Pseudo-Random Waveforms and Comb Calibration Signals

Evans Paschal

9/12/05

This note describes a technique for generating a signal which has a comb spectrum. It contains many harmonics of a given base frequency, and the amplitudes of all the harmonics are equal. A signal like this is a useful calibration signal when applied to the input of an amplifier under test. As long as the amplifier passes the signal without distortion, we can measure the gain and frequency response of the amplifier in one step. That is, since the amplitude and frequency of each component of the input comb signal is known, by measuring the amplitudes of these components in the output we can find the gain of the amplifier as a function of frequency, as shown in Figure 1.

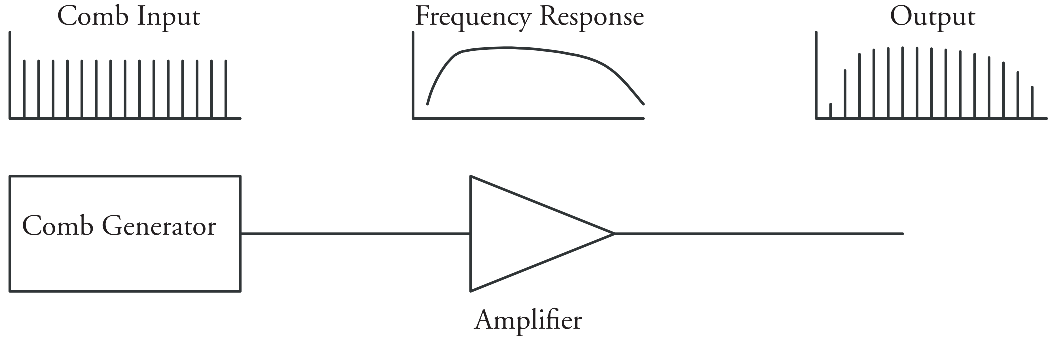


Figure 1. Using comb signal to measure amplifier frequency response.

Perhaps the simplest way to generate a comb signal is to use a short repetitive pulse. The pulse repetition frequency f_0 is the base frequency of the comb. As long as the width of the pulse is sufficiently narrow, the amplitudes of the harmonic components in the waveform at multiples of f_0 can be made as uniform as desired; that is, the envelope of the comb spectrum can be as flat as needed. This approach has a major limitation, however: all the power of the signal is contained in the short pulse; the rest of the time the waveform is zero. To obtain useful power in each of the comb components a very large pulse must be used, and such a pulse will be clipped in the amplifier. Once it's clipped, it can't be used to measure amplifier frequency response.

The approach we'll follow here is to generate a pseudo-random waveform. This is a signal whose waveform appears to be random over short intervals, but which is actually deterministic and repeats over a cycle time t_0. It has frequency components at multiples of $f_0 = 1/t_0$. And the peak to rms ratio of the waveform is low, meaning it is much less likely to be clipped in the amplifier.

A pseudo-random waveform, also known as pseudo-random noise, is the output of a psuedo-random binary sequence generator. This is often implemented as a shift register with feedback as shown in Figure 2.

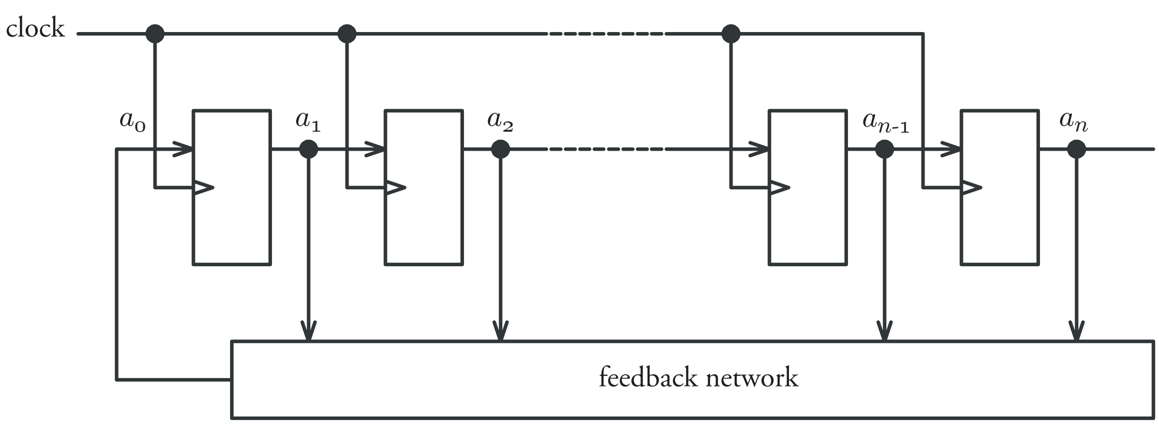


Figure 2. Shift register with feedback

The shift registers are clocked such that, at time p, $a_i(p) = a_{i-1}(p-1)$. That is, the previous state at a_{i-1} appears as state a_i after the clock pulse. State a_0 at the input to the shift register is some function of states $a_1, ..., a_n$, or, equivalently, of previous a_0's, as

$$a_0(k) = f(a_1(k), a_2(k), \ldots, a_n(k)) = f(a_0(k-1), a_0(k-2), \ldots, a_0(k-n)). \tag{1}$$

