



XAPP1018 (v1.0) October 22, 2007

Designing Efficient Wireless Digital Up and Down Converters Leveraging CORE Generator and System Generator

Authors: Helen Tarn, Kevin Neilson, Ramon Uribe, David Hawke

Summary

This application note demonstrates how efficient implementations of Digital Up Converters (DUC) and Digital Down Converters (DDC) can be done by leveraging the Xilinx DSP IP portfolio for increased productivity and reduced time to development. Step-by-step instruction is given on how to perform system-level trade off analysis and develop the most efficient FPGA implementation, thus allowing engineers a flexible, low-cost and low-power alternative to ASSP technologies.

To provide step-by-step guidance on how to perform these critical steps in any system-level study, two examples are used of systems that require cost and power efficient DUC/DDC designs. The two chosen examples are for UMTS and CDMA2000 in both Spartan™-DSP and Virtex™-5 FPGAs. As such, the system-level analysis may be different between said technologies simply due to different performance available from the target technology. The resultant designs for both these devices provide customers a cost-effective and flexible alternative for cellular infrastructure equipment, such as Femtocells, Picocells, or Macrocells.

Introduction

Designing DUC and DDC functions can be achieved easily with FPGAs. Engineers, however, need to know how to leverage the silicon architectures and system-level tradeoffs that result in the lowest cost implementation. Significant area savings can be achieved with careful choice of clock speed and evaluation of different filter architectures to realize the same function.

This application note provides detailed insight into tradeoffs that can be achieved with FPGAs and careful system-level design that is sensitive to the target technology.

Highlights:

- 3-carrier DUC and DDC design files for CDMA2000
- 1-carrier DUC and DDC design files for WCDMA
- Both designs available for Virtex-5 and Spartan-DSP FPGAs
- Designed using class-leading Xilinx DSP IP portfolio
- Designs offered with both System Generator and HDL/CORE Generator™ design flows
- Architectural considerations and hardware utilization trade-off discussions

Table 1 summarizes the DUC/DDC design example offering. It specifies the supported design flows and the target FPGAs corresponding to each design example.

Table 1: Summary of Reference Design Offering

Design Example		Virtex-5 FPGA	Spartan-DSP FPGA
1-carrier WCDMA DUC/DDC	System Generator	√	√
	HDL/Core Generator	√	
3-carrier CDMA2000 DUC/DDC	System Generator	√	√

Contents

Summary	1
Introduction	1
Acronyms	6
Overview	7
IF Sampling Rate and System Clock	7
Single-Carrier WCDMA Design Example	8
Digital Up-Converter	8
Spectral Mask Requirements	8
Adjacent Channel Leakage Ratio	9
Error Vector Magnitude	9
Peak Code Domain Error	9
DUC Filter Design	10
Architectural Consideration	10
Channel Filter	12
First Halfband Interpolator	12
Second Halfband Interpolator	13
Third Halfband Interpolator	14
Composite Filter Response	14
Frequency Translation	15
DUC Performance	16
DUC Implementation	17
Using System Generator Design Flow	17
Construction of Interpolation Filters Using the FIR Compiler Block	17
Generate Sinusoids Using the DDS Compiler Block	20
Implementing Complex Multiplier Using DSP48 Slices	23
Using DSP48E to Implement Complex Multiplier for the Virtex-5 FPGA Family	23
Using DSP48A to Implement Complex Multiplier for the Spartan-DSP FPGA Family	25
Using HDL Design Flow	27
FIR Compiler CORE Generator GUI and Parameters	27
Filter Information Panel	28
Filter Specification Panel	29
Load Filter Coefficients (COE File)	29
Filter Specification, Frequency Specification, and Datapath Options	29
Filter Summary	29
DDS Compiler CORE Generator GUI and Parameters	31
DDS Information Panel	31
DDS Specification Panel	31
DDS Summary	32
Digital Down-Converter	33
Performance Requirements	33
Frequency Translation	34
DDC Filter Design	34
Architectural Consideration	34
Notes on Estimating Filter Size	35
First Half-Band Decimator	36
Second Half-Band Decimator	36
Channel Filter	37
Composite Filter Response	37
DDC Performance	38
Wideband Data Test	38
Adjacent Channel Selectivity and Blocking	39
Intermodulation Response Rejection	40
Performance Summary	41
DDC Implementation	41
Using System Generator Design Flow	41
Generate Sinusoids Using the DDS Compiler Block	42
Implement Mixer Using XtremeDSP DSP48 Slices	42
Construct Decimation Filters Using the FIR Compiler Block	43
Implement Gain Control and Time Division De-Multiplexing	43
Using HDL Design Flow	44
Three-Carrier CDMA2000 Design Example	45
Digital Up-Converter	45
Performance Requirements	45
Spectral Mask Requirements	46
Frequency Response Limits	47
Mean Squared Error Limits	47

Phase Equalization	47
Adjacent Channel Power Ratio	48
Rho (ρ) and Error Vector Magnitude	48
DUC Filter Design	48
Filter Design Strategy	48
Channel Filter	49
First Interpolation-by-5 Filter	50
Second Interpolation-by-5 Filter	51
Composite Filter Response	51
Frequency Translation and Combining	53
DUC Performance	54
DUC Implementation	54
Construction of Interpolation Filters Using the FIR Compiler Block	56
Notes on Multi-Channel Implementations	57
Generate Sinusoids Using the DDS Compiler Block	57
Implement Mixer and Combiner	58
Digital Down-Converter	60
Performance Requirements	60
Frequency Translation	61
DDC Filter Design	61
Filter Design Strategy	61
First Decimation-by-5 Filter	61
Second Decimation-by-5 Filter	62
Channel Filter	63
Composite Filter Response	63
DDC Performance	64
Wideband Data Test	64
Adjacent Channel Selectivity	65
Single Tone Desensitization	66
Intermodulation Spurious Response	68
Performance Summary	69
DDC Implementation Using System Generator	69
DFE Resource Utilization Summary	71
Power Consumption	72
Interface Requirements	73
WCDMA DUC Interface Description	73
WCDMA DUC Interface Timing	75
System Generator Design	75
HDL Design	76
WCDMA DDC Interface Description	77
WCDMA DDC Interface Timing	78
System Generator Design	78
HDL Design	79
CDMA2000 DUC Interface Description	80
CDMA2000 DUC Interface Timing	81
CDMA2000 DDC Interface Description	82
CDMA2000 DDC Interface Timing	83
Latency	84
System Integration	84
Conclusion	85
References	85
Appendix A: Determining the Number of Interpolation/Decimation Stages	86
Appendix B: Brief Introduction on Filter Design Techniques	87
Calculating FIR Coefficients	87
Window Method of Coefficient Calculation	87
Frequency Sampling Method	87
Remez/Parks-McClellan	87
Root Raised Cosine (RRC) Filters	88
Lth-Band Filters	88
Filter Architectures	88
Fully-Parallel and Folded Architectures	88
Transverse, Transpose, and Systolic	89
Multiplier-Free Filter Designs	89
CSD	89
CIC	89
Interpolators/Decimators	90
Symmetry	90
Rounding and Finite Word Length Effects	90

Appendix C: Description of WCDMA Reference Design Files in System Generator	90
Appendix D: Hardware Co-Simulation and Incorporate HDL models into System Generator	94
Appendix E: Description of CDMA2000 Reference Design Files	96
Revision History	97
Notice of Disclaimer	97

List of Figures

Figure 1. High-Level Block Diagram for WCDMA	8
Figure 2. The FDATool Screen Shot for Channel Filter Design	11
Figure 3. WCDMA DUC Interpolation Filter Architectural Choice.	12
Figure 4. Magnitude Response of RRC Filter.	12
Figure 5. Magnitude Response of First Halfband Filter	13
Figure 6. Magnitude Response of Second Halfband Filter.	13
Figure 7. Magnitude Response of Third Halfband Filter.	14
Figure 8. Overall WCDMA DUC Filter Response	15
Figure 9. DDS Output when Carrier Frequency Tuned to 12 MHz.	16
Figure 10. PSD for Single-Carrier WCDMA Waveform Centered at 12 MHz with Spectral Mask Shaded.	16
Figure 11. Top-Level Block Diagram for WCDMA DUC Implementation	17
Figure 12. WCDMA DUC System Generator Implementation	17
Figure 13. FIR Compiler Token in System Generator	18
Figure 14. FIR Compiler GUI in System Generator	19
Figure 15. DDS Compiler Block Diagram.	20
Figure 16. DDS Compiler Token in System Generator	21
Figure 17. DDS Compiler GUI in System Generator	22
Figure 18. Virtex-5 FPGA DSP48E Block Diagram	23
Figure 19. Virtex-5 FPGA Mixer Implementation for WCDMA DUC	24
Figure 20. Virtex-5 FPGA DSP48E and Opmode Selection Block.	25
Figure 21. Spartan-DSP FPGA DSP48A Block Diagram	25
Figure 22. Spartan-DSP FPGA Mixer Implementation for WCDMA DUC	26
Figure 23. Spartan-DSP FPGA DSP48A and Opmode Selection Block	27
Figure 24. FIR Compiler in CORE Generator.	27
Figure 25. FIR Compiler Screen Shot (Page 1)	28
Figure 26. FIR Compiler Screen Shot (Page 2)	30
Figure 27. FIR Compiler Screen Shot (Page 4)	30
Figure 28. DDS Compiler Screen Shot (Page 1)	32
Figure 29. DDS Compiler Screen Shot (Page 5)	33
Figure 30. High-Level Block Diagram for WCDMA DDC	34
Figure 31. WCDMA DDC Decimation Filter Architectural Choice	35
Figure 32. Magnitude Response of First Halfband Decimator	36
Figure 33. Magnitude Response of Second Halfband Decimator	36
Figure 34. Magnitude Response of WCDMA DDC Channel Filter	37
Figure 35. PSD for WCDMA DDC Output	38
Figure 36. WCDMA DDC Input for Wideband Data Test	39
Figure 37. DDC Output from Wideband Data Test.	39
Figure 38. WCDMA DDC Input for ACS and Blocking Test (Centered at 15 MHz)	40
Figure 39. WCDMA DDC Input for Intermodulation Test	40
Figure 40. Top-Level Block Diagram for WCDMA DDC Implementation	41
Figure 41. WCDMA DDC System Generator Implementation	41
Figure 42. Virtex-5 FPGA Mixer Implementation for WCDMA DDC.	42
Figure 43. Spartan-DSP FPGA Family Mixer Implementation for WCDMA DDC	43
Figure 44. Gain Control and TDD Block for WCDMA DDC Implementation	44
Figure 45. High-Level Block Diagram for CDMA2000 DUC	45
Figure 46. Three CDMA2000 Carriers Positioned in a 5 MHz Bandwidth.	46
Figure 47. CDMA2000 DUC Interpolation Filters Chain	49
Figure 48. Magnitude Response of CDMA2000 DUC Channel Filter.	50
Figure 49. Magnitude Response of First Interpolation-by-5 Filter	50
Figure 50. Magnitude Response of Second Interpolation-by-5 Filter	51
Figure 51. Overall Filter Response vs. Frequency Response Limits	52
Figure 52. Overall Filter Response vs. Spectral Mask Requirements	52
Figure 53. Composite Impulse Response v.s. IS-95 Response	53
Figure 54. PSD for Three-Carrier CDMA2000 waveform	54
Figure 55. Block Diagram for CDMA2000 DUC Implementation Targeting Virtex-5 FPGA	55
Figure 56. Block Diagram for CDMA2000 DUC Implementation Targeting Spartan-DSP FPGA.	55
Figure 57. CDMA2000 DUC Implementation (Virtex-5 FPGA)	55
Figure 58. CDMA2000 DUC Implementation (Spartan-DSP FPGA)	56
Figure 59. High-Level Block Diagram for CDMA2000 DDC	60
Figure 60. CDMA2000 DDC Decimation Filters Chain.	61

Figure 61. Magnitude Response of First Decimation-by-5 Filter	62
Figure 62. Magnitude Response of Second Decimation-by-5 Filter	62
Figure 63. Magnitude Response of CDMA2000 DDC Channel Filter.	63
Figure 64. DDC Input with Three Channel CDMA2000 Composite Signal Plus AWGN	64
Figure 65. CDMA2000 DDC Output from Wideband Data Test	65
Figure 66. DDC input for Adjacent Channel Selectivity Test	65
Figure 67. DDC Output for Adjacent Channel Selectivity Test	66
Figure 68. DDC Output from Single Tone Desensitization Test 1	66
Figure 69. DDC Output from Single Tone Desensitization Test 2	67
Figure 70. DDC Output from Single Tone Desensitization Test 3	67
Figure 71. DDC Input for Intermodulation Test.	68
Figure 72. DDC Output from Intermodulation Test.	68
Figure 73. Block Diagram for CDMA2000 DDC Implementation Targeting a Virtex-5 FPGA	69
Figure 74. Block Diagram for CDMA2000 DUC Implementation Targeting a Spartan-DSP FPGA	69
Figure 75. CDMA2000 DDC Implementation (Virtex-5 FPGA)	70
Figure 76. CDMA2000 DDC Implementation (Spartan-DSP FPGA)	70
Figure 77. WCDMA DUC Top-Level Component.	74
Figure 78. Timing Diagram of Input Data Interface for WCDMA DUC in System Generator	75
Figure 79. Timing Diagram of Output Data interface for WCDMA DUC in System Generator.	75
Figure 80. Timing Diagram of Freq and Freq_we for WCDMA DUC in System Generator	76
Figure 81. Timing Diagram of Input Interface for WCDMA DUC in VHDL	76
Figure 82. Timing Diagram of Output Interface for WCDMA DUC in VHDL.	77
Figure 83. WCDMA DDC Top-Level Component.	77
Figure 84. Timing Diagram of Din, Freq, Gain Inputs for WCDMA DDC in System Generator	78
Figure 85. Timing Diagram of Output Data Interface for WCDMA DDC.	78
Figure 86. Timing Diagram of Data, Freq, Gain Inputs for WCDMA DDC	79
Figure 87. Timing Diagram of Output Data Interface for WCDMA DDC	79
Figure 88. CDMA2000 DUC Top-Level Component	80
Figure 89. Timing Diagram of Input Data Interface (with Data_Load) for CDMA2000 DUC	81
Figure 90. Timing Diagram of Output Data interface for CDMA2000 DUC	81
Figure 91. Timing Diagram of Frequency and Scaler Program for CDMA2000 DUC	82
Figure 92. CDMA2000 DDC Top-Level Component	82
Figure 93. Timing Diagram of Input Interface for CDMA2000 DDC	83
Figure 94. Timing Diagram of Output Data Interface for CDMA2000 DDC	83
Figure 95. A Sample Initialization File for WCDMA DUC Reference Design	93

Acronyms

3GPP	3 rd Generation Partnership Project
3GPP2	3 rd Generation Partnership Project 2
ACLR	Adjacent Channel Leakage Ratio
ACPR	Adjacent Channel Power Ratio
ACS	Adjacent Channel Selectivity
ADC	Analog-to-Digital Converter
BER	Bit Error Rate
BS	Base Station
CDE	Code Domain Error
CFR	Crest Factor Reduction
CIC	Cascaded Integrator Comb
CPE	Customer Premises Equipment
DAC	Digital-to-Analog Converter
DDC	Digital Down Conversion/Converter
DDS	Direct Digital Synthesizer
DFE	Digital Front End
DUC	Digital Up Conversion/Converter
EVM	Error Vector Magnitude
FIR	Finite Impulse Response
GUI	Graphical User Interface
HDL	Hardware Description Language
HOR	Hardware Over-Sampling Rate
ICI	Inter-Chip Interference
IF	Intermediate Frequency
IIR	Infinite Impulse Response
LCM	Least Common Multiple
LPF	Low Pass Filter
MAC	Multiply-Accumulate
MCPS	Mega Chips per Second
MSE	Mean Squared Error
MSPS	Mega Samples per Second
NCO	Numerically Controlled Oscillator
OEM	Original Equipment Manufacturer
PAPR	Peak-to-Average Power Ratio
PAR	Place and Route
PSD	Power Spectral Density
PCDE	Peak Code Domain Error
RMS	Root Mean Square
RRC	Root-Raised Cosine
SF	Spreading Factor
SFDR	Spurious-Free Dynamic Range
SINR	Signal-to-Interference-and-Noise Ratio
SNR	Signal-to-Noise Ratio
TDD	Time Division De-multiplex
TDM	Time Division Multiplex
XST	Xilinx Synthesis Technology

Overview

This document has two design examples described in separate sections. The first design example focuses on the WCDMA air interface, including both the DUC and DDC. After the description of system overview, system requirements, and design consideration, the performance is presented for the continuation before commencing with the implementation discussion using various IP and design flows.

The second design example is centered around the CDMA2000 air interface and is organized in the same way, except that the implementation is briefly described and the details are often referred back to the first example. The remaining sections include: DFE resource utilization, power measurement, interface requirements, latency, and system integration. These sections include data from both examples.

IF Sampling Rate and System Clock

Though WCDMA and CDMA2000 are standardized by different partnership projects, similar fundamental techniques are shared. The chip rate for WCDMA is 3.84 Mcps and the chip rate for CDMA2000 is 1.2288 Mcps. An IF sample rate that is common to both of these two designs is selected. The least common multiple (LCM) of these two rates is 30.72 Mps. To allow the user to mix to a higher IF and to increase sampling rate further to relax requirements on the post-DAC analog filters, $2 \times 30.72 \text{ Mps} = 61.44 \text{ Mps}$ is selected as the IF sampling rate.

Now the system clock for the Virtex-5 FPGA designs are selected. These criteria are used in the selection process:

- The system clock needs to be a multiple of the IF sample rate (61.44 Mps).
- For CDMA2000, the ratio of the system clock and the IF sample rate should be a multiple of three, and preferably, a multiple of six, so that the six channels (I and Q for three input channels) can be folded into a single set of filters. Selecting a ratio that is not a multiple of 3 or 6 requires that the folded FIR filters have dead cycles. Likewise, for WCDMA, the ratio should be a multiple of 2, so that the I and Q channels can be folded into a single set of filters.
- The system clock should be as high as possible, and near 400 MHz, to take advantage of the high frequencies possible with the FIR Compiler and DDS Compiler IP. At the same time, it should not be so high that major timing problems would be anticipated during place and route. Selecting an overly-optimistic system clock at this point will have strong repercussions later, as it will be hard to redesign the architecture if we find major timing issues in place and route.

With these criteria in mind, a system clock of $61.44 \text{ MHz} \times 6 = \mathbf{368.64 \text{ MHz}}$ is selected for both systems targeting the Virtex-5 FPGA.

A similar method was used to select the clocks for the Spartan-DSP systems. The Spartan-DSP architecture is not capable of such high clock speeds, so it is best to aim for a target frequency not exceeding 200 MHz. The same 61.44 MHz IF sample rate is kept. Again, for WCDMA, the ratio of the system clock to the IF sample rate should be a multiple of 2. With this in mind, a system clock of $61.44 \text{ MHz} \times 2 = 122.88 \text{ MHz}$ is selected for the WCDMA system targeting the Spartan-DSP devices. The next multiple of two exceeds 200 MHz by a large margin and would likely be too high. For CDMA2000, the ratio of the system clock to the IF sample rate should be a multiple of 3, and preferably 6. With this in mind, a system clock of $61.44 \text{ MHz} \times 3 = 184.32 \text{ MHz}$ is selected for the Spartan-DSP-based CDMA2000 design. The multiple of six would be too high. This system clock frequency requires the use of a dual 3-channel architecture to accommodate all six channels.

Single-Carrier WCDMA Design Example

Digital Up-Converter

The DUC provides pulse shaping, interpolation, and frequency translation to the single-carrier baseband WCDMA signal from 0 Hz to a set of specified center frequencies. It is designed to meet the 3rd Generation Partnership Project (3GPP) TS 25.104 specification [Ref 1], which defines the transmission and reception requirements for the base station radio.

Table 2 lists the summary of requirements for the downlink transmit path.

Table 2: Target Specification for WCDMA Downlink Transmit Path

Parameter	Value	Comments
Carrier Bandwidth	5.0 MHz	
Number of Carriers	1 carrier	
Baseband Chip Rate	3.84 MCPS	
IF Sample Rate	61.44 MSPS	16×3.84 MSPS
Transmit Spectral Mask	Up to 70 dB for frequency offsets over 3.5 MHz	Section 6.6.2.1 of [Ref 1] plus 20 dB margin
ACLR	45 dB for adjacent channel 50 dB for alternate channel	Section 6.6.2.2 of [Ref 1]
EVM	17.5% for QPSK 12.5% for 16-QAM	Section 6.8.2 of [Ref 1]
Input Signal Quantization	16-bit I and Q	Complex
Output Signal Quantization	16-bit I and Q	Complex output is more general than real output
Mixer Properties	Tunability: Variable Resolution: ~0.25 Hz SFDR: up to 115 dB	Various SFDR are supported by the DDS Compiler core (28 bits of frequency accumulator width)

A standard like WCDMA has many component signals summing up as the transmitted signal encounters the high peak-to-average-power ratio (PAPR) problem, which often requires processing modules, such as the Crest Factor Reduction (CFR) following the DUC. These blocks expect both the real and imaginary component to be present. Thus, an output with a complex format was chosen since it is a more generic solution. When optional CFR is not required or an RF structure that takes only real output is used, one can simply neglect the imaginary output. Figure 1 shows a high-level block diagram of the WCDMA DUC in the downlink transmit path.

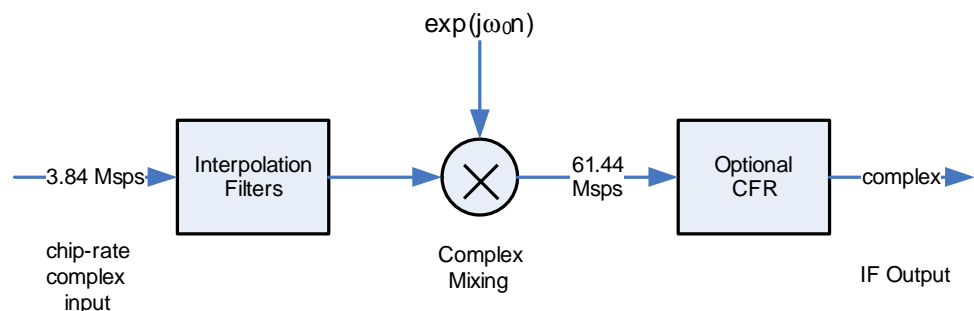


Figure 1: High-Level Block Diagram for WCDMA DUC

Spectral Mask Requirements

The pulse shaping filter for the downlink transmitter is a root-raised cosine (RRC) with a roll-off factor $\alpha = 0.22$. Obtained from section 6.6.2.1 of 3GPP TS 25.104 [Ref 1], the spectral mask

requirement for the $P \geq 43$ dBm case, which has the base station (BS) maximum output power, is shown in Table 3. The BS maximum output power is spread over the entire bandwidth of 3.84 MHz of the WCDMA signal. The filter attenuation requirements listed in Table 4 are generated by normalizing maximum power against the measurement bandwidth and then enhanced with a 20 dB margin. The margin adds extra security when CFR algorithm is used or analog distortion is present.

Table 3: 3GPP Spectral Mask for $P \geq 43$ dBm

Frequency Offset to Measurement Filter Edge (-3 dB cutoff)	Maximum Power in Measurement Bandwidth	Measurement Bandwidth
2.5 MHz to 2.7 MHz	-14 dBm	30 kHz
2.7 MHz to 3.5 MHz	-14 dBm \rightarrow -26 dBm	30 kHz
3.5 MHz to 4.0 MHz	-26 dBm	30 kHz
3.5 MHz to 12.0 MHz	-13 dBm	1 MHz

Table 4: Filter Attenuation Requirements for $P = 43$ dB

For Δf Within the Range	Minimum Attenuation (dB)	Attenuation with 20 dB Margin (dB)
2.5 MHz to 2.7 MHz	36	56
2.7 MHz to 3.5 MHz	36 \rightarrow 48	56 \rightarrow 68
3.5 MHz to 12.0 MHz	50	70

Adjacent Channel Leakage Ratio

Adjacent Channel Leakage Ratio (ACLR) is defined as the ratio of the in-band power to the power in an adjacent channel after each band is filtered by a matched Root-Raised Cosine (RRC) filter. The minimum requirement is listed in the section 6.6.2.2 of 3GPP TS 25.104 [Ref 1]. The first adjacent channel leakage ratio (ACLR1) is defined for an offset center frequency of 5 MHz. The second adjacent channel leakage ratio (ACLR2), a.k.a. alternate channel leakage ratio, is defined for an offset center frequency of 10 MHz. The limits for ACLR1 and ACLR2 are 45 dB and 50 dB, respectively.

Error Vector Magnitude

Error Vector Magnitude (EVM) is defined as the difference between the reference waveform and the measured waveform. As described in section 6.8.2 of 3GPP TS 25.104 [Ref 1], both waveforms have to pass through a matched RRC filter. The EVM can be expressed as:

$$EVM(\%) = \frac{RMS(error_vector)}{RMS(reference_signal)} \times 100\% \quad \text{Equation 1}$$

The minimum requirement defines that EVM shall be less than 17.5% when QPSK modulation is used and 12.5% when 16 QAM is used. The frequency, phase, amplitude, and chip clock timing can be selected to minimize the EVM value.

Peak Code Domain Error

Peak Code Domain Error (PCDE) is defined as the maximum value for the Code Domain Error (CDE) for all codes. It is computed by projecting the error vector onto the code domain at a specific spreading factor (SF). CDE is the ratio of the mean power of the projection onto that code to the mean power of the composite reference waveform. The minimum requirement for the PCDE is -33 dB at SF of 256.

DUC Filter Design

Architectural Consideration

When it comes to filter architectural consideration, both the system performance and the hardware resource usage should be taken into account. The interpolation filter chain in the DUC needs to pulse-shape and up sample the baseband data by a factor of $61.44/3.84=16$. There are several options to partition the specific rate change. Some structures might be more efficient or have better performance than others.

At first glance, one may attempt to design an upsample-by-16 interpolator with pulse shaping capability in one shot. This is impractical as it is often difficult to design such a filter to meet the required spectral mask within a reasonable filter length. Even worse, the total required computational complexity is extremely high. A better approach is to decompose the rate conversion into multiple interpolation stages. It is fairly straightforward to design an up-sampled-by-2 RRC channel filter with lower order and meet the system performance requirement.

After the channel filter, the signal still needs to be up sampled by 8 with aliasing effects removed in this process. Four possible configurations are considered:

1. A single up-sampled-by-8 filter
2. An up-sample-by-4 filter followed by a halfband filter
3. A halfband filter followed by an interpolation filter with a rate of 4
4. A cascade of three halfband interpolators

This design example does not contain a CFR module, so the system targets such as ACLR and EVM are fairly easy to meet for all the configurations. Therefore, the deciding factor on the filter architectural consideration is primarily the silicon real estate consumption. First, the filters should be designed in each configuration to obtain the filter length. This is produced by using the MATLAB® Filter Design and Analysis Tool (FDATool). The FDATool has a graphical user interface (GUI) that allows users to set filter specifications, quickly visualize the filter responses, and import and export the filter designs from and to the MATLAB workspace to do further analysis. [Figure 2](#) shows the screenshot of the FDATool for the RRC filter design with windowing.

After determining the filter length, one way to quickly evaluate the hardware utilization is to use the GUI resource estimator of the FIR compiler in the Xilinx DSP IP portfolio. It reports accurate metrics on block RAM and DSP48 slice usage. This feature is covered in greater detail in the [“FIR Compiler CORE Generator GUI and Parameters”](#) section. [Table 5](#) lists the filter length, sample rate, block RAM, and DSP48 slice usage for the anti-aliasing filters (channel filter excluded) of all four configurations. (For a formula to estimate filter resource requirements, see [“Notes on Estimating Filter Size.”](#))

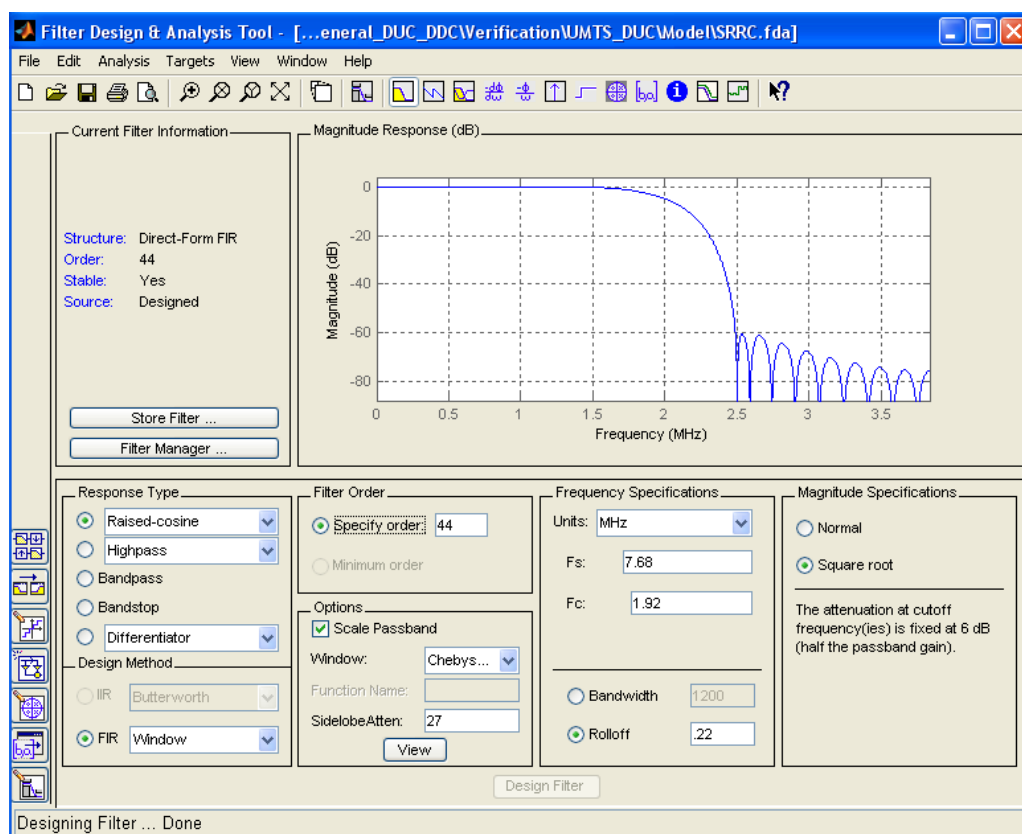


Figure 2: The FDATool Screen Shot for Channel Filter Design

Table 5: Filter Length and Sample Rate vs. Hardware Utilization for Four Configurations

Anti-Aliasing Filter	Filter Length & Sample Rate for 1 st Filter	Filter Length & Sample Rate for 2 nd Filter (if any)	Filter Length & Sample Rate for 3 rd Filter (if any)	Total Required DSP48Es for Virtex-5 Devices	Total Required DSP48As for Spartan-DSP Devices	Architecture of Choice
Configuration 1	91 taps ($\uparrow 8$) 61.44 MSPS			3	7	
Configuration 2	47 taps ($\uparrow 4$) 30.72 MSPS	11 taps ($\uparrow 2$) 61.44 MSPS		4	7	
Configuration 3	23 taps ($\uparrow 2$) 15.38 MSPS	25 taps ($\uparrow 4$) 61.44 MSPS		4	6	
Configuration 4	23 taps ($\uparrow 2$) 15.38 MSPS	11 taps ($\uparrow 2$) 30.72 MSPS	11 taps ($\uparrow 2$) 61.44 MSPS	3	5	✓

Users are encouraged to play with the software bundled with the reference design to further analyze the performance of all four configurations. Note that the system targets are harder to meet when a CFR module is used and that one configuration can be predominately more favorable than others. For the WCDMA DUC design discussed in the sections following, the focus is on configuration 4, an architecture that partitions the filter chain to the channel filter plus a cascade of three anti-aliasing halfband filters (shown in Figure 3). It requires minimum DSP48 usage for both the Virtex-5 and Spartan-DSP FPGA architectures, while the slice utilization for all four configurations is about equivalent.

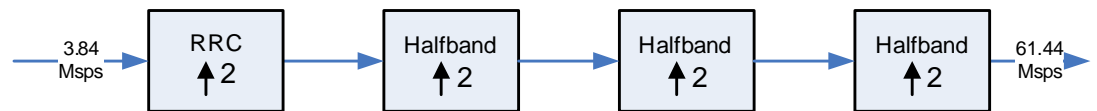


Figure 3: WCDMA DUC Interpolation Filter Architectural Choice

Channel Filter

The transmitted pulse shaper is an RRC filter with roll-off $\alpha = 0.22$. When it convolves with the matched RRC filter, the overall raised-cosine response has no inter-chip interference (ICI) because the zero crossings occur at chip intervals. However, it has an infinite response in the time domain. Moreover, the RRC filter also needs to be designed to meet the spectral requirements discussed in the “[Spectral Mask Requirements](#)” section.

Practical filters that approximate the overall desired RRC frequency response can be designed, but often require analyzing the tradeoff between system performance and hardware complexity. A common method is to use the window approach to design a filter within a manageable length. It was found that a 45-tap symmetric RRC filter with Chebyshev windowing (sidelobe parameter set to 27 dB) met all of the filter objectives. The Chebyshev windowing provides better sidelobe suppression than a rectangular window at the expense of some widening of the mainlobe. Also, it is a type of adjustable window that requires lower filter order to achieve similar performance than various windows, such as Hann, Blackman, and Hamming. The frequency response of the RRC filter with an interpolation factor of two is obtained and illustrated in [Figure 4](#).

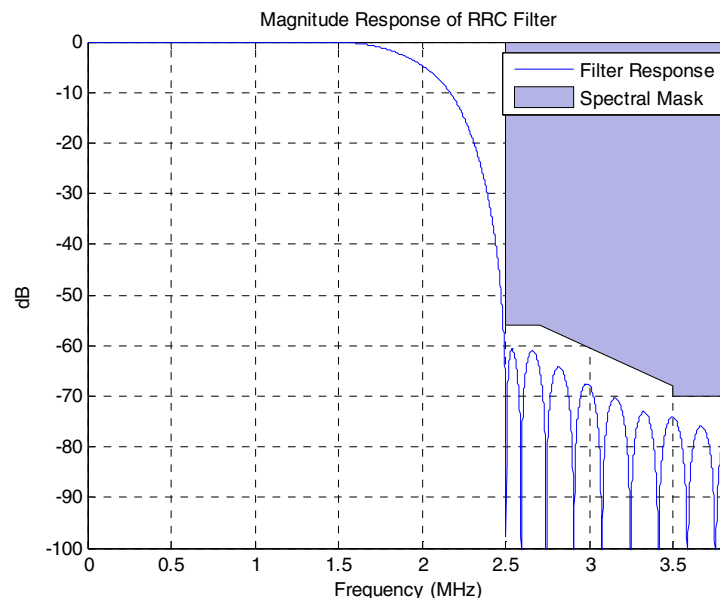


Figure 4: Magnitude Response of RRC Filter

First Halfband Interpolator

After the channel filter, a cascade of interpolation filters follows to remove the aliasing effect produced by up sampling. The signal is first up sampled by two using a half band interpolator. Halfband filters are a type of FIR filter where its transition region is centered at one quarter of the sampling rate, $F_s/4$. The end of its passband and the beginning of the stopband are equally spaced on either side of $F_s/4$. When it comes to implementing an interpolation filter with a rate of two, the halfband filter is often the structure of choice, since it requires much less computational power (and thus less hardware) for a filter realization. This results from the fact that every odd indexed coefficient in the time domain is zero except the center tap and even indexed coefficients are symmetric.

The output sample rate of the filter is four times the chip rate, that is, $F_s = 3.84 \times 4 = 15.36$ MSPS. The passband was set to $F_{\text{pass}} = 3.84 \times 1.22/2 = 2.34$ MHz and the passband ripple is chosen to be 0.002 dB. Using the MATLAB® FDATool, a 23-tap equiripple halfband filter was designed to this specification. If preferred, the passband can also be set using the normalized frequency $W_{\text{pass}} = 3.84 \times 1.22/14.36 = 0.305$ which leads to the identical filter design.

The frequency (magnitude) response of the filter is shown in Figure 5. With the filter coefficients quantized to 16 bits, it can be seen that the stopband attenuation is around 80 dB.

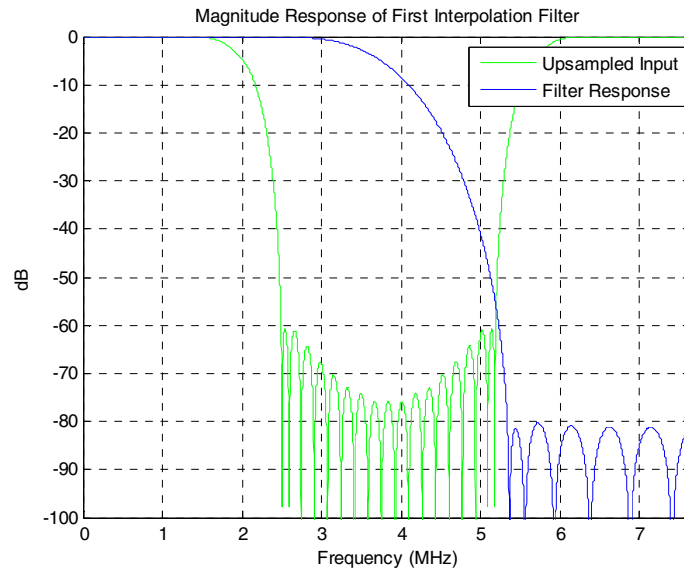


Figure 5: Magnitude Response of First Halfband Filter

Second Halfband Interpolator

After the first interpolation filter, the signal is again up sampled by a factor of two using a halfband interpolator. The output sample rate of the filter is eight times the chip rate, that is, $F_s = 3.84 \times 8 = 30.72$ MSPS. The passband is the same as the first interpolator, set to $F_{\text{pass}} = 2.34$ MHz. The passband ripple was chosen to be 0.002 dB. An 11-tap halfband filter was designed, and its magnitude response is shown in Figure 6. The stopband attenuation is better than 80 dB with the filter coefficients quantized to 16 bits.

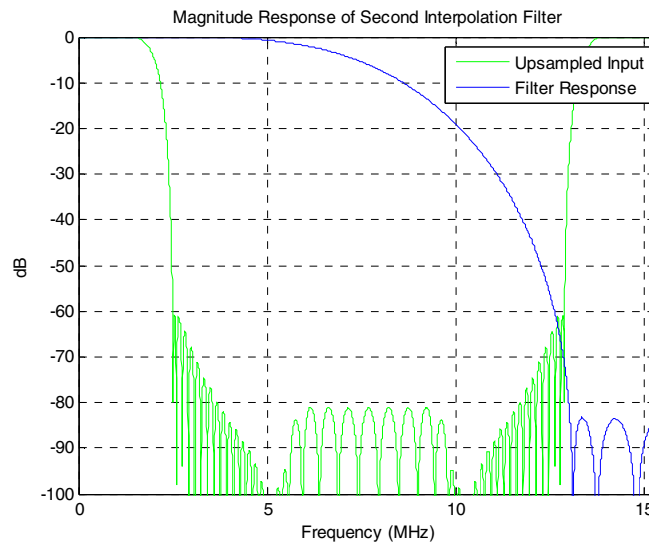


Figure 6: Magnitude Response of Second Halfband Filter

Third Halfband Interpolator

Finally, the signal enters the last halfband interpolator to reach an output sample rate of 61.44 MSPS. The passband was set to $F_{\text{pass}} = 3.84 \times 1.22/2 = 2.34$ MHz, identical to the first two interpolators. The passband ripple was chosen to be 0.001 dB. An 11-tap halfband filter was designed to this specification with its magnitude response shown in Figure 7. The stopband attenuation far exceeds 100 dB with the coefficients quantized to 16 bits.

