Tunable Gain Audio Amplifier

a.k.a TYGA9001

E.D. Money a.k.a. Criminal Jabs
Prince Forrest B. Fist
Wild Erin P. Cash

December 2014

1 System Overview

The TYGA9001 is a high–performance, stereo audio volume control designed for professional and high-end consumer audio systems. Using high performance differential amplifier stages and rail-to-rail circuitry internal to the TYGA9001 yields low noise and distortion, while providing the capability to drive 600Ω loads directly without buffering. The 3-wire serial control interface allows for connection to a wide variety of host controllers, in addition to support for daisy-chaining of multiple TYGA9001 devices.[1]

We have designed a stereo tunable gain amplifier capable of driving a 600Ω load with an SPI control interface.

It is entirely fair to draw comparisons between our chip, the TYGA9001, and the Texas Instruments PGA2311. The PGA2311 is a stereo volume control, like our chip. It features 120dB of dynamic range (as compared to our amplifier, which has 12dB of dynamic range). By removing the extra dynamic range, we have produced a more streamlined amplifier better for all applications.

1.1 Features And Specifications

- Digitally-controlled Analog Volume Control
  - Two Independent Audio Channels
  - Serial Control Interface
  - Lack Of Zero Crossing Detection
  - No Mute Function
- Wide Gain And Attenuation Range +12db To 0db With Several Steps
- Relatively Low Noise And Distortion
  - 12db Dynamic Range
  - Probably Only A Little THD+N At 1khz (u-grade)
  - Maybe Slightly More THD+N At 1khz (a-grade)
- Sort Of Noise-free Level Transitions
- Hopefully Low Interchannel Crosstalk

[1] One might notice certain similarities to the description for the PGA2311. These are entirely coincidental.

- Power Supplies: +5V Analog, +5V Digital
- Available In Dip-40 Package
- Neither Pin Nor Software Compatible With The Crystal CS3310
- Applications
  - Audio Amplifiers
  - Mixing Consoles
  - Multi-track Recorders
  - Broadcast Studio Equipment
  - Musical Instruments
  - Effects Processors
  - A/V Receivers
  - Car Audio Systems

To meet these specifications, we designed the system seen in Figure 1.

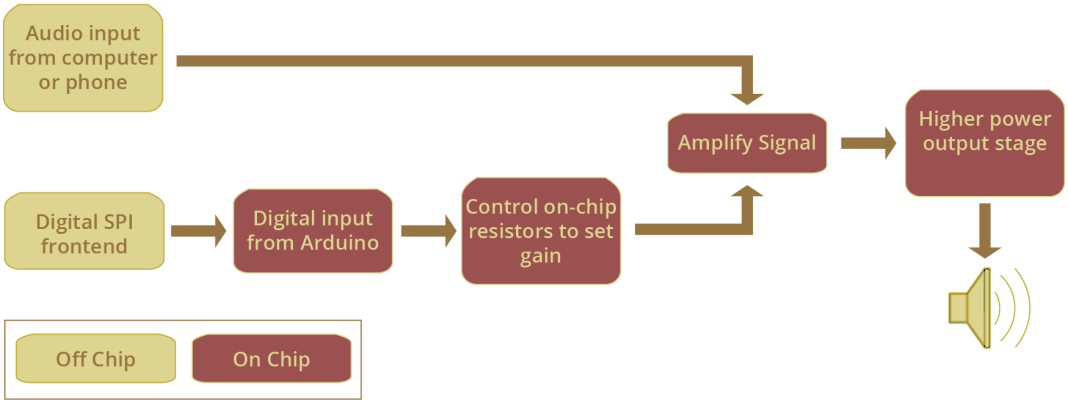

Figure 1: System Architecture

2 Amplifier

Our amplifier is based around the single-stage differential amplifier we built in Machine Problem 3, but with several key changes.

