LogiCORE IP DUC/DDC Compiler v1.1

DS766 September 21, 2010 **Product Specification**

Introduction

The Xilinx LogiCORE™ IP DUC/DDC Compiler implements high-performance, optimized Digital Up- and Down-Converter modules for use in wireless base stations and other suitable applications. In addition to a wide range of parameter options, resource trade-off options are available to tailor the core to a particular application.

Features

- Drop-in module for Virtex®-6, Virtex-5, and Spartan®-6 FPGAs
- Generates Digital Up-Converter modules for a range of RF sample rates between 61.44 and 245.76 MHz
- Generates Digital Down-Converter modules for a range of RF sample rates between 61.44 and 184.32 MHz
- Supports TD-SCDMA (1.6 MHz channel) and LTE (1.4, 3, 5, 10, and 20 MHz channels)
- Supports up to 18 carriers (maximum dependent upon wireless standard and channel bandwidth)
- Implementation options to configure clock rate, enable optional control signals, and set resource usage preferences
- Supports Fs/4 IF down-mixing in DDC configuration
- Supports programmable carrier frequencies (within the limits imposed by wireless standard)
- Supports fixed carrier phase offsets between 0 and 2π
- Supports selectable carrier relative gain levels
- Streaming data interface allowing simple integration into signal processing data flows
- Easy-to-use programming interface compliant with AMBA® 3 APB standard
- Xilinx CORE Generator™ graphical user interface (GUI) features: resource and latency estimation, frequency and phase raster reporting
- For use with Xilinx CORE Generator software v12.3 or later

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family[1]	Virtex-6, Virtex-5, Spartan-6				
Supported User Interfaces	AMBA APB (AMBA Advanced Peripheral Bus)				
	Resources[2]				**Frequency**
Configuration	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs[3]**	**Max. Freq.[4]**
LTE DUC 4A4Cx5	7167	9091	72	1	368.64
LTE DDC 4A2Cx20	6700	10984	112	2	368.64
TD-SCDMA DUC 4A9C	5547	8299	92	38	368.64
Provided with Core					
Documentation	Product Specification				
Design Files	Netlist				
Example Design	Not Provided				
Test Bench	Not Provided				
Constraints File	Not Provided				
Simulation Model	C Model				
Tested Design Tools					
Design Entry Tools	CORE Generator™				
Simulation	Mentor Graphics ModelSim 6.5d Cadence Incisive Enterprise Simulator (IES) v9.2 ISE Simulator 12.3				
Synthesis Tools	Not Provided.				
Support					
Provided by Xilinx, Inc.					

1. For a complete listing of supported devices, see the release notes for this core.
2. Resources listed here are for Virtex-6® devices. For more complete device performance numbers, see Tables M-N.
3. Based on 18K block RAMs
4. Performance numbers listed are for Virtex-6 FPGAs. For more complete performance data, see "Performance and Resource Utilization," page 29.

Overview

Digital Up-/Down-Converters (DUCs/DDCs) are key components in wireless communications systems, linking the baseband processing function with the radio front end.

A DUC forms part of the transmit path of digital radio front end (DFE) signal processing systems, and performs the function of filtering and up-converting the baseband signal to a higher sample rate to be passed to the radio front end via the Digital-to-Analog Converter (DAC), or to provide input to Crest Factor Reduction (CFR), Digital Pre-Distortion (DPD), I/Q offset correction, or other ancillary RF signal processing functions applied prior to the DAC. A DUC may include a multi-carrier mixing stage to combine multiple carriers into a composite passband signal.

A Digital Down-Converter (DDC) forms part of the receive path of a digital radio front-end signal processing system, following the Digital-to-Analog Converter (DAC), and Automatic Gain Control (AGC) or other ancillary RF signal processing functions. A DDC performs the function of filtering and down-converting the input RF sample rate to the baseband processing sample rate of the system (or an integer multiple thereof, for example 2x for symbol timing recovery.) The DDC may also perform frequency translation to shift each carrier of a multi-carrier system to baseband ready for de-modulation.

Channel selection filtering is normally incorporated into the filtering functions of DUC or DDC modules, and the sample rate conversion is normally performed most efficiently over multiple stages, with appropriate low-pass filtering for anti-aliasing or image rejection. The general architecture of a DUC or DDC therefore consists of multiple stages of filters and mixers, with the mixers being constructed variously from direct digital synthesizers, multipliers, and simple logic functions. This generalized architecture is illustrated in Figure 1 and Figure 2.

Each core configuration has been designed to meet the requirements of the relevant air standard with a target spectral mask margin of approximately 10 dB. The generated core will meet or exceed the ACS, ACLR, or EVM requirements for the relevant specifications, and EVM is further limited to 2% or less to provide maximum flexibility in other areas of the wireless digital front end (DFE).

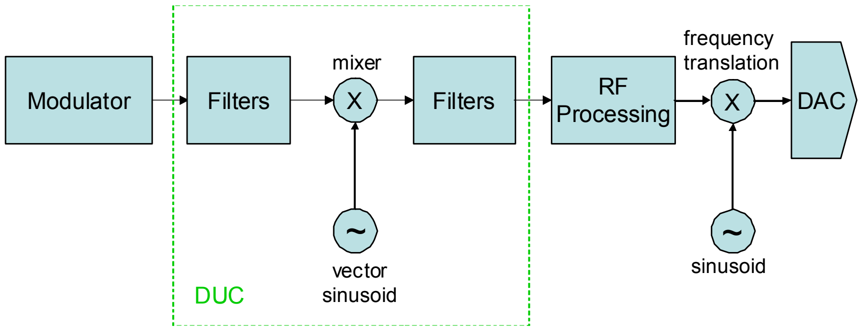


Figure 1: **Generalized DUC Architecture**

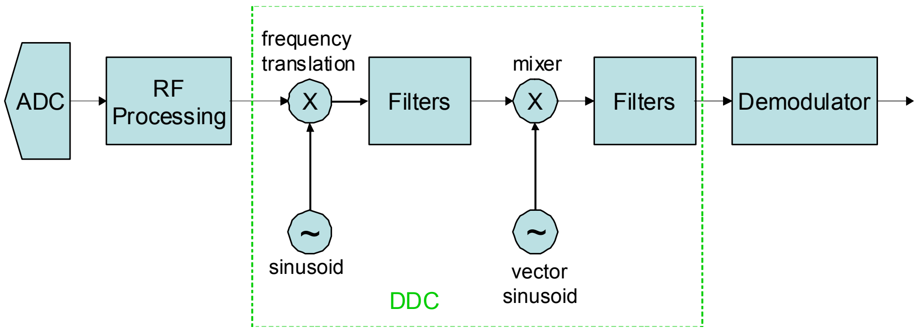


Figure 2: **Generalized DDC Architecture**

The DUC/DDC Compiler core provides an easy-to-use programming interface to allow carrier positions and relative gain levels to be programmed, as well as to provide configuration information and status reporting.

The DUC offers the capability to mix multiple carriers into a complex composite signal centered around zero Hz, while the DDC configuration offers the option of additionally translating a real passband composite signal centered at Fs/4 Hz, where Fs is the input sample rate (usually the ADC sample rate), to a complex composite signal at zero Hz. This two-stage mixing process is illustrated in Figure 3.

The core covers a wide range of parameter options, and automatically compiles a highly optimized filter cascade and mixer structure from the system-level specification. Advanced algorithms select appropriate mixing sample rates, filter types, data path configurations, and other meta-parameters to create an efficient design that meets the performance requirements of the relevant wireless air interface specification and achieves the resource usage goals of the user.

The DUC/DDC Compiler core provides an easy-to-use programming interface to allow carrier positions and relative gain levels to be programmed, as well as providing a status reporting mechanism. This interface complies with the AMBA 3 APB bus specification.

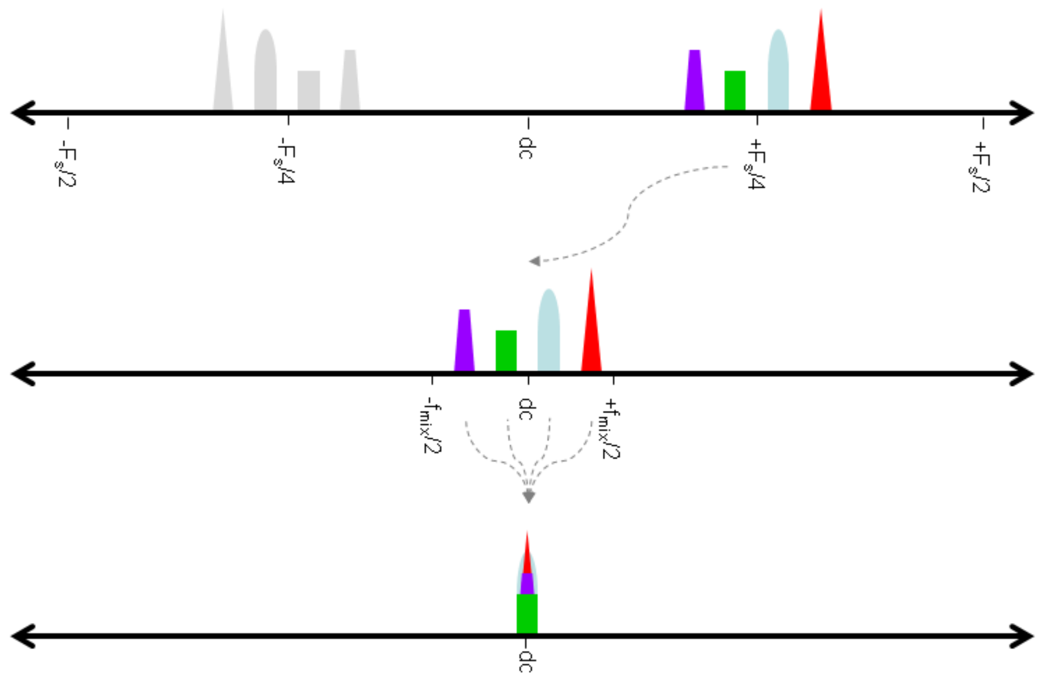


Figure 3: **Two-Stage Down-mixing of Multiple Carriers**

Core I/O Port Definitions

Table 1 defines the core port names and port functional descriptions.

Table 1: **Core Signal Pinout**

Name	Interface	Direction	Description
CLK	System	Input	Core clock (active rising edge.) Always present.
DATA_RESETn	System	Input	Synchronous reset (active low.) Asserting DATA_RESETn synchronously with CLK resets the core control logic functions, but does ***not*** reset the filter data memory contents. DATA_RESETn is an optional pin.
SDATA_VALID	Data slave	Input	Indicates that the input data is valid. Data is input into the core when both SDATA_VALID and SDATA_READY are asserted.
SDATA_READY	Data slave	Output	Core ready to receive input data. Used by core to indicate required input sample rate. At reset, SDATA_READY is held high until the first time SDATA_VALID goes high. Thereafter, SDATA_READY goes high for one cycle at a time, at the configured input sample rate. If input data is not provided at the required time (SDATA_READY is high but SDATA_VALID is low), then SDATA_READY is held high until SDATA_VALID goes high. The core continues to process and output data that is in its pipeline, but does not start processing any new data until new data is provided (indicated by SDATA_VALID going high.) This allows the external master to pause its supply of data. When this occurs, the core signals a missing input interrupt, see INT_MISSINPUT.
SDATA_R[M-1:0]	Data slave	Input	Real sample input data (DDC only), single antenna case. “M” denotes the input bit width.
SDATA_I[M-1:0]	Data slave	Input	Complex sample input data, in-phase portion, single antenna case. “M” denotes the input sample bit width.
SDATA_Q[M-1:0]	Data slave	Input	Complex sample input data, quadrature portion, single antenna case. “M” denotes the input sample bit width.
SDATA_C[M-1:0]	Data slave	Input	Complex sample input data in TDM format, single antenna case. “M” denotes the input sample bit width.
SDATA_R_Ax[M-1:0]	Data slave	Input	Real sample input data (DDC only), multiple antenna case. “M” denotes the input bit width while “x” signifies the antenna number.
SDATA_I_Ax[M-1:0]	Data slave	Input	Complex sample input data, in-phase portion, multiple antenna case. “M” denotes the input bit width while “x” signifies the antenna number.
SDATA_Q_Ax[M-1:0]	Data slave	Input	Complex sample input data, quadrature portion, multiple antenna case. “M” denotes the input bit width while “x” signifies the antenna number.
SDATA_C_Ax[M-1:0]	Data slave	Input	Complex sample input data in TDM format, multiple antenna case. “M” denotes the input bit width while “x” signifies the antenna number.
SDATA_LAST	Data slave	Input	Indicates the last sample of an input packet. DUC input packet contains complex data for each carrier, I then Q for TDM format, in ascending numerical carrier order. DDC input packet contains one complex data sample, I then Q for TDM format. Signal is not present if the input packet is one cycle long (DDC, not TDM format.) If SDATA_LAST is asserted for any sample other than the last sample of an input packet, or de-asserted for the last sample of an input packet, then the core ignores SDATA_LAST and signals a packet error interrupt, see INT_ERRPACKET.
MDATA_VALID	Data master	Output	Indicates that the output data is valid. Data is output from the core when both MDATA_VALID and MDATA_READY are asserted.

