Digital Pre-Distortion v8.1

LogiCORE IP Product Guide

Vivado Design Suite

PG076 July 3, 2017

XILINX
ALL PROGRAMMABLE™

Table of Contents

Chapter 5: Test Bench

Chapter 6: Application Software

Appendix A: Verification, Compliance, and Interoperability

Appendix B: Migrating and Upgrading

Appendix C: Debugging

Appendix D: Correction Performance

Appendix E: Additional Resources and Legal Notices

Introduction

The LogiCORE™ IP Digital Pre-Distortion (DPD) core negates the non-linear effects of a power amplifier (PA) when transmitting a wide-band signal. DPD allows a PA to achieve greater efficiency by operating at a higher output power while maintaining spectral compliance, and reducing system capital and operational expenditure.

Features

- Algorithms:
  - DPD correction with up to 40 dB of adjacent channel leakage ratio (ACLR) improvement
- Physical Configuration Parameters
  - Selection of phase options for datapath implementation allowing a resource/ sample rate trade-off
  - Selection of one, two, four, six or eight transmit antennas
  - Smaller filter and capture depth options for low cost solutions like micro remote radio head (RRU), distributed antenna system (DAS) and low power PA applications.
  - Independent control of filter memory depth, capture memory depth and acceleration levels allowing for resource versus performance trade-off
- Software
  - Added support for SMP mode under Linux Operating System.

See Feature Summary for more detailed feature information.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Zynq®-7000 and Zynq UltraScale+
Supported User Interfaces	AXI4, AXI4-Lite, AXI4-Stream.
Resources	
Provided with Core	
Design Files	Local Vivado® Repository
Example Design	Not Provided
Test Bench	See Test Bench
Constraints File	See Constraining the Core
Simulation Model	Not Provided
Supported S/W	Executable and linkable format files are now packaged along with DFE Subsystem XRF2 Reference Design which needs to be downloaded separately.
Tested Design Flows(2)	
Design Entry	Vivado Design Suite 2017.2
Simulation	Supported. See Test Bench for more details
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Pre-distortion negates the non-linear effects of a Power Amplifier (PA) generated when transmitting a wide-band signal. Pre-distortion allows a PA to achieve greater efficiency by operating at higher output power while still maintaining spectral compliance, reducing system capital and operational expenditure.

The solution is targeted for base stations used in third and fourth generation (3G/4G) mobile technologies and beyond. It is a combination of hardware and embedded software processes that between them realize pre-distortion correction along with features that make for a fully engineered, practical, robust and self-contained solution.

The DPD solution comprises both a hardware core (DPD HW) and software application (DPD SW). For hardware engineers, the design flow involves customizing and integrating the DPD HW component into the device subsystem. To facilitate this, a detailed description of the interface requirements is described in Chapter 3, Designing with the Core followed by a description of the customization options in Chapter 4, Design Flow Steps. Individual system applications can interact with the DPD SW using the various registers and commands described in Chapter 6, Application Software. This chapter also includes guidance for optimizing the correction performance.

Feature Summary

- Algorithms
  - DPD correction with up to 40 dB of ACLR improvement
  - Improved support for signal dynamics (including high PAPR signals possible with some DAS deployments)
  - Frequency Division Duplex (FDD) and Time Division Duplex (TDD) support with automatic data selection
  - Feedback receiver Quadrature modulator correction
  - PA saturation (overdrive) detection
  - Independent runtime parameters for each antenna path configuration for selected hardware datapath
  - Signal capture and analysis

- Easy integration and evaluation using the Debug Interface utility [Ref 1]

- Physical Configuration Parameters
  - Multi-Phase Data Interface to support wider pre-distortion bandwidth at the same processing clock frequency by accepting multiple samples in parallel
  - One, two, four, six, or eight transmit antennas
  - Increased filter/capture depth offers improved pre-distortion correction in some higher bandwidth or multi-carrier use cases. Additional smaller filter depths and capture depths allow better cost trade-offs for low power PAs used in small cells.
  - Option to reduce FPGA footprint significantly by selecting low precision coefficients with some performance trade-offs.

Applications

Digital pre-distortion forms part of the digital front-end (DFE) processing before the analog converters within the radio equipment. The transmit path in a typical DFE application system is shown in Figure 1-1.

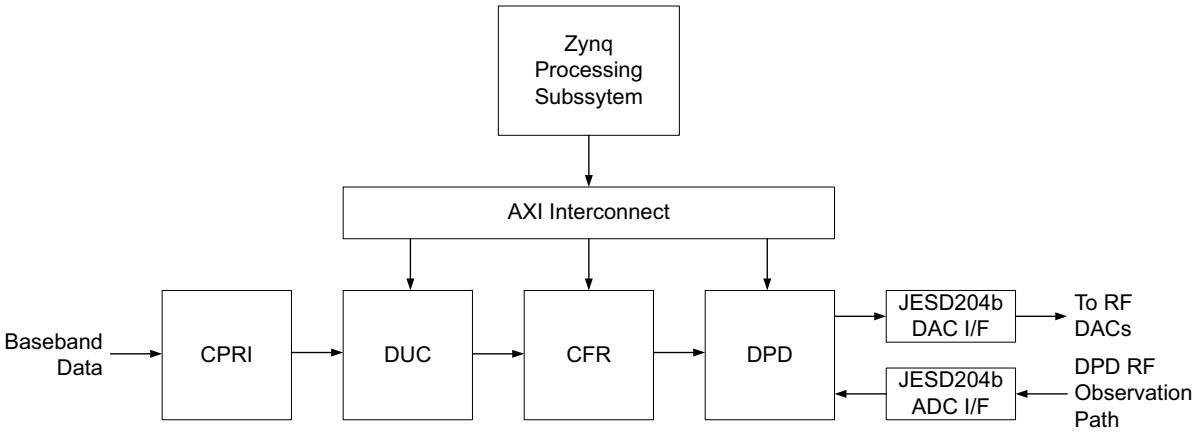


Figure 1-1: **Typical DFE Transmit Path**

Xilinx provides IP cores and application notes for the subsystems and blocks shown in the transmit path:

- **Common Packet Radio Interface (CPRI)** - The CPRI slave receives baseband IQ Data over the CPRI™ Link from the Radio Equipment Controller (REC) (see CPRI for information about this core).
- **Digital Up Converter (DUC)** - takes a number of antenna streams from the CPRI, performs mixing and up-conversion to rates required by subsequent stages (see DUC/DDC Compiler for information about this core). DUC design changes are based on end product requirements for remote radiohead (RRH), small cell and DAS.

- **Crest Factor Reduction (CFR)** - improves the peak-to-average power ratio (PAPR) of the signal (see PC-CFR for information about this core).
- **Digital Pre-Distortion (DPD)** - corrects for non-linearity in the PA behavior. It requires an observation path from the PA output to correct for dynamic behavior of the PA.
- **JESD204B** - provides SerDes serial transceiver links to and from the DACs and ADCs (see JESD204 for information about this core).
- **Zynq® Processing System** - sets up and controls the RF processing chain. Runs the software portion of DPD processing (see the *Zynq-7000 All Programmable SoC Overview* (DS190) [Ref 2] and *Zynq UltraScale+ MPSoC Overview* [Ref 21]

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado® Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability.

For more information, visit the Digital Pre-Distortion product web page.

Information about other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

The DPD v8.1 core is available as an evaluation version which operates for several hours, depending on the clock frequency. The data output is set to zero after the evaluation period ends. The host interface reports EVAL_LICENSE_TIMEOUT status value when the hardware times out.

Product Specification

Standards

The DPD IP core is designed to work with various cellular standards signals including WCDMA, WiMAX, CDMA2000, TD-SCDMA, LTE (both FDD and TDD modes), and any combination of these in a multi-Radio Access Technology (RAT) configuration.

IMPORTANT: *It is strongly recommended to contact Xilinx® and discuss your Global Systems for Mobile Communications (GSM) standard's deployment before proceeding to use this product. For various stringent GSM spectral emission mask (SEM) requirements, some additional functionality may be necessary that Xilinx can evaluate and suggest, as a customized DPD solution.*

RF Performance

Contact Xilinx Technical Support to receive example RF performance results (ACLR, SEM) with latest generation PA transistors for various cellular signal standards in different configurations.

IP Timing Performance

The DPD IP core timing performance was verified on Zynq®-7000 series and Zynq® UltraScale + devices. The clock speeds shown in Table 2-1 and Table 2-2 were achieved when a single DPD IP core is placed and routed as part of an example DFE system on appropriate Zynq/Zynq UltraScale+ devices.

The Vivado® Design Suite 2017.2 tools on a Zynq 7045 (-2 speed grade) were used to verify the timing performance shown in Table 2-1.

Table 2-1: **DPD Core Timing Performance - Zynq-7045 (-2)**

Number of Phases	dpd_aclk Speeds (MHz)	dpd_internal_aclk Speeds (MHz)	s_axi_user_aclk Speeds (MHz)	s_axi_ctrl_aclk Speeds (MHz)	s_axi_ctrl_2x_aclk Speeds (MHz)
1	368	368	200	200	400
2 (HYBRID = 0)	368	368	200	200	400
2 (HYBRID = 1)	245	491	200	200	400

The Vivado Design Suite 2016.3 tools on a Zynq 7015 (-2 speed grade) were used to verify the timing performance as shown in Table 2-2.

Table 2-2: **DPD Core Timing Performance - Zynq-7015 (-2)**

Number of Phases	dpd_aclk Speeds (MHz)	dpd_internal_aclk Speeds (MHz)	s_axi_user_aclk Speeds (MHz)	s_axi_ctrl_aclk Speeds (MHz)	s_axi_ctrl_2x_aclk Speeds (MHz)
1	245	245	150	150	300
2 (HYBRID = 0)	245	245	150	150	300
2 (HYBRID = 1)	184	368	150	150	300

Table 2-3 show the timing performance on a Zynq UltraScale+ ZU9EG (-1L speed grade) using the Vivado Design Suite 2016.3 tools.

Table 2-3: **DPD Core timing performance for Zynq UltraScale+ xczu9eg (-1L)**

Number of Phases	dpd_aclk (MHz)	dpd_internal_aclk (MHz)	s_axi_user_aclk (MHz)	s_axi_ctrl_aclk (MHz)	s_axi_ctrl_2x_aclk (MHz)
1	491.52	491.52	200	200	400
2 (Hybrid=0)	491.52	491.52	200	200	400
2 (Hybrid=1)	307.2	614.4	200	200	400

Resource Utilization

DPD IP (Hardware and Software) uses resources in Zynq-7000/Zynq UltraScale+ Programmable Logic (PL) and Zynq-7000/Zynq UltraScale+ Processing Subsystem (PS). The PL resource utilization for all supported configurations of the DPD IP core is provided in the DPD evaluation lounge.

The DPD software is loaded and run from PS double data rate (DDR) memory. For details on DDR memory configurations supported by particular parts, refer to *Zynq-7000 AP SoC Technical Reference Manual (TRM)* [Ref 13] and Zynq-UltraScale+ MPSoC Technical Reference Manual [Ref 18]. DPD requires 15 MB of storage when using DPD as a single processor bare-metal application. If asymmetric multi-processing (AMP) configuration is

used, DPD requires 15 MB of contiguous space on the DDR memory apart from the storage requirements of the operating system on the CPU0.

When running DPD as an application on SMP Linux, DPD executable will typically require 1.5MBytes in the root file system. Typically, resident set size (RSS) reported by `ps` command is 4.5 MBytes when running DCL with eight antennas. It is recommended to characterize DPD software running within your own embedded Linux System to understand its resource requirements and loading with respect to other processes in your system.

Datapath Latency

The latency of the core is primarily a function of filter memory depth and number of phases. Table 2-4 given below shows the latency of the dpd_aclk core in cycles. This is measured from input `s_axis_din_tlast` to output `m_axis_dout_tlast`.

Note: When you select Reduced Precision coefficients, then the latency reduces by 7 cycles for non-hybrid designs and by 4 cycles for hybrid designs.

Table 2-4: **Latency of dpd_aclk core**

Filter Memory Depth	Latency (cycles)		
	Phases=1	Phases=2	Phases=2(Hybrid)
1	73	71	47
2	79	75	50
4	89	81	55
5	93	83	57
6	97	85	59

Port Descriptions

Figure 2-1 displays the signal names of the DPD HW component. Table 2-5 defines these signals. The AXI4-Stream interfaces are parameterized by the number of transmit antennas (TX) and the number of phases (NUM_PHASES). For two phases, two samples per antenna path are input and output from the core on each clock cycle. For one phase, one sample per antenna path is input and output from the core on each clock cycle.

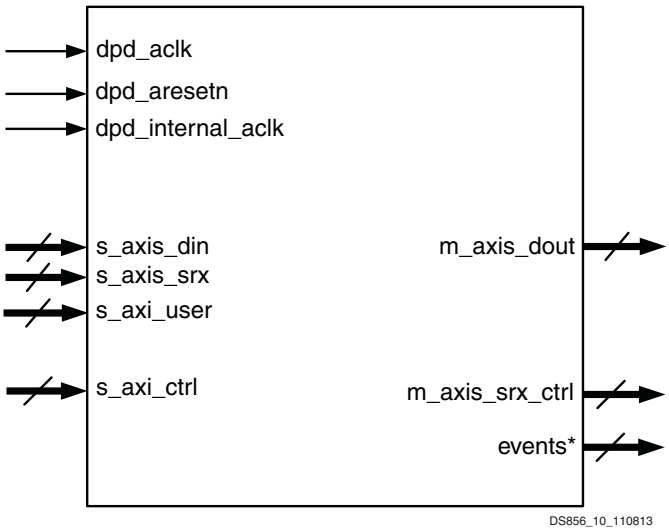


Figure 2-1: **DPD IP Symbol (Input/Output Ports)**

Table 2-5: **Top-Level I/O for the DPD IP Core**

Names	Direction	Description
dpd_aclk	Input	Rising edge clock for all AXI4-Stream data interfaces (s_axis_din, s_axis_srx, m_axis_srx_ctrl and m_axis_dout).
dpd_aresetn	Input	Common active-Low reset for datapath. Designed to immediately halt output data and initiate full DPD reset process. Synchronous to dpd_aclk.
dpd_internal_aclk	Input	Rising edge clock for internal datapath logic. When using the core in the two-phase hybrid mode, this must be phase aligned and two times the frequency of dpd_aclk. Otherwise this must be tied to the same clock as dpd_aclk.
DIN Interface (Slave AXI4-Stream)		
s_axis_din_tdata[16*2*NUM_PHASES*TX-1:0]	Input	16 bit Q and I component per transmit antenna cascaded.
s_axis_din_tvalid	Input	Valid handshake.
s_axis_din_tready	Output	Ready handshake.
s_axis_din_tuser[8*TX-1:0]	Input	Control information for each transmit antenna. Consists of frame/slot markers and a capture strobe.
s_axis_din_tlast	Input	For future expansion, should be tied Low.
Sample Receiver (SRX) Interface (Slave AXI4-Stream)		
s_axis_srx_tdata[16*2*NUM_PHASES-1:0]	Input	16 bit Q and I component for common feedback path between all antennas.
s_axis_srx_tvalid	Input	Valid handshake
s_axis_srx_tready	Output	Ready handshake

Table 2-5: **Top-Level I/O for the DPD IP Core *(Cont'd)***

Names	Direction	Description
s_axis_srx_tuser[7:0]	Input	Indicates the antenna path that the current feedback data is associated with and where burst analog-to-digital converter (ADC) mode is used indicates burst status.
DOUT Interface (Master AXI4-Stream)		
m_axis_dout_tdata[16*2*NUM_PHASES*TX-1:0]	Output	16 bit Q and I component per transmit antenna cascaded.
m_axis_dout_tvalid	Output	Valid handshake
m_axis_dout_tready	Input	Ready handshake
SRX Control Interface (Master AXI4-Stream)		
m_axis_srx_ctrl_tdata[7:0]	Output	Control output for SRX input stream to select antenna feedback data and support burst ADC interfaces.
m_axis_srx_ctrl_tvalid	Output	Valid handshake
m_axis_srx_ctrl_tready	Input	Ready handshake
Host Interface (AXI4-Lite)		
s_axi_user_aclk	Input	Rising edge clock
s_axi_user_aresetn	Input	Active-Low synchronous reset
s_axi_user_wdata[31:0]	Input	Write data
s_axi_user_wvalid	Input	Write data valid handshake
s_axi_user_wready	Output	Write data ready handshake
s_axi_user_awaddr[31:0]	Input	Write address (byte addressing)
s_axi_user_awvalid	Input	Write address valid handshake
s_axi_user_awready	Output	Write address ready handshake
s_axi_user_araddr[31:0]	Input	Read address (byte addressing)
s_axi_user_arvalid	Input	Read address valid handshake
s_axi_user_arready	Output	Read address ready handshake
s_axi_user_bresp[1:0]	Output	Write response
s_axi_user_bvalid	Output	Write response valid handshake
s_axi_user_bready	Input	Write response ready handshake
s_axi_user_rdata[31:0]	Output	Read data
s_axi_user_rvalid	Output	Read data valid handshake
s_axi_user_rready	Input	Read data ready handshake
s_axi_user_rresp[1:0]	Output	Read data response
s_axi_ctrl (AXI4-Lite) A single AXI4-Lite bus used by the DPD application for communication with the hardware components. This bus should be connected to the PS.		

Table 2-5: **Top-Level I/O for the DPD IP Core *(Cont'd)***

Names	Direction	Description
s_axi_ctrl_aclk	Input	Rising edge clock
s_axi_ctrl_2x_aclk	Input	Rising edge clock. This must be phase aligned and two times the frequency of s_axi_ctrl_aclk.
s_axi_ctrl_aresetn	Input	Active-Low synchronous reset. This is synchronous to s_axi_ctrl_aclk clock domain.
s_axi_ctrl_wdata[31:0]	Input	Write data
s_axi_ctrl_wvalid	Input	Write data valid handshake
s_axi_ctrl_wready	Output	Write data ready handshake
s_axi_ctrl_awaddr[31:0]	Input	Write address (byte addressing)
s_axi_ctrl_awvalid	Input	Write address valid handshake
s_axi_ctrl_awready	Output	Write address ready handshake
s_axi_ctrl_araddr[31:0]	Input	Read address (byte addressing)
s_axi_ctrl_arvalid	Input	Read address valid handshake
s_axi_ctrl_arready	Output	Read address ready handshake
s_axi_ctrl_bresp[1:0]	Output	Write response
s_axi_ctrl_bvalid	Output	Write response valid handshake
s_axi_ctrl_bready	Input	Write response ready handshake
s_axi_ctrl_rdata[31:0]	Output	Read data
s_axi_ctrl_rvalid	Output	Read data valid handshake
s_axi_ctrl_rready	Input	Read data ready handshake
s_axi_ctrl_rresp[1:0]	Output	Read data response
s_axi_ctrl_wstrb[3:0]	Input	When High, specifies the byte lanes of the data bus that contain valid information. There is one write strobe for each eight bits of the write data bus, therefore s_axi_ctrl_wstrb [n] corresponds to s_axi_ctrl_wdata[(8n)+7: (8n)].
s_axi_ctrl_arprot[2:0]	Input	Defines access permissions for read accesses.
s_axi_ctrl_awprot[2:0]	Input	Defines access permissions for write accesses.
events*		
event_s_din_underrun	Output	Underrun status flag for s_axis_din AXI4-Stream interface.
event_m_dout_overrun	Output	Overrun status flag for m_axis_dout AXI4-Stream interface.
event_s_srx_underrun	Output	Underrun status flag for s_axis_srx AXI4-Stream interface.

Register Space

The register space on the `s_axi_ctrl` interface is proprietary and not described in this document.

The host interface on the `s_axi_user` interface provides a 1Kx32 area of shared memory that is used to communicate with the application software. Details of this interface and its required memory map are described in Chapter 6, Application Software.

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

Clocking

This core requires five clock input signals.

dpd_aclk

Signal `dpd_aclk` is used to drive the main data interface logic within the design. Its clock frequency determines the data rate of the DPD IP core. All AXI4-Stream interfaces (`s_axis_din`, `m_axis_dout`, `s_axis_srx`, `m_axis_srx_ctrl`) are synchronous to this clock.

dpd_internal_aclk

Signal `dpd_internal_aclk` is used to clock the internal datapath logic. When using the two-phase hybrid configuration, this clock should be phase-aligned but double the frequency of `dpd_aclk`; for the single and non-hybrid two-phase configurations, this clock should be tied to the same source clock as `dpd_aclk`.

s_axi_user_aclk

Signal `s_axi_user_aclk` is a dedicated input for the AXI4-Lite interface that provides asynchronous access to the shared memory for passing instructions to, and reading status from the DPD IP software application running on the ARM® processor.

s_axi_ctrl_aclk

This `s_axi_ctrl_aclk` clock is used to time all transfers on the `s_axi_ctrl` interface.

s_axi_ctrl_2x_aclk

The s_axi_ctrl_2x_aclk is used to over-clock internal logic. This clock has a strict requirement that it is phase aligned and double the frequency of s_axi_ctrl_aclk. Hardware acceleration performance is closely tied to this clock.

RECOMMENDED: *Xilinx recommends that the maximum achievable frequency is applied to s_axi_ctrl_2x_aclk with s_axi_ctrl_aclk being set to exactly half that frequency.*

Resets

All reset signals have active-Low sensitivity.

In the design, dpd_aresetn is the main reset and should be synchronous to dpd_aclk; this is used to reset all system logic and the software state in the design. Asserting dpd_aresetn immediately disables the output data and initiates a software controlled reset process that reinitializes the system to a power-on state including clearing any parameters provided on the host interface. After deasserting dpd_aresetn, you must wait for the DPD to indicate that it is ready through the host interface and then reapply any configuration parameters specific to your system.

The AXI4-Lite user interface has its own dedicated reset, s_axi_user_aresetn, and must be synchronous to s_axi_user_aclk. It resets all AXI4-Lite interface logic and will prevent any communication on the user interface. Hence, it must only be applied when no transactions are pending on the user interface.

The reset s_axi_ctrl_aresetn is specific to the interface used by the DPD software application to communicate with the hardware.

Reset and Clock Sequencing

The standard power-on sequencing of reset and clocks is as follows:

1. Power-On: assert dpd_aresetn, s_axi_ctrl_aresetn, s_axi_user_aresetn
2. Bring-Up Bus Clocks: bring-up s_axi_ctrl_acllk, s_axi_ctrl_2x_aclk, s_axi_user_aclk. Once these clocks are stable, de-assert s_axi_ctrl_aresetn and s_axi_user_aresetn.
3. Start DPD Software: Once it is possible to communicate on s_axi_ctrl interface, then the DPD Software can start running. However, it will wait for de-assertion of dpd_aresetn.

4. Bring Up Data Rate Clocks: Once `dpd_aclk` and `dpd_2x_aclk` are stable, then `dpd_aresetn` can be de-asserted.
5. Follow the Instructions in DPD Application Software Boot Process section of Chapter 6, Application Software to start the DPD process.

To power-down or reset the system, the above steps must be reversed. When DPD Application is running, `dpd_aresetn` can be applied at anytime whilst the system is operating. It will disable the output and will initiate a reset process within the DPD Software. Operation of DPD can be re-started, starting from step 4.

To power-down the system further, the operation of the DPD Software must be terminated. To end operation of the DPD, the KILL_DPD control mode must be issued. See Software Control Modes for more information. Access on the `s_axi_user` interface should also be stopped. At this point, it is possible to apply the bus interface resets `s_axi_ctrl_aresetn` and `s_axi_user_aresetn`.

Protocol Description

The DPD IP core requires various interfaces to the DFE system for successful operation of the design. The DPD v8.0 core uses AXI4 interfaces for both data and control. The presence of AXI4, AXI4-Lite and AXI4-Stream interfaces brings standardization and enhanced interoperability of Xilinx® IP LogiCORE™ solutions. For further details on AXI4 interfaces see the *AMBA AXI4-Stream Protocol Specification* (IHI 0051A) [Ref 3], *Xilinx AXI Design Reference Guide* (UG1037) [Ref 4] and *ARM AMBA AXI and ACE Protocol Specification (IHI 0022D)* [Ref 5].

Datapath AXI4-Stream Interfaces

The input TX (baseband) data is received through the slave interface (`s_axis_din`) and outputs pre-distorted data through the master interface (`m_axis_dout`) with the width of the `tdata` streams sized according to the number of antennas and phases.

The basic element of the datapath is a 32-bit complex sample composed of 16 bits of real and imaginary signed data concatenated with the real component in the lower 16 bits of the 32-bit word. The total width of the datapath is then determined by the number of transmit antennas and the number of phases. For one phase, the input data for each antenna should be concatenated as per Table 3-1.

Table 3-1: **One Phase Per Antenna Concatenation**

Antenna 1		Antenna 0	
Q	I	Q	I
63:48	47:32	31:16	15:0

For two-phase data, the two complex samples per clock per antenna are concatenated as shown in Table 3-2 where sample S0 precedes sample S1 in the serial data stream.

Table 3-2: **Two Phases Per Antenna Concatenation**

Antenna 1				Antenna 0			
S1		S0		S1		S0	
Q	I	Q	I	Q	I	Q	I
127:112	111:96	95:80	79:64	63:48	47:32	31:16	15:0

For data widths other than 16 bits, pad or round the data as appropriate to form the input data for a particular transmit path based on an understanding of the signal statistics and dynamics. Where the bit width is less than 16 bits it should be zero-padded at the LSB, and where it is greater than 16 bits it should be symmetrically rounded; in both cases, it is important to ensure that the sign bit is retained. The data out contains a corresponding 32 bits for each transmit path which can be connected to a digital mixer or complex digital-to-analog converter (DAC). If the TX DAC has a lower bitwidth (14 bits, for example), symmetric rounding should be applied to convert the signals from 16 to 14 bits. Symmetric rounding is required to prevent any unwanted DC offset, which introduces a spur in the transmitted spectrum in cases where the signal itself has no DC energy, for example wideband code division multiple access (WCDMA) carrier configurations [0 0 1] and [1 0 1].

After power-up or reset, the core is idle until `s_axis_din_tvalid` has been asserted for the first time; `s_axis_din_tready` is held High and `m_axis_dout_tvalid` is held Low by the core until this occurs. After this, the core is synchronized and expects data input and provides data output every clock cycle. After synchronization, the core expects `s_axis_din_tvalid` to remain asserted, otherwise an underrun condition occurs. As the core is free-running, underrun causes the preceding data sample to be input for a second time internally. Underrun is signaled by asserting the `event_s_din_underrun` flag for a single clock cycle after the condition has occurred.

The `s_axis_din_tuser` field is used to supply sideband/control information alongside the `tdata` field in real time. It contains one byte per antenna, with byte 0 being used by antenna 0, byte 1 by antenna 1 and so on. Information on `s_axis_din_tuser` is output on `m_axis_dout_tuser` and is delayed to match the latency of the DPD Filter. The use of bits within each byte of TUSER and the field packing for each antenna byte is shown in Table 3-3.

Table 3-3: **TUSER Field Packing for each Byte**

Bits	Name	Description
7:4	user bits	Available for customer use. Demo System uses bit 4 to carry Power Amplifier Enable Signal in TDD system.
3	capture sync	Used as capture reference point by DPD
2	alternate frame marker	Alternative Frame Marker. For example, GSM Framing in Multi-RAT System

Table 3-3: **TUSER Field Packing *(Cont'd)*for each Byte**

Bits	Name	Description
1	frame marker	Primary framing signal. Typically 10ms time-base in 3G/4G Systems.
0	slot marker	Indicates Data Slot. Also used by Xilinx PC-CFR IP Core.

The frame and slot marker are not used but might be required for future feature enhancements. The capture synchronization strobe is used with capture mode 1 (See Updating and Reporting DPD Parameters). The capture is triggered CAPTUREDELAY samples after a rising-edge is detected on the capture synchronization bit (Capture Sync) which is internally sampled by `dpd_aclk`. You can assert this bit for one or more cycles of `dpd_aclk`; the capture is triggered relative to when it was asserted and it is irrelevant how many cycles it is asserted for.

After the core has received its first valid data on the `s_axis_din` interface, `m_axis_dout_tvalid` is asserted a number of cycles later equal to the core latency; after this, it remains permanently asserted to indicate valid output samples on the `m_axis_dout_tdata` port every cycle. The core expects the data to be read by asserting `m_axis_dout_tready` every cycle while the data updates with a new output sample. If an output sample is not read each cycle due to `m_axis_dout_tready` being deasserted, then the overrun condition occurs. As the core is free-running, overrun causes the next output sample to be dropped internally, as the current one has not been read and remains on the output port `m_axis_dout_tdata`. Overrun is signaled by asserting the `event_m_dout_overrun` flag for a single cycle.

Figure 3-1 shows an example timing diagram. The timing diagram shows the first slave valid pulse which synchronizes the interfaces and initiates the free running flow of data into and out of the core. The diagrams demonstrate invalid operation as signaled by the underrun and overrun flags.

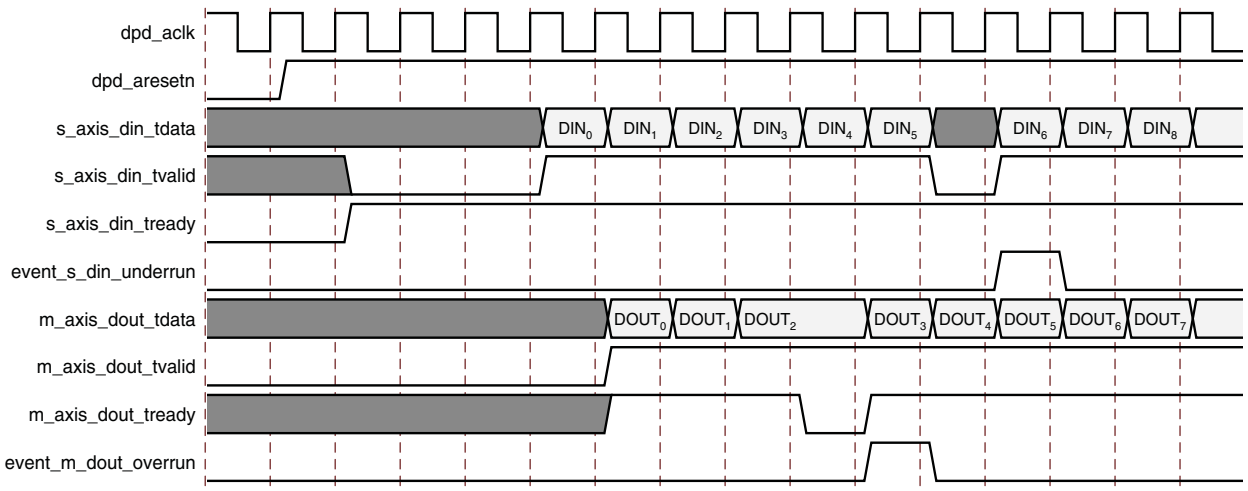


Figure 3-1: **AXI4-Stream Datapath Interface**

Sample Receiver (SRX) AXI4-Stream Interfaces

The slave AXI4-Stream `s_axis_srx` operates exactly the same as the slave data input interface documented in Datapath AXI4-Stream Interfaces. The relevant timing diagram demonstrating how this behaves on power up/reset is shown in Figure 3-1. Data can be provided on this interface independent of the main interface. Valid data should be indicated by asserting `s_axis_srx_tvalid`. When `s_axis_srx_tvalid` has been asserted, it is expected to remain High or an underrun event is flagged.

The data can be either complex (IQ) format or real samples, depending on the system design. The DPD software uses the `RXINPUTFORMAT` and `TXRXRATIO` parameters to interpret this interface. The required connection are defined in Table 3-4 and Table 3-5.

