

Agilent Technologies
Advanced Design System 2011.01
February 2011
Guide to Digital Predistortion

© Agilent Technologies, Inc. 2000-2011

5301 Stevens Creek Blvd., Santa Clara, CA 95052 USA

No part of this documentation may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Acknowledgments

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Mentor products and processes are registered trademarks of Mentor Graphics Corporation. * Calibre is a trademark of Mentor Graphics Corporation in the US and other countries. "Microsoft®, Windows®, MS Windows®, Windows NT®, Windows 2000® and Windows Internet Explorer® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Oracle and Java and registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC. FLEXIm is a trademark of Globetrotter Software, Incorporated. Layout Boolean Engine by Klaas Holwerda, v1.7 <http://www.xs4all.nl/~kholwerd/bool.html> . FreeType Project, Copyright (c) 1996-1999 by David Turner, Robert Wilhelm, and Werner Lemberg. QuestAgent search engine (c) 2000-2002, JObjects. Motif is a trademark of the Open Software Foundation. Netscape is a trademark of Netscape Communications Corporation. Netscape Portable Runtime (NSPR), Copyright (c) 1998-2003 The Mozilla Organization. A copy of the Mozilla Public License is at <http://www.mozilla.org/MPL/> . FFTW, The Fastest Fourier Transform in the West, Copyright (c) 1997-1999 Massachusetts Institute of Technology. All rights reserved.

The following third-party libraries are used by the NlogN Momentum solver:

"This program includes Metis 4.0, Copyright © 1998, Regents of the University of Minnesota", <http://www.cs.umn.edu/~metis> , METIS was written by George Karypis (karypis@cs.umn.edu).

Intel@ Math Kernel Library, <http://www.intel.com/software/products/mkl>

SuperLU_MT version 2.0 - Copyright © 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy). All rights reserved. SuperLU Disclaimer: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7-zip - 7-Zip Copyright: Copyright (C) 1999-2009 Igor Pavlov. Licenses for files are: 7z.dll: GNU LGPL + unRAR restriction, All other files: GNU LGPL. 7-zip License: This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. unRAR copyright: The decompression engine for RAR archives was developed using source code of unRAR program. All copyrights to original unRAR code are owned by Alexander Roshal. unRAR License: The unRAR sources cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified unRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver. 7-zip Availability: <http://www.7-zip.org/>

AMD Version 2.2 - AMD Notice: The AMD code was modified. Used by permission. AMD copyright: AMD Version 2.2, Copyright © 2007 by Timothy A. Davis, Patrick R. Amestoy, and Iain S. Duff. All Rights Reserved. AMD License: Your use or distribution of AMD or any modified version of AMD implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. AMD Availability: <http://www.cise.ufl.edu/research/sparse/amd>

UMFPACK 5.0.2 - UMFPACK Notice: The UMFPACK code was modified. Used by permission. UMFPACK Copyright: UMFPACK Copyright © 1995-2006 by Timothy A. Davis. All Rights Reserved. UMFPACK License: Your use or distribution of UMFPACK or any modified version of UMFPACK implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St,

Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. UMFPACK Availability: <http://www.cise.ufl.edu/research/sparse/umfpack> UMFPACK (including versions 2.2.1 and earlier, in FORTRAN) is available at <http://www.cise.ufl.edu/research/sparse> . MA38 is available in the Harwell Subroutine Library. This version of UMFPACK includes a modified form of COLAMD Version 2.0, originally released on Jan. 31, 2000, also available at <http://www.cise.ufl.edu/research/sparse> . COLAMD V2.0 is also incorporated as a built-in function in MATLAB version 6.1, by The MathWorks, Inc. <http://www.mathworks.com> . COLAMD V1.0 appears as a column-preordering in SuperLU (SuperLU is available at <http://www.netlib.org>). UMFPACK v4.0 is a built-in routine in MATLAB 6.5. UMFPACK v4.3 is a built-in routine in MATLAB 7.1.

Qt Version 4.6.3 - Qt Notice: The Qt code was modified. Used by permission. Qt copyright: Qt Version 4.6.3, Copyright (c) 2010 by Nokia Corporation. All Rights Reserved. Qt License: Your use or distribution of Qt or any modified version of Qt implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. Qt Availability: <http://www.qtsoftware.com/downloads> Patches Applied to Qt can be found in the installation at: `$HPEESOF_DIR/prod/licenses/thirdparty/qt/patches`. You may also contact Brian Buchanan at Agilent Inc. at brian_buchanan@agilent.com for more information.

The HiSIM_HV source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code, is owned by Hiroshima University and/or STARC.

Errata The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

Warranty The material contained in this document is provided "as is", and is subject to

being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this documentation and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license. Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at <http://systemc.org/> . This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

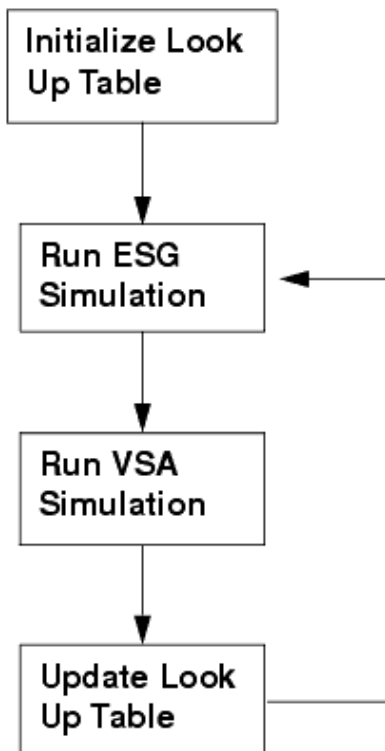
Restricted Rights Legend U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

Digital Predistortion Using Test Equipment	7
Step 1: Open the DesignGuide	7
Step 2: Initialize Look Up Table Coefficients	8
Step 3: Run ESG Simulation	9
Step 4: Run VSA Simulation	10
Time Averaging	10
Step 5: Update Coefficients	11
Step 6: Re-Run ESG Simulation	12
ESG and VSA Simulations	14
Hardware Set-up	14
Templates	31
Theory of Operation for Digital Predistortion	33
Digital Predistortion System	36

Digital Predistortion Using Test Equipment

Digital predistortion linearizes the non-linear response of a power amplifier over an operating region. It uses digital signal processing techniques to condition a baseband signal prior to modulation, up-conversion, and amplification by the power amplifier.

Digital Predistortion within Advanced Design System uses a series of system simulations and system sub-networks developed and implemented to evaluate the performance of digital predistortion. The implemented designs contain sufficient flexibility such that a variety of system sub-networks and a large number of system parameters can be readily changed so that the impact of those changes on system performance can be evaluated. The system simulations and sub-networks include the ability to interface the digital predistortion algorithm with an electronic signal generator (ESG) and a vector signal analyzer (VSA) so that the performance of the digital predistortion system can be evaluated using an actual amplifier. The following illustration summarizes the design flow for digital predistortion within ADS.



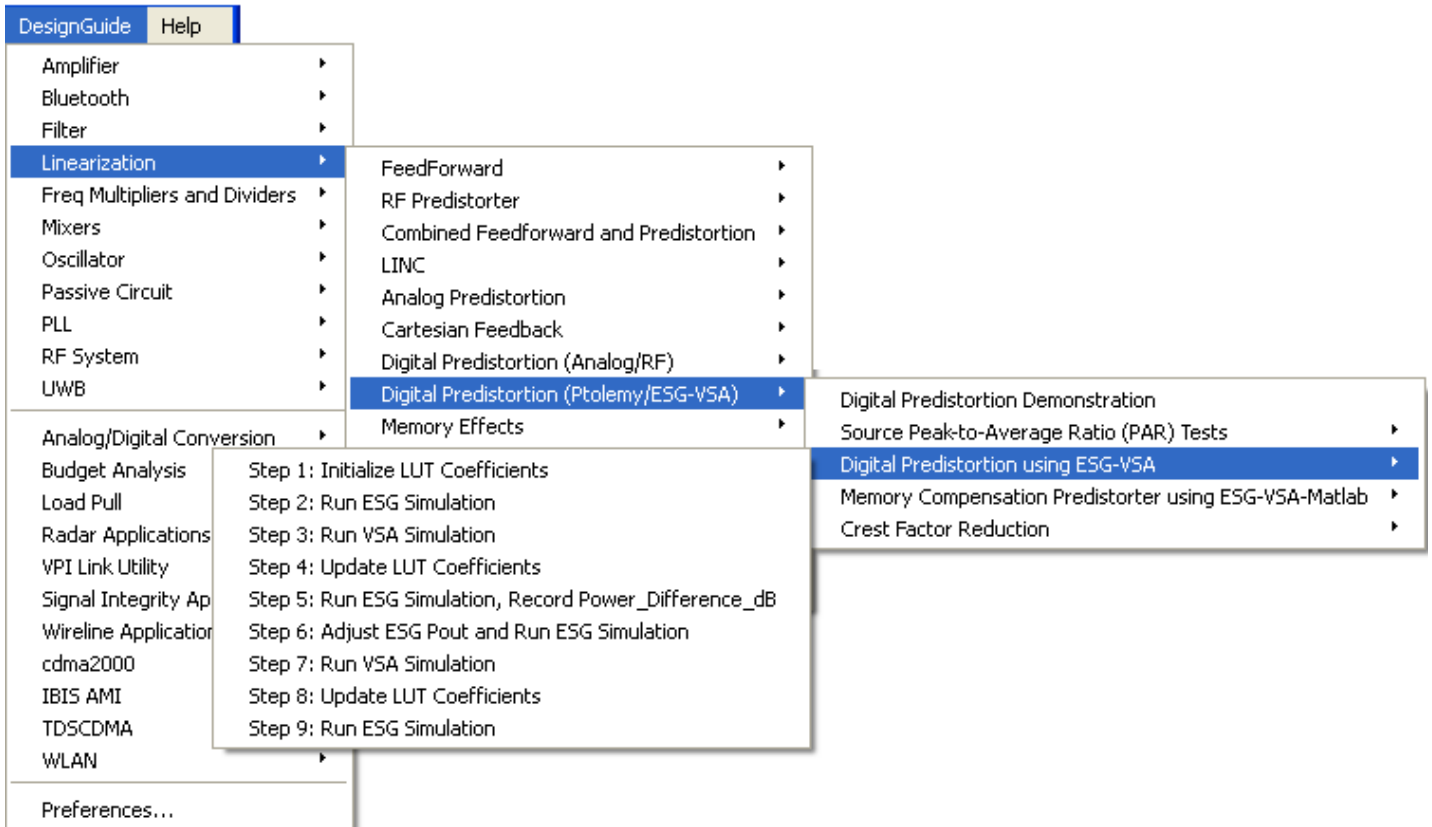
This process typically requires two iterations. You may need to perform the simulations until the desired digital predistortion improvement is obtained. After three iterations, sufficient linearization is usually obtained. Once the look-up table coefficients are finalized, adjust ESG power to obtain the desired output power and perform final measurements.

Step 1: Open the DesignGuide

The Digital Predistortion DesignGuide provides step-by-step instructions and templates to

create the predistorted versions of the desired modulation.

To use this DesignGuide, open a schematic window and choose **DesignGuide > Linearization > Digital Predistortion** .



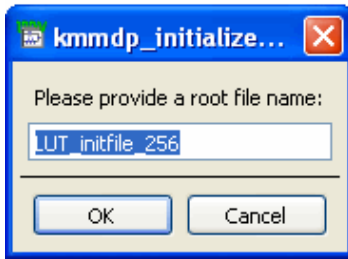
Note

Several of the signal sources provided in the DesignGuide require ADS Design Libraries. The cdma2000 sources require the cdma2000 Design Library, and the WCDMA sources require the 3GPP (3rd Generation Partnership Project) Design Library. The source used for the Digital Predistortion Demonstration is derived from a saved dataset, and does not require a Design Library. The source used in the test bench *Test_PARofModSig_OQPSK* also does not require a Design Library.

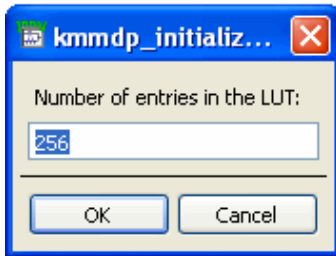
Step 2: Initialize Look Up Table Coefficients

Specify the number of entries desired in the Look-up table of coefficients.

1. Choose **DesignGuide > Linearization > Digital Predistortion (Ptolemy/ESG-VSA) > Digital Predistortion Using ESG-VSA > Step 1. Initialize LUT Coefficients** to display a dialog box pre-loaded with the filename *LUT_initfile_256*. This file supports a Look-up table with a size of 256 entries; this size is recommended for most applications.



- Click **OK** to display another dialog box where you can enter the number of Look-up table entries desired. A default value of 256 is entered. Click **OK** to reset the number of look-up entries to $1 + j \times 0.0$.

**Note**

For digital predistortion using ESG-VSA, begin with *Step 3: Run ESG Simulation* to first copy the Look-up table data files and ESG simulation template to your local workspace. Once this is done, continue with *Step 2: Initialize Look Up Table Coefficients*.

Step 3: Run ESG Simulation

Copy the ESG simulation template to your workspace.