Table 1: **Core Signal Pinout *(Cont'd)***

Name	Interface	Direction	Description
MDATA_READY	Data master	Input	Indicates that the external slave is ready to receive output data. The core will present valid output data until MDATA_READY is asserted. The core cannot accept back pressure. If MDATA_VALID is high but MDATA_READY is held low, then any future output samples that the core generates will be internally discarded. When this occurs, the core signals a lost output interrupt, see INT_LOSTOUTPUT. Slaves are recommended to tie MDATA_READY high.
MDATA_I[N-1:0]	Data master	Output	Complex sample output data, in-phase portion, single antenna case. “N” denotes the output sample bit width.
MDATA_Q[N-1:0]	Data master	Output	Complex sample output data, quadrature portion, single antenna case. “N” denotes the output sample bit width.
MDATA_C[M-1:0]	Data master	Output	Complex sample output data in TDM format, single antenna case. “N” denotes the output sample bit width.
MDATA_I_Ax[N-1:0]	Data master	Output	Complex sample output data, in-phase portion, multiple antenna case. “N” denotes the output bit width while “x” signifies the antenna number.
MDATA_Q_Ax[N-1:0]	Data master	Output	Complex sample output data, quadrature portion, multiple antenna case. “N” denotes the output bit width while “x” signifies the antenna number.
MDATA_C_Ax[N-1:0]	Data master	Output	Complex sample output data in TDM format, multiple antenna case. “N” denotes the output bit width while “x” signifies the antenna number.
MDATA_LAST	Data master	Output	Indicates the last sample of an output packet. DUC output packet contains one complex data sample, I then Q for TDM format. DDC output packet contains complex data for each carrier, I then Q for TDM format, in ascending numerical carrier order. Signal is not present if the output packet is one cycle long (DUC, not TDM format.)
MDATA_CLEAN	Data master	Output	Indicates when output data is clean, that is, calculated solely from input data, not from invalid data remaining within the data path since initialization or reset. As output data from FIR filters is dependent on both input data and internal state, it takes a number of output samples following initialization or reset until all internal state at the time of initialization or reset has been flushed from the filters. This signal is low for output samples following initialization or reset until the filter internal state has been flushed. Thereafter, this signal is high for all output samples (until the next reset.)
SREG_PRESETn	Programming	Input	Programming interface. Compliant with AMBA 3 APB protocol. See the AMBA 3 APB bus specification [Ref 7] for detailed information.
SREG_PADDR[11:0]	Programming	Input	
SREG_PSEL	Programming	Input	
SREG_PENABLE	Programming	Input	
SREG_PWRITE	Programming	Input	
SREG_PWDATA[31:0]	Programming	Input	
SREG_PREADY	Programming	Output	
SREG_PRDATA[31:0]	Programming	Output	
SREG_PSLVERR	Programming	Output	

Table 1: **Core Signal Pinout *(Cont'd)***

Name	Interface	Direction	Description
INT_MISSINPUT	Interrupt	Output	Interrupt flag indicating that a missing input sample condition has occurred. Valid input was not provided on the data input interface in the required clock cycle. The input interface paused until valid input was provided on the data input interface in some later clock cycle. The output data values will be unaffected, but there will be a corresponding pause in the output data rate once data has propagated through the core. Once asserted, this interrupt signal will remain high until the interrupt is cleared or disabled using the programming interface, or DATA_RESETn is asserted.
INT_ERRPACKET	Interrupt	Output	Interrupt flag indicating an error in input packet length has occurred. SDATA_LAST went high when not expected, so the input packet is too short, or did not go high when expected, so the input packet is too long. Output data may be incorrect. Once asserted, this interrupt signal will remain high until the interrupt is cleared or disabled using the programming interface, or DATA_RESETn is asserted.
INT_LOSTOUTPUT	Interrupt	Output	Interrupt flag indicating that a lost output sample condition has been detected. Valid output was not accepted on the data output interface in one of the required clock cycles. The following sample was dropped to maintain the output sample rate. Once asserted, this interrupt signal will remain high until the interrupt is cleared or disabled using the programming interface, or DATA_RESETn is asserted.
INT_DUCDDC	Interrupt	Output	Combined interrupt output. This signal is the logical OR of all other interrupt output signals. It indicates that one or more interrupts are active.

CORE Generator Graphical User Interface

The DUC/DDC Compiler core GUI has several pages with fields in which to set parameter values for the particular configuration required, while also providing some user feedback for information about the implementation. This section provides a description of each GUI field.

Tab 1: IP Symbol

The IP Symbol tab illustrates the core pinout.

Tab 2: Implementation

The Implementation tab displays:

- Resource estimation information
- Quantization and scaling details
- Estimated core latency in both clock cycles and time based on the specified clock rate

The estimated number of DSP slices is displayed in addition to an approximate count of the number of block RAM elements required to implement the design. Usage of general slice logic is not currently estimated. The results in the Resource Estimation are estimates only using equations that model the expected core implementation structure. Users should implement the core following generation for a more accurate report on resource usage (an ISE project file is created by Core Generator to ease integration with the IDS.) The final resource cost when integrated into the user's application system may differ due to additional routing congestion. It is not guaranteed that the resource estimates provided in the GUI match the results of a mapped core implementation.

The latency report box provides an indication of the approximate expected delay of the core from input to output, in terms of both clock cycles and absolute time. This information is a useful metric for the core in assessing suitability for particular applications. The figure is approximate and users are encouraged to confirm the actual latency in simulation following core generation.

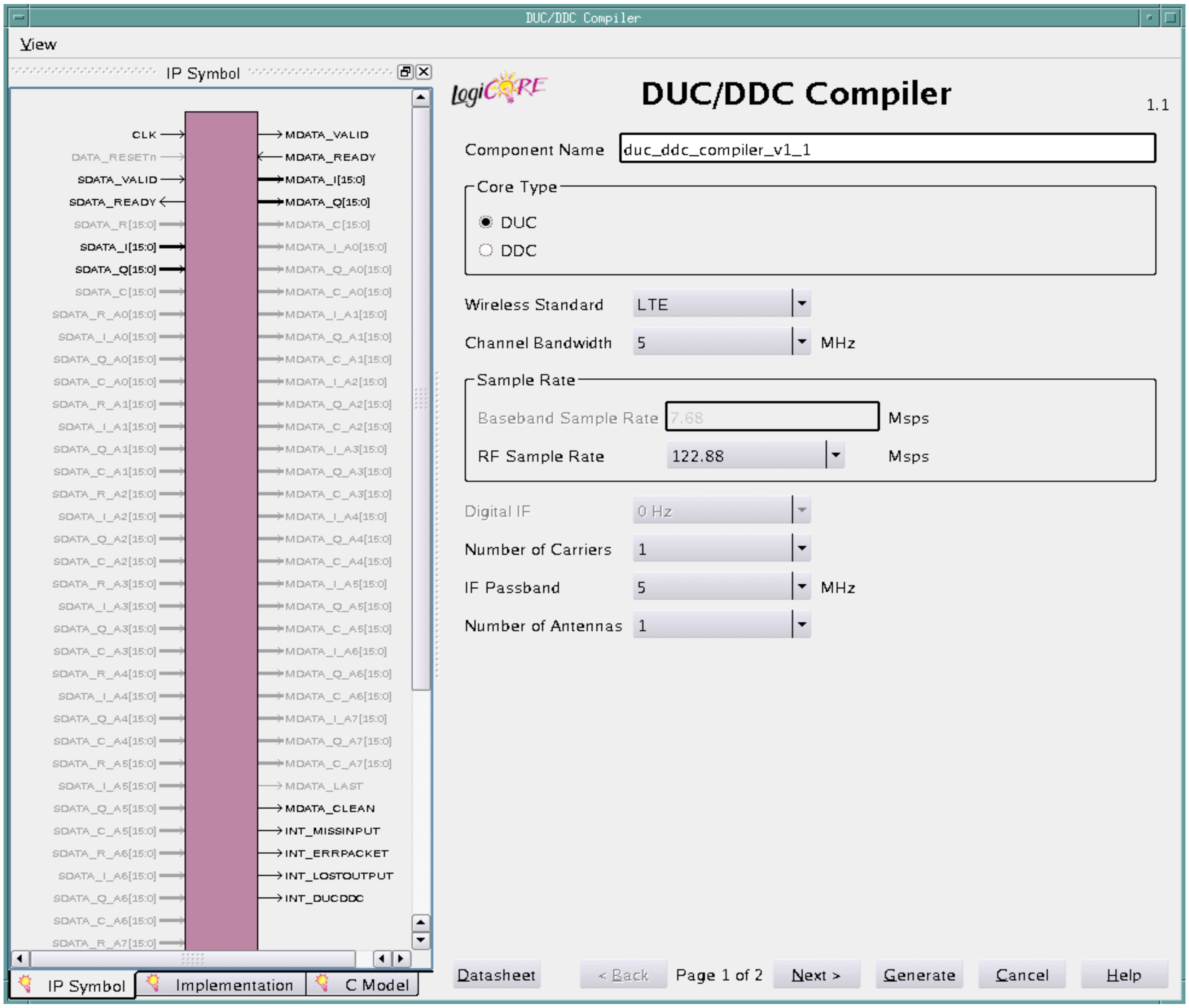
The Data Format box provides information on the quantization and scaling used by the core. Core inputs are assumed to range between -1.0 to +0.999.., i.e. 1 integer bit with the remaining bits being fractional. Core outputs are scaled by the integer bit growth through the core. Integer bit growth may be introduced in filters and mixers, and the core accumulates these bits to maintain full accuracy (no saturation or scaling stages are inserted in the data path processing chain.) The Data Format information is provided to allow the user to decide how to handle the output samples. For example, the user may decide to saturate and round the sample values at the core output based on the reduced fractional width, removing some or all of the additional integer bits.

Tab 3: C Model

This tab provides details on how to obtain a copy of the bit-accurate C model for this core.

Page 1

Page 1 of the GUI is shown in Figure 4.



Component Name: duc_ddc_compiler_v1_1

Core Type: [✓ DUC] DDC

Wireless Standard: LTE

Channel Bandwidth: 5 MHz

Sample Rate

Baseband Sample Rate: 7.68 Msps

RF Sample Rate: 122.88 Msps

Digital IF: 0 Hz

Number of Carriers: 1

IF Passband: 5 MHz

Number of Antennas: 1

Figure 4: **GUI Page 1, with IP Symbol Tab**

- **Component Name:** The name of the core component to be instantiated. The name must begin with a letter and be composed of the following characters: a to z, 0 to 9, and "_".
- **Core Type:** Select between DUC and DDC options.
- **Wireless Standard:** Select the required wireless air interface standard, either LTE or TD-SCDMA.
- **Channel Bandwidth:** Select between 1.4, 3, 5, 10, and 20 MHz channel bandwidth options for LTE. This field is set automatically to 1.6 MHz for TD-SCDMA.
- **Baseband Sample Rate:** The baseband sample rate value is selected automatically based on the Channel Bandwidth setting. The relevant values are: 1.28 Msps for TD-SCDMA 1.6 MHz channel; 1.92 / 3.84 / 7.68 / 15.36 / 30.72 Msps for LTE 1.4 / 3 / 5 / 10 / 20 MHz channels respectively.
- **RF Sample Rate:** Select the RF sample rate value to suit the data converter sample rates in the application system, or to match up with another intermediate sample rate. For DUC implementation, the allowable RF

sample rate values are: 61.44, 76.80, 92.16, 122.88, 153.60, 184.32, and 245.76 Msps; while for DDCs, the range is: 61.44, 76.80, 92.16, 122.88, 153.60, and 184.32 Msps.

- **Digital IF:** Select the intermediate frequency mixing option to implement, either zero IF or Fs/4. Only available for DDC configurations. The Fs/4 option adds an efficient quarter sample rate frequency translation stage to convert the RF input signal from a real passband signal centered at Fs/4, where Fs is the RF Sample Rate value, to a complex passband signal centered at 0 Hz, ready for extraction of individual carrier sample streams from the composite multi-carrier signal.
- **Number of Carriers:** Select the required number of carriers. Support for carrier options is dependent upon the Wireless Standard and Channel Bandwidth already selected. Table 2 shows the carrier options that are supported in the core. Some RF Sample Rate values will reduce the number of carriers supported.