Table 3-4: **Sample Receiver Interface for Single Phase Designs**

RXINPUTFORMAT	TXRXRATIO	srx_din(31:16)	srx_din(15:0)
Complex	1	Q0,Q1,Q2...	I0,I1,I2...
Complex	2	Q0,Q0,Q1,Q1...	I0,I0,I1,I1...
Real	1	S1,S3,S5...	S0,S2,S4...
Real	2	S0,S1,S2...	S0,S1,S2...

Table 3-5: **Sample Receiver Interface for Polyphase and Hybrid Designs**

RXINPUTFORMAT	TXRXRATIO	srx_din(63:48)	srx_din(47:32)	srx_din(31:16)	srx_din(15:0)
Complex	1	Q1,Q3,Q5...	I1,I3,I5...	Q0,Q2,Q4...	I0,I2,I4...
Complex	2	Q0,Q1,Q2...	I0,I1,I2...	Q0,Q1,Q2...	I0,I1,I2...
Real	1	S3,S7,S11	S2,S6,S10	S1,S5,S9	S0,S4,S8
Real	2	S1,S3,S5	S1,S3,S5	S0,S2,S4	S0,S2,S4

Additional Interfaces

When using the DPD IP core in a multi-antenna system, the core must be able to select observation data from among the different antennas. This can be done in two ways:

- Direct hardware control (SRX control stream)
- Software handshake

The direct hardware control stream is the preferred method when the antenna switch control is directly accessible to the device. If this is not the case, for example when the antenna control is provided by the application control plane, then a software handshake can be used. See Antenna Selection Options in a Multipath Installation for more information.

When using the software handshake the `m_axis_srx_ctrl_tready` should be tied High by the external system (see Figure 3-2). In this mode, the `m_axis_srx_ctrl` is not used by the DPD software but the port change request is echoed in this control interface along

with the host interface. The m_axis_srx_ctrl_tvalid can optionally be used to generate an interrupt for the system processor to indicate when a port change request needs to be serviced. An externally generated interrupt can be used to avoid requiring the system processor to continuously poll the host-interface.

SRX Control Stream

The SRX control stream (m_axis_srx_ctrl) is an optional AXIS interface used for direct hardware control of the antenna selection from the DPD software. All control requests are made automatically by the DPD software when new observation data is required.

The m_axis_srx_ctrl_tdata packs the antenna and request type request as follows:

Table 3-6: **SRX Field Packing**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Request type				Reserved	Antenna		

The request type values are defined in Table 3-7.

Table 3-7: **Request Types**

m_axis_srx_ctrl_tdata (7:4)	Request Type
0x0	Normal (forward/incident)
0x1	Hi-Resolution request when using certain types of export compliant ADCs.
0x2	Reflected signal (for VSWR power measurements)
0x3 ... 0xE	Reserved for future expansion
0xF	SRX available (unused by DPD)

Antenna Selection

The Figure 3-1 demonstrates the typical use case for a multi-antenna system using a standard ADC (hi-res flag always Low). The stream follows the standard AXI protocol and the antenna selection request is deemed to be accepted when tvalid and tready are both High. In the simplest case, the external hardware is always ready to accept an antenna change request and tready can simply be tied High by the external system. The DPD core presents the requested antenna number on tdata and asserts tvalid for a single cycle.

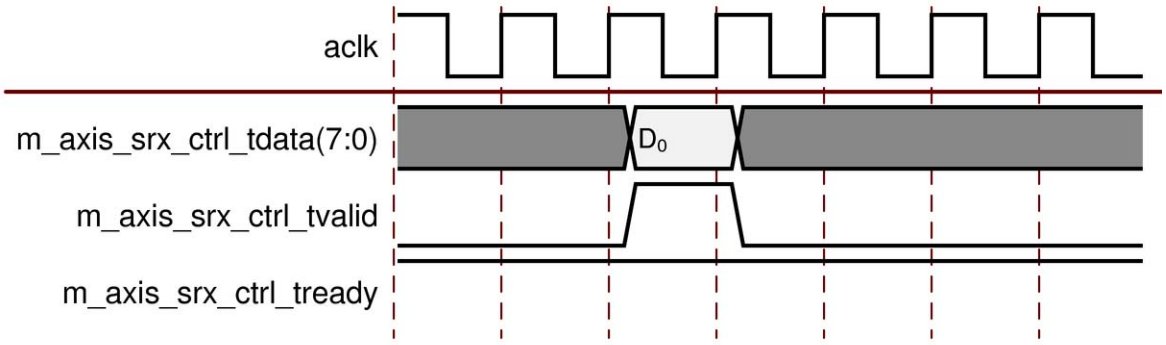


Figure 3-2: **m_axis_srx_ctrl Stream Behavior with tready Tied High**

In some circumstances, the external hardware might need to delay the acceptance of the request (possibly to meet some external system constraints). In this case, the external hardware can hold `tready` Low until it is ready to accept the antenna request and the DPD core leaves the data on the bus with `tvalid` remaining High. After the external hardware accepts the request, `tvalid` is dropped Low and `tdata` idles.

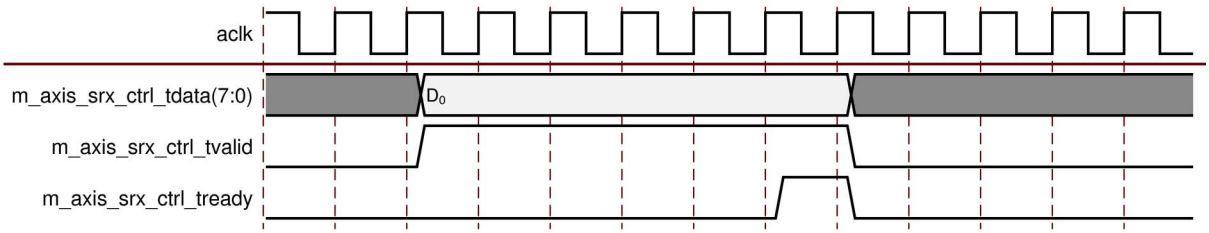


Figure 3-3: **m_axis_srx_ctrl Stream Behavior When an External System Holds off tready**

IMPORTANT: *In both cases, the antenna data is only presented for the period when `tvalid` is active. Therefore, if the external hardware requires a steady value (for example to drive a mux select) it must latch the `tdata` using `tvalid` as a strobe.*

Indicating the Active Antenna to the DPD Core Using SRX Control Stream

The DPD core needs to know when the antenna switch has completed and the SRX data is ready to be used in an update. When using the SRX control stream, there are two possible methods, configurable through the Host Interface by the `PORTCONTROL` parameter.

- Programmable software delay
- Append antenna information to `s_axis_srx_tuser`

Programmable software delay uses a simple programmable delay named `SRXSELECTDELAY`, which causes the DPD core to wait for a period of time after the antenna switch request before starting to use the data in an update. See Updating and Reporting DPD Parameters in Chapter 6 for more information.

Appending the antenna information to s_axis_srx_tuser is an optional enhancement where information is appended to s_axis_srx marking the antenna to which the data applies. If the switching time of the antenna selection is well known, this can provide a more precise way of indicating when the new antenna data is valid. To use this method the external hardware should add the antenna number to the s_axis_srx_tuser stream. This might require some number of cycles after the request is accepted to allow for any multiplexing switching delay or transients to settle (if required). See the following timing diagram which shows an antenna being requested through m_axis_srx_ctrl_tdata and some period of time later (after the switch has occurred) the s_axis_srx_tuser stream being updated to show the new antenna.

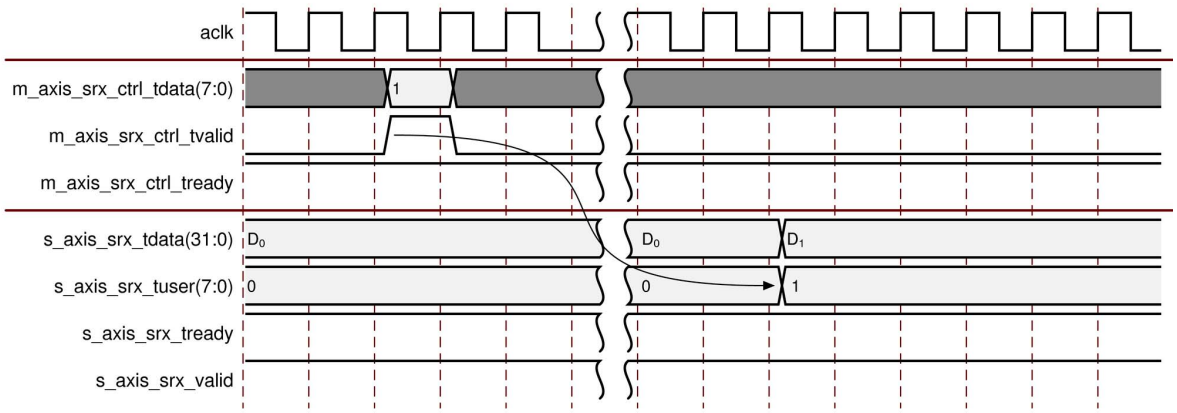


Figure 3-4: **Using s_axis_srx_tuser to Indicate Active Antenna**

When enabled using the PORTCONTROL parameter, this stream is monitored by the software to check that the requested antenna number has been presented on tuser before the stream is used for a capture.

Export Compliant ADCs

The core provides support of certain types of export compliant ADCs. These are devices that can only provide full resolution data for some period of time before being 'locked out' at a lower resolution such that the average resolution is reduced. To use this kind of ADC in an observation receiver, the core must be able to request high resolution data, and the incoming SRX stream must indicate when this is the case. This is done using the SRX control stream.

The following diagram shows m_axis_srx_ctrl used to request high resolution data. The core asserts of m_axis_srx_ctrl_tdata(4) along with tvalid. This is shown as setting the Request Type to 0x1 in Table 3-5. At some point in time later the ADC is able to output high resolution data and this is indicated to the core using s_axis_srx_tuser(4). The DPD core optionally monitors s_axis_srx_tuser(4) during a capture to ensure high resolution data is present for the whole period. Monitoring is enabled using the HIRESMONITOR parameter.

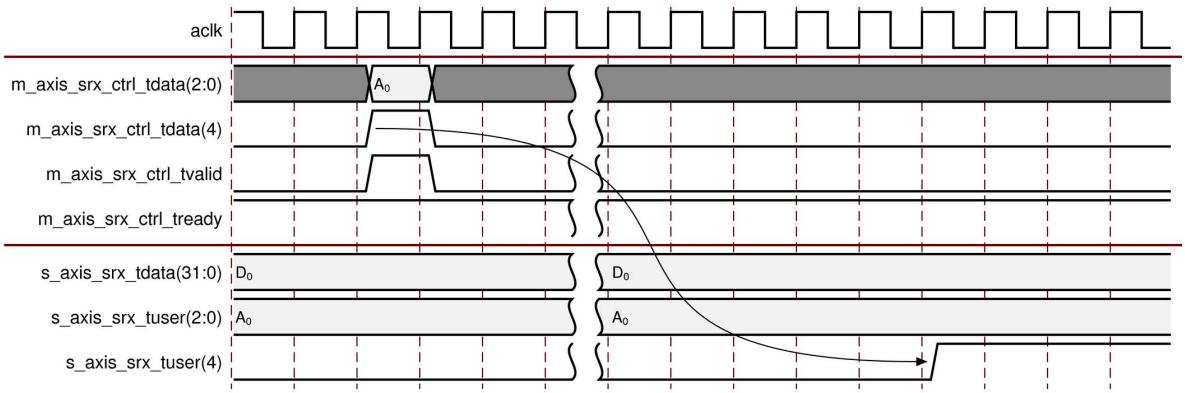


Figure 3-5: **Using s_axis_srx_tuser to Indicate a High Resolution Data Period**

IMPORTANT: *The current antenna is also repeated on tdata(2:0) when high resolution data is requested. This is necessary for AXI compliance; tdata must be correctly set when tvalid is asserted. The core however never requests a change of antenna and high resolution data simultaneously. Therefore it is safe to latch s_axis_srx_ctrl_tdata(2:0) with s_axis_srx_ctrl_tvalid to create an antenna multiplexing select signal as tdata(2:0) is guaranteed not to change due to a high resolution request.*

It is expected that a certain amount of "glue-logic" is needed to be implemented externally to connect the hi-res flag to the ADC, to handle cases where the core requests high resolution data during the lock out period. The specific implementation depends on the exact ADC used. The DPD control signals have been left as generic as possible to provide maximum flexibility.

Sharing the Observation Path with Non-DPD Functions

In many systems, it is desirable to share the DPD observation (or feedback) path with other non-DPD related functions (e.g. making VSWR measurements or RFIC calibration cycles). The Xilinx DPD solution provides two general methods for sharing access to the observation path. The first method assumes that the Xilinx DPD is the master and will always have highest priority to the observation path. The second method assumes that the Xilinx DPD has low priority access (i.e. the observation path can be taken away for other purposes at any time). An example of the `m_axis_srx_ctrl` while running the DCL routine with ENABLEVSWRPWRMSR enabled and PORTCONTROL set to zero is shown in Figure 3-6.

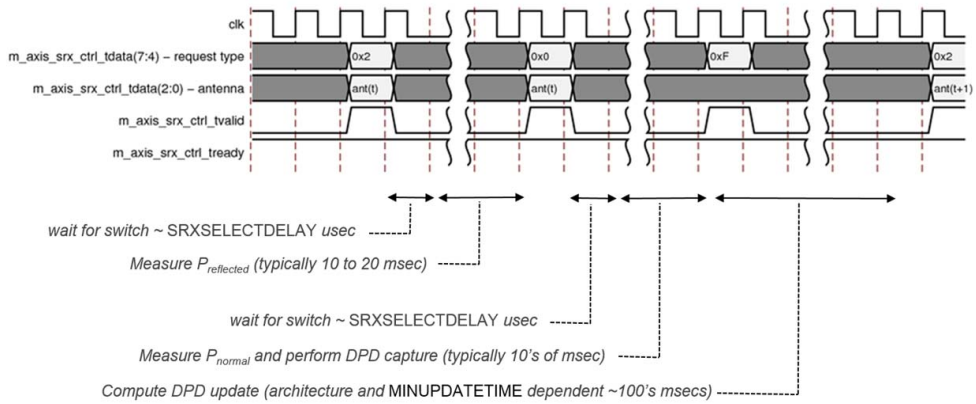


Figure 3-6: **Sharing the Observation Path with DPD as the master**

When Xilinx DPD is the master, the request type and antenna fields in the SRX control stream (`m_axis_srx_ctrl`) are used to indicate the required state of the observation path. The external system configures the observation path in accordance with the request and antenna and then acknowledge back to DPD when ready.

Additional information must be supplied to the Xilinx DPD core, using the `s_axis_srx_tuser` stream, when it is operating in a system with low priority access to the observation. The s_axis_srx_tuser signal is described inTable 3-8. This information is used by DPD to discard captures that do not contain the desired captures and to report only valid SRX power measurements in the DCL monitor.

Table 3-8: **s_axis_srx_tuser for low priority access**

bit	Name	Description
7	ONA	Observation not available. sample is masked. `s_axis_srx_tdata` shall be set to 0 on any cycle where ONA=1.
6:5	Reserved	set to zero

Table 3-8: **s_axis_srx_tuser for low priority access *(Cont'd)***

bit	Name	Description
4	Hi-Res Marker	Set to 1 when hi-resolution requested is granted after REQ_TYPE=1 on `m_axis_srx_ctrl`[7:4]. Always reset to 0 immediately when ONA occurs regardless of value on REQ_TYPE.
3	Reserved	set to zero
2.0	Antenna	Antenna Number

In this approach, the external system must communicate to the DPD core when the observation path is not available (ONA). A value of 1 for ONA indicates that the observation path is not available for DPD usage. The operation is as follows:

typo "m_axis_srx_ctrl_tdata"

1. DPD is configured to use the hi-resolution capture controls (see HIRESMONITOR). In this mode the DPD core will set the request type on `s_axis_srx_ctrl_tdata` to **b0001** at the beginning of every capture attempt. At the end of each capture, the DPD core monitors `s_axis_srx_tuser`(4) to see if the observation was good during the entire capture. This configuration allows the DPD to quickly discard any captures that cannot be used for DPD updates.
2. At the beginning of each measurement interval, the DPD core clears a flag that is controlled by `s_axis_srx_tuser`(7). This flag will latch any high signal seen on `s_axis_srx_tuser`(7). This flag is not cleared automatically. At the end of each measurement interval, the flag is monitored. If the flag is low, then the measurement is good and can be reported as a valid observation path power measurement. If the flag is high, the measurement is discarded and the reported power is set to zero.
3. Zeroing the `s_axis_srx_tdata` when the observation path is not available allows for an additional fast checks by the DPD core to see if the DPD updates should be attempted.

Host AXI4-Lite Interface

This interface provides settings and instructions to the ARM processor about the functions it needs to perform. This interface uses a 1024x32 block RAM in dual-port mode. The AXI4-Lite slave interface gives full access to one port of this memory. See the *AMBA AXI4-Stream Protocol Specification* [Ref 3] for more information on the AXI4 and AXI4-Lite ports and operation.

Host Interface Registers provides the detail and usage of the memory map of this interface.

Because there is only a single port of the block RAM available for accesses, write and read operations cannot take place simultaneously. The internal logic between the AXI4-Lite interface and the block RAM arbitrates, giving read accesses priority and therefore throttling write address/data, where required. If either write or read responses are not read (ready asserted) then after approximately eight responses of either operation have been buffered for output, the input of the corresponding operation is throttled by the core.

It is possible to enter corresponding write address and write data out of sync; however, if full bandwidth (back-to-back access) is required, these should be written simultaneously; otherwise, either the address or data channel is throttled depending on which has been written already and which is still pending. When writing write address and write data out of sync, the write bandwidth is limited to a maximum of 50%. As read accesses take priority, these always sustain full bandwidth at the expense of stalling write accesses.

DPD AXI4-Lite Control Interfaces

The AXI4-Lite interface `s_axi_ctrl` enables communication between the DPD hardware components and the DPD software application running on the processor subsystem. This bus must be connected to the processor subsystem. The `s_axi_ctrl` interface has the ability to take control of the host interface when accessed by the DPD Debug Interface (see UG989). In such cases, the value returned by reads of the host interface from `s_axi_user` interface will be zero.

The address space required for the DPD Control Interface is 26-bits or 0x4000000 bytes. This can be located on any suitable PL-PS address space within the Zynq or Zynq MPSoC Device. Further restrictions may apply depending on the choice of DPD Software. For details on the memory map, see Application Software Section.

The details of register mapping within the DPD IP Core are internal to the solution and are not detailed in this document. Accesses to unmapped address regions within the `s_axi_ctrl` address space result in decode errors (DECERR) being returned on `s_axi_ctrl_bresp`, that may be reported as Data Aborts. However, they will not cause failure of the core.

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 6]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 8]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 9]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado Design Suite.

RECOMMENDED: *The recommended design flow for instantiating DPD IP Core within a design is to use Vivado IP Integrator. See Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator (UG994)[Ref 6] for more information. IP Integrator simplifies the process of integrating IP and manages the address spaces and bus connections in a Zynq or Zynq MPSoC Processing Subsystem. A customer reference design using Vivado IP Integrator is available on the Digital Pre-Distortion Evaluation Lounge, and is the recommended starting point for designs.*

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Create or open a Vivado RTL project. Ensure that the part selected is one of the parts listed in “Device, Package and Speed Grade Selections”.
2. Unzip the DPD v8.1 archive. Click **Settings** in the Vivado IDE and select the **IP** icon. Use the **Add Repository...** button to add the `vivado` sub-directory of the archive.
3. Select **Create Block Design** under **IP Integrator** in Flow Navigator panel to create a new block diagram.

4. Right-click in the Block Diagram and **Add IP..** from the IP Catalog, The Digital Pre-Distortion core is listed under **Communications and Networking/Wireless**.
5. Double-click on the core to customize the Core.
6. A new window, **Customize IP,** shows various customizable parameters of the IP for DPD v8.1. These parameters are explained in the following sections. See Figure 4-1. A similar customization window is displayed, if the IP Core is created in an RTL design.

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE).This layout might vary from the current version.

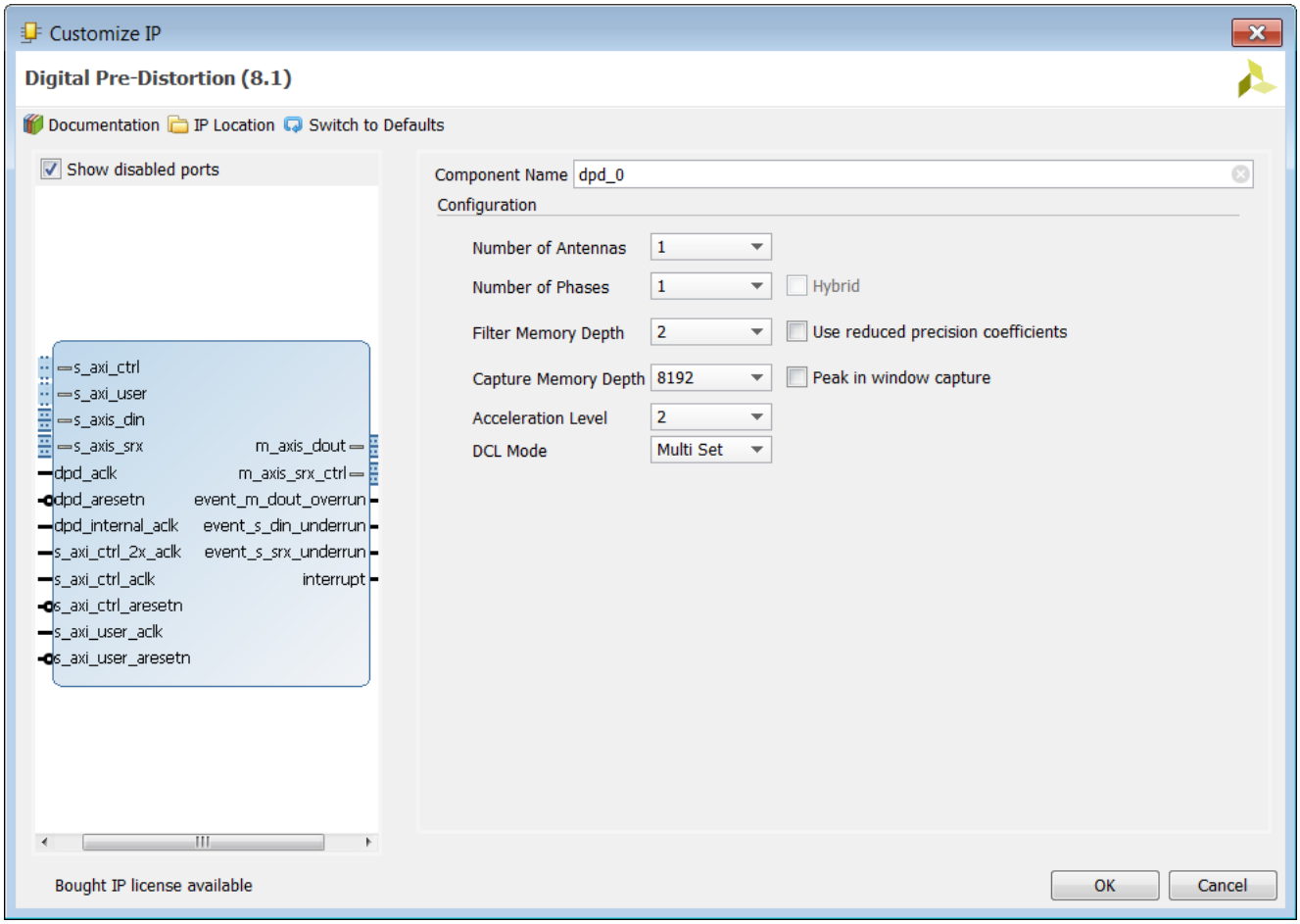


Figure 4-1: **Customize IP**

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7] and the "Working with the Vivado IDE" section in the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 8].

Component Name

The component name is used as the base name of the output files generated for the core. In Vivado IP Integrator, the component name is taken directly from the block diagram and is not available in the customization GUI.

Note: Names must begin with a letter and must be composed from the following characters: a through z, 0 through 9 and "_" (underscore).

Number of Antennas

Number of Antennas processed by the DPD core. The DPD v8.1 core supports 1, 2, 4, 6 or 8 antennas. The width of streams carrying antenna data (`s_axis_din` and `m_axis_dout`) scale with this parameter.

Number of Phases

The DPD core can be configured to accept either one or two complex samples per clock cycle. The two phase case allows the core to run at half the clock frequency but increases area usage.

Depending on the selected filter configuration, the width of buses (`s_axis_din`, `s_axis_srx`, and `m_axis_dout`) scale appropriately.

Hybrid

If the core is configured for two-phase input, a hybrid option becomes available. In this mode, two samples are presented on each clock cycle, but internally the filter uses the second clock to process single samples at double the interface rate. This offers some of the benefits of improved timing closure as with the standard two-phase case, but with only a small increase in area usage over the single-phase case.

Filter Memory Depth

Controls the size of the DPD filter. An increased depth can improve the core performance at the expense of increased update time and extra area.

Capture Memory Depth

This parameter controls the depth of the capture memory. It can be set to 8,192 or 16,384 and can be selectively set to 4,096, if the number of phases is 1. These reduced capture depths can be leveraged for low power PAs and small signal bandwidth cases.

Use Reduced Precision Coefficients

This parameter configures DPD filter coefficients with reduced precision. This helps reduce FPGA resources like Block RAMs and DSP48s with a potential performance trade-off. Default (with check-box unchecked) results in the use of full precision coefficients.

Peak in Window Capture

This parameter adds a hardware capture mechanism capable of capturing the maximum signal peak observed over an interval much longer than the depth of a capture is desired. This mode can be optionally turned off which results in resource savings.

Acceleration Level

This parameter controls the number of hardware engines used for coefficient computation. The default Acceleration Level shown in Vivado IDE is a function of the selected Filter Memory Depth. Selecting lower Acceleration Level values saves device resources with reduced coefficient update times. It is generally not beneficial to select Acceleration Level values higher than the default.

DCL Mode

Dynamic Control Layer (DCL) Mode allows storage of different sets of DPD coefficients which can be internally used by the software to track different signal conditions and apply those coefficients for those signal states. More details about the DCL Algorithm is in section Dynamic Control Layer (DCL) in Chapter 6.

- Single Set: Only a single set of coefficients is available in DPD hardware and used by DPD software.
- Multi-Set: Multiple sets of coefficients are available in DPD hardware and DPD software can take advantage of these multiple sets of coefficients to independently track signal states. DPD software also has a runtime parameter to force its usage to be single set DCL mode (see Table 6-8).

User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value[1]	User Parameter/Value[2]	Default Value
Number of Antennas	NUM_ANTENNAS	1
Number of Phases	NUM_PHASES	1
Filter Memory Depth	MEMORY_DEPTH	2
Hybrid	HYBRID	FALSE
Capture Memory Depth	CAPTURE_DEPTH	2
Use Reduced Precision Coefficients	REDUCED_PRECISION	FALSE
Peak in window capture	CAPTURE3	TRUE
Acceleration Level	NUM_VW_CENG	2
DCL Mode	DCL	Multi Set

Notes:

1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.
2. Provided for information only, cannot be changed.

Output Generation

For details, see "Generating IP Output Products" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

This section describes the constraints within the XDC file:

- In the design, there are two synchronous clock pairs `dpd_aclk`/`dpd_internal_aclk` and `dpd_ctrl_aclk`/`dpd_ctrl_2x_aclk`; all paths between the two domains that form a pair are timed due to their known phase and frequency relationship inferred through the correct generation of these clocks.
- Between the synchronous pairs and `s_axi_user_aclk`, everything is assumed to be asynchronous with appropriate clock-domain crossing logic between logic in the different clock zones. These clock-crossing paths are false-pathed in the `<component_name>_ooc.xdc` file. Similar constraints should be used in a customer design. If the asynchronous domain assumptions are not valid in the customer design, appropriate modifications to these example files should be made.

See Clock Frequencies for details on the recommended clock constraints.

Device, Package, and Speed Grade Selections

The DPD core is designed to run on Zynq-7000/Zynq UltraScale+ devices.

Clock Frequencies

The recommended maximum clock frequencies are shown in IP Timing Performance.

Clock Management

See Reset and Clock Sequencing for more details on clock management.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

This core supports simulation and a demo test bench can be generated for the core, See Test Bench for more information,

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation*(UG900)[Ref 9].

Synthesis and Implementation

This IP core will work with the standard Synthesis and Embedded Design Implementation flow. For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7].

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.

Demonstration Test Bench

When the core is generated using the Vivado IDE, a demonstration test bench can be created by using the following steps:

1. Right-click on the core name in **Project Manager > Sources**.
2. Select **Generate Output Products**.
3. In the **Manage Output Products** dialog box, select **Generate** in the Test Bench drop-down menu.

This is a simple VHDL test bench that exercises the core. The demonstration test bench source code is one VHDL file: `demo_tb/tb_<component_name>.vhd` in the Vivado output directory. The source code is comprehensively commented. The test bench is intended to show the DPD command shell handshake process [Refer Table 6-15 DPD Command Shell] between user interface and control interface. Simplified bus functional models are used to show transactions occurring on these interfaces. In addition to this the test bench shows behavior of clocks and reset along with behavior of the core when configured as pass through. It also calculates the input to output latency of the core and echoes it on the screen.

Using the Demonstration Test Bench

After the output products of the core are generated the test bench should be added by right clicking on Simulation Sources and choosing Add Sources. Navigate to the demo_tb folder and the file `tb_<component_name>.vhd`. After this, the behavioral simulation can be launched from **Flow Navigator -> Simulation -> Run Simulation**.

If target simulator is set to Vivado then this will launch the native Vivado Simulator. For other simulators compile the Xilinx libraries as instructed. After the test bench simulates view the test bench signals in the waveform viewer of your simulator to see the operations of the test bench.

More information on the demonstration test bench is available in the tb_readme.txt file that is generated as part of the test bench.

Application Software

The DPD Application Software is supplied in a separate archive. Contact Xilinx Support for gaining access to this archive. The archive contains the pre-compiled and linked DPD Application for a number of different application environments. Table 6-1 displays this list of application software. For bare-metal and AMP Configurations, ELF files are supplied for the DPD application. For Linux SMP configurations, executables are supplied. All Zynq-7000 ELF Files and executables are 32-bit. Zynq-UltraScale+ MPSoC ELFs and executables are 64-bit.

Table 6-1: **Application Software**

Application Software Configuration	Software Files
Zynq-7000 Single-Processor with Bare-Metal Configuration	dpd_a9_0_baremetal.elf
Zynq-7000 Dual-Processor with AMP Configuration	dpd_a9_1_amp.elf
Zynq-7000 Dual-Processor with SMP Configuration	dpd_smp
Zynq-UltraScale+ Single-Processor with Bare-Metal Configuration	dpd_a53_0_baremetal.elf
Zynq UltraScale+ MPSoC: SMP Configuration	dpd_smp

For more information, see the *Zynq-7000 All Programmable SoC Software Developers Guide User Guide (UG821)* [Ref 10] and the *Zynq UltraScale+ MPSoC Software Developers Guide (UG1137)[Ref 17]*.

Zynq-7000 Single-Processor Bare-Metal Configuration

In Zynq-7000 single-processor bare-metal configuration, the DPD application software runs in CPU0 of the Zynq-7000 PS. The second CPU1 of the Zynq-7000 PS is inactive. Figure 6-1 shows the DPD application in single-processor configuration.

