Downlink Subchannelization for WiMAX

February 2007, version 1.0 Application Note 451

Introduction

Altera provides building blocks to accelerate the development of a worldwide interoperability for microwave access (WiMAX) compliant basestations. This application note describes a reference design that demonstrates the suitability of the Altera® tools and devices for implementing the downlink subchannelization function.

WiMAX is an emerging broadband wireless technology that promises high-speed data services. The *IEEE 802.16e-2005* standard enables mobility. There is significant market potential for this technology and it is currently being deployed by equipment manufacturers. Altera devices are the ideal platform for high throughput DSP designs such as those found on a WiMAX basestation channel card, because of the dedicated multiplier blocks and inherent parallel structure. This structure gives a significant cost and performance advantage over general purpose processors for this type of design.

For more information on *IEEE 802.16e-2005*, refer to the *IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE P802.16e-2005, February 2006.*

In orthogonal frequency-division multiple access (OFDMA) systems, multiple end stations or subscriber stations (SSs) transmit at the same time to the access point (AP) or basestation. In the downlink path, the AP splits up the downlink bandwidth into different subchannels. Transmission to each SSs is allocated to one or more subchannels.

Downlink subchannelization is the process of amalgamating the data to be transmitted to the different SSs into OFDMA symbols (each symbol occupying the entire downlink bandwidth).

After the downlink subchannelization, the resultant frequency domain OFDMA symbols are converted into time domain OFDMA symbols (using inverse FFT). Then a cyclic prefix is added to each symbol to provide immunity against multipath propagation. Finally the signal undergoes frequency upconversion and amplification before it is transmitted from the basestation.

The downlink subchannelization reference design provides the following features:

- Maps subchannel data to physical subcarriers to generate OFDMA symbols
- Supports mandatory channelization schemes: full usage of subchannels (FUSC) and partial usage of subchannels (PUSC)
- Dynamically changes between schemes
- Generates pilot information and inserts into OFDMA symbols
- Supports FFT sizes of 128, 512, 1,024, and 2,048 points and offers synthesis time option
- Supports multiple antennas
- Includes Avalon® Streaming (Avalon-ST) interfaces for data input and output—ease of integration with other WiMAX designs

The downlink subchannelization reference design is compliant with the following WiMAX specification versions:

- *IEEE P802.16-Revd/D5-2004 "Part 16: Air Interface for Fixed Broadband Wireless Access Systems"*
- *IEEE Std 802.16e-2005 & IEEE Std 802.16-2004/Cor 1-2005 "Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1"*

The design complies to the following sections of the two specifications:

- *8.4.6.1.2.1 Symbol Structure for PUSC*
- *8.4.6.1.2.2 Symbol Structure for FUSC*
- *8.4.9.4 Modulation*

The design does not perform the data mapping to quadrature phase-shift keying (QPSK) or quadrature amplitude modulation (QAM). Only the data randomization of the symbol according to the pilot polarity is implemented.

However, the following sections are not applicable to this design:

- *8.4.9.4.3.1 Preamble pilot modulation*
- *8.4.9.4.3.2 Ranging pilot modulation*

The reference design includes the following items:

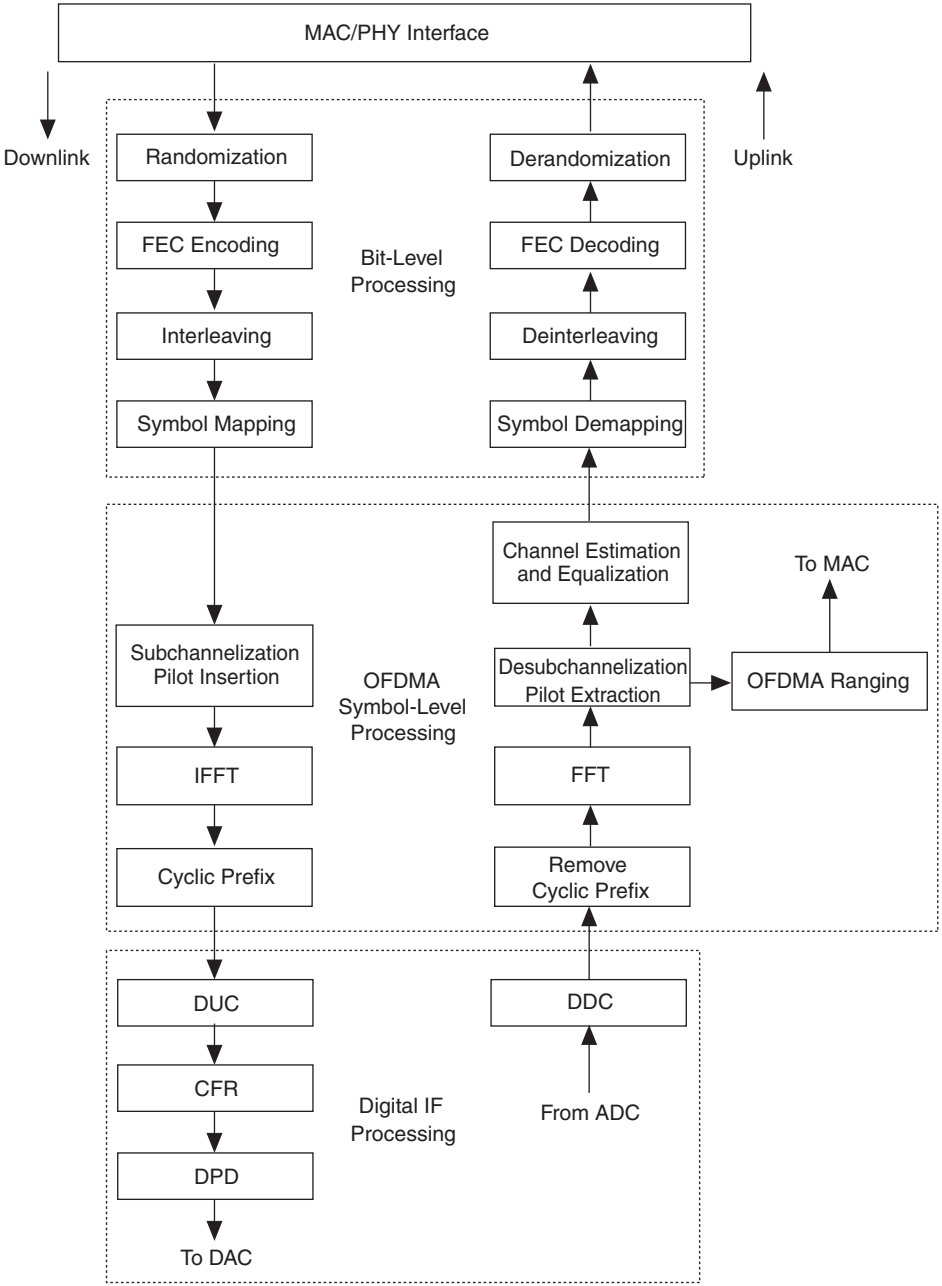
- VHDL code for design
- Different version for each FFT size
- VHDL self-checking testbench
- Same testbench for all FFT versions

- Test data (input and expected output) for range of conditions
- FFT size, `IDCell`, PUSC, and FUSC channelization schemes
- ModelSim RTL Simulation and Quartus® II synthesis Tcl scripts
- Perl script to automate RTL simulations for all test cases

WiMAX Physical Layer

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

Figure 1. WiMAX PHY 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 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 FFT (IFFT) MegaCore functions. Other 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 describes downlink subchannelization.

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

- *AN 412: A Scaleable OFDMA Engine for WiMAX*
- *AN 421: Accelerating DUC & DDC System Designs for WiMAX*
- *AN 430: OFDMA Ranging for WiMAX*
- *AN 434: Channel Estimation & Equalization for WiMAX*
- *AN 439: Constellation Mapper and Demapper for WiMAX*
- *AN 451: Downlink Subchannelization for WiMAX*
- *AN 452: An OFDM FFT Kernel for WiMAX*

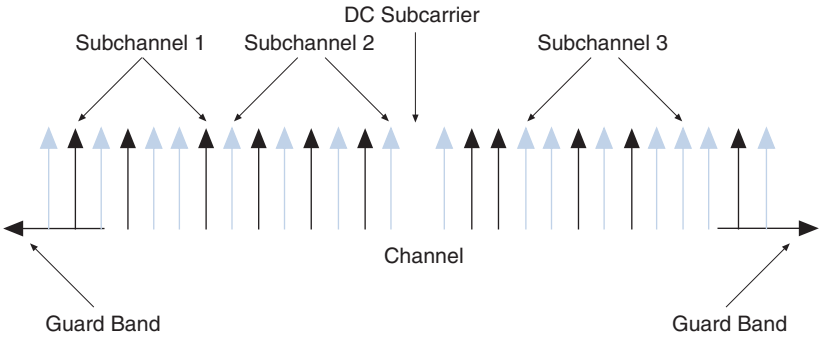
Overview

An OFDMA symbol consists of a number of carriers equal to the size of the Fourier transform (ignoring cyclic prefix for simplicity). 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 (see Figure 2).

Figure 2. OFDMA Frequency Description



Particular users are allocated a number of different subchannels to send and receive data.

The subchannelization design maps the raw constellation data allocated to different subchannels to physical subcarriers in the OFDMA symbol. The mapping formula varies for the 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:
  - The set of used subcarriers is partitioned into clusters (a group of 14 subcarriers)
  - Pilot subcarriers are allocated from within each cluster
  - Subchannels are allocated to clusters

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.