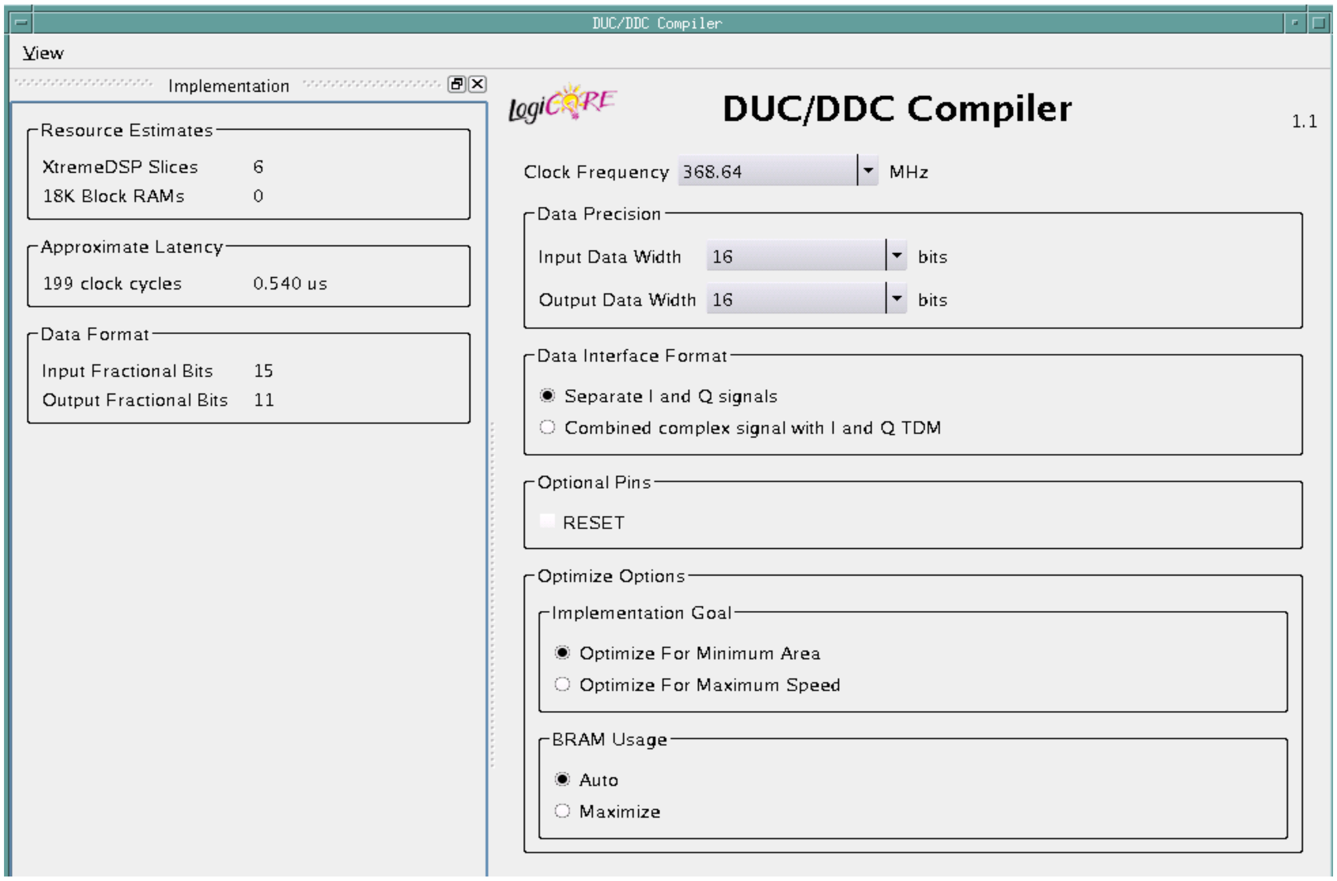
Table 2: **Carrier Support Options**

Wireless Standard	Channel Bandwidth	Supported Number of Carriers
LTE	1.4	1, 2, 3, 4, 5, 6, 8, 10, 12, 16
LTE	3	1, 2, 3, 4, 5, 6, 8, 10, 12, 16
LTE	5	1, 2, 3, 4, 5, 6, 8, 10, 12
LTE	10	1, 2, 3, 4, 5, 6
LTE	20	1, 2, 3
TD-SCDMA	1.6	1, 3, 6, 9, 12, 15, 18

- **IF Passband:** Select the required IF passband width in which you wish to place the carriers. The range of valid values for this option is: 5, 10, 15, 20, 30, 40, 50, and 60 MHz; however, some limits are applied to limit this range further. The minimum value will be the smallest option in which all carriers will fit, while the largest option will be no more than twice than the minimum, or half of the RF Sample Rate value, whichever is smaller.
- **Number of Antennas:** Select the number of antennas to be implemented, up to 8 antennas in total. A separate data path and rate conversion filter cascade will be implemented for each antenna, and appropriate I/O pins will appear on the IP symbol and in the output netlist for these data paths.

Page 2

Figure 5 shows the GUI when a single carrier is used.



Resource Estimates

XtremeDSP Slices	6
18K Block RAMs	0

Approximate Latency

199 clock cycles	0.540 us

Data Format

Input Fractional Bits	15
Output Fractional Bits	11

Clock Frequency 368.64 MHz

Data Precision

Input Data Width 16 bits

Output Data Width 16 bits

Data Interface Format

[✓ Separate I and Q signals]
Combined complex signal with I and Q TDM

Optional Pins

RESET

Optimize Options

Implementation Goal

[✓ Optimize For Minimum Area]
Optimize For Maximum Speed

BRAM Usage

[✓ Auto]
Maximize

Figure 5: **GUI Page 2, with Implementation Tab**

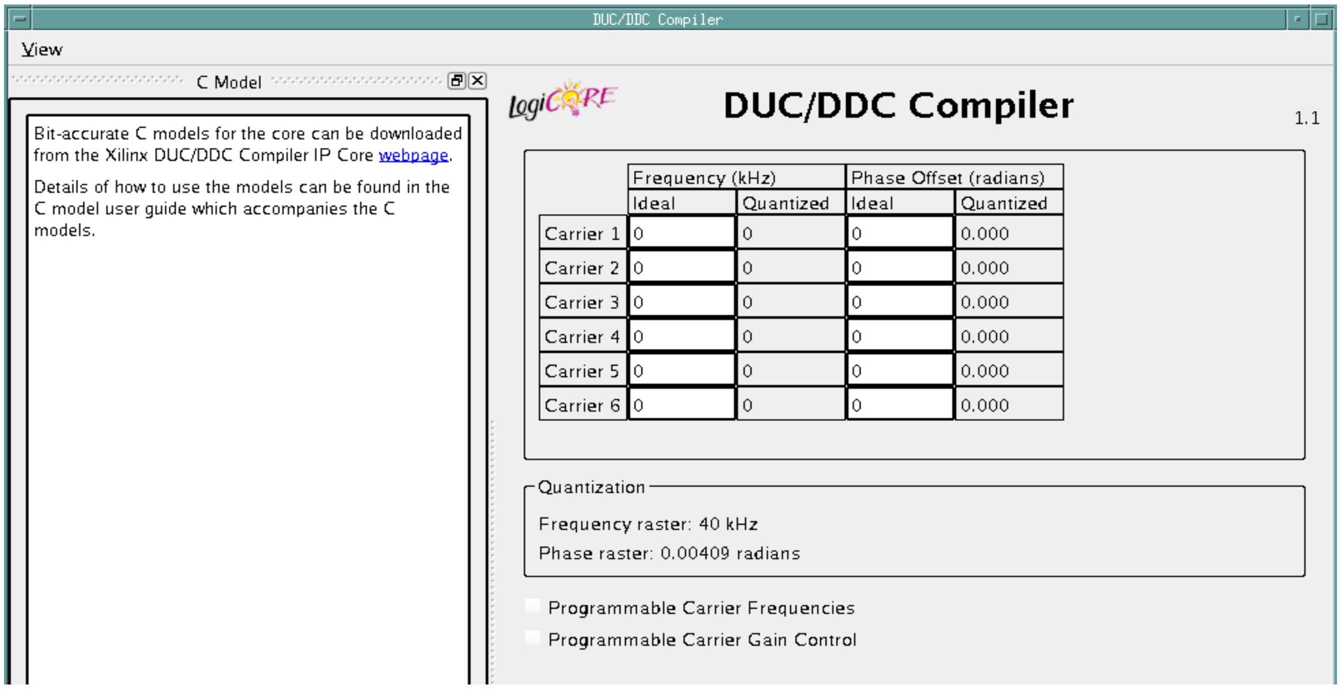
- **Clock Frequency:** The Clock Frequency is selected via a drop-down list which is limited to integer multiples of the RF Sample Rate, within the limits of the selected FPGA device family and the selected speed grade.
- **Data Precision – Input Data Width:** Select the required input data width, from 11 to 17 bits.
- **Data Precision – Output Data Width:** Select the required output data width, from 11 to 18 bits.
- **Data Interface Format:** Select the interface format used for both input and output data ports, either separate I & Q data ports or a single complex data port with I & Q supplied in a TDM format (In-Phase first.) This option is only available if the clock frequency is at least twice the RF Sample Rate value; if it is less than twice the RF Sample Rate, it is set to separate I & Q data ports. The option is also disabled when using a DDC with the digital IF set as Fs/4, in which case only a real valued data signal is provided as input, and the output is separate I & Q data ports.
- **Optional Pins – Reset:** Select whether or not to add reset capability to the core. This option adds two reset pins to the core, one for the main data path logic (`DATA_RESETn`, active low) and another for the programming port (`SREG_PRESETn`, active low.) Note that `DATA_RESETn` does not reset filter sample history.
- **Optimize Options – Implementation Goal**: Select either Minimum Area or Maximum Speed as an Implementation Goal. The recommendation for this setting is to use Minimum Area first, as generally this setting will also achieve maximum performance and does not require any additional resources; however, if timing goals are not being achieved, the architectural changes enabled by the Maximum Speed setting can improve results, at the expense of an increase in core resource usage.

- **BRAM Usage**: Select whether to use more block RAMs or leave the decision to the filter sub-core functions to select storage type appropriately. This optimization directive is a goal and will not remove all block RAM usage from the core.

Page 3

When multiple carriers are used, Page 3 (Figure 6) allows the specification of carrier frequencies and phase offsets for each carrier.

Note: Page 3 is not present when a single carrier configuration is used.



Bit-accurate C models for the core can be downloaded from the Xilinx DUC/DDC Compiler IP Core webpage.

Details of how to use the models can be found in the C model user guide which accompanies the C models.

	Frequency (kHz)		Phase Offset (radians)	
	Ideal	Quantized	Ideal	Quantized
Carrier 1	0	0	0	0.000
Carrier 2	0	0	0	0.000
Carrier 3	0	0	0	0.000
Carrier 4	0	0	0	0.000
Carrier 5	0	0	0	0.000
Carrier 6	0	0	0	0.000

Quantization

Frequency raster: 40 kHz

Phase raster: 0.00409 radians

☐ Programmable Carrier Frequencies

☐ Programmable Carrier Gain Control

Figure 6: **GUI Page 3, with C Model Tab**

- **Carrier Specification Table:** The Carrier Specification Table presents the user with a set of text entry cells in which to specify ideal carrier frequencies and phase offsets, and a matching set of report text cells that report the closest achievable quantized equivalent of these ideal values.
- **Quantization:** This text box provides the user with information on the quantization units that can be expected when checking the Quantized cells in the Carrier Specification Table. The quantized values for frequency and phase will be multiples of the specified quanta.
- **Programmable Carrier Frequencies:** Select the capability to re-program carrier frequencies via the programming port.
- **Programmable Carrier Gain Control:** Select the capability to scale carrier amplitudes prior to multi-carrier mixing by a programmable gain factor. This feature is only available for a DUC, and is limited to integer powers of 2 only, with a nominal maximum gain of 1.0 (2^0) and a minimum positive gain of 2^{-8}, with zero gain also available.

Using the DUC/DDC Compiler IP Core

The CORE Generator GUI performs error-checking on all input parameter sets, applying range or value limitations as appropriate and providing feedback to the user to guide configuration. Resource estimation and approximate latency information are also available.

Several files are produced when a core is generated, and customized instantiation templates for Verilog and VHDL design flows are provided in the .veo and .vho files, respectively. For detailed instructions, see the CORE Generator software documentation.

Simulation Models

The core has two options for simulation models:

- VHDL UniSim-based structural simulation model
- Verilog UniSim-based structural simulation model

The models required may be selected in the CORE Generator software project options. No behavioral HDL model is provided for the core.