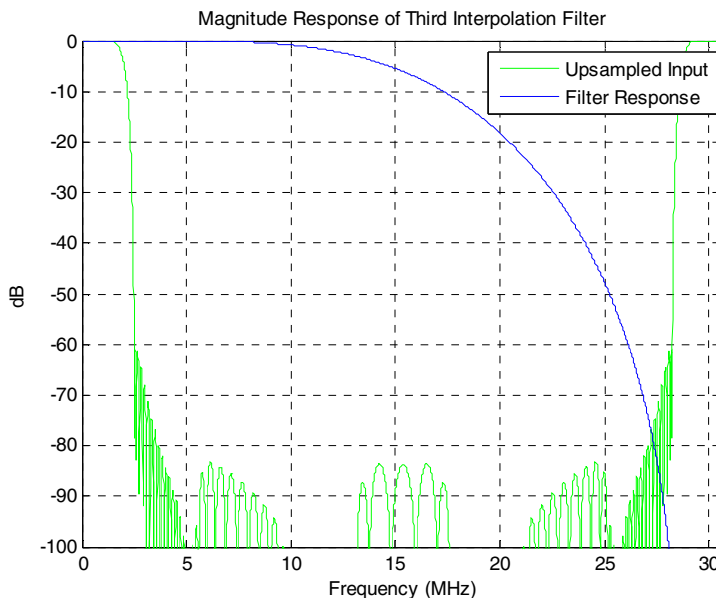


Figure 7: Magnitude Response of Third Halfband Filter

Composite Filter Response

The specifications of all four filters are summarized in Table 6, and the composite frequency response of all four filters is shown in Figure 8. The response has a peak-to-peak passband ripple of less than 0.015 dB. The filter response also meets the spectral emission mask, shown as the shaded area in the figure, after accounting for quantization effects.

Table 6: Summary of WCDMA Interpolation Filters

Filter Stage	Passband F_{pass} (MHz)	Output Sample Rate F_s (MSPS)	Peak-to-Peak Ripple (dB)	Filter Order
Windowed RRC	1.92	7.68	0.012	44
First Halfband	2.34	15.36	0.002	22
Second Halfband	2.34	30.72	0.002	10
Third Halfband	2.34	61.44	0.001	10

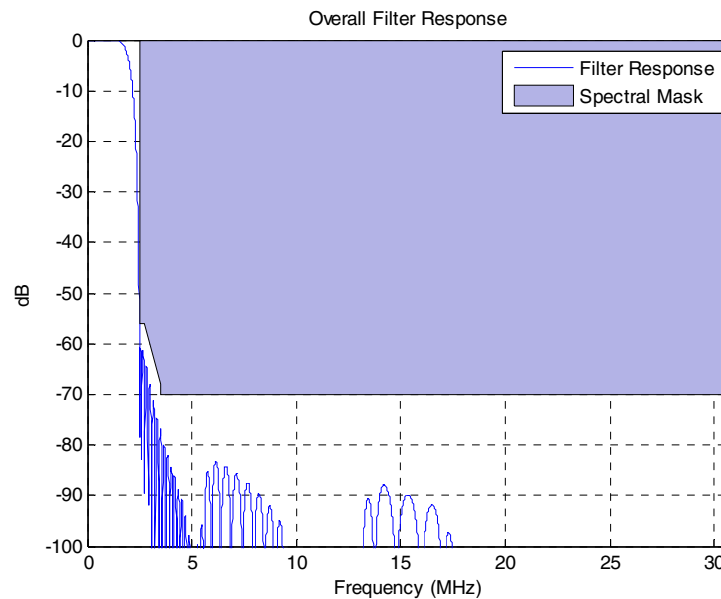


Figure 8: Overall WCDMA DUC Filter Response

Frequency Translation

After pulse shaping and up-converting the baseband signal, the signal spectrum is shifted from centered at 0 Hz to an intermediate frequency in the range of $[-F_s/2, F_s/2]$, where $F_s = 61.44$ MHz. This process requires a Direct Digital Synthesizer (DDS) and a mixer. In the single carrier WCDMA design, the mixer is basically a complex multiplier which multiplies the up-sampled signals (interpolation filter output) with the complex exponential generated from the DDS.

The DDS, also known as a numerically controlled oscillator (NCO), generates a local oscillator to mix with the incoming data. It effectively frequency shifts the incoming data up and down with a tunable carrier frequency.

The clarity of the DDS is crucial to the spectral purity of the DFE output, as the fidelity of the output signal is limited by the DDS's noise floor. This noise is caused by both the phase and amplitude quantization of the process. A phase dithering method is often used to improve the spurious-free dynamic range (SFDR) of the generated sinusoidal waveform by 12 dB without increasing the depth of the look-up table. It removes the spectral line structure associated with conventional phase truncation waveform synthesis architectures. Figure 9 shows the power spectral density (PSD) for a typical DDS output when the phase dithering method is used. The carrier frequency was tuned to 12 MHz and SFDR was set to 102 dB. A Taylor series corrected method can provide even higher SFDR (up to 115 dB) using the same depth of the sinusoidal look-up table.

Although the dithered DDS has a multiplier-free architecture, it can consume too many block RAMs for the quarter waveform storage in the high SFDR setting, which results in the deterioration of the design speed. The user could certainly add more pipelining stages to achieve the desired speed, but at the expense of increasing latency. Here, the Taylor series corrected method is used since it offers an excellent trade-off on the resource utilization. For example, to achieve a SFDR of 107 dB, the dithering method uses up to 16 block RAMs while the Taylor series corrected method uses only two DSP48 slices. In the following section (section “Generate Sinusoids Using the DDS Compiler Block”), DDS implementation consideration is discussed in detail.

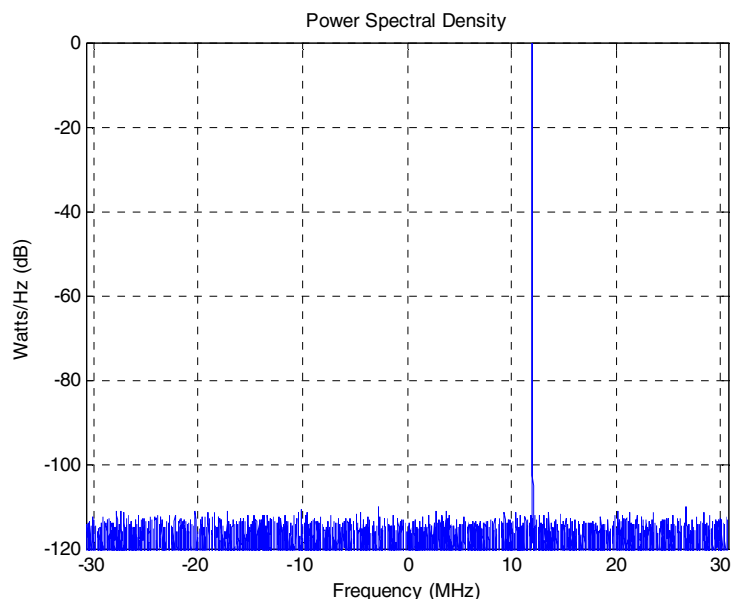


Figure 9: DDS Output when Carrier Frequency Tuned to 12 MHz

DUC Performance

Figure 10 shows the PSD for single carrier WCDMA centered at 12 MHz. As can be seen, the spectrum is well under the emission limits.

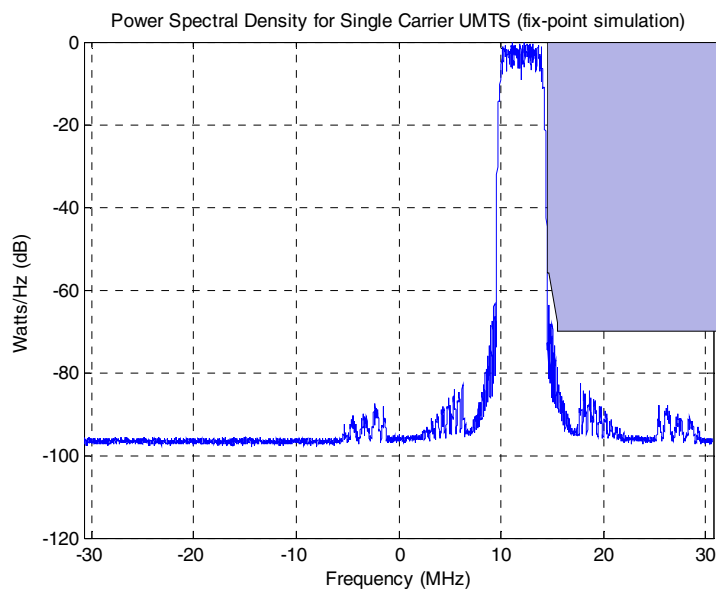


Figure 10: PSD for Single-Carrier WCDMA Waveform Centered at 12 MHz with Spectral Mask Shaded

Table 7 summarizes the DUC performance metrics generated from the fixed-point simulation. The input signal to the DUC was generated from test model 1 in the 3GPP document assuming QPSK modulation with SF of 128. Both ACLRs are measured by filtering the adjacent channel (offset by 5 and 10 MHz from the center frequency) by a matched RRC filter and meets the 3GPP requirements by a significant margin. The EVM generated by the DUC is 1.05%, conforming to the 3GPP requirement of 17.5%. The extra head room provides enough margins to account for distortion introduced by analog components. The PCDE is calculated to be -60.63 dB.

Table 7: Summary of WCDMA DUC Performance Metrics

Performance Metrics	Value	Requirement
ACLR1 (offset by 5 MHz)	78.97 dB	>45 dB
ACLR2 (offset by 10 MHz)	90.93 dB	>50 dB
EVM	1.05%	<17.5%
PCDE	-60.63 dB	<-33 dB

DUC Implementation

The top-level block diagram for the WCDMA DUC implementation is shown in Figure 11. The FPGA operation frequency for Virtex-5 and Spartan-DSP FPGA designs are 368.64 MHz and 122.88 MHz, respectively. The in-phase and quadrature phase components of the chip-rate data were first time division multiplexed (TDM'd) into a single stream before being processed by the interpolation filters.

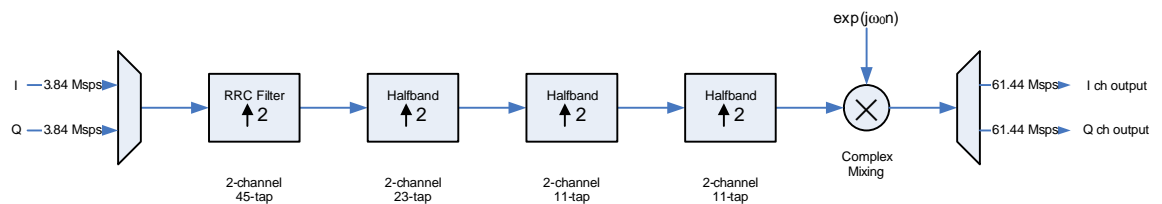


Figure 11: Top-Level Block Diagram for WCDMA DUC Implementation

Using System Generator Design Flow

The top-level screen shot for the System Generator Implementation of the WCDMA DUC is shown in Figure 12.

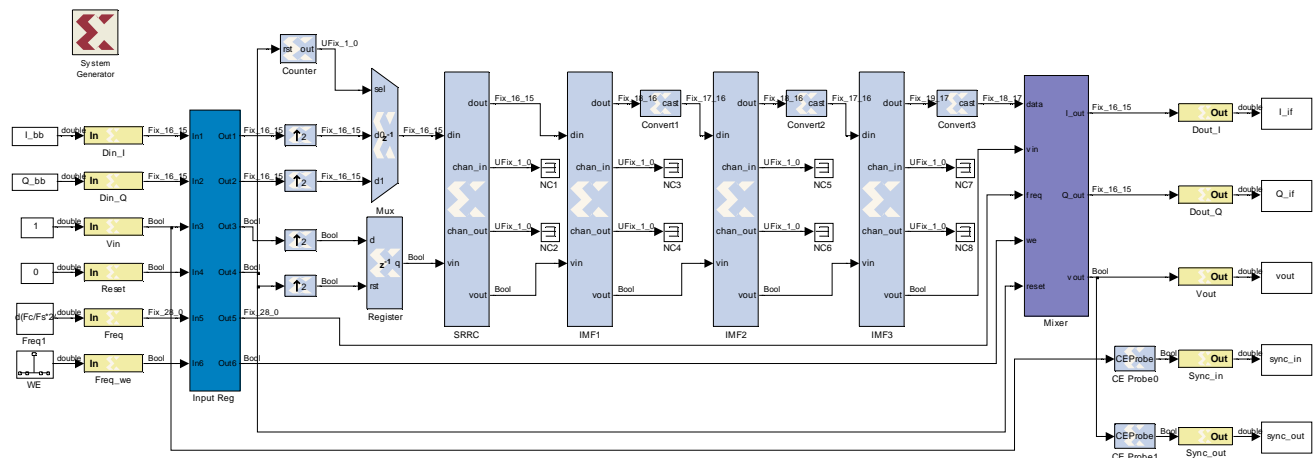


Figure 12: WCDMA DUC System Generator Implementation

Construction of Interpolation Filters Using the FIR Compiler Block

The four stage interpolators are implemented using Xilinx Finite Impulse Response (FIR) Compiler block. The FIR compiler block provides a common interface to generate a parameterizable, area-efficient, and high-performance filter module utilizing a Multiply-Accumulate (MAC) architecture. Multiple MACs may be used in achieving higher performance filter requirements, such as longer filter coefficients, higher throughput, or support for more channels.

The FIR Compiler block accepts a stream of input data and computes filtered output with a fixed delay based on the filter configuration. It has a simple interface and can be configured to have a number of optional ports in addition to din and dout ports which appear in all filter configurations. [Figure 13](#) shows the FIR Compiler v3.1 token from blockset library in System Generator and [Table 8](#) briefly describes its interface.

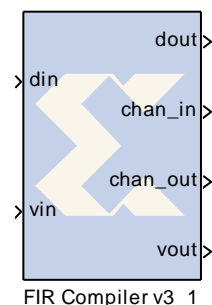


Figure 13: FIR Compiler Token in System Generator

Table 8: System Generator FIR Compiler Block Interface

Port Name	Port Direction	Port Description
din	Input	Input data for all channels is provided in a TDM manner through this port.
vin	Input	Indicates if the input data on the din port is valid (High) or invalid (Low).
dout	Output	Filtered output data for all the channels is provided in a TDM manner through this port.
vout	Output	Indicates if the output data on the dout port is valid (High) or invalid (Low).
chan_out	Output	Indicates to which channel the output data on the dout port belongs.
chan_in	Output	Indicates which channel's input data should be driven onto the din port next. As soon as the current input sample is clocked into the core, the value on this output port changes indicating the next channel for which a data input is required.

To program this block, double click the token and the block parameter dialog box is invoked, as shown in [Figure 14](#). Generic parameters, such as NTDM and MaxOverClock, are used in the GUI. These parameters are initialized through a MATLAB script before the design is simulated. More information on descriptions of the MATLAB script is covered in ["Appendix C: Description of WCDMA Reference Design Files in System Generator."](#)

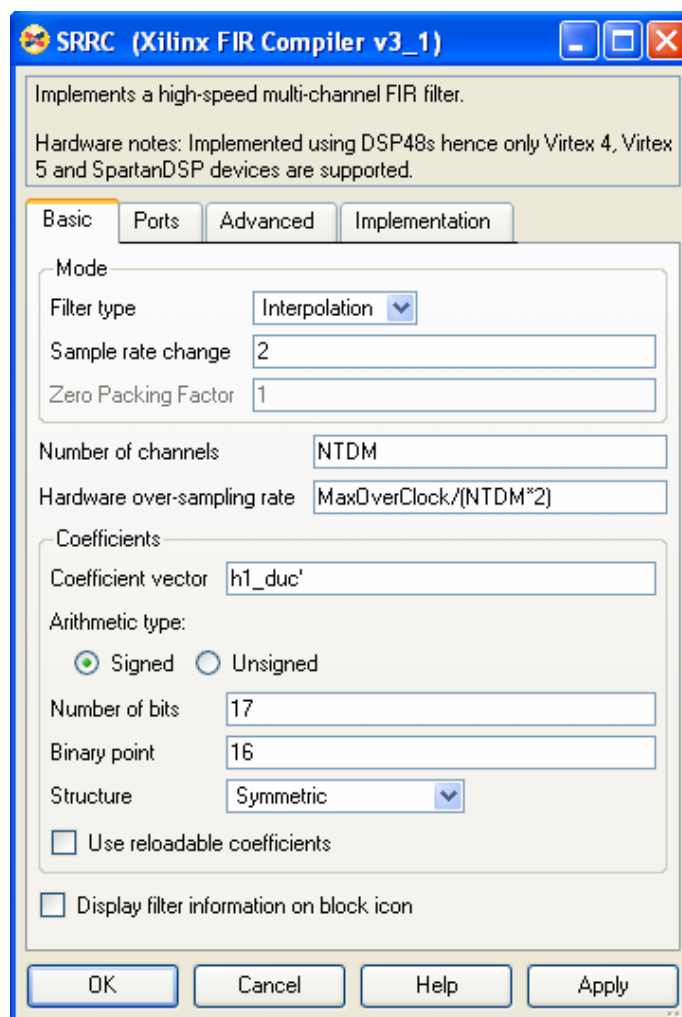


Figure 14: FIR Compiler GUI in System Generator

Table 9 lists a summary of important parameters for the WCDMA DUC design example. In this example, all four filters are interpolation-by-2 filters, and it can be considered a multi-channel design with two independent TDM channels. Knowing these parameters, it is straightforward to set the Filter Type, Sample Rate Change, and Number of Channel parameters.

Table 9: Parameters Used in System Generator FIR Block for WCDMA DUC

Parameter		RRC Filter	1 st Halfband	2 nd Halfband	3 rd Halfband
Filter Type		Interpolation	Interpolation	Interpolation	Interpolation
Sample Rate Change		2	2	2	2
Number of Channels		2	2	2	2
Hardware Over-Sampling Rate	Virtex-5 FPGA	24	12	6	3
	Spartan-DSP FPGA	8	4	2	1
Structure		Symmetric	Halfband	Halfband	Halfband

Setting the Hardware Over-Sampling Rate (HOR) parameter is the essential part of using the FIR compiler block. The HOR specifies the ratio between the FPGA clock rate and the faster data sample rate. The faster data sample rate is the faster of input data rate and the output data rate (consider the number of TDM channels as well). This ratio determines how much hardware folding can be done; in other words, how many MAC engines are required.

The first interpolator (RRC channel filter) in the Virtex-5 FPGA design is used as an example to demonstrate how to set the HOR parameter correctly. Since it is an interpolator, the faster data rate occurs at the output of the filter. The effective output data rate when accounting for the number of TDM channels is 15.36 MSPS. This is calculated by multiplying the output sample rate of 7.68 MSPS by number of channels (that is, 2). Therefore, the HOR is the FPGA clock rate of 368.64 MHz divided by 15.36 MSPS and is equal to 24. The HOR for the rest of the filters in Virtex-5 and Spartan-DSP FPGA designs are listed in [Table 9](#).

The coefficient vector can be specified using variables from the MATLAB workspace. Note that it has to be a row vector. The coefficient structure can be inferred from coefficients or specified explicitly as symmetric, non-symmetric, or halfband, as long as the vector of coefficients match the structure specified. All but the RRC filter use the efficient halfband structure.

Finally, the MAC engine in the FIR block is built using the various types of XtremeDSP™ slices, further discussed in the section [“Implementing Complex Multiplier Using DSP48 Slices.”](#) The XtremeDSP slice hardware structure is closely tied with the coefficient bit width consideration. For example, if extra system performance is required, when targeting the Virtex-5 family, one can increase the coefficient width up to 25 bits without triggering extra usage of the embedded DSP hardware resource. For the Virtex-4 and Spartan-DSP device families, up to 18 bit coefficients can be supported without incurring additional DSP resource. Of course, a slight increase of logic is expected for the coefficient storage.

Generate Sinusoids Using the DDS Compiler Block

The Direct Digital Synthesizer (DDS) Compiler block is used to generate a sinusoidal waveform needed by the frequency translate module. [Figure 15](#) shows a high-level view of the DDS Compiler block.

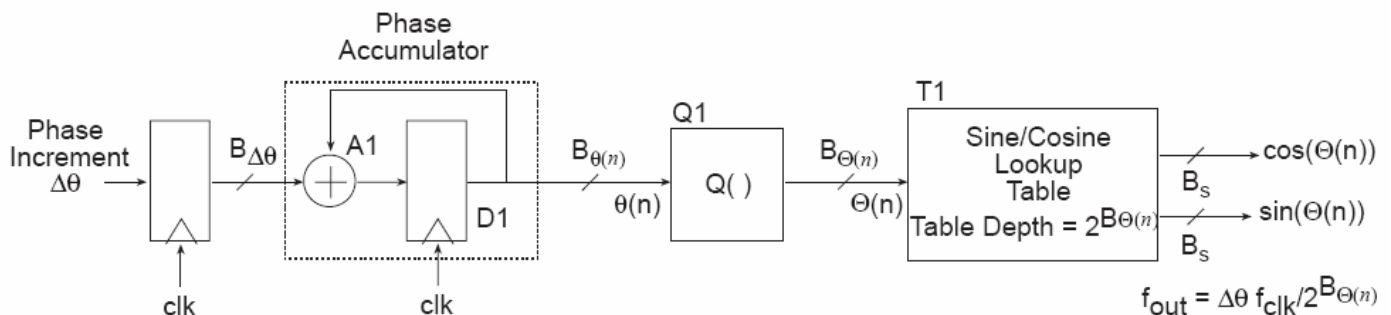


Figure 15: DDS Compiler Block Diagram

[Figure 16](#) shows the DDS Compiler v2.0 token from the block set library. A selective set of port configuration is described in [Table 10](#). To program this block, double click the DDS token. The block parameter dialog box is invoked as shown in [Figure 17](#). Again, the script that contains the generic parameters, such as SFDR, Fcr, and DucUpRate are described in [“Appendix C: Description of WCDMA Reference Design Files in System Generator.”](#)

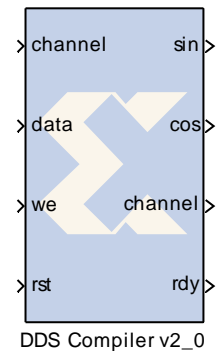


Figure 16: DDS Compiler Token in System Generator

Table 10: System Generator DDS Compiler Block Interface

Port Name	Port Direction	Port Description
channel	Input/Output	Specifies with which channel the current input (or output) is associated.
data	Input	Used for supplying values to the programmable frequency increment (or phase offset) memory.
we	Input	Write enable signal. Enables a write operation to the programmable frequency increment (or phase offset) memory.
rst	Input	Active-High synchronous reset. The internal registers of the block are reset when set to High.
Sin (±)	Output	Positive or negative Sine output value.
cos	Output	Cosine output value
rdy	Output	Sine/Cosine output data ready signal. Indicates when the output samples are valid.

Table 11 lists a summary of parameters for the DDS block. The DDS clock rate is the frequency at which the DDS block is clocked. In this single channel WCDMA example, it is equivalent to the DUC output sample rate, for example, 61.44 MHz. The SFDR defines the DDS output spectral purity in terms of the frequency domain requirements of the out-of-band noise level. It can be any number up to 115 dB and is defaulted to 102 dB in this configuration. The frequency resolution determines the granularity of the tuning frequency and is selected as 0.25 Hz, which

implies that the frequency accumulator width is $\left\lceil \log_2 \frac{61.44MHz}{0.25Hz} \right\rceil = 28$ bits.

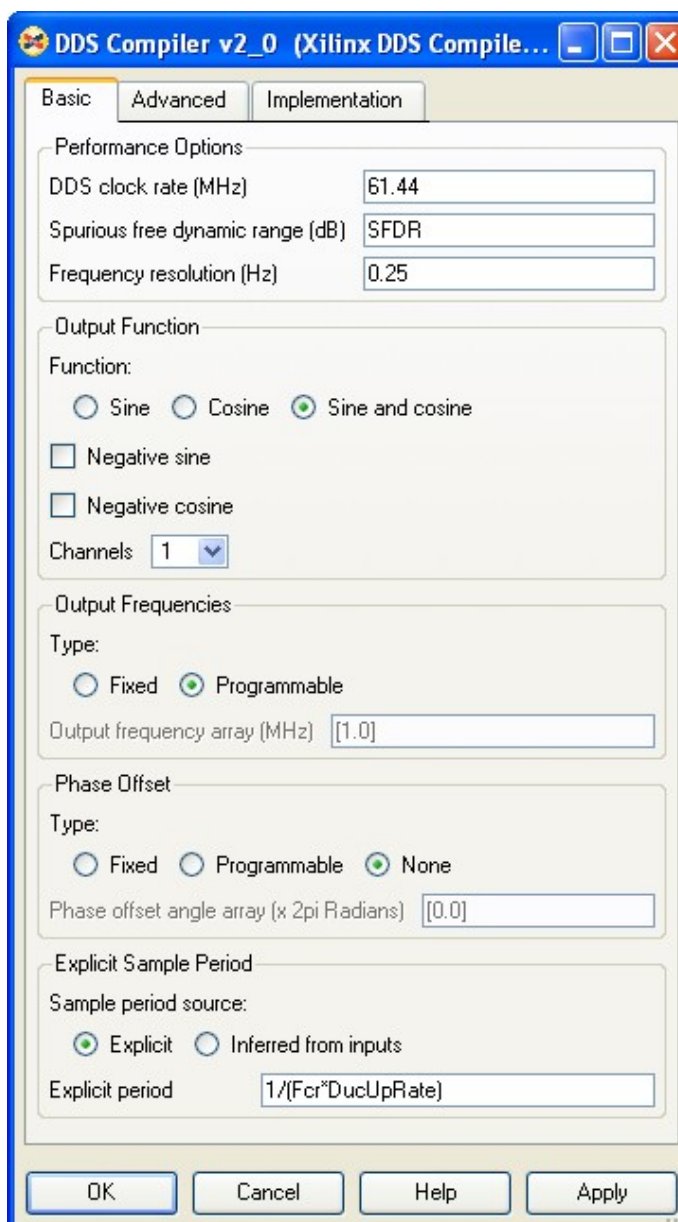


Figure 17: DDS Compiler GUI in System Generator

Table 11: Parameters Used in System Generator DDS Block for WCDMA DUC

DDS Clock Rate	61.44 MHz
Output Function	Both Sine (positive) and Cosine
SFDR	102 dB (can be specified up to 115 dB)
Frequency Resolution	0.25 Hz
Number of Channels	1
Output Frequency	Programmable
Noise Shaping	Taylor Series Corrected

The output frequencies and phase offset can be defined as constants or can be set dynamically through optional input ports. In this design example, the output frequency is programmable and the phase offset is set to zero. Both Sine and Cosine output are required. *Taylor Series Corrected* setting is selected for the noise shaping method, since it provides better SFDR than phase dithering while using a much smaller lookup table.

The output frequency value is defined in terms of cycles per sample. In other words, it is the normalized frequency. To generate the output frequency at F_c MHz from the DDS, the user has to feed the value of $\text{round}(F_c / F_s * 2^{\text{accumulator_width}})$ to the data port. For example, to shift the waveform to -5 MHz output frequency with a DDS clock rate of 61.44 MHz and a 28-bit accumulator width (0.25 Hz frequency resolution), the value of $\text{round}(-5/61.44 * 2^{28}) = -21845333_{(10)} = 1110\ 1011\ 0010\ 1010\ 1010\ 1010\ 1011_{(2)}$ has to be latched into the data port.

Note: The use of a negative frequency in this example means that the vector formed by the complex output of the DDS will rotate *backward*, that is, clockwise.

Implementing Complex Multiplier Using DSP48 Slices

Denote the sinusoidal waveform generated from the DDS as $A(n) = A_r(n) + j * A_i(n)$, the interpolator output as $B(n) = B_r(n) + j * B_i(n)$, where $A_r(n) = \cos(\omega_0 n)$ and $A_i(n) = \sin(\omega_0 n)$, B_r and B_i are the in-phase and quadrature components after interpolation filters. The mixer output is:

$$P = A \times B = (A_r + j \cdot A_i) \times (B_r + j \cdot B_i) = (A_r B_r - A_i B_i) + j \cdot (A_r B_i + A_i B_r) \equiv P_r + j \cdot P_i$$

Equation 2

P_r and P_i are defined as the real and imaginary part of the result from the complex multiplication. Each P_r or P_i term requires two multiplications and one addition/subtraction. This operation can be implemented using Xilinx high performance XtremeDSP slices, a.k.a. DSP48s. To implement the complex multiplication block efficiently, the engineer has to understand the architecture of the DSP48 slice. Since the clock rate is different for designs in various device families, the versatile DSP48 slice is briefly introduced for each family first, and then its corresponding implementation is discussed.

Using DSP48E to Implement Complex Multiplier for the Virtex-5 FPGA Family

The DSP48E slice in the Virtex-5 devices [Ref 2] not only can accelerate the algorithm but also lowers power consumption. It features a 25-bit by 18-bit, two's complement multiplier with full precision 48-bit result, which enables higher precision for greater dynamic range support. It also has a three-input, flexible 48-bit adder/subtractor with optional registered accumulation feedback. The block diagram of the DSP48E slice is illustrated in Figure 18.

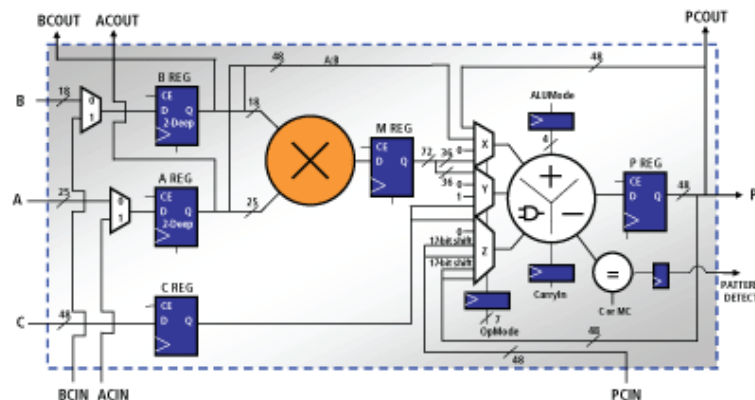


Figure 18: Virtex-5 FPGA DSP48E Block Diagram

The clock rate is 368.64 MHz, 6 times the required output sampling rate of 61.44 MHz, so there are six available operation cycles per output sample. Only one DSP48E slice is necessary to

perform the complex multiplication which requires four clock cycles. Moreover, the DSP48E slice has a built-in symmetric round function. The extra two cycles can be used to round the output of the complex multiplication.

The DDS output width is a function of the SFDR parameter and is calculated as $\left\lceil \frac{SFDR}{6} \right\rceil + 1$ for Taylor Series Corrected method and $\left\lceil \frac{SFDR}{6} \right\rceil$ for noise dithering method.

The width can be as wide as 20 bits for up to 115 dB SFDR. Therefore, it is generally a good idea to feed in the complex exponential from the A port of the DSP48E, as it can handle up to 25-bit input.

To stream in, align, and mix the DDS output and the interpolator output, some data muxing and scheduling are necessary. Users are encouraged to look at the reference design and work out the details. Figure 19 shows the System Generator design of the mixer block.

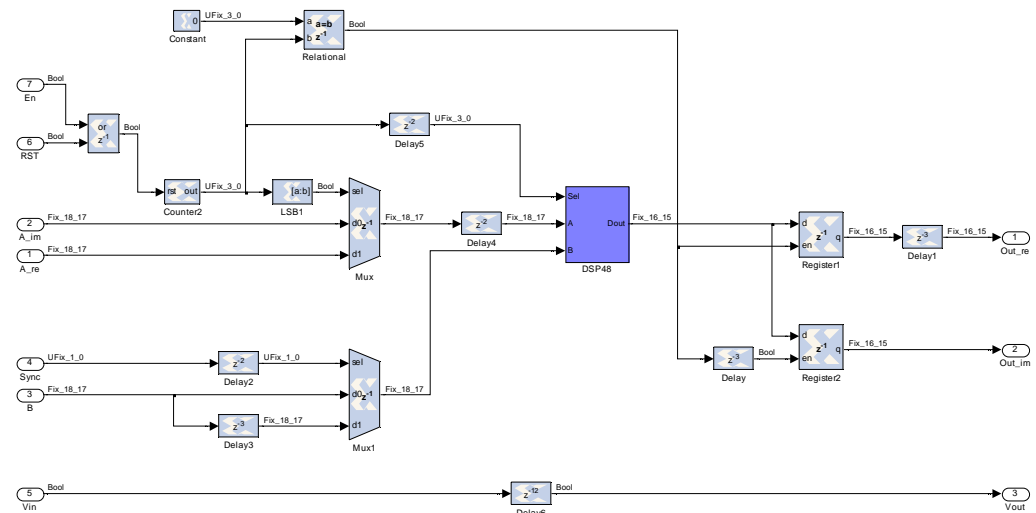


Figure 19: Virtex-5 FPGA Mixer Implementation for WCDMA DUC

Within the DSP48E slice, the operation for each of the six clock cycles is listed as follows:

Cycle 1: $P \leftarrow A_r \cdot B_r$

Cycle 2: $P \leftarrow P - A_i \cdot B_i$

Cycle 3: $P \leftarrow \text{round}(P)$

Cycle 4: $P \leftarrow A_r \cdot B_i$

Cycle 5: $P \leftarrow P + A_i \cdot B_r$

Cycle 6: $P \leftarrow \text{round}(P)$

As can be seen, the real product $P_r = \text{round}(A_r \cdot B_r - A_i \cdot B_i)$ is obtained after cycle 3, and the imaginary product $P_i = \text{round}(A_r \cdot B_i + A_i \cdot B_r)$ is computed after cycle 6. The System

Generator DSP48E primitive usage screen shot is shown in [Figure 20](#). Various opcodes are multiplexed into the primitive to control different DSP48E operations.

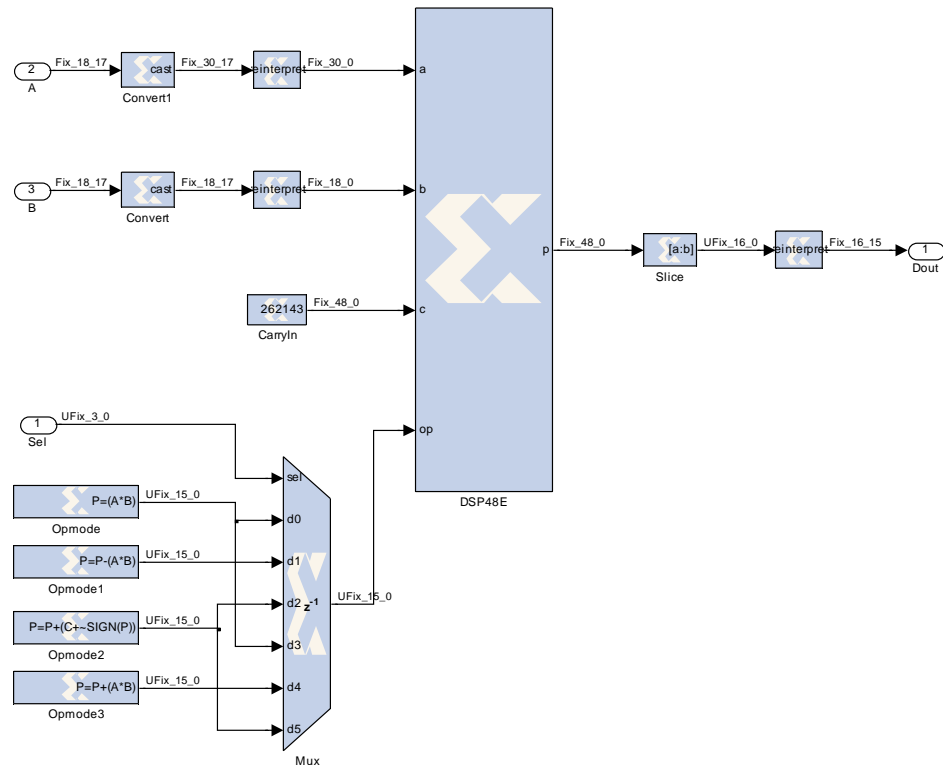


Figure 20: Virtex-5 FPGA DSP48E and Opmode Selection Block

Using DSP48A to Implement Complex Multiplier for the Spartan-DSP FPGA Family

The DSP48A Slice [\[Ref 3\]](#) provides an 18-bit by 18-bit, two's complement multiplier with full precision 36-bit results, an 18-bit pre-adder, and a two-input 48-bit post-adder/accumulator with optional registered accumulation feedback. The block diagram of the DSP48A Slice is shown in [Figure 21](#).

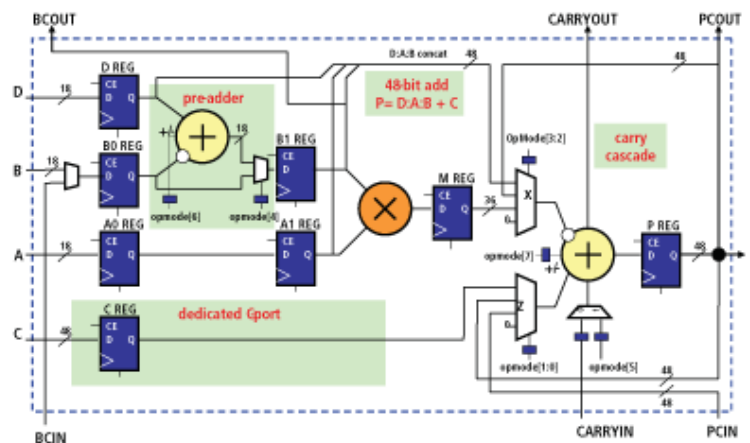


Figure 21: Spartan-DSP FPGA DSP48A Block Diagram

Recall that for the WCDMA design targeting the Spartan-DSP technology, the clock rate is 122.88 MHz, two times the data sampling rate of 61.44 MHz. Each of the real and imaginary terms require two MAC operations, so a total of two DSP48A slices are needed to perform the complex multiplication.

Figure 22 shows the System Generator design of the mixer block. The details on how to align and mix the DDS and interpolator outputs are again left for users to work out and can be found in the reference design.

