didn't connect to their pad. After probing each connection point to its end point(s) the testing was complete. The board did not have any breaks or shorts in any of the connections.

Cross Talk

The next part of testing the computer was to test the trace lines of the newly manufactured PCB boards from Advanced Circuits. The PCBs were taken to the Senior Project Lab to connect them up to an oscilloscope to measure the cross talk of the trace lines. The cross talk testing payed close attention to trace lines that were close together and had a 45^{o} angle in at least one of the two lines. The green lines in all the figures represent the voltage across one line at a frequency indicated in the figure. The orange line is the amount of voltage that crossed over to the other line.

To get a base line of how big the cross talk might be, the first test was done on two adjacent bus lines. Two adjunct bus lines should have the most cross talk. Cross talk is not an issue with bus lines because they are always on the same clock cycle. When the lines were tested some small cross talk did occur. **Figure 2** shows, that a small ±50mV spike accrued in the orange line when the green voltage changed from high to low voltage and from low to high voltage. The voltage of the green line in **Figure 3** was 1Vpp.

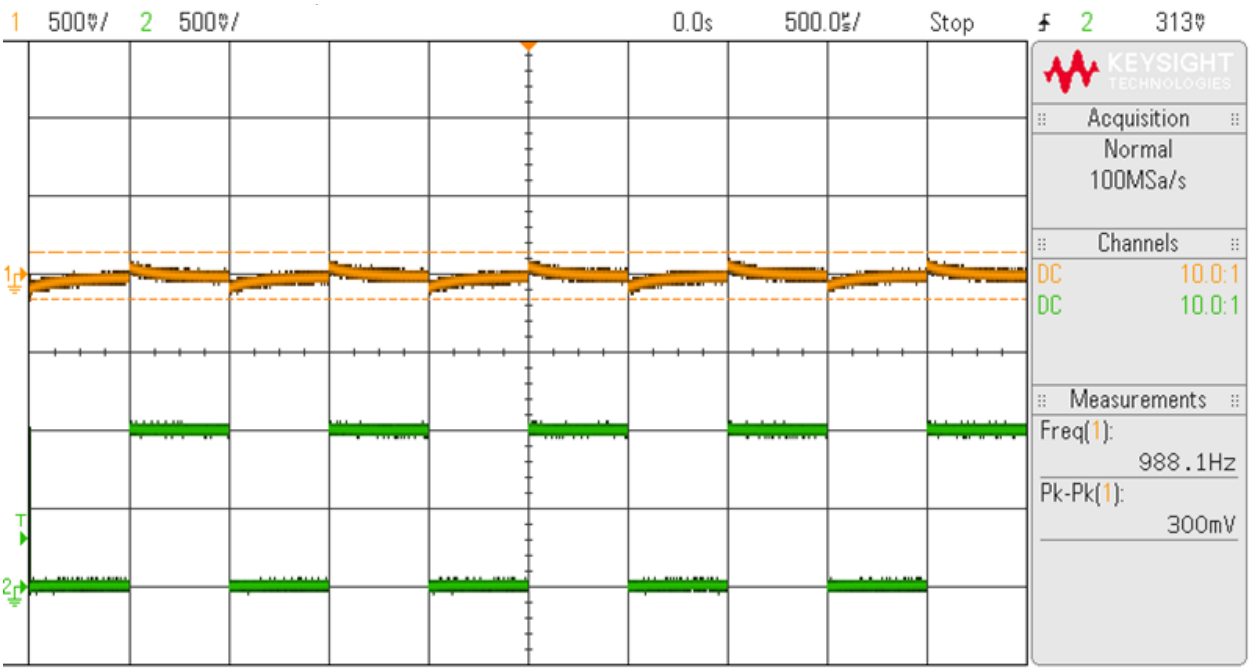

Figure 2: Cross Talk of Two Bus Lines

The next tested lines were the Clock Line and the Enable Line. As **Figure 3** shows, there was no cross talk between the two lines at 4Vpp. These lines were on two different layers but right next to each other but had no cross talk between them.

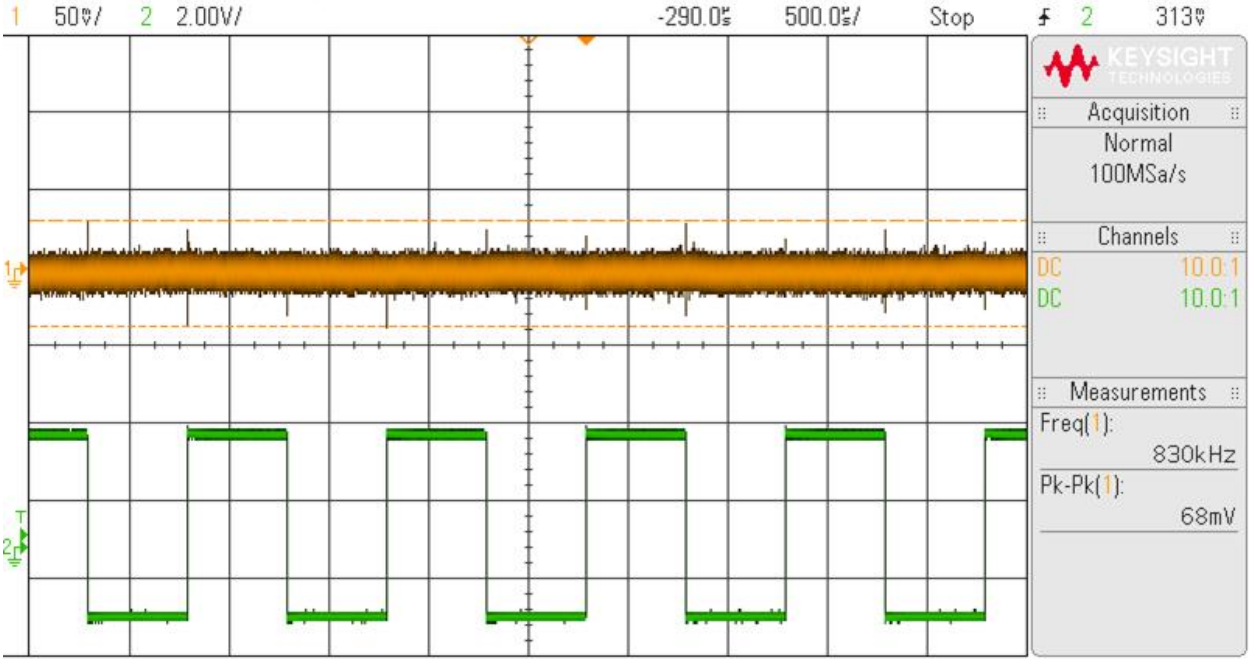

Figure 3: Cross Talk of Clock and Enable Line

As **Figure 4** shows there was no cross talk between the USB and the Bus lines at 1Vpp. These two lines were on the same layer and right next to each other.