Xilinx recommends that simulations utilizing UniSim-based structural models be run using a resolution of 1 ps. Some Xilinx library components require a 1 ps resolution to work properly in either functional or timing simulation. The UniSim-based structural simulation models may produce incorrect results if simulated with a resolution other than 1 ps. See the "Register Transfer Level (RTL) Simulation Using Xilinx Libraries" section in Chapter 6 of the *Synthesis and Simulation Design Guide* for more information. This document is part of the ISE® Software Manuals set available at www.xilinx.com/support/documentation/dt_ise.htm.

General Information

The DUC/DDC Compiler provides coverage of a wide range of parameters that affect the design implementation while presenting a simple interface to the user; detailed knowledge of the internal workings of the core are generally unnecessary. The core handles configuration by querying a compiled database of highly optimal design configurations, which provides configuration information on appropriate rate change steps, filter configurations and numbers of data paths at each stage, mixer sample rate and configuration and flow control signal handling.

The basic static configuration parameter options for the core and their effects are detailed in the preceding GUI section, while the programming interface and registers define dynamic configuration options and their effects. Where some additional understanding of the core configuration and implementation is beneficial to ease integration of the core into an application system, further information is provided in this section.

The DUC/DDC Compiler makes extensive use of the FIR Compiler LogiCORE IP as a component core, including multiple instances of FIR filters in cascades to achieve the desired up- or down-conversion function. Familiarity with the data sheet for that core is advised to better understand some of the features of this core, particularly with regard to behavior under reset conditions. The implications of FIR filter reset behavior for overall DUC/DDC as a whole are described in "Filter Sample History Persistence."

Input Range

The input data format is simply a binary word of user-configurable width with arbitrary scaling (nominally the range is considered as being between -1.0 and 0.99999...)

A further restriction must be considered when using an Fs/4 IF mixer stage. In such cases, the input will be saturated at -0.999... to 0.999... to avoid any increase in integer bit representation, as the Fs/4 sampling operation utilizes a logical inversion stage, and therefore the data input signal must have a symmetrical range about zero. Saturation is performed within the core; however, users should be aware of this fact and may wish to scale input data appropriately to minimize the frequency and impact of that saturation operation.

Internal Range

The output of each filter stage and each mixer operation is rounded using a "Convergent to Even" rounding scheme throughout. Internal data bit width representation retains as much precision as is practical. Overflow is handled by wrapping.

Output Range and Scaling

Growth in the number of integer bits through the various filter stages and multi-carrier mixer accumulator is controlled to maintain a balance between likelihood of overflow and achieving sufficient precision to achieve the desired performance specified by the relevant air standard. The GUI provides a text field to indicate the output integer bits utilized by the current configuration. Generally, the core adds 1 additional integer bit for headroom and then the filter chain maintains unity gain as far as is practical; the main exception to this pattern is the case of multi-carrier DUCs, in which case the mixer accumulation leads to additional integer bits.

Rasterized DDS: Specification and Programming

The multi-carrier mixing function uses an efficient form of Direct Digital Synthesizer which is commonly referred to as a "rasterized" DDS, in that it can only produce frequencies which fall on a "raster" of fixed frequency values separated by a constant frequency difference (the "raster step".) Wireless air interface standards generally have a specified raster of frequencies at which carriers may be located; for LTE, the minimum raster is 200 kHz, while for TD-SCDMA it is 100 kHz. The raster step used in the DDS for this core depends on the desired mixing frequency, and is sometimes smaller than that specified in the wireless standard (the implementation raster step is in fact the greatest common divisor of the wireless standard raster and the mixing sample rate selected by the core.)

The frequency raster step is shown in the CORE Generator GUI in the Quantization section, see "Page 3". All carrier frequencies are multiples of this frequency raster. Quantization of carrier frequencies to make them multiples of the frequency raster is performed automatically by the CORE Generator GUI.

Users are highly recommended to restrict the range of carrier frequencies such that all carriers lie within the width of the IF Passband, centered at zero Hz. The filters in the core are designed for this range of carrier frequencies, and carriers outside this range will be strongly attenuated.

Filter Sample History Persistence

The DUC/DDC core contains a number of internal FIR filter stages. Each filter stage has a sample history pipeline that contributes to the overall sample latency of the core. The core does not clear the contents of these pipelines at initialization and after reset events; therefore consistent operation is only guaranteed once these pipelines have flushed with known sample values, which takes a number of samples equal to the total sample latency, or, in other words, the impulse response length of the full filter cascade. The `MDATA_CLEAN` signal denotes the period following reset and initialization during which time the sample history may be unknown, should the user require that information. For further information on `MDATA_CLEAN` timing behavior, see "Master Data Output Interface."

Control Signals and Timing

The DUC/DDC Compiler v1.1 core has five interfaces:

- system interface: clock and reset signals
- slave data input interface
- master data output interface
- programming interface: AMBA 3 APB slave
- interrupt interface: a set of interrupt output pins

System Interface

The system interface consists of a single clock and a single synchronous reset.

All interfaces and all internal logic use the same single clock, `CLK`.

The synchronous reset input `DATA_RESETn` is active low, and applies to the data path, the data input interface, the data output interface, and the interrupt interface. A second reset input, `SREG_PRESETn`, is provided for resetting the programming interface and registers: see the "Programming Interface" section. Both reset inputs are present only when “Optional Pins - Reset” is selected in the CORE Generator GUI.

Slave Data Input Interface

The slave data input interface is a data streaming interface that receives data samples that are input to the core. The core asserts `SDATA_READY` when it is ready to receive a data sample. `SDATA_VALID` input indicates that the input data is valid. Data is input into the core when both `SDATA_READY` and `SDATA_VALID` are asserted.

After reset, when the core is ready to receive the first input data sample, the core holds `SDATA_READY` high until `SDATA_VALID` goes high to indicate the first input data sample. Thereafter, the core expects input data to be provided at a steady rate, and it indicates the expected data rate by asserting `SDATA_READY`. If input data is not provided at the expected time (`SDATA_READY` is high but `SDATA_VALID` is low), then the core holds `SDATA_READY` high until `SDATA_VALID` goes high, and signals a missing input interrupt on the `INT_MISSINPUT` interrupt output. The core continues to process and output data that is in its data path while it waits for new data. There will be a corresponding pause in the output data rate when data in the core data path has been output from the core.

The core provides data input signals to match the configuration, as defined in Table 3.

Table 3: **Data input signals**

Core Type	Digital IF	Data Interface Format	Number of Antennas	Data signals	Description
DUC or DDC	0 Hz	Separate I and Q signals	1	SDATA_I	Complex data, in-phase portion
				SDATA_Q	Complex data, quadrature portion
			2-8	SDATA_I_Ax(1)	Complex data, in-phase portion
				SDATA_Q_Ax(1)	Complex data, quadrature portion
		Combined complex signal with I and Q TDM	1	SDATA_C	Complex data, TDM format: in-phase portion first, quadrature portion second
			2-8	SDATA_C_Ax(1)	Complex data, TDM format: in-phase portion first, quadrature portion second
DDC	Fs/4	n/a	1	SDATA_R	Real data
			2-8	SDATA_R_Ax(1)	Real data

1. Multiple signals, where x signifies the antenna number, from 0 to (Number of Antennas - 1.)

`SDATA_LAST` input indicates the last sample of an input packet: this is the input data sample for the last carrier (in a multi-carrier DUC), the quadrature portion if TDM format is used. The core expects input data to be provided for each carrier in ascending carrier order (in a multi-carrier DUC), and in-phase then quadrature data if TDM format is used. If `SDATA_LAST` is asserted for any sample other than the last sample of an input packet, or is not asserted for the last sample of an input packet, then the core ignores `SDATA_LAST` and signals a packet error interrupt on the `INT_ERRPACKET` interrupt output. It is not possible to change the input sample order using `SDATA_LAST`. A DDC or a single carrier DUC with Data Interface Format of separate I and Q signals has an input packet containing only one sample: in this configuration, `SDATA_LAST` is not present.

Timing Diagrams

Figure 7, Figure 8, Figure 9 and Figure 10 show timing diagrams for four configurations of the core:

- Figure 7: DDC with Digital IF of Fs/4, real data input, 1 antenna
- Figure 8: DDC with Digital IF of 0 Hz, TDM format, 1 antenna
- Figure 9: DUC with 3 carriers, separate I and Q format, 1 antenna
- Figure 10: DUC with 3 carriers, TDM format, 1 antenna

In each case, the input packet is 8 clock cycles in length. The effects of `SDATA_VALID` not being high when `SDATA_READY` is high causing a missing input interrupt, and incorrect `SDATA_LAST` (where present) causing a packet error interrupt, are shown towards the end of each timing diagram. Interrupt signals are asserted 2 clock cycles after the interrupt event.

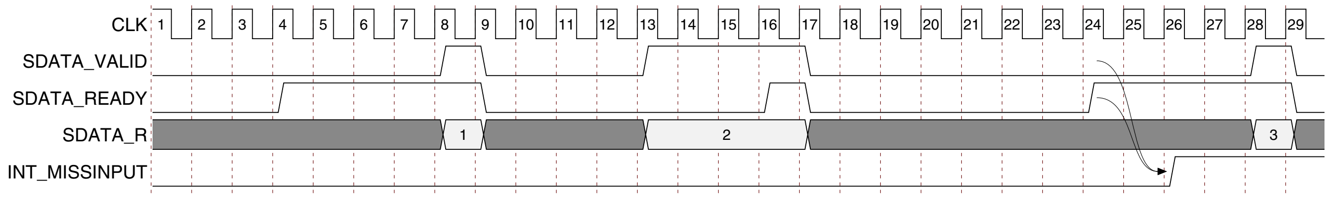


Figure 7: **Slave data input interface timing diagram: DDC, Fs/4, real data input, 1 antenna**

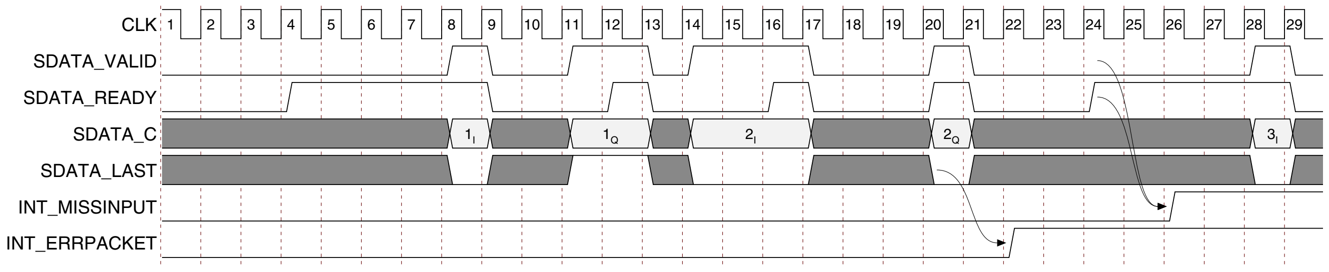


Figure 8: **Slave data input interface timing diagram: DDC, 0 Hz, TDM format, 1 antenna**

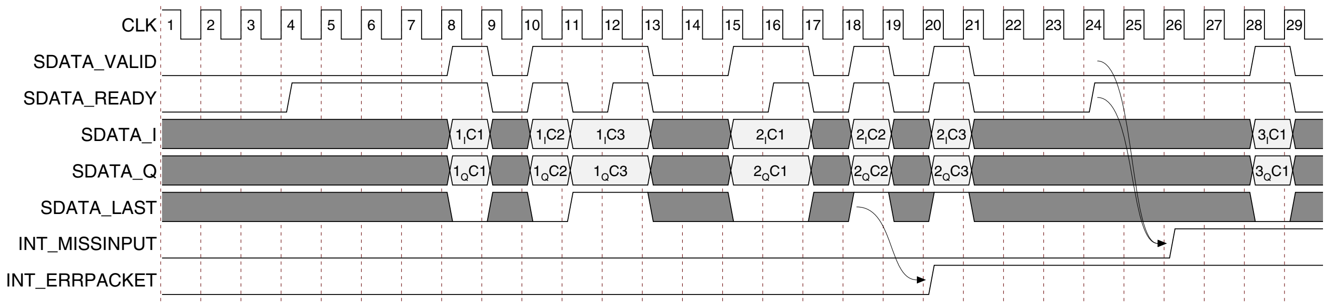


Figure 9: **Slave data input interface timing diagram: DUC, 3 carriers, separate I and Q format, 1 antenna**

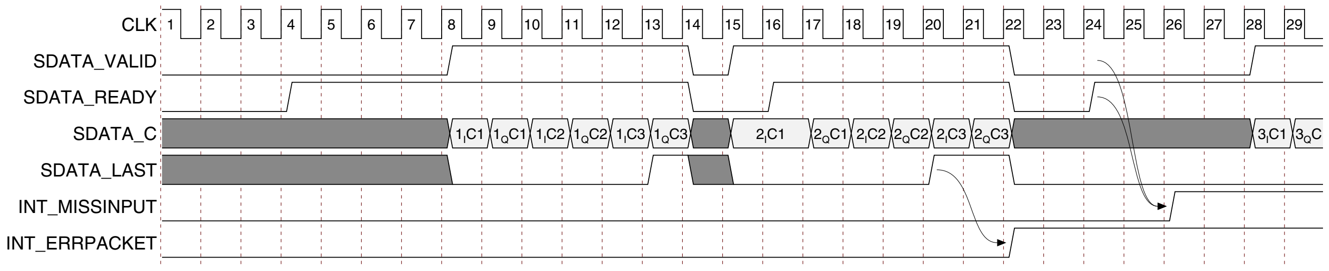


Figure 10: **Slave data input interface timing diagram: DUC, 3 carriers, TDM format, 1 antenna**

Master Data Output Interface

The master data output interface is a data streaming interface that outputs data samples from the core. The core asserts `MDATA_VALID` when new output data is valid and presented on the interface. The core holds `MDATA_VALID` high until `MDATA_READY` is asserted. Data is output from the core when both `MDATA_VALID` and `MDATA_READY` are asserted.