A single packet of user data is distributed over multiple OFDMA symbols.

Table 1 shows the number of subchannels for each mode.

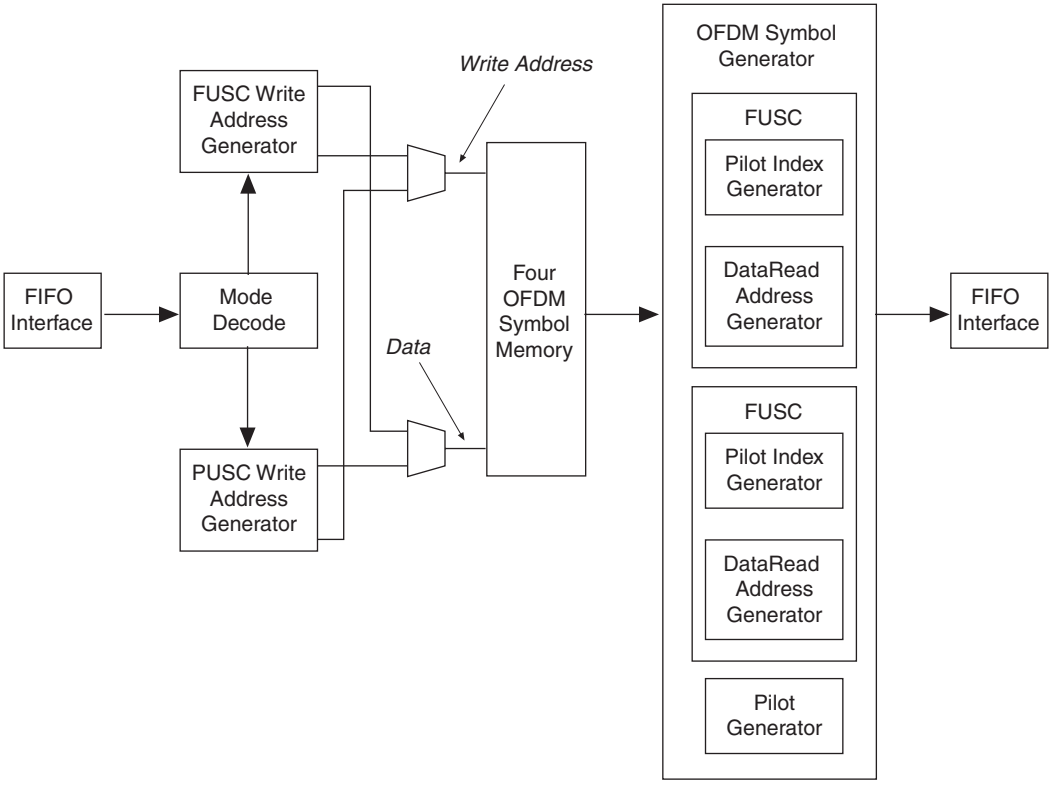
Table 1. Number of Subchannels for Each Mode

Mode	FFT Size (Points)			
	128	512	1,024	2,048
Downlink FUSC	2	8	16	32
Downlink PUSC	3	15	30	60

Functional Description

Figure 3 shows the subchannelization reference design block diagram.

Figure 3. Subchannelization Block Diagram



The data input and output interfaces are Altera Avalon® Streaming (Avalon-ST) compliant.

For more information on the Avalon-ST interfaces, refer to the *Avalon Streaming Interface Specification*.

The design accepts frequency domain data tagged with an information field. The design decodes the information field to determine which of the two channelization modes (PUSC or FUSC) to use.

Depending on the channelization mode, the design passes control to either the FUSC write address generator or the PUSC write address generator. In generic terms, both of these blocks perform the same high level task. They use the subchannel number in the information field, to determine where in the OFDMA symbol to insert the data (i.e., which frequency bin the data is for). A memory write address, relating to this position, is generated. The design writes the data to this address in the OFDM symbol memory. These blocks do not alter the data values.

The FUSC write address generator writes the data into the symbol memory in order of usable data subcarriers (starting from the lowest number 0).

The PUSC write address generator writes the data into the symbol memory in logical cluster order. Thus, the first 12 locations are for the data values comprising logical cluster 0, the next 12 are the logical cluster 1 and so on.

The OFDM symbol generator constructs the frequency domain OFDMA symbols. It determines which mode (PUSC or FUSC) relates to the current symbol in memory to be processed. The design then passes control to either the FUSC or PUSC subblocks. From a high level, they both perform the same following tasks:

- Generating the pilot values
- Modulating the data values read from the symbol memory with the polarity of the pilot generator (multiplication with +1 or -1)
- Constructing the OFDMA symbol by outputting either:
  - Null (left/right guard bands or dc subcarrier)
  - Pilot value
  - Modulated data value

To ensure high throughput, the symbol memory is double buffered. While the design writes data into one buffer, it creates an OFDMA symbol from data in the other buffer. This memory has a depth of four OFDMA symbols, as it must cope with the worst case of PUSC, which generates two symbols at a time.