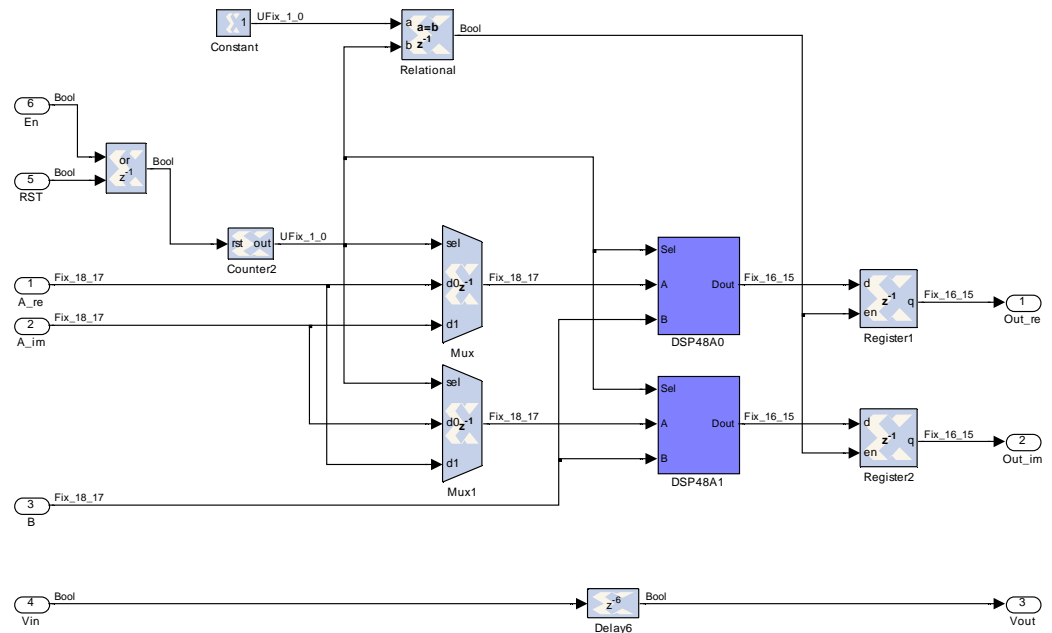


Figure 22: Spartan-DSP FPGA Mixer Implementation for WCDMA DUC

The rounding function can be performed outside of the DSP48A slice with additional logic. An innovative way to round without introducing extra hardware is chosen. In the first cycle operation when computing the partial product, the rounding constant (0.4999...) and a random bit of 0 or 1 with equal probability were added through the C port and Cin port, respectively. When handling a number that is halfway between two nearest integers (for example, 2.5), 50% of the time it is rounded towards infinity (that is, 3), and 50% of the time it is rounded towards 0 (that is, 2). Therefore, statistically this rounding method is symmetric and does not introduce any DC bias. The random bit can be generated either from a LFSR or fetched from other random sources.

The operation within the DSP48A slice for computing the real product is:

$$\text{Cycle 1: } P \leftarrow A_r \cdot B_r + C + C_{in}$$

$$\text{Cycle 2: } P \leftarrow P - A_i \cdot B_i$$

And for computing the imaginary product, the operation is:

$$\text{Cycle 1: } P \leftarrow A_r \cdot B_i + C + C_{in}$$

$$\text{Cycle 2: } P \leftarrow P + A_i \cdot B_r$$

After cycle 2, the calculation for both the real and imaginary terms is complete and appears at the DSP48A output. The DSP48A primitive and Omode selections are shown in Figure 23.

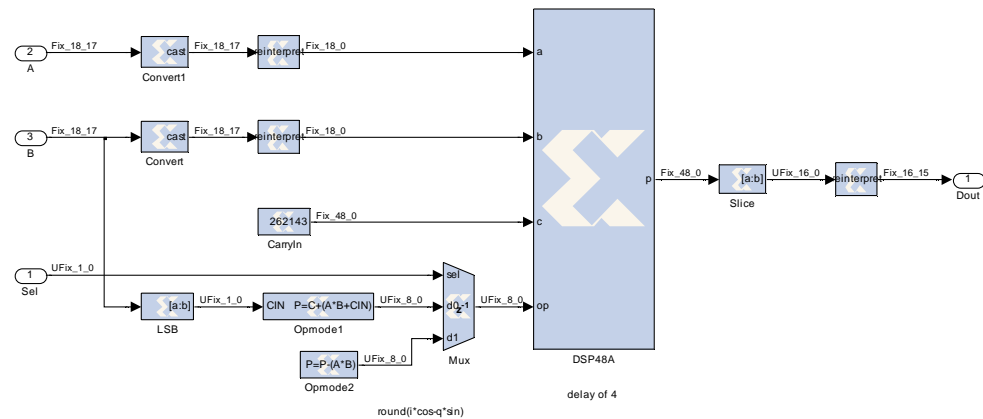


Figure 23: Spartan-DSP FPGA DSP48A and Opmode Selection Block

After the complex mixing, the frequency shifts the TDM data stream, then goes through the time division demultiplexing process. The in-phase and quadrature phase outputs are obtained using capture registers and down samplers.

Using HDL Design Flow

The design concept is similar to that described in DUC Implementation in “Using System Generator Design Flow” section. In this section, the focus is on introducing the FIR compiler v3.1 and the DDS compiler v2.0 in the CORE Generator [Ref 4].

FIR Compiler CORE Generator GUI and Parameters

The FIR compiler v3.1 can be found under the Digital Signal Processing Function, Filters category, as shown in Figure 24.

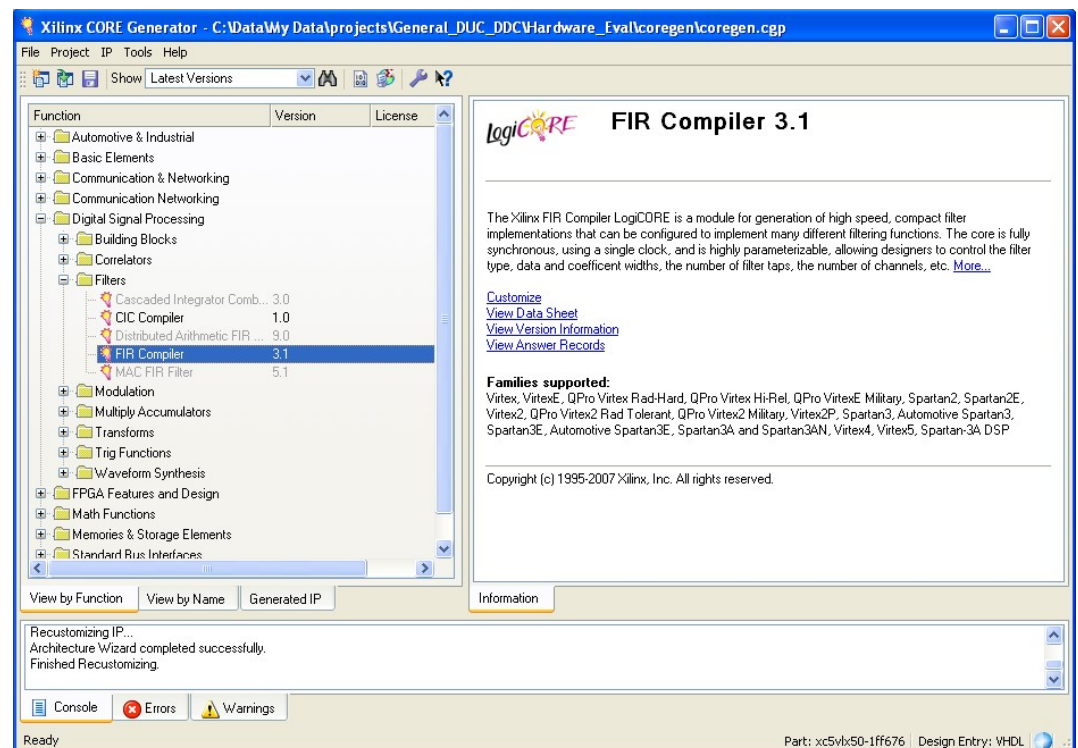


Figure 24: FIR Compiler in CORE Generator

Double click the FIR Compiler token to invoke the core's GUI (Figure 25). The FIR Compiler GUI consists of the informational panels on the left side and the specification screens on the right.

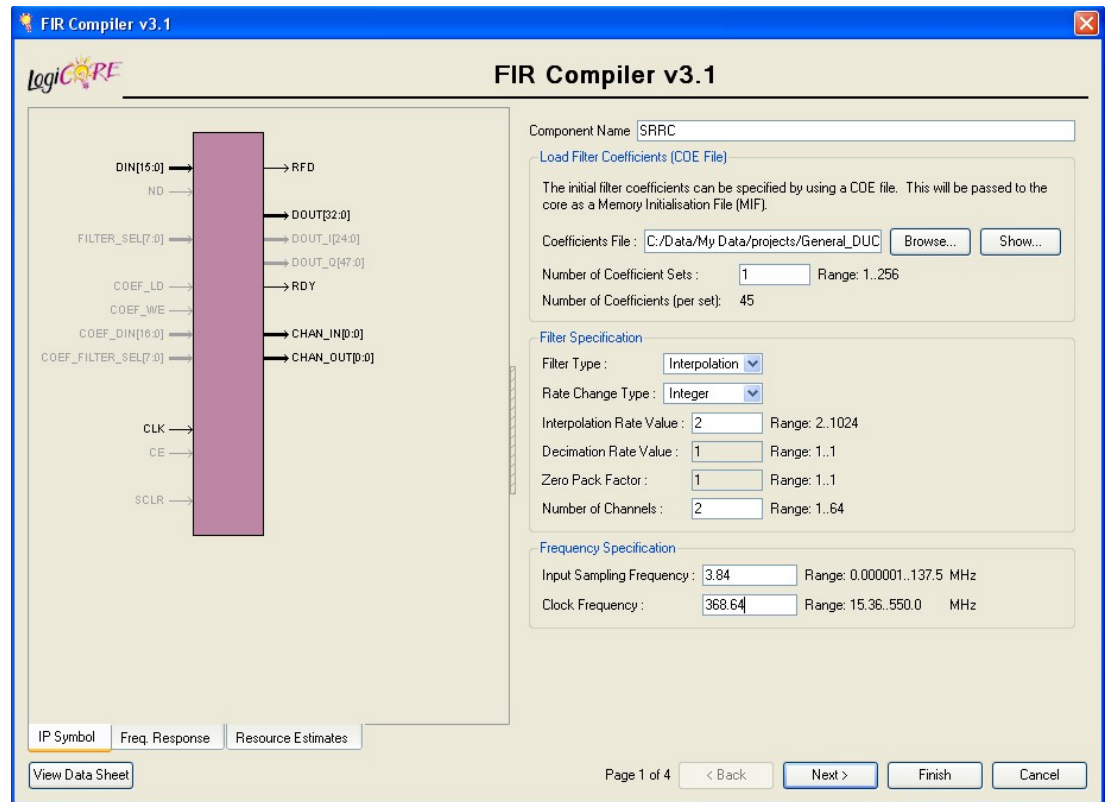


Figure 25: FIR Compiler Screen Shot (Page 1)

Note: If using the Project Navigator, the appearance of the GUI will be somewhat different. In Project Navigator, click the “New Source” icon and select “IP” to create a CORE Generator core.

Filter Information Panel

The three left side tabs, IP Symbol, Frequency Response, and Resource Estimates, lead to their corresponding informational panels. The first tab, IP Symbol, shows the core interface description. The filter's frequency response in magnitude and the passband and stopband filter response analysis boxes are displayed in the second tab. This tab is the default tab at start-up.

The third tab presents the resource estimation information which lists the number of DSP slices used along with the number of block RAM counts required to implement the design. This is a very useful function in filter design when the system performance and hardware usage tradeoff are evaluated. The usefulness of this tool has already been shown in “[Architectural Consideration](#)”. The value of the resource estimator is proven again in the CDMA2000 example, described in the “[Filter Design Strategy](#)” section.

As the title suggests, the usage counts are estimates only. Equations are used to model the expected core implementation structure to calculate the FPGA real estate consumption. Users can obtain a more accurate report on all resource usage by generating the core with the Resource Utilization option (within the CORE Generator) turned on.

Note that the panels can be enlarged to the full GUI window size by dragging fully to the right to facilitate easy viewing of the presented information, particularly for the frequency response window.

Filter Specification Panel

The 4-page Filter Specification panels at the right hand side define the basic configuration and performance of the filter. Here we selectively described the fields that are important to the filters in the DFE designs.

Load Filter Coefficients (COE File)

The filter coefficients are supplied to the FIR compiler using a coefficient file with a COE extension. This is an ASCII text file with a single-line header that defines the radix of the number representation used for the coefficient data, followed by the coefficient values themselves. The filter coefficient file format is shown below for an N-tap filter. The filter coefficients can be supplied as integers in base-10, base-16, or base-2 representation. This corresponds to *coefficient_radix*=10, *coefficient_radix*=16, and *coefficient_radix*=2, respectively.

```
radix=coefficient_radix;
coefdata=
a(0),
a(1),
a(2),
...
a(N-1);
```

The COE file is passed into the core as a Memory Initialization File (.mif). When clicking the “Show” button, the filter coefficients are displayed in a pop-up window. Since no reloadable coefficient is used in the DFE designs, only one set of coefficients is required.

Filter Specification, Frequency Specification, and Datapath Options

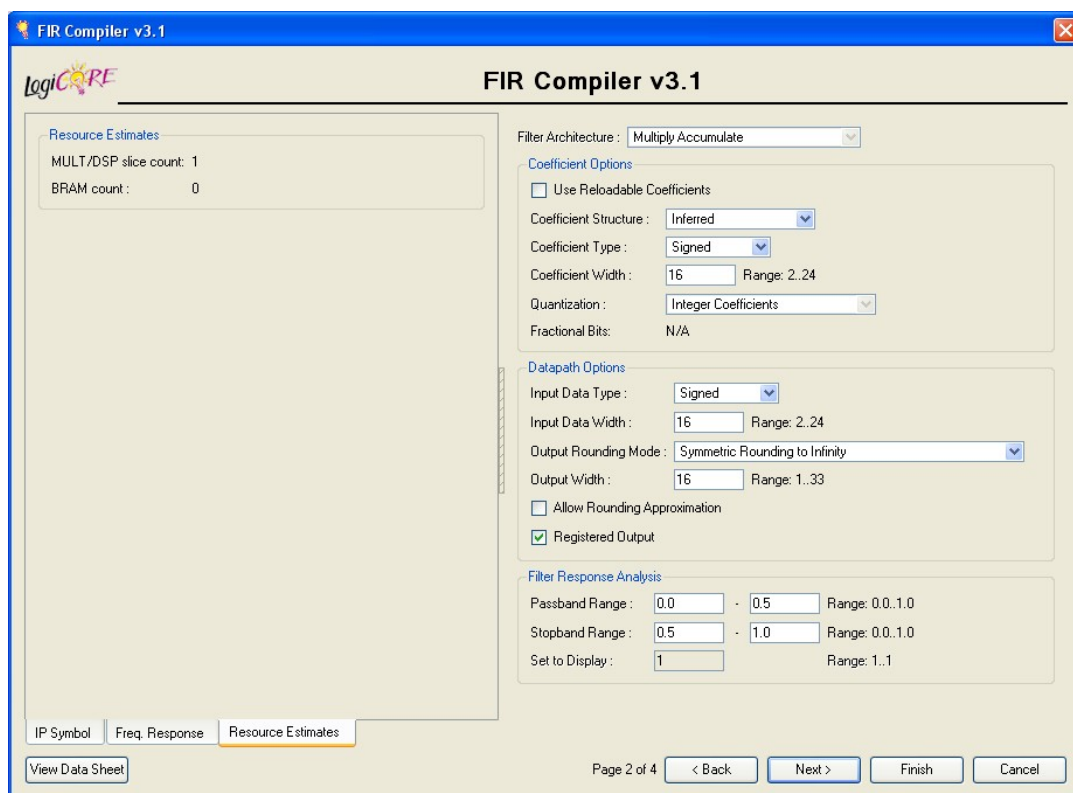
Table 12 lists the parameters used to specify the filter characteristics and datapath for the WCDMA DUC targeting Virtex-5 device family. Most fields are self-explanatory. For definitions for each field, refer to the FIR compiler v3.1 data sheet [Ref 5]. The first and second page of the RRC filter GUI is shown in Figure 25 and Figure 26. The control, memory, and DSP Slice Column options listed in page 3 of the GUI are kept as the default.

Table 12: Parameters Used in CORE Generator FIR Compiler for WCDMA DUC

Parameter		RRC Filter	1 st Halfband	2 nd Halfband	3 rd Halfband
Filter Type		Interpolation	Interpolation	Interpolation	Interpolation
Rate Change Type		Integer	Integer	Integer	Integer
Sample Rate Change		2	2	2	2
Number of Channels		2	2	2	2
Input Sampling Frequency		3.84 MHz	7.68 MHz	15.36 MHz	30.72 MHz
Clock Frequency	Virtex-5 FPGA	368.64 MHz			
	Spartan-DSP FPGA	122.88 MHz			
Output Rounding Mode		Symmetric Rounding to Infinity			

Filter Summary

The final page of the filter specification panel provides summary information about the core parameters selected, and presents the calculated cycle-latency value. The summary page for the RRC filter is shown in Figure 27. After all the selections are finalized, the user can hit the “finish” button and then the netlists will be generated.



FIR Compiler v3.1

LogiCORE

Resource Estimates

MULT/DSP slice count: 1
BRAM count: 0

Filter Architecture: Multiply Accumulate

Coefficient Options

☐ Use Reloadable Coefficients

Coefficient Structure: Inferred

Coefficient Type: Signed

Coefficient Width: 16 Range: 2..24

Quantization: Integer Coefficients

Fractional Bits: N/A

Datapath Options

Input Data Type: Signed

Input Data Width: 16 Range: 2..24

Output Rounding Mode: Symmetric Rounding to Infinity

Output Width: 16 Range: 1..33

☐ Allow Rounding Approximation

☒ Registered Output

Filter Response Analysis

Passband Range: 0.0 - 0.5 Range: 0.0..1.0

Stopband Range: 0.5 - 1.0 Range: 0.0..1.0

Set to Display: 1 Range: 1..1

IP Symbol Freq. Response **Resource Estimates**

View Data Sheet

Page 2 of 4 < Back Next > Finish Cancel

Figure 26: FIR Compiler Screen Shot (Page 2)

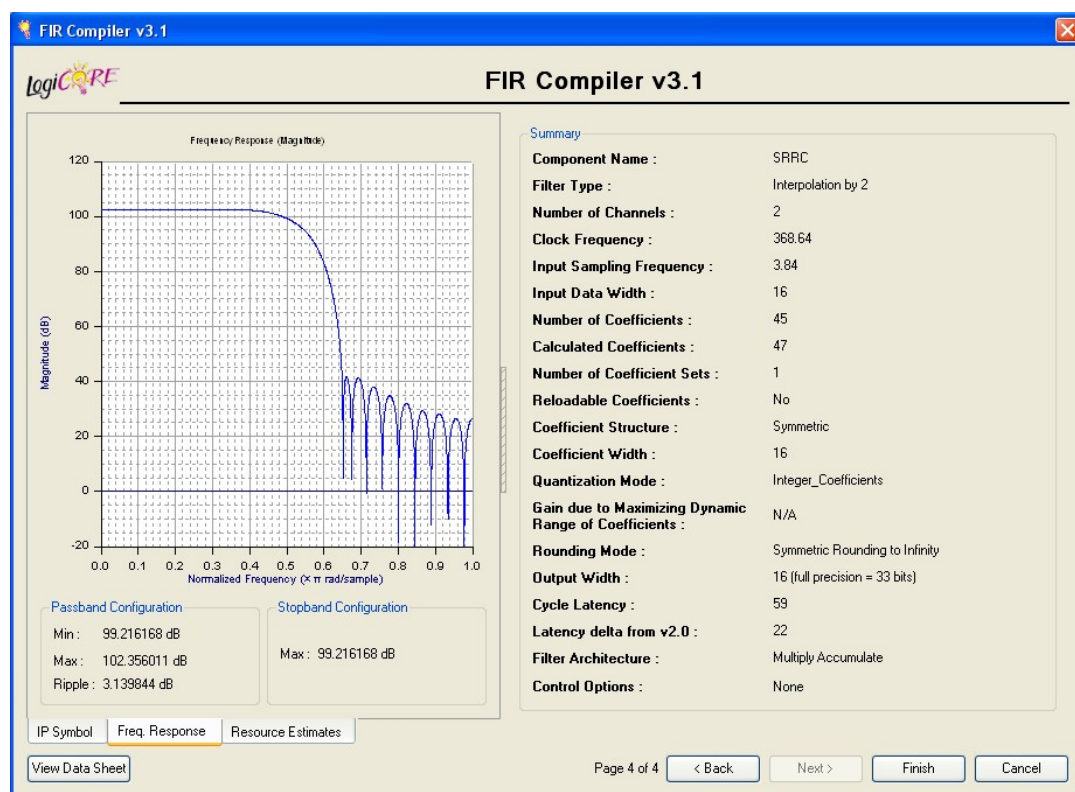


Figure 27: FIR Compiler Screen Shot (Page 4)

DDS Compiler CORE Generator GUI and Parameters

The DDS compiler v2.0 can be found under the Digital Signal Processing Function, Waveform Synthesis category. The DDS compiler core GUI also includes two parts: the informational panels on the left side, and the specification screens on the right.

DDS Information Panel

The IP Symbol tab shows the core interface. To add or remove some control signals, users can go to page 4 of the Specification panel to select or deselect the handshaking, clear, and clock enable options.

DDS Specification Panel

The 6-page Filter Specification panels at the right hand side define the basic configuration and performance of the DDS. [Table 13](#) lists the parameters used to specify the DDS characteristics for the WCDMA DUC targeting Virtex-5 device family. These include the DDS function, performance options, output frequencies, and phase offset angles. Refer to the DDS compiler v2.0 data sheet [\[Ref 6\]](#) for detailed definition for each field. The first page of the DDS GUI is shown in [Figure 28](#).

Table 13: Parameters used in CORE Generator DDS Compiler for WCDMA DUC

Output Selection	Both Sine (positive) and Cosine
Channels	1
DDS Clock Rate	61.44 MHz
SFDR	105 dB (can be specified up to 115 dB)
Frequency Resolution	0.25 Hz
Phase Increment	Programmable
Phase Offset	None
Noise Shaping	Taylor Series Corrected

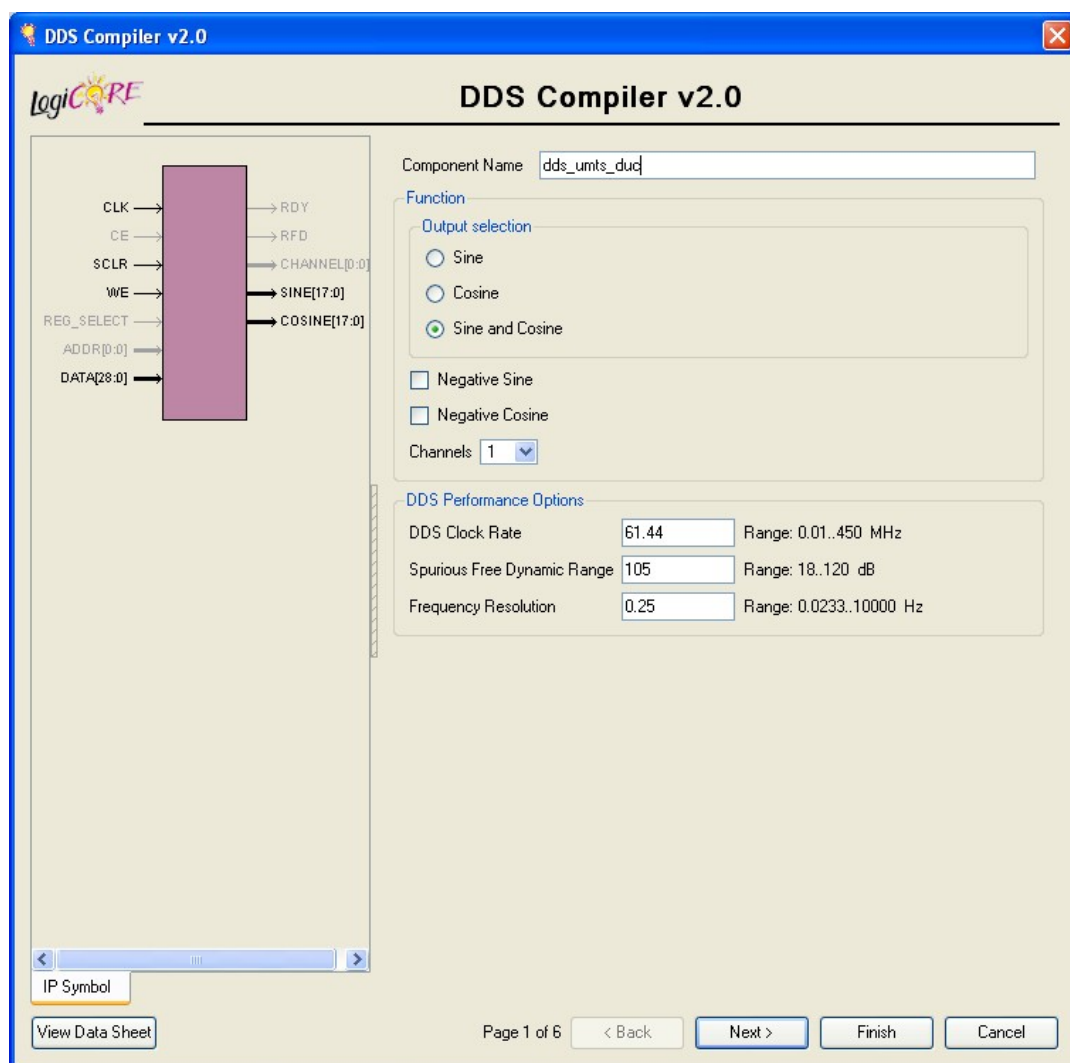


Figure 28: DDS Compiler Screen Shot (Page 1)

DDS Summary

The last page of the DDS GUI provides summary information about the core parameters selected, latency value, and DSP48s and block RAMs counts. The summary page for the DDS in the WCDMA DUC example is shown in [Figure 29](#).

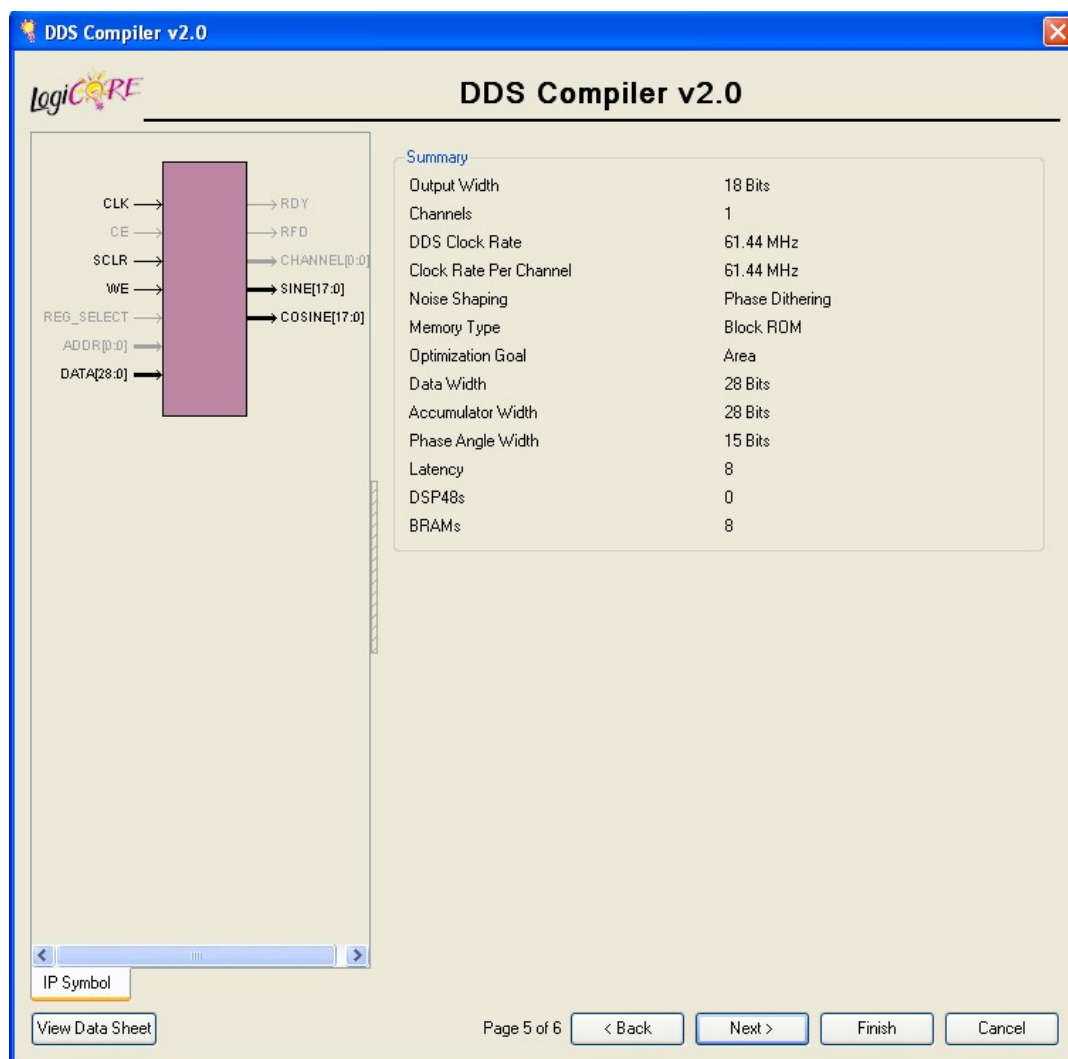


Figure 29: DDS Compiler Screen Shot (Page 5)

Digital Down-Converter

The DDC translates one or more intermediate IF channels from a set of specified center frequencies to 0 Hz. It also performs decimation and matched filtering to remove adjacent channels and maximize the received signal-to-noise ratio (SNR). For the reference design, the input to the DDC is a real signal sampled at $16 \times F_{\text{chip}} = 61.44$ MSPS and quantized to 14 bits. The complex baseband output are generated at a sampling rate of $2 \times F_{\text{chip}} = 7.68$ MSPS.

Performance Requirements

A summary of requirements for the uplink receive path is listed in [Table 14](#).

Table 14: Target Specification for WCDMA Uplink Receive Path

Parameter	Value	Comments
Carrier Bandwidth	5.0 MHz	
Number of Carriers	1 carrier	
IF Sample Rate	61.44 MCPS	16×3.84 MSPS
DDC Output Rate	7.68 MSPS	$2 \times F_{\text{chip}}$
Input Signal Quantization	14-bit	Real

Table 14: Target Specification for WCDMA Uplink Receive Path (Continued)

Parameter	Value	Comments
Output Signal Quantization	16-bit I and Q	Complex
Mixer Properties	Tunability: Variable Resolution: ~0.25 Hz SFDR: up to 115 dB	Various SFDR are supported by the DDS Compiler core (28 bits of frequency accumulator width)

The DDC input is assumed to be real, directly coming from the ADC. A simplified high-level block diagram of the WCDMA DDC in the uplink receive path is illustrated in [Figure 30](#).

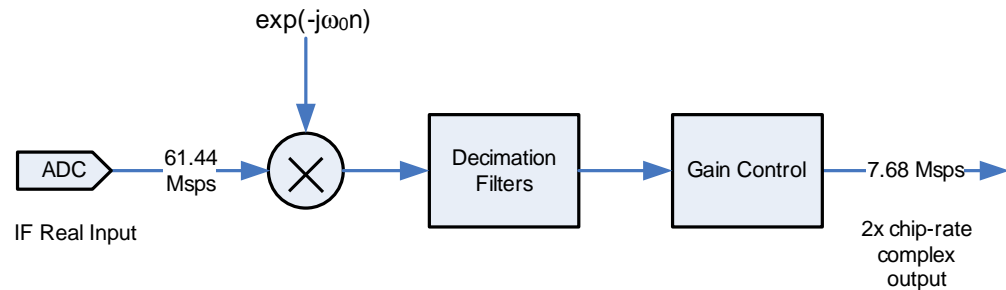


Figure 30: High-Level Block Diagram for WCDMA DDC

Frequency Translation

The mixer translates the real bandpass input signal from some intermediate frequency to a complex baseband signal centered at 0 Hz. Mathematically, the real input signal $X(n)$ is multiplied by a complex exponential $e^{-j\omega_0 n} = \cos(\omega_0 n) - j \sin(\omega_0 n)$ to produce a complex output signal with real and imaginary components given by $Y_r(n) = X(n) \cos(\omega_0 n)$ and $Y_i(n) = -X(n) \sin(\omega_0 n)$, respectively. The sinusoidal waveforms required to perform the mixing process is obtained by using the DDS. The discussion on the DDS can be found in the section [“Generate Sinusoids Using the DDS Compiler Block”](#).

DDC Filter Design

Architectural Consideration

As with the DUC, there are several architectural options for the receive signal path. The decimators in the DDC need to down sample the IF data from 61.44 MHz back to 2x chip rate. The factor of $61.44/7.68 = 8$ can be partitioned in a couple of ways.

The channel filter is the last component of the decimation chain. As mentioned previously, downsampling by eight at once will most likely result in an extremely long filter length and result in an inefficient hardware implementation. This is because of the reasons outlined in [“Appendix A: Determining the Number of Interpolation/ Decimation Stages.”](#) Additionally, separating the shaping filter allows the remaining stages to be implemented as two halfband filters, which are advantageous because half of the coefficients are zero. Instead, it was designed as a decimator with a rate of 2, matched to the channel filter in the DUC.

The remaining down-sample-by-4 task can be done directly, or by breaking it down to two stages of halfband filters. Again, we use a combination of FDATool and the FIR compiler GUI resource estimate tab (see [“FIR Compiler CORE Generator GUI and Parameters”](#)) to determine the preferred configuration, as shown in [Table 15](#). The channel filter is identical in either configuration so it is excluded from the table. [Table 15](#) shows that configuration 2 (shown in [Figure 31](#)) is the architecture of choice since it consumes less hardware.

Table 15: Filter Length and Sampling Rate vs. Hardware Utilization for WCDMA DDC

Anti-Aliasing Filter	Filter Length & Sample Rate for 1 st Filter	Filter Length & Sample Rate for 2 nd Filter (if any)	Total Required MACs for Virtex-5 FPGA	Total Required MACs for Spartan-DSP FPGA	Architecture of choice
Configuration 1	31 taps (↓4) 61.44 MSPS		3	5	
Configuration 2	11 taps (↓2) 61.44 MSPS	19 taps (↓2) 30.72 MSPS	2	4	✓

Note: Dual channel mode is used for the filters in both configurations as the input/output data is in a complex format.

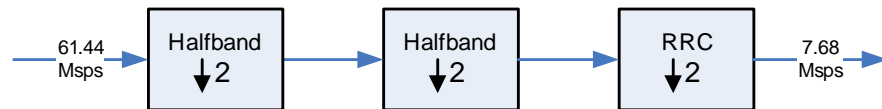


Figure 31: WCDMA DDC Decimation Filter Architectural Choice

The software included in the reference design bundle can be used to further analyze the performance of either configuration. In the following sections, the focus on the architecture that partition the filter chain to three stages of decimator, each with a rate change of 2.

Notes on Estimating Filter Size

To estimate the number of multipliers (or DSP48s) in a filter, one can calculate the number of multiply instructions per second (MIPS) required for a filter structure and then divide by the number of MIPS that can be processed per DSP48. Generally, the number of MIPS that can be processed by a DSP48 is F_c , the system clock frequency, since the DSP48 usually operates at the highest clock speed of the design. This yields the formula:

$$D = \left\lceil N \cdot C \cdot \frac{1}{L} \cdot \frac{f_s}{f_c} \right\rceil$$

where

- D is the number of DSP48s required;
- C is the number of channels;
- N is number of nonzero taps in filter;
- L is the interpolation rate (1 for non-interpolating filters);
- f_s is the output sample rate of the filter (per channel);
- f_c is the system clock frequency

For configuration 1 in Table 15, $C = 2$ (the filter is being used in a 2-channel mode to compute the real and imaginary parts of the complex input), $N = 31$, $L = 1$ (because this is a decimation filter), $f_c = 368.64$ MHz, $f_s = 61.44/4 = 15.36$ MHz. This yields $D = \text{ceil}(2.583) = 3$ DSP48s required.

For non-interpolating, non-decimating filters with symmetric coefficients, N can be halved if pre-adders are used.

The formula is an estimate and may not include extra DSP48s required for rounding or other architectural reasons.

First Half-Band Decimator

The first decimation filter is designed to reduce the sampling rate by 2. It is realized using the computationally efficient half-band filter structure, where every odd indexed coefficient is zero except for the center tap and even indexed coefficients are symmetric.

The input sample rate of the filter is 61.44 MSPS, and the output sample rate is 30.72 MSPS, 8x the chip rate. The passband was set to $F_{\text{pass}} = 3.84 \times 1.22/2 = 2.34$ MHz and the passband ripple is chosen to be 0.0002 dB. An 11-tap halfband filter was designed to this specification using the FDATool.

The frequency (magnitude) response of the filter is shown in Figure 32. With the coefficient quantized to 16 bits, it can be seen that the stopband attenuation is below 100 dB.

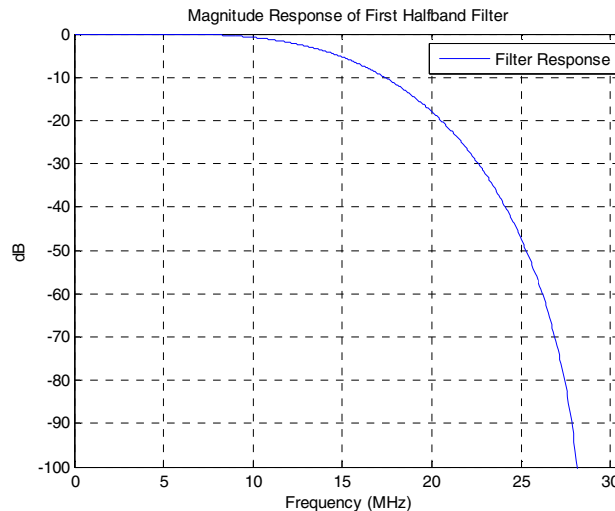


Figure 32: Magnitude Response of First Halfband Decimator

Second Half-Band Decimator

The second halfband filter further decimates the data by two, from 30.72 MSPS to 15.36 MSPS. The passband edge is set to 2.34 MHz, the same as that in the first halfband filter, determined by the chip rate and the RRC roll-off. The passband ripple is set to 0.0001 dB. A 19-tap halfband filter with coefficients quantized to 16 bits is sufficient to meet the requirements. From the magnitude response shown in Figure 33, it can be seen that the filter can reject the spectral image replica below 100 dB level.

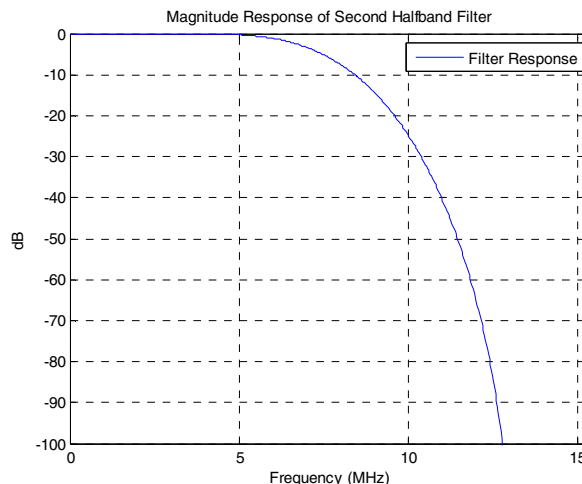


Figure 33: Magnitude Response of Second Halfband Decimator

Channel Filter

The receiver channel filter is an RRC filter matched to the one used in the uplink transmitter. The filter also provides a decimation-by-2 to reduce the sampling rate from 15.36 MSPS to 7.68 MSPS, 2 times the chip rate. This 2x over-sampling rate is needed in the timing recovery process to avoid the signal loss due to the sampling point misalignment. The filter was designed to have a cutoff frequency of 1.92 MHz and a roll-off of 0.22 MHz. By using a Chebyshev window with 50 dB sidelobe attenuation, a 95-tap filter with greater than 85 dB stopband attenuation is designed (Figure 34).

An interesting note here is that the filter was designed with steeper stopband attenuation than required. In this case, a filter order of 60 to 70 is probably sufficient to provide good system performance. In hardware, the extra 25 to 35 taps comes at no extra expense. The sampling rate of the filter is relatively low compared to the FPGA operation frequency, and it was found that a single processing engine can handle all the MAC operations. We chose to use the maximum taps allowed while not increasing the hardware consumption.

