

Introduction

The DFT reference design performs a discrete Fourier transform (DFT) or an inverse DFT (IDFT) of a complex input sequence and produces a complex output sequence.

The reference design performs the functions for either a DFT in the uplink or an IDFT in the downlink of a typical 3G long-term evolution (LTE) physical interface (PHY) implementation

The design inputs the transform length coincident with the first validated data sample. The design can configure the transform length at runtime (on a block-by-block basis) to any one of the 34 sizes specified by the proposal in *3GPP Technical Document R1-062852*, submitted to *TSG-RAN WG1 #46bis, Seoul Korea*.

The reference design uses Avalon® Streaming (Avalon-ST) interfaces for the inputs and the outputs. The input samples are in an integer format; the output samples are in block floating point format.

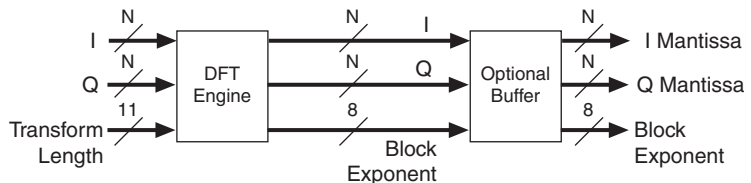
Parameters specify the transform mode (DFT or IDFT) and internal bit widths for the data-path and the twiddle value precision.

The reference design is targeted at Altera® Stratix® II or Stratix III devices and meets typical latency requirements while minimizing power consumption.

Functional Description

Figure 1 shows the reference design block diagram, where N is the datawidth.

Figure 1. Block Diagram



The input is an Avalon-ST sink interface with a ready latency of 1. The design does not use the `eop` signal at the input as it expects packets of exactly `length`. The transform commences within the design when all `length` samples have been input.

You can configure the output Avalon-ST interface to come from the DFT engine from an optional buffer. The buffer depth is user selectable. With the buffer there is a ready signal input, which prevents data streaming out as soon as the DFT engine finishes a transform. When the DFT engine finishes a transform is determined by: when data is input to the design, and the transform time for the current length.

Table 1 shows the 34 transform sizes defined for 3GPP LTE and the corresponding latencies.

<i>Table 1. Transform Times and Latencies (Part 1 of 2)</i>		
Transform Size	Calculation Time (Cycles)	Total Latency (Cycles)
12	24	36
24	72	96
36	108	144
48	144	192
60	180	240
72	288	360
96	384	480
108	432	540
120	480	600
144	576	720
180	720	900
192	768	960
216	1,080	1,296
240	960	1,200
288	1,440	1,728
300	1,200	1,500
324	1,620	1,944
360	1,800	2,160
384	1,920	2,304
432	2,160	2,592
480	2,400	2,880

Table 1. Transform Times and Latencies (Part 2 of 2)

Transform Size	Calculation Time (Cycles)	Total Latency (Cycles)
540	2,700	3,240
576	2,880	3,456
600	3,000	3,600
648	3,888	4,536
720	3,600	4,320
768	3,840	4,608
864	5,184	6,048
900	4,500	5,400
960	4,800	5,760
972	5,832	6,804
1080	6,480	7,560
1152	6,912	8,064
1200	6,000	7,200

If the output buffer is enabled, the ready input controls the data flow out of the buffer. The buffer always fills in bursts as the DFT engine completes individual transforms so you must ensure the buffer depth is adequate for your system to avoid overflow. The design behaves unpredictably if the buffer overflows.

The design provides the `sop` and `eop` signals at the output, which you can optionally use.



For more information on the Avalon-ST interface, refer to the [Avalon Streaming Interface Specification](#).

To perform an IDFT, set the `idft_mode` parameter to 1.



The design does not perform the $1/\text{length}$ scaling for an IDFT.

In the MATLAB comparison script (`compare_rtl_vs_matlab_results.m`) within the regression test, this scaling is applied (in IDFT mode only) to the RTL results before comparing to the MATLAB reference.

