



DesignWare Cores

SuperSpeed USB 3.0 Controller

Databook

DWC_usb3 – Product Codes

Copyright Notice and Proprietary Information

© 2018 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

Revision History	11
Preface	51
Chapter 1	
Product Overview	57
1.1 General Product Description	58
1.1.1 Applications	60
1.1.2 Compatibility and Quality	60
1.2 System Overview	61
1.3 Hardware and Software Partition	63
1.4 DWC_usb3 Block Diagrams	64
1.5 Features List	65
1.5.1 General Features	65
1.5.2 Application Interface Features	66
1.5.3 USB 3.0 Device Features	69
1.5.4 USB Class-Specific Device Features	69
1.5.5 USB 3.0 xHCI Host Features	70
1.5.6 USB 3.0 Dual-Role Device (DRD) Features	70
1.5.7 USB 3.0 Hub (SuperSpeed Only) Features	71
1.5.8 Power Optimization Features	71
1.5.9 High Speed Inter-Chip (HSIC) Features	73
1.5.10 Area Reduction Features	73
1.5.11 Performance Features	73
1.5.12 MIPS Reduction Features	74
1.5.13 Debug Features	74
1.5.14 Automotive Safety Features	74
1.5.15 System Bandwidth Requirements	75
1.6 Related Products	76
1.7 Speed and Clock Requirements	76
1.8 Area	77
1.9 Known Issues	78
1.9.1 Device	78
1.9.2 Host	78
1.9.3 Hub	79
1.9.4 Unsupported Features	79
Chapter 2	
Architecture	81

2.1	DWC_usb3 Block Diagrams	82
2.2	Logical Hierarchy, Clock Domains and Data Flow	83
2.2.1	Block Descriptions	84
2.2.2	Clock Domains	86
2.2.3	Detailed Hierarchy	87
2.2.4	Receiving a Data Packet	89
2.2.5	Reception Flow Details	90
2.2.6	Transmitting a Data Packet	93
2.2.7	Transmission Flow Details	94
2.3	System Memory Descriptor and Data Buffers	97
2.4	Configuration-Dependent Flexible Resource Allocation	98
2.4.1	Flexible Endpoint Mapping	98
2.4.2	Flexible IN Endpoint TxFIFO Allocation	99
2.4.3	Configuration-Dependent RAM Allocation	100
2.4.4	Smaller TxFIFO RAM Requirement	100
2.4.5	Threshold Mode and Store-and-Forward Mode Support	100
2.5	Clock Generation and Clock Tree Synthesis (CTS) Requirements	101
2.5.1	ram_clk	101
2.5.2	mac_clk, mac2_clk, mac3_clk	101
2.6	Reset Generation	110
2.6.1	Asynchronous Resets	111
2.6.2	Software Resets	112
2.7	USB 3.0 PHY Interface Unit (U3PIU)	114
2.7.1	U3 PIU Clocks	115
2.7.2	SS Tx Synchronous Data Transfer	115
2.7.3	SS Tx Asynchronous Data Transfer	116
2.7.4	SS PHY Control Transfer	117
2.7.5	SS Rx Synchronous Data Transfer	117
2.7.6	SS Rx Asynchronous Data Transfer	118
2.7.7	SS PHY Status Transfer	118
2.7.8	SS PHY Hard Wired Controls	119
2.8	USB 2.0 PHY Interface Unit (U2PIU)	120
2.9	USB 3.0 Link Power Management	121
2.9.1	LPM Functions	121
2.9.2	Transition from L0 to L1	125
2.9.3	L1 State Events	126
2.9.4	DWC_usb3 Device LPM Support	126
2.9.5	Device-Initiated L1 Exit	127
2.9.6	Host/Hub-Initiated L1 Exit	127
2.9.7	Reset Signaling-Initiated L1 Exit	127
2.9.8	Compile Time Configuration Options	128
2.9.9	Low Power Mode Operation in Host Mode	128
2.9.10	LPM Errata Support	128
2.10	USB 3.0 Link (U3LINK)	129
2.11	USB 3.0 MAC (U3MAC)	130
2.12	USB 3.0 Protocol Transaction Layer (U3PTL)	131
2.12.1	Block Diagram	131
2.12.2	Device OUT Transaction Engine (RXE)	132
2.12.3	Device IN Transaction Engine (TXE)	132

2.12.4 Host OUT Transaction Engine (HTXE)	133
2.12.5 Host IN Transaction Engine (HRXE)	133
2.12.6 Asynchronous Transaction Engine (AXE)	133
2.12.7 Control State Engine (CXE)	133
2.13 USB 2.0 MAC (U2MAC).....	134
2.13.1 U2MAC Block Description.....	134
2.14 USB 2.0 ROOT HUB (U2RHB).....	137
2.14.1 Port MUX (U2PRTMUX)	137
2.14.2 Synchronizers.....	138
2.14.3 Port Modules (U2PORT).....	138
2.15 Buffer Management Unit (BMU).....	139
2.15.1 Interfaces.....	139
2.15.2 High-level Flow	140
2.15.3 Block Description.....	142
2.15.4 TxFIFO Arbiters	145
2.16 List Processor (LSP).....	147
2.16.1 Device List Processor.....	148
2.16.2 Device Functional Description.....	150
2.16.3 Device Cache Memory Layout.....	150
2.16.4 Host List Processor	152
2.16.5 Debug Capability List Processor	157

Chapter 3

Memory Requirements.....	159
3.1 Synchronous Static RAM Requirement	160
3.2 Total Device RAM Requirement	162
3.2.1 Calculating RAM Size for Example Configurations.....	165
3.2.2 Device RxFIFO Data Allocation.....	167
3.2.3 Device TxFIFO Data Allocation.....	167
3.2.4 Number of Transfer Resources in Device	167
3.2.5 Number of Cached TRBs per Transfer in Device Mode	168
3.3 Total Host RAM Requirement	169
3.4 Packet Threshold and Burst Features for High Latency Systems	171
3.4.1 Transmit Path.....	171
3.4.2 Receive Path	172
3.5 Total DRD RAM Requirement	176
3.6 Rx/Tx FIFO Configuration	177
3.7 System Bus and RAM Bandwidth Requirements.....	182
3.7.1 Number of RAMs.....	183
3.7.2 2-Port or Single-Port RAM	183
3.7.3 Host Mode - Number of High-Speed and SuperSpeed Bus Instances	183
3.8 Additional RAM Requirements for xHCI Debug Capability	184
3.8.1 Additional TxFIFOs and RxFIFOs.....	184
3.8.2 DbC LSP Cache	185
3.8.3 DbC RAM-Based CSRs	186
3.9 RAM Requirements for USB 2.0-Only Mode	187

Chapter 4

Parameter Descriptions.....	189
------------------------------------	------------

4.1 Basic Config Parameters	191
4.2 PHY Config Parameters	203
4.3 Device Config Parameters	214
4.4 Host Config Parameters	220
4.5 Hub Config Parameters	229
4.6 Advanced Config Parameters	233

Chapter 5

Additional Configuration Details	257
---	------------

5.1 Recommendations for Selecting DWC_USB3_FAST_TAT_EN Value	258
5.2 Example Device Endpoint Mapping in Different Applications	259

Chapter 6

Signal Interfaces	263
------------------------------------	------------

6.1 System Bus Interface Overview	264
6.1.1 Master Interface	264
6.1.2 Slave Interface	265
6.2 AXI Master Interface	266
6.2.1 Burst Type	266
6.2.2 Page Boundary	267
6.2.3 Outstanding Request Limit	267
6.2.4 Transaction ID	267
6.2.5 AXI Low-Power Interface	268
6.2.6 Bus Error Address	268
6.2.7 AXI Master – Waveform Examples	269
6.3 AXI Slave Interface	273
6.3.1 Transfer ID	273
6.3.2 Burst Support	273
6.3.3 Read Vs. Write Channel	273
6.3.4 Slave Responses	273
6.3.5 AXI Low-Power Interface	274
6.3.6 Miscellaneous Info Signals	274
6.3.7 AXI Slave – Waveform Examples	274
6.4 Native Master Interface – Timing	277
6.5 Native Slave Interface – Timing	279
6.6 Native Slave Interface – Combination Request	280
6.7 Description of Synopsys Test Environment Interface Signals	281

Chapter 7

Signal Descriptions	297
--------------------------------------	------------

7.1 System Clock, Reset, and Control Signals	299
7.2 AHB Master Interface Signals	311
7.3 AHB Slave Interface Signals	315
7.4 Native Master Interface Signals	319
7.5 Native Slave Interface Signals	332
7.6 AXI Master Interface Signals	335
7.7 AXI Slave Interface Signals	345
7.8 RAM Interface Signals	357
7.9 USB 3.0 PIPE3 PHY Interface Signals	361

7.10 USB 2.0 UTMI+ Parallel Interface Signals	369
7.11 USB 2.0 UTMI+ Vendor Control Interface Signals	381
7.12 USB 2.0 UTMI+ OTG Interface Signals	382
7.13 USB 2.0 ULPI PHY Interface Signals	383
7.14 HSIC Interface Signals	387
7.15 LPM Interface Signals	388
7.16 Power Controller Interface Signals	390
7.17 Host Interface Signals	395
7.18 Hub Interface Signals	402
7.19 ACA Interface Signals	418
7.20 Miscellaneous Signal Interface Signals	420
7.21 JTAG Interface Signals	428
7.22 Synopsys Test Environment Interface Signals	430
7.23 Type-C Support Signals for Synopsys PHY	431
Chapter 8	
Hub	433
8.1 General Product Description	434
8.2 Hub Software Storage Requirement and Data Structure	437
8.2.1 Overriding the Hub Descriptor	439
8.3 Configuring the Hub	441
8.4 Hub Descriptor Storage and Initialization	441
8.5 Sharing ROM or Serial-Flash/PROM between USB 2.0 Hub and SS Hub	444
8.6 Overriding Descriptor Data through Input Ports	445
8.7 Hub RAM Requirements	445
8.8 Debug JTAG	446
8.9 Hub Controller Clock Scheme	452
8.10 Hub Controller Timing	453
8.11 Vendor Control Interface	454
8.11.1 Control Transfer Phases	454
8.11.2 Control Transfer Scenarios	455
8.11.3 Handling Vendor Command Control Transfers	462
8.11.4 Timing Diagrams	463
Chapter 9	
Battery Charger	467
9.1 Battery Charger	468
9.1.1 BC (ACA) Functions	468
9.1.2 Battery Charger Operation	469
9.1.3 BC Support for ULPI Interface	471
9.1.4 The bc_chirp_on Signal Behavior	473
9.2 Interrupt Hierarchy	478
Chapter 10	
Power Management	479
10.1 Managing Power in USB 3.0 by Using Different Power States	480
10.2 Hibernation	481
10.2.1 Overview of Hibernation	481
10.2.2 Hibernation Architecture	483

10.2.3 General Operation of Hibernation	485
10.2.4 Hibernation Memory Requirements.....	486
10.2.5 Configuring DWC_usb3 for Hibernation.....	486
10.2.6 Programming DWC_usb3 for Hibernation	487
10.2.7 Integrating DWC_usb3 with Hibernation Enabled	487
10.2.8 Hibernation Dependencies and Assumptions.....	494
10.3 Clock Gating.....	496
10.4 Reference Clock Turn Off Feature for Power Saving.....	498
10.4.1 Reference Clock Turn Off Feature Overview	498
10.4.2 Configuring the Controller to Enable Reference Clock Turn Off Feature.....	498
10.4.3 Integrating the DWC_usb3 with Reference Clock Turn Off Feature Enabled	499
10.4.4 Programming the Reference Clock Startup Time	499

Appendix A

Area, Speed, and Power.....	501
A.1 Device Area	502
A.2 Host Area Numbers.....	503
A.3 DRD Area Numbers.....	504
A.4 USB 2.0-Only Mode Area Numbers	504
A.5 Hub Area Numbers	504
A.6 Area Differences Due to Scan Ready and Clock Gating.....	505
A.7 Speed	505
A.8 Power-Compiler Clock Gating.....	505
A.9 Power Consumption	506
A.10 DFT Coverage.....	507
A.11 Performance	507
A.11.1 Host/Device/Hub Performance	508
A.12 Minimum Clock Frequencies: bus_clk, ram_clk	515
A.12.1 Recommended Configuration	516
A.12.2 Device bus_clk Frequency	516
A.12.3 Host bus_clk Frequency.....	516
A.12.4 Device ram_clk Frequency	516
A.12.5 Host ram_clk Frequency	516
A.12.6 Minimum AHB/AI and RAM Clock Frequencies for 4Gbs/8Gbs Data Rate	519
A.12.7 Minimum Clock Frequencies (MHz) for USB 2.0-Only Mode.....	520

Appendix B

Clock Domain Crossing.....	521
B.1 Types of Clock-Crossing Signals	522
B.2 CDC Implementation and Validation	523
B.2.1 Design Flow	524
B.2.2 Simulation Flow.....	525
B.2.3 Synthesis Flow	526
B.2.4 CDC Analysis Flow.....	527
B.3 Negedge-Sampled Signals and SDF Annotated Timing Check Off Flops	527
B.4 Synchronizer	527
B.4.1 Control Synchronizer (src/com/DWC_usb3_sync_ctl.v).....	528
B.4.2 Toggle Control and Data Synchronizer (src/com/DWC_usb3_sync_toggledata.v)	529
B.4.3 Data Synchronizer (src/com/DWC_usb3_sync_data.v).....	531

