

# Analog Shield Application Note 1: Direct Digital Synthesis Function Generator

Revised June 5, 2014  
Author: William Esposito

*This document was edited and adapted for MPIDE by Digilent. The original document was prepared by William Esposito at Stanford.*

## Overview

The Analog Shield Direct Digital Synthesis (DDS) Function Generator is a compact, low-frequency function generator capable of producing sine, square, triangle, or sawtooth waveforms (Fig. 1).

DDS works by means of “playing back” a prerecorded waveform stored in a lookup table at a precisely controlled rate. One full period of the waveform is stored in an array on the Arduino™. To create a wave output, the Arduino sends a single sample of the stored waveform out via the analog shield. It moves through the table outputting samples until it reaches the last sample in the array. Once it reaches the end, the program returns to the beginning of the table and repeats. In order to put out a waveform at a higher frequency, the program skips steps in the table. For example, if a step of 1 produces a 1Hz output, then outputting every 10<sup>th</sup> sample (a step of 10) would produce a 10 Hz output. By calculating how many samples it has to skip, it can precisely generate any frequency it needs, up to half the rate at which it puts out samples (the Nyquist rate, where there are a minimum of two samples per waveform period, which assumes a sine wave. More samples per period are necessary for more complex waveforms).



Figure 1. The Analog Shield DDS.

## Hardware

The Analog Shield DDS is built around the Arduino and uses the TI/Stanford Analog Shield (Fig. 2) for signal output, and is available to purchase off of the Digilent website. The Analog Shield DDS also uses the Adafruit® RGB LCD Shield kit for its user interface.

The Arduino UNO™ R3 can be found at:  
[http://store.arduino.cc/index.php?main\\_page=product\\_info&products\\_id=195](http://store.arduino.cc/index.php?main_page=product_info&products_id=195)

The Adafruit RGB LCD Kit can be found at: <https://www.adafruit.com/products/714>

The Analog Shield can be found at: <http://www.digilentinc.com/analogshield>



Figure 2. Analog Shield.

In order to get a meaningful signal at more than a kilohertz, the DDS also needs a “reconstruction” filter on its output. The quality of the reconstruction filter will make a big difference in the quality of the signal produced and it is discussed in its own section below.

## Building the Demo

In order to build the Analog Shield DDS, four nonstandard Arduino libraries are necessary. For help with the libraries, read the *First Time Setup* guide, or go to <http://arduino.cc/en/Guide/Libraries>.

The libraries required for the DDS are:

- The Adafruit RGB-LCD Shield and MCP23017 libraries (For the LCD ) <https://github.com/adafruit/Adafruit-RGB-LCD-Shield-Library>
- The TimerOne library (for the DDS sample clock) <https://code.google.com/p/arduino-timerone/>
- The Analog Shield library (for analog input) <http://diligentinc.com/analogshield>
- In case these libraries move in the future, the Adafruit LCD libraries are linked from the Adafruit Touch LCD Shield tutorial at <https://learn.adafruit.com/rgb-lcd-shield/>
- The code for the DDS is available with this document at <http://www.diligentinc.com/analogshield>.

A detailed discussion is included below in the *How the Code Works* section of this document.

Once these libraries are downloaded and unzipped in the Arduino/libraries folder, open the Arduino IDE and the “DDS\_Generator” sketch. It is important to note that if the Arduino IDE was already open before the libraries were added, the IDE will need to be restarted to recognize the libraries.

Stack the LCD and Analog Shield on your Arduino. Next, click the “Upload” arrow in the top left to upload the DDS Function Generator to the Arduino. Once the upload is complete, a menu will appear on the LCD.

## Controlling the Demo

At startup, the Arduino waits for the user to set the frequency and wave output type.

To change the frequency, press up and down. The left and right buttons move the cursor and selects which digit the up and down buttons effect, as displayed in Fig. 3. Moving the cursor past the lowest digit allows selection of the wave shape (sine, square, sawtooth, triangle).

Once a timeout (5 seconds) occurs or the user presses “select,” the system will output the desired wave. When running, the word “run” displays in the upper right corner of the LCD, as displayed in Fig. 4.

To set a new waveform, press reset to return to the menu.