- Choose **DesignGuide > Linearization > Digital Predistortion (Ptolemy/ESG-VSA) > Digital Predistortion Using ESG-VSA > Step 2. Run ESG Simulation** to display the ESG simulation design and data display, which has two pages, Main and LUT signals.
- Save these files to your Workspace. The Main data display page contains the *Power_Difference_dB* function. This value will later be entered into the VAR expression *ESGIndParameters* for *ESGAmplitude*. The test frequency, *ESGCarrierFrequency*, should also be set at this time, in units of Hz.
- Running the ESG Simulation will set the ESG output power, download the signal, trigger the VSA, and enable the RF output of the ESG. Refer to the *ESG_E4438C_Sink* (instruments) for more information on setting the interface parameters.



VAR

ESGIndParameters

ESGCarrierFrequency= 850e6

ESGAmplitude_dBm= -1.0

Step 4: Run VSA Simulation

Copy the VSA simulation template to your Workspace.

1. Choose **DesignGuide > Linearization > Digital Predistortion (Ptolemy/ESG-VSA) > Digital Predistortion Using ESG-VSA > Step 3. Run ESG Simulation and Record Response** to display the VSA simulation design and data display. When you save these files to your Workspace, the VSA setup file is also copied to your Workspace /data directory.
2. Start the VSA manually with the VSA setup file selected using **File > Recall > Recall Setup**.
3. Modify the center frequency using *MeasSetup > Frequency* as required, then re-save the file. Close the VSA window at this time; it will be re-opened when the simulation is run. Selecting this menu item will also open the associated display window, which has five pages.

When the VSA simulation is run, certain data from the ESG simulation is directly passed to the VSA simulation. In addition, the VSA Source downloads the DUT's output signal. At this point, the VSA may be manually set to run mode and the time domain waveform examined. The peak value of the ramp (V_{peak}) will be dependent upon the gain of the DUT, because the ESG signal is normalized to a magnitude of 1. Due to this gain, the value of *VSAScaleFactor* in the *VSAIndParameters* VAR component should be set to $1/V_{peak}$. For example, if V_{peak} is 3.07V, set *VSAScaleFactor* to $1/3.07$ or 0.326. This value will not need to be changed for subsequent simulations.

Once *VSAScaleFactor* has been set, run the VSA simulation again to obtain the first iteration of LUT coefficients.

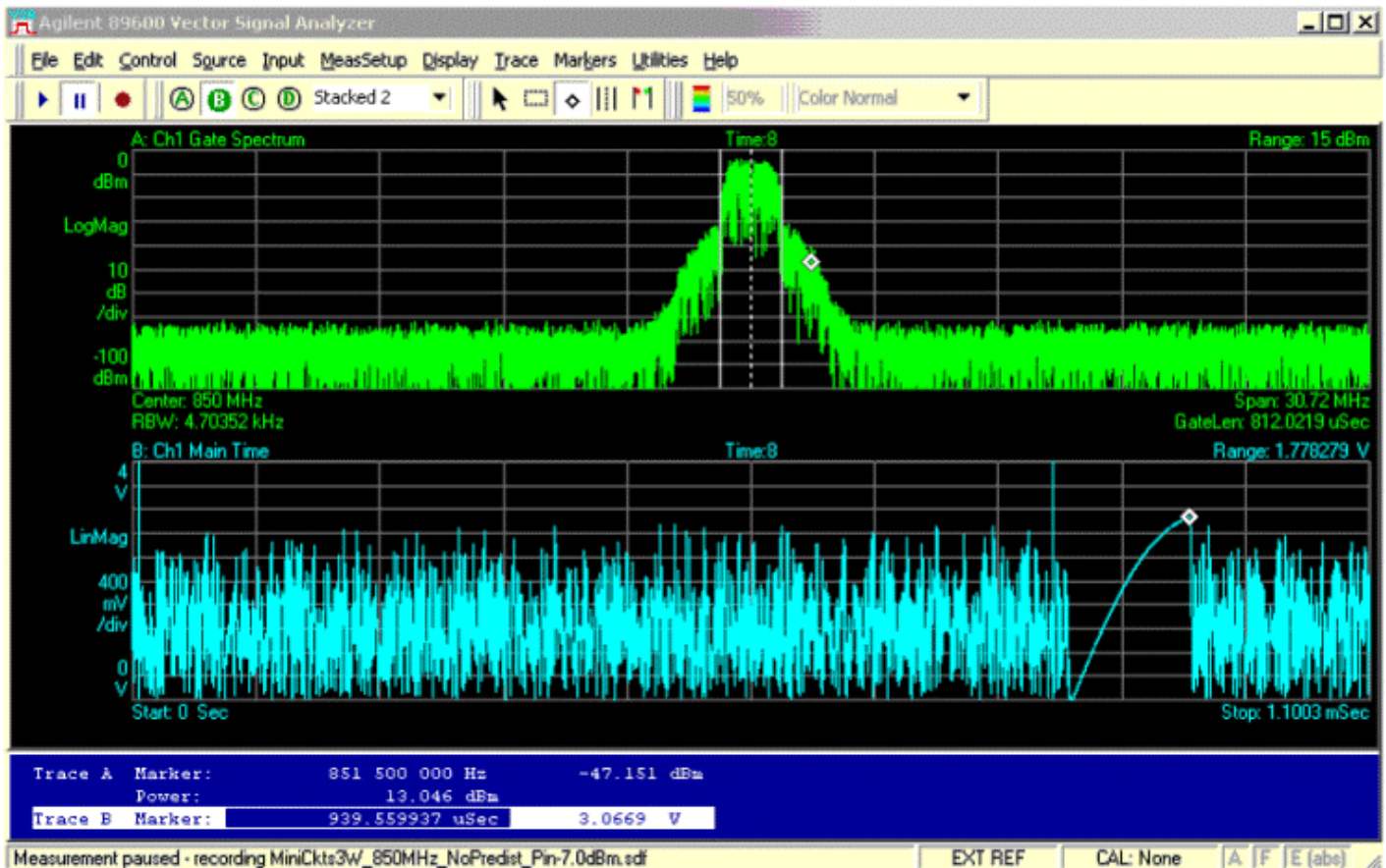
Time Averaging

Use time averaging to improve the dynamic range performance of the VSA. The included setup file enables averaging, as follows:

- Averaged measurements are not handled as a special case by the *VSA_89600_Source* component
- Each 89600 analyzer trace update is output to the simulation

The default 89600 analyzer average setup has "Fast Average" disabled, so the traces are updated each time new measurement results are added to the average, starting with the first measurement.

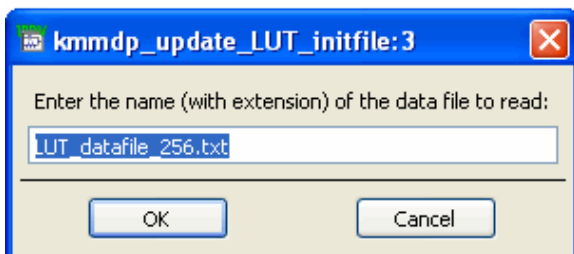
To force only the final, fully averaged result to be output to the simulation, choose *MeasSetup > Average* on the 89600 analyzer and check both the "Fast Average" and the "Same as Count" options. This will prevent the analyzer from updating the screen until the selected number of measurements have been averaged together.



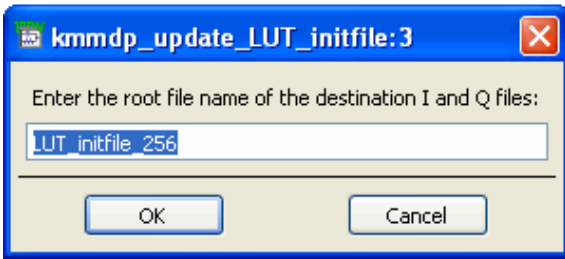
Step 5: Update Coefficients

Update the number of entries desired in the Look-up table of coefficients.

Choose **DesignGuide > Linearization > Digital Predistortion (Ptolemy/ESG-VSA) > Digital Predistortion Using ESG-VSA > Step 4. Update LUT Coefficients** to display a dialog box pre-loaded with the filename LUT_datafile_256.txt.



This file will be processed to obtain the I and Q files in the correct format for use by the look-up table. Click **OK** to display another dialog box where you can enter the number of Look-up table entries desired.



Step 6: Re-Run ESG Simulation

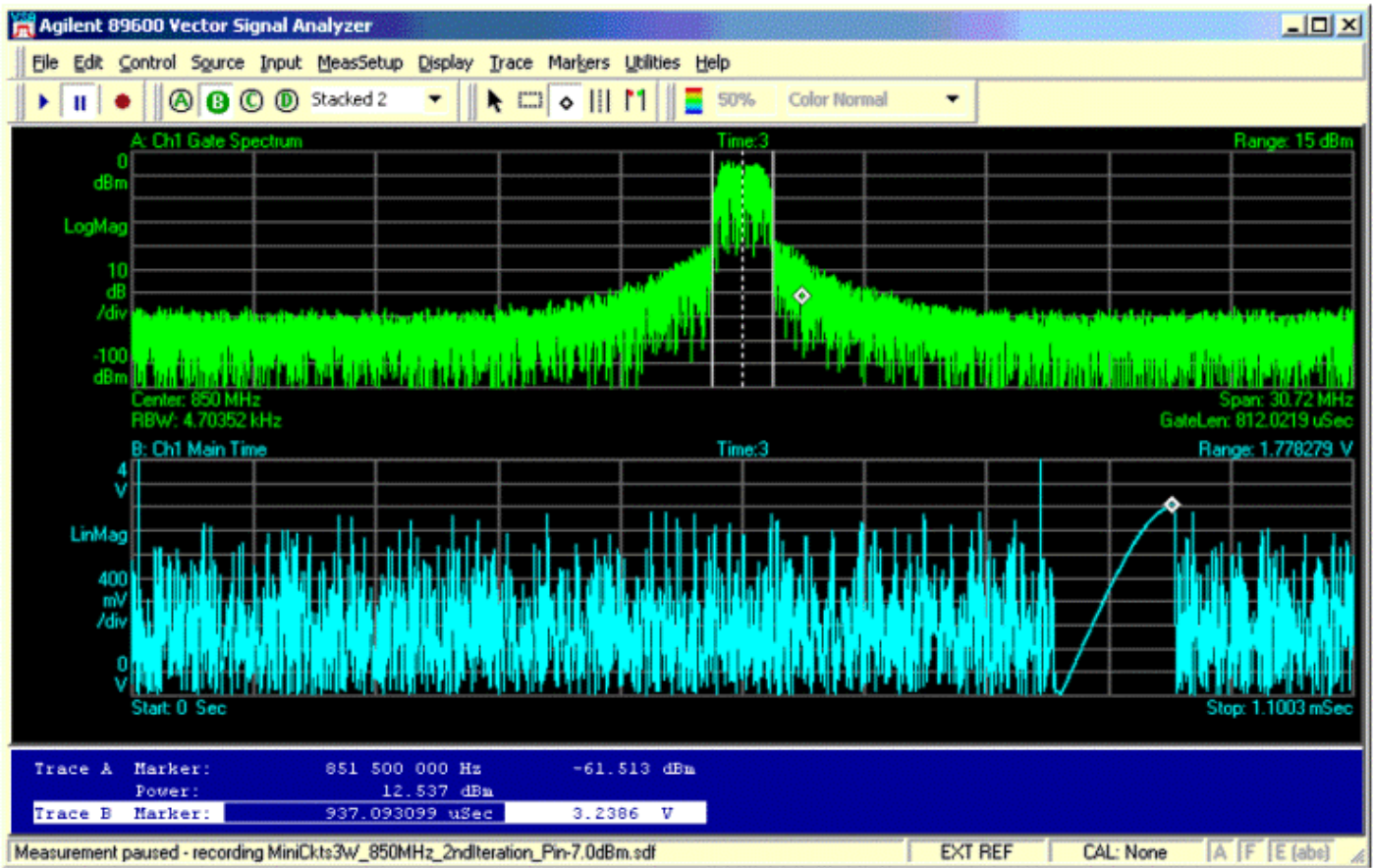
Re-run the ESG simulation, which will now use the look-up table coefficients obtained during the first iteration of Digital Predistortion. When the data display is shown, note the value of *Power_Difference_dB* on the Main page.

ComplexAdjustInMag_rms	ComplexAdjustInPower_dB
365.7847m	-8.735488

ComplexAdjustOutMag_rms	ComplexAdjustOutPower_dB
253.9457m	-11.90518

Power_Difference_dB
-3.169695

Add the value of *Power_Difference_dB* into the VAR expression *ESGIndParameters* for *ESGAmplitude* such that *ESGAmplitude* is adjusted by the value of *Power_Difference_dB*. It is necessary to compensate for gain expansion or compression due to the look-up table by offsetting the ESG output power by this amount. Run the ESG simulation again to drive the DUT at the corrected power level. Repeat this for each set of look-up table coefficients.