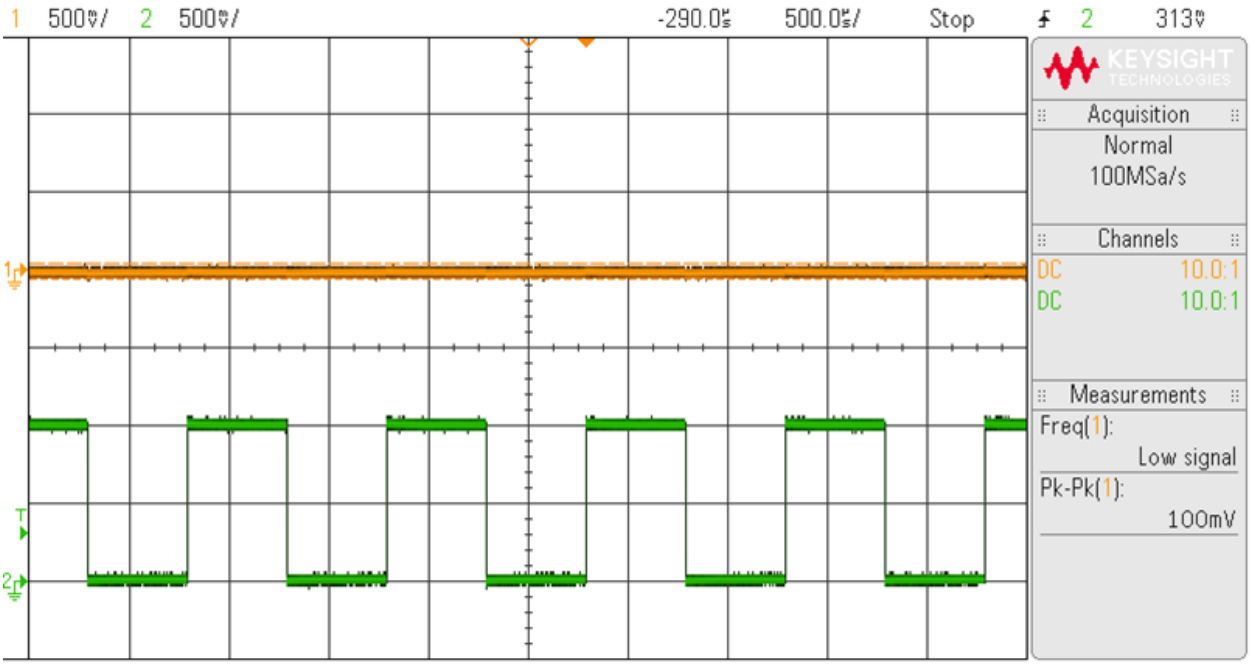

Figure 4: Cross Talk of USB Line and Bus Line at 1Vpp

The next test was to test the cross talk of the USB and Bus Lines at different voltages. We test it at 2.5Vpp and 5Vpp when the first test on these lines was only at 1Vpp. **Figure 5** and **6** shows that at higher voltages that bigger cross talk starts to accrue. At 2.5Vpp the cross talk was about ±8 to ±10 millivolts. At 5Vpp the cross talk was about ±12 to ±15 millivolts.

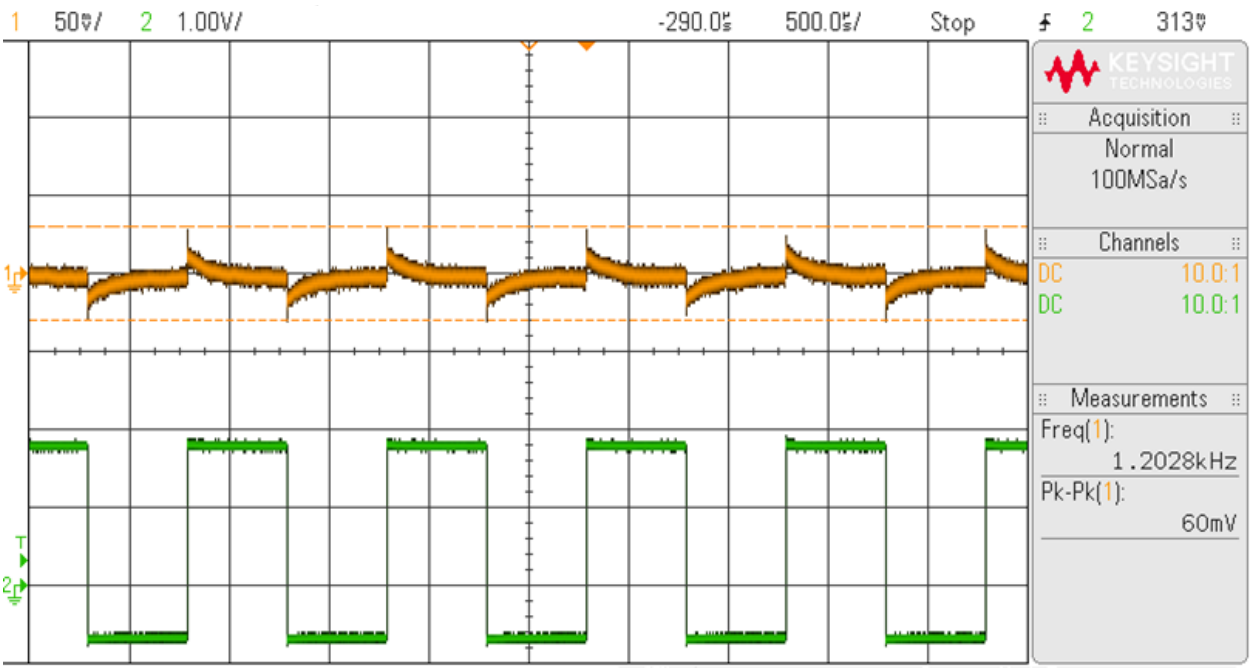

Figure 5: Cross Talk of USB and Bus Lines at 2.5Vpp

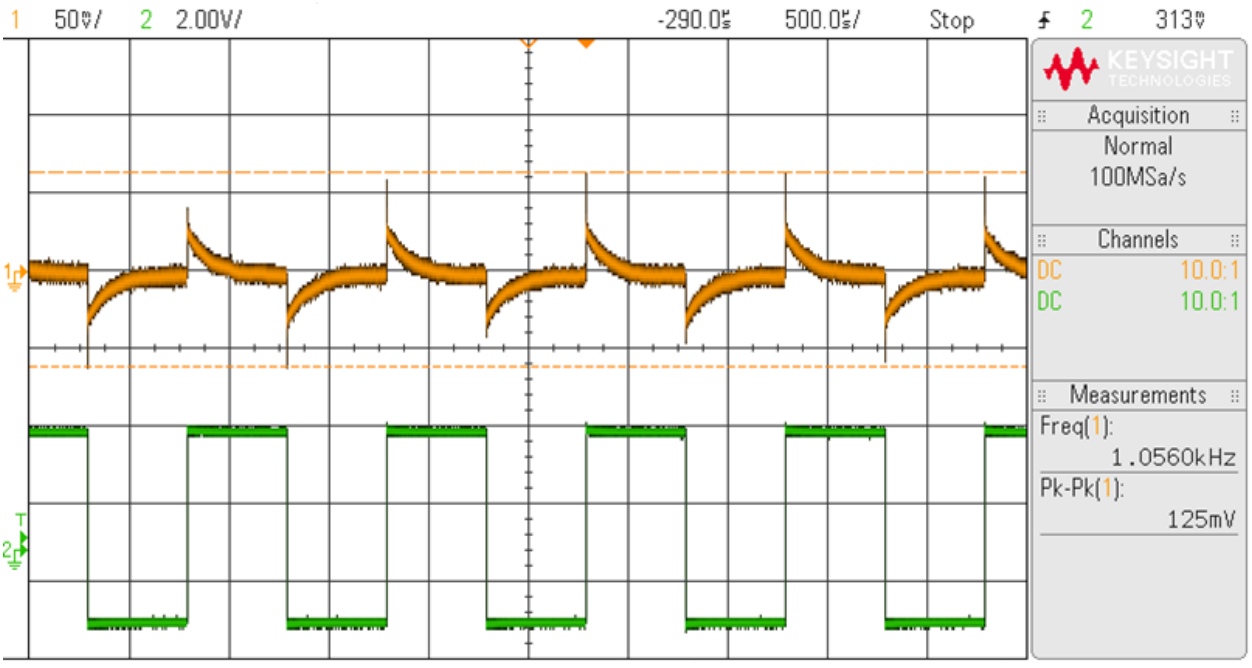

Figure 6: Cross Talk of USB and Bus Lines at 5Vpp

All of the crosstalk measured in the wire trace lines meets the requirement for maximum allowable voltage.

