Ray Mack, W5IFS

17060 Conway Springs Ct, Austin, TX 78717; w5ifs@arrl.net

SDR: Simplified

We resume the SDR series in this issue with a look at Cascaded Integrator Comb Filters.

The past year has been full of activities that kept me away from writing this series. That is all behind me now, so we can get back to experiments and learning about software defined radios.

Introduction to CIC Filters

We looked briefly at the theory of the sampling down converter (decimator) in the Nov/Dec 2009 issue and the sampling up converter (interpolator) in the Jan/Feb 2010 issue. An interpolator is typically used in a transmitter to increase the sample rate of a signal in preparation for frequency translation to the final frequency. A decimator is typically used in a receiver to lower the sample rate and also translate the signal to a lower frequency. In the interpolator, we add zero samples in between our existing samples and then low pass filter the sequence to eliminate images of the original. The result is a new sequence of samples that has the same spectrum as the original but with samples at a much higher rate. A decimator works similarly, but we throw away existing samples and low pass filter the sequence to eliminate unwanted aliases of higher frequencies. This new sequence also has the same spectrum (or an aliased spectrum) as the original, but with a lower sample rate.

The filter for an interpolator or a decimator can be a finite impulse response (FIR) or infinite impulse response (IIR) filter. The problem with both types of filters is that they require a large number of multiply operations, which consume a large number of DSP processor cycles. Eugene Hogenauer developed a very useful simplification of the sample conversion/filter configuration called a cascaded integrator comb (CIC) filter. He presented this design in an article in the *IEEE Transactions on Acoustics, Speech and Signal Processing* in April 1981.[1] The important aspect of CIC filters is that only addition, subtraction, and delay operations

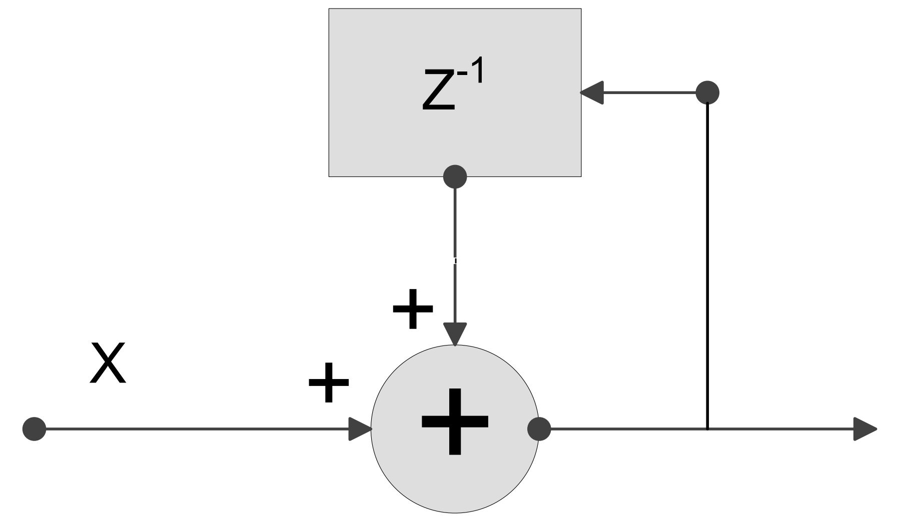

Figure 1 — Z Transform diagram of an integrator. The new output is the sum of the previous output and the new sample