B.4.4 Bus Synchronizer (src/com/DWC_usb3_bussync.v)	532
B.4.5 Pulse Synchronizer (src/com/DWC_usb3_sync_pulse.v)	533
Appendix C	
High Speed Inter-Chip (HSIC) Feature	535
C.1 HSIC Operation Model	535
C.1.1 Attach Sequence	535
C.1.2 Suspend/Resume	538
C.1.3 Suspend/Remote Wakeup	540
C.2 Limitations.....	541
C.3 Operational Restrictions in HSIC Mode	541
Appendix D	
Fixed ECNs.....	543
Appendix E	
External Buffer Control	557
E.1 Overview of EBC.....	558
E.1.1 External Buffer Control for OUT Endpoint.....	558
E.1.2 External Buffer Control for IN Endpoint.....	561
E.1.3 External Buffer Control for Trace Applications	566
Appendix F	
DWC_usb3 Automotive Safety	567
F.1 Overview of Automotive Safety Feature.....	568
F.2 Automotive Safety Package Documents	569
F.3 Configuring Automotive Safety Feature	569
F.4 Programming Automotive Safety Feature	570
F.4.1 ECC Operational Model and Programming Flow	570
Appendix G	
Internal Parameter Descriptions.....	573

Revision History

Date	Version	Description
May 2018	3.30b	No updates to this document
February 2018	3.30a	<p>Preface</p> <ul style="list-style-type: none"> ■ Updated xHCI specification version (1.0 to 1.1) <p>Chapter 2, “Architecture”</p> <ul style="list-style-type: none"> ■ Updated “NYET Response” on page 124 and rearranged “DWC_usb3 Device LPM Support” on page 126 ■ Updated Figure 2-12 on page 103 and Figure 2-13 on page 104 <p>Chapter 3, “Memory Requirements”</p> <ul style="list-style-type: none"> ■ Reference of bMaxBurstSize to DEPCFG.BrstSiz in Table 3-4 on page 174 <p>Chapter 4, “Parameter Descriptions”</p> <ul style="list-style-type: none"> ■ Added the following parameters: <ul style="list-style-type: none"> DWC_USB3_LPM_SUSP_OFF, DWC_USB3_GUCTL3, DWC_USB3_SSPHY_SUPPORT_P3CPM_P4, DWC_USB3_REF_CLK_OFF, DWC_USB3_PWR_SWITCH_POLARITY, DWC_USB3_CLK_SET_FALSE_PATH, DWC_USB3_LINK_SETTINGS_INIT, DWC_USB3_LINK_LUCTL_INIT, DWC_USB3_LINK_LPTMDPDELAY_INIT, and DWC_USB3_LU1LFPSRXTIM_INIT ■ Updated the following parameters: <ul style="list-style-type: none"> DWC_USB3_GUCTL1, DWC_USB3_GUCTL2, and DWC_USB3_PIPE_32BIT_ONLY <p>Chapter 5, “Additional Configuration Details”</p> <ul style="list-style-type: none"> ■ Updated notes in Table 5-2 on page 258 <p>Chapter 7, “Signal Descriptions”</p> <ul style="list-style-type: none"> ■ Added the following signals: <ul style="list-style-type: none"> - ram_clk_out, pmgt_bus_clk_ref_off, pmgt_ref_clk_off, ram_clk_in, pmgt_ref_clk_ok, pmgt_ref_clk_req_n, and dft_clk_mux_ctlval_ref_clk - Type-C Support Signals for Synopsys PHY <p>Chapter 10, “Power Management”</p> <ul style="list-style-type: none"> ■ Added “Reference Clock Turn Off Feature for Power Saving” on page 498 <p>Appendix A, “Area, Speed, and Power”: Updated</p>

(Continued)

Date	Version	Description
February 2018 <i>Cont'd</i>	3.30a <i>Cont'd</i>	<p>(Continued)</p> <p>Appendix B, “Clock Domain Crossing”</p> <ul style="list-style-type: none"> ■ Updated “Negedge-Sampled Signals and SDF Annotated Timing Check Off Flops” on page 527 <p>Removed support for the following features:</p> <ul style="list-style-type: none"> ■ OTG ■ ADP ■ SSIC <p>Restructured the databook as follows:</p> <p>Moved Clock gating information from “Feature List” section to Chapter 10, “Power Management”</p> <ul style="list-style-type: none"> ■ Combined the Architecture Overview and Architecture Details into Chapter 2, “Architecture” ■ Moved the following sections from “Architecture Details” chapter into new chapters/appendices: <ul style="list-style-type: none"> - “Memory Requirements” and “Mapping of GTXFIFOSIZn/GRXFIFOSIZn Registers to RAM Start Address and TxFIFO/RxFIFO Depths” sections to Chapter 3, “Memory Requirements” - System Bus Interface section to “Signal Interfaces” on page 263 - External Buffer Control to Appendix E, “External Buffer Control” ■ Created Chapter 5, “Additional Configuration Details” ■ Moved the following chapters/sections to the <i>DWC SuperSpeed USB 3.0 Programming Guide</i>: <ul style="list-style-type: none"> - Registers - Device Data Structures - Device Programming Model - Host Programming Model - Programming sections from Chapter 10, “Power Management”

(Continued)

Date	Version	Description
March 2017	3.20a	<p>This is a databook-only update.</p> <ul style="list-style-type: none"> ■ Updated information on PORTSC register in Table 6-126 ■ Removed Light Host Controller Reset step from Switch Host task in Figure 11-8 ■ Fixed typos: <ul style="list-style-type: none"> - DWC_USB3_HSPHY_INTERFACE parameter value in GUSB2PHYACC_UTMI register description - ECC Error event in Table 7-8 <p>Added:</p> <ul style="list-style-type: none"> ■ Support for Automotive Safety features ■ Appendix E, "DWC_usb3 Automotive Safety" <p>"Preface"</p> <ul style="list-style-type: none"> ■ Updated information on USB 3.0 Controller Linux Driver Software in "Related Product Documentation" <p>Chapter 1, "Product Overview"</p> <ul style="list-style-type: none"> ■ Added a limitation with 16-bit UTMI and multiple ports in "Host" <p>Chapter 3, "Architecture Details"</p> <ul style="list-style-type: none"> ■ Added <ul style="list-style-type: none"> - "DWC_usb3 Device LPM Support" - "Device-Mode Transmit Path" - A note on DCFG.NUMP ■ Rearranged "Packet Threshold and Burst Features for High Latency Systems" ■ Updated: <ul style="list-style-type: none"> - "USB 2.0 Extension Transaction to Support LPM" - "Device Response to Extended Token Packet With SubPID 0011B (LPM Token)" <p>Chapter 4, "Parameters"</p> <ul style="list-style-type: none"> ■ Added: <ul style="list-style-type: none"> - "Enable Automotive ECC Generation and Checking for RAMs?" parameter (DWC_USB3_EN_ECC) - "Recommendations for Selecting DWC_USB3_FAST_TAT_EN Value" ■ Updated: <ul style="list-style-type: none"> - "Enable UTMI-16bit Fast Turnaround Operation" parameter (DWC_USB3_FAST_TAT_EN) - "Remove pipe_clk mux for 2.0 mode?" parameter (DWC_USB3_REMOVE_PIPE_CLK_MUX_FOR_20_MODE) <p>Chapter 5, "Signal Descriptions"</p> <ul style="list-style-type: none"> ■ Changed hard-coded value to 1 for xm_csysack and xm_cactive ■ Changed width of ramN_p1_wr_n, ramN_p1_wdata, ramN_p2_wr_n, ramN_p2_wdata, ramN_p1_rdata ■ Removed reference to FPGA directory in pipe3_PhyStatus_async and pipe3_PhyStatus_async descriptions

(Continued)

Date	Version	Description
February 2017 <i>Cont'd</i>	3.20a <i>Cont'd</i>	<p>Chapter 6, "Registers"</p> <ul style="list-style-type: none"> ■ Added: <ul style="list-style-type: none"> - ECC registers - GUCTL2.EN_HP_PM_TIMER and NOLOWPWRDUR register fields - DEVTEEN.ECCERREN register field - GRXTHRCFG.UsbRxPktCntSel register field - DCFG.NUMP register field ■ Updated: <ul style="list-style-type: none"> - Notes in GRXTHRCFG register - GRXTHRCFG.UsbMaxRxBurstSize description - GCTL.PRTCAPDIR register field description - GUCTL1 register; added bit 10, RESUME_OPMODE_HS_HOST - GUCTL[15] to reserved - GUCTL register field descriptions: USBHstInAutoRetryEn DTCT, DTFT - Testable field for following: <ul style="list-style-type: none"> - GUSB3RRMICTL register fields: MPHystate, DisAutoSTALL, DisScrambling, DisLupLdnTxRx - GUSB3PIPECTL register fields: PHYSoftRst, StartRxDetU3RxDet - DCTL.TSTCTL - DSTS.COREIDLE - ADPCTL register fields: EnaPrb, EnaSns, ADPEn, ADPRes, WB - GSDBG.DbgWordn - GUSB2PHYACC_ULPI register description - GUSB3PIPECTL field descriptions: HstPrtCmpl, DELAYP1TRANS - DCFG.DEVSPD description - DCTL register field descriptions: LPM_NYET_thres, KeepConnect, L1HibernationEn - DSTS.CONNECTSPD Value After Reset field <p>Chapter 7, "Device Data Structures"</p> <ul style="list-style-type: none"> ■ Added: <ul style="list-style-type: none"> - Note in "Command 6: Start Transfer (DEPSTRTRXFER)" - New event in Table 7-8 <p>Chapter 8, "Device Programming Model": Added note in "Handling L1 Event During a Transfer" and "Worst-Case Response Time"</p> <p>Appendix B, "Clock Domain Crossing"</p> <ul style="list-style-type: none"> ■ Removed the src_pulse_edge_2r clock-crossing flop from "Negedge-Sampled Signals and SDF Annotated Timing Check Off Flops" <p>Appendix D, "Fixed ECNs": Updated Table D-1</p>

(Continued)

Date	Version	Description
March 2016	3.10a	<p>Updated the following chapters:</p> <p>Chapter 4, "Parameters": Fixed typos in the following parameters:</p> <ul style="list-style-type: none"> - Starting Depth of the SS Descriptors in the ROM or Serial-Flash (in 32-bit Dwords)? - Enable Vendor Control Command Interface - Total RAM0 Depth (32 to 65536) <p>■ Chapter 5, "Signal Descriptions"</p> <ul style="list-style-type: none"> - Updated registered and synchronous information for pipe3_PowerPresent[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0] - Renamed sleep signal as power_switch_control <p>■ Chapter 6, "Registers"</p> <ul style="list-style-type: none"> - Updated description of INCRBRSTENA - Updated SOFFN with information on SuperSpeed - Updated PORTSC and PORTPMSC registers in Table 6-126 - Added PORTHLPNC register to Table 6-126 <p>Chapter 7, "Device Data Structures"</p> <ul style="list-style-type: none"> ■ Added information on race condition in "Definitions" ■ Added a note on Set Periodic Parameters in Table 7-2 ■ Added a note on GUCTL2[Rst_actbitlater] in "Command 8: End Transfer (DEPENDXFER)" <p>Chapter 8, "Device Programming Model"</p> <ul style="list-style-type: none"> ■ Added BUFSIZ requirements for setup stage of control transfer in "Buffer Size Rules and Zero-Length Packets" ■ Updated Table 8-6 with additional requirements while adding a new endpoint ■ Added a note on stale TRB transfer in "Adding Intervals to a Transfer" <p>Chapter 12, "Hibernation"</p> <ul style="list-style-type: none"> ■ Replaced debug_u2pmu[0] with debug_u2pmu[DWC_USB3_NUM_U2_ROOT_PORTS*DWC_USB3_DEBUG_U2WAKEUP_W] in "Integration to Allow Hibernation to be Disabled" <p>Chapter 13, "SuperSpeed InterChip (SSIC) Feature": Added a new section "Recommendations on M-PHY Attribute Values"</p> <p>Appendix B, "Clock Domain Crossing": Removed references to VC CDC because it is no longer supported</p>

(Continued)

Date	Version	Description
May 2015	3.00a	<p>Updated the following chapters:</p> <ul style="list-style-type: none"> ■ Chapter 4, "Parameters" ■ Chapter 5, "Signal Descriptions" <ul style="list-style-type: none"> - Merged "Signal Interfaces" chapter with "Signal Descriptions" chapter - Updated "Description of Synopsys Test Environment Interface Signals" ■ Chapter 6, "Registers" <p>Chapter 1, "Product Overview"</p> <ul style="list-style-type: none"> ■ Fixed an error in the databook on clock gating information in USB 2.0 mode ("Clock Gating") <p>Chapter 3, "Architecture Details"</p> <ul style="list-style-type: none"> ■ Added a note on setting GRXTHRCFG.USBRxPktCntSel in "Device-Mode Receive Path" <p>Chapter 7, "Device Data Structures"</p> <ul style="list-style-type: none"> ■ Added value 9 for the field [9:4] (TRBCTL) and value 4'hF for field [31:28] (TRBSTS) in Table 7-1 ■ Added clarification to the note in "Commands 4 and 5: Set Stall and Clear Stall (DEPSSTALL, DEPCSTALL)" ■ Updated field [11:8] for values 14, 8, 6, and 4 in Table 7-8 <p>Chapter 8, "Device Programming Model"</p> <ul style="list-style-type: none"> ■ Added the following new sections: <ul style="list-style-type: none"> - "Initialization after U3 Exit" - "Interrupts" ■ Updated "Worst-Case Response Time" <p>Chapter 13, "SuperSpeed InterChip (SSIC) Feature"</p> <ul style="list-style-type: none"> ■ Updated the following sections: <ul style="list-style-type: none"> - "Functional Changes for SSIC within DWC_usb3" - "Configuration Requirements for SSIC Feature" - "Clock Requirement for the SSIC Feature"

(Continued)

Date	Version	Description
May 2015 <i>Cont'd</i>	3.00a <i>Cont'd</i>	<p>(Continued)</p> <ul style="list-style-type: none"> - “AutoEXIT_H8: Automatic Exit from HIBERN8 to LS MODE State” - “AutoROM_RRAP: Automatic Remote Register Access Protocol in LS MODE” ■ “Handling Disconnect Scenario as a SSIC Device” ■ Added a note on non-Synopsys M-PHY usage in “Organization of the SSIC ROM Locations” ■ Added 4 RRAP commands to Table 13-9, Table 13-10, and Table 13-11 ■ Updated Table 13-28 for pa_rx (3'b100) ■ Updated section “Analog Loopback Testing” ■ Updated step 10 in “SSIC Host Core Low Power Entry Flow” ■ Added the following subsections: <ul style="list-style-type: none"> - “SSIC Multi-Port Host Support” - “Switching from SSIC Mode to PIPE Mode” - “SSIC Multi-Port and ram_clk_sel Feature” - “SSIC Multi-Port Programming Flow Requirements” - “Support for Synopsys MPHY GLOBALHIBERN8 (PLL OFF) Mode” - “SSIC PSI Dwords for U3 Host Ports” - “Cold Attach Status for Host SSIC Ports” <p>Appendix A, “Area, Speed, and Power”</p> <ul style="list-style-type: none"> ■ Added estimated area for multi-port configurations supported by the SSIC core (Table A-9) <p>Appendix B, “Clock Domain Crossing”</p> <ul style="list-style-type: none"> ■ Updated examples in “Synthesis Flow” ■ Updated “CDC Analysis Flow” <p>Appendix D, “Fixed ECNs”</p> <ul style="list-style-type: none"> ■ Updated the table on fixed xHCI ECNs (Table D-1 and Table D-3)
November 2014	2.90a- Ip01	<ul style="list-style-type: none"> ■ Updated “Device-Mode Receive Path” to include the new register field (GRXTHRCFG.ResvISOCOUTSpc) ■ Added a new register field (ResvISOCOUTSpc) in GRXTHRCFG register.

