Implementing fft with On-Chip RAM

in FLEX 10K Devices

February 1998, ver. 3 Application Note 84

Introduction

The fast Fourier transform (FFT) function forms the fundamental building block in many digital signal processing (DSP) applications, including communications, voice recognition, spectrum analysis, quadrature amplitude modulation (QAM), asymmetric digital subscriber line (ADSL), radar, and image manipulation. The `fft_on_chip` reference design implements an FFT function using the Altera® `fft` MegaCore™ function and on-chip RAM.

This application note explains how to instantiate and simulate `fft` in a FLEX® 10K design, using the `fft_on_chip` reference design and the Altera MAX+PLUS® II software. The instructions in this application note are valid for both PCs and UNIX workstations and assume the following:

- MAX+PLUS II version 8.1 or higher is located in the default location, **c:\maxplus2** directory for PCs and **/usr/maxplus2** for UNIX workstations.
- The `fft` MegaCore files have been installed to the default location, **c:\megacore** directory for PCs ; there is no default directory for UNIX workstations.
- Your command search path includes the **megacore\bin** directory. Otherwise, you must enter a full pathname to execute **twiddle.exe**.
- You have added the **megacore\lib** directory as a user library in the MAX+PLUS II software.
- You have purchased a license for the `fft` MegaCore function or are using the Altera OpenCore™ feature.

You can use Altera's OpenCore feature to instantiate, compile, and simulate the `fft` MegaCore function. However, you must obtain a license from Altera before you can generate programming or configuration files.

fft_on_chip

The `fft_on_chip` reference design uses the FLEX 10K embedded array architecture to provide on-chip RAM for storing data and twiddle factors during the data processing stages. Two memory blocks store data during read and write transactions and one memory block stores the twiddle factors. Each memory block stores both real and imaginary elements that are read in parallel.

The following discrete Fourier transform (DFT) equation normally requires N^2 multiplications, where N represents the number of points in the FFT. However, by using the decimation in frequency (DIF) FFT algorithm implemented with `fft`, the required number of multiplications is reduced to $(N/2)\log_2 N$.

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{((-j2\pi)/N)^{nk}}$$

$$= \sum_{n=0}^{N-1} x(n)(e^{(-j2\pi)/N})^{nk}$$

$$= \sum_{n=0}^{N-1} x(n)(W_N)^{nk}$$

where:
- k = 0, 1, 2,...$(N/2) - 1$
- N = Number of points
- n = Current input data point
- j = $\sqrt{-1}$
- $(W_N)^{nk}$ = Twiddle factor

For more information on `fft_on_chip`, refer to *Functional Specification 7 (fft_on_chip Fast Fourier Transform)*. For more information on the `fft` MegaCore function, refer to the *fft Fast Fourier Transform Data Sheet*.

Instantiating fft_on_chip

The following steps explain how to instantiate the `fft_on_chip` reference design using the MAX+PLUS II software. The typical PC installation of MAX+PLUS II creates Windows program items so that all required programs can be launched from a Windows icon. In a UNIX environment, you can bring up the MAX+PLUS II graphical user interface by typing `maxplus2` at the UNIX command prompt.

For detailed information on how to use the MAX+PLUS II software, refer to MAX+PLUS II Help.

1. Create an **ffttest** directory on your computer and then change to this directory by typing the following commands at a DOS or UNIX prompt. This directory will be the project directory.

```
mkdir ffttest
cd ffttest
```

2. Copy the Simulator Channel File (**fft_on_chip_walk.scf**) from the FFT **walkthru** directory into your **ffttest** directory.

The **walkthu** directory is installed to **c:\megacore\fft\walkthru**.

3. Generate the twiddle factors. In the **ffttest** directory, type the following command at a DOS or UNIX prompt:

   DOS `c:\megacore\bin\twiddle 4 8 ↵`
   UNIX `/megacore/bin/twiddle 4 8 ↵`

   This command generates the **tw4_8.mif** file, which contains $2^4/2 = 8$ twiddle coefficients.