Signal Integrity Testing

A common method of testing signal integrity of a digital circuit is an eye diagram. An eye diagram is able to measure the amount of noise on a signal as well as the amount of clock jitter. To setup the test the board is probed using the oscilloscope at its clock and data lines and display persistence is turned on. By turning on persistence jitter will show up on the scope as a color gradient. The brightest portion would be the most common signal patterns and the less common and more extreme signals will show up faded. This test allows you to measure the maximum rise and fall times as perceived by the receiving device in order to determine if your circuit meets specification. The horizontal fluctuation will be increased with the more noise on the line. This is primarily important on parts that operate at higher frequencies, such as the memory which was the primary target of this test. The testing setup was not ideal as the board had to be extremely under clocked in order for a good scope capture. **Figure 7** shows the signals from the clock and data source which shows very little clock jitter. There is some noise but this is likely due to the testing setup and not from the processor itself.

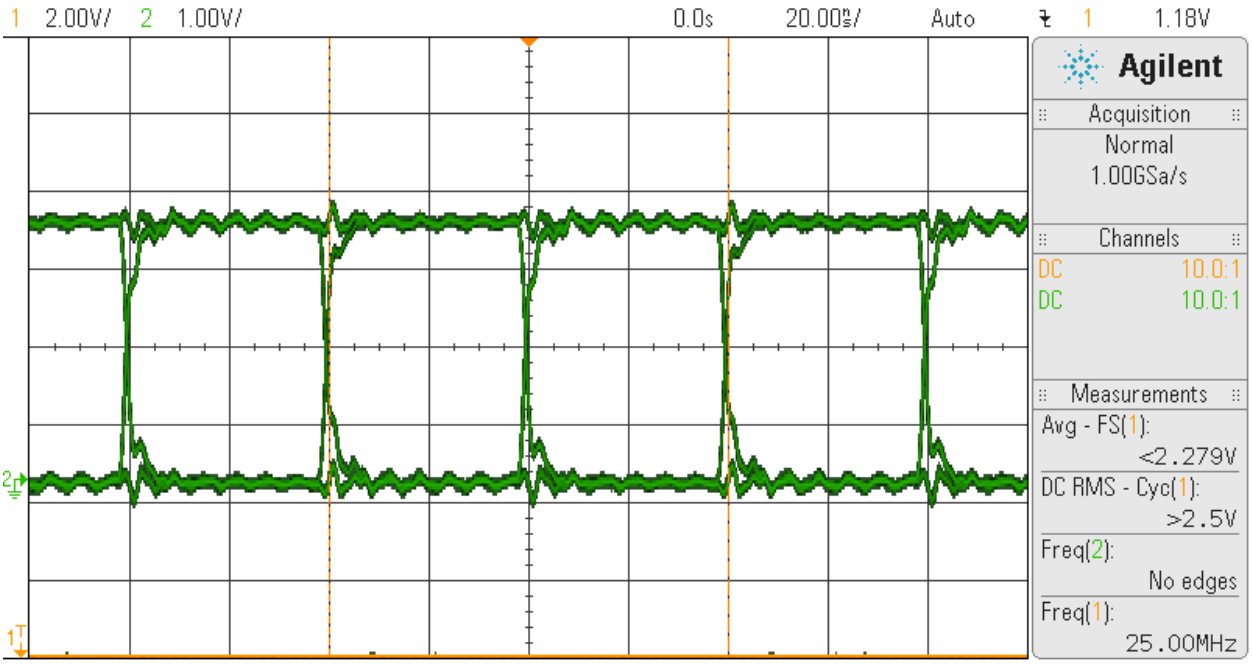

Figure 7: Eye Diagram at Clock and Signal Source (Only Signal Shown)

After the source was measured as a benchmark, the board was probed at the memory location. The result is shown in **Figure 8** and also shows very little clock jitter. The main difference that appeared in this test was the slight increase in rise and fall times. This time was measured as slightly less than 3ns for a change from the maximum input low of 0.8V to the minimum input high of 2V, which is within the specification for the access time of the memory module which is 5.4ns.

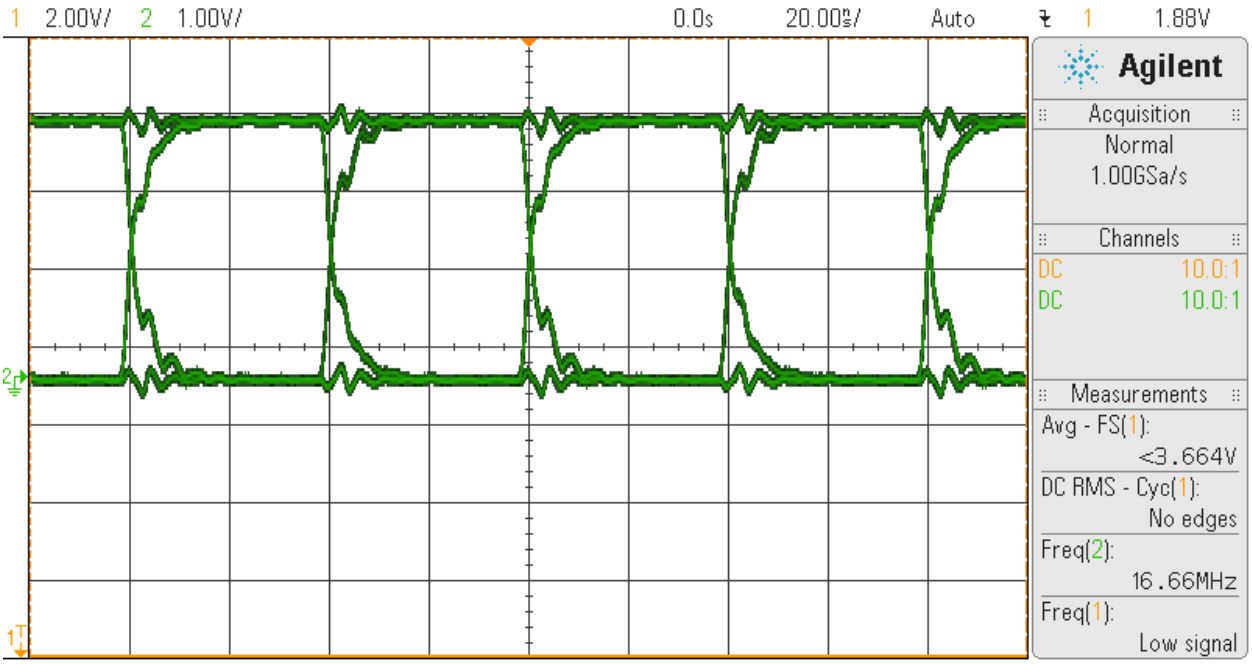

Figure 8: Eye Diagram as Received by Memory (Only Signal Shown)

Power Testing

A simple test was conducted to ensure the board would be able to supply current at the maximum amount specified by USB specifications for peripheral devices. The draw is 500mA max and was the minimum target of the test. To conduct this test a simple resistive load was applied to the board and the voltage supplied was slowly increased. With each increment the power traces were felt by hand to check if they were overheating at all. If the traces were to get noticeably hot than too much current is being passed through and would risk damaging the board. The final current that was reached was the near what the source could supply, approximately 2.4A as shown in **Figure 9**. The board was not damaged and traces did not detectably overheat, and it is likely that the board can handle higher currents. Power protection such as resettable fuses would be useful for later versions of the board, however, because the on-board voltage supplies are not rated for this high of current.