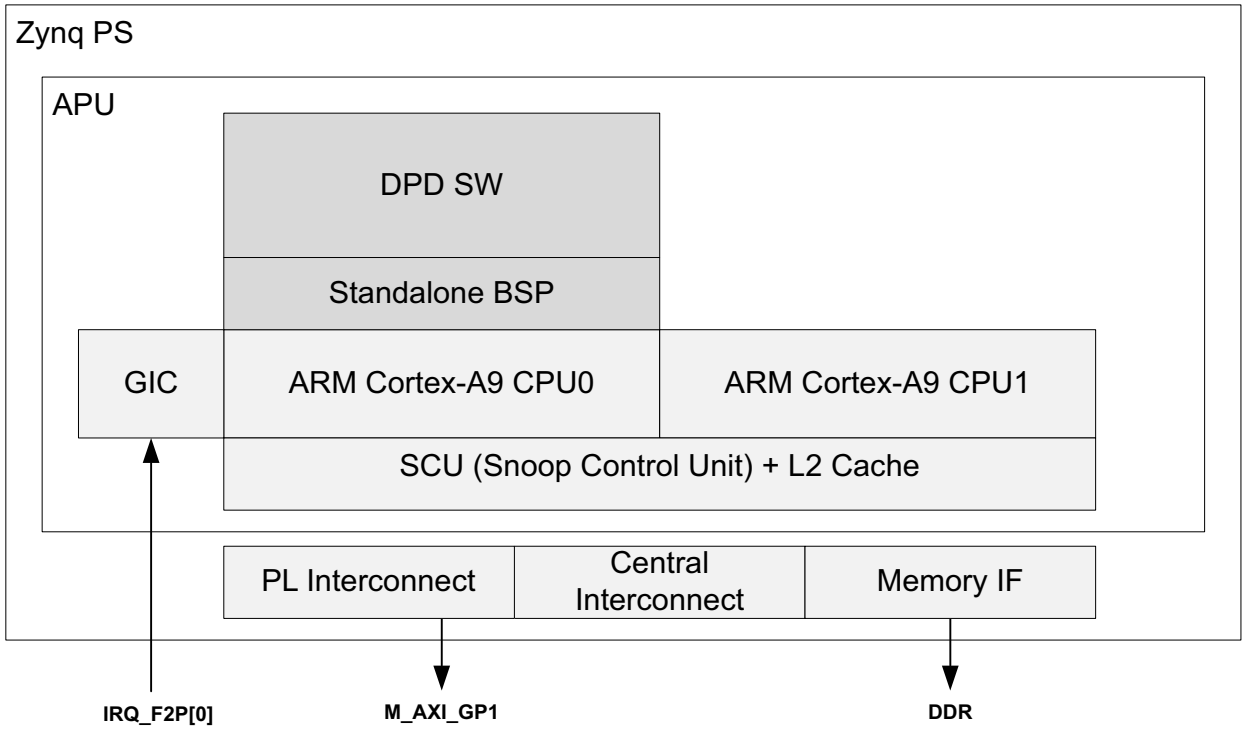


Figure 6-1: **DPD SW in Zynq-7000 Single Processor Bare-Metal Configuration**

Memory Map

The DPD software application is pre-compiled and linked for execution using the address ranges shown in Table 6-2.

Table 6-2: **Single-Processor Memory Map**

Section	Address Range	Notes
Program and Data, Heap, Stack	0x00100000 - 0x00FFFFFF	DPD execution stating address at 0x00100000
DPD Hardware Access on GP1(1)	0x80000000 - 0x83FFFFFF	Used only for internal communication between DPD software and DPD hardware components

Notes:

1. GP stands for Zynq-7000 PS General Purpose AXI Port. See the *Zynq-7000 All Programmable SoC Software Developers Guide User Guide (UG821)* [Ref 10].

Although the footprint of the DPD software might not exactly occupy the entire range of addresses for Program and Data, Heap, and Stack, it is recommended to allocate the indicated range to accommodate possible variations in code/data size in software updates.

Software Boot Process

The DPD application software ELF must be included in the boot process of the Zynq-7000 PS along with any other applications and OS. The specifics of the boot process depend on the boot mode, memory device, and OS to be used in a particular system. Refer to the *Zynq-7000 All Programmable SoC Software Developers Guide User Guide* (UG821) [Ref 10] for details of the different options to boot the Zynq-7000 PS.

The following steps describe the procedure to boot from Secure Digital (SD) RAM as an example:

1. Create a First-Stage Boot-Loader (FSBL) application in the software development kit (SDK). Use the following procedure to create this package:
   a. Select **File**->**New**->**Application Project**.
   b. Name your project. For example “dpd_fsbl”.
   c. Select **Create New** for Board Support Package.
   d. Click **Next**.
   e. Select the **Zynq FSBL** template.
   f. Click **Finish**.

   This creates the board support package required and uses this to create the FSBL executable elf file.
2. Select the FSBL project in the Project Explorer in SDK.
3. Select **Xilinx Tools**->**Create Zynq Boot Image**. The form that shows should already have a boot image partition defined with the FSBL elf.
   a. Add a new partition by clicking **Add** and then browsing to the bit-file to include in the boot image. This should contain the DPD hardware design and any other customer specific design.
   b. Add a new partition by clicking **Add** again and browsing to the DPD elf file that is provided with our IP core.
   c. Click **Create Image** to generate an output.bin file (by default) at the location specified by the 'Output path' selection at the bottom of the boot image form.
   d. Copy the `output.bin` file to a SD-RAM card and rename it BOOT.BIN.

With this, the SD-RAM card can be used to boot the DPD system.

Zynq-7000 Dual-Processor AMP Configuration

In Zynq-7000 Dual-Processor AMP configuration, the DPD software runs in CPU1 of the Zynq-7000 PS as a bare-metal application in an Asymmetric Multi-Processor (AMP) arrangement with CPU0. The following assumptions are made in AMP Configuration:

- CPU0 is the master in the AMP configuration.
- Software applications, and possibly an Operating System (OS), that are running in CPU0 are fully isolated in memory from the address range used by CPU1.
- CPU0 configures the ZYNQ APU Generic Interrupt Controller (GIC) to enable forwarding of interrupts to the CPU's.

An application running in CPU0 (for example, an Operations and Maintenance O&M application) can be used to control DPD. Figure 6-2 shows the DPD application in dual-processor configuration.

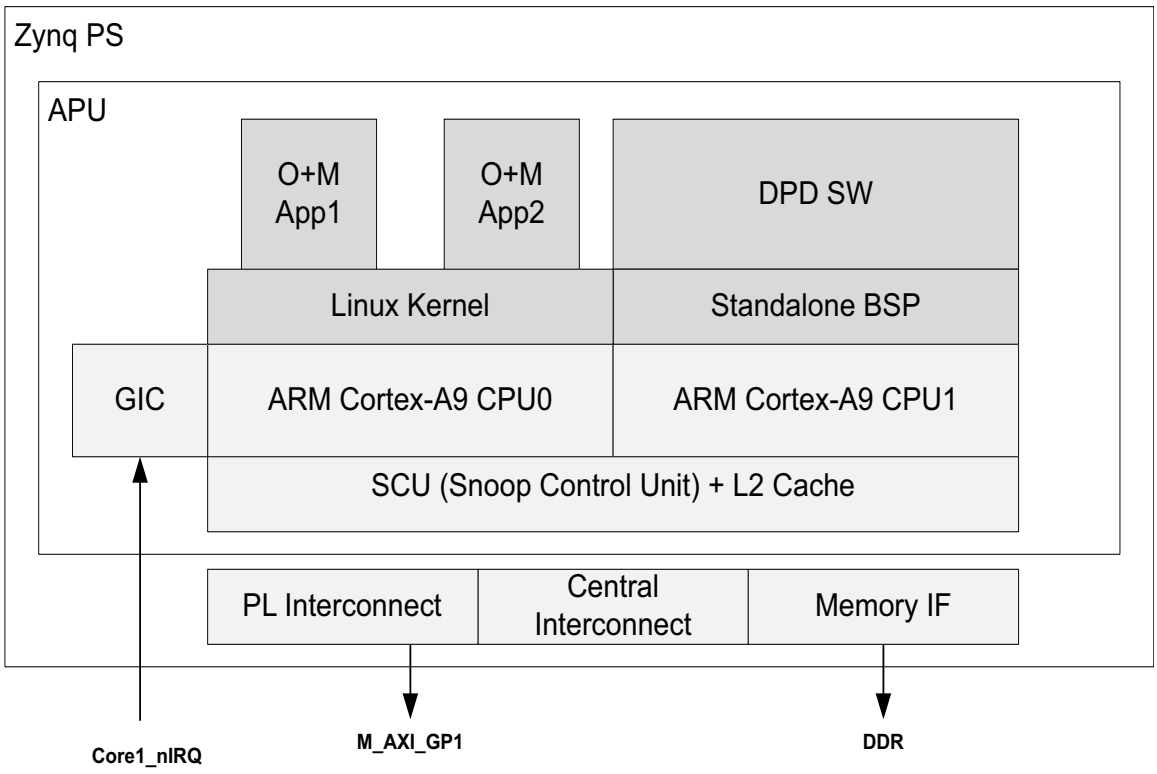


Figure 6-2: **DPD SW in Zynq-7000 Dual Processor AMP Configuration**

Memory Map

The DPD software application is pre-compiled and linked for execution using the address ranges shown in Table 6-3 in the dual-processor configuration.

Table 6-3: **Dual-Processor Memory Map**

Section	Address Range	Notes
Program and Data, Heap, Stack	0x00100000 - 0x00FFFFFF	DPD execution stating address at 0x00100000 in CPU1.
Non-DPD application(s)	0x01000000 - ...	Available for application(s) running in CPU0. Upper limit of range depends on memory size.
DPD Hardware Access on GP1(1)	0x80000000 - 0x83FFFFFF	Used only for internal communication between DPD software and DPD hardware components

Notes:

1. GP stands for Zynq-7000 PS General Purpose AXI Port. See *Zynq-7000 All Programmable SoC Software Developers Guide User Guide (UG821)* [Ref 10].

Although the footprint of the DPD software does not occupy the entire range of addresses for Program and Data, Heap, and Stack, it is recommended to allocate the indicated range to accommodate possible variations in code/data size in software updates.

Zynq-7000 Dual-Processor Linux SMP Configuration

In dual-processor configuration with Linux SMP, the Linux SMP Kernel runs on both CPU0 and CPU1. Both the DPD Software and Customer Overhead and Management (O+M) Software run on Linux SMP Kernel. They can be scheduled to run on either of the CPUs.

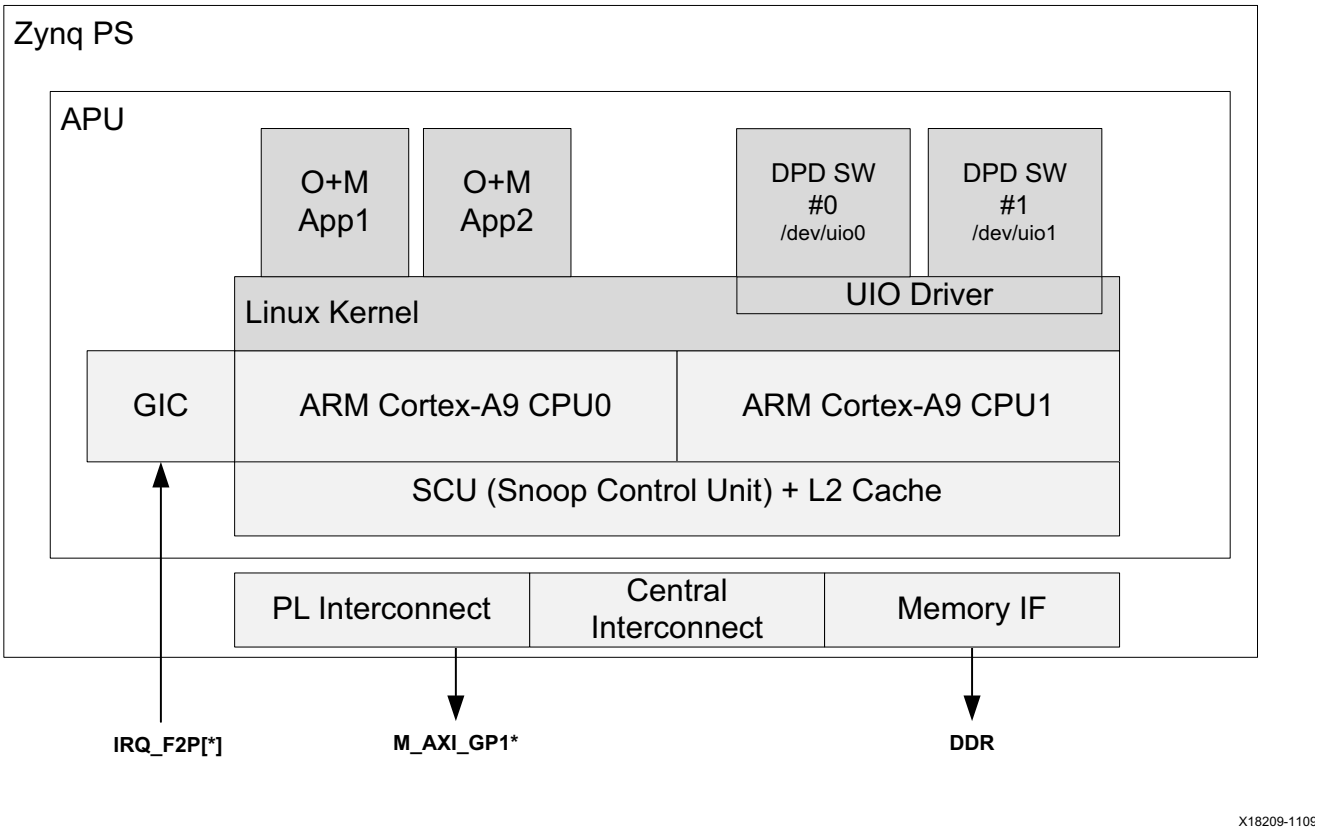


Figure 6-3: **Zynq-7000 Dual Processor Linux SMP Configuration**

The following assumptions are made in the Zynq-7000 Linux SMP Configuration:

- **Interrupts**: The DPD Interrupt(s) are connected to one of the Shared Peripheral Interrupts (SPI) connecting Programmable Logic (PL) to Processor System (PS). See Chapter 7 of *Zynq-7000 Technical Reference Manual (UG585)*[Ref 13] for more details on interrupts. The choice of interrupt used is determined by the linux device-tree and derived from the hardware configuration exported from Vivado.
- **Memory Map**: The recommended memory map for running is shown in Table 6-4. The Linux Kernel is in charge of the whole main DDR memory space. DPD Software can use any valid PS to PL Master Interface to communicate to control the DPD IP Core.
- **UIO Drivers**: The SMP Linux version of the DPD Software uses the UIO (Userspace IO) drivers to access the DPD Hardware. These drivers allow the DPD Hardware to be mapped into userspace within the Linux Kernel and provide support for controlling the DPD Interrupt.

Table 6-4: **Zynq-7000 Dual Processor Memory Map**

Section	Address Range	Notes
SMP Linux Physical Memory	0x00000000-0x3FFFFFFF	Maximum Address Space for DDR
DPD Hardware Access	Address determined by chosen PS-PL Master Interface (M_AXI_GP*).	See Chapter 4 of the *Zynq-7000 AP SoC Technical Reference Manual (TRM)(UG585)*[Ref 13] for system addresses.

Support is provided for multiple instances of DPD IP Core. For each instance of the DPD IP Core in the Programmable Logic (PL), one UIO device driver is created in the linux `/dev` area. Command line options on the `dpd_smp` software application can be used to determine which UIO device driver is connected to e.g. `dpd_smp -u 1` to connect to `/dev/uio1`.

Zynq UltraScale+ MPSoC Single Processor Bare-Metal Configuration

In the Zynq UltraScale+ MPSoC Single Processor Bare-Metal Configuration, DPD Software runs entirely on CPU0 with the other processor as inactive. The bare-metal configuration is largely provided for comparison purposes with Zynq-7000 and early stage system debug, and it is expected that the customers will migrate to a Linux SMP Implementation to exploit all the available processors on the Zynq UltraScale+ MPSoC Device.

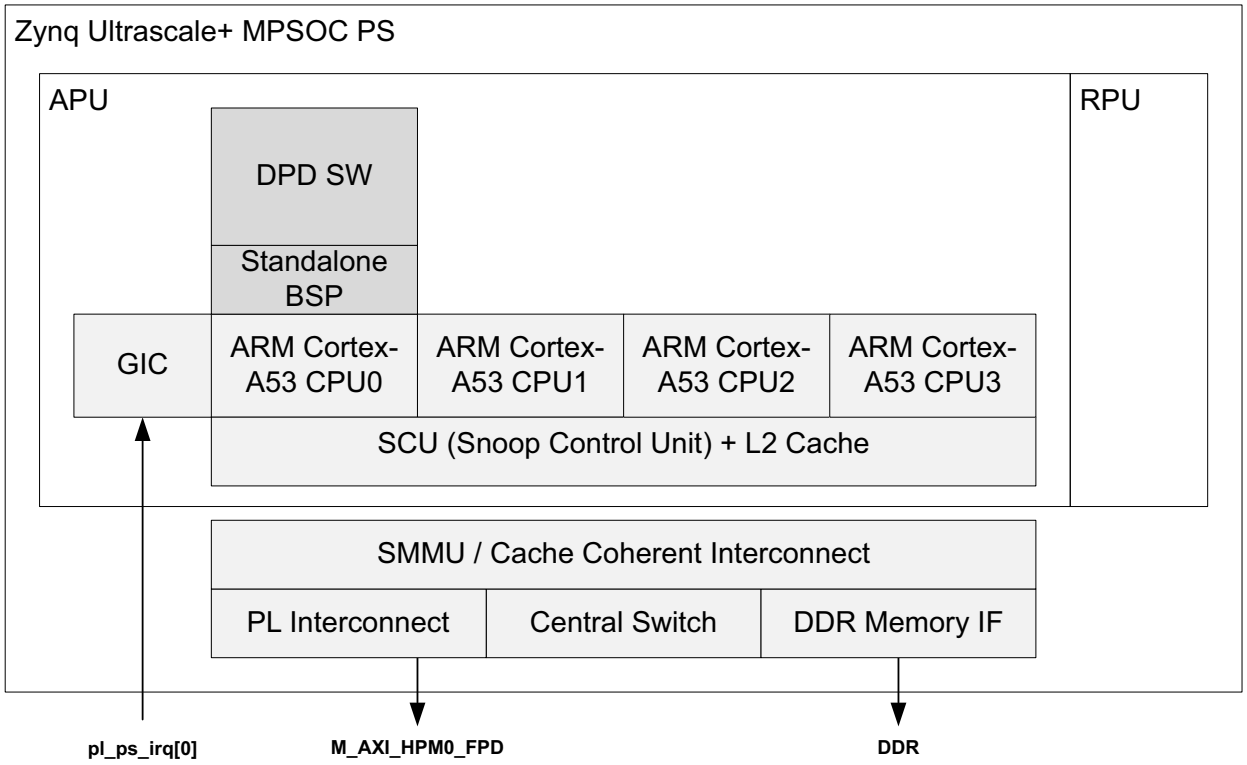


Figure 6-4: **Zynq UltraScale+ MPSOC Single Processor Bare-Metal Configuration**

The following assumptions are made in the Zynq-UltraScale+ MPSoC Single Processor Bare-Metal Configuration:

- **Interrupts**: the DPD Interrupt is connected to bit zero of the Shared Peripheral Interrupts (SPI) connecting Programmable Logic (PL) to Processor System (PS) - termed pl_ps_irq[0]. See *Zynq-UltraScale+ MPSoC Technical Reference Manual (UG1085)*[Ref 18] for more details on interrupts.
- **Memory Map**: the memory map for running is shown in Table 6-5. All the space allocated for Program, Data, Heap and Stack may not be used. However, this is a recommended range to allow for future variations in DPD Program Code and Data Size.

Table 6-5: **Zynq UltraScale+ MPSoC Single Processor Bare-Metal Memory Map**

Section	Address Range	Notes
Program, Data, Heap and Stack	0x00100000–0x00FFFFFF	DPD execution starting address at 0x00100000
DPD Hardware Access	0xA0000000–0xA3FFFFFF	Address Fixed by software compilation. Maps to address space of M_AXI_HPM0_FPD connection to Programmable Logic.

Zynq UltraScale+ MPSOC Multi-Processor Linux SMP Configuration

In multi-processor configuration with Linux SMP, the Linux SMP Kernel runs on all CPUs with the Zynq UltraScale+ APU. Both DPD Software and Customer Overhead + Management Software run on Linux SMP Kernel, and can be scheduled to run on any of the CPUs.

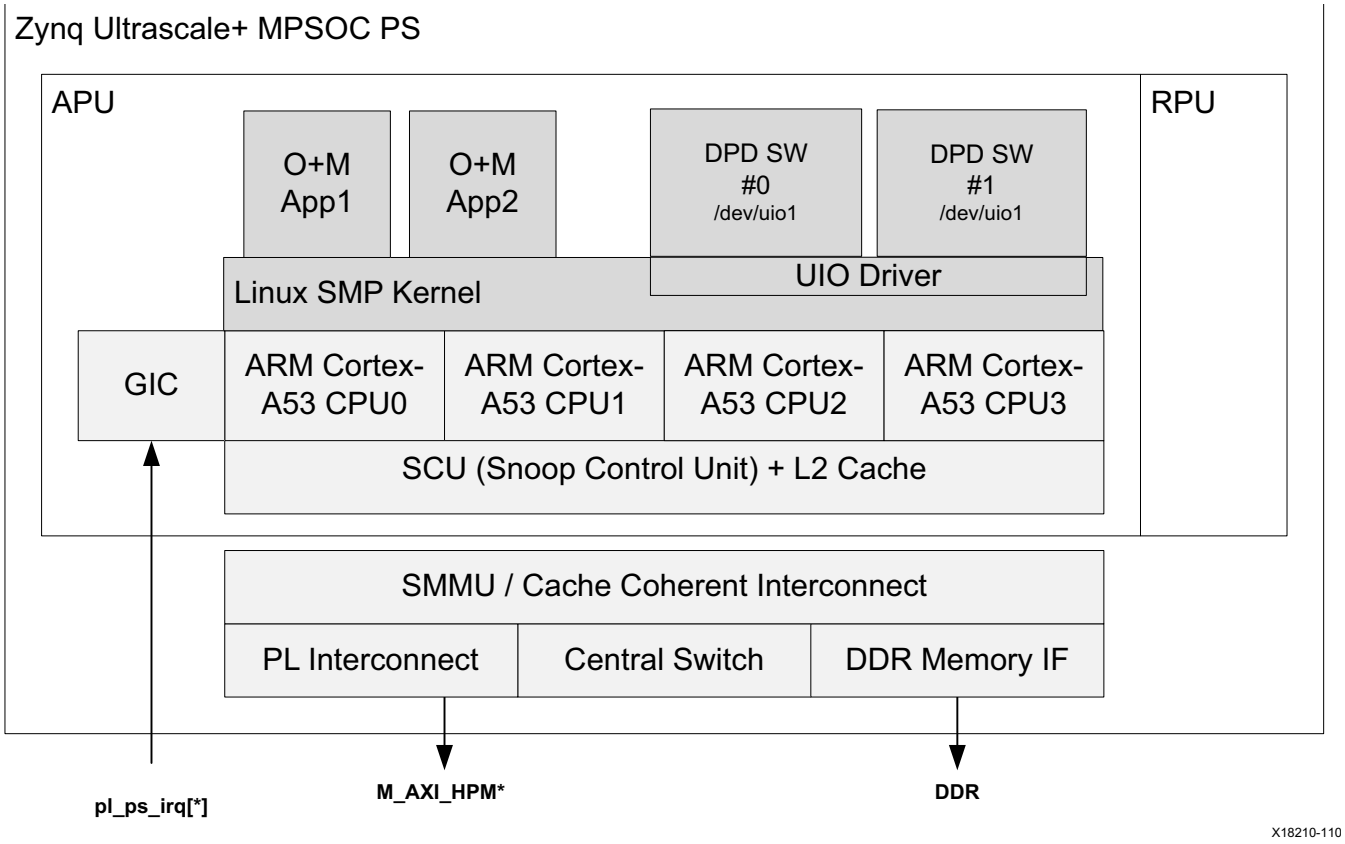


Figure 6-5: **Zynq UltraScale+ MPSOC Multi-Processor Linux SMP Configuration**

The following assumptions are made in the Zynq UltraScale+ MPSoC Linux SMP Configuration:

- **Interrupts**: The DPD Interrupts are connected to one of the Shared Peripheral Interrupts (SPI) connecting Programmable Logic (PL) to Processor System (PS). See *Zynq-Ultrascale+ MPSoC Technical Reference Manual (UG1085)*[Ref 18] for more details on interrupts. The choice of interrupt used is determined by the linux device-tree and derived from the hardware configuration exported from Vivado.

- **Memory Map**: The recommended memory map for running is shown in Table 2. The Linux Kernel is in charge of the whole main DDR memory space. DPD Software can use any valid PS to PL Master Interface to communicate to control the DPD IP Core.
- **UIO Drivers**: The SMP Linux version of the DPD Software uses the UIO (Userspace IO) Drivers to access the DPD Hardware. These drivers allow the DPD Hardware to be mapped into userspace within the Linux Kernel and provide support for controlling the DPD Interrupt.

Table 6-6: **Zynq UltraScale+ MPSoC Multi-Processor Linux SMP Memory Map**

Section	Address Range	Notes
Program, Data, Heap and Stack	0x00000000-0x7FFFFFFF	Lower Address Space for DDR. Exact Space determined by memory available
DPD Hardware Access	Any valid address in ranges supported by PS-PL Master Interfaces on Zynq-UltraScale+ devices.	See Chapter 10 of *Zynq-UltraScale+ MPSoC Technical Reference Manual* (UG1085)[Ref 18]for system addresses.

Support is provided for multiple instances of DPD IP Core. For each instance of the DPD IP Core in the Programmable Logic (PL), one UIO device driver is created in the linux `/dev` area. Command line options on the `dpd_smp` software application can be used to determine which UIO device driver is connected to e.g. `dpd_smp -u 1` to connect to `/dev/uio1`.

Creating Linux Images with DPD

This section provides an example for creating a basic Petalinux image to support operation with DPD for AMP and SMP configurations. The basic steps for AMP / SMP are the same. However, the differences exist in how some of the Linux configuration files are setup.

This example is based on Petalinux 2017.2 version tools. For more information on Petalinux commands and workflow, refer to the *Petalinux Tools Documentation Reference Guide* [Ref 16].

Note: Petalinux Tools have moved to using Yocto Tools to generate Linux Images. Hence, instructions referred to the Petalinux Documentation would have changed when moving from versions prior to 2017.2.

Table 6-7: **Creating Linux Images**

Step	Command/Description
1. Create Hardware Platform	Later Steps require the hardware definition for your design exported from Vivado. This hardware platform must meet the minimum configuration requirements to support Linux as described in the "Create Hardware Platform with Vivado" section in Petalinux Tools Documentation Reference Guide [Ref 16].
2. Export Hardware Platform	Run the Export Hardware command in Vivado as described in Export Hardware Platform to Petalinux Project in Petalinux Tools Documentation Reference Guide [Ref 16]. This will generate a directory containing the hardware definition file. (.hdf)

3. Create Petalinux Project	A Petalinux Project can be created from Petalinux Board Support Package (BSP) which is available for many Xilinx Development Boards.
	`petalinux-create -t project -s <BSP_FILE> -name <PROJECT_NAME>`
	If a BSP is not available, then a project can be created by CPU Type using
	`petalinux-create -t project --template <CPU_TYPE> -name <PROJECT_NAME>`
	where CPU_TYPE is **zynq** for Zynq SoC and **zynqMP** for Zynq MPSoC.

4. Add Applications to Root File system (Optional for AMP)	Depending on the system environment, you may want to add other application programs to Linux. For SMP Linux, it is recommended to have the DPD Application, dpd_smp, in the root filesystem. To do this, follow the instructions under **Including Prebuilt Applications** in Petalinux Tools Documentation Reference Guide [Ref 16]. To run applications at the start up, refer to the section **Application Auto Run at Startup** in the same document.
5. Configure Petalinux	Run the following command to configure the Petalinux Project with the generated hardware platform: `petalinux-config --get_hw_description=<HW_PLATFORM_DIR>` where HW_PLATFORM_DIR is the path to the hardware definition directory generated above. Once the file is read, petalinux will enter into a configuration menu. See next steps for using these menus.

Table 6-7: ***(Cont'd)*Creating Linux Images**

Step	Command/Description

6. Configure Boot Arguments	Perform the following under **Kernel Bootargs** in the menu:
	• Click **n** to deselect automatic generation of boot args
	• Select arguments on the line below, and click **Enter**
	
	Enter the Boot Arguments.
	
	For AMP on Zynq SoC, the following boot arguments are recommended:
	
	`console=ttyPS0,115200 earlyprintk` **`maxcpus=1 mem=256M clk_ignore_unused`**
	
	Additional options are in bold. The options limit linux to 1 CPU with 256M of main memory, and prevents it from managing clocks on the unused CPU.
	
	For SMP on Zynq SoC, the following boot arguments are recommended:
	
	`console=ttyPS0,115200 earlyprintk`
	`uio_pdrv_genirq.of_id=generic-uio`
	
	Additional options are in bold. This option maps the generic-uio driver to the installed driver name.
	
	For SMP on Zynq MPSoC, the following boot arguments are recommended:
	
	`earlycon=cdns,mmio,0xFF000000,115200n8`
	`console=ttyPS0,115200`
	`uio_pdrv_genirq.of_id=generic-uio`
	
	Additional options are in bold. This option maps the generic-uio driver to the installed driver name.

7. Configure Memory (Zynq-7000 AMP Only)	Change the memory configuration available for Linux kernel to make room for the DPD application running in CPU1 as follows: - select "Subsystem AUTO Hardware Settings->Memory Settings->kernel base address" - set to value 0x02000000. This creates a gap which can be used by DPD in CPU1 in AMP mode.

Table 6-7: ***(Cont'd)*Creating Linux Images**

Step	Command/Description
8. Add UIO Drivers in Kernel (SMP Only)	To add UIO Drivers to kernel: `petalinux-config -c kernel` which will start a menuconfig configuration screen. Navigate down Menus to "Device Drivers'Userspace I/O drivers'Userspace I/O platform driver with generic IRQ handling". Ensure that this kernel module is enabled to be loaded by typing "y" which will mark the entry with a "*"

9. Update Device Tree (SMP Only)	The device-tree needs to be updated to map DPD to the Generic UIO Drivers. See Device Tree Configuration section in the Petalinux Tools Documentation Reference Guide (UG1144)[Ref 16] for an overview of customizing the device tree.
	
	Update the following file:
	`project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi`.
	Following is an example system-conf.dtsi file:
	
	`/include/ "system-conf.dtsi"`
	`/ {`
	`};`
	`&TXCHAIN0_DFESS0_DPD_1 {`
	`  compatible = "generic-uio";`
	`};`
	
	TXCHAIN0_DFESS0_DPD_1 is the name of the node for DPD in the device tree, and can be found by looking in the following file:
	`components/plnx_workspace/device-tree-generation/pl.dtsi`
	If a design contains multiple instances of DPD, then multiple nodes will have to be changed.

10. Build Petalinux	Build the Petalinux Image by executing the command `petalinux-build` The result of this command will be in the sub-folder images/linux. The generated files can be then used for Linux boot up.

Control the Scheduling of DPD Application in Linux SMP

The utilization of processor and memory resources by the DPD application software can be affected by controlling the scheduling of corresponding process in Linux SMP. Depending on other software applications (e.g. Operation and Maintenance applications) executing concurrently in the Zynq multi-processor APU, care must be taken not to starve the DPD application to maintain adequate real-time performance. Analysis of real-time performance can be complex and it will depend highly on what other applications run concurrently with the DPD application.

