Digital Pre-Distortion v13.1

LogiCORE IP Product Guide

Vivado Design Suite

PG076 October 19, 2022

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

Table of Contents

Appendix D: Correction Performance

Appendix E: Additional Resources and Legal Notices

IP Facts

Introduction

The LogiCORE™ IP Digital Pre-Distortion (DPD) core compensates for the non linear transfer function including memory effects of an RF power amplifier. DPD allows a PA to achieve greater efficiency by operating in the non-linear but at higher output power and efficiency region while maintaining spectral compliance. Higher PA efficacy has a direct and significant impact of the energy consumption and cost of the radio.

Features

- Algorithms
  - DPD correction with up to 40 dB of adjacent channel leakage ratio (ACLR) improvement
  - Handles long term memory effects seen with Gallium Nitride (GaN) Amplifiers to meet stringent FCC and 3GPP MIMO requirements
- Physical Configuration Parameters
  - Selection of DPD datapath filter structure
  - Selection of phase options for datapath implementation allowing a resource/sample rate trade-off
  - Selection of one, two, four, six or eight transmit antennas
  - Multiple filter and capture depth options, delivers a highly adaptable solution which enables cost optimized solution for macro Base Stations, massive MIMO 5G RRH, micro Remote Radio Head (RRH), Distributed Antenna System (DAS), and low power PA applications, independent control of filter memory and capture memory depth and acceleration levels allowing for resource versus performance trade-off.
- Software
  - Support for SMP mode under Linux Operating System. Currently, only Xilinx SoCs can host DPD IP, as a PS running Linux is mandatory
  - C API for Host Processor
  - DFE Reference Design included

See Feature Summary for more detailed feature information.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family	Zynq® UltraScale+™ RFSoC: Gen 1/Gen 2/Gen 3 DFE Zynq UltraScale+
Supported User Interfaces	AXI4, AXI4-Lite, AXI4-Stream.
Resources	N/A
Provided with Core	
Design Files	Local Vivado® Repository
Example Design	See the Chapter 5, Reference Design.
Test Bench	See Test Bench
Constraints File	See Constraining the Core
Simulation Model	Not Provided
Supported S/W	DPD Binary included. See Chapter 5, Reference Design for more information.
Tested Design Flows[1]	
Design Entry	Vivado Design Suite
Simulation	Supported. See Test Bench for more details.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 66685
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

[1] For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Digital Pre-Distortion (DPD) compensates for the non linear transfer function, including the memory effects of a RF power amplifier. DPD allows the Power Amplifier (PA) to operate in the non linear region and thus, at higher efficiency while maintaining spectral compliance and in band performance. The PA typically accounts for over 50% of the energy consumption in a radio and higher efficiency has a direct and meaningful impact of the weight and operating costs of the radio.

The solution is targeted for base stations used in mobile technologies and similar applications. It is a combination of hardware and embedded software processes that, together, realize digital pre-distortion correction. Xilinx's DPD delivers a fully engineered, cost effective, 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 Host API which accesses various registers and commands described in Chapter 7, 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
  - Predistorts long term memory effects of GaN amplifiers
  - 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 select URAM in Zynq® UltraScale+™ RFSoC and UltraScale+ MPSoC with URAM
    - Multiple optimized Datapath filter structures offered:
        - Structure 0/Structure 8 (on RFSoC DFE devices) is similar to the DPD v9.0 and earlier IP versions
        - Structure 1/Structure 9 (on RFSoC DFE devices) enables an option that is beneficial to wider IBW/OBW scenarios
        - Structure 10 (on RFSoC DFE devices) enables an option for new flexible memory depth which can benefit Macro cell applications
        - Structure 11 (on RFSoC DFE devices) has similar performance to Structure 10 with reduced hardware utilization
    - Option to select Long Term Memory (LTM), especially when GaN amplifiers are to be predistorted

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.

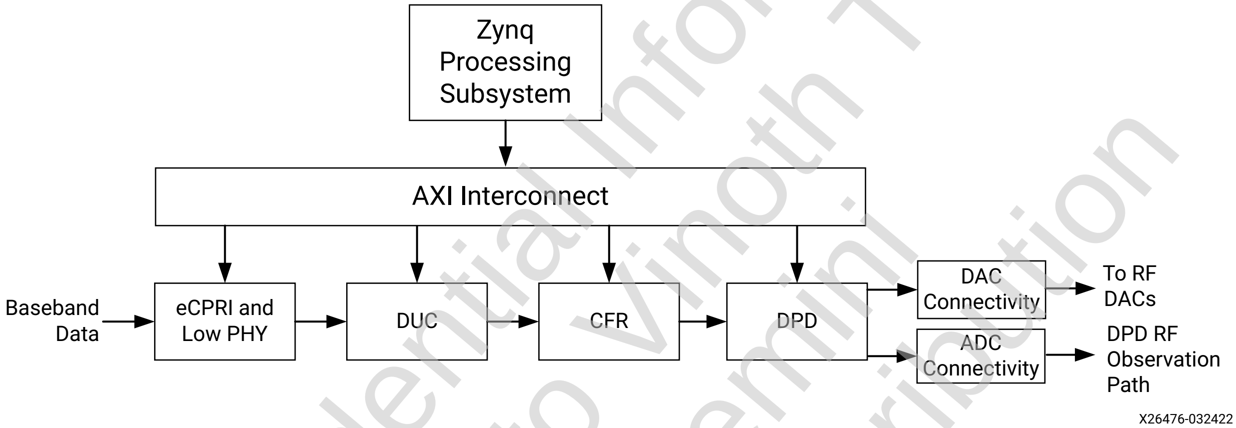


Figure 1-1: **Typical DFE Transmit Path**

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

- **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.

Note:

1. **DAC and ADC Connectivity**: JESD204 protocol provides SerDes serial transceiver links to and from the DACs and ADCs (see JESD204 for information about this core). For Xilinx Zynq UltraScale+ RFSoC devices, DAC and ADCs are integrated and connected through AXI4-Stream interface without any SerDes protocol.
2. **Zynq® Processing System**: Sets up and controls the RF processing chain. Runs the software portion of DPD processing (see the *Zynq UltraScale+ MPSoC Overview* (DS891) [Ref 18].

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided under the terms of the Xilinx Core License Agreement. The module is provided through the DPD Lounge. For full access to all core functionalities 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.

TIP: *To verify that you need a license, check the License column of the IP Catalog. Included means that a license is included with the Vivado Design Suite; Purchase means that you have to purchase a license to use the core.*

License Checkers

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

IMPORTANT: *Vivado Design tools can be used to get the license status for each IP using* `report_ip_status-license_status`.

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), 5G NR, and any combination of these in a multi-Radio Access Technology (RAT) configuration.

IMPORTANT: *DPD IP does not support GSM standards. Contact your local Xilinx sales representative with any questions related to GSM support.*

RF Performance

Contact Xilinx Technical Support to receive example RF performance results (ACLR, SEM) for selected RF PAs.

IP Timing Performance

The DPD IP core timing performance was verified on Zynq UltraScale+™ and Zynq UltraScale+ RFSoC 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 UltraScale+ devices.

Table 2-1 shows the timing performance on a Zynq UltraScale+ ZU9EG (-2LVI speed grade) and Zynq UltraScale+ RFSoC ZU28DR (-2LVI) using the Vivado Design Suite 2021.2 tools. Timing Performance on Zynq UltraScale+ RFSoC Gen 3 devices also meets similar results using Vivado Design Suite 2021.2 tools.

Table 2-1: **DPD Core Timing Performance for Zynq UltraScale+ XCZU9EG (-2LVI) and RFSoC ZU28DR (-2LVI)**

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

Table 2-2: **DPD Core Timing Performance for Zynq UltraScale+ RFSoC DFE (ZCU67DR) Device**

Number of Phases	Filter Structure	dpd_aclk (MHz)	dpd_internal_aclk (MHz)	s_axi_user_aclk (MHz)	s_axi_ctrl_aclk (MHz)	s_axi_ctrl_2x_aclk (MHz)
1	8	491.52	491.52	200	200	400
1	9	491.52	491.52	200	200	400
2	8	491.52	491.52	200	200	400
2	9	491.52	491.52	200	200	400
2	10	491.52	491.52	200	200	400
2	11	491.52	491.52	200	200	400
4	8	491.52	491.52	200	200	400
4	9	491.52	491.52	200	200	400

Resource Utilization

DPD IP (Hardware and Software) uses resources in Zynq UltraScale+/Zynq UltraScale+ RFSoC Programmable Logic (PL) and 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 local PS double data rate (DDR) memory. For details on DDR memory configurations supported by particular parts, refer to *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 15].

When running DPD as an application on SMP Linux, DPD executable requires about 360 KB in the root file system. Typically, resident set size (RSS) reported by ps command is 6.57 MB when running DCL with eight antennas. Xilinx recommends characterizing 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

The latency of the core is primarily a function of Filter Memory Depth and Number of Phases. Table 2-3 and Table 2-4 show the latency of the `dpd_aclk` core in cycles for PL-only IP. Table 2-5 shows latency of the `dpd_aclk` core in cycles for IP on RFSoC DFE (using PL + DFE NL FIR Primitives).This is measured from input `s_axis_din_tvalid/s_axis_din_tlast` to output `m_axis_dout_tvalid/m_axis_dout_tlast`.

Note: When a QMC is removed from transmit datapath, the latency reduces by 6 cycles for non-hybrid designs and by 3 clock cycles for hybrid designs (PL-only).

Table 2-3: **Latency of PL-Only Configuration of Core (dpd_aclk Cycles) (Long Term Memory Disabled)**

Filter Memory Depth	Latency (cycles)		
	Phases = 1	Phases = 2	Phases = 2 (Hybrid)
1	66	66	45
2	70	69	47
3	73	72	48
4	77	75	50
5	81	89	52
6	84	81	54

Table 2-4: **Latency of PL-Only Configuration of Core (dpd_aclk Cycles) (Long Term Memory Enabled)**

Filter Memory Depth	Latency (cycles)		
	Phases = 1	Phases = 2	Phases = 2 (Hybrid)
1	81	82	65
2	81	81	65
3	81	82	65
4	81	81	65
5	81	82	65
6	82	82	66

Table 2-5: **Latency of Core (dpd_aclk Cycles on RFSoC DFE Devices)**

Filter Structure	Filter Memory Depth	Latency (cycles)					
		Long Term Memory = 0			Long Term Memory = 1		
		Phases = 1	Phases = 2	Phases =4	Phases = 1	Phases = 2	Phases = 4
8	5	47	36	34	73	73	73
9	5	52	47	45	73	73	73
10	5	–	68	-	–	77	-
11	5	-	45	-	-	73	-

Port Descriptions

Figure 2-1 displays the signal names of the DPD HW component. Table 2-6 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.

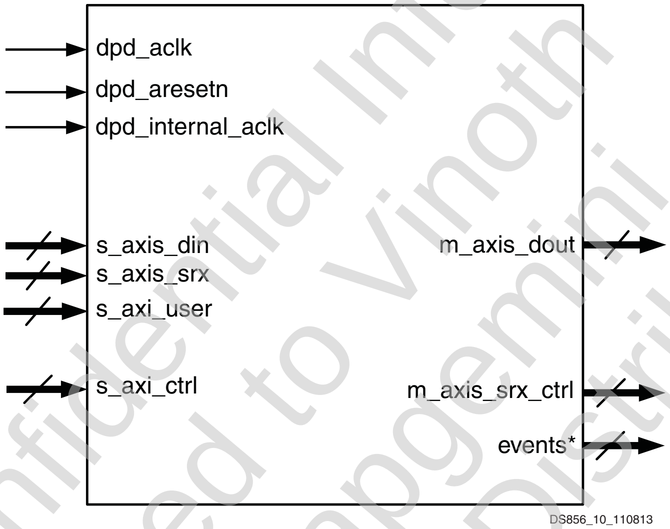


Figure 2-1: **DPD IP Symbol (Input/Output Ports)**

Table 2-6: **Top-Level I/O for the DPD IP Core**

Names	I/O	Description
dpd_aclk	I	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	I	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	I	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]	I	16-bit Q and I component per transmit antenna cascaded.
s_axis_din_tvalid	I	Valid handshake.
s_axis_din_tready	O	Ready handshake.
s_axis_din_tuser[8*TX-1:0]	I	Control information for each transmit antenna. Consists of frame/slot markers and a capture strobe.

Table 2-6: **Top-Level I/O for the DPD IP Core *(Cont'd)***

Names	I/O	Description
s_axis_din_tlast	I	For future expansion, should be tied Low.
Sample Receiver (SRX) Interface (Slave AXI4-Stream)		
s_axis_srx_tdata[16*2*NUM_PHASES-1:0]	I	16-bit Q and I component for common feedback path between all antennas.
s_axis_srx_tvalid	I	Valid handshake
s_axis_srx_tready	O	Ready handshake
s_axis_srx_tuser[7:0]	I	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]	O	16-bit Q and I component per transmit antenna cascaded.
m_axis_dout_tvalid	O	Valid handshake
m_axis_dout_tready	I	Ready handshake
SRX Control Interface (Master AXI4-Stream)		
m_axis_srx_ctrl_tdata[7:0]	O	Control output for SRX input stream to select antenna feedback data and support burst ADC interfaces.
m_axis_srx_ctrl_tvalid	O	Valid handshake
m_axis_srx_ctrl_tready	I	Ready handshake
Host Interface (AXI4-Lite) - s_axi_user		
s_axi_user_aclk	I	Rising edge clock
s_axi_user_aresetn	I	Active-Low synchronous reset
s_axi_user_wdata[31:0]	I	Write data
s_axi_user_wvalid	I	Write data valid handshake
s_axi_user_wready	O	Write data ready handshake
s_axi_user_awaddr[31:0]	I	Write address (byte addressing)
s_axi_user_awvalid	I	Write address valid handshake
s_axi_user_awready	O	Write address ready handshake
s_axi_user_araddr[31:0]	I	Read address (byte addressing)
s_axi_user_arvalid	I	Read address valid handshake
s_axi_user_arready	O	Read address ready handshake
s_axi_user_bresp[1:0]	O	Write response
s_axi_user_bvalid	O	Write response valid handshake
s_axi_user_bready	I	Write response ready handshake
s_axi_user_rdata[31:0]	O	Read data
s_axi_user_rvalid	O	Read data valid handshake

Table 2-6: **Top-Level I/O for the DPD IP Core *(Cont'd)***

Names	I/O	Description
s_axi_user_rready	I	Read data ready handshake
s_axi_user_rresp[1:0]	O	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.		
s_axi_ctrl_aclk	I	Rising edge clock
s_axi_ctrl_2x_aclk	I	Rising edge clock. This must be phase aligned and two times the frequency of s_axi_ctrl_aclk.
s_axi_ctrl_aresetn	I	Active-Low synchronous reset. This is synchronous to s_axi_ctrl_aclk clock domain.
s_axi_ctrl_wdata[31:0]	I	Write data
s_axi_ctrl_wvalid	I	Write data valid handshake
s_axi_ctrl_wready	O	Write data ready handshake
s_axi_ctrl_awaddr[31:0]	I	Write address (byte addressing)
s_axi_ctrl_awvalid	I	Write address valid handshake
s_axi_ctrl_awready	O	Write address ready handshake
s_axi_ctrl_araddr[31:0]	I	Read address (byte addressing)
s_axi_ctrl_arvalid	I	Read address valid handshake
s_axi_ctrl_arready	O	Read address ready handshake
s_axi_ctrl_bresp[1:0]	O	Write response
s_axi_ctrl_bvalid	O	Write response valid handshake
s_axi_ctrl_bready	I	Write response ready handshake
s_axi_ctrl_rdata[31:0]	O	Read data
s_axi_ctrl_rvalid	O	Read data valid handshake
s_axi_ctrl_rready	I	Read data ready handshake
s_axi_ctrl_rresp[1:0]	O	Read data response
s_axi_ctrl_wstrb[3:0]	I	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]	I	Defines access permissions for read accesses.
s_axi_ctrl_awprot[2:0]	I	Defines access permissions for write accesses.
events*		
event_s_din_underrun	O	Underrun status flag for s_axis_din AXI4-Stream interface.
event_m_dout_overrun	O	Overrun status flag for m_axis_dout AXI4-Stream interface.
event_s_srx_underrun	O	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 7, Application Software.

Chapter 3

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

The 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

The 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`. See Hybrid in Chapter 4 for more information.

s_axi_user_aclk

The 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

The `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` disables the output data immediately after one cycle of `dpd_aclk` 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_aclk`, `s_axi_ctrl_2x_aclk`, `s_axi_user_aclk`. Once these clocks are stable, deassert `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 deassertion of dpd_aresetn.

4. **Bring Up Data Rate Clocks**: Once dpd_aclk and dpd_internal_aclk are stable, then dpd_aresetn can be deasserted.
5. Follow the Instructions in DPD Application Software Boot Process section of Chapter 7, Application Software to start the DPD process.

The standard power-down sequence is as follows:

1. Apply dpd_aresetn.

   Stop or change the dpd_aclk and dpd_internal_aclk if necessary.
2. Terminate the DPD Application software.

   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.
3. Apply s_axi_ctrl_aresetn.

   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.

   Stop or change s_axi_ctrl_aclk, s_sxi_2x_aclk, and s_axi_user_aclk if necessary.
4. Turn off the power.

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 of the power up sequence.

Protocol Description

The DPD IP core requires various interfaces to the DFE system for successful operation of the design. The DPD IP 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 2], *Xilinx AXI Design Reference Guide* (UG1037) [Ref 3], and *Arm AMBA AXI and ACE Protocol Specification (IHI 0022D)* [Ref 4].

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 shown in 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 four-phase data, the four complex samples per clock per antenna are concatenated as shown in the following table where sample S0 precedes sample S1 and so on in the serial data stream.

Table 3-3: **Four Phases Per Antenna Concatenation**

Antenna 1								Antenna 0							
S3		S2		S1		S0		S3		S2		S1		S0	
Q	I	Q	I	Q	I	Q	I	Q	I	Q	I	Q	I	Q	I
255:240	239:224	223:228	207:192	191:176	175:160	159:144	143:128	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 chip interface 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 3-4.

Table 3-4: **TUSER Field Packing for Each Byte**

Bits	Name	Description
7:4	user bits	Available for customer use. ***Note:*** Xilinx DFE Demo System uses Bit[4] to carry Power Amplifier Enable Signal in TDD system.
3	capture sync	Used as capture reference point by DPD. Connection is required when using CAPTUREMODE=1 or DCLALGORITHM=4.
2	Uplink/Downlink marker	For TDD systems (0/1) indicating the Uplink/Downlink periods respectively. This bit is not used for FDD systems.
1	frame marker	Primary framing signal. Typically 10 ms time-base in 3G/4G/5G Systems.
0	slot marker	Indicates Data Slot. Also used by Xilinx PC-CFR IP Core.

The slot marker is 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. Typically, the capture sync bit should have a periodic pattern with length matching the signal frame or any special signal periodicity feature. For LTE/NR, tying this bit 3 to bit 1 (frame marker) is one option.

The `m_axis_dout` interface is delay matched with the core's latency with respect to `s_axis_din` interface. After the core comes out of reset, `m_axis_dout_tvalid` follows the behavior of `s_axis_din_tvalid` with a latency equal to core's latency. When `m_axis_dout_tvalid` is high, the core gives out N number of samples for every clock cycle, where N is one for single-phase designs and two for two-phase designs. The data is read by asserting `m_axis_dout_tready` every cycle while the data is updated 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.

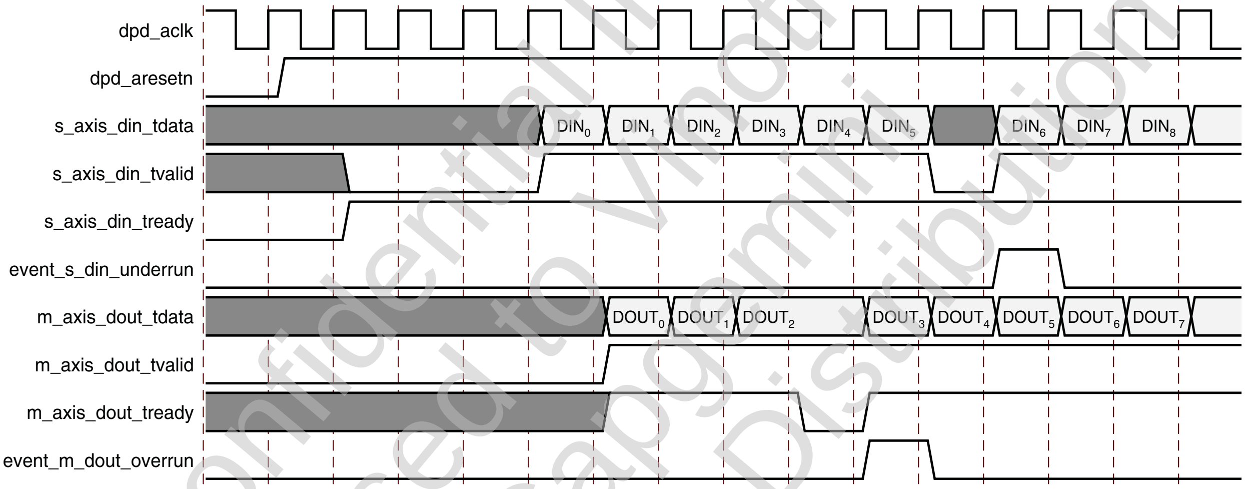


Figure 3-1: **AXI4-Stream Datapath Interface**

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 connections are defined in the following tables.

Table 3-5: **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…

Table 3-5: **Sample Receiver Interface for Single Phase Designs *(Cont'd)***

RXINPUTFORMAT	TXRXRATIO	srx_din(31:16)	srx_din(15:0)
Real	1	S1,S3,S5...	S0,S2,S4...
Real	2	S0,S1,S2...	S0,S1,S2...

Table 3-6: **Sample Receiver Interface for Two Phase 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

Table 3-7: **Sample Receiver Interface for Four Phase Designs**

RXINPUTFORMAT	TXRXRATIO	srx_din (127:112)	srx_din (111:96)	srx_din (95:80)	srx_din (79:64)	srx_din (63:48)	srx_din (47:32)	srx_din (31:16)	srx_din (15:0)
Complex	1	Q3,Q7, Q11...	I3,I7, I11...	Q2,Q6, Q10...	I2,I6, I10...	Q1,Q5,Q9...	I1,I5,I9...	Q0,Q4, Q8...	I0,I4,I8...
Complex	2	Q1,Q3,Q5...	I1,I3,I5...	Q1,Q3, Q5...	I1,I3,I5...	Q0,Q2,Q4...	I0,I2,I4...	Q0,Q2, Q4...	I0,I2,I4...
Real	1	S7,S15, S23...	S16,S14, S22...	S5,S13, S21...	S4,S12, S20...	S3,S11, S19...	S2,S10, S18...	S1,S9, S17...	S0,S8,S16...
Real	2	S3,S7,S11...	S3,S7, S11	S2,S6, S10...	S2,S6, S10...	S1,S5,S9...	S1,S5, S9...	S0,S4, S8...	S0,S4, S8...

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 3-8: **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 3-9.

Table 3-9: **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

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.

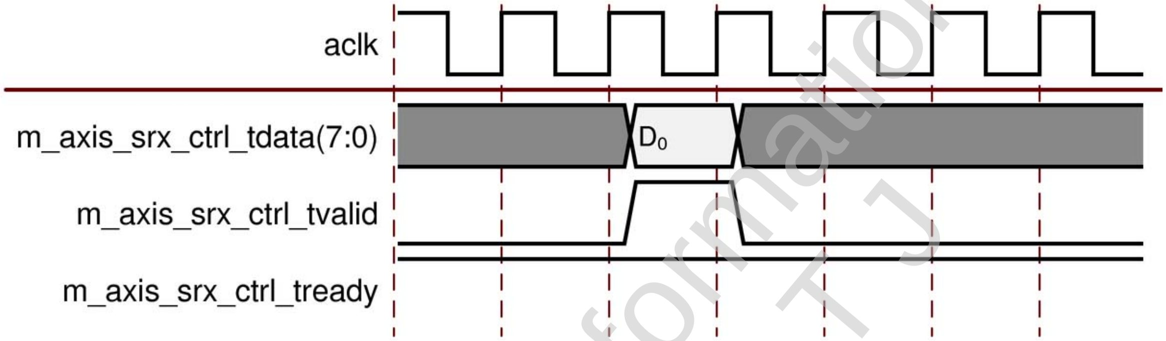


Figure 3-2: **m_axis_srx_ctrl Stream Behavior with tready Tied High**

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.

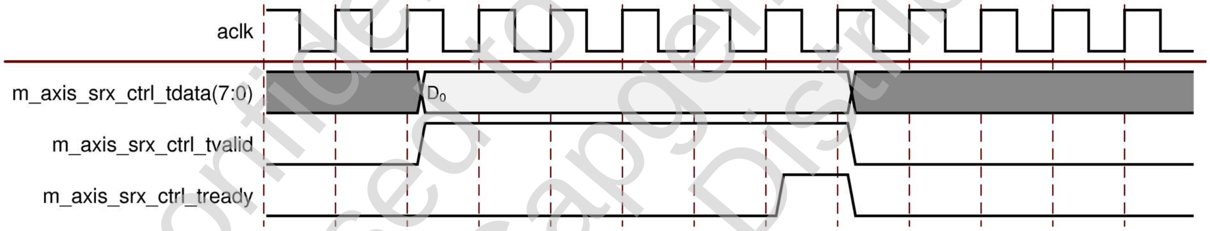


Figure 3-3: **m_axis_srx_ctrl Stream Behavior When an External System Holds Off tready**

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 7 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 not 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.

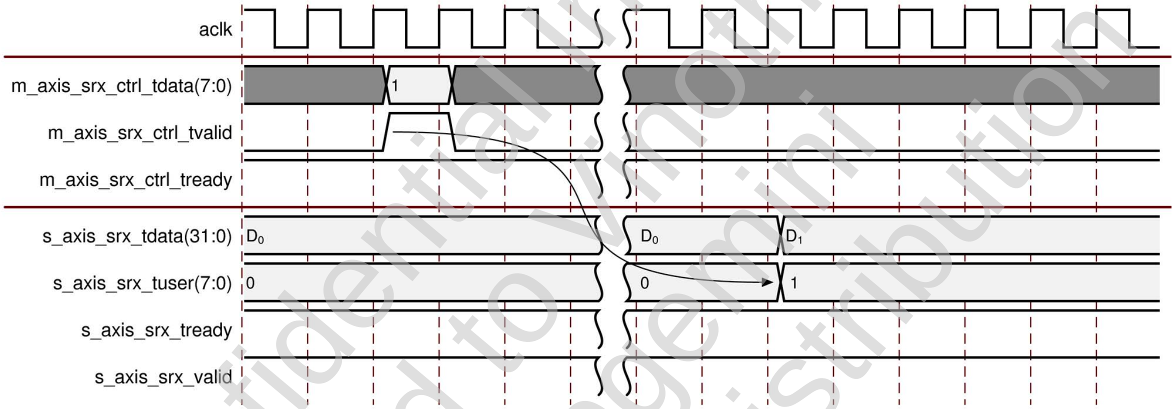


Figure 3-4: **Using s_axis_srx_tuser to Indicate Active Antenna**

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 3-9. 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.

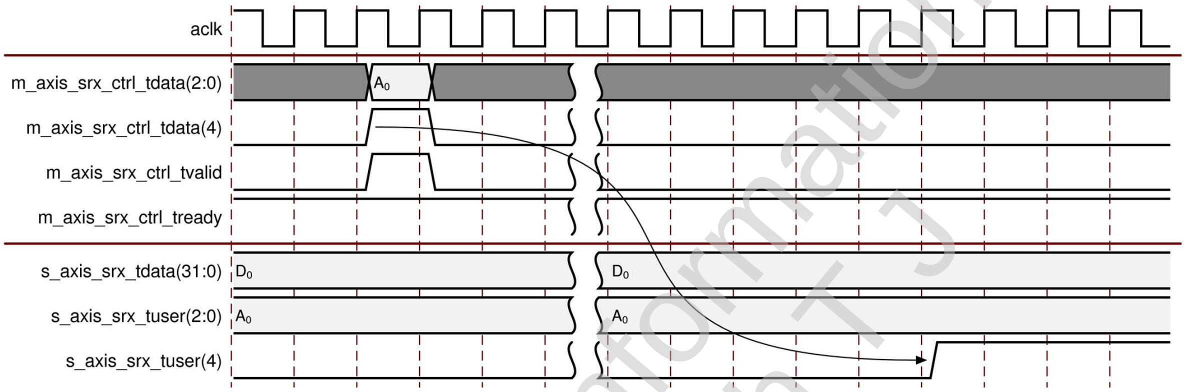


Figure 3-5: **Using s_axis_srx_tuser to Indicate a High Resolution Data Period**

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 preferred 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 set to 1 and PORTCONTROL set to 0 is shown in Figure 3-6.

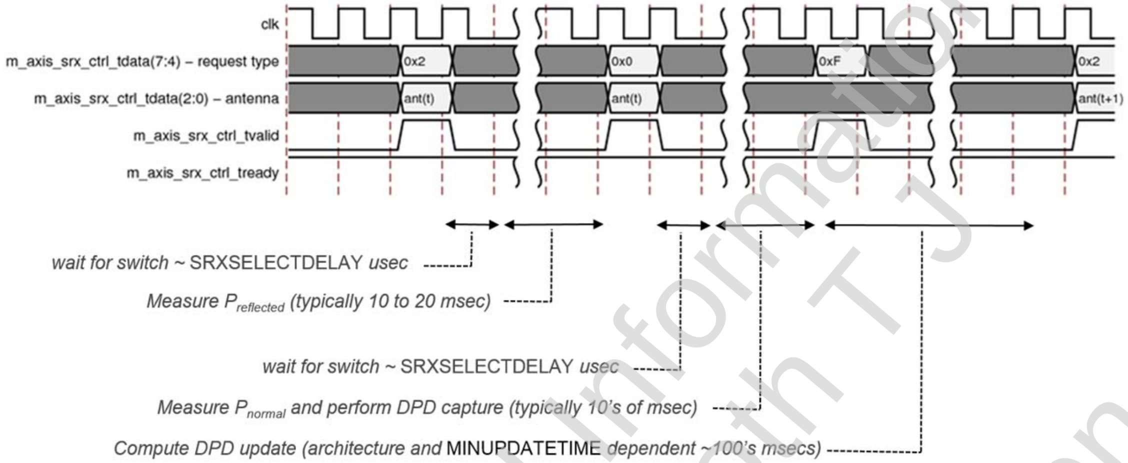

clk
m_axis_srx_ctrl_tdata(7:4) – request type
m_axis_srx_ctrl_tdata(2:0) – antenna
m_axis_srx_ctrl_tvalid
m_axis_srx_ctrl_tready
0x2
0x0
0xF
0x2
ant(t)
ant(t)
ant(t+1)
wait for switch ~ SRXSELECTDELAY usec
Measure $P_{reflected}$ (typically 10 to 20 msec)
wait for switch ~ SRXSELECTDELAY usec
Measure P_{normal} and perform DPD capture (typically 10's of msec)
Compute DPD update (architecture and MINUPDATETIME dependent ~100's msecs)

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 in Table 3-10. 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 3-10: **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. ONA must not be connected to UL/DL marker in TDD systems, as this results in constant 0 reported in SRX powers and DCL will not be able to operate.
6:5	Reserved	Set to zero
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:

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 2] 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 *Digital Pre-Distortion v13.1 Debug Interface User Guide* (UG989) [Ref 1]). 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 MPSoC or Zynq RFSoC Device. Further restrictions may apply depending on the choice of DPD Software. For details on the memory map, see the Chapter 7, Application Software.

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 might be reported as Data Aborts. However, they will not cause failure of the core.

Chapter 4

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 MPSoC Processing Subsystem. A customer reference design using Vivado IP integrator is described in Chapter 5, Reference Design, and design files are included along with DPD IP repository ZIP file and is the recommended starting point for designs.*

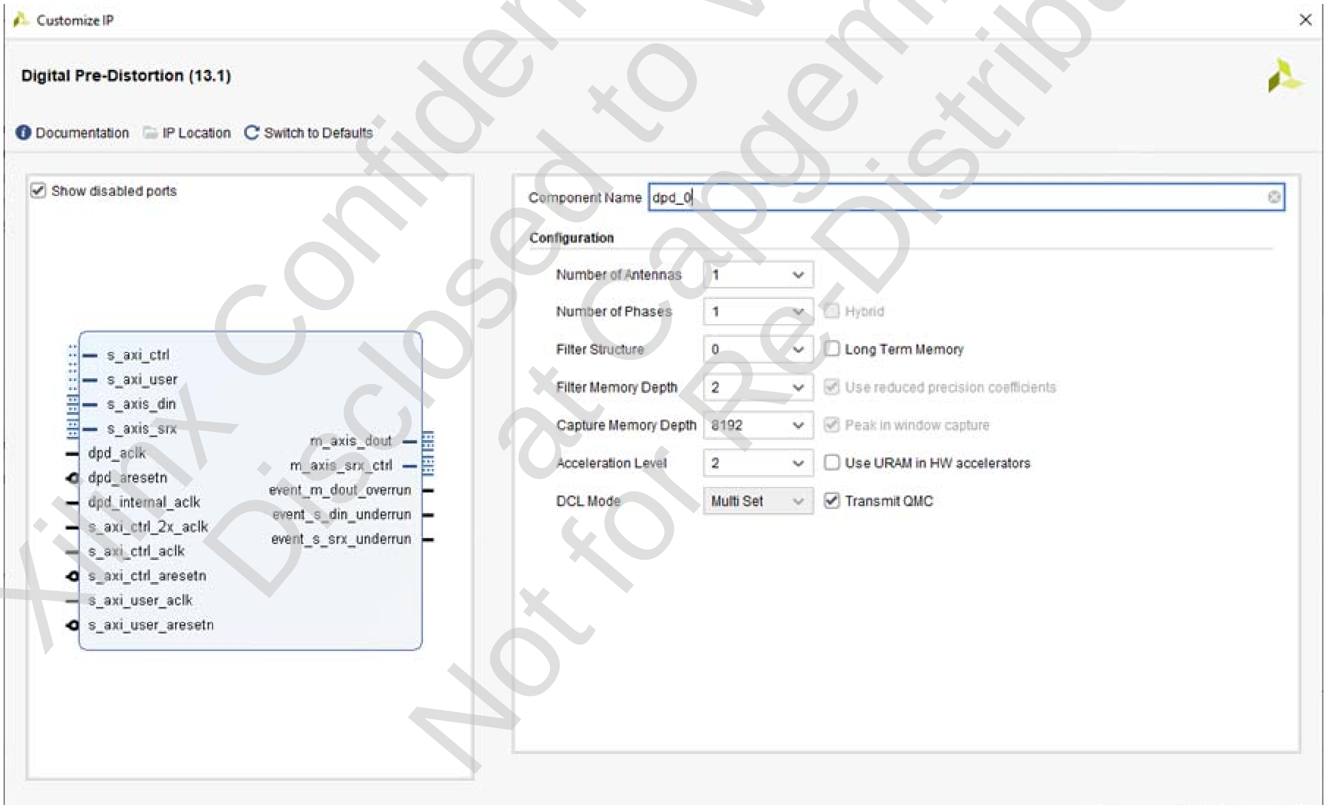
You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

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 v13.1 archive. Click **Settings** in the Vivado IDE and select the **IP** icon. Use the **Add Repository...** button to add the `ip_repos/repo/` 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 v13.1. 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.

Note: As of the DPDv13.1 IP release, some parameters are fixed to default values and are not user-configurable. These parameters include Reduced Precision, Peak in Window Capture Mode, and DCL Mode. These parameters are grayed out and disabled with their default values displayed on the IP GUI.