Table 2 shows the reference design parameters.

Name	Values	Description
Idft_mode	0 or 1	0 = DFT, 1 = inverse DFT.
datawidth	16 .. 24	Internal datapath and input widths in bits.
twidwidth	14 .. 24	Twiddle factor and butterfly weight coefficient widths in bits (internal to the design).
use_output_buffer	0 or 1	0 = no output buffer is instantiated and the output interface does not have a ready input (source_ready_in is inactive). 1 = an output buffer is instantiated, which allows the reading of data from the design to be paused. The ready latency is 0.
BUFFER_DEPTH	4 .. 2048	If use_output_buffer = 1, this parameter defines the buffer depth. Each buffer location corresponds to one pair of complex (I and Q) samples.
BUFFER_DEPTH_N	2 .. 11	Set so that $2^{\text{BUFFER_DEPTH_N}} \geq \text{BUFFER_DEPTH}$ For example, if BUFFER_DEPTH = 600, set BUFFER_DEPTH_N to 10.

Table 2 shows the reference design signals...

Name	Direction	Description
a_reset_n	Input	Active low asynchronous reset. Deassert synchronously to clk to avoid reset-release timing violations.
clk	Input	System clock. All logic in the design is synchronous to this clock.
exponent[7:0]	Output	Avalon-ST source data. Exponent of all samples in the packet. A single value is valid throughout the packet on this output.
sink_eop	Input	Avalon-ST sink end of packet. The design does not use this signal.
sink_imag[datawidth-1:0]	Input	Avalon-ST sink data. Imaginary (Q) input sample.
sink_length[10:0]	Input	Transform length. A straight binary encoding of one of the lengths in Table 1. The value on this input is sampled when sink_valid is high and sink_ready was high on the previous cycle.

Table 3. Signals (Part 2 of 2)

Name	Direction	Description
sink_ready	Output	Avalon-ST Ready. The input interface has a ready latency of 1.
sink_real[datawidth-1:0]	Input	Avalon-ST sink data. Real (I) input sample.
sink_sop	Input	Avalon-ST sink start of packet.
sink_valid	Input	Avalon-ST sink valid.
source_eop	Output	Avalon-ST source end of packet.
source_imag[datawidth-1:0]	Output	Avalon-ST source data. Mantissa of Imaginary (Q) output sample.
source_real[datawidth-1:0]	Output	Avalon-ST source data. Mantissa of Real (I) output sample.
source_ready_in	Input	Avalon-ST source ready. This signal is only enabled if the parameter <code>use_output_buffer</code> is set to 1. The ready latency is 0.
source_sop	Output	Avalon-ST source start of packet.
source_valid	Output	Avalon-ST source valid.

Fixed-Point Model

Altera provides a fixed-point MATLAB model to allow fast bit-accurate simulation of the design's arithmetic behavior. The model is as a matlab function `dft_model()` and is defined in the file `dft_model.m`. This top-level file and the sub-function MATLAB files are in the `/model` directory. Altera provides a MATLAB example that calls the model in file `simple_dft_example.m`. To run the example, follow these steps:

1. Open the MATLAB software.
2. Change the directory to `/model`.
3. Type the following command:

```
simple_dft_example
```
4. Open the results file `<DFT type>_length<length>_data<D_top>bit_tw<T_top>bit_rand<seed>.txt` and examine the data.

The `simple_dft_example.m` file defines the parameters.

From left to right there are the following pairs of columns:

1. Input data.

2. Model block Floating Point output.
3. Model integer output.
4. MATLAB output.
5. Absolute Difference between the Model and MATLAB outputs
6. Percentage difference between the Model and MATLAB outputs.

In each pair of columns the left hand column is the real data and the right hand column is the imaginary data. All data is in normal order with time 0 and bin 0 on the first row.


