

## Floating-Point FFT Processor (IEEE 754 Single Precision) Radix 2 Core

### Introduction

The floating-point fast fourier transform (FFT) processor calculates FFTs with IEEE 754 single precision (1 sign bit, 8 exponent bits, and 23 mantissa bits) accuracy. The processor uses extended precision arithmetic (1 sign bit, 8 exponent bits, and 32 mantissa bits) internally for higher accuracy. The floating point FFT processor radix 2 core can implement any powers of 2 length of FFT. It is optimized for the Stratix™ device family and takes advantage of the Stratix DSP blocks and M-RAM blocks.

You can parameterize the number of points at compile time. There are two versions of the core in the package, one version is optimized for internal device memory and the other for memory external to the device. Top-level reference designs, with open source code, allow you to integrate the core into your system.

The core is designed for simulation with the ModelSim simulator (version 5.6a) and synthesis using the Quartus® II software. Testbench generation and analysis utilities (MATLAB-based) are included for verifying the core in both environments.

### Parameters & Ports

Altera® provides two top-level system reference designs: **FFTTOPA2.VHD** and **FFTTOPB2.VHD**. The parameters can be set via the **PARMS.VHD** file. The FFT processor cores are **FFT2A.VHD** and **FFT2B.VHD**, respectively.

Table 1 shows the parameters.

*Table 1. Parameters*

| Parameter      | Description  |
|----------------|--|
| POINTS         | The number of points in the transform, any power of 2, value 16 or higher.   |
| DATAINDELAY    | Set to 0 for FFTOPA2. In FFTOPB2, any value 2 or greater, defines the latency between the core output and the data memory input.             |
| DATAOUTDELAY   | Set to 0 for FFTOPA2. In FFTOPB2, any value 2 or greater, defines the latency between the data memory output and the core input.             |
| TWIDINOUTDELAY | Set to 0 for FFTOPA2. In FFTOPB2, any value 2 or greater, defines the latency between the FFT control output and the FFT twiddle data input. |

Table 2 shows the input signals.

*Table 2. Input Signals*

| Signal                       | Description   |
|------------------------------|---|
| SYSCLK                       | SYSCLK is the system clock. All memory accesses and processing are at the system clock rate.  |
| RESET                        | The RESET input is active high, and prepares the FFT processor for another FFT operation.   |
| GO                           | The GO input enables FFT processing when high. It must be held high during calculation of the FFT.  |
| REALIN[32:1]<br>IMAGIN[32:1] | Real and imaginary data input ports, in IEEE754 single precision floating point format. The sign bit is bit 32, the exponents occupy from bits 24 to 31, and the mantissa the 23 LSBs.    |
| TWREAL[32:1]<br>TWIMAG[32:1] | Real and imaginary twiddle input ports, in IEEE754 single precision floating point format. The sign bit is bit 32, the exponents occupy from bits 24 to 31, and the mantissa the 23 LSBs. |

Table 3 shows the output signals.

*Table 3. Output Signals*

| Signal                         | Description   |
|--------------------------------|---|
| READADD[addwidth:1]            | Data read address output.   |
| WRITEADD[addwidth:1]           | Data write address output.  |
| TWIDADD[addwidth:1]            | Twiddle read address output.  |
| REALOUT[32:1]<br>IMAGOUT[32:1] | Real and imaginary data output ports, in IEEE 754 single precision floating point format. The sign bit is bit 32, the exponents occupy from bits 24 to 31, and the mantissa the 23 least significant bits (LSBs). |
| WRITE                          | This signal is high when valid data exists on the WRITEADD, REALOUT, and IMAGOUT ports.   |
| DONE                           | When DONE is high, the FFT processor has completed processing, and data can be read out of the FFT, after which a new data set can be loaded in.  |

## Architecture

The FFT processor core consists of a processing core for a radix 2 FFT. It does not include memory for data, intermediate storage, or twiddles, or any interfaces required to load and unload data memory. Altera provides two reference designs, **FFTTOPA2.VHD** and **FFTTOPB2.VHD** as examples of how to implement the memory interfaces. A DOS utility, **TWFP1.EXE** is provided to generate memory initialization files (Intel HEX format) for the on-chip ROM containing the twiddle factors.