   For help on the syntax of **twiddle.exe**, type `twiddle` with no arguments at the command prompt.

   tw4_8.mif is a Memory Initialization File (**.mif**) in the FLEX 10K embedded array block (EAB) memory format. Using any text editor, you can verify the following contents:

```
depth = 8;
width = 16;
address_radix = dec;
data_radix = hex;
content
begin
0: 8000;
1: 8A31;
2: A55B;
3: CF76;
4: 007F;
5: 3176;
6: 5B5B;
7: 7631;
end;
```

   The twiddle coefficients start after the `begin` line. After scaling for the requested bit depth (8 bits in this example), each line lists the twiddle factor in hexadecimal format. The real part of the twiddle number is listed first; then the imaginary part. For example, twiddle number `0` is `8000` hex. This means the real part of the twiddle number is `80` hex, and the imaginary part is `00` hex.

   For an *n*-point FFT, the twiddle factors are simply exp(jωt) values from the unit circle on the complex number plane. There are *n* such values equally spaced around the upper half (positive imaginary value) of the unit circle, with the first value on the positive-real axis.

4. Start the MAX+PLUS II software.

5. Choose **New** (File menu) to create a new Graphic Design File (**.gdf**).

6. Double-click in the Graphic Editor window. In the **Enter Symbol** dialog box, double-click on the *\megacore\lib* symbol library and then select **fft_on_chip** in the *Symbol Files* box. When the **Edit Ports/Parameters** dialog box opens, choose **Cancel**; you will enter this information at a later stage.

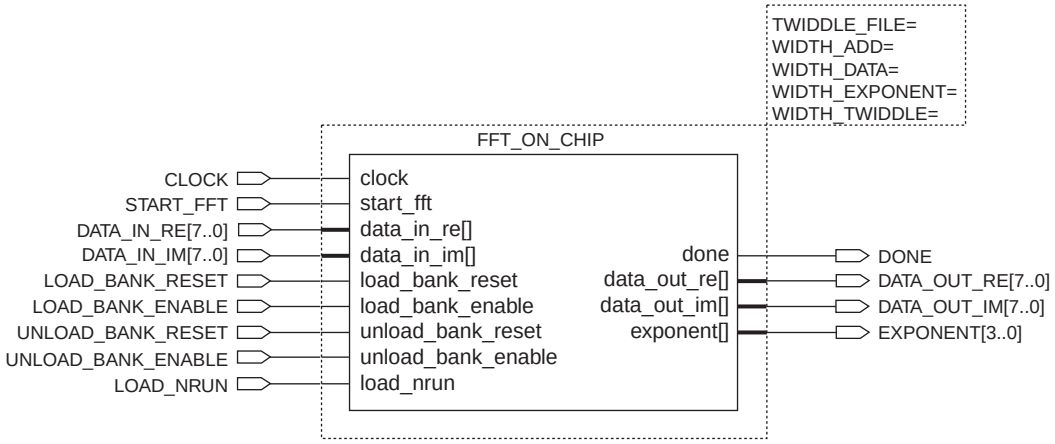
   If the **megacore\lib** symbol library does not display, you can add it as a user library with the **User Libraries** command (Options menu).

   You can use a version of the **fft_on_chip_walk.gdf** file that already contains all the inputs, outputs, and parameters. Copy **fft_on_chip_walk.gdf** from the **walkthru** directory into your **fft** directory. If you use this file, skip steps 7 through 10.

7. Enter, name, and connect the input and output pin symbols to the `fft_on_chip` symbol. For the simulation to work, the GDF pin names must match the `fft_on_chip` symbol port names, as shown in Figure 1.

Figure 1. Labeling the Input & Output Pins



8. Save the file as **fft_on_chip_walk.gdf** in your **ffttest** directory.

   Do not save the file as **fft_on_chip.gdf** because it will make the function recursive in the hierarchy tree, resulting in an error.