 The model and the reference design do not perform 1/length scaling when in IDFT mode, so the 1/length scaling is removed from the MATLAB results by multiplying all samples by the length.

Table 4 shows the model's output parameters.

Table 4. Model Output Parameters		
Variable Name	Type	Description
<code>vec_out</code>	Array of complex integers	Complex output array in normal order (provided <code>disable_reordering = 0</code>) Array length is the same as input array 'vec'
<code>blk_exponent</code>	Integer (-127 to +127)	Exponent value common to all values in output array <code>vec_out</code> . If you use the model for system modelling without reference to the RTL, set <code>invert_sign_of_exponent</code> to 0 so that the output sample values are: $vec_out \times 2^{blk_exponent}$

Table 5 shows the model's input parameters.

Table 5. Model Input Parameters			
Variable Name	Type	Default Value <i>Note (1)</i>	Description
vec	Array of complex integers	–	Input array to be transformed. The model performs a transform of length equal to the input array length. The array length must be one of the 34 lengths (see Table 1).
model_is_fixed_point	0 or 1	1	0 = double precision model; 1 = fixed point model.
idft_mode	0 or 1	–	0 = DFT; 1 = IDFT.
B_top	Integer 14 to 24	T_top (3)	Butterfly coefficient precision in bits.
D_top	Integer 14 to 24	–	RAM data path width in bits. This value is also the input data width and the mantissa width of the output samples.
T_top	Integer 14 to 24	–	Twiddle coefficient precision in bits.
bfly_mult_convergent_mode	0 or 1	1	Rounding mode at outputs of butterfly complex multipliers: 0 = symmetrical round-up 1 = convergent rounding (2)
tw_mult_convergent_mode	0 or 1	1	Rounding mode at outputs of twiddle complex multipliers: 0 = symmetrical round up 1 = convergent rounding (2)
limit_bfp_mux_size	0 or 1	1	0 = BFP scaling is done to the full precision of width D_top width. 1 = BFP scaling is limited to max_num_of_bfp_msb_shift_bits bits down from the MSB. (2)
max_num_of_bfp_msb_shift_bits	2 to (D_top – 1)	3	BFP scaler size. (2)
skip_bfp_on_last_pass	0 or 1	1	0 = BFP scaling is performed on all reads from data RAM; 1 = BFP scaling is omitted for the last read from data RAM. (2)

Table 5. Model Input Parameters

Variable Name	Type	Default Value <i>Note (1)</i>	Description
invert_sign_of_exponent	0 or 1	1 (Note4)	0 = output sample values are given by: $vec_out \times 2^{blk_exponent}$ 1 = output sample values are given by: $vec_out \times 2^{-blk_exponent}$ When set to 1 the behavior matches the RTL. If you use the model for system modelling without reference to the RTL, set <code>invert_sign_of_exponent</code> to 0 so that the output sample values are: $vec_out \times 2^{blk_exponent}$ <i>(2)</i>
disable_reordering	0 or 1	0	0 = output vector is in normal order; 1 = the output samples are not re-ordered.
debug_dump	0 or 1	0	1 = write out debug data file.
dump_order	0 or 1	1	0 = data appears in the debug dump file in linear data ram address order. 1 = data appears in the debug dump file in the same order that it is generated in the RTL so you can compare the internal RTL behavior.
dump_id	Positive integer	–	Tags the dump file for post-run debugging of regression tests.
debug_on_error	0 or 1	0	1 = break simulation and display additional debug data to console when datapath value specified by <code>debug_on_error_value</code> is detected. <i>(2)</i>
debug_on_error_value	Complex integer	–	Trigger value to be detected at RAM input for additional debug output.

Note to Table 5:

- (1) Typical settings to use when running the model or default value that you should use so that the model matches the RTL behavior.
- (2) These parameters are only valid for `fixed_point_model = 1`.
- (3) In the RTL, the value of `B_top` is always equal to the value set for `T_top`.