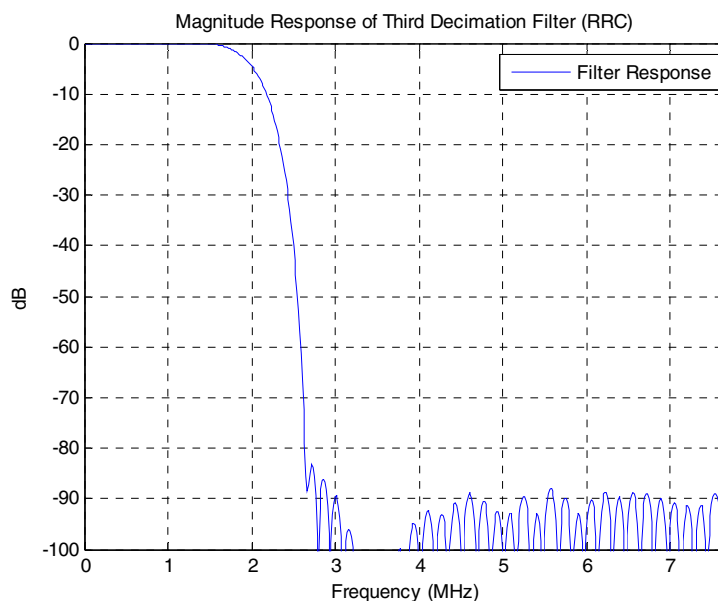


Figure 34: Magnitude Response of WCDMA DDC Channel Filter

Composite Filter Response

The specifications of all three filters are summarized in Table 16. When the real output of the DUC is used as the input of the DDC, the power spectral density of the DDC output is shown in Figure 35.

Table 16: Summary of WCDMA DDC Filters

Filter Stage	Pass Band Fpass (MHz)	Sample Rate Fs (MSPS)	Peak-to-Peak Ripple (dB)	Filter Order
First Halfband	2.34	61.44	0.0002	10
Second Halfband	2.34	30.72	0.0001	18
Windowed RRC	1.92	7.68	0.02	94

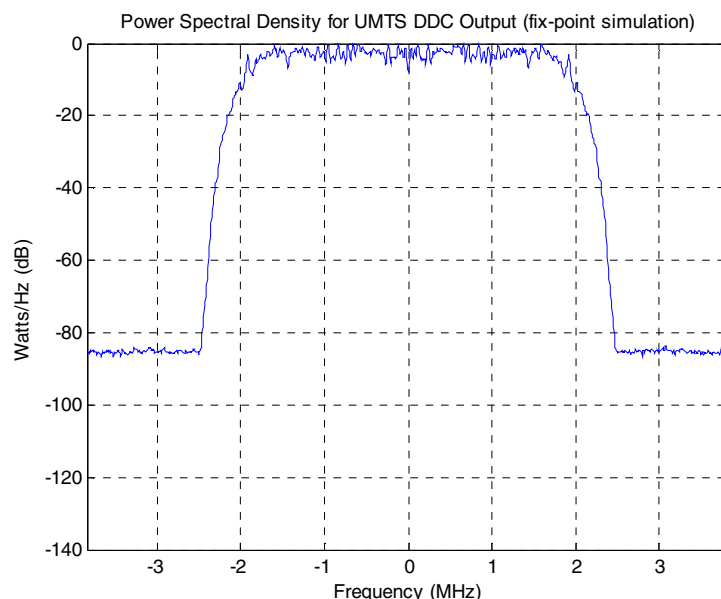


Figure 35: PSD for WCDMA DDC Output

DDC Performance

This section discusses the simulation results generated from several conformance tests defined in the Section 7 of the 3GPP standard [Ref 1]. It also defines that the BER shall not exceed 0.001 for those tests. The corresponding SNR requirements can be derived through the BER vs. SNR curve found in most communication literatures. A 25 dB de-spreading gain (since the reference measurement channel data rate is 12.2 kHz) is assumed and added to the final SNR calculation.

Wideband Data Test

The input to the wideband data test is a three-carrier WCDMA composite signals with total noise power (in a 61.44 MHz bandwidth) set at the level of 30 dB below the desired signal, as shown in Figure 36. The band of interest is the carrier centered at 15 MHz. Figure 37 illustrates the DDC output after channelizing the signals from adjacent bands and removing the noise. It can be observed that the noise floor is more than 85 dB down. The performance is further evaluated by the EVM figure. Comparing the received signal from the transmitted reference signal, the EVM is about 1.22%. The AWGN in-band noise at the input of the DDC is a more significant contributor to the EVM than the adjacent channels. The SNR after accounting for the de-spreading gain is 63.27 dB. Note the signal-to-interference-and-noise ratio (SINR) is a more appropriate term in this context, but it is used interchangeably throughout this document for convenience.

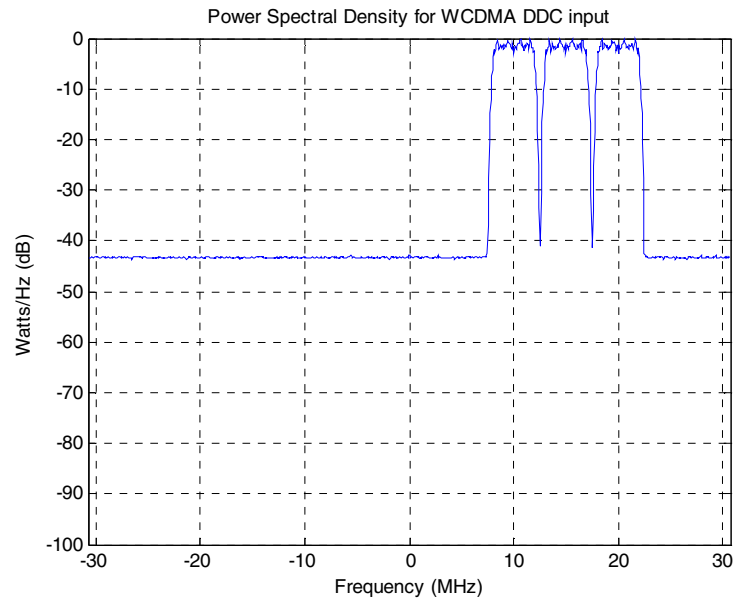


Figure 36: WCDMA DDC Input for Wideband Data Test

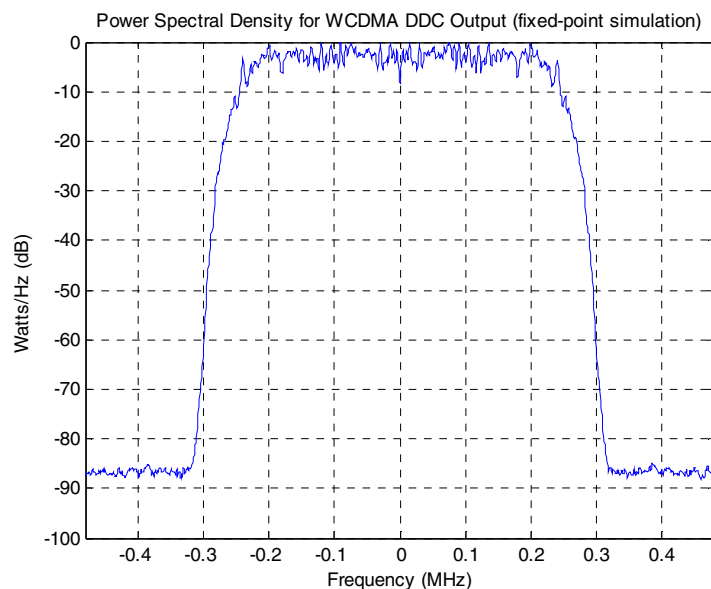


Figure 37: DDC Output from Wideband Data Test

Adjacent Channel Selectivity and Blocking

In the 3GPP standard, the Adjacent Channel Selectivity (ACS) and blocking tests are defined in separate sections. The ACS test measures the ability to receive a wanted signal at its assigned channel frequency in the presence of an adjacent channel, that is, two WCDMA interference signals placed at ± 5 MHz offset from the center frequency of the desired signal. Moreover, the power of the desired signal is 63 dB below the interference (for the wide area base station). For the blocking tests, the ability to receive a desired signal at its assigned channel with the presence of unwanted interference located on frequencies other than the adjacent signals is measured. The minimum offset of the interference blocker is located at ± 10 MHz, 75 dB above the wanted signal.

A test was created which combines both interference criteria from ACS and Blocking to simulate a more severe environment. The DDC input is presented in Figure 38 with the band of

interest centered at 15 MHz. The effective SNR is 29.45 dB, which is equivalent to 3.36% of EVM. This is significantly higher than the SNR requirements for a QPSK input format.

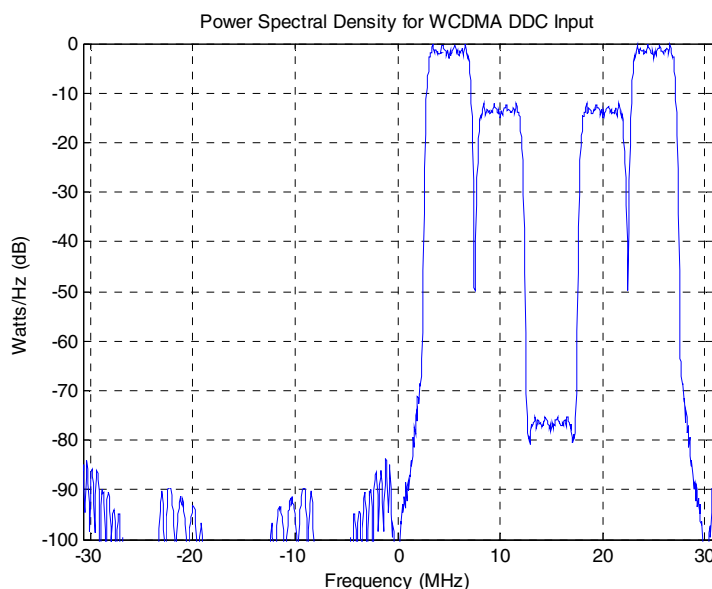


Figure 38: WCDMA DDC Input for ACS and Blocking Test (Centered at 15 MHz)

Intermodulation Response Rejection

Finally, the intermodulation test is performed. This is to test the effects that the third and higher order mixing of the two interfering RF signals introduce to the band of interest. In this test, four interferers 67 dB higher than the desired signal are generated. The two closer ones are single tone blockers, offset at ± 10 MHz. The other two are WCDMA signals at ± 20 MHz offset and uncorrelated to the desired signal. Figure 39 shows the DDC input with desired signal centered at 0 Hz.

The EVM for the down-converted output is 18.59%, which translates to 14.62 dB of SNR before de-spreading. After accounting for the coding gain, the SNR is around 40 dB.

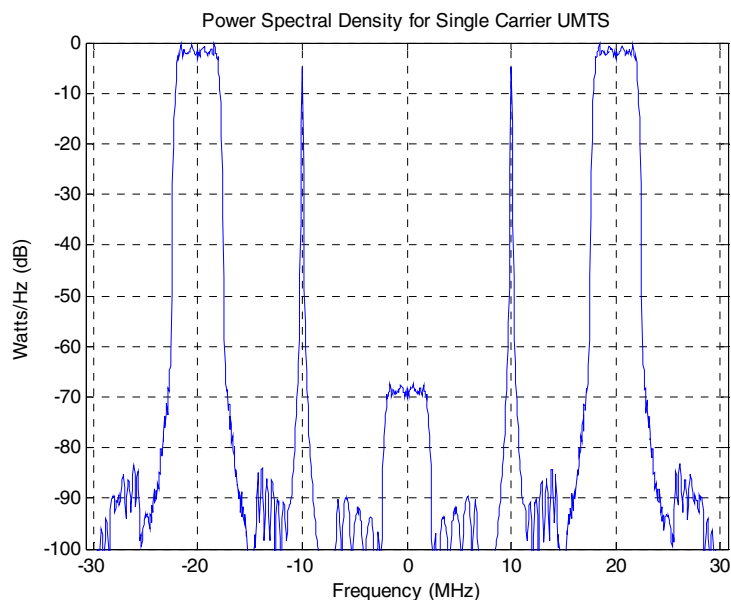


Figure 39: WCDMA DDC Input for Intermodulation Test

Performance Summary

The results from all the tests are summarized in [Table 17](#). In all the cases, the effective SNR is more than what is required for a communication system to receive the QPSK signal.

Table 17: Summary of WCDMA DDC Conformance Test Performance Metrics

Test	EVM (before de-spreading)	SNR (before de-spreading) (dB)	EVM (after de-spreading)	Effective SNR (after de-spreading) (dB)
Wideband Data with AWGN	1.22%	38.27	0.07%	63.27
ACS + Blocking	59.79%	4.47	3.36%	29.45
Intermodulation	18.59%	14.62	1.04%	39.59

DDC Implementation

The block diagram for the WCDMA DDC is presented in [Figure 40](#). The concept of the WCDMA DDC implementation is similar to the one used in the DUC implementation, except it is in reverse.

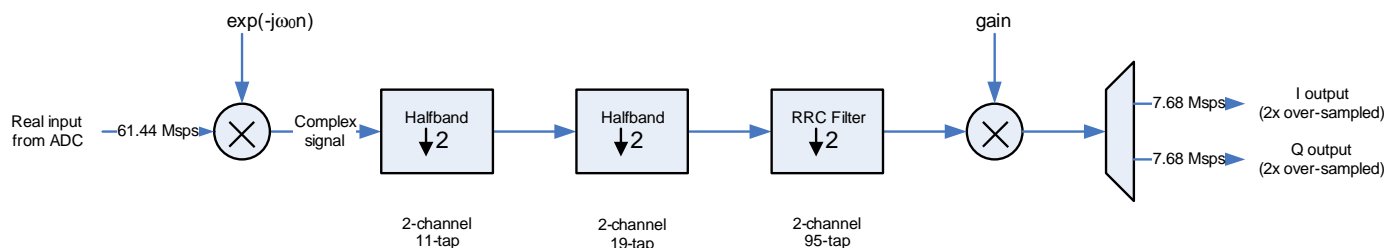


Figure 40: Top-Level Block Diagram for WCDMA DDC Implementation

Using System Generator Design Flow

The top-level screen shot for the System Generator Implementation of the WCDMA DDC is shown in [Figure 41](#).

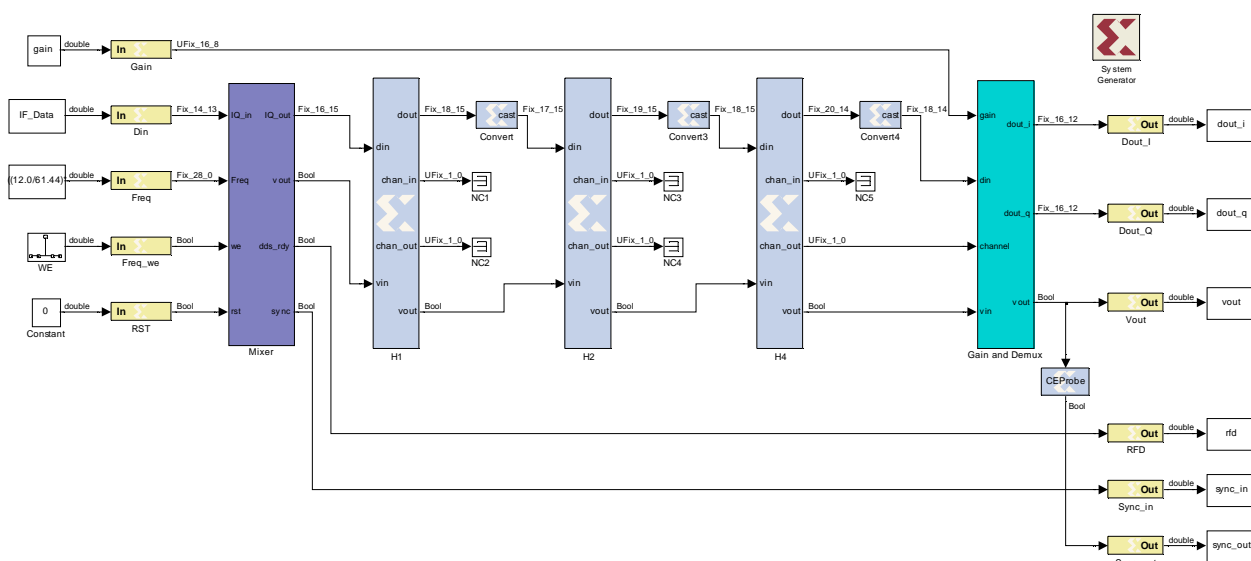


Figure 41: WCDMA DDC System Generator Implementation

Generate Sinusoids Using the DDS Compiler Block

The DDS Compiler block is used to generate sinusoidal waveform needed by the frequency translate function. The overview and block interface of the DDS Compiler are found in the [“Generate Sinusoids Using the DDS Compiler Block”](#) section.

In the DDC implementation, the sinusoidal pair $[\cos(\omega_0 n), -\sin(\omega_0 n)]$ is used. Therefore, the negative sine option needs to be selected in the GUI. The rest of the parameters are identical to that of the DUC implementation and are listed in [Table 18](#).

Table 18: Parameters used in System Generator DDS Block for WCDMA DDC

DDS Clock Rate	61.44 MHz
Output Function	Both Sine (negative) and Cosine
SFDR	102 dB (can be specified up to 115 dB)
Frequency Resolution	0.25 Hz
Number of Channels	1
Output Frequency	Programmable
Noise Shaping	Taylor Series Corrected

Implement Mixer Using XtremeDSP DSP48 Slices

The mixer block for design targeting Virtex-5 FPGA is shown in [Figure 42](#).

Here assume the DDS block output is $A(n) = A_r(n) + j \cdot A_i(n)$, where $A_r(n) = \cos(\omega_0 n)$ and $A_i(n) = -\sin(\omega_0 n)$, and the input signal to the DDC is a real signal $B(n)$. The mixer output is calculated as:

$$P = A \times B = (A_r + j \cdot A_i) \times B = (A_r B) + j \cdot (A_i B) \equiv P_r + j \cdot P_i \quad \text{Equation 3}$$

where P_r and P_i are defined as the real and imaginary part of the result from the complex multiplication. Each P_r or P_i term only requires one multiplication.

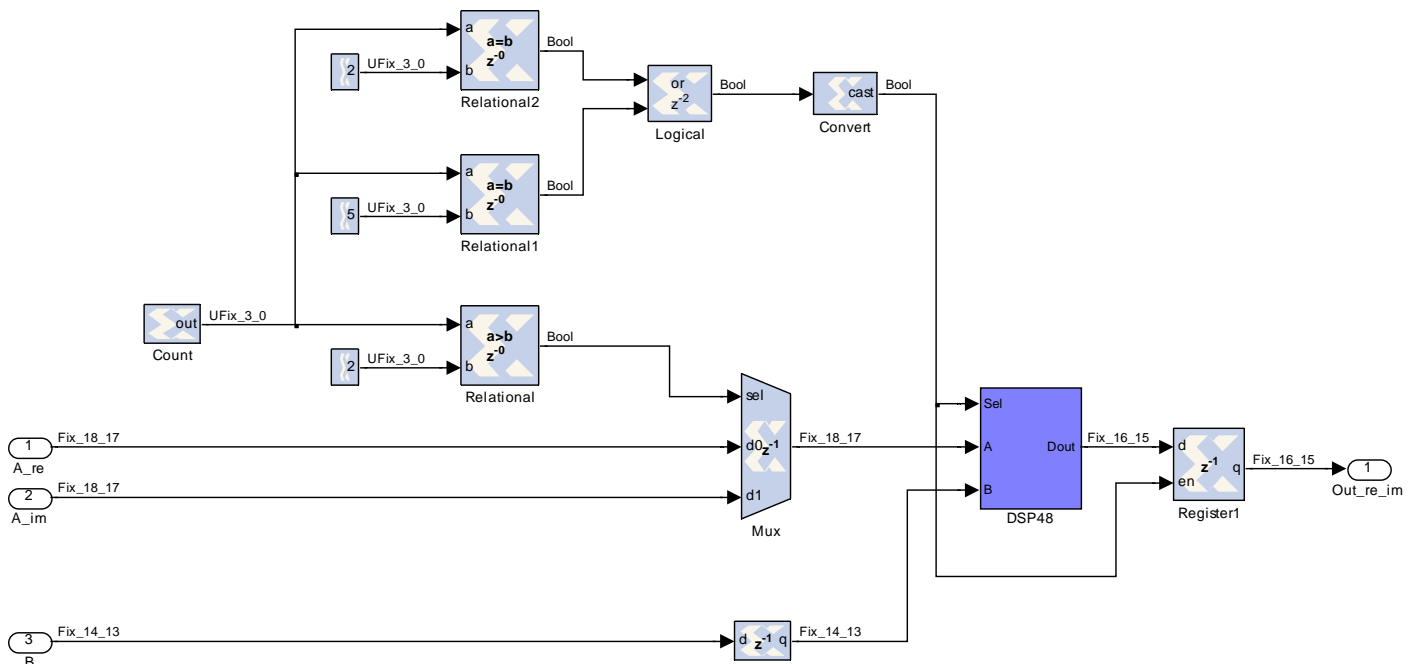


Figure 42: Virtex-5 FPGA Mixer Implementation for WCDMA DDC

The operations can be folded onto a single DSP48E meeting the throughput requirements for both the Virtex-5 and Spartan-DSP device families. The implementation details are omitted because the design concept is very similar (but simpler) to what has been described in the DUC section (see “[Implementing Complex Multiplier Using DSP48 Slices](#)”). [Figure 43](#) and [Figure 44](#) show the mixer implementation in the System Generator for Virtex-5 and Spartan-DSP devices, respectively.

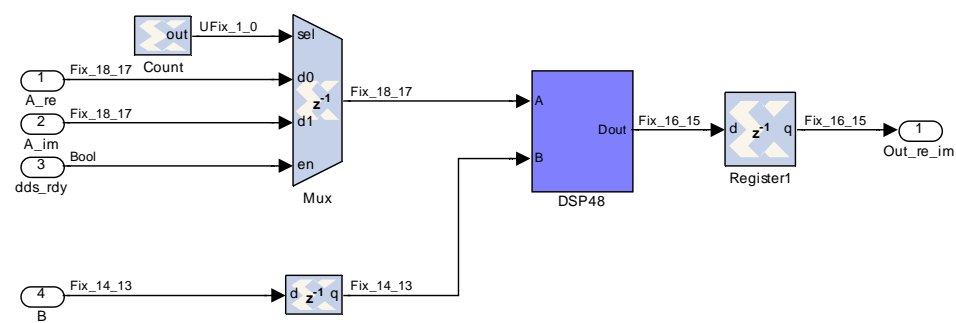


Figure 43: Spartan-DSP FPGA Family Mixer Implementation for WCDMA DDC

Construct Decimation Filters Using the FIR Compiler Block

The FIR Compiler block is used to implement the various decimation filters in the DDC design. The overview and interface of the FIR Compiler are introduced in detail in the “[Construction of Interpolation Filters Using the FIR Compiler Block](#)” section. [Table 19](#) lists a summary of important parameters for the WCDMA DDC design example. All but the RRC filter are using the efficient halfband structure.

Table 19: Parameters used in System Generator FIR Block for WCDMA DDC

Parameter		1 st Halfband	2 nd Halfband	RRC Filter
Filter Type		Decimation	Decimation	Decimation
Sample Rate Change		2	2	2
Number of Channels		2	2	2
Hardware Over-Sampling Rate	Virtex-5 FPGA	3	6	12
	Spartan-DSP FPGA	1	2	4
Structure		Symmetric	Halfband	Halfband

The hardware over-sampling rate parameter is defined as the ratio between the FPGA clock rate and the faster data rate of the input and output when accounting for the number of TDM channels. Use the first decimator (1st Halfband filter) in the Virtex-5 device design as an example; since it is a decimator, the faster data rate is the input data rate. The effective input data rate when accounting for the number of TDM channels is 122.88 MSPS. This is calculated by multiplying the input sample rate of 61.44 MSPS by the number of channels (that is, 2). Therefore, the HOR is the FPGA clock rate of 368.64 MHz divided by 122.88 MSPS and is equal to 3. The HOR for rest of the filters in Virtex-5 and Spartan-DSP device designs are listed in [Table 6](#).

Implement Gain Control and Time Division De-Multiplexing

The signal after the decimation process might be weak due to the removal of the adjacent channel interference. To boost the signal power, a gain control module using the Xilinx Multiplier core is implemented. This core computes the product of the data on its two input

ports, with appropriate internal pipelining inserted. This block is generic and can be used independently of the device families.

After the decimation filtering and gain stage, the TDM data stream is divided into the I and Q channels by the time division de-multiplexing (TDD) process which involves some capture registers and down samplers. The implementation of the Virtex-5 FPGA WCDMA DDC design is shown in [Figure 44](#). This gain or scaler stage may also be incorporated into an AGC loop by the user.

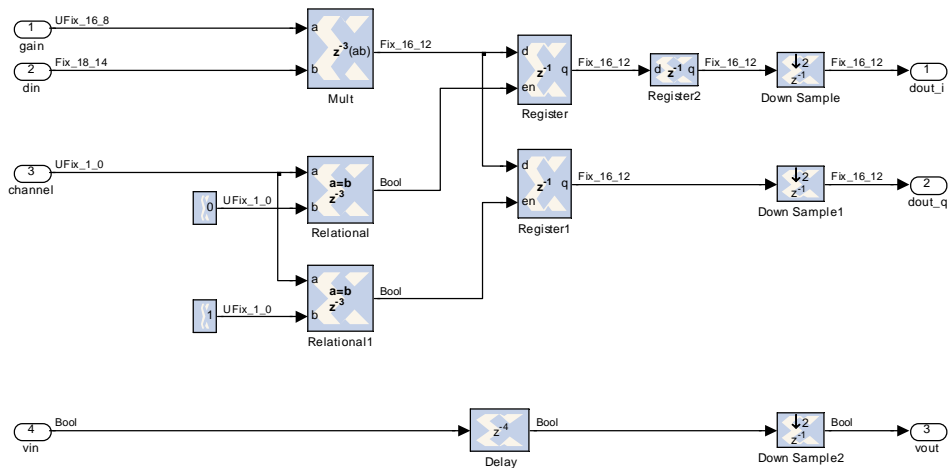


Figure 44: Gain Control and TDD Block for WCDMA DDC Implementation

Using HDL Design Flow

The design concept is identical to what was described in the [“Using System Generator Design Flow”](#) Section. [Table 20](#) and [Table 21](#) list the parameters used to specify the DDS and FIR compiler for the WCDMA DDC example targeting both the Virtex-5 and Spartan-DSP device families.

Table 20: Parameters Used in CORE Generator DDS Compiler for WCDMA DDC

Output Selection	Both Sine (negative) and Cosine
Channels	1
DDS Clock Rate	61.44 MHz
SFDR	105 dB (can be specified up to 115 dB)
Frequency Resolution	0.25 Hz
Phase Increment	Programmable
Phase Offset	None
Noise Shaping	Taylor Series Corrected

Table 21: Parameters used in CORE Generator FIR Compiler for WCDMA DDC

Parameter	1 st Halfband	2 nd Halfband	RRC Filter
Filter Type	Decimation	Decimation	Decimation
Rate Change Type	Integer	Integer	Integer
Sample Rate Change	2	2	2
Number of Channels	2	2	2
Input Sampling Frequency	61.44	30.72	15.36

Table 21: Parameters used in CORE Generator FIR Compiler for WCDMA DDC

Parameter		1 st Halfband	2 nd Halfband	RRC Filter
Clock Frequency	Virtex-5 FPGA	368.64 MHz		
	Spartan-DSP FPGA	122.88 MHz		
Output Rounding Mode		Symmetric Rounding to Infinity		

Three-Carrier
CDMA2000
Design Example

Digital Up-Converter

The DUC translates the three-carrier CDMA2000 complex baseband data samples at the chip rate to digital IF, between $-F_s/2$ to $F_s/2$, where F_s is the sample rate at the input of the DAC and is 61.44 MHz in this example. The DUC also performs necessary pulse shaping to the transmit waveform. Moreover, in a multi-carrier system, the waveforms in different bands are combined together. This example is designed to meet the 3rd Generation Partnership Project 2 (3GPP2) C.S0010-C specification [Ref 7], which defines the minimum performance standards for the CDMA2000 base station. The high-level block diagram for the CDMA2000 DUC is illustrated in Figure 45.

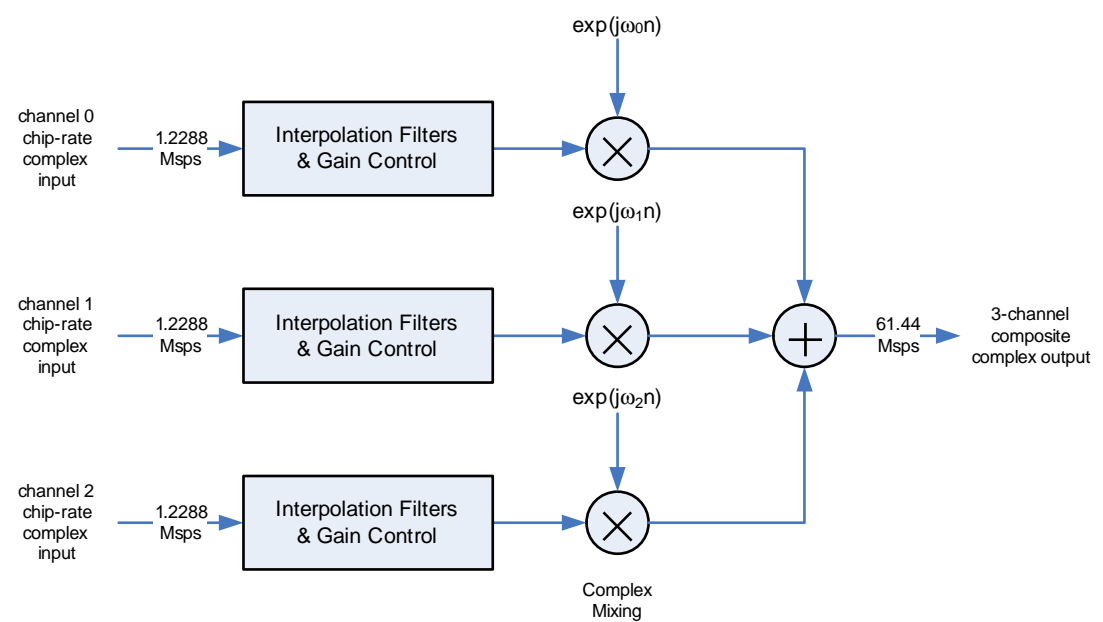


Figure 45: High-Level Block Diagram for CDMA2000 DUC

Performance Requirements

A summary of requirements for the DUC design is listed in Table 22.

Table 22: Target Specification for CDMA2000 Downlink Transmit Path

Parameter	Value	Notes
Carrier Bandwidth	1.25 MHz	
Number of Carriers	3 carriers	
Total Bandwidth	5.0 MHz	3 carriers positioned in a 5 MHz spectrum
Carrier Spacing	1.25 MHz	
Baseband Chip Rate	1.2288 MSPS	

Table 22: Target Specification for CDMA2000 Downlink Transmit Path (Continued)

Parameter	Value	Notes
IF Sample Rate	61.44 MSPS	50×1.2288 MSPS
Transmit Spectral Mask	Up to 87 dB for frequency offset over 6.0 MHz	Section 4.4 of [Ref 7] plus 20 dB margin
Mean Squared Error (MSE)	0.03	Compared to IS-95 waveform
Rho (ρ)	0.912	Measure modulation accuracy
Input Signal Quantization	16-bit I and Q	Complex
Output Signal Quantization	18-bit I and Q	Complex, maximize DSP48 capability
Mixer Properties	Tunability: Variable Resolution: ~0.25 Hz SFDR: up to 115 dB	Various SFDR are supported by the DDS Compiler core (28 bits of frequency accumulator width)

The base CDMA2000 chip rate is 1.2288 MCPS. Each carrier occupies a 1.25 MHz bandwidth and the composite waveform of three carriers is positioned in a 5 MHz bandwidth. Figure 46 illustrates how three adjacent CDMA2000 waveforms occupy a 5 MHz bandwidth with carrier spacing of 1.25 MHz and two 625 kHz guard bands.

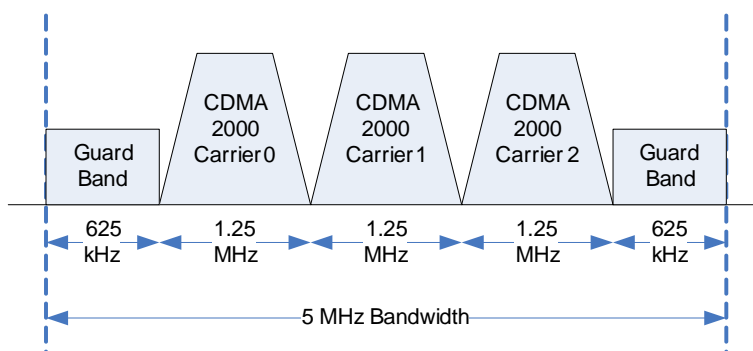


Figure 46: Three CDMA2000 Carriers Positioned in a 5 MHz Bandwidth

Spectral Mask Requirements

Section 4.4 of the 3GPP2 C.S0010-C [Ref 7] defines the transmitter emission limits for different band classes and ITU categories for CDMA2000 Base Stations. In addition, different emission limits are specified for single-carrier and multi-carrier cases.

The spectral requirements among all defined categories are summarized and only the most stringent minimum attenuation requirements are listed in Table 23. The most restrictive requirements happen in the case where $P = 33$ dBm, where P is the total inband power. For CDMA2000, the total inband power of the signal is measured in a 1.23 MHz bandwidth around the signal center frequency. Note that the emission limits listed in the standard have different measurement bandwidths, varying from 30 kHz to 1 MHz, and the table contains the normalized dB scale.

The spectral mask requirements are generated by adding a 20 dB to the minimum requirements. This ensures that the spectral mask can be met were there any distortion in the analog components or in any post processing modules, such as CFR.

Table 23: Single-Carrier Filter Attenuation Requirements for P = 33 dBm

For Δf Within the Range	Minimum Attenuation (dB)	Attenuation with 20 dB Margin (dB)
750 kHz to 885 kHz	29→44	49→64
885 kHz to 1.125 MHz	44→49	64→69
1.125 MHz to 1.98 MHz	49	69
1.98 MHz to 4.0 MHz	59	79
4.00 MHz to 6.0 MHz	62	82
> 6.0 MHz	67	87

Frequency Response Limits

Section 11.3.1.3.7.3 of the 3GPP2 C.S0024-B specification [Ref 8] lists the frequency response limits for the baseband filter. The passband edge frequency (F_{pass}) and stopband edge frequency (F_{stop}) are defined as $F_{pass} = 590$ kHz and $F_{stop} = 740$ kHz. The peak-to-peak passband ripple is constrained to 3 dB, and the minimum stopband attenuation should be greater than 40 dB.

In this case, the composite filter response generated by the DUC needs to meet the above frequency response limits.

Mean Squared Error Limits

Also defined in the section 11.3.1.3.7.3 of the 3GPP2 C.S.0024-B specification [Ref 8] is the Mean Squared Error (MSE) limits for the baseband filter. The impulse response of the baseband filter, $s(t)$, should satisfy the following equation:

$$MSE = \sum_{k=0}^{\infty} [\alpha s(kT_s - \tau) - h(k)]^2 \leq 0.03$$

Equation 4

where the constants α and τ are used to minimize the MSE, $T_s = 203.451$ ns, and $h(k)$ is a 48-tap symmetric filter that holds the characteristics of the IS-95 waveform.

Note: In this document, the term “IS-95 waveform” is used to describe the CDMA2000 waveform, which must match some spectral characteristics with the older IS-95 specification for legacy reasons.

Applied to the DFE design, the time domain composite filter response generated by the DUC should be within 0.03 compared to the IS-95 waveform $h(k)$, where the phase and scale can be selected to minimize the MSE. Note that in the equation, the composite response used for comparison is 4x oversampled to the chip rate of 1.2288 MSPS. To interpret the equation correctly, it is important to resample the overall filter response to the corresponding rate before performing the MSE calculation.

Phase Equalization

The 3GPP2 standard [Ref 8] lists a phase equalization requirement for the transmit signal path for a CDMA2000 system. The equalizing filter is essentially an all-pass IIR filter with a baseband transfer function:

$$H(\omega) = K \frac{\omega^2 + j\alpha\omega\omega_0 - \omega_0^2}{\omega^2 - j\alpha\omega\omega_0 - \omega_0^2}$$

Equation 5

where K is an arbitrary gain, $j = \sqrt{-1}$, $\alpha = 1.36$, ω_0 equals $2\pi \cdot 3.15 \cdot 10^5$, and ω is the radian frequency.

In the forward link, this all-pass IIR filter (virtually unity gain across its Nyquist bandwidth) provides the required phase characteristics and generally follows the pulse-shaping filter. In hardware implementation, one way to realize it is to approximate this IIR filter by a FIR filter. The phase equalization FIR filter is then convolved with the pulse shaping filter that meets the spectral mask requirements, frequency response limits, and the MSE limits to generate the channel filter. It has also been seen that this type of phase equalizer has been implemented in the analog domain rather than the digital domain.

In this reference design, this phase equalization is not included as part of the DUC. The user can design a filter with the right phase characteristics and reprogram the coefficients of the channel filter to obtain the desired response were it to be of interest.

Adjacent Channel Power Ratio

Adjacent Channel Power Ratio (ACPR) is another figure of merit for component testing in CDMA2000. It is defined as the ratio of the average power in a bandwidth away from the main signal to the average power within the main transmitted frequency channel. As can be implied, this definition is heavily dependent upon the locations and the bandwidth of the distortion signal.

In the CDMA2000 standards, two types of ACPR are generally measured. The ACPR1 is the ratio of the distortion product power (measured in a 30 kHz bandwidth) that is 885 kHz away from the main channel to the main channel signal power. More specifically, this is to measure the distortion at $0.885 + 1.25 = 2.135$ MHz in a multi-channel configuration with carrier spacing of 1.25 MHz. The ACPR2, also known as alternate channel power ratio, measure the ratios of power in a bandwidth two channels away, for example, $1.98 + 1.25 = 3.23$ MHz from the main signal in a 30 KHz bandwidth to the main channel signal power.

Rho (ρ) and Error Vector Magnitude

Rho (ρ) is the ratio of the correlated power to the total power and is a special quality metric for CDMA systems. The correlated power is calculated by a cross correlation between the measured signal and an ideal reference signal. EVM is a more familiar figure of merit that identifies the difference between the measured to reference signals.

Rho and EVM work in reverse. For example, suppose the measured signal is identical to the reference signal, no error exists so the EVM = 0%. In this case $\rho = 1.00$ is achieved. Rho is always within the range of 0 and 1. The higher the Rho, the better the signal accuracy. The relationship between Rho and EVM can be calculated as [Equation 1](#).

$$\rho = \frac{1}{1 + (\text{EVM}/100)^2}$$

Equation 6

The 3GPP2 specification [\[Ref 7\]](#) requires ρ to be greater than 0.912.

DUC Filter Design

Filter Design Strategy

In the DUC, the input sample is up-sampled by 50 to an output sampling rate of 61.44 MSPS by a cascade of interpolation filters. This can be broken into a cascade of three filters as shown in [Figure 47](#) – an interpolation-by-2 channel filter to provide pulse shaping, and two interpolation-by-5 filters to remove the aliasing images introduced by up sampling. All three filters have a reasonable length.