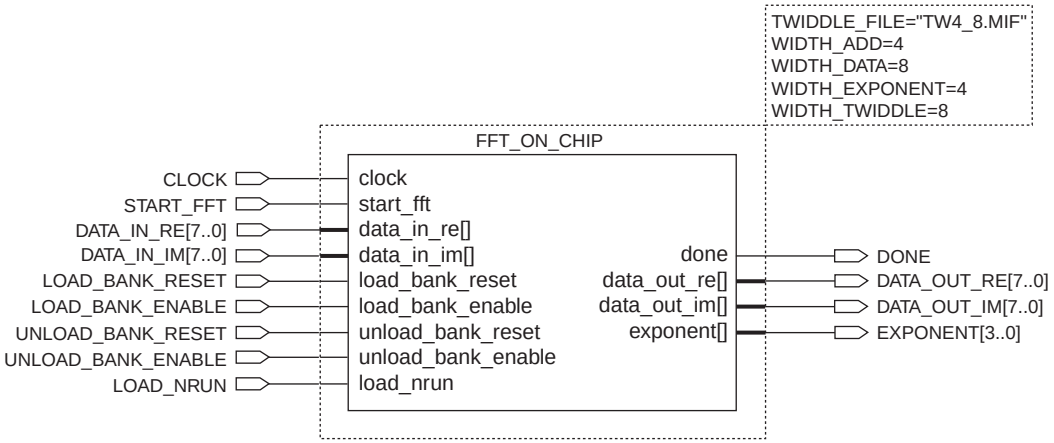
9. Set the parameter values for the `fft_on_chip` symbol with the **Edit Ports/Parameters** command (Symbol menu). Table 1 shows the parameter values for the design described in this application note.

Table 1. Parameter Values

Parameter	Value
TWIDDLE_FILE	"TW4_8.MIF"
WIDTH_ADD	4
WIDTH_DATA	8
WIDTH_EXPONENT	4
WIDTH_TWIDDLE	8

Figure 2 shows how the **fft_on_chip_walk.gdf** file should appear after you set the parameter values.

Figure 2. fft_on_chip_walk.gdf File



10. Save your file by choosing **Save** (File menu).

11. Set your project to the current file by choosing the **Project\Set Project to Current File** command (File menu).

12. Set the synthesis options for your design using the **Device** command (Assign menu). Select *FLEX 10K* as the target device family and *AUTO* as the target device.

13. To obtain the best synthesis results, set the following options in the **Define Synthesis Style** dialog box, which is available from the **Global Project Logic Synthesis** dialog box (Assign menu). Turn on the *Use LPM for AHDL Operators* option and set the *Style* to *FAST*.

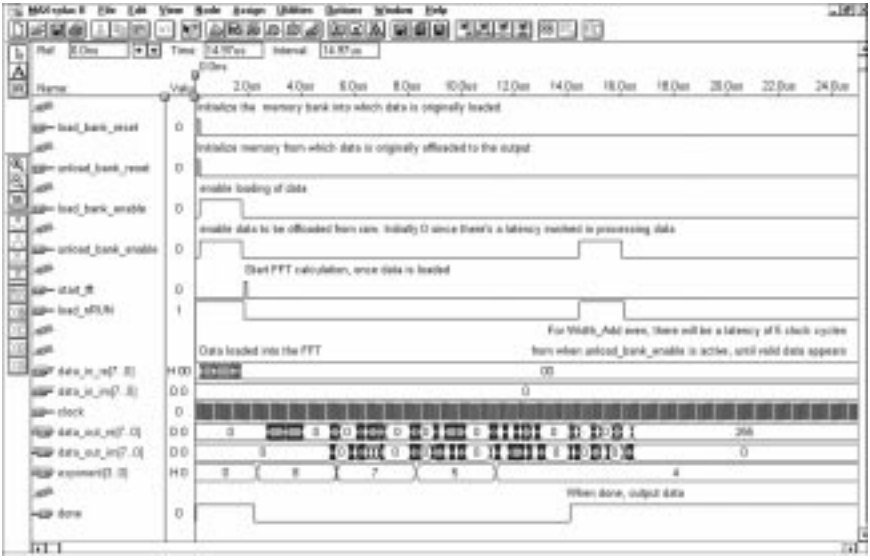
14. Save and compile the design.

Simulating fft_on_chip

The following steps show how to simulate `fft_on_chip` using the MAX+PLUS II development system.