The usual feedback is "linear" feedback of the form

$$a_0(k) = c_1 a_1(k) \oplus c_2 a_2(k) \oplus \cdots \oplus c_n a_n(k), \tag{2}$$

where each c_i is 0 or 1, depending if state a_i is fed back or not, and $\oplus$ indicates modulo-2 summation (that is, the XOR function).

With the proper choice of c_i's we can generate a maximal length sequence, whose sequence length is $m = 2^n-1$, as shown in Figure 3. Note that an n-stage shift register has a total of 2^n different states, but that the all-zero state makes $a_0 = 0$, which gives the all-zero state again. Only at most 2^n-1 other states can be in a sequence. Our design must prevent the shift register from starting in the all-zero state.

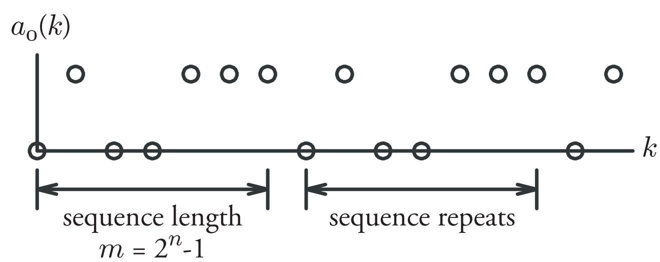


Figure 3. Maximal length shift register sequence

A maximal length shift register sequence has several interesting statistical properties.* For instance, the number of zeros and ones is nearly equal (in fact, there is always exactly 1 more ones than zeros). There are two runs (of constant value) of length p for every run of length $p+1$. And, most important for our use, the autocorrelation function

$$R(i) \equiv \frac{1}{m} \sum_{k=0}^{m-1} a_0(k) * a_0(i-k) \tag{3}$$

is two-valued, with a peak at 0 phase (and multiples of the sequence length m) and a very small value elsewhere, as shown in Figure 4. (Note that we have considered the sequence to take on the values -1 and 1 here, rather than 0 and 1.)

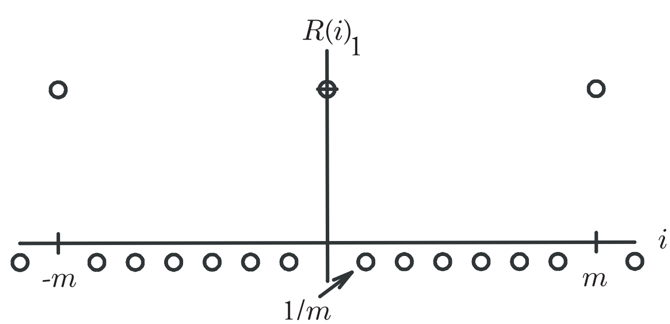


Figure 4. Autocorrelation of maximal length sequence

*For further details, the reader is referred to the definitive work: *Shift Register Sequences,* Solomon W. Golomb, Holden-Day, 1967.

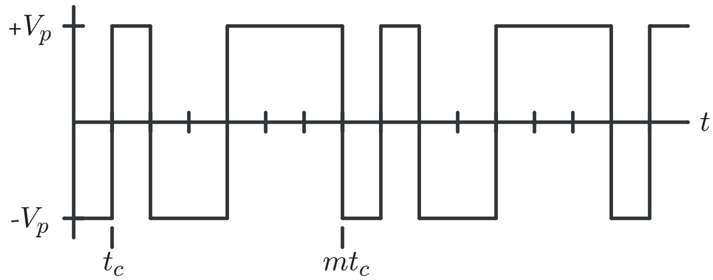


Figure 5. Shift register output viewed as a voltage waveform

Now consider the output of the shift register as a voltage waveform $v(t)$ as a function of time, as shown in Figure 5. The shift register is clocked every t_c seconds. The waveform $v(t)$ can have two values, either +V_p or -V_p, and can change only at clock transitions. The waveform repeats every mt_c seconds. The autocorrelation $R(\tau)$ of this waveform is given by Equation 4, where

$$\begin{aligned} R(\tau) &= \lim_{T\to\infty} \frac{1}{T}\int_{-T/2}^{T/2} v(t)v(\tau - t)dt \\ &= V_p^2\left[(1+\frac{1}{m})\Lambda(\frac{\tau}{t_c}) - \frac{1}{m}\Pi(\frac{\tau}{mt_c})\right] * \sum_{k=-\infty}^{\infty}\delta(\tau - kmt_c) \end{aligned} \tag{4}$$

the $*$ operator indicates convolution. $R(\tau)$ is shown in Figure 6.