(Continued)

Date	Version	Description
September 2014	2.90a	<ul style="list-style-type: none"> ■ Renamed Chapter 7 as “Device Data Structures”, and moved the information on commands and content of event buffers from Chapter 6, “Registers” to Chapter 7, “Device Data Structures” (see “Device Command Structure” and “Device Event Buffer Structure”) ■ Moved the following sections to Chapter 5, “Signal Descriptions”: <ul style="list-style-type: none"> - SSIC signals from SSIC chapter - Power Controller Interface signals from Power Management chapter ■ Moved the following sections to Chapter 3, “Architecture Details” <ul style="list-style-type: none"> - FIFO and DMA-related sections from Registers chapter - System bus-related sections from Signals chapter and Registers chapter ■ Implemented Auto-extraction (signal, parameter, and register descriptions and details are extracted from the respective RTL files): <ul style="list-style-type: none"> - Replaced tables in Chapter 4, “Parameters” with auto-extracted tables - Replaced tables in Chapter 5, “Signal Descriptions” tables with auto-extracted tables; Added Chapter 5, “Signal Interfaces” chapter which now has all the interface-level details - Replaced tables in Chapter 6, “Registers” with auto-extracted tables <p>Chapter 1, “Product Overview”</p> <ul style="list-style-type: none"> ■ Added more information on GCTL.RAMClkSel in “Speed and Clock Requirements” ■ Updated “Known Issues” <p>Chapter 3, “Architecture Details”</p> <ul style="list-style-type: none"> ■ Removed reference to DWC_USB3_ENABLE_LPM parameter from “Core Handshake Response to LPM Transaction” ■ Added a new section “External Buffer Control for Trace Applications” <p>Chapter 4, “Parameters”</p> <ul style="list-style-type: none"> ■ Updated description of DWC_USB3_EN_FPGA, DWC_USB3_GCTL_INIT, and DWC_USB3_HOST_RAM_CLK_FREQ_PERF ■ Updated possible values for DWC_USB3_RMMI_DATA_WIDTH

(Continued)

Date	Version	Description
September 2014 <i>Cont'd</i>	2.90a <i>Cont'd</i>	<ul style="list-style-type: none"> ■ Removed “BMU-PTL Source and Sink Buffer Type” parameter Chapter 5, “Signal Descriptions” ■ Updated description of suspend_clk, bus_clk_early, and vcc_reset_n ■ Updated clock accuracy requirement for ref_clk to 50 ppm Chapter 6, “Registers” ■ GRXTHRCFG(#n)[15]: Updated access type to Reserved ■ GUSB2PHYCFG(#n)[29]: Added a note to ULPI_LPM_WITH_OPMODE_CHK on setting this bit to 0 for Synopsys PHY. ■ GUSB2PHYCFG(#n): Added a note to Enable utmi_sleep_n and utmi_I1_suspend_n (EnbISlpM) and Suspend USB2.0 HS/FS/LS PHY (SusPHY) ■ GUSB3PIPECTL(#n)[29]: Added a note to P3 OK for U2/SSInactive ■ GUSB3PIPECTL(#n)[10] and [11]: Updated description ■ GCTL[10]: Removed reference to DWC_USB3_ENABLE_LPM parameter from SOFITPSYNC ■ GCTL[7:6]: Updated description ■ GUCTL1: Added new fields ■ SEVTn[4]: Added note on M-PHY State Machine Changed Event (MPHYStChng) stating it is only for debug purpose ■ GSCFG: Added more information in ls_to_hs_exit time ■ GSDBG[27:25] and GSDBG[24]: Updated description ■ DSTS[22]: Updated description Chapter 7, “Device Data Structures” ■ Added a new scenario to “Transfer Resource Usage and Transfer State” ■ Added a new field [14] to command 1 (Table 7-3) ■ Updated description of Parameter 2 in command 1 (Table 7-3) Chapter 8, “Device Programming Model” ■ Added a new scenario to “Transfer Resource Usage and Transfer State” Chapter 12, “Hibernation” ■ Added information on DCTL.KeepConnect bit in “Hibernation While Connected” ■ Updated step 8 in “Software or PHY-Initiated Wakeup While Connected” for USB 2.0 mode ■ Moved PMUs inside DWC_usb3 controller Chapter 13, “SuperSpeed InterChip (SSIC) Feature” ■ Updated “Host-Initiated Disconnect” Appendix A, “Area, Speed, and Power” ■ Updated “SSIC Numbers” Appendix D, “Fixed ECNs” ■ Updated the table on fixed xHCI ECNs (Table D-3)

(Continued)

Date	Version	Description
February 2014	2.80a	<p>Chapter 1, "Product Overview"</p> <ul style="list-style-type: none"> ■ Updated "Known Issues" (fourth bullet under Host section). ■ Added STAR 9000714241 to "Known Issues" ■ Added STAR 9000714246 to "Known Issues" ■ Updated MAC clock domains in "General Features" ■ Added bullet item at end of section <p>Chapter 2, "Architecture Overview"</p> <ul style="list-style-type: none"> ■ Updated bullet "MAC3 Clock" in "Clock Domains" ■ Updated third bullet in section "Clock Domains" <p>Chapter 3, "Architecture Details"</p> <ul style="list-style-type: none"> ■ Removed information on two-port configuration in "Synchronous Static RAM Requirement" <p>Chapter 4, "Parameters"</p> <ul style="list-style-type: none"> ■ Added a note to "In Host Mode, Enable USB2.0 Suspend during Disconnect?" configuration parameter. ■ Replaced Gear speed for SSIC ports parameter with Max gear speed for SSIC ports in "Max gear speed for SSIC ports" ■ Changed default value for "Global USB3 RMMI Control register's Power-On Initialization Value" <p>Chapter 5, "Signal Descriptions"</p> <ul style="list-style-type: none"> ■ Updated xm_csysreq signal in Table 5-9. (This signal is currently not supported and must be hardwired to 1 (meaning non-active.)) ■ Updated xs_csysreq signal in Table 5-10. (This signal is currently not supported and must be hardwired to 1 (meaning non-active.)) ■ Updated code examples in "Description of Synopsys Test Environment Interface Signals". ■ Added a new signal "sb2m3l_ssic_gear [DWC_USB3_NUM_U3_ROOT_PORTS*2-1:0]". Item numbers for rows Table 5-34 were renumbered.

(Continued)

Date	Version	Description
February 2014 <i>Cont'd</i>	2.80a <i>Cont'd</i>	<p>Chapter 6, "Registers"</p> <ul style="list-style-type: none"> ■ GUCTL[8:0]: Updated description for 156.25MHz clock in "Global User Control Register: GUCTL" ■ GUCTL[12]: Updated description of External Extended Capability Support Enable (ExtCapSuptEN) in Table 7-14 ■ SEVTn[7]: Updated description of RRAP Write Command from the Test Equipment to the RRAP TEST_MODE Register Event (MPHY.TestEn) in Table 7-81 ■ SEVTn[4]: Updated description of "M-PHY State Machine Changed Event (MPHYStChng)" in Table 7-81 ■ Added SSIC tDPRResponse time[5:4] to "Global SSIC Configuration Register (GSCFG)" ■ Added ls_to_hs_exit_time [11:6] to "Global SSIC Configuration Register (GSCFG)" ■ Added rxcfgupd_on_h8exit[12] to "Global SSIC Configuration Register (GSCFG)" ■ ADPEVT[15:0]: Updated description of (RAMP TIME (RTIM)). Reversed order of scaledown ramp_timeout values. ■ DEPCFG: Updated Parameter 1 [30] of Command 1 as reserved (Table 6-73) <p>Chapter 7, "Device Data Structures"</p> <ul style="list-style-type: none"> ■ Updated 5th bullet item under "Definitions" , list of Device mode TRB characteristics <p>Chapter 8, "Device Programming Model"</p> <ul style="list-style-type: none"> ■ Removed the column "Bulk-based" from Table 8-12 <p>Chapter 13, "SuperSpeed InterChip (SSIC) Feature"</p> <ul style="list-style-type: none"> ■ Revised first paragraph in "SSIC PHY Adapter (PA) Functions" ■ Updated "SSIC Host Port ITP Generation" and "Inaccuracy in ITS.Delta Value" ■ Removed Gear Speed for SSIC Ports parameter from Table 14-4 ■ Added the following new sections: <ul style="list-style-type: none"> - "Clock Requirement for the SSIC Feature" - "Clock Selection and Programming Reference" - Analog Loopback Testingunder "Programming Flow for M-PHY Test" - "SSIC Peak Throughput Calculation" ■ Added new signal ssic_soc_pa_clk_freq[1:0] to Table 13-7 ■ Updated description for ssic_soc_pa_clk in Table 13-7 ■ Updated "RRAP Target" ■ Updated "Warm Reset" ■ Updated "M-PHY Test Mode Entry" and "M-PHY Test Mode of Operation" (under Receive Burst Testing and Analog Loopback Testing sections)

(Continued)

Date	Version	Description
February 2014 <i>Cont'd</i>	2.80a <i>Cont'd</i>	<ul style="list-style-type: none"> ■ Updated “M-PHY Test Exit” ■ Updated “Clock Gating” Appendix A, “Area, Speed, and Power” ■ Updated “Power Consumption”
December 2013	2.70a	<p>Chapter 1, “Product Overview”</p> <ul style="list-style-type: none"> ■ Updated the third example under AXI Master Interface (see “Application Interface Features”) ■ Updated “High Speed Inter-Chip (HSIC) Features” ■ Updated “Known Issues” <p>Chapter 3, “Architecture Details”</p> <ul style="list-style-type: none"> ■ Removed information on two-port configuration in “Synchronous Static RAM Requirement” ■ Updated requirements of “External Buffer Control” ■ Updated the following EBC diagrams for better accuracy: <ul style="list-style-type: none"> - Figure 3-60 - Figure 3-61 - Figure 3-63 - Figure 3-64 <p>Chapter 4, “Parameters”</p> <ul style="list-style-type: none"> ■ Added a note to In Host Mode, Enable USB2.0 Suspend during Disconnect? configuration parameter ■ Updated description of the following parameters: <ul style="list-style-type: none"> - Enable receiver detection in PHY power state P3? - Number of Interrupters (1-16) - Enable USB 2.0-only mode? ■ Added Enable Separate Descriptor Queues ■ Updated dependencies of the following configuration parameters: <ul style="list-style-type: none"> - Enable HSIC Support for USB 2.0 Ports? - Global Receive Threshold Configuration Register (GRXTHRCFG) Power-On Initialization Value (0x0-ffffffff) ■ Updated “Example Device Endpoint Mapping in Different Applications”

(Continued)

Date	Version	Description
December 2013 <i>Cont'd</i>	2.70a <i>Cont'd</i>	<p>(Continued)</p> <p>Chapter 5, "Signal Descriptions"</p> <ul style="list-style-type: none"> ■ System Clock, Reset, and Control Signals: <ul style="list-style-type: none"> - Added a note on suspend_clk when hibernation is enabled - Updated description of bus_clk_early ■ AXI Master Interface Signals signals: Updated the note on output signal value of xm_awprot[2:0] and xm_arprot{2:0} signals ■ Updated "Transaction ID" ■ HSIC Interface Signals: Updated description of if_select_hsic [`DWC_USB3_NUM_U2_ROOT_PORTS-1:0] ■ Updated sub-states of RX_DET in Table 5-37 <p>Chapter 6, "Registers"</p> <ul style="list-style-type: none"> ■ GCTL[16]: Updated description of U2RSTECN ■ GCTL[7:6]: Updated description of RAM Clock Select (RAMClkSel) ■ GCTL[5:4]: Updated description of Scale-Down Mode (ScaleDown) in SuperSpeed mode ■ GUCTL1[17:15]: Added new bits (PARKMODE_DISABLE_SS, PARKMODE_DISABLE_HS, and PARKMODE_DISABLE_FSLs) ■ GDBGLTSSM[30]: Added a new field (Rx Elec Idle (RxEleclidle)) ■ GUSB2PHYCFGn[13:10]: Added a note that this field (USB 2.0 Turnaround Time (USBTrdTim)) is valid only in device mode ■ GUSB2PHYCFGn[26]: Updated description of INV_SEL_HSIC ■ DCFG[24]: Added a note on Stop On Disconnect (StopOnDisconnect) ■ OSTS[13:12]: Corrected typos in DevRunStp and xHciRunStp ■ ADPEVT[26]: Updated description of ADPEVTInfo[2] ■ GPMSTS[27:17]: Updated from [29:17] to [27:17] in "Global Power Management Status Register: GPMSTS". ■ Added tables listing signals that make up DbgWordn (Table 7-156, Table 7-157, Table 7-158, Table 7-159, Table 7-160, and Table 7-161) <p>Chapter 8, "Device Programming Model"</p> <ul style="list-style-type: none"> ■ Added a note on Vendor Device Test in "Event Buffers" <p>Chapter 13, "SuperSpeed InterChip (SSIC) Feature": Added a new chapter on SuperSpeed InterChip (SSIC) feature.</p> <p>Appendix C, "High Speed Inter-Chip (HSIC) Feature"</p> <ul style="list-style-type: none"> ■ Updated "Attach Sequence" and "Limitations"

