Daniel Beierl

6.101 Design project – final report

Alphabet Soup: Displaying Text with an Analog Waveform Generator

1. Goals

The goal of this project was to create a system capable of displaying characters like letters and numbers using an analog, x-y controlled display. Originally, I had hoped to adapt the design to use a laser galvo, but due to time, I just used an oscilloscope in x-y mode. In both cases, character shapes can be generated through the two signals that control the x- and y-deflection of the imaging beam. Varying the shape and timing of these two signals changes the shape of the image generated by the display.

The main source of challenge and interest in this design problem is that it tries to solve a naturally digital problem, displaying text and characters, using analog circuits. That is, there is no systematic way to translate a *value* of '1' into the *shape* of '1,' and so on for the rest of the characters the system might be expected to display. It seems unlikely that any system could produce a wide range of text characters on command, unless it somehow encodes the shape associated with each value. In light of this, the system was designed as a more generic curve tracer that could be tuned or configured to produce a variety of different letters.

2. Design Overview: General Approach

2.1. Scope

Because of the nature of letter characters, it is extremely difficult to come up with a single circuit that is capable of producing every single one. Striving for completeness in this respect would introduce prohibitive complexity to the design. Instead, I sought a compromise between simplicity of design and completeness of functionality. Most of my design time early in the project period was spent reasoning about how to produce the greatest variety of characters without unreasonable implementation complexity.

I ultimately found a reasonably large family of letters that could be drawn in a single pass from left to right and have either odd or even symmetry. That is, they can be expressed in x-y coordinates as simple odd or even functions. For example, the letter 'N' is an odd function about its center, whereas the letters 'A' or 'X' are not functions at all, despite their attractive symmetry. Many of these simple, function-like letters, including M, W, N, V, Y, and T, can be drawn using combinations of a small set of triangle waves and delta functions, as shown in Figure 1. By starting with this small, simple model of character drawing, I was able to make a system that could later be generalized to produce a more significant family of characters and images.

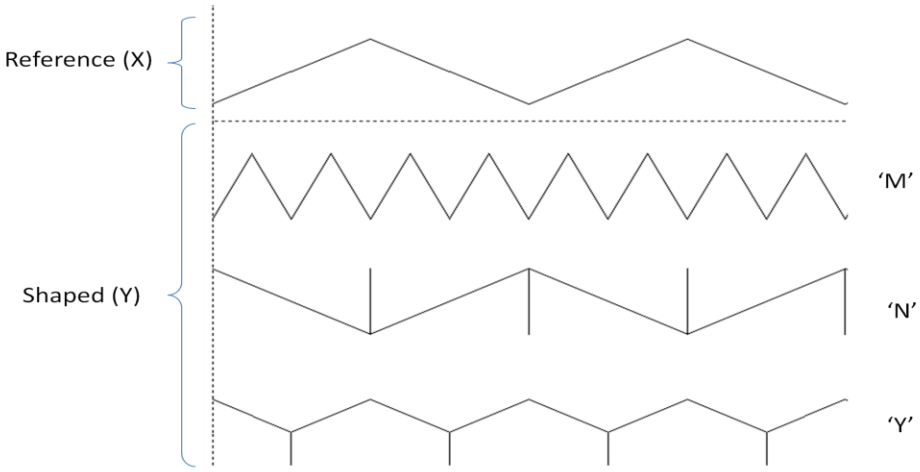

Figure 1: example x-y signals for producing several simple characters

The central challenge and core functionality of the project was the production and composition of these phase-matched triangle and delta functions. Once this framework was in place, however, other possibilities began to emerge. The most straightforward of these, which I did implement, was to change these triangle waves into sinusoidal (technically quadratic) waveforms to make curvilinear shapes. Other additional features could have further expanded the capabilities of the system, but were not implemented due to time. Some of these potential features will be described in a later section. Ultimately, although this basic implementation can only generate a handful of letters, the modularity of the system makes it straightforward to extend the design beyond the initial scope of the project, which is one of the strengths of this general approach.