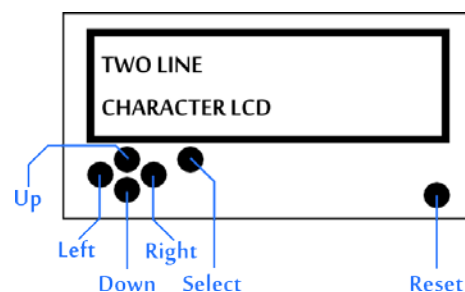


Figure 3. DDS Controls.



Figure 4. LCD with cursor and DDS running.

## Output

Now that there is a signal coming out of D0 on the analog shield, the next step is to do something with the signal. One option is to directly connect the output (D0) to an oscilloscope (as well as a ground from the shield to the ground on your oscilloscope probe, to complete the circuit).

With a direct output, an oscilloscope will show a fairly clear waveform at low frequencies (up to 1kHz or so), as displayed in Fig. 5, although it will have a somewhat stair-step nature. At higher frequencies, the waveform will break down. This is due to the nature of the DDS. At higher frequencies, the DDS skips more steps and each step will, therefore, make a larger vertical jump. At a high enough frequency, the signal begins to look like a rolling square wave. To allow the DDS to work at higher frequencies, a reconstruction filter is needed, which will be addressed below.

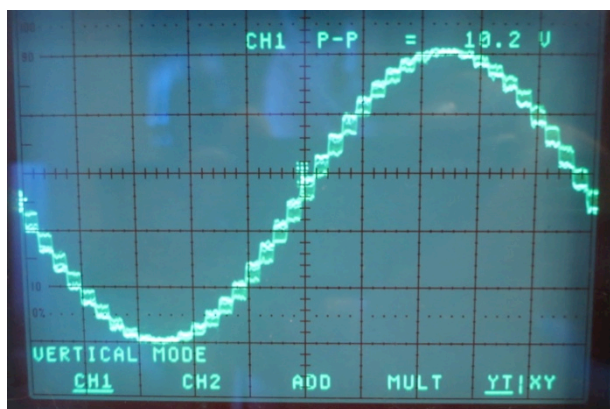


Figure 4. 1 kHz sine wave with no reconstruction filter.

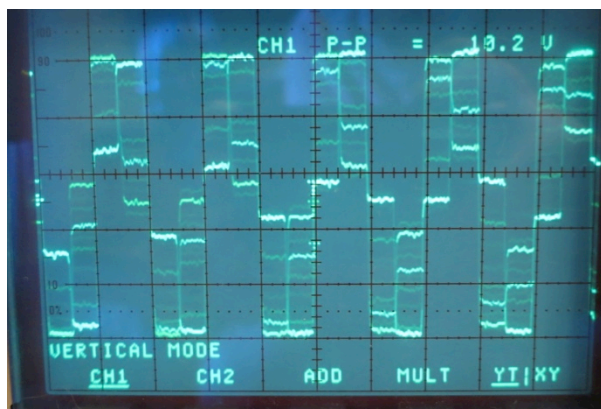


Figure 3. 10 kHz sine wave with no reconstruction filter.

If an oscilloscope isn't handy, the output can be heard with headphones. One advantage of this is that most headphones naturally act as a low-pass filter with a cutoff at or below 20 kHz. To connect your headphones, you will need a **headphone jack** and a **resistor with a value of at least 3.3 kΩ**. The resistor is critical because it will limit the volume output through the headphones, saving your ears and your headphones from damage. Figures 7 and 8 show this circuit.

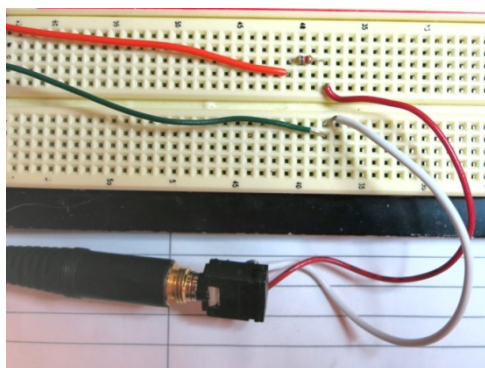


Figure 6. Headphone jack output. Orange wire (top) to D0, green wire (bottom) to the ground on the Analog Shield.