(Continued)

Date	Version	Description
July 2013	2.60a	<p>Added information on High Speed Inter-Chip (HSIC) feature:</p> <ul style="list-style-type: none"> ■ Added the following configuration parameters: <ul style="list-style-type: none"> - Enable HSIC Support for USB 2.0 Ports? - Enable Filters for UTMI and Overcurrent-related inputs from the PHY in PMU? ■ Added “HSIC Interface” ■ GUSB2PHYCFGn: Added two new fields: <ul style="list-style-type: none"> - HSIC_CON_WIDTH_ADJ - INV_SEL_HSIC ■ Increased the width of npi_usb2phy_config[18:0] signal by two bits ■ Added Appendix C, “High Speed Inter-Chip (HSIC) Feature” ■ Updated Appendix D, “Fixed ECNs” <p>Added information on USB 2.0-only mode:</p> <ul style="list-style-type: none"> ■ Added “RAM Requirements for USB 2.0-only Mode” ■ Added the configuration parameter – Enable USB 2.0-only mode? ■ Updated the following parameters: <ul style="list-style-type: none"> - Enable USB 2.0 or 3.0 OTG Capability? - Master Bus (DMA Bus) Interface Type - Number of RAMs - Global USB2 PHY Configuration Register's (GUSB2PHYCFG) Power-On Initialization Value (0x0-ffffffff) - Enable xHCI Debug Capability? - Number of USB3 Root Hub Ports (1-15) - Number of Cached Endpoints for Each SuperSpeed USB Instance (1-32) - Number of Cached TRBs for Each Cached SuperSpeed Endpoint (2-126) - Size of the SuperSpeed RxFIFO in Number of 1024-byte Packets - Per SS Bus-Instance (1-32) - Size of the SuperSpeed TxFIFO in Number of 1024-byte Packets - Per SS Bus-Instance (1-32) - Enable receiver detection in PHY power state P3? - Global USB3 PIPE Control Register's (GUSB3PIPECTL) Power-On Initialization Value (0x0-ffffffff) - Enable Pipelining on the Pipe Interface - Pipe3_RxTermination signal hardware reset value? - PIPE Interface is always 32-bit?

(Continued)

Date	Version	Description
July 2013 <i>Cont'd</i>	2.60a <i>Cont'd</i>	<p><i>(Continued)</i></p> <ul style="list-style-type: none"> ■ Added notes on the following signals in USB 2.0-only mode: <ul style="list-style-type: none"> - hub_port_overcurrent[N-1:0] - hub_vbus_ctrl[N-1:0] - host_num_u3_port[3:0] - host_u3_port_disable['DWC_USB3_NUM_U3_ROOT_PORTS-1:0] - dft_clk_mux_ctlval_rx_mx_pcclk[1:0] - dft_clk_mux_ctlval_tx_mx_pcclk[1:0] - dft_clk_mux_ctlval_mac3_clk[2:0] - suspend_clk ■ Updated the following signals: <ul style="list-style-type: none"> - utmi_sleep_n - utmi_l1_suspend_n ■ GUCTL1: Added the following new fields: <ul style="list-style-type: none"> - L1_SUSP_THRLD_EN_FOR_HOST - L1_SUSP_THRLD_FOR_HOST - Host ELD Enable (HELDEn) - Host Parameter Check Disable (HParChkDisable) - Updated Table 9-1 (Power-On or Soft Reset Register Initialization) ■ Added “USB 2.0-only Mode Area Numbers” <p>Removed information on NPI (Non-Processor Interface)</p> <p>Preface</p> <ul style="list-style-type: none"> ■ Added details on product codes for DWC_usb3 (“coreConsultant User Guide, Synopsys, Inc. (included with the coreConsultant tool) https://www.synopsys.com/dw/doc.php/doc/coretools/latest/coreconsultant_user.pdf) <p>Chapter 1, “Product Overview”</p> <ul style="list-style-type: none"> ■ Updated “Known Issues” ■ Added a bullet on single-port support only in host mode under “USB 2.0 OTG Features” <p>Chapter 3, “Architecture Details”</p> <ul style="list-style-type: none"> ■ Fixed an error in “FIFO Configuration for Host Mode” and “FIFO Configurations for DRD Mode” by updating the number of instances to two HS instead of one. ■ Updated “External Buffer Control”

(Continued)

Date	Version	Description
July 2013 <i>Cont'd</i>	2.60a <i>Cont'd</i>	<p>(Continued)</p> <p>Chapter 4, “Parameters”</p> <ul style="list-style-type: none"> ■ Added the “Preserve logic_analyzer_trace mux select during reset for hardware debug?” configuration parameter ■ Updated the value range for Master ID Port Width in Table 4-1 ■ Updated label for the DWC_USB3_HOST_NUM_PERIODIC_EP configuration parameter (Number of Periodic Endpoints (32-510)) ■ Added definition of DWC_USB3_MAX_ACTIVE_U2_INSTANCES in the following: <ul style="list-style-type: none"> - Global Host FIFO DMA Priority Register's (GDMAHLRATIO) TX SS:HSFSL Ratio Power-On Initialization Value (0x0-1f) - Global Host FIFO DMA Priority Register's (GDMAHLRATIO) RX SS:HSFSL Ratio Power-On Initialization Value (0x0-1f) ■ Table 4-4: Added more recommendations on setting the parameter Number of Cached TRBs per Transfer (2-32) ■ Updated the area numbers for the following parameters in Table 4-5 <ul style="list-style-type: none"> - Number of USB2 Root Hub Ports (1-15) - Number of USB3 Root Hub Ports (1-15) - Number of SuperSpeed USB Bus-Instances (1-4) - Number of High-Speed USB Bus-Instances (1-4) ■ Updated the following in Table 4-7 <ul style="list-style-type: none"> - Minimum ram_clk frequency information in Always use posedge-clock synchronizers? <p>Chapter 5, “Signal Descriptions”</p> <ul style="list-style-type: none"> ■ System Clock, Reset, and Control signals: Updated “dev_usb_outep_pkt_buff_avail[15:0]” and “dev_usb_inep_pkt_buff_avail[15:0]” signals ■ LPM Signal Interface: Updated information on the assertion and deassertion of the utmi_l1_suspend_n signal ■ Miscellaneous Signal Interface: Added more information on “logic_analyzer_trace[63:0]” ■ Added the “rxdet_poll_fail_us”, “hub_desc_rd_done”, and “mphy_cmd_state[2:0]” to “Synopsys Test Environment Interface”, created Table 5-34 to show example debug width, and changed the order of signals listed in Table 5-35 to match the RTL snippet.

(Continued)

Date	Version	Description
July 2013 <i>Cont'd</i>	2.60a <i>Cont'd</i>	<p>(Continued)</p> <p>Chapter 6, "Registers"</p> <ul style="list-style-type: none"> ■ Updated the reset values of all Reserved and Reserved1 fields to X. ■ Corrected the starting address for Debug Capability Structure as Addr2 + 10h in "xHCI 1.0 Host Registers Map": <p>Global Registers</p> <ul style="list-style-type: none"> ■ Added two new registers: <ul style="list-style-type: none"> - "Global General Purpose Input/Output Register (GGPIO)" - "Global Error Injection Control Register 2 (GERRINJCTL_2)" ■ GCTL[1]: Updated access type of GblHibernationEn ■ GCTL[10]: Updated SOFITPSYNC ■ GUSB2PHYCFGn[30]: Removed a note on the requirement of setting this bit based on the ref_clk counter, GCTL.SOFITPSYNC=1, or GFLADJ. GFLADJ_REFCLK_LPM_SEL=1, and updated the reset value to include DWC_USB3_GUSB2PHYCFG_INIT[30] ■ GUSB2PHYACn[15:8]: Updated 6-bit address to 8-bit address for ULPI mode ■ GRXTHRCFG: Removed the following note because it is no longer valid <ul style="list-style-type: none"> ■ "All the fields in GRXTHRCFG register are valid only in Host mode." ■ GUCTL[14]: Added a note stating that this bit is also applicable in device mode ■ GUCTL[20:18]: Updated this field (PSQExtrResSp) as Reserved ■ GDBGFIFO SPACE: Increased the width of FIFO/Queue Select (or) Port-Select field to nine bits. <p>Device Registers</p> <ul style="list-style-type: none"> ■ DCTL[23:20]: Updated the reset value of LPM_NYET_thres (when LPM is enabled) ■ DCTL[[28:24]: Added a note on setting HIRD Threshold (HIRD_Thres) field when operating in SS mode ■ DCFG[2:0]: Updated the reset value of Device Speed (DevSpd) for USB 2.0-only mode ■ Updated "Commands 4 and 5: Set Stall and Clear Stall (DEPSSTALL, DEPCSTALL)" ■ Updated Run SoC Bus LoopBack Test to be more clear ■ Updated Burst Size (BrstSiz) in Table 6-73 ■ DALEPENA[[31:0]: Updated "USB Active Endpoints (USBActEP)" to clarify that hardware clears these bits for all endpoints (except EP0-OUT and EP0-IN) after USB reset ■ DGCM: Updated the access type of Command Type (CmdTyp) and Command Interrupt on Complete (CmdIOC) as R_W ■ DEPCMDn[11]: Added ClearPendIN option for HighPriority/ForceRM (HiPri_ForceRM)

(Continued)

Date	Version	Description
July 2013 <i>Cont'd</i>	2.60a <i>Cont'd</i>	<p>(Continued)</p> <p>Host Registers</p> <ul style="list-style-type: none"> ■ HCSPARAMS2: Added information on Max Scratchpad Bufs Hi ■ HCCPARAMS: Added two new fields: <ul style="list-style-type: none"> - Stopped EDLTA Capability (SEC) - Stopped - Short Packet Capability(SPC) <p>Chapter 7, "Device Data Structures"</p> <ul style="list-style-type: none"> ■ Added another option "TransferInProgress" in the description of the "TRB Status (TRBSTS)" field in Table 7-1 <p>Chapter 8, "Device Programming Model"</p> <ul style="list-style-type: none"> ■ Removed the requirement of soft reset after power-on reset ("Device Power-On or Soft Reset") ■ Added a requirement on future microframe time in "Starting a Transfer" ■ Added another scenario when the endpoint is considered to be paused in "Low Power Operation" <p>Chapter 9, "Host Programming Model"</p> <ul style="list-style-type: none"> ■ Updated information on NumP field in ACK TP in "Bursting Behavior for OUT and IN Transfers" <p>Chapter 12, "Hibernation"</p> <ul style="list-style-type: none"> ■ Updated bus_clk to bus_clk_early in "DWC_usb3 Signals Related to Power Management" ■ Updated "Software or PHY-Initiated Wakeup While Connected" <p>Appendix A, "Area, Speed, and Power"</p> <ul style="list-style-type: none"> ■ Updated Appendix A.13, "Minimum Clock Frequencies: bus_clk, ram_clk" ■ Updated the area numbers for the following: <ul style="list-style-type: none"> - "Device Area" - "Host Area Numbers" - "DRD/OTG Area Numbers" - "SSIC Numbers" <p>Appendix B, "Clock Domain Crossing"</p> <ul style="list-style-type: none"> ■ Added another negedge flip-flop to "Negedge-Sampled Signals and SDF Annotated Timing Check Off Flops" <p>Appendix D, "Fixed ECNs"</p> <ul style="list-style-type: none"> ■ Updated the table on fixed xHCI ECNs (Table D-3)

(Continued)

Date	Version	Description
November 2012	2.50a	<ul style="list-style-type: none"> ■ Modified and removed references to unsupported RX byte alignment as this feature is now supported in the core. The following sections are modified: <ul style="list-style-type: none"> - Removed reference to unsupported RX byte alignment from “Known Issues” and Table 7-1 - Modified references to unsupported RX byte alignment from “xHCI Debug Capability” ■ All rid_* signals are now prefixed with bc (for example, bc_rid_*). In addition, the chirp_on signal is now bc_chirp_on ■ Fixed a databook error: Updated utmiotg_bvalid as utmisrp_bvalid ■ Modified and removed references to xHCI 0.96 as it is no longer supported. <ul style="list-style-type: none"> - Removed the DWC_USB3_HC_HCIVERSION parameter from Table 4-5 - Removed the xHCI 0.96 Host Registers Map from Chapter 6, “Registers” - Removed the xhci_revision signal from “Host Interface” - Updated the reset value of HCIVERSION (see Table 7-92) - Updated “Structural Parameters 2 (HCSPARAMS2)” and “Capability Parameters (HCCPARAMS)” ■ LPM Errata feature support is currently a Limited Customer Availability feature (no longer an Early Adopter feature) <p>Chapter 1, “Product Overview”</p> <ul style="list-style-type: none"> ■ Added information on Device, Host, and DRD/OTG usage in “General Product Description” ■ Added information on UTMI(ULPI)_CLK requirements in “Speed and Clock Requirements” ■ Added information about DWC_usb3 behavior based on GCTL.SOFTITPSYNC and GFLADJ.GFLADJ_REFCLK_LPM_SEL bits in “Speed and Clock Requirements” <p>Chapter 2, “Architecture Overview”</p> <ul style="list-style-type: none"> ■ Updated the LSP and pwrn blocks in Figure 2-3 <p>Chapter 3, “Architecture Details”</p> <ul style="list-style-type: none"> ■ Updated the requirements of the EBC feature in “External Buffer Control” ■ Fixed a databook error: The width of the ports is 8/16/32 and not 6/16/31 in Figure 3-5, Figure 3-6, and Figure 3-7