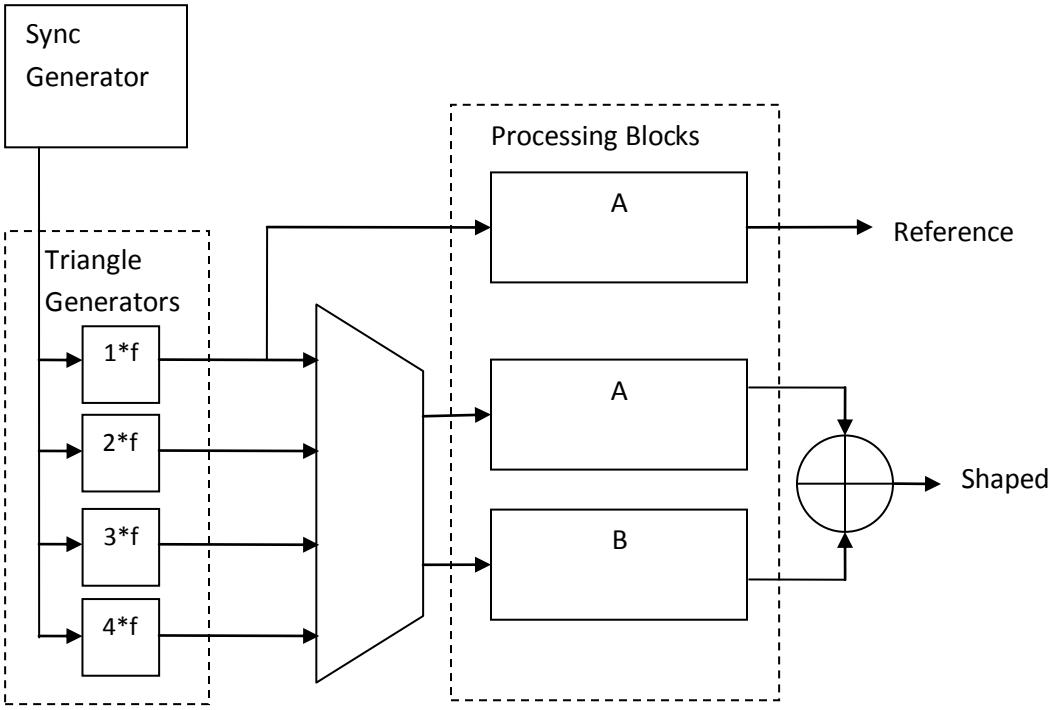

Figure 2: High level system block diagram. The 'A' processing blocks can produce triangles and sinusoids, while the 'B' block produces delta pulses. These are adequate to display a compelling variety of character shapes (see Figure 1)

2.2. Note on Modularity

One of the things which I found most pleasing about this design is the way it leveraged modularity in a way that I have not often used before. Yes, modularity is great for enabling different stages of a system to be designed independently, but this might be the most extensively that I have been able to reuse particular blocks throughout the design, which is something I am particularly proud of. For example, in Figure 2, all four of the triangle generators used the same circuit topology and element values, but could be independently tuned to the different frequencies. In the processing stage, not only did both the shaped and reference signals use the same 'A' processing topology, but both the 'A' and 'B' processing circuits used the same phase-shifting comparator (see Figure 5). Assembling, testing, and debugging such a large circuit would have been extremely difficult without these common topologies.

3. Design Overview: Details

3.1. Sync Generation

One of the central challenges in this design was the generation of signals at precise, integral frequencies and stable, matched phase. This might have been straight forward using digital counters as frequency dividers, but this would have violated the spirit of the project. Another possibility would have been to use a PLL, but that would certainly have been beyond my ability to implement with discrete components. Given these constraints, I decided to construct four independent signal generators that are externally synchronized at a fixed interval. While it was relatively simple to implement, it required each of the four generators to be delicately tuned to a precise frequency to avoid horrible clipping and distortion, making this a weak feature of the design.