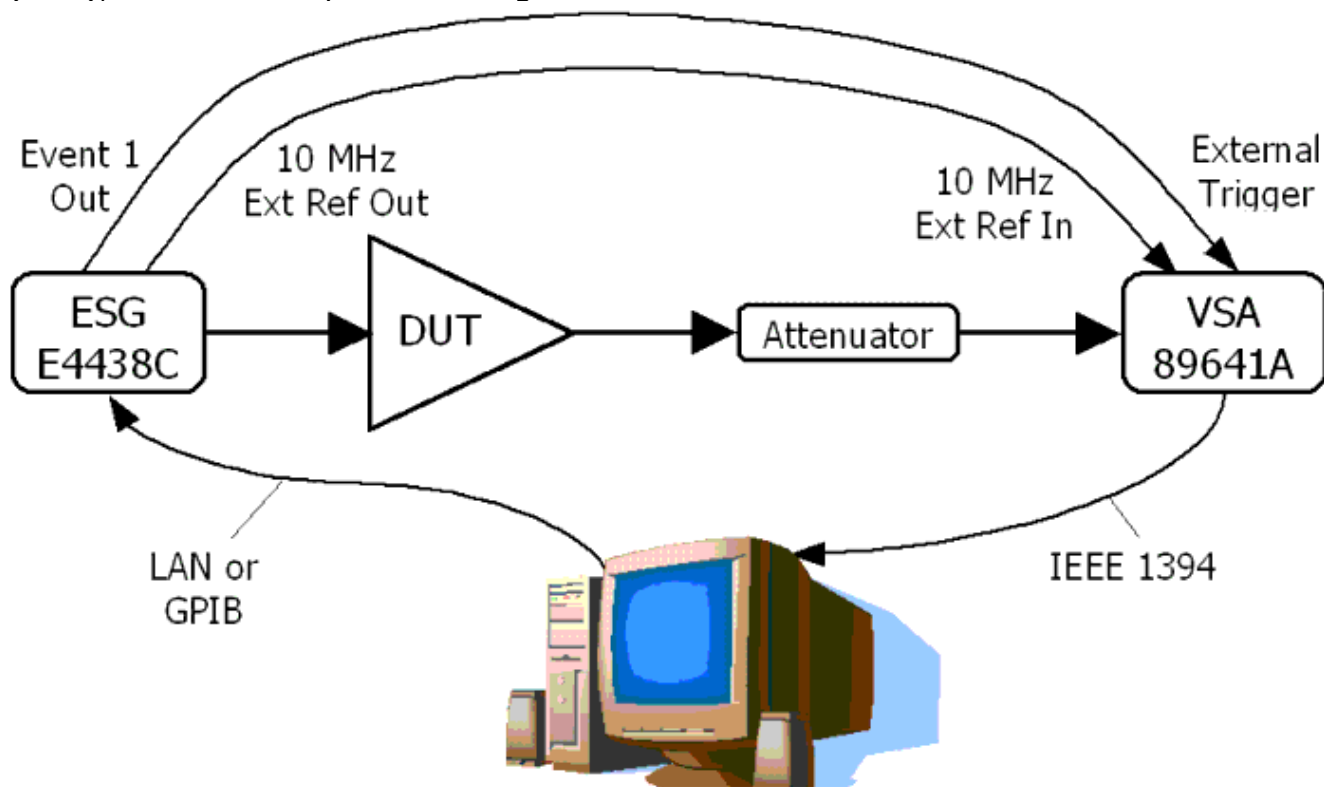
ESG and VSA Simulations

The simulations in the *DigitalPredistortion_v5p0_ESGSink* ("ESG simulation") and *DigitalPredistortion_v5p0_VSASource* ("VSA simulation") design files define a pair of simulations that complement one another. These simulations are intended to be executed repeatedly one after the other, with the ESG simulation being executed first.

The ESG simulation generates and predistorts a modulated signal, passing it on to an ESG where the signal is output on an RF carrier. The RF output of the ESG is fed to a power amplifier that is connected through an attenuator to the VSA. The VSA simulation is then run to capture the downconverted received signal, bring that signal into the ADS domain and perform signal processing on it to update the set of coefficients used to predistort the signal in the ESG simulation.

Hardware Set-up

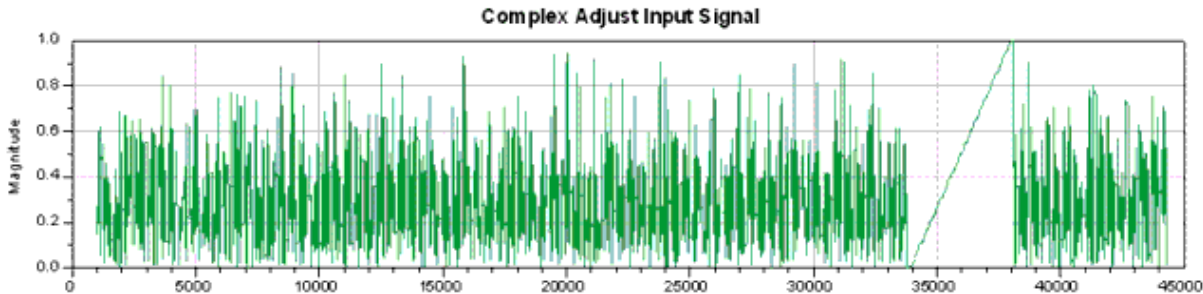
The following figure illustrates the set-up for interconnecting the ESG, amplifier under test (DUT), VSA and computer running ADS.



Besides the RF output of the ESG, two other outputs of the ESG are employed in the set-up. The ESG's Event 1 trigger and 10 MHz External Reference Out, both located on the back panel of the ESG, are connected to the VSA's External Trigger and 10 MHz External Reference input, respectively. It is also acceptable to connect the VSA's 10 MHz External Reference Out to the ESG's 10 MHz External Reference input, if preferred.

ESG Simulation

The *DigitalPredistortion_v5p0_ESGSink* simulation generates a complex baseband signal, predistorts that signal, and downloads it to the arbitrary waveform generators of an ESG. The complex baseband signal is a modulated signal that is gated off temporarily while a training signal is inserted into the signal path. The following plot shows the envelope of a typical baseband signal that is generated during the course of simulation.



Typical Complex Baseband Signal (envelope) to be Predistorted

The initial period of the modulated signal has two purposes:

1. To provide a modulated signal that can be used to monitor the predistorter's performance; and
2. As a basis for synchronizing the input signal and feedback signal prior to feeding both signals to the adaptation algorithm processing block.

The training signal is used by the adaptation algorithm to estimate the predistortion function and determine a new set of look-up table coefficients. The second period of the modulated signal is included to increase the proportion of the total modulated signal time to training signal time. The automatic level control mechanism of the ESG will operate more favorably if the total modulated signal time is greater than the training signal time.

Sample Rate and Complex Baseband Input Signal

All samples are passed between components at the top level of the simulation hierarchy at a uniform rate. This rate is equal to the `DSPRate` parameter, with the time step being equal to the `DSPTStep` parameter.

The ESG simulation sample rate should be set so that it does not exceed the maximum input sample rate of the VSA automatic level controls.

In the ESG simulation, the `DSPRate` parameter is dependent upon the parameters defining the modulation employed, specifically, the number of simulation samples per modulated symbol/chip and the symbol/chip rate of the modulation format. It is convenient to tie the simulation sampling rate to the modulation rate so that resampling of the modulated signal is not necessary.

A typical set of parameters defines the modulation signal and its format may include the

following.

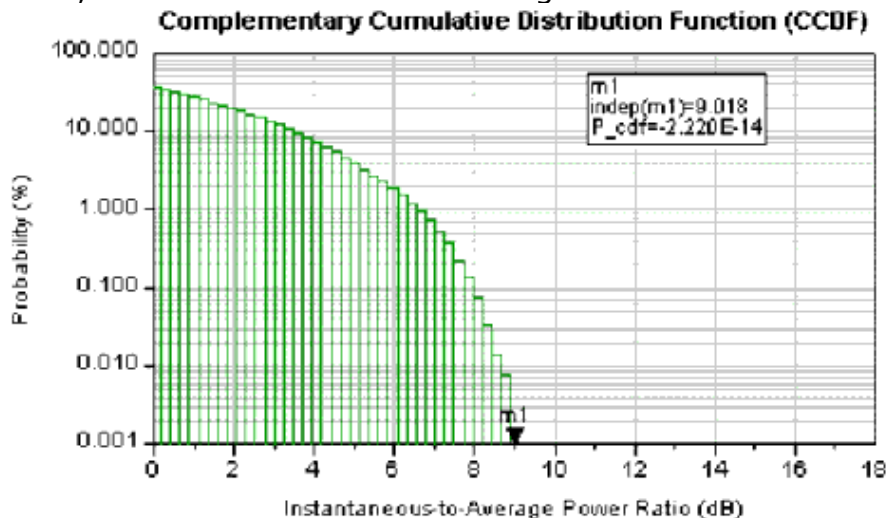
- Symbol/chip rate
- Number of samples per symbol/chip
- Length of the pulse shaping filter, as well as an excess bandwidth (roll-off factor) if a (square-root) raised cosine filter is used

In the ESG simulation that employs the *kmm_CDMA2K_FwdRCSrc* component to generate a forward link cdma2000 base station signal, typical values of the system parameters defining the signal are as follows.

- *cdma2000ChipRate* (1.2288e6)
- *cdma2000PulseShapingFilterLengthSymbols* (64)
- *cdma2000SamplesperChip* (32)
- *cdma2000XSBW* (0.25)
- *cdma2000PeakMagnitude* (0.75)

The first four parameters define those elements of the modulated signal described above. The final parameter, *cdma2000PeakMagnitude*, is the peak magnitude that the modulated signal source will produce. The peak magnitude of the modulated signal source is determined by running a separate simulation that uses the modulated signal source with the same parameter values that it will have within the ESG simulation. This separate simulation should be over a sufficient number of samples to accurately determine the peak magnitude that the signal source can generate.

An example of this can be seen in the design *Test_PARofModSig_CDMA2kFwdSrc* and its data display *Test_PARofModSig.dds*. The following displays are obtained using *Test_PARofModSig_CDMA2kFwdSrc*, and indicates the Peak Magnitude of the signal to be 0.75 V, as well as the CCDF of the signal.



Eqn $\text{PeakMagnitude} = \max(\text{mag}(\text{ComplexBBEnvelope}))$

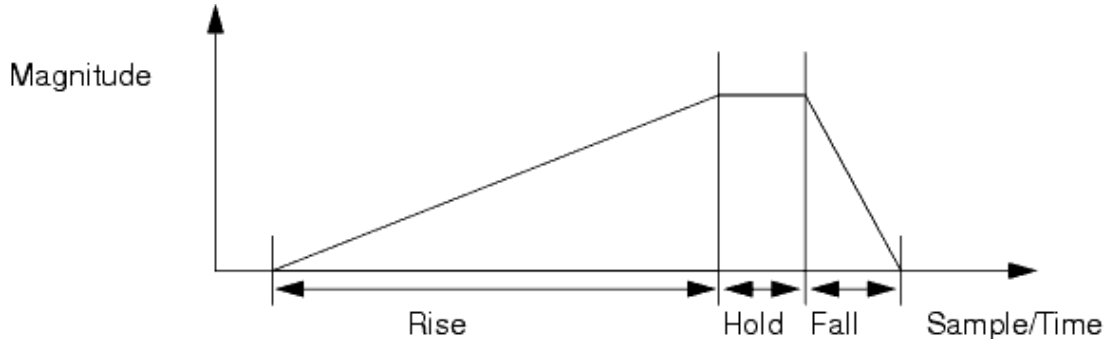
PeakMagnitude	PeakPower
753.5554m	510.5666m

CCDF and PeakMagnitude Display for Test_PAR...Test Benches

Training Signal

The training signal is a tone having an increasing magnitude, i.e. a ramp. The tone is generated by uniformly increasing the amplitude of the in-phase and quadrature component of the complex baseband signal.

In the ESG simulation, the ramp is divided into 3 segments as shown in the following illustration.



Training Signal (Ramp)

The length (in samples) of each portion of the training signal is determined by the following simulation parameters:

- RampRampLength - the length of the rising portion of the ramp
- RampRampHold - the duration for which the peak ramp magnitude is held
- RampDownRampLength - the length of the trailing ramp

Only the rising portion of the ramp is used by the adaptation algorithm in the calculation of the predistortion function. The hold and trailing ramp portions are included to smooth the transition between power levels and minimize the impact of finite length filters on the latter part of the rising portion of the ramp.

The shape of the ramp can be changed to accommodate the particular look-up table indexing scheme being employed. The shape of the ramp can be controlled through the parameters of the *Async_Ramp* component. For instance, if magnitude addressing is the index scheme being used for the look-up table, the magnitude of the ramp should increase linearly so that an equal number of simulation sample points are distributed to each look-up table entry.

The shape of the falling portion of the ramp can also be adjusted through the parameters of the *Async_Ramp* component that controls its shape.

The magnitude of the ramp at its outset is determined by the value of the RampVOffset parameter. In some cases it may be desirable that the initial magnitude of the ramp be greater than 0.0 - for instance, to avoid low level noise that may be present in the system

that is impacting system performance. If RampVOffset is set greater than zero, the user should ensure that the look-up table is being addressed over the desired range. That is, the look-up table address generator, *AddressCalc* component, does not necessarily take into account an offset magnitude on the incoming baseband signal; thus, if the initial ramp magnitude is non-zero, the lowest address generated (at the outset of the ramp) may not be 0, i.e. one or more of the initial addresses in the look-up table will not be referenced.

The point during the simulation at which the modulated signal is gated off and the training signal inserted into the baseband signal stream is set by the user using the CoefCalcStartStep parameter. The start of the ramp, however, does not occur at the simulation step defined by the CoefCalcStartStep parameter; rather, the ramp starts at the conclusion of a waiting period having a duration given by the CoefCalcDelaySteps parameter. The simulation step where the training signal is inserted should be set bearing in mind the number of samples required for reasonably accurate synchronization of the input and feedback signals in the VSA simulation. The other CoefCalcModulationOffPeriod is a dependent system parameter that defines the number of steps for which the modulated signal is gated off.

