



## **Hideo Okawara's Mixed Signal Lecture Series**

### **DSP-Based Testing – Fundamentals 40 F-matrix Equalization**

*Verigy Japan  
September 2011*

#### **Preface to the Series**

ADC and DAC are the most typical mixed signal devices. In mixed signal testing, analog stimulus signal is generated by an arbitrary waveform generator (AWG) which employs a D/A converter inside, and analog signal is measured by a digitizer or a sampler which employs an A/D converter inside. The stimulus signal is created with mathematical method, and the measured signal is processed with mathematical method, extracting various parameters. It is based on digital signal processing (DSP) so that our test methodologies are often called DSP-based testing.

Test/application engineers in the mixed signal field should have thorough knowledge about DSP-based testing. FFT (Fast Fourier Transform) is the most powerful tool here. This corner will deliver a series of fundamental knowledge of DSP-based testing, especially FFT and its related topics. It will help test/application engineers comprehend what the DSP-based testing is and assorted techniques.

#### **Editor's Note**

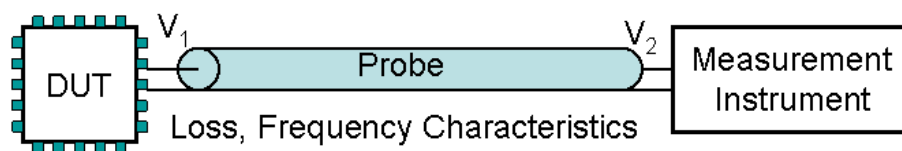
For other articles in this series, please visit the Verigy web site at [www.verigy.com/go/gosemi](http://www.verigy.com/go/gosemi).

## Preface

Signal integrity of high-speed digital signals could be degraded by the frequency characteristics of the signal path on the DUT board. So the DUT board must be carefully designed to have enough analog bandwidth along with using excellent board materials. If we knew the characteristics of the signal path, we could compensate the deterioration with using DSP technique. It is known as equalization. The theme of this month issue is equalization to a waveform data by DSP.

## F-matrix

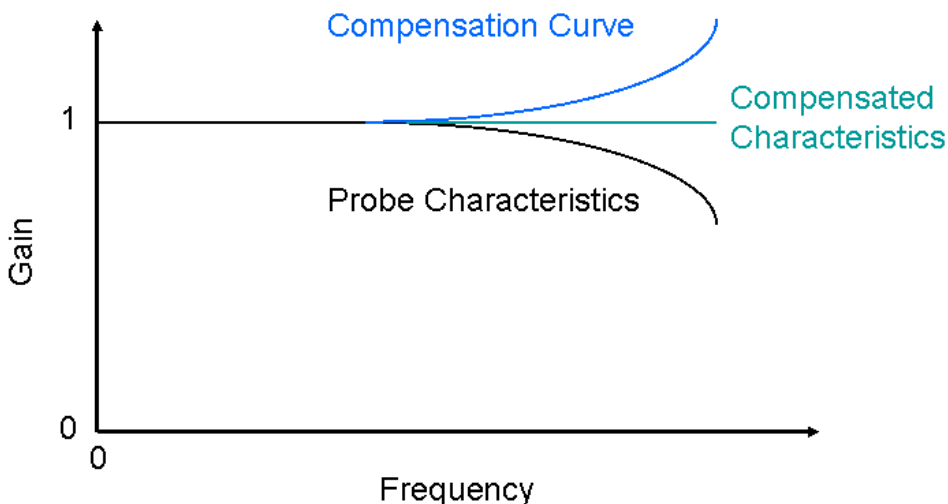
Figure 1 illustrates that a measurement instrument observes a DUT signal with using a probe. The DUT output  $V_1$  is measured as  $V_2$  at the instrument so that the measured signal should be slightly deteriorated by the probe.