Multiple Antenna Support

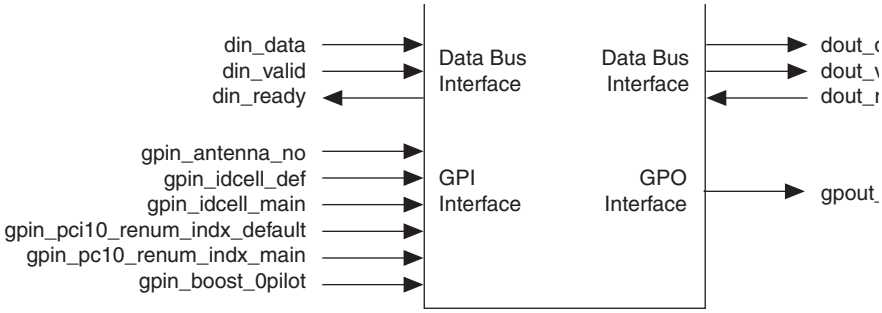
You can share the subchannelization design can be shared among multiple antennas. There is an antenna number signal that you feed into this design. The OFDMA symbol data output is accompanied with an antenna number.

You must run the design at the appropriate clock frequency to allow timesharing among the different antennas.

Interfaces

The design operates with a single clock domain. All transitions are synchronous to the rising edge of the clock. Figure 4 shows the interfaces and signals. There are two input interfaces: a data interface and a general purpose input (GPI) interface. Similarly, there are two output interfaces: a data interface and a general purpose output (GPO) interface.

Figure 4. Subchannelization Design Interfaces



Input Interface

Table 2 shows the input interface signals.

Table 2. Input Interface Signals (Part 1 of 2)

Signal	Width	Direction	Description
Data Bus Interface			
din_data	44, 46, 47, or 48	Input	Frequency domain data plus information field.
din_valid	1	Input	Signifies validity of all data bus inputs.
din_ready	1	Output	Signifies whether design can accept more data.
GPI Interface			
gpin_antenna_no	4	Input	Antenna number.
gpin_idcell_def	6	Input	Default IDCell value.
gpin_idcell_main	6	Input	Main IDCell value.
gpin_pcI0_renum_idx_def	3, 5, 6, or 7	Input	Default PUSC renumbering index start.

Table 2. Input Interface Signals (Part 2 of 2)

Signal	Width	Direction	Description
gpin_pcI0_renum_idx_main	3, 5, 6, or 7	Input	Main PUSC renumbering index start.
gpin_boost_0pilot	16	Input	Boosted value of pilot 0.

Data Bus Input Interface

The data bus interface is Avalon-ST compliant and uses a ready latency of one clock cycle. It applies backpressure to the upstream agent driving data into it, by deasserting din_ready when it can not accept data.

The data bus width can be 44 to 48 bits wide, depending on the FFT size the subchannelization design is working to. Table 3 shows the composition of the data bus—the data component and the different fields that make up the information part.

Table 3. Input Data Bus

Data Bus Fields	Data Bus Bit Slices for Different FFT Sizes			
	128 (44 bits)	512 (46 bits)	1,024 (47 bits)	2,048 (48 bits)
Real data	15:0	15:0	15:0	15:0
Imaginary data	31:16	31:16	31:16	31:16
Subchannel number	33:32	35:32	36:32	37:32
Symbol offset	38:34	40:36	41:37	42:38
Segment number	40:39	42:41	43:42	44:43
Reserved	41	43	44	45
Mode PUSC	43:42	45:44	46:45	47:46

The data parts are fixed at 16 bits for real and 16 bits for imaginary. This width was selected as the upper limit required for most users. If the requirement is for less than 16 bits, you can still use this reference design. by padding the upper unwanted bits with the sign bit value.

For more than 16 bits, contact Altera.

The subchannel number indicates which subchannel the data is for.

The symbol offset is the current OFDMA symbol number modulo 32 relative to the start of data region, the current data sample refers to. This value does not change on a sample by sample basis, but on the first sample for the next OFDMA symbol (for PUSC mode, it must always equal the symbol offset for the lower of the symbols being created).

A cell site can be divided up into three different geographical regions (or segments). Thus segment number can be 0, 1, or 2.

The `mode_pusc` field signals which channelization mode to implement and which of the `IDCell` values (provided on the GPIs) to use.

Table 4 shows the coding for the `mode_pusc` signal. During the downlink frame, at the start the `IDCell` for the first few symbols is assumed to be zero. Thus, all SS listening in always know what `IDCell` value to use when decoding the downlink frame. After the decode, the SSs know what `IDCell` the rest of the frame is coded with. The subchannelization design provides the provision to use two `IDCell` values (the default value does not have to be zero, but should be for IEEE specification compliance).