Signal Levels

The peak magnitude of the ramp is determined by the value of the RampVPeak parameter. This parameter is arbitrarily set to 1.0 and is the value at which the ramp is held during the hold period of the ramp. A more important system parameter than the peak magnitude of the ramp is the relative magnitude of the modulated signal and the ramp. Scaling the magnitude of the modulated signal so that its peak magnitude coincides with the peak magnitude of the ramp sets the relative magnitude of the two signals. The dependent system parameter, *cdma2000ScaleFactor*, in the case where the *kmm_CDMA2K_FwdRCSrc* component is employed, determines the magnitude scaling required at the output of the modulated signal source.

It may be desirable in some instances to change the relative level of the modulated signal and the ramp, however, note that the peak magnitude of the ramp is also the peak input magnitude that is corrected by the predistorter.

Address Generator, LUT and Coefficient Functions

The *AddressCalc* component translates the magnitude of the baseband input signal into a look-up table address using one of three alternative addressing schemes:

- linear (magnitude)
- power
- log power

The *AddressMode* system parameter determines which addressing scheme is employed. A scaling factor must also be passed to the *AddressCalc* component, namely, the system parameter *AddressScale*.

The look-up table is implemented by the *LUT_RAM* component. The number of entries in

the look-up table is controlled by the *LUTSize* parameter. The look-up table is initialized at the outset of the simulation using values read from a pair of text files. The text files are passed to the *LUT_RAM* component using the *LUTIInit* and *LUTQInit* parameters. The text files would typically be located in the data directory of the Workspace so a path need not be provided along with the name of the text file.

The format of the text files is described in more detail in the notes of *LUT_RAM* component .

The *LUT_RAM* initialization files can be generated automatically using two DesignGuide menu functions designed for use with the ESG and VSA digital predistortion simulations. These functions may be accessed via the Linearizer DesignGuide menu.

- **Initialize LUT Coefficients** will automatically generate the inphase and quadrature initialization files so that every LUT entry is set to $1 + j \times 0.0$.
- **Update LUT Coefficients** reads a text file (produced by the VSA simulation) containing a set of LUT entries in floating point complex notation and translates those entries into the appropriate format and writes the data to an inphase and quadrature initialization file.

Note
these functions assume that the fixed point precision being used in the look-up table is 32.32.

Before running the ESG simulation for the first time, all entries in the look-up table initialization files should be reset to the $1 + j \times 0.0$ value using *Initialize LUT Coefficients* . In other words, for the first run of the ESG simulation, the baseband signal that is generated is not predistorted, as each look-up table entry is unity. After the VSA simulation, the look-up table initialization files can be loaded with the newly calculated coefficient values from the VSA simulation using *Update LUT Coefficients*. When the ESG simulation is then executed for the second pass, the look-up table initialization files will contain the updated coefficient values and the baseband signal will be predistorted using the coefficient values.

ESG Interface and Simulation Length

The complex baseband modulated signal, with the inserted training signal, is predistorted in the ESG simulation prior to being loaded into the arbitrary waveform generator of the ESG. The ADS *ESG_E4438C_Sink* component is employed to link the ADS simulation environment to the ESG.

The carrier frequency of the signal output by the ESG is set using the *ESGCarrierFrequency* parameter. The portion of the predistorted baseband signal that is sent to the ESG is determined by the dependent system parameters *ESGStartDSPPoint* and *ESGStopDSPPoint*. The *ESGStartDSPPoint* parameter is dependent upon the transient period of the generated baseband signal, due to the length of the pulse shaping filter. In the case where the *kmm_CDMA2K_FwdRCSrc* component is employed, the transient period is given by the *cdma2000TransientSamps* parameter. The *ESGStopDSPPoint* parameter is set to be equal to the total number of steps in the simulation, which is defined by the user through the *SimDSPPoints* parameter. The value of the *SimDSPPoints* parameter should be set bearing in mind the number of samples required for the

following.

- Synchronization of the input and feedback signals
- Adaptation algorithm to calculate a new set of LUT coefficients
- New coefficients to be written to a text file

The output power of the ESG is controlled by the user via the `ESGAmplitude_dbm` parameter. The `ESGAmplitude_dbm` parameter is likely to be the only parameter that is required to be varied between each execution of the ESG simulation. Predistortion of the baseband signal will introduce expansion and thus will increase the signal's average power. However, if the expanded signal is downloaded to the ESG and the `ESGAmplitude_dbm` parameter is not adjusted, the expansion is lost and the absolute change in the magnitude of the baseband signal induced by predistortion will not be realized. The foregoing assumes that the scaling function of the *ESG_E4438C_Sink* component is enabled and that the ESG's automatic leveling control circuitry is able to maintain the average output power of the ESG. The automatic leveling control circuitry may not necessarily be able to maintain the output power level of the ESG and may force it into an 'unlevel' condition. In such a case the output power of the ESG must be measured using a power meter and adjusted manually. In addition, the bandwidth of the automatic leveling control loop is potentially problematic. For the ESG, the loop's bandwidth is 100 Hz; therefore, the automatic leveling control will maintain a constant relationship between the baseband signal's amplitude and the absolute output amplitude of the ESG over the course of the ramp as long as the ramp is sufficiently short in relation to the total duration of the modulated signal.

The value of the `ESGAmplitude_dbm` parameter is initially set to an appropriate value given the amplifier under test and the peak input power of the PA that is correctable using predistortion. Following execution of the ESG simulation the *DigitalPredistortion_v5_ESGSink* data display file can be accessed where the average power of the baseband signal sent to the ESG is calculated. The absolute magnitude of this average power is not critical, rather, its deviation from one pass of the ESG and VSA simulation pair to the next should be noted so that the `ESGAmplitude_dbm` parameter can be set appropriately.

 **Note**

After executing the VSA simulation and updating the look-up table coefficients, the ESG simulation should be run twice before starting the VSA simulation. After the first ESG simulation, the change in the average power of the baseband signal from the prior set of look-up table coefficients to the present must be taken into account in the value of the `ESGAmplitude_dbm` parameter. Thus, with the `ESGAmplitude_dbm` parameter updated the absolute output amplitude of the ESG will remain constant from one set of coefficients to the next.

A modulated signal is typically described in terms of its average power (P_{av}) and its peak to average power (PAR). The peak to average power is simply the peak power (P_{pk}) divided by the average power (P_{av}) and is often given in dB, i.e. $10 * \log (P_{pk} / P_{av})$. The ESG output level is specified in terms of P_{av} only, not in terms of P_{pk} . Furthermore, the *ESG 4438C ADS* component scales the signal being downloaded to the ESG so that the peak magnitude is equal to 1.

For a given set of look-up table coefficients (perhaps all set to 1, as at the outset) the modulated signal that is being sent to the ESG sink will be described by PAR_1 and P_{av1} . Similarly, when the look-up table coefficients change, the modulated signal's statistics change and result in PAR_2 and P_{av2} . Because the 4438C sink scales the peak magnitude of the modulated signal to 1, we can write:

$$P_{av2} \times PAR2 = P_{av1} \times PAR1.$$

If PAR1 and PAR2 are calculated, since we have set the ESG to P_{av1} , the new output power of the ESG must be $P_{av2} = P_{av1} \times PAR1 / PAR2$. If the P_{av} 's and PAR's are described in dB, P_{av2} is found by simply adding or subtracting dB values. Furthermore, the PAR of a signal does not change if it is scaled, i.e. the PAR of $\sin(t)$ is equal to the PAR of $2 * \sin(t)$, thus, the PAR of the signal that is to be downloaded to the ESG can be calculated prior to the scaling of the peak magnitude to unity.

The adjustment required to the ESG level may therefore be directly obtained from the Main page of the ESG data display, where the difference between the average power of the input and output signals is shown in the following figure.

ComplexAdjustInMag_rms	ComplexAdjustInPower_dB
365.7847m	-8.735488

ComplexAdjustOutMag_rms	ComplexAdjustOutPower_dB
253.9457m	-11.90518

Power_Difference_dB
-3.169695

Input/Output Power Difference in ESG Data Display

Triggering

The ESG's EventMarkers parameter is set to "Event1" so that the ESG will generate a trigger on its Event 1 output each time the first sample of the predistorted waveform is read and transmitted by the ESG. The ESG's Event 1 trigger output is connected to the VSA's external trigger input.

Link to VSA Simulation

Three signals from the ADS ESG simulation domain are intended to be carried forward to the ADS VSA simulation domain:

- (N4) the input signal to the complex adjuster (InputSignal)
- (N6) the predistorted signal at the output of the complex adjuster (ComplexAdjustOut)
- (N5) the integer trigger signal that gates the modulated signal off and initiates the ramp training signal (TriggerSignal)

These three signals are input to the VSA simulation by being read from the ESG simulation dataset using NumericSource components (N4-N6). All three signals are of the numeric type, with identical start and stop sample capture periods - ESGStartDSPPoint and ESGStopDSPPoint, respectively.

ESG Simulation Data Display

The Data Display file for the ESG simulation, *DigitalPredistortion_v5p0_ESGSink*.dds has two pages - Main and LUT Signals.

Main Page

The Main page shows plots of the timed input and output signal from the complex adjuster over the length of the entire simulation. This page also includes a power calculation for these two signals taken over the samples that are forwarded to the ESG and the VSA simulation. The average power of the input signal will not change between runs of the ESG simulation (provided only the ESG output power is changed with each run) and is provided for reference. The average power of the output signal from the complex adjuster will change every time the look-up table coefficients change. The difference between the average power of the input and output signals is shown on the Main page as well. This difference can be utilized by the user to set the output power of the ESG from one run to the next to ensure that the output level of the ESG is calibrated to the first run of the ESG simulation.

LUT Signals Page

The LUT Signals page contains two plots - one showing the look-up table read address and the other the output of the look-up table. These two plots can be used to check the operation of the predistorter.

VSA Source Simulation

The VSA Source simulation extracts a time record of the downconverted RF signal that is transmitted by the ESG from a properly configured VSA. The extracted time record is an input to the VSA simulation and forms the feedback signal that is used by the adaptation algorithm to calculate the open loop gain of the system and determine a new set of look-up table coefficients.

Like the ESG simulation, the VSA Source simulation relies upon a set of independent and dependent system parameters. The parameters of the VSA Source simulation can be further sub-divided into a set of parameters that are identical to those from the ESG simulation and a set of additional parameters. For simplicity, all of the parameters from the ESG simulation are copied to the VSA simulation. Although all of the ESG simulation parameters are copied, the VSA simulation does NOT utilize every one. The ESG

simulation system parameters that are employed by the VSA simulation have exactly the same meaning in the VSA simulation as they have in the ESG simulation.

The length of the VSA Source simulation depends upon the length of the ESG simulation. The VSA simulation dependent system parameter `VSASimStopDSPPoint` is set to be equal to the length of the data to be read from the ESG dataset and the length of the data record loaded into the ESG.

VSA Interface

The time record of the feedback signal is extracted from the VSA using the `VSA_89600_Source` component. The use of a timed signal ensures that the VSA's sampling rate is set appropriately when the simulation is executed and the data is extracted from the VSA. The VSA simulation employs a time step equal to that used in the ESG simulation. The `VSATStep` parameter is set to the ESG simulation time step.

Although the sampling rate of the VSA will be automatically set, other configuration settings of the VSA are set and maintained for each execution of the VSA simulation using a setup file. The name of the setup file is assigned to the `VSASetupFile` system parameter. A setup file, `VSA_Setup_One.set`, is included in the data directory of the digital predistortion workspace. The `VSA_89600_Source` will automatically look in the data directory for the setup file.

Some of the more important VSA configuration settings that are defined in the file include:

- VSA trace 'B' be set to 'MainTime'
- VSA record length be greater than the number of points to be extracted from the VSA
- VSA input is triggered by the externally supplied (ESG Event1 trigger output) trigger signal
- VSA input range be set to maximize the dynamic range of the VSA

The VSA simulation parameter, `VSAScaleFactor`, must be set by the user to appropriately scale the data read from the VSA. The `VSAScaleFactor` should be set to be equal to the inverse of the desired linear gain, k , of the predistorter and amplifier cascade. The appropriate value for `VSAScaleFactor` can be determined following the execution of the VSA simulation for the first time using the loop gain calculation and plot found on the Feedback and Gain page of the VSA Data Display file (`DigitalPredistortion_v5p0_VSASource.dds`). The plot titled "Loop Gain (uncorrected)" shows the open loop gain of the system. The loop gain is described as uncorrected because the G_{lin} factor has not yet been applied to the loop gain calculation. The portion of the plot that is of interest is that which coincides with the training ramp. The open loop gain over the length of the ramp should, ideally, be unity, however, with no predistortion present the non-linear gain of the amplifier will cause the loop gain to decrease as the power of the training signal increases. Normally, the amplifier will have a region at lower power within which it does operate in a relatively linear manner. The inverse value of the loop gain at a point within this linear operating region will determine the value of the `VSAScaleFactor` parameter. After the `VSAScaleFactor` parameter is determined the VSA simulation is calibrated and should be executed again to calculate the predistortion function coefficient values. Following this initial calibration step, the `VSAScaleFactor` parameter should not be changed over the course of the iterations of the coefficients that

follow. Note that if the range of the VSA input is changed between iterations, the `VSA_ScaleFactor` will need to be altered to reflect the change in the extracted signal magnitude.

