A Scalable OFDMA Engine for WiMAX

May 2007, Version 2.1 Application Note 412

Introduction

The Altera® scalable orthogonal frequency-division multiple access (OFDMA) engine for mobile worldwide interoperability for microwave access (WiMAX) can be used to accelerate the development of mobile broadband wireless networks based on the *IEEE 802.16* standard.

For more information on *IEEE 802.16*, refer to the *IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE P802.16-REVd/D5-2004, May 2004.

Scalable OFDMA is a key technology behind mobile WiMAX and is widely regarded as an enabling technology for future broadband wireless protocols including the 3GPP and 3GPP2 long term evolution standards.

This reference design demonstrates the suitability of Cyclone® II, Cyclone III, Stratix® II, and Stratix III, FPGAs for implementing advanced OFDMA symbol-level processing algorithms.

Key Features of the Reference Design

The scalable OFDMA engine has the following key features:

- Support for 128, 512, 1K, and 2K FFT sizes to address variable bandwidths from 1.25 to 20 MHz
- Support for both downlink partial usage of subchannels (PUSC) and full usage of subchannels (FUSC) and uplink PUSC mandatory schemes
- Support for both fixed and variable pilots and runtime configurable cyclic prefix insertion
- Parameterizable design
- Optimized for efficient use of Cyclone II, Cyclone III, Stratix II, and Stratix III device resources

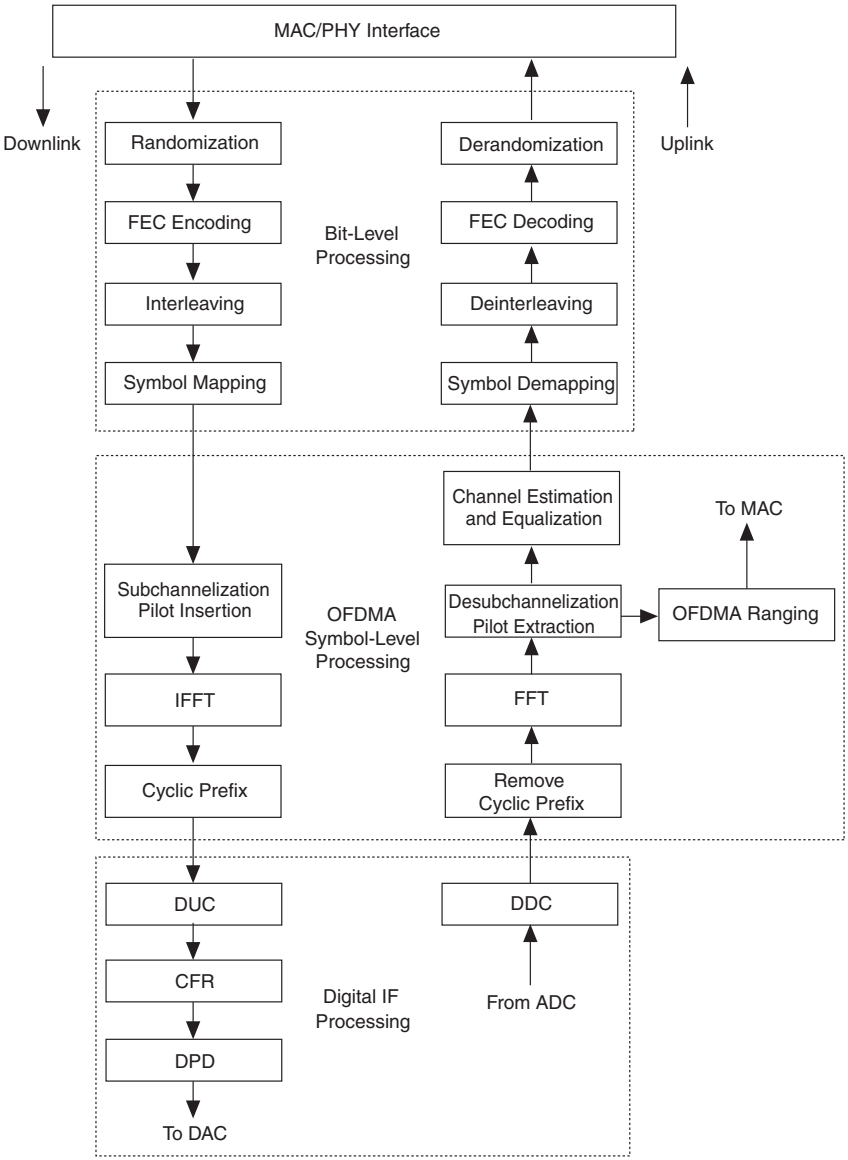
You can use the reference design as a starting point to accelerate designs based on WiMAX or 3GPP long term evolution protocol. Altera supplies the reference design as clear-text register transfer level (RTL) HDL.