(Continued)

Date	Version	Description
November 2012 <i>Cont'd</i>	2.50a <i>Cont'd</i>	<p>(Continued)</p> <ul style="list-style-type: none"> ■ Updated the note on USB and System Bus sub-packet threshold mode in “Total Device RAM Requirement” <p>Chapter 4, “Parameters”</p> <ul style="list-style-type: none"> ■ Changed the valid range of DWC_USB3_SIDWIDTH from 1-16 to 1-32 in Table 4-1 ■ Updated dependencies and added a note on Hibernation for the following parameters in Table 4-1: <ul style="list-style-type: none"> - DWC_USB3_EN_AD - DWC_USB3_EN_BC ■ Added information on bits 29 and 30 in DWC_USB3_GUSB2PHYCFG_INIT (see Table 4-2) ■ Updated area information for the following parameters in Table 4-5: <ul style="list-style-type: none"> - “Number of USB2 Root Hub Ports (1-15)” - “Number of USB3 Root Hub Ports (1-15)” - “Number of SuperSpeed USB Bus-Instances (1-4)” - “Number of High-Speed USB Bus-Instances (1-4)” ■ Changed the minimum value of DWC_USB3_DBC_TRBS_PER_TRANSFER configuration parameter from 2 to 16 in Table 4-5 ■ Added the following parameters to the Table 4-7 <ul style="list-style-type: none"> - DWC_USB3_EN_SYNC_ALL_POSEdge - DWC_USB3_DEV_RAM_CLK_FREQ_PERF - DWC_USB3_DEV_RAM_CLK_FREQ_FUNC - DWC_USB3_HOST_RAM_CLK_FREQ_PERF - DWC_USB3_HOST_RAM_CLK_FREQ_FUNC ■ Updated description of DWC_USB3_RAM_CLK_TO_BUS_CLK in Table 4-7 ■ Updated DWC_USB3_DEV_EN_SCATTER_PACKETS_OF_8_TO_15_TRBS in Table 4-7: Extended support in Debug Capability mode <p>Chapter 5, “Signal Descriptions”</p> <ul style="list-style-type: none"> ■ Modified information for the following signals: <ul style="list-style-type: none"> - “ref_clk” - “fladj_30mhz_reg[5:0]” - “utmi_suspend_com_n” - “utmi_l1_suspend_com_n” - “debug_mclk_usof_number[18:0]” ■ Added the “mphy_cmd_state[2:0]” debug signal to “Synopsys Test Environment Interface”

(Continued)

Date	Version	Description
November 2012 <i>Cont'd</i>	2.50a <i>Cont'd</i>	<p>(Continued)</p> <p>Chapter 6, “Registers”</p> <ul style="list-style-type: none"> ■ Added reset information for all registers (see “Register Resets”) <p>Global Registers</p> <ul style="list-style-type: none"> ■ Added three new registers: <ul style="list-style-type: none"> - “Global Status Register (GSTS)” - “Global User Control Register 1 (GUCTL1)” - “Global Frame Length Adjustment Register (GFLADJ)” ■ Fixed the following document errors in “Global Registers Map”: <ul style="list-style-type: none"> - GSBUSCFG0: Fixed reset values of DescBigend and DatBigEnd - GUSB2PHYCFGn[16]: Updated to match the Table 7-41 - GUSB3PIPECTLn[30:28]: Updated to match the Table 7-44 ■ GSBUSCFG0[0]: Updated description of INCRBurstEna in Table 7-5 ■ GCTL[16]: Added a note stating that GCTL[16] (U2RSTECN) is valid only in device mode (see Table 7-9) ■ GCTL[10]: Modified bit description for SOFITPSYNC bit in “Global Core Control Register (GCTL)” ■ GSTS[1:0]: Updated the access type and reset value, and removed the DRD option (see Table 7-10) ■ GUCTL[31:22]: Modified bit description for REFCLKPER bit in “Global User Control Register (GUCTL)” ■ GHWPARAMS1[31]: Updated the reset value to be coreConsultant parameter instead of 1'b0 (see Table 7-21) ■ GHWPARAMS3[9:8] is Reserved (see Table 7-24) ■ GHWPARAMS4[27:24]: Updated the field to show DWC_USB3_BMU_PTL_DEPTH-1 instead of DWC_USB3_BMU_PTL_DEPTH (see Table 7-25) ■ GHWPARAMS4[16:13]: Updated the description to show DWC_USB3_HIBER_SCRATCHBUFS is valid only in the device mode (see Table 7-25) ■ GHWPARAMS6[6]: Updated as DWC_USB3_EN_DBG_PORTS (see Table 7-27) ■ GHWPARAMS6[11]: Updated reset value of this field to DWC_USB3_EN_OTG != 0 & DWC_USB3_MODE = 2 (see Table 7-27) ■ GUSB2PHYCFGn[30] and [29]: Added two new fields (U2_FREECLK_EXISTS and ULPI_LPM_WITH_OPMODE_CHK) in Table 7-41 ■ GUSB2PHYCFGn[9]: Added a new field XCVRDLY (see Table 7-41) ■ GDBGLSPMUX[15]: Added a new field EnDbC (see Table 7-36) ■ Added more information on “Global FIFO Size Registers” and “Global DMA Priority Registers”

(Continued)

Date	Version	Description
November 2012 <i>Cont'd</i>	2.50a <i>Cont'd</i>	<p>(Continued)</p> <ul style="list-style-type: none"> ■ Updated the offset range of GEVNTADRn, GEVNTSIZn, and GEVNTCOUNTn (see “Global Event Buffer Address Register (GEVNTADRn)”, “Global Event Buffer Size Register (GEVNTSIZn)”, and “Global Event Buffer Count Register (GEVNTCOUNTn)”) ■ DEVT[11:8]: Value of 6 indicates an USB Suspend Entry Event (see Table 6-53) ■ DGCMD[8] and DEPCMDn[8]: Added a note on dependency of this bit on DCTL.RunStop (see “Device Generic Command Register (DGCMD)”, and “Device Physical Endpoint-n Command Register (DEPCMDn)”) <p>xHCI Registers</p> <ul style="list-style-type: none"> ■ SUPTPRT2_DW2[20]: Added a new bit BLC to SUPTPRT2_DW2 ■ Updated the access type of all Reserved fields from RO to Rsvd to be consistent with the xHCI Specification <p>Chapter 8, “Device Programming Model”</p> <ul style="list-style-type: none"> ■ Added more information on DEPCFG command in Table 8-5 and Table 8-6 <p>Chapter 9, “Host Programming Model”</p> <ul style="list-style-type: none"> ■ Added the following new sections <ul style="list-style-type: none"> - “Debug Support in Multi-Port and One-Port Configurations” - “Disabled Endpoint Handling (DCCTRL.DRC/DCR)” - “FIFO Sizes Impact” ■ Updated Table 9-3 <ul style="list-style-type: none"> - Added HCRST behavior - Updated Chained TRBs greater than TRB cache size - Removed Multiple Ports since it is supported now ■ Updated “TRB Cache Size and Transfers” <p>Chapter 11, “OTG v2/v3 Add-Ons and Battery Charger”</p> <ul style="list-style-type: none"> ■ Added a note on A-Peripheral to A-Host flow in “A-Host -> A-Peripheral -> A-Host Flow” <p>Chapter 12, “Hibernation”</p> <ul style="list-style-type: none"> ■ Updated “Hibernation While Connected” and “Hibernation While Disconnected” ■ Updated step 8 for DSTS.USBLnkSt values other than 3, 14, and 15 in “Software or PHY-Initiated Wakeup While Connected” ■ Updated width of the Core/PMU data bus to 32 bits (see Figure 12-8 and Table 12-6) <p>Chapter 14, “Device Mode Non-Processor Interface (NPI)”</p> <ul style="list-style-type: none"> ■ Updated description of npi_dev_config[31:0] in Table 14-5 <p>Appendix D, “Fixed ECNs”</p> <ul style="list-style-type: none"> ■ Added a new table on Fixed xHCI ECNs (Table D-3)

(Continued)

Date	Version	Description
August 2012	2.40a	<ul style="list-style-type: none"> ■ Added information on the xHCI Debug Capability (Chapter 1, 3, 4, 6, and 9) ■ Added information on the LPM Errata (Chapter 3, 4, 6, and Appendix D) ■ Renamed Chapter 7 as “Device Descriptor Structures”: Removed “USB 3.0” from the title <p>Chapter 1, “Product Overview”</p> <ul style="list-style-type: none"> ■ Updated the speed and clock requirements for the HS mode ■ Updated section “Known Issues” <p>Chapter 2, “Architecture Overview”</p> <ul style="list-style-type: none"> ■ Section “Transmitting an Endpoint Ready Packet”: Moved contents to section “Device List Processor” of Chapter 3, “Architecture Details” <p>Chapter 3, “Architecture Details”</p> <ul style="list-style-type: none"> ■ Section “USB 2.0 Extension Transaction to Support LPM”: Removed the line “This bit is common to both the host and the device.” because the LPMCap is valid only in the device mode ■ Updated Figure “Software Resets and PHY Clock Sequencing and Requirements” ■ Added information on the LPM errata <p>Chapter 4, “Parameters”</p> <ul style="list-style-type: none"> ■ Split the configuration parameter table into 7 tables based on the coreConsultant tabs ■ Added section “Example Device Endpoint Mapping in Different Applications” ■ Updated the description and default value for DWC_USB3_GCTL_INIT ■ Updated the description of DWC_USB3_HOST_NUM_PERIODIC_EP ■ Added the following parameters: <ul style="list-style-type: none"> - DWC_USB3_GTXFIFOPRIDEV_INIT - DWC_USB3_GTXFIFOPRIHST_INIT - DWC_USB3_GRXFIFOPRIGHST_INIT - DWC_USB3_GFIFOFRIDBC_INIT - DWC_USB3_GDMAHLRATIO_TX_INIT - DWC_USB3_GDMAHLRATIO_RX_INIT - DWC_USB3_LPM_ERRATA_ENABLE

(Continued)

Date	Version	Description
August 2012 <i>Cont'd</i>	2.40a <i>Cont'd</i>	<p>(Continued)</p> <p>Chapter 5, "Signal Descriptions"</p> <ul style="list-style-type: none"> ■ System Clock, Reset, and Control Signals: <ul style="list-style-type: none"> - Updated the description of the bus_clk_early signal ■ AXI Master Interface Signals <ul style="list-style-type: none"> - Updated the width of the xm_wstrb signal ■ AXI Slave Interface Signals <ul style="list-style-type: none"> - Updated the width of the xs_wstrb signal ■ LPM Interface Signals <ul style="list-style-type: none"> - Added more information on the utmi_sleep_n signal ■ Hub Interface Signals <ul style="list-style-type: none"> - Added a new bit to the hub_desc_override_params signal ■ Synopsys Test Environment Interface Signals <ul style="list-style-type: none"> - Added a new signal pmgt_gate_bus_clk_ext <p>Chapter 6, "Registers"</p> <p>Global Registers:</p> <ul style="list-style-type: none"> ■ GSBUSCFG0: Updated [15:12] from Reserved to Reserved1 ■ GCTL <ul style="list-style-type: none"> - [11]: Added a note on the CoreSoftReset field - [7:6]: Added more information on the RAMClkSel field for the host mode - [2]: Added more information on the U2EXIT_LFPS bit ■ GUCTL[13]: Added a new bit EnOverlapChk ■ GUSB3PIPECTLn[8]: Added more information on the RX_DETECT to Polling.LFPS Control <p>Device Registers:</p> <ul style="list-style-type: none"> ■ DCFG[21:17]: Added information on the NumP field ■ DCTL[31]: Added more information on the Run/Stop bit ■ DCTL[16:17]: Updated the access type of the CRS and CSS bits from WO to R_W, and added a note on the value of these fields when read ■ DSTS[25]: Fixed a typo in the RSS description ■ Section "Command 6: Start Transfer (DEPSTRTRXFER)": Added a note on the link state change during an IN transfer <p>OTG and Battery Charger Registers</p> <ul style="list-style-type: none"> ■ OCFG[4:5]: Added these two bits to the "OTG and Battery Charger Registers Map" to match the description of the OCFG register in the Table "OTG Configuration Register: OCFG" <p>Host Registers:</p> <ul style="list-style-type: none"> ■ Updated the xHCI registers map to be more clear on the register sets' base address calculation

(Continued)

Date	Version	Description
August 2012 <i>Cont'd</i>	2.40a Cont'd	<p>(Continued)</p> <p>Chapter 7, "Device Data Structures"</p> <ul style="list-style-type: none"> ■ Table "Device Descriptor Structure Field Definitions": Updated the description of the BPTRL field <p>Chapter 8, "Device Programming Model"</p> <ul style="list-style-type: none"> ■ Section "Initialization on Disconnect Event": On Disconnect event, the application must set DCTL[8:5] to 5. ■ Section "Buffer Size Rules and Zero-Length Packets": Added more information on the IN endpoint rules ■ Figure "Transfer Usage State Diagram": Removed SETUP because XferNotReady(Setup) will never occur ■ Section "Handling L1 Event During a Transfer": Added ■ Figure "Software Flow for Control Transfers": Removed the loop "XferNotReady(Data)-expected dir" ■ Section "Three-Stage Control Transfer Programming Model": Removed the bullet on XferNotReady(Data) ■ Section "Data Movement Within a Stream": Removed the software requirement of setting the high priority bit in response to the XferNotReady event with the EventStatus field indicating "XferNotActive" <p>Chapter 11, "OTG v2/v3 Add-Ons and Battery Charger"</p> <ul style="list-style-type: none"> ■ Section "A-Host -> A-Peripheral -> A-Host Flow" <ul style="list-style-type: none"> - Added a note on LTSSM state and Device Notification TP ■ Updated and "B-Peripheral -> B-Host -> B-Peripheral Flow" <ul style="list-style-type: none"> - Added a note on LTSSM state and Device Notification TP - Added another note on handling any unexpected error scenario <p>Chapter 12, "Hibernation"</p> <ul style="list-style-type: none"> ■ Added description of debug_u2pmu[0] and debug_u3pmu[0] signals to the Table "Power Controller Interface Signals" <p>Chapter 14, "Device Mode Non-Processor Interface (NPI)"</p> <ul style="list-style-type: none"> ■ Added two new bits to the npi_ssphylink_config signal