ESG Simulation Signals

Four system parameters are employed within the VSA simulation to identify the signals that are read from the ESG simulation dataset:

- `ESGSimDataset`
- `ESGSimInputSignalVariable`
- `ESGSimTriggerSignalVariable`
- `ESGSimComplexAdjustOut`

The name of the ESG simulation dataset and the three signals from the dataset that are to be read are assigned to the four system parameters appropriately.

Note

the `ComplexAdjustOut` signal that is extracted from the ESG simulation dataset is not used by the VSA simulation. The `ComplexAdjustOut` signal is simply passed onto the VSA simulation dataset using a numeric sink. In order to prevent an error message being generated at run-time it is necessary that the signal flow from the `NumericSource` that sources the `ComplexAdjustOut` signal be tied into the signal flow graph of the simulation. The VSA simulation includes a dummy multiplier and `NumericSink` at the output of the `NumericSource` that sources the `ComplexAdjustOut` signal to ensure that an error is not generated.

Feedback Delay Estimation

The delay in the feedback path is estimated by the `Delay_Adjust_v2` sub-network. The `Delay_Adjust_v2` sub-network will estimate the delay between the input signal and the feedback signal using a specified level of granularity (interpolation). The input and output sample rate of the `Delay_Adjust_v2` sub-network is the same, thus, even though the delay between the input and feedback signals is not an integer number of sample periods, the `Delay_Adjust_v2` sub-network will delay the input signal by an integer number of sample periods, ensuring that the feedback signal is delayed an appropriate fraction of a sample period so that at the output of the sub-network the two signals are synchronized. The `Delay_Adjust_v2` sub-network generates two additional outputs that indicate the number of integer sample periods that the input signal is delayed (coarse adjust) and the number of fractional sample periods that the feedback signal is delayed (fine adjust).

Several VSA Source simulation system parameters are employed to control the synchronization of the input and feedback signals. The `FineAdjustGranularity` parameter determines the interpolation rate that the `Delay_Adjust_v2` sub-network will employ to estimate the delay between the input and feedback signals and delay the feedback signal by a fractional amount. The `Delay_Adjust_v2` sub-network employs Lagrange interpolation to increase the rate of the input and feedback signals. As the value of `FineAdjustGranularity` is raised the simulation execution time will increase.

The correlation within the `Delay_Adjust_v2` sub-network is performed on a block-by-block basis over a user defined number of lags. The `CorrelationSymbols` parameter determines

the size of the block in terms of the number of modulation symbols. The dependent system parameter, `CorrelationBlockSize`, is the number of input samples contained in the block and is dependent upon the `CorrelationSymbols` parameter and parameter determining the number of samples per modulated symbol.

The simulation execution time can be kept to a minimum by ensuring that the correlation within the `Delay_Adjust_v2` sub-network is performed over a small number of lags. The combination of the VSA Source simulation `CorrelationMaxDelay` and `DelayUnderestimate` parameters are utilized to set the maximum number of lags for the correlation calculation and to keep it to a minimum. The value of the `CorrelationMaxDelay` specifies the maximum difference in simulation steps that the `Delay_Adjust_v2` sub-network can correct for. The value of `CorrelationMaxDelay` can be kept low by setting `DelayUnderestimate` to a value that is close to, but not over, the feedback delay. The `DelayUnderestimate` value is translated into a fixed delay that is inserted into the input signal path prior to the `Delay_Adjust_v2` sub-network.

The `Delay_Adjust_v2` sub-network includes the ability to latch the estimated delay between the input and feedback signals. The `CorrelationBlockstoProcess` parameter sets the number of blocks that the sub-network will process prior to latching the estimated delay. The dependent system parameter, `CorrelationWindowLength`, translates the value of `CorrelationBlockstoProcess` into a window that is used to control the latch of the `Delay_Adjust_v2` sub-network.



Note

the `Delay_Adjust_v2` sub-network will introduce a delay of two sample periods into the input and feedback signal flows and that this delay is NOT included in the coarse adjust value at the output of the sub-network.

Gain Linearization

Immediately following synchronization of the input and feedback signals the loop gain of the system is calculated. This loop gain calculation includes scaling by the `VSAScaleFactor` and should be close to unity. Furthermore, the calculated loop gain could be used directly to determine a new set of predistortion function coefficients. Unfortunately, due to an arbitrary phase offset that is present on the samples extracted from the VSA, the coefficients determined in this manner would also suffer from an arbitrary phase offset that would change from one iteration to the next. The phase offset is not, in general, a problem, however, is easy to remove and when removed leads to a set of coefficients that are more easily compared from one iteration to the next. The arbitrary phase is removed by selecting a sample from the loop gain and using the phase of the sample to uniformly adjust the phase of all other loop gain samples. This procedure assumes that the phase offset is constant across the entire training signal period.

The VSA simulation includes a section of signal processing that attempts to locate the point during the training signal at which the loop gain is the most linear, i.e. where the slope of the loop gain is a minimum. The magnitude of the gain at the point where the loop gain is a minimum is referred to here as the linearized gain and the `Gain_Linearization` sub-network is designed to determine that point and the gain.

The `Gain_Linearization` sub-network determines the slope of the magnitude of the loop gain using the magnitude of the training signal as the independent variable. The slope is

calculated using a difference relation (i.e rise / run) between adjacent samples. Due to noise that will appear in the feedback signal path, the calculated slope is passed through a sliding window averager prior to determining the point where the slope is a minimum. The magnitude of the training signal is utilized as the independent variable in the gain linearization calculation because the training signal may not necessarily be linear in magnitude. For instance, if the training signal ramp has a quadratic shape, the change in the magnitude of the training signal from one simulation time step to the next is not uniform.

The independent system parameters of the VSA simulation that control the operation of the gain linearization processing are:

- GainLinRunGainLinearization
- GainLinPresetGainLinMagnitude
- GainLinSmoothingWindowLength

The GainLinRunGainLinearization parameter is intended to be assigned "Yes" or "No", depending upon whether the user desires that the gain linearization be performed. The default setting is "**No**". If gain linearization is NOT to be performed, the value of the GainLinPresetGainLinMagnitude parameter is utilized as the linearized gain and thus the magnitude of G_{lin} is as follows.

$$G_{lin} = \frac{\text{GainLinPresetGainLinMagnitude}}{\text{VSAScaleFactor}}$$

If GainLinPresetGainLinMagnitude is set to unity, the *VSAScaleFactor* parameter can be used to completely determine the value of G_{lin} . If gain linearization is to be performed, G_{lin} will depend upon the magnitude of the minimum gain output by the *Gain_Linearization* sub-network and the value of the *VSAScaleFactor* parameter. If gain linearization is to be performed, it should only be performed the first time that the VSA simulation is executed. After the first execution, the magnitude of the linearized gain should be noted from the Feedback and Gain page of the VSA simulation's Data Display file and entered into the GainLinPresetGainLinMagnitude parameter.

Whether gain linearization is to be performed or not, the *Gain_Linearization* sub-network is always executed to select a loop gain sample whose phase is used as the basis for the removal of the phase offset from all other loop gain samples. Hence, the GainLinSmoothingWindowLength parameter should be set to an appropriate value. The length of the smoothing window will depend upon the amount of noise in the system and the length of the training signal. Due to the presence of the smoothing window, the *Gain_Linearization* sub-network skips the initial and final GainLinSmoothingWindowLength / 2 samples of the loop gain over the course of the training signal. Therefore, the value of the GainLinSmoothingWindowLength parameter should be sufficient to reliably result in the minimum slope of the loop gain being located, but also ensure that an excessive number of samples at the start of the training signal are not skipped.

The VSA simulation includes a series of four dependent system parameters that are used to control the signal processing of the *Gain_Linearization* sub-network:

- GainLinSmoothingWindowDelaySteps
- GainLinControlWindowDelaySteps
- GainLinControlWindowLength
- GainLinWaitingSteps

These parameters are primarily concerned with maintaining the correct timing of the various signal data flows in the simulation.

Accumulator Initialization Files

Although the power of the training signal is increased smoothly over its duration, the predistortion function is quantized to the number of entries in the look-up table. In order to quantize the calculated loop gain values an accumulator is employed. The accumulator separately integrates the calculated loop gain that corresponds to each of the quantization levels of the predistortion function. The mean value of the accumulator output for each quantization level is the loop gain utilized to update the predistortion function coefficient values. In order to calculate the mean loop gain at each quantization level, the accumulator maintains a running total of the number of loop gain points integrated at each quantization level and outputs that number alongside the integrated loop gain.

The accumulator must be provided the file names of two initialization files that are used to preset the accumulator entries. The VSA simulation system parameters `AccumRAMCoefInit` and `AccumRAMNumInit` are assigned the name of a pair of initialization files for the accumulator's RAM associated with the integrated totals and the number of points integrated. The integrated totals are fixed-point values, while the number of points integrated are integer values. The files for a 256-entry accumulator are copied to the `/data` sub-directory by the DesignGuide. A series of initialization files for the fixed-point and integer entries for various accumulator sizes can be found in the `/data` sub-directory of the Linearizer DesignGuide in `HPEESOF_DIR/DesignGuides/projects`.

Note

The accumulator is reset before the start of the integration period, thus, the data transferred to the accumulator from the initialization files is obliterated before it is used. The VSA simulation can be changed so that the accumulator is not reset before execution with the data loaded into the accumulator (all zeros) at the start of the simulation determining the state of the accumulator entries at the start of the integration period.

Observe that the accumulator actually integrates the inverse of the calculated loop gain. In addition, the accumulator may be a Cartesian accumulator or a polar accumulator. The Linearizer DesignGuide includes sub-networks that implement both types of accumulators while maintaining the same set of component parameters in each case - see `Coefficient_Accumulator_RAM` (the default) and `Coefficient_Polar_Accumulator_RAM_v2`.

Convergence Control

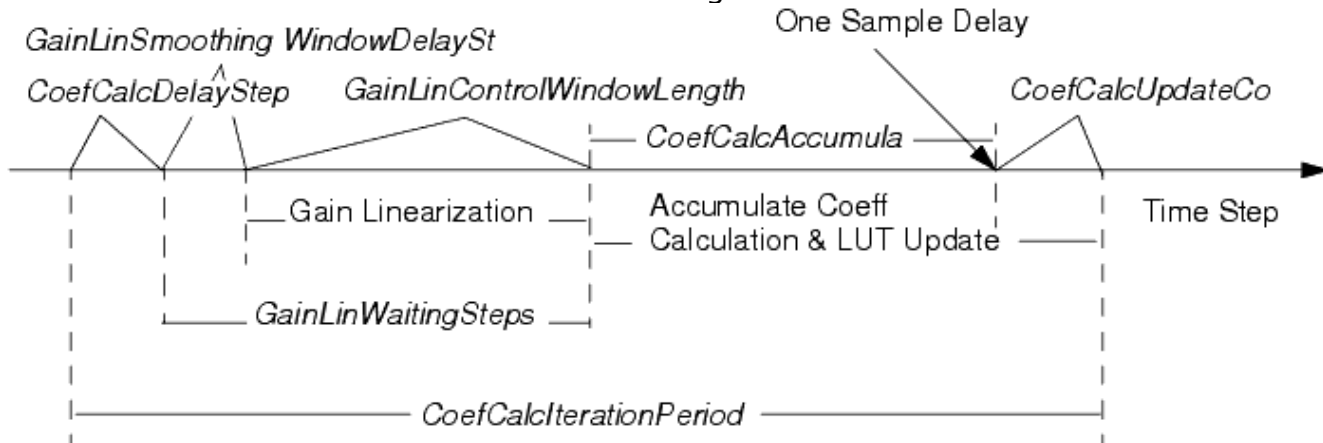
The rate at which the adaptation algorithm converges can be controlled by the `LUTMagConvFactor` and `LUTPhaseConvFactor` system parameters. The parameters may simply be left at unity, however, if a slower rate of convergence is desired the value of the parameters should be changed accordingly. The text within the `Convergence_Control` sub-network describes how the convergence control is implemented and how to set the parameters.

Coefficient Calculation and Output

The primary steps employed within the VSA Source simulation to calculate a new set of predistortion function coefficients include:

- loop-gain calculation
- determination of the linear gain and phase offset
- scaling of the loop gain by the linear gain and correction for the phase offset
- integrating (accumulating) the (scaled and phase corrected) loop gain over each entry of the quantized predistortion function
- calculation of the new predistortion function coefficients using the average integrated (scaled and phase corrected) loop gain and loading of those coefficients into the LUT

To perform these steps and maintain synchronization of the various data and control signal flows in the VSA Source simulation, a number of dependent system parameters are utilized. A number of these parameters have been discussed above. Those not discussed are those having the *CoefCalc*, *AddressWrite*, and *UpdateLUT* prefix. The relationship of a number of the VSA simulation parameters to the steps involved in the calculation of the new coefficient values is shown in the following illustration.



Relationship of Various VSA Simulation Parameters

The gain linearization (when *GainLinRunGainLinearization* = "YES"; default is "No") is performed coincident with the training signal received through the feedback path, but for a delay introduced by the smoothing window of the gain linearization signal processing block (*GainLinSmoothingWindowDelaySteps*). The waiting time from the start of the ramp to the completion of the gain linearization is given by the value of *GainLinWaitingStep*.

