Application Report
SPRA554B

Implementation of the Double-Precision Complex FFT for the TMS320C54x DSP

Mike Hannah — *TI MSDS Emerging Markets and Technologies Group*
Aaron Kofi Aboagye — *TI C5000 DSP Software Applications Group*

Abstract

A double-precision complex Fast Fourier Transform (FFT) C-callable code library has been developed for the Texas Instruments (TI™) TMS320C54x fixed-point digital signal processor (DSP). The library includes routines and the necessary 32-bit coefficients to process both the FFT and the inverse FFT (IFFT) for 32-bit complex-valued input data. The routines can accommodate input data lengths of 8, 16, 32, 64, 128, 256, 512, and 1024 points. The library was written and optimized in assembly to minimize processor utilization. Benchmarking results and memory utilization are summarized in this application report.

Contents

Figures

Tables

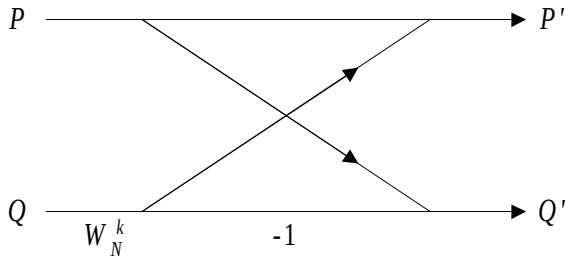
Algorithm Overview

The double-precision complex FFT algorithm was developed using a radix-2 decimation-in-time FFT butterfly.[1] All of the calculations are performed in place to conserve memory. There is no bit reversal inside of the FFT. Inputs to the FFT must be in bit-reversed order to generate ordered outputs. Inputs must be ordered with the real component of the data point followed by the corresponding imaginary component with the most significant word first for each component.

Fast Fourier Transform (FFT)

The radix-2 butterfly signal flowgraph is shown in Figure 1, where P and Q are consecutive complex-valued double-precision input data points, W_N^k is the complex-valued, double-precision FFT coefficient, or twiddle factor, and P' and Q' are the complex-valued double-precision output data points. W_N^k is equal to $e^{-j2\pi k/N}$, where $k = 0,\ldots,N-1$, and N is the length of the FFT.

Figure 1. Radix-2 Butterfly Signal Flowgraph



The inputs and outputs can be separated into real components P_R and Q_R and imaginary components P_I and Q_I for the FFT calculations as:

$$P'_R = P_R + (Q_R W_R - Q_I W_I)$$

$$Q'_R = P_R - (Q_R W_R - Q_I W_I)$$

$$P'_I = P_R + (Q_I W_R + Q_R W_I)$$

$$Q'_I = P_R - (Q_I W_R - Q_R W_I)$$

These equations were implemented in a highly optimized loop for the butterfly kernel with the equations for $W = 1$, that is, $W_R = 1$ and $W_I = 0$, calculated outside the butterfly loop because the double-precision multiplication by coefficients could be eliminated. The resulting equations implemented outside the butterfly kernel loop were then:

$$P'_R = P_R + Q_R$$

$$Q'_R = P_R - Q_R$$

$$P'_I = P_R + Q_I$$

$$Q'_I = P_R - Q_I$$

The butterfly kernel was used for all stages of the FFT and IFFT except the first, second, and third stages. Other simplifications were made to reduce computations for those stages of the FFT. Stages 1 and 2 were combined into one routine using a radix-4 butterfly. Using a radix-4 FFT that has an FFT coefficient of unity eliminates all multiplication. The equations that describe the radix-4 butterfly output for the combined first and second FFT stages are:

$$R'_1 = R_1 + R_2 + R_3 + R_4$$

$$I'_1 = I_1 + I_2 + I_3 + I_4$$

$$R'_2 = R_1 - R_2 + I_3 - I_4$$

$$I'_2 = I_1 - I_2 - R_3 + R_4$$

$$R'_3 = R_1 + R_2 - R_3 - R_4$$

$$I'_3 = I_1 + I_2 - I_3 - I_4$$

$$R'_4 = R_1 - R_2 - I_3 + I_4$$

$$I'_4 = I_1 - I_2 + R_3 - R_4$$

where R is the real component of the output and I is the imaginary component of the output.