(Continued)

Date	Version	Description
July 2012	2.30a	<ul style="list-style-type: none"> ■ Updated databook (Chapter 4, 5, and 10) to include Vendor Control Interface (VCI) ■ Removed the vaux_reset_n input to the DWC_usb3 core, and moved it's behavior to the vcc_reset_n signal. The vcc_reset_n signal now has two behaviors depending on whether the hibernation feature is enabled or not. (Chapter 3, 5, 6, and 12) <p>“Preface”:</p> <p>Updated xHCI errata from January 17 2011 to June 13 2011</p> <p>Chapter 1, “Product Overview”</p> <ul style="list-style-type: none"> ■ Section “Speed and Clock Requirements”: Added new requirements ■ Updated ‘Version affected’ under star 9000498951 in Known Issues section ■ Updated “Known Issues” <p>Chapter 2, “Architecture Overview”</p> <p>Chapter 3, “Architecture Details”</p> <ul style="list-style-type: none"> ■ Section “Clock Generation and Clock Tree Synthesis (CTS) Requirements”: Updated information on RAM clock selection in host mode ■ Section “Device RxFIFO Data Allocation”: Fixed a typo (Changed 4 MDWIDTH-bytes to 16 bytes) ■ Figure “Two-Port RAM Interface (when DWC_USB3_SPRAM_TYP=0)": Removed three signals (ram<RAM#>_p1_wr_n, ram<RAM#>_p1_wdata, and ram<RAM#>_p2_rdata) ■ “Receive Path” subsection of “Packet Threshold and Burst Features for High Latency Systems”: Fixed typos ■ Added “Device Mode Receive Path” <p>Chapter 4, “Parameters”</p> <p>Updated:</p> <ul style="list-style-type: none"> ■ Descriptions of the following parameters in Table 4-7: <ul style="list-style-type: none"> - DWC_USB3_VENDOR_CTL_INTERFACE - DWC_USB3_GUSB3PIPECTL_INIT ■ Dependencies of the following parameters in Table 4-7: <ul style="list-style-type: none"> - DWC_USB3_VENDOR_CTL_INTERFACE

(Continued)

Date	Version	Description
July 2012 <i>Cont'd</i>	2.30a <i>Cont'd</i>	<p><i>(Continued)</i></p> <ul style="list-style-type: none"> ■ Default value of the following parameters in Table 4-7: <ul style="list-style-type: none"> - DWC_USB3_GCTL_INIT - DWC_USB3_GUSB2PHYCFG_INIT - DWC_USB3_GUSB3PIPECTL_INIT - DWC_USB3_HOST_NUM_CACHE_EP_PER_SS_INSTANCE - DWC_USB3_HOST_NUM_CACHE_EP_PER_HS_INSTANCE ■ Removed DWC_USB3_HOST_EN_EXTERNAL_CAP (this feature can now be enabled/disabled using GUCTL[12]) ■ Moved DWC_USB3_GSBUSCFG0_INIT and DWC_USB3_GSBUSCFG1_INIT from Host Configuration to Advanced Configuration section ■ Added DWC_USB3_HUB_ENABLE_VCI <p>Chapter 5, "Signal Descriptions"</p> <ul style="list-style-type: none"> ■ System Clock, Reset, and Control Signals: Updated the minimum frequency, and the accuracy requirement of ref_clk. ■ System Bus Interface: Updated section 'Master Interface' ■ Updated gmw_mcache_reqinfo, xm_awcache, xm_arcache, hm_hprot signal descriptions ■ UTMI+ Vendor Control Interface: Added a note on NPI configuration ■ Host Interface: Added a note on host_u3_port_disable ■ Hub Signal Interface: Added a new bit (1760) to hub_desc_override_params, a new signal hub_dsport_state, and VCI signals ■ Miscellaneous Signal Interface Signals: Added dft_clk_mux_ctlval_bus_clk[1:0], and updated the width of dft_clk_mux_ctlval_mac2_wpc_clk from 1 bit to 3 bits <p>Chapter 6, "Registers"</p> <p>Global Registers:</p> <ul style="list-style-type: none"> ■ GRXTHRCFG: Added information on device mode for USBRxPktCntSel, USBRxPktCnt, and USBMaxRxBurstSize fields ■ GCTL[10]: Added information on SOFITPSYNC requirement, and disabled-SOFITPSYNC scenario ■ GCTL[7:6]: Added a note on restriction in host mode ■ GCTL[2]: Added a new field U2EXIT_LFPS ■ GUCTL[31:22]: Updated description of REFCLKPER ■ GUCTL[12]: Added a new field ExtCapSuptEN ■ GUSB2PHYACFn: Added more information to the description of various fields ■ GUSB3PIPECTL[23]: Updated access type of StartRxDetU3RxDet as WO ■ GUSB3PIPECTL[8]: Added a new bit "RX_DETECT to Polling.LFPS Control" ■ DEPEVT: Added more information on EventParam ■ DEVT: Updated description of various fields ■ Added DMA Priority Registers

(Continued)

Date	Version	Description
July 2012 <i>Cont'd</i>	2.30a <i>Cont'd</i>	<p>(Continued)</p> <p>Device Registers:</p> <ul style="list-style-type: none"> ■ DCFG[21:17]: Added information on NumP dependency on GRXTHRCFG.USBRxPktCntSel ■ DCTL[31]: Updated description ■ DCTL[18], [19]: Added note on access type of these fields ■ Updated Table “Minimum Duration for Soft Disconnect” ■ DEV滕[11:10]: Updated as Rsvd ■ DEV滕[6]: Added a new field U3L2L1SuspEn ■ DSTS: Added more information on DSTS.SOFFN ■ Table “Command 1: Set Endpoint Configuration Parameters: DEPCFG”: Updated Parameter 2 description ■ Section ‘Command 6: Start Transfer (DEPSTRXFER)’: Added a note on Start Transfer Command <p>Chapter 7, “Device Data Structures”</p> <p>Table “Device Descriptor Structure Field Definitions: Added more information on TRBCTL field</p> <p>Chapter 8, “Device Programming Model”</p> <ul style="list-style-type: none"> ■ Section “Non-Isochronous OUT Transfers”: Updated with information on GRXTHRCFG.USBRxPktCntSel ■ Table “Control OUT Transfer”: Updated Step 12 ■ Section “Multiple Device Interrupt Support”: Updated to be more clear <p>Chapter 9, “Host Programming Model”</p> <p>Chapter 10, “Hub”</p> <p>Added a new section Vendor Control Interface</p> <p>Chapter 11, “OTG v2/v3 Add-Ons and Battery Charger”</p> <p>Chapter 12, “Hibernation”</p> <ul style="list-style-type: none"> ■ Section “Initialization for Hibernation Support”: Updated with more information on HIRD_Thres ■ Section “Software or PHY-Initiated Wakeup While Disconnected”: Updated with more information <p>Chapter 14, “Device Mode Non-Processor Interface (NPI)”</p> <ul style="list-style-type: none"> ■ Table: “Non Processor Interface Control/Status/Configuration Signals”: Updated description of npi_ep_config[17:13] ■ Section “NRDY, ERDY, NUMP, and EOB Management”: Updated information on Burst Size field ■ Added a new section “Burst Size and NumP” ■ Added a section on ‘UTMI Vendor Access’

(Continued)

Date	Version	Description
July 2012 <i>Cont'd</i>	2.30a <i>Cont'd</i>	<p>(Continued)</p> <p>Appendix A, “Area, Speed, and Power”</p> <ul style="list-style-type: none"> ■ Added a note on MBs <p>Appendix B, “Clock Domain Crossing”</p> <ul style="list-style-type: none"> ■ Added src/com/DWC_usb3_sync_2edge.v/in_p_1d to SDF list <p>Appendix D, “Fixed ECNs”</p>
April 2012	2.20a	<p>Chapter 1, “Product Overview”</p> <ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - Table 1-1 - Description of USB 3.0 Static Dual-Role Device (DRD/OTG) controller - Table 1-3 and Table 1-4 - “Application Interface Features” - “USB 3.0 Device Features” - “USB Class-Specific Device Features” - “USB 3.0 xHCI Host Features” - Multiple additions and deletions (see change bar version of databook) - “Power Optimization Features” - “Area Reduction Features” - “Performance Features” - “Debug Features” - Host section under “Known Issues” ■ Removed: <ul style="list-style-type: none"> - USB 3.0 Supported Features section <p>Chapter 2, “Architecture Overview”</p> <ul style="list-style-type: none"> ■ Updated with information on Not Active state (see “Reception Flow Details”) ■ Added a note on bus clock requirement (see “Clock Domains”) <p>Chapter 3, “Architecture Details”</p> <p>Updated note in “ADP Module”</p> <p>Chapter 4, “Parameters”</p> <ul style="list-style-type: none"> ■ Added DWC_USB3_DEV_EN_SCATTER_PACKETS_OF_8_TO_15_TRBS <p>Updated:</p> <ul style="list-style-type: none"> ■ Descriptions of the following in Table 4-7: <ul style="list-style-type: none"> - Description of DWC_USB3_GCTL_INIT

(Continued)

Date	Version	Description
April 2012 <i>Cont'd</i>	2.20a <i>Cont'd</i>	<p>(Continued)</p> <ul style="list-style-type: none"> - Description of DWC_USB3_SBUS_TYPE to include AXI - Description of bits [24] and [21:19] of the DWC_USB3_GUSB3PIPECTL_INIT - Description DWC_USB3_GCTL_INIT PM ■ Dependencies of the following: <ul style="list-style-type: none"> - DWC_USB3_EN_ADP - DWC_USB3_EN_BC - DWC_USB3_SBUS_TYPE - DWC_USB3_MDWIDHT - DWC_USB3_SDWIDHT - DWC_USB3_AWIDHT - DWC_USB3_IDWIDHT - DWC_USB3_SIDWIDHT - DWC_USB3_NUM_RAMs - DWC_USB3_SPRAM_TYP - DWC_USB3_USERID - DWC_USB3_GUCTL - DWC_USB3_GCTL_INIT - DWC_USB3_VENDOR_CTL_INTERFACE - DWC_USB3_SUSPEND_ON_DISCONNECT_EN - DWC_USB3_DEVICE_NUM_INT - DWC_USB3_NUM_EPS - DWC_USB3_NUM_IN_EPS - DWC_USB3_CACHE_TRBS_PER_TRANSFER - DWC_USB3_NUM_SS_USB_INSTANCES - DWC_USB3_NUM_HS_USB_INSTANCES - DWC_USB3_NUM_FSLS_USB_INSTANCES - DWC_USB3_GSBUSCFG0_INIT - DWC_USB3_GSBUSCFG1_INIT - DWC_USB3_GTXTHRCFG_INIT - DWC_USB3_SSPHY_INTERFACE_NUM_PIPE - DWC_USB3_EN_LOG_PHYS_EP_SUPT - DWC_USB3_ATSPEED_DFT - DWC_USB3_RAM_CLK_TO_BUS_CLK - DWC_USB3_EN_BUS_FILTERS ■ Default value of DWC_USB3_HOST_NUM_CACHE_EP_PER_SS_INSTANCE (was 8 now 10) <p>Removed DWC_USB3_SSPHY_INTERFACE from Table 4-7 Chapter 5, “Signal Descriptions” Added a note on At-Speed DFT control ports (see Table 5-33) Updated:</p> <ul style="list-style-type: none"> ■ Information on Hub in “RAM Interface” ■ Width of hub_desc_override_params <p>Removed usb2_phy_clk from “UTMI+ Parallel Interface” because it is no longer used</p>

(Continued)

Date	Version	Description
April 2012 <i>Cont'd</i>	2.20a <i>Cont'd</i>	<p>Chapter 6, "Registers"</p> <ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - Reset value of acronym a0 in "Device Registers Map" - "Global SoC Bus Configuration Register 0 (GSBUSCFG0)" (description and Table 7-5) - Access type of GRXTHRCFG[15] as R_W (see "Global Rx Threshold Control Register: GRXTHRCFG") - Description of GCTL.SOFITPSYNC (see "Global Control Register: GCTL") - "Global User Control Register: GUCTL" <ul style="list-style-type: none"> - default value (Device Mode and Host Mode) - GCTL[11] to add GCTL and GUCTL to list of registers not cleared by this bit - GUCTL[13:12] Rsvd with R_W access - Description of GUSB3PIPECTLn[30] (see "Global USB3 PIPE Control Register (GUSB3PIPECTLn)") - Note in GDBGLTSSM[29], [28], and [29] (see "Global Debug LTSSM Register: GDBGLTSSM") - Descriptions of bits 22 and 23 of "Global USB3 PIPE Control Register (GUSB3PIPECTLn)" - DEVT[11:8]: <ul style="list-style-type: none"> - "End of Periodic Frame Event (EOPF)" as Reserved (see "Device-Specific Events: DEVT") - 11: Event Buffer Overflow Event (EvntOverflow) - Reset value for DCFG[2:0] DevSpd as 3'b100 (see "Device Configuration Register: DCFG") - DCFG[11:10] PerFrInt as Reserved (see "Device Configuration Register: DCFG") - DCTL[8:5] ULStChngReq – removed references to loopback (see "Device Control Register: DCTL") - DEVTE[6] EOPFEn as Reserved (see "Device Event Enable Register: DEVTE") - DEVTE[10] CmdCmpltEn as Reserved (see "Device Event Enable Register: DEVTE") - DEVENT[11] is now reserved (see "Device Event Enable Register (DEVTE)") - 07h (Transmit Device Notification) in Device Generic Command Types (see "Device Generic Command Types") - Reset value for HCCPARAMS[5] LHRC as 1 (see "Capability Parameters Register (HCCPARAMS)") - "ADP Configuration Register (ADPCFG)" - Value of of DWC_USB3_CSR_ACCESS_TIMEOUT in "CSR Timeout Mechanism" - Default values of HCSPARAMS2[31:27] and [25:21] ■ The following commands are not supported from 2.20a release, and are hence removed from "Device Generic Command Types": <ul style="list-style-type: none"> - Command 01h (Transmit Set Link Function LMP) - Command 03h (Transmit Function Wake Device Notification) - Command 06h (Transmit Function Host Role Request Device Notification)