Following gain linearization the scaled and phase adjusted loop gain is fed to the accumulator. The accumulator is reset prior to the arrival of the loop gain data from the training ramp. The accumulator is reset over a period of time steps equal to the number of entries in the accumulator starting at the time step coinciding with the first sample of the training ramp. The length of the accumulation period is equal to the length of the training ramp and is specified by the *CoefCalcAccumulate* parameter. For the purpose of controlling the write address of the accumulator the *AddressWriteDelaySteps* and *AddressWriteLength* parameters specify the relative starting time step of the accumulation period and the length of the accumulation period.

The accumulated loop gain data output by the accumulator is averaged, used to calculate

the new predistortion function coefficients and written to the look-up table over a period of time steps equal to the size of the accumulator/look-up table plus one. The `CoefCalcUpdateCoefDelaySteps` system parameter controls the relative time step at which the process of reading and averaging the data from the accumulator begins.

The predistortion function coefficients are updated and loaded into the look-up table by multiplying the averaged output data from the accumulator with the existing coefficient data being read simultaneously from the look-up table. The inactive look-up table RAM that contains the updated coefficients is activated following the update period. The `UpdateLUTDelaySteps` system parameter specifies the relative time step at which the look-up table update signal is generated to activate the new set of coefficients. The `UpdateLUTDelaySteps` system parameter is dependent upon the `CoefCalcIterationPeriod` system parameter. The latter specifies the number of time steps in the coefficient calculation period.

Just before the end of the simulation, the new set of predistortion function coefficients are read from the look-up table and output to a text file using the *Printer* sink. The name of the text file is assigned to the `CoefCalcLUTDataFilename` system parameter, while the initial and final samples of the data to be written are given by the `CoefCalcStartStepOutputLUTValues` and `CoefCalcStopStepOutputLUTValues`, respectively.

VSA Source Simulation Data Display

The Data Display file for the VSA simulation, *DigitalPredistortion_v5p0_VSASource.dds* contains five pages - Main, Spectrums, Correlator Signals, Feedback and Gain and Coefficient Values.

Main Page

The Main page contains four plots. The two upper plots show the magnitude of the input signal from the ESG simulation and the magnitude of the scaled feedback signal extracted from the VSA over the entire duration of the simulation. The lower two plots show the magnitude, and the real and imaginary components, of the same input signal and scaled feedback signal over a block of 512 samples.

The upper two plots are useful for charting the performance of the predistorter as each iteration of the predistortion function coefficients is performed. Ultimately, the two plots should be identical if the predistorter and amplifier cascade is resulting in linear amplification.

The lower two plots provide additional insight into the relationship of the input signal and the feedback signal, specifically, the difference in their magnitudes and phase.

Spectrums Page

The Spectrums page is intended to compare the spectrum of the predistorter output signal

to the spectrum of the scaled feedback signal. The page includes time domain plots of the two signals over the duration of the entire simulation as well. The spectrums are derived from the segment of the time domain signals occurring before the insertion of the training signal. The time limits of this segment are controlled by the value of the `ki1` and `ki2` parameters. The average power of the signals over the length of the segment is also calculated and displayed. Note that the spectrums that are plotted are a smoothed version of those actually calculated.

Correlator Signals Page

The Correlator Signals page displays the `CoarseAdjustDelay` and `FineAdjustDelay` values determined during the execution of the VSA simulation. A review of these values provides the user with a check on the operation of the VSA simulation. The page also shows two plots that compare the magnitude of the delay adjusted input signal to that of the feedback signal. The upper plot compares the two signals over a 256 sample window, while the lower plot compares the two signals over a 100 sample window and over a smaller magnitude range. The plots can be used a check on the correct operation of the delay adjustment block. Prior to the delay adjustment point the signals should be offset from one another, however, after the delay adjustment they should overlap. The lower plot can also be used to view the magnitude of the noise that is present on the feedback signal.

Feedback and Gain Page

The Feedback and Gain page provides basic information from the VSA simulation that can be used to calibrate the feedback signal and also ensure that the simulation is running properly. Displayed on the page are the `CoarseAdustDelay` and `FineAdjustDelay` values, as they were on the Correlator Signals page, as well as the magnitude and phase of the linear gain that was used during simulation to scale and phase correct the calculated loop gain.

The upper two plots on the Feedback and Gain page displays the magnitude of the difference in the magnitude of the input and feedback signals after synchronization. The two plots show the error measurement over different periods of the simulation, with the upper most plot taken over the length of the training signal ramp and the lower over the entire length of the simulation. The error measurement is useful over the course of the training signal to show how effectively the predistorter is reducing the difference between the signals. A lack of synchronization or proper scaling is readily identified from the value of the error measure. The error measure taken over the length of the simulation shows the difference in the signals prior to synchronization and following synchronization. The change in the error measure following the first estimation of the feedback delay should be visible from the plot. The amount of change will depend upon the degree to which the signals are not synchronized initially and the quality of the estimation of the feedback delay from the synchronization processing block.

The bottom plot on the Feedback and Gain page shows the calculated loop gain taking into account only the `VSAScalingFactor` parameter, i.e. prior to gain linearization. This plot can be used to calibrate the VSA simulation by adjusting the `VSAScalingFactor` parameter so

that the loop gain of the linear operating region of the amplifier under test falls at the desired magnitude - normally around unity.

Coefficient Values Page

The Coefficient Values page plots the LUT read address and the output of the LUT coefficients in terms of their magnitude and phase. The 'EntryMarker' of the LUT Read Address plot is employed to extract the coefficients from the LUT output signal starting at the appropriate simulation step. At the end of the VSA simulation the LUT read address is swept sequentially through the entire LUT address range. Thus, if the data of the LUT Read Address plot is moved to show the LUT read address ramp, the 'EntryMarker' can be placed by the user on the first address (zero) of the address sweep. With the marker in place, the lower plot will display the predistortion function coefficients that were determined and stored in the LUT.

Using Other Wireless Signals and Multi-Carrier Signals

The ESG and VSA simulations are provided with an included cdma2000 single-carrier forward link signal source. Other signal sources are provided under the Source Peak-to-Average Ratio (PAR) Tests menu and may be substituted for the cdma2000 source. There are several source-related VAR items on the both the ESG and VSA simulations; these should be compared to the desired source test bench obtained using the Source Peak-to-Average Ratio (PAR) Tests menu and modified accordingly. The multi-carrier signals, having a wider bandwidth, require more samples per bit or chip than do the single-carrier signals.

To substitute another signal source, perform these steps:

1. Transfer the relevant VAR item parameters from the PAR test bench to the ESG and VSA test benches, as well as the source subnetwork
2. Run the PAR test bench and note the value of the peak magnitude (see [CCDF and PeakMagnitude Display for Test PAR...Test Benches](#)) and ensure that this value is included in the expression, e.g.:

```
cdma2000ScaleFactor= RampVPeak / cdma2000PeakMagnitude
```

where RampVPeak is typically set to 1 V.

Templates

(Please also add the top-level designs to the Reference section under the heading Digital Predistortion using ESG-VSA)

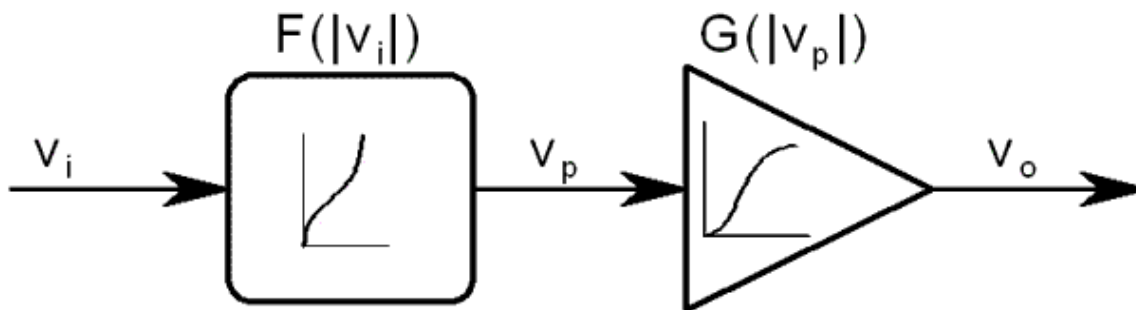
```
DigitalPredistortion_v5p0_ESGSink
AddressCalc
AddressCalc_subcomp1
```

Async_Ramp
Async_Triggers
Edge2Pulse
Edge2Pulse
kmm_CDMA2K_FwdRCsrc
LUT_RAM
DigitalPredistortion_v5p0_VSASource
AddressCalc
AddressCalc_subcomp1
Address_Generator
Async_Ramp_v2
Coefficient_Accumulator_RAM
Convergence_Control
Delay_Adjust_v2
Lagrange_Interp
Edge2Pulse
Gain_Linearization
Async_FindMin
Edge2Pulse
LUT_RAM
Magnitude_Limit
Test_PARofModSig_OQPSK
Test_PARofModSig_ADS_WCDMA
Test_PARofModSig_ADS_4xWCDMA
Test_PARofModSig_CDMA2K_FwdRCsrc
Test_PARofModSig_cdma2000_ForwardLink_Generator
Data Display Files (.dds)
DigitalPredistortion_v5p0_ESGSink.dds
DigitalPredistortion_v5p0_VSASource.dds

Theory of Operation for Digital Predistortion

The typical non-linear response of most power amplifiers is undesirable because it results in (in-band and out-of-band) distortion of the amplified signal. An amplifier with a linear response does not produce such distortion. Digital predistortion linearizes the non-linear response of a power amplifier over an operating region. It uses digital signal processing techniques to condition a baseband signal prior to modulation, up-conversion, and amplification by the power amplifier. As a result of this conditioning, the cascade of the digital predistortion response and the power amplifier response produces the desired linear response. Please refer to the *Digital Predistorter Bibliography* in the *Linearizer DesignGuide* (dglin) documentation for references associated with the following explanations.

The figure below shows a high-level representation of a signal passing through a digital predistorter and then into a power amplifier.

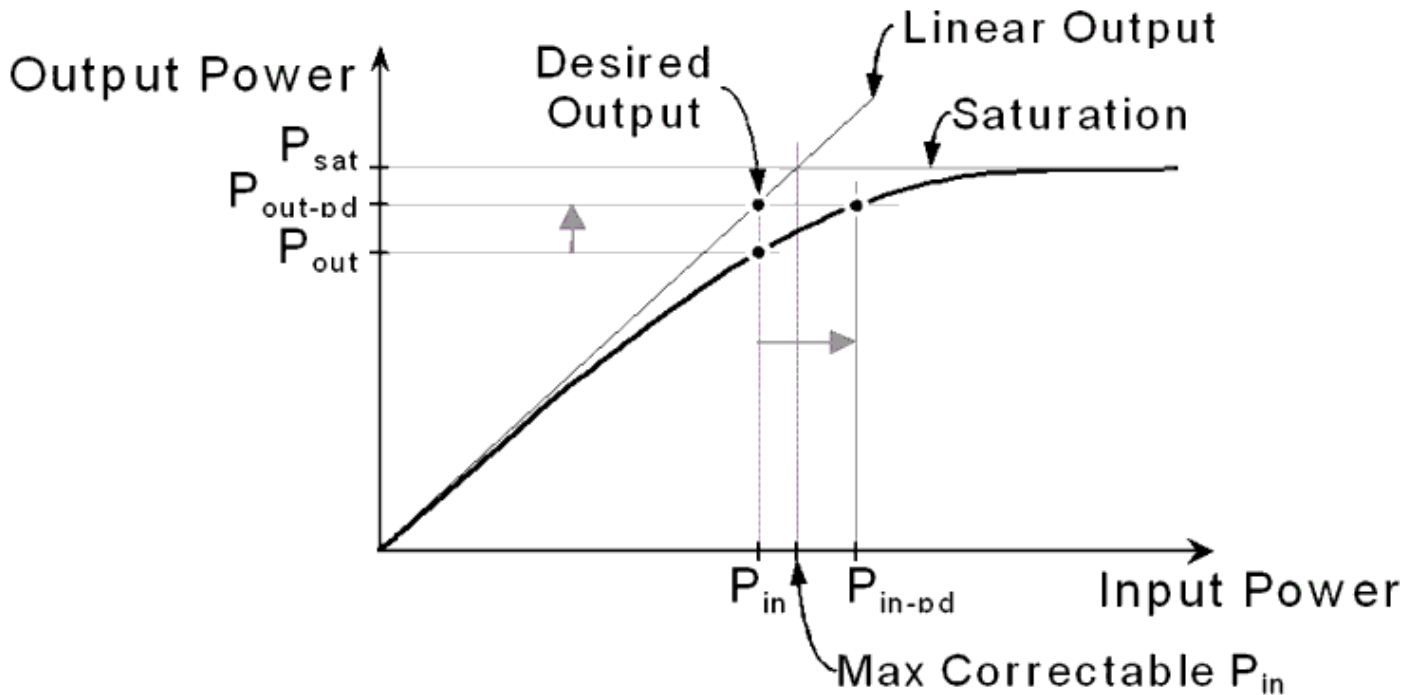


Digital Predistorter Followed by a Power Amplifier

The gain, G , of the PA is modeled as a function of the magnitude of the power amplifier input signal, v_p . The function G is, in general, memoryless and non-linear in amplitude and phase (i.e. complex, when converted to an equivalent baseband model). The use of a memoryless model that is dependent upon input signal magnitude only is a simplification of the actual response of a typical power amplifier. Other variables will affect the power amplifier response, including, most notably, frequency and operating temperature.