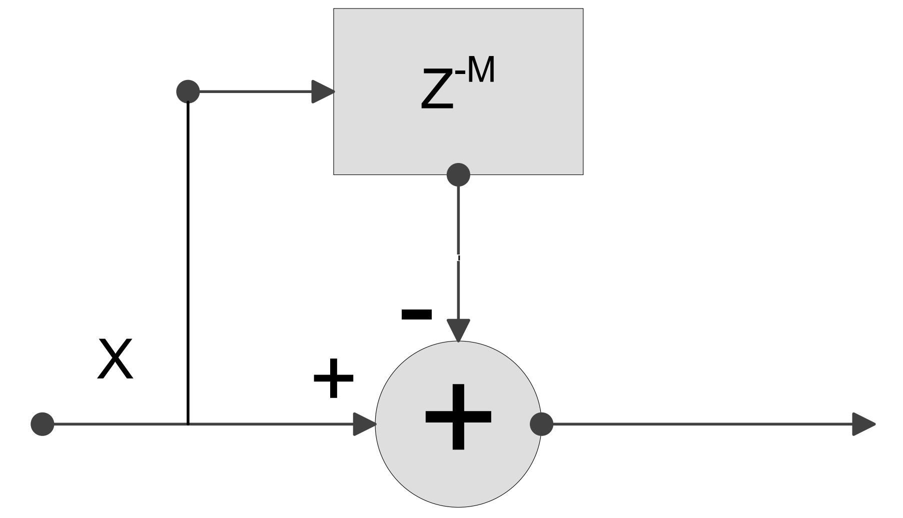

Figure 2 — Z Transform diagram of a comb. The new output is the difference of the present sample and a delayed sample. The number *M* designates how many steps happened during the delay.

[1]Notes appear on page 36.

are required for implementation.

As with most things in life, improving one aspect of a system requires compromise in other aspects. This is also true of CIC filters where we trade the simplification of eliminating multipliers for restricting the filter response. A CIC filter can only be low pass. Additionally, there is a limited subset of possible low pass responses constrained by the sample rate change and number of stages in the comb and integrator stages. The most important property of a CIC filter is that it can be very easily implemented in hardware either in an FPGA or as part of the dedicated logic of an IC such as the AD9874 and AD9957.

How a CIC Filter Works

Matthew Donadio has written a very good description on his website of how a CIC filter works, along with the associated mathematics (in case you want to see what the Z-transform equations look like).[2] I have borrowed several of his examples. We haven't used standard Z-transform graphical notation up to now, but I believe it will help you understand the framework.

The integrator is an infinite impulse response filter. Figure 1 shows how it works and how simple it is. The integrator holds a running total of all previous samples. The integrator adds the last output value (z^{-1}) to the current input value (x). Ordinarily, we would worry about overflow in an integrator because a dc component in the signal will cause the integrator to overflow. The combination of the comb and the integrator, however, cancels any problems with overflow (see Donadio for details). The integrator is a single pole low pass filter with infinite

Listing 1
GnuPlot

```
R=8
N=4
M=1

set angles radians
#set the grid lines to dots
set grid linetype 13, linetype 13
#turn on the x and y tic marks
set grid mxtics
set grid mytics
#set the intervals for major and minor grids
set xtics 0.05
set mxtics 2
set mytics 5
#set a large number of samples to create a smooth plot
set samples 100000
#set the x axis to span fs*-0.125 to fs*0.5
set xrange [-0.125:0.5]
#set the y span from -80dB to 0 dB
set yrange [-80:0]
set ylabel "dB"
Set xlabel "Frequency (f/fs)"
sinc(x) = (sin(3.14 * x * R))/(sin(3.14 * x))
CIC(x) = abs(sinc(x))
db_response(x) = (20 * log10 ( (CIC(x)**N)/(R**N) ) )
plot db_response(x) with lines 1
```

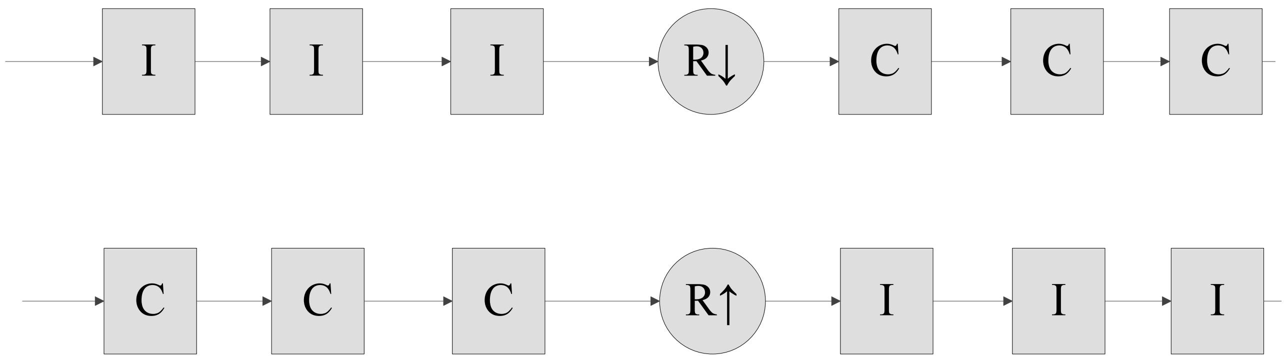

Figure 3 — The top diagram shows a decimator, where the sample rate is reduced by a value "*R*." The bottom diagram shows an interpolator where the sample rate is increased by a value "*R*." Note that the difference between the two is the order of the combs and integrators as well as the direction of the rate change. The combs are on the low sample rate side of both systems.