Figure 9: Power Testing of Supply Traces Final Result

Operating System Testing

The O.S. was tested in a few and quick ways. First way we tested the O.S. was to login into the board using ssh and an Ethernet cable. Because the Linux kernel was used from a computer running the Debian O.S. and from the setup of the TFTP most of the settings were setup and ready to go. The login was easy and it verified that an O.S. was installed on the board.

The second way the O.S. was verified was to run a few commands on the Linux operating system. Commands like: pwd, lscpu, cat /proc/meminfo, cat /proc/version, and other commands. The outputs from each command were verified against the expected output that was calculated or evaluated by hand. The last test done on the O.S. was to run a very simple C program on it. A “Hello World!” program was written, compiled, and ran. When the out on the screen showed “Hello World!” then the installation of the O.S. was verified.

Project Management

Since this is a hardware related project there are exaggerated time constraints compared to a project that only requires the creators to compile and debug. Sending the board to be fabricated could take several days, even weeks for some services, as well as shipping time to receive parts meant that time had to be used effectively and deadlines set to ensure something would be built and tested by the end of the project. **Figure 10** shows the approximate deadlines set for certain parts of the project. Some areas were flexible, while others less so. The different shades for some tasks indicates it was taking longer than desired and was critical to complete soon. Board Design had to be completed before May to allow turnaround for the board to be fabricated, shipped, and then assembled by hand. Software was important to the research and selection of components, and thus went on early in the project and then paused once it was understood the steps that needed to be taken to upload Linux to the board and get it running. It resumed as the hardware portion was nearing completion. The chart is color coded to represent the primary roles, however, throughout the process both members assisted one another with their primary tasks and also undertook a shared responsibility for the completion of other parts of the project. In order to maximize available time, development continued during breaks as well.

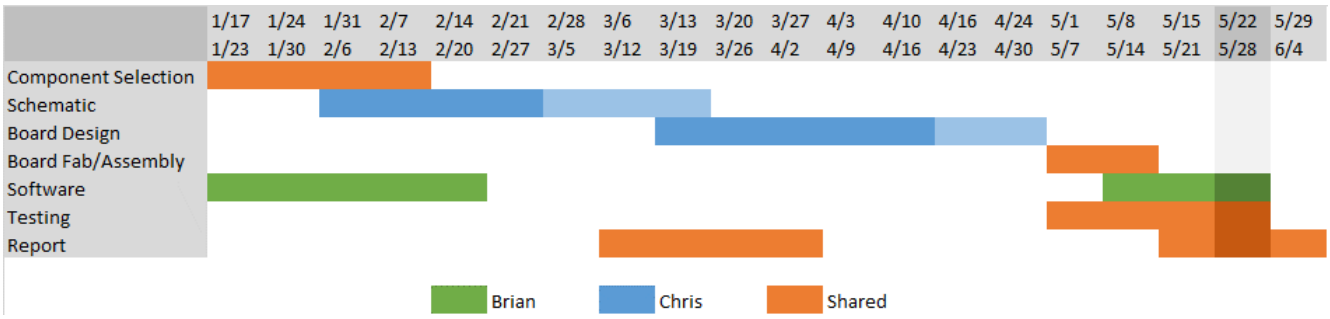

Figure 10: Rough Gantt Chart Used to Keep Track of Progress and Approximate Deadlines.

Conclusion and Future Work

This project provided a lot of insight into the process of designing larger systems such as computer mainboards. When selecting devices for the board, a familiarity with all of the components must be gained by examining each component's datasheet before the board can confidently be created and thought to actually function properly. Once the schematic is finally created, the board layout process begins where parts must be strategically laid out and connected in a way that ensures reliable signals and also keeps the board size as small as practical. Finally, the board must receive a special Linux build, as the processor has an architecture different from mainstream computers. The lessons learned through the development of this board will be applied to future revisions and expansion boards where even more features of the ARM9 processor will be taken advantage of, or even for possibly adventuring into attempting BGA layouts which will allow for even more modern and advanced technologies now that we are comfortable with SMT board design. More precise versions of the tests conducted already can be done on future versions to ensure a completely reliable device. This would specifically involve more comprehensive testing of crosstalk and signal integrity to test all signals. Many lessons were learned by completing this project, virtually all concepts demonstrated through this report were new. It also provided invaluable insight into obstacles that come about when creating a hardware product, such as turnaround times and costs that aren't normally associated with software projects.

References

1. Advanced Circuits Trace Width Calculator. Retrieved from http://www.4pcb.com/trace-width-calculator.html

2. Ott, Henry. PCB Stack-Up. Henry Ott Consultants. Retrieved from http://www.hottconsultants.com/techtips/pcb-stack-up-2.html

3. High Speed Analog Design and Application Seminar Notes. Retrieved from http://www.ti.com/lit/ml/slyp173/slyp173.pdf

4. Alfke, Peter. Printed Circuit Board Design Considerations. Xilinx. Retrieved from http://web.ewu.edu/groups/technology/Claudio/ee260/LabDocuments/pcb_xilinx.pdf

5. Olimex Ltd (2013) OlinXino-Micro Open-source single-board Linux computer User's Manual. Retrieved from https://www.olimex.com/Products/OLinuXino/iMX233/iMX233-OLinuXino-MICRO/resources/iMX233-OLINUXINO-MICRO.pdf

6. Olimex Ltd. "IMX233-OlinuXino get started – making the ARCH Linux SD card." Olimex. 06 September 2012. 04/28/2016. https://olimex.wordpress.com/2012/09/06/imx233-olinuxino-get-started-making-the-arch-linux-sd-card/

7. Olimex Ltd (2013) IMX233-OlinuXino-MAXI an open-source embedded Linux board User's Manual. Retrieved from https://www.olimex.com/Products/OLinuXino/iMX233/iMX233-OLinuXino-MAXI/resources/iMX233-OLINUXINO-MAXI.pdf

8. Texas Instruments Inc. (2010) Linux PSP User's Guide for AM1x/OMAP-L1/DA8x. Retrieved from http://software-dl.ti.com/sitara_linux/esd/AM17xSDK/latest/index_FDS.html

9. Mentor Graphics Inc. (2010) CodeSourcery Tools. Retrieved from https://www.mentor.com/embedded-software/sourcery-tools-services

10. "DULG." RSS. Web. 04 June 2016. Retrieved from http://www.denx.de/wiki/view/DULG/UBoot