The following are mechanisms available in Linux to control the priority of execution and allocation of resources to the DPD application to ensure adequate real-time operation.

- Use the `renice` command to control the priority of the DPD software application process. This provides a basic mechanism for prioritizing the scheduling of the DPD process among other application processes running in the APU. For example, executing `renice -20 pid` with `pid` being the process ID of the running DPD process can be used to set the scheduling priority to be the highest.
- Use `cpuset` and `isolcpus` to control the exclusivity of resources and binding of the DPD application process. This provides a more robust mechanism to allocate one of the CPUs for execution of the DPD application. In particular, use the following:
  - Use the boot argument `isocpus=cpu_num` to isolate one of the CPUs (`cpu_num`) from scheduling of user space processes by the Linux scheduler.
  - Use the `cpuset` mechanism to create a CPU set (e.g one with a single CPU `cpu_num`) and then bind the DPD process to that single CPU set.

The use of any of these Linux mechanisms will depend on the particular applications running concurrently in the APU.

Host Interface Registers

Communication with the application software is through the shared memory, accessed over the `S_AXI_USER` bus. The shared memory register map that is available for DPD control and status is documented in Table 6-8. Detailed use of the registers is described in Host Interface Operations Guide.

Table 6-8: **Memory Map**

Word Address	Mnemonics	Notes
0	CONTROLMODETRIGGER	A control mode is triggered by toggling this register from zero to 0xABCDEF12.
1	CONTROLMODEREGISTER	The number of the control mode to execute. See Software Control Modes for more information.
2	PORTNUM	Specify the port to which you apply the command. For non-DCL commands, specify the port and Request Type to apply on the `m_axis_srx_ctrl` stream (see SRX Control Stream). bits[31:16] - Request Type bits[15:0] - Antenna number
3	WAITINGONPORTSWITCH	This register is used by the host to acknowledge that the requested port has been switched in to the SRX path.
4	PARAMETER_0	Register used to pass a single parameter for various commands.
5 - 15	reserved	
16	COMMANDSTATUS	Status response for all non-DCL commands. See Table 6-14.
17	EXECUTEDCOMMAND	Echoes CONTROLMODEREGISTER on completion of the control mode
18	TRIGGERACK	Trigger Acknowledge (see DPD Application Software Boot Process)
19	CODEPOINTER	See DPD Application Software Boot Process or Antenna Selection Options in a Multipath Installation.
20	ACTIVEPORT	See Antenna Selection Options in a Multipath Installation.
21	EXECUTINGCOMMAND	Reports the currently executing command.
22	CAPTURERAMSTATE	bits [17:16] - TXRXRATIO bits [15:0] - Capture RAM state Capture RAM states: • 2: raw samples after CAPTURE_NEW_SAMPLES command completes. • 14: aligned samples after CAPTURE_AND_ALIGN command completes. • 15: AM/AM and AM/PM responses. Other values are undefined.
23	ACTIVESRXPORTREQUEST	Echo of value currently applied to `m_axis_srx_ctrl`.
24 - 31	reserved	
32	REPOSITORY	Software build Change List (CL) number.
33	BUILDDATE	Software build date: - day = mod(BUILDDATE,32) - month = mod((BUILDDATE-day)/32,12) - year = 2000 + ((BUILDDATE-day)/32 - month)/12
34	BUILDTIME	Software build time: - seconds = mod(BUILDTIME,60) - minutes = mod((BUILDTIME-seconds)/60,60) - hour = ((BUILDTIME-seconds)/60 - minutes)/60

Table 6-8: **Memory Map *(Cont'd)***

Word Address	Mnemonics	Notes
35	BUILDSETTINGS	bits[3:0] - number of antennas bits[6:4] - number of phases bit[7] - hybrid mode bits[11:8] - filter memory depth bits[15:12] - capture RAM depth, L = 2^(bits[15:12] + 9) bits[19:16] - acceleration level bit[20] - Peak in Window (PIW) capture mode bit[21] - Dynamic control layer (DCL) mode bit[22] - low resolution coefficients
36	HWVERSION	Hardware version - bits[31:24]-Device Family - bits[23:16]-Major version - bits[15:8] -Minor version - bits[7:0] - Revision Device Family 1 - Automotive Zynq 2 - Zynq-7000 3 - Zynq UltraScale+ 4 - Defense grade Zynq-7000
37	SWVERSION	- bits[23:16]-Major version - bits[15:8] -Minor version - bits[7:0] - Revision
38 - 61	reserved	
62	SRXQDC	Accumulation of the Q channel over the METERLENGTH interval (S32.30 value)
63	SRXIDC	Accumulation of the I channel over the METERLENGTH interval (S32.30 value)
64	RUNTIMELSW	32-bit LSW of a time monitor that counts the number of measurement intervals. The timer is reset when the meter length is updated or when the DCL parameters are set.
65	RUNTIMEMSW	32-bit MSW of a time monitor that counts the number of measurement intervals. The timer is reset when the meter length is updated or when the DCL parameters are set.
66	SRXLSW	32-bit LSW of the receiver power [31:28] - active SRX port value [27:0] - lower 28 bits of power meter
67	SRXMSW	32-bit MSW of the receiver power [31:28] - active SRX port value [27:0] - upper 28 bits of power meter
68	TXPOWERLSW_A	32-bit LSW of the transmit power at the input to DPD of port A
69	TXPOWERMSW_A	32-bit MSW of the transmit power at the input to DPD of port A
70	TXPOWERCOUNT_A	Number of non-zero samples in the power measurement of port A

Table 6-8: **Memory Map *(Cont'd)***

Word Address	Mnemonics	Notes
71	FCM_A	Proprietary frequency content metric (FCM)
72	ACTIVE_SID_A	Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation.
73	TXPOWERLSW_B	32-bit LSW of the transmit power at the input to DPD of port B
74	TXPOWERMSW_B	32-bit MSW of the transmit power of port at the input to DPD B
75	TXPOWERCOUNT_B	Number of non-zero samples in the power measurement of port B
76	FCM_B	Proprietary frequency content metric (FCM)
77	ACTIVE_SID_B	Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation.
78	TXPOWERLSW_C	32-bit LSW of the transmit power at the input to DPD of port C
79	TXPOWERMSW_C	32-bit MSW of the transmit power at the input to DPD of port C
80	TXPOWERCOUNT_C	Number of non-zero samples in the power measurement of port C
81	FCM_C	Proprietary frequency content metric (FCM)
82	ACTIVE_SID_C	Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation.
83	TXPOWERLSW_D	32-bit LSW of the transmit power at the input to DPD of port D
84	TXPOWERMSW_D	32-bit MSW of the transmit power at the input to DPD of port D
85	TXPOWERCOUNT_D	Number of non-zero samples in the power measurement of port D
86	FCM_D	Proprietary frequency content metric (FCM)
87	ACTIVE_SID_D	Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation.
88	TXPOWERLSW_E	32-bit LSW of the transmit power at the input to DPD of port E
89	TXPOWERMSW_E	32-bit MSW of the transmit power at the input to DPD of port E
90	TXPOWERCOUNT_E	Number of non-zero samples in the power measurement of port E
91	FCM_E	Proprietary frequency content metric (FCM)
92	ACTIVE_SID_E	Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation.
93	TXPOWERLSW_F	32-bit LSW of the transmit power at the input to DPD of port F
94	TXPOWERMSW_F	32-bit MSW of the transmit power at the input to DPD of port F
95	TXPOWERCOUNT_F	Number of non-zero samples in the power measurement of port F
96	FCM_F	Proprietary frequency content metric (FCM)
97	ACTIVE_SID_F	Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation.
98	TXPOWERLSW_G	32-bit LSW of the transmit power at the input to DPD of port G
99	TXPOWERMSW_G	32-bit MSW of the transmit power at the input to DPD of port G
100	TXPOWERCOUNT_G	Number of non-zero samples in the power measurement of port G
101	FCM_G	Proprietary frequency content metric (FCM)

Table 6-8: **Memory Map *(Cont'd)***

Word Address	Mnemonics	Notes
102	ACTIVE_SID_G	Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation.
103	TXPOWERLSW_H	32-bit LSW of the transmit power at the input to DPD of port H
104	TXPOWERMSW_H	32-bit MSW of the transmit power at the input to DPD of port H
105	TXPOWERCOUNT_H	Number of non-zero samples in the power measurement of port H
106	FCM_H	Proprietary frequency content metric (FCM)
107	ACTIVE_SID_H	Status indicating which coefficient set is loaded into the predistortion LUTs. Can be used for monitoring during DCL operation.
108 - 159	reserved	
160 - 191	shared DPD parameter Sets. See DPD Parameter sets for more details.	
192 - 255	reserved	
256 - 511	DCL Diagnostics (antenna 0 through 7) (read only). See Table 6-27.	
512 - 1023	Page transfer	Paging memory used to transfer large data sets to/from the user environment.

Hardware and Software Interaction

Xilinx DPD is a combination of hardware and software processes that between them realize the PA distortion inverse model and the estimation algorithm.

Figure 6-6 depicts the main elements of the DPD solution. The software processes are run in the ARM® processor core. There is also an Overdrive Detection (ODD) software process not indicated in the diagram.

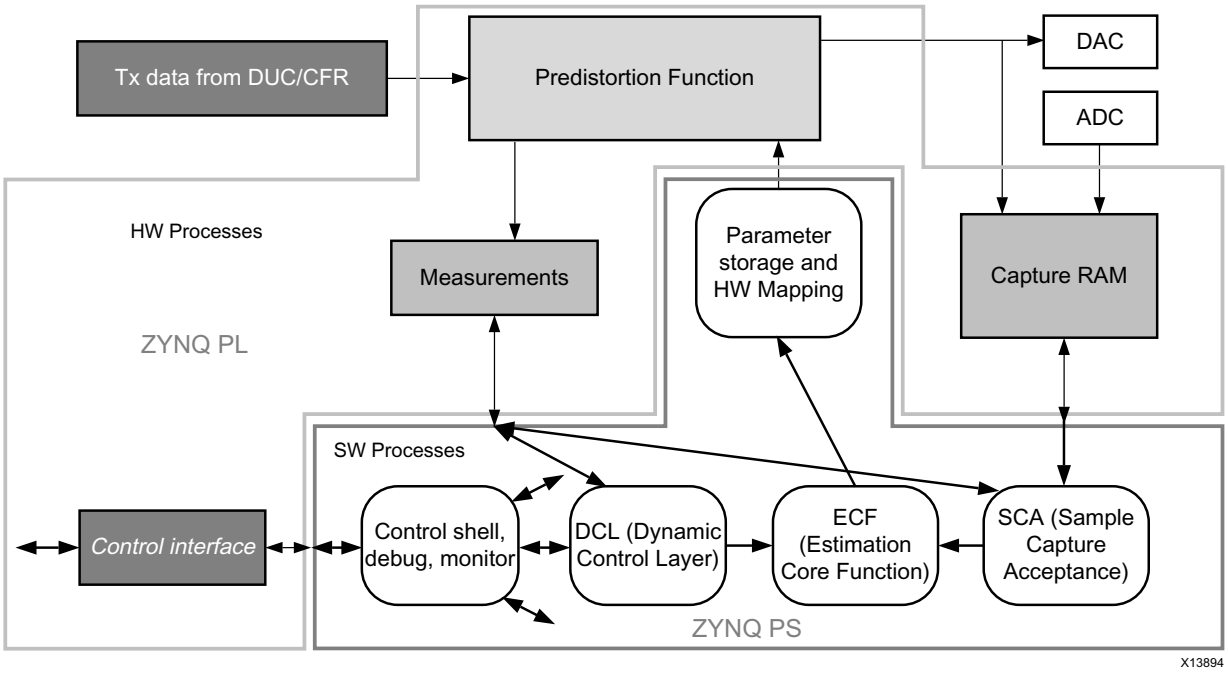


Figure 6-6: **Hardware Datapath and Major Software Processes**

Capture RAM and Estimation Core Function (ECF)

The capture RAM collects complex samples of the transmitted data and the appropriate samples from the observation path ADC, depending on the receiver configuration.

The ECF performs digital down-conversion and then alignment and coefficient estimation.

The ECF performs checks on the signals and reports any adverse status (see Software Control Modes) if there is an issue, in which case, the pre-distortion function parameters are not updated.

Measurements Block and Sample Capture Acceptance (SCA)

The capture RAM is able to capture tens of microseconds of data. To obtain optimum estimation results, the capture needs to contain data that is representative of the signal over the lifetime of the coefficients, a time much longer than the capture length. In particular, if the capture is taken during a period of time in which the signal amplitude is small, for example during the low period of a data pulse, the estimation can be poor for higher amplitude signals that might occur over a longer timescale. This is similar in principle to fitting a polynomial curve to points in a scatter diagram. The function beyond the region where there are data points is an extrapolation, yielding unpredictable behavior.

In some cases there is a frame structure with a known good period that can be used for estimation. For this case, DPD can be configured to allow the capture to be triggered by an external sync pulse, with a programmable delay. Otherwise, the SCA feature can be enabled.

SCA takes captures at random and applies acceptance criteria to the samples in the capture RAM to ensure that the samples are representative of the full signal. The acceptance criteria are based on the average power and statistical measures (histograms) of the transmitted data in the capture buffers, and their comparison to the average power and statistics of the transmitted signal over a defined measurement interval (typically greater than 10 ms). The Measurements block depicted in Figure 6-6 provides the real-time processing of the signals required to form the powers and histogram. This data is also available through the control interface to aid system debug and monitoring.

When SCA is running, TDD signals are automatically catered for – a capture taken in the uplink period fails and only data transmitted in the downlink period is ever used.

An optional Peak-In-Window capture hardware mechanism is available for cases where capturing the maximum signal peak observed over an interval much longer than the depth of a capture is desired.

Setting Valid Capture Windows

The CAPWIN parameter set can be used to define up to 10 valid capture windows. These valid capture windows are optional, but, can be used to limit where in a frame captures can occur. This can be beneficial in TDD systems, where it is good to avoid the time immediately after the PA turns on.

When left un-defined, the capture processes are free to capture anywhere in the signal so long as the capture passes all SCA criteria for the current mode. Adding valid capture windows has the effect of adding an additional criteria that must be passed by the SCA process. The added criteria requires that the location of the capture relative to the last capture sync is within one of the defined capture windows. Use of the CAPWIN parameter set requires that the system supplies DPD with the capture sync signal.

The details of the operation of the valid capture windows will depend on which CAPTUREMODE is being utilized.

CAPTUREMODE 0 - use smart capture with SCA

For this capture mode, the definition of valid capture windows will directly extend the existing SCA criteria to include a check that the current capture location falls within any of the defined valid windows.

CAPTUREMODE 1 - capture at fixed delay from supplied capture sync signal.

For this capture mode, the definition of valid capture windows will cause the DPD software to select a random delay from within a single window. The capture at this delay will be checked against the SCA criteria and if all checks are passed, the capture is accepted. If any SCA check fails, the DPD software will select another random delay from the next valid window. This process continues until a capture is found that passes all SCA criteria.

CAPTUREMODE 2 - use software SCA with no hardware assistance.

For this capture mode, the definition of valid capture windows will directly extend the existing SCA criteria to include a check that the current capture location falls within any of the defined valid windows.

CAPTUREMODE 3 - use PIW

For this capture mode, (when the PIW hardware is available) the definition of valid capture windows will cause DPD to return the capture containing the largest signal peak observed within the current window. When multiple valid windows are defined, the capture process will cycle through each window in a round-robin mode.

Dynamic Control Layer (DCL)

The strategy for dealing with average power dynamics, such as would occur via call-load and power control in a 3G/4G cell site, is driven by the characteristics of a PA and DPD filter when predistorted. Understanding what happens when the signal changes (for example, its power or frequency content) after the coefficients were calculated is key to achieving acceptable performance at a reasonable hardware cost.

A key distinguishing feature among PAs, and one that has a profound effect on the complexity of the DPD solution, is whether a single coefficient set can be applied over the range of signal powers that are transmitted. Where this is the case, after this coefficient set is found it does not need to be adapted when the transmitted power changes. Other PAs are sensitive to the power level and the DPD must be re-adapted whenever the power level changes (either higher or lower).

These two types are depicted qualitatively in Figure 6-7. Figure 6-7(a) indicates that acceptable adjacent channel power (ACP) is maintained for all power levels below the power used for training the DPD coefficients. Figure 6-7(b) depicts a case where acceptable ACP is only achieved for a region close the power used for training the DPD.

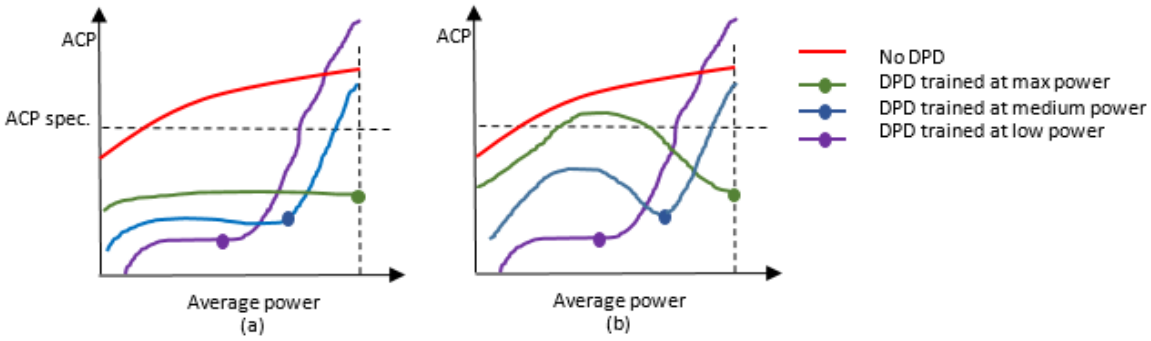


Figure 6-7: **Average power variation (a) Single set capable (b) Multi-set required**

In addition to average power considerations the variations in frequency content of input signals must also be taken into account. PAs are designed to support the desired signal

bandwidth; so changes in the signals frequency content tend to have minimal impact on the PA when compared to the effect that the DPD filter has on the performance. The frequency response of DPD filters that contain memory terms will generally be valid only for the frequencies that are present when the DPD filter adaptation occurs (a memory-less predistorters response is valid at all frequencies). The response beyond these frequencies can vary widely; this phenomena is shown qualitatively in Figure 6-8.

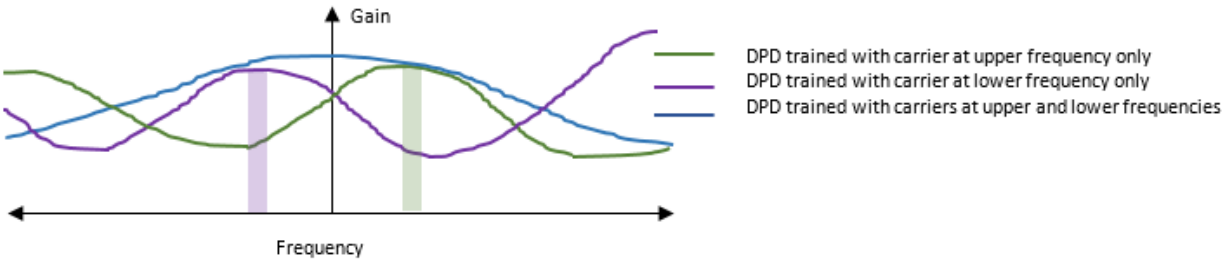


Figure 6-8: **Average frequency response of DPD filter**

Note that these descriptions are valid when the transmitted power is varied by changing the digital gain of the signal before predistortion. It is assumed that the analog gain is fixed. Indeed, predistortion performance is very sensitive to the analog gain after DPD (because that affects the accuracy of the transfer characteristic model) and, in practice, the analog gain drifts and should be tracked. It should be emphasized that the DPD solution here is not intended to be used in any system having fast analog power dynamics, except in the case of a TDD system where the amplifier can be turned off during uplink periods to no ill effect.

DCL for Single-Set System

The previous considerations imply an update strategy for a system suitable for a single coefficient set is to get the coefficient set trained at Pmax as soon as possible. When trained at Pmax, continue to track analog gain and temperature drift (these processes occur over time scales like tens of seconds). A triggered update based only on the power measurement (assumes no significant change in the frequency content of the input signal) can be used.

The method is to store the coefficients used for predistortion together with a record of the power at which they were estimated (SS_PSET0). Then estimate and replace the coefficients whenever the current measured power (P) crosses a specific threshold. With traffic dynamics, SS_PSET0 becomes closer and closer to Pmax if the trigger is P > SS_PSET0. However, to track analog gain drift it is necessary to ensure that the predistortion coefficients are unconditionally recalculated within a certain time span. Therefore, the trigger condition needs to be modified to allow for situations when P falls below SS_PSET0. This is obtained by resetting the trigger threshold to SS_PSET0 initially and whenever an estimation is performed. After the threshold has been reset, it decays with time, at a user programmable rate (DCLPSTEP). When the threshold falls to P, the coefficients are re-estimated. An example of the behavior of this algorithm is shown in Figure 6-9.

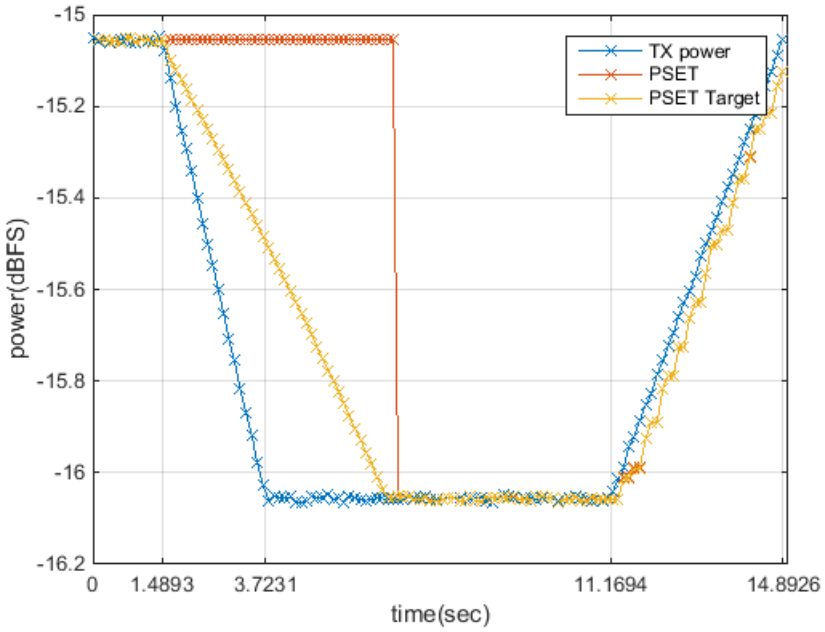


Figure 6-9: **Single set DCL PSET training**

The dynamic performance depends on the signal ramp-up rates. If the ramp up is slow, the coefficients never need to be used for powers greatly above their estimated power and no spectral violation is observed. If, on the other extreme, the power ramps up instantaneously, the algorithm does not exhibit optimum predistortion until the coefficients are estimated. However, this is a one-time effect, as subsequent ramp-ups present no problems. To cover cases where this is a problem, off-air calibration can be performed and optimum initial coefficients can be pre-loaded.

DCL for Multi-Set Systems

Consider the curves in Figure 6-7(b) and Figure 6-8. Although there is clearly a continuum of possibilities, the idea is to show the problem case; as the power backs-off or the frequency content of the signal changes, the performance gets worse and can even break mask.

This behavior and much previous work on DPD is related to how to deal with tracking this power dependence. The brute force approach would be to have an estimation cycle that is fast enough to track the power dynamics, but this requires a significant cost of hardware resources and might not be sufficient for fast frequency content changes. Therefore, the Xilinx DPD solution includes a mechanism to allow for the use of multiple coefficient sets, each trained at different power levels and potentially different frequency contents. Figure 6-10 indicates the desired performance that would be achieved by switching between coefficients sets as the power level changes.

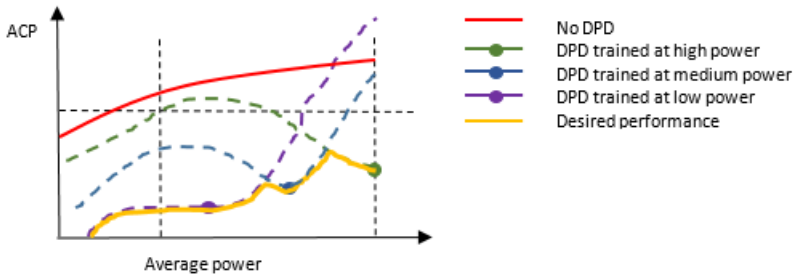


Figure 6-10: **Desired Performance**

The Xilinx approach is to devise a feature that allows power and frequency content to be responded to quickly, while retaining the benefits of a software solution for estimation, namely, minimal additional hardware resource.

The principle is to learn multiple sets of coefficients, trained at different powers and frequency content, then jump from one set to the other, and thus, by staying on the yellow curve, always meet mask. The rate at which to jump must be fast, and this is achieved with an interrupt service routine (ISR) triggered on the measurement interval (for example, 10 ms). Concurrent with the ISR is a training algorithm that tracks power and frequency content and updates the coefficient sets opportunistically.

Multipath Handling

The estimation process can update only one path at a time. The multipath DCL attends to each path in turn.

In the limiting case where only one path ever satisfies the criteria for coefficient update, that path is continuously re-estimated. This also means that if one path fails, DPD operates correctly on the others.

Quadrature Modulator Correction (QMC)

The DPD solution supports Quadrature Modulator Correction (QMC) using no special waveforms, training sequences, or additional RF hardware; it is fully integrated into the DPD solution and is able to operate virtually transparent while DPD is operational.

The use of QMC is only supported at either the TX or RX side. It cannot be applied to both the TX and RX side simultaneously because the QMC, as implemented, cannot distinguish between QM imperfections on the TX side versus imperfections on the RX side if direct conversion were used on both.

The optional TX ZIF QMC correction is only available (to enable) while using real ADC sampling on the receive side. The RX QMC correction is automatically enabled when complex ADC sampling is used on the received side. Table 6-9 describes the TX and RX data format, IF choices, and QMC support.

Table 6-9: **DPD QMC Support**

TX data format	TX analog IF	Fb RX data format	QMC Support
Complex	Zero	Complex	RX QMC (TX QMC to be managed outside DPD)
		Real	TX QMC + RX DC
	Non-zero	Complex	RX QMC (TX QMC to be managed outside DPD)
		Real	RX DC (TX QMC to be managed outside DPD)

Power Amplifier Protection

Setting the maximum drive level for an RF power amplifier under operational conditions can be a non-trivial task. The use of digital pre-distortion allows the PA to be driven such that the peaks of the predistorted waveform can be driven very close to the saturation point of the PA. However any form of pre-distortion begins to fail when the peaks of the waveforms enter the saturating region of the amplifier.

DPD contains functions that dynamically detect when the signal is being driven too hard. One method is predictive, which means that an overdrive condition can be detected before it actually occurs.

In some cases the predicted expansion can be inaccurate so a more reliable method based on actual signal measurement is also available. The predictive method is referred to as Overdrive Detection (ODD) and the measurement-based method is referred to as Peak Saturation.

Overdrive Detection (ODD)

After each pre-distortion coefficient estimation, ODD computes the estimated peak expansion value. The estimated expansion is compared to an overdrive threshold (which is a user parameter), and the result is used to report an overdrive status for that estimation, which can be monitored. There is a second user-defined threshold that is used to prevent the coefficients from being used.

Figure 6-11 shows an example of the returned status value for a system operating in the saturation region of a power amplifier. For this example, the default ODDEXPTHRESHOLDS value is used. The default sets the overdrive detect threshold at 4 dB of expansion and the overdrive protect threshold at 5 dB of expansion. A small DAMPINGVALUE is used so the expansion at each iteration of the DPD update can be easily observed. The system starts with pass-through coefficients (that is, no expansion). Each iteration increases the expansion applied by the DPD filter as it works to linearize the system. For estimated expansions below 4dB the returned status value is SUCCESSFUL. When the estimated expansion is between 4dB and 5dB the returned status is OVERDRIVE_DETECTED, this value can be used as a system warning to indicate that expansion is getting high but does nothing to affect the normal update process. When the estimated expansion is greater than 5dB the returned status is OVERDRIVE_PROTECTED.

The new coefficients that are estimated to cause more than 5 dB expansion are NOT loaded into the DPD filter, hence the large expansion never actually occurs. The coefficients remain as the last set that did not cause the OVERDRIVE_PROTECTED condition.

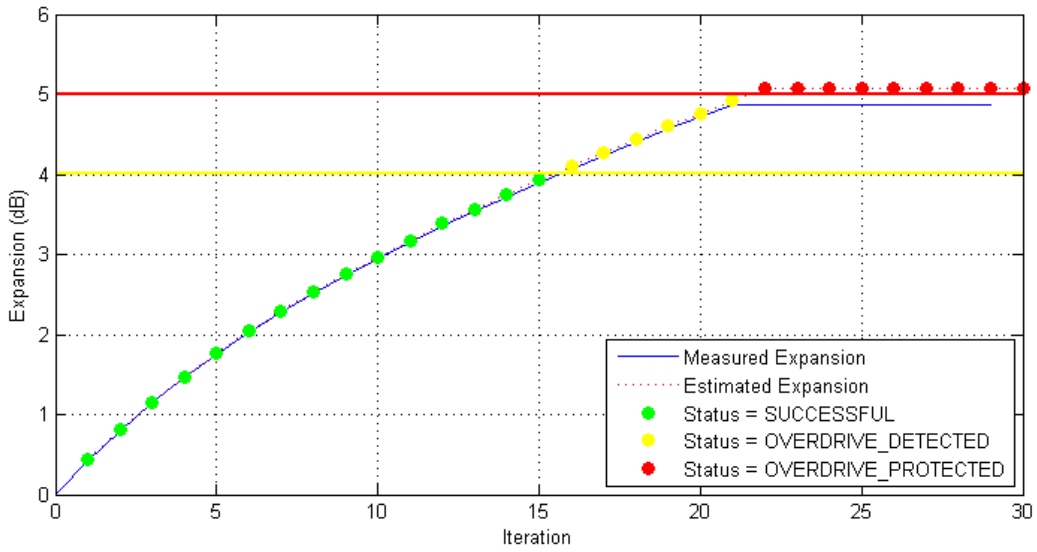


Figure 6-11: **Returned Status Value**

CAUTION! *The estimated expansion can be inaccurate when the input signal is sparsely populated with a narrow-bandwidth signal. In these conditions it is recommended that ODPENABLE be disabled or the Peak Saturation method be enabled.*

ODDEXPTHRESHOLDS

For ODPENABLE = 2, the thresholds are set relative to the peaks of the input signals. This mode limits the amount of expansion than can be applied to the signal regardless of the input power level.