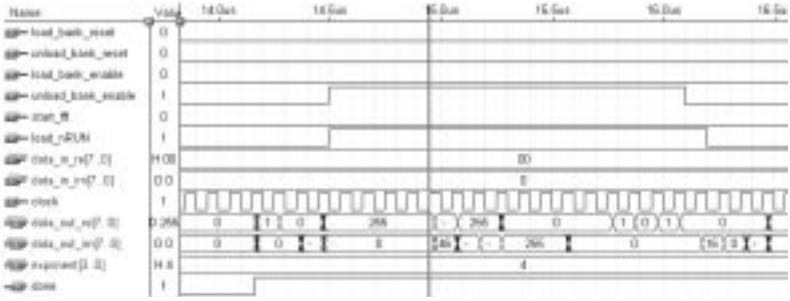
1. Open the MAX+PLUS II Simulator.
2. Choose **Inputs/Outputs** (File menu) and select **fft_on_chip_walk.scf** as the file to simulate.
3. Choose the **Start** button to start the simulation. Figure 3 shows the waveforms generated from **fft_on_chip_walk.scf**.

Figure 3. Simulator Channel File fft_on_chip_walk.scf



As shown in Figure 3, after `done` goes high, the processed data appears in the 14 µs to 16 µs range on the fifth rising clock edge after the `unload_bank_enable` and `load_nrun` inputs go high. Figure 4 shows the Figure 3 waveforms in greater detail.

Figure 4. Detailed View of fft_on_chip.scf



Timing Analysis

The MAX+PLUS II Timing Analyzer can analyze the timing performance of a project after it has been optimized by the Compiler. To begin a timing analysis of `fft_on_chip_test`, choose **Registered Performance** (Analysis menu) in the Timing Analyzer window and then choose the **Start** button. The Timing Analyzer should show the FFT performing at approximately 60 MHz.

Signal

The `fft_on_chip_test` reference design described in this application note uses a signal of the form: $y(t) = \sin(t) - (1/3)\sin(3t)$. The 16 data points from this signal were plotted in the range $[-\pi, \pi]$, scaled, and then used as input data for the `fft_on_chip _test` simulation. Figure 5 shows the plot of the input signal data.

Figure 5. Input Data Points

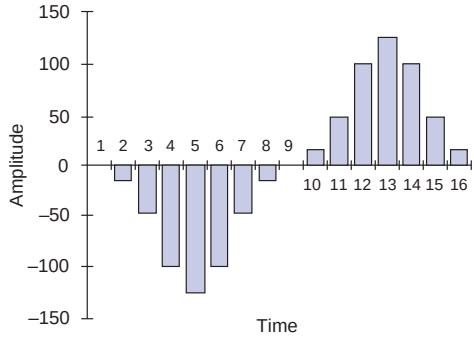


Figure 6 shows the resulting plot of the magnitude vs. frequency response for the sinusoid in Figure 5.

Figure 6. Magnitude Response

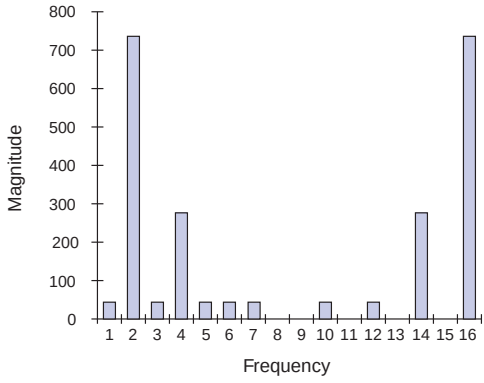
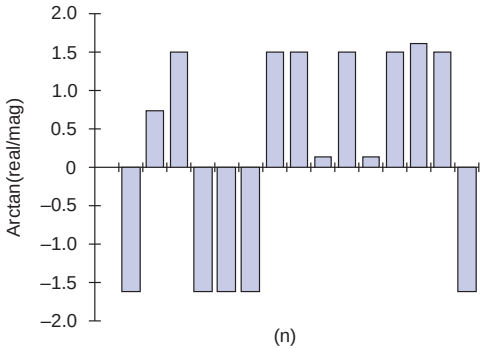


Figure 7 shows the phase response for the signal shown in Figure 5.

Figure 7. Phase Response



Conclusion

The `fft_on_chip` reference design instantiates the `fft` MegaCore function and also takes advantage of the FLEX 10K embedded array to provide on-chip RAM for storing data and twiddle factors. This application note demonstrates how to implement and simulate the `fft_on_chip` reference design.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com
Applications Hotline:
(800) 800-EPLD
Customer Marketing:
(408) 544-7104
Literature Services:
(888) 3-ALTERA
lit_req@altera.com