Figure 47: CDMA2000 DUC Interpolation Filters Chain

It is worth noting that when designing filters, paying a little more attention to the filter length can go a long way, especially in terms of hardware resource consumption. For example, the first up-sample-by-5 interpolator in the following section can be designed with the passband ripple of 0.01 dB and the stopband attenuation of 85 dB in 47 taps. Instead, the passband ripple requirements is loosened to 0.025 dB with the same stopband parameter, which results in a filter length of 43. It can be observed that all the system criteria are still being met. However, in this process, one DSP48 was saved simply by reducing the length by 4 taps. The 47 tap filter requires three DSP48s; the 43 tap filter with competitive performance requires only two DSP48s. The resource utilization estimate page in the FIR compiler GUI is introduced in the previous section (see “[Construction of Interpolation Filters Using the FIR Compiler Block](#)”), which is a very powerful tool in combination with the MATLAB Fdatool when performing filter design.

Channel Filter

Some baseband chipset manufacturers provide selectable options between 1x data and 2x shaped data; therefore, some OEM applications require that the DFE do pulse shaping, others do not. In this application note, it is assumed that the CDMA2000 radio input is 1x raw data (pre-processed data before the baseband filter).

As defined in the previous section, the overall filter response needs to fulfill the spectral masks requirements and be within the frequency response and MSE limits. Since the channel filter has the most significant impact on the final spectrum, it has to be carefully designed based on those criteria.

A way to design the channel filter is to convolve the IS-95 response $h(k)$ with a low pass filter (LPF). This ensures that the time domain response of the channel filter is closely matched to $h(k)$ to meet MSE limits. It is observed that the passband peak-to-peak ripple of $h(k)$ is around 2 dB, 1 dB away from the requirements, and its stopband attenuation is only fair (less than 40 dB). Therefore, the LPF required here not only needs to have enough attenuation to suppress the stopband even further to meet the 40 dB frequency response limits (especially at 740 kHz stopband edge) and the very stringent spectral masks up to 87 dB (up to 64 dB below 885 kHz), but also needs to have a relatively small passband ripple.

A 67-tap filter that is generated from convolving a 47-tap LPF with the *resampled* version of $h(k)$ meets the above requirements. The LPF is designed using the Remez filter design methodology, and its passband and stopband edge are 590 kHz and 740 kHz, respectively. The sampling frequency is 2.4576 MHz, and the filter has passband peak-to-peak ripple of 0.01 dB and stopband attenuation of 42 dB. As implied from [Equation 4](#), $h(k)$ is 4x over-sampled but the channel filter is an interpolation-by-2 filter. Therefore, $h(k)$ has to be resampled before convolving the LPF to obtain a correct response. [Figure 48](#) shows the magnitude response of the up-sampled-by-2 channel filter.

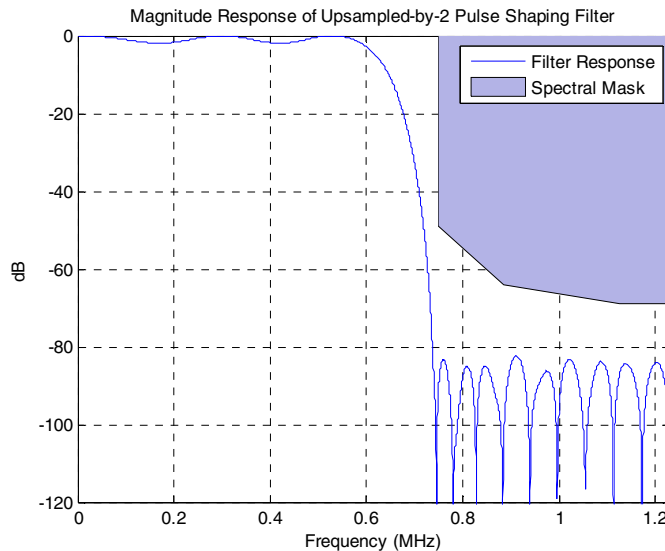


Figure 48: Magnitude Response of CDMA2000 DUC Channel Filter

First Interpolation-by-5 Filter

Following the pulse shaping filter is an interpolation filter with an up sampling rate of 5. The purpose of this filter is to remove the aliasing signal image. It does not influence the shape of the transmitted signal.

The output sample rate of the filter is 12.288 MSPS, ten times the chip rate. The filter was designed using the lowpass equiripple (Remez) method with the passband set to $F_{\text{pass}} = 0.59$ MHz as defined in the standard, and the stopband set to $F_{\text{stop}} = 2.4576 - 0.74 = 1.7176$ MHz. This keeps the desired passband signal while suppressing the image produced by the interpolation effect.

With the passband ripple chosen to be 0.025 dB and the stop attenuation selected to be 85 dB, a 43-tap equal ripple filter was designed to the specification. Although the stopband attenuation is set to 85 dB, the filter obtained has an F_{stop} just less than 80 dB. This is normal when the transition band is much narrower than the sampling rate. When designing a filter, consider that extra margin might need to be added and always check the magnitude response to ensure the performance is met after the filter is designed. Figure 49 shows the magnitude response of the filter after the quantization effects.

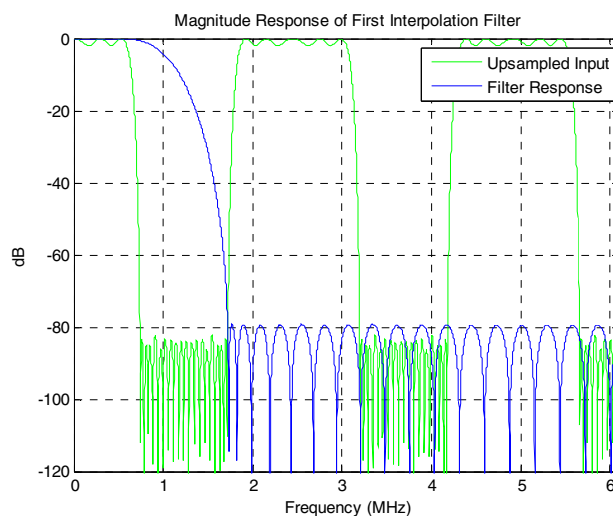


Figure 49: Magnitude Response of First Interpolation-by-5 Filter

Second Interpolation-by-5 Filter

The second interpolation-by-5 filter further up samples the signal and suppresses the aliasing image generated by the interpolation process. The output sample rate of the filter is 61.44 MSPS. Using the MATLAB FDATool, a 21-tap equiripple filter is designed when setting the passband frequency $F_{pass} = 0.59$ MHz and the stopband frequency $F_{stop} = 12.288 - 0.59 = 11.698$ MHz. The passband peak-to-peak ripple is chosen to be 0.05 dB and the stopband attenuation is selected to be 95 dB. The stringent stopband attenuation requirement is to ensure that the spectral masks of up to 87 dB can be achieved. The filter response of the filter after the quantization effects is presented in Figure 50.

Composite Filter Response

The specifications of all the interpolators are summarized in Table 24. Figure 51 shows that the overall response meets the frequency response limits. The worst case passband ripple is $2 + 0.01 + 0.025 + 0.05 = 2.085$ dB, well constrained within the 3 dB limit. Also, the stopband is below the 40 dB limit starting at 740 MHz.

Table 24: Summary of CDMA2000 Interpolation Filters

Filter Stage		Pass Band F_{pass} (MHz)	Stop Band F_{stop} (MHz)	Output Sample Rate F_s (MSPS)	Peak-to-Peak Ripple (dB)	Stop Band Attenuation (dB)	Filter Order
Channel Filter	$h(k)$	48-tap symmetric per spec and resampled to 2x					
	LPF	0.59	0.76	2.4576	0.01	42	46
First Interpolator- by-5 Filter		0.59	1.7176	12.288	0.025	85	42
Second Interpolator-by-5 Filter		0.59	11.698	61.44	0.05	95	21

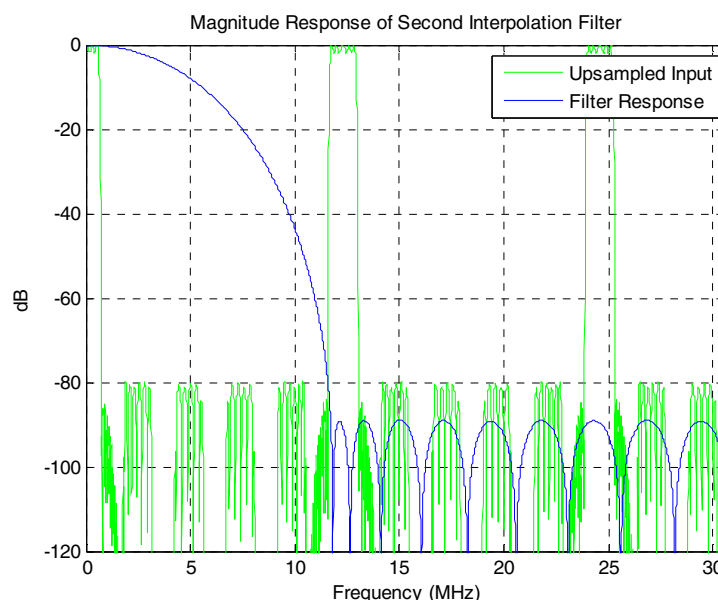


Figure 50: Magnitude Response of Second Interpolation-by-5 Filter

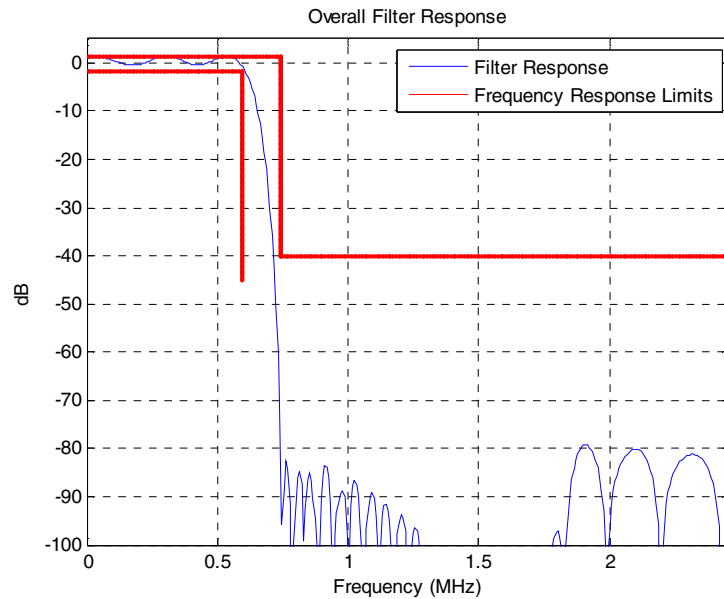


Figure 51: Overall Filter Response vs. Frequency Response Limits

Figure 52 illustrates that after accounting for quantization effects, the overall response meets the stringent spectral masks up to 87 dB.

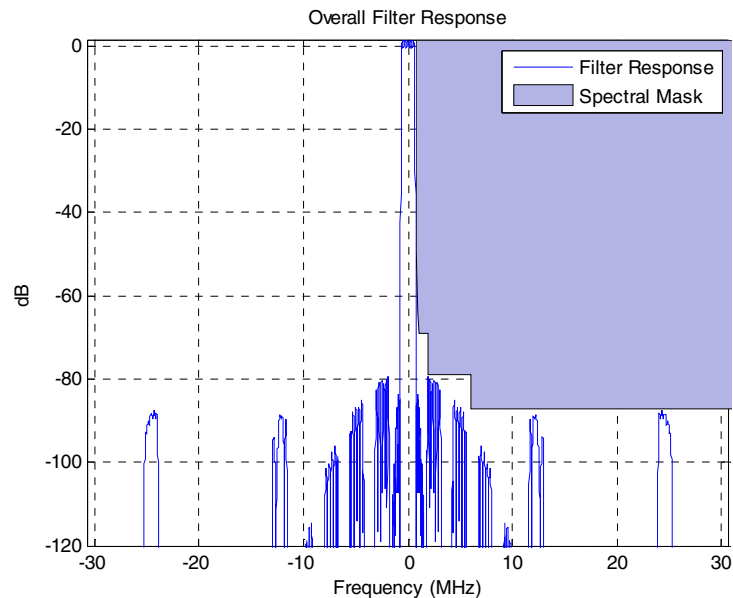


Figure 52: Overall Filter Response vs. Spectral Mask Requirements

The MSE of the composite filter response is 0.0047 compared to the IS-95 waveform. This is well below the limits of 0.03. The impulse response (adjusted to the phase and scale that

minimized the MSE) is overlaid on the top of the IS-95 time domain response $h(k)$, shown in Figure 53. It can be seen that two responses are very close to each other.

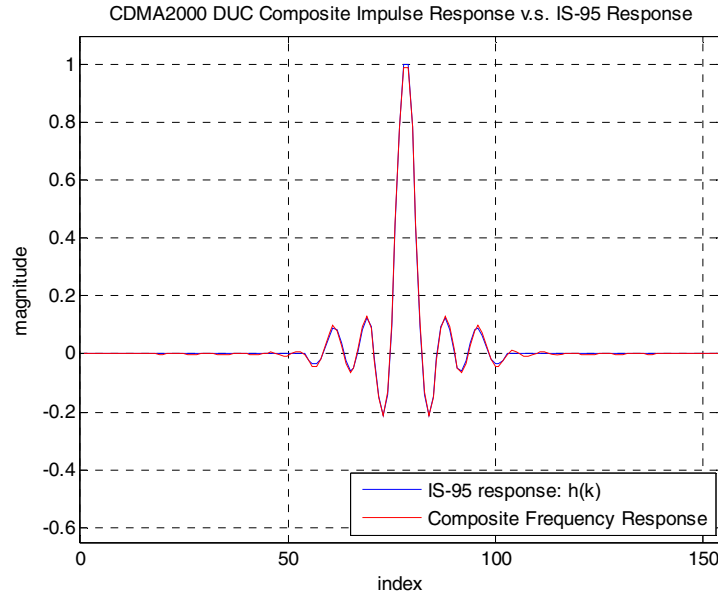


Figure 53: **Composite Impulse Response v.s. IS-95 Response**

Frequency Translation and Combining

The DDS and mixer function for the CDMA2000 example is identical to what is described in the WCDMA example. It is basically to shift the waveform to a specific intermediate frequency using the DDS. The detail has been included in the earlier section and is not repeated here. The only difference is that in a multi-carrier scenario, individual outputs from all three carriers need to be combined to form one complex output stream after frequency translation.

Mathematically, the overall mixing and combining function can be expressed as

$$y(n) = \sum_{k=1}^3 y_k(n) = \sum_{k=1}^3 x_k(n) * e^{jw_k n} \quad \text{Equation 7}$$

where $y(n)$ is the final complex output, $x_k(n) = x_{I_k}(n) + jx_{Q_k}(n)$ is the up-sampled input signal, and $e^{jw_k n}$ is the complex exponential which serves as the orthonormal basis.

After a few more steps of derivation, it can be found that the real and imaginary parts of the output are

$$y_I(n) = \sum_{k=1}^3 [x_{I_k}(n) * \cos(w_k n) - x_{Q_k}(n) * \sin(w_k n)] \quad \text{Equation 8}$$

$$y_Q(n) = \sum_{k=1}^3 [x_{I_k}(n) * \sin(w_k n) + x_{Q_k}(n) * \cos(w_k n)] \quad \text{Equation 9}$$

DUC Performance

The power spectral density for 3-Carrier CDMA2000 (middle carrier centered at 0 Hz is shown in Figure 54. The emission requirements are satisfied so that the interferers to adjacent bands are minimal.

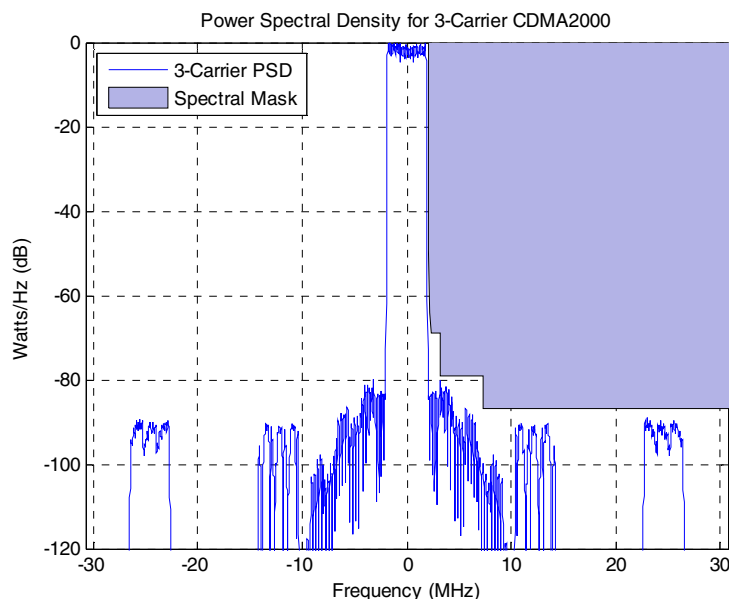


Figure 54: PSD for Three-Carrier CDMA2000 waveform

Table 25 summarizes the performance metrics for the DUC generated from the simulation results. The ACPR1 and ACPR2 are measured when all three carriers are enabled. Therefore, the integral BW of the transmit signal is 3.75 MHz. The ACPR1 and ACPR2 are measured in a 30 kHz BW at 885 kHz and 1.98 MHz offset from the center frequency of the highest carrier (located at 1.25 MHz), and both are nicely sitting below 80 dB.

Table 25: Summary of CDMA2000 Performance Metrics

Performance Metrics	Value
ACPR1 (offset by 0.885 MHz)	81.92 dB
ACPR2 (offset by 1.98 MHz)	86.78 dB
Rho	0.9989
EVM	3.3%

The measured cross correlation coefficients Rho is 0.9989 and met the 3GPP2 requirement of 0.912 by a significant margin. This provides head room for distortion introduced by other analog components or digital front end modules, such as the CFR and DPD. The 3GPP2 specification does not define an EVM requirement, but it is listed here as it is a common metric to use in evaluating radio system performance. The DUC has an EVM of 3.3%.

DUC Implementation

In the “IF Sampling Rate and System Clock” section, the relationship between the IF data rate and FPGA operation rate is discussed. In the Virtex-5 FPGA example, since there are a total of six available clocks per output sample ($F_{clk} = 368.64$ MHz), the input data from six independent channels (three complex channels) can be TDM'd into a single data stream to be processed by the interpolation filter chain. For the design targeting the Spartan-DSP FPGA, 184.32 MHz was the clock rate and there are only 3 cycles available for every output sample. A convenient choice is to construct two TDM data streams, one for three I channels and the other

for three Q channels, which feed into two filter paths. At the end of the final interpolator, the samples then go through the complex mixing and combining stage. The block diagram for DUC architecture for the Virtex-5 FPGA is shown in Figure 55, and that for Spartan-DSP FPGA is presented in Figure 56.

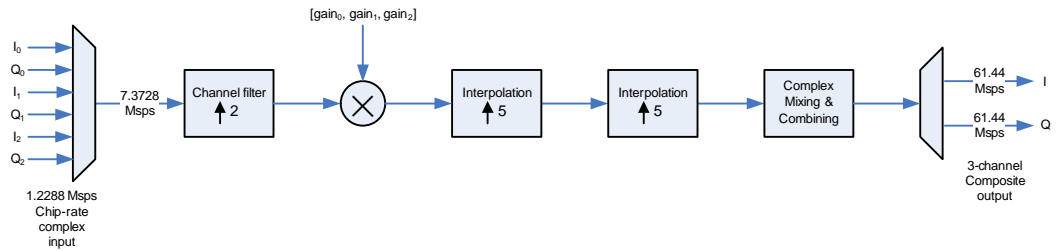


Figure 55: Block Diagram for CDMA2000 DUC Implementation Targeting Virtex-5 FPGA

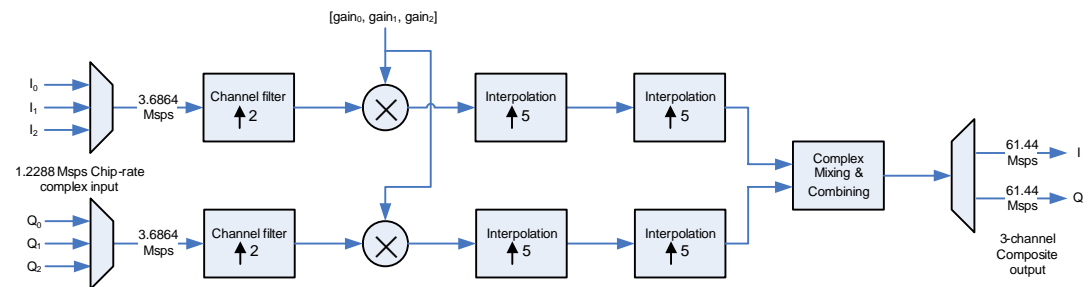


Figure 56: Block Diagram for CDMA2000 DUC Implementation Targeting Spartan-DSP FPGA

The top-level screen shots for the System Generator Implementation of the CDMA2000 DUC targeting Virtex-5 and Spartan-DSP devices are shown in Figure 57 and Figure 58, respectively.

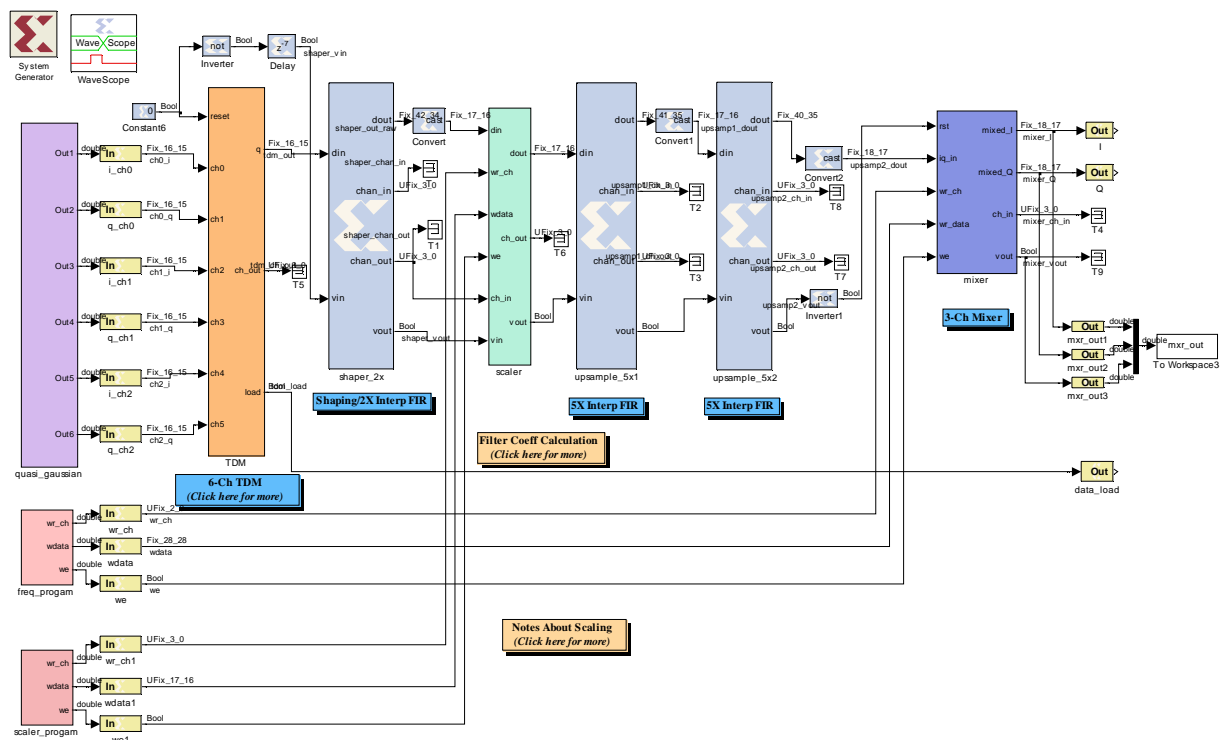


Figure 57: CDMA2000 DUC Implementation (Virtex-5 FPGA)

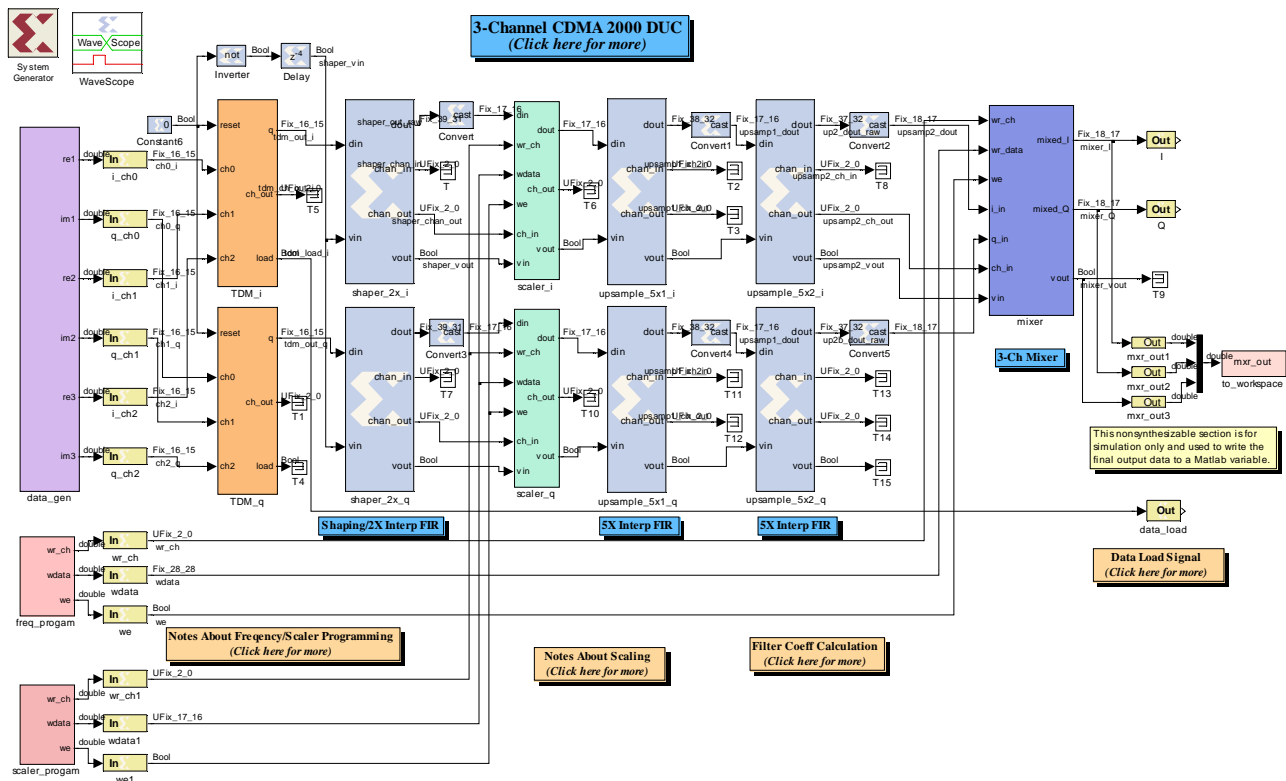


Figure 58: CDMA2000 DUC Implementation (Spartan-DSP FPGA)

Construction of Interpolation Filters Using the FIR Compiler Block

The interpolation filters in the DUC chain shown in Figure 55 and Figure 56 are implemented using Xilinx FIR Compiler block. The overview and block interface of the FIR Compiler has been discussed in detail in the WCDMA “Construction of Interpolation Filters Using the FIR Compiler Block” section and is not repeated here. Table 26 lists a summary of important parameters for the CDMA2000 DUC design example.

Table 26: Parameters Used in System Generator FIR Block for CDMA2000 DUC

Parameter		Channel Filter	Image Removing Filter 1	Image Removing Filter 2
Filter Type		Interpolation	Interpolation	Interpolation
Sample Rate Change		2	5	5
Number of Channels	Virtex-5 FPGA	6	6	6
	Spartan-DSP FPGA	3	3	3
Hardware Over-sampling Rate (HOR)		25	5	1
Structure		Symmetric	Symmetric	Symmetric

The HOR parameter specifies the ratio between the FPGA clock rate and the faster data rate, for example, the faster input data rate and output data rate when accounting for the number of TDM channels. This ratio determines how much hardware folding can be done; in other words, how many MAC engines are required.

The channel filter is used in the Virtex-5 FPGA design as an example to demonstrate how to set the HOR parameter correctly. Since it is an interpolator, the faster data rate comes from the output data rate. The effective output data rate when accounting for the number of TDM channels is 14.7456 MSPS. This is calculated by multiplying the output sample rate of 2.4576 MSPS by the number of channels (that is, 6). Therefore, the HOR is the FPGA clock rate of 368.64 MHz divided by 14.7456 MSPS and is equal to 25.

As the second example, the image removing filter 1 in the Spartan-DSP FPGA is an up-sampled-by-5 filter. The effective output data rate is calculated by multiplying the output sample rate of 12.288 MSPS by number of channels (that is, 3) and is equal to 36.864 MSPS. The FPGA clock rate of 184.32 MHz divided by the output data rate is 5, which is the HOR that is expected. The HOR for the rest of the filters in Virtex-5 and Spartan-DSP FPGA designs are listed in [Table 26](#).

Notes on Multi-Channel Implementations

All of the designs described in this application note have been implemented as multi-channel designs with the data multiplexed on TDM buses. Why did we not choose an architecture comprising multiple single-channel instantiations? A multi-channel architecture is much more modular. If extra channels need to be added, it is, while not trivial, easier to mux more channels onto the existing TDM buses. Such an implementation is also more likely to be resource-efficient. For example, there are filter stages in the Virtex-5 FPGA CDMA2000 design which use two DSP48s shared among six channels. Were the channels not sharing a single TDM stream, each filter stage would require its own DSP48, meaning there would be a total of six, and each would only be used every third clock cycle. There would also be no sharing of hardware, such as coefficient ROMs, even though these are common among the six channels.

The DDS provides another example of the advantage of multi-channel implementation. Because the DDS can operate at a faster rate than the sample rate of any single channel, it can be “folded” into all channels using a TDM scheme. The sine ROM is the same for all channels, so it makes no sense to replicate this if not necessary. There is a hardware cost for the multi-channel DDS. Most of the storage registers must be replaced with small RAMs. However, as long as the channel count is less than 16, this may not increase the slice count, since the RAMs can be implemented efficiently in distributed RAM. The additional hardware is much less than would be required if using a separate DDS for each channel.

Generate Sinusoids Using the DDS Compiler Block

If interested the user can refer back to the WCDMA “[Generate Sinusoids Using the DDS Compiler Block](#)” section for the DDS overview and its block interface.

[Table 27](#) lists a summary of parameters for the DDS block. The DDS clock rate is the frequency at which the DDS block is clocked. In multi-channel case, it is equivalent to the DUC output sample rate multiply with the number of channels. Thus, the DDS clock rate is $61.44 \times 3 = 184.32$ MHz for the 3-carrier CDMA2000. The SFDR defines the DDS output spectral purity in terms of the frequency domain requirements of the out-of-band noise level. It can be any number up to 115 dB, and is defaulted to 102 dB in this configuration. The frequency resolution determines the granularity of the tuning frequency and is selected to be 0.25 Hz.

Table 27: Parameters Used in System Generator DDS Block for CDMA2000 DUC

DDS Clock Rate	184.32 MHz
Output Function	Both Sine (positive) and Cosine
SFDR	102 dB (can be specified up to 115 dB)
Frequency Resolution	0.25 Hz
Number of Channels	3
Output Frequency	Programmable
Noise Shaping	Taylor Series Corrected

Both Sine and Cosine output are required in this example. The output frequency is set to be programmable and the phase offset is set to zero.

The output frequency field requires the normalized frequency. To generate the output frequency at F_c MHz from the DDS, the value of

$$\text{round}(F_c / F_s * 2^{\text{accumulator_width}})$$

needs to be fed to the data port. For example, to shift the waveform to 1.25 MHz output frequency with a DDS clock rate of 61.44 MHz and 30 bit accumulator width, the value of $\text{round}(1.25/61.44 * 2^{30}) = 21845333_{(10)} = 01\ 0100\ 1101\ 0101\ 0101\ 0101\ 0101_{(2)}$ has to be latched into the data port.

Implement Mixer and Combiner

The 3-channel mixer and combiner are implemented using the multiple DSP48 blocks and the opmode muxes. From Equation 8 and Equation 9, it can be found that six MAC operations are required to calculate each sum of the real and imaginary part.

For the Virtex-5 FPGA design, since there are six available cycles per output sample, a total of two DSP48s are needed to perform the operation, one for the real and one for the imaginary. Because the DDS outputs sine and cosine at the same time, both the real and imaginary parts are simultaneously available in two independent paths.

Use the real sum as an example, the algorithm follows six steps over the course of six clock cycles:

1. DSP48 calculates a portion of the real product for channel 1, that is, $x_{I_1}(n) * \cos(w_1 n)$, and stores it in its P register. Also, a rounding constant is added into the product from the C and Cin ports.
2. DSP48 calculates the remaining portion of the real product for channel 1, that is, $x_{Q_1}(n) * \sin(w_1 n)$, and subtracts it from P. This gives the partial results, $x_{I_1}(n) * \cos(w_1 n) - x_{Q_1}(n) * \sin(w_1 n)$, as the new P value.
3. DSP48 calculates a portion of the real product for channel 2, that is, $x_{I_2}(n) * \cos(w_2 n)$, and accumulates it to P.
4. DSP48 calculates the remaining portion of the product for channel 2, that is, $x_{Q_2}(n) * \sin(w_2 n)$, and subtracts it from P.
5. DSP48 calculates a portion of the real product for channel 3, that is, $x_{I_3}(n) * \cos(w_3 n)$, and accumulates it to P.
6. DSP48 calculates the remaining portion of the real product for channel 3, for example, $x_{Q_3}(n) * \sin(w_3 n)$, and subtracts it from P.

After step 6, the P register holds the composite value

$$\sum_{k=1}^3 [x_{I_k}(n) * \cos(w_k n) - x_{Q_k}(n) * \sin(w_k n)]$$

for the real part. This is captured, down sampled, and rounded (the constant is added in step 1) to the output bit width and then presents as the final output. The cycle begins again at step 1.

For the Spartan-DSP FPGA design, there are three available cycles per output sample. At least two DSP48As are required to calculate each of the real and imaginary composite value. The DDS also outputs sine and cosine at the same time, so both the real and imaginary parts are simultaneously available.

In reality, to map each of the real and imaginary composite value calculation in two DSP48As requires some data scheduling consideration because the DSP48A only has a two-input adder (Figure 21) unlike the DSP48E which has a three-input adder (shown in Figure 18). The method described below was used to work around this limitation and take advantage of the built-in Pout port.

The real output can be calculated by the algorithm below:

In the first DSP48A:

1. DSP48A calculates the real product for channel 1, that is, $x_{I_1}(n) * \cos(w_1 n)$, and stores it in the P register. Also, a rounding constant is added into the sum from the C port.
2. DSP48A calculates the real product for channel 2, that is, $x_{I_2}(n) * \cos(w_2 n)$, and adds it to the results stored at P.
3. DSP48A calculates the real product for channel 3, that is, $x_{I_3}(n) * \cos(w_3 n)$, and adds it to P. This gives the partial sum of

$$\sum_{k=1}^3 [x_{I_k}(n) * \cos(w_k n)]$$

In the second DSP48A:

1. DSP48A calculates the imaginary product for channel 1, that is, $x_{Q_1}(n) * \sin(w_1 n)$, subtracts it from the partial sum (latched in through the Pin port) produced by the first DSP48A in step 3. This new value is stored in register P.
2. DSP48A calculates the imaginary product for channel 2, that is, $x_{Q_2}(n) * \sin(w_2 n)$, and subtracts it from the results stored at P.
3. DSP48A calculates the real product for channel 3, that is, $x_{Q_3}(n) * \sin(w_3 n)$, and subtracts it from P. This gives the required value of

$$\sum_{k=1}^3 [x_{I_k}(n) * \cos(w_k n) - x_{Q_k}(n) * \sin(w_k n)]$$

Note: The three steps used in first and second DSP48A are not simultaneous. Some pipeline registers are needed to delay the data before entering the second DSP48A. After step 3, the second DSP48A holds the composite value for the real part. This is then captured, down sampled, and rounded (the constant has been added in step 1 in the first DSP48A) to the output bit width and then is presented as the final output. After that, another 3-step process begins.

Digital Down-Converter

The Digital Down-Converter shows the high-level block diagram of the CDMA2000 DDC. Unlike the WCDMA example, the DDC input is assumed to be complex, coming from two different ADCs. It is sampled at $50 \times F_{\text{chip}} = 61.44$ MSPS and quantized to 16 bits. This is done to demonstrate how another possible DDC architecture can be efficiently implemented. A real input can also be used by grounding the Q component.

The DDC translates the IF signals back to 0 Hz and down converts the samples $2 \times$ chip rate 2.4576 MSPS using a cascade of filters. Three complex baseband outputs are generated at a sample rate of $2 \times F_{\text{chip}} = 2.4576$ MSPS.

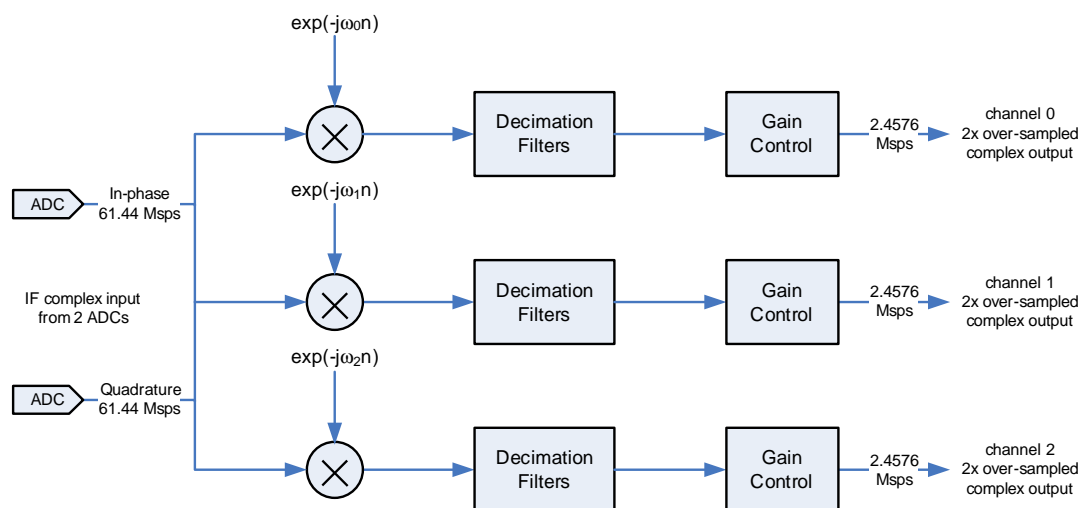


Figure 59: High-Level Block Diagram for CDMA2000 DDC

Performance Requirements

A summary of requirements for the DDC design is listed in [Table 28](#).