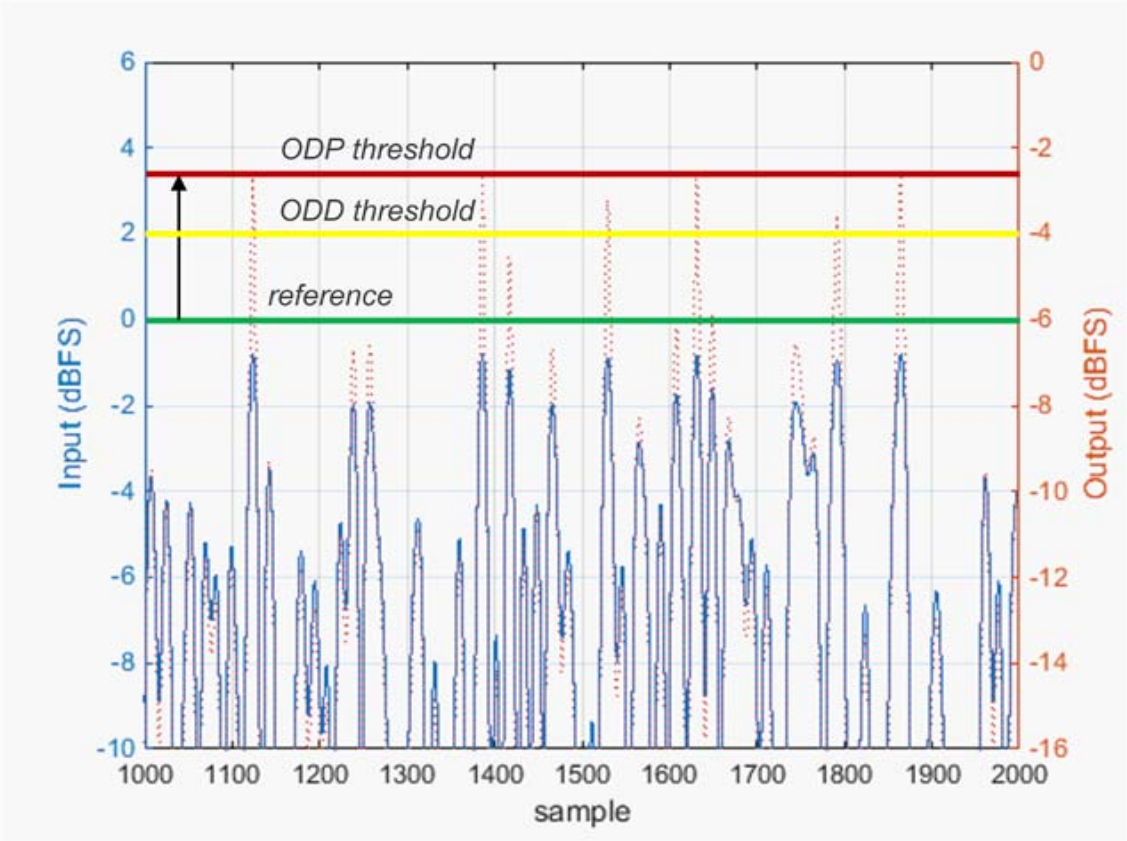


Figure 6-12: **Example of ODDPSOTHRESHOLDS for ODPENABLE = 2**

Peak Saturation

During each signal capture the peak value at the output of the DPD filter is measured during the capture interval. If the peak value is above the warning threshold and below the protection threshold, a successful update will return the EXPANSION_SATURATION_WARNING message. If the peak value is detected to be above the protection threshold, the coefficient update equation will become w(n+1) = w(n)*0.999 + e(n)*0.001 and the EXPANSION_SATURATION message is returned. This will have the effect of stopping the expansion until you intervene (for example, back-off the PA).

For ODPENABLE = 1, the thresholds are set relative to 0dBFS of the output. This mode operates by limiting the magnitude of the maximum peaks, it does not limit the amount of expansion.

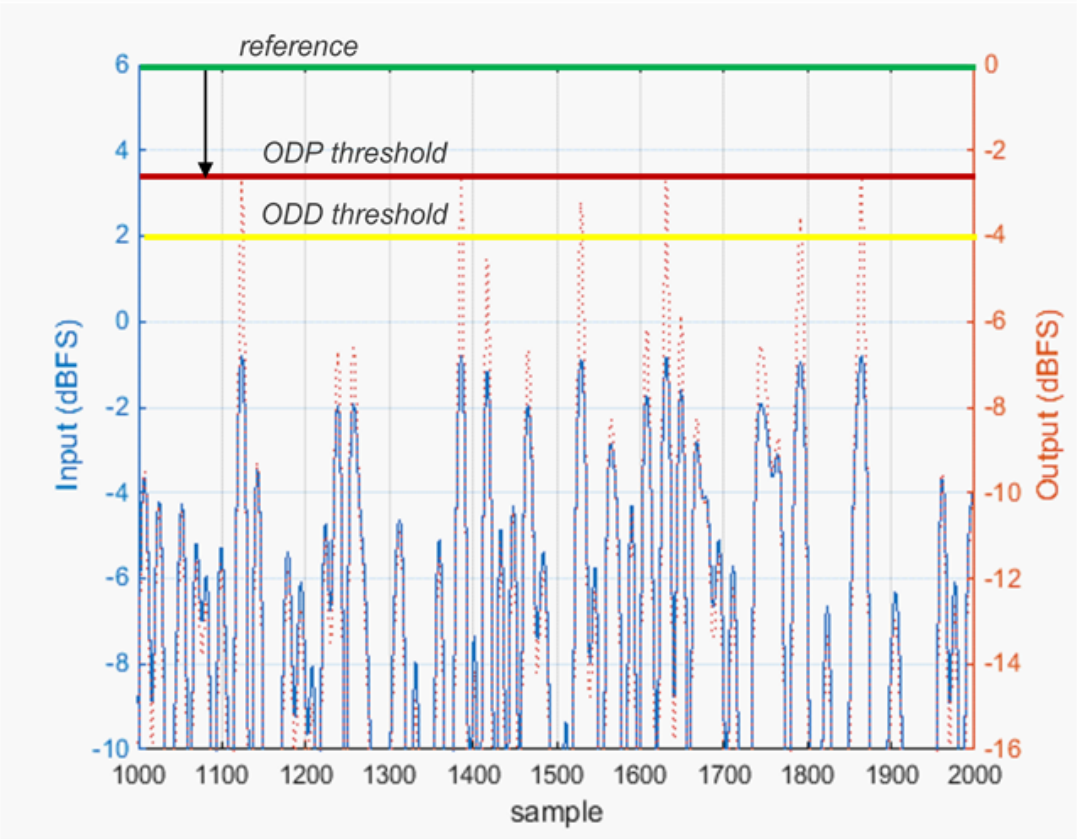


Figure 6-13: **Example of ODDPSOTHRESHOLDS for ODPENABLE = 1**

Host Interface Operations Guide

The DPD core is controlled using the host interface RAM (see Host Interface Registers). DPD core operations are activated by writing data to addresses in the RAM. Status, results, diagnostics, and data are accessed by reading data from addresses.

The host interface memory map is organized into the regions as shown in Table 6-10. In what follows, individual addresses are specified. For ease of reference and support, uppercase mnemonics are defined for key addresses, parameters, control modes and values.

Table 6-10: **DPD Host Interface Memory Map Outline**

Address Range	Usage
0–31	Control and Status
32–127	Continuous Monitors (read only)
160–191	Parameters
256–511	DCL Diagnostics (antenna 0 through 7) (read only)
512–1023	Page Transfer

Software Control Modes

DPD features are executed using triggered control modes (see DPD Application Software Boot Process). They are like function calls with optional parameters that influence the behavior of the datapath and internal state, in addition to returning results. Control modes are provided that allow you to configure the DPD core, run single-step estimation, run the DCL, access measurements, and read data and status information for setup, debug, and monitoring. Table 6-11 shows a full list of available control modes and Table 6-12 shows the general registers associated with control mode operation.

Table 6-11: **DPD Control Modes**

Value	Mnemonic	Description
0	WAIT_STATE(2)	NO-OP
1	READ_CONFIGURATION(2)	Refresh the host interface REPOSITORY, BUILDDATE, BUILDTIME, BUILDSETTINGS and HWVERSION registers.
2	RESTORE_DEFAULTS(2)	Restore the default configuration.
3	RUN_UPDATE_COEFFICIENTS_V2(1)	Perform a full update of the DPD coefficients. This command is recommended for high PAR signals, to improve performance. This update mode works only with the "smart capture with SCA" signal capture processing. The user defined CAPTUREMODE is not utilized.
4	Reserved	
5	RESET_ALIGN_CALIBRATION(1)	Reset alignment calibration, this causes a new calibration on the next attempted alignment. This command must be sent whenever the path between the DPD output and the SRX input is changed (e.g. gain changes, RF component reconfiguration etc.)
6	RUN_UPDATE_COEFFICIENTS(1)	Perform a full update of the DPD coefficients. This includes capturing new samples, using the method selected with the CAPTUREMODE parameter, ECF processing and updating the datapath parameters.
7	CAPTURE_NEW_SAMPLES(1)	Trigger a new sample capture sequence. The capture follows the rules set by the CAPTUREMODE parameter.
8	RESET_COEFFICIENTS(1)	Set all coefficient sets to unity gain and update the datapath for pass-through.
9	DPD_OFF(1)	Update the datapath with pass-through unity values without modifying the internally stored coefficients.

Table 6-11: **DPD Control Modes *(Cont'd)***

Value	Mnemonic	Description
10	DPD_ON[(1)]	Update the datapath from the internally stored coefficients.
11	REPORT_COEFFICIENTS[(1)]	This command is used to copy a selected set of coefficients from the internal memory to the host interface. See Reading and Loading Coefficient Set.
12	LOAD_COEFFICIENTS[(1)]	This command is used to copy a selected set of coefficients from the host interface to the internal memory. See Reading and Loading Coefficient Set.
13-18	Reserved	
19	RUN_DCL[(2)]	Directs the processor to run the DCL control function. This function has a unique use model which should be understood fully prior to executing it.
20	EXIT_DCL[(2)]	Directs the processor to exit the DCL control function.
21	RESET_DCL[(2)]	This command is used to reset the internal state machine for the DCL routine. The DPD coefficients are not reset, but the sideband information associated with the coefficients and used by the DCL routine is cleared (for example, the power level when the coefficients were computed).
22	GET_CAPTURE_RAM_PAGE[(2)]	Present the page, specified by PARAMETER_0, of the capture RAM data at the host interface RAM page transfer area.
23	GET_HISTOGRAM[(1)]	Present the 256 bins of the transmit histogram at the DPD input at the host interface RAM page transfer area.
24	GET_CAPTURE_HISTOGRAM[(2)]	Present 256 bins of the transmit capture histogram at the host interface RAM page transfer area.
25	READ_CAPTURE_POWER_METERS[(2)]	Present the values from the DPD input capture TX power meter at addresses 512(LSW) and 513(MSW) and the capture RX power at addresses 514(LSW) and 515(MSW).

Table 6-11: **DPD Control Modes *(Cont'd)***

Value	Mnemonic	Description
26	GET_SWEPT_CAPTURE_POWER(1)	DPD will perform 512 evenly distributed captures using CAPTUREMODE=1 with CAPTUREDELAY of between 0 and PARAMETER_0 samples. The captured power at each delay is reported in the page transfer area. See GET_SWEPT_CAPTURE_POWER Function for more information
27	RUN_ODD_AMAM(1)	Run ODD algorithm on the current coefficients and report the estimated AM/AM response in the host interface. See RUN_ODD_AMAM Function for more information.
28	CLEAR_C2L_OVERFLOW(1)	Clear latched coefficient overflow warning.
29	GET_CAPTURE_PSD(2)	Compute power spectral density of the contents in a capture RAM. See GET_CAPTURE_PSD Function for more information.
30	CAPTURE_AND_ALIGN(1)	Capture samples and run signal alignment routines.
31	QMC_RESET(1)	Reset the internally stored QMC coefficients and set the QMC datapath block to pass-through.
32	QMC_OFF(1)	Update the QMC datapath block for pass-through without changing the internally stored coefficients.
33	QMC_ON(1)	Update the QMC datapath block from the internally stored coefficients.
34	REPORT_QMC_COEFFICIENTS(1)	Copy the internal QMC coefficients to the host interface.
35	LOAD_QMC_COEFFICIENTS(1)	Read QMC coefficients from the host interface to the internal QMC memory. This can be used to initialize the QMC performance.

Table 6-11: **DPD Control Modes *(Cont'd)***

Value	Mnemonic	Description
36	MONITOR_CAPTURE_LOCATION[1]	DPD will perform 512 captures using the user defined CAPTUREMODE. The location within the frame of each capture (number of samples since the last capture sync) is reported in the page transfer area. All page transfer address values are initialized to 0xFFFFFFFF at the start of this function. This function can be used to help validate the CAPWIN parameter setting. See MONITOR_CAPTURE_LOCATION Function for more information.
37	COMPUTE_AMAM_AMPM_RESPONSES[1]	Capture samples and compute the AM/AM and AM/PM responses between DPD input and DPD FbRx Input.
38	COMPUTE_PA_AMAM_AMPM_RESPONSES[1]	Capture samples and compute the AM/AM and AM/PM responses of the PA (i.e. DPD output v/s DPD Feedback Input).
39	DISABLE_OUTPUT[1]	Disable the DPD filter output.
40	ENABLE_OUTPUT[1]	Enable the DPD filter output after being disabled using the DISABLE_OUTPUT command.
41	REPORT_DIAGNOSTICS[1]	Report requested DPD diagnostics to the host interface.
42	CLEAR_DIAGNOSTICS[1]	Clear DPD diagnostics for the current port.
43	UPDATE_ECF_PARAMETERS[2]	Update the ECF parameters for specified port(s).
44	REPORT_ECF_PARAMETERS[2]	Report the ECF parameters for a specific port.
45	UPDATE_ARCH_PARAMETERS[2]	Update the ARCH parameters for specified port(s).
46	REPORT_ARCH_PARAMETERS[2]	Report the ARCH parameters for a specific port.
47	UPDATE_CAPTURE_PARAMETERS[2]	Update the CAP parameters for specified port(s).
48	REPORT_CAPTURE_PARAMETERS[2]	Report the CAP parameters for a specific port.
49	UPDATE_DCL_PARAMETERS[2]	Update the DCL parameters for specified port(s).

Table 6-11: **DPD Control Modes *(Cont'd)***

Value	Mnemonic	Description
50	REPORT_DCL_PARAMETERS[(2)]	Report the DCL parameters for a specific port.
51	UPDATE_ODD_PARAMETERS[(2)]	Update the ODD parameters for specified port(s).
52	REPORT_ODD_PARAMETERS[(2)]	Report the ODD parameters for a specific port.
53	UPDATE_MET_PARAMETERS[(2)]	Update the MET parameters for specified port(s).
54	REPORT_MET_PARAMETERS[(2)]	Report the MET parameters for a specific port.
55	UPDATE_TXQMC_PARAMETERS[(2)]	Update the TXQMC parameters for specified port(s).
56	REPORT_TXQMC_PARAMETERS[(2)]	Report the TXQMC parameters for a specific port.
57	UPDATE_CAPWIN_PARAMETERS[(2)]	Update the CAPWIN parameters for specified port(s).
58	REPORT_CAPWIN_PARAMETERS[(2)]	Report the CAPWIN parameters for a specific port.
0xDEADC0DE	KILL_DPD[(2)]	Disable DPD interrupts and exit the DPD application. This command can be used to stop all AXI transactions initiated by the DPD core. Once you run this command, DPD application has to be restarted. Use this command ONLY when the DPD stops responding.

Notes:
1. Port specific command, this command will only affect the port defined in PORTNUM during command triggering.
2. Port agnostic command, PORTNUM changes will be conveyed on the m_axis_srx_ctrl_tdata signal but does not affect the response of the requested command (i.e. the effect of the command will be the same regardless of the value of PORTNUM). An invalid PORTNUM values will still result in a failed command with response of INVALID_COMMAND and the requested command is not executed.

Table 6-12: **DPD Host Interface Control Registers**

Address	Mnemonic	Description
0	CONTROLMODETRIGGER	A control mode is triggered by toggling this register from zero to 0xABCDEF12.
1	CONTROLMODEREGISTER	The number of the control mode to execute.
2	PORTNUM	Specify the port to which you apply the command. For non-DCL commands, Specify the port Request Type to which you apply `m_axis_srx_ctrl` stream (see SRX Control Stream). bits[31:16] - Request Type bits[15:0] - Antenna number
3	WAITINGONPORTSWITCH	This register is used by the host to acknowledge that the requested port has been switched in to the SRX path.

Table 6-12: **DPD Host Interface Control Registers *(Cont'd)***

Address	Mnemonic	Description
4	PARAMETER_0	Register used to pass a single parameter for various commands.
5-15		<reserved> future expansion
16	COMMANDSTATUS	See Table 6-14.
17	EXECUTEDCOMMAND	Echoes CONTROLMODEREGISTER on completion of the control mode.
18	TRIGGERACK	Trigger Acknowledge
19	CODEPOINTER	See DPD Application Software Boot Process or Antenna Selection Options in a Multipath Installation.
20	ACTIVEPORT	See Antenna Selection Options in a Multipath Installation.
21	EXECUTINGCOMMAND	Reports the currently executing command.
22	CAPTURERAMSTATE	Reports the current state of the capture RAMs to aid in parsing the samples
23	ACTIVESRXPORTREQUEST	Echo of value currently applied to m_axis_srx_ctrl.
24- 31		<reserved> future expansion

DPD Application Software Boot Process

The DPD application software ELF must be included in the boot process of the Zynq-7000 PS along with any other applications and OS. The specifics of the boot process depend on the boot mode, memory device, and OS to be used in a particular system. For a reference example of how to tailor a boot scheme based on a slightly modified version of the First Stage Boot Loader (FSBL) from SDK, see *Simple AMP Running Linux and Bare-Metal System on Both Zynq SoC Processors* (XAPP1078) [Ref 11]. The modifications to the FSBL described in this application note can serve as the basis to boot the DPD application in CPU1 while running Linux (or other OS) in CPU0.

The DPD hardware must be out of reset with stable clocks before the DPD software is allowed to initialize and configure the system. See Reset and Clock Sequencing for more information. To facilitate the proper sequencing, the DPD software will wait at the beginning of the main function until the CONTROLMODETRIGGER register is set to 0xA5A5A5A5 before initializing the DPD system and starting the DPD command shell. Figure 6-14 shows the DPD software boot sequence.

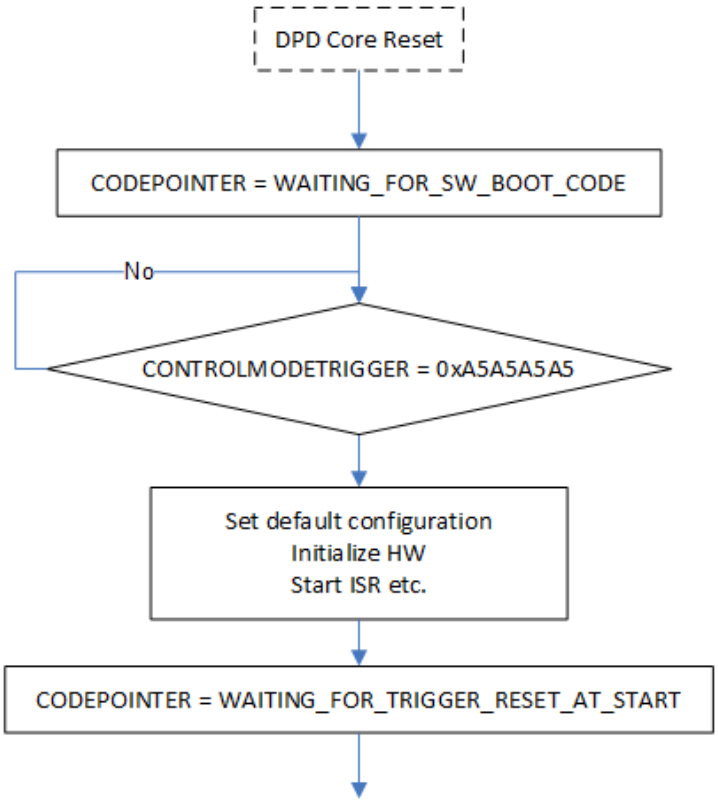


Figure 6-14: **DPD SW Boot Sequence**

Control Handshake

The control modes should be triggered using the CODEPOINTER and TRIGGERACK registers to facilitate handshaking between the DPD core and the host environment. The valid values for the CODEPOINTER register are shown in Table 6-13. When operating with the debug interface (See *Xilinx LogiCORE IP Digital Pre-Distortion v8.1 Debug Interface User Guide*(UG989) [Ref 1]), the values read from registers over `s_axi_user` are zero, when debug interface has taken control of the host interface via the `s_axi_ctrl` bus. Customer software should be aware of this and should time out when polling for values in steps below, and should return to step 1 in the handshake.

Table 6-13: **Code Pointer Values**

Value	Mnemonic	Description
0	UNDEFINED	Indicates that `s_axi_ctrl` interface has taken control of host interface during debug stage.
1	COMMAND_IN_PROGRESS	Indicates that the processor is busy in executing a requested command.
16	PERFORMING_CAPTURE_PROCESS	The capture process is running. During this time the SRX path should be dedicated to the requested feedback path.

Table 6-13: **Code Pointer Values *(Cont'd)***

Value	Mnemonic	Description
125	WAITING_FOR_SW_BOOT_CODE	Waiting to detected 0xA5A5A5A5 code in CONTROLMODETRIGGER register at beginning of main() function indicating that it is okay to proceed with boot-up.
126	WAITING_FOR_GLOBAL_TIMER_ENABLE	The software is waiting to detect the global timer is enabled.
127	WAITING_FOR_DPD_ARESETN_CLEAR	The software is waiting for `dpd_aresetn` to be cleared.
128	WAITING_FOR_TRIGGER	The code is ready to execute a new command.
129	WAITING_FOR_TRIGGER_RESET	The current command has completed. When the CONTROLMODETRIGGER is reset the code is ready for the next command.
130	WAITING_FOR_TRIGGER_RESET_AT_START	The CONTROLMODETRIGGER needs to be reset before completing the bootup.
131	WAITING_FOR_PORT_SWITCH	The code is sitting idle waiting for an acknowledgment that the RF port has been switched as requested. (See Antenna Selection Options in a Multipath Installation)
132	READING_HW_SETTING	During boot up, the software is reading the hardware settings registers.
133	SET_DEFAULT_CONFIG	During boot up, the software is configuring the default parameters.
134	RESET_QMC_REGISTERS	During boot up, the software is resetting the QMC registers.
135	SKIP_RESET_QMC_REGISTERS	During boot up, the software is instructed to skip the QMC register resetting.
136	RESET_FILTER_COEFFS	During boot up, the software is resetting the DPD filter.
137	SKIP_RESET_FILTER_COEFFS	During boot up, the software is instructed to skip the DPD filter resetting.

The sequence of events is as follows (see Figure 6-15):

1. Verify CODEPOINTER is WAITING_FOR_TRIGGER.
2. Write the number of the required control mode to CONTROLMODEREGISTER.
3. Write the port (antenna) number (0 to 7) and Request Type to PORTNUM.
4. Write optional parameters as specified in Updating and Reporting DPD Parameters.
5. Write trigger value to CONTROLMODETRIGGER register.
6. Wait for trigger acknowledge (read TRIGGERACK until it goes to 1).
7. Wait for command to complete (wait for CODEPOINTER to equal WAITING_FOR_TRIGGER_RESET).

8. Reset the trigger (write 0 to CONTROLMODETRIGGER).
9. Wait for trigger acknowledge to reset (TRIGGERACK goes to zero).
10. Read COMMANDSTATUS.

Except DCL modes, modes terminate and return a status value in COMMANDSTATUS. Possible returned values are shown in Table 6-14.

Table 6-14: **Status Values**

Value	Mnemonic	Description
-911	PANIC_RESET_DETECTED	ERROR: A panic reset pin assertion has been detected.
-511	EVAL_LICENSE_TIMEOUT	ERROR: The evaluation license has timed out.
-415	HWA_ERROR_0	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-414	HWA_ERROR_1	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-413	HWA_ERROR_2	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-412	HWA_ERROR_3	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-411	HWA_ERROR_4	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-410	HWA_ERROR_5	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-409	HWA_ERROR_6	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-408	HWA_ERROR_7	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.

Table 6-14: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-407	HWA_ERROR_8	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-406	HWA_ERROR_9	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-255	CONFIG_FAILURE_PORTSEQUENCING	ERROR: Failed to successfully update the DCL parameter set because of invalid PORTSEQUENCING parameter.
-254	CONFIG_FAILURE_ENABLEVSWRPWRMSR	ERROR: Failed to successfully update the DCL parameter set because of invalid ENABLEVSWRPWRMSR parameter.
-253	CONFIG_FAILURE_ARCH_SEL_B	ERROR: Failed to successfully update the ARCH parameter set because of invalid ARCH_SEL parameter (B value).
-252	CONFIG_FAILURE_ARCH_SEL_C	ERROR: Failed to successfully update the ARCH parameter set because of invalid ARCH_SEL parameter (C value).
-251	CONFIG_FAILURE_ARCH_SEL_ABC	ERROR: Failed to successfully update the ARCH parameter set because of invalid ARCH_SEL parameter (value outside 0xABC).
-250	CONFIG_FAILURE_TXRXRATIO	ERROR: Failed to successfully update the ECF parameter set because of invalid TXRXRATIO parameter.
-249	CONFIG_FAILURE_PORTSKIP	ERROR: Failed to successfully update the DCL parameter set because of invalid PORTSKIP parameter.
-248	CONFIG_FAILURE_CAPTUREMODE	ERROR: Failed to successfully update the CAP parameter set because of invalid CAPTUREMODE parameter.
-247	CONFIG_FAILURE_HIRESMONITOR	ERROR: Failed to successfully update the CAP parameter set because of invalid HIRESMONITOR parameter.
-246	CONFIG_FAILURE_DCLALGORITHM	ERROR: Failed to successfully update the DCL parameter set because of invalid DCLALGORITHM parameter.
-245	CONFIG_FAILURE_METERLENGTH_TWO_PHASE	ERROR: Failed to successfully update the MET parameter set because of invalid METERLENGTH parameter (must be factor of two).
-244	CONFIG_FAILURE_MINDELAY_MAXDELAY_WIN_SIZE_BIG	ERROR: Failed to successfully update the ECF parameter set because of invalid MAXDELAY and MINDELAY parameters (MAXDELAY - MINDELAY too big).

Table 6-14: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-243	CONFIG_FAILURE_MINDELAY_MAXDELAY_WIN_SIZE_SMALL	ERROR: Failed to successfully update the ECF parameter set because of invalid MAXDELAY and MINDELAY parameters (MAXDELAY - MINDELAY too small).
-242	CONFIG_FAILURE_MINDELAY_MAXDELAY_WIN_SIZE_ORDER	ERROR: Failed to successfully update the ECF parameter set because of invalid MAXDELAY and MINDELAY parameters (MINDELAY > MAXDELAY).
-241	CONFIG_FAILURE_SAMPLES2PROCESS	ERROR: Failed to successfully update the ECF parameter set because of invalid SAMPLES2PROCESS parameter.
-240	CONFIG_FAILURE_DATAPATH_GAIN	ERROR: Failed to successfully update the ARCH parameter set because of invalid DATAPATH_GAIN parameter.
-239	CONFIG_FAILURE_METERLENGTH	ERROR: Failed to successfully update the MET parameter set because of invalid METERLENGTH parameter.
-238	CONFIG_FAILURE_ENABLELINEARCORRECTION	ERROR: Failed to successfully update the ECF parameter set because of invalid ENABLELINEARCORRECTION parameter.
-237	CONFIG_FAILURE_NZCOUNTTHRESHOLD	ERROR: Failed to successfully update the MET parameter set because of invalid NZCOUNTTHRESHOLD parameter.
-236	CONFIG_FAILURE_RXPHASESTEP	ERROR: Failed to successfully update the ECF parameter set because of invalid RXPHASESTEP parameter.
-235	CONFIG_FAILURE_CORRECTION_BW	ERROR: Failed to successfully update the ECF parameter set because of invalid CORRECTION_BW parameter.
-234	CONFIG_FAILURE_ODDEXPTHRESHOLD	ERROR: Failed to successfully update the ODD parameter set because of invalid ODDEXPTHRESHOLD and ODPEXPTHRESHOLD parameters (ODDEXPTHRESHOLD > ODPEXPTHRESHOLD).
-233	CONFIG_FAILURE_RXINPUTFORMAT	ERROR: Failed to successfully update the ECF parameter set because of invalid RXINPUTFORMAT parameter.
-232	CONFIG_FAILURE_ODDENABLE	ERROR: Failed to successfully update the ODD parameter set because of invalid ODPENABLE parameter.
-231	CONFIG_FAILURE_PORTCONTROL	ERROR: Failed to successfully update the DCL parameter set because of invalid PORTCONTROL parameter.
-230	CONFIG_FAILURE_QMCNUMESTAVERAGE	ERROR: Failed to successfully update the QMC parameter set because of invalid QMCNUMESTAVERAGE parameter.

Table 6-14: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-229	CONFIG_FAILURE_QMCTXENABLE	ERROR: Failed to successfully update the QMC parameter set because of invalid QMCTXENABLE parameter.
-228	CONFIG_FAILURE_CAPTUREMODE_PIW	ERROR: Failed to successfully update the CAP parameter set because of invalid CAPTUREMODE parameter (PIW hardware is not available).
-227	CONFIG_FAILURE_LEAKAGEVALUE	ERROR: Failed to successfully update the ECF parameter set because of invalid LEAKAGEVALUE parameter.
-226	CONFIG_FAILURE_DAMPINGVALUE	ERROR: Failed to successfully update the ECF parameter set because of invalid DAMPINGVALUE parameter.
-225	CONFIG_FAILURE_SPECTRALINVERSION	ERROR: Failed to successfully update the ECF parameter set because of invalid SPECTRALINVERSION parameter.
-224	CONFIG_FAILURE_MINDELAY_BIG	ERROR: Failed to successfully update the ECF parameter set because of invalid MINDELAY parameter (MINDELAY too big).
-223	CONFIG_FAILURE_LS_REGULARIZATION	ERROR: Failed to successfully update the ECF parameter set because of invalid LS_REGULARIZATION parameter.
-222	CONFIG_FAILURE_QMCGAINMU	ERROR: Failed to successfully update the QMC parameter set because of invalid QMCGAINMU parameter.
-221	CONFIG_FAILURE_QMCPHASEMU	ERROR: Failed to successfully update the QMC parameter set because of invalid QMCPHASEMU parameter.
-220	CONFIG_FAILURE_QMCOFFSETMU	ERROR: Failed to successfully update the QMC parameter set because of invalid QMCOFFSETMU parameter.
-219	CONFIG_FAILURE_MINUPDATETIME	ERROR: Failed to successfully update the DCL parameter set because of invalid MINUPDATETIME parameter.
-218	CONFIG_FAILURE_DCLSTARTUPDAMPING	ERROR: Failed to successfully update the DCL parameter set because of invalid DCLSTARTUPDAMPING parameter.
-217	CONFIG_FAILURE_ODDPS0THRESHOLD	ERROR: Failed to successfully update the ODD parameter set because of invalid ODDPS0THRESHOLD and ODPPS0THRESHOLD parameters (ODDPS0THRESHOLD > ODPPS0THRESHOLD).
-216	CONFIG_FAILURE_INVALID_PORT	ERROR: Invalid port was selected.