2.1 Rail-to-Rail Input Stage

We added a rail-to-rail input stage, as shown in Figure 2 on the next page. This input stage is a significant improvement over the input stage from Machine Problem 3[2]. In addition to allowing a rail-to-rail input swing, it has an input impedance of 625kΩ depending on the gain of the amplifier. All the transistors in the input stage are of analog unit size, so the addition of this stage doesn't *significantly* increase the size of our amplifier.

[2] `http://madvlsi.olin.edu/madvlsi/handouts/141002_mp3.pdf`

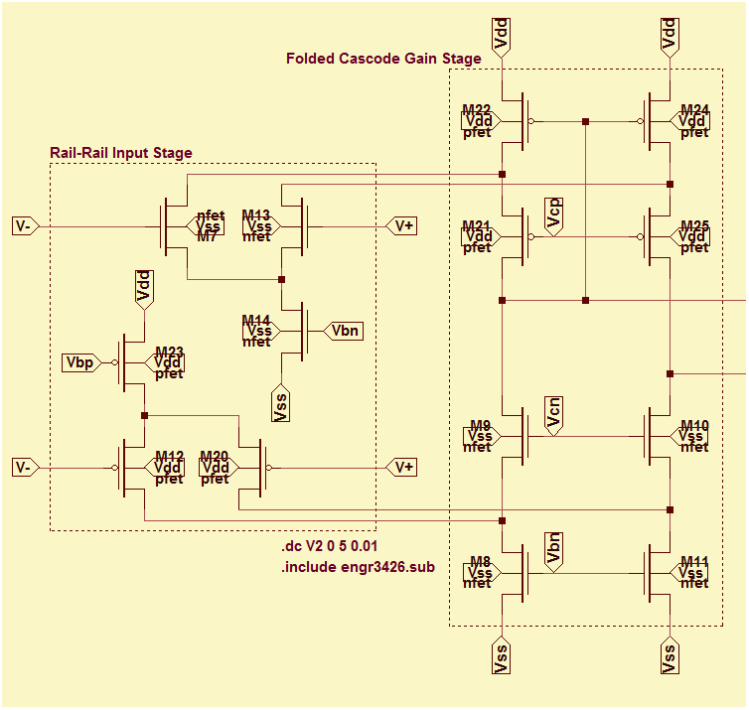

Figure 2: Input Stage

2.2 High Current Rail-to-Rail Output Stage

We initially set out to drive a 600Ω load, without first assessing the feasibility of this goal. 600Ω is a standard from the early days of telephones[3], where the unit reference "0 dBm" was the voltage necessary to drive 1 mW over a 600Ω load. Regarding output levels, since we are limited to a 5V supply voltage, we're limited to $\frac{5V}{\sqrt{2}} = 3.53V_{RMS}$, or 13 dBu of output. This, fortunately, is well within the range of both pro- and consumer-level audio equipment.

Our output driver transistors are effectively 1.8μm long by 3600μm wide (implemented as 50 interleaved 1.8μm by 72μm transistors), giving them a strength ratio of 2000, one hundred times the strength of our standard analog transistors. As it happens, these are able to drive our 600Ω load with room to spare. In simulation our output impedance is the incredibly low 9.29Ω, giving us an output short current of nearly 250mA!