Appendix A –Parts List

Table 2: Parts List

Part	Part Number	Quantity	Price
Thin Film Resistors - 4.87K ohm	RT1206DRE074K87L	1	$0.28
Headers 2X09P RCPT HV-100 TE	215307-9	1	$2.48
Headers FULL STICK HDR/36POS	929834-02-36-RK	1	$1.84
Headers 2.54MM SHUNT	969102-0000-DA	6	$0.05
SolderSoldrPaste 2 PartMix 15g, LeadFree, LowTemp	SMDLTLFP15T4	1	$19.95
Aluminum Electrolytic Capacitors - SMD10UF	EEE-HD1C100AR	2	$0.27
Aluminum Electrolytic Capacitors - SMD1UF	EEE-HD1H1R0R	1	$0.28
Multilayer Ceramic Capacitors MLCC - SMD/SMT 0.1uF	12065C104KAT2A	4	$0.03
Multilayer Ceramic Capacitors MLCC - SMD/SMT 220nF	CC1206KKX7R9BB224	1	$0.03
Multilayer Ceramic Capacitors MLCC - SMD/SMT 20pF	12061A200JAT2A	2	$0.39
Multilayer Ceramic Capacitors MLCC - SMD/SMT 0.01uF	12065C103KAT2A	2	$0.06
Thin Film Resistors - SMD 110 ohm	RT1206FRE07110RL	2	$0.03
Thin Film Resistors - SMD 1.5K ohm	RT1206FRE071K5L	1	$0.03
Thin Film Resistors - SMD 2.2K ohm	RT1206FRE072K2L	3	$0.03
Thin Film Resistors - SMD 49.9k ohm	RN732BTTD4992F25	5	$0.29
Thin Film Resistors - SMD 10K ohm	RT1206FRE0710KL	9	$0.03
Crystals 20.000MHZ 18pF 10ppm	ABL-20.000MHz-B1U	1	$0.62
Microprocessors - MPUSitara ARM MPU	AM1705DPTP4	1	$14.50
USB Connectors SMT USB B-TYPE RBI T&R	KUSBEX-ASFS1N-B	2	$0.56
Ethernet Connectors RJ-45 w/ Transformer	RB1-125BAG1A	1	$2.72
Flash Memory 8Gb NAND Flash	S34ML08G101TFI200	1	$9.86
Ethernet ICsSGL Port 10/100 Mb/s Transceiver	DP83848IVVX/NOPB	1	$5.24
EMI Filter Beads 50 ohm	BLM31PG500SN1L	2	$0.17
Voltage Regulators - 3.3v	ADP2108AUJZ-3.3-R7	1	$1.67
Voltage Regulators - 1.8v	ADP2108AUJZ-1.8-R7	1	$1.67
Voltage Regulators - 1.3v	ADP2108AUJZ-1.3-R7	1	$1.67
Standard Clock Oscillators 50MHz	ASVMB-50.000MHZ-XY-T	1	$2.79
SDRAM 512Mb 133MHz x8	MT48LC64M8A2TG-75:C	1	$12.07
		Total	$79.59

Appendix B - Analysis of Senior Project Design

Summary of Functional Requirements

This project is a single-board-computer that is capable of running Linux. This board is useful for many tasks, and can be used as a control board for smaller electronics or mechanical applications. The on-board peripherals included allow this interfacing with the world around it and allows for the user to connect to the board from remote locations if the board is connected to the internet. This makes it useful for creating "Internet of Things" devices.

Primary Constraints

One of the most significant constraints in this project was experience. This is the main factor that limited the capabilities of the board. It would be possible to pick out components that would make surpass SBCs on the market today but there would be even more nervousness with whether or not the board would work at all. Instead, performance was balanced with designer capability – capability that was mostly all learned through the process of working on this project so naturally the goals weren't set very high. One of the physical constraints that existed is cost. It would not be possible to have the board manufactured with industrial robots that are capable of extreme precision, nor was it possible to have an entirely comprehensive design or testing process. Assembly of the board was done by hand meaning parts had to be of a manageable size. Board simulations were not possible because the software cannot be afforded by students. Finally, newer technologies weren't able to be used without the ability to comprehensively check traces for continuity and fully analyze solder joints of BGAs. To leap to a more complicated system while being unable to fully verify the boards condition would only add to an enormous potential list of problems if the board did not function. As a result of these constraints, the goal of the board was to keep it as simple as possible while still supporting a general purpose processor that can run Linux. This project was made difficult because it was done completely using a Learn-By-Doing approach, but as a result it was also a very rewarding project.

Economic

The costs of materials for prototyping totaled $343.59 which breaks down to $264.00 for PCBs and $79.59 for parts. Software to perform board layout was an additional $169.00. The bill of materials is shown in Appendix 1. The total development time was 19 weeks. There were no initial timeline projections for the project, instead some fixed deadlines were set that had to be met in order to ensure project completion.
If manufactured on a commercial basis
The board is not a commercially viable product in its current state and therefore would not likely sell at all. Better boards already exist for far cheaper than it would be possible to produce the current design, such as the Raspberry Pi Zero for $5.
Environmental
As the board was not made to ROHS specifications it would need to be disposed of properly. The board uses parts that are not ROHS certified and lead was used when soldering components in place.
Manufacturability
The board would not offer any significant challenges if it were manufactured on a large scale. It uses standard PCB techniques and also requires assembly on only one side.

Sustainability

The design and manufacturing of the board does not lead to any issues or challenges with maintaining the completed device. The board only requires only 5Vs to operate so, it does not use a lot of resources. The board's design does not have a GPU at this time. To upgrade the board a FPGA could be added using the SPI connections on the GPIO. The FPGA could be program to function like a GPU for the board. The biggest issue from upgrading the FPGA would be to program it to become a GPU. Another design upgrade that could be done with the board is to change the CPU that uses flatpack package to a ball grid array. With this upgrade the board's design would not be a limiting as it was. The issue with changing to a ball grid array design is making sure that the IC where solder was done correctly.

Ethical

The board would not have any ethical implications when comes to the design of the board. The board's design is typical of other boards of similar size and shape. The manufacturing of the board does not have many ethical implications also. The manufacturing of the board could be done anywhere, including manufacturing plants that already exist. The misuse of the project is just as likely as the misuse of any general computer operating system.

Health and Safety

The board's design, manufacturing, and use do not have any more or less health and safety concerns than the other single-board computers that are on the market today.
Social and Political
The board's design, manufacturing, and use do not have any more or less social and political concerns than the other single-board computers that are on the market today.

Development

In this project we had to use many new tools and techniques for development of the board. First, when we were designing the PCB board we had to design all the circuitry ourselves. It had to be all done with the datasheets which we have only had limited practice with. Both of us had never used the Eagle Program to design a PCB board before. Another new technique we employed in this project was the use a reflow oven to solder the IC and other components to the PCB board.

When it came to installing the operating system on the board all the development was new to us. Every technique and tool used to install the O.S. had to be researched and learned. The U-Boot program, Linux kernel image, flash writers, CodeSourcery, TI SDK, User Boot Loader, and TFTP server were all new to us and we had to learn how to use each one. Even though we had use CCStudio before, we had not used it this way before for this propose.