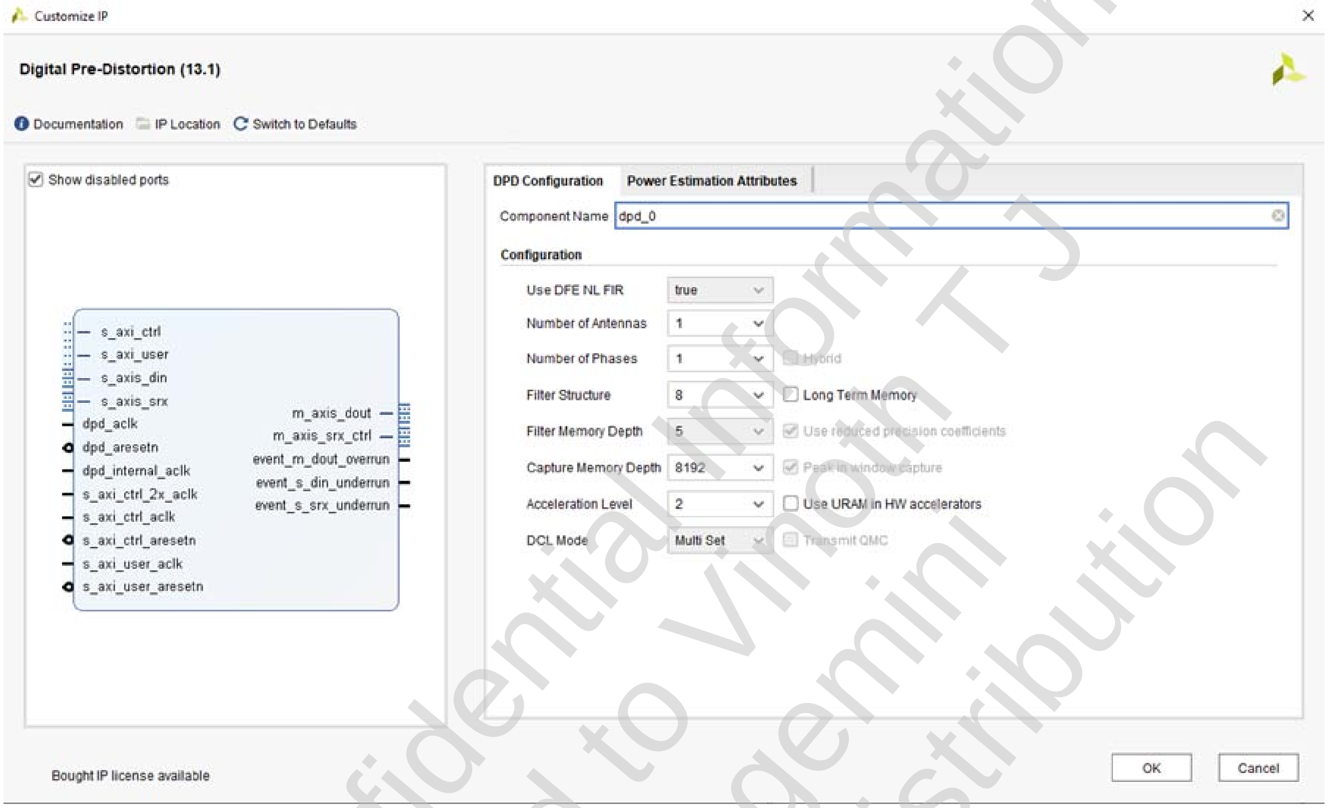
☑ Show disabled ports

Component Name: dpd_0

Configuration

Number of Antennas: 1
Number of Phases: 1 ☐ Hybrid
Filter Structure: 0 ☐ Long Term Memory
Filter Memory Depth: 2 ☑ Use reduced precision coefficients
Capture Memory Depth: 8192 ☑ Peak in window capture
Acceleration Level: 2 ☐ Use URAM in HW accelerators
DCL Mode: Multi Set ☑ Transmit QMC

Figure 4-1: **Customize IP (When Targeting MPSoC/RFSoC Gen 1/2/3)**



Digital Pre-Distortion (13.1)

☑ Show disabled ports

Component Name: dpd_0

Configuration

Use DFE NL FIR: true

Number of Antennas: 1

Number of Phases: 1 □ Hybrid

Filter Structure: 8 □ Long Term Memory

Filter Memory Depth: 5 ☑ Use reduced precision coefficients

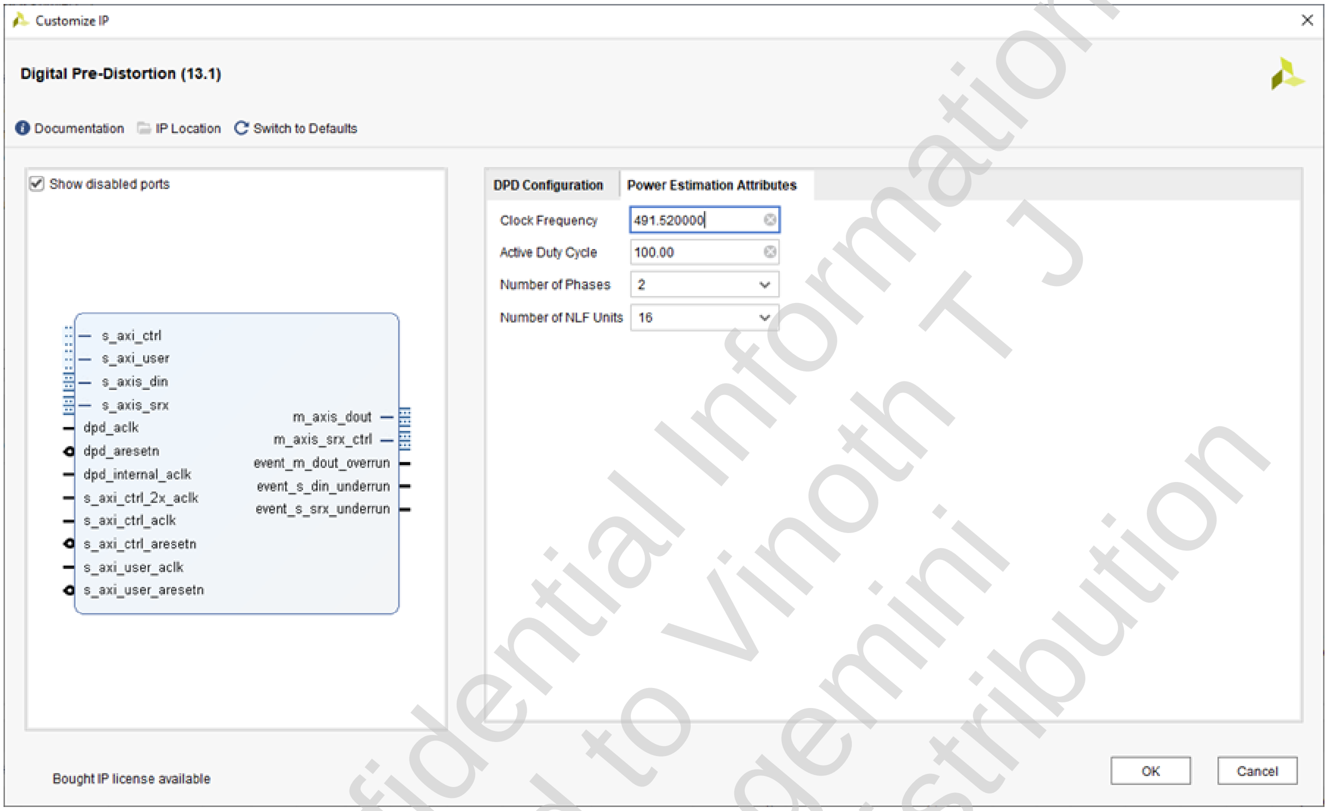
Capture Memory Depth: 8192 ☑ Peak in window capture

Acceleration Level: 2 □ Use URAM in HW accelerators

DCL Mode: Multi Set □ Transmit QMC

Bought IP license available

Figure 4-2: **Customize IP (RFSoC DFE Devices)**


Customize IP

Digital Pre-Distortion (13.1)

Documentation | IP Location | Switch to Defaults

☑ Show disabled ports

DPD Configuration | Power Estimation Attributes

Clock Frequency: 491.520000

Active Duty Cycle: 100.00

Number of Phases: 2

Number of NLF Units: 16

Bought IP license available

Figure 4-3: **Power Estimation Attributes Only (RFSoC DFE Devices)**

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 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 (on all supported devices) or four (RFSoC DFE devices only) complex samples per clock cycle. 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. As an example, to operate DPD at 491.52 MSps, with this mode, DPD interface will operate at 245.76 MHz clock (`dpd_aclk`), while internal to DPD, some of the functionality will operate at 491.52 MHz (`dpd_internal_aclk`). This allows the user to build their design and its interface to DPD at a lower rate and leverage DPD's optimized IP that can run at that internal clock rate.

Note: This option is NOT available on RFSoC DFE devices.

Filter Structure

On MPSoC and RFSoC devices, this parameter allows selection of one of four filter structures. Filter structure 0 creates a filter with the original structure used until DPD v9.0. Filter structure 1 creates a filter that can be beneficial with wideband signals.

On RFSoC DFE devices, the parameter allows selection of one of four filter structures. Filter Structure 8 and Filter Structure 9 are similar to Filter Structure 0 and Filter Structure 1, respectively. Filter Structures 10 and 11 may improve performance for Macro PA applications.

Note: When using a Filter Structure set to 10 or 11, the DPD Number of phases supported is 2.

Table 4-1: **Filter Structures**

	Filter Structure		Description
Programmable Logic Datapath	0	00000	Legacy datapath available since DPD v7.0
	1	00001	Improved performance for wideband
Reserved	7:2	RSVD	Reserved for future expansion
DFE NL FIR Datapath	8	01000	Similar to Structure 0 of PL ONLY mode
	9	01001	Improved performance for wideband (similar to structure 1)
	10	01010	Deeper memory options utilizing two DFE NL FIR primitives per Antenna path (beneficial for Macro PA)

Table 4-1: **Filter Structures *(Cont'd)***

	Filter Structure		Description
	11	01011	Similar performance to Filter Structure 10 using only one DFE NL FIR primitive
Reserved	15:12	RSVD	Reserved for future expansion

Long Term Memory

Electron trapping and thermal effects seen with Gallium Nitride (GaN) transistors based amplifiers, exhibits various long term memory effects that DPD has to compensate, on top of traditional memory effects seen. When enabled, this parameter adds additional datapath functionality to handle these long term memory effects. To use this feature, some of the other parameters are also forced such as- capture memory depth (set to 16384) and Transmit QMC (false). When this parameter is enabled, additional run time parameters are available to manage, as described in the UPDATE_ARCH_PARAMETERS (Table 7-21). When this parameter is enabled, there is a additional PL resources used by DPD, which are available to review in the resource spreadsheet available on the DPD Lounge.

Note: When using long term memory, the `capture_sync` and `frame_marker` bits in the `s_axis_din_tuser` (see Table 3-4) must be connected to the systems `frame_marker`.

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.

Note: When using DFE NL FIR primitive, Filter Memory Depth is fixed.

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. When Long Term Memory is selected, this is fixed to 16384.

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 the Acceleration Level values higher than the default selected in the GUI.

Use URAM in HW Accelerators

This parameter allows the use of UltraRAM instead of block RAM in hardware accelerators when necessary. By default, block RAMs are used. Select this option to use UltraRAMs to reduce the overall number of block RAMs in the HW accelerator. Except in a few instances, most UltraScale+ devices support UltraRAM. Xilinx recommends checking the corresponding data sheet before enabling UltraRAM.

Note: USE URAM in HW accelerators is fixed to TRUE when targeting RFSoC DFE devices.

Transmit QMC

This parameter controls the selection of QMC in the transmit path. The transmit QMC is optional for all devices where a few design resources can be saved when removed from the transmit chain. This parameter is available by default for non-RFSoC devices. However, it can be removed by setting the parameter to FALSE.

User Parameters

Table 4-2 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-2: **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 Structure	FILTER_STRUCTURE	PL-only IP: 0 PL+DFE NL FIR: 8
Filter Memory Depth	MEMORY_DEPTH	PL-only IP: 2 PL+DFE NL FIR: 5 (Fixed)
Hybrid	HYBRID	PL-only IP: Hybrid PL+DFE NL FIR: N/A
Capture Memory Depth	CAPTURE_DEPTH	2
Acceleration Level	HWA_LEVEL	2
Use URAM in HW accelerators	USE_URAM_HWA	PL-only IP: FALSE PL+DFE NL FIR: TRUE (Fixed)
Transmit QMC	HAS_TX_QMC	PL-only IP: FALSE (Optional) PL+DFE NL FIR: FALSE (Fixed)

Table 4-2: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value[1]	User Parameter/Value[2]	Default Value
Long Term Memory	LONG_TERM_MEMORY	0

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 the Generating IP Output Products section 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 IP Core XDC file. The Vivado flow takes care of propagating all the IP Core's synthesis constraints. It is assumed that only clock specifications come externally from the user and all other constraints are within the IP package.

- In the design, there are two synchronous clock pairs `dpd_aclk/dpd_internal_aclk` and `s_axi_ctrl_aclk/s_axi_ctrl_2x_aclk` all paths between the two domains that form a pair. These are timed due to their known phase and frequency relationship which comes from common clock source, typically a MMCM within the clock wizard IP.
- 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 have an appropriate timing exception applied 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 UltraScale+ and Zynq UltraScale+ RFSoC devices. Both the hardware instance (using Vivado) and software binary (dpd-smp) should operate on same device.

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].

Reference Design

This chapter describes the IP Reference Design for DPD Zynq UltraScale+™, and Zynq RFSoC UltraScale+ devices ONLY. The design is provided for Xilinx® ZCU111/ZCU102/ZCU208/ZCU216 evaluation boards and includes the following flows.

Note: To understand the reference design details for RFSoC DFE and ZCU670 evaluation board, see the *Zynq UltraScale+ RFSoC DFE Targeted Reference Design* (UG1530) [Ref 22].

Hardware Flow

- Vivado® IP integrated design includes subsystems for clocking and reset, Digital Front End (DFE), RF Interface, Zynq Processor, and Vector Source. The design can be synthesized and implemented to generate a bitstream for each board.
- Source code includes HDL for the IP using the module reference flow in the Vivado IP integrator and design constraints.
- IP repository for DPD IP core.

Software Flow

- Xilinx PetaLinux scripts to generate embedded Linux boot image.
- XSDB scripts to boot image over JTAG.
- Source code for the example host application that communicates via DPD host API to DPD application software.
- Xilinx XSCT scripts to create the Vitis™ software platform project and to build the host application.

Table 5-1 shows the top-level directory structure of archive and where the particular files are located:

Table 5-1: **Folder Structure**

Directory	Description
bin/	DPD application binaries
psu_cortexa53	
common/	Common sources for all reference designs
embedded_sw/	Embedded software sources
ip_repos/	IP repositories
doc/	Documentation
html/	Doxygen HTML documentation for embedded software
ip_ref_design/	IP reference design
zcu102_*	ZCU102 reference designs
zcu111_*	ZCU111 reference designs
zcu208_*	ZCU208 reference designs
zcu216_*	ZCU216 reference designs

Subsequent sections provide an overview of systems and subsystems followed by instructions for building each reference design.

System Overview – RFSoC Designs

Figure 5-1 shows the top-level block diagram of the IP Reference Design for RFSoC Designs. Some of the IP Cores and interconnects have been omitted from the diagram for clarity, but are described in the subsequent sections.

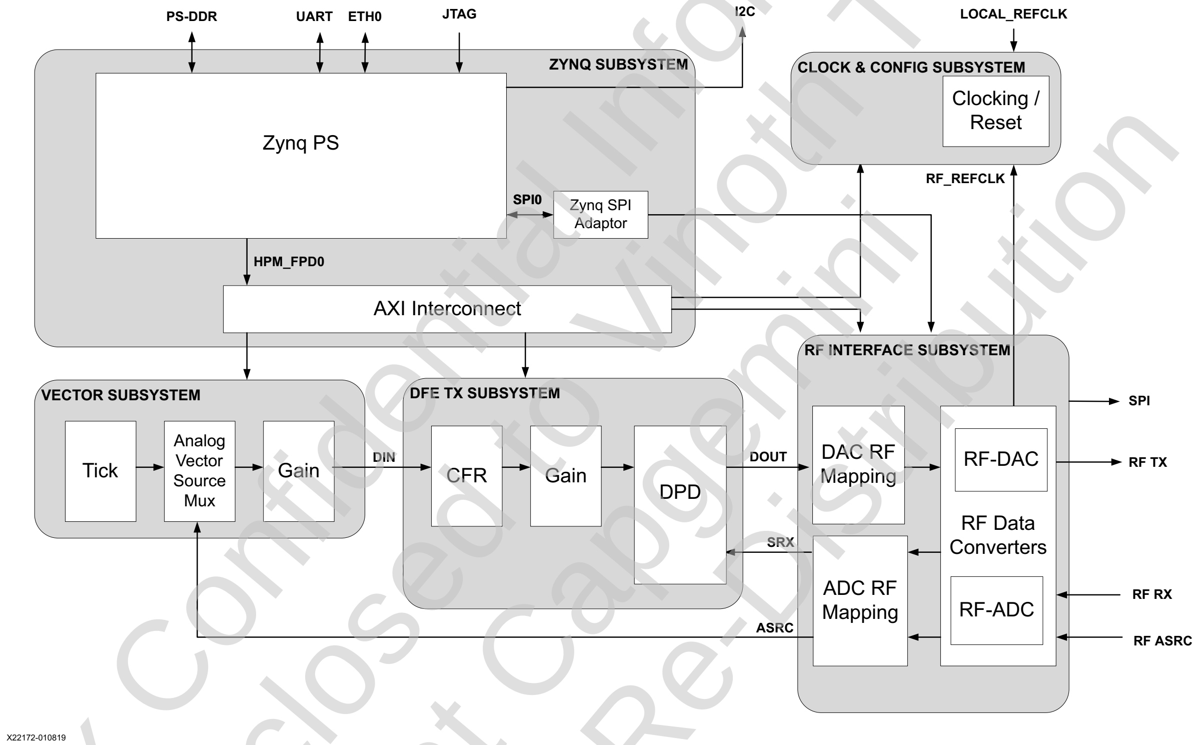


Figure 5-1: **IP Reference Design for RFSoC Designs**

The reference design comprises of five subsystems:

- **DFE Transmit Subsystem**: The Digital Front-End (DFE) Transmit Subsystem contains the CFR and DPD IP Cores. For more information, see DFE Transmit Subsystem.
- **Zynq Subsystem**: Provides AXI bus connections to other subsystems in the design. For more information, see Zynq Subsystem.
- **RF Interface Subsystem**: Provides access to the RF-DAC and RF-ADC blocks using the RF Data Converter IP Core. Support is provided for using one ADC input as an Analog Vector Source (ASRC). The subsystem contains IP Cores to map the digital data to the format used by the RF Data Converters and SRX Path Selection. For more information, see RF Interface Subsystem (RFSoC).
- **Vector Subsystem**: The reference design uses one of the RF-ADC inputs to act as an Analog Vector Source (ASRC input) from third-party test equipment. Basic frame timing

is provided from the Tick IP Core. For more information, see Vector Subsystem (RFSoC Designs).

- **Clocking/Reset/Configuration Subsystem**: These provide clocking, reset, and configuration support to the design. They also include support for generating user control and DFE clocking. For more information, see Clocking, Reset, and Configuration Subsystem.

System Overview – JESD204C Designs

Figure 5-2 shows the top-level block of the IP Reference Design for JESD204b designs which comprises the same subsystems as described previously. In this design, there is no support for Analog Vector Source, instead a simple tone data is generated for the design to show basic operation.

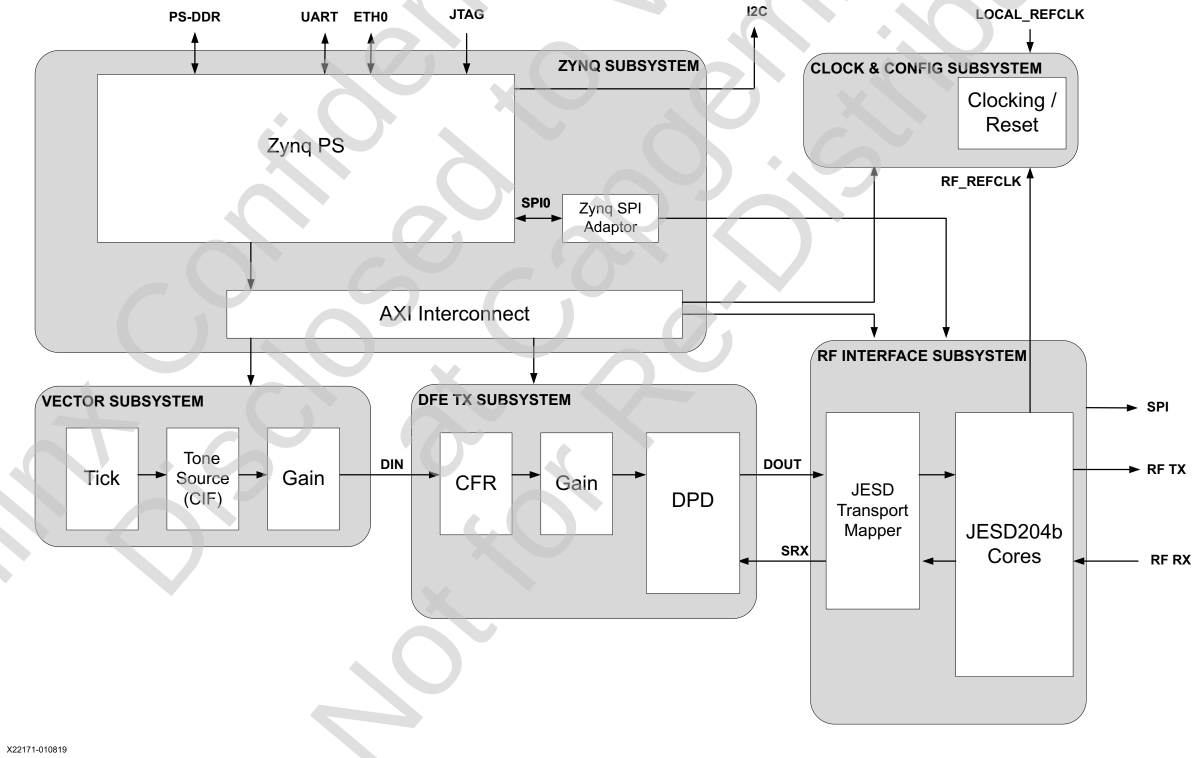


Figure 5-2: **IP Reference Design for JESD204b Designs**

ZCU111 Board Level Connectivity

An overview of the board level connectivity for the reference design is shown in Figure 5-3. The ZCU111 provides interfaces to a host PC via JTAG / UART and Ethernet interfaces. The RF-DAC and RF-ADC interfaces are connected out via the LPAF Connector on the ZCU111. Clocking to the RF-DAC and RF-ADC are provided by the on-board RF Clocking using LMK04208 and LMX2594 parts. The LPAF is connected to a Xilinx® HW-FMC-XM500 which provides a variety of single-ended and differential input/output options for the RF signals.

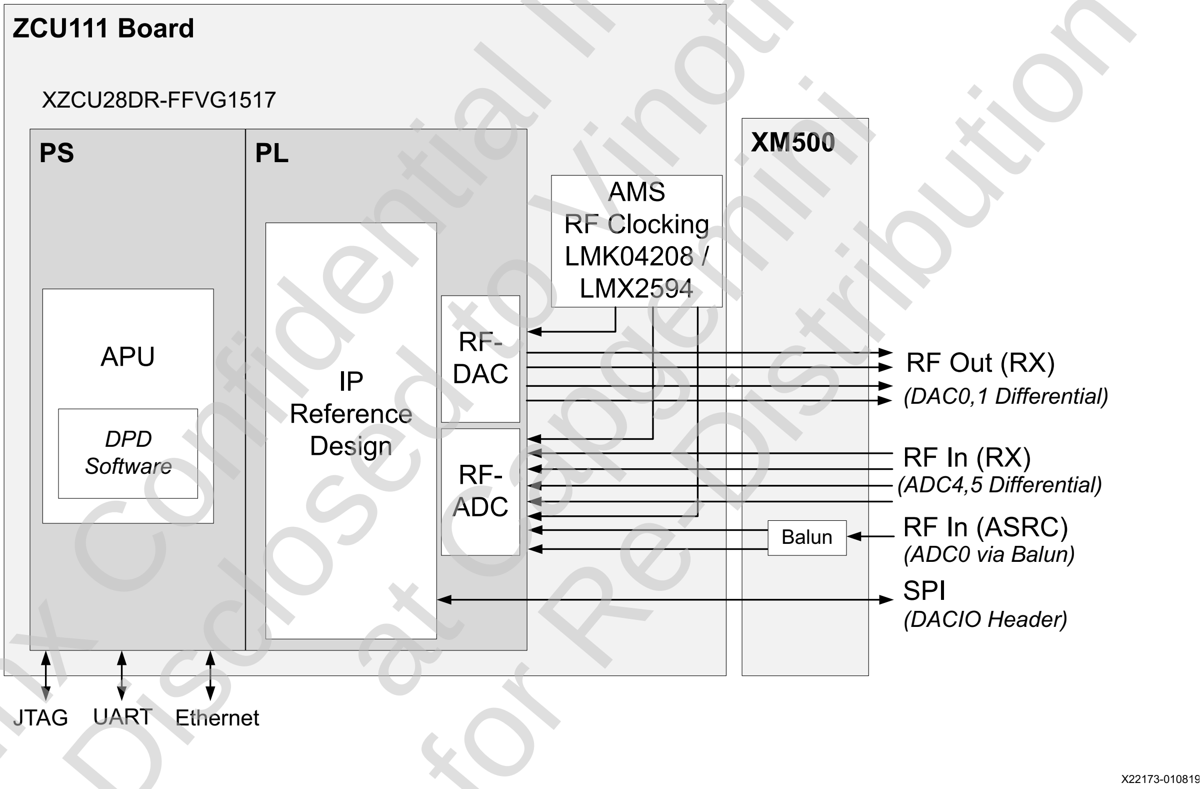


Figure 5-3: **Board Level Connectivity for ZCU111**

An SPI output is also provided on the DACIO header of the HW-FMC-XM500 to allow control of external RF Boards.

ZCU102 Board Level Connectivity

Figure 5-4 shows the board-level connectivity for IP Reference Designs targeted to ZCU102 boards using JESD204b to communicate with an external RF Board. In this case, a Xilinx XRF2 board is used. This board includes RF Clocking, JESD204b DAC and ADC and a RF Front-End comprising IQ Modulation/Demodulation, Filtering, and Attenuation.

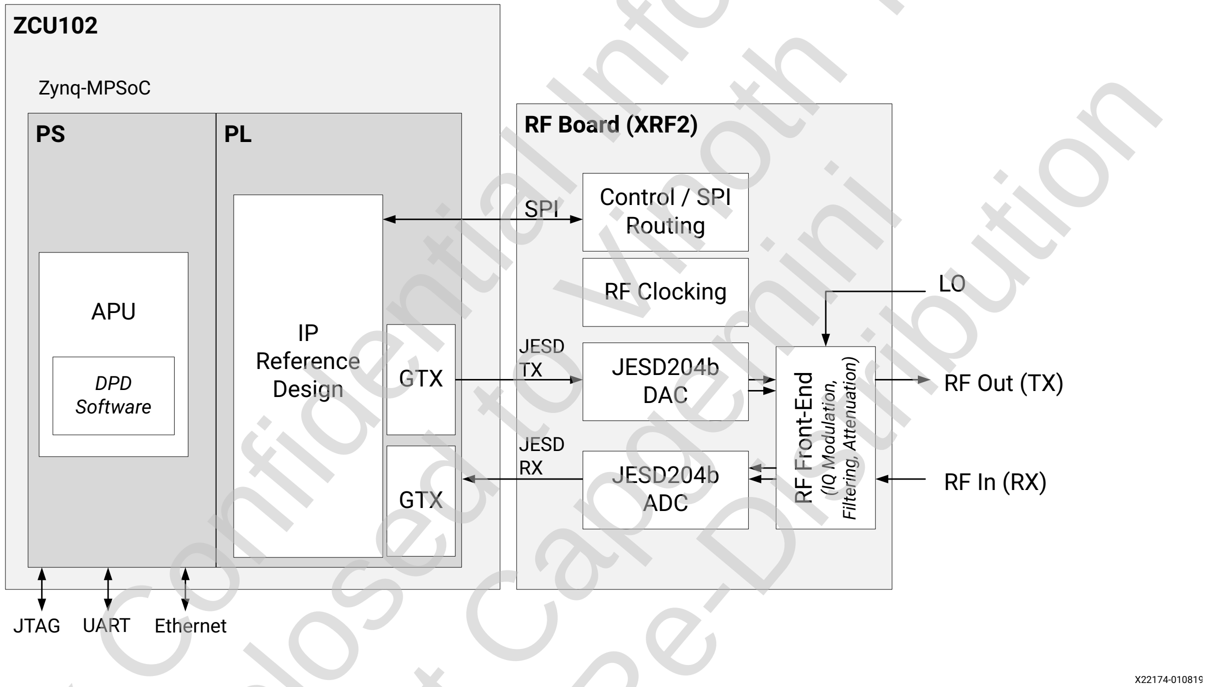


Figure 5-4: **IP Reference Designs for ZCU102**

ZCU208/ZCU216 Board Level Connectivity

An overview of the board level connectivity for the reference design is shown in Figure 5-5. The ZCU208 or ZCU216 provide interfaces to a host PC via JTAG/UART and Ethernet interfaces. The RF-DAC and RF-ADC interfaces are connected out via the LPAF Connector on the ZCU208 or ZCU216. Clocking to the RF-DAC or RF-ADC are provided by the external CLK104 module. CLK104 implements LMK04828 and two LMX2594 parts for reference or full RF clock generation for the RF-DAC as well as RF-ADC. The LPAF is connected to a Xilinx HW-FMC-XM655 which provides a variety of single-ended and differential input/output for the RF signals. An SPI output is also provided on the DACIO header of the HW-FMC-XM655 to control external RF Boards. Figure 5-5 is one example in which external cables are used to connect DAC output with ADC input via Baluns.

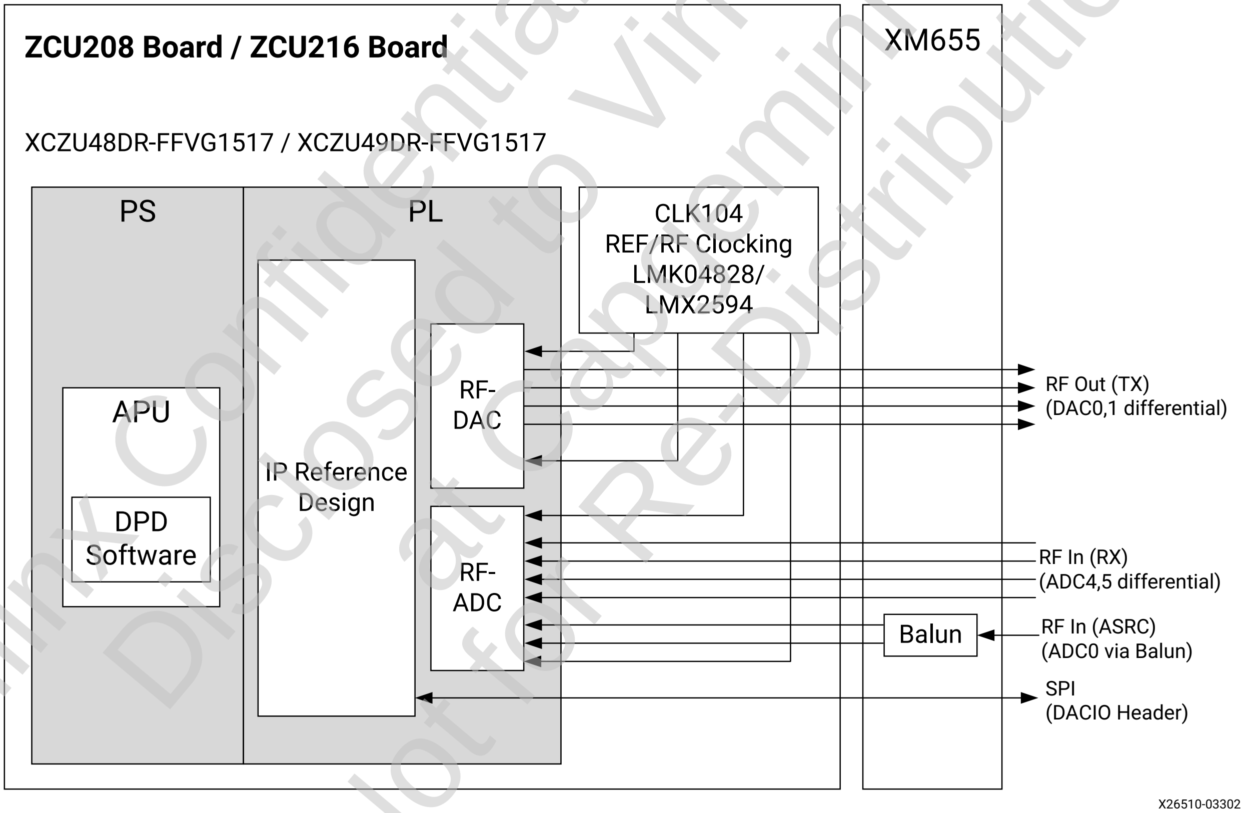

ZCU208 Board / ZCU216 Board
XCZU48DR-FFVG1517 / XCZU49DR-FFVG1517
PS
PL
APU
DPD Software
IP Reference Design
RF-DAC
RF-ADC
CLK104 REF/RF Clocking LMK04828/ LMX2594
XM655
Balun
RF Out (TX) (DAC0,1 differential)
RF In (RX) (ADC4,5 differential)
RF In (ASRC) (ADC0 via Balun)
SPI (DACIO Header)
X26510-03302

Figure 5-5: **IP Reference Designs for ZCU208/ZC216**

For ZCU670 board connection information, refer to the 'Board Connectivity' section of the *Zynq UltraScale+ RFSoC DFE Targeted Reference Design* (UG1530), registration required.

IP Reference Design Subsystems

The following sections describe the individual subsystems that are used to construct the IP Reference Design.

DFE Transmit Subsystem

The DFE Transmit Subsystem is shown in Figure 5-6. The DFE Transmit Subsystem contains the main signal processing functions for the IP Reference Design.

The start of the DFE Processing Chain is a Xilinx Crest Factor Reduction core (see PG097 - registration required) that reduces the overall Peak-to-Average Power of the signal. A Gain block is placed after the CFR to allow adjustment of signal power at input to the Xilinx DPD IP Core. The final stage in the processing is the Xilinx DPD Core.

The address space required for both the DPD Control Interface and the DPD User Interface is described in Protocol Description. This address space can be located on any suitable PL-PS address space.

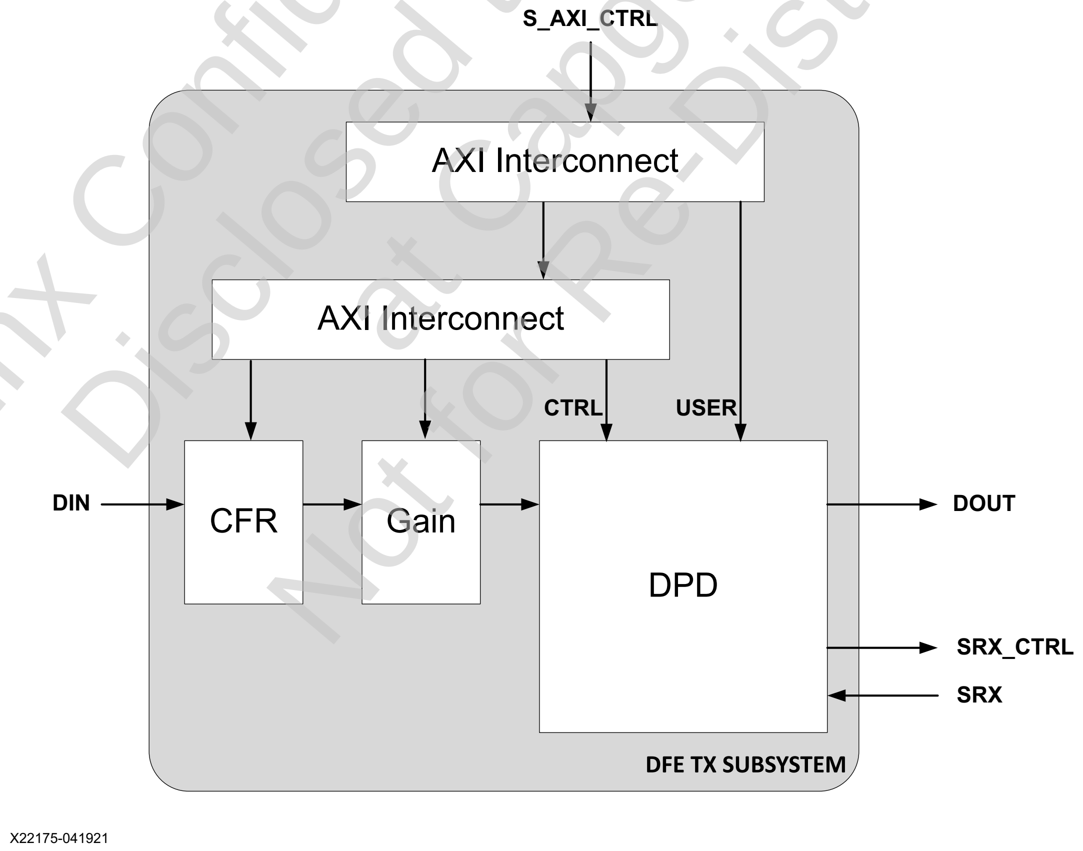


Figure 5-6: **DFE Transmit Subsystem**

The AXI Interconnect connect the Data Processing Cores to the Zynq PS. A separate interconnect is used to split the bus into Control Accesses and User Host Interface Accesses to the DPD Core. The User Host Interface runs on a different clock (`s_axi_user_aclk`) from the clock used for other control accesses (`s_axi_ctrl_aclk`) to allow connection to a separate host system. In the reference design, a top-level AXI interconnect is provided to supply both `s_axi_user` and `s_axi_ctrl` from the Zynq PS.