**Figure 1: Measurement via Probe**

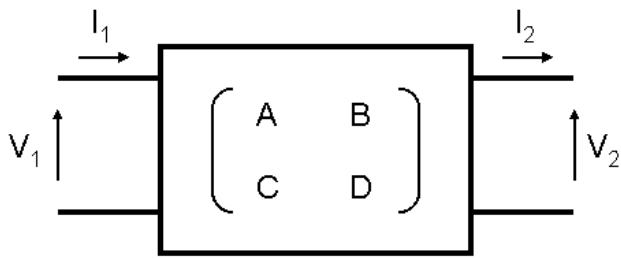
The higher the test signal frequency is, the more the signal would be degraded. We want to measure  $V_1$  right at the DUT terminal but we have to see  $V_2$  instead at the probe output. If we know the characteristics of the probe, we could mathematically compensate the degradation by the probe.

Figure 2 illustrates that the probe gain would gradually reduce along with the frequency. By multiplying the reciprocal characteristics of the probe, the total characteristics would be flattened. This compensation technique is often called equalization in the data communication arena. A physical circuit "equalizer" may be employed to perform equalization in the practical transmission systems or measurement systems.



**Figure 2: Frequency Response Compensation = Equalization**

Let's review about F-matrix. Figure 3 illustrates a 2-port network. The input voltage  $V_1$  and current  $I_1$  can be described with using the ABCD parameters and the output voltage  $V_2$  and current  $I_2$  as Equation (1).



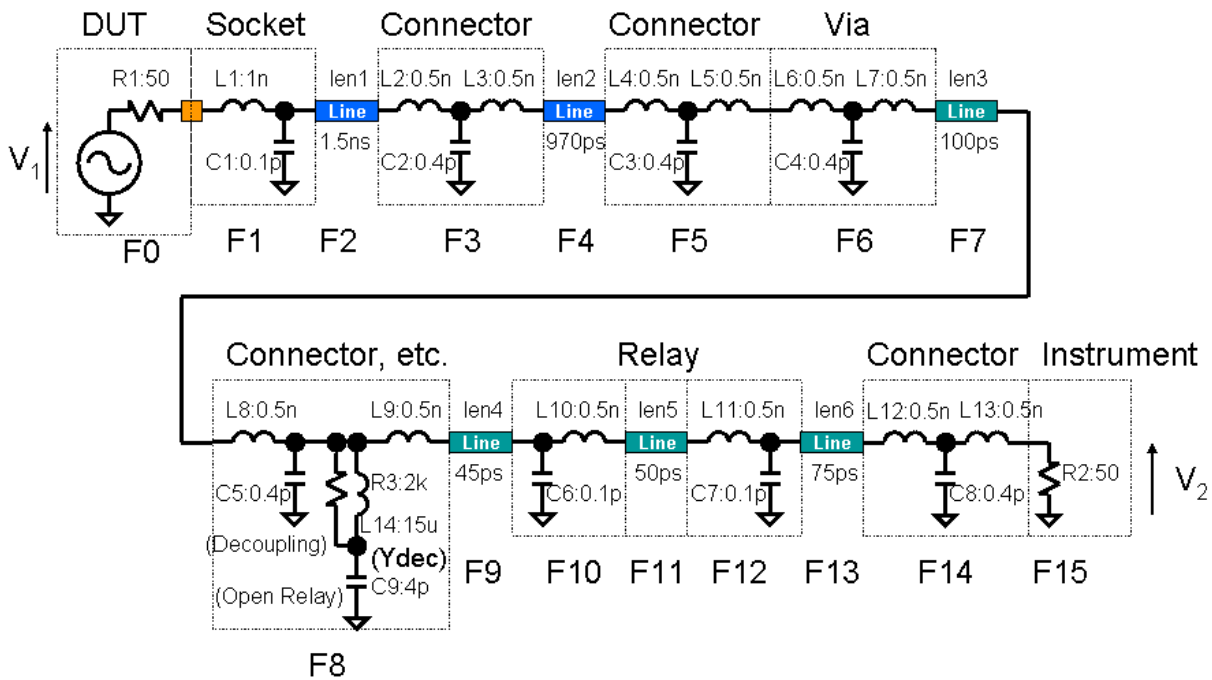
**Figure 3: F-matrix**

$$\begin{aligned}
 V_1 &= AV_2 + BI_2 \\
 I_1 &= CV_2 + DI_2
 \end{aligned}
 \tag{1}$$

If the output port is open,  $I_2=0$  so that Equation (1) becomes  $V_1=AV_2$ . So if we know the characteristics of the network parameter  $A$ , we can estimate the input signal  $V_1$  by manipulating the output signal  $V_2$ . The equation  $V_1=AV_2$  represents the equalization.