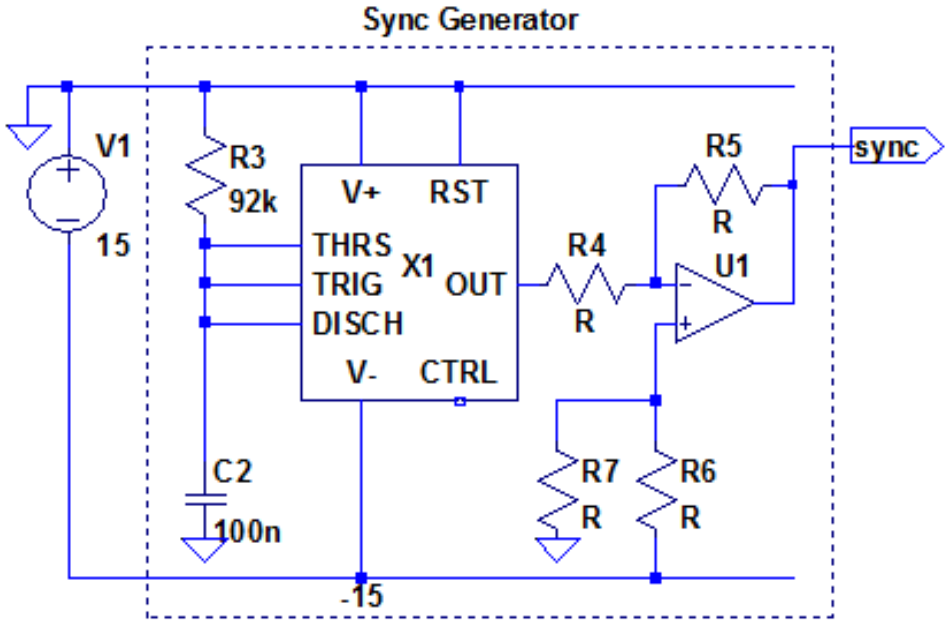

Figure 3: A 555 sync pulse generator. The resistor values for the op-amp inverter have not been specified because they have little effect on the circuit's operation.

The circuit that produces the timed sync pulses is shown in Figure 3. This is just a simple astable oscillator set to operate at around 100 Hz. Notably, because there is little or no resistance on the discharge path (a choice that might bear reconsideration for the sake of safety), the circuit emits a very

short pulse once per cycle. U1 acts as a simple inverter. The actual process of synchronization takes place in the triangle generators.

3.2. Triangle Generators

The triangle generator circuit is shown in Figure 4. In an earlier implementation of this design, the sync signal was a low pulse that was driven into the 555 timer's reset input. While at first this seemed to accomplish the goal of forcing a discharge of C1, the 555's reset behavior has subtly different timing from a normal discharge, which was leading to undesirable distortion. I solved this problem by instead using the configuration in Figure 4, with a diode (D1) and a large resistor (R2) acting as a crude kind of analog OR. Although the tuning of the frequency generators was still delicate, this change made the generated signals dramatically cleaner and more stable. The second half of the triangle wave generator consists of a comparator and integrator that convert the sawtooth to a square wave, and then to a triangle wave.