The data RAM and twiddle ROM memories are synchronous memories, and the data RAM is dual port also (one read port and one write port). For a ROM, an external asynchronous memory can be made to appear synchronous by adding another stage of delay to the input and output of the ROM. For the data RAM, this is more difficult, as the **WE** pulse generally requires a set-up and hold time for both the data and address inputs to the RAM. However, there are several available commercial synchronous dual-port RAMs that can be used.

Any delays added to external memories, whether to make asynchronous memories appear synchronous, or to improve fitting or performance of the system, can be compensated for by the FFT core through the **DATAINDELAY**, **DATAOUTDELAY**, and **TWIDINOUTDELAY** parameters. In the **FFTTOPB2.VHD** reference design, pipelining only occurs between the twiddle ROM and the FFT inputs, which keeps the reference design simple. In an actual design, the pipelining could be split in any ratio before and after the ROM, as long as **TWIDINOUTDELAY** was set to the sum of both latencies.

The additional pipeline delays can also be used to convert the FFT from using dual-port memories to single-port memories. In this case, two separate banks of memories are required.

The FFT uses a decimation in frequency (DIF) algorithm, hence the input data samples are loaded into the memory in natural order, and the transform is stored in bit or digit reversed order. The reference designs include functions to read out the transformed data in natural order.

## Compiling the FFT Processor

This section details the actions to compile the FFT processor.

### *Setting Parameters*

To set the parameters, perform the following steps:

1. Specify the number of points in the **PARMS.VHD** file. If you are using the **FFTTOPA2.VHD** reference design, set **DATAINDELAY**, **DATAOUTDELAY**, and **TWIDINOUTDELAY** to 0.

*or*

If you are using the **FFTTOPB2.VHD** reference design, set the delay parameters to some value greater than 2.

2. Generate the HEX files to initialize the twiddle memories, using the **TWFP1.EXE DOS** utility. Call the utility by typing the following command:

```
TWFP1 points 2 ↵
```

where *points* is the number of points, and the 2 optimizes the twiddle memories for radix 2 operation.

Three files are generated: **WREAL.HEX**, **WIMAG.HEX**, and **WQ.HEX**. The **WREAL.HEX** and **WIMAG.HEX** files are flat files containing every twiddle factor required by a radix 2 FFT, and are used by the **FFTTOPB2.VHD** reference design. The **WQ.HEX** file is an area optimized file that is used by the **FFTTOPA2.VHD** reference design.

### Memory Requirements

The core has the following memory requirements:

- $64 \times \text{POINTS}$  data RAM bits
- $8 \times \text{POINTS}$  twiddle ROM bits

For example, a 1K FFT requires 72-Kbits memory.

### Compiling with ModelSim

The core has been verified with ModelSim 5.6a. The following libraries from the **EDA/SIM\_LIB** directory in the Quartus II software are required: **220PACK.VHD**, **220MODEL.VHD**, and **ALTERA\_MF.VHD** (or **ALTERA\_MF\_93.VHD**). The core can be compiled with VHDL 87 or VHDL 93 language support, except for the testbenches, which require VHDL 93 support.

The hierarchy (from bottom up as required by ModelSim) is:

- **PARMSUB.VHD**
- **PARMS.VHD**
- **LS1.VHD**
- **RS1.VHD**
- **CLZ1.VHD**
- **SELONE.VHD**
- **FPM1.VHD**
- **ALU1.VHD**
- **DFT2.VHD**
- **BFLY2.VHD**
- **CTL2.VHD**
- **FFT2A.VHD** (or **FFT2B.VHD**)
- **TWMEM2.VHD** (or **TWEXT2.VHD**)
- **FFTTOPA2.VHD** (or **FFTTOPB2.VHD**)

The testbench created by the included MATLAB utilities is **TB\_FFTFP.VHD**.