Table 4. mode_pusc Encoding

Mode	IDCell Value	mode_pusc[1:0]
FUSC	Default	00
	Main	01
PUSC	Default	10
	Main	11

For more information on how these information fields can determine which physical subcarrier the data is for, refer to the IEEE WiMAX specifications.

Input Data Order

This section outlines the restrictions on the order in which you can feed data into the subchannelization design.

For a particular segment and particular antenna you must feed all the following data into the subchannelization design, before any other data:

- One OFDMA symbol (FUSC mode) or
- Two OFDMA symbols (PUSC mode)

Within this restriction, you can feed the different subchannels' (each comprising 48 subcarriers) data in any order but the subcarriers within each subchannel must be applied in order. Table 5 shows a valid sequence for the input data.

Table 5. Input Data Valid Sequence

Sample Number	Subchannel Number	Subcarrier Number
1	6	1
2	6	2
3	4	1
4	20	1
5	13	1
6	6	3
7	20	2
...	...	...
T	20	48
T + 1	6	47
T + 2	6	48
T + 3	13	48

GPI Interface

All GPIs are clocked into the design after the first data sample for each new OFDMA symbol is clocked in. Thus, these signals should be stable with their new values before the last sample of the current OFDMA symbol is clocked in.

The `gpin_antenna_no` 4-bit wide field allows a theoretical maximum of 16 antennas to be supported (limiting factor is the maximum clock frequency that the design can be clocked at).

The `gpin_idcell_def` and `gpin_idcell_main` fields are the two values of `IDCell` that the design uses, depending on the state of the `mode_pusc` signal (part of information field).

Set the `gpin_pcI0_renum_idx_def` and `gpin_pcI0_renum_idx_main` fields should be set to the following values:

```
gpin_pcl0_renum_indx_default = (13 * idcell_default) %
Nc
gpin_pcl0_renum_indx_main = (13 * idcell_main)% Ncc
```

Where Nc is equal to the total number of clusters.

The design uses these two signals for PUSC mode only. Their bit width varies depending on the FFT size as the total number of clusters varies. Table 6 shows the bit width for different FFT sizes.

Table 6. Bit Width of gpin_pcl0_renum_indx for Different FFT Sizes

FFT Size	Number of Clusters	gpin_pcl0_renum_indx Bit Width
128	6	3
512	30	5
1,024	60	6
2,048	120	7

The complex frequency domain data input has a 16-bit width for real and imaginary parts. Internally, the design generates the pilot values to output. As the design does not have knowledge of the number format of the input data, it does not know how to represent the 0 (+1) or 1 (–1) pilot generator output in 16 bits. Furthermore, according to the specifications, you can use boosted pilot values (values greater than +1 or less than –1). Altera provides a general purpose 16-bit wide input (gpin_boost_0pilot), which you must set to the value that the design should use to represent 0 (+1) coming out of the pilot generator. Internally the design calculates the following value to use for pilot generator output of 1 (–1):

```
- gpin_boost_0pilot
```

Output Interface

Table 7 shows the output interface signals.

Table 7. Output Interface Signals (Part 1 of 2)

Signal	Width	Direction	Description
Data Bus Interface			
dout_valid	1	Output	Signifies validity of data bus.
dout_ready	1	Input	Signifies whether design should output more data.
dout_data	32	Output	Frequency domain data.

Table 7. Output Interface Signals (Part 2 of 2)

Signal	Width	Direction	Description
GPO Interface			
gpout_antenna_no	4	Output	Antenna number.

Data Bus Output Interface
This interface is Avalon-ST compliant with a ready latency of 1 clock cycle. It can accept backpressure by the downstream agent sinking data from it (when dout_ready = 0).

The data bus is 32-bits wide. The lowest 16 bits are the real part of the complex data; the upper 16 bits are the imaginary part. This width is predetermined by the width of the complex frequency domain data that is you feed into the design.

Each OFDMA symbol is output starting from frequency bin $-N/2$, through to $+N/2 - 1$ (where N = FFT size).

GPO
There is only one GPO, gpout_antenna_no, which provides the antenna number for the current OFDMA symbol that is output. This value is updated, when the first sample of each OFDMA symbol is output.

Getting Started

This section describes the system requirements, installation and other information about using the downlink subchannelization reference design.