Please contact your local **Altera sales representative** for a copy of the reference design.

WiMAX Physical Layer

Figure 1 shows an overview of the *IEEE 802.16e-2005* scalable orthogonal frequency-division multiple access (OFDMA) physical layer (PHY) for WiMAX basestations.

Figure 1. WiMAX Physical Layer Implementation



Altera's WiMAX building blocks include bit-level, OFDMA symbol-level, and digital intermediate frequency (IF) processing blocks. For bit-level processing, Altera provides symbol mapping/demapping reference designs and support for forward error correction (FEC) using the Reed-Solomon and Viterbi MegaCore® functions.

The OFDMA symbol-level processing blocks include reference designs that demonstrate subchannelization and desubchannelization with cyclic prefix insertion supported by the fast Fourier transform (FFT), and inverse fast Fourier transform (IFFT) MegaCore functions. This reference design consists of these symbol level modules integrated together to form the scalable OFDMA Engine. Other OFDMA symbol-level reference designs illustrate ranging, channel estimation, and channel equalization.

The digital IF processing blocks include single antenna and multi-antenna digital up converter (DUC) and digital down converter (DDC) reference designs, and advanced crest-factor reduction (CFR) and digital predistortion (DPD).

This application note illustrates how the integration was achieved for both the downlink and uplink scalable OFDMA engine.

For detailed information on the modules in isolation, please refer to the following application notes:

- *AN-450 Uplink Desubchannelization for WiMAX*
- *AN-451 Downlink Subchannelization for WiMAX*
- *AN-452 An OFDM FFT Kernel for WiMAX*

For more information on related Altera WiMAX solutions, refer to the following application notes:

- *AN 421: Accelerating DUC & DDC System Designs for WiMAX*
- *AN 430: WiMAX OFDMA Ranging*
- *AN 434: Channel Estimation & Equalization for WiMAX*
- *AN 439 Constellation Mapper and Demapper for WiMAX*

OFDMA Modulation

The physical layer is based around OFDMA modulation. Data is mapped in the frequency domain onto the available carriers. For this data to be conveyed across a radio channel, it is transformed into the time domain using an inverse fast Fourier transform (IFFT) operation. To provide multipath immunity and tolerance for synchronization errors, a cyclic prefix is added to the time domain representation of the data.

Multiple OFDMA modulation modes are supported to accommodate variable channel bandwidths. This scalable architecture is achieved by using different FFT/IFFT sizes. Table 2 shows the supported channel bandwidths. This reference design supports all of these modes.

Table 1. Supported Channel Bandwidths

Channel Bandwidth (MHz)	FFT Size
1.25	128
5	512
10	1,024
20	2,048

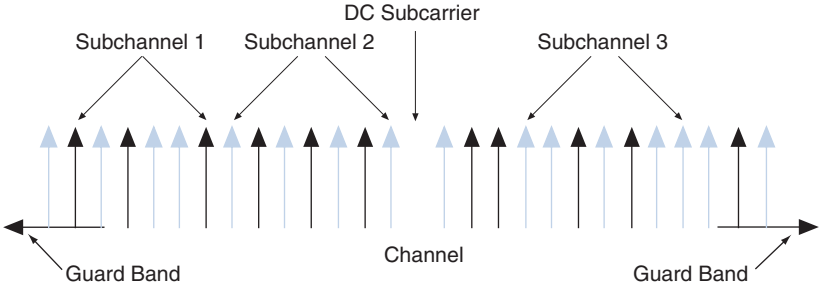
Subchannelization

An OFDMA symbol consists of a number of carriers equal to the size of the Fourier transform. The OFDMA symbols are constructed from data, pilot, and null carriers:

- Data carriers—for data transmission
- Pilot carriers—the magnitude and phase of these carriers are known to the receiver and they are used for channel estimation
- Null carriers—there is no transmitted energy on these carriers to enable the signal to naturally decay and prevent leakage of energy into adjacent channels

To support multiple access, the data subcarriers are divided into groups that make up subchannels. The subcarriers that make up a subchannel are distributed across all of the available carriers. Particular users are allocated a number of different subchannels to send and receive data (see Figure 2).

Figure 2. OFDMA Frequency Description



The subchannelization and desubchannelization modules map and demap the raw constellation data to particular subcarriers within subchannels. A permutation formula maps the subchannels to physical subcarriers in the OFDMA symbol. The formula varies for the uplink and downlink and for the FUSC and PUSC modes.