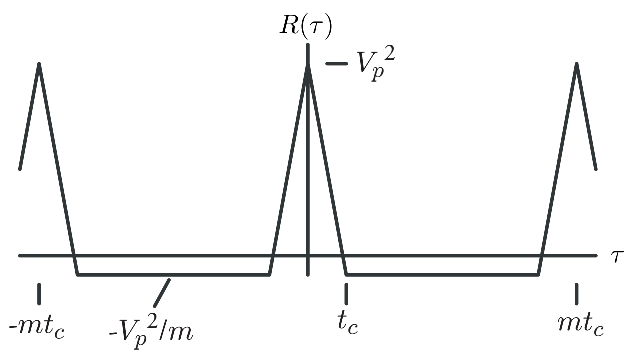


Figure 6. Autocorrelation of shift register output

Note that this is the same as the autocorrelation of a short repetitive pulse, except for the small negative value -$V_p{}^2/m$. The power spectrum of the shift register output $G(f)$ is the Fourier transform of the autocorrelation function, as

$$\begin{aligned} G(f) &= F\{R(\tau)\} \\ &= V_p^2\left[(\frac{m+1}{m^2})\,\mathrm{sinc}^2(ft_c) - \frac{1}{m}\,\mathrm{sinc}(fmt_c)\right]\cdot\sum_{k=-\infty}^{\infty}\delta(f - \frac{k}{mt_c}), \end{aligned} \tag{5}$$

where the sinc function is defined by $\mathrm{sinc}(x) \equiv \sin(\pi x)/\pi x$. The power spectrum $G(f)$ is zero everywhere except at multiples of $1/mt_c$, that is, at harmonics of the sequence repetition rate. If we filter the signal around one of these harmonics (including power at negative as well as positive frequency), we find the rms voltage of the spectral component at a given harmonic frequency is

$$V(0) = V_p/m, \tag{6}$$

$$\text{and} \qquad V(kf_0) = V_p\frac{\sqrt{2}(m+1)^{1/2}}{m}\,\mathrm{sinc}(k/m), \qquad k = 1, 2, \ldots, \tag{7}$$

where $f_0 = 1/mt_c$ is the sequence repetition rate. The term $V(0)$ is a small dc voltage, which we can ignore (use capacitive coupling to the rest of the circuit). The other components are all multiples of the comb frequency f_0. (Note: while we know the rms amplitudes of the comb components, we don't know their relative phases, which have no simple relationship.)

Design Procedure

1. The bandwidth of the amplifier to calibrate is known. Choose the frequency f_0 and factor k such that the frequency comb from f_0 to kf_0spans the frequency interval of interest. (If a finer comb spacing is desired, choose the comb spacing f_0 and find integers j and k such that jf_0 to kf_0 spans the interval of interest.) For example, assume we want to measure our amplifier response from 100 Hz to 30 kHz. If we chose f_0 = 100 Hz, then the comb component at 30 kHz will be the 300th component, or k = 300.

2. Decide how flat the comb needs to be. The rms voltage of higher-frequency components is proportional to sinc(k/m). For example, assume we want the 300th component to be down no more than 0.5 dB (0.944) from the first component. We need sinc(k/m) > 0.944. For k = 300 we find m >= 1613.

3. The sequence length m must be of the form 2^n-1. Find the integer n that gives an m greater than the minimum found in the previous step. (Making n even bigger will, of course, make the comb even flatter, but at the expense of a greater peak voltage for a given component rms voltage.) For example, for m >=1613 we find n = 11, giving m = 2047, and sinc(300/m) = 0.965 = -0.31 dB.

4. Calculate the shift register clock frequency $f_c = mf_0$. For example, 2047x100Hz = 204.700 kHz. This may not be a convenient frequency. Depending on how much effort we want to go to, we may need to compromise our original specifications and adjust f_0 to give a better clock frequency.

5. Finally, calculate the rms voltage of each comb component using Equation 7. For example, assume we use the capacitive-coupled output of a shift register built with 5-volt logic. The output swings from 0 to +5 V, so the ac signal after the capacitor will swing from -2.5 V to +2.5 V, or V_p = 2.5 V. For m = 2047, we find V(100 Hz) = 78.1 mV (and V(30 kHz) = 75.4 mV). We may need to amplify or attenuate the 2.5 V "square-wave" output of the shift register to get the amplitude we need.