Appendix C – Schematic

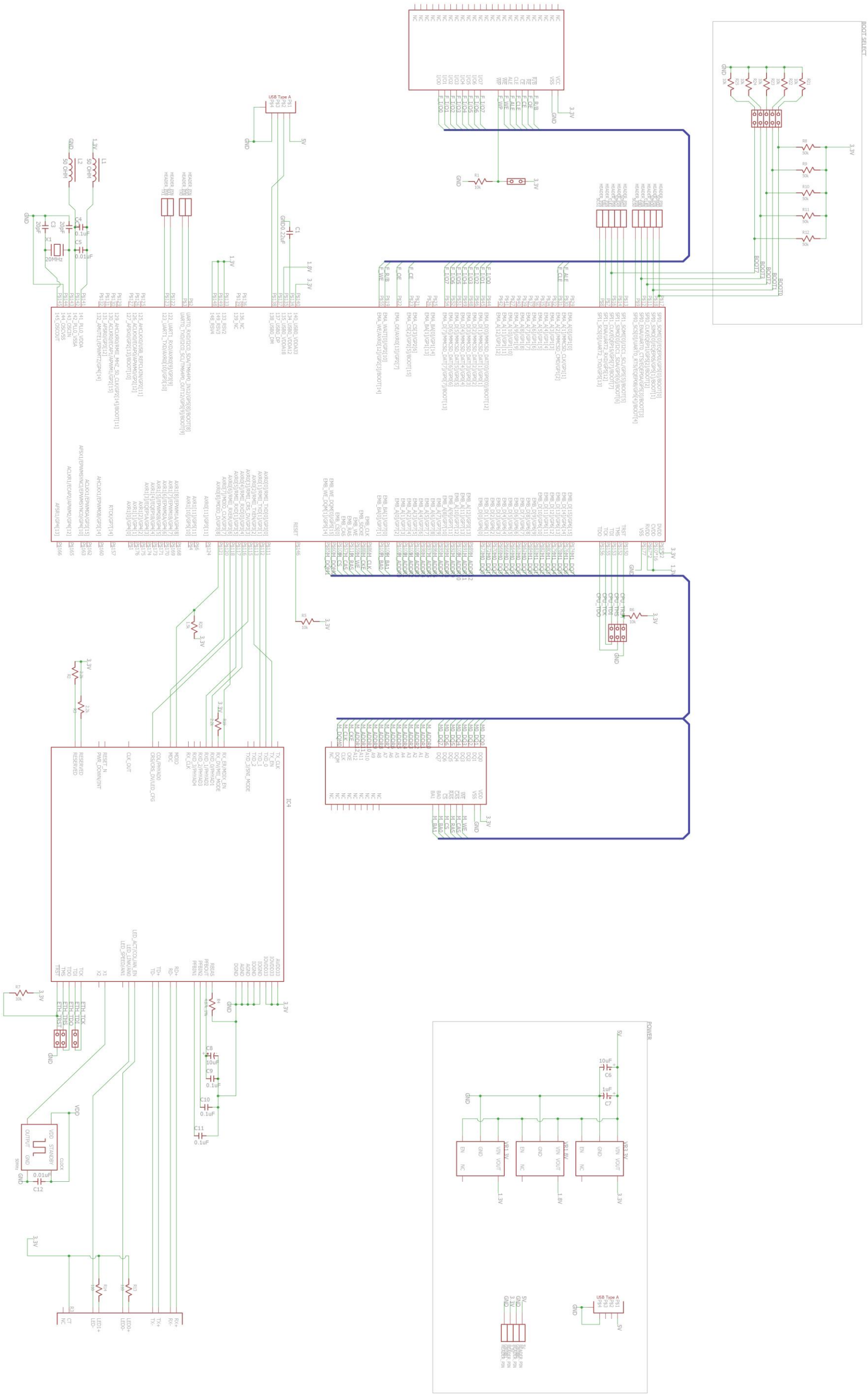

Figure 11: Schematic

Appendix D – Board Layout

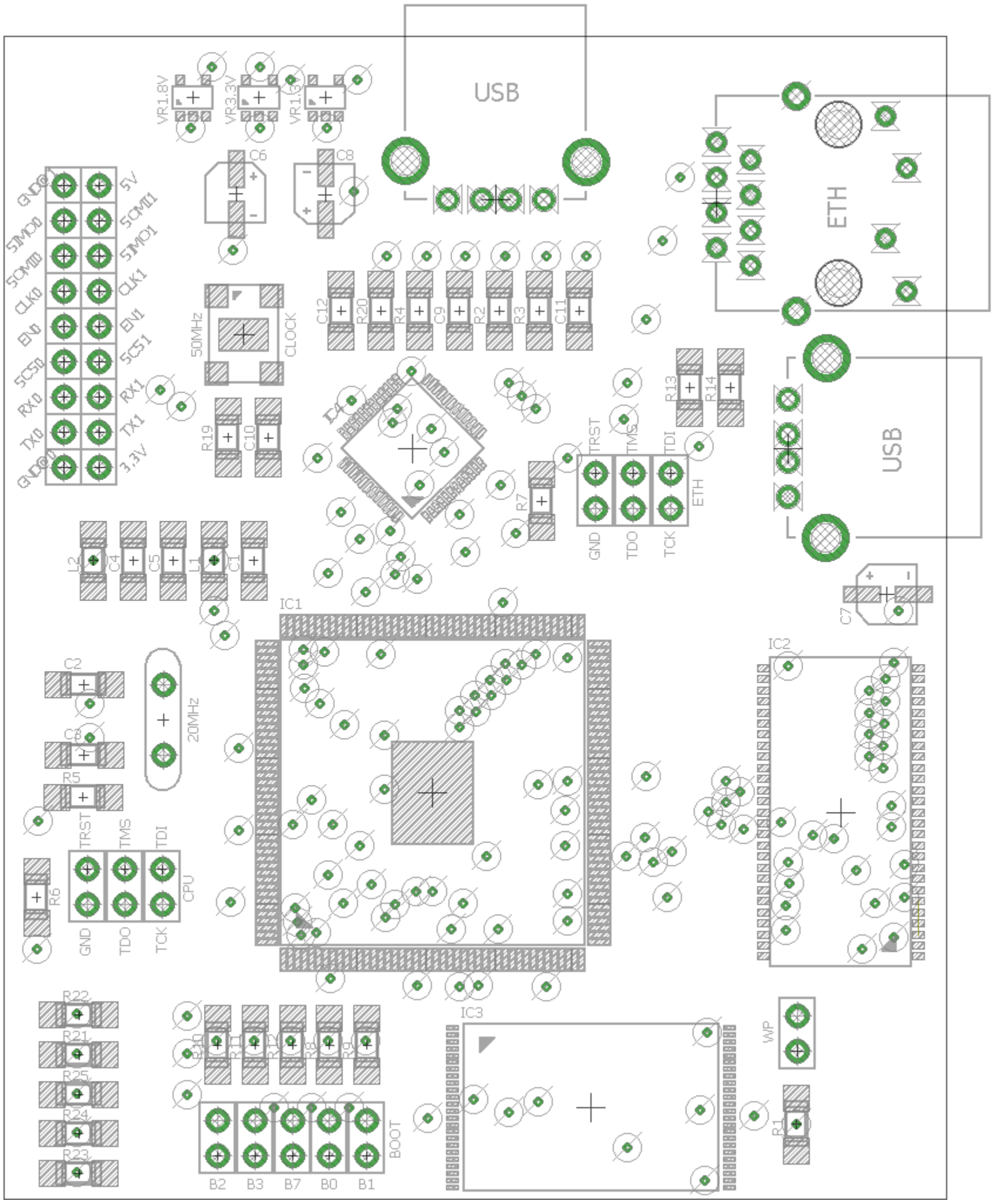