Table 6-14: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-215	CONFIG_FAILURE_CAPTUREWINDOWS	ERROR: Failed to successfully update the CAPWIN parameter set because minimum window delay is greater than maximum window delay (MIN_X > MAX_X).
-127	ALIGNMENT_PHASEOFFSET_DETECTION_FAILURE	ERROR: Signal alignment failure (phase offset detection failed).
-126	ALIGNMENT_ZERO_PEAKS_DETECTED	ERROR: No peaks were detected in time delay cross correlation function. Adjust the center of the search window or increase the window size.
-125	ALIGNMENT_RX_OVERFLOW_GAIN_ALIGN	ERROR: Gain alignment overflow. Rx samples cannot gain align to Tx without overflowing the fixed point storage, reduce input signal level to DPD.
-124	AVERAGE_FILTER_GAIN_FAILURE	ERROR: The average gain of the DPD filter is beyond the limit set by the AVERAGE_PWR_CHANGE parameter in the ODD set.
-123	ALIGNMENT_FAILURE_ONE_PEAK	ERROR: Signal alignment failure (failed to achieve coarse alignment metric for single strong peak).
-122	ALIGNMENT_PRE_MSE_FAILURE	ERROR: The mean squared error between the aligned TX and RX samples is too high.
-121	ALIGNMENT_CALIBRATED_LOOPGAIN_FAILURE	ERROR: The loop gain computed for the current capture was different from the loop gain computed during alignment calibration.
-120	ALIGNMENT_FAILURE_NO_CORRELATION	ERROR: Signal alignment failure (failed to detect any correlation between captured TX and RX samples).
-119	DPD_COEF_LUT_OVERFLOW_FAILURE	ERROR: Indicates that a overflow occurred while converting a coefficient set to LUT values.
-118	DPD_COEF_FIXED_POINT_STORAGE_FAILURE	ERROR: Indicates that a overflow occurred while converting a double precision coefficient set to fixed point values.
-115	DPD_COEF_LEASTSQUARES_FAILURE	ERROR: A numerical issue was encountered during the least squares processing (for example, divide by zero during matrix inversion).
-114	CAPTURE_SCA_FAILURE	ERROR: Indicates a failure to capture a statistically sufficient set of samples that passes all SCA criteria for ECF processing.
-113	CAPTURE_HARDWARE_CAPTURE_FAILURE	ERROR: Indicates a failure to capture new samples in the hardware.
-111	HISTOGRAM_FAILURE	ERROR: Failed to detected the histogram complete signal
-110	CAPTURE_DCL_SCA_FAILURE	ERROR: Failed to achieve a capture threshold that matched DCL power.

Table 6-14: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-109	SRX_ZERO_CAPTURES	ERROR: Zeros were captured in the RX capture buffer.
-108	CAPTURE_PEAK_THRESHOLD_FAILURE	ERROR: Peak capture failed
-107	COEF_STORAGE_FAILURE_RX_MODEL	ERROR: Failed to store fixed point rx model coefficients accurately
-106	SRX_CONTROL_WAIT_FOR_IDLE_TIMEOUT	ERROR: sRX control timeout occurred.
-105	SRX_CONTROL_ANTENNA_MATCH_TIMEOUT	ERROR: sRX waiting for antenna match timed out.
-104	SRX_EXTERNAL_CONTROL_TIMEOUT	ERROR: Port switch with external controls timed out.
-103	SRX_DCL_SWITCH_CONTROL_TIMEOUT	ERROR: Port switch during DCL timed out.
-102	C2L_CONVERSION_COMPLETE_TIMEOUT	ERROR: c2l engine timed out before completing conversion.
-101	CAPTURE_INVALID_1FS_REAL	ERROR: Format of rx samples in capture do not appear to be real 1fs mode.
-100	CAPTURE_CONSTANT_SID_FAILURE	ERROR: Capture process captured multiple sids.
-99	CAPTURE_HIRESMONITOR_FAILURE	ERROR: Capture failed, hi-res signal dropped during capture.
-98	CAPTURE_FAILURE_HIST_COUNT_INCREMENT	ERROR: Capture failed, capture histogram count failed to change.
-97	CAPTURE_COMPLETE_CLEAR_FAILURE	ERROR: Capture failed, the capture complete flag failed to clear.
-64	STATE_UNCERTAINTY	ERROR: Detected channel state change during update.
-63	DCL_MULTISET_STATE_MISMATCH	ERROR: State mismatch between channel and capture with multi-set DCL.
-62	STATE_UNCERTAINTY_AT_CAPT_START	ERROR: Detected channel state change during capture.
-3	INVALID_QMC_COMMAND	ERROR: Invalid QMC command.
-2	INVALID_CAPTURE_RAM	ERROR: Requested invalid capture RAM access
-1	INVALID_COMMAND	ERROR: Indicates an invalid command was requested.
0	ZERO	Undefined: During debug, may indicate that `s_axi_ctrl` interface is being used to control host interface.
2	SUCCESSFUL	Indicates successful completion of the requested command.
3	SUCCESSFUL_FRACTIONAL_DELAY_WARNING	Successful update but calibrated fractional delay is above threshold, system may be un-stable.

Table 6-14: **Status Values *(Cont'd)***

Value	Mnemonic	Description
15	SUCCESSFUL_WITH_LOW_RELIABLITY	Successful alignment but with LOW reliability, ensure correct delay.
32	LOW_TX_POWER	The tx power is below minimum level, defined by DCLTXLOWPOWER, to perform DPD updates while DCL is running.
33	LOW_RX_POWER	The rx power is below minimum level, defined by DCLRXLOWPOWER, to perform DPD updates while DCL is running.
34	PORT_SKIPPED	user has requested to skip updating this port during DCL routines
35	PORT_TRACK_SKIPPED	user has requested to skip updating and tracking this port during DCL routines
36	PORT_REQUEST_FORWARD	The DCL routine is currently requesting a forward port.
37	PORT_REQUEST_REFLECTED	The DCL routine is currently requesting a reflected port.
255	OVERDRIVE_DETECTED	Indicates that Over-Drive was detected (estimated expansion is beyond the limit set by ODDEXPTHRESHOLD) for the current ECF update, the coefficients are NOT blocked.
256	OVERDRIVE_PROTECTED	Indicates that Over-Drive was detected (estimated expansion is beyond the limit set by ODPEXPTHRESHOLD) for the current ECF update, the coefficients ARE blocked (that is, not switched into the datapath)
257	EXPANSION_SATURATION_WARNING	Indicates that the measured expansion is beyond the limit set by ODDPS0THRESHOLD for the current ECF update, the coefficients are NOT blocked.
258	EXPANSION_SATURATION	Indicates that the measured expansion is beyond the limit set by ODPPS0THRESHOLD for the current ECF update, future coefficient updates are limited.

Figure 6-15 shows the command shell for the DPD core processes.

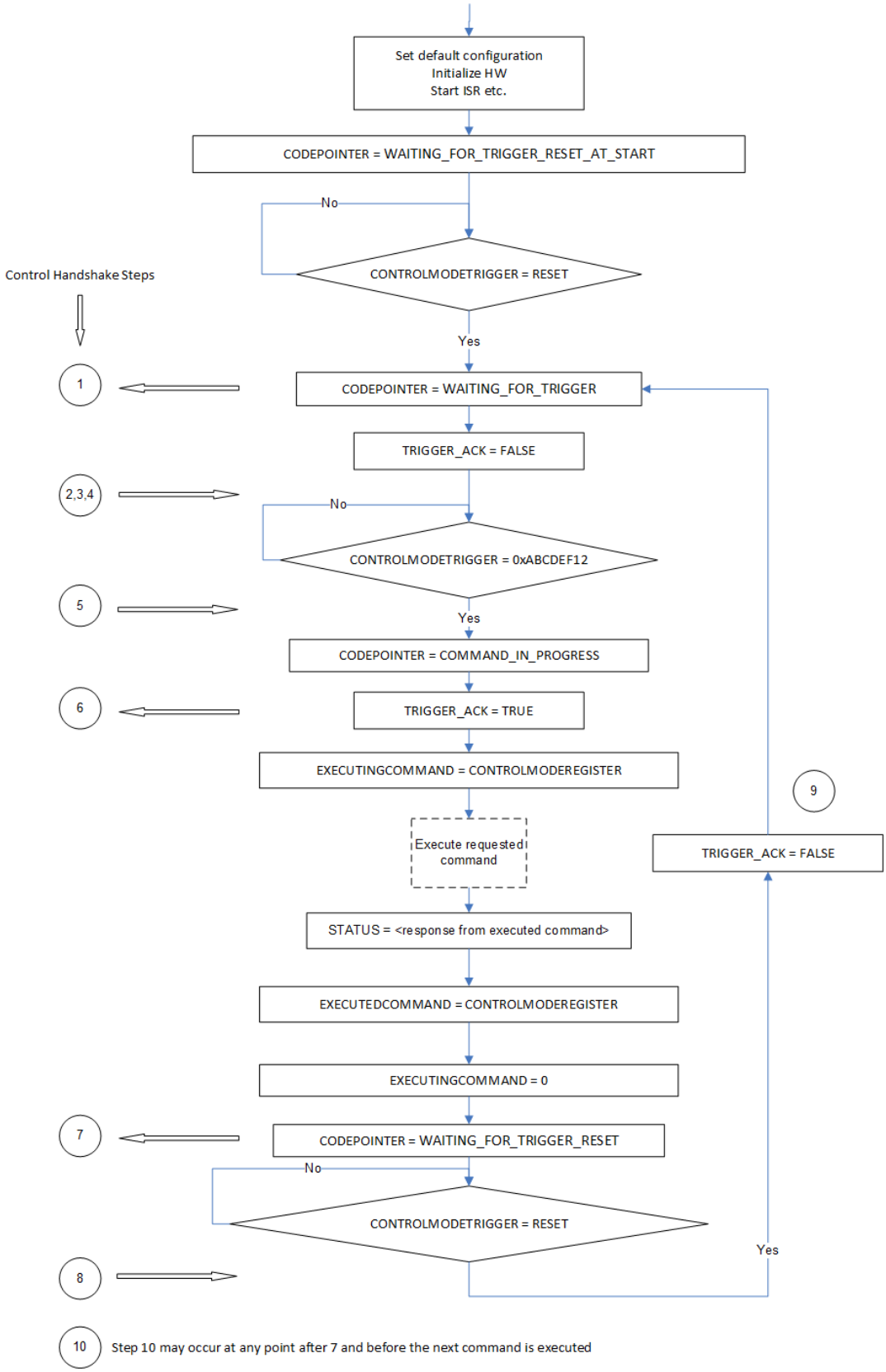


Figure 6-15: **DPD Command Shell**

Updating and Reporting DPD Parameters

The parameters that control the DPD core are detailed in Tables 1 through 7. Some parameters need to be set to suit the installation, particularly for the observation path configuration. For other parameters, the defaults usually provide good performance. Except where indicated, values are unsigned integers written to 32-bit registers. U32.x and S32.x represent unsigned or signed fractional values with x bits after the binary point.

Beginning with DPD v8.0, unless otherwise specified, parameters can be set with port specific values. To facilitate this potentially large increase in unique parameters, DPD v8.1 now utilizes paged parameter sets that share a common region in the host interface. Parameters can be written into the host interface RAM at any time, but are not activated until a control mode is executed.

The process for reporting a DPD parameter set for a specific port via the host interface is:

1. Verify DPD is ready to accept commands (see DPD Application Software Boot Process)
2. (optional) Clear the common parameter region in the host interface.
3. Write the port number of the desired parameter set to PARAMETER_0 register.
4. Send triggered command value corresponding to reporting the desired parameter set (see DPD Application Software Boot Process).
5. Read parameters from common parameter space in the host interface.

The process for updating DPD parameters is:

1. Verify DPD is ready to accept commands (see DPD Application Software Boot Process).
2. (optional) Clear the common parameter region in the host interface.
3. (suggested) Send the triggered command to report the current values of the parameter set to be updated. This step can be skipped as long as all parameters are written to the host interface. Reading the current value immediately before updating allows updating only a subset of parameters when desired.
4. Write bit mask indicating which ports parameter set should be updated to PARAMETER_0 register. Any number of ports can be updated simultaneously using a value of 0x0123XXXX, where XXXX contains the bit mask for which ports to update (LSB is mapped to port 0). All ports are updated if the 0x0123XXXX prefix is not present.
5. Write updated parameter values to the common parameter region.
6. Send triggered command value corresponding to updating the desired parameter set (see DPD Application Software Boot Process).
7. Verify whether the command was successfully executed, if the command returns an error find the offending parameter or port value and correct prior to repeating step 5.

DPD Parameter sets

The parameter sets available in DPD are described here, the corresponding tables describe each parameter associated with each set. The offset described in the tables are relative the base address of the Parameters region of the host interface (see Table 6-9 DPD Host Interface Memory Map Outline).

- Architecture (ARCH) - The ARCH set (see Table 6-15) contains parameters associated with defining the structure of the DPD filter.
- Estimation Core Function (ECF) - The ECF set (see Table 6-16) contains parameters associated with the estimation of the updates of the DPD coefficients. These cover a range of items from defining proper settings for achieving signal alignment between the transmitted and received signals to the general stability of the DPD updates.
- Capture (CAP) - The CAP set (see Table 6-17) is used to define the signal capture method to be utilized.
- Dynamic Control Layer (DCL) - The DCL set (see Table 6-18) contains parameters associated with how DPD autonomously tracks input signal dynamics and determines when to perform DPD coefficient updates.
- Quadrature Modulator Correction (QMC) - The QMC set (see Table 6-19) contains parameters associated with the optional QMC updates.
- Over-Drive Detection (ODD) - The ODD set (see Table 6-20) contains parameters associated with power amplifier protection.
- Meter Length (MET) - The MET set (see Table 6-21) is used to set the measurement interval for the power meters and histograms.
- Capture Windows (CAPWIN) - The CAPWIN set (See Table 6-22) is used to define the optional valid capture windows.

Table 6-15: **Register Set Using UPDATE_ARCH_PARAMETERS and REPORT_ARCH_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
0	ARCH_SEL(1)			When the core is instantiated, the performance architecture IDE selection sets the hardware provision for the degree of the Predistortion function. By default, the ECF recognizes this and computes the appropriate number of coefficients. However, for evaluation purposes the ECF can use a lesser degree than provisioned in the core. Changes to ARCH_SEL force a reset of the DPD coefficients.
1	DATAPATH_GAIN	1 or 2	1	Gain of DPD filter. 1 = -6dB or 2 = -12dB, equates to the maximum expansion that can be applied by the filter. When the gain is increased, the output is first disabled to prevent accidental damage to a PA. The output must be enabled using the ENABLE_OUTPUT command

Notes:

1. When the core is instantiated, the performance architecture selection sets the hardware provision for the pre-distortion function. By default, the ECF recognizes this and computes the appropriate number of coefficients. However, for evaluation purposes the ECF can use a lesser degree than provisioned in the core. ARCH_SEL can be set using the following relationship to enable non-default configurations.

   ```
   ARCH_SEL = 0xABC
   ```

   A: Model reduction (default = 0), increasing this value reduces the order of the modeling. The parameter can be used to reduce the number of parameters in a given model which leads to reduced update time at the potential loss of ACLR correction performance.

   B: Memory complexity (default is set by hardware). This value can be reduced from the maximum (default) down to zero in steps of one.

   C: Model form (default 3). Valid values are 0, 1, 2 or 3 which select four unique models of increasing complexity.

Table 6-16: **Register Set Using UPDATE_ECF_PARAMETERS and REPORT_ECF_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
0	SAMPLES2PROCESS	2000 to CRD	floor (CRD/ 2^{10})*1000	Number of samples to use for the DPD update algorithm; can be reduced to speed up estimation times if performance is assured. CRD is the capture RAM depth of the hardware.
1	LEAKAGEVALUE	0 to 1 (U32.31)	0.999	Tuning parameters for the general update equation: W(n) = W(n-1)*(LEAKAGEVALUE) + e(n)*(DAMPINGVALUE)
2	DAMPINGVALUE	0 to 1 (U32.31)	0.4	Tuning parameters for the general update equation: W(n) = W(n-1)*(LEAKAGEVALUE) + e(n)*(DAMPINGVALUE)
3	LS_REGULARIZATION	S32.0	-10	Matrix regularization constant added during Least Squares (LS) processing. Actual value is 2^X when X is not zero.
4	TXRXRATIO	1 or 2	1	Ratio between the TX and RX complex sample rates.
5	RXINPUTFORMAT	0 or 1	1	0 - the receiver is supplying real IF data. 1 - the receiver is supplying IQ baseband data.
6	SPECTRALINVERSION	0 or 1	0	Baseband spectral inversion. When equal to 1, the I and Q channels are swapped.
7	RXPHASESTEP	S32.32	0.25	(If frequency/fs)*2^{32} for all receiver modes. This is the frequency step required to bring the captured samples down to base-band. For real IF data, the RXPHASESTEP must be negative.
8	CORRECTION_BW	50 to 100	80	Limit correction bandwidth between 50 to 100% (relative to the effective RX I/Q sampling rate).

Table 6-16: **Register Set Using UPDATE_ECF_PARAMETERS and REPORT_ECF_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
9	ENABLELINEARCORRECTION	0 or 1	1	Change to 0, if the linear correction (e.g. gain flatness) within the instantaneous bandwidth (IBW) is overcompensated due to DPD compensating linear distortions on the feedback analog path.
10	MAXDELAY	1000 ≥Δ≥4	1000	Maximum integer sample delay to search over during initial delay adjustment. Δ = (MAXDELAY - MINDELAY).
11	MINDELAY	1000 ≥Δ≥4	0	Minimum integer sample delay to search over during initial delay adjustment. Δ = (MAXDELAY - MINDELAY).

Table 6-17: **Register Set Using UPDATE_CAPTURE_PARAMETERS and REPORT_CAPTURE_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
0	CAPTUREMODE	0,1,2 or 3	0	Capture mode. 0 - use smart capture with SCA. 1 - capture at fixed delay from supplied capture_sync signal. 2 - use software SCA with no hardware assistance. 3 - use PIW (if hardware is available).
1	CAPTUREDELAY	0 to 2^{32}-1	0	For CAPTUREMODE = 1, capture delay from sync in samples at fs.
2	CAPTUREPIWSIZE	0 to 2^{32}-1	0	For CAPTUREMODE = 3, window duration in milliseconds (if hardware is available). When set to 0, the window uses intervals between 1 and 2 METERLENGTH (default).
3	HIRESMONITOR	0 or 1	0	Enable monitoring of hi-res bit.

Table 6-18: **Register Set Using UPDATE_DCL_PARAMETERS and REPORT_DCL_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
0	DCLALGORITHM(1)	0, 1, 2 or 3	0	0 - Multi-set with user defined capture mode and RUN_UPDATE_COEFFICIENTS updates. 1 - Multi-set with enhanced high PAPR signal capture using RUN_UPDATE_COEFFICIENTS_V2 updates. 2 - Single-set with user defined capture mode and RUN_UPDATE_COEFFICIENTS updates. 3 - Single-set with enhanced high PAPR signal capture using RUN_UPDATE_COEFFICIENTS _V2 updates.
1	PORTCONTROL(1)	0, 1 or 2	0	SRX antenna select method 0 - Fixed time delay. 1 - Software handshake. 2 - Hardware SRX control stream.
2	PORTSEQUENCING(1)	0 or 1	0	SRX antenna port sequence 0 - linear 1 - random
3	SRXSELECTDELAY(1)	0 to 2^{32}-1	0	Define time delay for antenna select. 0 - Uses default of between 1 and 2 METERLENGTH intervals (ensures 1 full interval before switch). Non-Zero - User defined delay (t = (SRXSELECTDELAY / dpd_aclk).
4	DCLSKIPPORT(1)	0 to (2^{NP} – 1)	0	Bit mask indicating which ports to skip for DCL DPD updates. A '1' in bit location n of this word will cause the DCL to skip over port n while running in DCL mode. NP is the number of ports.
5	ENABLEVSWRPWRMSR(1)	0 or 1	0	Enable VSWR power measurements.

Table 6-18: **Register Set Using UPDATE_DCL_PARAMETERS and REPORT_DCL_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
6	MINUPDATETIME[(1)]	0 to 1000	0	Set the minimum amount of time that should be spent on each antenna in milliseconds. If the per antenna update time is very fast (or non-existent as is the case when there is a low power condition), this parameter can control the minimum amount of time that is spent in the SRX available cycle.
7	DCLPSTEP	U32.0	1000	Threshold decay step for single set DCL mode.
8	DCLTXLOWPOWER	U32.0	2102	Low TX power threshold to stop attempting DPD coefficient updates. The threshold in dBFS is 10*log10(threshold/2^22).
9	DCLRXLOWPOWER	U32.0	4204	Low RX power threshold to stop attempting DPD coefficient updates. The threshold in dBFS is 10*log10(threshold/2^22).
10	DCLSTARTUPDAMPING	0 to 100	10	Transition period used for transitioning the damping used for DPD updates from 1.0 linearly down to DAMPINGVALUE. This parameter can be used to improve the initial convergence times (larger damping) while keeping good tracking performance (smaller damping). This cycle is reset when the DCL parameters are updated or the DCL is reset.

Notes:

1. These parameters cannot be set uniquely for each port. All ports will take the last value set.

Table 6-19: **Register Set Using UPDATE_TXQMC_PARAMETERS and REPORT_TXQMC_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
0	QMCTXENABLE	0 or 1	0	Enable single tap TX QMC. This feature is only available when RXINPUTFORMAT = 0 and when the TX signal is transmitted using zero-IF (ZIF) architecture.
1	QMCNUMESTAVERAGE	1 - 256	1	QMC updates to average prior to updating the hardware registers (after initial convergence).
2	QMCGAINMU	U32.24	1	Scaling term applied to the gain error prior to updating the register (after initial convergence).
3	QMCPHASEMU	U32.24	1	Scaling term applied to the phase error prior to updating the register (after initial convergence).
4	QMCOFFSETMU	U32.24	1	Scaling term applied to the offset errors prior to updating (after initial convergence).

Table 6-20: **Register Set Using UPDATE_ODD_PARAMETERS and REPORT_ODD_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
0	AVERAGE_PWR_CHANGE	U32.24	1.2589	Maximum average power change allowed (10*log10(x) for dB).
1	ODPENABLE	0, 1 or 2	1	0 - Disable protection. 1 - Enable peak saturation referenced to 0dBFS at the output. 2 - Enable predictive ODD protection.

Table 6-20: **Register Set Using UPDATE_ODD_PARAMETERS and REPORT_ODD_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
2	ODDPS0THRESHOLD	U32.15	0.7943	Threshold for Over-Drive Detection with peak saturation method. In dB 20*log10(X).
3	ODPPS0THRESHOLD	U32.15	0.8912	Threshold for Over-Drive Protection with peak saturation method. In dB 20*log10(X).
4	ODDEXPTHRESHOLD	U32.8	1.5859	Threshold of Over-Drive Detection with estimated expansion method. In dB 20*log10(X).
5	ODPEXPTHRESHOLD	U32.8	1.7773	Threshold of Over-Drive Protection with estimated expansion method. In dB 20*log10(X).

Table 6-21: **Register Set Using UPDATE_MET_PARAMETERS and REPORT_MET_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
0	METERLENGTH	2^{18} to 2^{32}-1	2457600	Number of samples for measurements block processing. Should be a multiple of any frame size and ≥ 10 ms. Must be a multiple of 2 for two-phase or two-phase hybrid implementations. The start of the measurement interval is not synchronized to the data framing so care should be taken when setting the METERLENGTH equal to the frame length. This is especially true when testing with repeated data frames. In these cases it is best to set the METERLENGTH = 1.05 * frame length to ensure best performance.
1	NZCOUNTTHRESHOLD	0 to 2^{11}-1	0	Magnitude threshold used to determine the number of samples used in computing the power meter averages. In dBFS the threshold level is 20*log10(NZCOUNTTHRESHOLD / 2^{11})

Table 6-22: **Register Set Using UPDATE_CAPWIN_PARAMETERS and REPORT_CAPWIN_PARAMETERS**

Parameters base Offset	Mnemonic	Values	Default	Notes
0	MIN_0	S32.0	-1	Start delay for window 0.
1	MAX_0	S32.0	-1	Stop delay for window 0.
2	MIN_1	S32.0	-1	Start delay for window 1.
3	MAX_1	S32.0	-1	Stop delay for window 1.
4	MIN_2	S32.0	-1	Start delay for window 2.
5	MAX_2	S32.0	-1	Stop delay for window 2.
6	MIN_3	S32.0	-1	Start delay for window 3.
7	MAX_3	S32.0	-1	Stop delay for window 3.
8	MIN_4	S32.0	-1	Start delay for window 4.

Table 6-22: **Register Set Using UPDATE_CAPWIN_PARAMETERS and REPORT_CAPWIN_PARAMETERS *(Cont'd)***

Parameters base Offset	Mnemonic	Values	Default	Notes
9	MAX_4	S32.0	-1	Stop delay for window 4.
10	MIN_5	S32.0	-1	Start delay for window 5.
11	MAX_5	S32.0	-1	Stop delay for window 5.
12	MIN_6	S32.0	-1	Start delay for window 6.
13	MAX_6	S32.0	-1	Stop delay for window 6.
14	MIN_7	S32.0	-1	Start delay for window 7.
15	MAX_7	S32.0	-1	Stop delay for window 7.
16	MIN_8	S32.0	-1	Start delay for window 8.
17	MAX_8	S32.0	-1	Stop delay for window 8.
18	MIN_9	S32.0	-1	Start delay for window 9.
19	MAX_9	S32.0	-1	Stop delay for window 9.

Notes:

1. The optional valid capture windows can be defined for systems that supply a valid capture sync signal (see Table 3-3: **TUSER Field Packing for each Byte**) to the DPD core. Valid windows are defined when MAX_X >= MIN_X and MAX_X is less than the capture sync interval. To disable a window the MAX_X and MIN_X should be set to -1.
2. See Setting Valid Capture Windows for more details on various CAPTUREMODE usage.

Monitors

Information available from the host interface RAM is detailed in Table 6-23.

Table 6-23: **Monitors**

Address	Mnemonic	Description
32	REPOSITORY	Software build Change List (CL) number. See AR#66684 for more information.
33	BUILDDATE	Software build date: - day = mod(BUILDDATE,32) - month = mod((BUILDDATE-day)/32,12) - year = 2000 + ((BUILDDATE-day)/32 - month)/12
34	BUILDTIME	Software build time: - seconds = mod(BUILDTIME,60) - minutes = mod((BUILDTIME-seconds)/60,60) - hour = ((BUILDTIME-seconds)/60 - minutes)/60

Table 6-23: **Monitors *(Cont'd)***

Address	Mnemonic	Description
35	BUILDSETTINGS	bits[3:0] - number of antennas bits[6:4] - number of phases bit[7] - hybrid phases bits[11:8] - filter memory depth bits[15:12] - capture RAM depth, L = 2^(bits[15:12] + 9) bits[19:16] - acceleration level bit[20] - PIW capture available bit[21] - Multi set DCL available bit[22] - low resolution coefficients
36	HWVERSION	Hardware version - bits[31:24]-Device family - bits[23:16]-Major version - bits[15:8] -Minor version - bits[7:0] -Revision Device family: 1 - Automotive Zynq 2 - Zynq-7000 3 - Zynq UltraScale+ 4 - Defense grade Zynq-7000
62	SRXQDC	Accumulation of the Q channel over the METERLENGTH interval (S32.30 value)
63	SRXIDC	Accumulation of the I channel over the METERLENGTH interval (S32.30 value)
64	RUNTIMELSW	32-bit LSW of a time monitor that counts the number of measurement intervals. The timer is reset when the meter length is updated or when the DCL parameters are set.
65	RUNTIMEMSW	MSW of the time monitor.
66	SRXLSW	32-bit LSW of the receiver power -bits [31:28] - active SRX port value -bits [27:0] - lower 28 bits of power meter
67	SRXMSW	32-bit MSW of the receiver power -bits [31:28] - active SRX port value -bits [27:0] - upper 28 bits of power meter
68 + 5n	TXPOWERLSW	32-bit LSW of the transmit power of port n (where n ranges from 0 to number of ports minus one).
69 + 5n	TXPOWERMSW	32-bit MSW of the transmit power of port n (where n ranges from 0 to number of ports minus one).
70 + 5n	TXPOWERCOUNT	Number of non-zero samples in the power measurement of port n (where n ranges from 0 to number of ports minus one).

Table 6-23: **Monitors *(Cont'd)***

Address	Mnemonic	Description
71 + 5n	FCM	Proprietary frequency content metric (FCM) of port n (where n ranges from 0 to number of ports minus one).
72 + 5n	ACTIVE_SID	Active signal identifier of port n (where n ranges from 0 to number of ports minus one). bits [15:0] - active signal identifier. bit [16] - latched overflow indicator. bit [17] - overflow indicator for currently active signal identifier. bits [31:18] - signal identifier associated with latched overflow.

All TX ports have their own dedicated power meters at the input of the filter. Each transceiver power is a 64-bit value representing the sum of the individual powers of the number of samples specified in METERLENGTH, divided by 256. The 64-bit power reading is, power = (SRXLSW) | ((SRXMSW) << 32). To convert to average power in dBFS, the formula is:

$$10 \times \log10(256 \times power/(2^{30} \times TXPOWERCOUNT)).$$

The TXPOWERCOUNT is the number of samples that should be included when computing the average power (0 < TXPOWERCOUNT <= METERLENGTH). TXPOWERCOUNT is the number of TX samples whose magnitude is above the NZCOUNTTHRESHOLD parameter. By default the NZCOUNTTHRESHOLD is set to zero. This threshold can be utilized in TDD systems to ensure the reported average power corresponds to the active part of the waveform. In some applications the input TX sample may have some noise floor that prevents absolute zero values from being applied to input of DPD, in these situations the NZCOUNTTHRESHOLD can be tuned to remove these low signal level samples from being included in the TXPOWERCOUNT (although all samples regardless of their magnitude are included in the power accumulation), see Figure 6-16.

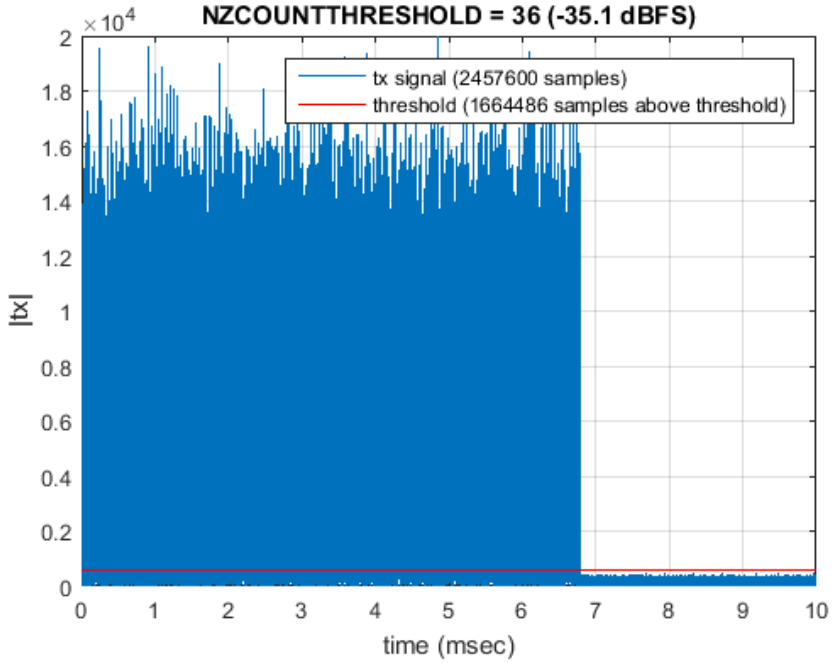


Figure 6-16: **XPOWERCOUNT example**

The SRX power meter, registers (SRXLSW and SRXMSW), is shared between all ports (for normal and VSWR power measurements). The registers have the active SRX port number embedded in the upper nibble of each register. These nibbles can be used to determine which port the SRX power is reporting. The 56-bit SRX power value is srx_power = (SRXLSW & 0x0FFFFFFF) | ((SRXMSW & 0x0FFFFFFF) << 28). To convert the srx_power to average power in dBFS:

$$10 \times \log10(256 \times srx\_power/(2^{30} \times TXPOWERCOUNT).$$

The SRX power reported in the SRXLSW and SRXMSW registers is an unbiased power measurement. The D.C. bias removed in the reported SRX power is reported in the SRXIDC and SRXQDC registers.

Reading the SRX power in when using non-DCL operation

The srx port selection is not synchronized with the power measurement interval. This causes potential power measurements that are partial measurements of two different ports immediately after switching. During these times, both SRXLSW and SRXMSW report 0xF000000 to indicate unknown power reading. This value is present for no more than one measurement interval.