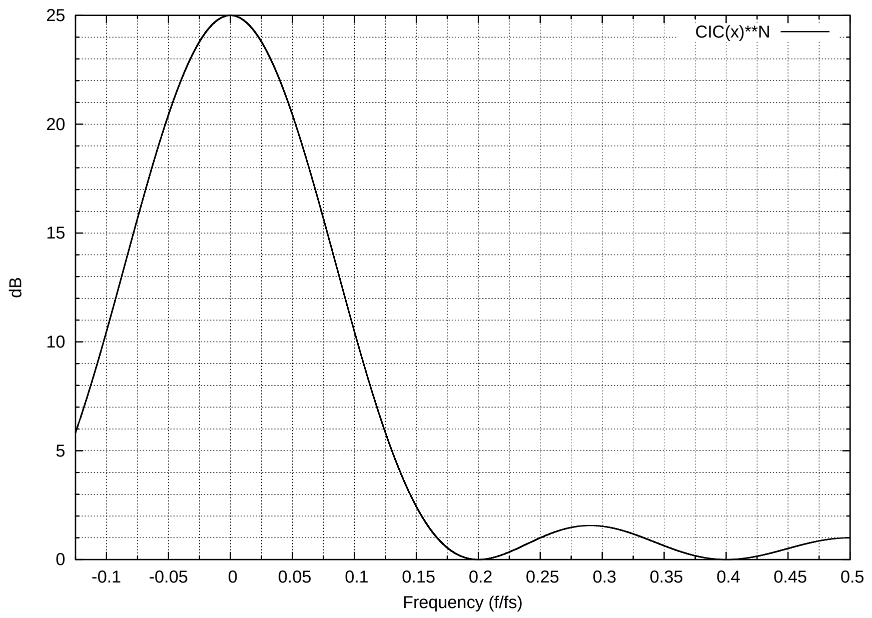

Figure 4 — The sin x/x shape of a CIC filter. The lobes decrease quickly because the sin x/x function is raised to a power.

Listing 2
C Program CIC filter

```
int accumulator_1, accumulator_2, accumulator_3, sample;
int comb_1_out, comb_1_delay;
int comb_2_out, comb_2_delay;
int comb_3_out, comb_3_delay;
#define RATE_CHANGE   9  // 1 + desired rate change

int integrator (int new_sample, int integrator_accumulator)
{
  return (accumulator + new_sample);
}
int main(void)
{
int i;
  i = 0;
  while (1)
  {
    sample = read_adc();
    // do the accumulation in reverse order so it ripples
    // in the correct way
    accumulator_3 = integrator(accumulator_2, accumulator_3);
    accumulator_2 = integrator(accumulator_1, accumulator_2);
    accumulator_1 = integrator(sample, accumulator_1);
    if ((i % RATE_CHANGE) == 0)
    {
      comb_3_out = comb_2_out - comb_3_delay;
      comb_2_out = comb_1_out - comb_2_delay;
      comb_1_out = accumulator_3 - comb_1_delay;
      comb_1_delay = accumulator_3;
      comb_2_delay = comb_1_out;
      comb_3_delay = comb_2_out;
      write_output_dac(comb_3_out);
      i = 0;
    }
    i++;
  }
}
```

gain at dc. Hogenauer figured out that the system doesn't care about overflow as long as the integrators are implemented with adders using two's complement addition that allow wrap around when overflow occurs, and that the number of bits in the word is as big as the expected output word.