(Continued)

Date	Version	Description
April 2012 <i>Cont'd</i>	2.20a <i>Cont'd</i>	<p>Chapter 7, "Device Data Structures" Chapter 8, "Device Programming Model"</p> <ul style="list-style-type: none"> ■ Figure 8-1 "Transfer Usage State Diagram": Removed the text "Update TRB Status and release the stream" in the "End Transfer" state ■ Table 8-1 "Power-On or Soft Reset Register Initialization": Removed reference to loopback ■ "Buffer Size Rules and Zero-Length Packets" (regarding E-Star 9000535038: In device mode, increased the maximum number of TRBs that may be used to describe an IN packet from 7 to 15.) ■ Replaced text in Step 4 of Table 8-13 <p>Chapter 9, "Host Programming Model"</p> <ul style="list-style-type: none"> ■ Table 9-1 "Overriding Global Registers in Host Configuration": Removed reference to loopback <p>Chapter 11, "OTG v2/v3 Add-Ons and Battery Charger"</p> <ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - Note in "Core OTG Functions" - Command value from 06h to 07h (see "A-Host -> A-Peripheral -> A-Host Flow" and "B-Peripheral -> B-Host -> B-Peripheral Flow Diagram") - Figure 11-10, "A-Device Flow Diagram" - Figure 11-20, "B-Device Flow Diagram" <p>Chapter 12, "Hibernation"</p> <ul style="list-style-type: none"> ■ Removed reference to HSIC in Figure 12-3 "Hibernation Architecture in Host Mode", since it is not supported ■ Updated link in "Software or PHY-Initiated Wakeup while Disconnected" to Section 12.2.5.3 ■ Replaced size and description of pm_pmu_config_strap in Table 12-5 <p>Chapter 14, "Device Mode Non-Processor Interface (NPI)"</p> <ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - npi_tx_isoc_eof and npi_tx_isoc_leof description (see Table 14-4) - npi_dev_config[31:0] (see Table 14-5) - Description of npi_dev_ctl[25:0] <p>Appendix A, "Area, Speed, and Power"</p> <p>Appendix D, "Fixed ECNs"</p> <ul style="list-style-type: none"> ■ Added "Fixed USB 2.0 ECNs"

(Continued)

Date	Version	Description
January 2012	2.10a	<p>Global:</p> <ul style="list-style-type: none"> ■ The DWC_usb3 2.10 release includes a new manual, DesignWare Cores SuperSpeed USB 3.0 Controller User Guide. Chapters and sections from previous releases of this databook have been moved to the new User Guide or Installation Guide (where noted): <ul style="list-style-type: none"> - “Deliverables” (formerly Section 1.6) – DesignWare Cores SuperSpeed USB 3.0 Controller Installation Guide - “Select Configuration Parameters” section – DesignWare Cores SuperSpeed USB 3.0 Controller User Guide - “Hardware Integration” chapter – DesignWare Cores SuperSpeed USB 3.0 Controller User Guide - “Verification” chapter – DesignWare Cores SuperSpeed USB 3.0 Controller User Guide ■ Corrected instances of BIinterval to blinterval <p>Preface</p> <ul style="list-style-type: none"> ■ Updated “Reference Documentation” <p>Chapter 1, “Product Overview”</p> <ul style="list-style-type: none"> ■ Added a note for the “Add-On” features relating to licensing <p>Chapter 3, “Architecture Details”</p> <ul style="list-style-type: none"> ■ Updated <ul style="list-style-type: none"> - “Clock Generation and Clock Tree Synthesis (CTS) Requirements” - Figure 3-10, Figure 3-11, Figure 3-12 <p>“Synchronous Static RAM Requirement” Chapter 4, “Parameters”</p> <ul style="list-style-type: none"> ■ Updated default value for “In Host Mode Enable USB2.0 suspend during Disconnect?” configuration option in Table 4-7 <p>Chapter 5, “Signal Descriptions”</p> <ul style="list-style-type: none"> ■ Added: <ul style="list-style-type: none"> - ref_clk (see “System Clock, Reset, and Control Signals”)

(Continued)

Date	Version	Description
January 2012 <i>Cont'd</i>	2.10a <i>Cont'd</i>	<ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - dft_* description in Table 5-33 - fladj_30mhz_reg[5:0] - “Native Master Interface” - “Details of the AXI Master Interface” - usb2_phy_clk now unused - utmi_suspend_com_n, utmi_l1_suspend_com_n (updated description regarding sups end state for conditions that make signals assert high/low) - gs_mbe (description) ■ Corrected bit widths for following signals: xm_awsize, xm_wstrb, xs_wstrb, xm_arlen, xm_awmisc_info, xm_bmisc_info, xm_armisc_info, xm_rmisc_info, xs_awmisc_info, xs_bmisc_info, xs_armisc_info, xs_rmisc_info ■ Removed: <ul style="list-style-type: none"> - gs_idle from Figure 5-6 and Table 5-6 - core_usb2phy_reset - phy_usb2phy_reset <p>Chapter 6, “Registers”</p> <ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - GCTL[9], GCTL[10], and GUCTL[31:22] are no longer reserved. - GHWPARAMS4[21] not reserved, now DWC_USB3_EXT_BUFF_CONTROL - GUSB2PHYCFGn[16] to Reserved (see Section 7.2.5.2) - GUSB2PHYCFGn[31] - DCFG[2:0] – Reset value - DCTL. ULStChngReq (see bits 8:5 in Table 7-57) - Bits 29 and 21:18 of “Device Status Register (DSTS)” - DEPCFG.blInterval_m1 (see Table 6-73) - Bits [30:28] in Section 7.2.5.5: “Global USB3 PIPE Control Register (GUSB3PIPECTLn)” ■ Added: <ul style="list-style-type: none"> - Information regarding Address bits A0 to A19 used by the core (1 MB memory space in “Core CSR Memory Map”) - GUCTL[11] (see Table 7-14) - Bits 28 and 29 to GUSB3PIPECTLn register (see Table 7-44) - 07h: Transmit Device Notification <p>Chapter 8, “Device Data Structures”</p> <ul style="list-style-type: none"> ■ Updated Section 8.1.1 <p>Chapter 9, “Device Programming Model”</p> <ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - Added note to Section 9.1.7 - Definition of blInterval in Section 9.3.2 - “Two-Stage Control Transfer Programming Model” - “Three-Stage Control Transfer Programming Model”

(Continued)

Date	Version	Description
January 2012 <i>Cont'd</i>	2.10a <i>Cont'd</i>	<p>Chapter 11, "OTG v2/v3 Add-Ons and Battery Charger"</p> <ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - Figure 11-39 <p>Chapter 12, "Hibernation"</p> <ul style="list-style-type: none"> ■ Updated: <ul style="list-style-type: none"> - "OTG Programming Model" - PHY Wakeup Events table and Notes in Section 12.2.7 ■ Removed item 10 from Section 12.2.13 <p>Chapter 14, "Device Mode Non-Processor Interface (NPI)"</p> <ul style="list-style-type: none"> ■ Updated Table 14-5 <p>Appendix A, "Area, Speed, and Power"</p> <ul style="list-style-type: none"> ■ Replaced: <ul style="list-style-type: none"> - Sections: "Internal RAM, Bus Bandwidth, and Clock Frequency Requirements" and "Minimum RAM and BUS Clock Frequencies" with "Minimum Clock Frequencies: bus_clk, ram_clk" ■ Updated: <ul style="list-style-type: none"> - Appendix A.2, "Host Area Numbers" - Appendix A.3, "DRD/OTG Area Numbers"
October 2011	2.00a	<p>Global:</p> <ul style="list-style-type: none"> ■ Replaced bus_reset_n, poweron_reset_n and core_reset_n signals by vcc_reset_n and vaux_reset_n signals <p>Preface:</p> <ul style="list-style-type: none"> ■ Added cross references to the chapters that are newly added ■ Added OTG 3.0 Specification under Reference Documentation <p>Chapter 1, "Product Overview"</p> <ul style="list-style-type: none"> ■ Updated Table "DWC_usb3 Configurable Parameters Summary" (for details, refer to Revision History of Chapter 2) <p>Chapter 2, "Parameters"</p> <ul style="list-style-type: none"> ■ Updated the following parameters: <ul style="list-style-type: none"> - DWC_USB3_EN_PWROPT - DWC_USB3_IDWIDTH - DWC_USB3_SIDWIDTH - DWC_USB3_NUM_RAMs - DWC_USB3_HSPHY_INTERFACE - DWC_USB3_SSPhy_INTERFACE_NUM_PIPE - DWC_USB3_FAST_TAT_EN - DWC_USB3_GUSB2PHYCFG_INIT

(Continued)

Date	Version	Description
October 2011 <i>Cont'd</i>	2.00a <i>Cont'd</i>	<p><i>(Continued)</i></p> <ul style="list-style-type: none"> - <i>DWC_USB3_PIPE_RXTERM_RESET_VAL</i> - <i>DWC_USB3_DEVICE_NUM_INT</i> - <i>DWC_USB3_DEV_TXF0_DEPTH</i> - <i>DWC_USB3_HOST_NUM_INTERRUPTER_SUPT</i> - <i>DWC_USB3_NUM_DEVICE_SUPT</i> - <i>DWC_USB3_FREECLK_USB2_EXIST</i> - <i>DWC_USB3_HOST_NUM_PERIODIC_EP</i> - <i>DWC_USB3_GRXTHRCFG_INIT</i> - <i>DWC_USB3_GCTL_INIT</i> <ul style="list-style-type: none"> ■ <i>DWC_USB3_EN_BUS_FILTERSDWC_USB3_HOST_NUM_CACHE_EP_PER_SS_INSTANCE, DWC_USB3_HOST_NUM_CACHE_TRB_PER_SS_EP, DWC_USB3_HOST_NUM_CACHE_EP_PER_HS_INSTANCE, DWC_USB3_HOST_NUM_CACHE_TRB_PER_HS_EP, DWC_USB3_HOST_NUM_CACHE_EP_PER_FSLS_INSTANCE, DWC_USB3_HOST_NUM_CACHE_TRB_PER_FSLS_EP</i> ■ <i>DWC_USB3_NUM_RXF_SS_PKTS, DWC_USB3_NUM_RXF_HS_PKTS, DWC_USB3_NUM_RXF_FSLS_PKTS, DWC_USB3_NUM_TXF_SS_PKTS, DWC_USB3_NUM_TXF_HS_PKTS, DWC_USB3_NUM_TXF_FSLS_PKTS</i> ■ <i>Removed the following parameters:</i> <ul style="list-style-type: none"> - <i>DWC_USB3_NUM_OUTSTANDING_TXDMA</i> (<i>This parameter is no more configurable</i>) - <i>DWC_USB3_PERIODIC_PARAMS</i> - <i>DWC_USB3_SSPHY_INTERFACE</i> - <i>DWC_USB3_PERIODIC_PARAMS</i> <p><i>Chapter 3, "Signal Descriptions"</i></p> <ul style="list-style-type: none"> ■ <i>System Clock, Reset, and Control Signals</i> <ul style="list-style-type: none"> - <i>Added information on suspend_clk</i> - <i>Updated ram_clk_gated description</i> - <i>Added a new signal 'ram_clk_gated_ram0'</i> ■ <i>System Bus Interface: Added information on wait cycles on the system bus</i> ■ <i>Native Master Interface Signals</i> <ul style="list-style-type: none"> - <i>Added two new signals: 'gmw_mbigendian' and 'gmr_mbigendian'</i> ■ <i>Updated information and added example waveforms for AXI Master and Slave Interface</i> ■ <i>RAM Interface Signals</i> <ul style="list-style-type: none"> - <i>Added information on ram_clk_gated and ram_clk_gated_ram0 clock domains</i> ■ <i>USB 3.0 PIPE3 PHY Interface Signals</i> <ul style="list-style-type: none"> - <i>Added information on pipe3_PhysStatus and pipe3_PhysStatus_async</i> - <i>Removed pipe3_common_rx_pclk and pipe3_common_tx_pclk</i> ■ <i>UTMI+ Parallel Interface Signals</i> <ul style="list-style-type: none"> - <i>Updated utmiotg_vbusvalid and utmisrp_bvalid as registered signals</i> - <i>Added information on utmi_suspend_n</i>

(Continued)