In non-DCL operation the normal or VSWR power can be measured and reported in the SRXLSW and SRXMSW registers. The normal or VSWR signal is request using the PORTNUM register (see Control Handshake).

Note: There is no explicit protection against sending any DPD command while the port is configured for VSWR in non-DCL operation.

Reading the SRX power when using DCL operation

The state of the SRX feedback path is automatically configured when operating under the autonomous DCL control. In this mode valid power measurements are only reported when the Request Type applied to the `m_axis_srx_ctrl` stream is either Normal or VSWR power. When the `m_axis_srx_ctrl` is set to SRX available (unused by DPD) the reported SRX power is set to zero. Under DCL operation the amount of time the SRX is configured to Normal or VSWR power is very short, hence the values reported in the shared SRXLSW and SRXMSW will not be stable long enough to reliably read.

With DCL operation the measured powers are reported into the port specific DCL Monitor registers, see Table 6-25 for DCL Monitors.

Running the DCL

Normally the DPD core is activated using the DCL control modes in Table 6-24. PORTNUM does not need to be specified because the DCL automatically operates on all available ports. Various status monitors are provided in Table 6-24 and can be used to implement error handling. In a multi-port installation, *n* ranges from 0 to the number of ports minus one.

In a system where DPD has been running successfully, the appearance of an undesired status such as, `UPDATE_INPROGRESS` or `LAST_UPDATED_STATUS`, typically indicates an abnormal signal condition such as a failed observation receiver or the occurrence of severe interference. If the ECF does encounter an error, the coefficients are not updated or stored. All ECF parameters are relevant to the DCL.

While operating under DCL control, the DPD core cannot respond to commands using the normal non-DCL command interface. A limited number of commands can be serviced by DPD while DCL is running using the interface described in DCL Commanding.

Table 6-24: **DCL Control Modes**

Mode Number	Mnemonic	Description
19	RUN_DCL	Run the DCL.
20	EXIT_DCL	Stop the DCL while retaining internal state.
21	RESET_DCL	Reset the DCL internal state. Executing SET_DCL_PARAMETERS also does this.

Note: When contacting Xilinx Technical Support, it is useful to have a record of the DCL monitors.

Table 6-25: **DCL Monitors**

Address	Mnemonic	Value	Description
256 + 32*n	FCM	S32.16	Proprietary frequency content metric
257 + 32*n	TX_POWER_NORMAL	U32.30	Average tx power during normal measurement (convert to dBFS as 10*log10(256*x))
258 + 32*n	RX_POWER_NORMAL	U32.30	Average SRx power during normal measurement (convert to dBFS as 10*log10(256*x))
259 + 32*n	LOOPGAIN	U32.24	Calibrated feedback loop gain (convert to dB as 20*log10(x)).
260 + 32*n	COUNTER	U32.0	The total number of ECF updates attempts that have occurred since the function was started. This is the last register written in this table. Detecting a change in this register can be used to indicate all values in this table are stable for reading, subject to the beginning of the next update.
261 + 32*n	UPDATE_INPROGRESS	U32.0	0 - no ECF updates are currently occurring 1 - an ECF update is being computed 32 - low tx power 33 - low rx power 34 - port skipped
262 + 32*n	LAST_UPDATED_SID	U32.0	Last updated signal identifier bits[7:0] - updated SID bits[15:8] - captured SID
263+ 32*n	LAST_UPDATED_STATUS	S32.0	Indicates the returned status of the ECF update code (see Table 6-14).
264+32*n	UPDATING_SID	U32.0	Indicates the currently updating SID.
265+32*n	FCM_CAPT_METRIC	S32.16	Frequency content metric for the latest capture.
267+32*n	SS_PSET0	U32.30	Power level associated with single training. 10*log10(256*SS_PSET0) to convert to dBFS.
268+32*n	SS_PSET0_TARGET	U32.30	Decay level for single set update threshold. 10*log10(256*(SS_PSET0-SS_PSET_DELTA)) for current threshold in dBFS.
269+32*n	MAX_MIN_LUT_VALUE	two packed signed 16 bit values	msw contains most positive value in all datapath LUTs. lsw contains the most negative value in all datapath LUTs.
270 + 32*n	MEASUREDPEAKEXPANSION	U32.24	Measured maximum expansion in the DPD filter (convert to dB as 10*log10(x)).

Table 6-25: **DCL Monitors *(Cont'd)***

Address	Mnemonic	Value	Description
271 + 32*n	MSE	U32.30	Mean squared error between the input TX and received RX signal (convert to dB as 10*log10(x)).
273 + 32*n	IBW_OBW_VALUES	two packed U16.9 values	Estimate of the Instantaneous and Occupied bandwidths (IBW and OBW) of the latest capture samples. bits[15:0] - IBW bits[31:16] - OBW Multiply by the DPD sampling rate to convert these values to Hz.
274 + 32*n	ODDPEAKEXPANSION	U32.8	Peak expansion computed during ODD processing (20*log10(x))
276+ 32*n	PRE_MSE	U32.30	Mean squared error of raw aligned capture (convert to dB as 10*log10(x)).
277 + 32*n	CAPT_LOOPGAIN	U32.24	Current feedback loop gain (convert to dB as 10*log10(x)).
278 + 32*n	TX_POWER_REFLECTED	U32.30	Average tx power during reflected measurement (convert to dBFS as 10*log10(256*x))
279 + 32*n	RX_POWER_REFLECTED	U32.30	Average SRx power during reflected measurement (convert to dBFS as 10*log10(256*x))
281 + 32*n	QMC_GAIN_ERROR	U32.16	Use 20 x log10(QMC_GAIN_ERROR) to convert to dB error.
282 + 32*n	QMC_PHASE_ERROR	S32.16	Use QMC_PHASE_ERROR x 180/pi to convert to error in degrees.
283 + 32*n	QMC_I_OFFSET_ERROR	S32.16	LSB DC offset.
284 + 32*n	QMC_Q_OFFSET_ERROR	S32.16	LSB DC offset.
285 + 32*n	QMC_UPDATE_COUNTER	U32.0	Total number of QMC updates after the function was started.
286 + 32*n	SUCCESSFUL_COUNTER	U32.0	Number of successful DPD updates since the function was started.
287 + 32*n	MEASURED_AVEPWR_GAIN	U32.24	Current average power gain of DPD filter (10*log10(x) for dB)

DCL Commanding

While DCL is operating, the DPD continuously services each antenna sequentially. This loop periodically looks for the DCL_EXIT command or for a request to interleave in another command. A high-level view of this loop is shown in Figure 6-17.

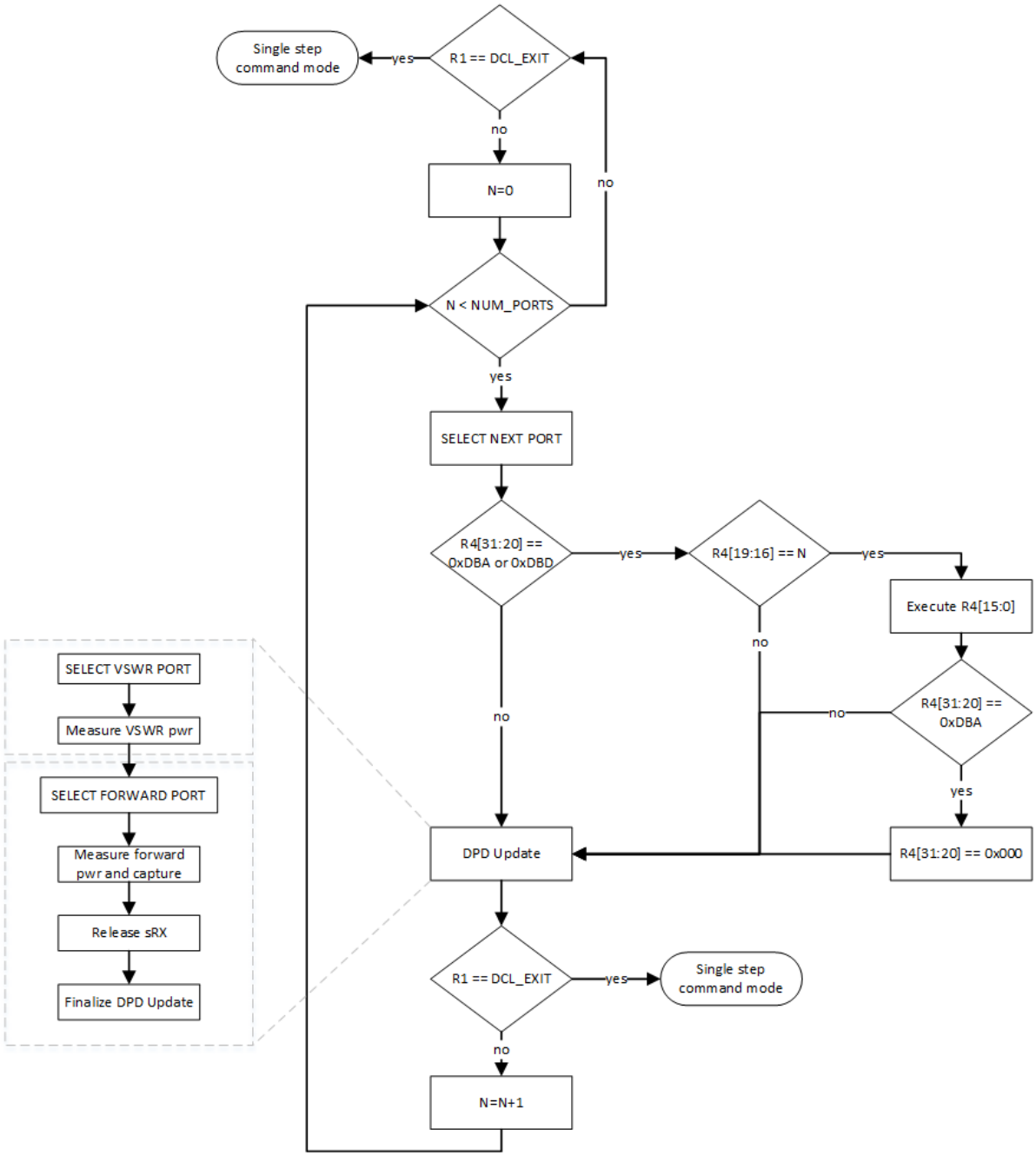


Figure 6-17: **DCL Commanding Loop**

In Figure 6-17, R1 refers to the CONTROLMODEREGISTER register and R4 refers to the PARAMETER_0 register in the host interface.

Commands are requested by writing the proper value into the PARAMETER_0 register while DCL is operating. Each command is a single 32-bit value (i.e. continuously re-run the command once every time the port is activated).

Command structure is:

- 0xDBAPMMXY for "one-shot" commands (i.e. run the command a single time), upon completion of the command this value will change to 0x000PMMXY.
- 0xDBDPMMXY for continuous commands (i.e. continuously re-run the command once every time the port is activated).

"DBA" and "DBD" initiate the command, "P" indicates which port to apply the command, "MM" defines which command and "XY" are command specific. Table 6-26 shows a list of available DCL commands.

Responses are typically copied to the Page transfer region of the host interface (R512 to R1023).

Table 6-26: **DCL Commands**

DCL Command Name	MM	XY	Notes
DCL_AMAM_AMPM	0	--	R512-R681 lsw- ampm (S16.12), msw-amam (U16.15) R682-R851 lsw – mag input for amam/ampm msw-mag input for pa_amam/pa_ampm (U16.15) R852-R1021 lsw- pa_ampm (S16.12), msw-pa_amam (U16.15)
DCL_REPORT_DIAGNOSTICS	1		Y indicates which diagnostics to report ('0' for documented diagnostics) R512-R524.
DCL_ODD_AMAM	2		XY specifies which set to process. 0xFF will use the active set. Any value greater than the available sets will map to set 0.
DCL_PSD	3		Y indicates which capture RAM to process (0-TX pre, 1-RX, 2-TX post), no error checking. X indicates the decimation rate (2X), no error checking.
DCL_RESET_ALIGNMENT	4	--	P=0xF will apply to all ports or this command.
DCL_RESET_COEFFICIENTS	5		Y indicates which set to reset, XY = 0xFF will reset all coefficient sets for the port. P=0xF will apply to all ports or this command.
DCL_GET_HISTOGRAM	6	--	R512-R767 last capture histogram R768-R1023 long term histogram
DCL_CLEAR_C2L_OVERFLOW	7	--	Clears latched overflow error message. P=0xF will apply to all ports or this command.

Table 6-26: **DCL Commands *(Cont'd)***

DCL Command Name	MM	XY	Notes
DCL_GET_SKIP_PORT	8	--	R512 Report current skip port value. Skip port value is a bit mask indicating which antenna ports should be skipped during DCL updating. LSB corresponds to port 0.
DCL_SET_SKIP_PORT	9	--	R512 Set current skip port value. Skip port value is a bit mask indicating which antenna ports should be skipped during DCL updating. LSB corresponds to port 0. While skipped the UPDATE_INPROGRESS will report 34, all other DCL monitor for the skipped port will be zero.

Exiting the DCL

To exit the DCL operating mode the EXIT_DCL command value is written to the CONTROLMODEREGISTER without the normal command triggering process. The DCL routine will periodically check the CONTROLMODEREGISTER for the EXIT_DCL command.

Once the EXIT_DCL command is observed in the CONTROLMODEREGISTER the DCL will exit and DPD will be ready to accept triggered commands again (see Figure 6-17). The delay between writing the EXIT_DCL command and the actual exiting of the DCL routine can be up to a full update interval.

Single Stepping

For fine control, parameter adjustment, debug, general understanding and non-standard applications, DPD software features can be individually activated using the control modes given in Table 6-27. PORTNUM needs to be specified. The registers in Table 6-27 can be used for additional diagnostics. These registers are valid only after the RUN_UPDATE_COEFFICIENTS or CAPTURE_AND_ALIGN commands are executed and apply only to the current antenna.

Table 6-27: **Single Stepping Control Modes**

Mode Number	Mnemonic	Description
1	READ_CONFIGURATION	Refresh the host interface REPOSITORY, BUILDDATE, BUILDTIME, BUILDSETTINGS and HWVERSION registers.
2	RESTORE_DEFAULTS	Restore the default configuration.
3	RUN_UPDATE_COEFFICIENTS_V2	Perform a full update of the DPD coefficients. This command is recommended for high PAR signals, to improve performance. This update mode works only with the "smart capture with SCA" signal capture processing. The user defined CAPTUREMODE is not utilized.

Table 6-27: **Single Stepping Control Modes *(Cont'd)***

Mode Number	Mnemonic	Description
5	RESET_ALIGN_CALIBRATION	Reset alignment calibration. This causes a new calibration on the next attempted alignment.
6	RUN_UPDATE_COEFFICIENTS	Perform a full update of the DPD coefficients. This includes capturing new samples, using the method selected with the CAPTUREMODE parameter, ECF processing and updating the datapath parameters.
7	CAPTURE_NEW_SAMPLES	Trigger a new sample capture sequence. The capture follows the rules set by the CAPTUREMODE parameter.
8	RESET_COEFFICIENTS	Set all coefficient sets to unity gain and update the datapath for pass-through.
9	DPD_OFF	Update the datapath with pass-through unity values without modifying the internally stored coefficients.
10	DPD_ON	Update the datapath from the internally stored coefficients.
27	RUN_ODD_AMAM	Run ODD algorithm on the current coefficients and report the AM/AM response in the host interface. See RUN_ODD_AMAM Function for more information.
28	CLEAR_C2L_OVERFLOW	Clear latched coefficient overflow warning.
29	GET_CAPTURE_PSD	Compute power spectral density of the contents in a capture RAM. See GET_CAPTURE_PSD Function.
30	CAPTURE_AND_ALIGN	Capture samples and run signal alignment routines.
37	COMPUTE_AMAM_AMPM_ RESPONSES	Capture samples and compute the AM/AM and AM/PM responses. See Compute AM/AM and AM/PM Functions.
38	COMPUTE_PA_AMAM_AMPM_RESPONSES	Capture samples and compute the AM/AM and AM/PM responses of the PA. See Compute AM/AM and AM/PM Functions.
39	DISABLE_OUTPUT	Disable the DPD filter output by writing zero coefficients to the datapath.
40	ENABLE_OUTPUT	Enable the DPD filter output after being disabled using DISABLE_OUTPUT command.
41	REPORT_DIAGNOSTICS	Report DPD diagnostics for the current port to host interface.
42	CLEAR_DIAGNOSTICS	Clear DPD diagnostics for the current port.

Single-step diagnostics are stored for each antenna. Use the REPORT_DIAGNOSTICS command to dump the desired antenna diagnostics to the page transfer area of the host interface. Table 6-28 shows the available diagnostics.

Table 6-28: **Signal Stepping Diagnostic Registers**

page transfer base offset	Mnemonic	Value	Description
0	CAPTURED_FCM	S32.16	Frequency content metric (FCM) of the currently captured samples.
1	MEASUREDPEAKEXPANSION	U32.24	Measured maximum expansion in the DPD filter (convert to dB as 10*log10(x)).
2	DCOFFSET	Two packed 16 bit values (S16.0) LSW-Idc, MSW-Qdc	Amount of DC offset removed during signal alignment.
3	LOOPGAIN	U32.24	Gain adjust made to the RX samples during signal alignment.
4	PHASEREAL	S32.30	Real component of the phase rotation applied during signal alignment.
5	PHASEIMAG	S32.30	Imaginary component of the phase rotation applied during signal alignment.
6	ILOOPDELAY	U32.1	Integer delay estimated during signal alignment.
7	FLOOPDELAY	S32.16	Fractional delay estimated during signal alignment.
8	CORRRATIO	S32.30	Ratio of correlation strength between the tx and rx samples and the auto correlation strength of the tx samples.
9	MSE	U32.30	Mean-squared-error (convert to dB as 10*log10(x)).
10	RELIABILITY	U32.3	Reliability metric for alignment routine. Values < 2 indicate potentially incorrect alignment, typically occurs when instantaneous bandwidth (IBW) of signal is >> occupied bandwidth (OBW) of the signal (for example, two widely spaced GSM carriers). DPD command response will return SUCCESSFUL_WITH_LOW_RELIABLITY.
11	ODDPEAKEXPANSION	U32.8	Peak expansion computed during ODD processing (20*log10(x) for dB).

Table 6-28: **Signal Stepping Diagnostic Registers *(Cont'd)***

page transfer base offset	Mnemonic	Value	Description
12	MEASURED_AVEPWR_GAIN	U32.24	Measured average power gain of the DPD filter (10*log10(x) for dB).
13	IBW_OBW_VALUES	two packed U16.9 values	Estimate of the Instantaneous and Occupied bandwidths (IBW and OBW) of the latest capture samples. bits[15:0] - IBW bits[31:16] - OBW Multiply by the DPD sampling rate to convert these values to Hz.
14	BLIND_RX_QMC	two packed S16.12 values	Memoryless RX QMC applied to RX capture. bits[15:0] - Phase imbalance bits[31:16] - Gain imbalance
15	CAPTURED_PWR	U32.15	Power in the signal capture (20*log10(x) for dB).

Signal Analysis

To aid setup and debug, the control modes shown in Table 6-29 give access to the signals processed by the DPD core and measurements made by the DPD core on those signals. Because these analyses are specific to a particular port, ensure that `PORTNUM` is specified appropriately. A capture can be triggered and the captured data, the (transmit) power and histogram can be read out. Transfer of bulk data uses a paged mechanism. Each page is 512 words long and is available at addresses 512-1023.

The histogram integrated over METERLENGTH can be read out. The histograms are 256 samples long. The histogram bins are the number of samples of signal amplitude when the amplitude is divided by 128. The capture power is the sum of the capture signal power over the number of samples specified in SAMPLES2PROCESS.

Table 6-29: **Signal Analysis Control Modes**

Mode Number	Mnemonic	Description
7	CAPTURE_NEW_SAMPLES	Trigger a new sample capture sequence. The capture follows the rules set by the CAPTUREMODE parameter.
22	GET_CAPTURE_RAM_PAGE	Present the page, specified by PARAMETER_0, of the capture RAM data at the host interface RAM page transfer area.

Table 6-29: **Signal Analysis Control Modes *(Cont'd)***

Mode Number	Mnemonic	Description
23	GET_HISTOGRAM	Present the 256 bins of the transmit histogram at the host interface RAM page transfer area.
24	GET_CAPTURE_HISTOGRAM	Present 256 bins of the capture histogram at the host interface RAM page transfer area.
25	READ_CAPTURE_POWER_METERS	Present the values from the capture TX power meter at addresses 512 (LSW) and 513 (MSW) and the capture RX power at addresses 514 (LSW) and 515 (MSW).
26	GET_SWEPT_CAPTURE_POWER	DPD will perform 512 evenly distributed captures using CAPTUREMODE=1 with CAPTUREDELAY between 0 and PARAMETER_0 samples. The captured power at each delay is reported in the page transfer area. See GET_SWEPT_CAPTURE_POWER Function for more information.
29	GET_CAPTURE_PSD	Compute power spectral density of the contents in a capture RAM. See GET_CAPTURE_PSD Function.
37	COMPUTE_AMAM_AMPM_ RESPONSES	Capture samples and compute the AM/AM and AM/PM responses.
38	COMPUTE_PA_AMAM_AMPM_ RESPONSES	Capture samples and compute the AM/AM and AM/PM responses of the PA.

Examples of uses for the signal analysis control modes are to:

- Check the transmit and receive spectra (by analyzing the captured data in a tool such as MATLAB®) and thereby verify that the signal source, RF paths, core interfaces and relevant DPD parameters are correct.
- Check the CFR configuration (by examining the transmit histogram).
- Determine appropriate settings for CAPTUREDELAY in capture mode 1 by examining the capture histogram and powers relative to the measurements over METERLENGTH.

IMPORTANT: *When contacting Xilinx Technical Support, it is useful to have the signal analysis data described in this section. Xilinx offers a fully featured MATLAB®-based GUI debug environment that can be used to quickly evaluate and explore the full capabilities of the DPD solution. Contact Xilinx Technical Support for access to the debug interface environment [Ref 1].*

Reading the Capture RAM Contents

The content of the capture RAM can be read after the following commands have been executed by DPD

- CAPTURE_NEW_SAMPLES
- CAPTURE_AND_ALIGN
- COMPUTE_AMAM_AMPM_RESPONSES
- COMPUTE_PA_AMAM_AMPM_RESPONSES

The current state of the capture RAM is stored in the CAPTURERAMSTATE register. There are three separate capture RAM that can be read. Each capture RAM is L samples long (the actual value of L depends on the size of the capture RAM built in the DPD hardware, see BUILDSETTINGS register) and therefore requires N = L/512 pages to accesses all values in each capture of RAM using the page transfer mechanism. The GET_CAPTURE_RAM_PAGE command along with PARAMETER_0 specifies a page number from 0 to 3N-1. The three capture RAMs are formatted as follows after the CAPTURE_NEW_SAMPLES or CAPTURE_AND_ALIGN commands:

- The first N pages are the transmit data at the input of the DPD filter. Each 32-bit value consists of a concatenation of two 16-bit two's-complement data for the I (LSW) and Q (MSW) samples.
- The second N pages are the receive data that are also a concatenation of two 16-bit two's-complement data as I (LSW) and Q (MSW) (when operating with RXINPUTFORMAT = 0 the receive capture RAM contains real samples after the CAPTURE_NEW_SAMPLES command. In this case the format is RX (n+1) (LSW) and RX (n) (MSW)).

  Note: Starting from DPD v8.1, the RX samples will be -6dB below the TX samples after running the CAPTURE_AND_ALIGN command.
- The third N pages are the transmit data at the output of the DPD filter for non-PIW capture modes. Each 32-bit value consists of a concatenation of two 16-bit two's-complement data for the I (LSW) and Q (MSW) samples.

Compute AM/AM and AM/PM Functions

The DPD software can convert the TX and RX captured samples from I/Q samples into the AM/AM and AM/PM format. A capture is performed and the original samples in the capture RAM are overwritten with AMAM and AMPM responses. These samples can be accessed using the page transfer mechanism.

- The first N pages are the AMAM response. Each 32-bit data consists of a concatenation of two 16-bit values. The LSW is the unsigned input (U16.15) magnitude. The MSW is an unsigned value (U16.12) with the AMAM response.

- The second N pages are the AMPM response, each 32-bit data consists of a concatenation of two 16-bit values. The LSW is the unsigned input (U16.15) magnitude and the MSW is a signed value (S16.13) for the AMPM value.
- The third N pages are unused for the AM/AM and AM/PM reporting.

Figure 6-18 and Figure 6-19 show examples using both the COMPUTE_PA_AMAM_AMPM_RESPONSES and the COMPUTE_AMAM_AMPM_RESPONSES commands. The input magnitude signals associated with the COMPUTE_PA_AMAM_AMPM_RESPONSES will be reduced by the programmed DATAPATH_GAIN. Figure 6-18 and Figure 6-19 have scaled the input magnitude signals from the COMPUTE_PA_AMAM_AMPM_RESPONSES prior to plotting the results.

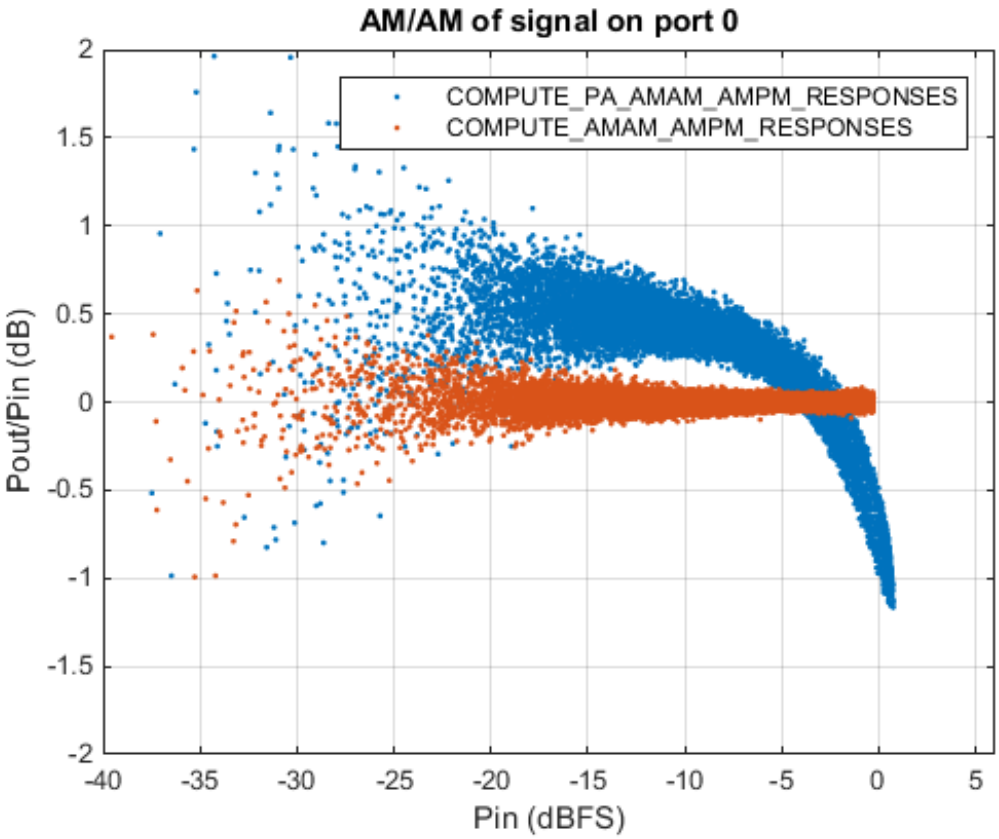


Figure 6-18: **AM/AM Response for Corrected and Uncorrected Signals**

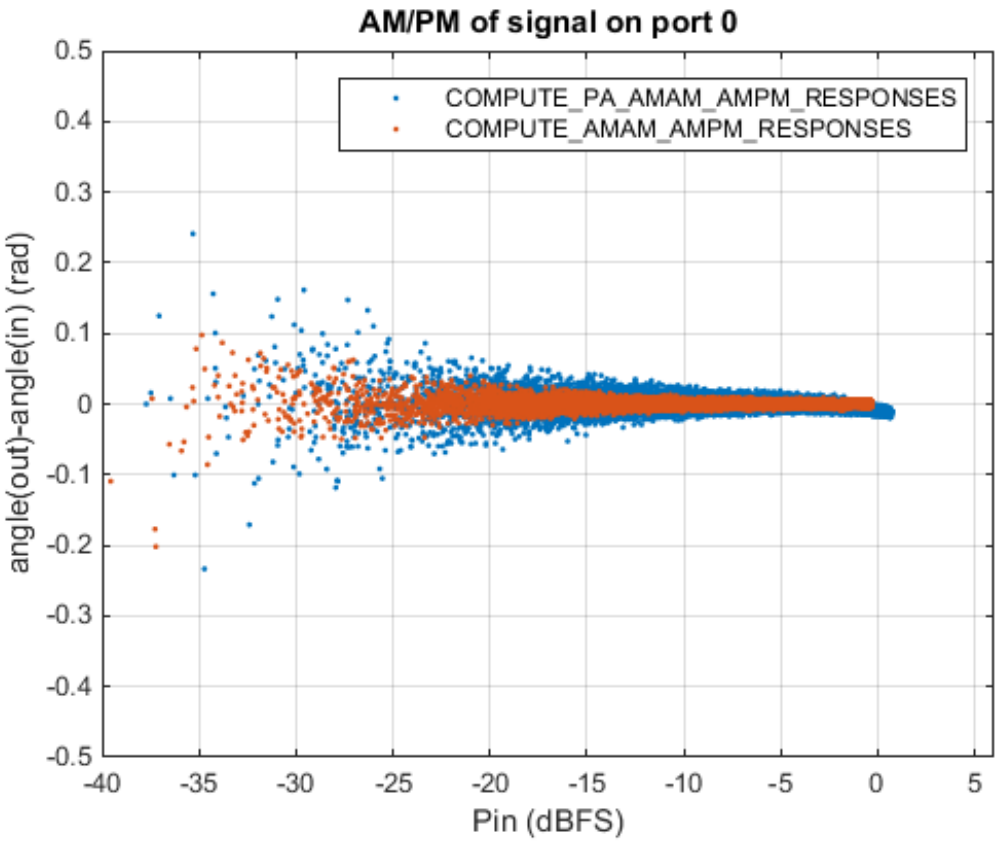


Figure 6-19: **AM/PM Response for Corrected and Uncorrected Signals**

GET_CAPTURE_PSD Function

When external analysis of the capture RAM contents is not readily available, the power spectral density (PSD) of the capture RAMs can be computed in the embedded software. The resulting 128 point complex PSD is stored in the page transfer area.

The capture RAM to be analyzed is selected by writing the desired parameters to the PARAMETER_0 register prior to trigger the command.

PARAMETER_0 = X + (Y * 2^8)

where

X = 0 for TX or 1 for RX capture RAM

and

Y = power of two decimation to apply before power spectral estimation.

The logarithmic result of the PSD estimate is returned in the first 128 registers of PAGE_TRANSFER area with the format of U32.16. Figure 6-20 show an example of the normalized results using four WCDMA carrier sampled at 491.52 MHz. The three plots are for X=0 and Y=0,1 and 2 (that is, no decimation, 2x decimation and 4x decimation).