[Wikipedia tells us that "the two's complement of a binary number is defined as the value obtained by subtracting the number from a large power of two (specifically, from 2^N for an N-bit two's complement). The two's complement of the number then behaves like the negative of the original number in most arithmetic, and it can coexist with positive numbers in a natural way." — *Ed.*][3]

The comb is a finite impulse response stage that subtracts a previous sample from the present sample. The amount of delay between the present sample and the delayed sample is called the differential delay and is denoted as M by most authors. Figure 2 shows the operation of the comb. A real implementation of a CIC filter is composed of multiple integrator-comb sections that are cascaded. A CIC filter has exactly the same number of integrators as combs. Remember the associative property of math from elementary school: you can rearrange the order of the additions in a sequence and the result of the sequence does not change ($a + (-b) + c + d + (-e) + (-f)$ is identical to $a + c + d - b - e - f$). A CIC filter with rate change uses that property to group all of the integrators together and to group all of the combs together. We place either a down sample or up sample rate changer between the combs and integrators. Figure 3 shows that a decimator is an integrator section followed by a down rate change, which is then followed by a comb section. An interpolator turns the system around and puts the comb section first, followed by an up rate changer, which is followed by an integrator section. It is very useful for a hardware implementation that the number of integrators and combs is independent (within reason) from the rate change and that, in general, you can rearrange the inputs, outputs, and rate change to create a decimator and interpolator with the same blocks.

Richard Lyons has a third way of explaining the operation of a CIC filter.[4] Unfortunately, Lyons, Donadio, and Hogenauer do not do a really good job of connecting the relationship between the rate change and the differential delay in the comb. Lyons comes closest when he describes the sinc shape in terms of the differential delay in the comb section. What he did not explicitly say is that the sinc shape of a combined comb and integrator is the product of the rate change (*R*) and *M*. Both Donadio and Lyons express the number of nulls in the CIC response as a function of the differential delay, *M*, but they leave out the very important requirement that it applies when *R* = 1 (no rate increase or decrease). A normal implementation places the rate change ahead of the comb, so the effect is to transfer the number of nulls from *M* to *R*. When we add a rate change, we swap the values, so *R* has a value greater than one and *M* = 1.

Implementation Details

There are three parameters that affect the implementation of a CIC filter. "*R*" is the up sample rate or down sample rate. "*M*" is the delay in the comb section and is almost always either one or two. "*N*" is the number of stages in the comb section (which is required to be the same as the number of integrator sections). The simplification of separating the combs into one section and the integrators into a second section is advantageous for the speed required of the storage elements for the combs. The combs always operate at the low frequency end of the system and the integrators work at the high speed side of the system.

Notice that the associative property would also allow a decimator with the comb first and the integrator after the down converter. Either configuration will give the same results. The reason we always put the comb on the low sample rate side of the system is pragmatic. The comb requires one additional storage element (for the usual *M* = 1 situation) over what is required for an integrator. Each storage element consumes power when it is clocked. A faster clock and more storage registers translate directly into additional power dissipation. The additional power is an issue when implementing a CIC filter in hardware such as an FPGA.

Figure 4 shows the (sin *x*) / *x* (also called the sinc function) shape of the frequency response of a CIC filter. The full equation is:

$$\mathrm{H}(f) = \left[\frac{\sin \pi M f}{\sin \frac{\pi f}{R}} \right]^N \qquad \text{[Eq 1]}$$

If *x* is very small, then sin(*x*) is approximately *x*. When *R* is a large value the denominator becomes $\pi f / R$ so you can approximate the shape of the frequency response in the first region (with the help of a lot of Algebra) as:

$$\mathrm{H}(f) = \left[RM \frac{\sin \pi RMf}{\pi RMf} \right]^N \qquad \text{[Eq 2]}$$

The frequency response has nulls when *f* is equal to a multiple of 1 / *RM*. We looked in detail at the sinc function when studying the frequency response of a real world DAC (Jan/Feb 2010 *QEX*). A CIC filter suffers from the same frequency droop issue when the output bandwidth is substantial compared to the output sample frequency. Another important characteristic of the CIC filter is that the aliases for decimation and images for interpolation occur centered on the nulls of the response. Figure 5 shows a representation of the response of a decimator with *R* = 8, *M* = 1 and *N* = 4, where the original data is sampled at 48 kHz and the desired bandwidth is 1.2 kHz. The shading shows both the baseband energy as well as a non-trivial amount of energy in the higher frequencies that is aliased into the base band