System Requirements

The reference design requires the following hardware and software:

- A PC running the Windows 2000/XP operating system
- Quartus II software version 6.0, SP1
- ModelSim SE 5.7d (mixed VHDL-Verilog HDL license)

Install the Reference Design

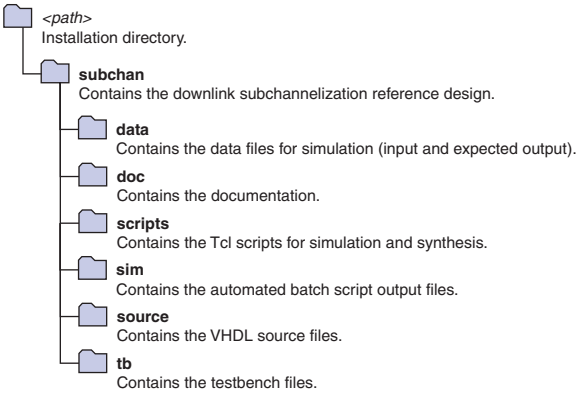
The reference design ships with the scalable OFDMA engine.

For more information and installation instructions on the scalable OFDMA engine, refer to *AN412: A Scalable OFDMA Engine for WiMAX*.

The reference design installs by default into the **c:\altera\reference_designs** directory, but you can change the default.

Figure 5 shows the directory structure, where <*path*> is the top-level directory, **wimax_ofdma\source\rtl\dl_rx**.

Figure 5. Directory



After you install the reference design, follow these steps:

1. Browse to the <*Quartus II install directory*>**\libraries\vhdl\altera**.
2. Make a backup copy of the existing **alt_cusp_package.vhd** file.
3. Copy the **alt_cusp_package.vhd** file from**\source\dl_subchan\dump**directory and paste to the <*Quartus II install directory*>**\libraries\vhdl\altera**.

Table 8 describes the design files.

Table 8. Files

File Name	Directory	Description
run_dl_subchan_tb_msim.tcl	**\scripts**	ModelSim simulation script.
dl_subchan_batch_msim.pl	**\scripts**	Perl batch script for RTL simulations.

Table 8. Files

File Name	Directory	Description
build_quartus.tcl	**\source\dl_subchan_128\dump** **\source\dl_subchan_512\dump** **\source\dl_subchan_1024\dump** **\source\dl_subchan_2048\dump**	Quartus II Tcl synthesis script.
dl_subchan_tb.vhd	**\tb**	RTL testbench.
dl_subchan_tb_pkg.vhd	**\tb**	Package file for testbench.
dl_subchan_tb_n128_pkg.vhd	**\tb**	Package file for testbench.
dl_subchan_tb_n512_pkg.vhd	**\tb**	Package file for testbench.
dl_subchan_tb_n1024_pkg.vhd	**\tb**	Package file for testbench.
dl_subchan_tb_n2048_pkg.vhd	**\tb**	Package file for testbench.

Use the Testbench

The design has different HDL files for each different FFT size.

You must copy the relevant HDL files for the FFT size to be simulated to the **\subchan\source\dl_subchan** directory.

You can use the same testbench to simulate all versions of the subchannelization (for each different FFT size).

The testbench reads input data from a text file. It checks the outputs from the design with the golden data read from another text file and logs any differences to a log files.

The testbench expects an input file **dl_subchan_ipdata.txt**; and an output file **dl_subchan_opdata.txt**.

It expects both files to be located in the simulation directory. The simulation script (**run_dl_subchan_tb_msim.tcl**) simulates the design in the **\subchan\sim** directory. You must copy any data files to this directory.

The testbench generates a log file in the simulation directory, **dl_subchan_log_file.txt**.

The testbench requires a VHDL package file, **\subchan\tb\dl_subchan_tb_pkg.vhd**.

This package file is different depending on which FFT size the design operates on. Package files are in the **\subchan\tb** directory for each different FFT size. Each has a name **dl_subchan_tb_n**<*FFT size*>**_pkg.vhd**.

Before simulation you must copy the appropriate package file to the name the testbench expects.

Simulate in the ModelSim Simulator

To simulate in the ModelSim simulator, follow these steps:

1. Copy files for the relevant FFT size from the **\source\dl_subchan_**<*FFT size*> to the **\source\dl_subchan** directory.

2. Copy the input data file from **\data\fusc\ip** or **\data\pusc\ip** to **\sim\dl_subchan_ipdata.txt**.

3. Copy the output data file from **\data\fusc\exp** or **\data\pusc\exp** to **\subchan\sim\dl_subchan_opdata.txt**.

4. Copy the appropriate package file from **\tb\dl_subchan_tb_**<*FFT size*>**_pkg.vhd** to **\tb\dl_subchan_tb_pkg.vhd**.

5. Open the simulation script **\scripts\ run_dl_subchan_tb_msim.tcl** in a text editor and modify the expected locations of the design.

6. To ensure that the waveform viewer is opened in the ModelSim simulator with appropriate signals loaded into it, set following variable defined in the script to the appropriate value:

```
set batch_mode 0
```

7. In the ModelSim simulator, execute the simulation script **run_dl_subchan_tb_msim.tcl**.

8. After the simulation has completed, view the signals in the waveform viewer and output the log file **\sim\dl_subchan_log_file.txt**.

Run Automated Batch Mode Simulations

Altera supplies 24 different data sets with this design. There are six different sets per FFT size. Half of the data sets test PUSC mode; the other half test FUSC mode.