Zynq Subsystem

The block diagram of a Zynq MPSoC Subsystem is shown in Figure 5-7. The Zynq UltraScale+ MPSoC that connects it to the peripherals on the board and other subsystem in the IP Reference Design. The Zynq Subsystem provides a wrapper around the peripheral connections that are made to UART, Ethernet, PS-DDR, and I2C (both I2C0 and I2C1) on the ZCU111 or ZCU208 or ZCU216 boards. Connection to a 4-wire SPI Interface is made for control of RF Daughter boards.

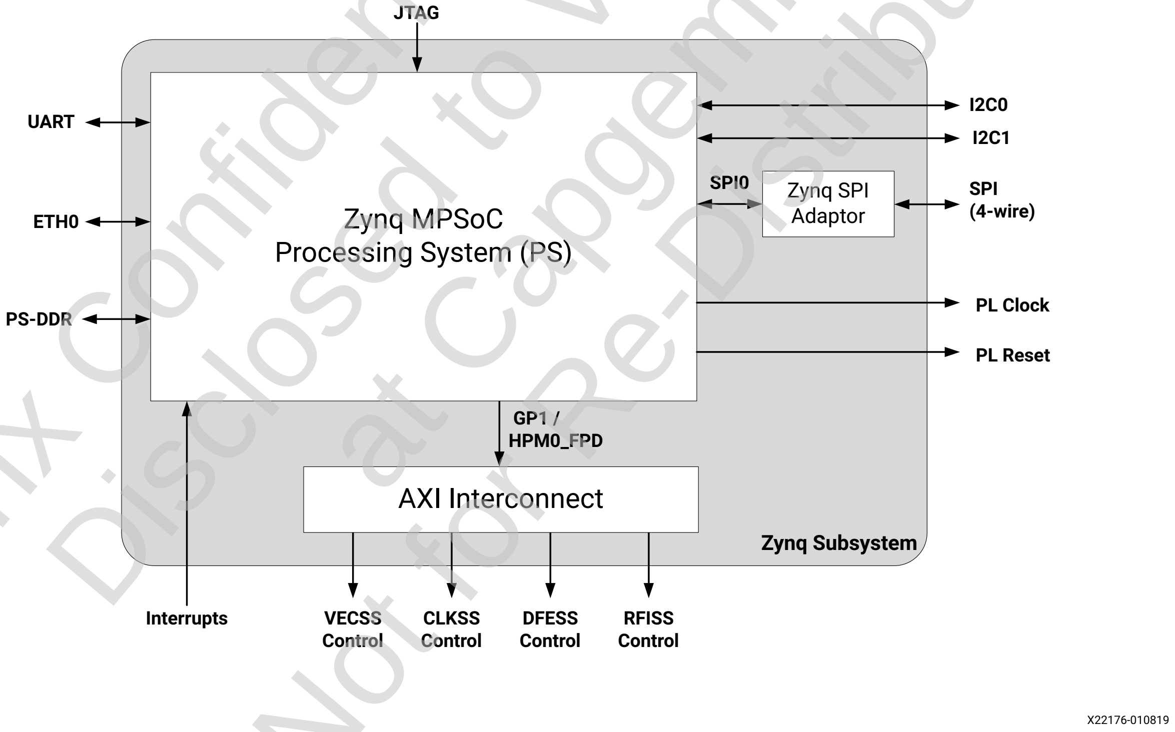


Figure 5-7: **Zynq Subsystem – Block Diagram**

The Zynq Subsystem provides a wrapper around Clocks and Resets are generated and fed to the clocking, reset, and configuration subsystem. The provided AXI Interconnect routes control accesses to the different subsystems in the design from the HPM0_FPD bus on the PS. A Protocol Converter (not shown) converts accesses from AXI4 Memory Map to the AXI4-Lite Standard to minimize the size of the control bus. Interrupts from the DPD Core and RF Data Converter are concatenated and fed into the PS.

RF Interface Subsystem (RFSoC)

The block diagram of the RF Interface Subsystem is shown in Figure 5-8. It provides interfaces from the DAC Transmit Path to the PA and to a number of RF Receive Paths that Feedback the PA input of DPD Core (`s_axis_srx`). A path is also provided for an Analog Vector Source (ASRC) from Third-Party Test Equipment to be fed into the design.

The RF Interface Subsystem contains a number of blocks that control the format and movement of data to and from the RF Data Converters. The SRX Multiplexer is used to select the SRX feedback path from a number of RF-ADC input streams. The selection is controlled by the SRX_CTRL input from the DPD Core.

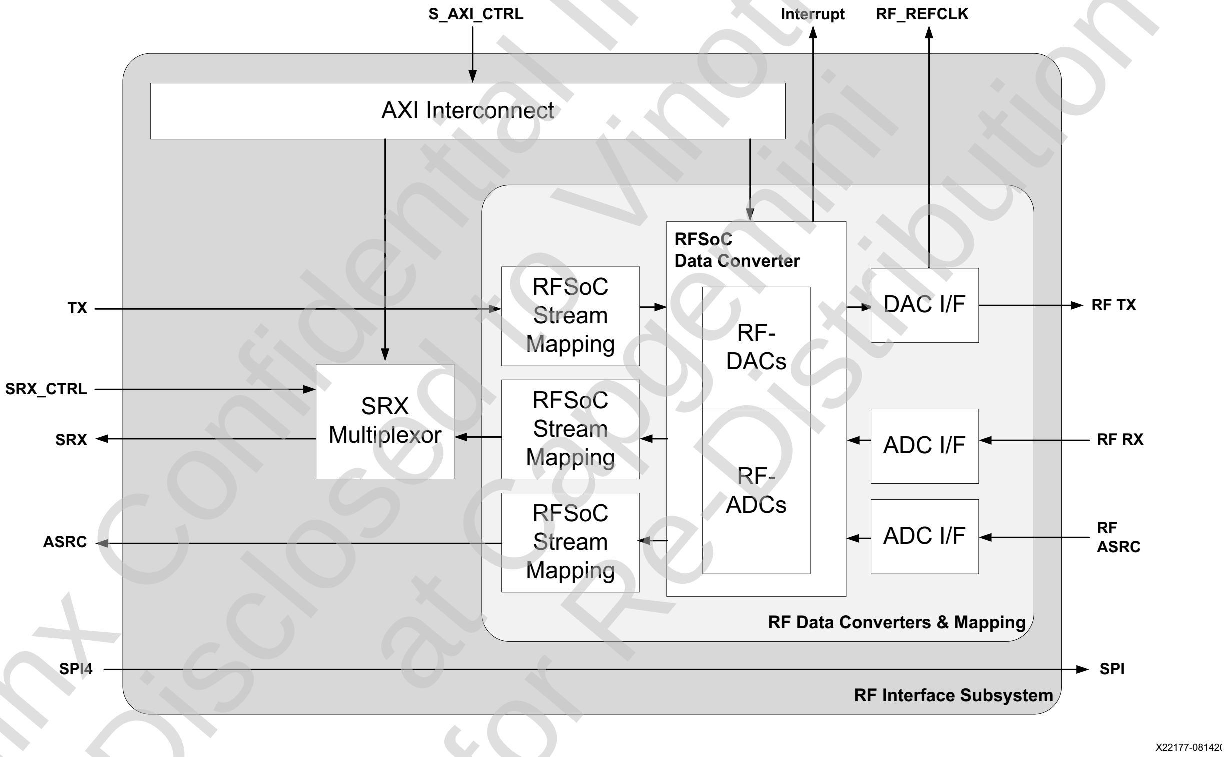
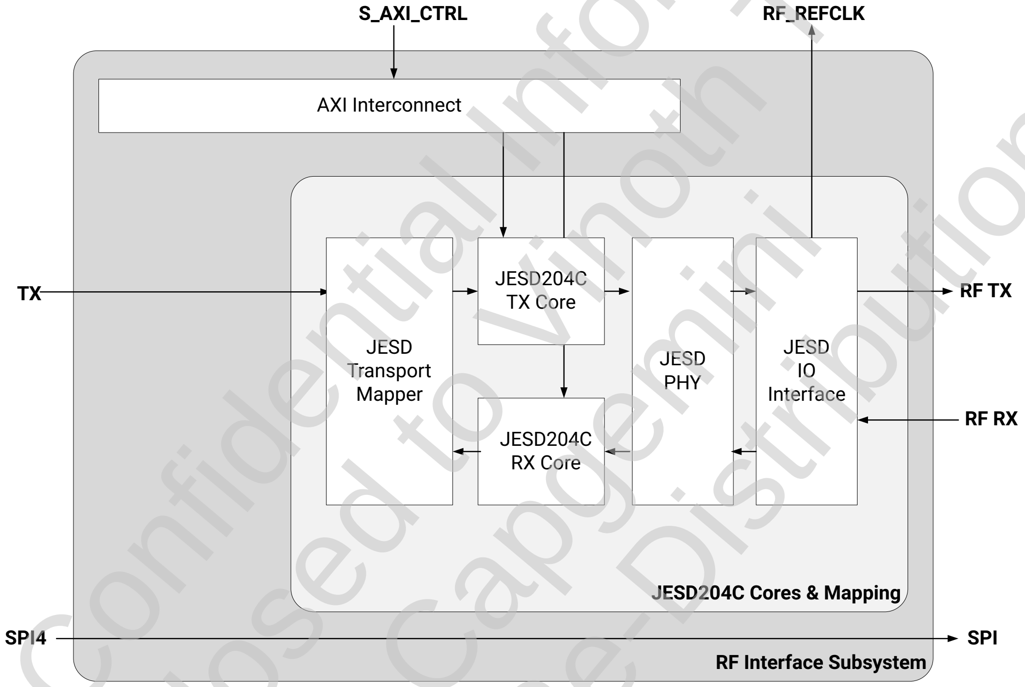


Figure 5-8: **RF Interface Subsystem – Block Diagram**

The RF Data Converter has number of stream mapping blocks on RF-DAC inputs and RF-ADC outputs. The mapping blocks used in the Vivado IP will vary depending on the RF configuration, but include Xilinx AXI4-Stream Combiners, AXI Subset Converter IP from the Vivado IP Catalog which reformats data to correct format for input into the RF Data Converters. The DAC and ADC IO Interface blocks are used to simplify display in the Vivado IP integrator, by combining multiple RF-DAC and RF-ADC signals from the RF Data Converter IP onto interfaces.

RF Interface Subsystem (JESD204C)

Figure 5-9 shows the block diagram for RF Interface Subsystem when using JESD204C to connect to external DACs and ADCs. Compared to the RFSoC Reference Design, no Analog Vector Source is provided with the JESD204C interface. The Reference Design for JESD204C supplied are provided with a single transmit and receive path, so no SRX multiplexer is shown to select the feedback path.



X22178-042821

Figure 5-9: **RF Interface Subsystem (JESD204C)**

Vector Subsystem (RFSoC Designs)

The block diagram of a Vector Subsystem is shown in Figure 5-10. This block takes the Analog Vector Source from the RF Interface Subsystem and uses it to create a vector source for the Transmit Subsystem. The Tick block is used to add a timing information on the TUSER signals on the AXI4-Stream. The Gain block is used to adjust signal levels on the vector stream.

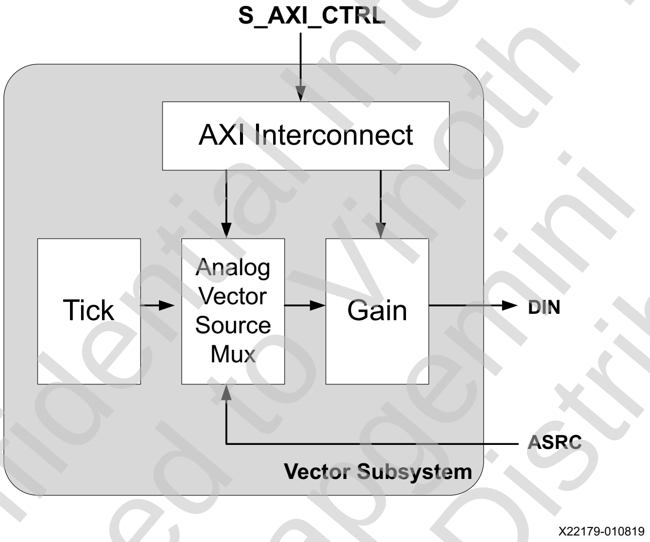


Figure 5-10: **Vector Subsystem (RFSoC Designs)**

Vector Subsystem (JESD204b Designs)

The Vector Subsystem for JESD204b designs is shown in Figure 5-11. Instead of Analog Vector Source, a tone source is used (the implementation is a complex mixer (CIF) mixing applied to a constant input).

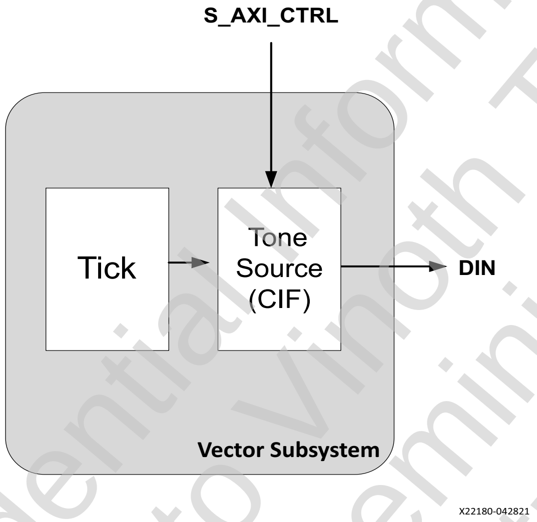


Figure 5-11: **Vector Subsystem (JESD204b Designs)**

Clocking, Reset, and Configuration Subsystem

Figure 5-12 shows the Clocking, Reset, and Configuration Subsystem. The clocks generated and their associated resets can be categorized into three main groups:

- **Control Clocking**: A clock wizard provides clocks for the AXI Interconnect in the design that is used to provide access to the control registers of the blocks. These clocks must be brought up first in the system. A Processor System Reset block provides resets, and will not release the system resets until the clocks are stable. The control clocking generates a main clock (s_axi_ctrl_1x_aclk) which is used by the AXI4-Lite Interconnect in the design.
- **User Clocking**: A clock wizard is also used to provide a user clock, which can be used to provide a separate control clock running at a different rate. This is used for the user S_AXI_USER interface on the DPD IP Core. It is using a similar structure to the control clocking sub-block above.
- **DFE Clocking**: The Digital Front End (DFE) Clocking block provides clocks for the DFE processing in the design. These clocks are used for the data processing in the design. These clocks are brought up under software control, once the control and user clocks are brought up. The clock wizard allows a choice of input clocks. The first input is from a local reference clock taken from a Silicon Device Si570 on the board which connects

directly into the FPGA fabric. In the case of ZCU111 board, the second input clock is from the dedicated RF clocking on the board which connects into the RF Data Converters on the RFSoC Device. In the case of ZCU208 or ZCU216 board, the second input clock is from CLK104 clocking module. The clocking wizard generates output clock rates, clkoutN which depend on the design. The clock mapping connections, maps the output clocks to different functional clocks for different parts of the design. The functional clocks also connect to the DFE Clock Control Block, which generates resets for each functional clock. The DFE Clock Control Block also provides status connections to LEDs and control of the DFE Input Clock Source.

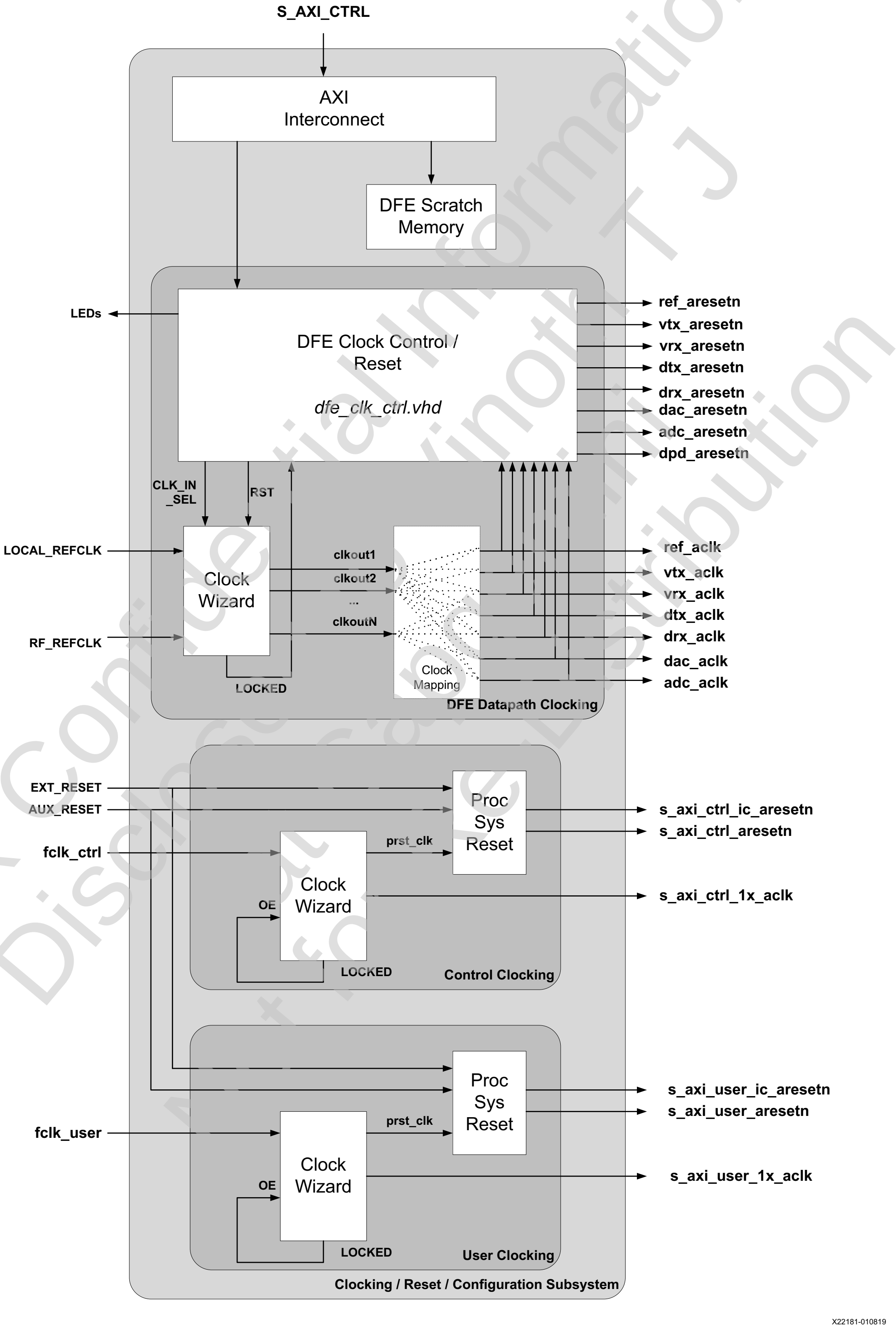


Figure 5-12: **Clocking, Reset, and Configuration Subsystem**

A list of clock and resets generated is given in Table 5-2 for reference.

The subsystem also includes support for system bring-up and configuration. It includes a small scratch RAM which is used by DPD Demonstration Software during system boot-up.

Table 5-2: **Clocks and Resets**

Clock Name	Reset Name	Description
-	EXT_RESET	Active-High External Reset.
-	AUX_RESET	Active-Low Auxiliary Reset
fclk_ctrl	-	Input Clock for Control Clocking
s_axi_ctrl_1x_aclk	s_axi_ctrl_aresetn	Control Output Clock/Reset
-	s_axi_ctrl_ic_aresetn	Control Interconnect Reset
fclk_user		Input Clock for User Clocking
s_axi_user_1x_aclk	s_axi_user_aresetn	User Output Clock/Reset
	s_axi_user_ic_aresetn	User Interconnect Reset
LOCAL_REFCLK	-	Local Reference Clock
RF_REFCLK	-	RF Reference Clock
ref_aclk	ref_aresetn	Reference Clock/Reset
vtx_aclk	vtx_aresetn	Vector Subsystem TX Clock/Reset
vrx_aclk	vrx_aresetn	Vector Subsystem RX Clock/Reset
dtx_aclk	dtx_aresetn	Digital Front End (DFE) TX Clock/Reset
drx_aclk	drx_aresetn	Digital Front End (DFE) RX Clock/Reset
dac_aclk	dac_aresetn	RF Interface DAC Clock/Reset
adc_clk	adc_aresetn	RF Interface ADC Clock/Reset
-	dpd_aresetn	Reset for DPD Core Only. Synchronous to dtx_aresetn.

Implementation Flow

Setting Up the Environment

Setup the environment before running the software flow on Linux.

- Setup Vivado using `settings64.sh` script. For more information, see *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973)* [Ref 5].
- Setup PetaLinux using `setting.sh` script. For more information, see Chapter 2 of *PetaLinux Tools Documentation Reference Guide (UG1144)* [Ref 13].
- You can use `settings64.sh` a Xilinx Software Command-line Tool (XSCT) script to setup the environment.

Note: You may encounter incompatibilities while setting up the environment between PetaLinux and Vitis software platform. Xilinx recommends you to run PetaLinux and XSCT in separate shells to avoid problems. Instructions are provided for each step as to which tool environment to use (Vivado, PetaLinux, or Vitis software platform).

Hardware Flow: Generating Vivado Design

The IP Reference Design is supplied as part of the DPD IP Core Delivery archive. To create a reference design, you need to run the `run.tcl` script. Table 5-3 describes the ways to create a reference design. The `run.tcl` script generates a Vivado project and creates a Vivado IP integrator design using an IP from the Vivado IP Catalog. DPD IP repository and IP are defined using an IP integrator module reference flow. It then synthesizes and implements the design and exports a Xilinx® shell archive (XSA) file to be used in the software implementation flow.

Table 5-3: **Reference Design Steps**

Method	Steps
GUI	Start Vivado Tools. Go to **Tools**, select Run **Tcl Script** menu option to run the `run.tcl` script from the ip_ref_design/<ref_design_name>/hw_flow directory.

Vivado Tcl Console	Start Vivado Tools. In **Tcl Console Window**, enter the following command:
	`cd ip_ref_design/<ref_design_name>/hw_flow`
	`source run.tcl`

Command Line	Ensure Vivado tools are on your path. Execute the following command:
	`cd ip_ref_design/<ref_design_name>/hw_flow`
	`vivado -mode tcl -source run.tcl`

Table 5-4 shows the `ip_ref_design/hw_flow`.

Table 5-4: **Hardware Flow**

Directory	Description
`run.tcl`	Runs the hardware flow in the Vivado
`project.tcl`	Subscript that generates the Vivado project
`design_bd.tcl`	Vivado IP integrator block design
`src/`	Project sources
`hdl/`	HDL source files for IP modules
`interfaces/`	Additional interface definitions for Vivado IP integrator
`xdc/`	Project constraints

Running the Software Flow

The `ip_ref_design` / `sw_flow` directory provides a set of scripts to build PetaLinux and to compile a host application using the DPD host API. Table 5-5 shows the `ip_ref_design/sw_flow`.

Table 5-5: **Software Flow**

Directory	Description
`plnx_run1.sh`	Shell script to create PetaLinux project, to update configuration files, and to create sysroot.
`run1_files/`	Additional configuration files for PetaLinux first stage. XSCT script `compile_dpd_host_app.tcl` to create the Vitis software platform project and to compile an example host application.
`plnx_run2.sh`	Second stage PetaLinux script. Adds the application binaries to PetaLinux image and rebuilds.
`copy_products.sh`	Copies products to products area and includes boot script.
`xsdb_jtag_boot.tcl`	XSDB boot script for booting PetaLinux image over JTAG.
`boot.cmd`	UBoot JTAG boot script commands. Can be used to set additional UBoot environment variables.
`boot.scr`	UBoot JTAG boot script image. Memory image for JTAG download produced by mkimage utility.
`xsa/`	Exports directory from the hardware flow containing the XSA file
`plnx_cleanup.sh`	Cleans up PetaLinux directory by removing temporary files.

Perform the following steps to build PetaLinux images and to compile an example host application. These commands automate the steps described in Table 7-12.

Table 5-6: **Building PetaLinux Images and Compiling an Example Host Application**

Step		Command/Description
1	Vivado: Run the hardware flow and export the XSA file	See Hardware Flow: Generating Vivado Design section on generating `system.xsa` file.

2	PetaLinux: First Stage PetaLinux Build	`cd ip_ref_design/<ref_design>/sw_flow`
		`./plnx_run1.sh`
		This script creates PetaLinux project from a BSP. BSPs for each board must be downloaded as described in *Vitis Unified Software Platform Documentation: Embedded Software Development* (UG1400) [Ref 21]. The script uses the environment variable $BSP_DIR to allow the default BSP location to be changed. The script also updates the PetaLinux configuration files and then imports the XSA file generated by the hardware flow. It builds an initial PetaLinux image and then uses it to create a sysroot for the software compilation.

3	XSCT: Compile Host Application	`xsct compile_dpd_host_app.tcl`
		This XSCT script creates a Vitis software platform project and adds an example DPD host application to it. The script compiles using the source from the `common/embedded_sw` directory of the archive. The script compiles and links against the sysroot generated by PetaLinux in the previous step. Use the `vitis -workspace work_dpd-app-smp` command to open the Vitis software platform project after running XSCT.
		`xsct compile_*_app.tcl`
		If any other application is present, run those XSCT script to create the corresponding Vitis software platform project for those example control application as well (see CFR and other IP applications in the References).

4	PetaLinux: Second Stage PetaLinux Build	`./plnx_run2.sh`
		`./copy_products.sh`
		This script runs PetaLinux to add application binaries for the example host application and for the DPD application executable to the Linux root file system. The `copy_products.sh` script copies products to the `./products` directory together with the `xsdb_jtag_boot.tcl` script which can be used to bring up the design on board.

Running the Example DPD Host Application Software

Chapter 7, Application Software describes controlling DPD Software from Host Software using the Host Application Programming Interface (API). An Example Host Application is provided with the IP Release Archive which illustrates use of the API. Details of running the Example Host Application are provided in this section.

Perform the following steps to boot Linux images and to run the example DPD host application. For more information on the board connectivity steps see *DFE Demo User Guide* (UG1163) [Ref 16].

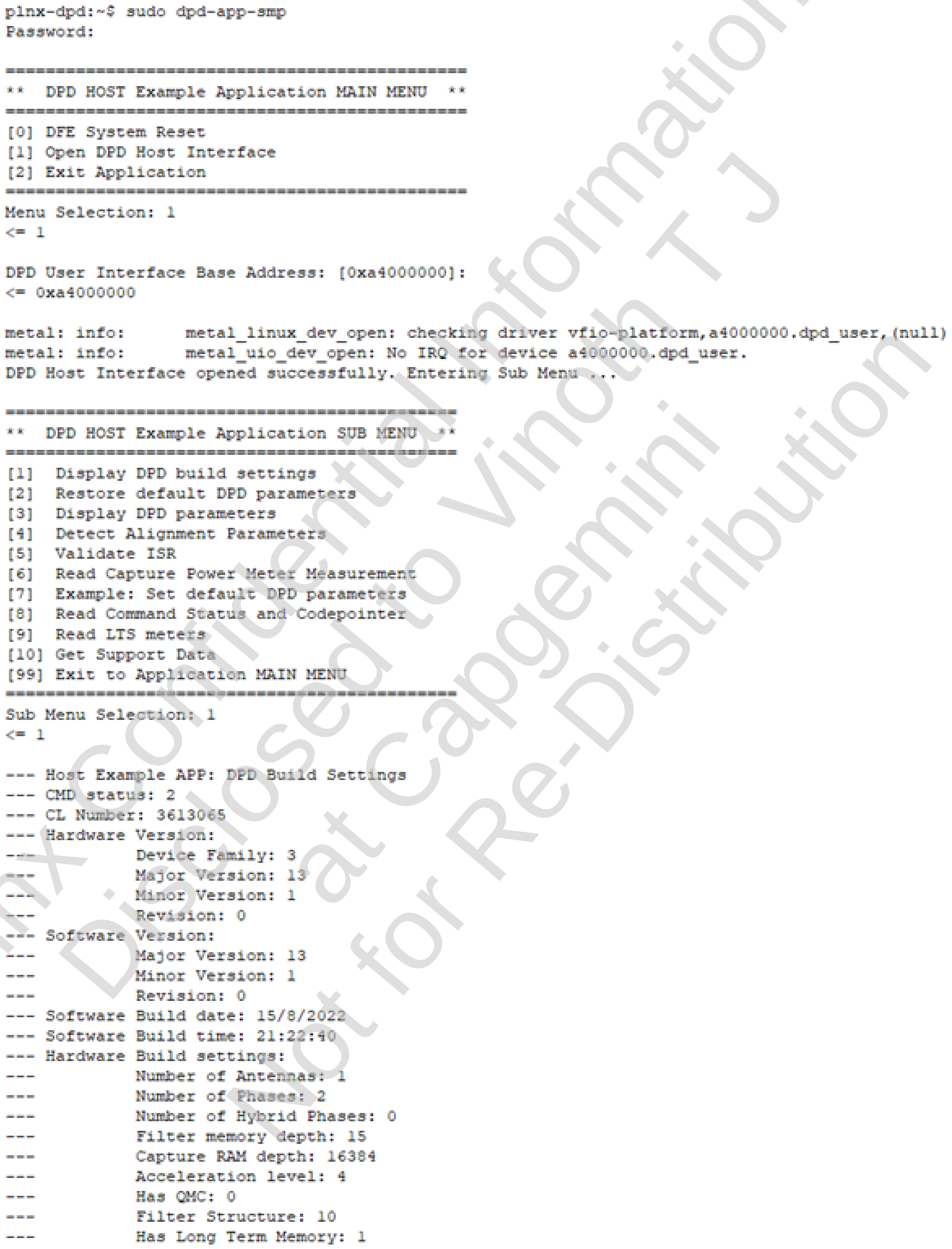
Table 5-7: **Booting Linux Images and Running the Example DPD Host Application**

Step		Command/Description
1	Setup UART and JTAG connections to the board and remove the SD Card	Power up and connect to the Xilinx Board and set up connections to JTAG and UART. Refer to the respective board user guide for more information. JTAG connection is managed via `hw_server` connection which may be running on the local machine or remotely via cable such as SmartLynq. Ensure that MODE pins are set to JTAG boot and remove the SD card from board as applicable.
2	Program the clock using the system controller	The reference design requires a correct clock frequency to be set for the `LOCAL_REFCLK` which is fed from on board Silicon Device Si570. Exact frequency varies for each design. You should check the required clock in the Vivado IP integrator by checking the `LOCAL_REFCLK` clock frequency in the DFE datapath clocking hierarchical block of the clocking and reset subsystem. ***Note:*** The DFE Demo automates this set-up directly over I2C interface to Si570 device, it does not require the clock to be programmed via the system controller. For more information see, *DFE Demo User Guide* (UG1163) [Ref 16].

3	Boot using XSDB	First, start XSDB:
		`cd ../products`
		`xsdb`
		In the XSDB shell,
		`connect -url TCP:<hw_server_ip_addr>:3121`
		`source xsdb_jtag_boot.tcl`

4	Start the applications	From the UART terminal, you can login to the embedded Linux using `petalinux` as your username and `plnx` as your password. From the cmd prompt, you can start the DPD application. Type `sudo dpd-smp help` command to access help on the application.
		The most important option is the `-u` option that specifies which UIO device to be used to run the DPD application. Multiple UIO devices may exist on the board. To find the particular UIO device associated with an instance of DPD, check the emebedded linux sysfs directory (`/sys/devices/platform/amba_pl@0/a0000000.dpd/uio`) on the MPSoC and RFSoC devices. The exact path is dependent on the device tree. For example, if you find the UIO device listed as uio4, then the command to start the DPD application would be:
		`sudo dpd-smp -u 4 &`
		To run the example host application:
		`sudo dpd-app-smp`

Figure 5-13 shows the Example Host Application while it is running. The CLxyzabc reported in the line `Build: Linux SMP: CLxyzabc` is just an example. For accurate CL (Change List) number, see the Release Notes.



```
plnx-dpd:~$ sudo dpd-app-smp
Password:

===============================================
**  DPD HOST Example Application MAIN MENU  **
===============================================
[0] DFE System Reset
[1] Open DPD Host Interface
[2] Exit Application
===============================================
Menu Selection: 1
<= 1

DPD User Interface Base Address: [0xa4000000]:
<= 0xa4000000

metal: info:      metal_linux_dev_open: checking driver vfio-platform,a4000000.dpd_user,(null)
metal: info:      metal_uio_dev_open: No IRQ for device a4000000.dpd_user.
DPD Host Interface opened successfully. Entering Sub Menu ...

==============================================
**  DPD HOST Example Application SUB MENU  **
==============================================
[1]  Display DPD build settings
[2]  Restore default DPD parameters
[3]  Display DPD parameters
[4]  Detect Alignment Parameters
[5]  Validate ISR
[6]  Read Capture Power Meter Measurement
[7]  Example: Set default DPD parameters
[8]  Read Command Status and Codepointer
[9]  Read LTS meters
[10] Get Support Data
[99] Exit to Application MAIN MENU
==============================================
Sub Menu Selection: 1
<= 1

--- Host Example APP: DPD Build Settings
--- CMD status: 2
--- CL Number: 3613065
--- Hardware Version:
---          Device Family: 3
---          Major Version: 13
---          Minor Version: 1
---          Revision: 0
--- Software Version:
---          Major Version: 13
---          Minor Version: 1
---          Revision: 0
--- Software Build date: 15/8/2022
--- Software Build time: 21:22:40
--- Hardware Build settings:
---          Number of Antennas: 1
---          Number of Phases: 2
---          Number of Hybrid Phases: 0
---          Filter memory depth: 15
---          Capture RAM depth: 16384
---          Acceleration level: 4
---          Has QMC: 0
---          Filter Structure: 10
---          Has Long Term Memory: 1
```

Figure 5-13: **Running DPD Host Application**

The Example Host Application is a console based application with a simple interface that allows you to:

- Reset DFE System
- Open DPD Host Interface

After you open the DPD Host Interface by choosing the option, a sub-menu interface allows you to:

- Display Build Settings
- Restore Default DPD Parameters
- Display DPD Parameters
- Detect Alignment Parameters
- Validate ISR
- Read Capture Power Meter Measurement and provides an example option to set default DPD parameters.
- Set default DPD parameters
- Read Command Status and Codepointer
- Read LTS meters
- Get Support Data

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 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 7-21 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.

Chapter 7

Application Software

DPD Application Software

Xilinx® DPD is an integrated Hardware and Software solution for Zynq UltraScale™ MPSoC/RFSoC parts with the DPD Application Software running on Zynq Processing System (PS) within these devices. The DPD Application Software is supplied as part of the IP Release Archive and can be found in the `software/bin` directory.

Customers control the DPD Application software with their own Host Application, communicating with DPD Host Interface. Both the DPD Application software and host application will operate on the Zynq PS, preferably on a single OS on the APU. These software may be assigned a priority to operate on different APU cores. A host API (Application Programming Interface) is provided to simplify writing of customer Host Applications. This can be found in the `common/embedded_sw/dpd_host_api` directory of the release archive. The required Operating System (OS) for running DPD is Embedded Linux (such as Xilinx PetaLinux) which supports Symmetric Multi Processing (SMP). SMP allows applications to run on any of the available processor cores in the device. Figure 7-1 shows the high-level software architecture for running the DPD Application Software communicating with Customer Host Application on Embedded Linux.

Linux SMP is supported for all devices (Zynq UltraScale+ MPSoC/RFSoC). Further details can be found on this OS configuration in the Software Configurations section.