At reset and initialization, the core holds `MDATA_VALID` low until input data has been received and processed by the core data path, and is ready to be output. Thereafter, the core will attempt to output data at a steady rate, and it asserts `MDATA_VALID` when each output sample is available.

If input data was not provided at the slave data input interface when the core expected it, the core waits until input data is provided, but continues to process and output data that is in its data path. When all data samples in the data path have been output, the core holds `MDATA_VALID` low until new input data has been received and processed by the core data path, and is ready to be output. This will result in a pause in the steady rate of output data. The core indicates that this will happen by signalling a missing input interrupt on the `INT_MISSINPUT` interrupt output when the input data was not provided at the expected time.

The core cannot accept back pressure. It must produce output samples at the rate determined by the input data rate and the sample rate change performed by the core. If MDATA_VALID is high but MDATA_READY is held low, then any future output samples that the core generates will be internally discarded, and the core signals a lost output interrupt on the INT_LOSTOUTPUT interrupt output. Slaves are recommended to tie MDATA_READY high.

The core provides data output signals to match the configuration, as defined in Table 4.

Table 4: **Data output signals**

Data Interface Format	Number of Antennas	Data signals	Description
Separate I and Q signals	1	MDATA_I	Complex data, in-phase portion
		MDATA_Q	Complex data, quadrature portion
	2-8	MDATA_I_Ax(1)	Complex data, in-phase portion
		MDATA_Q_Ax(1)	Complex data, quadrature portion
Combined complex signal with I and Q TDM	1	MDATA_C	Complex data, TDM format: in-phase portion first, quadrature portion second
	2-8	MDATA_C_Ax(1)	Complex data, TDM format: in-phase portion first, quadrature portion second

1. Multiple signals, where x signifies the antenna number, from 0 to (Number of Antennas - 1.)

MDATA_LAST output indicates the last sample of an output packet: this is the output data sample for the last carrier (in a multi-carrier DDC), the quadrature portion if TDM format is used. A DUC or a single carrier DDC with Data Interface Format of separate I and Q signals has an output packet containing only one sample: in this configuration, MDATA_LAST is not present.

Timing Diagrams

Figure 11, Figure 12, Figure 13 and Figure 14 show timing diagrams for four configurations of the core:

- Figure 11: DUC with separate I and Q format, 1 antenna
- Figure 12: DUC with TDM format, 1 antenna
- Figure 13: DDC with 3 carriers, separate I and Q format, 1 antenna
- Figure 14: DDC with 3 carriers, TDM format, 1 antenna

In each case, the output packet is 8 clock cycles in length. The effect of MDATA_READY being held low for long enough to cause a lost output interrupt is shown towards the end of each timing diagram. Interrupt signals are asserted 2 clock cycles after the interrupt event.

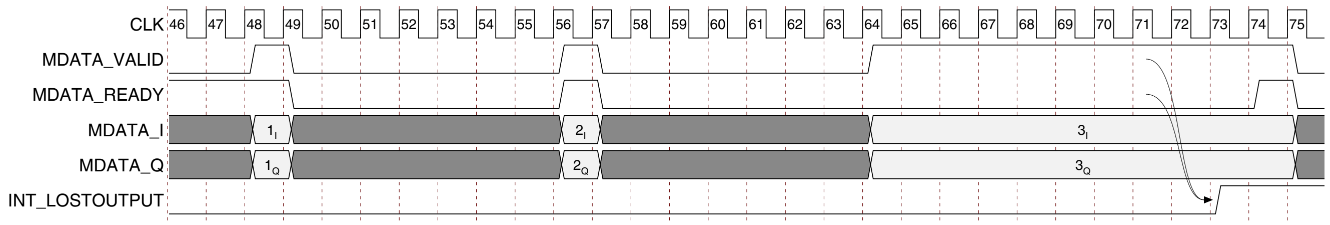


Figure 11: **Master data input interface timing diagram: DUC, separate I and Q format, 1 antenna**

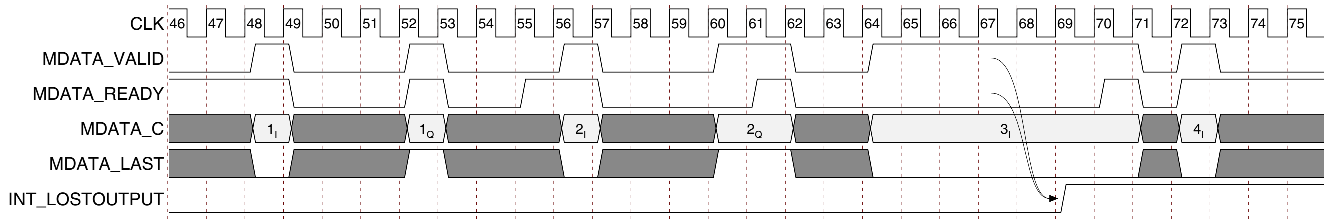


Figure 12: **Master data input interface timing diagram: DUC, TDM format, 1 antenna**

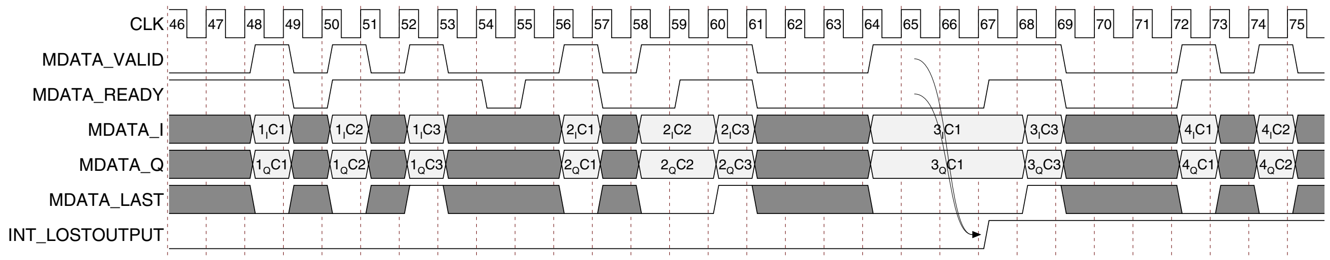


Figure 13: **Master data input interface timing diagram: DDC, 3 carriers, separate I and Q format, 1 antenna**

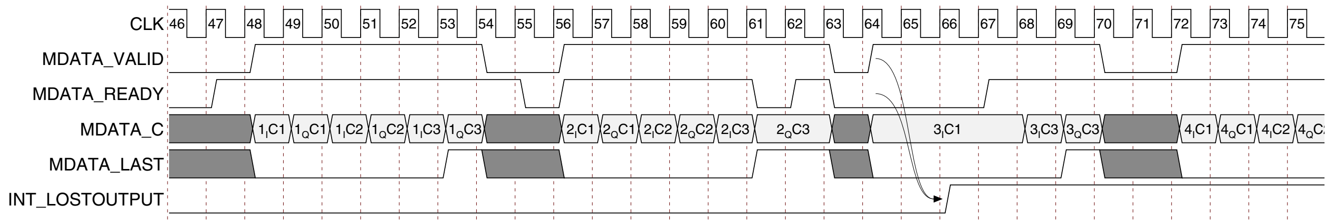


Figure 14: **Master data input interface timing diagram: DDC, 3 carriers, TDM format, 1 antenna**

Programming Interface

The programming interface is an AMBA 3 APB slave interface for programming carrier frequencies and carrier gain control for providing configuration, status and error information about the core. The programming interface is always present.

The programming interface complies with the AMBA 3 APB interface specification [Ref 7]. See this specification for detailed information about the interface.

The core uses SREG_PREADY to insert wait states to extend APB transactions. All transactions on the programming interface use at least 3 wait states. Reads and writes to "Frequency Programming Registers" and "Gain Control Programming Registers" sometimes use more wait states, up to a maximum of 9.

SREG_PRESETn is the reset signal for the programming interface and registers. This reset signal is synchronous and active low. SREG_PRESETn is registered internally to aid timing closure, and takes effect one cycle after it is synchronously asserted.

Interrupt Interface

The interrupt interface is a set of interrupt output pins, each corresponding to a particular interrupt type, plus one combined interrupt output pin that indicates an interrupt of any type:

- `INT_MISSINPUT` indicates a missing input interrupt, see "Slave Data Input Interface" for details and timing diagrams.
- `INT_ERRPACKET` indicates an input packet length error interrupt, see "Slave Data Input Interface" for details and timing diagrams.
- `INT_LOSTOUTPUT` indicates a lost output interrupt, see "Master Data Output Interface" for details and timing diagrams.
- `INT_DUCDDC` indicates an interrupt of any type, and is the logical OR of the above three interrupt signals.

Interrupt signals are asserted 2 clock cycles after the corresponding interrupt event.

All interrupt outputs are always present. Each interrupt type can be independently enabled or disabled using the "Interrupt Enable Register". When an interrupt is disabled, the corresponding interrupt signal is held low at all times, whether an interrupt has occurred or not. `INT_DUCDDC` cannot be independently enabled or disabled: it is the logical OR of the other three interrupt signals, and therefore takes into account the individual interrupt enables.

The current status of interrupts and interrupt signals is also available in the "Raw Interrupt Status Register" and "Masked Interrupt Status Register" respectively.

All interrupt outputs are sticky, and once they go high they will stay high until disabled by writing to the "Interrupt Enable Register", or cleared by writing to the "Interrupt Clear Register" or resetting the core by asserting `DATA_RESETn`.

Programming Interface Registers

Register Map

For compatibility with SoCs that use APB to communicate with a number of peripherals, the DUC/DDC Compiler register map is limited to a 4KB (12 bit) address space, and the address bus, `SREG_PADDR`, is 12 bits wide.

The DUC/DDC Compiler's register map is shown in Figure 15.

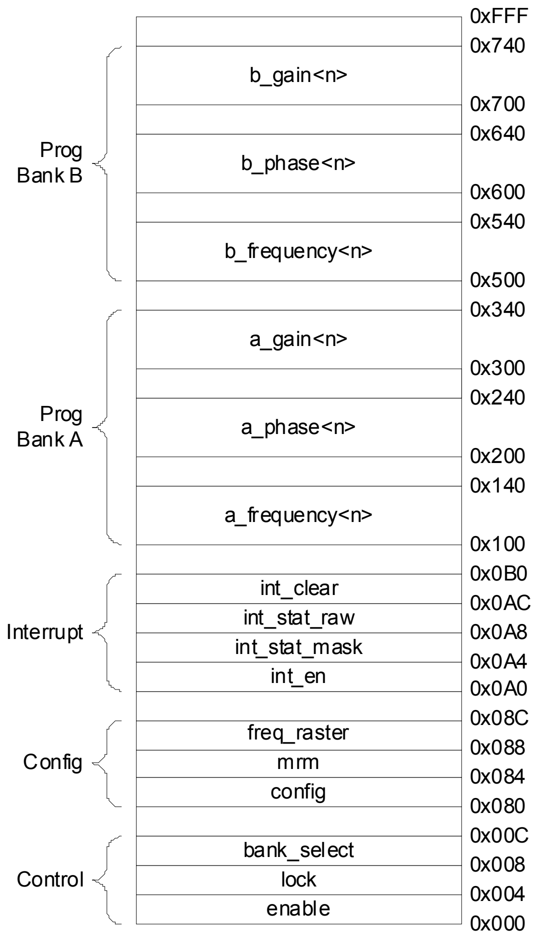


Figure 15: **Register Map**

Register Definitions

The registers are summarized in Table 5, and described in detail in the following sections. All registers are 32 bits wide.