Table 28: Target Specification for CDMA2000 Uplink Receive Path

Parameter	Value	Notes
Carrier Bandwidth	1.25 MHz	
Number of Carriers	3 carriers	
Total Bandwidth	5.0 MHz	3 carriers positioned in a 5 MHz spectrum
IF Sample Rate	61.44 MSPS	50×1.2288 MSPS
DDC Output Rate	2.4576 MSPS	$2 \times$ Chip Rate
Input Signal Quantization	16-bit I and Q	Complex
Output Signal Quantization	17-bit I and Q	Complex
Mixer Properties	Tunability: Variable Resolution: ~ 0.25 Hz SFDR: up to 115 dB	Various SFDR are supported by the DDS Compiler core (28 bits of frequency accumulator width)

Frequency Translation

The mixer translates three IF channels from a set of specified center frequencies to 0 Hz by multiplying the complex input signal with the appropriate complex exponential for each channel. Mathematically, the mixer output for channel k , where $k = 1, 2, 3$, is:

$$y_k(n) = x_k(n) \cdot e^{-j\omega_k n} = x_k(n) \cdot (\cos(\omega_k n) - j \sin(\omega_k n))$$

Where $x_k(n) = x_{I_k}(n) + j \cdot x_{Q_k}(n)$

A three-channel DDS generates the $[\cos(\omega_k n), -\sin(\omega_k n)]$, $k = 1, 2, 3$. The DDS is driven by three separate frequency commands, which drive a time-shared frequency-to-phase accumulator that generates a sawtooth waveform that addresses a cosine/sine lookup table.

The real and imaginary components at the output of the mixer are

$$y_{I_k}(n) = x_{I_k}(n) \cos(\omega_k n) + x_{Q_k}(n) \sin(\omega_k n) \text{ and}$$

$$y_{Q_k}(n) = x_{Q_k}(n) \cos(\omega_k n) - x_{I_k}(n) \sin(\omega_k n).$$

DDC Filter Design

Filter Design Strategy

The DDC needs to decimate and pulse shape the samples from 61.44 MSPS to 2x chip rate of 2.4576 MSPS. This down sample factor of 25 is easily partitioned to two stages of decimator with rate change of 5. However, to embed a decimation rate of 5 to the matched pulse shaper with reasonable system performance is not practical to achieve in reality. Instead, the matched pulse shaping filter is better designed as a single rate filter.

The shaper follows two anti-aliasing decimation-by-5 filters, as shown in Figure 60. The lengths of these two filters are 17 and 47 taps, respectively. To design a direct down-sample-by-25 filter with the same performance requires 158 taps at least.



Figure 60: CDMA2000 DDC Decimation Filters Chain

First Decimation-by-5 Filter

The first decimation filter reduces the sampling rate of each down converted channel by 5, from 61.44 MSPS to 12.288 MSPS. To preserve all the information in the band of interest and avoid any aliasing effect during the down sampling process, the passband edge is set to 0.59 MHz and the stopband edge is set to $61.44/5 - 0.59 = 11.698$ MHz. The stopband attenuation is chosen to be 85 dB to reject the spectral image replica below a certain level. The passband ripple was the least important of the four, since the SINR is dominated by the inter-chip interference in the CDMA2000 standard, and a small ripple in the passband does not degrade the overall SINR performance too much. Further details are discussed later in the “Channel Filter” section.

A 17-tap symmetric filter with coefficients quantized to 18 bits is sufficient to meet the requirements. The magnitude response of this decimation-by-5 filter is shown in Figure 61.

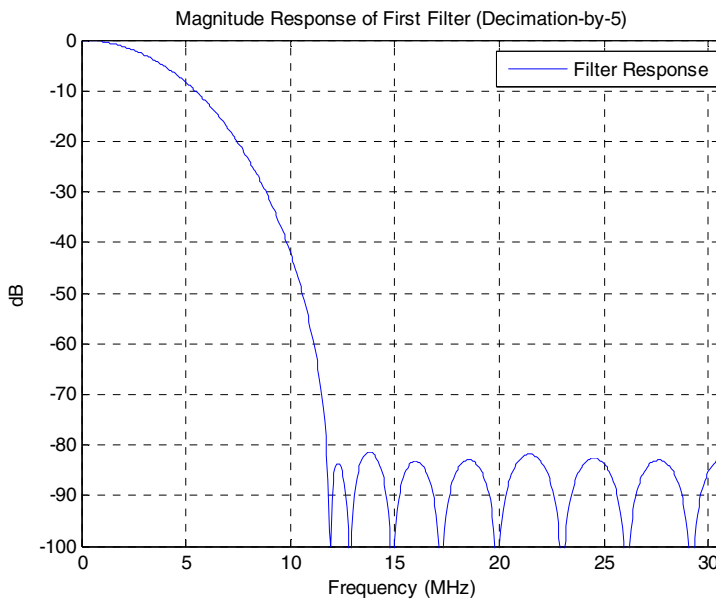


Figure 61: Magnitude Response of First Decimation-by-5 Filter

Second Decimation-by-5 Filter

The input sampling rate of the second decimator is 12.288 MSPS. The filter further decimates the sampling rate by 5 to a rate of 2.4576 MSPS. The designed decimator has a passband edge of 0.59 MHz (same as the first decimator), a stopband edge of $12.288/5 - 0.74 = 1.7176$ MHz, a peak-to-peak ripple of 0.01 dB, and a stopband attenuation of 85 dB. A symmetric filter with an order of 46 (47-tap) is designed, and with the coefficient quantized to 18 bits the filter meet the specification. Figure 62 shows the magnitude response of this decimation-by-5 filter.

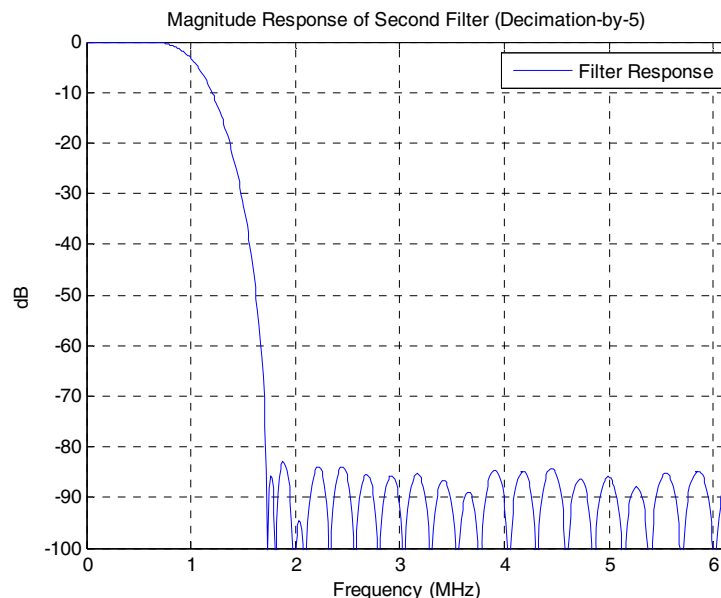


Figure 62: Magnitude Response of Second Decimation-by-5 Filter

Channel Filter

As mentioned in the WCDMA example, 3GPP defines a Nyquist type pulse shaping filter. Both the transmitter and receiver channel filters are RRC filters. Therefore, no inter-symbol interference is introduced in this process. However, unlike the 3GPP standard, the CDMA2000 standard specifies a non-Nyquist filter as the pulse shaping filter—a 48-tap IS-95 shaped FIR filter. Even with a matched filter at the receive side, a significant inter-symbol and inter-chip interference is seen. It has been shown in the 3GPP2 literature that the SINR is 16.55 dB (before including the spreading gain) when using a matched IS-95 filter without the presence of noise.

Some research introduced ways to design an optimal receiver filter to increase the SINR in the CDMA2000 system [Ref 9]. However in this application note, the channel filter in the DDC was designed to match the transmitter pulse shaper as the emphasis is on the FPGA implementation. The optimal receiver design approach can be applied to the CDMA2000 radio using the methodology shown in this document.

As with the DUC, the same methodology was used to design the channel filter. Two basic filters serve as the basis: one is the IS-95 response $h(k)$ which provides the desired *shape* at the passband, and the other is an LPF with steeper attenuation (>40 dB) to provide better adjacent channel rejection. The channel filter is obtained by convolving these two filters. Note that the LPF has the same specification as the DUC: a passband edge of 0.59 MHz, a stopband edge 0.74 MHz, passband ripple of 0.01 dB, and a stopband attenuation of 42 dB.

A 69-tap channel filter meets the requirements. The magnitude response of the single-rate channel filter is shown in Figure 63. Note that this channel filter is single rate—both the input and output sampling rates are 2.4576 MSPS. This results in two samples per data chip. The 2x chip rate is because the RAKE receiver follows the DDC generally requires higher-resolution sampling than one chip to reduce the potential signal loss from the sampling point misalignment in the timing recovery process.

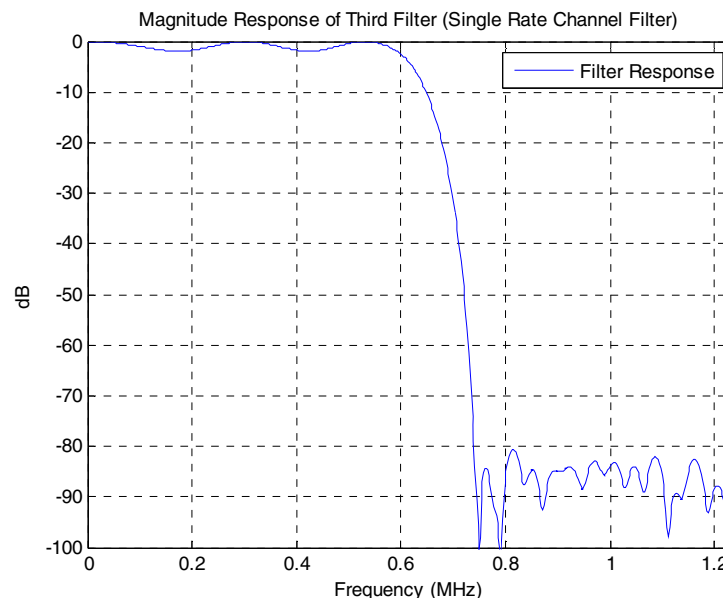


Figure 63: Magnitude Response of CDMA2000 DDC Channel Filter

Composite Filter Response

The specifications of all three filters are listed in Table 29. A 16-bit quantization for the input and 17-bit for the output is used in the DDC design.

Table 29: Summary of CDMA2000 DDC Filters

Filter Stage		Pass Band Fpass (MHz)	Stop Band Fstop (MHz)	Sample Rate Fs (MSPS)	Peak-to-Peak Ripple (dB)	Stop Band Attenuation (dB)	Filter Order
First Decimation-by-5 Filter		0.59	11.698	61.44	0.2	85	16
Second Decimation-by-5 Filter		0.59	1.7176	12.288	0.01	85	46
Channel Filter	h(k)	48-tap symmetric per spec and resampled to 2x					
	LPF	0.59	0.76	2.4576	0.01	42	45

DDC Performance

Several conformance tests defined in the 3GPP2 standard were performed. The results are illustrated and presented in this section. Since the data bit rate is at 9.6 kHz, it is assumed that the de-spreading gain is 21 dB.

Wideband Data Test

Here a three-channel CDMA2000 input is assumed with the AWGN presented and the SNR is set to 10 dB in a 61.44 MHz bandwidth. Figure 64 shows the DDC input. The DDC serves as a band selector, channelizing the signal from the channel centered at 0 Hz. The measured SINR after de-spreading is 37.33 dB. The DDC output is illustrated in Figure 65.

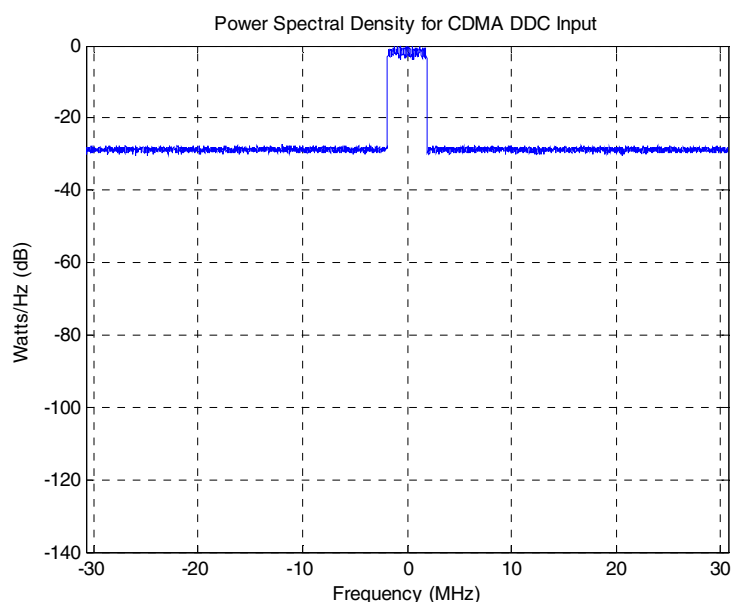


Figure 64: DDC Input with Three Channel CDMA2000 Composite Signal Plus AWGN

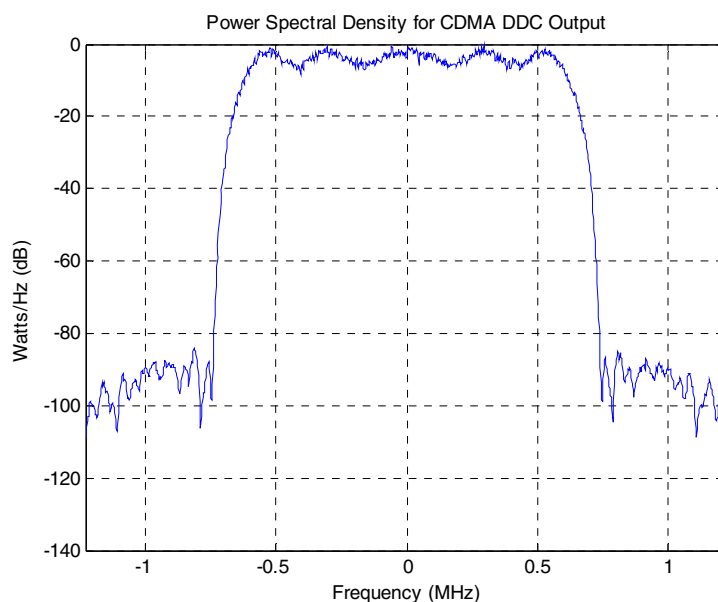


Figure 65: CDMA2000 DDC Output from Wideband Data Test

Adjacent Channel Selectivity

The adjacent channel selectivity test measures the ability of the DDC to receive the CDMA signal in the *desired* channel, with the presence of more two or more CDMA channels at a higher power level.

In the test, four other CDMA channels are adjusted to -53 dBm, and placed at ± 2.5 MHz and ± 5 MHz away from the center frequency of the desired channel (power level of -117 dBm). The input to the DDC is shown in Figure 66 after normalization.

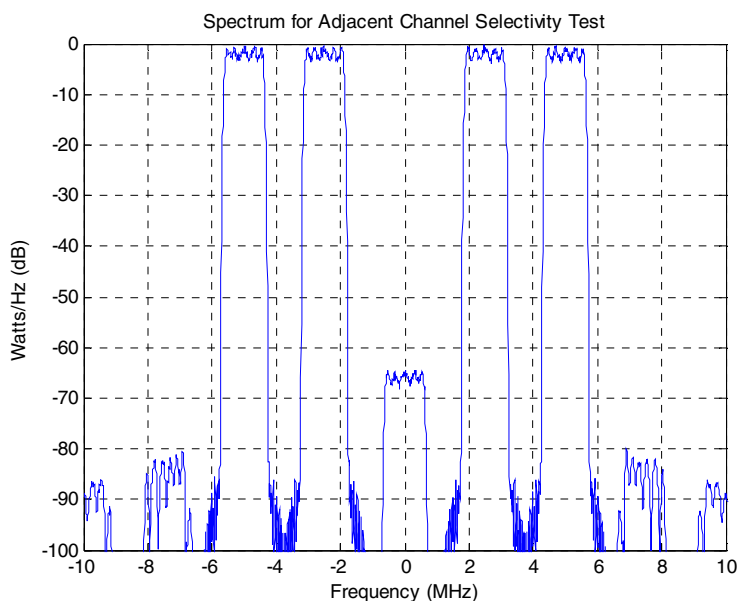


Figure 66: DDC input for Adjacent Channel Selectivity Test

The DDC output is shown in Figure 67. The SINR is 14.83 dB before de-spreading. With coding gain, the effective SINR is 35.90 dB.

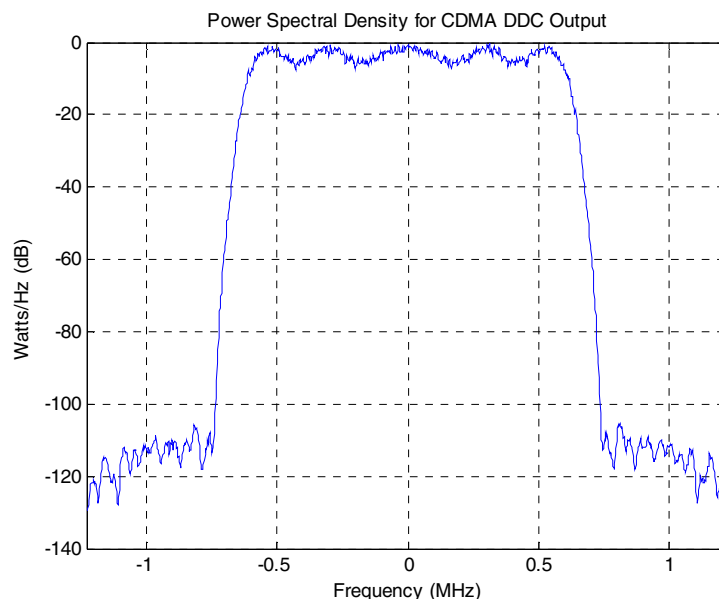


Figure 67: DDC Output for Adjacent Channel Selectivity Test

Single Tone Desensitization

The single tone desensitization is a measure of the ability to receive a CDMA signal on the assigned channel frequency in the presence of a single tone that is offset from the center frequency of the assigned channel. Dependent upon the band class, CW tones with different power level is used to simulate narrow-band interferer.

Three single tone tests to the DDC were performed, assuming a single-carrier scenario. The first single tone test has CW blockers 50 dB above the desired CDMA carrier power level, located at +750 kHz and -750 kHz from the lowest CDMA carrier center frequency. The receive spectrum is shown in Figure 68, with a measured SNR of 19.49 dB before de-spreading.

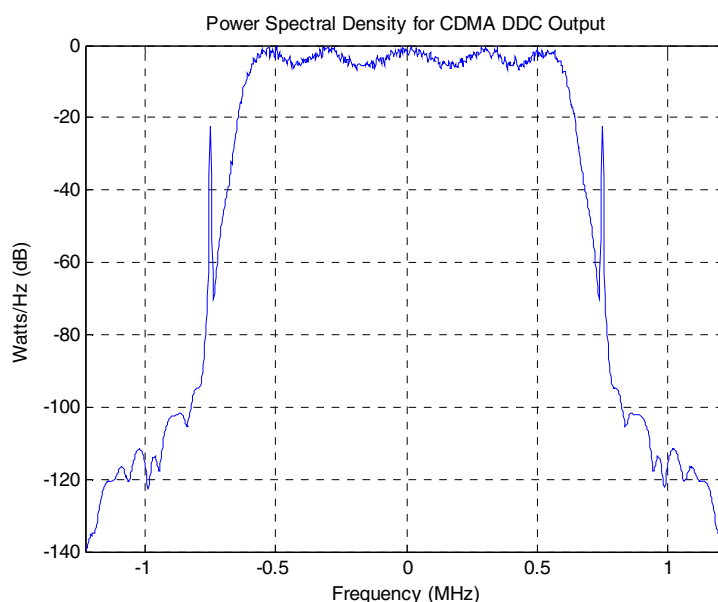


Figure 68: DDC Output from Single Tone Desensitization Test 1

The second test specifies the most stringent requirement—two blockers located at ± 900 kHz are above the desired signal power level by 87 dB. Under test, the designed filter was still able to channelize the desired signal, as shown in Figure 69. The SNR before de-spreading is -1.61 dB. Though this is below 0 dB, the effective SNR is 19.46 dB after adding the coding gain. This is well above the necessary SNR level to demodulate the CDMA2000 signal in the baseband.

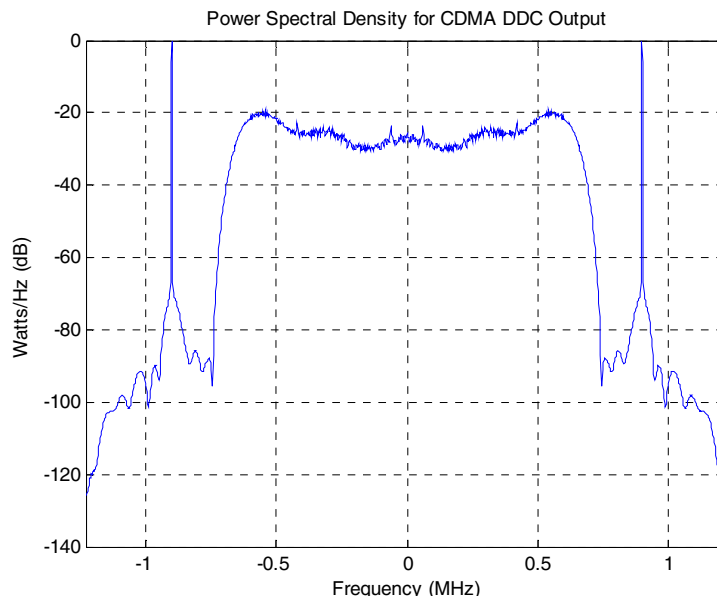


Figure 69: DDC Output from Single Tone Desensitization Test 2

Finally, the third test lists two blockers at ± 1.25 MHz. Though the tone is 80 dB above the desired signal level, it is farther away and does not create too much interference to the original signal. The effective SNR after accounting for the coding gain is 25.82 dB, and the power spectral density of the DDC output is shown in Figure 70.

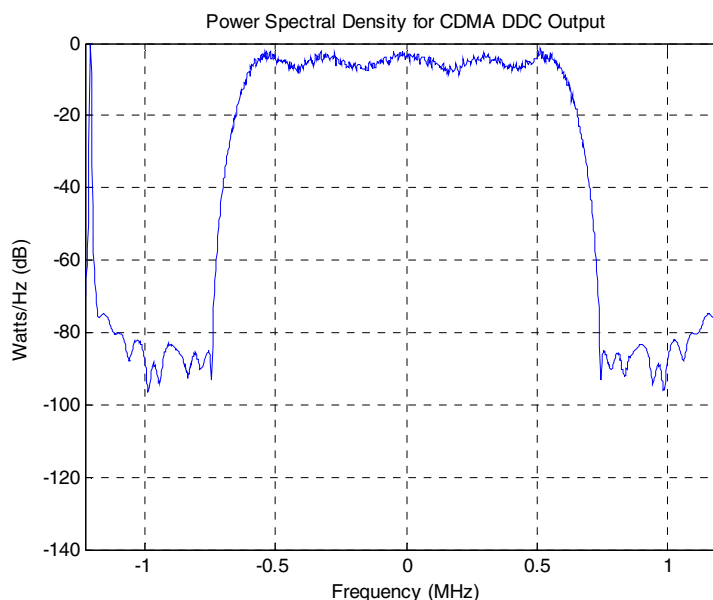


Figure 70: DDC Output from Single Tone Desensitization Test 3

Intermodulation Spurious Response

For the intermodulation conformance test, two pairs of narrow-band interferences are in present adjacent to the desired channel. The test was done using the band 0 specifications because the interference tones are closest to the desired signal and has the highest power of 72 dB above. The input to the DDC is shown in Figure 71. The DDC output is decent, showing its ability to receive the desired CDMA signal in the presence of non-linear elements caused by the third order mixing from those tones. The spectrum shown in Figure 72 has an effective SNR of 26.88 dB.

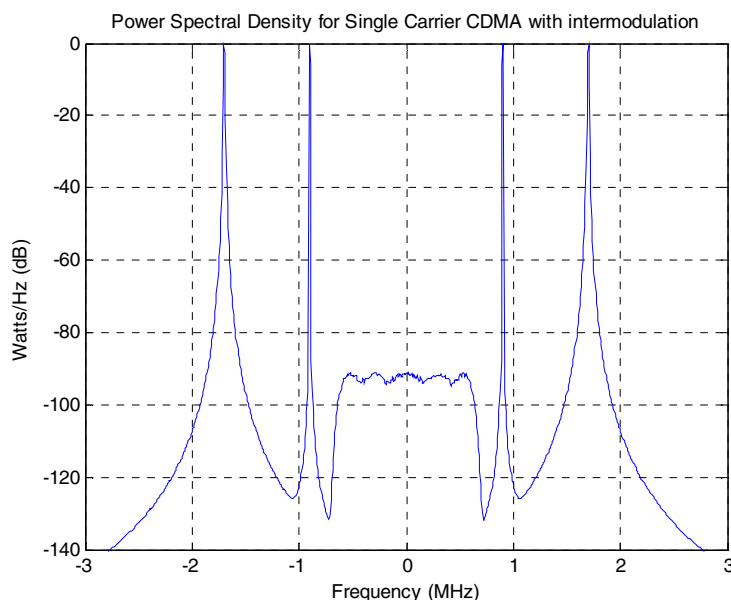


Figure 71: DDC Input for Intermodulation Test

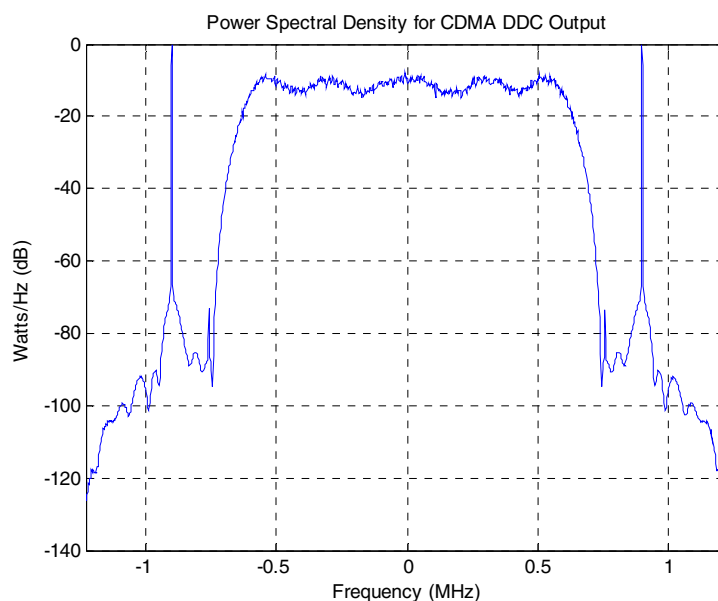


Figure 72: DDC Output from Intermodulation Test

Performance Summary

The results from all the tests are summarized in [Table 30](#).

Table 30: Summary of CDMA2000 DDC Conformance Test Performance Metrics

Test	SINR before de-spreading (dB)	Effective SINR after de-spreading (dB)	Coded Error Vector Magnitude (%)	Modulation Accuracy (Rho)
DDC input w/ AWGN	16.26	37.33	1.36	0.9998
ACS	14.83	35.90	1.60	0.9997
Single Tone 1	19.49	40.57	0.94	0.9999
Single Tone 2	-1.61	19.46	10.64	0.9888
Single Tone 3	4.75	25.82	5.11	0.9974
Intermodulation	5.80	26.88	4.53	0.9980

DDC Implementation Using System Generator

[Figure 73](#) and [Figure 74](#) show the block diagrams of the Virtex-5 and Spartan-DSP FPGA implementation, respectively. The concept to the CDMA2000 DDC implementation is similar to what used in the DUC implementation, except in reverse. Thus, instead of going into details, we only list the parameters used in the DDS and FIR compiler. The user should be able to derive the way to implement the complex multiplier based on the example given in the DUC section. The DDC also contains a scaler block after decimation filters which can boost the signal power to the appropriate level. The range is between 1/256 to 255. This scaler may be incorporated into an AGC loop by the user.

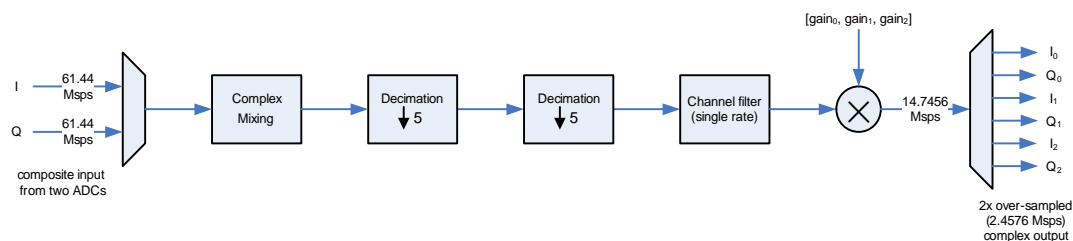


Figure 73: Block Diagram for CDMA2000 DDC Implementation Targeting a Virtex-5 FPGA

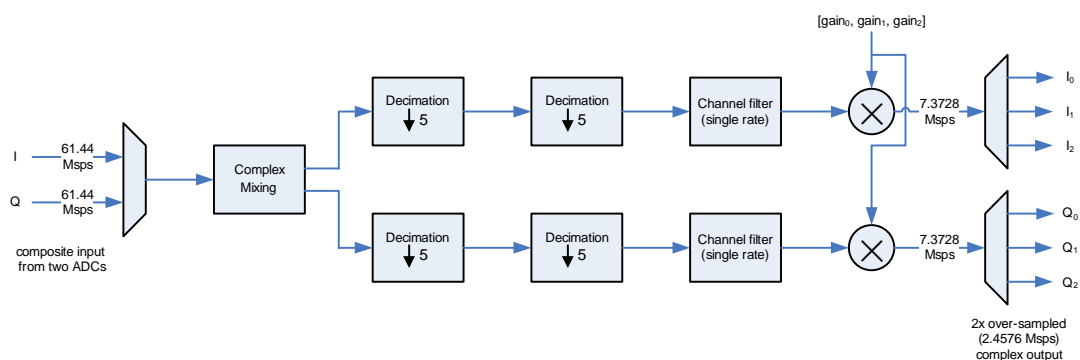


Figure 74: Block Diagram for CDMA2000 DUC Implementation Targeting a Spartan-DSP FPGA

The top-level System Generator design screen shots for Virtex-5 and Spartan-DSP device families are shown in Figure 75 and Figure 76, respectively.

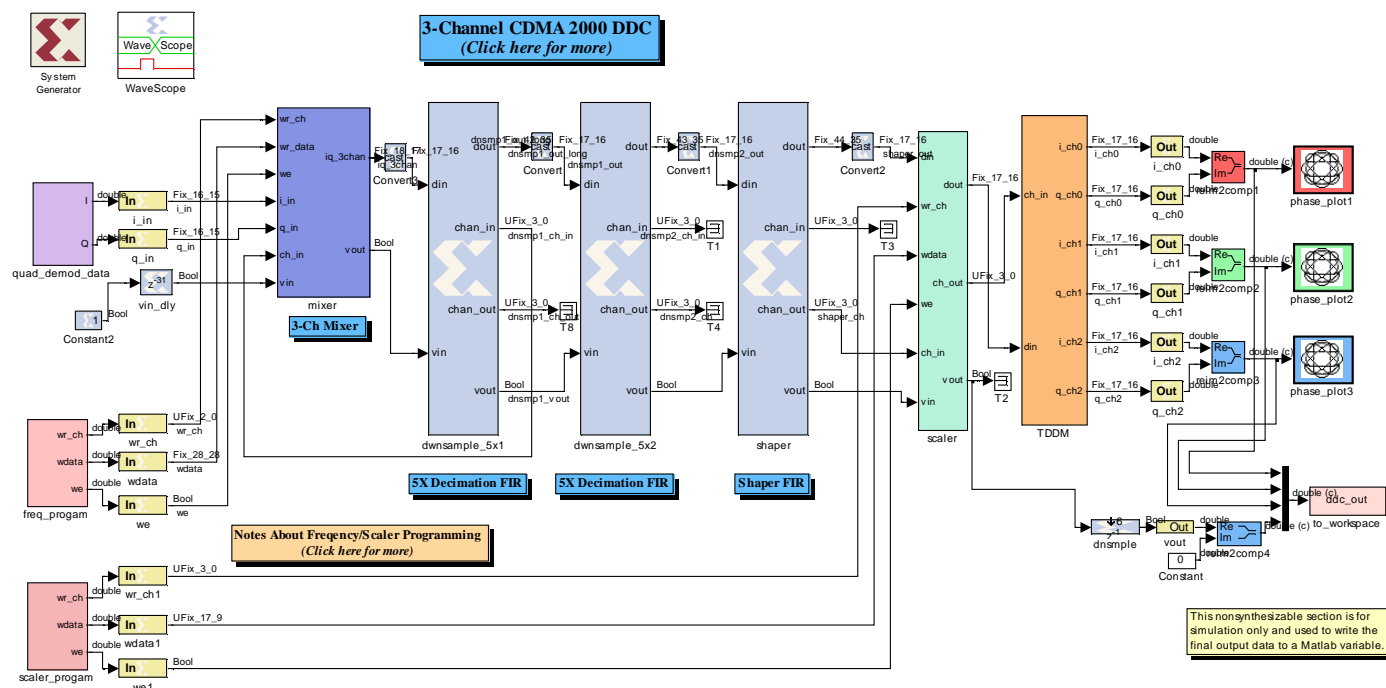


Figure 75: CDMA2000 DDC Implementation (Virtex-5 FPGA)

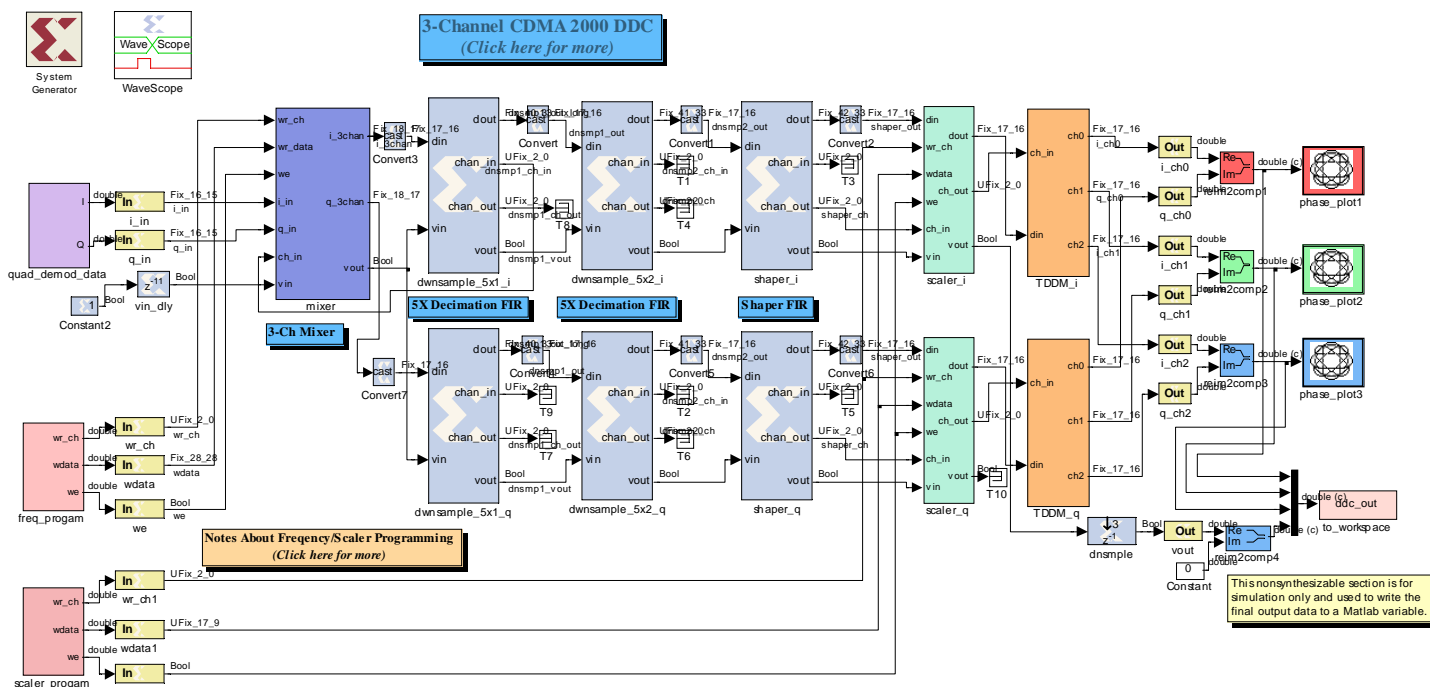


Figure 76: CDMA2000 DDC Implementation (Spartan-DSP FPGA)

Table 31 and Table 32 list the summary of parameters used in the FIR compiler and DDS compiler block, respectively, for the CDMA2000 DDC.

Table 31: Parameters Used in System Generator DDS Block for CDMA2000 DDC

DDS Clock Rate	184.32 MHz
Output Function	Both Sine (negative) and Cosine
SFDR	102 dB (can be specified up to 115 dB)
Frequency Resolution	0.25 Hz
Number of Channels	3
Output Frequency	Programmable
Noise Shaping	Taylor Series Corrected

Table 32: Parameters Used in System Generator FIR Compiler for CDMA2000 DDC

Parameter		Image Removing Filter 1	Image Removing Filter 2	Channel Filter
Filter Type		Decimation	Decimation	Single-Rate
Sample Rate Change		5	5	1
Number of Channels	Virtex-5 FPGA	6	6	6
	Spartan-DSP FPGA	3	3	3
Hardware Over-Sampling Rate (HOR)		1	5	25
Structure		Symmetric	Symmetric	Symmetric

DFE Resource Utilization Summary

The resources required by the DFE for the WCDMA and CDMA2000 air interfaces when targeting the Virtex-5 FPGA family are summarized in Table 33, Table 34, and Table 35. The slice counts for both the downlink transmit (DUC) and uplink receive (DDC) portions of the DFE were obtained using the default MAP packing factor. The numbers listed in parentheses in the tables are the percent utilizations of a Xilinx Virtex-5 XC5VSX50T device which contains 288 DSP48Es, 132 block RAMs (36k bits) and 8160 slices (52224 logic cells and 32640 Flip-Flops). Note that the resource utilization numbers for all designs were generated with registers on all the inputs and outputs.

Table 36 and Table 37 summarize the DFE resource utilization when targeting the Spartan-DSP FPGA family. As with the Virtex-5 FPGA designs, the numbers listed in parentheses in the tables are the percent utilizations of a Xilinx Spartan-DSP FPGA XC3SD1800A device which contains 84 DSP48As, 84 block RAMs (18k bits) and 37440 Logic Cells.

The DFE designs (except the ones developed in VHDL flow) were implemented using the System Generator 9.2 and Xilinx ISE 9.2i software. Xilinx Synthesis Technology (XST) tool was selected for synthesis. A high overall effort level was used for the place and route (PAR) tool. A clock frequency of 368.64 MHz was achieved for all the DFE design in a XC5VSX50T-1 device (slowest speed grade). For all designs in a Spartan-DSP FPGA XC3SD1800A -4 device, a clock speed of 122.88 MHz and 184.32 MHz was reached for the WCDMA and CDMA2000 DFE, respectively. As can be seen from the Fmax reported in Table 33 through Table 37, there are enough margins to account for clock jitter.