The data and pilot subcarrier indexes are generated differently for the FUSC and PUSC modes:

- Downlink FUSC:
  - Fixed and Variable pilot tones are added for each OFDMA symbol independently
  - Remaining subcarriers are divided into subchannels that are used exclusively for data

- Downlink PUSC and uplink PUSC:
  - The set of used subcarriers is partitioned into subchannels
  - Pilot subcarriers are allocated from within each subchannel

In FUSC, there is one set of common pilot subcarriers; in PUSC, each subchannel contains its own set of pilot subcarriers.

Users are allocated slots for data transfer and these slots represent the smallest possible data unit. A slot is defined by a time and subchannel dimension and it varies depending on the following operating modes:

- For downlink FUSC, one slot is a single subchannel by one OFDMA symbol
- For downlink PUSC, one slot is a single subchannel by two OFDMA symbols.
- For uplink PUSC, one slot is a single subchannel by three OFDMA symbols.

A single packet of user data is distributed over multiple OFDMA symbols for the PUSC modes. For more information on distributed subcarrier permutations, refer to the *IEEE 802.16-2004* base specification and the *IEEE 802.16e-2005* amendment sections 8.4.6.1.2.2, 8.4.6.1.2.2.2, 8.4.6.1.2.1 and 8.4.6.2.1. Table 2 shows the number of subchannels for each mode.

Table 2. Number of Subchannels for Each Mode

Mode	FFT 128	FFT 512	FFT 1,024	FFT 2,048
Downlink FUSC	2	8	16	32
Downlink PUSC	3	15	30	60
Uplink PUSC	4	17	35	70

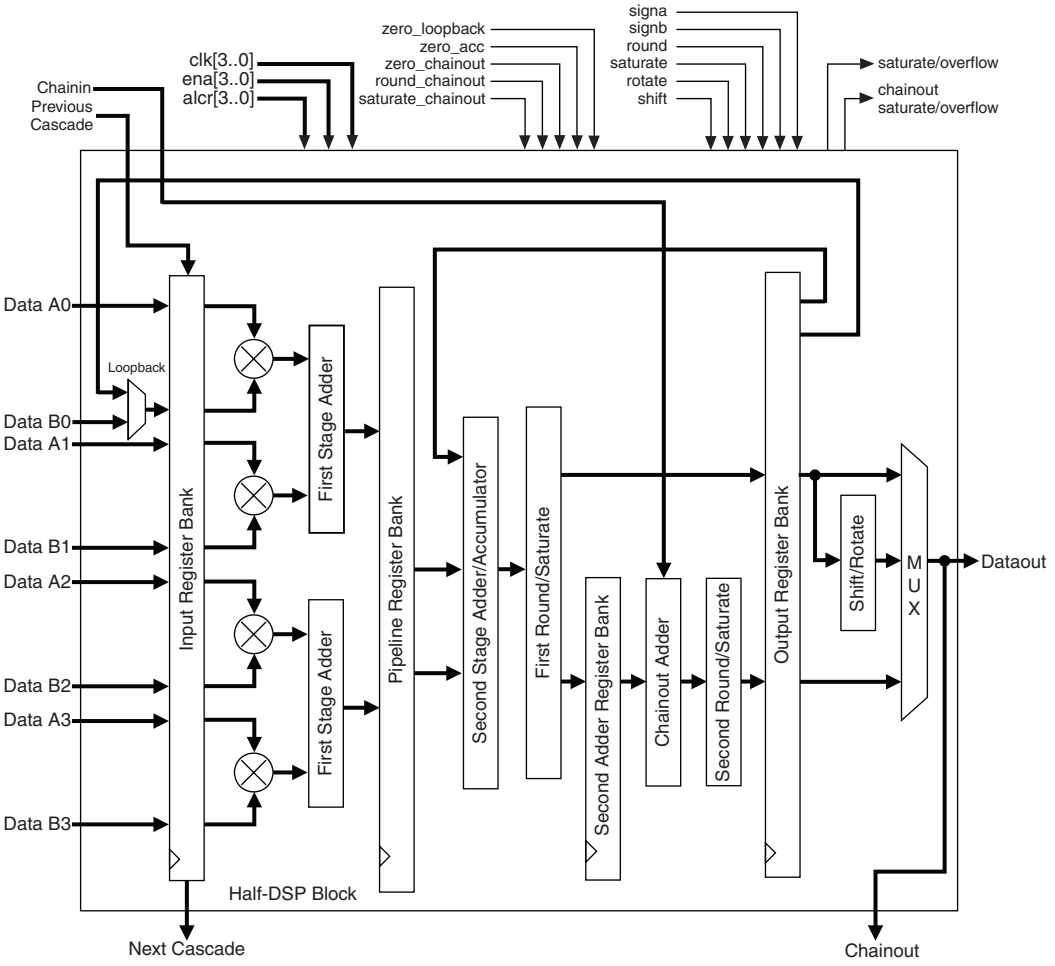
Implementation

FPGAs are well suited to FFT and IFFT processing because they are capable of high speed complex multiplications.