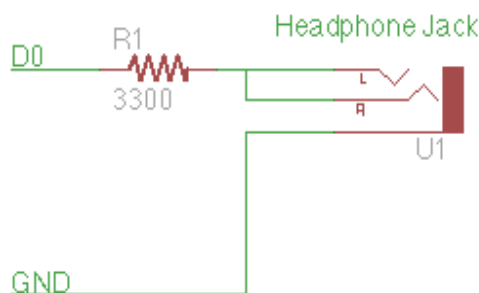


Figure 5. Headphone jack connector circuit

## Reconstruction Filter

To improve the DDS, a reconstruction filter can be added. The purpose of the reconstruction filter is to remove the stair-step nature of our sampled sine wave. Figure 9 shows the relatively low frequency 1 kHz output after passing through the high quality filter, which will be discussed in a moment. Note the drastic improvement in this output when compared to the same wave seen without a filter in Fig. 5.

The reconstruction filter should be a low-pass filter that removes the sharp (40 kHz high frequency) steps in the output without weakening the lower frequency (1Hz – 16 kHz) signal, which is the actual intended output.

The simplest filter that can be implemented is a passive RC (resistor-capacitor) low-pass filter. Such a filter will improve the signal, but due to its simplicity, it will only weakly attenuate higher frequencies. This means that the resulting signal is less of an improvement when compared to a better filter.

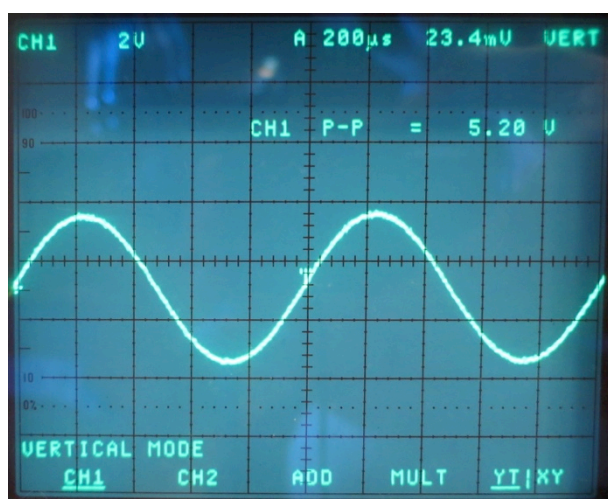


Figure 8. 1 kHz sine through high quality filter.

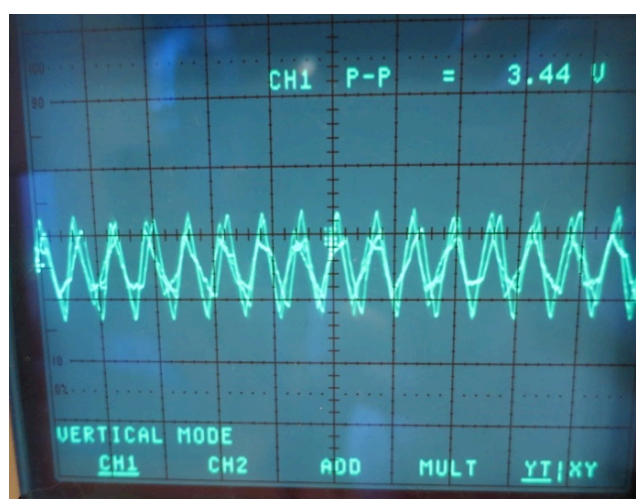


Figure 7. 10 kHz signal through RC filter, with the same scale as figure 9.

To make a simple RC Filter, connect a resistor of 2 kΩ between D0 on the Analog Shield and our output, and a capacitor of 22 nanofarads (nF) between our output and ground, as shown in Fig. 11. This will create a filter with a “cutoff” (half power) point at:

$$f_{cutoff} = \frac{1}{2 * \pi * 2000 * .000000022}$$

$$f_{cutoff} = 3617 \text{ Hz}$$

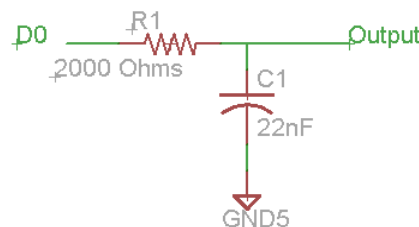


Figure 9: Simple RC filter.

This will help our resulting signal by attenuating high frequency noise. It will not provide optimal performance, however, as the signal of interest is often above 10 kHz and will be attenuated as well. Figure 10 shows the 10 kHz signal as observed through this RC filter. Note that it is closer to a sine wave than Fig. 5 above, but is still a poor quality output.