Table 33: Resource Utilization for WCDMA DFE (Virtex-5 FPGA Architecture, System Generator)

Design	DSP48Es	36 kbit Block RAMs	Slices	Flip-Flops	Logic Cells	Fmax (MHz)
DUC	7	1	664	1564	1143	435
DDC	10	1	637	1604	1165	397
Total DFE	17	2	1301	3168	2308	–

Table 34: Resource Utilization for WCDMA DFE (Virtex-5 FPGA Architecture, VHDL Flow)

Design	DSP48Es	36 kbit Block RAMs	Slices	Flip-Flops	Logic Cells	Fmax (MHz)
DUC	7	1	537	1247	1103	394
DDC	9	1	574	1241	1193	382
Total DFE	16	2	1111	2488	2296	–

Table 35: Resource Utilization for CDMA2000 DFE (Virtex-5 FPGA Architecture)

Design	DSP48Es	36 kbit Block RAMs	Slices	Flip-Flops	Logic Cells	Fmax (MHz)
DUC	12	3	764	1720	1240	401
DDC	12	3	711	1865	1346	383
Total DFE	24	6	1475	3585	2586	–

Table 36: Resource Utilization for WCDMA DFE (Spartan-DSP FPGA Architecture)

Design	DSP48As	18 kbit Block RAMs	Slices	Flip-Flops	Logic Cells	Fmax
DUC	13	1	1034	1320	1016	223
DDC	17	1	1181	1560	1301	223
Total DFE	30	2	2215	2880	2317	–

Table 37: Resource Utilization for CDMA2000 DFE (Spartan-DSP FPGA Architecture)

Design	DSP48As	18 kbit Block RAMs	Slices	Flip-Flops	Logic Cells	Fmax
DUC	22	9	1959	2404	2103	208
DDC	22	9	1668	2422	1984	196
Total DFE	44	18	3627	4826	4087	–

Power Consumption

The dynamic power measurements listed in this section were obtained using real hardware. The Virtex-5 FPGA power was measured using an AFX-FF1136-400 Prototype Board populated with an XC5VSX50T-1 FPGA. The Spartan-DSP FPGA power was measured using a Spartan-3A FG676 AFX-II Board populated with an XC3SD3400A -4 FPGA. Separate power supplies were used for the VCCINT (1.0V for the Virtex-5 FPGA board, 1.2V for the Spartan-3A FPGA board) voltage sources. The designs were clocked over a range of frequencies using a Stanford Research Systems CG635 synthesized clock generator. All measurements were performed at room temperature.

Static power was not measured because it is dependent on the density of the target device. For information on static power, refer to Xpower or the Power Tools section on the Xilinx website. The I/O power was also omitted because these designs have more I/O than would typically be used in a production design, where most of the configuration registers are likely driven from microprocessor interface logic. Likewise, the input data and output data need to interface to additional FIFO logic.

For the DFE design, a hardware test bench was created to provide stimulus to the ports of the design. The data inputs were driven by an internal block RAM based pattern generator. The dynamic power was only measured for designs that were implemented using the System Generator flow.

The dynamic power results for Virtex-5 and Spartan-DSP FPGAs for WCDMA and CDMA2000 designs are summarized in Table 38 and Table 39, respectively. The current draw from the design was calculated by subtracting the quiescent current measured with the clock turned off from the ICCINT at a full operating clock rate, for example, 368.64 MHz for Virtex-5 FPGAs, 122.88 MHz or 184.32 MHz for Spartan-DSP FPGAs. Therefore, the dynamic power consumption for the DFE design under test is $VCCINT \times (ICCINT \text{ at full clock rate} - ICCINT \text{ when clock turned off})$.

Table 38: Dynamic Power Summary for the Virtex-5 and Spartan-DSP FPGA WCDMA Designs

Design	Virtex-5 FPGA Power (W) at 368.64 MHz	Spartan-DSP FPGA Power (W) at 122.88 MHz
DUC	0.1653	0.2548
DDC	0.1725	0.2459
Total DFE	0.3378	0.5007

Table 39: Dynamic Power Summary for the Virtex-5 and Spartan-DSP FPGA CDMA2000 Designs

Design	Virtex-5 FPGA Power (W) at 368.64 MHz	Spartan-DSP FPGA Power (W) at 184.32 MHz
DUC	0.3397	0.5625
DDC	0.3525	0.5466
Total DFE	0.6922	1.1091

Interface Requirements

This section describes the interface of the DUC and DDC designs for both the WCDMA and CDMA2000 radios along with timing diagrams of the interface signals.

WCDMA DUC Interface Description

The top-level interface of the WCDMA DUC design is illustrated in Figure 77 and the port definitions are listed in Table 40. The number formats shown in Table 77 use the notation from System Generator. For example, Fix_16_12 represents a two's complement signed number that is quantized to 16 bits (includes the sign bit) with 12 bits of fraction. As a second example, the format Ufix_16_15 represents an unsigned number that is quantized to 16 bits with 15 bits of fraction.

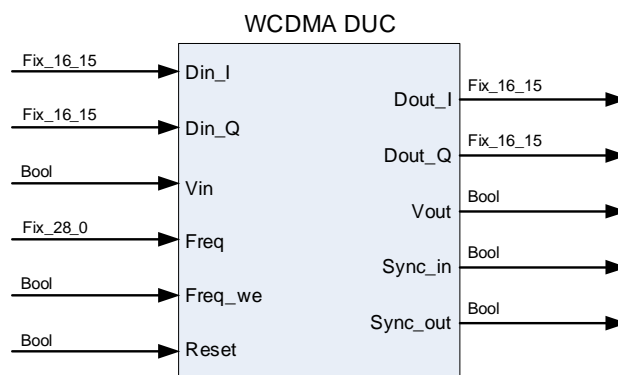


Figure 77: WCDMA DUC Top-Level Component

Table 40: WCDMA DUC Port Definitions

Signal	Direction	Description
Din_I[15:0]	Input	I component of chip-rate data input
Din_Q[15:0]	Input	Q component of chip-rate data input
Vin	Input	Active-High data valid signal, indicates arrival of new I/Q chip-rate data
Freq[27:0]	Input	Programmable center frequency for carrier
Freq_we	Input	Active-High center frequency write enable signal
Reset	Input	Active-High synchronous reset
Dout_I[15:0]	Output	I component of DUC output
Dout_Q[15:0]	Output	Q component of DUC output
Vout	Output	Active-High data valid signal for DUC output
Sync_in	Output	Sync signal that pulses once every few cycles based on the rate of the data input (only available in System Generator Design)
Sync_out	Output	Sync signal that pulses once every few cycles based on the rate of the data output (only available in System Generator Design)

The interface timing for designs using the System Generator flow and the VHDL flow are described in separate sections due to the fundamental clocking differences between the two flows. The design using the System Generator flow is based on a single clock, but implicit derived clock enable signals are used to control the multi-rate circuit. The advantage of this technique is that signals between sender and receiver logic do not need to travel at the full rate; it has multiple clock cycles to work with so that designs can generally more easily meet the timing requirements.

On the other hand, all timing to and from the designs using VHDL flow is based on a single clock domain, and all signals are assumed to be synchronous to this domain. This approach often results in a lower latency design without potential clock enable fan-out issues.

Note: The VHDL reference design bundle only includes the Virtex-5 FPGA designs. The Spartan-DSP FPGA design is implemented only in System Generator, but it should be straightforward enough to use the concept described in this application note if users were to implement it in VHDL.

WCDMA DUC Interface Timing

System Generator Design

The timing of the DUC input data interface using the System Generator flow is illustrated in [Figure 78](#). The V_{in} signal indicates the arrival of the chip-rate I/Q input data components. As mentioned earlier, implicitly derived clock enables are used throughout this design. Inside the DUC design, there is a set of input registers in front of the chip-rate data such as $D_{in_I/Q}$, V_{in} , and Reset signals. Another set of input registers with rate of 61.44 MHz are inserted before $Freq$ and $Freq_we$. This is to ensure that there is no combinational path between input ports to any internal logic. It is still the user's responsibility to perform timing analysis to guarantee that the data path from the external logic to this first set of registers meets the timing requirements, either 3.84 MHz or 61.44 MHz of the system.

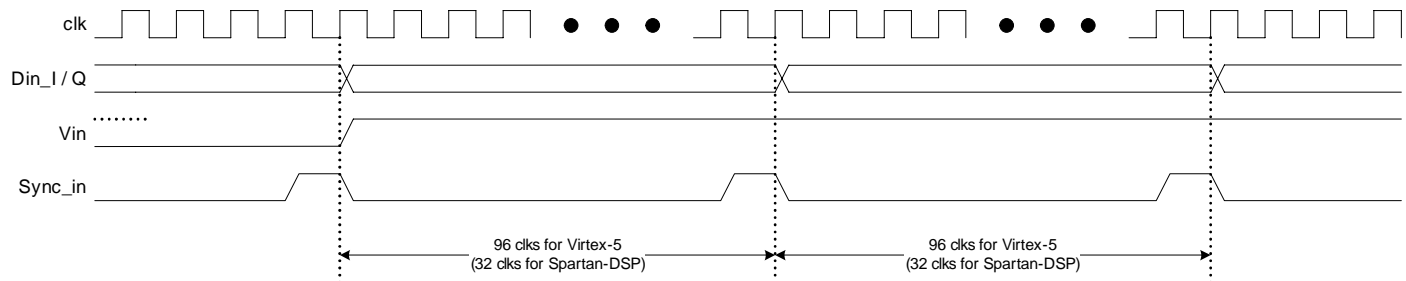


Figure 78: Timing Diagram of Input Data Interface for WCDMA DUC in System Generator

The timing of the DUC output data interface is illustrated in [Figure 79](#). The output data is a continuous data stream at a sample rate of 61.44 MSPS, which gives six clocks per output sample.

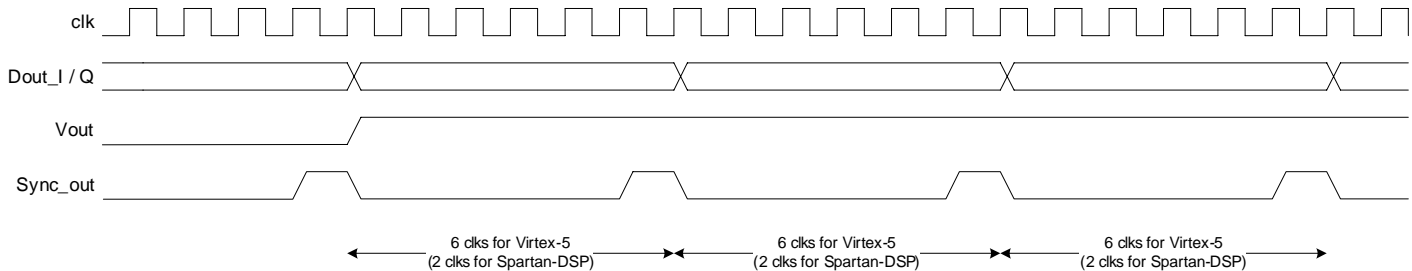


Figure 79: Timing Diagram of Output Data interface for WCDMA DUC in System Generator

Two extracted derived clock enable signals, $Sync_in$ and $Sync_out$, are offered at the output of the DUC design, as shown in [Figure 78](#) and [Figure 79](#), respectively. These cyclical pulses hold the behavior of an ideal clock enable signal used in the hardware implementation of a multi-rate circuit. For the Virtex-5 FPGA design, $Sync_in$ and $Sync_out$ pulse once every 96 and 6 cycles of FPGA clock, respectively. For the Spartan-DSP FPGA design, they pulse once every 32 and 2 cycles of FPGA clock, respectively, since the clock rate of 122.88 MHz is one third of that in the Virtex-5 design. Note that $Sync_in$ can serve as a good handshaking signal for chip-rate input data. Also, $Sync_out$, combined (AND'ed) with Valid signal, can be used as a valid signal to qualify the output data of the DUC and as the write enable to an interface FIFO.

The timing diagram of the $Freq$ and $Freq_we$ inputs is shown in [Figure 80](#). Note that to set the frequency increment content (at start-up or after reset), users have to load a target value on the $Freq$ port, and then assert $Freq_we$ to latch in the new value.

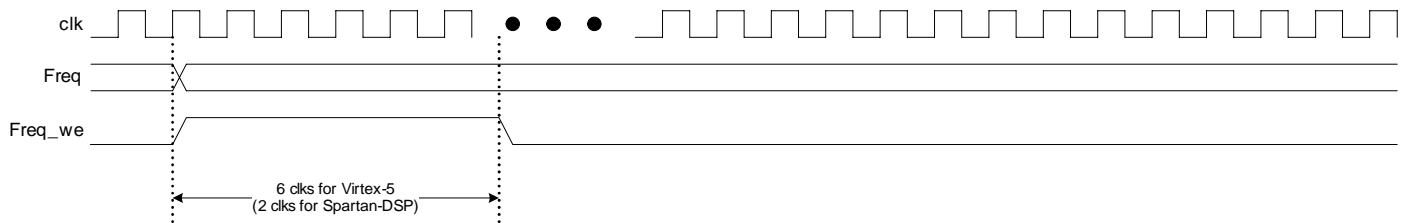


Figure 80: Timing Diagram of Freq and Freq_we for WCDMA DUC in System Generator

HDL Design

The timing of the DUC input data interface using the VHDL flow is illustrated in Figure 81. It is required that all the input data and control signals be synchronous to the rising edge of the clock as shown. Inside the DUC design, a set of high rate (368.64 MHz for Virtex-5 FPGA design) input registers is inserted to ensure that there is no combinational path between input ports to any internal logic. However, it is still the user's responsibility to perform timing analysis to guarantee that the data path from the external logic to this first set of registers meets the timing requirements of the system.

The RST is an Active High synchronous clear signal that resets most internal counters and state variables to their initial values. It can be asserted any time and as short as one clock pulse. The register that holds the frequency increment value is not affected by the RST signal. To set the frequency increment content (at start-up or after reset), users have to load a target value on the Freq port, and then assert Freq_we for at least one cycle to latch in the new value.

The Vin signal serves as an input data enable signal indicating the arrival of the chip-rate I/Q input data components. It can be asserted any time during the 96 clock cycle period when the same input data presents.

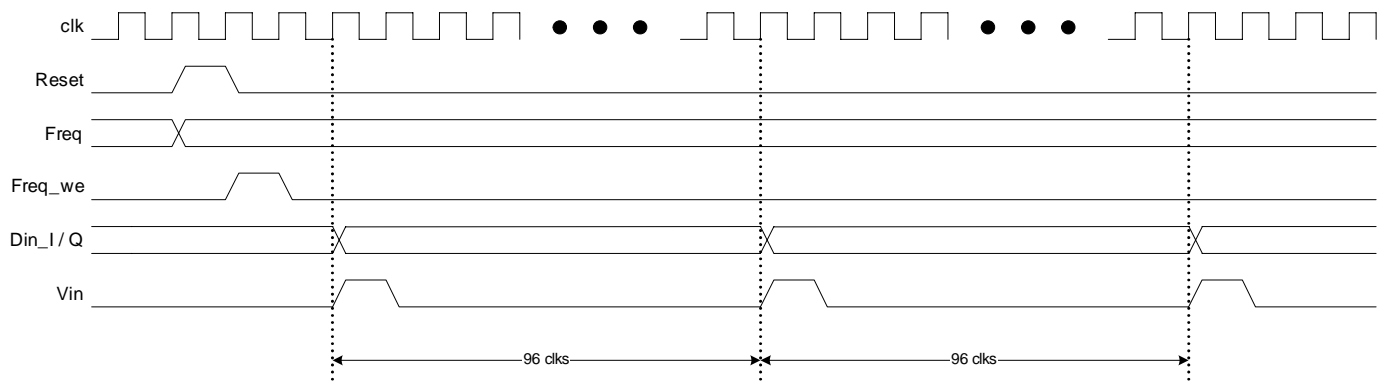


Figure 81: Timing Diagram of Input Interface for WCDMA DUC in VHDL

The timing of the DUC output data interface is illustrated in Figure 82. The output data is a continuous data stream at a sample rate of 61.44 MSPS, which gives 6 clocks per output sample. The Vout is an active-high output data valid indicator the can also be used as the write enable to an interface FIFO. The outputs of the DUC design are registered. However, additional timing analysis must be performed during integration to guarantee that the data path from the output registers to the interface FIFO logic meets the timing requirements of the overall system.

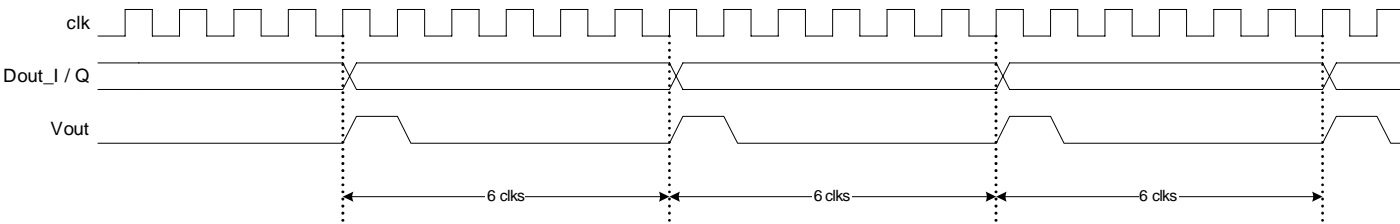


Figure 82: Timing Diagram of Output Interface for WCDMA DUC in VHDL

WCDMA DDC Interface Description

The top-level interface of the DDC is illustrated in Figure 83 and the port definitions are listed in Table 41. As with the WCDMA DUC, the interface timing for designs using the System Generator and the VHDL flows are described in separate sections, because of the fundamental clocking concept differences. Also, only the Virtex-5 FPGA designs are implemented using both the System Generator and the VHDL flows. The Spartan-DSP FPGA design is implemented only in System Generator.

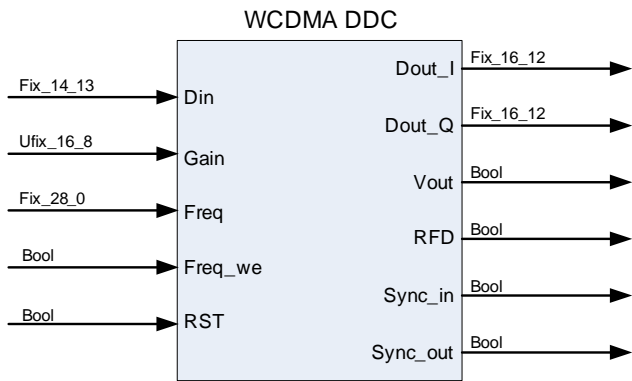


Figure 83: WCDMA DDC Top-Level Component

Table 41: WCDMA DDC Port Definitions

Signal	Direction	Description
Din[13:0]	Input	IF data input from antenna
Gain[15:0]	Input	Gain applied to DDC output
Freq[27:0]	Input	Programmable Center frequency
Freq_we	Input	Active-high center frequency write enable signal
RST	Input	Active-high synchronous reset
Dout_I[15:0]	Output	I component of DDC output
Dout_Q[15:0]	Output	Q component of DDC output
Vout	Output	Active-high data valid signal for DDC output
RFD	Output	Active-high ready for IF data signal
Sync_in	Output	Sync signal that pulses once every few cycles based on the rate of the data input (only available in System Generator Design)
Sync_out	Output	Sync signal that pulses once every few cycles based on the rate of the data output (only available in System Generator Design)

WCDMA DDC Interface Timing

System Generator Design

The timing of the DDC inputs is illustrated in [Figure 84](#). All of the inputs are registered at the input of the design at a rate of 61.44 MHz clock. The interface to these input registers should be on the same clock domain to guarantee that the signals meet the period constraint of the 61.44 MHz.

At circuit start-up or after reset, a target frequency increment value has to be loaded on the Freq port by asserting the Freq_we signal. The RFD indicator will go High once the circuit is ready to accept IF input data, an approximately 54 clock cycle after Freq_we is inserted dependent upon the DDS setting.

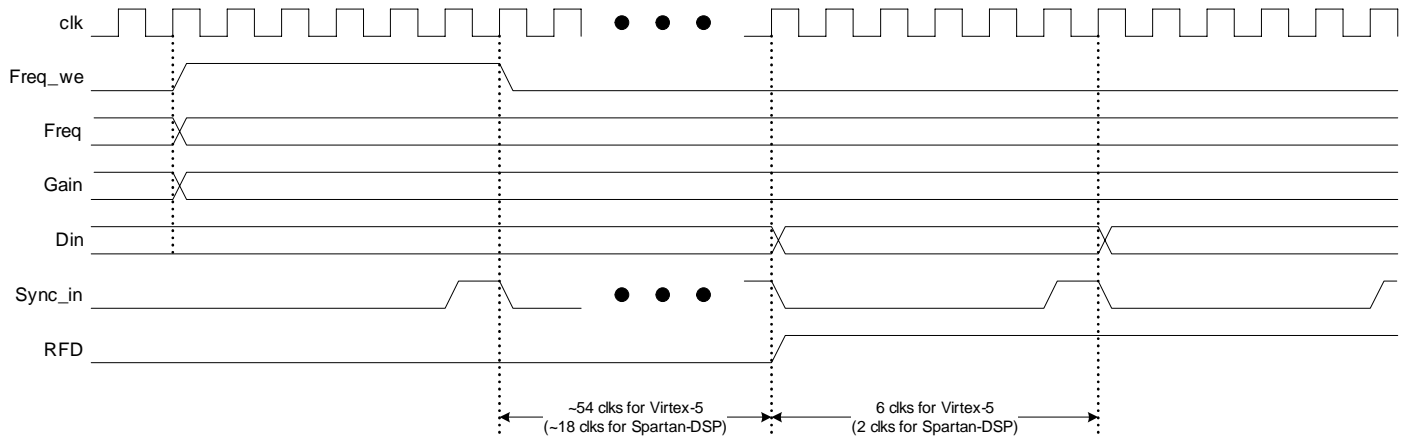


Figure 84: Timing Diagram of Din, Freq, Gain Inputs for WCDMA DDC in System Generator

The interface timing for the DDC output data is illustrated in [Figure 85](#). The output data has the throughput of 7.68 MSPS, and thus the spacing between samples is 48 clock cycles for the Virtex-5 FPGA design, and 16 clock cycles for the Spartan-DSP FPGA design.

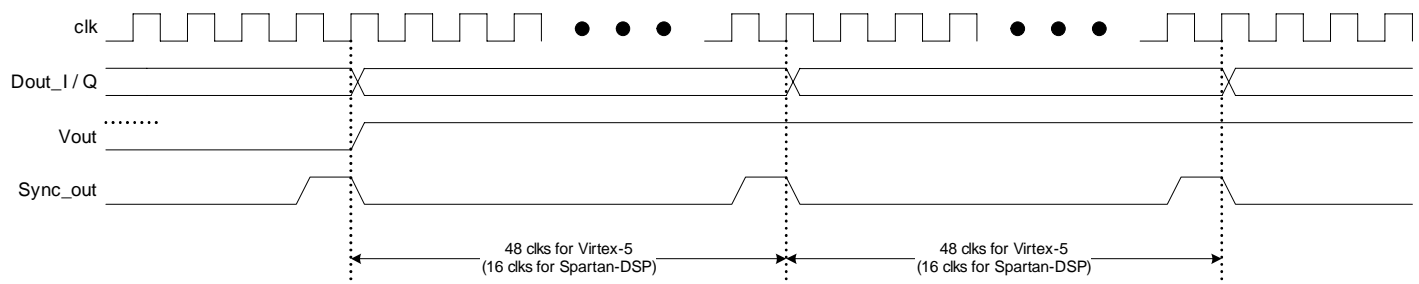


Figure 85: Timing Diagram of Output Data Interface for WCDMA DDC

As with interface in the DUC, Sync_in, and Sync_out signals shown in [Figure 84](#) and [Figure 85](#), respectively, are derived clock enable output signals that cyclically pulse once every 6 and 48 clock cycles for the Virtex-5 FPGA design. For the Spartan-DSP FPGA design, they pulse once every 2 and 16 clock cycles. They behave like ideal clock enable signals used in the hardware implementation of a multi-rate circuit. Sync_in can be used as a read enable to the interface FIFO for the DDC input, while Sync_out not only serves as a valid signal for the DDC output but can be used as a write enable to an interface FIFO.

HDL Design

The timing of the DDC inputs is illustrated in [Figure 86](#). All of the inputs are registered at the input of the design using the rising edge of the high rate (368.64 MHz for Virtex-5 FPGA) clock. The interface to these input registers should be on the same clock domain to guarantee that the signals meet the period constraint of the high rate clock. This is true regardless of how often the data actually switches. For example, the Din input have a sampling rate of 61.44 MSPS, which means there are six clock cycles per sample. Regardless of this fact, the interface to the Din register must meet the same period constraint as if the data were sampled at only one clock per sample.

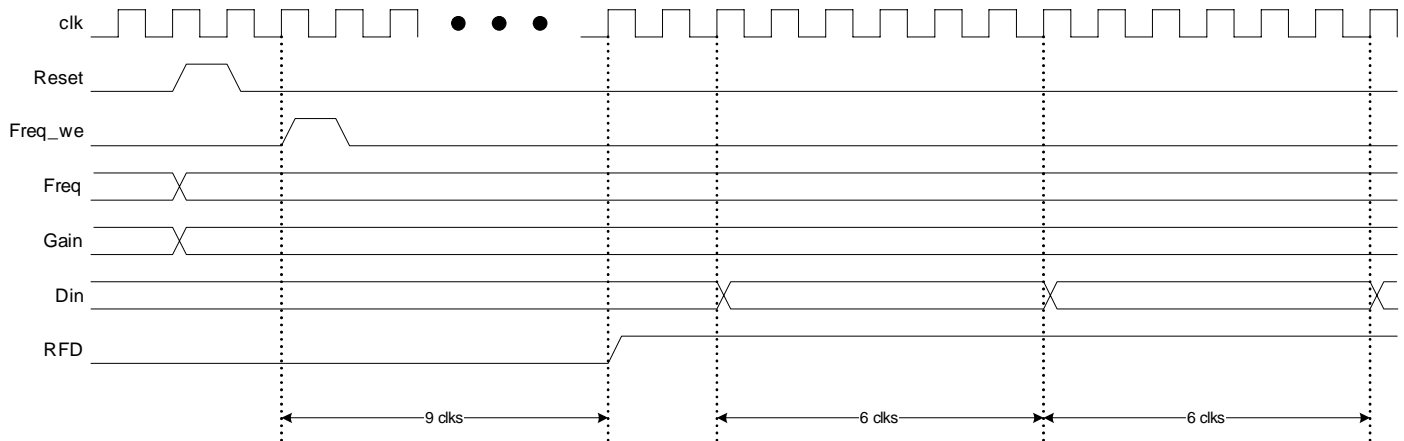


Figure 86: Timing Diagram of Data, Freq, Gain Inputs for WCDMA DDC

The RST is an active High synchronous clear signal that resets most internal counters and state variables to their initial values. It can be asserted any time and be as short as a one clock pulse. At circuit start-up or after reset, a target frequency increment value has to be loaded on the Freq port with Freq_we asserted for at least one cycle. This is because the internal frequency increment register is not reset by the RST signal. The RFD indicator goes High once the circuit is ready to accept IF input data, an approximately 9 clock cycle after Freq_we is inserted dependent upon the DDS setting.

The interface timing for the DDC output data is illustrated in [Figure 87](#). The output data has the throughput of 7.68 MSPS, and thus the spacing between samples is 48 clock cycles. The Vout signal indicates the arrival of the valid baseband output samples.

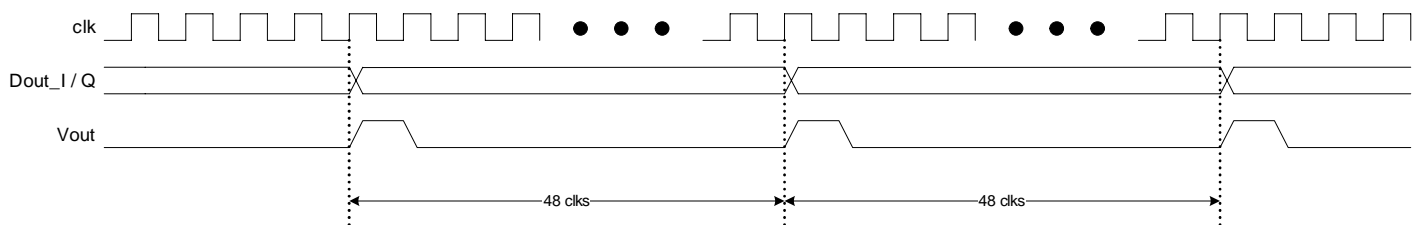


Figure 87: Timing Diagram of Output Data Interface for WCDMA DDC

CDMA2000 DUC Interface Description

The top-level interface of the CDMA2000 DUC design is illustrated in Figure 88 and the port definitions are listed in Table 42. As with the WCDMA DUC, the format used in Figure 1 follows the System Generator notations.

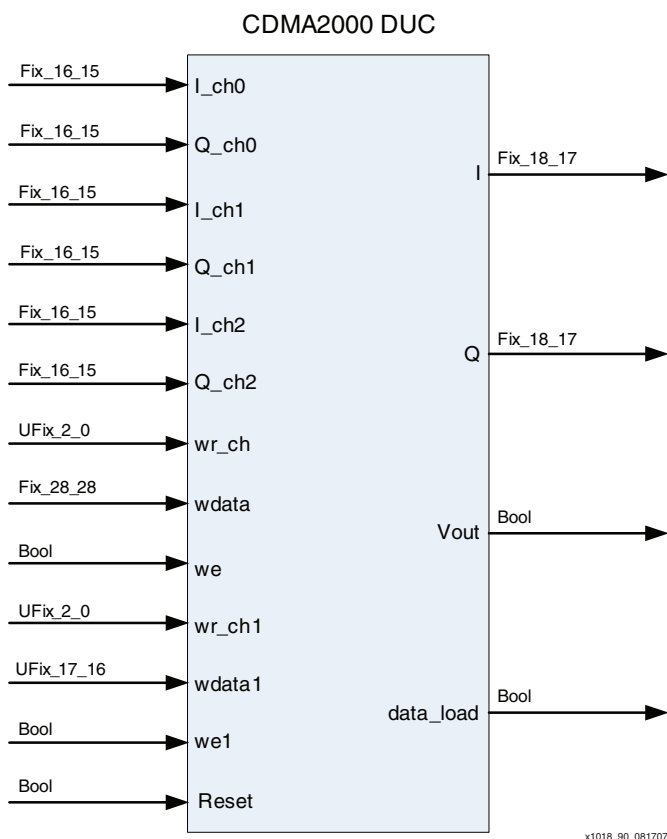


Figure 88: CDMA2000 DUC Top-Level Component

Table 42: CDMA2000 DUC Port Definitions

Signal	Direction	Description
I_ch0[15:0]	Input	I component of chip-rate data input for channel 0
Q_ch0[15:0]	Input	Q component of chip-rate data input for channel 0
I_ch1[15:0]	Input	I component of chip-rate data input for channel 1
Q_ch1[15:0]	Input	Q component of chip-rate data input for channel 1
I_ch2[15:0]	Input	I component of chip-rate data input for channel 2
Q_ch2[15:0]	Input	Q component of chip-rate data input for channel 2
wr_ch[2:0]	Input	Channel address to program center frequency
wdata[27:0]	Input	Programmable center frequency for carrier
we	Input	Active-High center frequency write enable signal
wr_ch1[2:0]	Input	Channel address to program scaling factor
wdata1[16:0]	Input	Programmable scaling factor for carrier
we1	Input	Active-High scaling factor write enable signal
Reset	Input	Active-High synchronous reset

Table 42: CDMA2000 DUC Port Definitions (Continued)

Signal	Direction	Description
I[17:0]	Output	I component of DUC output
Q[17:0]	Output	Q component of DUC output
Vout	Output	Active-High data valid signal for DUC output
Data_load	Output	Sync signal that is used to show when the six data inputs are latched into the design

CDMA2000 DUC Interface Timing

The timing of the 3-channel CDMA2000 DUC input data interface with data load output signal is illustrated in Figure 89. The inputs to the DUC are three complex chip-rate signals, where each complex input consists of I and Q pairs. A set of input registers are in front of the chip-rate data before those signals being up sampled and time-multiplexed into a single data stream. This is to ensure that there is no combinational path between input ports to any internal logic. It is user's responsibility to perform timing analysis to guarantee that the data path from the external logic to this first set of registers meets 1.2288 MHz timing requirements.

The output signal, data_load, is used to show when the six data inputs (ch0_i ~ ch2_q) are being latched into the TDM. The data inputs are latched on the last rising edge of the clock when data_load is asserted. The data_load signal is asserted for 100 cycles and 50 cycles for Virtex-5 and Spartan-DSP FPGA designs, respectively, so that its sample rate is six times the input data rate, i.e., $6 \times 1.2288 \text{ MHz} = 7.3608 \text{ MHz}$. The data_load signal can be upsampled to a faster rate by the user, if desired, and it is meant to be used to maximize the setup time on the data inputs.

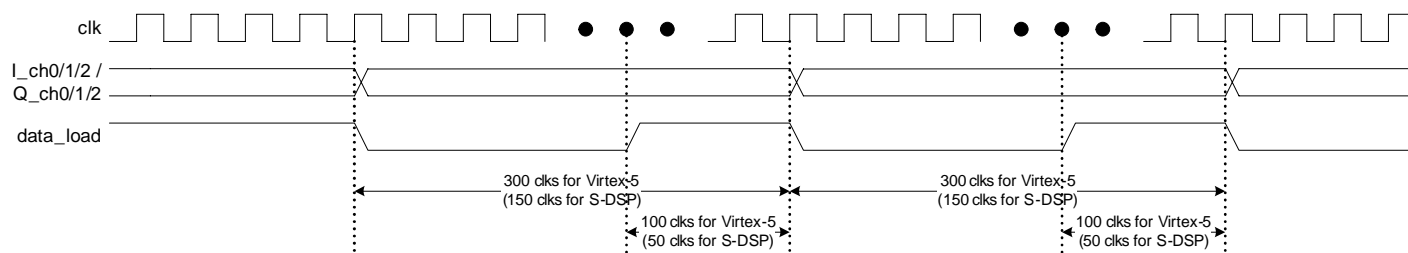


Figure 89: Timing Diagram of Input Data Interface (with Data_Load) for CDMA2000 DUC

The timing of the DUC output data interface is illustrated in Figure 90. The output data is a continuous data stream at a sample rate of 61.44 MSPS, which gives six clocks per output sample for the Virtex-5 FPGA design, and three clocks per output sample for the Spartan-DSP FPGA design. The Vout output signals when the valid data is generated.

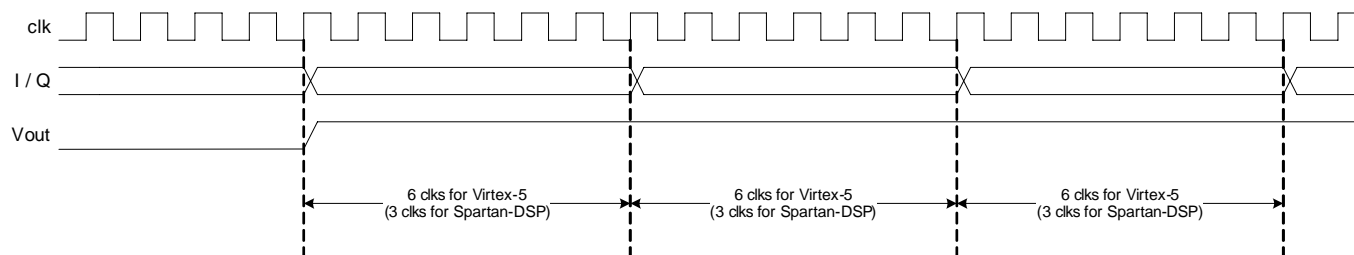


Figure 90: Timing Diagram of Output Data interface for CDMA2000 DUC

The timing diagram of the frequency and scaler programming ports are shown in Figure 91. The frequency and scaler programming ports are used to set the mixing frequencies and scale factors, respectively. The ports can be used by a microprocessor or connected to a bus. There

are three addresses for each port, 0-2, which correspond to the three complex input channels. To write to a port, set the address (wr_ch) to the desired channel, put the desired data on wdata, and assert the write enable (we). All writes are synchronous.

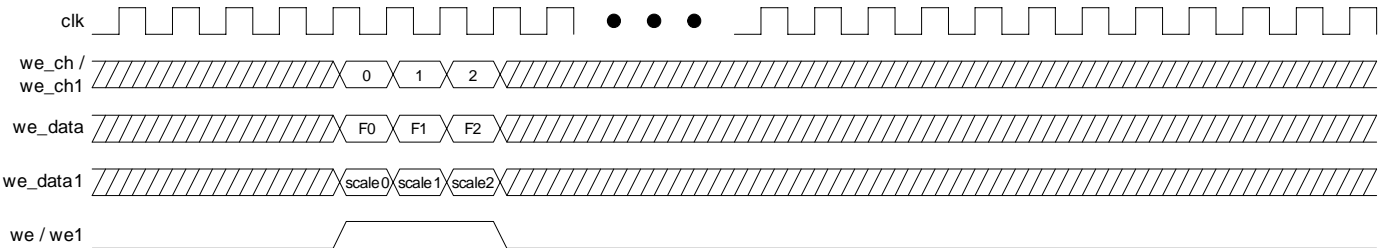


Figure 91: Timing Diagram of Frequency and Scaler Program for CDMA2000 DUC

CDMA2000 DDC Interface Description

The top-level interface of the DDC is illustrated in Figure 92 and the port definitions are listed in Table 43.

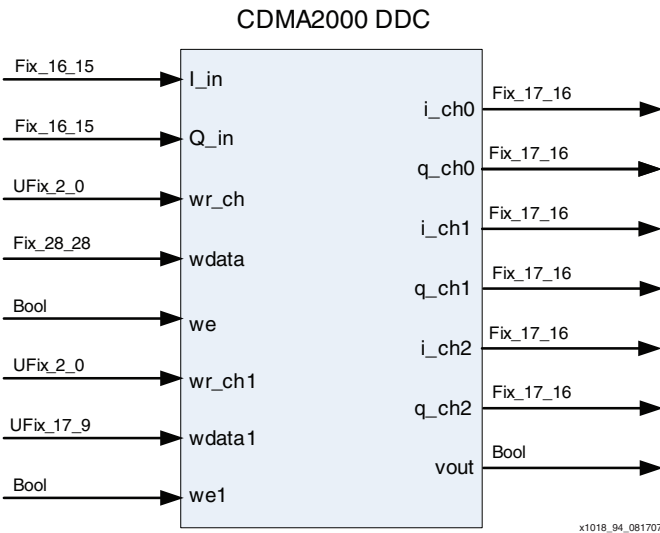


Figure 92: CDMA2000 DDC Top-Level Component

Table 43: DDC Port Definitions

Signal	Direction	Description
i_in[15:0]	Input	I channel IF data input from antenna
q_in[15:0]	Input	Q channel IF data input from antenna
wr_ch[2:0]	Input	Channel address to program center frequency
wdata[27:0]	Input	Programmable center frequency for carrier
we	Input	Active-high center frequency write enable signal
wr_ch1[2:0]	Input	Channel address to program scaling factor
wdata1[16:0]	Input	Programmable scaling factor for carrier
we1	Input	Active-High scaling factor write enable signal
i_ch0[16:0]	Output	I component of DDC output for carrier 0
q_ch0[16:0]	Output	Q component of DDC output for carrier 0
i_ch1[16:0]	Output	I component of DDC output for carrier 1

Table 43: DDC Port Definitions (Continued)

Signal	Direction	Description
q_ch1[16:0]	Output	Q component of DDC output for carrier 1
i_ch2[16:0]	Output	I component of DDC output for carrier 2
q_ch2[16:0]	Output	Q component of DDC output for carrier 2
vout	Output	Active-High data valid signal for DDC output

CDMA2000 DDC Interface Timing

The timing of the DDC inputs is illustrated in Figure 93. The data inputs are registered at a rate of 61.44 MHz clock. At circuit start-up, a target frequency value for each carrier has to be loaded on the wdata port by setting the corresponding wr_ch value and asserting we signal. The scaler programmer works the same way. With the corresponding channel address on the wr_ch1 and the correct scale factor on the wdata1 port, the scaler value is latched into the design by asserting the we1 to High.