DSP devices typically have up to eight dedicated multipliers, whereas the Stratix III EP3SE110 FPGA has 112 DSP blocks that offer a throughput of nearly 500 GMACs and can support up to 896 18x18 multipliers, which is an order of magnitude higher than current DSP devices.

Figure 3 shows the embedded digital signal processing (DSP) blocks in an Altera Stratix III device.

Figure 3. Embedded DSP Blocks Architecture in Stratix III Devices



Such a massive difference in signal processing capability between FPGAs and DSP devices is further accentuated when dealing with basestations that employ advanced, multiple antenna techniques such as space time codes (STC), beam forming, and multiple-input multiple-output (MIMO) schemes.

The combination of OFDMA and MIMO is widely regarded as a key enabler of higher data rates in current and future WiMAX and 3GPP long term evolution (LTE) wireless systems. When multiple transmit and receive antennas are employed at a basestation, the OFDMA symbol processing functions have to be implemented for each antenna stream separately before MIMO decoding is performed.

The symbol-level complexity grows linearly with the number of antennas implemented on DSPs that perform serial operations. For example, for two transmit and two receive antennas the FFT and IFFT functions for WiMAX take up approximately 60% of a 1-GHz DSP core when the transform size is 2,048 points.

In contrast, a multiple antenna-based implementation scales very efficiently when implemented with FPGAs. Using Altera devices, you can exploit parallel processing and time-multiplexing between the data from multiple antennas. The same 2×2 antenna FFT/IFFT configuration uses less than 10% of a Stratix II 2S60 device.

Design Methodology

Altera provides the reference designs as clear-text VHDL. The modules are integrated together using some custom glue logic modules and top level wrappers.

To accelerate integration of the modules, they have all been designed so that their interfaces support the Altera Avalon® Streaming (Avalon-ST) interface specification.

For more information, refer to the *Avalon Streaming Interface Specification*.

Altera has verified the RTL behavior against a fixed point MATLAB model of the algorithms. This reference design includes synthesizable RTL with testbenches.

Functional Description

The Scalable OFDMA Engine consists of two parts:

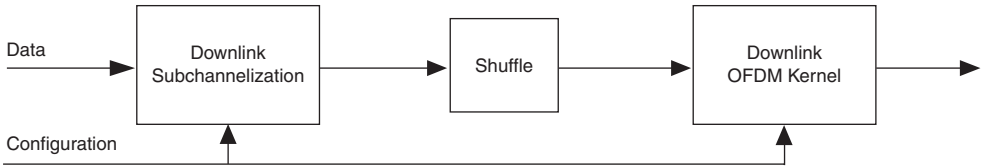
- Transmit and receive orthogonal frequency division multiplexing (OFDM) kernel
- Transmit and receive subchannelization

Downlink OFDMA Engine

The downlink OFDMA engine consists of the downlink subchannelization module and the downlink OFDM kernel.

Figure 4 shows the block diagram of the integrated module and the glue logic that ensures compatibility between the two modules.

Figure 4. Downlink OFDMA Engine



For more information on the individual functionality of the modules, refer to the appropriate application note.

Shuffle Block

Although the OFDM kernel and subchannelization blocks support the Avalon-ST interface specification, the ordering of the packet at the output of the downlink subchannelization block differs to the packet that is expected by the downlink OFDM Kernel. Hence it was necessary to design a small adapter that applies the necessary time slot arrangement.

The FFT MegaCore function in the OFDM kernel accepts packets of data that are in the format [0..*N*-1] where *N* is the FFT size. The downlink subchannelization block outputs OFDMA symbols that are defined from $[-N/2 ... N/2-1]$ and it is necessary to condition the output data so that it is in a suitable format for the downlink kernel. The two packet formats are shown in Figure 5 and Figure 6. In addition, this block generates Avalon-ST control signals.