Figure 12: Board Layout – Placement, Pads, and Vias

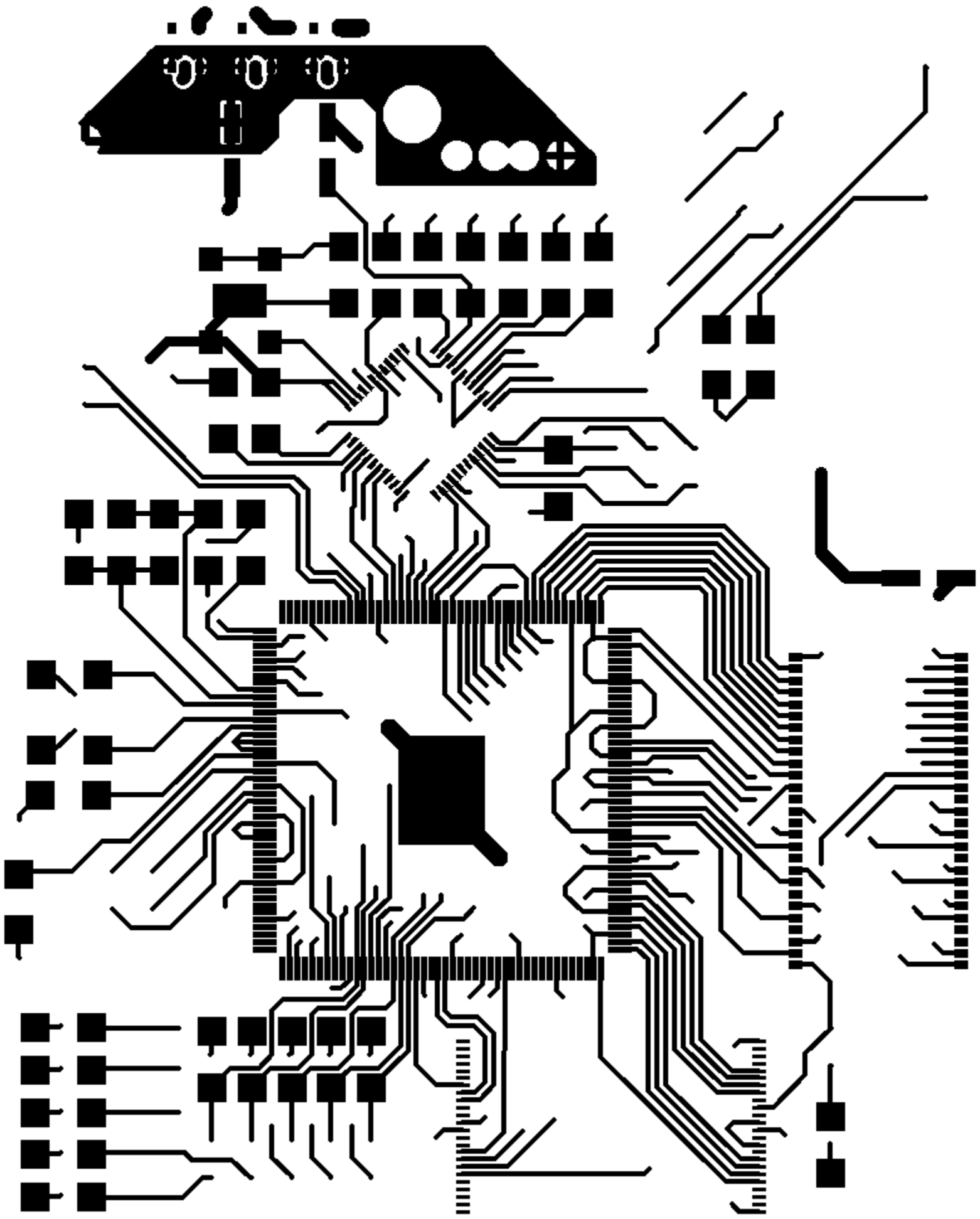

Figure 13: Top Layer (Route 1)