Note the advantage of the pseudo-random comb generator in the example above. If we had generated the comb signal using a short rectangular pulse, we would have used a pulse 4.89 µs wide ($1/mf_0$), repeated every 10 ms. To get the same 78 mV component amplitude, however, the short pulse would need an amplitude of $m^{1/2}V_p$ or 113 V! Our amplifier would need an additional 33 dB of headroom to pass this pulse without distortion. The problem with a short pulse, of course, is that all the energy of the signal is contained in the pulse, whereas the pseudo-random waveform spreads the energy out and is always running at constant power. In terms of the peak voltage required for a given component amplitude, the pseudo-random waveform is the optimum solution.

Sample Designs

Figures 7 and 8 show two comb filter designs that are currently used at field stations for calibrating ELF and VLF receivers. The circuit in Figure 7 is used in the Noise Survey project receivers (Tony Fraser-Smith's experiment). The comb generator is implemented in 4000-series CMOS logic running from a 15-volt supply. Three 4015 4-stage shift registers are wired in series to give a 12-bit shift register; only the first 11 bits are used (m = 2047). Note that the feedback logic here uses XNOR (inverting XOR) logic. This works because, while we're loading the inverse feedback into the shift register, and thus looking at the Q-bar outputs instead of the Q outputs, the XOR of two Q-bar outputs gives the same result as the XOR of two Q outputs. We're really generating the inverse of the sequence. In this design, the all-ones state is the state to be avoided. This is accomplished by using the 4015 reset inputs to be sure the shift register starts off in the all-zeros state. The clock frequency required is synthesized using a divide-by-m counter and a phase-locked-loop referenced to an external standard. Since this circuit is designed to calibrate ELF as well as VLF receivers, with the ELF comb going as low as 10 Hz, dc coupling is used for the comb signal. The Disable Offset input to the low-pass filter compensates for the dc offset while the shift register is held off in its all-zeros state.

Figure 8 shows the comb calibrator in the new HAIL2 preamps currently being deployed by Stanford. In this circuit the pseudo-random sequence is generated by a PIC12F629 microprocessor. The microprocessor generates the maximum length sequence of a 10-stage shift register (m = 1023). With a crystal oscillator input of f_x = 8.192 MHz, the internal microprocessor clock runs at $f_x/4$ = 2.048 MHz. The cycle time for the shift register is 8 microprocessor instructions, so the virtual shift register clock is $f_c = f_x/32$ = 256 kHz. With a sequence length of m = 1023, the comb components are spaced every 250.244 Hz, and the spectral envelope is down by 0.22 dB at 32 kHz. The desired comb spacing was f_0 = 250.000 Hz. This would have required a crystal oscillator frequency of 8.184 MHz. This is not available in an off-the-shelf oscillator, so the closest available frequency was used. Rather than using XOR (or XNOR) feedback of the register contents to determine the next input bit to the shift register, the program in the microprocessor tests the shift register output and, if zero, XORs the feedback value into the register (broadside, as it were). We can think of this as running the circuit in Figure 2 in time-reversed order.