Table 5: **Registers**

Address	Label	Type	Init/Reset	Name
0x000	lock	RW	0x00000000	Lock Register
0x004	bank_select	RW	0x00000000(1)	Programming Bank Selection Register
0x008-0x07C	-	-	-	Reserved
0x080	config	RO	(2)	Configuration Register
0x084	mrm	RO	(2)	Mixing Rate Multiple Register
0x088	freq_raster	RO	(2)	Frequency Raster Register
0x08C-0x09C	-	-	-	Reserved
0x0A0	int_en	RW	0x00000007	Interrupt Enable Register
0x0A4	int_stat_mask	RO	0x00000000	Masked Interrupt Status Register
0x0A8	int_stat_raw	RO	0x00000000	Raw Interrupt Status Register
0x0AC	int_clear	WO	-	Interrupt Clear Register
0x0B0-0x0FC	-	-	-	Reserved
0x100-0x144	a_frequency<*n*>	RW	(1)(3)	Frequency Programming Registers, bank A, <*n*> = 1 to Number of Carriers
0x148-0x1FC	-	-	-	Reserved
0x200-0x244	a_phase<*n*>	RO	(1)(4)	Phase Offset Programming Registers, bank A, <*n*> = 1 to Number of Carriers
0x248-0x2FC	-	-	-	Reserved
0x300-0x344	a_gain<*n*>	RW	0x00010000(1)	Gain Control Programming Registers, bank A, <*n*> = 1 to Number of Carriers
0x348-0x4FC	-	-	-	Reserved
0x500-0x544	b_frequency<*n*>	RW	(1)(3)	Frequency Programming Registers, bank B, <*n*> = 1 to Number of Carriers
0x548-0x5FC	-	-	-	Reserved
0x600-0x644	b_phase<*n*>	RO	(1)(4)	Phase Offset Programming Registers, bank B, <*n*> = 1 to Number of Carriers
0x648-0x6FC	-	-	-	Reserved
0x700-0x744	b_gain<*n*>	RW	0x00010000(1)	Gain Control Programming Registers, bank B, <*n*> = 1 to Number of Carriers
0x748-0xFFC	-	-	-	Reserved

1. Register is not reset by SREG_PRESETn input.
2. Initial value is calculated by the core based on several parameters.
3. Initial value of each register <*n*> is the quantized carrier frequency for carrier <*n*> divided by the frequency raster.
4. Initial value of each register <*n*> is the quantized carrier phase offset for carrier <*n*> divided by the phase raster.

If Programmable Carrier Frequencies is enabled, the Frequency Programming Registers are read/write; otherwise these registers are read only and an attempt to write them results in a SLVERR response.

If Programmable Carrier Gain Control is enabled, the Gain Control Programming Registers are read/write; otherwise they are read only and an attempt to write them results in a SLVERR response.

Lock Register

The Lock Register is a 32-bit read/write register at address 0x000 that enables or disables write access to all other registers accessible through the programming interface. The format of this register is shown in Figure 16.

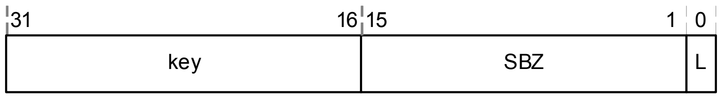


Figure 16: **Lock Register**

The register bits are shown in Table 6.

Table 6: **Lock Register Bits**

Bit	Field	Type	Init/Reset	Description
[31:16]	key	WO	-	Key: write only, Read As Zero. To lock, write the value 0xC705 to this field. To unlock, write the value 0x5E5A to this field.
[15:1]	-	-	0	Should Be Zero
0	L	RO	0	Lock: read only, ignored on writes. 0 = unlocked: programming interface writes to registers will proceed as normal. 1 = locked: all programming interface writes except writes to the Lock Register will fail with a SLVERR response.

Writes to the Lock Register are not affected by the L bit. Writes with an incorrect key (neither the lock nor unlock key) are silently ignored and will not affect the L bit. This register is reset by the `SREG_PRESETn` reset input.

Programming Bank Selection Register

The Programming Bank Selection Register is a 32-bit read/write register at address 0x004 that selects the bank of programming registers to be active. The format of this register is shown in Figure 17.

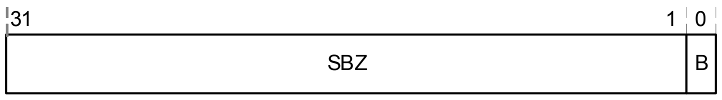


Figure 17: **Programming Bank Selection Register**

The register bits are shown in Table 7.

Table 7: **Programming Bank Selection Register Bits**

Bit	Field	Type	Init	Description
[31:1]	-	-	0	Should Be Zero
0	B	RW	0	Bank selection: 0 = programming registers bank A is active, bank B is shadowing. 1 = programming registers bank B is active, bank A is shadowing.

There are two banks of programming registers – bank A and bank B. Each bank of registers contains a complete set of frequency, phase offset, and gain programming registers (see "Frequency Programming Registers," "Phase Offset Programming Registers," and "Gain Control Programming Registers," respectively.) At any time, one bank of programming registers is active and the other bank is shadowing. The active bank is used by the core data path, and is read only; writes to registers in the active bank result in a `SLVERR` response. The shadowing bank is not used by the core data path, and is read/write, so can be used for programming new frequency, phase offset, and gain values. When the new values are correctly programmed in the shadowing bank, write the Programming Bank Selection Register to change the B bit, and the new values become the active bank and are used by the core data path. This register is not reset by the `SREG_PRESETn` reset input.

Any write to the Programming Bank Selection Register, whether the value of the B bit is changed or not, forces an internal core data path reset, as if DATA_RESETn had been asserted. This is required to allow the DDS and mixer to start using the new programmed frequency, phase offset and gain values in a consistent and predictable manner. The MDATA_VALID signal goes low to indicate that the core has been reset, and will remain low until new input data has been processed and has propagated through the core. The internal data path reset does not clear the contents of internal sample history pipelines. See "Filter Sample History Persistence" for the implications of this on output data values.

Configuration Register

The Configuration Register is a 32-bit read only register at address 0x080 that shows the value of a number of key core parameters. The format of this register is shown in Figure 18.

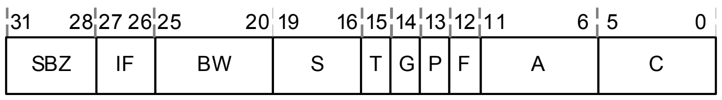


Figure 18: **Configuration Register**

The register bits are shown in Table 8.

Table 8: **Configuration Register Bits**

Bit	Field	Type	Description
[31:28]	-	-	Should Be Zero
[27:26]	IF	RO	Digital IF: 00: 0 Hz 01: Fs/4 All other values are reserved.
[25:20]	BW	RO	Channel bandwidth option, also depends on S field. See Table 9 for details.
[19:16]	S	RO	Wireless standard: 0000: LTE 0001: TD-SCDMA All other values are reserved.
15	T	RO	Core type: 0: DUC 1: DDC
14	G	RO	Programmable gain control: 0 = no programmable gain control 1 = programmable gain control present
13	P	RO	Programmable carrier phase offsets. Reserved for future use; always reads as zero.
12	F	RO	Programmable carrier frequencies: 0 = fixed carrier frequencies 1 = programmable carrier frequencies
[11:6]	R	RO	Number of antennas. Possible values are 1 to 8; all other values are reserved.
[5:0]	C	RO	Number of carriers. Possible values are 1 to 18; all other values are reserved.

The channel bandwidth is determined from the S and BW fields as shown in Table 9. All combinations of S and BW not shown are reserved.

Table 9: **Channel Bandwidth**

S	Wireless Standard	BW	Channel Bandwidth
0000	LTE	000001	1.4 MHz
		000011	3 MHz
		000101	5 MHz
		001010	10 MHz
		001111	15 MHz
		010100	20 MHz
0001	TD-SCDMA	000010	1.6 MHz

The C field, which reports the number of carriers, is expected to be the most used, to allow driver software to generate programmable carrier frequencies and gains. Therefore this field is in the LSBs of the register so that a register read and a single `AND` **`0x3F`** software instruction can return the number of carriers.

The value of this register is constant and does not change during run time.

Mixing Rate Multiple Register

The Mixing Rate Multiple Register is a 32-bit read only register at address 0x084 that indicates the ratio between the mixing sample rate and the frequency raster. It shows the number of possible carrier positions for both carrier frequency and carrier phase offset. The format of this register is shown in Figure 19.

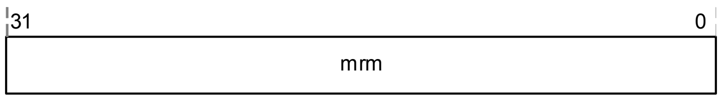


Figure 19: **Mixing Rate Multiple Register**

The register bits are shown in Table 10.

Table 10: **Mixing Rate Multiple Register Bits**

Bit	Field	Type	Description
[31:0]	mrm	RO	Mixing Rate Multiple, unsigned integer value

The mixing rate multiple (mrm) indicates the number of possible values for carrier frequency and carrier phase offset. Carrier frequency values in "Frequency Programming Registers" are in the range -mrm/2 to mrm/2-1. Carrier phase offset values in "Phase Offset Programming Registers" are in the range 0 to mrm-1.

The phase raster can be calculated from the mixing rate multiple using the following formula:

$$\text{phase raster (radians)} = 2\pi / \text{mrm}$$

The sample rate at which multi-carrier mixing occurs can be calculated from the mixing rate multiple and the frequency raster (see "Frequency Raster Register") using the following formula:

$$\text{mixing sample rate in Hz} = \text{frequency raster in Hz} \times \text{mrm}$$

The value of this register is constant and does not change during run time.

Frequency Raster Register

The Frequency Raster Register is a 32-bit read only register at address 0x088 that indicates the frequency raster in Hz for the implemented wireless standard. The format of this register is shown in Figure 20.

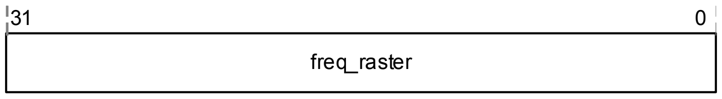


Figure 20: **Frequency Raster Register**

The register bits are shown in Table 11.

Table 11: **Frequency Raster Register Bits**

Bit	Field	Type	Description
[31:0]	freq_raster	RO	Frequency Raster in Hz, unsigned integer value

See "Rasterized DDS: Specification and Programming" for details of the frequency raster.

The value of this register is constant and does not change during run time.

Interrupt Registers

There are four interrupt registers:

- "Interrupt Enable Register"
- "Masked Interrupt Status Register"
- "Raw Interrupt Status Register"
- "Interrupt Clear Register"

All four registers have the same format, which is shown in Figure 21.

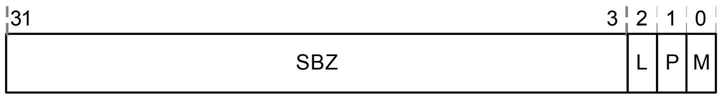


Figure 21: **Interrupt Registers**

Each field in the interrupt registers corresponds to an interrupt type, as shown in Table 12.

Table 12: **Interrupt Register Bits**

Bit	Field	Description
[31:3]	-	Should Be Zero
2	L	Lost output sample interrupt
1	P	Input packet length error interrupt
0	M	Missing input sample interrupt

Interrupt Enable Register

The Interrupt Enable Register is a 32-bit read/write register at address 0x0A0 that enables or disables interrupts. The format and register bits are shown in Figure 21 and Table 12.

When a bit in the Interrupt Enable Register is set, the interrupt for that bit is enabled, and an interrupt shown in the Raw Interrupt Status Register is also signalled by the corresponding interrupt output going high. Clearing a bit disables the interrupt for that bit, and the corresponding interrupt is masked (the interrupt output is held low) regardless of the interrupt status.

All interrupts are enabled (all bits corresponding to interrupts are 1) at reset. This register is reset by the `SREG_PRESETn` reset input.

Masked Interrupt Status Register

The Masked Interrupt Status Register is a 32-bit read only register at address 0x0A4 that provides the interrupt status taking into account interrupt enabling. This is the `AND` of the Raw Interrupt Status Register and the Interrupt Enable Register. The Masked Interrupt Status Register directly indicates the status of the interrupt output pins. The format and register bits are shown in Figure 21 and Table 12.

When a bit in the Masked Interrupt Status Register is high, the interrupt for that bit is triggered and enabled. When a bit is low, the interrupt for that bit is either not triggered or is not enabled, and so has been masked.

All bits are 0 initially. This register is not reset by the `SREG_PRESETn` reset input. Interrupts are reset (cleared) by the `DATA_RESETn` reset input, and the Interrupt Enable Register is reset by the `SREG_PRESETn` reset input; therefore this register may change value on either reset.