The pearl script **\scripts\dl_subchan_batch_msim.pl** performs steps 1 through 4 in "Simulate in the ModelSim Simulator" on page 18, then it opens the ModelSim simulator and runs the simulation. Finally it stores the output files to **\sim** with unique name for each simulation.

To run automated batch mode simulations, follow these steps:

1. Open the **\scripts\ run_dl_subchan_tb_msim.tcl** simulation script in text editor and modify the expected locations of the design.

2. To ensure that the ModelSim simulator is run in command mode and the waveform viewer does not open, set following variable defined in the script to the appropriate value:

```
set batch_mode 1
```

3. Open a command prompt.

4. Change the directory to **\subchan\scripts**.

5. Enter the following command:

```
dl_subchan_batch_msim.pl –rtl_sim
```

As the script runs, information is printed to the command window, which indicates which files it is copying, and from where and to where.

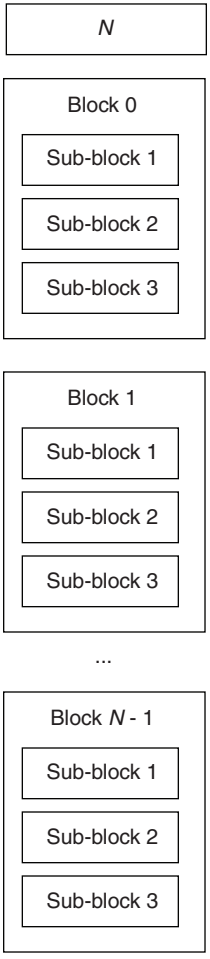
Examine the perl script (**dl_subchan_batch_msim.pl**) to determine which data files comprise each test case and where they are stored. Alternatively, just run the script and examine the messages printed to the command prompt.

Data File Format

This section describes the format of the input and output data files.

Figure 6 shows the general structure of these files. The data is arranged into several blocks. Each block has three sub-blocks. All fields must contain decimal numbers only.

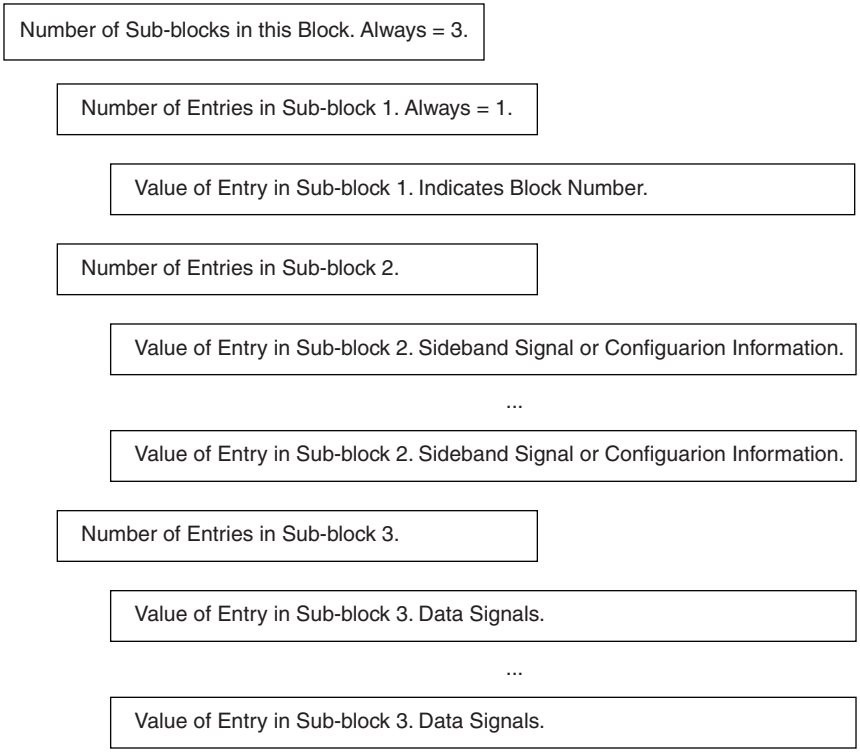
Figure 6. General Data Text File Structure



The first line contains a single number *N*. This number indicates the number of blocks in the file. If the number is zero, ignore its value, as the number of blocks in the file is unknown or not calculated.

After the first line, each block of data follows. Each block comprises three sub-blocks. Figure 7 shows the structure of each block in more detail. The first line in each block contains a single number, which indicates the number of sub-blocks in this current block and should always be 3.

Figure 7. Structure of Single Block in File



The next line contains a single number that indicates the number of entries in sub-block 1, which should always be 1. The following line contains a single number, which is the value of the entry for sub-block 1. Sub-block 1 only contains a block number ID, given by this value.