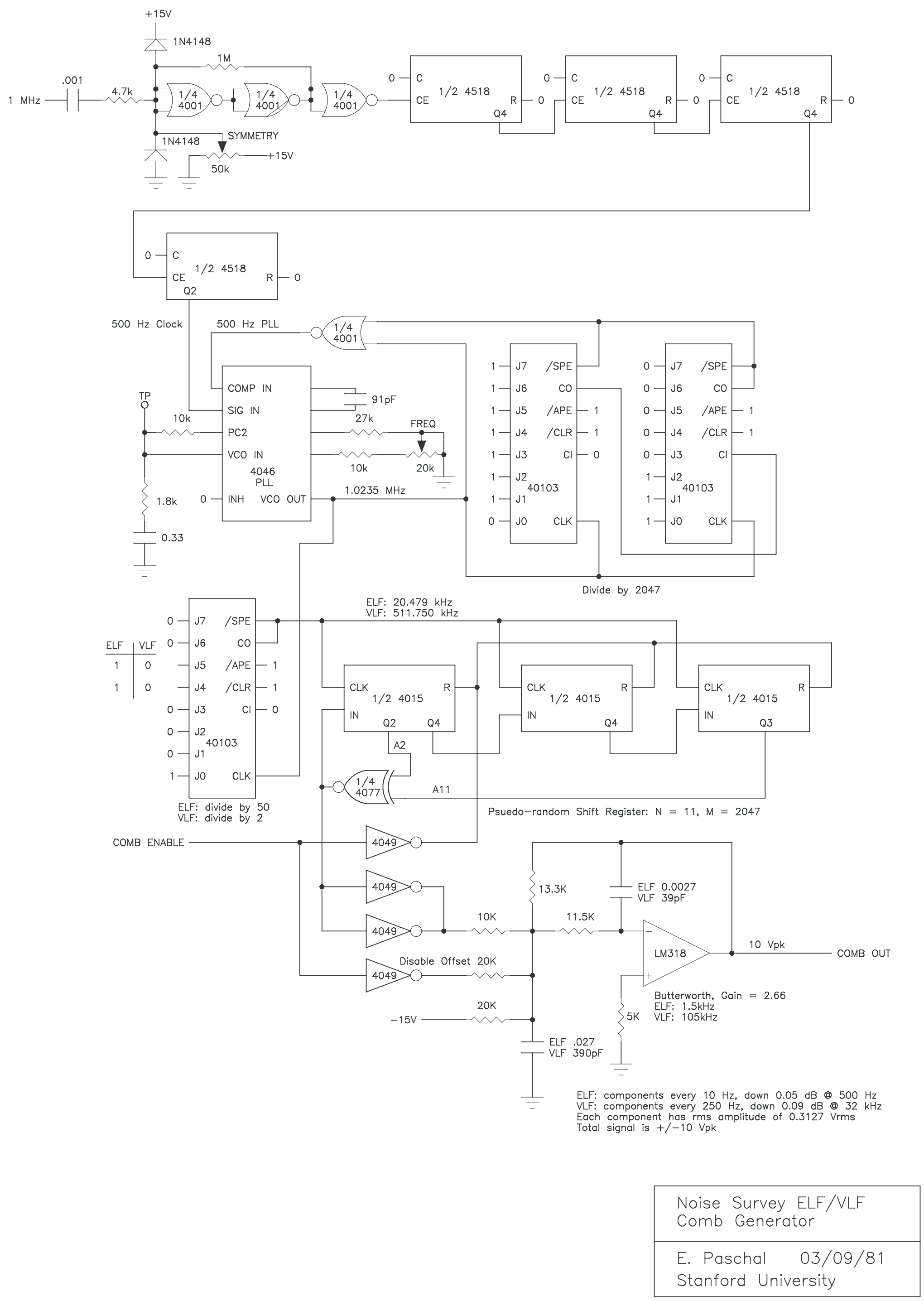
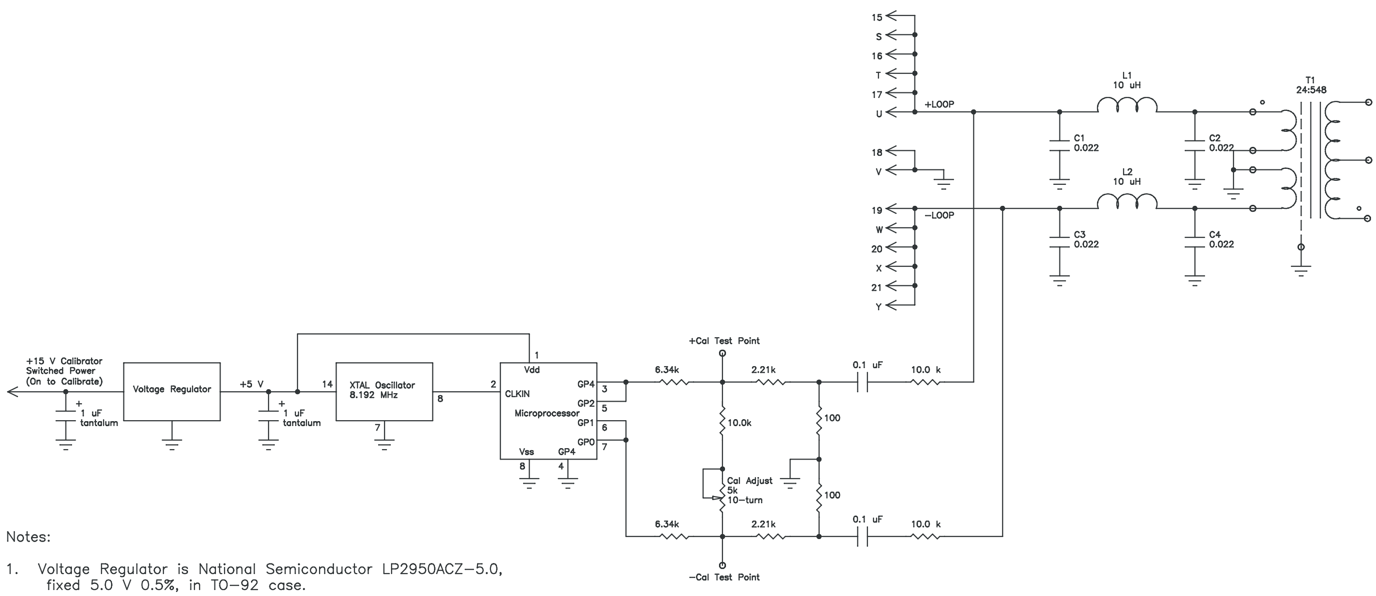


Figure 7. Comb generator using 15-volt 4000-series CMOS logic.