[3] `http://en.wikipedia.org/wiki/Line_level`

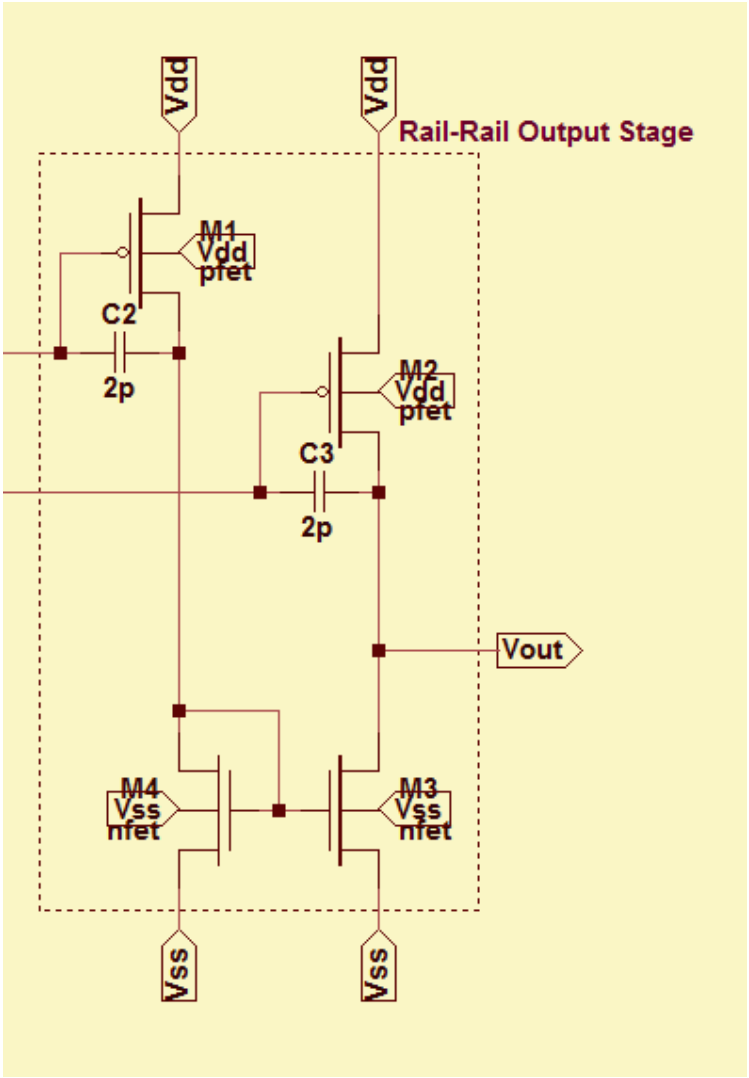

Figure 3: Output Stage

3 SPI Interface

Our SPI interface functions similarly to most SPI peripherals—our timing diagram is shown in Figure 4. We have no serial output, and the last four bits of the serial input determine the gain setting. Any bits before these are ignored. When `CSn` goes high, the serial data is passed to the amplifier and the new gain setting takes effect.

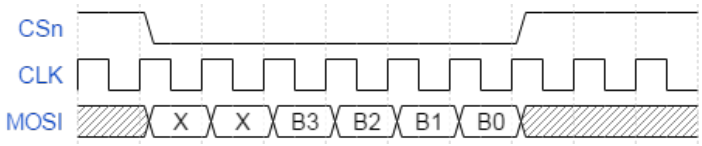

Figure 4: SPI Timing diagram

Fortunately, SPI interfaces are fairly easy to implement using shift registers. For the most basic SPI interface, we merely needed to add an enable input to each D flip-flop. We did this by adding a second pair of pass transistors between the two latches, as shown in Figure 5. A simple shift register is a perfectly acceptable SPI interface. However, the issue

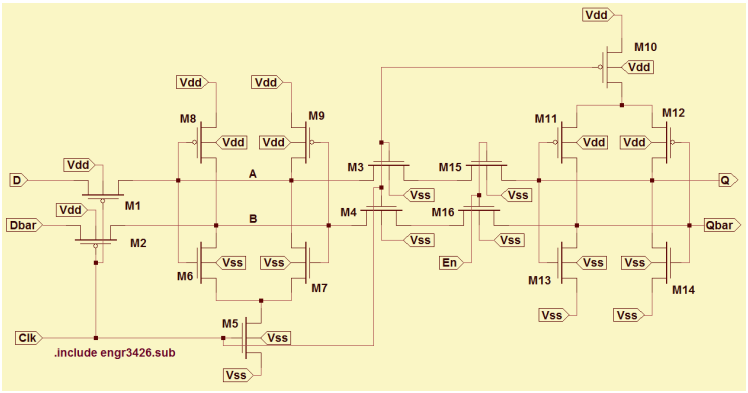

Figure 5: D Flip Flop with Enable

with this in our application is that the parallel bits change to several intermediate values between the initial and final values as the serial bits are shifted in. Though this glitch is fairly short (and depends on the SPI clock speed) we decided to implement another set of transparent latches to prevent the new data from changing the output while Chip Select is still enabled. The transparent latches' clock inputs are falling-edge triggered and connected to the inverted `CSn` signal. This structure is seen in Figure 6. An additional problem (for