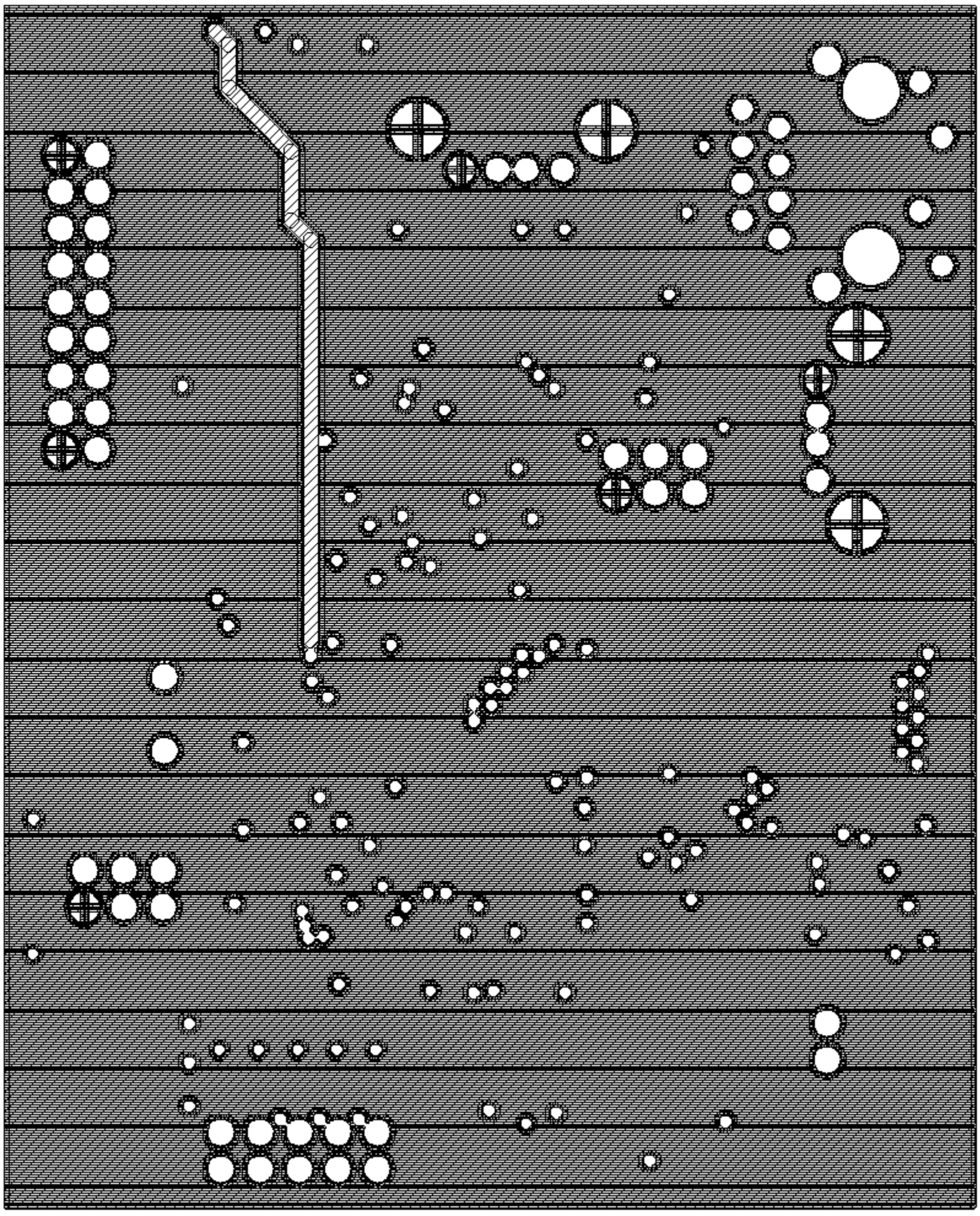

Figure 14: Route 2

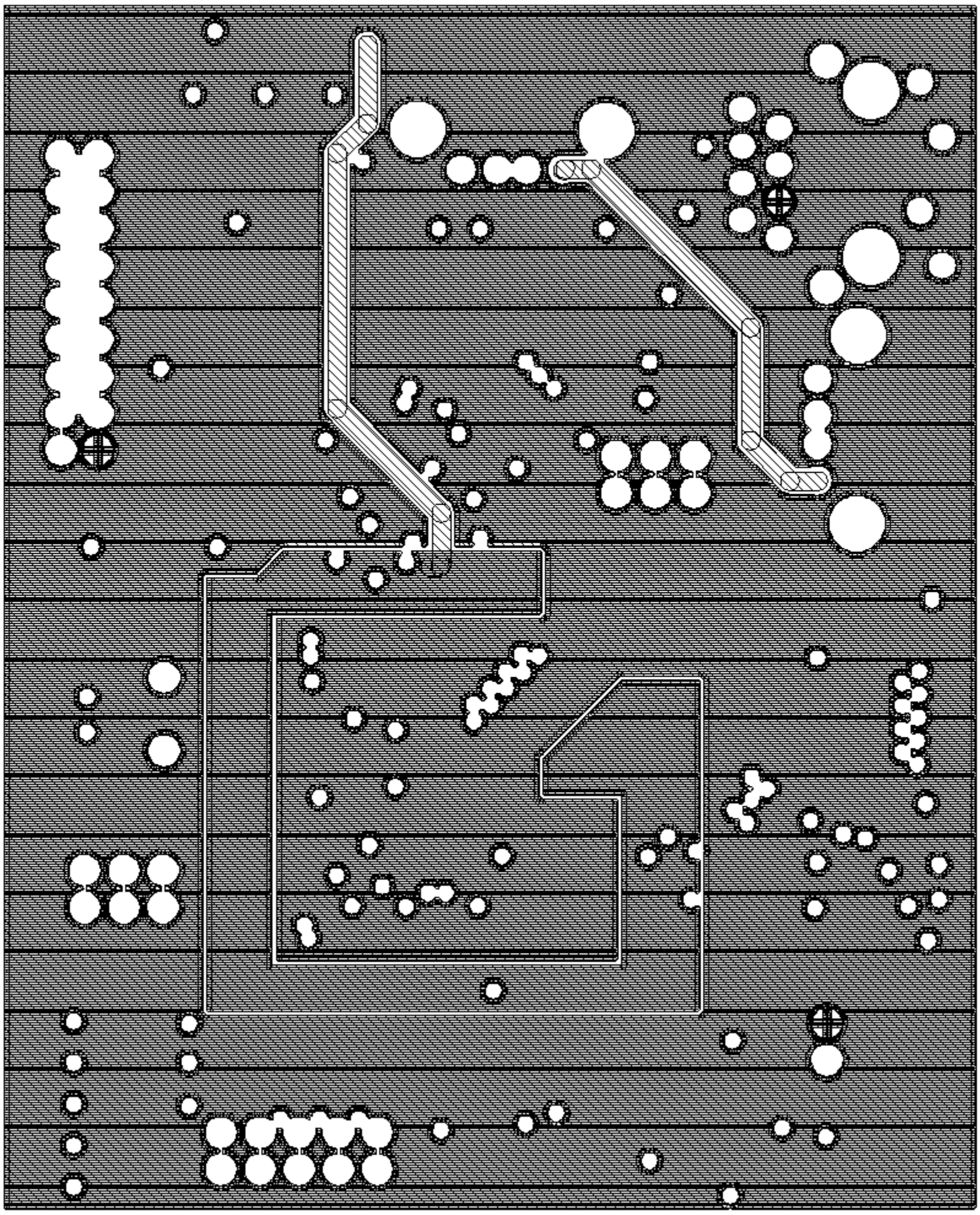

Figure 15: Route 15

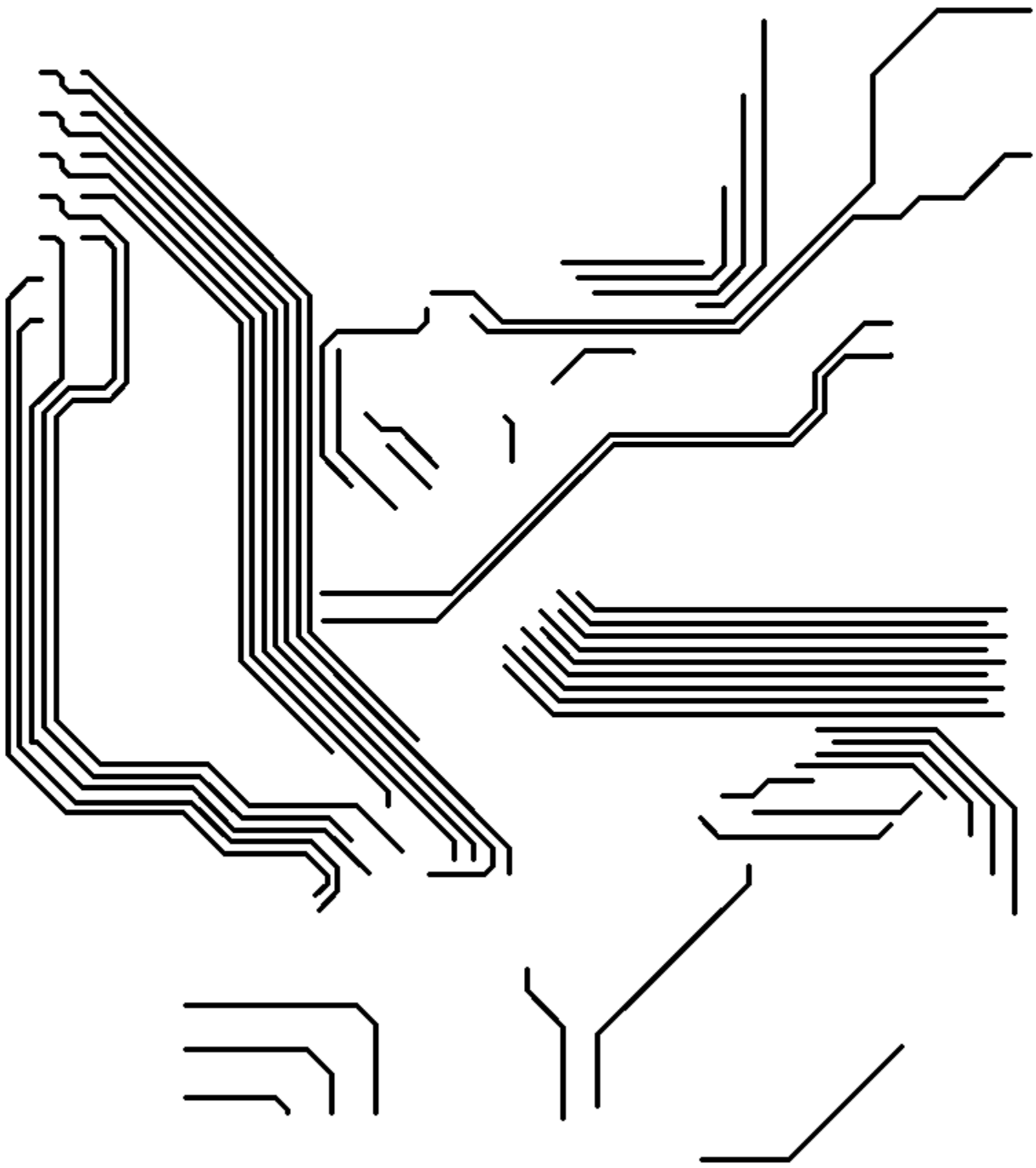

Figure 16: Bottom Layer (Route 16)