Getting Started

This section describes the following topics:

- [System Requirements](#)

- Install the Reference Design
- Simulate the Reference Design
- Synthesize the Reference Design

System Requirements

The reference design requires the following hardware and software.

- MATLAB version R2006B
- Quartus® II software version 7.1

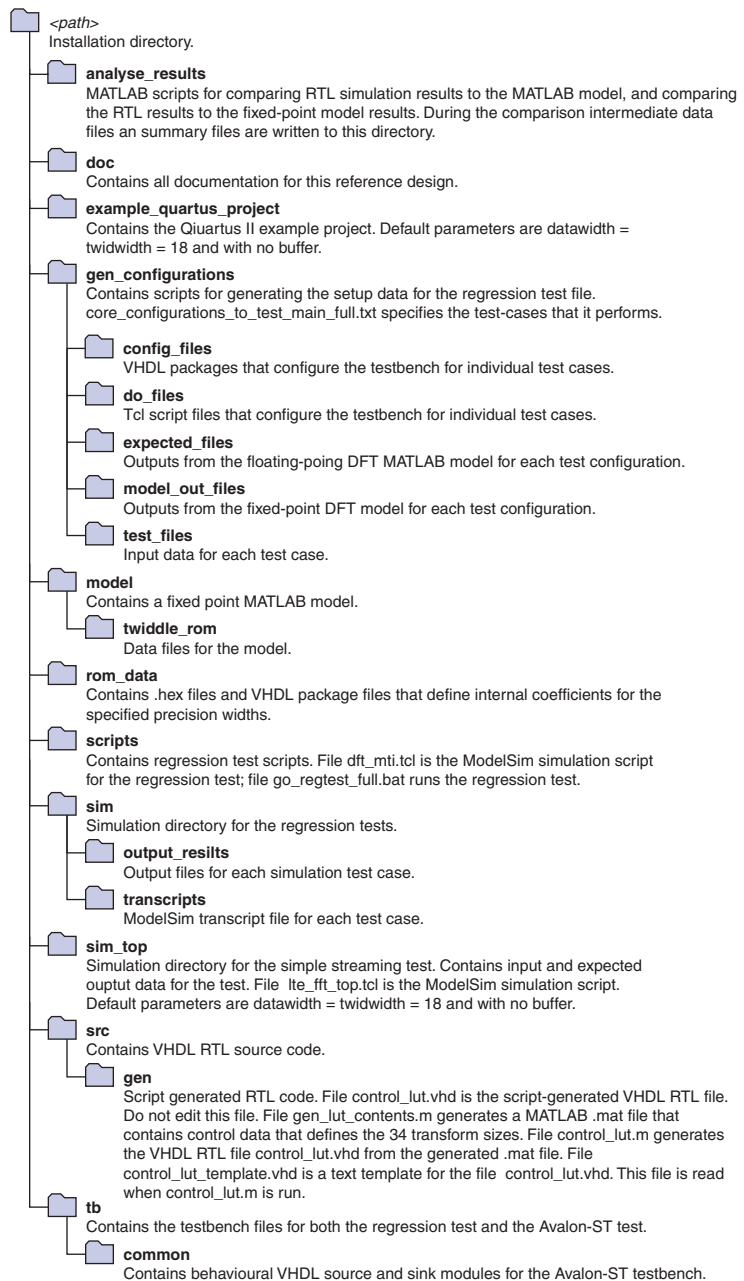
Install the Reference Design

To install the reference design, run the **an464.zip** file.



You may need to add the **/model** directory to the MATLAB path.

Figure 2 shows the directory structure after installation.

Figure 2. Reference Design Directory Structure

Simulate the Reference Design

The reference design includes two simulation environments.

The Avalon-ST testbench and script uses a single fixed set of parameters and runs a sequence of transforms with the transform length changing. This testbench quickly demonstrates the integration of the reference design and the Avalon-ST interfaces.