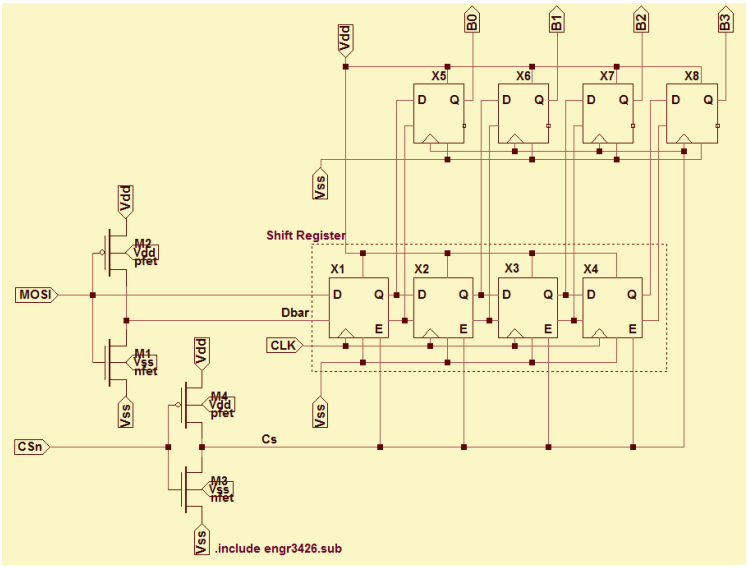

Figure 6: Shift Register with Transparent Latches

possible future investigation) is what happens to the signal at the moment the gain changes. Currently, the amplifier changes gain immediately upon the master device releasing `CSn`. To facilitate a more smooth gain change, it would be best to develop a zero-crossing detector. We imagined a system with two comparators and voltages very close to zero—when both

comparators indicated the output was close to zero, a pulse could be generated (using a single D Flip-Flop) that would clock the transparent latches and change the gain.

4 Full System

Our fully integrated system is shown in Figure 7.

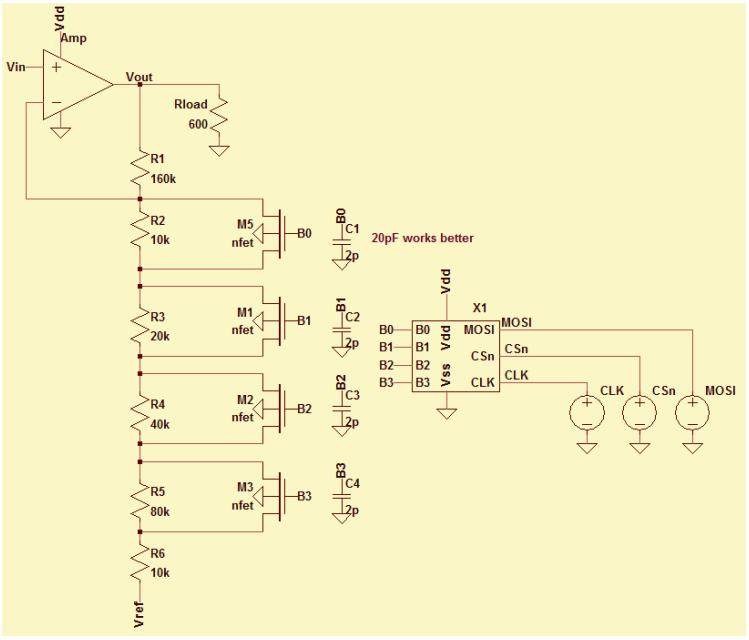

Figure 7: Our full system

Here we short resistors in the feedback loop of our amplifier to set the gain.