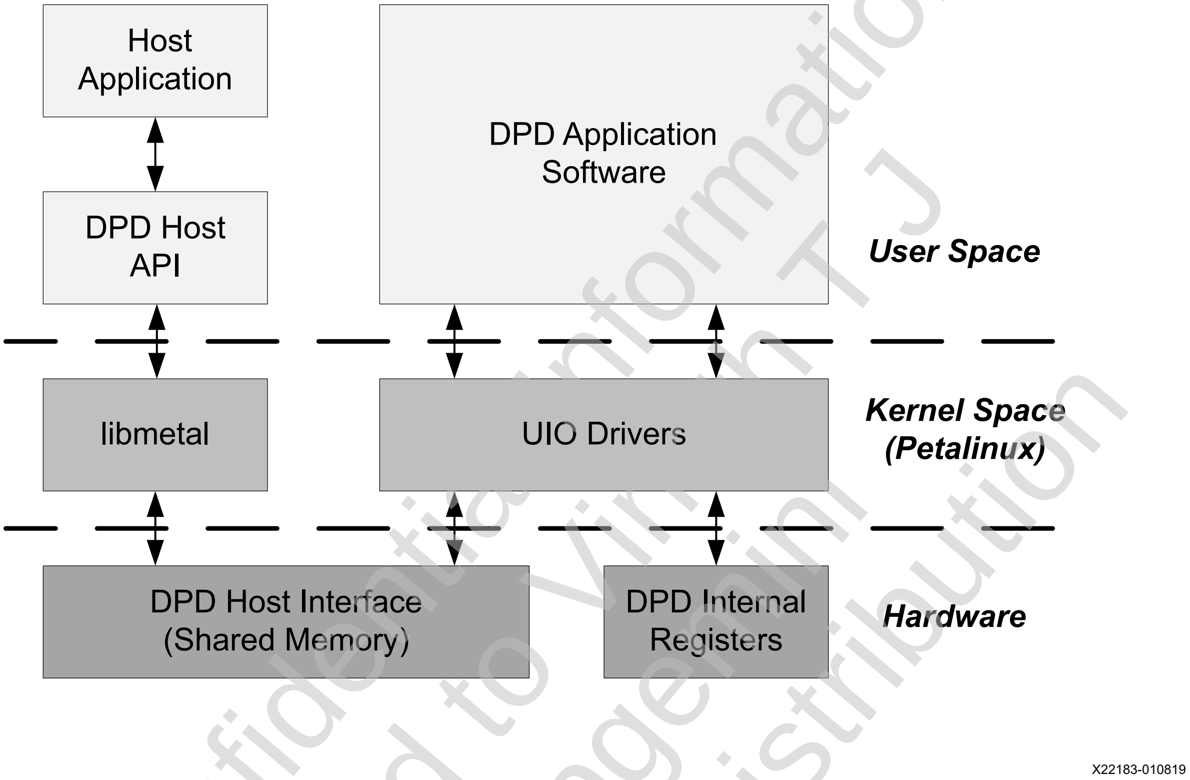


Figure 7-1: **High-Level Software Architecture (Linux)**

The following table lists the software devices that are supported.

Table 7-1: **Software Device Support**

Device Family	OS Configuration	Description/Archive Location

Zynq UltraScale+ RFSoC
Zynq UltraScale+ MPSoC
	Linux SMP	All A53 Cores.
		`bin/psu_cortexa53/dpd-smp/newline`
		`(aarch64 64-bit executable)`

For more information on Software Development on Xilinx Devices, see the *Zynq UltraScale+ MPSoC Software Developers Guide* (UG1137) [Ref 14].

DPD Host API

This section details the DPD Host Application Programming Interface (API). The DPD Host API is provided as C code and compiles using the *Libmetal and OpenAMP for Zynq Devices User Guide* (1186) [Ref 19] on the Linux operating system. Libmetal abstracts details of the operating system and provides support for other operating systems, but testing of the Host

API has been restricted to Embedded Linux running in SMP mode. Figure 7-1 shows how the DPD Host API fits into the overall software architecture for Embedded Linux.

The DPD Host API is supplied with an Example Host Application to illustrate use of the API. The Example Host Application is also supplied as part of the IP Release Archive. It can be useful to show that the DPD Hardware and DPD Software are working correctly. Further details can be found in the IP Reference Design Section Implementation Flow.

Communication with the DPD Software is made through the host interface shared memory. For more information on the Host interface and its operation, see the Host Interface Operations Guide. The DPD Host API typically provides a one-to-one relationship with the Host Interface Command Codes. It abstracts away details of managing the host interface protocol and type encoding to simplify integration into customer applications.

DPD Host API Operation Modes

The Host API has three main modes of operation:

- **Initialization**: In this mode of operation the valid sequence of API calls is `xdpd_open_host_if()` which will get a handle for Host Interface and `xdpd_init()` which initializes the DPD Software into the single-step mode.
- **Single-Step Mode**: In this mode of operation, a complete set of commands are available for configuration, running and monitoring of the DPD Algorithm. Control of SRX Port Selection is done explicitly. This mode is used primarily during system configuration and debug.
- **Dynamic Control Layer (DCL) Mode**: In this mode of operation, a smaller set of commands are available, with the DPD updates and SRX Port Selection being run automatically by the DCL Software. This is the main operating mode of DPD.

Single-Step Commands are typically used for system configuration, initialization and debug. Single-Step commands are indicated in tables Table 7-4, Table 7-5, Table 7-6, Table 7-7, Table 7-8, Table 7-9, and Table 7-10 by existence of Single-Step Command Code (CMD) for the command.

DCL Commands are used for controlling the DCL operation. DCL Commands are indicated in tables Table 7-4, Table 7-5, Table 7-6, Table 7-7, Table 7-8, Table 7-9, and Table 7-10 by the existence of DCL Command Code (MM) for the command. The word ‘auto’ for a command code, indicates that DCL is automatically running these commands.

Many API Calls can be run in both Single-Step and DCL Modes. If an API Call relies on switching the SRX Port, then in DCL Mode it will wait for this port to be scheduled by the DCL Layer, whilst the SRX Port switch will start immediately in Single-Step Mode.

Some API calls do not use the control handshake and directly read host interface registers, so that it can be run in either Single-Step or DCL modes.

DPD Host API Command Status

Each Host API call returns a command status. This uses the error control codes defined for the host interface with additional error codes to indicate API-Level Errors which are shown in Table 7-2.

Table 7-2: **Host API Additional Error Codes**

Value	Name	Description
-3000	XDPD_STATUS_API_TIMEOUT	Timeout polling register in Handshake Protocol. See Figure 7-14.
-3001	XDPD_STATUS_API_METAL_FAIL	Libmetal Error Occurred during xdpd_host_if_open()
-3002	XDPD_STATUS_API_UNEXPECTED_CMD	Unexpected Command found when completing long command. See finish_swept_capture_power() and finish_monitor_capture_location() API calls.
-3003	XDPD_STATUS_API_UNEXPECTED_OPMODE	Unexpected Operating Mode during API call.
-3004	XDPD_STATUS_API_UNEXPECTED_PAGE	Unexpected page number access requested in API call.
-3005	XDPD_STATUS_API_UNEXPECTED_RAMSTATE	Unexpected capture RAM state during API call.

DPD Host API Parameters

The DPD Host API uses a common set of names for API parameters. The parameters are shown in Table 7-3 to simplify understanding and description of each API call. For example, calls with the `portnum` parameter applies the command only to the port specified, and in some cases changes the current SRX port to the one specified. Commands with the `portmap` parameter indicate that a set of ports can be specified and never requires an SRX port change.

Table 7-3: **DPD Host API Parameters**

Argument Name	Type	Description
host_if	xdpd_host_interface	Host Interface Handle. Initialized with xdpd_sys_init() call. Provides handle for API to access Host Interface Registers
base_addr	unsigned int	Physical Base Address of DPD s_axi_ctrl Interface
portnum	unsigned int	Specific Port Number. Causes change of SRX Port using m_axis_srx_ctrl port on IP Core in Single-step Mode. API will uses PORTNUM register to select port. Commands using portnum argument to change SRX Port are known as Port Specific commands.

Table 7-3: **DPD Host API Parameters *(Cont'd)***

Argument Name	Type	Description
portmap	unsigned int	Port Map. API Call may specify more than one port for command to operate on. The portmap parameter is a 32-bit value specified with the upper 16 bits set to 0x0123 and the lower 8 bits used as a bit-mask to specify which port(s) the command should be applied. The LSB maps to port 0. For example, to send a command to ports 0, 2, 3, and 6, the value for portmap would be 0x0123004D. portmap[31:16] = 0x0123 portmap[15:8] = reserved portmap[7:0] = port bit-mask ***Note:*** The API automatically inserts 0x0123 prefix.
sid	unsigned int	Specific Coefficient Set Identifier (SID)
ecf_parameter	xdpd_ecf_params	ECF Parameters Type. See Table 7-22.
arch_parameter	xdpd_arch_params	Architecture Parameters. See Table 7-21.
cap_parameter	xdpd_cap_params	Capture Parameters. See Table 7-23.
dcl_parameter	xdpd_dcl_params	DCL Parameters. See Table 7-24.
odd_parameter	xdpd_odd_params	Over-Drive Detections Parameters. See Table 7-26.
met_parameter	xdpd_met_params	Meter Length Parameters. See Table 7-27.
qmc_parameter	xdpd_qmc_params	QMC Parameters. See Table 7-25.
capwin_parameter	xdpd_capwin_params	Capture Window Parameters. See Table 7-28.
msp_parameter	xdpd_msp_params	Multi-Set Power Parameters, See Table 7-29.
coeffs	unsigned int	Coefficients
ram	unsigned int	Identifies RAM to use for transfer.
page	unsigned int	Identifies RAM Page with RAM to transfer for capture/responses. Each RAM Page is 512 x 32-bit words in size. Number of RAM Pages is provided by capture RAM Size in Build Settings.
decim	unsigned int	Decimation
mode	unsigned int	Mode for Snapshot
monitors	xdpd_monitors	DPD Continuous Monitors for single port
dcl_monitors	xdpd_dcl_monitors	DCL Monitors for specified port
diag	xdpd_diagnostics	DPD Diagnostics
settings	xdpd_build_settings	DPD Build Settings
<analysis>_data	double, float, unsigned int	Signal Analysis Data return types
cap_pwr	xdpd_cap_pwr_measure	See READ_CAPTURE_PWR_METERS command.

DPD Host API Calls

The tables - Table 7-4, Table 7-5, Table 7-6, Table 7-7, Table 7-8, Table 7-9, and Table 7-10 provide a summary of the DPD Host API Calls. The tables list each of the API calls together with a summary of command codes used in the Host Interface for Single-Step and DCL modes, with a list of parameters and a description. Cross-references to the relevant Host Interface documentation are provided where applicable.

The full API can be found by reference to the header file dpd_host_api.h. The code also includes comments for the Doxygen Code Documentation Tool. Output from the Doxygen tool can be found in the IP Release Archive in the software/doc/html directory.

Each API Call has a common structure:

- Takes pointer to `xdpd_host_interface* host_if`: The host interface is used to store interface regarding the host interface and libmetal handles
- Takes zero or more additional arguments. These are detailed in the documentation. Structured data types are used for more complex parameter types.
- Returns an integer error code as documented in the Host Interface operation guide. Negative values indicate command failure.

The DPD Host API Calls are broken into the following main groups:

- **System Initialization**: These are used to initialize or close access to the host_interface using the libmetal drivers.
- **Parameter Setup**: These calls are used to configure the DPD Application Software, and typically take structured data types as parameters. Matching update report calls are provided for each parameter group.
- **DPD Control**: These calls are used to control the DPD Algorithm once it has been configured correctly. They are used to reset algorithm, do capture and alignment, and run updates.
- **DCL Control**: Commands to enter/exit and control DCL (Dynamic Control Layer) Mode.
- **DPD Monitors and Diagnostics**: API calls allow diagnostics and monitors for DPD operation to be reported and cleared, monitoring of error histograms, and to log a history of values using Snapshot monitor calls.
- **TX Quadrature Modulation Correction (QMC) Control**: API Calls to enable/disable QMC on Transmit Interface, and to load/store coefficients.
- **Signal Analysis**: API Calls to allow capture to be made for analysis and variety of histograms to be collated.

Note: In the following API description, a pointer to the host_if structure is required for all calls. It is not listed in the list of arguments to save space.

System Initialization

The `xdpd_open_host_if()` call is used to open access to the host interface and populates the xdpd_host_interface structure. It takes two additional arguments, one is the interface name in the device tree, and the second is the offset of the `s_axi_ctrl` interface in this space which will be 0x1C00000 for current designs. The `xdpd_read_configuration()` call can be used to get data on hardware and software build settings. The System Initialization API Calls are listed in Table 7-4.

Table 7-4: **System Initialization API Calls**

API Call	OPMODE	Description
xdpd_open_host_if()		Initialize Host Interface use offset = 0x1C00000 when accessing via s_axi_ctrl interface memory, or 0x0 when accessing via s_axi_user interface memory. Creates host_interface handle structure.
xdpd_init()	RESET, DCL or SINGLE_STEP	Return to Single Step Operation Mode. If boot_only, then will only do this if part of boot-up sequence, otherwise will also do this from DCL mode.
xdpd_close_host_if()		Shutdown Host Interface. Release host interface. Optionally stop_dpd by forcing single-step mode and using KILL_DPD command code.
xdpd_read_configuration()	SINGLE_STEP	Reads back the DPD configuration
xdpd_get_opmode()		Report Operation Mode.

Parameter Setup

The Parameter Setup API Calls are listed in Table 7-5. For more information, see Updating and Reporting DPD Parameters.

Table 7-5: **Parameter Setup API Calls**

API Call	OPMODE	Description
xdpd_restore_default()	SINGLE_STEP	Restores default configuration
xdpd_configure_structure()	SINGLE_STEP	Configure or Reconfigure the NLF hardware
xdpd_update_ecf_parameters()	SINGLE_STEP	Sets Estimation Core Parameters
xdpd_report_ecf_parameters()	DCL or SINGLE_STEP	Gets Estimation Core Parameters
xdpd_update_arch_parameters()	SINGLE_STEP	Sets Architecture Parameters
xdpd_report_arch_parameters()	DCL or SINGLE_STEP	Gets Architecture Parameters
xdpd_update_cap_parameters()	SINGLE_STEP	Sets Capture parameters
xdpd_report_cap_parameters()	DCL or SINGLE_STEP	Reports Capture parameters
xdpd_update_msp_paramters()	SINGLE_STEP	Sets Multi-Set Power parameters
xdpd_report_msp_parameters()	DCL or SINGLE_STEP	Reports Multi-Set Power parameters

Table 7-5: **Parameter Setup API Calls *(Cont'd)***

API Call	OPMODE	Description
xdpd_update_dcl_parameters()	SINGLE_STEP	Sets DCL parameters
xdpd_report_dcl_parameters()	DCL or SINGLE_STEP	Reports DCL parameters
xdpd_update_odd_parameters()	SINGLE_STEP	Sets ODD parameters
xdpd_report_odd_parameters()	DCL or SINGLE_STEP	Reports ODD parameters
xdpd_update_met_parameters()	SINGLE_STEP	Sets MET parameters[1]
xdpd_report_met_parameters()	DCL or SINGLE_STEP	Reports MET parameters
xdpd_update_txqmc_parameters()	SINGLE_STEP	Sets TX QMC parameters
xdpd_report_txqmc_parameters()	DCL or SINGLE_STEP	Reports TX QMC parameters
xdpd_update_capwin_parameters()	SINGLE_STEP	Sets CAPWIN parameters
xdpd_report_capwin_parameters()	DCL or SINGLE_STEP	Reports CAPWIN parameters
xdpd_dcl_update_ltm_tc()	DCL	Update the LTM time constant values
xdpd_dcl_update_damping_leakage()	DCL	Update the ECF damping and leakage values
xdpd_dcl_update_fixed_loopgain()	DCL	Update FIXED_GAIN_VALUE parameter.
xdpd_dcl_get_skip_port()	DCL	Get DCL Skip Port Mask
xdpd_dcl_set_skip_port()	DCL	Sets DCL Skip Port Mask

Notes:

1. METERLENGTH parameter is applied for all ports whereas NZCOUNTTHRESHOLD parameter can be applied for a specific port.

DPD Control

The DPD Control API Calls are listed in Table 7-6.

Table 7-6: **DPD Control API Calls**

API Call	OPMODE	Description
xdpd_change_srx_port()	SINGLE_STEP	Changes SRX Port (uses NO Operation Opcode)
xdpd_reset_coefficients()	DCL or SINGLE_STEP	Sets all DPD coefficients to unity gain
xdpd_reset_align_calibration()	DCL or SINGLE_STEP	Reset alignment calibration
xdpd_capture_and_align()	SINGLE_STEP	Captures samples and run alignment routines
xdpd_run_update_coefficients()	SINGLE_STEP	Performs an update of the DPD coefficients
xdpd_run_update_coefficients_v2()	SINGLE_STEP	Performs an update of the DPD coefficients
xdpd_run_update_coefficients_v3()	SINGLE_STEP	Performs an update of the DPD coefficients

Table 7-6: **DPD Control API Calls *(Cont'd)***

API Call	OPMODE	Description
xdpd_report_coefficients()	SINGLE_STEP	Reports a selected set of coefficients
xdpd_load_coefficients()	SINGLE_STEP	Loads a selected set of coefficients
xdpd_dpd_off()	SINGLE_STEP	Turn off DPD. Use 0dB Gain Pass-through Coefficients.
xdpd_dpd_on()	SINGLE_STEP	Turn on DPD. Use current coefficients
xdpd_disable_output()	SINGLE_STEP	Disable Filter Output.
xdpd_enable_output()	SINGLE_STEP	Enables Filter Output. Use current coefficients.

DCL Control

After `run_dcl()` call, DCL will automatically run schedule captures, alignment and coefficient updates for each port not masked by `portmap` specified by xdpd_dcl_set_skip_port() API. DCL operation can be monitored using `xdpd_dcl_get_monitors()` call. DCL exits once the `xdpd_exit_dcl()` command is executed.

The DCL Control API Calls are listed in Table 7-7.

Table 7-7: **DCL Control API Calls**

API Call	OPMODE	Description
xdpd_run_dcl()	SINGLE_STEP	Enables the DCL Scheduling Loop
xdpd_exit_dcl()	DCL or SINGLE_STEP	Disables the DCL Scheduling Loop
xdpd_reset_dcl()	DCL or SINGLE_STEP	Resets the DCL Scheduling Algorithm
xdpd_dcl_get_monitors()	DCL	Get DCL Monitors
xdpd_dcl_reset_coef_align()	DCL	Resets the coefficients and alignment calibration for specified ports.

DPD Monitors and Diagnostics

DPD Monitors and Diagnostic API Calls are listed in Table 7-8.

Table 7-8: **DPD Monitors and Diagnostic API Calls**

API Call	OPMODE	Description
xdpd_get_monitors()	DCL or SINGLE_STEP	Get Continuous Monitors for specified Port.
xdpd_report_diagnostics()	DCL or SINGLE_STEP	Reports the DPD update diagnostics.
xdpd_report_hwa_diagnostics()	DCL or SINGLE_STEP	Reports the HWA diagnostics.
xdpd_dcl_report_amam_ampm()	DCL	Report 256 samples of the latest AM/AM and AM/PM responses.

Table 7-8: **DPD Monitors and Diagnostic API Calls *(Cont'd)***

API Call	OPMODE	Description
xdpd_clear_diagnostics()	SINGLE_STEP	Clears the DPD diagnostics.
xdpd_clear_c2l_overflow()	DCL or SINGLE_STEP	Clears the coefficient overflow warning.
xdpd_dcl_reset_dcl_counters()	DCL	Reset the various counters associated with the DCL, including error histogram.
xdpd_reset_error_histogram()	DCL or SINGLE_STEP	Resets the error histogram counter.
xdpd_enable_snapshot_monitor()	DCL or SINGLE_STEP	Enable the snapshot monitor for selected ports.
xdpd_pause_snapshot_monitor()	DCL or SINGLE_STEP	Pause the updating of the snapshot monitor.
xdpd_snapshot_monitor_status()	DCL or SINGLE_STEP	Read the SNAPSHOTSTATUS register.
xdpd_dump_snapshot_monitor()	SINGLE_STEP	Dump a page of the snapshot buffer.
xdpd_dump_error_histogram()	DCL or SINGLE_STEP	Dumps error histogram for specified port

TX QMC Setup

The TX QMC API Calls are listed in Table 7-9.

Table 7-9: **TX QMC API Calls**

API Call	OPMODE	Description
xdpd_qmc_reset()	SINGLE_STEP	Resets the QMC coefficients
xdpd_qmc_off()	SINGLE_STEP	Updates QMC datapath with for pass-through
xdpd_qmc_on()	SINGLE_STEP	Updates QMC datapath without changing the coefficients
xdpd_report_qmc_coefficients()	SINGLE_STEP	Reports the internal QMC coefficients
xdpd_load_qmc_coefficients()	SINGLE_STEP	Updates the internal QMC coefficients

Signal Analysis

The Signal Analysis API Calls are listed in Table 7-10. For more information, see in the Host Interface Operations Guide section.

Table 7-10: **Signal Analysis API Calls**

API Call	OPMODE	Description
xdpd_capture_new_samples()	SINGLE_STEP	Triggers a new sample capture sequence
xdpd_read_capture_power_meters()	SINGLE_STEP	Presents the capture power meter values
xdpd_get_capture_ram_page()	SINGLE_STEP	Presents the specified Capture RAM page data

Table 7-10: **Signal Analysis API Calls *(Cont'd)***

API Call	OPMODE	Description
xdpd_get_histogram()	DCL or SINGLE_STEP	Presents the transmit histogram
xdpd_get_current_histogram()	SINGLE_STEP	Presents the transmit current histogram
xdpd_get_capture_psd()	DCL or SINGLE_STEP	Captures new samples and computes the power spectral density
xdpd_compute_amam_ampm_responses()	SINGLE_STEP	Captures samples and computes the AM/AM and AM/PM responses
xdpd_compute_pa_amam_ampm_responses()	SINGLE_STEP	Captures samples and computes the AM/AM and AM/PM responses
xdpd_get_response_ram_page()	SINGLE_STEP	Presents the specified Computed Response RAM page data
xdpd_run_odd_amam()	DCL or SINGLE_STEP	Run ODD algorithm on current coefficients
xdpd_start_swept_capture_power()	SINGLE_STEP	Presents the capture power at the specified delay
xdpd_finish_swept_capture_power()	SINGLE_STEP	Finishes Swept Capture Power (Long Command, returns XDPD_STATUS_API_TIMEOUT if command not finished)
xdpd_start_monitor_capture_location()	SINGLE_STEP	Performs 512 captures using the defined CAPTUREMODE
xdpd_finish_monitor_capture_location()	SINGLE_STEP	Finishes Monitor Capture Location (Long Command, returns XDPD_STATUS_API_TIMEOUT if command not finished)
xdpd_get_amam_page()	SINGLE_STEP	Read AMAM values after xdpd_compute_amam_ampm _responses() or xdpd_compute_pa_amam_a mpm_responses() API call
xdpd_get_ampm_page()	SINGLE_STEP	Read AMPM values after xdpd_compute_amam_ampm _responses() or xdpd_compute_pa_amam_a mpm_responses() API call
xdpd_report_dcl_monitor_views()	DCL or SINGLE_STEP	Presents DCL Monitor view at the host interface RAM screen
xdpd_update_dcl_monitor_views()	SINGLE_STEP	Updates DCL Monitor view with the data from the host interface RAM screen

Software Configurations

Zynq UltraScale+ MPSoC/RFSoC 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.

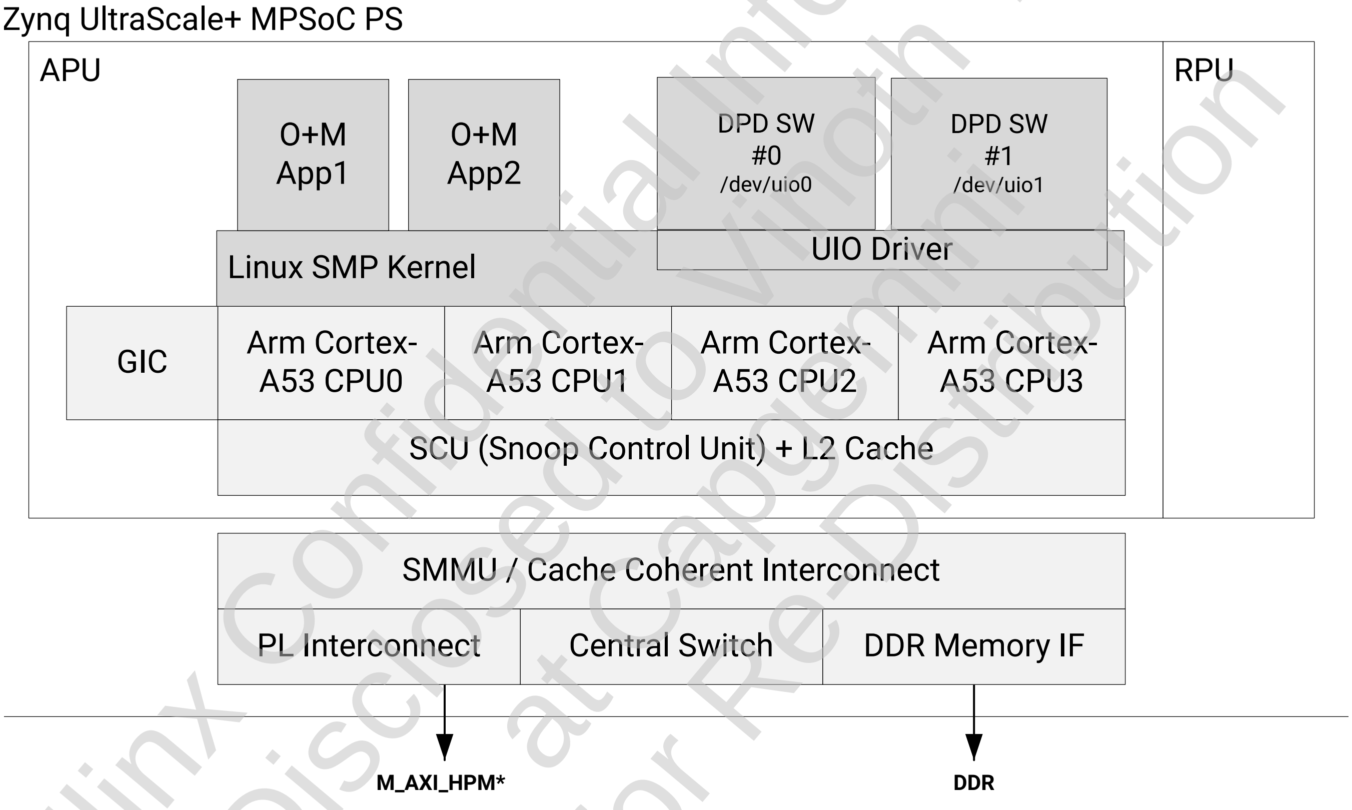


Figure 7-2: **Zynq UltraScale+ MPSoC/RFSoC/RFSoC DFE Multi-Processor Linux SMP Configuration**

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

- **Memory Map**: The recommended memory map for running is shown in Table 7-11. 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 7-11: **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 15] 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 guidance for creating a basic PetaLinux image to support operation with DPD for SMP configurations. The Software Flow for the DPD Reference Design provides example PetaLinux scripts which generate a PetaLinux embedded Linux image for the DPD Reference Design. The instructions provided below are only illustration of changes required. The reference design software scripts are customized for each particular reference design and should be followed for more precise guidance.

For more information on PetaLinux commands and workflow, see the *PetaLinux Tools Documentation Reference Guide* (UG1144) [Ref 13].

Table 7-12: **Creating Linux Images**

Step		Command/Description
1	Create Hardware Platform	Later Steps require the hardware definition for your design exported from Vivado as .xsa file. This hardware platform must meet the minimum configuration requirements to support Linux as described in **Creating a Project** in *PetaLinux Tools Documentation Reference Guide* (UG1144) [Ref 13].

2	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 **zynqMP** for Zynq MPSoC.

Table 7-12: **Creating Linux Images *(Cont'd)***

Step		Command/Description
3	Add Applications to Root File system	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* (UG1144) [Ref 13]. To run applications at the start up, see the section **Application Auto Run at Startup** in the same document.
4	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.
5	Configure Boot Arguments	Perform the following under **Kernel Bootargs** in the menu: • Click **n** to deselect automatic generation of boot arguments • Select arguments on the line below, and click **Enter** • Enter the Boot Arguments. On Zynq MPSoC/RFSoC, the following boot arguments are recommended: `earlycon=cdns,mmio,0xFF000000,115200n8  console=ttyPS0,115200` **`uio_pdrv_genirq.of_id=generic-uio cpuidle.off=1 clk_ignore_unused`** **`root=/dev/ram0`** Additional options are in bold. This option maps the generic-uio driver to the installed driver name.
6	Add UIO Drivers in Kernel	To add UIO Drivers to kernel: `petalinux-config -c kernel` Which will start a menuconfig configuration screen. Navigate 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 "*"
7	Add libmetal to rootfs	To add libmetal to rootfs: `petalinux-config -c rootfs` Which will start a menuconfig configuration screen. Navigate to **Filesystem Packages**->**libs**->**libmetal**. Ensure that libmetal is selected and the entry is marked with a "*"

Table 7-12: **Creating Linux Images *(Cont'd)***

<table>
<tr><th colspan="2">Step</th><th>Command/Description</th></tr>
<tr><td>8</td><td>Update Device Tree</td><td>The device-tree needs to be updated to map DPD to the Generic UIO Drivers. See <b>Device Tree Configuration</b> in the <i>PetaLinux Tools Documentation Reference Guide</i> (UG1144) [Ref 13] for an overview of customizing the device tree.
Update the following file:<pre>
project-spec/meta-user/recipes-bsp/device-tree/files/
system-user.dtsi.
</pre>Following is an example system-conf.dtsi file:<pre>
/include/ "system-conf.dtsi"
/ {
};
&amba_pl {
    TXCHAIN0_DFESS0_DPD_1_USER: dpd_user@a4000000 {
        compatible = "generic-uio";
        reg = <0x0 0xa4000000 0x0 0x20000>;
};
};
&TXCHAIN0_DFESS0_DPD_1 {
    compatible = "generic-uio";
    reg = <0x0 0xa0000000 0x0 0x4000000>;
};
</pre>TXCHAIN0_DFESS0_DPD_USER_1 is a new node in the device tree which is created to provide a separate node for DPD s_axi_user interface so that it can be used by the DPD Host API. TXCHAIN0_DFESS0_DPD_1 is the name of the node for DPD s_axi_ctrl interface in the device tree, and can be found by looking in the following file:<pre>
components/plnx_workspace/device-tree-generation/pl.dtsi
</pre>If a design contains multiple instances of DPD, then multiple nodes will have to be changed.
<b><i>Note:</i></b> The example <code>reg = <...></code> property in the device tree node is a tuple of 32-bit words and two words are required on 64-bit architectures such as MPSoC to specify addresses. Each reg property specifies an address range consisting of the base address followed by the size. The default PetaLinux device tree node for DPD combines the address ranges for both <code>s_axi_ctrl</code> and <code>s_axi_user</code> interfaces. But the changes in the <code>system-user.dtsi</code> above creates two separate nodes that simplify initialization of libmetal interface used by the DPD Host API. These nodes also change the default compatible property to allow the use of UIO drivers by the DPD Host Application.</td></tr>
<tr><td>9</td><td>Build PetaLinux</td><td>Build the PetaLinux Image by executing the command<pre>
    petalinux-build
</pre>The result of this command will be in the sub-folder images/linux. The generated files can be then used for Linux boot up. See Table 5-7 for booting system using JTAG.</td></tr>
</table>

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 mechanisms are 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 7-13.

Table 7-13: **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	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

Table 7-13: **Memory Map *(Cont'd)***

Word Address	Mnemonics	Notes
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	PARAMETER_1	Register used to pass a single parameter for various commands.
6	PARAMETER_2	Register used to pass a single parameter for various commands.
7	DCL_AUTO_EXIT_RANGE	Bits[31:16] 16-bit signed status value Bits[15:0] 16-bit signed status value The upper and lower 16 bits are used to define the range of status values (inclusive) that will cause the DCL to automatically exit. The limits may be specified as lower-value:higher-value or higher-value:lower-value.
8	DCL_AUTO_EXIT_PORTS	Specify the desired ports to monitor when DCL_AUTO_EXIT is enabled. bits [31:16] used to specify port specific or all port monitoring. bits [15:0] used to specify which ports to monitor. If bits [31:16] = 0xDBAE then each of the lower bits [7:0] are used as a bit mask to indicate which ports to monitor. Bit 0 corresponds to port 0. Bits with value of 1 are monitored, bits with value of 0 are ignored. When bits [31:16] != 0xDBAE the lower bits are ignored and all ports will be monitored for DCL_AUTO_EXIT feature.
9-15	Reserved	
16	COMMANDSTATUS	Status response for all non-DCL commands. See Table 7-19.
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[20:18] - PORTNUM active during capture • 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`.

Table 7-13: **Memory Map *(Cont'd)***

Word Address	Mnemonics	Notes
24	SNAPSHOTSTATUS	Reports current status of the snapshot buffer and DCL enabled. • Bit[0] - Indicates whether DCL routine is enabled or not. • Bit[1] - Indicates whether the snapshot monitor is enabled or not. • Bit[2] - Indicates whether the snapshot monitor is paused or not. • Bits[7:3] - Reserved • Bits[15:8] - Reports which snapshot monitor mode is enabled. • Bits[23:16] - Reports which ports are active in the snapshot. • Bits[31:24] - Reports how full the snapshot buffer is (in %).
25-29	Reserved	
30	ALT_RESULT0	Alternate status result
31	ALT_RESULT1	Alternate status result
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 ***Note:*** Month is computed as zero-based value (i.e. 0 - January).
34	BUILDTIME	Software build time: • seconds = mod(BUILDTIME,60) • minutes = mod((BUILDTIME-seconds)/60,60) • hour = ((BUILDTIME-seconds)/60 - minutes)/60
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 (CRD), L = $2^{(Bits[15:12] + 9)}$ • Bits[19:16] - Acceleration level • Bits[22-20] - Reserved • Bit[23] - has_qmc • Bit[24] - Long Term Memory • Bits[29:25] - Filter Structure
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: • Zynq UltraScale+ (also for RFSoC) ***Note:*** HWVERSION is available in the host interface before the DPD software has booted.

Table 7-13: **Memory Map *(Cont'd)***

Word Address	Mnemonics	Notes
37	SWVERSION	• Bits[23:16]- Major version • Bits[15:8] - Minor version • Bits[7:0] - Revision
38-59	Reserved	
60[1]	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.
61	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.
62	METERLENGTH	Reports the current METERLENGTH value programmed into the hardware
63	SRXQDC	Accumulation of the Q channel over the METERLENGTH interval (S32.0 value) Average D.C. in lsb's is (16*SRXQDC/METERLENGTH).
64	SRXIDC	Accumulation of the I channel over the METERLENGTH interval (S32.0 value) Average D.C. in lsb's is (16*SRXQDC/METERLENGTH).
65	SRXAVE	Average SRX power: • Bits[31:28] - Active port value • Bits[27:0] - Average SRX power value (U32.30 convert to dBFS using 10*log10(x))
66	DPDOUTPUT_PAPR	DPD output peak to power ratio: • Bits[31:28] - Active port value • Bits[27:0] - DPD output Peak to Average Power Ratio (U28.16 in dB)
67	DPDOUTPUTAVE	Average DPD output power value: • Bits[31:28] - Active port value • Bits[27:0] - Average DPD output power value (U32.30 convert to dBFS using 10*log10(x))
68 +5*n	TXPOWERAVE	DPD input average power of port n (U32.30 convert to dBFS using 10*log10(x))
69 +5*n	TXPAPR	DPD input Peak to Average Power Ratio of port n (U32.16 in dB)
70 +5*n	TXPOWERCOUNT_N	Number of non-zero samples in the power measurement of port n
71 +5*n	FCM_REG_N	Proprietary frequency content metrics (FCM) of port n
72 +5*n	ACTIVE_SID_N	Status indicating which coefficient set is loaded into the predistortion LUTs on port n 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 Monitors (antenna 0 through 7) (read only). See Table 7-33.	
512-1023	Page transfer	Paging memory used to transfer large data sets to/from the user environment.

Notes:

1. This value increments by one every METERLENGTH/fs seconds (typically 10-20msec), where fs is the DPD sample rate.

Hardware and Software Interaction

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

Figure 7-3 depicts the main elements of the DPD solution. The software processes are run in the Arm® processor core.