For the third FFT stage, the general radix-2 butterfly kernel was implemented with some slight modifications. Because some of the FFT coefficient values were based on the sine of 45 degrees, the equivalent real and complex components allowed for reuse of some of the multiplication products.

Inverse Fast Fourier Transform (IFFT)

Although forming the complex conjugate of the input data and reusing the FFT routines could be used to carry out the IFFT, separate IFFT routines were developed. By the IFFT definition, the IFFT routines have the appropriate sign changes to the butterfly equations to account for the opposite sign of the complex component of the coefficients. By having routines for both the FFT and the IFFT, you are not forced to deal with sign issues. Also, the same coefficients are used for both the FFT and IFFT to prevent wasting memory with separate coefficient tables for the differences in complex components. The equations that describe the radix-4 butterfly output for the combined first and second FFT stages of the IFFT are:

$$R'_1 = R_1 + R_2 + R_3 + R_4$$

$$I_1' = I_1 + I_2 + I_3 + I_4$$

$$R_2' = R_1 - R_2 - I_3 + I_4$$

$$I_2' = I_1 - I_2 + R_3 - R_4$$

$$R_3' = R_1 + R_2 - R_3 - R_4$$

$$I_3' = I_1 + I_2 - I_3 - I_4$$

$$R_4' = R_1 - R_2 + I_3 - I_4$$

$$I_4' = I_1 - I_2 - R_3 + R_4$$

Scaling

Unless the input signal amplitude is very low, the computation of the FFT of a signal often leads to overflows in the outputs. To prevent this, both the FFT and IFFT functions allow you to specify whether or not the outputs should be scaled to avoid overflow. If the scaling option is chosen; that is, the scale parameter is not zero (0), the outputs are scaled by a factor of 1/N. This is achieved by scaling down the data points by a factor of 2 using a right shift at the output of each FFT or IFFT stage, except for the combined first and second stage, where the output data points are scaled down by a factor of 4. By FFT convention, if scaling is used during the FFT, it must not be used by the IFFT and vice-versa because the IFFT is defined with a scale factor of 1/N.

Using the Double-Precision Complex FFT

To use the double-precision FFT or IFFT for the C54x, the DSP library C header file `dsplib.h` must be included in the file calling the double-precision FFT or IFFT function.

Version 1.1 or above of the C54x DSPLIB includes the cfft_32.asm and cifft_32.asm in library format. In this way, the necessary routines are included only by linking with the 54xdsp.lib library. (Note: Version 1.1 of the DSPLIB is expected to be released in December of 1999, the code will be available on the TI FTP site prior to that date).

If you have version 1.0 of the C54x DSPLIB, you must link the cfft_32.asm and cifft32.asm files. The `cfft_32.asm` file contains all of the common functions necessary to implement the double-precision FFT. The `cifft_32.asm` file contains all of the common functions necessary to implement the double-precision IFFT.

The FFT and IFFT functions have been written to utilize the same coefficient tables contained in the file sintab.q31. This file contains directives ensuring that only the coefficient tables necessary for the FFT or IFFT size being implemented are linked in.

The FFT or IFFT of length 8 is a special case. If the double-precision FFT or IFFT of length 8 is implemented, access to coefficient tables is not required and, as such, there is no need to link in any coefficient table.

Input to the double-precision FFT or IFFT can be in normal order or bit-reversed as long as the input is ordered with the real component of the data point followed by the corresponding imaginary component with most significant word first for each component. Figure 2 shows a representation of memory for a normal-ordered input to the FFT or IFFT. MSW is the most significant 16-bit word and LSW is the least significant 16-bit word. Output data resides in the same memory space as the input data.

Figure 2. Example of Double-Precision Complex FFT/IFFT Normal-Ordered Input Data

```
Addr   |<---------------------- Data Point ---------------------->|
0000   Real X(0) MSW | Real X(0) LSW | Imag X(0) MSW | Imag X(0) LSW
0004   Real X(1) MSW | Real X(1) LSW | Imag X(1) MSW | Imag X(1) LSW
0008   Real X(2) MSW | Real X(2) LSW | Imag X(2) MSW | Imag X(2) LSW
 .           .               .               .               .
 .           .               .               .               .
 .           .               .               .               .
```

The double-precision complex FFT or IFFT can be called using the prototypes:

```
void cfft_32( LDATA x, NX, short scale);
void cifft_32( LDATA x, NX, short scale);
```

LDATA x is the pointer to the FFT/IFFT input data vector containing NX complex elements stored in real-imaginary component order. The LDATA type is a 32-bit-long data type that holds data with 31 fractional bits; therefore, you should allocate 2*NX elements of 32-bit (or LDATA) width to the x array.