To test our full system, we fed the amplifier with a 5kHz sine wave and watched the output (attached to a 600 ohm load) as the gain decreased logarithmically through the 16 levels.

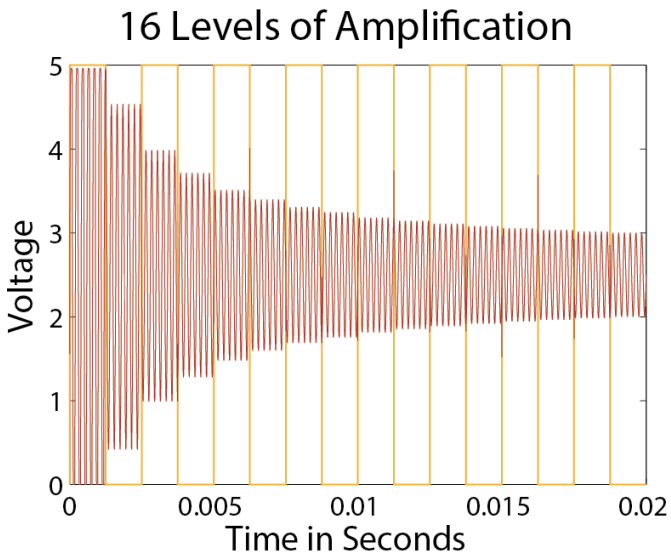

Figure 8: Full System Test

5 Layout

Our overall layout, including the routed padframe, can be seen in Figure 9. To layout the padframe, traditional "dumb" autorouter techniques were used, including using horizontal metal3 wires and vertical metal2 wires. Power and ground connections were routed in metal1.

Additionally, we imported images of our faces onto metal3. As far as we know, we will be the only three people in the world with our smiling visages on an integrated circuit.

We tried to take to heart more of the analog layout best practices that were taught in class. This required completely redoing the analog layout of the gain stage of the amplifier from Machine Problem 3, since none of us had done a very good job the first time around. This time, putting the input stages in the center of the diff amp and mirroring the gain stages around them made the amp more compact, more symmetrical, and much prettier (as seen in Figure 11 on page 10). Hopefully its performance will take after its layout.