### Path from DUT to Instrument

Figure 4 illustrates an example equivalent circuit from a DUT output to a measurement pin.



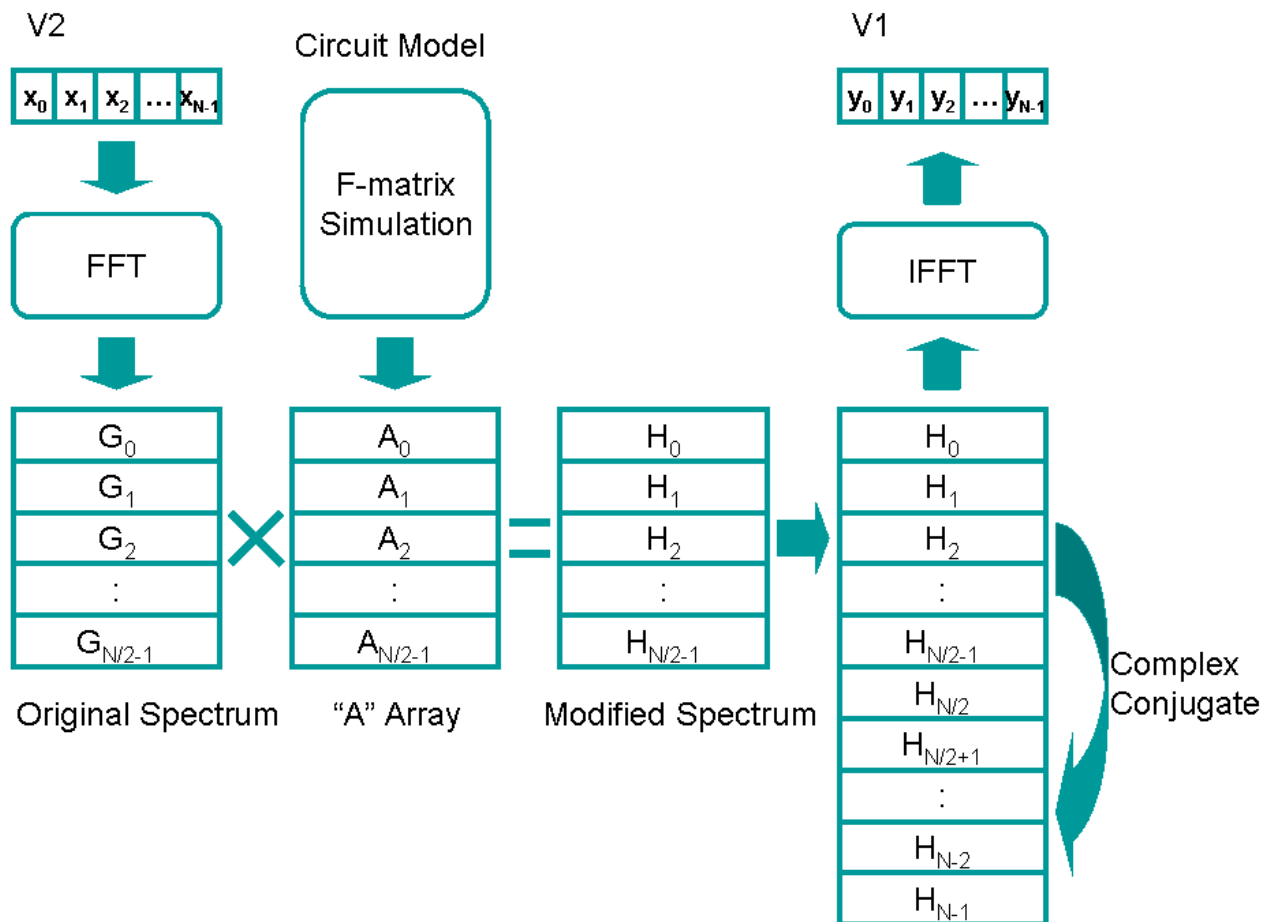
**Figure 4: Equivalent Circuit Model of DUT to Instrument**