NX is the length of the FFT or IFFT. NX must be a constant (not a variable) and can equal the values 8, 16, 32, 64, 128, 256, 512, or 1024.

scale is a flag that determines if the FFT or IFFT output data is scaled down by NX. If scale is set to 0, scaling occurs; if scale is set to any value other than 0, no scaling of the data is performed. Scaling is the overflow prevention methodology implemented in the double-precision FFT and IFFT.

Benchmarks

Table 1 details double-precision FFT benchmarks for different FFT lengths. If multiple FFT lengths are implemented, the code size will not be the sum of the two code sizes because most of the routines are common modules. The actual code size will only be slightly larger than the largest code size of the lengths being implemented. In the same regard, the coefficient size will correspond to the largest of the FFT lengths being implemented because all smaller lengths can use a subset of that particular coefficient table.

Table 1 and

Table 2 compare the single-precision FFT and the double-precision FFT benchmarks.

Table 1. C54x Benchmarks for the Double-Precision Complex FFT

FFT Length	C54x Clock Cycles	Code Size (Words)	Coefficient Size (Words)
8	297	389	0
16	796	407	26
32	2097	429	74
64	5263	452	170
128	12749	475	362
256	30059	498	746
512	144205	521	1514
1024	408051	544	3050

Notes: 1) Benchmarks are identical for the IFFT.
2) Assumes all data is in on-chip DARAM and there are no bus conflicts.

Table 2. C54x Benchmarks for the Single-Precision Complex FFT

FFT Length	C54x Clock Cycles	Code Size (Words)	Coefficient Size (Words)
8	149	109	0
16	322	151	11
32	733	199	34
64	1672	247	81
128	3795	295	176
256	8542	343	367
512	19049	391	750
1024	42098	439	1517

Notes: 1) Benchmarks are identical for the IFFT.
2) Assumes all data is in on-chip DARAM and there are no bus conflicts.

References

[1] "Implementation of the Fast Fourier Transform with the TMS32020," *Digital Signal Processing Applications with the TMS320 Family*, Volume 1, (SPRA012), 1989, pp. 69-168.

TMS320C54x DSPLIB User's Guide, (SPRA480), December 1998

TI Contact Numbers

INTERNET

TI Semiconductor Home Page
www.ti.com/sc

TI Distributors
www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

Americas
Phone +1(972) 644-5580
Fax +1(972) 480-7800
Email sc-infomaster@ti.com

Europe, Middle East, and Africa
Phone
Deutsch +49-(0) 8161 80 3311
English +44-(0) 1604 66 3399
Español +34-(0) 90 23 54 0 28
Francais +33-(0) 1-30 70 11 64
Italiano +33-(0) 1-30 70 11 67
Fax +44-(0) 1604 66 33 34
Email epic@ti.com

Japan
Phone
International +81-3-3344-5311
Domestic 0120-81-0026
Fax
International +81-3-3344-5317
Domestic 0120-81-0036
Email pic-japan@ti.com

Asia
Phone
International +886-2-23786800
Domestic
Australia 1-800-881-011
TI Number -800-800-1450
China 10810
TI Number -800-800-1450
Hong Kong 800-96-1111
TI Number -800-800-1450
India 000-117
TI Number -800-800-1450
Indonesia 001-801-10
TI Number -800-800-1450
Korea 080-551-2804
Malaysia 1-800-800-011
TI Number -800-800-1450
New Zealand 000-911
TI Number -800-800-1450
Philippines 105-11
TI Number -800-800-1450
Singapore 800-0111-111
TI Number -800-800-1450
Taiwan 080-006800
Thailand 0019-991-1111
TI Number -800-800-1450
Fax 886-2-2378-6808
Email tiasia@ti.com

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE (“CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty, or endorsement thereof.