### Synthesizing with the Quartus II Software

The core has been verified with Quartus II integrated synthesis. The core achieves the following example push-button result using Quartus II version 2.2 for **POINTS = 1024** and in a Stratix EPS1S10F484C5:

- 2704 LCs
- 2 DSP blocks
- 72 Kbits memory
- 185 MHz

The hierarchy (from top down as required by the Quartus II software) is:

- **PARMSUB.VHD**
- **PARMS.VHD**
- **FFTTOPA2.VHD** (or **FFTTOPB2.VHD**)
- **TWMEM2.VHD** (or **TWEXT2.VHD**)
- **FFTA2.VHD** (or **FFTB2.VHD**)
- **CTL2.VHD**
- **BFLY2.VHD**
- **DFT2.VHD**
- **ALU1.VHD**
- **FPM1.VHD**
- **SELONE.VHD**
- **CLZ1.VHD**
- **RS1.VHD**
- **LS1.VHD**

## Reference Designs

Altera provides two reference designs, which you may use as given, or you can modify to meet your requirements.

### *FFTTOPA2*

FFTTOPA2 is a reference design that uses on-board memory for both the data RAM and twiddle ROM. It uses an FFT processor, **FFT2A.VHD**, as the core of the design, and the generic Stratix memory, `altsyncram` to implement both the data RAM and twiddle ROM.

FFTTOPA2 includes ports to write data into the FFT system, and ports to read data out of the FFT system.

The actions of the ports may also be seen from the testcase generated by the MATLAB utilities **FFTFPTB1.M** (for ModelSim) or **FFTFPVEC1.M** (for Quartus II). For more information, see “Testing” on page 5.

Table 4 shows the FFTTOPA2 input signals.

*Table 4. FFTTOPA2 Input Signals*

| <b>Signal</b> | <b>Description</b>  |
|---------------|---|
| SYSCLK        | Main system clock.  |
| RESET         | Resets FFT processor, active high.  |
| GO            | Enables FFT processor, active high.   |
| LOAD          | When high, enables write into data memory. When low, data and address inputs are ignored. |
| UNLOAD        | When high, enables read out of data memory. When load, address inputs ignored.            |
| LOADADD       | Address used for writing data into data memory bank.                                      |
| UNLOADADD     | Address used for reading data out of data memory bank.                                    |
| REALSIGNIN    | Sign bit for real data input.   |
| REALEXPIN     | Exponent for real data input.   |
| REALMANIN     | Mantissa for real data input.   |
| IMAGSIGNIN    | Sign bit for imaginary data input.  |
| IMAGEXPIN     | Exponent for imaginary data input.  |
| IMAGMANIN     | Mantissa for imaginary data input.  |

Table 5 shows the FFTOPA2 output signals.

Table 5. FFTOPA2 Output Signals

| Signal      | Description  |
|-------------|--|
| REALSIGNOUT | Sign bit for real data output.   |
| REALEXPOUT  | Exponent for real data output.   |
| REALMANOUT  | Mantissa for real data output.   |
| IMAGSIGNOUT | Sign bit for imaginary data output.                                      |
| IMAGEXPOUT  | Exponent for imaginary data output.                                      |
| IMAGMANOUT  | Mantissa for imaginary data output.                                      |
| DONE        | When high, FFT processing is complete, data may be read out of the core. |

### FFTOPB2

FFTOPB2 has an almost identical structure and ports to FFTOPA2. FFTOPB2 is designed to work with memory outside the device.

The only difference in the system design between the two examples is the additional level of pipelining in and out of all of the memory components in FFTOPB2. This additional pipelining allows better timing between the FFT processor inside the Altera device, and the memory components external to the device. The FFT processor automatically compensates for the additional latency, which is controlled using the DATAINDELAY, DATAOUTDELAY, and TWID-INOUTDELAY parameters.

For more information on these parameters see “Parameters & Ports” on page 1.

## Testing

This section details the actions to test the FFT processor.

### Example Test Flow

Note: the following example does not assume any directory structure. You may have to copy files between directories, i.e., the MATLAB **work** or mapped directories, the ModelSim **work** or mapped directories, and the Quartus II **work** or mapped directories.

There are four MATLAB utilities for testing the core: two for use with the ModelSim simulator, and two for use with the Quartus II software.

### ModelSim Testing

**FFTFPTB1.M** is a MATLAB utility that creates a VHDL testbench for both reference designs. The format of the utility call is:

```
FFTFPTB1 (input_vector, radix, datadelay) ←
```

where `datadelay` parameter is the sum of `DATAINDELAY` and `DATAOUTDELAY`.