The signal  $V_1$  at the DUT output is conveyed through the network and the transmission lines to the measurement pin and it is captured as the signal  $V_2$ , which is deteriorated by the loss and the frequency characteristics of the path. "Line" in Figure 4 represents 50Ω coaxial cables (colored blue) and printed circuit patterns (colored green). They are simply approximated as coaxial cables in the simulation processing. The total delay time adds up to 2.74ns. Cascading the sixteen

F-matrices can be combined as a single F-matrix by F-matrix multiplication processing.<sup>1</sup> As discussed in the previous section, the parameter  $A$  of the combined F-matrix is used for the equalization processing.

### FFT & IFFT Method

The purpose of the work is to compensate the characteristics of the signal path and estimate the true DUT output waveform  $V_1$  based on the measured waveform  $V_2$ . The processing strategy is illustrated in Figure 5. The measured signal  $V_2$  is the original waveform data set  $\{x_k\}$ . The FFT converts it into the frequency spectrum set  $\{G_k\}$ . The frequency characteristics of the signal path model in Figure 4 can be compiled as a single F-matrix. Then the parameter  $A$  can be figured out at each FFT frequency bin as the set  $\{A_k\}$ . As discussed previously,  $V_1=AV_2$  so that the equalized frequency components can be calculated as  $G_k \cdot A_k = H_k$ . With making  $\{H_k\}$  as complex conjugate, the IFFT converts  $\{H_k\}$  into the real number waveform set  $\{y_k\}$ , which is the equalized waveform  $V_1$ .



**Figure 5: FFT & IFFT Method**

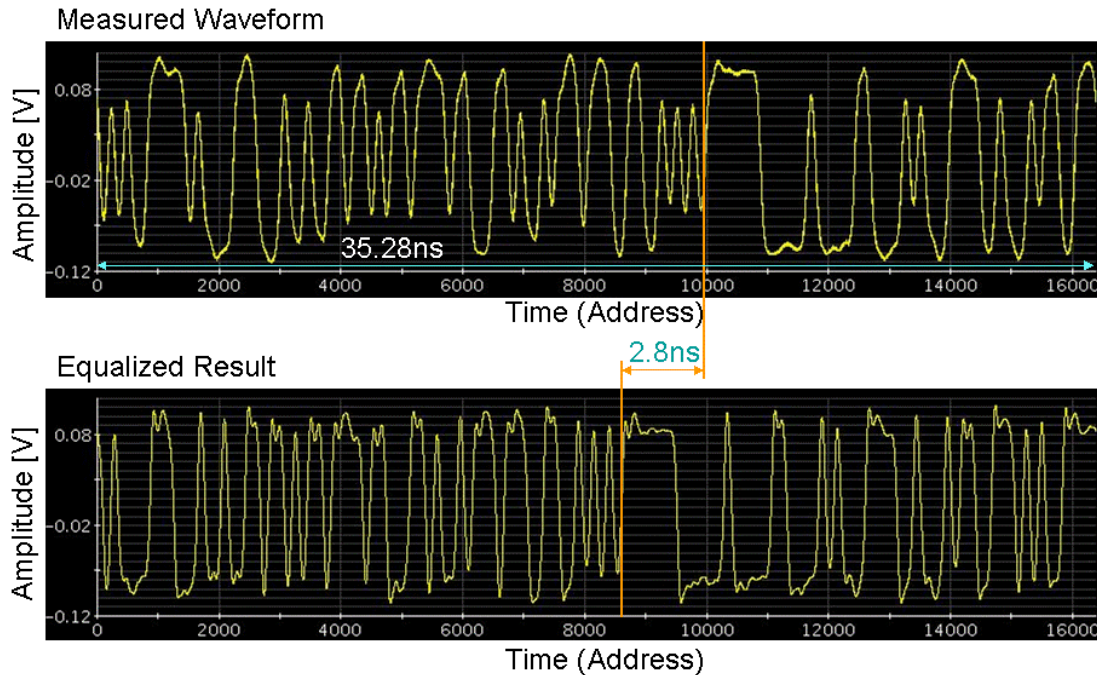
### Experimental Result

Figure 6 shows the actual measurement and equalized results. The measured waveform is a PRBS7 bit stream at 3.6Gbps. The amplitude of narrow pulses in the measured waveform is significantly attenuated than wide pulses. This is the effect of the frequency bandwidth of the signal path. The amplitude of the narrow pulses in the equalized waveform is very much improved and becomes