The digital predistorter precedes the power amplifier, conditioning the input signal, V_i , before amplification by impressing upon it a complex gain given by the digital predistortion response, F . Similar to G , F is made to be a function of the magnitude of the input signal to the digital predistortion, namely, V_i . Thus, the cascade of the digital predistortion and power amplification will result in the desired linear response when $F(|V_i|) G(|V_p|) = k$, where k is a constant and $V_p = V_i F(|V_i|)$.

The following figure illustrates the typical relationship between the input power and output power of a power amplifier.



Power Amplifier Pout versus Pin and Digital Predistortion

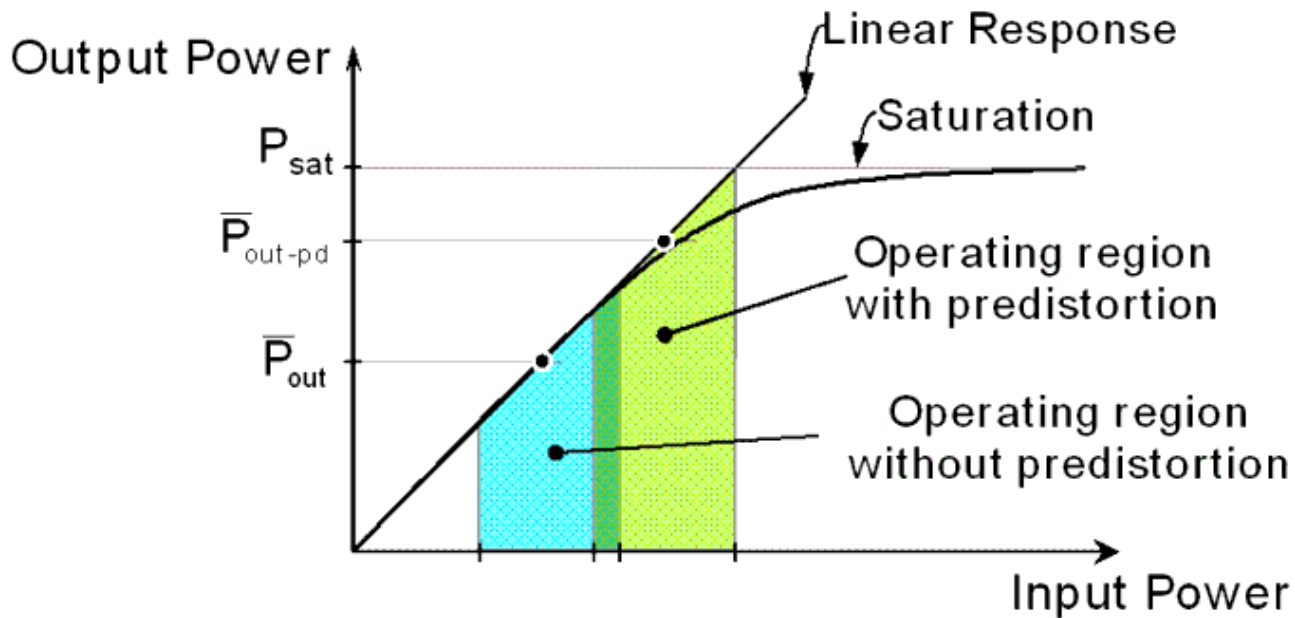
The heavier dark curve shows that in the absence of digital predistortion the power amplifier's P_{out} versus P_{in} curve is highly non-linear. However, through the introduction of digital predistortion, the P_{out} versus P_{in} curve is made to have a linear response over a large range of input power levels. The desired linear response of the power amplifier is illustrated by the Linear Output curve shown in the preceding figure. The slope of the Linear Output curve is the desired linear gain of the power amplifier.

The manner in which digital predistortion works can be understood from the preceding figure. When the amplifier is operating in compression, the P_{out} versus P_{in} curve falls below the Linear Output curve, hence, the actual output power of the power amplifier is not sufficient for linear operation. The inclusion of digital predistortion prior to the power amplification has the effect of introducing expansion - the amplitude of the input signal is increased so that the desired output power (falling on the Linear Output curve) is achieved. The expansion effect of digital predistortion can be observed in the preceding figure where the input power, P_{in} (resulting in P_{out} before PD), is increased to P_{in-pd} so that the power amplifier output power is raised to P_{out-pd} which coincides with the Linear Output curve.

The region of the P_{out} versus P_{in} curve which can be linearized using digital predistortion is not unlimited. The maximum input amplitude that will be amplified linearly is dependent upon the desired linear gain and the saturation output power of the PA, P_{sat} . The output power cannot be expanded beyond the saturation level, thus, the intersection of the Linear Output curve with the P_{sat} level determines the maximum input amplitude that will result in linear amplification.

The amplitude of the input signal to the power amplification may be constant, as in the

case of a GMSK input signal - characterized by a constant envelope. When the input signal is based upon a non-constant envelope modulation scheme, for instance, QPSK, OQPSK, or Mary-QAM, the amplitude of the input signal to the power amplification will vary continuously and rapidly. The upper and lower limits of the amplitude variation of the input signal defines an operating region of the power amplifier. Two operating regions are shown in the following figure.



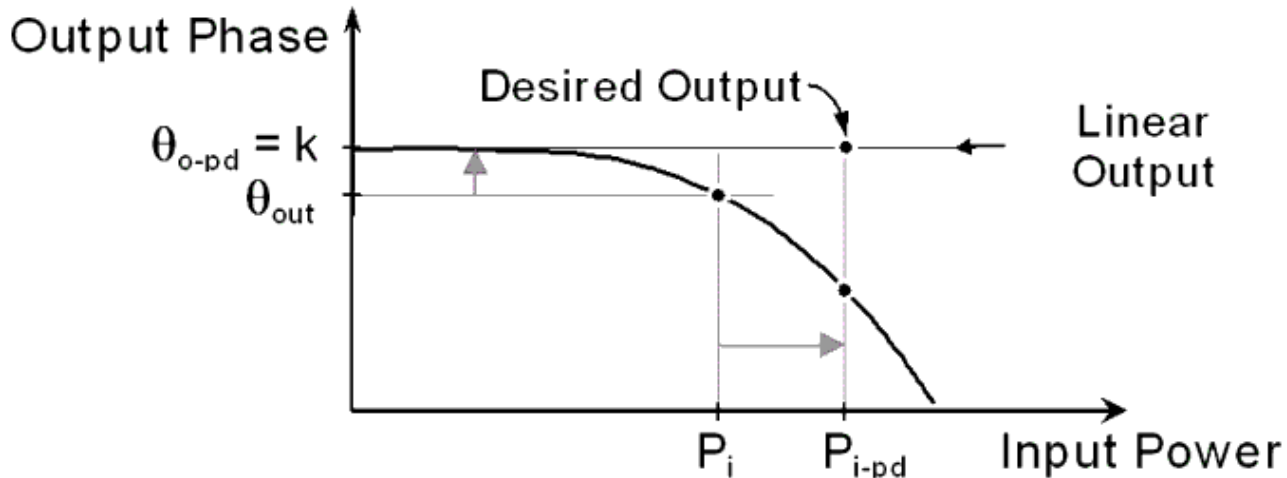
Power Amplifier Operating Region With and Without Digital Predistortion

Each operating region is characterized by an average output power level, \overline{P}_{out} . The magnitude of \overline{P}_{out} is typically described in terms of its relation to the peak output power associated with the signal being amplified by a peak-to-average power ratio (PAR). The PAR is dependent upon the statistics of the signal being amplified.

The non-linear portion of the P_{out} versus P_{in} curve is normally located at the higher output power levels, as saturation is approached (depending on the type of amplifier, non-linear behavior may also be found within other portions of the P_{out} versus P_{in} curve). Generally, to minimize the distortion introduced by the non-linear region, the input power is curtailed or backed-off, so that the operating region falls predominantly within the linear portion of the P_{out} versus P_{in} curve.

It is desirable that the operating region be as close to saturation as possible because amplifier efficiency increases as the average output power increases. Efficiency only increases to a certain point, however, and degrades as more and more of the operating region encompasses saturation. As is shown in the preceding figure, the linearization of the P_{out} versus P_{in} curve through digital predistortion enables the average output power of the power amplifier to be elevated above the average output power of the power amplifier without digital predistortion. Raising the average output power level enables the power amplifier to be operated more efficiently.

The phase response of a power amplifier is also non-linear. Thus, Digital predistortion is also used to linearize the typical non-linear phase response (as illustrated in [Digital Predistorter Followed by a Power Amplifier](#) and [Power Amplifier Pout versus Pin and Digital Predistortion](#)) and the amplitude response. The following figure shows a typical phase response of a power amplifier and the linearizing effect of digital predistortion. The desired phase response is one that is constant at all input power levels.

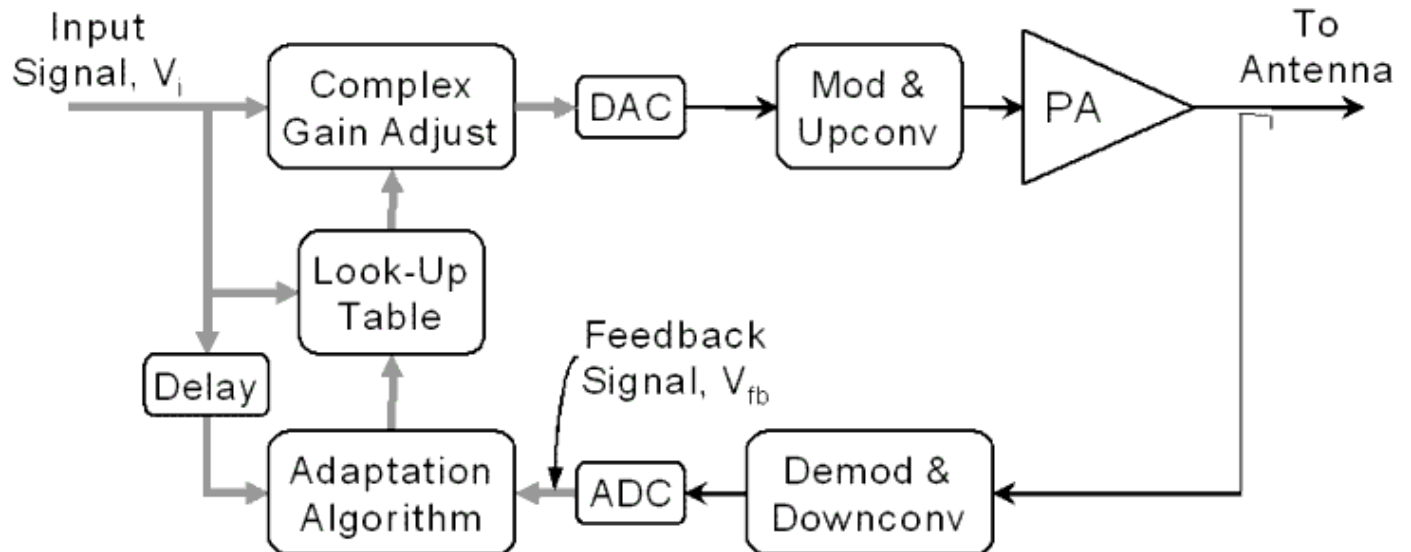


Typical Power Amplifier Phase Response

Digital Predistortion System

A system level block diagram of a digital predistortion system is shown in the following figure. The entire digital predistortion system is housed in the transmitter and requires the addition of several components to the standard transmitter components. The primary components augmenting the modulator, upconverter, and the power amplifier are:

- Complex Gain Adjust block in the baseband signal path
- Look-Up Table for storing a set of complex coefficients
- Demodulator and Downconverter whose input is drawn from the output of the power amplifier
- Adaptation Algorithm processing block that calculates the complex coefficients that are stored in the look-up table



Digital Predistortion Block Diagram

With the inclusion of digital predistortion, the digital complex baseband input signal samples are multiplied prior to the DAC by complex coefficients drawn from the look-up table. The look-up table coefficients implement the predistortion function, F , shown in [Digital Predistorter Followed by a Power Amplifier](#). The adaptation algorithm determines the values of the coefficients by comparing the feedback signal and a delayed version of the input signal.

Note

Ideally, the feedback signal should be a scaled reproduction of the input signal (i.e. linear amplification), thus, the role of the adaptation algorithm is to derive a set of coefficients that forces the error between the scaled feedback signal and the delayed input signal to zero.