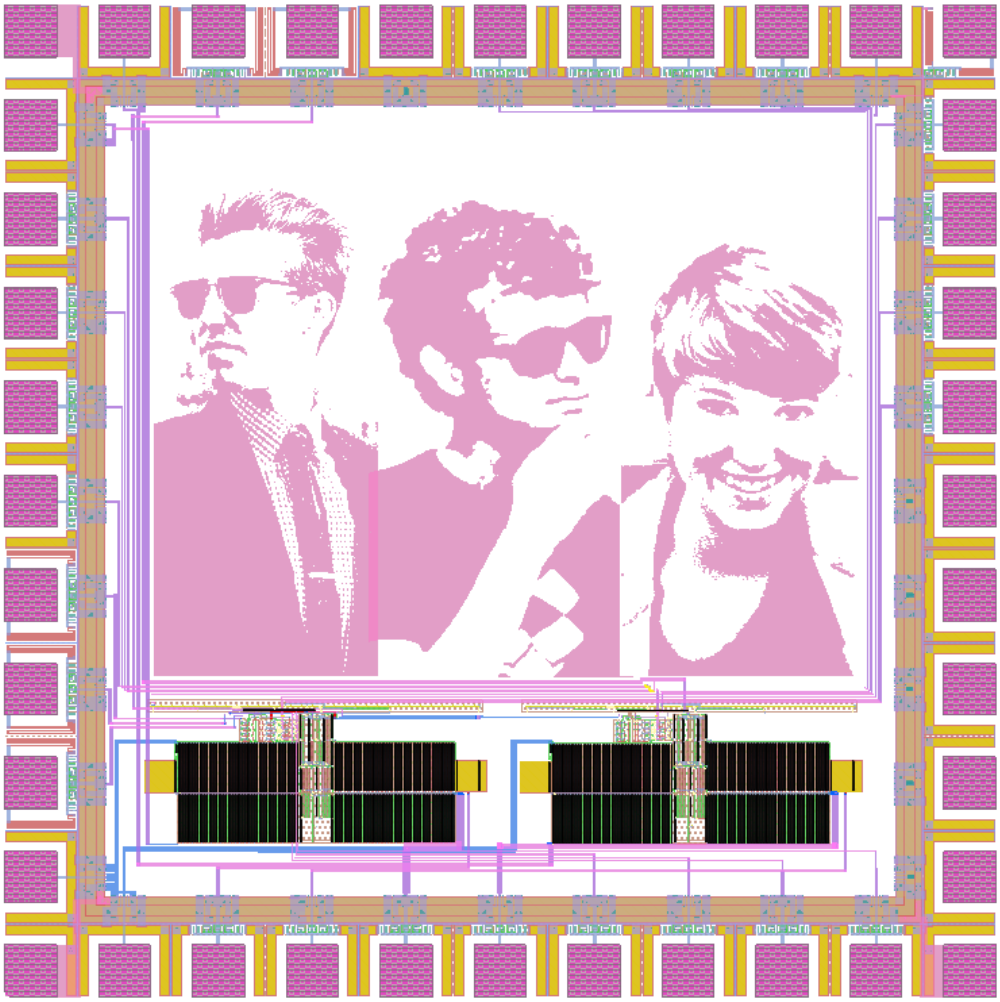

Figure 9: Padframe Layout

The layout of one of our overall amplifier and SPI interfaces can be seen in Figure 10 on the following page. The resistors used for the amplifier are at the top.

A closeup of the SPI interface is in Figure 12 on page 11.