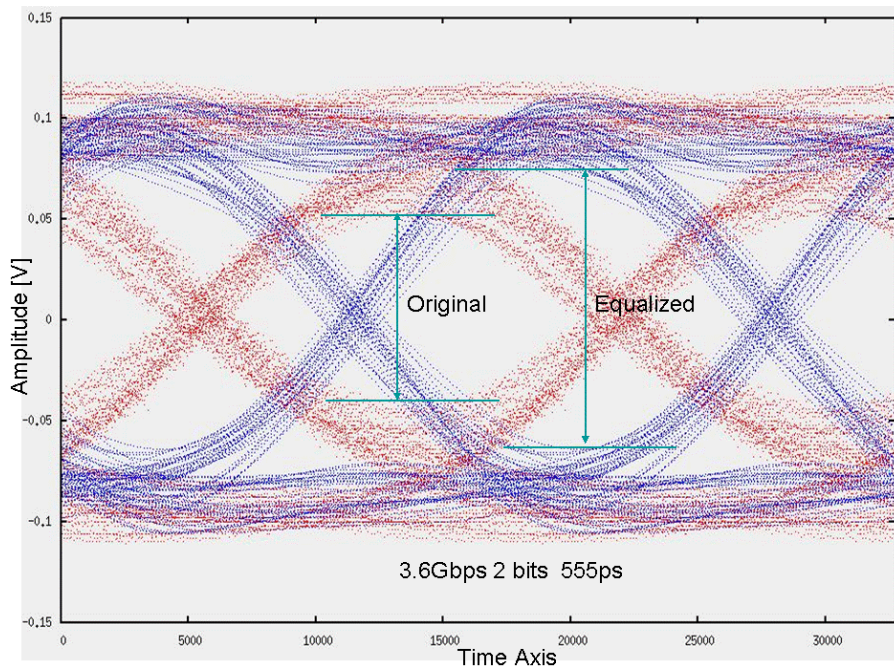
<sup>1</sup> Mixed Signal Lecture Series DSP-Based Testing – Fundamentals 35 F-matrix Simulation

close to the amplitude of wide pulses. On the other hand, this result correctly shows the time relationship between the measured and the equalized waveforms. The measured waveform is the output of the network so that it is delayed from the network input. Therefore the equalized result is ahead of the measured waveform by the time to travel in the network. This is the way the F-matrix simulation keeps the time consistency.

How much improved by the equalization can be easily recognized with comparing the eye patterns as Figure 7, where RMS values of both waveforms are matched.



**Figure 6: Measured Waveform and Equalized Result**



**Figure 7: Eye Pattern**

## Appendix

### Example Program Code

The physical dimensions of the coaxial cable and physical constants are as follows;

```
01: DOUBLE dOD=0.8382e-3;           // Insulator Outer Diameter 0.8382mm
02: DOUBLE dID=0.3048e-3;           // Inner Conductor Diameter 0.3048mm
03: DOUBLE K1=0.939;                 // (7x38) Braid Effect Factor for 7
04: DOUBLE dw=0.0799e-3;            // Braid Wire Diameter
05: DOUBLE rho_Cu=1.72e-8;           // Copper Resistivity [ohms.m]
06: DOUBLE rho_Ag=1.62e-8;          // Silver Resistivity [ohms.m]
07: DOUBLE dODthick=0.4572e-3;      // Outer Conductor Thickness
08:
09: DOUBLE mu=4.0*M_PI/1.0e7;        // Absolute Permeability
10: DOUBLE e0=8.854e-12;             // Absolute Dielectric Constant [F/m]
11: DOUBLE er=2.1;                   // Relative Dielectric Constant of TFE Teflon
12: DOUBLE Epsilon=e0*er;            // Dielectric Constant
13:
```