To maximize the frequency that can be reproduced with the DDS, a complicated 8<sup>th</sup> order active low-pass filter can be implemented. Such a filter uses operational amplifiers (op-amps) to allow numerous successive stages of high frequency removal without a loss in signal strength. Figure 12 shows the same signal as Fig. 10, passed through one of these high quality filters.

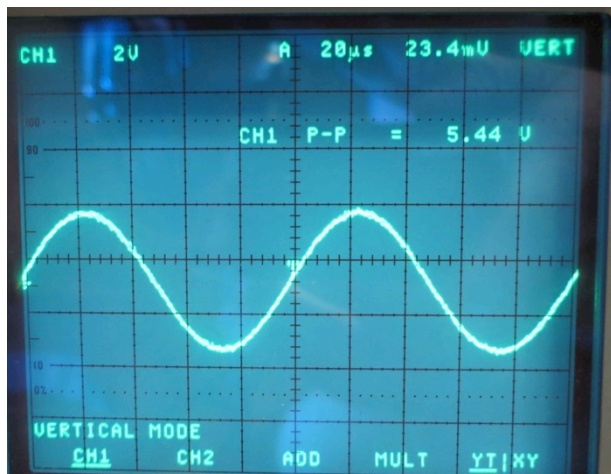


Figure 11. The same 10 kHz sine wave through an 8<sup>th</sup> order reconstruction filter, with an 18 kHz cutoff.

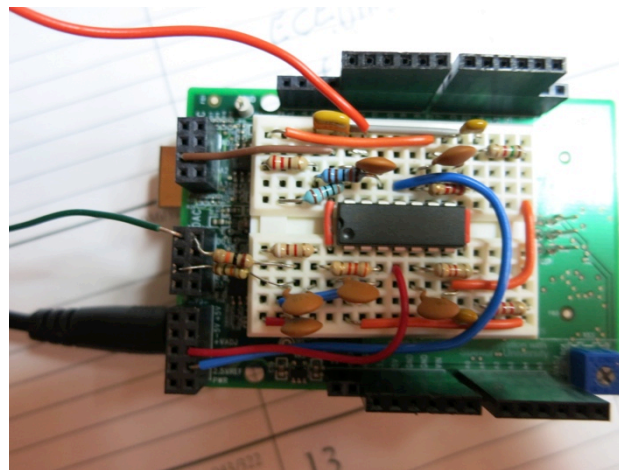


Figure 10. 8<sup>th</sup> order reconstruction filter with an 18 kHz cutoff constructed on the prototyping area of the Analog Shield.

A full filter design has been attached to this document for an 8<sup>th</sup> order filter with an 18 kHz cutoff that will allow for a clear DDS output up to a 16 kHz sine wave. The design for this filter has been included as Fig. 17 at the end of this document.

There is a caveat, however. Sometimes, the desired output waveform needs to have sharp edges. In an extreme example, the DDS can output a square wave, which has the sharpest possible edge. By passing that signal through the reconstruction filter, the output waveform would be reduced to an approximation of the sine wave output at 10 kHz (see Figs. 14 and 15). This is due to the nature of a square wave itself, whose sharp edges are the result of harmonics which are attenuated through the reconstruction filter.