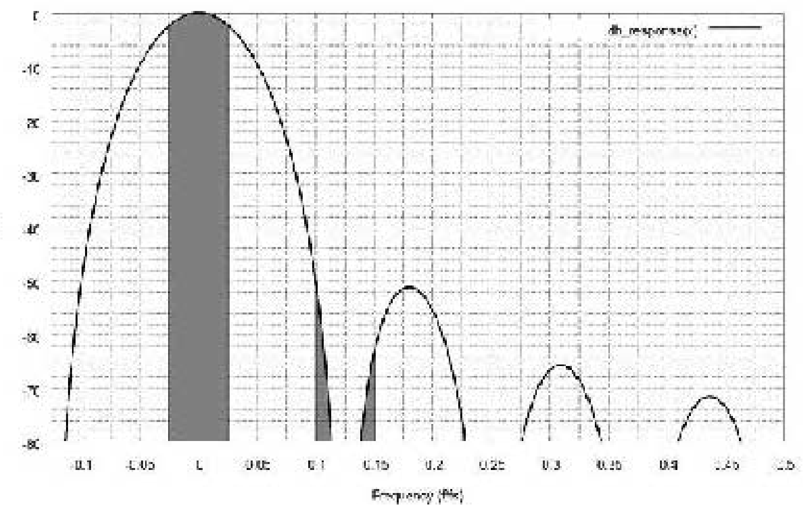

Figure 5 — The response of a CIC decimator filter with *R* = 8, *M* = 1, and *N* = 4. The shaded areas show the energy in the input spectrum that is included in the output of the filter.

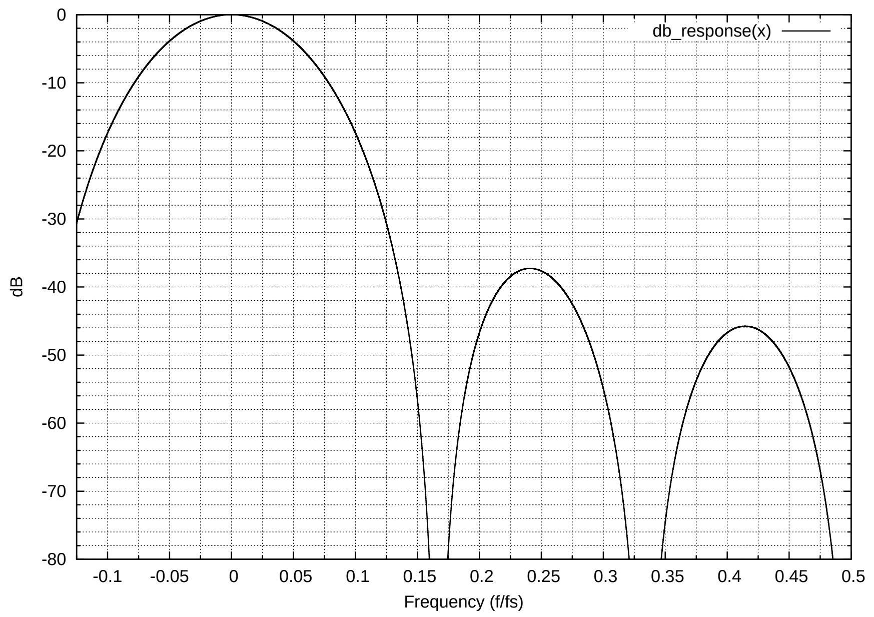

Figure 6 — The response of a CIC decimator filter with *R* = 6, *M* = 1, and *N* = 3. The droop is less in this filter, but alias energy is larger by 10 dB.

and contributes to noise in the system. The alias energy in the first Nyquist zone is down only 52 dB at 1.2 kHz. The baseband energy has 2 dB of droop over the pass band. Figure 6 shows a different filter where $R = 6$ and $N = 3$. Here we see that lowering N and R has improved the amount of droop to only 1 dB and the alias energy only occupies three Nyquist zones. The alias energy in the first Nyquist zone at 1.2 kHz is only down about 42 dB, however. Listing 1 is a *Gnuplot* program that will plot the normalized response of a CIC filter. It was used to create Figures 5 and 6.

The solution to the droop issue is to implement a combination system where the initial rate conversion is aggressive, followed by an FIR droop compensation filter. The compensation filter should have a shape that is the inverse of the sinc shape over the desired baseband frequencies. In our example of a 48 kHz sampled signal that has been reduced by a factor of 8 to a 6 kHz sampled signal, we would want an FIR that increases response up to perhaps 3 kHz to compensate for the droop.