Figure 5. Subchannelization Output Packet Format

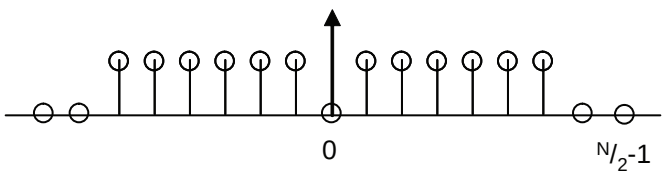
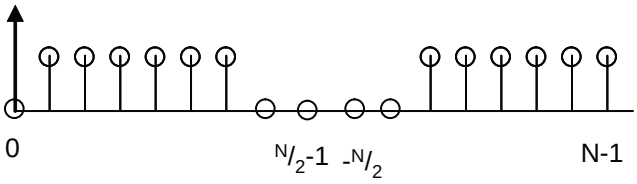


Figure 6. Figure 24: Required OFDMA Input Packet Format

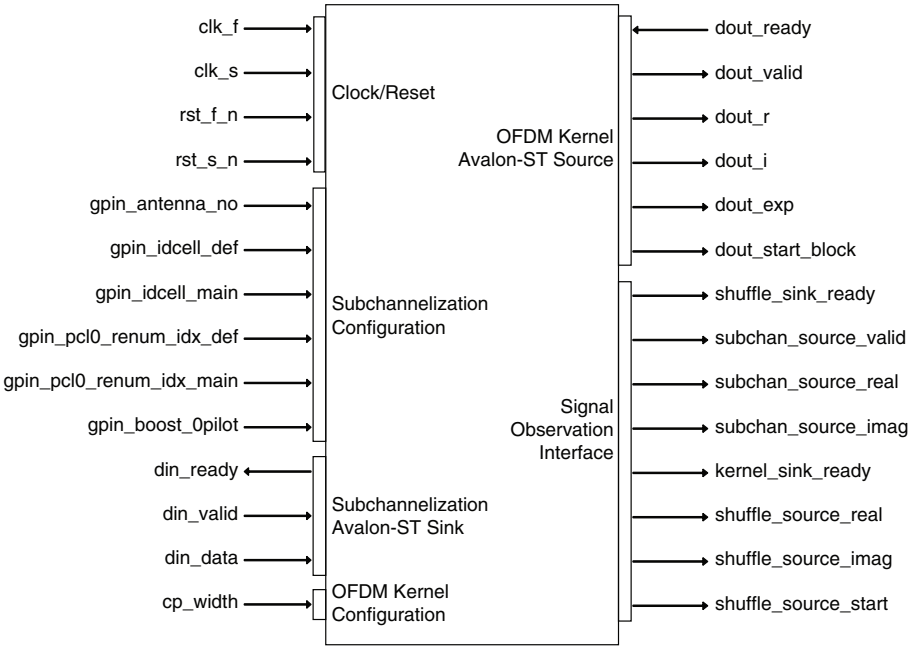


The shuffle operation is achieved through the use of a state machine and an additional internal memory.

Top Level Ports

The top level interface ports for the Downlink OFDMA interface are shown in Figure 7.

Figure 7. Downlink OFDMA Engine Interface Ports

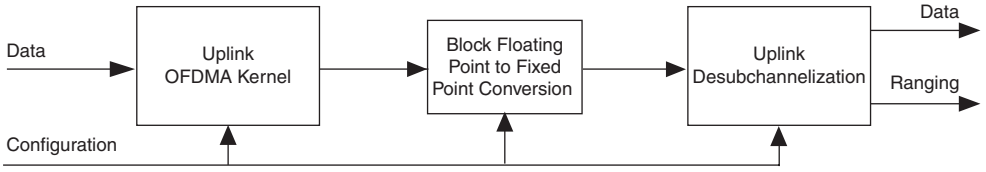


Uplink OFDMA Engine

The Uplink OFDMA engine consists of the uplink OFDM Kernel and the uplink desubchannelization module.

Figure 8 shows the block diagram of the integrated module and the glue logic that ensures compatibility between the two modules.

Figure 8. Uplink OFDMA Engine



For more information on the individual functionality of the modules, refer to the appropriate application note.

Block Floating Point to Fixed Point Conversion

The output data from the OFDM Kernel is in block floating point format. Utilizing this particular format maximizes the output dynamic range for the given input and output data widths.

For more information on block floating point data, refer to *AN83: Binary Number Systems*.

This format is not suitable to input into the uplink desubchannelization module and so it is necessary to convert this block floating point format into a fixed point format.

To ensure that the radix point of each output data packet is equivalent, the output fixed point data must be shifted by the number of places given by the output exponent. For the radix point to be in the same place for each output packet, scaling proportional to the output exponent must take place.