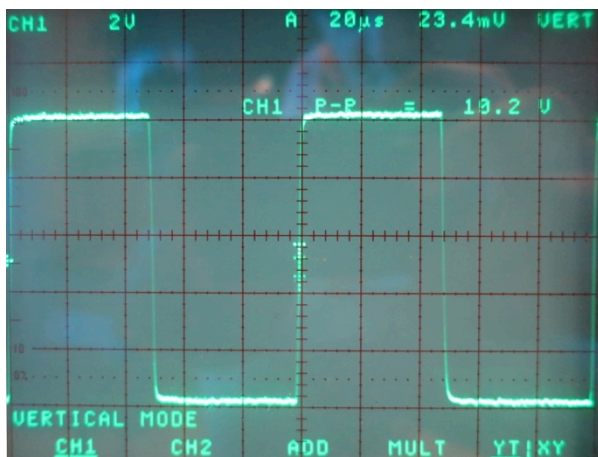


Figure 13. 10 kHz square wave with no reconstruction filter

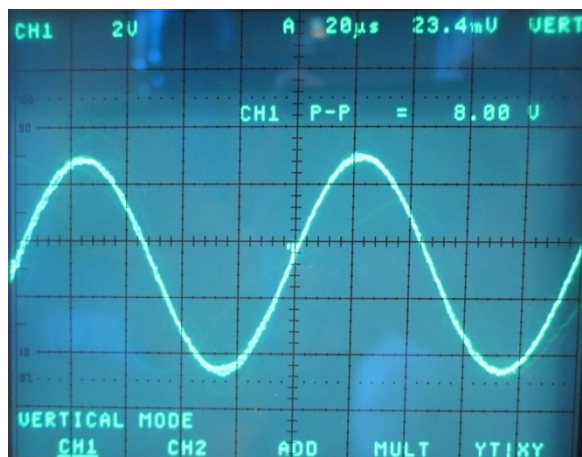


Figure 12. 10 kHz square wave through 8<sup>th</sup> order reconstruction filter with an 18 kHz cutoff.

## How the Code Works

On the Analog Shield, the DDS has been implemented in a very standard configuration.

The DDS stores a phase offset, which represents a position in a hypothetical infinite waveform. Each sample cycle, the DDS routine increments that phase offset by its stride size (tuning word). Since only one period of the waveform is actually stored in memory, the DDS computes the offset with respect to a single wave cycle and uses that as a table lookup. The resulting value is then sent to the DAC for conversion to a voltage output. By storing the phase in a higher precision word than the table, the program avoids having to perform complicated carry offset operations every time it generates a sample (Fig. 16).

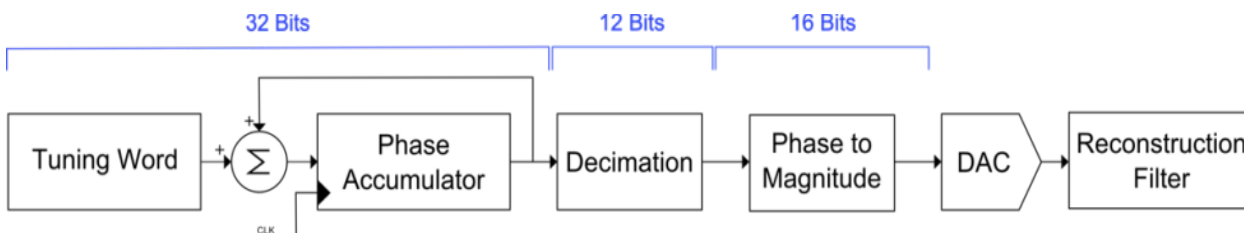


Figure 14. DDS Software block diagram.

The actual DDS routine is moderately complex as it supports multiple waveforms and controls the LCD and menu system. A shortened version of the main code, which can be found below, shows how the block diagram can be implemented with careful coding.

```

/* Direct Digital Synthesis */

//12 bit long (4096 length) sine wave table
PROGMEM prog_uint16_t isinTable16[] = {<sine wave table omitted for space>};
#include <analogShield.h> //Analog Shield
#include <avr/pgmspace.h> //Sine Wave Storage
#include <TimerOne.h> //Timer Control

//DDS variables
volatile unsigned long tuningWord = 0;
volatile unsigned long phaseAccumulator = 0; //32 bits
bool toggle = false;

void setup(){}

void loop() { //takes user input and the goes into DDS forever.
    Timer1.initialize(16); //Setup the timer for DDS
    Timer1.attachInterrupt(dds); //start DDS
    while (1) {}; //wait forever. Reset Arduino for new frequency.
}

//DDS code
void dds(){ //Direct Digital Synthesis Lives Here.
    unsigned int value; //value to output on DAC
    unsigned long tempPhase; //calculated phase for this sample

    //Increment the phase accumulator by the value stored in the tuning word.
    phaseAccumulator += tuningWord; //32 bits
  
```

```
//use only top 12 bits of 32 bit phase accumulator
tempPhase = (unsigned long)(phaseAccumulator >> 20);
//Get Sine Value
value = pgm_read_word(isinTable16 + tempPhase);
//write the result to the output.
analog.write(0, value);
}
```

## Additional Caveats about the DDS

The waveform of a DDS itself is band-limited by the sample clock used to drive the digital-to-analog converter. In the case of the Analog Shield DDS, the sample clock runs around 40,000 samples/second. This means that a sharper edge (as found in the square and sawtooth waveform) can't be accurately reproduced. This phenomenon results in apparent jitter on the sharp edges of the waveform, as they appear to move back and forth in time by one sample.