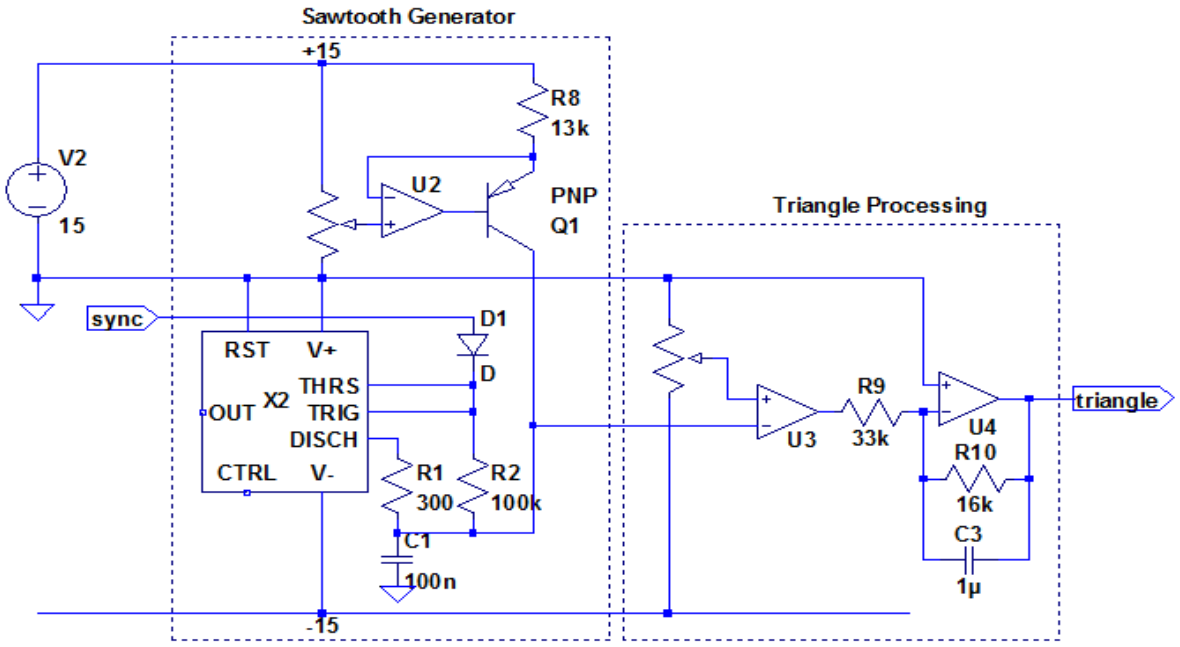

Figure 4: A synchronized 555 triangle wave generator. The full system had four such generators, tuned to different frequencies using the voltage control on U2.

3.3. Processing Blocks

Figure 5 describes the processing pathways used to generate shaped signals for character generation. Both circuits shown have the same basic processing path. First, use a Schmitt trigger with adjustable hysteresis to create a square wave with a particular phase shift from the base signal. From the square wave, either integrate or differentiate to produce different wave shapes. The phase shift turned out to be one of the coolest features of the system. It gave a range of somewhat less than 90 degrees, allowing "rotation" of the displayed image.

One negative consequence of using integrators with a fixed cutoff frequency is that the same integrator is meant to be used at four different frequencies. This means that higher frequency base signals end up being more heavily attenuated by double integration, which made sinusoidal shapes fairly tiny and unstable. To some extent, this could be solved with some adjustable gain at the ouput.

Unfortunately, the triangle waves often contained barely perceptible 100 Hz artifacts from synchronization, which sometimes dwarfed the higher frequency sinusoids because the 100 Hz components were less attenuated by the integrators. This is one of the reasons the frequency tuning had to be so delicate. In practice, I addressed this problem by replacing R5 (in Figure 5) with a potentiometer, allowing both the gain and the effective pass-band of the second integrator to be adjusted dynamically.

In addition to the differentiation and integration circuits that are shown in figure 5, there were several later processing blocks that allowed, for example, rectification of the delta pulses that were produced by the differentiator block. Notice, for example, in figure 1, that the 'N' shaped waveform has both negative and positive pulses, whereas the 'Y' waveform has only negative pulses. So, characters like 'Y,' 'T,' or '7' could not be depicted at all without rectifying the delta waveform. This feature was omitted from the figure because the rectification circuitry was simple, but the switching network was complex, making it difficult to diagram concisely. Luckily, the existence of those circuits is more important than the details of their topology.