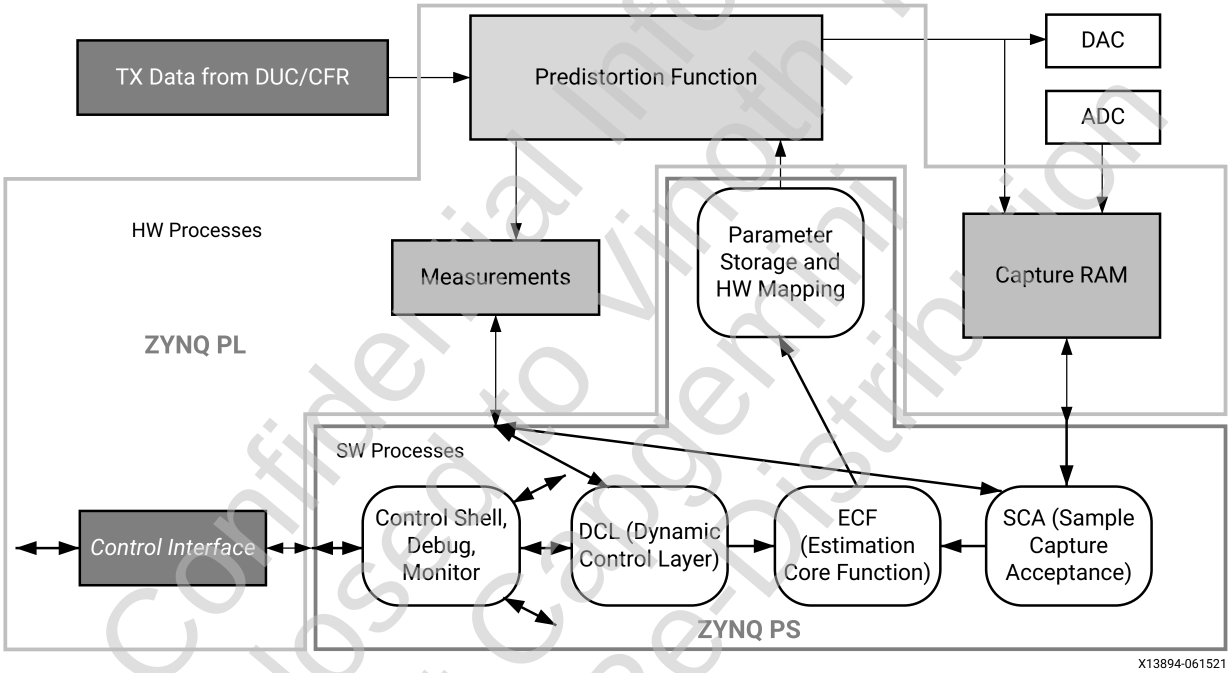

TX Data from DUC/CFR
Predistortion Function
DAC
ADC
HW Processes
Measurements
Parameter Storage and HW Mapping
Capture RAM
ZYNQ PL
SW Processes
Control Interface
Control Shell, Debug, Monitor
DCL (Dynamic Control Layer)
ECF (Estimation Core Function)
SCA (Sample Capture Acceptance)
ZYNQ PS
X13894-061521

Figure 7-3: **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 Status Values) 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 7-3 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.

A 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 undefined, 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. CAPWIN specification are not needed when operating with DCLALGORITHM=4.

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, 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 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 7-4. Figure 7-4(a) indicates that acceptable adjacent channel power (ACP) is maintained for all power levels below the power used for training the DPD coefficients. Figure 7-4(b) depicts a case where acceptable ACP is only achieved for a region close to the power used for training the DPD.

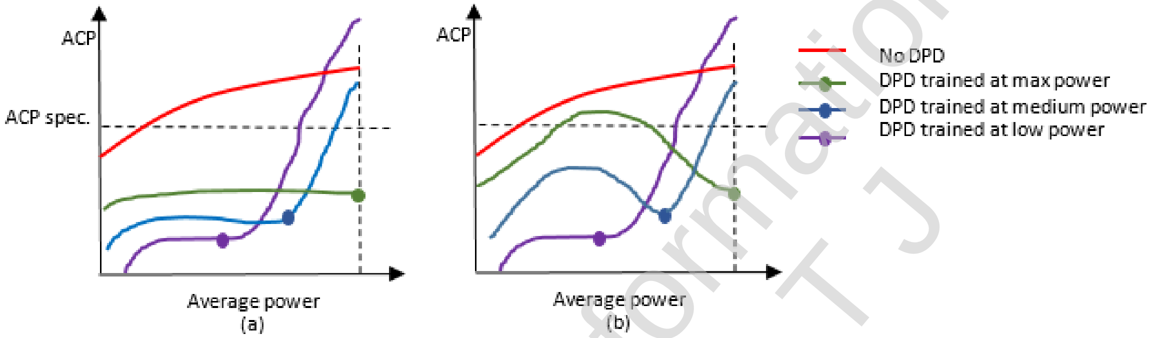


Figure 7-4: **Average Power Variation (a) Single Set Capable (b) Multi-Set Required**

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 7-5.

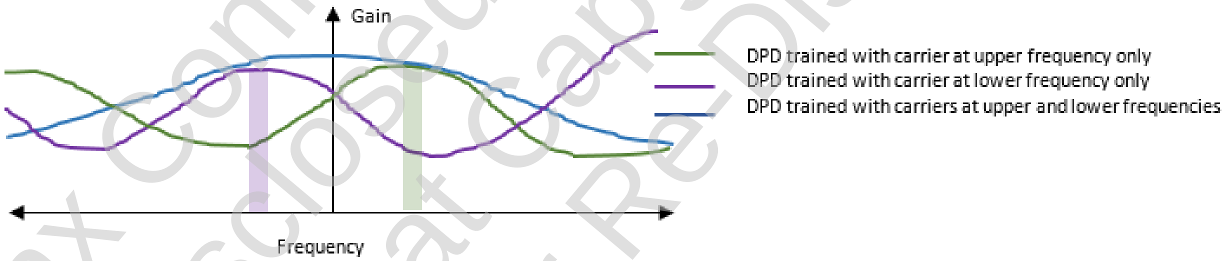


Figure 7-5: **Average Frequency Response of DPD Filter**

Note: 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 Multi-Set Systems

Consider the curves in Figure 7-4(b) and Figure 7-5. 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 7-6 indicates the desired performance that would be achieved by switching between coefficients sets as the power level changes.

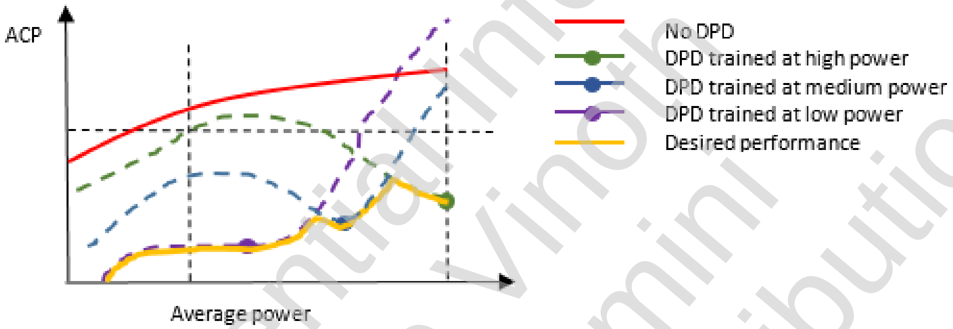


Figure 7-6: **Desired Performance**

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, 20 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 7-14 describes the TX and RX data format, IF choices, and QMC support.

Table 7-14: **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. This is a second user-defined threshold that is used to prevent the coefficients from being used.

Figure 7-7 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.

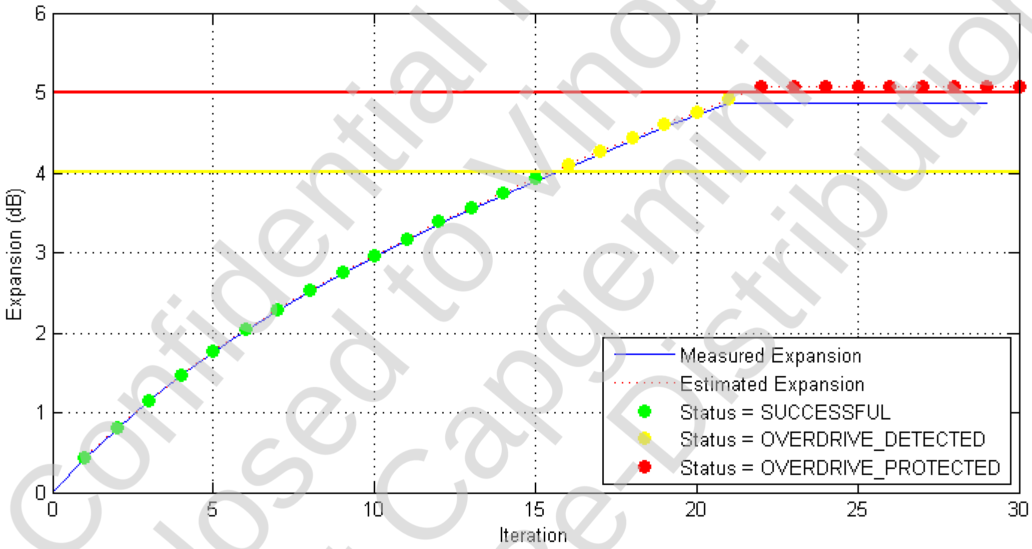


Figure 7-7: **Returned Status Value**

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.

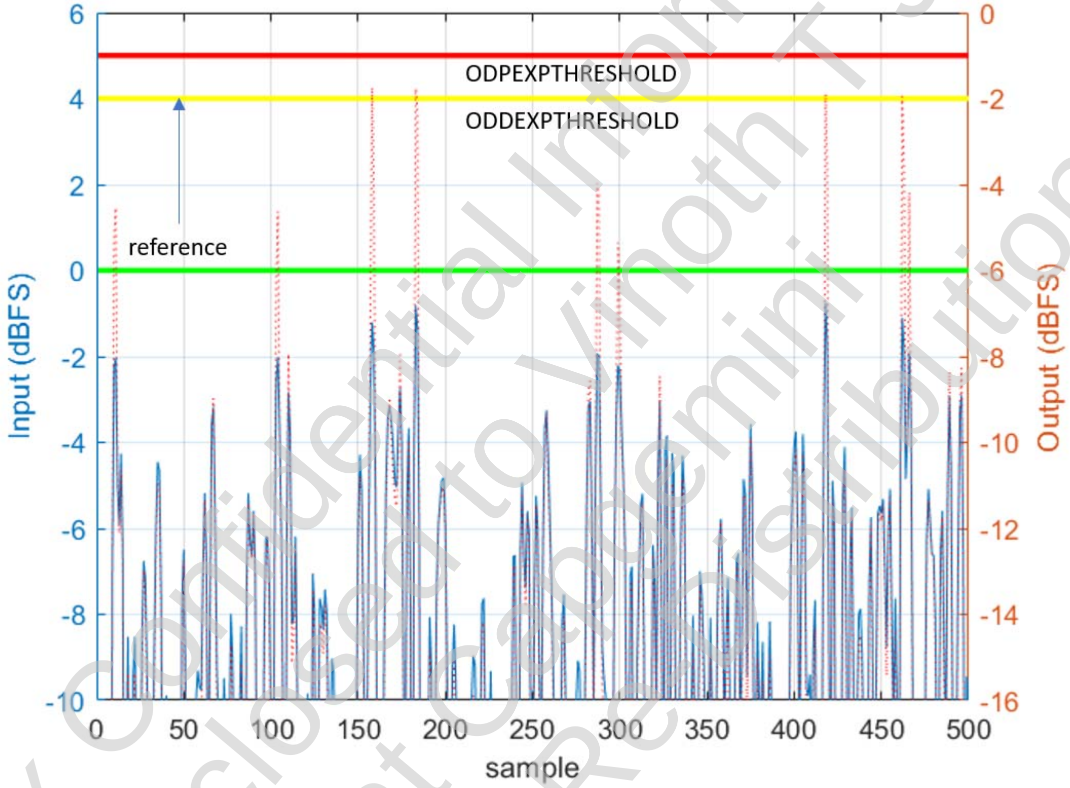


Figure 7-8: **Example of ODDEXPTHRESHOLDS for ODPENABLE = 2**

Peak Saturation (ODDPS0THRESHOLDS)

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 0 dBFS of the output. This mode operates by limiting the magnitude of the maximum peaks, it does not limit the amount of expansion.

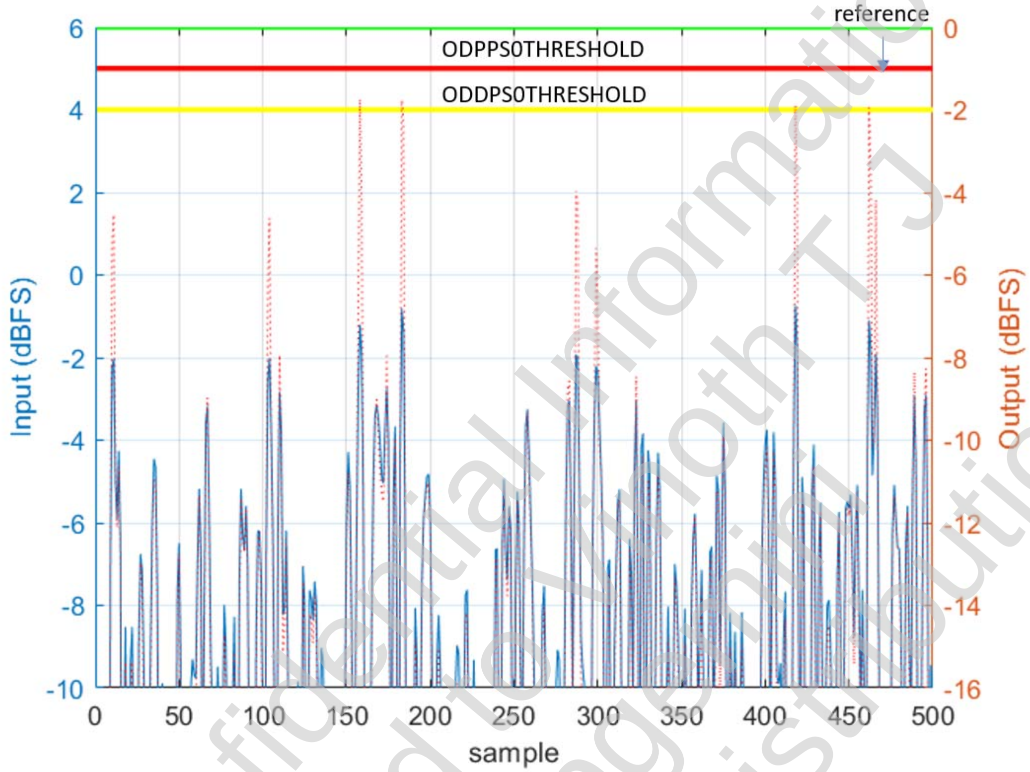


Figure 7-9: **Example of ODDPSOTHRESHOLDS for ODPENABLE = 1**

Coefficient Fall-back Feature

The DPD will attempt to automatically recover from a limited set of error conditions. When one of the following conditions occur the DPD has detected an issue with the current coefficient update. In these cases the DPD will fall-back or revert to previous set of coefficients (i.e., the coefficients that were loaded before the issue was detected). The coefficient fall-back is triggered under the following conditions:

- When AVERAGE_FILTER_GAIN_FAILURE is detected it is assumed that the coefficients that were last updated have caused this condition. In this case the previous coefficients will be re-loaded. The AVERAGE_FILTER_GAIN_FAILURE is not detected until the next update attempt on a given port, so this state will exist for the time it takes DPD to update all other ports.
- DPD_COEF_LUT_OVERFLOW_FAILURE indicates that the coefficients that were just loaded cannot be applied without causing overflows. When this occurs the new coefficient LUT values are not loaded and thus never applied to the actual signal. The coefficients fall-back to the currently loaded values.

- OVERDRIVE_PROTECTED prevents a new coefficient set from being loaded because the estimated expansion is beyond the user specified limits, and hence, the coefficients fall-back to the currently loaded values.

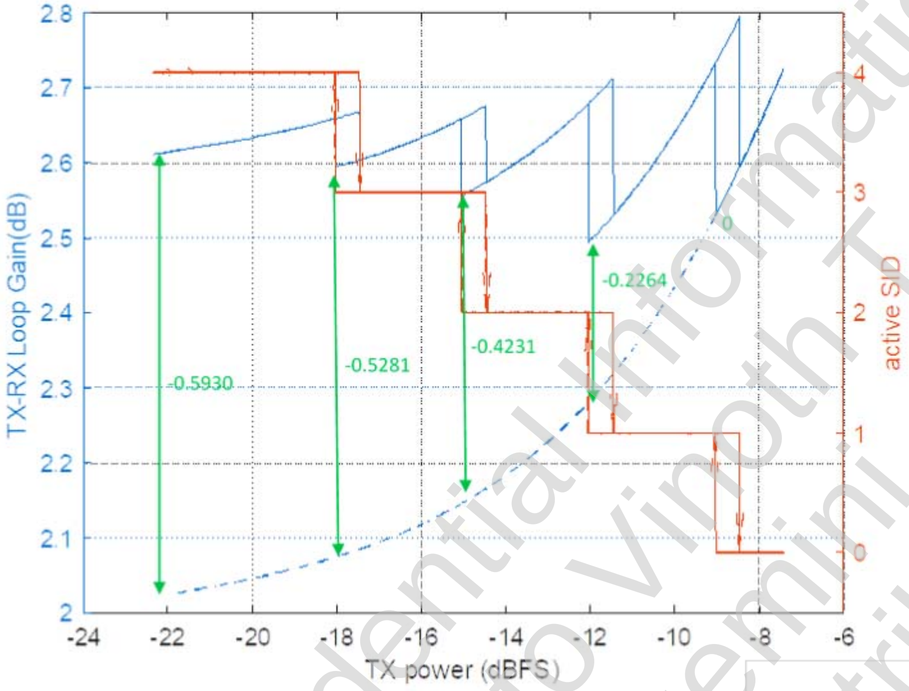
DPD Gain Tracking Options

Multi-Set Power (Table 7-30) controls (see DPD Parameter Sets) have been added to allow the selection of DPD average power gain tracking modes. These controls can select between a **Fixed DPD gain** mode or a **Fixed Loop gain** mode.

The Fixed DPD gain mode keeps the average power through the DPD filter fixed at defined levels. This is the mode used in DPD v10.0 and prior with all levels set to 0dB gain through the filter. When using this mode, the DPD feedback loop gain can be changed at any time without notifying the DPD core. Hence, this mode does not compensate for the average power change at different operating points of the Power Amplifier (PA). Multi-Set Power controls to select the gain level for each of the SID's utilized in the multi-set DCL mode are exposed along with the input power level used to switch between SID coefficient sets.

The example below shows the loop gain (TX power - RX power) as a function of input TX power along with the active SID. In this example the switching thresholds are set to -9, -12, -15 and -18 dBFS. There is 0.5 dB of hysteresis applied to the threshold, so when the power is increasing the actual switching levels will be -8.5, -11.5, -14.5, and -17.5 dBFS.

The dashed blue curve shows the response when all gain tables are set to 0 dB and the solid blue curve shows the response with unique gains for each SID.



	- - - - - - - - -	———
AVE_PWR_GAIN_0	0	0
AVE_PWR_GAIN_1	0	-0.2264
AVE_PWR_GAIN_2	0	-0.4231
AVE_PWR_GAIN_3	0	-0.5281
AVE_PWR_GAIN_4	0	-0.5930

Figure 7-10: **Fixed DPD Gain Mode**

The **Fixed Loop** gain mode uses the AVE_PWR_GAIN_X parameters for the reset coefficients state and when coefficients are inherited from one SID to another as a starting point. Once updates begin to occur for a given SID the DPD updates maintain a **Fixed Loop** gain by appropriately adjusting the average power gain for each SID. In this mode, any change in the gain of the feedback path must be conveyed to DPD to ensure proper tracking.

It is expected that utilizing the **Fixed DPD gain** mode, while initially configuring any attenuators after DPD, is useful since no information needs to be sent back to DPD. Then optionally, if the system loop gain remains quasi-constant, the DPD can be switched into the **Fixed Loop gain** mode. Switching between the operating modes can be achieved using the DCL command structure.

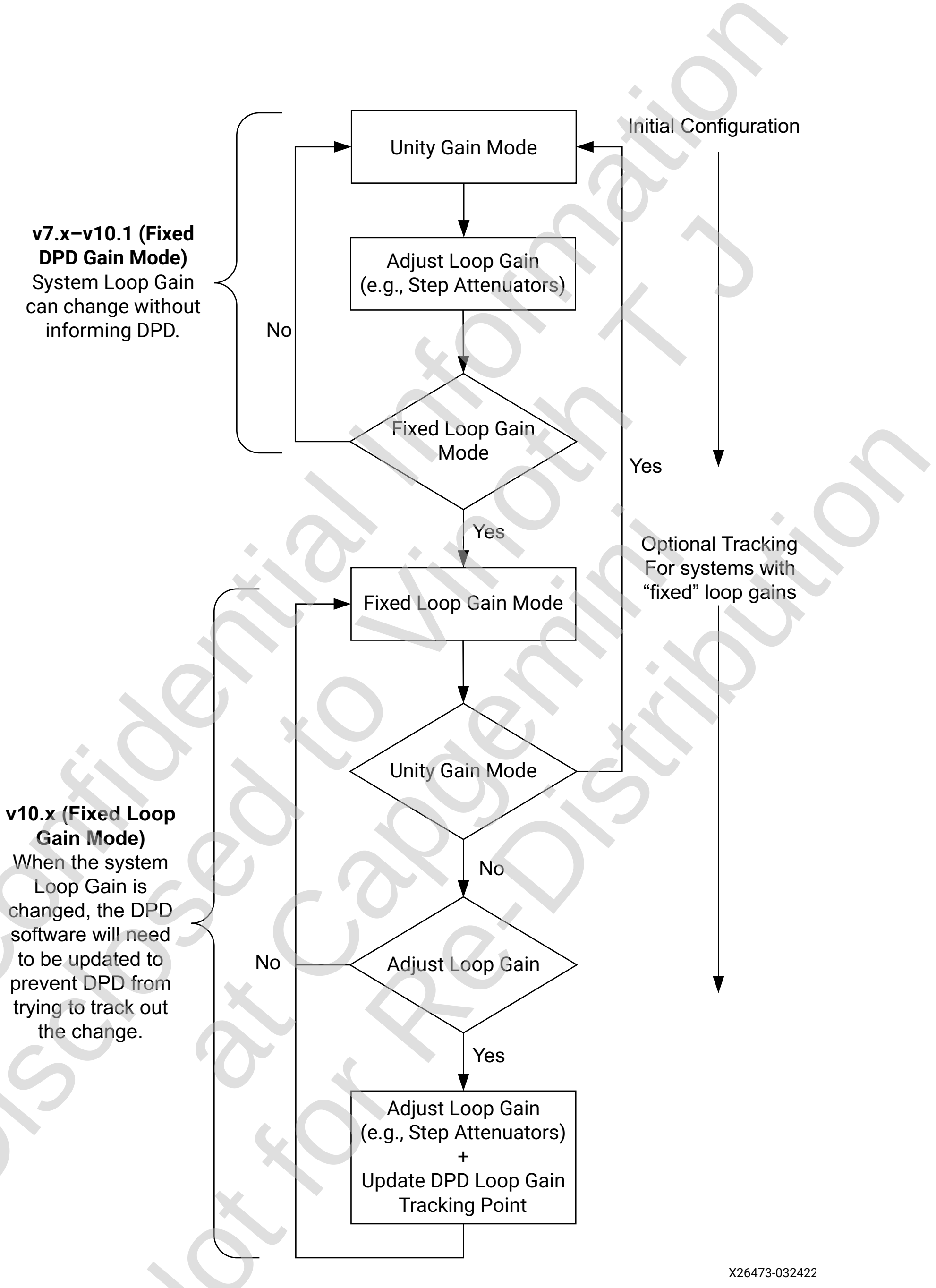

v7.x–v10.1 (Fixed DPD Gain Mode)
System Loop Gain can change without informing DPD.
Unity Gain Mode
Initial Configuration
Adjust Loop Gain (e.g., Step Attenuators)
No
Fixed Loop Gain Mode
Yes
Yes
Optional Tracking For systems with "fixed" loop gains
Fixed Loop Gain Mode
Unity Gain Mode
v10.x (Fixed Loop Gain Mode)
When the system Loop Gain is changed, the DPD software will need to be updated to prevent DPD from trying to track out the change.
No
No
Adjust Loop Gain
Yes
Adjust Loop Gain (e.g., Step Attenuators) + Update DPD Loop Gain Tracking Point
X26473-032422

Figure 7-11: **Suggested Usage**

Example for Adjusting Loop Gain During Fixed Loop Gain Mode

Occasionally, the loop gain of the system may need to be adjusted. During these updates the new loop gain value must be updated in DPD to prevent DPD from compensating for the change in loop gain. The following is the suggested process:

1. Read current LOOPGAIN from desired port in the DCL monitors.
2. Compute what the new expected LOOPGAIN will be after the loop gain is adjusted, that is, if a step attenuator is to be used to remove 0.25 dB of gain in the loop, then the expected new LOOPGAIN will be 0.25 dB higher than the current LOOPGAIN. So LOOPGAIN_NEW = round(LOOPGAIN*10^(+0.25/20)).
3. Write LOOPGAIN_NEW to PARAMETER_2 of the host interface.
4. Execute DCL_UPDATE_FIXED_LOOPGAIN command.
5. Adjust actual loop gain (e.g., a step attenuator).

Example

Below shows a simple example of an input signal that is switching between two power levels. The system is started in **Fixed DPD gain** mode, then switch into **Fixed Loop gain** mode and finally goes back to **Fixed DPD gain** mode. The traces are gathered using the snapshot monitor feature of DPD.

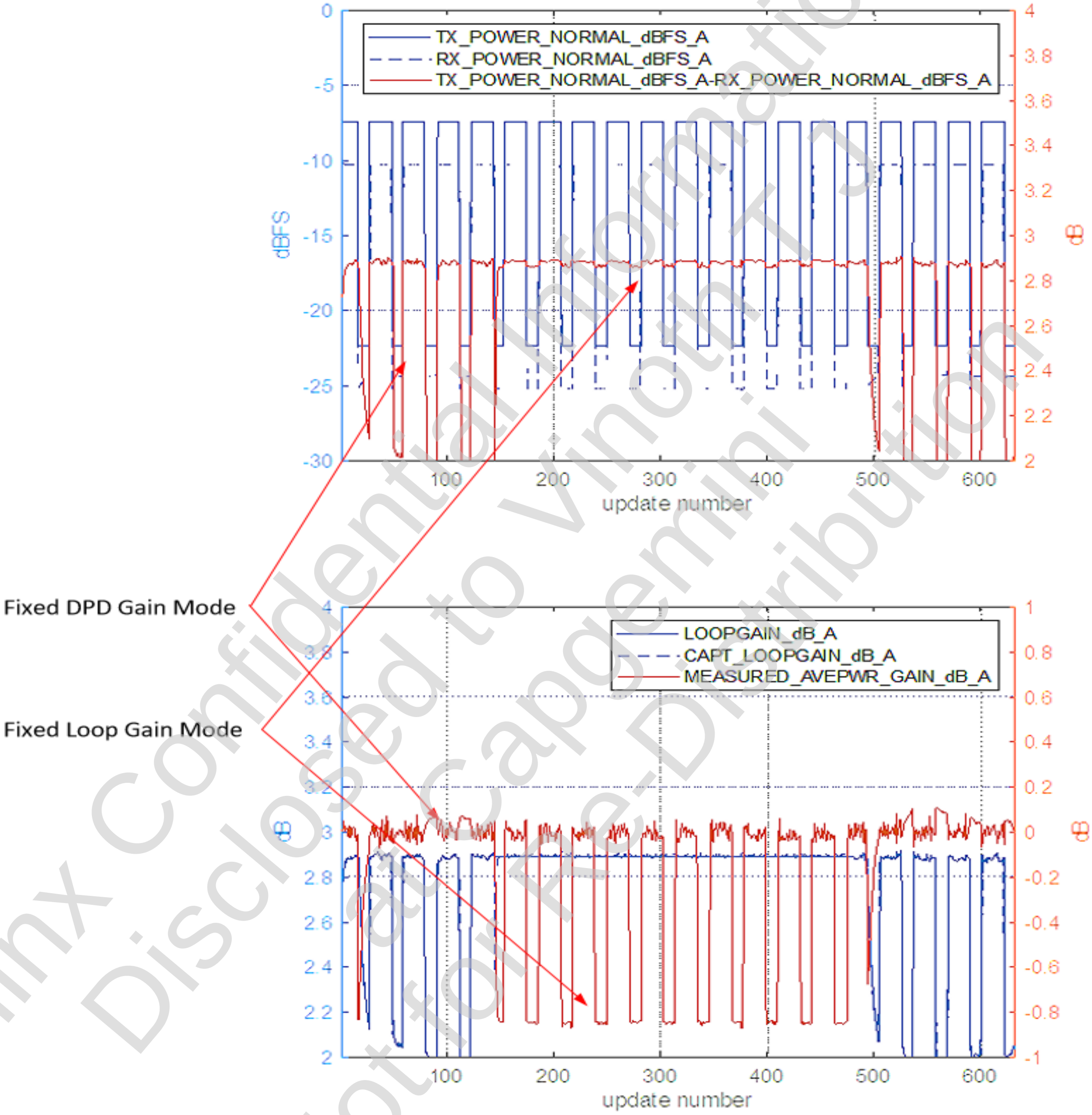


Figure 7-12: **Input Signal Example**

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 7-15. 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 7-15: **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 7-16 shows a full list of available control modes and Table 7-17 shows the general registers associated with control mode operation. The interface for all commands is updated to ensure consistent usage of portnum and portmap in the API. The API portnum is used for commands that can only be applied to one port at a time while portmap is used for commands that can be applied to multiple ports at the same time. The commands can be grouped into four categories as follows:

- Port agnostic commands
- Commands that require an SRX port change (uses portnum). The portnum value is written to the PORTNUM register prior to triggering the command, appropriate SRX port switch controls are required.
- Commands that do NOT require an SRX port change and can be applied to multiple ports (uses portmap). The portmap is written to PARAMETER_0 prior to triggering the command. If portmap is not used the PARAMETER_0 should be set to 0.

  When portmap is not used the UPDATE_*_PARAMETERS commands will be applied to all port, the others will be applied only to the active port.
- Commands that do NOT require an SRX port change and can only be applied to a single port (uses portnum). The portnum is written to PARAMETER_0 prior to triggering the command.

Table 7-16: **DPD Control Modes**

Value	Mnemonic	Description
	Port Agnostic Commands	
0	WAIT_STATE	NO-OP
1	READ_CONFIGURATION	Refresh the host interface REPOSITORY, BUILDDATE, BUILDTIME, BUILDSETTINGS and HWVERSION registers.
2	RESTORE_DEFAULTS	Restore the default configuration.
4	Reserved	
15	ENABLE_SNAPSHOT_MONITOR	Enable snapshot monitor feature (see Snapshots Monitors).
16	DUMP_SNAPSHOT_MONITOR	Read snapshot monitor pages (see Snapshots Monitors).
17	PAUSE_SNAPSHOT_MONITOR	Toggles the pause/start of an active snapshot monitor. (see Snapshots Monitors).
18	CONFIGURE_STRUCTURE	Reconfigure the NLF primitive for supported Filter Structures. Specify the desired filter structure (FS) by setting PARAMETER_0 to (FS-8) prior to sending this command. • FS10 builds may be configured as FS8, FS9, FS10 or FS11 • Non-FS10 builds may be configured as FS8 in addition to the build-time Filter Structure.
19	RUN_DCL	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	Directs the processor to exit the DCL control function.
21	RESET_DCL	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	Present the page, specified by PARAMETER_0, of the capture RAM data at the host interface RAM page transfer area.
24	GET_CURRENT_HISTOGRAM	Present 256 bins of the current transmit histogram for the active port at the host interface RAM page transfer area.
25	READ_CAPTURE_POWER_METERS	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). The integration period for the measurement is returned in 516(L) and the PORTNUM that was active during the capture is returned in 517. To convert to dBFS use: 10*log10((LSW + MSW*232)/230/L)
29	GET_CAPTURE_PSD	Compute power spectral density of the contents in a capture RAM. See GET_CAPTURE_PSD Function for more information.
59	UPDATE_DCL_MONITOR_VIEWS	Update the DCL monitor view terms with values read from the host interface register 512 to 543.
60	REPORT_DCL_MONITOR_VIEWS	Report the DCL monitor view terms in the host interface registers 512 to 543.

Table 7-16: **DPD Control Modes *(Cont'd)***

Value	Mnemonic	Description
0xDEADC0DE	KILL_DPD	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.
Commands that require a SRX port change (Uses portnum)		
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.
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.
26	GET_SWEPT_CAPTURE_POWER	DPD will perform 512 evenly distributed captures using CAPTUREMODE=1 with CAPTUREDELAY of between 0 and PARAMETER_0 samples. The captured DPD input power along with either the DPD output power or SRX power at each delay is reported in the page transfer area. See GET_SWEPT_CAPTURE_POWER Function for more information
30	CAPTURE_AND_ALIGN	Capture samples and run signal alignment routines.
36	MONITOR_CAPTURE_LOCATION	DPD performs 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	Capture samples and compute the AM/AM and AM/PM responses between DPD input and DPD FbRx Input.
38	COMPUTE_PA_AMAM_AMPM_RESPONSES	Capture samples and compute the AM/AM and AM/PM responses of the PA (i.e., DPD output v/s DPD Feedback Input).
Commands that do NOT require SRX port change, can be applied to multiple ports (uses portmap), if portmap is not supplied these commands will be applied to the active port.		
5	RESET_ALIGN_CALIBRATION	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.).

Table 7-16: **DPD Control Modes *(Cont'd)***

Value	Mnemonic	Description
8	RESET_COEFFICIENTS	Set coefficient sets to unity gain and update the datapath for pass-through. By default, all coefficients associated with the various SID are reset. To specify a single SID to reset, write the SID number to PARAMETER_0 register with format of 0xC0EF00XX (where XX is the desired SID/SET number). The value of XX can range from 0 to 34.
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. By default, the coefficient set with SID = 0 is loaded. To specify an SID to load, write the SID number to PARAMETER_0 register using the format 0xC0EF00XX (where XX is the desired SID/SET number). The value of XX can range from 0 to 34.
13	RESET_ERROR_HISTOGRAM	Reset the error histogram counter for the current port(s) as defined by the portmap.
28	CLEAR_C2L_OVERFLOW	Clear latched coefficient overflow warning.
31	QMC_RESET	Reset the internally stored QMC coefficients and set the QMC datapath block to pass-through.
32	QMC_OFF	Update the QMC datapath block for pass-through without changing the internally stored coefficients.
33	QMC_ON	Update the QMC datapath block from the internally stored coefficients.
39	DISABLE_OUTPUT	Disable the DPD filter output.
40	ENABLE_OUTPUT	Enable the DPD filter output after being disabled using the DISABLE_OUTPUT command.
42	CLEAR_DIAGNOSTICS	Clear DPD diagnostics for the current port.
43	UPDATE_ECF_PARAMETERS	Update the ECF parameters for specified port(s).
45	UPDATE_ARCH_PARAMETERS	Update the ARCH parameters for specified port(s).
47	UPDATE_CAPTURE_PARAMETERS	Update the CAP parameters for specified port(s).
49	UPDATE_DCL_PARAMETERS	Update the DCL parameters for specified port(s).
51	UPDATE_ODD_PARAMETERS	Update the ODD parameters for specified port(s).
53	UPDATE_MET_PARAMETERS	Update the MET parameters for specified port(s).[1]
55	UPDATE_TXQMC_PARAMETERS	Update the TXQMC parameters for specified port(s).
57	UPDATE_CAPWIN_PARAMETERS	Update the CAPWIN parameters for specified port(s).
61	UPDATE_MSP_PARAMETERS	Update the MSP parameters for specified port(s).
Commands that do NOT require a SRX port change and can only be applied to a single port (uses portnum)		
44	REPORT_ECF_PARAMETERS	Report the ECF parameters for a specific port.
46	REPORT_ARCH_PARAMETERS	Report the ARCH parameters for a specific port.
48	REPORT_CAPTURE_PARAMETERS	Report the CAP parameters for a specific port.

Table 7-16: **DPD Control Modes *(Cont'd)***

Value	Mnemonic	Description
50	REPORT_DCL_PARAMETERS	Report the DCL parameters for a specific port.
52	REPORT_ODD_PARAMETERS	Report the ODD parameters for a specific port.
54	REPORT_MET_PARAMETERS	Report the MET parameters for a specific port.
56	REPORT_TXQMC_PARAMETERS	Report the TXQMC parameters for a specific port.
58	REPORT_CAPWIN_PARAMETERS	Report the CAPWIN parameters for a specific port.
62	REPORT_MSP_PARAMETERS	Report MSP parameters for a specified port.
11	REPORT_COEFFICIENTS	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	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.
14	DUMP_ERROR_HISTOGRAM	Present the 512 bin error histogram at the host interface RAM page transfer area. Each bin value represents the corresponding error number, unknown conditions will be counted in bin 0 (e.g., R512 is bin 0, R636 (512+124) corresponds to error -124).
23	GET_HISTOGRAM	Present the 256 bins of the transmit histogram associated with the last capture attempt at the host interface RAM page transfer area. R512-R767 last capture histogram R768-R1023 long term histogram associated with last capture attempt.
27	RUN_ODD_AMAM	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.
34	REPORT_QMC_COEFFICIENTS	Copy the internal QMC coefficients to the host interface.
35	LOAD_QMC_COEFFICIENTS	Read QMC coefficients from the host interface to the internal QMC memory. This can be used to initialize the QMC performance.
41	REPORT_DIAGNOSTICS	Report requested DPD diagnostics to the host interface.