Notes:

1. Voltage Regulator is National Semiconductor LP2950ACZ-5.0, fixed 5.0 V 0.5%, in TO-92 case.

2. Crystal Oscillator is 8.192 MHz 100ppm in 14-pin DIP package, EPSON SG-51 8.1920MC or similar.

3. Microprocessor is Microchip PIC12F629 in 8-pin DIP.

4. 1 uF tantalum capacitors at voltage regulator can be 20 or 35 V units, Kemet T350A105M035AS, for instance.

5. All resistors RN55D 1% metal film.

6. 0.1 uF capacitors X7R (not Z5U) ceramic 5%, any voltage.

Comb Calibrator generates calibration components of equal amplitude every 250 Hz throughout VLF spectrum.
Adjust "Cal Adjust" pot for 1 V peak-to-peak at +Cal and -Cal testpoints (or 2 V p-p between them).
This will give each component an amplitude of 1.00 mV rms at the 10 k cal resistors (equivalent to 0.69 pT or 208 uV/m when using 4.9m square 1-ohm 1-mH loop.

HAIL2 VLF Line Receiver Comb Calibration Generator Rev 1
E. Paschal 2/28/04 Whistler Radio Services

Figure 8. Comb generator implemented with a microprocessor.

Appendix 1: Program to List Maximal-Length Sequence Feedback Taps

The following C language program lists all XOR feedback taps that give maximal-length shift register sequences for $n = 3$ to $nmax$ (nominally 16). The (edited) output of this program is listed in Appendix 2.

```
/*PRSeqList.cpp
*
*       Pseudo-random shift-register sequence feedback list program
*       E. Paschal
*       2/19/04
*
*       This program is used to list feedback taps for shift registers which
*       give maximal-length sequences using XOR feedback.  The program lists
*       all feedback taps for shift register lengths from 3 to nmax.
*/

#include <stdio.h>

/*  Global variables  */
int             nmax;           /* maximum shift-register length, 3 to 31        */
int             n;              /* current shift-register length                 */
unsigned long   regist;         /* shift register, up to 31 bits                 */
unsigned long   feedbk;         /* feedback taps                                 */
unsigned long   rmask;          /* register mask, 0s except n 1s on right end    */
unsigned long   m;              /* (2**n)-1 = maximal length (= rmask)           */
unsigned long   mp1;            /* 2**n                                          */
unsigned long   count;          /* shift counter                                 */
unsigned long   soluts;         /* number of maximal-length solutions            */
FILE            *listfile;

int main()
{
    listfile = fopen("Seqlist.txt","a");
    nmax = 16;                                  /* don't make it too big unless you have
                                                   a fast machine.  must be 31 or less */
    for (n = 3; n <= nmax; n++) {               /* do next sequence length      */
        int i;                                  /* loop counters                */
        rmask = 0;
        for (i = 0; i < n; i++) {
            rmask = (rmask << 1) + 1;           /* shift in n 1s to mask        */
        }
        m = rmask;                              /* 2**n -1                      */
        mp1 = m + 1;                            /* 2**n                         */
        soluts = 0;                             /* reset solution count for this n */
        printf("Registers n = %d, Sequence length m = %d\n",n,m);
        fprintf(listfile,"Registers n = %d, Sequence length m = %d\n",n,m);
                                                /* test all possible feedbk's */
                                                /* first feedback mask is '1000..000' */
        for (feedbk = m/2; feedbk < mp1-1; feedbk++) {
            regist = 0;                         /* start with register clear    */
            for (count = 1; count <= mp1; count++) {/* shift till back to 0      */
                unsigned long regtemp;
                int bitct;
                int j;
                regtemp = regist & feedbk;      /* mask feedback bits           */
                bitct = 0;
                for (j = 0; j < n; j++) {       /* count masked bits set        */
                    if (regtemp & 1) bitct++;
                    regtemp = regtemp >> 1;
                }
                regist = regist << 1;           /* shift register left          */
                                                /* shift in a 1 if even feedback bits */
                if ((bitct & 1) == 0) regist++;
                regist = regist & rmask;        /* mask to n bits               */
                if (regist == 0) break;         /* done when we're back to 0 */
            }
            if (count == m) {
                soluts++;
                printf ("Feedback %6x = ",feedbk);
                fprintf (listfile,"Feedback %6x = ",feedbk);
                regist = feedbk;
                for (i = 0; i < n; i++) {
                    regist = regist << 1;
                    printf ("%1x",(mp1 & regist) >> n);
                    fprintf (listfile,"%1x",(mp1 & regist) >> n);
                }
                printf ("\n");
                fprintf (listfile,"\n");
            }
        }
        printf("Number of maximal-length solutions = %d\n\n",soluts);
        fprintf(listfile,"Number of maximal-length solutions = %d\n\n",soluts);
    }
    return(0);
}
```