Raw Interrupt Status Register

The Raw Interrupt Status Register is a 32-bit read only register at address 0x0A8 that provides the interrupt status ignoring interrupt enabling. The Raw Interrupt Status Register indicates the status of interrupts from the core before masking. The Raw Interrupt Status Register may differ from the status of the interrupt output pins if one or more interrupts are disabled using the Interrupt Enable Register. The format and register bits are shown in Figure 21 and Table 12.

When a bit in the Raw Interrupt Status Register is high, the interrupt for that bit is triggered. When a bit is low, the interrupt for that bit is not triggered.

All bits are 0 initially. This register is not reset by the `SREG_PRESETn` reset input. Interrupts are reset (cleared) by the `DATA_RESETn` reset input; therefore this register may change value on `DATA_RESETn`.

Interrupt Clear Register

The Interrupt Clear Register is a 32-bit write only register at address 0x0AC for clearing interrupts. The format and register bits are shown in Figure 21 and Table 12.

Writing 1 to a bit in the Interrupt Clear Register clears the corresponding bit in the Raw Interrupt Status Register, thereby clearing the interrupt and setting the corresponding interrupt output pin low. Writing 0 to a bit has no effect.

Frequency Programming Registers

The Frequency Programming Registers are two banks, bank A and bank B, of C 32-bit read/write registers, where C is the number of carriers (reported in the C field of the "Configuration Register".) The registers in bank A are at sequential word addresses starting at address 0x100; the registers in bank B are at sequential word addresses starting at address 0x500. The a_frequency<*n*> register for carrier *n* is at address (0x100 + 4 × (*n* - 1)), and the b_frequency<*n*> register for carrier *n* is at address (0x500 + 4 × (*n* - 1)). Each a_frequency<*n*> and b_frequency<*n*> register holds the frequency for carrier *n* as a multiple of the frequency raster, given by the "Frequency Raster Register."

The format of each a_frequency<*n*> and b_frequency<*n*> register is shown in Figure 22.

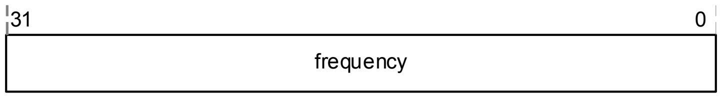


Figure 22: **a_frequency<*n*> and b_frequency<*n*> Registers**

The register bits are shown in Table 13.

Table 13: **a_frequency<*n*> and b_frequency<*n*> Register Bits**

Bit	Field	Type	Init	Description
[31:0]	frequency	RW	(1)	Carrier *n* frequency as a multiple of the frequency raster. 2's complement integer in the range -*mrm*/2 to *mrm*/2-1, where *mrm* is the mixing rate multiple, see "Mixing Rate Multiple Register."

1. Initial value of each register <*n*> is the quantized carrier frequency for carrier <*n*> divided by the frequency raster.

Software drivers can calculate the correct value to write to a Frequency Programming Register using the required frequency in Hz and the frequency raster; see "Frequency Raster Register."

frequency programming value = required frequency in Hz / frequency raster in Hz

The "Programming Bank Selection Register" selects which bank of Frequency Programming Registers is active (used by the core data path) and which is shadowing (not used by the core data path but available for programming.) Frequency Programming Registers in the active bank are read only – an attempt to write an active bank Frequency Programming Register results in a `SLVERR` response. Frequency Programming Registers in the shadowing bank are read/write if Programmable Carrier Frequencies is enabled, and read only otherwise. Values written to Frequency Programming Registers in the shadowing bank are not used by the core data path until the Programming Bank Selection Register is modified to swap the active and shadowing banks.

An attempt to write a Frequency Programming Register with a value that is out of the legal range results in a `SLVERR` response and does not change the register value. An attempt to read or write a Frequency Programming Register that does not exist (that is, where *n* is greater than the number of carriers) results in a `SLVERR` response.

For a single carrier, the single Frequency Programming Register in each bank is read only and its value is set to zero.

These registers are *not* reset by the `SREG_PRESETn` reset input.

Phase Offset Programming Registers

The Phase Offset Programming Registers are two banks, bank A and bank B, of C 32-bit read only registers, where C is the number of carriers (reported in the C field of the "Configuration Register".) The registers in bank A are at sequential word addresses starting at address 0x200; the registers in bank B are at sequential word addresses starting at address 0x600. The a_phase<*n*> register for carrier *n* is at address (0x200 + 4 × (*n* - 1)), and the b_phase<*n*> register for carrier *n* is at address (0x600 + 4 × (*n* - 1)). Each a_phase<*n*> and b_phase<*n*> register holds the phase offset for carrier *n* as a multiple of the phase raster, which can be calculated from the "Mixing Rate Multiple Register."

The format of each a_phase<*n*> and b_phase<*n*> register is shown in Figure 23.

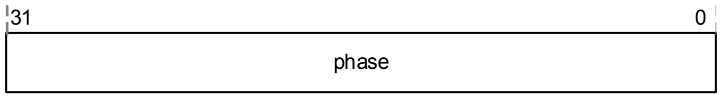


Figure 23: **a_phase<*n*> and b_phase<*n*> Registers**

The register bits are shown in Table 14.

Table 14: **a_phase<*n*> and b_phase<*n*> Register Bits**

Bit	Field	Type	Init	Description
[31:0]	phase	RO	(1)	Carrier *n* phase offset as a multiple of the phase raster. Unsigned integer in the range 0 to *mrm*-1, where *mrm* is the mixing rate multiple, see "Mixing Rate Multiple Register."

1. Initial value of each register <*n*> is the quantized carrier phase offset for carrier <*n*> divided by the phase raster.

The value of a Phase Offset Programming Register is calculated from the corresponding carrier phase offset in radians and the phase raster, which is derived from the mixing rate multiple, see "Mixing Rate Multiple Register."

Phase Offset Programming Register value = carrier phase offset in radians / phase raster in radians

All Phase Offset Programming Registers in both bank A and bank B are read only. Carrier phase offsets cannot be changed at run time.

Gain Control Programming Registers

The Gain Control Programming Registers are two banks, bank A and bank B, of C 32-bit read/write registers, where C is the number of carriers (reported in the C field of the "Configuration Register".) The registers in bank A are at sequential word addresses starting at address 0x300; the registers in bank B are at sequential word addresses starting at address 0x700. The a_gain<*n*> register for carrier *n* is at address (0x300 + 4 × (*n* - 1)), and the b_gain<*n*> register for carrier *n* is at address (0x700 + 4 × (*n* - 1)). Each a_gain<*n*> and b_gain<*n*> register holds the gain for carrier *n*.

The format of each a_gain<*n*> and b_gain<*n*> register is shown in Figure 24.

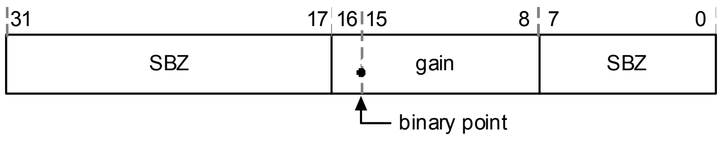


Figure 24: **a_gain<*n*> and b_gain<*n*> registers**

The register bits are shown in Table 15.

Table 15: **a_gain<*n*> and b_gain<*n*> Register Bits**

Bit	Field	Type	Init	Description
[31:17]	-	-	0	Should Be Zero
[16:8]	gain	RW	100000000 (that is,1.0)	Carrier *n* gain. Unsigned fixed point value in the range 0.0 to 1.0. The binary point is fixed, between bits 16 and 15. Only power of 2 values are allowed, that is, gain field must be one-hot or all zeros.

Gain control allows the relative amplitude of carriers to be adjusted, for example for power management, by attenuating one or more carriers. Each carrier's amplitude is attenuated by the programmed value. For example, writing a gain value of 001000000 to carrier 1 Gain Control Programming Register corresponds to a gain of 0.25, and all data values for carrier 1 will be multiplied by 0.25 prior to multi-carrier mixing.

The "Programming Bank Selection Register" selects which bank of Gain Control Programming Registers is active (used by the core data path) and which is shadowing (not used by the core data path but available for programming.) Gain Control Programming Registers in the active bank are read only – an attempt to write an active bank Gain Control Programming Register results in a SLVERR response. Gain Control Programming Registers in the shadowing bank are read/write if Programmable Carrier Gain Control is enabled, and read only otherwise. Values written to Gain Control Programming Registers in the shadowing bank are not used by the core data path until the Programming Bank Selection Register is modified to swap the active and shadowing banks.

An attempt to write a Gain Control Programming Register with an illegal value or with a value that is out of the legal range results in a SLVERR response and does not change the register value. An attempt to read or write a Gain Control Programming Register that does not exist (that is, where *n* is greater than the number of carriers) results in a SLVERR response.

For a single carrier, the single Gain Control Programming Register in each bank is read only and its value is set to 100000000 (that is, 1.0).

Gain control is not available for DDCs: all Gain Control Programming registers are read only in DDCs.

These registers are *not* reset by the SREG_PRESETn reset input.

Performance and Resource Utilization

Resource requirements and performance are dependent on a wide range of core parameters, but mainly on these factors: difference in input and output sample rates; number of carriers; and number of antennas. The last of these options has the largest impact, as the resource usage of the core is almost linear with respect to this value (actually slightly less due to certain resource sharing between antenna data paths and a single programming interface and register set.)

Table 16, Table 17, and Table 18 provide resource requirements for Virtex-6, Virtex-5, and Spartan-6 devices, respectively. For all configurations, the default data width of 16-bits is used at both data input and output interfaces.

The results are obtained by double-registering input and output ports to reduce dependence on I/O placement. The inner level of registers use a separate clock signal to measure the path from the input registers to the first output register through the core.

The resource usage results do not include these double registers on inputs and outputs, and represent the true logic used by the core to implement a single instance. LUT counts include SRL16s or SRL32s (according to device family.)

Each configuration is constrained to achieve the target clock frequency, including 300 ps total peak-to-peak clock jitter margin (sufficient for most clock sources used in FPGAs.)

Results are obtained using Xilinx ISE Design Suite 12.2.

The map options used are: `map -pr b -ol high`

The par options used are: `par -t 1 -ol high`

The achievable clock frequency and the resource counts may also be affected by other tool options, routing congestion due to additional logic in the FPGA device, using a different version of Xilinx tools, and other factors. No advanced constraint usage is employed in achieving these results. Area group constraints and other enhanced placement techniques may improve performance and further reduce area, or allow the use of a lower speed grade.

To aid timing closure, inputs are immediately registered inside the core where possible, and all outputs are driven directly from registers. The handshaking protocol in the slave data input interface requires `SDATA_VALID` to be used in combinatorial logic without registering it, so this input needs additional timing slack.

Core Optimization: Area vs. Speed

The DUC/DDC Compiler can generate a wide range of core configurations targeted at three different device families with a number of sub-types and speed grades. Unfortunately, it is not possible to characterize the performance and resource usage of all permutations. Consequently, the performance tables below are only an indication. The core includes some implementation options to allow users to tailor the core for their device or application.

The selection of “Area” or “Speed” for the Optimization Goal option is an additional measure for customers who experience difficulty in achieving the desired timing for the core in their device. Normally, the goal should be kept as “Area”, the default setting, as this normally achieves good timing results. However, there are certain combinations of device and core configuration that may result in the core failing to achieve the timing goal without intervention. If the device resource levels allow, the “Speed” setting for Optimization Goal adds additional logic resources to try to improve the chances of achieving the desired timing in the target device.

Guidance on Suitable Device Selection

One significant factor in achieving the desired timing performance is the availability of DSP slices and DSP slice column separation within the targeted device. This is due to the use of filter modules within the DUC or DDC architectures since filters make extensive use of cascaded DSP slices to implement efficient filter structures. The SX devices have a high ratio of DSP slices to logic, and many columns with low separation distances. This is in contrast to other Virtex devices (for example, LX and FX) that contain a lower ratio of DSP slices to logic, and fewer columns with greater separation distances. For large DUC or DDC designs, there is a greater chance that the core resources are spread over multiple DSP slice columns. As a result, it is easier to achieve timing closure in SX devices than LX devices due to the shorter distances between columns.

Customers who attempt to implement a larger core in an LX device and cannot achieve the desired core timing have some options for alleviating the timing problem. One option is to set the Optimization Goal option to “Speed”, which adds some additional logic targeted at increasing the chances of achieving timing closure, at the expense of increased resource usage. Another option is to implement fewer antennas in a single core and use multiple core instances to create all the antenna data paths. Each core instance can then be constrained by an Area Group to achieve the desired timing. The only disadvantage in splitting up the antennas across a number of core implementations is duplication of the resources used for the DDS module and APB peripheral bus interface. However, these are generally a relatively small portion of the overall resources for each core.

