Mini Project 3 Report

The goal of this project was to produce a sinusoidal waveform through a 10-bit R-2R ladder digital-to-analog converter while minimizing used memory storage by leveraging the symmetries in a sine wave. The code written for this project pre-stores 128 sine values representing one-quarter of the sine wave and then uses a finite state machine to cycle through these values, reconstructing the full sine wave through symmetry. The code consists of 3 modules: a memory module that stores the sine wave samples, a finite state machine module that controls how the waveform is sequenced, and a top module to connect the IceBlink Pico to the 10-bit R-2R ladder.

The memory module is a read-only memory module that is initialized with a text file, sine.txt, that holds 9-bit sine wave samples. It uses the address to locate the memory and retrieve its sine value.

The finite state machine sequences through the memory addresses to reconstruct the sine wave by defining four states within the sine wave (0° to 90°, 90° to 180°, 180° to 270°, and 270° to 360°). For the first state it reads the memory sequentially, but for the second state it reads the memory in reverse as the first and second quadrant of a sine wave are symmetrical. The third and fourth quadrants follow the same forward and backward pattern as the first and second, except they are inverted. Thus, the data is read as inverted as well. By using the same 128 bits of stored information and translating it cleverly, the FSM can transition through these defined states every 128 clock cycles and still output a sine wave.

The top-level module is perhaps the simplest. It instantiates the FSM and connects its output to specific pins on the IceBlink Pico.

When tested in simulation, the code produced a sine wave as shown in the screengrab below. It also worked on the hardware, and produced a sine wave form as measured by the oscilloscope, although I admittedly did have to resolder the pins as I put them on backwards the first time.

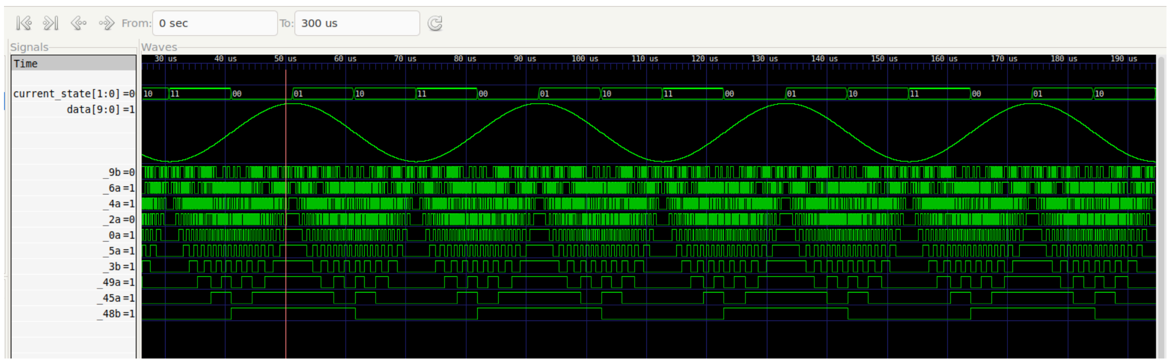

Fig. 1: Screen grab of the GTK wave produced by the code