Appendix 2: Partial Listing of Maximal-Length Sequence Feedback Taps

This listing shows feedback taps that will give maximal-length pseudo-random sequences for shift registers from 3 to 16 bits long. For instance, the feedback used in the comb generator in Figure 7 (bits a_2 and a_{11}) is shown below in bold as the first entry in the table for n = 11. Note that for many shift register lengths it is possible to generate a maximal-length sequence using only two feedback taps. However, for lengths n = 8, 12, 13, 14, and 16, four taps must be used.

```
Registers n = 3, Length m = 7
Feedback      5 = 101
Feedback      6 = 110
Number of solutions = 2

Registers n = 4, Length m = 15
Feedback      9 = 1001
Feedback      c = 1100
Number of solutions = 2

Registers n = 5, Length m = 31
Feedback     12 = 10010
Feedback     14 = 10100
Feedback     17 = 10111
Feedback     1b = 11011
Feedback     1d = 11101
Feedback     1e = 11110
Number of solutions = 6

Registers n = 6, Length m = 63
Feedback     21 = 100001
Feedback     2d = 101101
Feedback     30 = 110000
Feedback     33 = 110011
Feedback     36 = 110110
Feedback     39 = 111001
Number of solutions = 6

Registers n = 7, Length m = 127
Feedback     41 = 1000001
Feedback     44 = 1000100
Feedback     47 = 1000111
Feedback     48 = 1001000
Feedback     4e = 1001110
Feedback     53 = 1010011
Feedback     55 = 1010101
Feedback     5c = 1011100
Feedback     5f = 1011111
Feedback     60 = 1100000
Feedback     65 = 1100101
Feedback     69 = 1101001
Feedback     6a = 1101010
Feedback     72 = 1110010
Feedback     77 = 1110111
Feedback     78 = 1111000
Feedback     7b = 1111011
Feedback     7e = 1111110
Number of solutions = 18

Registers n = 8, Length m = 255
Feedback     8e = 10001110
Feedback     95 = 10010101
Feedback     96 = 10010110
Feedback     a6 = 10100110
Feedback     af = 10101111
Feedback     b1 = 10110001
Feedback     b2 = 10110010
Feedback     b4 = 10110100
Feedback     b8 = 10111000
Feedback     c3 = 11000011
Feedback     c6 = 11000110
Feedback     d4 = 11010100
Feedback     e1 = 11100001
Feedback     e7 = 11100111
Feedback     f3 = 11110011
Feedback     fa = 11111010
Number of solutions = 16

Registers n = 9, Length m = 511
Feedback    108 = 100001000
Feedback    10d = 100001101
Feedback    110 = 100010000
Feedback    116 = 100010110
Feedback    119 = 100011001
Feedback    12c = 100101100
Feedback    12f = 100101111
Feedback    134 = 100110100
Feedback    137 = 100110111
Feedback    13b = 100111011
        ...
Feedback    1da = 111011010
Feedback    1dc = 111011100
Feedback    1e3 = 111100011
Feedback    1e5 = 111100101
Feedback    1e6 = 111100110
Feedback    1ea = 111101010
Feedback    1ec = 111101100
Feedback    1f1 = 111110001
Feedback    1f4 = 111110100
Feedback    1fd = 111111101
Number of solutions = 48

Registers n = 10, Length m = 1023
Feedback    204 = 1000000100
Feedback    20d = 1000001101
Feedback    213 = 1000010011
Feedback    216 = 1000010110
Feedback    232 = 1000110010
Feedback    237 = 1000110111
Feedback    240 = 1001000000
Feedback    245 = 1001000101
Feedback    262 = 1001100010
Feedback    26b = 1001101011
        ...
Feedback    3aa = 1110101010
Feedback    3ac = 1110101100
Feedback    3b1 = 1110110001
Feedback    3be = 1110111110
Feedback    3c6 = 1111000110
Feedback    3c9 = 1111001001
Feedback    3d8 = 1111011000
Feedback    3ed = 1111101101
Feedback    3f9 = 1111111001
Feedback    3fc = 1111111100
Number of solutions = 60

Registers n = 11, Length m = 2047
Feedback    402 = 10000000010
Feedback    40b = 10000001011
Feedback    415 = 10000010101
Feedback    416 = 10000010110
Feedback    423 = 10000100011
Feedback    431 = 10000110001
Feedback    432 = 10000110010
Feedback    438 = 10000111000
Feedback    43d = 10000111101
Feedback    446 = 10001000110
        ...
Feedback    7c8 = 11111001000
Feedback    7cb = 11111001011
Feedback    7cd = 11111001101