Resource Utilization (Virtex-6)

Table 16 provides characterization data for Virtex-6 FPGAs using an XC6VSX475T-1FF1759, with similar results achieved using an XC6VSX240T-1FF1759. The common 368.64 MHz clock rate is used throughout these examples, but lower clock rate designs will also meet their timing goals, although with increased resources. Low clock rate designs are significantly larger due to a reduction in resource time sharing on DSP slices, which in turn have a detrimental effect on timing (net delays due to long interconnect lines and congestion will increase.)

Block RAM counts listed are for 18K blocks, which are often amalgamated into pairs for mapping to 36K locations where possible; it is important to note this when comparing these values with map results for your configuration.

Table 16: **DUC/DDC Compiler Resource Utilization in Virtex-6 FPGAs**

Wireless Air Standard	DUC / DDC	Channel BW (MHz)	Carriers	Antennas	Passband (MHz)	RF Sample Rate (Msps)	Fs/4 Digital IF	Clock Frequency (MHz)	Optimize for Area or Speed (A/S)	Maximize Block RAMs	Frequency and Gain Programming	DSP48E1s	18K Block RAMs	Slices
LTE	DUC	5	4	1	20	122.88	N	368.64	A	N	N	18	1	896
LTE	DDC	5	4	1	20	122.88	N	368.64	A	N	N	20	1	1197
LTE	DDC	5	4	1	20	122.88	N	368.64	A	Y	Y	20	37	713
LTE	DUC	5	4	2	20	122.88	N	368.64	A	N	N	36	1	1667
LTE	DDC	5	4	2	20	122.88	N	368.64	A	N	N	40	1	2241
LTE	DUC	5	4	4	20	122.88	N	368.64	A	N	N	72	1	3208
LTE	DUC	5	4	4	20	122.88	N	368.64	A	Y	N	72	109	1925
LTE	DDC	5	4	4	20	122.88	N	368.64	A	N	N	80	1	4325
LTE	DUC	10	2	1	20	122.88	N	368.64	A	N	N	15	1	593
LTE	DDC	10	2	4	20	122.88	N	368.64	A	N	N	64	1	2480
LTE	DUC	10	5	1	50	184.32	N	368.64	A	N	N	52	3	1084
LTE	DDC	10	5	1	50	184.32	N	368.64	A	N	N	58	1	1133
LTE	DDC	10	5	1	50	184.32	Y	368.64	A	N	N	58	1	1197
LTE	DUC	10	5	4	50	184.32	N	368.64	A	N	N	208	6	3855
LTE	DDC	10	5	4	50	184.32	N	368.64	A	N	N	232	6	4523
LTE	DUC	20	1	4	20	122.88	N	368.64	A	N	N	44	0	1333
LTE	DDC	20	1	4	20	122.88	N	368.64	A	N	N	56	0	1808
LTE	DUC	20	1	8	20	122.88	N	368.64	A	N	N	88	0	2596
LTE	DDC	20	1	8	20	122.88	N	368.64	A	N	N	112	0	3472
LTE	DUC	20	2	4	40	122.88	N	368.64	A	N	N	100	2	2046
LTE	DDC	20	2	4	40	122.88	N	368.64	A	N	N	112	1	2608
LTE	DUC	20	2	4	40	184.32	N	368.64	A	N	N	108	2	2182
LTE	DDC	20	2	1	40	184.32	N	368.64	A	N	N	29	1	763
TD-SCDMA	DUC	1.6	9	1	15	92.16	N	368.64	A	N	N	23	11	749

Table 16: **DUC/DDC Compiler Resource Utilization in Virtex-6 FPGAs *(Cont'd)***

Wireless Air Standard	DUC / DDC	Channel BW (MHz)	Carriers	Antennas	Passband (MHz)	RF Sample Rate (Msps)	Fs/4 Digital IF	Clock Frequency (MHz)	Optimize for Area or Speed (A/S)	Maximize Block RAMs	Frequency and Gain Programming	DSP48E1s	18K Block RAMs	Slices
TD-SCDMA	DUC	1.6	9	1	15	92.16	N	368.64	A	N	Y	23	11	875
TD-SCDMA	DDC	1.6	9	1	15	92.16	N	368.64	A	N	N	19	13	763
TD-SCDMA	DUC	1.6	9	4	15	92.16	N	368.64	A	N	N	92	38	2535
TD-SCDMA	DDC	1.6	9	4	15	92.16	N	368.64	A	N	N	76	46	2745
TD-SCDMA	DUC	1.6	12	1	20	92.16	N	368.64	A	Y	N	26	27	685
TD-SCDMA	DDC	1.6	12	1	20	92.16	N	368.64	A	N	N	22	15	871
TD-SCDMA	DUC	1.6	12	2	20	92.16	N	368.64	A	N	N	52	26	1371
TD-SCDMA	DDC	1.6	12	2	20	92.16	N	368.64	A	N	N	44	28	1588
TD-SCDMA	DUC	1.6	12	4	20	92.16	N	368.64	A	N	N	104	50	2757
TD-SCDMA	DDC	1.6	12	4	20	92.16	N	368.64	A	N	N	80	1	4390

Notes:

1. Clock frequency determined using a -1 speed grade and 300ps clock jitter allowance

Resource Utilization (Virtex-5)

Table 17 provides characterization data for Virtex-5 FPGAs using an XC5VSX240T-2FF1738. Block RAM counts listed are for 18K blocks, which are often amalgamated into pairs for mapping to 36K locations where possible; it is important to note this when comparing these values with map results for your configuration.

Table 17: **DUC/DDC Compiler Resource Utilization in Virtex-5 FPGAs**

Wireless Air Standard	DUC / DDC	Channel BW (MHz)	Carriers	Antennas	Passband (MHz)	RF Sample Rate (Msps)	Fs/4 Digital IF	Clock Frequency (MHz)	Optimize for Area or Speed (A/S)	Maximize Block RAMs	DSP48Es	18K Block RAMs	Slices
LTE	DUC	5	4	1	20	122.88	N	368.64	A	N	18	1	1284
LTE	DDC	5	4	1	20	122.88	N	368.64	A	N	20	1	1527
LTE	DDC	5	4	1	20	122.88	Y	368.64	A	N	19	1	1496
LTE	DUC	5	4	4	20	122.88	N	368.64	A	N	72	1	4571
LTE	DDC	5	4	4	20	122.88	N	368.64	A	N	80	1	6127
LTE	DUC	10	2	1	20	122.88	N	368.64	A	N	15	1	1026
LTE	DDC	10	2	1	20	122.88	N	368.64	A	N	17	1	1326
LTE	DUC	20	1	1	20	122.88	N	368.64	A	N	11	0	591
LTE	DDC	20	1	1	20	122.88	N	368.64	A	N	14	0	916
LTE	DUC	20	2	1	40	122.88	N	368.64	A	N	25	2	1114
LTE	DDC	20	2	1	40	122.88	N	368.64	A	N	28	2	1418
LTE	DUC	20	2	1	40	122.88	N	368.64	A	Y	25	30	1138
LTE	DDC	20	2	1	40	122.88	N	368.64	A	Y	28	43	1245
TD-SCDMA	DUC	1.6	9	1	15	122.88	N	368.64	A	N	23	11	1193
TD-SCDMA	DUC	1.6	9	1	15	122.88	N	368.64	A	N	19	13	1161
TD-SCDMA	DUC	1.6	9	4	15	122.88	N	368.64	A	N	92	38	4501
TD-SCDMA	DUC	1.6	9	4	15	122.88	N	368.64	A	N	76	46	4127
TD-SCDMA	DUC	1.6	12	1	20	122.88	N	368.64	A	N	26	14	1226
TD-SCDMA	DUC	1.6	12	1	20	122.88	N	368.64	A	N	22	15	1296

Notes:

1. Clock frequency determined using a -2 speed grade and 300ps clock jitter allowance.

Resource Utilization (Spartan-6)

Table 18 characterization data for Spartan-6 FPGAs using an XC6SLX150T device in an FGG900 package; the speed grade required varies depending on the target clock rate and design size; the speed grade used for each configuration is indicated in the final column. Block RAM counts quoted are for 9K blocks, which are often amalgamated into pairs for mapping to 18K locations where possible; it is important to note this when comparing these values with map results for your configuration.

Table 18: **DUC/DDC Compiler Resource Utilization in Spartan-6 FPGAs**

Wireless Air Standard	DUC / DDC	Channel BW (MHz)	Carriers	Antennas	Passband (MHz)	RF Sample Rate (Msps)	Fs/4 Digital IF	Clock Frequency (MHz)	Optimize for Area or Speed (A/S)	Maximize Block RAMs	DSP48A1s	9K Block RAMs	Slices	Speed Grade
LTE	DUC	5	4	1	5	122.88	N	245.76	A	N	23	2	1133	-4
LTE	DDC	5	4	1	5	122.88	N	245.76	A	N	26	2	1296	-4
LTE	DUC	10	2	1	20	122.88	N	245.76	A	N	20	2	866	-4
LTE	DDC	10	2	1	20	122.88	N	245.76	A	N	23	2	974	-4
LTE	DUC	20	1	1	20	122.88	N	245.76	A	N	16	0	539	-3
LTE	DDC	20	1	1	20	122.88	N	245.76	A	N	20	0	630	-3
LTE	DUC	20	2	1	40	122.88	N	245.76	A	N	35	2	816	-3
LTE	DDC	20	2	1	40	122.88	N	245.76	A	N	40	2	884	-3
TD-SCDMA	DUC	1.6	9	2	15	76.80	N	230.40	S	N	28	7	1158	-4
TD-SCDMA	DDC	1.6	9	2	15	76.80	N	230.40	S	N	22	11	1424	-4
TD-SCDMA	DUC	1.6	9	1	15	92.16	N	184.32	A	N	49	18	1240	-3
TD-SCDMA	DDC	1.6	9	1	15	92.16	N	184.32	A	N	40	18	1200	-3
TD-SCDMA	DUC	1.6	9	1	15	92.16	N	276.48	A	N	28	18	956	-4
TD-SCDMA	DDC	1.6	9	1	15	92.16	N	276.48	A	N	24	16	1150	-4
TD-SCDMA	DUC	1.6	12	1	20	92.16	N	184.32	A	N	51	14	1646	-3
TD-SCDMA	DDC	1.6	12	1	20	92.16	N	184.32	A	N	42	22	1611	-3

Notes:

1. Clock frequency determined using 300ps clock jitter allowance.

Simulation Support

The core has been tested with the following simulators:

- Mentor Graphics ModelSim v6.5d
- Cadence Incisive Enterprise Simulator (IES) v9.2
- ISE Simulator 12.3

Glossary

Table 19: **Glossary**

Acronym	**Definition**
3GPP	3rd Generation Partnership Project
ACLR	Adjacent Channel Leakage Ratio
ACS	Adjacent Channel Selectivity
ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
AMBA	Advanced Microprocessor Bus Architecture
APB	AMBA Peripheral Bus
BRAM	Block Random Access Memory
CFR	Crest Factor Reduction
DAC	Digital-to-Analog Converter
dB	deciBels. Unit of logarithmic ratio measurement
DDC	Digital Down-Converter
DDS	Direct Digital Synthesizer
DFE	Digital Front End
DUC	Digital Up-Converter
EVM	Error Vector Magnitude
FIR	Finite Impulse Response (filter)
GUI	Graphical User Interface
IF	Intermediate Frequency
LSB	Least Significant Bit
MRM	Mixing Rate Multiple: the integer multiple of the frequency raster to reach the mixing frequency, that is, the mixing frequency divided by the frequency raster.
Raster	A regularly spaced sequence of discrete frequencies, used to define carrier positioning. A minimum raster size is specified by the wireless air standard, but a lower raster value is often used as the step size of a rasterized DDS.
RF	Radio Frequency
RO	Read Only
RW	Read/Write
TDD	Time Division Duplex
WO	Write Only

References

1. DS534, "FIR Compiler v5.0 Product Specification"
2. XAPP1018, "Designing Efficient Wireless Digital Up and Down Converters Leveraging CORE Generator and System Generator"
3. XAPP1113, "Designing Efficient Digital Up and Down Converters for Narrowband Systems"
4. XAPP1123, "3GPP LTE Digital Front End Reference Design"

5. 3GPP TS 36.104, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception; (Release 8)"
6. 3GPP TR 25.105, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Base Station (BS) radio transmission and reception (TDD) (Release 8)"
7. ARM IHI0024B, "AMBA 3 APB Protocol," v1.0

Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY.*

Refer to the *IP Release Notes Guide* (XTP025) for further information on this core. There will be a link to all the DSP IP and then to the relevant core being designed with.

For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

Ordering Information

This core may be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator software v12.3 and later. The Xilinx CORE Generator system is shipped with Xilinx ISE Design Suite software.

To order Xilinx software, contact your local Xilinx sales representative.

Information on additional Xilinx LogiCORE IP modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
07/23/10	1.0	Initial Xilinx release.
09/21/10	1.1	ISE 12.3 update with minor corrections.

Notice of Disclaimer