Notes on using Analog Devices' DSP, audio, & video components from the Computer Products Division
Phone: (800) ANALOG-D or (781) 461-3881, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com

Choosing and Using FFTs for ADSP-21xx

Last Modified: 06/18/96

FFT is a fast algorithm for computing DFT. There are several different implementations on fixed point ADSP-21xx. This EZ-Notes will help you choose the different implementations and use the results.

The equation for computing DFT is,

$$X(k) = \Sigma \{x(n)*W(n)\}$$

Choosing different FFT implementation

The main considerations in choosing FFT implementations are: speed and accuracy. The factors include radix, points in FFT, windowing and scaling. Since the DIT and DIF give the same accuracy at the same speed, choosing one or the other won't affect performance.

1. DIT and DIF

FFT has a special processing order. This results in a requirement on a special input order (address bit-reversed) or the FFT will produce the output in the special order. The FFT which takes the bit-reversed ordered input and generates a sequential ordered output is called 'decimation in frequency' (DIF) and the FFT which takes the sequential ordered inputs but generates a bit-reversed ordered output is called 'decimation in time' (DIT). The two approaches give the same results if the orders are adjusted appropriately. In the DIT case, an input is in original data order and, after FFT processing, the output should be adjusted (unscrambled). In DIF case, a input is pre-adjusted(scrambled) and then FFT is called and the output is in the correct order. The flow diagrams for two implementation is as below.

DIT
inputs
|
scramble the inputs
|
FFT
|
outputs

DIF
inputs
|
FFT
|
unscramble the outputs
|
outputs

2. Radix 2 vs. radix 4

The FFT divides DFT into smaller DFTs. If the division ratio is two, the FFT is called radix-2. If the ratio is four, the FFT is called radix-4. The results from two approaches are the same. Compared with radix-2, the radix-4 trades more complex data addressing and twiddle factors with less computation. Due to the advanced addressing mechanism, the ADSP-21xx can process the radix-4 FFT significant faster than radix-2 FFT.

For instance, 1024 points radix-2 DIT(and DIF) FFT takes 73928 cycles while radix-4 takes 37016 cycles, which saves about half of cycles radix-2 takes.

3. Points in FFT

The number of points in FFT depends on the resolution we need. An FFT output can be thought as a series of bins at specific frequencies evenly spread over the range 0 to f/2 (f is the sampling frequency) with the space of f/N. If the N input samples contain full period or multiple period, the frequency component falls into a frequency bin. If the N input points does not contain one full period or multiple periods, some frequency components will split to two adjacent bins. Therefore f/N decides the resolution of the frequency domain. Note that, there is a tradeoff between the higher resolution and more computation. The larger the N the better the resolution and the more DSP horsepower needed.

graph illustration

4. Windowing

If exactly one period or a multiple of periods of a signal can not be captured exactly, the leakage(side lobe) will result. The windowing is a technique to reduce the leakage. The window function is selected for two characteristics: reducing the effects of side lobe and narrowing the main lobe of the window. There are two

commonly used windows: Hanning and Hamming windows as shown below.

Hanning: w(n) = 1/2[1-cos(2*pi*n/(N-1))]
Hamming: w(n) = 0.54 - 0.46*cos(2*pi*n/(N-1))

The N input samples to your FFT are multiplied by the windowing coefficients before executing the FFT. A 'Rectangular window' is the default and results when no multiply is performed (think of it as multiplying your input data by a rectangle of 1's).

5. Scaling

The scaling is an important difference between the floating point FFT and fixed point FFT implementations. At each calculation stage of fixed point FFT butterfly, the computation could cause the data to grow by two bits from the input to output. To avoid the overflow on the fixed-point chips, we have to scale the data. There are three possible ways to do this: input data scaling, unconditional block floating-point scaling and conditional block floating-point scaling.

The input scaling (IS) FFT doesn't scale at all, so the input data must be pre-scaled by $\log_2 N+1$. This means that for a 1024-pt FFT, you'd have to scale the data by 10+1 or 11 bits. For the 16-bit 21xx family this would leave you with only 5 bits of valid data giving you a large truncation error. However, input scaling is the fastest and may be a good choice for small N FFTs. The unconditional block floating-point scaling (UCBFS) scales the input data unconditionally at each butterfly stage. The data must still be pre-scaled to give two guard bits. The conditional block floating-point scaling (CBFS) scales the input data at each butterfly stage by 2 IF NECESSARY. The number of scaling down bits is equal to the number of growth bits of the previous butterfly stage. The UCBFS is the special case of CBFS which assumes the data always grows two bits at each butterfly stage.

As for the accuracy, the CBFS gives the best result and the UCBFS is next. The IS gives worst accuracy. As for the speed, the IS is fastest. The CBFS and UCBFS nearly the same speed (both have butterfly inner loops of 12 instructions) which is about 30% slower than IS (whose butterfly inner loop is 9 instructions). Since the UCBFS has almost the same speed as CBFS but with less accuracy, the CBFS is recommended for large FFTs.

Using of FFTs on ADSP-21xx

The output of FFT processing is the complex DFT X(k) of the input signal where k is the discrete frequency point. The square of magnitude of the spectrum is

real{X(k)} * real{X(k)} - imag{X(k)} * imag{X(k))}

and phase is

arctan [imag{X(k)}/real{X(k)}]

In most cases, you use the square of magnitude of the spectrum. In some cases such as in image processing, the phase information may also be used.

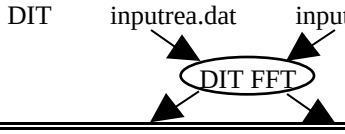
1. Example files

Several versions of FFTs have been coded in ADI assembly language and are available on the FTP and in the *Digital Signal Processing Applications Using the ADSP-2100 Family applications* book. The example files for different fixed point FFT implementations are

DSP files:		Inputs
DIT	dit_main.dsp	inputrea.dat
	dit_fft.dsp	inputima.dat
	scramble.dsp	twidreal.dat
	bfp_adj.dsp	twidimag.dat
DIF	dif_main.dsp	inplaced.dat
	dif_fft.dsp	(interleaved real
	unscramb.dsp	and imaginary data)
	bfp_adju.dsp	twidreal.dat
		twidimag.dat
Radix-4	m4n1024.dsp	input.dat (interleaved)
	m4n1024.dsp	cos1024.dat

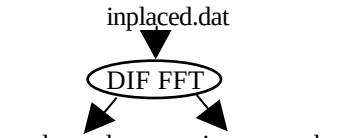
2. The inputs and outputs

The inputs can be loaded from a file on your hard drive by reassembling the program or by loading the memory directly from a file. When finishing the execution of a program, the outputs are saved in simulator memory. The outputs for both DIT and DIF are separated into real and imaginary parts while the outputs are interleaved and saved as one file in Radix-4 FFT. The diagram of testing the DIT and DIF is as below.

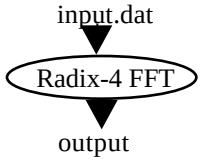


inplacereal inplaceimag

DIF



Radix-4



All the input and output files are in hexadecimal format.

3. Checking the results

The FFT result can be checked by plotting the result in the ADSP-21xx simulator data memory. The simulator command is,

plot dm(address)/length n

where n indicates interleaving span.