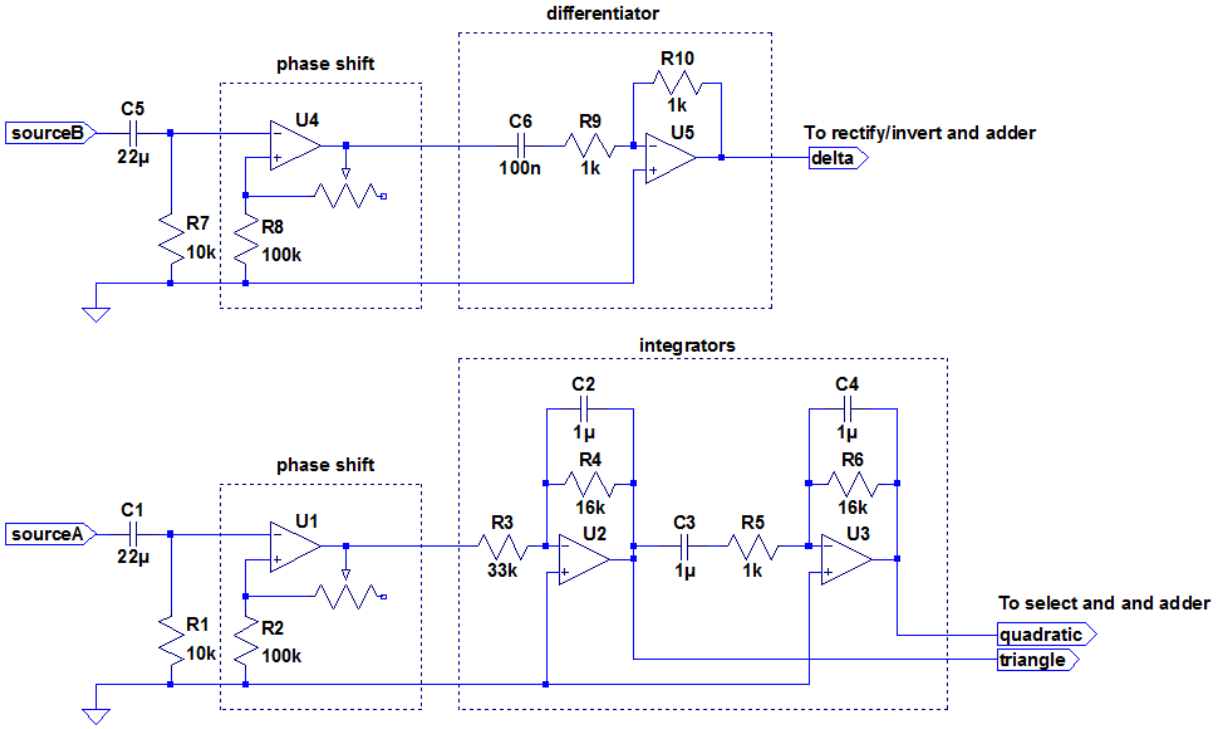

Figure 5: Processing blocks. The top circuit corresponds to the 'B' block in figure 2, while the bottom circuit corresponds to 'A.' The source signals for both circuits are triangle waves.

3.4. Etcetera

For the sake of brevity, these circuits have omitted the inter-stage selection and additional processing circuitry that were necessary for the system to actually work. In Figure 2, these are represented by a generic multiplexer and an adder block. Simply be aware that the full system included a network of switches and some op-amp adders and inverters that were necessary for operation, but whose implementations were not particularly interesting or noteworthy.

4. Outcomes

4.1. Performance

The system's performance was fairly satisfactory, overall. It was able to display a reasonable variety of characters, which consisted of:

E F I L M N O S T U V W Y Z 1 2 3 5 7 8

There were a few flaws in its performance. Because the tuning of the triangle generators had to be so precise, any error in this tuning would result in wildly unstable or distorted images, and it was often difficult to really eliminate this effect, particularly for the sinusoidal shapes. Perhaps the most fatal flaw in this design is that the arduous tuning prohibits the rapid generation of different characters. This means that it would be exceedingly difficult, if not impossible, to adapt this design to display a string of text. More than any other part of the design, this damages the essential purpose of the system: what good is a text generator if it can only generate one character at a time? While this was disappointing, the direction of the project had shifted more towards a waveform generator, rather than a robust or complete text display. In this respect, it was still successful.

4.2. Further Work

As mentioned in section 2.1, there are a few features that would have broadened the range of images this display could produce, but which were not implemented due to time. One of the most enticing would have been the addition of a third control signal: the gate. Carefully timed pulses on this input would break the display beam, enabling characters like 'C' or, 'X,' which would otherwise be quite difficult to draw. Other possibilities would basically involve adding more shaping options to the display signals, such as adding delta pulses to the reference signal (useful for making 'A'), or rectifying a sinusoidal shaped signal to make characters like 'D' and 'B.' The staging of the design would allow these features to be added independently of one another, and without dramatically altering the existing stages.

One other possibility intrigues me: while talking to other students in the lab, someone brought up the idea of making a circuit that could do multiplications with rotation matrices. This strikes me as a wildly cool idea, and could be used to great effect in similar analog display projects, either making a rotating x-y image, or even making an angular "sonar sweep" effect. This would be a lot of fun to explore further.

5. Conclusions

I am under no illusion that the system I created is a practical, or even plausible, solution to the problem it attempts to solve. However, the fun and difficulty of this project was not in making a system practical or effective in its own right, but rather in solving a familiar, even trivial, problem in an unconventional way. While I see no good reason to ever generate text images with analog waveforms, this work has opened my eyes to many of the surprising subtleties of signal generation and circuit design.

Although the external synchronization worked well enough for this project, it turns out to be a poor way to produce clean, stable waveforms. A better design would use PLL's or digital counters to create signals at different frequencies or control phase shifts, and this would both simplify the circuitry as well as reduce the need for delicate tuning. It might also use clean sinusoids as base signals, which would avoid the attenuation problems caused by double integration. Unfortunately, phase comparators

and other such circuits for working in frequency control are yet foreign to me. This more ambitious approach would be an exciting target if I were to try a project like this again.