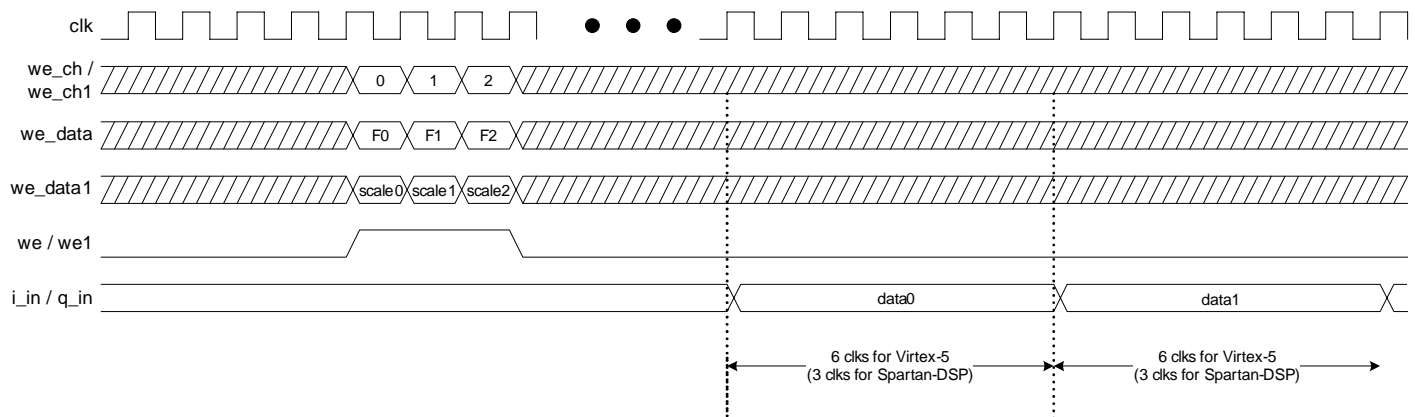


Figure 93: Timing Diagram of Input Interface for CDMA2000 DDC

The interface timing for the DDC output data is illustrated in Figure 94. The output data has the throughput of 2.4576 MSPS, and thus the spacing between samples is 150 clock cycles for the Virtex-5 FPGA design and 75 clock cycles for the Spartan-DSP FPGA design.

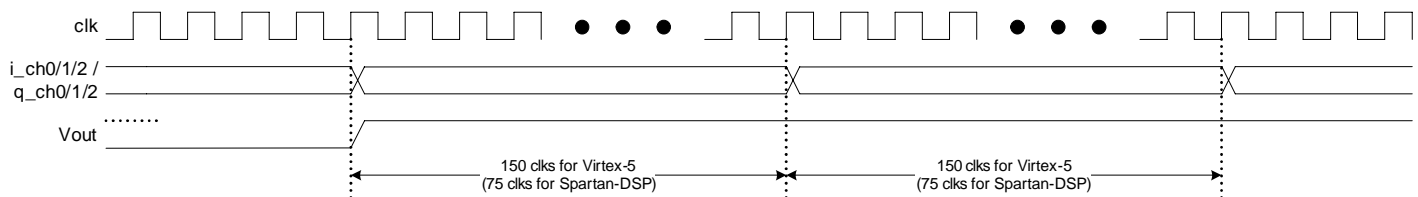


Figure 94: Timing Diagram of Output Data Interface for CDMA2000 DDC

Latency

This section summarizes the latency of each design, where latency is defined as the total delay from input to output, including both the algorithmic delays (for example, filter group delay) as well as implementation delays (for example, pipeline delays).

The latency was measured using the impulse response method. For the DUC, an impulse located at a certain input sample was used to stimulate the In-phase component of one carrier with its center frequency set to 0 MHz, while the Quadrature component of the carrier and the inputs of other two carriers were kept to zero. The peak locations of the DUC output were measured in clock cycles, and the difference between the input pulse and the output peaks are the data latency. This latency is dominated by the filter group delays, particularly from the channel filter.

A similar technique was used to measure the latency of the DDC design. A 96-clock cycle long input pulse was used to stimulate the Virtex-5 FPGA design, with one of the sub-channel frequencies offset set to zero. For the Spartan-DSP FPGA design for the WCDMA and CDMA2000 air interface, a 32- and 48-clock cycle long input pulses were used, respectively. The peak of the In-phase component output from that specific sub-channel was measured to calculate the data latency. The latency results for the DUC and DDC designs are listed in [Table 44](#). All numbers are approximate.

Table 44: Latency for DFE Designs

Design	Delay (clocks)	Delay (μ s)
Virtex-5 FPGA WCDMA DUC (System Generator)	1956	5.3
Virtex-5 FPGA WCDMA DDC (System Generator)	1608	4.36
Virtex-5 FPGA WCDMA DUC (VHDL)	1789	4.85
Virtex-5 FPGA WCDMA DDC (VHDL)	1426	3.87
Virtex-5 FPGA CDMA2000 DUC	6810	18.47
Virtex-5 FPGA CDMA2000 DDC	6300	17.09
Spartan-DSP FPGA WCDMA DUC	704	5.73
Spartan-DSP FPGA WCDMA DDC	584	4.75
Spartan-DSP FPGA CDMA2000 DUC	2823	15.32
Spartan-DSP FPGA CDMA2000 DDC	3228	17.51

System Integration

This section briefly describes how to integrate a System Generator design into a larger system using an NGC netlist and a VHDL black-box flow. For a more complete description, refer to [\[Ref 1\]](#), also available through the MATLAB Full Product Family Help menu.

There are several compilation targets that can be selected from the System Generator GUI (for example, HDL Netlist, NGC Netlist, Bitstream, etc.). For the design flow described here, the NGC Netlist option should be selected. An NGC netlist is a binary file recognized by the Xilinx ISE™ tools that contains the design netlist along with any constraint information. This format allows the design to be pulled into a larger design as a black box by NGDDBuild (Project Navigator Translation). When using the default options, the NGC netlist is stored in a file named <design_name>_cw.ngc.

In addition to creating the NGC netlist, System Generator typically creates two VHDL files for the design. The first file is named <design_name>.vhd and the second file is named <design_name>_cw.vhd. The _cw.vhd file is a top-level wrapper containing the System Generator design as a component along with a clock-driver module. The clock driver generates any inferred clock enables that can be in the design along with uniquely named clock signals for each clock enable. Even though the clock signals can have unique names, they are driven from the same clock in a single-clock design.

If the design is to be used as a black box, the appropriate synthesis attribute must be set in the code that instantiates it. For Synplify, the user must attach the **syn_black_box** attribute to the component and set it to true. For XST, the **box_type** attribute must be attached to the component and set to **black_box**.

When implementing the top-level design, the user must copy the NGC netlist from the System Generator subdirectory and place it in the project directory of the top-level design, allowing the ISE tools to pull the module into the top-level design during NGDBuild. The VHDL files, however, should not be copied to this directory because they can interfere with the NGC netlist black-box flow. For simulation purposes, the user can compile the VHDL files into a simulation library while leaving the source code in a separate directory. Although the VHDL files should remain in a different directory, any Memory Initialization Files (.mif) and Coefficient (.coe) files should be copied to the top-level project directory.

Conclusion

This application note and reference design example illustrates creative techniques in the design and realization of efficient and cost effective solutions to meet the demands of constant cost reduction requirements in wireless infrastructure and CPE equipment.

It also demonstrates in great detail that there is no one solution to any DUC/DDC configuration. Detailed up-front investigation of differing architectures and associated clock rates can reveal significantly smaller results than initially anticipated.

Using the class leading Xilinx DSP IP Portfolio solutions for any DUC/DDC design can be implemented with the minimum of effort on the engineers' behalf. The techniques in this application note by no means are exclusive to DUC/DDC for wireless, as they are generic techniques. It is highly likely, however, that with higher rate changes, such as encountered in many military and medical applications, architectures may be better suited to a Cascaded Integrator Comb (CIC) filter structure in addition to the standard FIR filter techniques shown in this application note.

References

1. 3GPP TS 25.104 v6.13.0, *Base Station Radio Transmission and Reception (FDD)*, June 2006.
2. [UG193](#), *Virtex-5 XtremeDSP Design Considerations User Guide*
3. [UG431](#), *XtremeDSP DSP48A for Spartan-3A DSP FPGAs User Guide*
4. [CORE Generator User Guide](#)
5. [DS534](#), *LogiCORE FIR Compiler v3.1 Data Sheet*
6. [DS558](#), *LogiCORE DDS Compiler v2.0 Data Sheet*
7. 3GPP TS 25.141 v6.15.0, *Base Station Conformance Testing (FDD)*, Oct. 2006.
8. 3GPP2 C.S0002-D v2.0, *Physical Layer Standard for CDMA2000 Spread Spectrum Systems*, Sept. 2005.
9. F. Wang, et al, *Optimal Receiver Filter Design with Applications in IS-2000 CDMA Systems*, IEEE Wireless Communications and Networking, March 2003.
10. [System Generator for DSP User Guide](#)
11. 3GPP2 C.S0010-C v2.0, *Recommended Minimum Performance Standards For CDMA2000 Spread Spectrum Base Stations*, Feb. 2006.
12. [XAPP921c v2.1](#),¹ *High Density WCDMA Digital Front End Reference Design*, Ed Hemphill, Helen Tarn, Michel Pecot, David Hawke, Jorge Seoane, and Paul Popescu, April 2007.
13. [XAPP569 v1.0.1](#), *Digital Up and Down Converters for the CDMA2000 and UMTS Base Stations*, August 2006.

1. Users must register to obtain this document.

14. Rabiner, Crochiere, *Optimum FIR Digital Filter Implementations for Decimation, Interpolation, and Narrow-Band Filtering*, IEEE Transactions on Acoustics, Speech, and Signal Processing, October 1975.
15. Ifeachor, Jervis, *Digital Signal Processing: A Practical Approach*, 1993.

Appendix A: Determining the Number of Interpolation/ Decimation Stages

Selecting the ideal number of stages for the interpolation/decimation process is somewhat of an art. In general, it is better to perform the interpolation/decimation in a cascade of smaller rate change stages than in one single stage with a large rate alteration. There are reasons from both system design and hardware implementation perspectives. In system design, a smaller rate change results in wider transition band (normalized to the sampling rate), which leads to much fewer taps in the filter design. From a hardware implementation perspective, when multiple stages are used, the earlier interpolation (or later decimation) stages operate at a lower sample rate, allowing the multipliers to be folded by a factor. Thus, the composite interpolator/decimator uses less hardware than a single stage, and the computational efficiency is improved significantly. (Folding a multiplier means to use it more than once during the calculation of a single sample.) In addition, the successive filtering stages have successively wider transition bands ([Ref 14], section III).

There is no closed-form solution to finding the optimal number of stages when the number of stages $I > 2$ ([Ref 15], p.505). This requires an exhaustive search. Two heuristics can help limit the search. First, the optimal number of stages rarely exceeds four ([Ref 15], p.505). Second, the decimation rates in each stage of an optimal decimator are decreasing, that is, $M_1 > M_2 > M_3 > \dots > M_i$, where M_i are the decimation rates of each stage i ([Ref 15], p.505). (The rates are increasing in an interpolator.) This latter heuristic requires that the decimation rates are allowed to be non-integers, whereas in this application note, we limit our search to integer values of M_i , meaning we may not find the optimal solution.

Let us use the CDMA2000 design as an example. The desired overall interpolation rate for the design is 50, which has a prime factorization of $2 \times 5 \times 5$. Using only integer interpolation techniques, we have a few options: a single stage of 50; two stages of 2, 25, or 10, 5 with two combinations of those; or three stages of 2, 5, 5 with three combinations of that. Only a true exhaustive search, performing the Parks-McClellan algorithm on all those possibilities to determine the exact number of taps required per stage, will determine the smallest possible implementation. Limiting our search to the integer solutions with increasing rates, that is, 50; 2, 25; and 2, 5, 5; we find that 2, 5, 5 requires the fewest multipliers.

The analysis of the problem is further complicated by a few factors listed below.

- A required filter for a particular stage may require a non-integer number of multipliers to meet its requirements. In that case, extra taps may be added “for free” until the multipliers are fully utilized. This can decrease passband ripple and relax requirements for successive stages.
- There are formulae to estimate the minimum number of taps required per stage, but no exact closed-form solution, so the Parks-McClellan algorithm may have to be performed over a range of tap numbers to find the optimal solution.
- One must take into account that some of the stages may be halfband (or, more generally, L^{th} -band) filters which use less hardware than the number of taps might imply. Shaping, normally done to reduce ISI, is typically done in the first stage. Since the typical shaping filter is not a Nyquist filter, it cannot be an L^{th} -band filter. (A raised-cosine shape does meet the Nyquist criterion, but a root-raised cosine shape does not.)
- The optimal solution may require fractional interpolation/decimation stages. This complicates the solution search. In this paper, we limit the search to integer solutions.

For more details on this subject, refer to [Ref 14] listed in the “References” section.

Appendix B: Brief Introduction on Filter Design Techniques

There are many methods available to determine filter coefficients and to build filter hardware. For those not versed in filter design, a brief survey of topics is covered here. This overview is by no means comprehensive and is not a substitute for a textbook on the material. As a preface to this discussion, an important concept should be kept in mind: the DSP48s in the various Xilinx FPGA architectures (such as Virtex-4, Virtex-5, and Spartan-DSP devices) should be embraced and exploited whenever possible. DSP48s are embedded hard cores; therefore, they are very fast and consume much less die space and power than reconfigurable fabric. Moreover, there are plenty of DSP48s on these Xilinx devices. Though there are many elegant and clever filter architectures that can save on multipliers, some of these date from an earlier era when multipliers were a scarce resource and may not be advantageous when implementing in these Xilinx FPGAs.

Calculating FIR Coefficients

Window Method of Coefficient Calculation

The window method for calculating filter coefficients is usually covered in most DSP textbooks and was at one time heavily used in practice. A window is selected and closed-form formulae allow for easy calculation of coefficients. The basic form is for a lowpass filter, but the method can easily be used for bandpass or highpass filters. This simplest window is the *rectangular* window, which is used for brick-wall filters. This is not a practical filter, because the impulse response, a sinc, is infinite in length and the coefficients drop off slowly. Other windows include *Hanning*, *Hamming*, and *Kaiser*. These all have a larger transition band and make various tradeoffs. The time-domain in pulse response of these windows has been pre-calculated in textbooks to simplify the determination of coefficients.

The window method is no longer frequently used because iterative methods like Parks-McClellan came into usage after computer programs for it were designed. The window method does not yield an *optimal* response—the desired response could have been obtained with fewer coefficients. Windows are still used to condition a signal prior to an FFT to remove *edge effects*.

Frequency Sampling Method

An arbitrary frequency response can be sampled at a regular interval and an inverse FFT may be performed on those points. This yields coefficients that, when used, yield a frequency response that is identical to the sampled points *at the sampling frequencies*. However, though this method is simple, it yields a poor response, which can vary widely between the sampled points, and ripple cannot be controlled. It is very susceptible to the locations of the points in the transition band.

Remez/Parks-McClellan

For general-purpose filters, this is the technique that is frequently used. For most filter designs, the engineer wishes to specify these parameters:

- Maximum ripple tolerance in passband(s)
- Width of transition band (between passband(s) and stopband(s))
- Minimum stopband attenuation of stopband(s)

Given these criteria, the optimal filter (that is, the one meeting these criteria having the fewest taps) has equiripple in the passband and stopband ([Ref 15], p.305). (That is, the peaks and troughs of the ripple are the same. All filters have ripple and the number of peaks in the ripple is directly related to the number of taps.)

Parks, McClellan, and Rabiner presented in 1975 a method using the Remez exchange algorithm of iteratively determining the optimal coefficients when given the criteria above. The number of passbands and stopbands, each with their own ripple and attenuation specifications, is arbitrary. The actual shape of the transition band is considered to be a *don't care* parameter. The Parks-McClellan program, as it is known, is available in MATLAB (as the *firpm* function)

and in many other filter-design software packages and may be downloaded at no cost from the Internet.

Root Raised Cosine (RRC) Filters

It is often necessary to use *shaping* filters in a radio design. One of the most common of these shaping filters is the RRC. A *raised cosine* (RC) filter response is one in which the transition band is shaped like a cosine curve. This response meets the *Nyquist criterion* because of its anti-symmetry. This means that it will have zero intersymbol interference (ISI), and the reconstructed received signal will have zero-crossings for each symbol's impulse response placed in such a way that at each sampling instant, the sample's value will be due to only a single symbol.

An RC response is characterized by α , the *rolloff factor*. This determines the *excess bandwidth*, or the amount of bandwidth used in excess of the minimum required to transmit data at the symbol rate. It is the width of the cosine-shaped transition band. The value of α is arbitrary (zero corresponds to a *brick wall* filter) and the tradeoffs are listed:

- A big α implies greater bandwidth requirements for the same data rate.
- A big α requires fewer coefficients, because the impulse response rolls off quicker due to the larger transition band.
- A big α implies less ISI due to errors in sampling instants, because the impulse response decays more quickly with time.
- A big α implies a smaller peak-to-average ratio, because the impulse response decays quicker. This means the excursions out of the constellation will be smaller. This has implications in the design of the amplifier hardware.

The RC response is usually split equally between the transmitter and receiver, yielding an RRC filter on each end. The RRC coefficients may be easily obtained from filter design software. The shape of the RRC impulse is a closed-form function, and it is simply sampled at the sampling instants to obtain the coefficients. The coefficients are truncated according to the amount of ripple desired.

L^{th} -Band Filters

L^{th} -band filters, of which the most popular is the halfband (where $L = 2$), can be used to reduce hardware complexity because many of the coefficients are zero. When a coefficient is zero, the product of the multiplication is zero, so that particular multiplication may be omitted. In the case of the halfband filter, every other coefficient is zero. No multipliers are required to calculate those taps. A halfband filter can be used to interpolate/decimate by two. The frequency response must meet the Nyquist criterion, which means that the filter curve is anti-symmetric (as seen on a linear, not log plot) about the point where the amplitude response is $\frac{1}{2}$. A brick-wall filter meets this criterion. The impulse response of the brick-wall filter is a sinc curve, which has zero-crossings at every other sample. A raised-cosine (but not root raised-cosine) frequency response also meets this criterion.

Filter Architectures

Fully-Parallel and Folded Architectures

A fully-parallel FIR filter has a multiplier for every tap. This is the version seen in textbooks and is often shown in the *direct* or *transverse* form (though the *transpose* or *systolic* forms are better for Virtex designs). Depending upon the speed at which the multipliers must operate, it may not be resource-efficient. If the clock speed is less than that at which the multipliers can operate or if the multipliers have *dead* cycles, then the filter may be able to be *folded*. At the most extreme example of this, the filter has only a single multiplier. These are sometimes called *MAC* or *MACC* (Multiply ACCumulate) FIR filters, because the central element is a MAC unit of the type found in a standard digital signal processor chip. The MAC FIR uses N clock cycles (where N is the number of taps, or coefficients) to calculate a single output sample, whereas the fully-

parallel version calculates an output sample for every cycle. The MAC version is hardware efficient but only appropriate for lower sample rates.

In between the fully-parallel and the MAC filter are intermediate levels of folding. For example, if one is building a 16-tap FIR and the system clock is 500 MHz and the FIR's output sample rate is 500 MHz, then a fully-parallel version containing 16 multipliers must be used. If, instead, the required output sample rate is only 500 MHz/4, then one may design a filter in which each multiplier may be used four times (with four different coefficients) during the four cycles to calculate each sample. This design requires only four multipliers. It is said to be *folded by a factor of four*.

Filters may also be folded into several channels. That is, in a multi-channel FIR, a multiplier may not only be used multiple times for each sample, but may also be reused for multiple channels.

More information on this topic can be found in the *XtremeDSP™ for Virtex-4 FPGA User's Guide*.

Transverse, Transpose, and Systolic

The textbook transverse FIR uses an adder tree. Binary adder trees are easy to understand, but they must be carefully placed on the die in a geometry that eliminates routing bottlenecks. Furthermore, they are not amenable to embedded hard cores because the tree size needs to be known *a priori* for ideal placement. The cascaded adder avoids these issues. The cascaded adder is used in the transpose and systolic FIR structures. The DSP48s have dedicated cascade routing, so the question of which type of adder to use in the Virtex architecture has already been decided. Use FIRs that are transpose or systolic. The FIR Compiler tool does this automatically.

More information on this topic can be found in the *Xtreme DSP for Virtex-4 FPGA User's Guide*.

Multiplier-Free Filter Designs

CSD

Using the method of Canonic Signed Digit (CSD) can reduce hardware usage, but only for constant-coefficient adder-based filters. The coefficients are written in a CSD format, in which each coefficient bit is no longer limited to 0 or 1, but to 0, 1, or -1. When the coefficient is written in this format, it can always be written in such a way that at least half the bits (and, on average, two-thirds) are 0. Then constant coefficient multipliers may be built of adds (or subtracts, for bits that are -1) and shifts, and the number of adds is minimized. No full multipliers are used. While elegant, this method is usually not appropriate for Virtex architectures, because the DSP48 multipliers are faster, smaller, and more power efficient than a series of fabric-based shift-adds.

Simulated annealing and other techniques can also be used to reduce the non-zero bits in the coefficients. Again, these may be advantageous in some cases, but often may not be appropriate for Virtex architectures. These techniques only work for constant-coefficient multipliers, in which one of the multiplicands is a constant. If *folding* is used to reuse a multiplier or the coefficient changes because the multiplier is part of an *interpolator* or *decimator*, then full multipliers must be built, negating any possible advantage of performing the multiplication in the FPGA's fabric rather than in the embedded DSP48 multiplier.

CIC

The Cascaded-Integrator Comb filter, used for interpolation or decimation, was codified by Hogenauer in 1984 and has been popular since. The main reason for its popularity is that it is extremely parsimonious with hardware, using nothing but integrators (accumulators) and combs (an adder, subtracter, and delay line). The coefficients for each stage of the filter are all '1'. The CIC has a frequency response with a lot of droop in the first lobe and large lobes in the stopband. It must often be coupled with a compensation filter to counteract the droop. However, it is useful in certain applications. It is a hardware-efficient method to accomplish very large sampling rate changes since it has no multipliers and can operate at a high IF.

Interpolators/Decimators

When rate changes are required, an *interpolator* or *decimator* type of filter is required. An interpolator is used to increase the sampling rate, that is, to add more samples to the data stream. This is done by first upsampling and then lowpass filtering. This is used, for example, in a digital up-converter. The sampling rate must be increased to allow mixing to an IF or to reduce requirements on the post-DAC analog reconstruction filter. When interpolating by an integer factor L , the impulse response is sampled at $N \cdot L$ instants, where N is the number of taps. A different set of N *phases* is used for each sample until the L phases have been cycled through.

Decimators are used to reduce the sampling rate when the desired bandwidth is lower than the sampling rate implies. This is done to ease processing burdens. Decimation is done by lowpass filtering and then decimating (throwing away samples). In practice, this is usually done by lowpass filtering at only every M^{th} sample, where M is the integer decimation rate.

Both of these structures have implications for filter structure. The filter will be optimally designed when the FIR Compiler is used. More information on this topic can be found in the *Xtreme DSP for Virtex-4 FPGA User's Guide*.

Symmetry

Many impulse responses may be sampled so the coefficients are symmetric (or, in the case of Hilbert transformers, anti-symmetric). Because the coefficients are symmetric, pre-adders may be used to combine two input samples before the multiplication occurs, halving the number of required multipliers. This means that while N additions must still be performed, where N is the number of taps, only $\lceil N/2 \rceil$ multiplications must be performed.

The pre-adders may be implemented in fabric. Certain Virtex families contain pre-adders such as Spartan-DSP family in the DSP48 which may be used for this purpose.

Rounding and Finite Word Length Effects

Finite word length effects, such as those that occur when using fixed-point arithmetic to represent the samples and coefficients, is a subject that is covered in most DSP textbooks ([Ref 15], p.348). Quantization errors occur when intermediate truncation/rounding occurs and at the filter outputs when the result is truncated/rounded. Truncation usually introduces a DC bias to a filter output. Convergent rounding or symmetric rounding is usually preferable to remove any DC bias. Even simple rounding is generally better than truncation. The DSP48s allow for various roundings to be performed. Several rounding methods were used in the DUC/DDC designs in this application note. For example, symmetric rounding is used for Virtex-5 FPGA designs, and a hybrid method that reduces hardware consumption but statistically has no DC bias is used for Spartan-DSP designs. More information on rounding hardware can be found in the *Xtreme DSP for Virtex-4 FPGA User's Guide*.

Appendix C: Description of WCDMA Reference Design Files in System Generator

This section describes the files that are included in the WCDMA reference design archive that are used in the System Generator tool flow. The WCDMA reference design zip file contains two directories: DUC and DDC. Each directory contains the `Model` and `Implementation` sub-directories. The MATLAB simulation model and various scripts are located under the `Model` directory, while the System Generator and VHDL source files and associated pre- and post-scripts are under the `Implementation` directory.

In each `Implementation` directory, there are three sub-folders: `Sp3a`, `Virtex5`, and `VHDL`. The `sp3a` and `Virtex5` directory contains the System Generator source targeting the Spartan-DSP FPGA and Virtex-5 FPGA, respectively. The VHDL directory is described in detail in ["Appendix D: Hardware Co-Simulation and Incorporate HDL models into System Generator"](#)

.” Table 45 and Table 46 list the files for the WCDMA DUC and DDC designs and their descriptions, respectively.

Table 45: WCDMA DUC Files and Descriptions

Directory		File Name	Descriptions
DUC\Model		umts_duc_fix_point_model.m	MATLAB fixed-point model
		calculate_metrics.p	Script used to calculate performance metrics for the WCDMA DUC
		duc_performance_metrics.p	
		filter2_conf2.m	Filter coefficients for various image removal filters
		filter2_conf3.m	
		filter3_conf4.m	
		halfband1.m	Filter coefficients for various halfband filters
		halfband2.m	
		halfband3.m	
		srrc.m	Filter coefficients for RRC filter
		plot_psd_umts_duc_mask.p	Emission mask plotting function
		umts_duc_filter_conf1.m	4 different filter configurations
		umts_duc_filter_conf2.m	
		umts_duc_filter_conf3.m	
		umts_duc_filter_conf4.m	
		wcdma_input_tm1.p	Input data for WCDMA generated from test model 1
		wcdma_input_tm3.p	Input data for WCDMA generated from test model 3
		xquantize.p	Special quantize function used in the MATLAB model
		imf_up4.fda	Filter design and analysis files using MATLAB Fdatool
		imf_up4_2.fda	
		imf_up8.fda	
		srrc.fda	
DUC\Implementation	\Sp3a	duc_umts_sp3a_v1_0.mdl	System Generator source file
		duc_umts_sp3a_init.m	Initialization file
		duc_umts_sp3a_post.m	Post-simulate script
	\Virtex5	duc_umts_virtex5_v1_0.mdl	System Generator source file
		duc_umts_init.m	Initialization file
		duc_umts_post.m	Post-simulate script
	\VHDL	VHDL source code and test benches. Refer to “Appendix D: Hardware Co-Simulation and Incorporate HDL models into System Generator” for details.	

Table 46: WCDMA DDC Files and Descriptions

Directory		File Name	Descriptions	
DDC\Model		umts_ddc_fix_point_model.m	MATLAB fixed-point model	
		acs_test_input.p	ACS test input generation script	
		dynamic_range_test_input.p	Dynamic range test input generation script	
		intermodulation_input.p	Intermodulation test input generation script	
		ddc_conf1_h1.m	Filter coefficients for various image removal filters	
		ddc_conf2_h1.m		
		ddc_conf2_h2.m		
		halfband1.m	Filter coefficients for various halfband filters	
		halfband2.m		
		halfband3.m		
		ddc_srrc.m	Filter coefficients for RRC filter	
		plot_psd_umts_duc_mask.p	Emission mask plotting function	
		umts_ddc_filter_conf1.m	Two different filter configurations	
		umts_ddc_filter_conf2.m		
		umts_ddc_input.p	Input data generation script	
		umts_duc_filter_conf1.m	WCDMA DUC filter configuration 1 coefficient generation script	
		wcdma_input_tm1.p	WCDMA Input data generated from test model 1	
		wcdma_interference.p	WCDMA Interference data generating script	
		xquantize.p	Special quantize function used in the MATLAB model	
		conf_h1.fda	Filter design and analysis files using MATLAB FDATool	
		conf2_h1.fda		
		conf2_h2.fda		
		ddc_srrc.fda		
DDC\Implementation		\Sp3a	ddc_umts_sp3a_v1_0.mdl	System Generator source file
			ddc_umts_sp3a_init.m	Initialization file
			ddc_umts_sp3a_post.m	Post-simulate script
		\Virtex 5	ddc_umts_virtex5_v1_0.mdl	System Generator source file
			ddc_umts_init.m	Initialization file
			ddc_umts_post.m	Post-simulate script
		\VHDL	VHDL source code and test benches. Refer to “ Appendix D: Hardware Co-Simulation and Incorporate HDL models into System Generator ” for details.	

The WCDMA reference design uses an initialization file to set the appropriate parameters. Figure 95 is a sample init file: (. \DUC\Implementation\Virtex5\duc_umts_init.m)

for the WCDMA DUC design targeting the Virtex-5 FPGA. Every parameter is associated with comments and is self-explanatory. The init file is called every time when the user simulates the design. The FIR compiler and DDS compiler discussed in a previous section were programmed using these variables.

```
% Run the fixed point simulation model to get filter coeff and input data
% The PSD plot performance metric calculation will only run when data doesn't exist
% or when sim_performance is set to 1

% add the path that includes simulation scripts and sub-functions
addpath ../../model

sim_performance = 0;      % set to 1 if simulation metric needs to be run (slower)
run umts_duc_fix_point_model;

% load baseband data input from simulation model
I_bb = real(u0_q);
Q_bb = imag(u0_q);

% chip rate - set chip rate to 1 as the base unit
% (it's not advisable to set it to a large number such as 3.84e6 as the reciprocal will be
% quite small and thus causes some numerical issues
Fcr = 1;
%Fcr = 3.84e6; %don't use

% number of TDM channels
NTDM = 2;

% input sampling rate for TDM signal
Fsin = NTDM*Fcr;

% DUC up-sampling rate
DucUpRate = 16;

% overclocking relative to IF (IF rate = 61.44 Msps)
OverClock = 6;

% desired FPGA operating frequency (FPGA rate = 368.64 MHz)
Fmax = Fcr*DucUpRate*OverClock;

% over-clocking rate
MaxOverClock = Fmax/Fcr;

% latency relative to MATLAB model
FiltersLatency = 86;

% startup latency
DDSStartup = 10;

% SFDR
SFDR = 105;

% remove the path that includes simulation scripts and sub-functions
rmpath ../../model
```

Figure 95: A Sample Initialization File for WCDMA DUC Reference Design

The user can explore the WCDMA reference design by using the following instructions.

Simulate the Design

For the DUC design, open the model `duc_umts_virtex5_v1_0.mdl` (for Virtex-5 FPGA) or `duc_umts_sp3a_v1_0.mdl` (for Spartan-DSP FPGA) inside the System Generator environment.

For the DDC design, open the model `ddc_umts_virtex5_v1_0.mdl` (for Virtex-5 FPGA) or `ddc_umts_sp3a_v1_0.mdl` (for Spartan-DSP FPGA) in the System Generator environment. Press the **Start simulation** button.

Note: When simulating the design for the first time, it takes a while to initialize the various blocks. Thereafter, simulations are much faster because only the System Generator blocks which have been updated are compiled.

Verify the Design

In MATLAB, run the model `duc_umts_post.m` (for Virtex-5 FPGA) or `duc_umts_sp3a_post.m` (for Spartan-DSP FPGA) for the DUC design. Or, run the model `ddc_umts_post.m` (for Virtex-5 FPGA) or `ddc_umts_sp3a_post.m` (for Spartan-DSP FPGA) for the DDC design. This simulates the DUC/DDC and compare the results to those from the System Generator simulation. Statistics and an overlay plot are shown to verify a match.

Generate HDL

Open the System Generator model, double-click on the System Generator token, and press the **Generate** button. This creates HDL (and some NGC core netlists) in the `netlist` directory.

Place and Route

After Generating the HDL, open the Xilinx ISE project file (`*.ise`) under the netlist directory. This brings up the ISE project navigator. From here, the user can synthesize, place and route the design, and create a bit file for loading into a Xilinx FPGA.

Appendix D: Hardware Co- Simulation and Incorporate HDL models into System Generator

The hardware co-simulation was used to verify all the reference designs described in this document. Co-simulating with hardware in the loop provides faster simulation and is an effective way to perform design verification on the real hardware. System Generator uses a special co-simulation block to control the design hardware during Simulink® simulations. The block's ports have names and types that match the ports on the original System Generator subsystem. The simulation behavior of the co-simulation block is bit- and cycle- accurate when compared to the behavior of the original subsystem. This feature enables significant simulation speedups, provides incremental hardware verification capabilities, and removes many of the hurdles to getting a design up and running in an FPGA. Refer to *Hardware Co-simulation* section in [Ref 10] for more details.

While it is simply a push-button flow to perform a hardware co-simulation for the design developed in the System Generator flow, it is equally straightforward to incorporate hardware description language (HDL) models into System Generator—the blackbox block found in basic elements in Xilinx Blockset is the agent to perform this function. The block is used to specify both the simulation behavior in Simulink and the implementation files to be used during code generation with System Generator. A black box's ports produce and consume the same sorts of signals as other System Generator blocks. When a black box is translated into hardware, the associated HDL entity is automatically incorporated and wired to other blocks in the resulting design.

The black box can be used to incorporate either VHDL or Verilog into a Simulink model. Black box HDL can be co-simulated with Simulink using the System Generator interface to either ISE simulator or the ModelSim® simulation software from Model Technology, Inc. The WCDMAHDL reference design package includes a System Generator Blackbox example to demonstrate the capability to perform hardware co-simulation in the System Generator framework. The help page of the Black Box and the Importing HDL Modules in [Ref 9] describes the requirements and procedures in detail.

Table 47 and Table 48 list the WCDMA reference design files using the VHDL for the DUC and DDC, respectively, and then verifies through the black box flow in the System Generator. These files are located under `\DUC\Implementation` directory.

Table 47: WCDMA DUC Design Files using VHDL and System Generator Black Box

Directory	File Name	Descriptions
VHDL\Cores	Contains all the cores generated through Core Generator	
VHDL\Modelsim	ModelSim compilation directory – this directory is automatically generated when blackbox co-simulation is used	
VHDL	duc_umts_rtl_bb.mdl	The blackbox model that contains the HDL entity in System Generator
	duc_umts_bb_init.m	A MATLAB initialization file (called by duc_umts_rtl_bb.mdl before simulation)
	duc_umts_post.m	Post simulation script
	duc_umts_v5_bb_wrapper_config.m	Configuration file executed in the System Generator blackbox flow
	duc_umts_v5.vhd	Top level VHDL file
	duc_umts_v5_bb_wrapper.vhd	A black box wrapper file that instantiate duc_umts_v5.vhd and create the appropriate interface for the System Generator flow
	complex_mult_v5.vhd	VHDL source code for various functions used in the DUC design
	freq_translate.vhd	
	mac_v5.vhd	
	mixer_v5.vhd	

Table 48: WCDMA DDC Design Files using VHDL and System Generator Black Box

Directory	File Name	Descriptions
VHDL\Cores	Contains all the cores generated through Core Generator	
VHDL\Modelsim	ModelSim compilation directory – this directory is automatically generated when blackbox co-simulation is used	
VHDL	ddc_umts_rtl_bb.mdl	The blackbox model that contains the HDL entity in System Generator
	ddc_umts_bb_init.m	A MATLAB initialization file (called by ddc_umts_rtl_bb.mdl before simulation)
	ddc_umts_post.m	Post simulation script
	ddc_umts_v5_bb_wrapper_config.m	Configuration file executed in the System Generator blackbox flow
	ddc_umts_v5.vhd	Top level VHDL file
	ddc_umts_v5_bb_wrapper.vhd	A black box wrapper file that instantiate ddc_umts_v5.vhd and create the appropriate interface for the System Generator flow
	complex_mult_v5.vhd	VHDL source code for various functions used in the DDC design
	freq_translate.vhd	
	mac_v5.vhd	
	mixer_v5.vhd	

Appendix E: Description of CDMA2000 Reference Design Files

This section describes the files that are included in the CDMA2000 reference design archive. The CDMA2000 reference design zip file contains four directories. The `V5_DUC` and `S3A_DUC` directories contain the System Generator source for the DUC targeting the Virtex-5 FPGA and Spartan-DSP FPGA, respectively. The `V5_DDC` and `S3A_DDC` directories contain the System Generator source for the DDC targeting the Virtex-5 FPGA and Spartan-DSP FPGA, respectively. A MATLAB model is included in all directories for verification purposes. [Table 49](#) lists the files in each directory and their descriptions.

Table 49: CDMA2000 Files and Descriptions

Directory	File Name	Descriptions
V5_DUC and S3A_DUC	<code>cdma_duc.mdl</code>	System Generator source
	<code>cdma_duc_model.m</code>	MATLAB model used for verification
	<code>cdma_duc_filter.m</code>	MATLAB code used to generate filter coefficients
	<code>rand_binary.p</code>	Function used to generate quasi-Gaussian input
	<code>scale_coeffs.m</code>	Used in the filter coefficients generation process
	<code>imf1_update.m</code>	
	<code>imf2_update.m</code>	
	<code>channel_fir.m</code>	Script to calculate the DUC performance metrics
	<code>performance_metrics.p</code>	
	<code>plot_psd_cdma_duc_mask.p</code>	
V5_DDC and S3_DDC	<code>cdma_ddc.mdl</code>	System Generator source
	<code>cdma_ddc_model.m</code>	MATLAB model used for verification
	<code>cdma_ddc_filter.m</code>	MATLAB code used to generate filter coefficients
	<code>rand_binary.p</code>	Function used to generate quasi-Gaussian input
	<code>scale_coeffs.m</code>	Used in the filter coefficients generation process
	<code>dec1.m</code>	
	<code>dec2.m</code>	
	<code>channel_fir.m</code>	Matrix with output from System Generator DUC simulation
	<code>composite_iq.mat</code>	
	<code>ddc_performance_metrics.p</code>	Script to calculate DDC performance metrics

The user can explore the CDMA2000 reference design by using the following instructions.

Simulate the Design

For the DUC design, open the model `cdma_duc.mdl` inside the System Generator environment. For the DDC design, open the model `cdma_ddc.mdl` in the System Generator environment. Press the **start simulation** button. At the end of the simulation, the output data is written to a file so it may be compared to the MATLAB model.

Note: When simulating the design for the first time, it takes a while to initialize the various blocks. Thereafter, simulations are much faster because only the System Generator blocks which have been updated are compiled.

Verify the Design

In MATLAB, run the model `cdma_duc_model.m` for the DUC design. Or, run the model `cdma_ddc_model.m` for the DDC design. This simulates the DUC/DDC and compares the results to those from the System Generator simulation. Statistics and an overlay plot are shown to verify a match.

Generate HDL

Open the System Generator model, double-click on the System Generator token, and press the **Generate** button. This creates HDL (and some NGC core netlists) in the netlist directory.

Place and Route

After Generating the HDL, open the Xilinx ISE project file (*.ise) under netlist directory, that is, `cdma_duc_cw.ise` for the DUC, `cdma_ddc_cw.ise` for the DDC. This brings up the ISE project navigator. From here, the user can synthesize, place and route the design, and create a bit file for loading into a Xilinx FPGA.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/22/07	1.0	Initial Xilinx release.

Notice of Disclaimer

Xilinx is disclosing this Application Note to you "AS-IS" with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.