## Notes

[1] Maxim Semiconductor has produced a whitepaper, MT-085 which explains DDS much more rigorously. It can be found at: <http://www.analog.com/static/imported-files/tutorials/MT-085.pdf>

**Disclaimer:** This code and circuit was developed by William Esposito, Ph.D. Candidate in Electrical Engineering, Stanford University, in the Kovacs/Giovangrandi Laboratory in collaboration with Texas Instruments, Incorporated. All code herein is free and open source, but is provided as-is with no warranties implied or provided. Use of this code and the associated documentation is at the user's own risk.

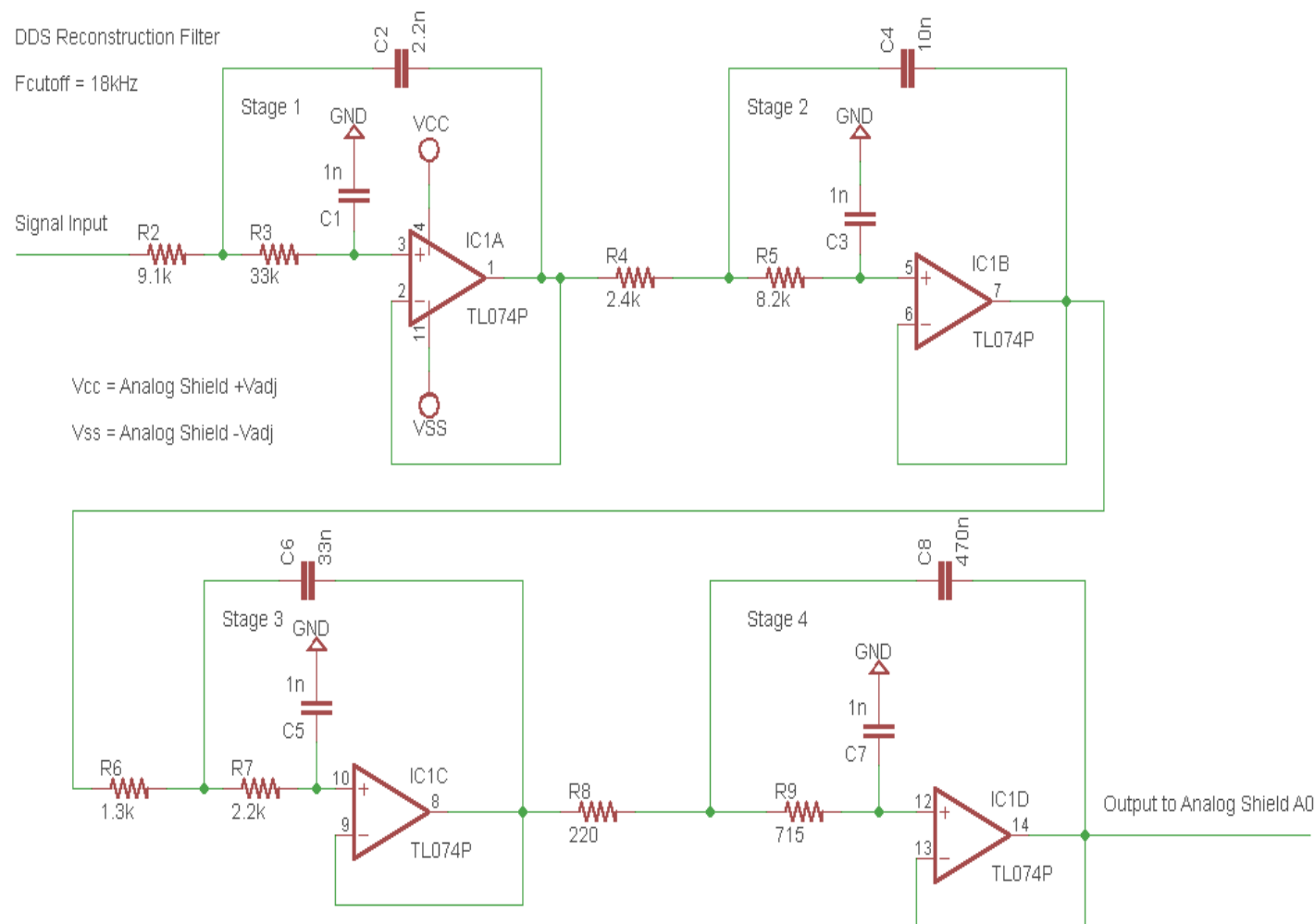


Figure 15: DDS reconstruction filter with 18 kHz cutoff.