The next line contains a single number that indicates the number of entries in sub-block 2. Sub-block 2 contains all sideband signal and configuration information. The number of entries in this sub-block varies (depending on which file is being referred to). For example, for 12 entries, the next 12 lines in the files contain the sideband signal and configuration information.

After these, the next line contains a single number, *m* that indicates the number of entries in sub-block 3. The next *m* lines contain information on the data signal values.

Sub-Block 2: Sideband Information

The following code is an example of the possible contents of sub-block 2:

```
12
100 128
101 16
102 0
103 1
104 0
105 13
106 5461
107 1
110 2
120 0
130 1
29 0
```

The first line contains 12, which means that there are 12 entries in sub-block 2. The next 12 lines contain the sideband signal and configuration information. There are two numbers on each line. The first number is a code that indicates which sideband signal or configuration information is referred to. The second number gives the value for this sideband signal or configuration information. Table 9 shows the sub-block 2 field codes.

Table 9. Sub-Block 2 Field Codes

Code	Signal
29	Antenna number.
100	FFT size.
101	Data bit width.
102	`IDCell` default.
103	`IDCell` main.
104	`Pcl0_renum_idx_default`.
105	`Pcl0_renum_idx_main`.
106	`Boost_0pilot_value`.
107	`Pusc_mode`.
108	Segment number.
109	Symbol offset.
110	Reserved.

Sub-Block 3: Data Values

This section describes the sub-block 3 input and output data files.

Input Data File
The input file is arranged so that each block represents data for a particular OFDMA symbol.

The following codes shows an example of the start of sub-block 3 (for a 1,024 FFT size):

```
720
25844 20589 0
-4091 -25986 0
-24208 -21142 0
-10205 6870 0
12621 -14822 0 ............
```

The first line contains the number of data samples in an OFDMA symbol, which is 720 in this example. The 1,024-point FFT size contains 720 data values and the rest are guard bands and pilots. This value is equal to the number of usable subcarriers. Each line contains three numbers: the first is the real part of the sample; the second is the imaginary part; the third is the subchannel number.

Output Data File
Sub-block 3 represents the data for a particular OFDMA symbol.

The following codes shows an example of the possible contents of sub-block 3:

```
28
0 0
0 0
0 0
0 0
... ...
-10780 30676
14527 26341
32502 24782
-7671 9194
-5461 0
-13574 -30198
```

The first line indicates that there are 128 subsequent lines in this sub-block, which equates to the number of data samples in one OFDMA symbol. This example is a 128-point FFT size.

Each subsequent line contains information about each slot sample. Each line always contains two values. The first (furthest left value) is the real part of the data output sample; the second is the imaginary part.

The samples are ordered from frequency bin $-N/2$ through to $+N/2 - 1$ (where N is the FFT size).

Performance

This section shows the synthesis results and throughput.

Synthesis Results

To run synthesis, follow these steps:

1. Open the Tcl synthesis script for the relevant FFT size **\source\dl_subchan_**<*FFT size*>**\dump\ build_quartus.tcl** in a text editor. Modify any path locations to match the locations on your PC.

2. Execute the Tcl script from the Quartus II software.

The scripts stores synthesis files in the **\source\dl_subchan_**<*FFT size*>**\dump\db** directory.

Table 10 shows the synthesis results for all FFT sizes. The results assume 16-bit inputs for the real and imaginary parts of the input data that is fed into the design.

Table 10. Synthesis Results

Device	FFT Size	LEs/ALUTs	Memory (M4K)	9 × 9 Multipliers	f_{MAX} (MHz)
Cyclone II 2C35C6	128	2692 (8%)	6 (6%)	2 (3%)	207
	512	3367 (10%)	18 (17%)	2 (3%)	197
	1,024	3927 (12%)	34 (32%)	2 (3%)	197
	2,048	4857 (15%)	66 (63%)	2 (3%)	180
Stratix II 2S15C4	128	2487 (20%)	6 (8%)	2 (2%)	219
	512	3162 (25%)	18 (23%)	2 (2%)	208
	1,024	3570 (29%)	34 (44%)	2 (2%)	193
	2,048	4290 (34%)	66 (85%)	2 (2%)	203

Throughput

The design continuously outputs OFDMA symbols at a data rate appropriate to the FFT size. The design must be able to output an OFDMA symbol within the time taken to clock in the data for the next OFDMA symbol.

Table 11 shows that to meet throughput the design must be clocked at a minimum of four times the data rate for all FFT sizes, except 128-point FFTs, which need to be clocked at six times the data rate.

Table 11. Minimum Clock Rate to Meet Throughput

FFT Size	Data Rate (Msps)	Minimum Clock Frequency (MHz)
128	1.25	7.5
512	5	20
1024	10	40
2048	20	80

Revision History

Table 12 shows the revision history for this application note.

Table 12. Revision History

Version	Date	Description
1.0	February 2007	First release.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
literature@altera.com