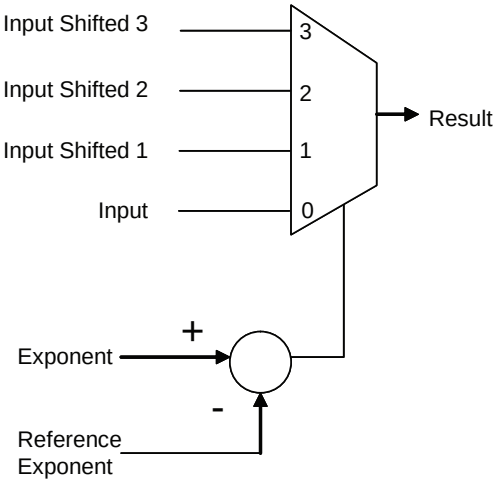
However, a left shift (that is, a multiplication) would lead to overflow since the dynamic range of the output packets is already maximized by the FFT MegaCore function. It is therefore necessary to determine the maximum output exponent for OFDM symbols and shift other output packets relative to this.

This operation could be performed by a barrel shifter. However, the hardware for this type of algorithm is expensive and unnecessary. The input distribution of the OFDM symbols is similar, and this leads to a similar exponent for every symbol processed by the FFT engine. Thus, a simple parameterizable converter that only supports shifts by a small number of values is a more economical solution.

Using this converter it is possible to parameterize the number of shifts to be any power of two, and also to specify the input bit width. The reference exponent is set externally and incorrect selection of this value could lead to a reduction in dynamic range.

This block-to-fixed point converter block is illustrated by the simplified block diagram in Figure 9.

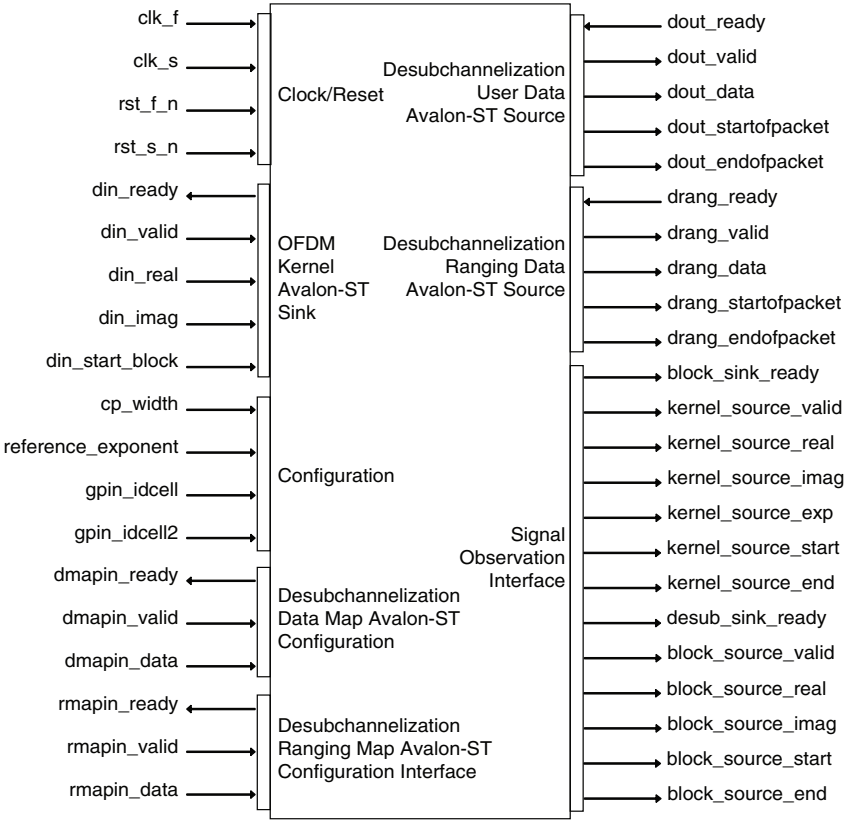
Figure 9. Block-to-Fixed Point Conversion



Top Level Ports

The top level interface ports for the Uplink OFDMA interface are shown in Figure 10 on page 12.

Figure 10. Uplink OFDMA Engine Interface Ports



Getting Started

This section describes the system requirements, installation and other information about using the scalable OFDMA engine reference design.

System Requirements

The scalable OFDMA engine requires the following hardware and software:

- A PC running the Windows 2000/XP operating system
- Quartus II version 6.1 (to support Stratix® III)
- ModelSim SE 5.7d
- Altera FFT MegaCore function

You can download the FFT MegaCore function from **www.altera.com**.