Notes:

1. METERLENGTH parameter is applied for all ports whereas NZCOUNTTHRESHOLD parameter can be applied for a specific port.

Table 7-17: **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.

Table 7-17: **DPD Host Interface Control Registers *(Cont'd)***

Address	Mnemonic	Description
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.
4	PARAMETER_0	Register used to pass a single parameter for various commands.
5	PARAMETER_1	Register used to pass a single parameter for various commands.
6	PARAMETER_2	Register used to pass a single parameter for various commands.
7	DCL_AUTO_EXIT_RANGE	Bits[31:16] - 16-bit signed status value Bits[15:0] - 16-bit signed status value The upper and lower 16 bits are used to define the range of status values (inclusive) that will cause the DCL to automatically exit. The limits may be specified as lower-value:higher-value or higher-value:lower-value.
8	DCL_AUTO_EXIT_PORTS	Specify the desired ports to monitor when DCL_AUTO_EXIT is enabled. Bits[31:16] used to specify port specific or all port monitoring. Bits[15:0] used to specify which ports to monitor. If Bits[31:16] = 0xDBAE then each of the lower bits [7:0] are used as a bit mask to indicate which ports to monitor. Bit 0 corresponds to port 0. Bits with value of 1 are monitored, bits with value of 0 are ignored. When bits [31:16] != 0xDBAE the lower bits are ignored and all ports will be monitored for DCL_AUTO_EXIT feature.
9-15		Reserved
16	COMMANDSTATUS	See Table 7-19.
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- 29		Reserved for future expansion
30	STATUS_ALT_RESULT0	Alternate status result
31	STATUS_ALT_RESULT1	Alternate status result

DPD Application Software Boot Process

The DPD application software ELF must be included in the boot image of the 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.

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 7-13 shows the DPD software boot sequence.

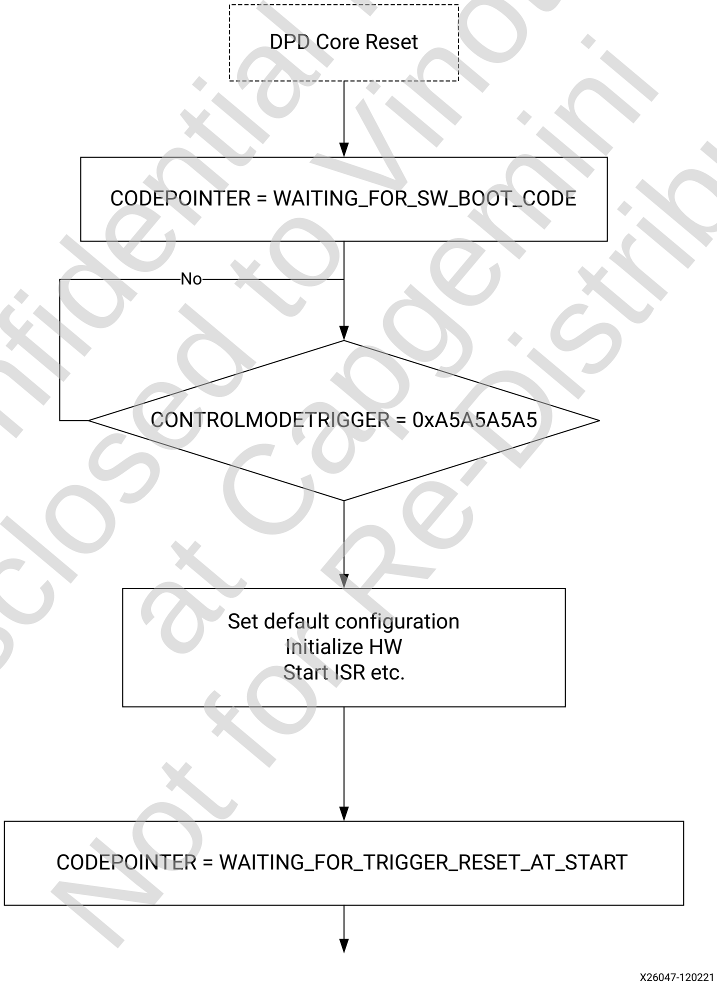

DPD Core Reset
CODEPOINTER = WAITING_FOR_SW_BOOT_CODE
No
CONTROLMODETRIGGER = 0xA5A5A5A5
Set default configuration
Initialize HW
Start ISR etc.
CODEPOINTER = WAITING_FOR_TRIGGER_RESET_AT_START
X26047-120221

Figure 7-13: **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 7-18. When operating with the debug interface (See *Xilinx LogiCORE IP Digital Pre-Distortion Debug Interface User Guide v13.1* (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 7-18: **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.
124	WAITING_FOR_PORT_SWITCH_TIMEOUT	Waiting for port switch acknowledgment has timed out.
125	WAITING_FOR_SW_BOOT_CODE	Waiting to detect 0xA5A5A5A5 code in CONTROLMODETRIGGER register at beginning of main() function indicating that it is okay to proceed with boot up.
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 boot up.
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.
138	ADVANCED_ALIGNMENT_ROUTINE	The software is currently in the advanced alignment routine which may require an extended timeout (up to 10 seconds).
139	PORT_RECONFIGURATION	The software is currently reconfiguring for the new port parameters.
140	PERFORMING_CAPTURE_PROCESS	The capture process is running. During this time the SRX path should be dedicated to the requested feedback path.

The sequence of events is as follows (see Figure 7-14):

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.

   Note: For all commands, PARAMETER_0 and PARAMETER_1 should be set to 0 when not used.
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. Read COMMANDSTATUS.
9. Reset the trigger (write 0 to CONTROLMODETRIGGER).
10. Wait for trigger acknowledge to reset (TRIGGERACK goes to zero).

Note: Events 2, 3, and 4 can occur at any time and in any sequence before the trigger is applied in event 5. The CONTROLMODEREGISTER, PORTNUM and any optional parameters will be read by the DPD software during event 5.

The following figure shows the command shell for the DPD core processes.

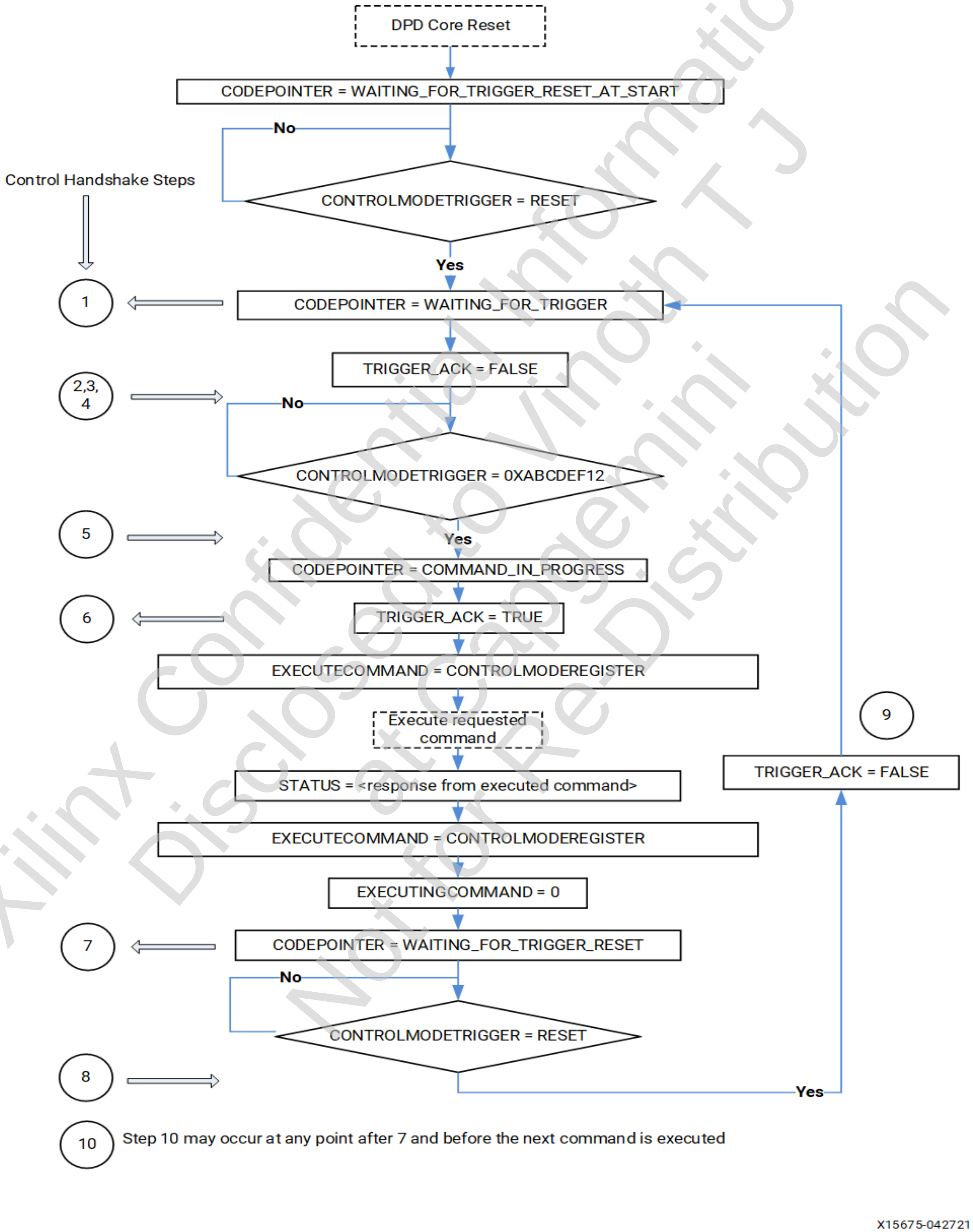

DPD Core Reset
CODEPOINTER = WAITING_FOR_TRIGGER_RESET_AT_START
No
Control Handshake Steps
CONTROLMODETRIGGER = RESET
Yes
1
CODEPOINTER = WAITING_FOR_TRIGGER
2,3, 4
TRIGGER_ACK = FALSE
No
CONTROLMODETRIGGER = 0XABCDEF12
5
Yes
CODEPOINTER = COMMAND_IN_PROGRESS
6
TRIGGER_ACK = TRUE
EXECUTECOMMAND = CONTROLMODEREGISTER
Execute requested command
9
STATUS = <response from executed command>
TRIGGER_ACK = FALSE
EXECUTECOMMAND = CONTROLMODEREGISTER
EXECUTINGCOMMAND = 0
7
CODEPOINTER = WAITING_FOR_TRIGGER_RESET
No
CONTROLMODETRIGGER = RESET
8
Yes
10
Step 10 may occur at any point after 7 and before the next command is executed
X15675-042721

Figure 7-14: **DPD Command Shell**

Except DCL modes, modes terminate and return a status value in `COMMANDSTATUS`. Possible returned values are shown in Table 7-19 and defined in `xdpd_host_interface.h` of the API source code.

Table 7-19: **Status Values**

Value	Mnemonic	Description
-911	PANIC_RESET_DETECTED	ERROR: A dpd_aresetn reset pin assertion has been detected.
-910	SOFTWARE_HARDWARE_MISMATCH	ERROR: The DPD major SW version number does not match the major HW version number.
-511	EVAL_LICENSE_TIMEOUT	ERROR: The evaluation license has timed out.
-451 to -450	NLF_ERROR_0 to NLF_ERROR_1	ERROR: Please contact Xilinx support. This error is associated with the operation of the NLF engine and indicates a fundamental problem with the build.
-443 to -400	HWA_ERROR_0 to HWA_ERROR_43	ERROR: Please contact Xilinx support. This error is associated with the operation of the HWA engine and indicates a fundamental problem with the build.
-304	DATAPATH_INTERNAL_OVERFLOW_PREVENTION	ERROR: Update blocked because the new coefficients could cause the datapath to overflow.
-303	FAILURE_MAX_MAG	ERROR: Update blocked because max([tx]) > 1.
-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.

Table 7-19: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-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_MULTI_PHASE	ERROR: Failed to successfully update the MET parameter set because of invalid METERLENGTH parameter (must be a multiple of the number of NUM_PHASES).
-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).
-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.

Table 7-19: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-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.
-229	CONFIG_FAILURE_QMCTXENABLE	ERROR: Failed to successfully update the QMC parameter set because of invalid QMCTXENABLE parameter.
-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.

Table 7-19: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-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.
-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).
-214	CONFIG_FAILURE_LTETDDCONFIGURATION	ERROR: Failed to successfully update the CAP parameter set because of invalid LTETDDCONFIGURATION parameter.
-213	CONFIG_FAILURE_NZCOUNTSEL	ERROR: Failed to successfully update the MET parameter set because of invalid NZCOUNTSEL parameter.
-212	CONFIG_FAILURE_SNAPSHOT_MODE	ERROR: Failed to successfully update the snapshot mode.
-211	CONFIG_FAILURE_LOAD_COEFFICIENTS	ERROR: Failed to successfully load new coefficients from host interface.
-210	CONFIG_FAILURE_DCL_MONITOR	ERROR: Failed to successfully update the DCL monitor view.
-208	CONFIG_FAILURE_ENABLEBLINDRXQMC	ERROR: Failed to successfully update the ECF parameter set because of invalid ENABLEBLINDRXQMC parameter.
-207	CONFIG_FAILURE_MULTI_SET_POWERS	ERROR: Failed to successfully update the MSP parameter set: • FIXED_GAIN_MODE > 1 • DAMPINGVALUE > 1.0 • SWITCH_0_1 < -15 dBFS • SWITCH_X_Y threshold are not monotonic or there is less than 2dB between adjacent switch thresholds • |AVE_PWR_GAIN_X| is > 3 dB
-205	CONFIG_FAILURE_LTM_TC	ERROR: Failed to successfully update the ARCH parameter set because of invalid LTM_TC parameter(s).

Table 7-19: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-204	CONFIG_FAILURE_DCLALGORITHM4_FSI	ERROR: Failed to successfully update the DCL parameter set because selected DCLALGORITHM parameter requires non-zero FRAME_SYNC_INTERVAL. Verify the frame marker connection in s_axis_din_tuser.
-203	CONFIG_FAILURE_LTM_TC_FSI	ERROR: LTM_TC_0 or LTM_TC_1 parameter could not be set because the FRAME_SYNC_INTERVAL is reporting zero value. Verify the frame marker connection in s_axis_din_tuser.
-202	CONFIG_FAILURE_INVALID_PAGE_NUMBER	ERROR: Requested page number of out of range for requested command.
-128	ALIGNMENT_LOOPDELAY_CHANGE_DETECTED	ERROR: Signal alignment detected a change in loop delay. Signal alignment will be re-calibrated on the next update attempt.
-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.
-117	ALIGNMENT_FAILED_FRACTIONAL_DELAY	ERROR: Indicates the detected fractional delay was beyond 1/2 of a sample.
-116	DPD_PHASE_ROTATION_ERROR	The detected average rotation of the signal cause by the DPD filter is beyond the default limit. Please contact Xilinx support.

Table 7-19: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-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. Can occur when using CAPTUREMODE = 1 or DCLALGORITHM = 4 with no capture sync connected on s_axis_din_tuser.
-111	HISTOGRAM_FAILURE	ERROR: Failed to detected the histogram complete signal
-109	SRX_ZERO_CAPTURES	ERROR: Zeros were captured in the RX capture buffer.
-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.
-96	CONSTANT_RX_LIMIT	ERROR: Alignment detected a section of constant values in the RX capture. The current update will stop as this may indicate that a bad capture has occurred.
-95	NORMALIZED_ERROR_POWER_FAILURE	ERROR: Power in the error signal is too high compared to TX capture power. The current update will stop as this may indicate that a bad capture has occurred.
-93	SRX_CONTROL_WAIT_FOR_ISR_ENABLE	ERROR: Waiting too long for interrupt routine to complete.

Table 7-19: **Status Values *(Cont'd)***

Value	Mnemonic	Description
-92	ALIGNMENT_PREMSE_MSE_CHECK_FAILURE	ERROR: PREMSE vs MSE check failed.
-91	LTM_CAPTURE_FAILURE	ERROR: Capture associated with Long Term Memory updates failed.
-90	DATAPATH_RESET_TIMEOUT	ERROR: Datapath took too long to come out of reset.
-63	DCL_MULTISET_STATE_MISMATCH	ERROR: State mismatch between channel and capture with multi-set DCL. Infrequency occurrences of this message while operating under dynamic signal conditions can be safely ignored.
-62	STATE_UNCERTAINTY_AT_CAPT_START	ERROR: Detected channel state change during capture. Infrequency occurrences of this message while operating under dynamic signal conditions can be safely ignored.
-61	STATE_MULTISET_CAPT_MISMATCH	ERROR: The captured state was not consistent during capture process. Infrequency occurrences of this message while operating under dynamic signal conditions can be safely ignored.
-60	STATE_CAPTURE_START_MISMATCH	ERROR: The capture process failed to find matching SID's in allocated time. Infrequency occurrences of this message while operating under dynamic signal conditions can be safely ignored.
-6	INVALID_DIAGNOSTIC_REQUEST	ERROR: Requested diagnostic value was invalid.
-5	INVALID_DCLALGORITHM_SETTING	ERROR: The attempted command can only be executed when DCLALGORITHM is set to 4.
-4	INVALID_NLF_COMMAND	A command associated with the DFE NL FIR primitive was attempted when the DFE NL FIR primitive is not used in the design or when an invalid structure is requested while reconfiguring a structure 10 build.
-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.
1	IN_PROGRESS	Indicates the processor is currently executing a command.
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 unstable.

Table 7-19: **Status Values *(Cont'd)***

Value	Mnemonic	Description
13	SUCCESSFUL_NOT_LOADED_DUE_TO_STATE_CHAN GE	The current update is successful but the new coefficients were not loaded into the filter because the signal state changed before the update was completed.
14	SUCCESSFUL_WITH_LATCHED_OVERFLOW	The current update is successful. However, there is a latched overflow present. The latched overflows can be cleared using `CLEAR_C2L_OVERFLOW` or `DCL_CLEAR_C2L_OVERFLOW`
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, tracking is still enabled.
35	PORT_TRACK_SKIPPED	user has requested to skip updating and tracking this port during DCL routines
36	PORT_REQUEST_NORMAL	The DCL routine is currently requesting a forward port.
37	PORT_REQUEST_REFLECTED	The DCL routine is currently requesting a reflected port.
225	DPD_PHASE_ROTATION_WARNING	The detected average rotation of the signal cause by the DPD filter is beyond the default limit. Random occurrences of this warning can be ignored.
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.

The error codes are classified to help group the errors as defined in Figure 7-20

Table 7-20: **Error Code Classification**

Error Code Classification	Description	Suggested Action
15	Potential low level Hardware Issues	Stop DCL, Reset Coefficient immediately
14	DPD performance issue, possible PA damage	Reset Coefficients and Alignment immediately
9	DPD performance issue, low risk to PA damage	Reset Coefficients and Alignment if SUCCESSFUL_COUNTER/COUNTER ratio is low. (see Bits[31:24] in UPDATE_MONITOR_0)
8	DPD performance issue, generally around signal dynamics	Reset Coefficients and Alignment if SUCCESSFUL_COUNTER/COUNTER ratio is low. (see Bits[31:24] in UPDATE_MONITOR_0)
2	Parameter configuration issues	Correct parameters
1	General commanding issues	Correct command
0	Unclassified / Undocumented	N/A

The error codes and associated classification are defined in the c-API file `xdpd_host_interface.h`.

Updating and Reporting DPD Parameters

The parameters that control the DPD core are detailed in tables Table 7-21 to Table 7-28. 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. Unless specified, 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.

Unless otherwise specified, parameters can be set with port specific values. To facilitate this potentially large increase in unique parameters, the DPD IP 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 (portmap) 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 the ports that need to be updated (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 to the base address of the Parameters region of the host interface (see Table 7-14 DPD Host Interface Memory Map Outline).

- **Architecture (ARCH)**:- The ARCH set (see Table 7-21) contains parameters associated with defining the structure of the DPD filter.
- **Estimation Core Function (ECF)**: The ECF set (see Table 7-22) 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 7-23) is used to define the signal capture method to be utilized.
- **Dynamic Control Layer (DCL)**: The DCL set (see Table 7-24) 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 7-25) contains parameters associated with the optional QMC updates.
- **Over-Drive Detection (ODD)**: The ODD set (see Table 7-26) contains parameters associated with power amplifier protection.
- **Meter Length (MET)**: The MET set (see Table 7-27) is used to set the measurement interval for the power meters and histograms.

- **Capture Windows (CAPWIN)**: The CAPWIN set (see Table 7-28) is used to define the optional valid capture windows.
- **Multi-Set Power (MSP)**: The MSP set (see Table 7-29) is used to define the gain tracking options of DPD.

Table 7-21: **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 = -6 dB or 2 = -12 dB, 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
2	LTM_TC_1[(2)]	0 to 65535	0	First time constant for Long Term Memory solution in µs. A value of 0 will remove these terms from the architecture, must be 0 for non-LTM builds. Time Constants must be programmed after the FRAME_SYNC_INTERVAL is reporting accurate non-zero values.
3	LTM_TC_2[(2)]	0 to 65535	0	Second time constant for Long Term Memory solutions in µs. A value of 0 will remove these terms from the architecture, must be 0 for non-LTM builds. Time Constants must be programmed after the FRAME_SYNC_INTERVAL is reporting accurate non-zero values.

Table 7-21: **Register Set Using UPDATE_ARCH_PARAMETERS and REPORT_ARCH_PARAMETERS *(Cont'd)***

Parameters Base Offset	Mnemonic	Values	Default	Notes
4	RESERVED	0	0	Reserved for future enhancements, must be set to 0.

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 (up to 15) 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. It is desirable to have smallest B value to keep the number of coefficients necessary for desired performance. This value helps to improve the computation time.

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

2. An alternate programming method for the LTM_TC_X value is available for situations where these parameters need to be programmed before the valid FRAME_SYNC_INTERVAL is available.

   LTM_TC_X = 0xXXYYZZZZ;
   The XX is an "enable" control and YY is used to specify the clock rate as a multiple of 30.72 MHz. ZZZZ is the time constant as normally defined. Both `LTM_TC_1` and `LTM_TC_2` must have the same XXYY values to enable this programming. To enable, set XX = A5.

   Example: To specify the time constants of 1000 and 100 with a 491.52MHz clock the parameters would be set as:

   LTM_TC_1 = 0xA51003E8;

   LTM_TC_2 = 0xA5100064;

   This will use 491.52MHz as the sample rate (30.72*16 = 491.52) and disable the checking that the frame sync is operating.

   * There will be no error checking that the YY eventually corresponds to the correct frame sync.

   * Read back of the correct LTM_TC_X values is not available until the frame sync is working.

   * The read back of the LTM_TC_X values will not contain the XXYY prefix.

Table 7-22: **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 - Receiver is supplying real IF data 1 - 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	S32.0	0x40000000	(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). Reducing this value can be a useful trade-off to improve close-in ACLR performance versus wideband performance. It is very useful when Occupied Bandwidth (OBW) is much smaller than the sampling rate.

Table 7-22: **Register Set Using UPDATE_ECF_PARAMETERS and REPORT_ECF_PARAMETERS *(Cont'd)***

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	$1000 \geq \Delta \geqslant 4$	1000	Maximum integer sample delay to search over during initial delay adjustment. Δ = (MAXDELAY - MINDELAY).
11	MINDELAY	0, 1...1000	0	Minimum integer sample delay to search over during initial delay adjustment. Δ = (MAXDELAY - MINDELAY).
12	ENABLEBLINDRXQMC	0 or 1	0	Initial stage of blind RX QMC correction. This can be changed to 0 for systems with real ADC sampling or when the RX QMC is handled outside of DPD.

Table 7-23: **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
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 ms. When set to 0, the window uses intervals between 1 and 2 METERLENGTH (default).

Table 7-23: **Register Set Using UPDATE_CAPTURE_PARAMETERS and REPORT_CAPTURE_PARAMETERS**

Parameters Base Offset	Mnemonic	Values	Default	Notes
3	HIRESMONITOR	0 or 1	0	Enable monitoring of hi-res bit.
4	LTETDDCONFIGURATION	1 to 1023 or 0x800003FF	x800003FF	LTE TDD frame configuration mask. A 10-bit value with each bit representing either Downlink/Special(1) or Uplink(0) subframe. The MSB represents subframe zero. The decimal values for standard configurations are: • 0 - DSUUUDSUUU -> 792 • 1 - DSUUDDSUUD -> 825 • 2 - DSUDDDSUDD -> 891 • 3 - DSUUUDDDDD -> 799 • 4 - DSUUDDDDDD -> 831 • 5 - DSUDDDDDDD -> 895 • 6 - DSUUUDSUUD -> 793 • FDD -> 1023 This parameter is primarily used to set the DCL power meter mode to FDD or TDD and sets the default CAPWIN parameters. These parameters are used to guide the capturing process, but not required for basic operation. When Bits[0:9] are set to 1023 the DCL power meters will use METERLENGTH as the divisor when computing average power for FDD mode. Otherwise TXPOWERCOUNT_X is used as the divisor (where X is the port) for TDD mode. Setting Bit[31] to 1 can be used to select TXPOWERCOUNT_X as the divisor when Bits[0:9] are set to 1023. Bit[31] is ignored when Bits[0:9] are not equal to 1023.

Table 7-24: **Register Set Using UPDATE_DCL_PARAMETERS and REPORT_DCL_PARAMETERS**

Parameters Base Offset	Mnemonic	Values	Default	Notes
0	DCLALGORITHM[1][2]	0, 1, or 4	1 for builds without LTM hardware 4 for builds with LTM hardware	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. 4 - Multi-set using RUN_UPDATE_COEFFICIENTS _V3. This operating mode requires connection of s_axis_din_tuser capture sync signal (connect to the frame marker signal) and can only be used in systems that support the following enforced parameter settings: • TXRXRATIO = 1 • RXINPUTFORMAT = 1 • RXPHASESTEP = 0 • ENABLEBLINDRXQMC = 0 • SAMPLES2PROCESS = 16000 (when LTM is used) Systems utilizing LTM require 16k capture buffers.
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 10 ms. Non-Zero - User defined delay in microseconds.
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.

Table 7-24: **Register Set Using UPDATE_DCL_PARAMETERS and REPORT_DCL_PARAMETERS *(Cont'd)***

Parameters Base Offset	Mnemonic	Values	Default	Notes
5	ENABLEVSWRPWRMSR	0 to 1024	0	Enable VSWR power measurements. For non-zero values a VSWR power reading will be requested when (COUNTER % ENABLEVSWRPWRMSR) == 0.
6	MINUPDATETIME	0 to 1000	0	Set the minimum amount of time that should be spent on each antenna in ms. 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	RESERVED			
8	DCLTXLOWPOWER	U32.30	107264	Low TX power threshold to stop attempting DPD coefficient updates. The threshold in dBFS is 10*log10(threshold). ***Note:*** Low power thresholds below the defaults must be validated in your system.
9	DCLRXLOWPOWER	U32.30	10752	Low RX power threshold to stop attempting DPD coefficient updates. The threshold in dBFS is 10*log10(threshold). ***Note:*** Low power thresholds below the defaults must be validated in your system.
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.
11	RESERVED	0	0	Reserved for future enhancement, must be set to 0.

Notes:

1. These parameters cannot be set uniquely for each port. All ports will take the last value set.
2. The CAPTUREMODE parameter is ignored for DCLALGORITHMS 1,3 and 4.

Table 7-25: **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	QMCNUMESTAVER AGE	1 - 256	1	QMC updates to average prior to updating the hardware registers (after initial convergence).
2	QMCGAINMU	U32.24	0.1	Scaling term applied to the gain error prior to updating the register (after initial convergence).
3	QMCPHASEMU	U32.24	0.1	Scaling term applied to the phase error prior to updating the register (after initial convergence).
4	QMCOFFSETMU	U32.24	0.1	Scaling term applied to the offset errors prior to updating (after initial convergence).

Table 7-26: **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, 2, or 3	3	0 - Disable protection 1 - Enable peak saturation referenced to 0 dBFS at the output 2 - Enable predictive ODD protection 3 - Enable both modes 1 and 2
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 7-27: **Register Set Using UPDATE_MET_PARAMETERS and REPORT_MET_PARAMETERS**

Parameters Base Offset	Mnemonic	Values	Default	Notes
0	METERLENGTH[1]	2^{18} to 2^{32}-1	2457600	Number of samples for measurements block processing. The following guidelines should adhered to: • Must be a multiple of 2 for two-phase or two-phase hybrid implementations. • Should be a multiple of any frame size and ≥ 10 ms. ***Note:*** 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 round(fs*(N*0.01 + 0.0005)) to ensure best performance. • Must be set sufficiently long to cover large variation in the peak signal power. For example an LTE system with very low user traffic will experience large bursts of power every 20 ms from the SIB1/MIB. In this case the METERLENGTH should be set > 20 ms to ensure stable performance in low power conditions.
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})
2	NZCOUNTSEL	0, 1, or 2	1	Non-zero count selection • 0 - Count samples with magnitudes > NZCOUNTTHRESHOLD • 1 - Count samples with both I and Q samples equal to zero • 2 - Count samples using UL/DL marker form TUSER

Notes:

1. METERLENGTH cannot be set uniquely for each port. All ports take the last value set.

Table 7-28: **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.

Table 7-28: **Register Set Using UPDATE_CAPWIN_PARAMETERS and REPORT_CAPWIN_PARAMETERS**

Parameters Base Offset	Mnemonic	Values	Default	Notes
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.
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 3-4: **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.
3. The default windows are set when LTECONFIGURATION is programmed. To over-ride the defaults send the command to update these after the LTECONFIGURATION command.

Table 7-29: **Register Set Using UPDATE_MSP_PARAMETERS and REPORT_MSP_PARAMETERS**

Parameters Base Offset	Mnemonic	Values	Default	Notes
0	FIXED_GAIN_MODE	0 or 1	0	Fixed Gain mode: • 0 - Keep the average power through DPD at the levels defined by AVE_PWR_GAIN_X parameters • 1 - Keep the average power of the DPD feedback loop gain constant
1	DAMPINGVALUE	0 to 1 (U32.31)	0.1	Tuning parameter for the DPD filter average power gain control.

Table 7-29: **Register Set Using UPDATE_MSP_PARAMETERS and REPORT_MSP_PARAMETERS**

Parameters Base Offset	Mnemonic	Values	Default	Notes
2	FIXED_GAIN_VALUE	U32.24	0	Fixed loop gain value to use when FIXED_GAIN_MODE is set to 1, convert to dB using 20*log10(X). When this parameter is set to 0, the LOOPGAIN determined during alignment calibration will be used.
3	AVE_PWR_GAIN_0	U32.30	1	Average power gain for maximum power coefficients, convert to dB using 20*log10(X).
4	SWITCH_0_1	U32.30	0.1	TX input average power switching threshold between max. power coefficients and second power coefficients, convert to dBFS using 10*log10(X).
5	AVE_PWR_GAIN_1	U32.30	1	Average power gain for second power coefficients, convert to dB using 20*log10(X).
6	SWITCH_1_2	U32.30	0.05	TX input average power switching threshold between second and third power coefficients, convert to dBFS using 10*log10(X).
7	AVE_PWR_GAIN_2	U32.30	1	Average power gain for third power coefficients, convert to dB using 20*log10(X).
8	SWITCH_2_3	U32.30	0.025	TX input average power switching threshold between third and fourth power coefficients, convert to dBFS using 10*log10(X).
9	AVE_PWR_GAIN_3	U32.30	1	Average power gain for fourth power coefficients, convert to dB using 20*log10(X).
10	SWITCH_3_4	U32.30	0.0125	TX input average power switching threshold between fourth power coefficients and min. power coefficients, convert to dBFS using 10*log10(X).
11	AVE_PWR_GAIN_4	U32.30	1	Average power gain for minimum power coefficients, convert to dB using 20*log10(X).

Monitors

Information available from the host interface RAM is detailed in Table 7-30.

Table 7-30: **Monitors**

Address	Mnemonic	Description
32	REPOSITORY	Software build Change List (CL) number. See Xilinx Answer 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 ***Note:*** Month is computed as a zero-based value (i.e., 0 - January).
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 = $2^{(\text{Bits}[15:12] + 9)}$ • Bits[19:16] - Acceleration level • Bits[22-20] - Reserved • Bit[23] - has_qmc • Bit[24] - Long Term Memory • Bits[29:25] - Filter Structure
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 - Reserved • 2 - Reserved • 3 - Zynq UltraScale+ (also for RFSoC) • 4 - Reserved
60	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.
61	RUNTIMEMSW	MSW of the time monitor.
62	METERLENGTH	Reports the current METERLENGTH value programmed into the hardware.
63	SRXQDC	Accumulation of the Q channel over the METERLENGTH interval (S32.0 value). Average D.C. in lsb's is (16*SRXQDC/METERLENGTH).

Table 7-30: **Monitors *(Cont'd)***

Address	Mnemonic	Description
64	SRXIDC	Accumulation of the I channel over the METERLENGTH interval (S32.0 value). Average D.C. in lsb's is (16*SRXQDC/METERLENGTH).
65	SRXAVE	Average SRX power: • Bits[31:28] - Active port value • Bits[27:0] - Average SRX power value (U32.30 convert to dBFS using 10*log10(x))
66	DPDOUTPUT_PAPR	DPD output peak to power ratio: • Bits[31:28] - Active SRX port value • Bits[27:0] - DPD output Peak to Average Power Ratio (U28.16 in dB)
67	DPDOUTPUTAVE	Average DPD output power value • Bits[31:28] - Active port value • Bits[27:0] - Average DPD output power value (U32.30 convert to dBFS using 10*log10(x))
68 + 5*n	TXPOWERAVE	DPD input average power of port n (U32.30 convert to dBFS using 10*log10(x))
69 + 5*n	TXPAPR	DPD input Peak to Power Ratio of port n (U32.16 in dB)
70 + 5*n	TXPOWERCOUNT	Number of non-zero samples in the power measurement of port n (where n ranges from 0 to number of ports minus one).
71 + 5*n	FCM_REG	Proprietary frequency content metrics (FCM) of port n (where n ranges from 0 to number of ports minus one).
72 + 5*n	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[29:18] - Signal identifier associated with latched overflow. • Bit[30] - Datapath overflow warning indicator. • Bit[31] - Datapath overflow indicator.

All TX ports have their own dedicated power meters at the input of the filter.

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 7-15.

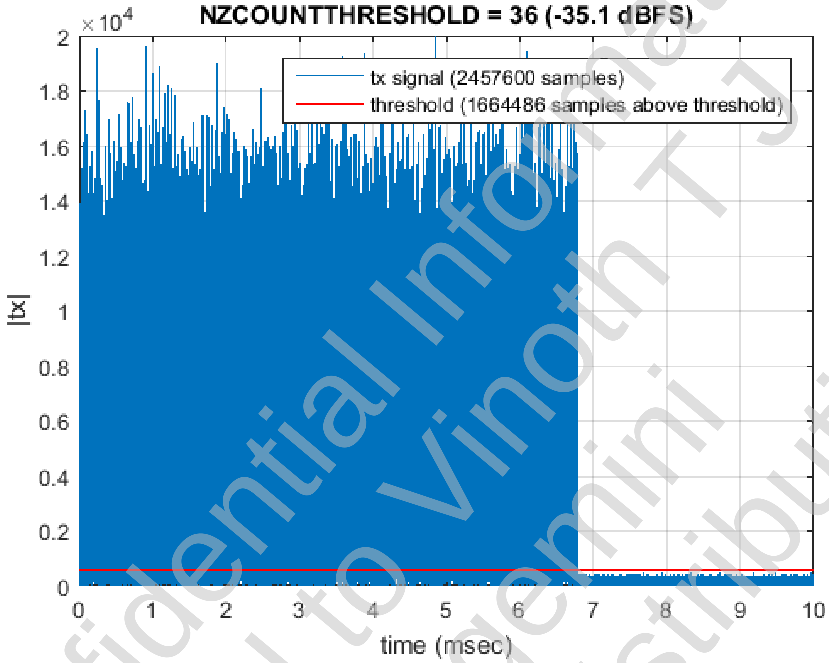


Figure 7-15: **XPOWERCOUNT Example**

The SRX power meter is shared between all ports (for normal and VSWR power measurements). This register has the active SRX port number embedded in the upper nibble of the register. This nibble can be used to determine which port the SRX power is reporting.