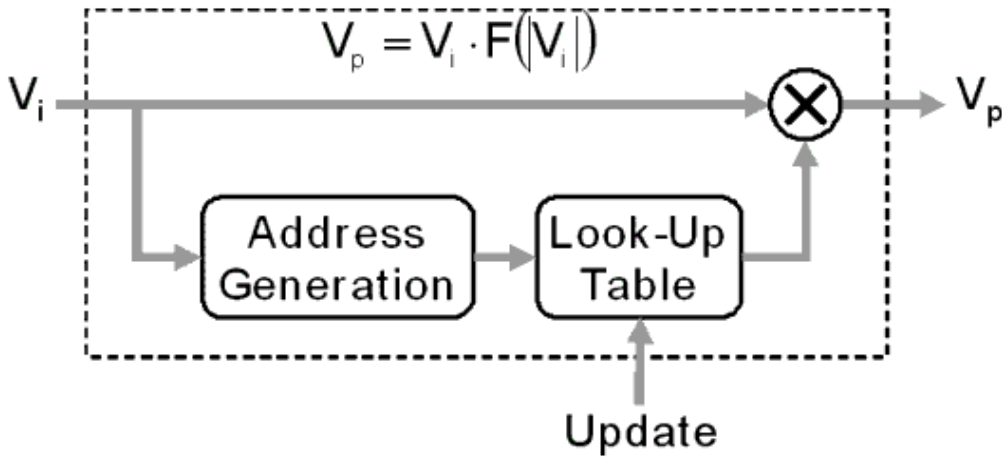
A number of critical design parameters/issues can be readily identified from the digital predistortion system shown in the preceding figure:

- Sampling rate used in the DSP portions of the transmitter
- Number of bits used for quantization by DAC and ADC
- Shape of the reconstruction filter at the output of the DAC
- Bandwidth of the low pass filter at the input of the ADC
- Stability of the feedback loop/adaptation algorithm
- Precision (fixed-point) of the look-up table and DSP processing, if any
- Memory available for the look-up table and adaptation algorithm
- Complexity of the adaptation algorithm and the amount of DSP horsepower required
- Accuracy of the determination of the feedback delay

The intent of implementing the digital predistortion system in ADS is to be able to capture and analyze the impact on performance of some of these design parameters.

Implementing the Predistortion Function

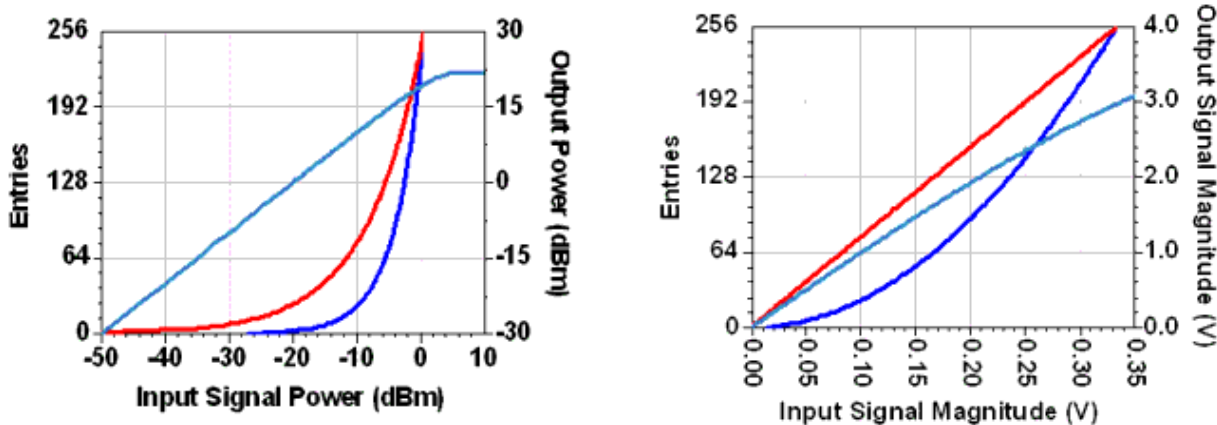
The Complex Gain Adjust and Look-Up Table blocks of the preceding figure are shown in more detail in the following illustration.



Complex Gain Adjust and Look-Up Table

The predistorting function, F , is assumed to be a function of the magnitude of the input signal. The predistorting function is implemented using a complex multiplier, a Look-Up Table and an Address Generation block that selects the appropriate coefficient from the look-up table, given the magnitude of the input signal. The coefficients stored in the look-up table are the value of the predistortion function at certain input signal magnitudes. Thus, the predistortion function is not implemented in an analytic manner, rather, it is only calculated at a specified number of points.

The size of the look-up table employed determines the number of points at which the predistortion function is calculated. In addition, the distribution of the predistortion function points need not necessarily be evenly distributed across the range of the input signal magnitude. Instead, it may be desirable to distribute the predistortion function points across the range of the input signal magnitude using a squared (power) or logarithmic relationship. The following plots illustrate how the predistortion function may be indexed in the look-up table by the magnitude of the input signal and by the power of the input signal. Both plots show the same information, however, the scale of the x-axis is in dBm in the left hand plot and in Volts in the right-hand plot.



Cumulative Distribution of Look-up Table Entries for Magnitude and Power Indexing

For a look-up table with 256 entries, the plots show the cumulative distribution of the entries over the range of the input signal magnitude for the magnitude and power distributions. Overlaid upon the distributions is the amplifier response, illustrating where the entries fall along the amplifier response. The 256th entry in both distributions falls at the maximum input power level that can be linearized (approximately 0.4 dBm). The look-up table entries are equally spaced over the range of the magnitude of the input signal in the case of magnitude indexing, whereas, more look-up table entries are distributed at the higher end of the range in the case of power indexing.

Adaptation Algorithm

The function of the adaptation algorithm is to derive the predistortion function, F , i.e. the inverse characteristic of the amplifier response. The predistortion function may be derived using either a modulated signal input (random signal) or a known training signal input. The adaptation algorithm and its implementation are fundamentally different depending upon which type of input signal is utilized. The algorithms that are based upon the use of a modulated signal, employ statistical signal processing and, typically, some type of curve fitting algorithm to generate a smooth predistortion function. The complexity of the adaptation algorithm and its implementation can be significantly simplified, however, by using the alternative input signal - a known training signal.

Training Signal

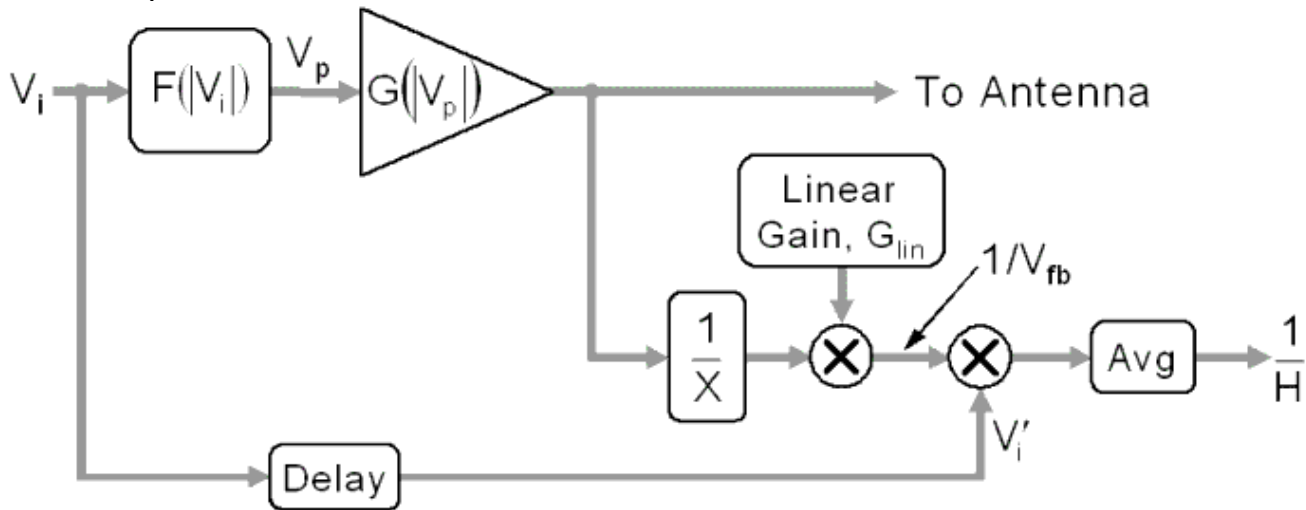
The adaptation algorithm chosen for implementation is based upon the use of a training signal. The training signal is a single tone having a frequency equal to the carrier frequency and whose power is ramped up over the duration of the training period. The power of the tone is set to zero at the start of the training period and will typically peak at, or just below, the maximum correctable input power of the amplifier.

The use of a single tone whose power is ramped as a training signal greatly simplifies the adaptation algorithm and its implementation. However, the use of the training signal does require that the modulated signal being transmitted be interrupted while the training signal is transmitted. In addition, because the training signal is a single tone, the digital predistorter is only correcting for the operation of the amplifier at a single frequency, not across the entire transmission bandwidth. If the amplifier's passband is quite flat, the use of a single tone training signal in this manner will enable the predistortion function to be determined accurately. In any event, the use of a single complex coefficient to correct for distortion of the amplifier at a particular power level presupposes that any amplifier memory effects are minimal, i.e. the amplifier has a flat passband.

Implementation

The following illustration is a block diagram of the adaptation algorithm that is employed. The algorithm is based upon the determination of the open loop gain, H , of the

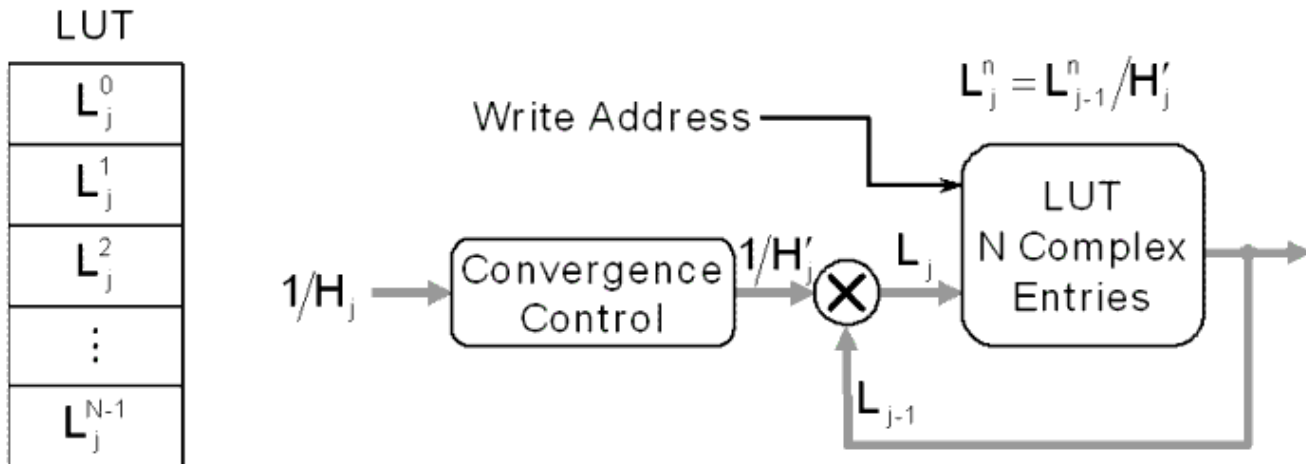
predistorter and amplifier combination at the power level associated with each look-up table entry.



$$\frac{1}{H} = \frac{V_i'}{V_{fb}} = \frac{G_{lin}}{F(|V_i|) \cdot G(|V_p|)} \Rightarrow 1 + j \times 0$$

Implementation of the Adaptation Algorithm

Recall that the desired linear response of the predistorter and amplifier cascade requires that $F(|V_i|) G(|V_p|) = k$ for all inputs. Hence, if G_{lin} is set to be equal to k , the desired open loop gain of the system is unity. If the calculated open loop gain is not equal to unity, the predistortion function must be adjusted in a manner to drive the open loop towards unity. This is can be achieved as illustrated in the following manner.



Calculation of New Predistortion Function

The predistortion function is defined by a set of coefficients stored in the look-up table, L_n , where each n corresponds to an input signal magnitude which is mapped to a look-up table address. In order to drive the open loop gain to unity, the predistortion function

coefficients are updated by dividing each coefficient by the calculated open loop gain (or by the calculated open loop gain adjusted to slow the rate at which the coefficients will change).

Synchronization

The accuracy of the open loop gain calculation is dependent upon the accuracy of the estimation of the delay in the feedback path. As shown in [Implementation of the Adaptation Algorithm](#), the input signal, V_i , must be delayed precisely by an amount equal to the delay in the feedback path.

The delay in the feedback path is estimated by calculating the correlation between the magnitude of the input signal and the magnitude of the feedback signal. The use of the magnitude of the signals has the benefit of not requiring phase synchronization in the feedback path. Because the delay in the feedback path will not necessarily be equal to an integer number of DSP sample periods, interpolation is employed to more precisely align the input and feedback signals.

The correlation between the input and feedback signal is performed on a modulated signal that precedes the training signal because the gain compression of the amplifier makes the accuracy of the correlation over the training signal suspect. In addition, because the envelope of the modulated signal will typically have a pdf such that it spends much of its time within the linear operating region of the amplifier, correlation using the modulated signal becomes more reliable. However, because the modulated signal is stochastic, the statistics of the modulated signal, as well as the size of the data block over which the correlation operation is performed will impact the accuracy of the delay estimation. In general, the accuracy of the estimation improves as the block size increases. Unfortunately, a larger block size requires more memory and takes longer to perform the estimate.

Convergence

The predistortion function cannot be exactly determined following the transmission of single training ramp and recalculation of the look-up table coefficients thereafter. A series of training ramps will have to be transmitted - although, significant improvements in the ACPR of the amplifier should be observed even after a single training ramp. Simulations have shown that the predistortion function can converge to a solution after only six training ramps.