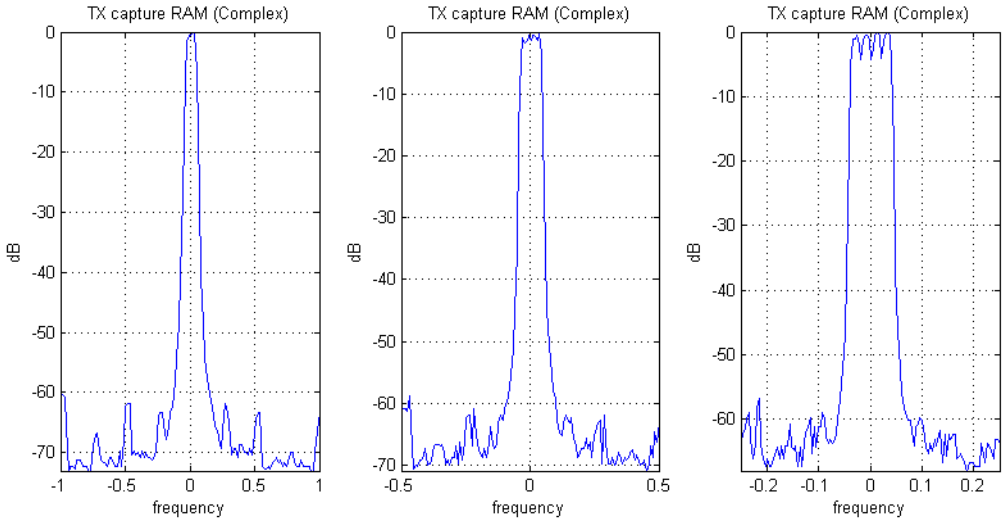


Figure 6-20: **GET_CAPTURE_PSD Results**

RUN_ODD_AMAM Function

An estimate of the DPD filters AM/AM (or expansion) curve can be computed using the same algorithm used for over-drive detection. On completion of the RUN_ODD_AMAM function, the estimated expansion curve is written to the first 256 addresses in the page transfer area (512- 768). The linear gain values are stored as a signed 32-bit value with 16 fractional bits.

Figure 6-20 shows an example expansion curve. The x-axis is fixed at x=1/256, 2/256, … 1 and the y-axis are the values returned from the RUN_ODD_AMAM command. Each axis can be converted to dB/dBFS respectively using 20*log(x).

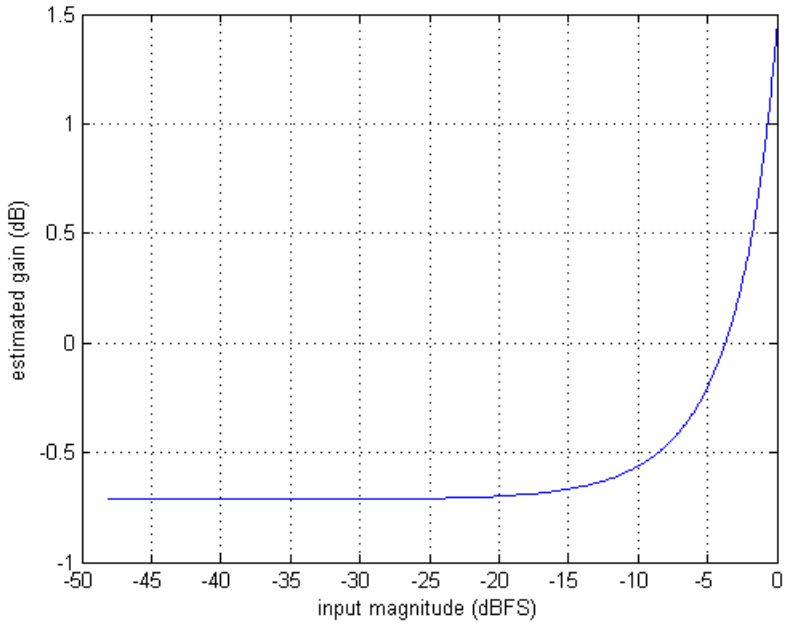


Figure 6-21: **Expansion Curve**

GET_SWEPT_CAPTURE_POWER Function

DPD will perform 512 evenly distributed captures using CAPTUREMODE=1 with CAPTUREDELAY between 0 and PARAMETER_0 samples. The captured power at each delay is reported in the page transfer area.

All page transfer address values are initialized to 0xFFFFFFFF at the start of this function. The first capture occurs with a delay equal to 0 and the power is reported in address 512. The delay is increased by floor(PARAMETER_0/512) and a new capture is performed and its power reported at the next page transfer address. This process continues for the full 512 captures as shown in the example below (Pseudo code for this function):

```
for n=0:511
    CAPTUREDELAY(n+1) = n*floor(PARAMETER_0/512);
        if(capture_sucessful)
            host_interface(512+n) = CAPTURED_PWR;
        else
            host_interface(512+n) = CAPTURE_ERROR_CODE;
        end
end
```

Upon completion of this command, any value with MSB of 1 in the page transfer area indicates that the capture failed.

An example of plotting CAPTUREDELAY against 20*log10(host_interface(512:1024)/2^15) after running this command while transmitting a repeated TM2.0 waveform at 245.76MHz with PARAMETER_0 set to 2457600 is shown in Figure 6-22.

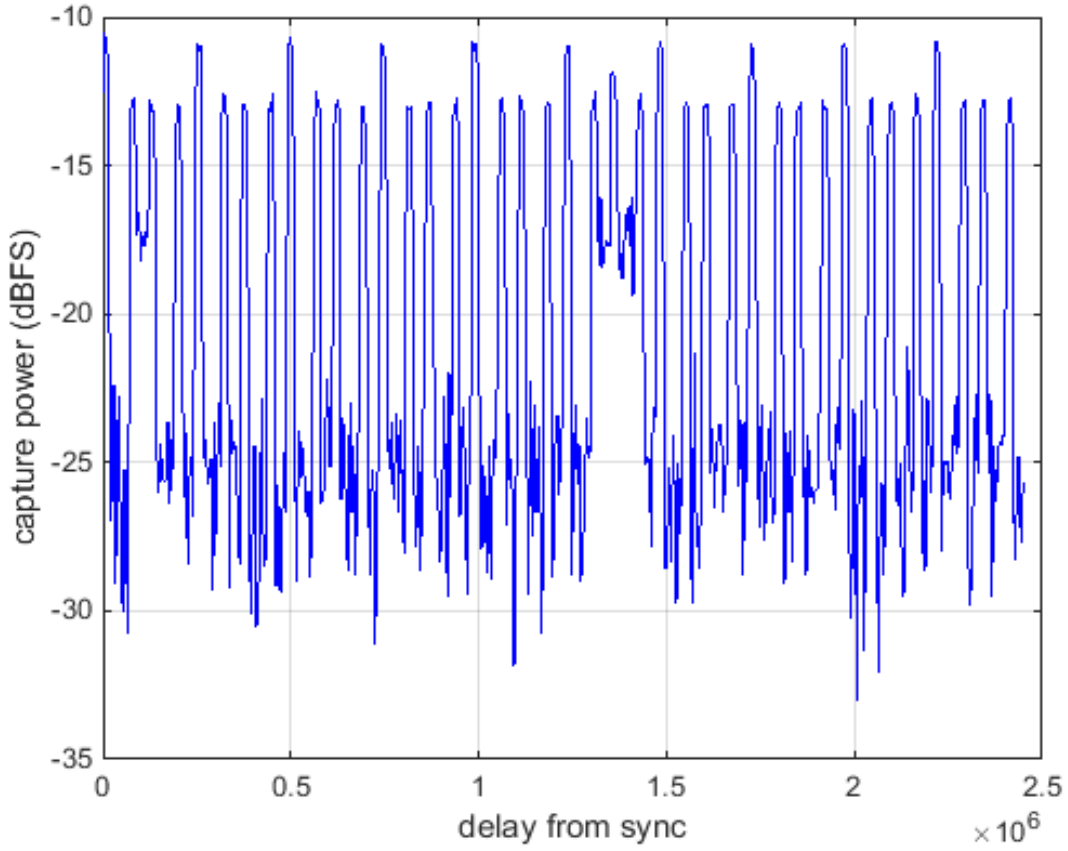


Figure 6-22: **Result of GET_SWEPT_CAPTURE_POWER While Transmitting a Repeated TM2.0 Waveform**

Note: Each capture will occur in different frames, so the result from this command is only deterministic when transmitting a repeated signal.

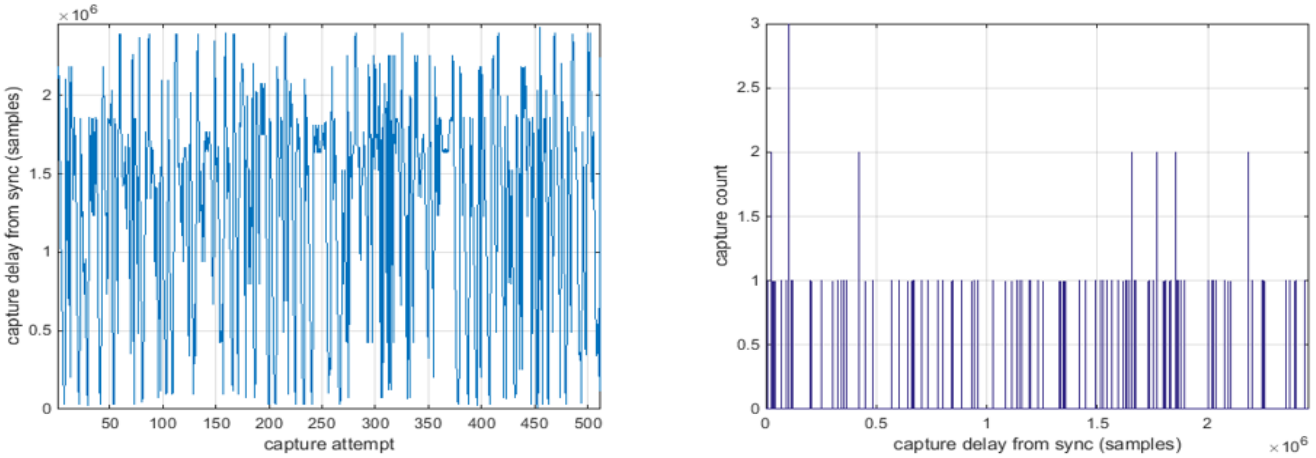
MONITOR_CAPTURE_LOCATION

DPD will perform 512 captures using the user defined CAPTUREMODE. The location within the frame of each capture (number of samples since the last capture sync) is reported in the page transfer area. Pseudo code for this function is shown below.

```
for n=0:511
        if(capture_sucessful)
            host_interface(512+n) = sample since capture sync;
        else
            host_interface(512+n) = CAPTURE_ERROR_CODE;
        end
end
```

All page transfer address values are initialized to 0xFFFFFFFF at the start of this function. Upon completion of this command, any value with MSB of 1 in the page transfer area indicates that the capture failed.

This function can be used to help validate the CAPWIN parameter setting by viewing the histogram of the captured locations against the valid capture windows defined by the CAPWIN parameters (see Setting Valid Capture Windows).



(a) capture location Vs capture attempt

(b) histogram of capture location

Figure 6-23: **Result of MONITOR_CAPTURE_LOCATION While Transmitting a Repeated TM3.1 Waveform**

Reading and Loading Coefficient Set

The DPD and QMC coefficients can be read from the DPD core or loaded into the DPD core for initialization purposes. These processes can be executed only when DCL is not operational.

DPD Coefficient Handling

The coefficient values can only be accessed while DCL is disabled.

Reading DPD Coefficients Sets

1. Write required port to read coefficients from to address PORTNUM in the host interface.
2. Send DPD command REPORT_COEFFICIENTS. See DPD Application Software Boot Process for more information.
3. Read values 512 thru 1,023 (512 values) from the host interface. These values are the coefficient set represented in a proprietary format.

Loading DPD Coefficient Sets

1. Write required port to write coefficients to in address PORTNUM.

2. Write the 512 values read during the "Reading DPD coefficients sets" process into registers 512 thru 1,023 for the desired page number.
3. Trigger DPD command LOAD_COEFFICIENTS. This command loads the coefficients into the DPD memory, but does NOT load them into the DPD filter.
4. (Optional) Trigger command DPD_ON to load the required coefficient set into the DPD filter.

QMC Coefficient Handling

The coefficients values can only be accessed while DCL is disabled.

Reading QMC Coefficients Sets

1. Write required port to read coefficients from to address PORTNUM in the host interface.
2. Trigger the DPD command, REPORT_QMC_COEFFICIENTS
3. Read values 512 through 515 (four values) from the host interface. These values are the coefficient set represented in a proprietary format.

Loading QMC Coefficient Sets

1. Write required port to write coefficients to in address PORTNUM.
2. Write the four values read during the Reading QMC Coefficients Sets process into registers 512 thru 515.
3. Trigger DPD command, LOAD_QMC_COEFFICIENTS. This command loads the coefficients into the DPD memory, but does NOT load them into the QMC registers.
4. (Optional) Trigger command QMC_ON to load the required coefficient set into the QMC registers.

Antenna Selection Options in a Multipath Installation

For multiple-antenna applications, the DPD core assumes that there is an RF or digital switch selecting the various observation paths to route to the sample receiver. The most transparent mode of operation is if the switch control is available as signals in the device, in which case they should be wired to the `m_axis_srx_ctrl` bus of the core. See Sample Receiver (SRX) AXI4-Stream Interfaces in Chapter 3 for more information.

If the switch is accessible only through the application control plane, a software handshake protocol is provided for switching the receiver. This is enabled by setting the appropriate value in the PORTCONTROL parameter for software handshake.

To use this software select mode:

1. Poll CODEPOINTER until the value 131 is read. This is the request for a port switch.

- To avoid constant polling a host software interrupt can be generated outside DPD by monitoring the `m_axis_srx_ctrl_tvalid` signal. This interrupt can be used to indicate when an antenna change request needs to be serviced.

2. Read the ACTIVEPORT register to see which port needs to be switched in to the receive path.
3. Switch the port and acknowledge by writing 0xA5A5A5A5 into the WAITINGONPORTSWITCH register.

In the non-software controlled modes, ACTIVEPORT indicates which port is active, and WAITINGONPORTSWITCH is ignored.

Configuring Software for a New PA

Basic Operational Checks

1. Read the addresses from the host interface; the stated default values should be seen.
2. Execute (for example) the RESET_COEFFICIENTS control mode (see Software Control Modes) to check termination with successful status.

Software Setup and Signal Validation

1. Set up DPD parameters as described in Updating and Reporting DPD Parameters.
2. Read the DPD monitors detailed in Table 6-25.
3. Determine whether the values for the transmit and receive powers are as expected.
4. Perform required operations as detailed in Signal Analysis to ensure that the signal inputs conform to the recommendations in Appendix D, Correction Performance.

Pre-distortion Operations and Achieving Performance

1. Adjust DPD parameters and external setup with the aid of the single-stepping commands (see Exiting the DCL), external measurements, signal analysis operations and interpretation of diagnostics as required.
2. In certain circumstances it is desirable to initialize the DPD coefficients using a known good set of coefficients. The Xilinx DPD solution allows for the reading and loading of coefficients to facilitate the initialization feature (see Reading and Loading Coefficient Set).

Run the DCL with diagnostic monitoring to experience the full operational capability of DPD.

XILINX
ALL PROGRAMMABLE™

Appendix A

Verification, Compliance, and Interoperability

Hardware Testing

Xilinx has conducted extensive functional coverage testing of software and hardware features using the ZC706 and ZCU102 boards with a MATLAB®-based extensive automated test environment. This functional testing was conducted using simplified digital models of a power amplifier. For performance evaluation of various features of the core, the core was integrated into a DFE system and tested with standards-compliant test vectors. Multiple different RF capable Transceiver boards were used as the radio platforms. Various power amplifier pallets, custom, and off-the-shelf packaged power amplifiers were used to validate the performance of the algorithm.

Compliance Testing

The DPD core was tested in a radio design with other radio cores from Xilinx for interface interoperability. Also, during performance testing, standards-compliant signals were used for validating the distortion correction performance and industry standard test equipment was used to validate spectral performance.

XILINX
ALL PROGRAMMABLE™

Appendix B

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

DPD v8.0 is not backwards compatible with any of the previous versions of DPD that were available in ISE and direct migration of any of the previous versions is not possible.

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

Parameter Changes

The DPD v8.0 is not backwards compatible with previous versions and offers a limited set of parameters. The user parameter Embedded Software is removed from the IP GUI options. The required embedded software is delivered through a separate reference design.

A new user parameter named REDUCED_PRECISION has been added. This corresponds to the **Use reduced precision coefficients** checkbox in the IP GUI. See the description for this parameter in Chapter 4, Design Flow Steps for more details. The definition of the LSREGULARIZATION parameter has changed between DPD v7.1 rev2 and DPD v8.0. To maintain compatibility, you must adjust the LSREGULARIZATION values used in DPD v7.1 rev2 by `-round(log2(SAMPLES2PROCESS))`. For example, a system using LSREGULARIZATION = -8 and SAMPLES2PROCESS = 16000 in DPD v7.1 rev2, you must set LSREGULARIZATION = -22 in DPD v8.0 to maintain the same regularization effect.

The general method of setting DPD parameters has changed between DPD v7.1 and DPD v8.0. It now supports per-antenna parameter values. Additionally, there have been a large number of parameter changes. There are a few new parameters. Some parameters have been unpacked and some parameters have been removed. A close review of the new parameters and methods for update and reporting parameters is required before upgrading existing DPD v7.1 installations.

The format of the AMPM response obtained using either COMPUTE_AMAM_AMPM_RESPONSES or COMPUTE_PA_AMAM_AMPM_RESPONSES has changed from S16.16 to S16.13 to support full -pi to +pi values.

Port Changes

The usage of the `m_axis_srx_ctrl` stream has been updated.

Functionality Changes

Starting DPD v8.1 the RX samples will be -6dB below the TX samples after running the CAPTURE_AND_ALIGN command.

DPD v8.0 implements an optional new algorithm for QMC correction, reporting optional VSWR power measurement along with indicating when the srx is not being utilized by DPD. Most of the functionality is identical to DPD v7.1, although there are significant changes to the DPD parameter set usage. There is a support for new devices. The DPD software now supports dual processor SMP mode in addition to bare metal and AMP modes. See Chapter 6, Application Software for more information.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

A MATLAB®-based Advanced Debug Interface is available to assist with the initial setup, learning, and debug of DPD v8.1. This is available through the Digital Pre-Distortion Evaluation Lounge. See the associated User Guide, L*ogiCORE IP Digital Pre-Distortion v8.1 Debug Interface* (UG989) [Ref 1] for a description of the Advanced Debug Interface.

Finding Help on Xilinx.com

To help in the design and debug process when using the DPD, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the DPD. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the DPD Core: AR 54474

Technical Support

Xilinx provides technical support at the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx support web page.

There are many tools available to address DPD design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 12].

Reference Boards

The ZC706 and ZCU102 Xilinx development boards support the DPD v8.1 core. These boards can be used to prototype designs and as a system integration platform. Suitable sample bitstreams are supplied with the Digital Pre-Distortion v8.1 tool, as described in the *Digital Pre-Distortion v8.1 Debug Interface User Guide* (UG989) [Ref 1] as well as with the DFE Demo v3.1 kit, as described in the *DFE Demo v3.1 User Guide* (UG1163)[Ref 19]

Correction Performance

Overview

Because DPD is a system-level function involving subtle interactions between digital, RF and a power amplifier design, it is difficult to make hard-and-fast rules about when the demonstrated performance can be achieved. Prototyping is highly recommended, and Xilinx Technical Support can help with the provision of evaluation platforms in certain circumstances. Nonetheless, the following sections provide some guidance.

Sample Rates

Performance depends on the sample rate of the DPD core. A general guideline is that the pre-distortion bandwidth f_s should be at least five times the signal bandwidth. However factors such as the PA design, the degree of correction required and the signal type come into play. Non-contiguous carrier configurations generally require a higher DPD sample rate than contiguous carrier configurations.

Required Signal Levels and Properties

The supplied design works with 16-bit transmit and up to 16-bit receive signals. The Sample Receiver (SRX) AXI4-Stream Interfaces details how to use fewer bits.

In a live base transceiver system (BTS), the transmit digital signal power varies with call load dynamics. At the maximum output power of the BTS, the scaling of the transmit signal should be such that it is optimally at -15 dBFS RMS power at the input to CFR processing. The output of the CFR must have a known maximum peak value that is determined by the CFR threshold or peak saturation logic. For optimal DPD performance the output of CFR should be scaled such that the peak value is close to full scale. In DPD v8.0, some headroom at the DPD input was required to allow the captured RX samples to be gain aligned to the reference TX samples without overflowing the 16-bit space. In DPD v8.1, the required headroom at the DPD input is minimized by performing the RX gain alignment at -6dB below the desired level. This allows the input signal to be applied near full scale without experiencing the RX overflows after gain alignment.

Figure D-1 shows an example signal line up assuming a maximum input power signal with a specified Peak to Average Power Ratio (PAPR) at the output of CFR. The histogram feature available within DPD can be used to assist in proper setting of the transmit signal levels. By default the DPD core allows for up to 6 dB of expansion.

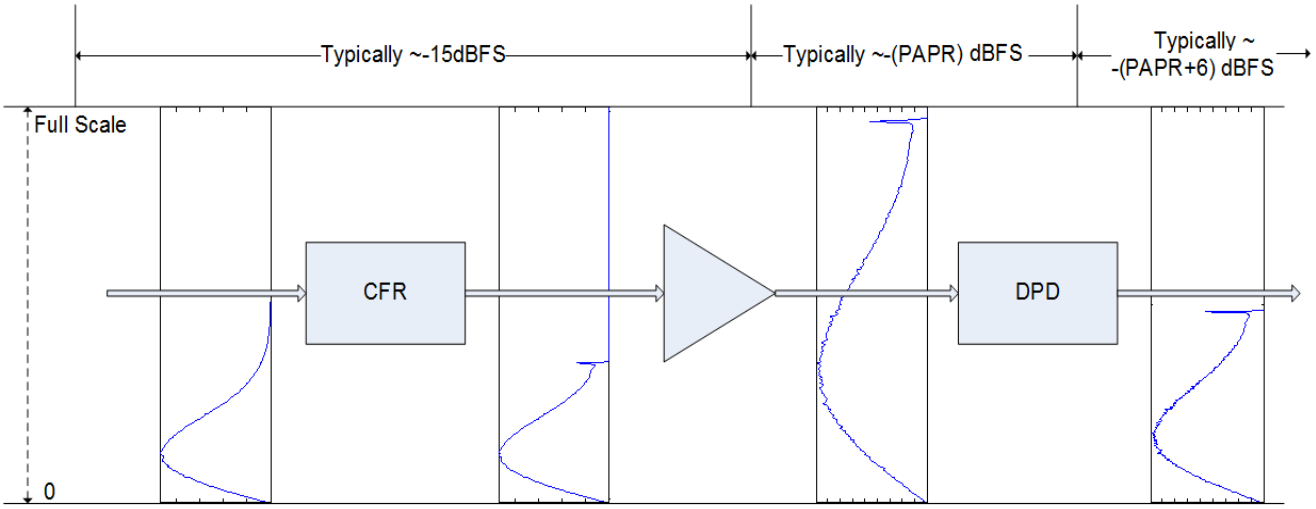


Figure D-1: **Example of Typical Histograms Along DFE Processing Chain**

DPD can, however, operate correctly at lower signal levels, but in any case, the software is set to not attempt pre-distortion at signal levels less than -30 dBFS while running under DCL control (single updates are not subject to the minimum power level checks in software).

The power in the sample receiver should be at -12 dBFS measured on the received signal, as reported by the internal DPD power meters. The signal level control is in the user domain. The test results (available upon request from Xilinx Technical Support) were collected with this scaling, and with a 14-bit ADC. Reasonable pre-distortion correction can be obtained with fewer ADC bits, but this should be explicitly verified in the intended installation.

RECOMMENDED: *CFR is helpful for good pre-distortion performance and highly recommended as a means of optimizing PA efficiency.*

CAUTION! *To ensure good DPD performance the RESET_ALIGN_CALIBRATION or DCL_RESET_ALIGNMENT command must be sent whenever the path between the DPD output and the SRX input is changed (e.g. gain changes, RF component reconfiguration etc.)*

RF Performance

The performance of DPD is intimately related to the quality of the RF design. The RF bandwidth (after accounting for the DAC interpolation filter bandwidth) should be at least five times the signal bandwidth, but special considerations might apply at the edge of the band, depending on the RF filter line-up. These are considerations outside the scope of the digital design that Xilinx offers. Within the RF bandwidth, Xilinx are unable to put limits on the amplitude and phase error that can be tolerated.

Parameters

The default and user-controllable settings described here normally give sufficient control for successful performance in most operational scenarios. However, the DPD core has several internal parameters and settings, and in some cases performance issues can be addressed by changing these. Contact Xilinx Technical Support for assistance.

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this product guide:

1. *Xilinx LogiCORE IP Digital Pre-Distortion v8.1 Debug Interface User Guide* (UG989), registration required
2. *Zynq-7000 All Programmable SoC Overview* (DS190)
3. *AMBA AXI4-Stream Protocol Specification* (ARM IHI 0051A)
4. *Xilinx AXI Design Reference Guide* (UG1037)
5. *AMBA AXI and ACE Protocol Specification* (ARM IHI 0022D)
6. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
7. *Vivado Design Suite User Guide: Designing with IP* (UG896)
8. *Vivado Design Suite User Guide: Getting Started* (UG910)
9. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
10. *Zynq-7000 All Programmable SoC Software Developers Guide User Guide* (UG821)
11. *Simple AMP Running Linux and Bare-Metal System on Both Zynq SoC Processors* (XAPP1078)
12. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
13. *Zynq-7000 AP SoC Technical Reference Manual (TRM)* (UG585)
14. *Vivado Design Suite User Guide: Implementation* (UG904)
15. *ISE to Vivado Design Suite Migration Guide* (UG911)

16. Petalinux Tools Documentation Reference Guide (UG1144)
17. Zynq UltraScale+ MPSoC Software Developers Guide (UG1137)
18. Zynq-UltraScale+ MPSoC Technical Reference Manual (UG1085)
19. *DFE Demo v3.1 User Guide* (UG1163)
20. Xilinx Peak Cancellation Crest Factor Reduction (PC-CFR) product page
21. *Zynq UltraScale+ MPSoC Overview*(DS891)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/03/2017	8.1	• Added support for 2017.2 tools.
06/20/2017	8.1	• Added Support for Filter Memory Depth of 5 • Simulation test bench added to IP and described in Test Bench section • Added Reset and Clock Sequencing section in Chapter 3, Designing with the Core • Added GET_SWEPT_CAPTURE_POWER, MONITOR_CAPTURE_LOCATION, UPDATE_CAPWIN_PARAMETERS, and REPORT_CAPWIN_PARAMETERS in Chapter 6, Application Software • Added new status values CONFIG_FAILURE_CAPTUREWINDOWS PORT_REQUEST_FORWARD, PORT_REQUEST_REFLECTED in Chapter 6, Application Software • Added Table 6-22 - Register Set Using UPDATE_CAPWIN_PARAMETERS and REPORT_CAPWIN_PARAMETERS in Chapter 6, Application Software • Added CAPTURED_PWR in Table 6-28 in Chapter 6, Application Software • Restricted HW accelerators =1 for Memory Depth 6

Date	Version	Revision
12/09/2016	8.0	• Moved to 2016.3 Tools • Added Support for Filter Memory Depth of 6 • Added low resolution coefficient build option • Updated format of embedded AMPM values • Added support for SMP mode under Linux Operating System • Added independent parameters on each port • Caused large changes in the host interface usage, not backward compatible with v7.x • Unpacked various parameters (e.g. MINMAX_) • Added parameter to optionally enable/disable linear correction • Added HWVERSION register • Added support for measuring normal or reflected power and releasing the SRX feedback for other use. • Updated m_axis_srx_ctrl usage • Removed D.C. bias from SRX power meter • Added software boot hold off • Added minimum update time parameter • Added new parameters such as ENABLELINEARCORRECTION, MAXDELAY, MINDELAY, PORTCONTROL, PORTSEQUENCING, PORTSKIP, ENABLEVSWRPWRMSR, MINUPDATETIME, DCLTXLOWPOWER, DCLRXLOWPOWER, DCLSTARTUPDAMPING, CAPTUREPIWSIZE, HIRESMONITOR, NZCOUNTTHRESHOL, ODDPS0THRESHOLD, ODPPS0THRESHOLD, ODDEXPTHRESHOLD, ODPEXPTHRESHOLD • Removed parameter MINMAX_A, MINMAX_B ... MINMAX_H, PAGENUMBER • Replaced PAGENUMBER with PARAMETER_0
06/30/2016	8.0	**EA**: Xilinx Confidential Draft. Approved for external release under NDA only • Removed support for example design • Added sections for Zynq UltraScale + device and SMP mode in Chapter 6, Application Software.

Date	Version	Revision
12/24/2015	7.1	• Added example procedure for Linux image with Petalinux for DPD in AMP configuration. • Added two features to the IP Facts table. • Added step 5 to the Customizing and Generating the Core section. • Updated Capture Memory Depth and Acceleration Level sections. • Added parameters to Table 4-1. • Added new fields DCL Mode and Advanced Capture Mode to the Customizing and Generating the Core section. • Updated figures; changed 7.0 to 7.1. • Made the following changes in the Memory Map table. ◦ Added Capture RAM state to CAPTURERAMSTATE ◦ Added bit[20] and bit[21] to BUILDSETTINGS. ◦ Added option to CAPTUREMODE. ◦ Updated CAPTUREDELAY, METERLENGTH, ODDEXPTHRESHOLDS, ODPENABLE descriptions. ◦ Added DATAPATH_GAIN, AVERAGE_PWR_CHANGE, and DCLPSTEP. ◦ Removed Word Addresses 195 through 211. • Added paragraph about optional Peak-In-Window capture to Measurements Block and Sample Capture Acceptance (SCA) section. • Updated Dynamic Control Layer (DCL) section. • Updated Quadrature Modulator Correction (QMC) section. • Added Peak Saturation section. • Updated address range 192-255 to be reserved in Table 6-5. • Added RUN_UPDATE_COEFFICIENTS_V2 and new values 37 through 42 to Table 6-6. • Added values for 134 and 135 in Table 6-8. • Added values -225, -224, -124, -123, and 257 in Table 6-9. • Made the following changes in Table 6-10. ◦ Changed max value for addresses 140-147 to 1000. ◦ Updated the notes for CAPTUREMODE, CAPTUREDELAY, METERLENGTH, ODPENABLE, and DCLALGORITHM ◦ Added AVERAGE_PWR_CHANGE for address 163 and DCLPSTEP for address 158.
12/15/2014	7.0	• Synchronize document version with core version • Vivado 2014.4 support • Added support for export compliant ADCs • AXI interconnect and address mapping now present and defined within the core • Faster update time due to improved algorithm acceleration • Architecture selection for reduced resource in lower bandwidth applications • Various algorithm improvements
12/18/2013	DRAFT	• Added AMP support

Date	Version	Revision
11/28/2013	DRAFT	• New update algorithm • Vivado Design Suite support added • Zynq series support added
10/16/2012	1.0	Initial Xilinx release. This Product Guide is derived from DS856. The following changes to the core have been made: • ISE 14.3 software support • Extended architectures introduced to improve wideband performance • Optional polyphase data path to increase sample rate support • Hardware accelerator performance improvements • Various resource optimizations • Various sample capture enhancements

Please Read: Important Legal Notices

NOTICE: This pre-release document contains confidential and proprietary information of Xilinx, Inc. and is being disclosed to you as a participant in an early access program, under obligation of confidentiality. You may print one (1) copy of this document for evaluation purposes. You may not modify, distribute, or disclose this material to anyone, including your employees, coworkers, or contractors (these individuals must apply for separate access to the program). This document contains preliminary information and is subject to change without notice. Information provided herein relates to products and/or services not yet available for sale, and provided solely for information purposes and are not intended, or to be construed, as an offer for sale or an attempted commercialization of the products and/or services referred to herein.

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.