The cable parameters are formalized as follows;

```
20: DOUBLE dLe=(mu/(2.0*M_PI)*log(dOD/(K1*dID))); // External Inductance
21: DOUBLE dC=2.0*M_PI*Epsilon/log((dOD+1.5*dw)/(K1*dID)); // Capacitance
22:
23: DOUBLE dskinDepthOD;              // Skin Depth of Outer Conductor (Copper)
24: DOUBLE dskinDepthID;              // Skin Depth of Inner Conductor (Silver)
25: DOUBLE dG=0.0;                    // No leakage assumption
26:
27: DOUBLE dTdelay=sqrt(dLe*dC);       // Delay Time by the cable
28: DOUBLE dvelocity=1.0/dTdelay;     // Traveling speed of the waveform on the cable
29:
```

F-matrix variables and component variables are defined as follows;

```
30: Fmatrix F0,F1,F2,F3,F4,F5,F6,F7,F8,F9;
31: Fmatrix F10,F11,F12,F13,F14,F15,F16;
32:
33: DOUBLE L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14;
34: DOUBLE C1,C2,C3,C4,C5,C6,C7,C8,C9;
35: DOUBLE R1,R2,R3;
36: DOUBLE Len1,Len2,Len3,Len4,Len5,Len6;
37: DOUBLE t1,t2,t3,t4,t5,t6;
38: COMPLEX Zdec,Ydec;
39:
```

Component values are defined according to the equivalent circuit in Figure 4 as follows;

```
41: R1=50.0; // 50ohms
42: L1=1.0e-9; // 1nH
43: L2=L3=L4=L5=L6=L7=L8=L9=L10=L11=L12=L23=0.5e-9; // 0.5nH
44: C1=C6=C7=0.1e-12; // 0.1pF
45: C2=C3=C4=C5=C8=0.4e-12; // 0.4pF
46: t1=1.5 ns; t2=970.0 ps; t3=100.0 ps;
47: t4=45.0 ps; t5=50.0 ps; t6=75.0 ps;
48: R2=50.0; // 50ohms
49: R3=2.0e3; // 2kohms (Decoupling ckt.)
50: L14=15.0e-6; // 15uF
51: C9=4.0e-12; // 4pF
52:
```

Some more variables and conditions are defined as follows;

```
60: INT i, N, Nsp, Nx;
61: DOUBLE dUTP, dFs, dFresln, dFx, dFreq, w;
62: DOUBLE dRMS1, dRMS2, dMean;
63: ARRAY_D dwave1, dwave2, dsp1, dsp2, dEye1, dEye2;
64: ARRAY_COMPLEX Cwave1, Cwave2, CSp1, CSp2;
65:
66: Nbits=127; // 127 bits
67: dFbps=3.6 GHz; // 3.6 Gbps
68: N=16384;
69: dUTP=Nbits/dFbps;
70: dFs=1.0/(dUTP/N);
71: dFresln=dFs/N;
72: Nsp=N/2;
```

The measured waveform is stored in the array "dWave2[]."

```
80:
81: // dwave2[] holds the measured waveform data.
82:
```

Processing of the FFT & IFFT method in Figure 5 starts as follows;

```
90: DSP_MEAN(dwave2, &dMean, &dRMS2, N); // RMS of dwave2[]
91:
92: DSP_CONV_D_C(dwave2, Cwave2, 1.0, 0.0); // For full-range spectrum
93: DSP_FFT(Cwave2, CSp2, RECT); // Complex Spectrum
94:
95: dFx=6.0 GHz; // Equalizing Bandwidth
96: Nx=(INT)(dFx/dFresln+0.5); // Upper Limit Bin Number
97: for (i=Nx; i<=(N-Nx); i++) CSp2[i]=Zero; // Noise Suppression
98: // Equalizing band is DC to Bin(Nx-1)
```

The analog bandwidth for equalization should be limited to a moderate frequency which is 6GHz in this example. Otherwise the signal power of the core part would be overly suppressed.

