6 One-Dimensional FFTs

6.7 LEAKAGE

The input to an FFT is not an infinite-time signal as in a continuous Fourier transform. Instead, the input is a section (a truncated version) of a signal. This truncated signal can be thought of as an infinite signal multiplied by a rectangular function. For a DFT, the product of the signal and the rectangular function is sampled (multiplied by a series of impulses). Because multiplication in the time domain corresponds to convolution in the frequency domain, the effect of truncating a signal is seen in the FFT results (Brigham, 1974). Figure 6.19 illustrates the effect truncation and sampling have on the Fourier transform.

Figure 6.19a shows z(t), a continuous cosine wave with a period of T_0. Its Fourier transform, Z(f) is two impulses, at $1/T_0$ and $-1/T_0$. Figure 6.19c shows the product of z(t) and u(t), a sampling function. The sampled signal z(t) x u(t) is truncated by multiplication with w(t), a rectangular function. Figure 6.19e shows the resulting signal, y(t), and its Fourier transform, Y(f) (the convolution of Z(f), U(f), and W(f)).

The DFT interprets its input as one complete cycle of a periodic signal. To create a periodic signal from the N samples (y(t)), we convolve y(t) with v(t), a series of impulses at intervals of T_0. T_0 is the length of the rectangular function as well as exactly one period of the input signal z(t). Notice that V(f), the Fourier transform of v(t), is a series of impulses located at multiples of $1/T_0$. Because the zero values of the side lobes of Y(f) are also located at multiples of $1/T_0$, multiplying Y(f) by V(f) in Figure 6.19g produces the same transform as in Figure 6.19c (the transform of the non-truncated signal).

If the length of the rectangular function (w(t)) is not equal to one period or a multiple of periods of the input signal, leakage effects appear in the DFT output. Figure 6.20 illustrates these effects. Notice that in this case T_1, the width of the rectangular function w(t), is not equal to T_0, the period of z(t). Because of the convolution of impulses at locations $-1/T_0$ and $1/T_0$ with W(f), which has zero values at multiples of $1/T_1$, Y(f) has zero values at frequencies other than multiples of $1/T_1$ or $1/T_0$. Convolution in the time domain of y(t) and v(t) in Figure 6.20g corresponds to multiplication of Y(f) and V(f) in the frequency domain. Because the samples in V(f) spaced at $1/T_1$ do not correspond to zero values in Y(f), noise (or leakage) in the DFT output is produced.

Another way to think of leakage is to conceptualize the DFT output as a series of bins at specific frequencies. If f_s is the sampling frequency and N

Figure 6.19 Development of DFT, Window = Input Period

6 One-Dimensional FFTs

Figure 6.20 Development of DFT, Window≠ Input Period

is the number of samples, the bins range from 0 Hz to $f_s/2$ Hz and are equally spaced in frequency f_s/N Hz apart. If the N input samples contain one period or a multiple of periods of the input signal, each of the frequency components falls into a frequency bin. If the N input samples do not contain one period or a multiple of periods, at least one of the frequency components falls between bins. The energy of this frequency component is distributed to the surrounding bins, producing a spectrum similar to Figure 6.20g (Brigham, 1974).

In a real system, it is difficult to capture exactly one period or a multiple of periods of a signal. In most cases, leakage in the FFT output will result. One method of reducing this leakage is called windowing.

Although windowing has many applications, we use it here to reduce leakage. Truncation of a signal is a form of windowing in which the window function is rectangular. Leakage caused by truncation can be reduced by selecting a non-rectangular window function with specific characteristics.

The window function is selected for two characteristics (Brigham, 1974). First, to reduce the effect of side lobe multiplication, the side lobes in the Fourier transform of the window function should be significantly smaller than those of the rectangular window function's Fourier transform. Second, the main lobe of the window function's Fourier transform should be sufficiently narrow so that important signal information is not lost. Two examples of window functions that exhibit these characteristics are the Hanning and the Hamming windows.

Hanning: $w(n) = 1/2\,[1 - \cos(2\pi n/(N-1))] \qquad 0 \le n \le N-1$

Hamming: $w(n) = 0.54 - 0.46\cos(2\pi n/(N-1)) \qquad 0 \le n \le N-1$

Noise reduction is accomplished by dividing the selected window function into N equally spaced samples (called window coefficients) and multiplying each FFT input sample by the corresponding coefficient. The module *window*, shown in Listing 6.37, performs this calculation. This module works with both the radix-2 and radix-4 DIF subroutines. It assumes that the *inplacedata* buffer contains sequentially ordered data organized with real and imaginary values interleaved. It is written for a 1024-point FFT, but the window size can be changed by changing the constant *N_x_2*.

The *window_coeffs* buffer is initialized with the window coefficients. This buffer is organized with real and imaginary values interleaved. The buffer is initialized with an external file *window_coeffs.dat* that contains the precalculated window coefficients.