The command-line regression test runs different combinations of parameters with different transform lengths. The transform results are compared to the output of a MATLAB fixed-point model fed with the same input data. This regression test takes much longer to run.

To run the Avalon-ST testbench, follow these steps:

1. Open the ModelSim simulator.
2. Change the directory to `/sim_top`.
3. Type the following command:

```
source ltefft_top.tcl
```



Ignore the address out of range warnings.

4. Compare the output file `output_data.txt` with the expected output `expected_output_data_random_16_blocks.txt`, to ensure they match except for the block length comments in the expected file.



The test input file is `input_stream_random_17_blocks.txt`. The sequence of sizes are listed in the `readme_random_17_blocks.txt` file in the `\sim_top` directory.

To run the command-line parameter regression test, which only exercises the kernel and does not use the Avalon-ST interfaces, follow these steps:

1. Open a command prompt.
2. Type the following commands, to run the 68 tests defined in file `gen_configurations/core_configurations_to_test_main_full.txt`:

```
cd <install path>/dft/scripts
```

```
go_regtest_full
```

3. Wait until `*** All Done. ***` is displayed.

To compare the RTL simulation results to the MATLAB results, follow these steps:

1. Open MATLAB.
2. Change the MATLAB directory to `<install path>/dft/analyse_results`.
3. Type the following command, to compare each of the 68 RTL simulation results to the expected file produced by MATLAB's FFT or IFFT functions, as appropriate, using double precision floating point:

```
>> compare_rtl_vs_matlab_results
```

The results display as an RMS error with all errors >10% taken as exceptions and excluded from the result. You can see the exceptions in files `exceptions_<id>.txt`.

A summary of the results is in the file `summary.txt`.



The first 34 tests are the different DFT sizes in descending order from 1,200 down to 12.

To compare the RTL simulation results to the fixed-point model results, follow these steps:

1. Open MATLAB.
2. Change the MATLAB directory to `<install path>/dft/analyse_results`.
3. Type the following command:

```
>> compare_rtl_vs_model_results
```

Each test case gains a pass or fail and a summary displays at the end. The `rtl_vs_model_summary.txt` file includes a transcript.

Synthesize the Reference Design

Altera provides an example Quartus II project in the directory `/example_quartus_project`.

To synthesize the reference design, follow these steps:

1. Open file `/src/new/ltefft_top.vhd` and edit the following top-level parameters as desired.



These default setting have no effect on the simulation runs.

```
ENTITY ltefft_top IS
  GENERIC (
    idft_mode : integer := 0;
    datawidth : positive := 18;
    twidwidth : positive := 18;
    use_output_buffer : boolean := false;
  );
END ENTITY ltefft_top;
```

2. Copy the two files `/rom_data/bits_<twidwidth>/rom_slots0to3.hex` and `rom_slots4.hex` to the `/example_quartus_proj` directory.
3. Copy the file `butterfly_coef_pkg.vhd` from the `dft/rom_data/bfly_coef_pkgs/bits_<twidwidth>` directory.



Skip this step if you are not changing the default parameters.

4. Open the Quartus II software.
5. Open the project in the `/example_quartus_proj/` directory.
6. Change the family or device as appropriate (Stratix II or Stratix III device).
7. Compile the design.

Performance

Table 6 shows the performance of the design with Stratix III devices, for the following parameters:

- `twidwidth = 18`
- `datawidth = 18`,
- `no buffer`

Table 6. Performance			
18-Bit DSP Elements	Memory (M9K)	Combinational ALUTs	f_{MAX} (MHz)
32	17	2,600	260

Document Revision History

Table 7 shows the revision history for this application note.

<i>Table 7. Document Revision History</i>		
Date and Document Version	Changes Made	Summary of Changes
June 2007 v1.0	First release.	—



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support/
Literature Services:
literature@altera.com

Copyright © 2007 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.