The main part of the equalization processing is as follows; DC is a singular point so that it is handled independently. In F-matrix simulation, DC often becomes a singular point which must be separately

processed. For example,  $1/(\omega C)$  cannot be resolved at DC. It may be worked around by substituting the frequency as 1Hz instead of 0Hz.

```

100:   for (i=0;i<Nsp;i++) {
101:       dFreq=dFresln*i;           // Bin-wise Frequency
102:       w=2.0*M_PI*dFreq;         // Omega=2*PI*Freq (Angular frequency)
103:
104:       dSkinDepthOD=sqrt(rho_Cu/(M_PI*dFreq*mu)); // Skin depth of outer conductor
105:       dSkinDepthID=sqrt(rho_Ag/(M_PI*dFreq*mu)); // Skin depth of inner conductor
106:       dR=(1.0/M_PI)*(rho_Cu/(dOD*dSkinDepthOD)+rho_Ag/(dID*dSkinDepthID));
107:       dL=dLe+dR/w;              // dR/w is the skin effect inductance.
108:
109:       if (dFreq==0.0) {         // DC bin is the singular point.
110:           dFreq=1.0;           // "1Hz" is perfunctory to avoid overflow.
111:           w=2.0*M_PI*dFreq;
112:           dR=rho_Cu*len1*(1.0/(M_PI*(dID/2.0)*(dID/2.0))+1.0/(M_PI*dOD*dODthick));
113:           dL=dLe;
114:       }
115:
116:       F0=Fz(COMPLEX(R1,0.0));
117:       F1=Fzy(COMPLEX(0.0,w*L1),COMPLEX(0.0,w*C1));
118:       F2=Fxline(dR,dG,dL,dC,len1,dFreq);
119:       F3=Fzyz(COMPLEX(0.0,w*L2),COMPLEX(0.0,w*C2),COMPLEX(0.0,w*L3));
120:       F4=Fxline(dR,dG,dL,dC,len2,dFreq);
121:       F5=Fzyz(COMPLEX(0.0,w*L4),COMPLEX(0.0,w*C3),COMPLEX(0.0,w*L5));
122:       F6=Fzyz(COMPLEX(0.0,w*L6),COMPLEX(0.0,w*C4),COMPLEX(0.0,w*L7));
123:       F7=Fxline(dR,dG,dL,dC,len3,dFreq);
124:       Zdec=COMPLEX((1.0/R3),(-1.0/(w*L14)))+COMPLEX(0.0,(-1.0/(w*C9)));
125:       Ydec=One/Zdec;
126:       F8=Fzyz(COMPLEX(0.0,w*L8),COMPLEX(0.0,w*C5)+Ydec,COMPLEX(0.0,w*L9));
127:       F9=Fxline(dR,dG,dL,dC,len4,dFreq);
128:       F10=Fyz(COMPLEX(0.0,w*C6),COMPLEX(0.0,w*L10));
129:       F11=Fxline(dR,dG,dL,dC,len5,dFreq);
130:       F12=Fzy(COMPLEX(0.0,w*L11),COMPLEX(0.0,w*C7));
131:       F13=Fxline(dR,dG,dL,dC,len6,dFreq);
132:       F14=Fzyz(COMPLEX(0.0,w*L12),COMPLEX(0.0,w*C8),COMPLEX(0.0,w*L13));
133:       F15=Fy(COMPLEX((1.0/R2),0.0));
134:
135:       F16=F0*F1*F2*F3*F4*F5*F6*F7*F8*F9*F10*F11*F12*F13*F14*F15;
136:       // F-matrix Multiplication. Calculation order sensitive.
137:
138:       cSp1[i]=F16.a*cSp2[i];     // v1=A*v2
139:       if (i!=0) cSp1[N-i]=Conjugate(cSp1[i]); // Complex Conjugate
140:   }
141:   cSp1[Nsp]=Zero;              // Complex Zero
142:
143:   DSP_IFFT(cSp1,Cwave1);
144:   dwave1=Cwave1.getReal();     // Equalized Result
145:   DSP_MEAN(dwave1,&dMean,&dRMS1,N); // RMS of dwave1[]
146:   DSP_MUL_SCL(dRMS2/dRMS1,dwave1,dwave1); // Scaling by RMS matching
147:
148:   DSP_SHUFFLE(dwave1,dEye1,Nbits); // Eye Pattern (Equalized)
149:   DSP_SHUFFLE(dwave2,dEye2,Nbits); // Eye Pattern (Measured)
150:

```