Install the Scalable OFDMA Engine

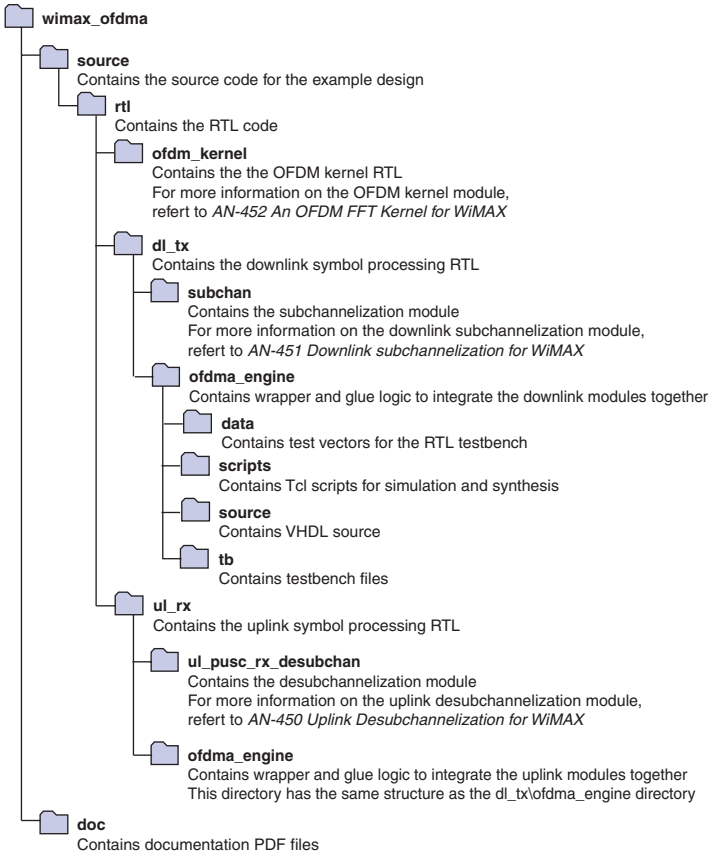
To install the scalable OFDM engine, run the executable file to launch Installshield and follow the installation instructions.

The reference design is installed by default in the directory **c:\altera\reference_designs** but you can change the default directory during the installation.

Figure 11 on page 13 shows the directory structure after installation.

Additional **sim** and **build** directories are created in the directory structure when you simulate and synthesize the design.

Figure 11. Directory Structure



After installing the reference design perform the following:

1. Browse to **<***Quartus II install directory***>\libraries\vhdl\altera**

1. Make a backup copy of the existing **alt_cusp_package.vhd** file.

2. Copy the **alt_cusp_package.vhd** file from *<reference design install directory>***\source\rtl\dl_tx\subchan\source\dl_subchanX \dump** to the **<***Quartus II install directory***>\libraries\vhdl\altera** directory.

Simulation Scripts

Altera provides a number of Tcl scripts to simulate and synthesize the modules that make up the Scalable OFDMA Engine reference design. These scripts are located in the **scripts** directories of the various modules.

To simulate the uplink or downlink OFDMA engine, browse to the appropriate directory and locate the **ofdma_engine_msim.tcl** file

You can modify variables in these scripts which specify the installation path for the modules, and design parameters (for example, FFT size).

After modifying the variables, select **Execute Macro** in the Modelsim Tools directory to setup the simulation and run the test vectors through the testbench.

Synthesis

Altera provides synthesis scripts for each module in the appropriate **scripts** directory.

To synthesize the uplink or downlink OFDMA engine, browse to the appropriate directory and locate the **ofdma_engine_quartus.tcl** file.

Before you synthesize your design in the Quartus II software, open the relevant Tcl script in a text editor and change the variables defining directory locations to match your setup. The variables are defined at the top of the scripts.

Performance

The tables in this section show the resource usage and maximum frequency of operation for Cyclone II, Cyclone III, Stratix II, and Stratix III devices using the Quartus® II software with no special optimizations.

Table 3 shows the Downlink performance for Cyclone II devices.

Table 3. Integrated Downlink OFDMA Engine Performance—Cyclone II (EP2C70F672C7)

FFT Size	Combinational ALUTs	Logic Registers	Memory (Bits)	Memory M4K	Multipliers		f_{MAX} (MHz)
					9×9	18×18	
128	3,130	3,764	31,439	11	0	4	178
512	3,721	4,370	125,483	33	0	4	170
1,024	4,171	4,644	250,916	63	0	4	159
2,048	5,176	5,300	501,762	124	0	4	158

Table 4 shows the Downlink performance for Cyclone III devices.

Table 4. Integrated Downlink OFDMA Engine Performance—Cyclone III (EP3C80F780C7)

FFT Size	Combinational LUTs	Logic Registers	Memory (Bits)	Memory M9K	Multipliers		f_{MAX} (MHz)
					9×9	18×18	
128	3,132	3,766	31,439	8	0	4	161
512	3,724	4,372	125,483	18	0	4	156
1,024	4,176	4,646	250,916	33	0	4	172
2,048	5,142	5,300	501,762	63	0	4	172

Table 5 shows the Downlink performance for Stratix II devices.