Feedback    7d3 = 11111010011
Feedback    7d6 = 11111010110
Feedback    7da = 11111011010
Feedback    7e6 = 11111100110
Feedback    7e9 = 11111101001
Feedback    7f2 = 11111110010
Feedback    7f4 = 11111110100
Number of solutions = 176
```

```
Registers n = 12, Length m = 4095
Feedback    829 = 100000101001
Feedback    834 = 100000110100
Feedback    83d = 100000111101
Feedback    83e = 100000111110
Feedback    84c = 100001001100
Feedback    868 = 100001101000
Feedback    875 = 100001110101
Feedback    883 = 100010000011
Feedback    88f = 100010001111
Feedback    891 = 100010010001
       ...
Feedback    f47 = 111101000111
Feedback    f71 = 111101110001
Feedback    f88 = 111110001000
Feedback    f8d = 111110001101
Feedback    f93 = 111110010011
Feedback    fb8 = 111110111000
Feedback    fcc = 111111001100
Feedback    fdd = 111111011101
Feedback    fde = 111111011110
Feedback    fe4 = 111111100100
Number of solutions = 144

Registers n = 13, Length m = 8191
Feedback   100d = 1000000001101
Feedback   1013 = 1000000010011
Feedback   101a = 1000000011010
Feedback   1029 = 1000000101001
Feedback   1032 = 1000000110010
Feedback   1037 = 1000000110111
Feedback   1045 = 1000001000101
Feedback   1046 = 1000001000110
Feedback   104f = 1000001001111
Feedback   1052 = 1000001010010
       ...
Feedback   1fab = 1111110101011
Feedback   1fb0 = 1111110110000
Feedback   1fc1 = 1111111000001
Feedback   1fc4 = 1111111000100
Feedback   1fc8 = 1111111001000
Feedback   1fd5 = 1111111010101
Feedback   1fda = 1111111011010
Feedback   1ff1 = 1111111110001
Feedback   1ffb = 1111111111011
Feedback   1ffe = 1111111111110
Number of solutions = 630

Registers n = 14, Length m = 16383
Feedback   2015 = 10000000010101
Feedback   201c = 10000000011100
Feedback   2029 = 10000000101001
Feedback   202f = 10000000101111
Feedback   203d = 10000000111101
Feedback   2054 = 10000001010100
Feedback   2057 = 10000001010111
Feedback   205d = 10000001011101
Feedback   205e = 10000001011110
Feedback   2067 = 10000001100111
       ...
Feedback   3f9f = 11111110011111
Feedback   3fa6 = 11111110100110
Feedback   3faa = 11111110101010
Feedback   3fb8 = 11111110111000
Feedback   3fc5 = 11111111000101
Feedback   3fc6 = 11111111000110
Feedback   3fcf = 11111111001111
Feedback   3fe2 = 11111111100010
Feedback   3fe8 = 11111111101000
Feedback   3ff3 = 11111111110011
Number of solutions = 756

Registers n = 15, Length m = 32767
Feedback   4001 = 100000000000001
Feedback   4008 = 100000000001000
Feedback   400b = 100000000001011
Feedback   4016 = 100000000010110
Feedback   401a = 100000000011010
Feedback   402f = 100000000101111
Feedback   403b = 100000000111011
Feedback   4040 = 100000001000000
Feedback   4043 = 100000001000011
Feedback   4049 = 100000001001001
       ...
Feedback   7fb0 = 111111110110000
Feedback   7fb9 = 111111110111001
Feedback   7fbf = 111111110111111
Feedback   7fc8 = 111111111001000
Feedback   7fd9 = 111111111011001
Feedback   7fe3 = 111111111100011
Feedback   7fec = 111111111101100
Feedback   7ff4 = 111111111110100
Feedback   7ff7 = 111111111110111
Feedback   7ffe = 111111111111110
Number of solutions = 1800

Registers n = 16, Length m = 65535
Feedback   8016 = 1000000000010110
Feedback   801c = 1000000000011100
Feedback   801f = 1000000000011111
Feedback   8029 = 1000000000101001
Feedback   805e = 1000000001011110
Feedback   806b = 1000000001101011
Feedback   8097 = 1000000010010111
Feedback   809e = 1000000010011110
Feedback   80a7 = 1000000010100111
Feedback   80ae = 1000000010101110
       ...
Feedback   ff41 = 1111111101000001
Feedback   ff74 = 1111111101110100
Feedback   ff82 = 1111111110000010
Feedback   ff99 = 1111111110011001
Feedback   ff9a = 1111111110011010
Feedback   ff9c = 1111111110011100
Feedback   ffb8 = 1111111110111000
Feedback   ffd2 = 1111111111010010
Feedback   fff5 = 1111111111110101
Feedback   fff6 = 1111111111110110
Number of solutions = 2048
```