The SRX power reported in the SRXAVE register is an unbiased power measurement. The D.C. bias, that is removed in the reported SRX power, is reported in the SRXIDC and SRXQDC registers.

The DPD output power meter is shared between all ports. This register has the active port number embedded in the upper nibble of the register. This nibble can be used to determine which port the DPD output power is reporting.

Note: The METERLENGTH power meters decimate the input signal by a factor of two (simple dropping of every other sample).

Reading the SRX and DPDOUTPUT 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 SRX and DPDOUTPUT registers will report 0xE000000 to indicate unknown power reading. This value is not reported for more than one measurement interval.

In non-DCL operation the normal or VSWR power can be measured and reported in the SRXAVE register. 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 and DPDOUTPUT 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 0xF000000. 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 SRXAVE 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 7-33 for DCL Monitors.

Snapshots Monitors

The DPD snapshot monitor has the capability to log various diagnostics to an internal snapshot buffer that can be read for external analysis at a later time. The snapshot buffer can operate by taking a single snapshot of the selected diagnostic (fill the buffer once) or it can be configured to operate continuously using a cyclic buffer approach.

Snapshot Enabling

The snapshot monitor can be enabled whether or not the DCL is running as follows:

1. If DCL is enabled, use the `DCL_ENABLE_SNAPSHOT_MONITOR` to select the desired monitor and ports to enable. Use PARAMETER_1 to select which ports to monitor and DCL command parameter XY to specify the snapshot mode.
2. When DCL is disabled the `ENABLE_SNAPSHOT_MONITOR` command is used to enable the snapshot monitor.

   Write the desired port enable bit mask and snapshot mode to the PARAMETER_0 register.

   - Bits[7:0] snapshot mode
   - Bits[15:8] port monitor bit mask
3. Run the `ENABLE_SNAPSHOT_MONITOR` command. The supported snapshot modes are
   - 0 - monitor the TX_POWER
   - 1 - continuously monitor the TX_POWER

  - 2 - monitor the DCL_MONITOR registers
  - 3 - continuously monitor the DCL_MONITOR registers
  - 4 - monitor the TX_POWER, SID (signal I.D.), and FCM_REG
  - 5 - continuously monitor the TX_POWER, SID (signal I.D.), and FMC_REG
  - 6 - monitor the TX_POWER, RX_POWER, FCM_REG and SID (signal I.D.)
  - 7 - continuously monitor the TX_POWER, RX_POWER, FCM_REG and SID (signal I.D.)

Snapshot Monitoring

The current state of the snapshot monitor can be read at any time from the SNAPSHOTSTATUS register. The bit field representation for the SNAPSHOTSTATUS register is defined in Table 7-31.

Table 7-31: **SNAPSHOTSTATUS Bit Fields**

Bit	Name	Description
Bit[0]	dcl_enabled	Indicates whether DCL routine is enabled or not.
Bit[1]	monitor_enabled	Indicates whether the snapshot monitor is enabled or not.
Bit[2]	monitor_paused	Indicates whether the snapshot monitor is paused or not.
Bits[7:3]	unused	
Bits[15:8]	monitor_mode	Reports which snapshot monitor mode is enabled.
Bits[23:16]	monitor_ports	Reports which ports are active in the snapshot.
Bits[31:24]	monitor_full	Reports how full the snapshot buffer is (in %).

Snapshot Pausing

The snapshot monitor updates can be paused using the DCL_PAUSE_SNAPSHOT_MONITOR command when DCL is running or the SNAPSHOT_MONITOR_PAUSE command when DCL is disabled. Pausing the updates can be useful when reading the monitor while it is updating in one of the continuous modes.

Snapshot Reading

The Snapshot size is 32768x32. This buffer must be read in pages (similar to the reading of the capture RAMs). Each page is 512x32 so 64 page reads are required to read the entire snapshot buffer.

To read the snapshot (DCL must be disabled):

1. Write the page_number to register PARAMETER_0 (page_number from 0 to 63).
2. Send the DUMP_SNAPSHOT_MONITOR command to dump the requested page.
3. Read raw snapshot values from page transfer region (registers 512 to 1023).

4. Read the last written address into the snapshot monitor from the ALT_COMMANDSTATUS0 register.

TX Power Snapshot Format

Average TX power for selected antenna(s) is stored at every METERLENGTH interval as a 32-bit value.

- The buffer can store ~16/N minutes for a 30 ms METERLENGTH, where N is the number of antenna powers being saved.
- Antenna powers are saved interleaved in the snapshot memory starting from the lowest antenna value.
- Convert average power to dBFS using 10*log10(*X/2^{30});
- For continuous monitoring the last written address is returned in ALT_COMMANDSTATUS0 register after the `DUMP_SNAPSHOT_MONITOR` command is executed. This register can be used to unwrap the relative time of the samples in the snapshot.

DCL Monitor Snapshot Format

- Snapshot size is 32768x32
- DCL monitors for selected antenna(s) are stored after each update of an antenna (after COUNTER increments in the DCL monitor).
- Can store 1024/N DCL updates, where N is the number of antenna DCL monitors being saved.
- DCL monitors are stored in 32-word sections for each antenna. Register 16 of each section is tagged with the corresponding antenna value and current RUNTIMELSW value.
- Register 16 Bits[27:0] RUNTIMELSW & 0x00FFFFFF
- Register 16 Bits[31:28] antenna
- For continuous monitoring the last written address is returned in ALT_COMMANDSTATUS0 register after the DUMP_SNAPSHOT_MONITOR command is executed. This register can be used to unwrap the relative time of the samples in the snapshot.
- TX Power, SID, FCM_REG Snapshot Format
  - tx_pwr = data(1:3:end);
  - sid = data(2:3:end);
  - tmpFCM = data(3:3:end);
- TX Power, RX_Power, SID, FCM_REG Snapshot Format
  - tx_pwr = data(1:4:end);

- rx_pwr = data(2:4:end);
- sid = data(3:4:end);
- tmpFCM = data(3:4:end)

Running the DCL

Normally the DPD core is activated using the DCL control modes in Table 7-32. PORTNUM does not need to be specified because the DCL automatically operates on all available ports. Various status monitors are provided in Table 7-32 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 in registers 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 7-32: **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.

DCL Monitors

The DCL monitor register in the host interface can be configured to monitor any of the available DCL monitors.

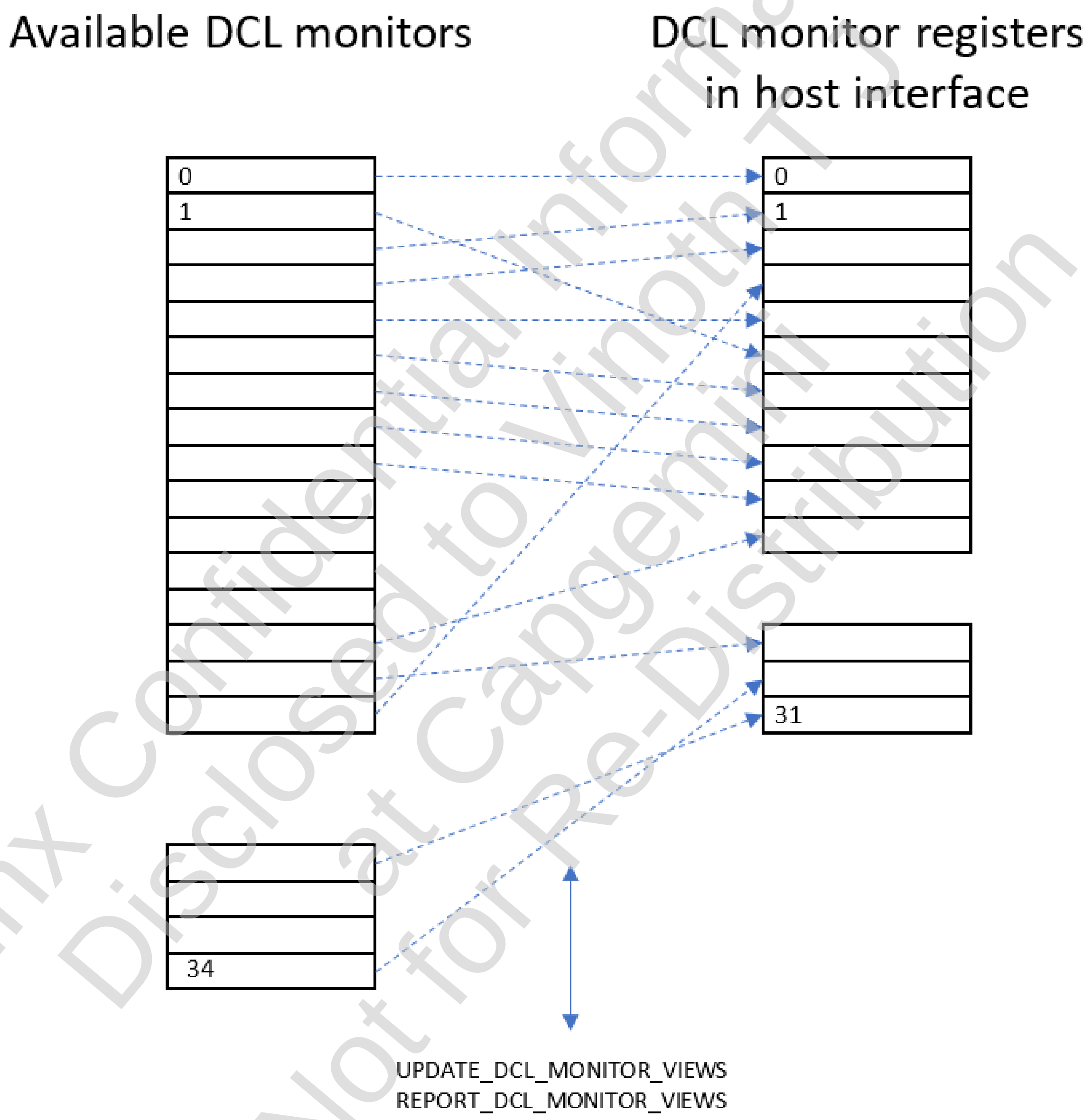


Figure 7-16: **DCL Monitoring**

The values can be updated and reported using the UPDATE_DCL_MONITOR_VIEWS and REPORT_DCL_MONITOR_VIEWS respectively. The DCL monitor selection is common for all antennas. The available DCL monitor signals are shown in Table 7-33. The default values reported are 0 through 31.

Table 7-33: **DCL Monitors**

Value	Mnemonic	Format	Description
0	FCM_REG	PACKED	Proprietary frequency content metric Bits[15:0] (S16.14) FCM1 Bits[31:16] (U16.14) FCM2
1	TX_POWER_NORMAL	U32.30	Average TX power during normal measurement (convert to dBFS as 10*log10(x))
2	RX_POWER_NORMAL	U32.30	Average SRx power during normal measurement (convert to dBFS as 10*log10(x))
3	LOOPGAIN	U32.24	Calibrated feedback loop gain (convert to dB as 20*log10(x)).
4	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.
5	UPDATE_INPROGRESS	U32.0	Possible values: 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 (coefficient selection is active) 35 - Port update skipped (coefficient selection is not active) 36 - Request port change for DPD update 37 - Request port change for VSWR measurement
6	LAST_UPDATED_SID	PACKED	Last updated signal identifier Bits[15:0] (S16.0) - Updated SID Bits[31:16] (U16.0) - Captured SID
7	LAST_UPDATED_STATUS	PACKED	The last update status values and a count of the number of consecutive failed updates since the last successful update. Bits[15:0] (S16.0 format) Indicates the returned status of the ECF update code Bits[31:16] (U16.0 format) A count of how many consecutive failed updates have occurred since the last successful update. The counter resets any time a successful update occurs and will saturate at 0xFFFF.
8	UPDATING_SID	PACKED	Indicates the currently updating SID. Bits[15:0] (S16.0) - Indicates the currently updating SID Bits[31:16] (U16.0) - Reserved

Table 7-33: **DCL Monitors *(Cont'd)***

Value	Mnemonic	Format	Description
9	FCM_CAPT_METRIC_REG	PACKED	Proprietary frequency content metrics Bits[15:0] (S16.14) FCM1_CAPT_METRIC Bits[31:16] (U16.14) FCM2_CAPT_METRIC
10	LOOP_PHASE_ROTATION	PACKED	Phase rotation between TX input and SRX Bits[15:0] (S16.14) PHASEROT_REAL Bits[31:16] (S16.14) PHASEROT_IMAG
11	UPDATE_MONITOR_0	PACKED	Current update status and classification along with highest classification since last reset of error histogram. Bits[15:0] (S16.0 format) Indicates the returned status of the ECF update code Bits[19:16] (U4.0 format) Indicated the classification of the ECF update code Bits[23:20] (U4.0 format) Highest classification of the ECF update codes in the error histogram. Bits[31:24] (U8.5 format) Ratio of number of passing updates in previous 32 attempts.
12	UPDATE_MONITOR_1	U32.0	Bit-mask with previous 32 updates pass/fail indication (lsb is current update).
13	MAX_MIN_LUT_VALUE	PACKED	Bits[15:0] (S16.0) contains the most negative value in all datapath LUTs. Bits[31:16] (S16.0) contains most positive value in all datapath LUTs.
14	MEASUREDPEAKEXPANSION	U32.24	Measured maximum expansion in the DPD filter (convert to dB as 10*log10(x)).
15	MSE	U32.30	Mean squared error between the input TX and received RX signal (convert to dBFS as 10*log10(x)).
16*	PAPR	U32.18	Peak to Average Power Ratio. Use 10*log10(PAPR) for dB value.
17	IBW_OBW_VALUES	PACKED	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.
18	ODDPEAKEXPANSION	U32.8	Peak expansion computed during ODD processing (20*log10(x))
19	POWER_METER_UPDATE_COUNTER	PACKED	Bits[15:0] (U16.0) TX_POWER_NORMAL and RX_POWER_NORMAL update counter Bits[31:16] (U16.0) TX_POWER_REFLECTED and RX_POWER_REFLECTED update counter

Table 7-33: **DCL Monitors *(Cont'd)***

Value	Mnemonic	Format	Description
20	PRE_MSE	U32.30	Mean squared error of raw aligned capture (convert to dB as 10*log10(x)).
21	CAPT_LOOPGAIN	U32.24	Current feedback loop gain (convert to dB as 20*log10(x)).
22	TX_POWER_REFLECTED	U32.30	Average TX power during reflected measurement (convert to dBFS as 10*log10(x))
23	RX_POWER_REFLECTED	U32.30	Average SRx power during reflected measurement (convert to dBFS as 10*log10(x))
24	LOOP_DELAY	U32.18	Delay in samples between TX input and SRX
25	QMC_GAIN_ERROR	S32.16	Use 20 x log10(QMC_GAIN_ERROR) to convert to dB error.
26	QMC_PHASE_ERROR	U32.16	Use QMC_PHASE_ERROR x180/pi to convert to error in degrees.
27	QMC_DC_OFFSET_I_ERROR	S32.16	LSB DC offset
28	QMC_DC_OFFSET_Q_ERROR	S32.16	LSB DC offset
29	QMC_UPDATE_COUNTER	U32.0	Total number of QMC updates after the function was started.
30	SUCCESSFUL_COUNTER	U32.0	Number of successful DPD updates since the function was started.
31	MEASURED_AVEPWR_GAIN	U32.24	Current average power gain of DPD filter (10*log10(x) for dB)
32	NMSE	U32.30	Normalized mean squared error between the input TX and received RX signal (convert to dB as 10*log10(x)).
33	PRE_NMSE	U32.30	Normalized mean squared error of raw aligned capture (convert to dBFS as 10*log10(x)).
34	DPD_PHASE_ROTATION	PACKED	Phase rotation between DPD input and DPD output Bits[15:0] (S16.14) PHASEROT_REAL Bits[31:16] (S16.14) PHASEROT_IMAG
35	DPD_IO_PAPR	PACKED	PAPR of DPD input and output signals Bits[15:0] (S16.8) TX_PAPR Bits[31:16] (S16.8) DPDO_PAPR
36	OUTPUT_PWR	U32.30	Average DPD output power (convert to dBFS as 10*log10(x))

Register 16 in the DCL monitors is not available with SNAPSHOT monitoring. When SNAPSHOT monitor is utilized, register 16 is over-written with the count and port number values.

Note: TX_POWER_REFLECTED and RX_POWER_REFLECTED are unused when VSWR monitoring is not enabled. QMC_GAIN_ERROR, QMC_PHASE_ERROR, QMC_DC_OFFSET_I_ERROR, QMC_DC_OFFSET_Q_ERROR and QMC_UPDATE_COUNTER are unused when TX QMC feature is not enabled.

DCL Error Histogram

DCL maintains a histogram count of all errors that are detected on each port. This histogram can be used to help identify any long term or sporadic error conditions. The error histogram for each port can be reset using the `RESET_ERROR_HISTOGRAM` or the `DCL_RESET_ERROR_HISTOGRAM` commands. The current error histogram for each port can be read using the `DUMP_ERROR_HISTOGRAM` or the `DCL_DUMP_ERROR_HISTOGRAM` commands. The histogram maintains 512 bins and each bin value represents the corresponding error number. Unknown conditions are counted in bin 0 (for example, R512 is bin 0 and R636 (512+124) corresponds to error -124). The bins saturate at 2^{32-1}.

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 7-17.

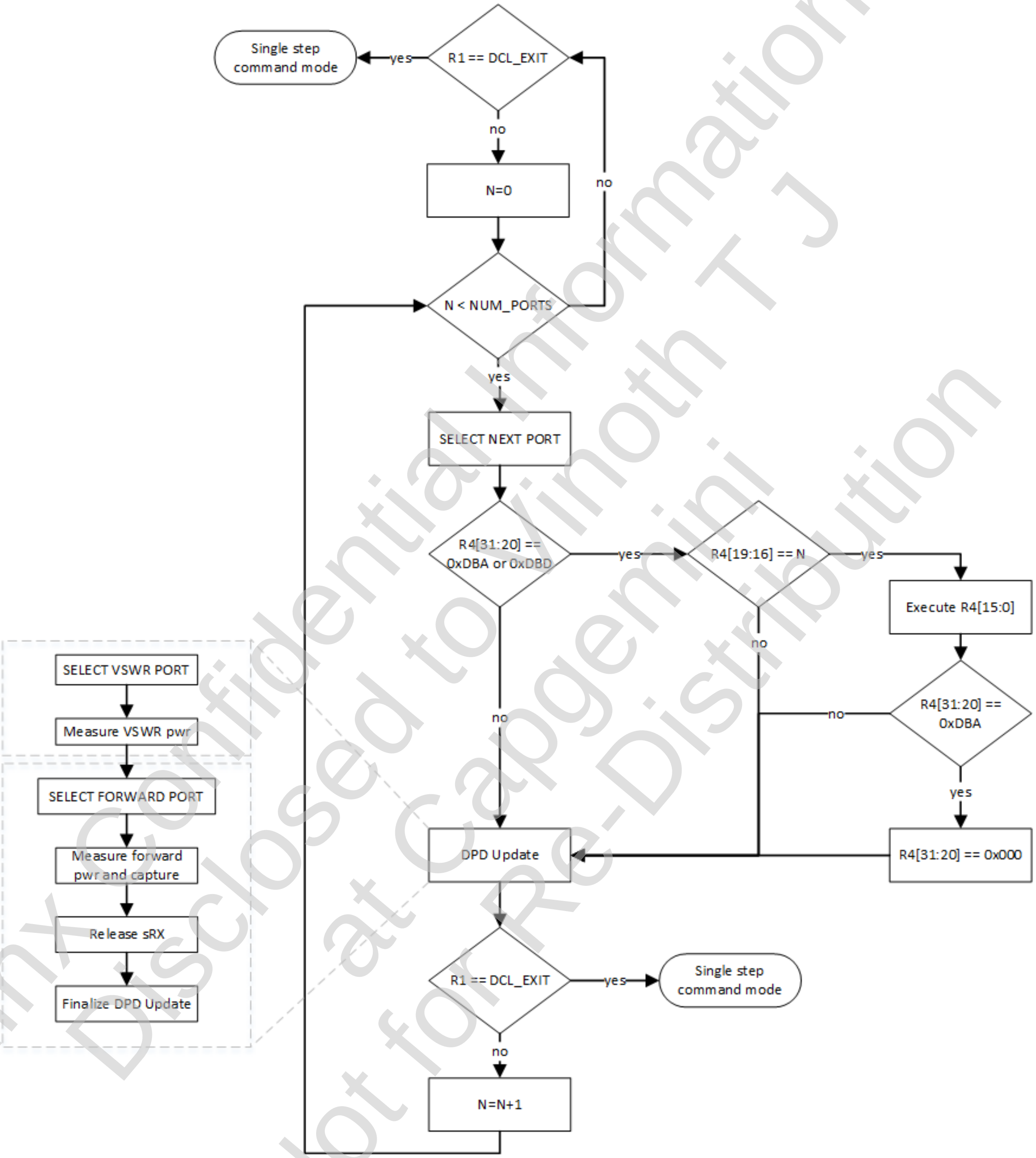


Figure 7-17: **DCL Commanding Loop**

In Figure 7-17, R1 refers to the CONTROLMODEREGISTER register and R4 refers to the PARAMETER_0 register in the host interface.

Commands are requested by writing proper values into the PARAMETER_0, PARAMETER_1, and PARAMETER_2 registers while DCL is operating. Each command is a single 32-bit value.

The command structure is:

- 0xDBAPMMXY for commands that should be run only once upon completion of the command this value will change to 0x000PMMXY.
- 0xDBDPMMXY for commands that re-run continuously each 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 7-34 shows a list of available DCL commands.

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

Table 7-34: **DCL Commands**

DCL Command Name	MM	Notes
portnum ("P") is unused		
DCL_GET_SKIP_PORT	8	Report the current skip port value to R512. 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	Set the current skip port value using the value specified in R512. 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.
DCL_PAUSE_SNAPSHOT_MONITOR	15	Toggles the pause/start of an active snapshot monitor.
DCL_UPDATE_DCL_MONITOR_VIEW	17	Update the DCL monitor view terms with values read from R512-R543.
DCL_REPORT_DCL_MONITOR_VIEW	18	Report the DCL monitor view terms in R512- R543.
Single portnum or portmask is supported (for portmask set P=0xF and portmask value in PARAMETER_1)		
DCL_RESET_ALIGNMENT	4	P=0xF will apply to all ports specified as portmap in PARAMETER_1 for this command.
DCL_RESET_COEFFICIENTS	5	XY indicates which set to reset, XY = 0xFF will reset all coefficient sets for the port. P=0xF will apply to all ports specified as portmap in PARAMETER_1 for this command.
DCL_CLEAR_C2L_OVERFLOW	7	Clears latched overflow error messages (both C2L and datapath). P=0xF applies to all ports specified as portmap in PARAMETER_1 for this command.
DCL_RESET_COEF_ALIGN	10	Reset all coefficients and sets alignment at the same time. P=0xF applies to all ports specified as portmap in PARAMETER_1 for this command. XY is not used.
DCL_RESET_ERROR_HISTOGRAM	12	Reset the current state of the error histogram. P=0xF applies to all ports specified as portmap in PARAMETER_1 for this command.
DCL_RESET_DCL_COUNTERS	13	Reset the counters associated with the DCL routine (includes error histograms).

Table 7-34: **DCL Commands *(Cont'd)***

DCL Command Name	MM	Notes
DCL_UPDATE_DAMPING_LEAKAGE	29	Update the ECF DAMPINGVALUE and LEAKAGEVALUE.
DCL_ENABLE_SNAPSHOT_MONITOR	14	Resets the snapshot monitor and defines which signals to monitor, the snapshot is enabled. XY is used to indicate which signal to monitor (see Snapshots Monitors)
DCL_UPDATE_DAMPING_LEAKAGE	29	New LEAKAGEVALUE and DAMPINGVALUE parameters loaded from PARAMETERS_2. Format is two packed 16-bit values (U16.15 format): PARAMETERS_2 = DAMPINGVALUE + (LEAKAGEVALUE + 2^16).
Only portnum specified in "P" is supported		
DCL_REPORT_AMAM_AMPM	0	R512-R681 lsw- ampm (S16.13), msw-amam (U16.12) 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 (U16.12)
DCL_REPORT_DIAGNOSTICS	1	Y indicates which diagnostics to report - 0 for DPD update diagnostics. See Table 7-36 for registers to reads. - 1 for HWA update time estimates. See Table 7-37 for registers to reads.
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. R512-R767 will contain the resulting ODD_AMAM response.
DCL_GET_HISTOGRAM	6	R512-R767 last capture histogram R768-R1023 long term histogram associated with last capture attempt
DCL_DUMP_ERROR_HISTOGRAM	11	Dump the current state of the error histogram for the specified port to R512-R1023
DCL_REPORT_QMC_COEFFICIENTS	16	Report the QMC coefficients for the specified port to R512- R515.
DCL_REPORT_CAPWIN_PARAMETERS	19	Report the CAPWIN parameters for the specified port to shared parameter region.
DCL_REPORT_TXQMC_PARAMETERS	20	Report the TXQMC parameters for the specified port to shared parameter region.
DCL_REPORT_ODD_PARAMETERS	21	Report the ODD parameters for the specified port to shared parameter region.
DCL_REPORT_MET_PARAMETERS	22	Report the MET parameter for the specified port to shared parameter region.
DCL_REPORT_ECF_PARAMETERS	23	Report the ECF parameters for the specified port to shared parameter region.
DCL_REPORT_ARCH_PARAMETERS	24	Report the ARCH parameters for the specified port to shared parameter region.
DCL_REPORT_CAPTURE_PARAMETERS	25	Report the CAPTURE parameters for the specified port to shared parameter region.

Table 7-34: **DCL Commands *(Cont'd)***

DCL Command Name	MM	Notes
DCL_REPORT_DCL_PARAMETERS	26	Report the DCL parameters for the specified port to shared parameter region.
DCL_UPDATE_FIXED_LOOPGAIN	27	New gain tracking level loaded from PARAMETER_2. Format is U32.24 (convert to dB as 20*log10(x)). A value of 0 switches to the Fixed DPD gain mode. Non-zero values use Fixed Loop gain mode.
DCL_UPDATE_LTM_TC	28	New LTM time constant parameters loaded from PARAMETERS_2. Format is two packed 16-bit values: PARAMETERS_2 = LTM_TC_1 + (LTM_TC_2 + 2^16). ***Note:*** To change LTM_TC_1 or LTM_TC_2 with this DCL command the corresponding parameter must be set to a non-zero values before the DCL is started using UPDATE_ARCH_PARAMETERS command.
DCL_REPORT_MSP_PARAMETERS	30	Report the MSP parameters for the specified port to shared parameter region.
Only portnum specified in "P" is supported, the command is executed when SRX port for "P" is active		
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. A new capture is always performed prior to computing the psd() which is reported in R512-R639.

Exiting the DCL

Bit[0] of the SNAPSHOTSTATUS register indicates when the DCL is enabled, 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 7-17). The delay between writing the EXIT_DCL command and the actual exiting of the DCL routine can be up to a full update interval.

Automatic DCL Exit for Debugging

When DPD is operating in the DCL mode it will continuously update each port while reporting status for each update in the DCL monitor region. To aid in system debug the DCL may be configured to automatically stop and exit when then the resulting status for the latest updated meets user specified criteria. This gives the capability to stop the DCL and preserve the DPD state or condition for additional diagnostic gathering. The DCL automatic exit feature is configured using the registers described in Table 7-35.

The automatic DCL exit is enabled by prepending 0xDBAE to the upper 16-bits of the EXIT_DCL command. The range of status values that will cause DCL to exit is written to the DCL_AUTO_EXIT_RANGE register and the associated ports to monitor are defined in the DCL_AUTO_EXIT_PORTS register. For example, if we wanted the DCL to exit when the DPD_COEF_LUT_OVERFLOW_FAILURE status is seen on port 1 or 2:

- Write 0xDBAE0006 to DCL_AUTO_EXIT_PORTS to define which ports to monitor
- Write 0xFF89FF89 to DCL_AUTO_EXIT_RANGE to define the range of status values to exit on
- Write 0xDBAE0014 to CONTROLMODEREGISTER to enable the DCL auto exit feature

At this point the DCL will continue to operate until the conditions specified in DCL_AUTO_EXIT_PORTS and DCL_AUTO_EXIT_RANGE. If these conditions are met the DCL will automatically exit, thus preserving the current state of DPD for further diagnostics gathering and debugging if needed.

Table 7-35: **DCL Automatic Exit Control Registers**

Page Transfer Base Offset	Mnemonic	Description
7	DCL_AUTO_EXIT_RANGE	Bits[31:16] - 16-bit signed status value Bits[15:0] - 16-bit signed status value The upper and lower 16 bits are used to define the range of status values (inclusive) that will cause the DCL to automatically exit. The limits may be specified as lower-value:higher-value or higher-value:lower-value.
8	DCL_AUTO_EXIT_PORTS	Specify the desired ports to monitor when DCL_AUTO_EXIT is enabled. Bits[31:16] used to specify port specific or all port monitoring. Bits[15:0] used to specify which ports to monitor. If Bits[31:16] = 0xDBAE then each of the lower Bits[7:0] are used as a bit mask to indicate which ports to monitor. Bit[0] corresponds to port 0. Bits with value of 1 are monitored, bits with value of 0 are ignored. When Bits[31:16] != 0xDBAE the lower bits are ignored and all ports will be monitored for DCL_AUTO_EXIT feature.

Single Step Diagnostics

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 7-16. The registers in Table 7-27 can be used for additional diagnostics. These registers are valid only after the `RUN_UPDATE_COEFFICIENTS`, `RUN_UPDATE_COEFFICIENTS_V2`, or `CAPTURE_AND_ALIGN` commands are executed and apply only to the current antenna.

Single-step diagnostics are stored for each antenna. Use the `REPORT_DIAGNOSTICS` or `DCL_REPORT_DIAGNOSTICS` command to dump the desired antenna diagnostics to the page transfer area of the host interface. Table 7-36 and Table 7-37 show the available diagnostics. The diagnostics for a requested port are reported using the following process:

1. Write portnum to PARAMETER_0 register.
2. Write diagnostic set number to PARAMETER_1 register.
   a. 0 for DPD update diagnostics. See Table 7-36 for registers to reads.
   b. 1 for HWA update time estimates. See Table 7-37 for registers to reads.
3. Send REPORT_DIAGNOSTICS command.

Table 7-36: **Signal Stepping DPD Update Diagnostic Registers**

Page Transfer Base Offset	Mnemonic	Value	Description
0	CAPTURED_FCM_REG	Two packed signed16.14 values	Frequency content metrics (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	Calibrated feedback loop gain (convert to dB as 20*log10(x)). ***Note:*** The LSB of this register indicates the value of FIXED_GAIN_MODE and the MSP parameters.
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.

Table 7-36: **Signal Stepping DPD Update Diagnostic Registers *(Cont'd)***

Page Transfer Base Offset	Mnemonic	Value	Description
9	MSE	U32.30	Mean-squared-error (convert to dBFS 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 greater than the 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).
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
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 dBFS).
16	FRAME_SYNC_INTERVAL	U32.0	Clock cycles between frame. sync. pulses (must be a multiple of 16).

Table 7-37: **Single Stepping HWA Diagnostic Registers**

Page Transfer Base Offset	Mnemonic	Value	Description
0	INITIAL_FULL_UPDATE_TIME	U32.0	Time (in µs) required for initial full DPD update (potentially includes delay search).
1	INITIAL_DELAY_SEARCH_TIME	U32.0	Time (in µs) required for initial delay search. This can be reduced by reducing MAXDELAY-MINDELAY range.
2	CURRENT_FULL_UPDATE_TIME	U32.0	Time (in µs) required for latest full DPD update.

Table 7-37: **Single Stepping HWA Diagnostic Registers *(Cont'd)***

Page Transfer Base Offset	Mnemonic	Value	Description
3	HWA_LEVEL_1	U32.0	Estimated Time (in µs) required for latest full DPD update if HWA level is set to 1 at build time.
4	HWA_LEVEL_2	U32.0	Estimated Time (in µs) required for latest full DPD update if HWA level is set to 2 at build time.
5	HWA_LEVEL_4	U32.0	Estimated Time (in µs) required for latest full DPD update if HWA level is set to 4 at build time.
6	HWA_LEVEL_8	U32.0	Estimated Time (in µs) required for latest full DPD update if HWA level is set to 8 at build time.

Signal Analysis

To aid setup and debug, the control modes shown in Table 7-38 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 7-38: **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.
23	GET_HISTOGRAM	Present the 256 bins of the transmit histogram associated with the last capture attempt at the host interface RAM page transfer area. R512-R767 last capture histogram R768-R1023 long term histogram associated with last capture attempt.
24	GET_CURRENT_HISTOGRAM	Present 256 bins of the current histogram for the active port at the host interface RAM page transfer area.

Table 7-38: **Signal Analysis Control Modes *(Cont'd)***

Mode Number	Mnemonic	Description
25	READ_CAPTURE_POWER_METERS	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). The integration period for the measurement is returned in 516(L) and the PORTNUM that was active during the capture is returned in 517. To convert to dBFS use: $10*\log10((LSW + MSW*2^{32})/2^{30/L})$
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.
27	RUN_ODD_AMAM	Estimate Expansion Curve using Overdrive Detection Algorithm
29	GET_CAPTURE_PSD	Capture new samples and compute the power spectral density of the contents in a capture RAM. See GET_CAPTURE_PSD Function.
36	MONITOR_CAPTURE_LOCATION	Monitor Capture Location to assess setting of Capture Window Parameters.
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.

For example, the signal analysis control modes are used 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 six 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 in Table 7-13) 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 6N-1. The six capture RAMs are formatted as follows after the `CAPTURE_NEW_SAMPLES` or `CAPTURE_AND_ALIGN` commands:

Table 7-39: **Capture RAMs**

Page Range	Mnemonic	Description
0 to N-1	TX_PRE	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.
N to 2N-1	RX	The SRX 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:*** The RX samples are -6dB below the TX samples after running the `CAPTURE_AND_ALIGN` command.
2N to 3N-1	TX_POST	The transmit data at the output 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.
3N to 4N-1	TX_PRE_RAW	Same format as TX_PRE, can be read regardless of CAPTURERAMSTATE.
4N to 5N-1	RX_RAW	Same format as RX, can be read regardless of CAPTURERAMSTATE.
5N to 6N-1	TX_POST_RAW	Same format as TX_POST, can be read regardless of CAPTURERAMSTATE.

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 7-18 and Figure 7-19 show examples using both the COMPUTE_PA_AMAM_AMPM_RESPONSES and the COMPUTE_AMAM_AMPM_RESPONSES commands.