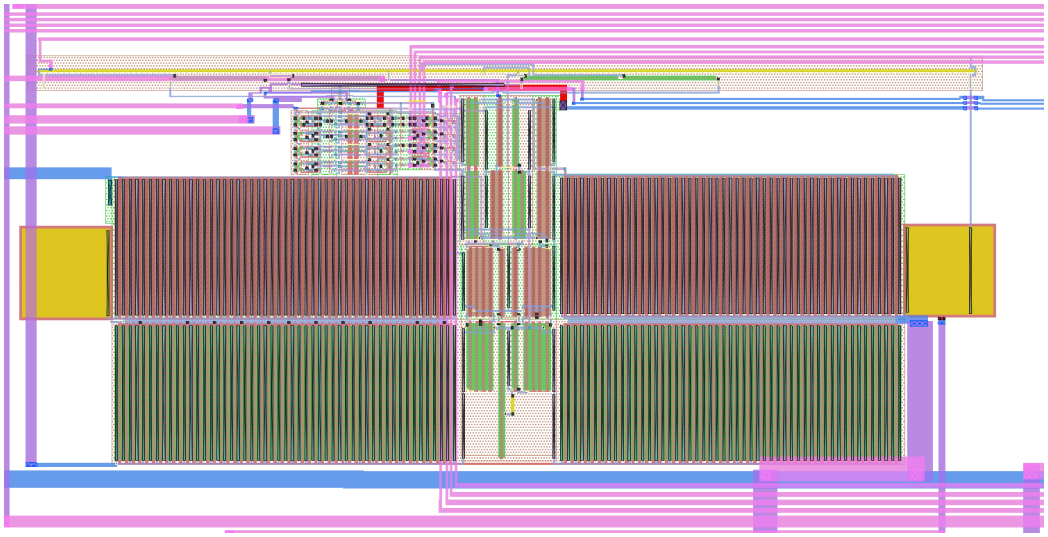

Figure 10: Amplifier and SPI Interface Layout

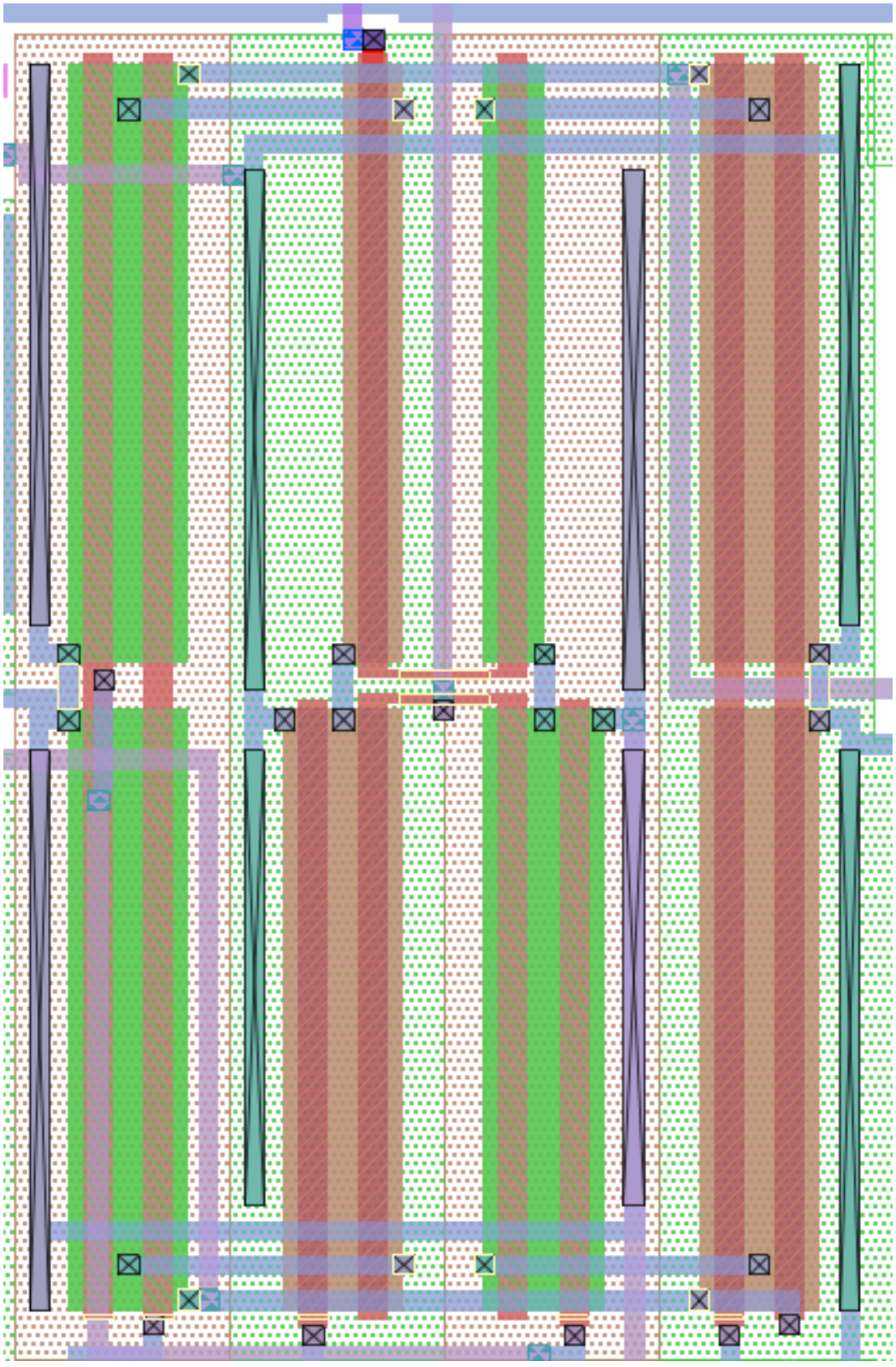

Figure 11: Diff Amp Layout

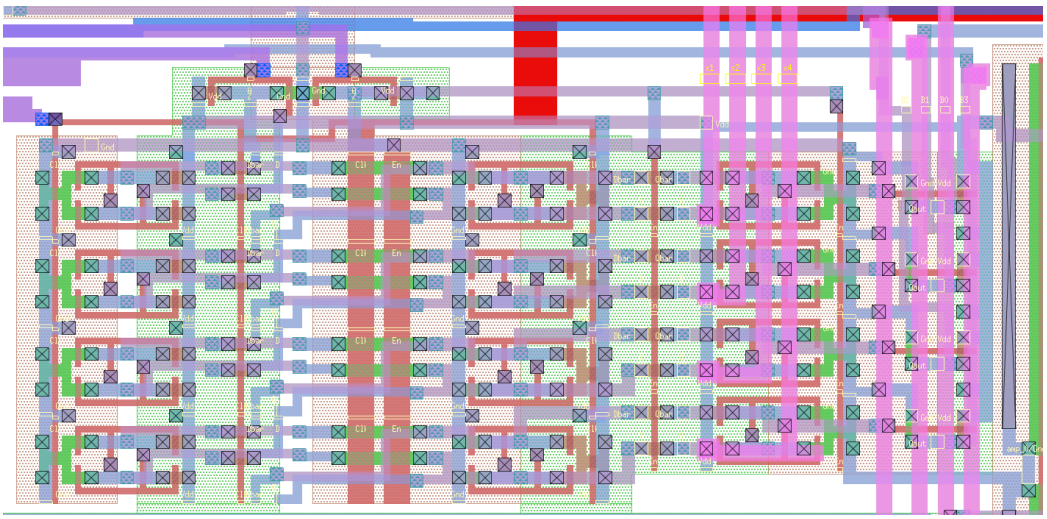

Figure 12: SPI Interface Layout

6 Test Plan

6.1 Basic Function

When we first get the chips back, our first test will be to connect it to power and see if it catches on fire. If it does not catch on fire, we will proceed to setting up an Arduino to output a clock signal and iterate through all combinations of gain. We will observe the shift register test points and confirm that our desired gain bits are shifting through on the clock signal. This will verify if our SPI interface is working. We will then set the gain to 0000, which is the minimum, and feed a 1kHz sine wave with an amplitude of 500mV into the analog input. We will scope the output and verify that the signal does come through and is amplified by approximately 2. We will reduce the amplitude of the input to 100mV and run through all the possible gain cases, confirming that we can observer increasing gain on the scope. Next, we will connect our 600Ω speaker to the output, with a 5k potentiometer in series with it. We will continue running through all amplification stages with the 1KHz wave and listen to the speaker. We will slowly reduce the potentiometer resistance, monitoring the TYGA9001 for signs of fire. Hopefully will be able to reduce the pot all the way to 0, indicating that we can indeed drive a 600Ω load. At this point we will select our first song to play, likely Music Sounds Better With You by Stardust. We will scope the song first and adjust its amplitude, and then we will listen to it as it get louder and softer as our Arduino script runs. We will then preform a celebratory group hug.

6.2 Amplifier

Probably using a SMU we will attempt to measure some of the key characteristics of our amplifier, including:

- Open Loop Gain
- Slew Rate
- Input Impedance
- Output Impedance
- Unity Gain Transfer Characteristic

We can then compare this to our spice simulation. We will also use a good source meter to test what the maximum output current is.

6.3 SPI

We will keep increasing the clock speed until either our shift registers or the Arduino begin to mess up, and note at what frequency this occurred.

6.4 Full system

For each possible gain value, we will input a small 20kHz signal, measure the output, and compare the two to see exactly how much gain we were able to get. For each gain level, we can also take a bode plot to see what the phase characteristics are.