Table 5. Integrated Downlink OFDMA Engine Performance—Stratix II (EP2S60F1020C4)

FFT Size	Combinational ALUTs	Logic Registers	Memory (Bits)	Memory Blocks			Multipliers		f_{MAX} (MHz)
				M512	M4K	M-RAM	9×9	18×18	
128	2,668	3,664	31,439	0	11	0	0	4	233
512	3,256	4,266	125,483	0	33	0	0	4	234
1,024	3,658	4,548	250,916	0	63	0	0	4	228
2,048	4,456	5,198	501,762	0	60	1	0	4	194

Table 6 shows the Downlink performance for Stratix III devices.

Table 6. Integrated Downlink OFDMA Engine Performance—Stratix III (EP3SE50F484C3)

FFT Size	Combinational ALUTs	Logic Registers	Memory (Bits)	Memory Blocks			Multipliers		f_{MAX} (MHz)
				M-ALUTs	M9K	M144K	12×12	18×18	
128	2,619	3,667	31,439	0	8	0	1	3	274
512	3,244	4,273	125,483	0	18	0	1	3	258
1,024	3,610	4,544	250,916	0	33	0	1	3	252
2,048	4,403	5,203	501,762	0	63	0	1	3	249

Table 7 shows the Uplink performance for Cyclone II devices.

Table 7. Integrated Uplink OFDMA Engine Performance—Cyclone II (EP2C70F672C7)

FFT Size	Combinational ALUTs	Logic Registers	Memory (Bits)	Memory M4K	Multipliers		f_{MAX} (MHz)
					9×9	18×18	
128	3,908	4,426	38,790	24	2	3	138
512	3,975	4,511	137,384	57	2	3	125
1,024	3,866	4,462	271,780	77	2	3	131
2,048	4,044	4,567	535,950	142	2	3	129

Table 8 shows the Uplink performance for Cyclone III devices.

Table 8. Integrated Uplink OFDMA Engine Performance—Cyclone III (EP3C80F780C7)

FFT Size	Combinational LUTs	Logic Registers	Memory (Bits)	Memory M9K	Multipliers		f_{MAX} (MHz)
					9×9	18×18	
128	3,909	4,427	38,790	19	2	3	149
512	3,976	4,512	137,384	35	2	3	146
1,024	3,834	4,461	271,780	57	2	3	140
2,048	4,015	4,566	535,950	77	2	3	144

Table 9 shows the Uplink performance for Stratix II devices.

Table 9. Integrated Uplink OFDMA Engine Performance—Stratix II (EP2S60F1020C4)

FFT Size	Combinational ALUTs	Logic Registers	Memory (Bits)	Memory Blocks			Multipliers		f_{MAX} (MHz)
				M512	M4K	M-RAM	9×9	18×18	
128	3,483	4,320	38,790	2	22	0	2	3	188
512	3,561	4,406	137,384	2	5	0	2	3	125
1,024	3,450	4,357	271,780	2	35	1	2	3	178
2,048	3,588	4,460	535,950	2	60	1	2	3	164

Table 10 shows the Uplink performance for Stratix III devices.

Table 10. Integrated Uplink OFDMA Engine Performance—Stratix III (EP3SE50F484C3)

FFT Size	Combinational ALUTs	Logic Registers	Memory (Bits)	Memory Blocks			Multipliers		f_{MAX} (MHz)
				M-ALUTs	M9K	M144K	9×9	18×18	
128	3,439	4,323	38,790	0	19	0	2	3	224
512	3,494	4,408	137,384	0	35	0	2	3	223
1,024	3,387	4,358	271,780	0	57	0	2	3	226
2,048	3,536	4,460	535,950	0	37	4	2	3	384

Conclusion

This application note has outlined the advantages of using Altera FPGAs for implementing a IEEE 802.16e compliant system.

A flexible, high-throughput DSP platform needs an FPGA-based implementation platform. In addition, this reference design demonstrates the implementation of the symbol processing portions and how it is possible to achieve rapid deployment of a scalable system by saving up to 18 months of development time.

Revision History

Table 11 shows the revision history for the *AN-412: A Scalable OFDMA Engine for WiMAX* application note.

Table 11. AN-412 Revision History

Version	Date	Errata Summary
2.1	May 2007	Updated for Quartus II version 7.1, Stratix III and Cyclone III devices.
2.0	February 2007	Updated for version 6.1 of the Quartus II software.
1.0	June 2006	First release of this application note.

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