Pointers are set to point to the *inplacedata* and *window_coeffs* buffers. The initial data fetch of a window coefficient and a sample is done before the loop. Inside the loop, a coefficient and sample are multiplied at the same time as the next coefficient and sample are read. After each multiplication, the product is written over the original FFT input sample. The loop is repeated for N samples (real and imaginary parts).

```
.MODULE       windowing;

{             Calling Parameters
                 FFT input data in the inplacedata buffer

              Return Values
                 Windowed FFT input data in the inplacedata buffer

              Altered Registers
                 I0,I1,I4,M0,M4,MX0,MY0,MR
}

.CONST        N_x_2=2048;

.VAR/PM       window_coeffs[N_x_2];

.INIT         window_coeffs:<window_coefs.dat>;

.ENTRY        window;


window:       I0=^inplacedata;         {I0 --> 1st sample in FFT input data}
              I1=I0;
              I4=^window_coeffs;       {I4 --> 1st window coefficient}
              M0=1;
              M4=1;
              CNTR=N_x_2-1;
              MX0=DM(I0,M0),MY0=PM(I4,M4);    {Read 1st sample and coefficient}
              DO window_loop UNTIL CE;        {Window N_x_2-1 samples}
                 MR=MX0*MY0(RND),MX0=DM(I0,M0),MY0=PM(I4,M4);
window_loop:     DM(I1,M0)=MR1;
              MR=MX0*MY0(RND);         {Multiply last sample and coefficient}
              DM(I1,M0)=MR1;           {Last sample updated}
              RTS;
.ENDMOD;
```

Listing 6.37 Windowing Routine

6.8 BENCHMARKS

Benchmarks for the optimized radix-2 DIT FFT and radix-4 DIF FFT routines are given in this section.

Straight-line code occupies much more memory than looped code, is hard to understand and is tedious to debug. All programs used to generate the benchmarks presented in this section are relatively short and uncomplicated looped programs.

The FFT benchmarks in Table 6.1 are worst-case. For example, the 1024-point radix-2 FFT benchmark assumes that the data grows by two bits in every stage.

Routine	*Number of Points*	*Number of Cycles*	*Execution Time* (12.5MHz ADSP-2100A)
Radix-2 DIT Input scaling	1024	52911	4.23 ms
Radix-2 DIT Conditional BFP*	1024	113482	9.08 ms
Radix-4 DIF Input scaling	64	1381	0.11 ms
	256	7372	0.59 ms
	1024	37021	2.96 ms
Radix-4 DIF Input scaling Built-in digit-reverse	64	1405	0.11 ms
	256	7423	0.59 ms
	1024	37203	2.98 ms

* BFP = Block Floating-Point Scaling

Table 6.1 Benchmarks for FFT Routines

6 One-Dimensional FFTs

Table 6.2 lists the benchmarks for the other routines presented in this chapter (bit reversal, digit reversal, and windowing).

Routine	*Number of Points*	*Number of Cycles*	*Time* (12MHz ADSP-2100A) *(μs)*
Bit-Reverse (scramble, real data)	64	138	11.04
	128	266	21.28
	256	522	41.76
	1024	2058	164.64
Bit-Reverse (unscramble, complex data)	64	270	21.60
	128	526	42.08
	256	1038	83.04
	1024	4110	328.80
Digit-Reverse (unscramble, complex data)	64	430	34.40
	256	1812	144.96
	1024	7188	575.04
Window (complex data)	64	267	21.36
	128	523	41.84
	256	1035	82.80
	1024	4107	328.56

Table 6.2 Benchmarks for Other Routines

6.9 REFERENCES

Brigham, E.O. 1974. *The Fast Fourier Transform*. Englewood Cliffs, N.J.: Prentice-Hall, Inc.

Burrus, C.S. and T.W. Parks. 1985. *DFT and Convolution Algorithms*. New York, NY: John Wiley and Sons.

Dudgeon, D. and Mersereau, R. 1984. *Multidimensional Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall Inc.

Gonzalez, R. and Wintz, P. 1977. *Digital Image Processing*. Reading, MA: Addison-Wesley Publishing Company.

Hakimmashhadi, H. 1988. "Discrete Fourier Transform and FFT." *Signal Processing Handbook*. New York, NY: Marcel Dekker, Inc.

Haykin, S. 1983. *Communication Systems*. New York: John Wiley and Sons.

Oppenheim, A. V., and Schafer, R. W. 1975. *Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall, Inc.

Proakis, J. G. and D. G. Manolakis. 1988. *Introduction to Digital Signal Processing*. New York, NY: Macmillan Publishing Company.

Rabiner, L. R. and Gold, B. 1975. *Theory and Applications of Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall, Inc.

Rabiner, Lawrence R. and Gold, Bernard. 1975. *Theory and Applications of Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall, Inc.

6 One-Dimensional FFTs