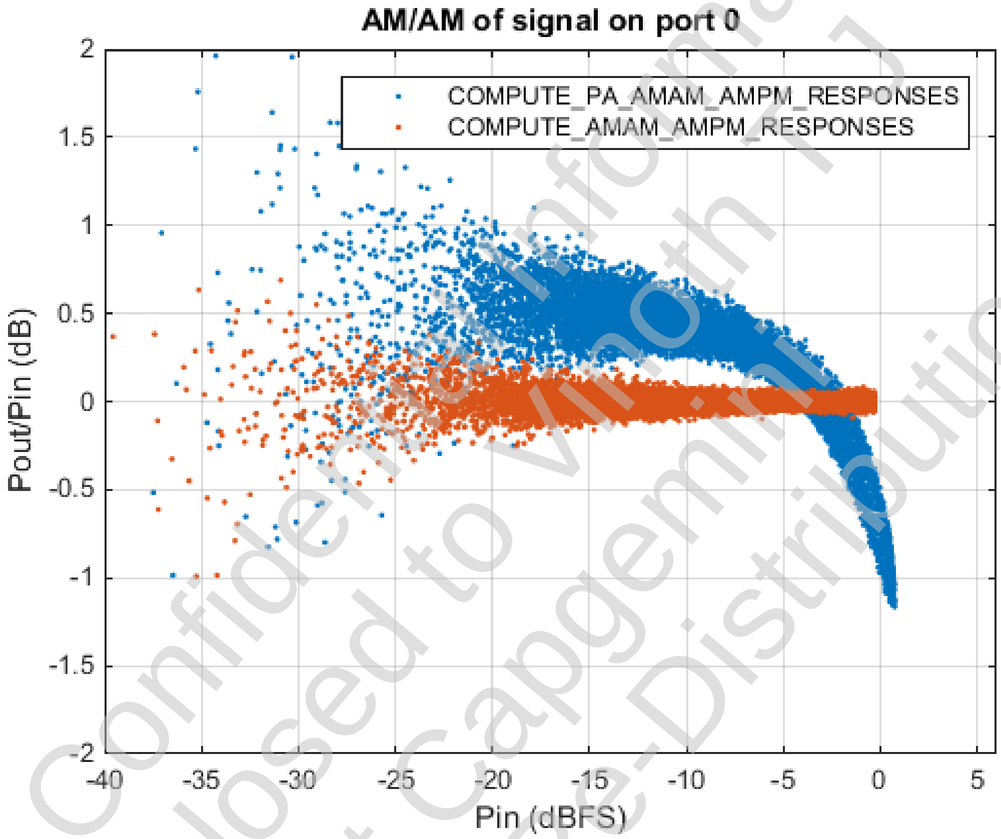


Figure 7-18: **AM/AM Response for Corrected and Uncorrected Signals**

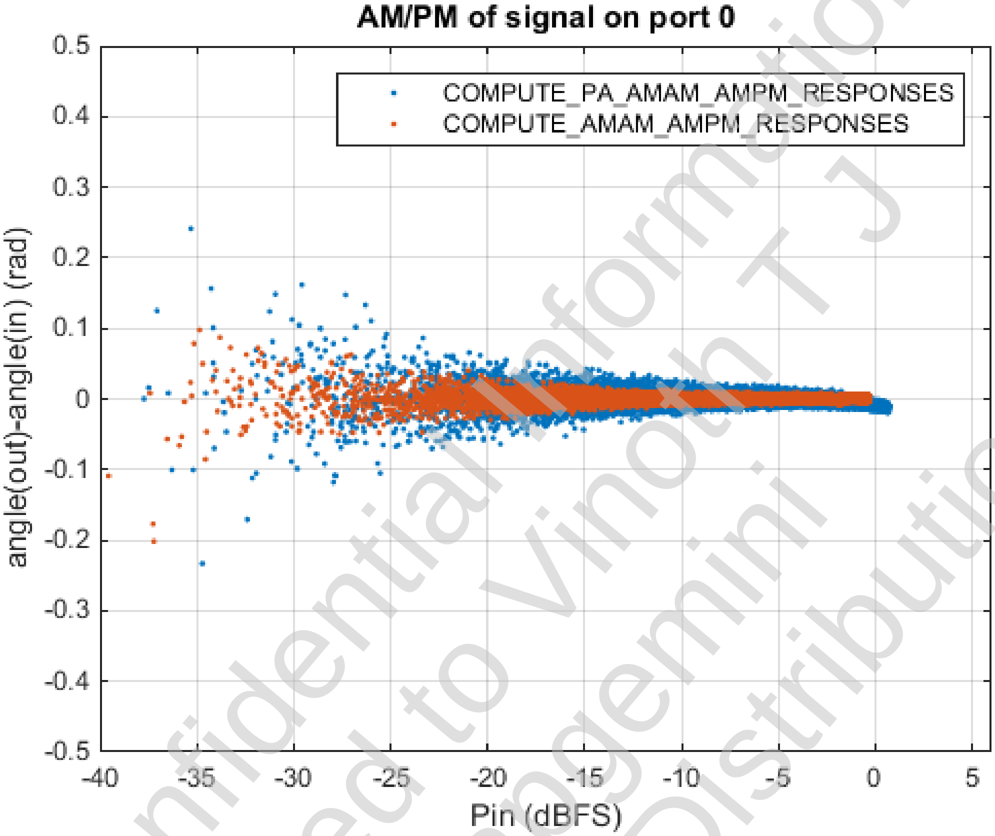


Figure 7-19: **AM/PM Response for Corrected and Uncorrected Signals**

The input magnitude signals associated with the COMPUTE_PA_AMAM_AMPM_RESPONSES will be reduced by the programmed DATAPATH_GAIN. Figure 7-18 and Figure 7-19 have scaled the input magnitude signals from the COMPUTE_PA_AMAM_AMPM_RESPONSES prior to plotting the results.

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 512 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. 1 for RX or 2 for DPD output 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 512 registers of PAGE_TRANSFER area with the format of U32.16. Figure 7-20 show an example of the normalized results using two LTE 5 MHz carriers at -15 and 15 MHz offsets sampled at 307.2 MHz. The three plots are for X = 0 and Y = 0, 1, and 2 (that is, no decimation, 2x decimation and 4x decimation).

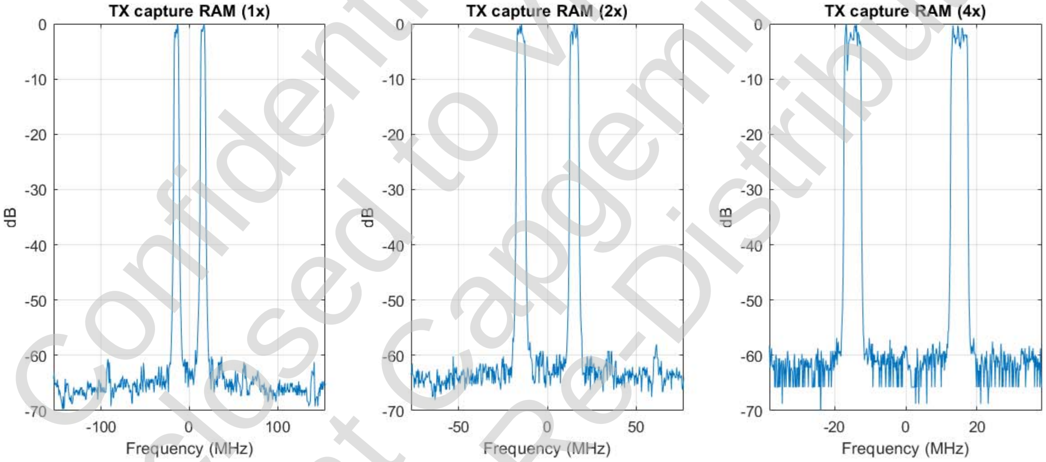


Figure 7-20: **GET_CAPTURE_PSD Results**

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 13 fractional bits.

Figure 7-21 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 using 20*log(x).

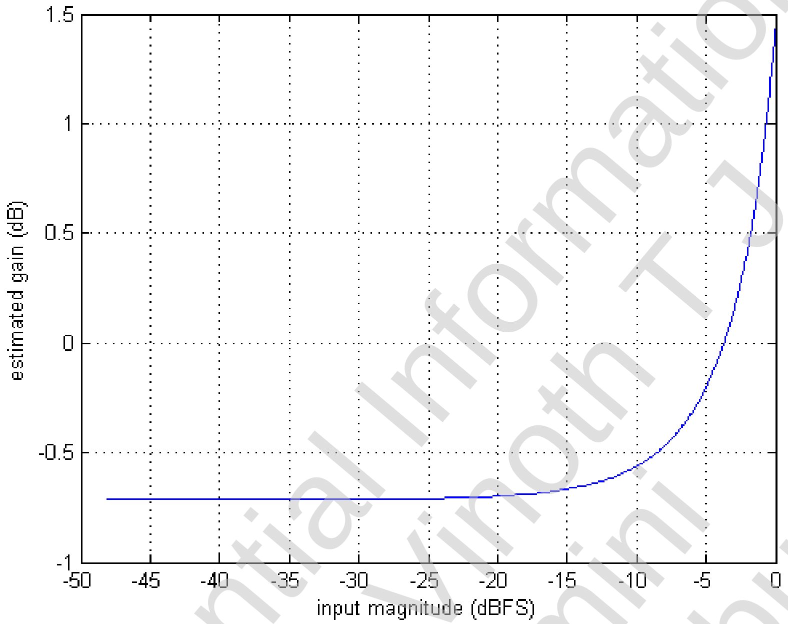


Figure 7-21: **Expansion Curve**

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 along with either the DPD output power or the SRX power at each delay is reported in the page transfer area.

The delay range to sweep across along with the alternate power to report is selected using PARAMETER_0 and PARAMETER_1 as follows:

- PARAMETER_0 = `max_delay`
- PARAMETER_1[30:0] = `min_delay`
- PARAMETER_1[31] = `alt_src` (0-SRX or 1-TX post DPD)

The captured power at each delay is report in the PAGE_TRANSFER area. The DPD input power is in the LSW and the alternate power is in the MSW of each address.

All page transfer address values are initialized to 0xFFFFFFFF at the start of this function. The first capture occurs with a delay equal to `min_delay` and the power is reported in address 512. The delay is increased by floor `(max_delay-min_delay)`/512, a new capture is performed and its associated 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) = min_delay+n*floor(max_delay-min_delay)/512;
        if(capture_sucessful)
            host_interface(512+n) = [CAPTURED_TX_PWR | (CAPTURED_ALT_PWR << 16)];
        else
            host_interface(512+n) = CAPTURE_ERROR_CODE;
        end
end
```

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 307.2 MHz with PARAMETER_0 set to 3072000 is shown in Figure 7-22.

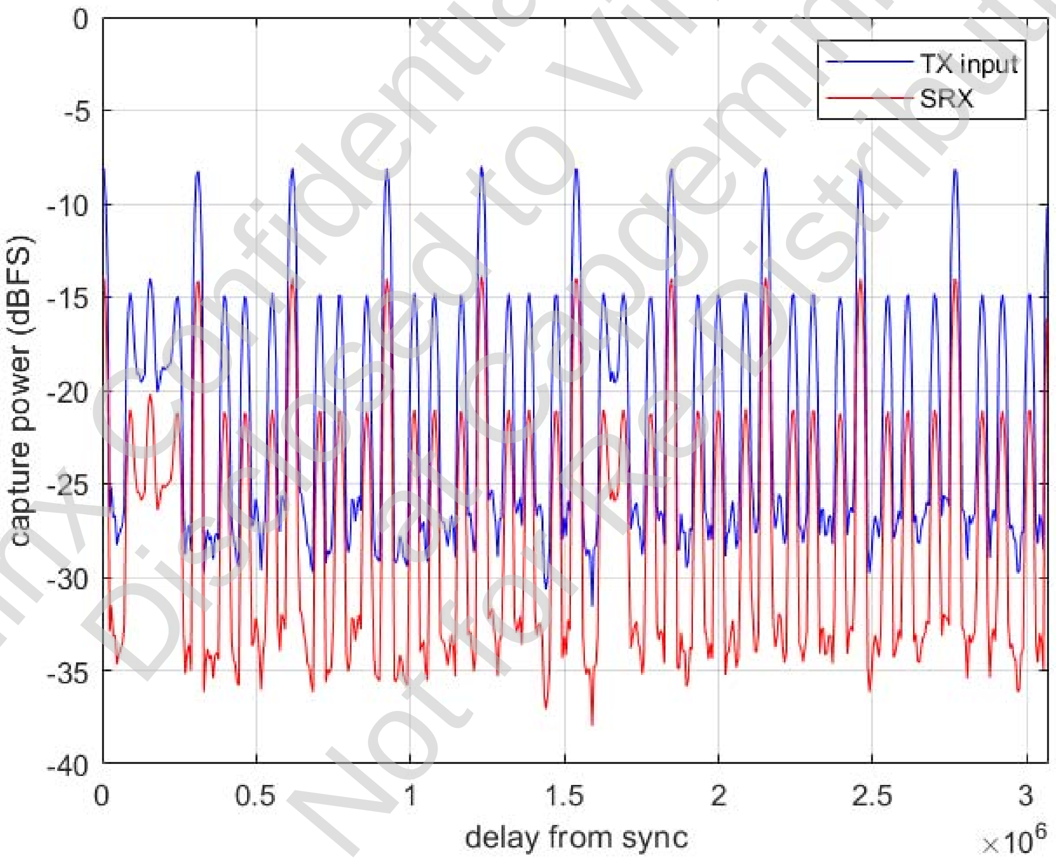


Figure 7-22: **Result of GET_SWEPT_CAPTURE_POWER While Transmitting a Repeated TM2.0 Waveform**

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

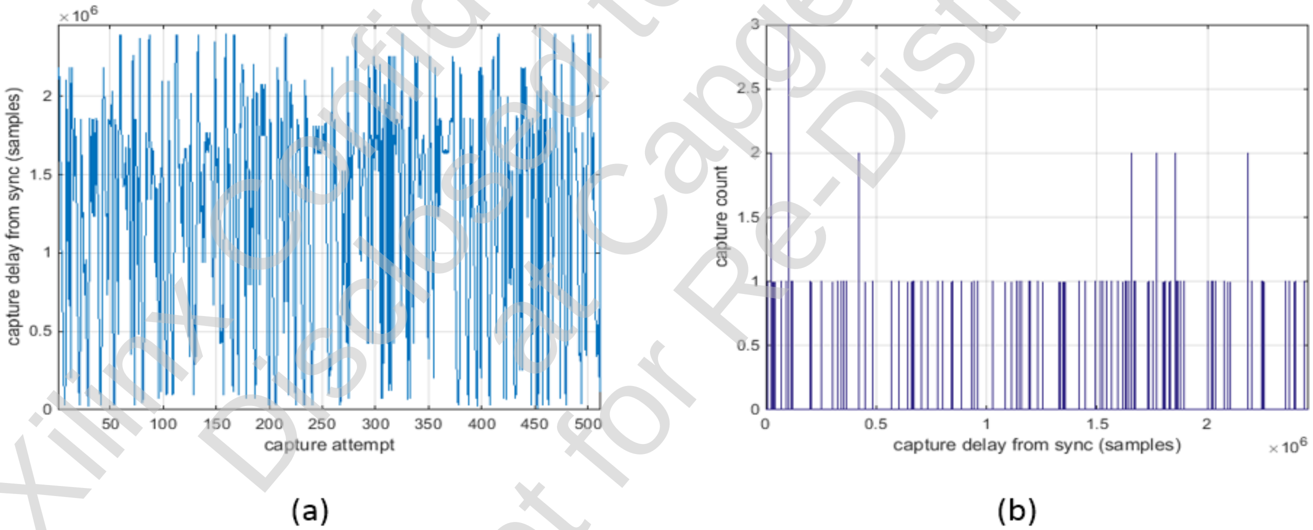
MONITOR_CAPTURE_LOCATION

DPD will perform 512 captures using the parameter `CAPTUREMODE` as defined by you. 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 7-23: **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 the port number, from which you want to read coefficients, to address PORTNUM in the host interface.

   By default the set for SID = 0 is read. To read other SID associated coefficient sets, write the desired SID number to the PARAMETER_0 register in the format 0xC0EF00XX (where XX is the desired SID/SET number). The value of XX can range from 0 to 24.

2. Run the DPD command `REPORT_COEFFICIENTS`.

   See DPD Application Software Boot Process for more information.

3. Read values 512 to 1023 (512 values) from the host interface. These values are the coefficient set represented in a proprietary format.

Loading DPD Coefficient Sets

1. Write the port number, to which you want to load the coefficient, in the PORTNUM register.
2. By default all SID sets are loaded. To write individual sets associated with an SID, write the desired SID number to PARAMETER_0 register in the format 0xC0E00XX (where XX is the desired SID/SET number). The value of XX can range from 0 to 2F4.
3. Write the 512 values that are read during the Reading DPD Coefficients Sets process into registers 512 to 1023 for the desired page number.
4. Trigger th DPD command `LOAD_COEFFICIENTS`.

   This command loads the coefficients into the DPD memory, but does NOT load them into the DPD filter.

5. (Optional) Trigger command DPD_ON to load the required coefficient set into the DPD filter.

Note: The coefficients are read in a scrambled format with a scrambling key that is continually changing. Hence, multiple reads of the same coefficient set are not guarantee to return the same value. To aid in validation of coefficient loading, coefficient reads that occur within 100 METERLENGTH intervals of a coefficient load will use the same scrambling key that was used in the load.

QMC Coefficient Handling

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

Reading QMC Coefficients Sets

1. Write the port number, to which you want to load the coefficient, to the PORTNUM register.
2. Trigger the DPD command `REPORT_QMC_COEFFICIENTS`.
3. Read values 512 to 515 (four values) from the host interface. These values are the coefficient set represented in a proprietary format.

Loading QMC Coefficient Sets

1. Write the port number to load the coefficient to PORTNUM register.
2. Write the four values that are read during the Reading QMC Coefficients Sets process into registers 512 to 515.
3. Trigger the 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 the 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 this case, these 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 the 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 ACTIVESRXPORTREQUEST 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

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 7-33.
3. Determine whether the values for the transmit and receive powers are as expected.
4. Perform required operations as detailed in to ensure that the signal inputs conform to the recommendations in Appendix D, Correction Performance.

Pre-distortion Operations and Achieving Performance

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.

Appendix A

Verification, Compliance, and Interoperability

Hardware Testing

Xilinx has conducted extensive functional coverage testing of software and hardware features using the supported evaluation 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.

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 v13.1 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.

DPD v13.0 to v13.1

- DPD IP Parameters
    - New Build Time option of Filter Structure (Filter Structure = 11) added.
- Host Interface Register Updates
    - SNAPSHOTSTATUS register moved to status region (register 61 changed to 24)
    - Various register locations for signal monitors have moved/renamed
        - RUNTIMELSW (register 64 changed to 60)
        - RUNTIMEMSW (register 65 changed to 61)
        - METERLENGTH (62)
        - SRXQDC (register 62 changed to 63)
        - SRXIDC (register 63 changed to 64)
        - SRXLSW changed to SRXAVE (register 66 changed to 65)
        - SRXMSW changed to DPDOUTPUT_PAPR (register 67 changed to 66)

        - DPDOUTPUT_AVE (67)
        - TXPOWERLSW_N changed to TXPOWERAVE_N
        - TXPOWERMSW_N changed to TXPAPR_N
    - DCL Monitor Updates
        - Removed SS_PSET0 and SS_PSET0_TARGET DCL monitors
        - Added UPDATE_MONITOR_0 and UPDATE_MONITOR_1 DCL monitors
        - Added DPD_IO_PAPR and OUTPUT_PWR DCL monitors
- DPD Parameter Updates
    - Added an alternate method for programming LTM_TC_1 and LTM_TC_2 parameters
    - Removed DCLPSTEP from UPDATE_DCL_PARAMETERS and REPORT_DCL_PARAMETERS
    - Changed value format for DCLTXLOWPOWER and DCLRXLOWPOWER from U32.0 to U32.30
    - Changed value format for SWITCH_0_1, SWITCH_1_2, SWITCH_2_3 and SWITCH_3_4 from U32.22 to U32.30
- DPD Command Updates
    - GET_HISTOGRAM changed from returning current histogram of requested port to returning last captured and long term histogram associated with latest update on requested port. Now similar to DCL_GET_HISTOGRAM.
    - GET_CAPTURE_HISTOGRAM change from returning the last captured histogram to returning the current histogram of the requested port.
    - Renamed DCL_REPORT_AMAMAMP to DCL_REPORT_AMAM_AMPM
    - Removed DCL_ENABLE_SNAPSHOT_PORTS command
    - Changed DCL_ENABLE_SNAPSHOT_MONITOR from command 13 to 14 and added support for portmap specification
    - Update DCL_RESET_COEF_ALIGN notes
    - Added new DCL_UPDATE_DAMPING_LEAKAGE and DCL_RESET_DCL_COUNTERS commands
- DPD Message Updates
    - Updated message values
        - PERFORMING_CAPTURE_PROCESS changed from 16 to 140
        - INVALID_DIAGNOSTIC_REQUEST changed from -94 to -6
    - Removed error messages
        - CONFIG_FAILURE_CAPTUREMODE_PIW

  - CAPTURE_PEAK_THRESHOLD_FAILURE
  - CAPTURE_DCL_SCA_FAILURE
- c-API Updates
  - Remove dcl_ss_pset0 from DPD c-API
  - Remove DCL_ENABLE_SNAPSHOT_PORTS command
  - Changed command number for DCL_ENABLE_SNAPSHOT_MONITOR
  - Make DCL_ENABLE_SNAPSHOT_MONITOR similar to ENABLE_SNAPSHOT_MONITOR
  - DCL_AMAM_AMPM changed to DCL_REPORT_AMAM_AMPM

DPD v12.0 to v13.0

- Zynq-7000 device support is deprecated. Customers targeting ZYNQ devices must use DPDv12.0 or below i.e., upgrade to DPDv13.0 is not supported when ZYNQ-7000 is targeted.

DPD IP Programmable Logic Changes (DPD v11.0 to v12.0)

- When targeting RFSoC DFE devices, DPD v12.0 is first release and no upgrade from v11.0 is supported. Refer to Filter Structure for new structures (8, 9,10) supported on RFSoC DFE.
- When targeting other MPSoC/RFSoC devices:
  - New Internal Parameter `INT_PARAM_3` is added.
  - Default value of `INT_PARAM_3` is 0 when target device selected is MPSoC/RFSoC.

Command Changes (DPD v11.0 to DPD v12.0)

- User parameters associated with UPDATE_MSP_PARAMETERS and REPORT_MSP_PARAMETERS have changed.
- User parameters associated with UPDATE_MET_PARAMETERS and REPORT_MET_PARAMETERS have changed.

DPD IP Programmable Logic Changes (DPD v10.1 to DPD v11.0)

- `XCI PARAMETERS ADDITION:` The DPD IP has a new internal parameters `INT_PARAM_D`. These internal parameters starting from `INT_PARAM_*` are generated based on other DPD IP parameters and should NOT be modify in the `xci` file manually. An unexpected change, may impact the functionality or change IP resource usage incorrectly.

- COMPILE TIME PARAMETERS ADDITION: A new parameter, LONG_TERM_MEMORY is added to handle long memory effects seen with Gallium Nitride (GaN) Amplifiers to meet stringent FCC and 3GPP MIMO requirements

Command Changes (DPD v10.1 to DPD v11.0)

- New value for DCLALGORITHM available for builds with LTM support enabled.
- Updated default for LS_REGULARIZATION
- Updated default for ENABLEBLINDRXQMC
- Updated default decimation rate used for IBW/OBW estimation
- New LTM_TC_1 and LTM_TC_2 parameters added to the ARCH parameters
- Increased FFT size with improved D.C. response in GET_CAPTURE_PSD command

Command Changes (DPD v10.0 to DPD v10.1)

- REPORT_DIAGNOSTICS and DCL_REPORT_DIAGNOSTICS now take a second parameter.
- GET_CAPTURE_RAM_PAGE supports additional CAPTURE_RAM.
- UPDATE_ECF_PARAMETERS and REPORT_ECF_PARAMETERS have new ENABLEBLINDRXQMC parameter.
- UPDATE_ARCH_PARAMETERS and REPORT_ARCH_PARAMETERS have a new RESERVED parameter.

Parameter Changes (DPD v10.0 to DPD v10.1)

- LTETDDCONFIGURATION has new default value.
- DCLTXLOWPOWER and DCLRXLOWPOWER have new default values.

Command Changes (DPD v9.0 to DPD v10.0)

- Update use of portnum and portmap for all commands.
- Add general use of PARAMETER_1.
- New UPDATE_DCL_MONITOR_VIEW and REPORT_DCL_MONITOR_VIEWS commands.

Parameter Changes (DPD v8.0 to DPD v9.0)

- ENABLEVSWRPWRMSR: This parameter is port specific and is used to configure the frequency of the VSWR measurements.
- MINUPDATETIME: This parameter is port specific.

- ODPENABLE: This parameter supports mode 3.
- SRXSELECTDELAY: units are changed from dpd_aclk referenced to µs and the default is changed to 10 ms.

Command Changes (DPD v8.0 to DPD v9.0)

- REPORT_COEFFICIENTS: Added PARAMETER_0 usage.
- LOAD_COEFFICIENTS: Added PARAMETER_0 usage.
- DPD_ON: Added PARAMETER_0 usage.
- RESET_COEFFICIENTS: Added PARAMETER_0 usage.
- UPDATE_CAPTURE_PARAMETERS: LTETDDCONFIGURATION parameter added.
- UPDATE_DCL_PARAMETERS: RESERVED parameter added.

Parameter Changes (DPD v7.1 to DPD v8.0)

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 Customize IP dialog box | IP catalog options. The required embedded software is delivered through a separate reference design.

A new user parameter named REDUCED_PRECISION has been added which corresponds to the **Use reduced precision coefficients** checkbox in the Customize IP dialog box | IP catalog. 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 AM-PM 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 (DPD v7.1 to DPD v8.0)

The usage of the m_axis_srx_ctrl stream was updated.

Functionality Changes (DPD v7.1 to DPD v8.0)

The RX samples are -6 dB below the TX samples after running the CAPTURE_AND_ALIGN command.

The DPD IP 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. Although most of the functionality is identical to DPD v7.1, there are significant changes to the DPD parameter set usage. There is a support for new devices as well. The DPD software now supports dual processor SMP mode in addition to bare metal and AMP modes. See Chapter 7, 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 the DPD IP. This is available through the Digital Pre-Distortion Evaluation Lounge. See the associated User Guide, L*ogiCORE IP Digital Pre-Distortion v13.1 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 66685

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 what is allowed in the product documentation.
- Change any section of the design that labeled as 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 10].

Reference Boards

The ZCU102, ZCU111, ZCU208, ZCU216, and ZCU670 (for RFSoC DFE) Xilinx development boards support the DPD v13.1 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 Debug Interface tool, as described in the *Digital Pre-Distortion Debug Interface User Guide* (UG989) [Ref 1] as well as with the DFE Demo v8.0 kit, as described in the *DFE Demo User Guide* (UG1163) [Ref 16].

Appendix D

Correction Performance

Overview

Since DPD is a system-level function that involves 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 signals 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 the DPD IP, some headroom at the DPD input is required to allow the captured RX samples to be gain aligned to the reference TX samples without overflowing the 16-bit space. However, 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.

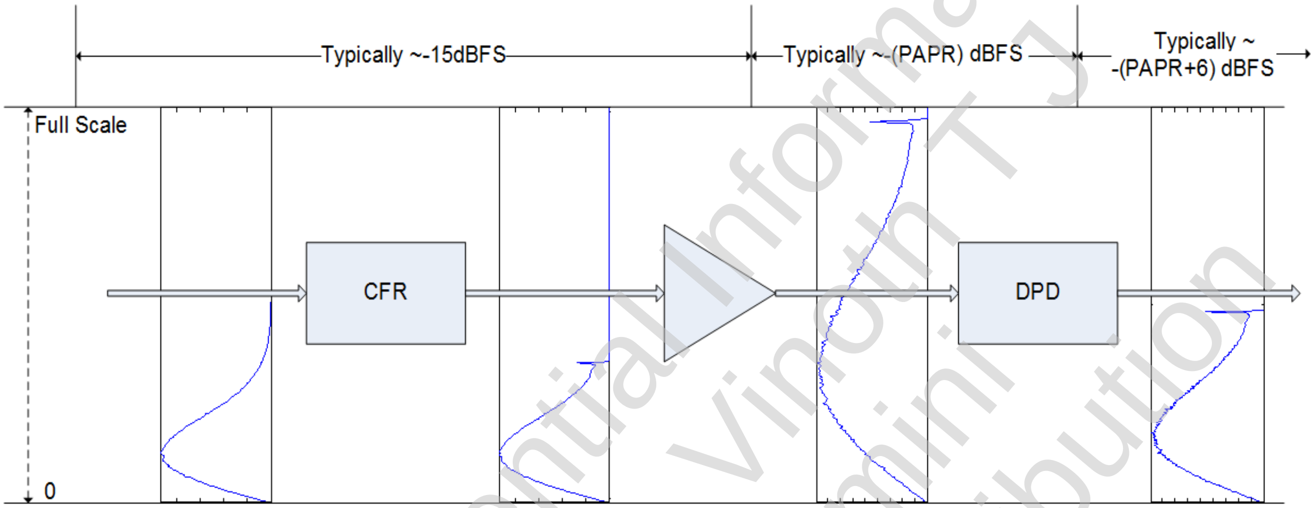


Figure D-1: **Example of Typical Histograms Along DFE Processing Chain**

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

The measured power of the received signal in the sample receiver should be at -12 dBFS, that is 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 that are 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.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. *Xilinx LogiCORE IP Digital Pre-Distortion Debug Interface User Guide* (UG989), registration required
2. *AMBA AXI4-Stream Protocol Specification* (ARM IHI 0051A)
3. *Xilinx AXI Design Reference Guide* (UG1037)
4. *AMBA AXI and ACE Protocol Specification* (ARM IHI 0022D)
5. Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973)
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. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
11. *Vivado Design Suite User Guide: Implementation* (UG904)
12. *ISE to Vivado Design Suite Migration Guide* (UG911)
13. *PetaLinux Tools Documentation Reference Guide* (UG1144)
14. *Zynq UltraScale+ MPSoC Software Developers Guide* (UG1137)
15. *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085)
16. *DFE Demo User Guide* (UG1163)
17. *Xilinx Peak Cancellation Crest Factor Reduction (PC-CFR)* product page
18. *Zynq UltraScale+ MPSoC Overview* (DS891)
19. *Libmetal and OpenAMP for Zynq Devices User Guide* (UG1186)
20. *Xilinx Software Command Line Tool (XSCT) Reference Guide* (UG1208)
21. *Vitis Unified Software Platform Documentation: Embedded Software Development* (UG1400)
22. *Zynq UltraScale+ RFSoC DFE Targeted Reference Design* (UG1530), registration required

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/19/2022	13.1	Added support for 2022.2 tools. Bug fixes: • Update SRX port release controls • DCL_UPDATE_FIXED_LOOPGAIN doesn't work with DCLALGORITHM=4 when multiple captures are used. • RUN_UPDATE_COEFFICIENTS_V3 can be called without proper ECF settings, can cause a crash Build time options: • Add new Filter Structure 11 • Remove DPD build options for single-set DCL • Remove DPD build options for no PIW • Remove DPD build options for non-reduced precision coefficients Feature enhancements: • Add classification for each error code. • Report TX and RX power in average power and add new DPD output and PAPR measurements • All captures performed after alignment calibration will capture integer delay aligned samples for all capture modes. • Add support for DCLAGORITHM=4 (no-LTM) with 4k and 8k capture buffers • Make structure 9 builds user programmable to structure 8 • LTS solution optimizations ◦ power meters operate on 2x decimated signal ◦ shared histogram c-API Updates: • Create xilinx_support_data equivalent for the c-API supplied in xdpd_lib.c/.h files • Add xdpd_enable_snapshot_monitor • Add xdpd_pause_snapshot_monitor • Add xdpd_snapshot_monitor_status • Add xdpd_dump_snapshot_monitor • Add xdpd_dcl_report_amam_ampm • Add xdpd_dcl_update_ltm_tc • Add xdpd_dcl_update_damping_leakage • Add xdpd_dcl_reset_dcl_counters

Date	Version	Revision
		• Add xdpd_host_writeword • Add xdpd_host_readword • Add xdpd_host_writeblock • Add xdpd_host_readblock • Add xdpd_dcl_reset_coef_align • Remove individual register xdpd_reg_write* and xdpd_reg_read* functions • Fix xdpd_handshake_get_opmode • Fix various data formats in get_diagnostics • Fix xdpd_update_dcl_monitor_views • Fix xdpd_report_dcl_monitor_views • Fix xdpd_dcl_update_fixed_loopgain • Update xdpd_host_interface.h • Update xdpd_report_ecf_parameters for DCL operation • Update xdpd_report_arch_parameters for DCL operation • Update xdpd_report_capture_parameters for DCL operation • Update xdpd_report_dcl_parameters for DCL operation • Update xdpd_report_odd_parameters for DCL operation • Update xdpd_report_met_parameters for DCL operation • Update xdpd_report_txqmc_parameters for DCL operation • Update xdpd_report_capwin_parameters for DCL operation • Update xdpd_report_msp_parameters for DCL operation • Update xdpd_reset_dcl for DCL operation • Update xdpd_report_qmc_coefficients for DCL operation • Update xdpd_get_monitors for new monitors
04/21/2022	13.0	• Added support for 2022.1 tools. • Added details for 4-phase DPD up to 1.96608 GSps on RFSoC DFE devices. • Removed Zynq-7000 and ZC706 board support. • Added new DPD command CONFIGURE_STRUCTURE.

Date	Version	Revision
12/15/2021	12.0	• Added support for 2021.2.1 tools • Added support for RFSoC DFE • Updated filter structures in Feature Summary • Updated timing performance description in IP Timing Performance • Added Table 2-2 • Updated DPD and RSS numbers in Resource Utilization • Updated Table 2-5 • Added capture sync description in Datapath AXI4-Stream Interfaces • Updated Filter Structure • Updated Figure 5-13 • Removed xdpd_init() API note in Table 7-4 • Updated BUILDSETTINGS Bits[29:24] notes in Table 7-13 • Updated group description in Software Control Modes • Added 303, 204, 203, 202, 93, 91, 90, and updated 113 in Table 7-19 • Updated notes in offset 2 and 3 in Table 7-21 • Updated note in offset 0 in Table 7-24 • Added offset 2 in Table 7-27 • Updated BUILDSETTINGS Bits[29:24] and ACTIVE_SID notes in Table 7-30
04/30/2021	11.0	• Added support for 2020.2 tools • Added the new `Long Term Memory` parameter • New optional support for pre-distortion of power amplifiers (PA's) that exhibit long term memory (LTM) (e.g., GaN PA's) ◦ Limited build and run time customization available • New DCLALGORITHM 4 ◦ Limited build and run time customization available • New Multi-Set Power parameters • New support for Fixed-Loop Gain operation • New LTM_TC_1 and LTM_TC_2 added to the ARCH parameter set. • New DCL_UPDATE_LTM_TC added to DCL commands • Interrupt from DPD HW to PS replaced with internal PS interrupt • Increased FFT size with improved D.C. response in GET_CAPTURE_PSD command • Simplify enabling updates with no effect by setting LEAKAGEVALUE=1.0 and DAMPINGVALUE=0.0 • Updated default Multi-Set Power thresholds

Date	Version	Revision
08/21/2020	10.1	• Added support for 2020.1 tools. • Added reporting of effect of HWA level on update times in DCL_REPORT_DIAGNOSTICS and REPORT_DIAGNOSTICS. • Added RESERVED field to UPDATE_ARCH_PARAMETERS and REPORT_ARCH_PARAMETERS. • Added ENABLEBLINDRXQMC parameter to **UPDATE_ECF_PARAMETERS** and **REPORT_ECF_PARAMTERS**. • Added new range for **GET_CAPTURE_RAM_PAGE**. • Updated definition of bit 2 in TUSER. • Updated notes on METERLENGTH parameter. • Updated format for SRXIDC and SRXQDC. • Updated values and default for LTETDDCONFIGURATION. • Updated default values for DCLTXLOWPOWER and DCLRXLOWPOWER. • Updated notes for UPDATE_INPROGRESS. • Corrected values field for MINDELAY. • Corrected notes for SRXSELECTDELAY.
03/16/2020	10.0	• Added support for 2019.2 tools. • FCM_v2 (reduces resources + improved performance with symmetric carrier dynamics) • New support for MEMORY_DEPTH=3 at build time • New DPD Filter structure support • New expanded DCL monitors • Removed AMP/Baremetal support • General LUT/FF resource reductions • Updated use of portnum and portmap for all commands • Changed portmap to portnum in all xdpd_report_*_parameters inputs • Changed portnum to portmap for the following functions: ◦ xdpd_reset_coefficients ◦ xdpd_reset_align_calibration ◦ xdpd_dpd_off ◦ xdpd_dpd_on ◦ xdpd_disable_output ◦ xdpd_enable_output ◦ xdpd_clear_diagnostics ◦ xdpd_clear_c2l_overflow ◦ xdpd_reset_error_histogram ◦ xdpd_qmc_reset ◦ xdpd_qmc_off ◦ xdpd_qmc_on • Vivado scripts updated to generate XSA file and export it to the embedded software flow • Software compile scripts updated to support latest XSCT commands used to generate Vitis software platform. • Reference design now includes example scripts for PetaLinux embedded Linux image generation • Example Host Application updated to include additional menu options

Date	Version	Revision
02/14/2019	9.0	• Added support for 2018.3 tools. • Improved DPD Resource utilization • New HAS_QMC option • Revamped reference design for DPD and CFR • New C API for host processor
05/25/2018	8.1	• Added support for 2018.1 tools • Added Zynq® UltraScale+™ RFSoC support • Added documentation for SUCCESSFUL_WITH_LATCHED_OVERFLOW status value. • Updated notes for ARCH_SEL. • Updated notes for CORRECTION_BW. • Updated notes for DCLTXLOWPOWER and DCLRXLOWPOWER. • Clarified Control Handshake usage. • Clarified the usage of SRXLSW and SRXMSW. • Updated notes for UPDATE_INPROGRESS. • Corrected formatting for DCL_AMAM_AMPM. • Correction to antenna selection options (replaced ACTIVEPORT with ACTIVESRXPORTREQUEST).
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 7, Application Software • Added new status values CONFIG_FAILURE_CAPTUREWINDOWS PORT_REQUEST_FORWARD, PORT_REQUEST_REFLECTED in Chapter 7, Application Software • Added Table 7-28 Register Set Using UPDATE_CAPWIN_PARAMETERS and REPORT_CAPWIN_PARAMETERS in Chapter 7, Application Software • Added CAPTURED_PWR in Table 7-36 in Chapter 7, 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 7, 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
11/28/2013	DRAFT	• New update algorithm • Vivado Design Suite support added • Zynq series support added

Date	Version	Revision
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 https://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 https://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.