The last issue is bit growth that results from the filter gain. The gain is $(RM)^N$ from Equation 2. The number of bits required at the output of the CIC filter is:

$$B_{OUT} = N \log2 RM + B_{IN}$$

where B_{OUT} is the number of bits in each output word and B_{in} is the number of input bits. Each integrator and comb section will require B_{OUT} bits in order to avoid the problem with overflow in the integrator sections. The bit growth is not likely to be an issue for a DSP processor which will likely have 32 bit registers. It can be a problem for FPGA based systems, however. The integrators and combs need to be built with enough bits to handle the largest combination of sections and rate change. Once such a system is built, though, any rate change below that maximum will be accommodated by the design.

Listing 2 shows a *C* program that implements a rudimentary CIC filter in software on a DSP chip. Read_adc() and write_dac() are left as exercises for the reader. Code very similar to that shown could be used to implement an FPGA version, which uses a VHDL compiler.

Notes

[1]E. B. Hogenauer, "An Economical Class of Digital Filters For Decimation and Interpolation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume 29, April 1981 pp:155-162.

[2]Matthew P. Donadio, "CIC Filter Introduction", 18 July 2000, **dspguru.com/dsp/tutorials/cic-filter-introduction**.

[3]See the Wikipedia entry at **http://en.wikipedia.org/wiki/Two's_complement**.

[4]Richard Lyons, "Understanding Cascaded Integrator-Comb Filters," *EE Times*, March 31, 2005.

Array Solutions Your Source for Outstanding Radio Products

Top-ranked Measurement Equipment from Array Solutions

Announcing the NEW: PowerAIM 120
Vector Impedance Analyzer for Broadcast Engineers

- Patented, unique technology offers the broadcast engineer the full capabilities of a single port network analyzer
- Small, lightweight, software-driven instrument
- Easy to carry on airlines and in the field.
- Very simple to set up and use.
- Safe measurements in RF-dense broadcast environments.
- Time Domain Reflectometer (TDR) Functions.

PowerMaster II

- New Larger, Sharp & Fast LCD Display
- Reduced Energy consumption
- USB and RS-232 interface built-in
- New - Both 3kW and 10kW couplers on one display - switched
- Hi / Lo Power Level Monitoring
- Supports 2 like couplers simultaneously (3kW & 3kW, 3kW & V/UHF, 10kW & 10kW)
- SWR Threshold Protection (with amp PTT bypass)

Single and Dual Rack Mount available
New "Power Master Basic" Software FREE!

Vector Network Analyzer Model VNA 2180

Measures impedance magnitude, phase and transmission parameters for antennas, filters, and discrete components - using one or two ports.

- Frequency range is 5KHz to 180MHz.
- Data plots include: impedance, SWR, return loss, S11 and S21.
- Plots can be saved for before and after comparisons.
- Dual Smith charts with zoom and rotation.
- Time Domain Reflectometer (TDR) Functions.
- New – 6 port VNA multiplexer for measuring directive arrays including Phase/Magnitude vector scope software.

AIM *uhf* Analyzer

- Frequency range from 5 kHz to 1 GHz.
- Data plots include SWR, RL, R + X, series and parallel, magnitude, phase, and more.
- Dual Smith charts with rotation and 20 markers.
- Plots and calibration files can be saved and used anytime in CVS and dynamic formats.
- AIM 4170C is still in production covering 5kHz to 180 MHz.
- Time Domain Reflectometer (TDR) Functions.

Other Quality Products from Array Solutions...

ACOM Sales and Service for Amplifiers and Accessories
Phillystran, Inc. Official Worldwide Phillystran Distributor
RigExpert Analyzers and Interfaces
Prosistel Rotators Strongest Rotators on the Market
OptiBeam Antennas German Engineering means High Performance
Hofi® Surge Arrestors & Antenna Switches

ARRAY SOLUTIONS

www.arraysolutions.com

Sunnyvale, Texas USA
Phone 214-954-7140
sales@arraysolutions.com
Fax 214-954-7142

Array Solutions analyzers are used by amateur, commercial, and professional broadcast engineers. See our web site for other products and additional details on these analyzers.