MASSACHUSETTS INSTITUTE OF TECHNOLOGY
HAYSTACK OBSERVATORY
WESTFORD, MASSACHUSETTS 01886

August 9, 2001

Telephone: 978-692-4764
Fax: 781-981-0590

To: SRT Group

From: Alan E.E. Rogers

Subject: Calculation of the Gray Chip filter response

The GrayChip GC1011A digital receiver chip performs the following calculations:

1. Multiplies 12-bit inputs by 12-bit digital sine and cosine functions
2. reduces the products to 13-bits 2s complements
3. passes the sine and cosine outputs into a Hogenauer filter (Hogenauer, IEEE trans on acoustics, ASSP-29, No. 2, April 1981 pp. 155-162) consisting of a C1C filter of 4 integrators a decimator by factor of R followed by 4 comb filters.
4. A two's complement rescaler
5. a decimate by 4 low pass FIR filter.
6. 16 bit outputs for real and imaginary components.

The C1C filter is equivalent to 4 FIR filters in cascade each filter consisting of R coefficients of 1. The impulse response of this filter can be calculated by 3 successive convolutions of the R wide uniform filters. The impulse response of the overall GrayChip can then be obtained by convolving every 4th point of the C1C impulse response with the impulse response of the final 64 tap FIR filter. The overall "bandpass" for a white Gaussian input can be computed by averaging the spectrum of the impulse response over all possible time placements of an input impulse that will produce a non zero output in the time window of the FFT.

$h_i(t) = 1$ for τ=0,....R-1 and zero elsewhere

$$h_{C1C}(\tau) = h_0 \otimes h_1 \otimes h_2 \times h_3$$

$h_{FIR}(\tau) = 2, -1, -2, -11 \ldots$(see program listing)

$$h_{C1C4}(\tau) = h_{C1C}(4\tau)$$
$$hGray(\tau) = h_{C1C4} \otimes h_{FIR}$$
$$x_\tau(t) = \mu_0(\tau) \otimes h_{Gray}(\tau)$$

where $x(t)$ is the input to the FFT

bandpass $(w) = \left\langle \left| \chi_\tau(w) \right|^2 \right\rangle$

```
#include <stdio.h>
#include <math.h>

#define PI 3.1415926536
void main(void) /* computes Graychip bandpass for 64 point FFT */
{
double px[128];
double theta,sumr,sumi;
double pwr[64],gray[64],hcic[2000],hcicp[2000],htemp[2000];
double cf[32] = {
2,-1,-2,-11,-23,-30,-36,-25,-1,25,56,59,35,-7,-71,-106,-100,
-51,56,148,198,173,34,-143,-312,-387,-264,39,509,1050,1497,1773
        };
int i,j,k,kk,n,m,mm,ndec,ncent,ncent2;
ncent = 128;    /* array index of center lag */
ncent2 = ncent*2;
ndec = 40;      /* won't get to 80 with 128 */
for(i=0;i<ncent2;i++) hcic[i]=hcicp[i]=0.0;
for(i=0;i<ndec;i++) hcic[ncent+i-ndec/2]=hcicp[ncent+i-ndec/2]=1.0;
for(n=0;n<3;n++) {
for(i=0;i<ncent2;i++) htemp[i]=0.0;
for(i=-ncent;i<ncent;i++){ for(j=-ncent;j<ncent;j++)
 if(ncent+j+i>0) htemp[ncent+i] += hcic[ncent+j]*hcicp[ncent+i+j]; }
for(i=0;i<ncent2;i++) hcic[i]=htemp[i];
}
/*
for(i=0;i<ncent2;i++) printf("i=%d %f\n",i,hcic[i]);
*/
for(i=0;i<=32;i++) pwr[i]=0.0;
for(kk=0;kk<ndec;kk++){
for(i=0;i<ncent2;i++) htemp[i]=0.0;
for(i=-ncent;i<ncent;i++){
 for(j=-ncent;j<ncent;j++){
 k=32+i+j;
 if(k>31) k=63-k;
 mm = ncent+ndec*j+kk;
 if(k>=0 && k<32 && mm>=0 && mm<ncent2) htemp[ncent+i] += hcic[mm]*cf[k];
 }
 }
/*
if(kk==0)for(i=0;i<ncent2;i++) printf("  i=%d %f\n",i,htemp[i]);
*/
for(n=-ncent2;n<=ncent2;n++) {
for(i=0;i<64;i++) {
m=4*i+n;
if(m >=0 && m<ncent2) px[i]=htemp[m];
else px[i]=0.0;
}
```

```
for(i=0;i<=32;i++){
sumr=sumi=0.0;
for(j=0;j<64;j++){
theta = i*j*PI/(32.0);
sumr += px[j]*cos(theta);
sumi += px[j]*sin(theta);
}
pwr[i] += sumr*sumr+sumi*sumi;
}
}
}
for(i=0;i<=32;i++){
gray[i]=pwr[i]/pwr[0];
printf("i %2d pwr %f\n",i,gray[i]);
}
}
```

results:

1.000000,1.006274,1.022177,1.040125,1.051102,1.048860,1.033074,1.009606,
0.987706,0.975767,0.977749,0.991560,1.009823,1.022974,1.023796,1.011319,
0.991736,0.975578,0.972605,0.986673,1.012158,1.032996,1.025913,0.968784,
0.851774,0.684969,0.496453,0.320612,0.183547,0.094424,0.046729,0.026470,
0.021300