To create an input vector in MATLAB, with completely random data, perform the following example (for a 256-point FFT):

1. Enter the following values at the MATLAB command prompt:

```
rr = rand(1,256); % create a random vector
ss = rand(1,256);
vec =rr + i * ss;
```

2. Call the MATLAB utility, by typing the following command:

```
FFTFPTB1 (vec, 2, 0) ←
```

Using a `datadelay` of 0 instantiates the FFTTOPA2 design into the testbench.

3. Before loading the design into ModelSim, generate the twiddle files using **TWFP1.EXE**. To generate twiddle files, perform the following example (for a 256-point FFT).

```
TWFP1 256 2 ←
```

4. To simulate, compile the generated test bench, **TB\_FFTFP.VHD**, in ModelSim, and load the design. You do not have to inspect the WAVE, the testbench writes out a results file, **FFTFP.TXT**. Another MATLAB utility, **FFTFPCHK1.M**, extracts the contents of the testbench generated output. To run **FFTFPCHK1.M**, type the following command:

```
[real result, imaginary result] = FFTFPCHK1(input vector) ←
```

For this example:

```
[x, y] = FFTFPCHK1(vec);
```

**FFTFPCHK1.M** automatically plots the differences between the simulation results and the expected results (MATLAB FFT). The differences can also be manually examined in more detail.

### *Quartus II Testing*

The utilities included for testing the core in the Quartus II environment support both functional and timing simulations—the procedure is identical for both. The following example is for the FFTTOPA2 design (`datadelay = 0`):

1. First, create a complex vector (example shown in Modelsim Testing section, above).
2. Create a vector file by typing the following command:

```
FFTFPVEC1 (vec, 2, 0)
```

3. Create the twiddle factors by typing the following command:

```
TWFP1 256 2
```

4. In the Quartus II software, select the mode (functional or timing), stimulus (**FFTFP.VEC**), and run the simulation.
5. When the simulation is complete, write a **.tbl** file from the waveform window. Call this file **FFTFP.TBL**.
6. Extract the results, by typing the following command:

```
y = FFTFPTBL1 (vec, 0);
```

The differences between the simulated and expected results automatically plot, but you can also manually examine them.

### List of Utilities & Files

The FFT processor core includes the following utilities and files:

- **TWFP1.EXE**—utility run from a DOS window that generates twiddle factors
  - Input parameters: points, radix
  - Output files: **WREAL.HEX**, **WIMAG.HEX**, **WQ.HEX**
- **FFTFPTB1.M**—MATLAB M file that generates the testbench for ModelSim
  - Inputs: complex data vector, radix, datadelay
  - Output file: **TB\_FFTFP.VHD**
- **FFTFCHK1.M**—MATLAB M file that compares ModelSim results with the expected results
  - Inputs: complex data vector
  - Input Files: **FFTFP.TXT**
  - Outputs: real and imaginary components of simulation results
- **FFTFPVEC1.M**—MATLAB file that creates a vector file for the Quartus II simulation
  - Inputs: complex data vector, radix, datadelay
  - Output file: **FFTFP.VEC**
- **FFTFPTBL1.M**—MATLAB file that compares the Quartus II results with the expected results
  - Inputs: complex data vector, datadelay
  - Input file: **FFTFP.TBL**
  - Outputs: complex vector of simulation results

### Performance

The performance of the FFT is dependant on two factors: the clock rate of the FFT system, and the number of clocks required to calculate the FFT.

The number of clocks required is given by:

*Number of passes required* × *number of clocks per pass*

where:

*number of passes* =  $\log_2(\text{points})$

*number of clocks per pass* =  $\text{points} + 32 + \text{datadelay}$

For example, if  $\text{points} = 1,024$  and  $\text{datadelay} = 7$

*Number of passes* = 10

*Clocks per pass* = 1,063

Total Clocks = 10,630

With a 185 MHz system clock, the FFT requires  $10,630/185 = 57 \mu\text{s}$  to compute the FFT.



101 Innovation Drive  
San Jose, CA 95134  
(408) 544-7000  
www.altera.com

Copyright © 2003 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.