Date	Version	Description
October 2011 <i>Cont'd</i>	2.00a <i>Cont'd</i>	<p>(Continued)</p> <ul style="list-style-type: none"> ■ UTMI+ OTG Interface Signals <ul style="list-style-type: none"> - Updated utmiotg_valid description ■ Host Interface Signals <ul style="list-style-type: none"> - Updated pm_power_state_request description - Removed current_power_state - Updated xhc_bme description ■ Miscellaneous Signal Interface <ul style="list-style-type: none"> - Updated dft_en_ram_clk_gated description - Removed pmgt_clamp_n and power_up_req since they are no longer used - Updated 'bus_filter_bypass' description - Added a new signal 'logic_analyzer_trace' ■ Synopsys Test Environment Interface Signals <ul style="list-style-type: none"> - Updated dbg_ram_clk_gated description - Updated sm2bl_cur_mode as b2rl_cur_mode - Updated debug_mclk_usof_number[18:0] description <p>Chapter 4, "Architecture Overview"</p> <p>Chapter 5, "Architecture Details"</p> <ul style="list-style-type: none"> ■ Updated the following diagrams with minimum bus_clk and ram_clk as 60 MHz, and removed mac2_clk: <ul style="list-style-type: none"> - Figure "Device/DRD/OTG Clock Generation and CTS Requirements" - Figure "Host Clock Generation and CTS Requirements" ■ Updated the following figures with ram_clk_gated_ram0 signal: <ul style="list-style-type: none"> - "Device/DRD/OTG Clock Generation and CTS Requirements" - "Host Clock Generation and CTS Requirements" ■ Section 'SS PHY Status Transfer': Updated section with a timing diagram and details on top level connections for PhyStatus ■ Updated fourth bullet in Note after Figure 5-27 <p>Chapter 6, "Hardware Integration"</p> <ul style="list-style-type: none"> ■ Section 'Synchronous Static RAM Requirement': Added information on 2-port RAM requirement ■ Updated the following figures with ram_clk_gated_ram0 signal: <ul style="list-style-type: none"> - 'Single-Port RAM Interface (when DWC_USB3_SPRAM_TYP=1)' - '2-Port RAM Interface (when DWC_USB3_SPRAM_TYP=0)' ■ Updated Section 'OTG Mode – Filters, VBUS, and Over-Current Connection' ■ Figure 'Port Connection in OTG Model': Removed utmiotg_availd ■ Table 'Usage of IDDIG and VBUS-Related Signals in UTMI Interface': utmiotg_valid is not used. ■ Section 'Integrating the SuperSpeed PHY': Updated with information on DWC_USB3_PIPE_RXTERM_RESET_VAL ■ Figure 'GUSB3PIPECTL[27] Timing Diagram (Ux Exit in Px)': Updated phy_rx_serial_line as phy_tx_serial_line

(Continued)

Date	Version	Description
October 2011 <i>Cont'd</i>	2.00a <i>Cont'd</i>	<p>(Continued)</p> <ul style="list-style-type: none"> ■ Added a new section ‘Integrating the Dual Power Rail (Hibernation) Feature’ ■ Section ‘Little-Endian and Big-Endian’: Updated Data[127:0] to include B11, B10, B9, and B8 <p>Chapter 6, “Registers”</p> <p>Global Registers:</p> <ul style="list-style-type: none"> ■ GSBUSCFG1 <ul style="list-style-type: none"> - [11:8]: Updated BREQLIMIT as PipeTransLimit ■ GCTL <ul style="list-style-type: none"> - [17]: Added a new field ‘BYPSETADDR’ - [2]: Updated as Reserved - [1]: Changed field as GblHibernationEn ■ GUCTL – updated description ■ GBUSERRADDR: Added information on AHB and AXI configurations ■ GUSB3PIPECTLn <ul style="list-style-type: none"> - [10]: Access type updated as R_W ■ GHWPARAMS4 <ul style="list-style-type: none"> - [16:13]: Added a new field ‘DWC_USB3_HIBER_SCRATCHBUFS’ ■ GDBGLSPMUX <ul style="list-style-type: none"> - [23:16]: Added a new field “logic_analyzer_trace” Port MUX Select’ ■ SUPTPRT3_DW3 in “xHCI 1.0 Host Registers Map” ■ Table ‘Device-Specific Events: DEVT’: <ul style="list-style-type: none"> - [11:8]: Added ‘Hibernation Request Event’ <p>Device Endpoint Registers:</p> <ul style="list-style-type: none"> ■ DEPCMDn – added a note to each command section regarding Star 9000498951

(Continued)

Date	Version	Description
October 2011 <i>Cont'd</i>	2.00a <i>Cont'd</i>	<p>(Continued)</p> <p>Device Registers:</p> <ul style="list-style-type: none"> ■ DCTL <ul style="list-style-type: none"> - [29]: Updated Light Soft Reset as Reserved - [19]: Added a new field 'KeepConnect' - [18]: Added a new field 'L1HibernationEn' - [17]: Added a new field 'CRS' - [16]: Added a new field 'CSS' - [12]: Updated description - [10]: Updated description - [8:5]: Added a note on Hibernation ■ DEPEVT: <ul style="list-style-type: none"> - [12]: Updated as Reserved for XferComplete or XferInProgress event ■ DEV滕 <ul style="list-style-type: none"> - [5]: Added a new field 'HibernationReqEvtEn' ■ DSTS <ul style="list-style-type: none"> - [29]: Added a new field 'DCNRD' - [28]: Added a new field 'SRE' - [25]: Added a new field 'RSS' - [24]: Added a new field 'SSS' - [21:18]: Added additional states ■ Updated Table 'Device Generic Command Descriptions': <ul style="list-style-type: none"> - Added commands 04h and 05h for Power Management ■ Command 1: Set Endpoint Configuration: DEPCFG <ul style="list-style-type: none"> - Parameter0[31:26]: Removed 'Ignore Sequence Number' and 'Data Sequence Number', and added 'Config Action'. Other bits are Reserved. ■ Command 3: Changed 'Get Data Sequence' to 'Get Endpoint State' ■ Command 6: Start Transfer (DEPSTRTXFER) <ul style="list-style-type: none"> - Updated information on HighPri field ■ Command 8: Added information on End Transfer Command With Hibernation Enabled <p>USB OTG and Battery Charger Registers</p> <ul style="list-style-type: none"> ■ Added a note on OTG 3.0 register fields ■ OCFG <ul style="list-style-type: none"> - [3]: Updated description of OTGSftRstMsk - [1]: Updated description of HNPCap - [0]: Updated description of SRPCap ■ OCTL: <ul style="list-style-type: none"> - [7]: Added a new field OTG3_GOERR - [6]: Updated the reset value of PeriMode - [1]: Updated description of DevSetHNPEn - [0]: Updated description of HstSetHNPEn

(Continued)

Date	Version	Description
October 2011 <i>Cont'd</i>	2.00a <i>Cont'd</i>	<p>(Continued)</p> <ul style="list-style-type: none"> ■ OEVTE - [23]: Added a new field HRRConfNotifEvnt - [22]: Added a new field HRRInitNotifEvnt - [20], [19], [18], [17], [16], [11], [10], [9], [8], [3], [2], [1]: Updated description ■ OEVTEEN - [23]: Added a new field HRRConfNotifEvntEn - [22]: Added a new field HRRInitNotifEvntEn ■ OSTS - [11:8]: Updated description <p>Chapter 7, "Device Data Structures"</p> <p>Chapter 8, "Device Programming Model"</p> <ul style="list-style-type: none"> ■ Section 'Device Power-On or Soft Reset': Added information on GUSB2PHYCFG ■ Section 'Core Behavior During an Interval': Updated with information on XferNotReady event ■ Section 'Stream Selection and Stream Programming Model': Updated <p>Chapter 9, "Host Programming Model"</p> <p>Chapter 10, "Hub"</p> <p>Chapter 11, "OTG v2/v3 Add-Ons and Battery Charger"</p> <p>Chapter 12, "Hibernation": Added a new chapter on Power Management</p> <p>Chapter 14, "Device Mode Non-Processor Interface (NPI)"</p> <p>Chapter 14, "Verification" (now a part of User Guide)</p> <ul style="list-style-type: none"> ■ Section 'Non-NPI Device Testbench and Test Files': Added a note on loopback test ■ Updated Table 'List of Host Tests Provided in the SoC Integration Example' ■ Updated section 'OTG Testbench and Test Files' ■ Section 'Host Rx.Detect Entry': Added a note on PORTSC[PP] ■ Added a new section 'Measuring Performance' <p>Appendix A, "Installation and Workspace Files"</p> <p>Appendix A, "Area, Speed, and Power"</p> <p>Appendix B, "Clock Domain Crossing"</p> <p>Appendix C, "Optional coreConsultant Flows" (now a part of User Guide)</p> <p>Appendix D, "FPGA Implementation of DWC_usb3 Core" (now a part of User Guide)</p> <p>Appendix D, "Fixed ECNs" (new appendix)</p>

Preface

This databook describes the implementation and use of the Synopsys DesignWare® Cores SuperSpeed USB 3.0 controller. The terms “system bus” and “SoC bus” are used interchangeably.

For more details on USB 3.0 protocol, refer to *USB 3.0 Protocol Overview* training module of the Synopsys *USB University* series.

Product Codes

[Table 4](#) lists the products and product codes for the DWC_usb3 controller.

Table 4 Product Codes for DWC SuperSpeed USB 3.0 Controller

Component Name or Add-On	Product Code
Base Products	
DWC USB 3.0 Device-AHB ^a	6793-0
DWC USB 3.0 Host-AHB ^b	6897-0
DWC USB 3.0 Dual Role Device-AHB ^c	7567-0
DWC USB 3.0 Hub	6921-0
DWC AP USB 3.0 Device	C234-0
DWC AP USB 3.0 Host	C233-0
DWC AP USB 3.0 Dual Role Device	C232-0
Add-Ons	
DWC USB 3.0 Device-Isoch Add-On	7571-0
DWC USB 3.0 Host-Isoch Add-On	7593-0
DWC USB 3.0 DRD-Isoch Add-On	7568-0
DWC USB 3.0 Hibernation Add-On	9228-0
DWC USB 3.0 HSIC Add-On	7387-0
DWC USB 3.0 OTG v3.0 Add-On ^d	7991-0
DWC USB 3.0 OTG v2.0 Add-On	7572-0

a. DWC USB 3.0 Device-AHB includes DWC USB 3.0 Device-AXI Add-On and DWC USB 3.0 Device-Isoch Add-On.

b. DWC USB 3.0 Host-AHB includes DWC USB 3.0 Host-AXI Add-On and DWC USB 3.0 Host-Isoch Add-On.

c. DWC USB 3.0 Dual Role Device-AHB includes DWC USB 3.0 DRD-AXI Add-On and DWC USB 3.0 DRD-Isoch Add-On.

d. DWC USB 3.0 OTG v3.0 Add-On includes DWC USB 3.0 OTG v2.0 Add-On.

Databook Organization

The chapters and appendices of this databook are organized as follows:

- [Chapter 1, "Product Overview"](#), provides an introduction to the DWC_usb3 controller's features, functional blocks, and applications.
- [Chapter 2, "Architecture"](#), describes the system, sub-block modules, design files, and operation of the DWC_usb3 controller.
- [Chapter 3, "Memory Requirements"](#), provides the system memory requirements of the DWC_usb3 controller.
- [Chapter 4, "Parameter Descriptions"](#), describes the configuration parameters of the DWC_usb3 controller, which are selected using the Synopsys coreConsultant tool.
- [Chapter 6, "Signal Interfaces"](#), describes the interfaces of the DWC_usb3 controller.
- [Chapter 7, "Signal Descriptions"](#), describes the signal interfaces and signals with which the DWC_usb3 controller can be integrated into an application environment.
- [Chapter 8, "Hub"](#), describes the USB 3.0 hub controller configuration.
- [Chapter 9, "Battery Charger"](#), describes the functionality of OTG and Battery Charger for DWC_usb3 controller.
- [Chapter 10, "Power Management"](#), describes the power management options available in DWC_usb3 controller.
- [Appendix A, "Area, Speed, and Power"](#), describes the area, speed, and power requirements for several example configurations of the DWC_usb3 controller.
- [Appendix B, "Clock Domain Crossing"](#), describes the types of Clock Domain Crossing (CDC) signals, CDC implementation and validation, and the types of synchronizer modules used in DWC_usb3 controller.
- [Appendix C, "High Speed Inter-Chip \(HSIC\) Feature"](#), describes enumeration, suspend/resume and suspend/remote-wakeup sequences, and limitation of HSIC feature in the DWC_usb3 controller.
- [Appendix D, "Fixed ECNs"](#), describes the released ECNs to date and the status of the controller in terms of whether these ECNs are applied or outstanding.
- [Appendix E, "External Buffer Control"](#), provides information about the device External Buffer Control (EBC) interface.
- [Appendix F, "DWC_usb3 Automotive Safety"](#), describes the automotive safety features supported by the USB 3.0 Controller (requires an additional license).
- [Appendix G, "Internal Parameter Descriptions"](#), describes the internal parameters that might be indirectly referenced in expressions in the Signals, or Parameters chapters.

Related Product Documentation

USB 3.0 Controller

Before installing the USB 3.0 Controller, you can download the latest document set, including the Datasheet, Installation Guide, QuickStart, Databook, and Release Notes, at:

Device: http://www.synopsys.com/dw/ipdir.php?c=dwc_usb_3_0_device

Host: http://www.synopsys.com/dw/ipdir.php?c=dwc_usb_3_0_host

Dual-Role Device (DRD): http://www.synopsys.com/dw/ipdir.php?c=dwc_usb_3_0_drd

Hub: http://www.synopsys.com/dw/ipdir.php?c=dwc_usb_3_0_hub

(SolvNet ID required)

USB 3.0 Controller Linux Driver Software

You can download the Linux Driver Software at:

<https://www.kernel.org/>

Directory: /drivers/usb/dwc3

Synopsys supports these USB drivers.

Verification IP (VIP)

To run simulations in coreConsultant, you must download and install the Synopsys Verification IP for USB 3.0 and AMBA from the SolvNet Download Center, at:

<https://solvnet.synopsys.com/DownloadCenter/dc/product.jsp>

For more information, refer to the *DesignWare Cores SuperSpeed USB 3.0 Installation Guide*.

Synopsys Tools

- *coreConsultant User Guide*, Synopsys, Inc. (included with the coreConsultant tool)
https://www.synopsys.com/dw/doc/doc/coretools/latest/coreconsultant_user.pdf

Web Resources

- DesignWare IP product information: <http://www.designware.com>
- Your custom DesignWare IP page: <http://www.mydesignware.com>
- Documentation through SolvNet: <http://solvnet.com> (Synopsys password required)
- Synopsys Common Licensing (SCL): <http://www.synopsys.com/keys>

Reference Documentation

The following standards are applicable to the USB 3.0 Controller product:

USB Protocol (from USB Implementers Forum)

- *Universal Serial Bus 3.0 Specification*, Revision 1.0, November 12, 2008
- *Universal Serial Bus Mass Storage Class UAS Protocol Specification*, Revision 0.6, June 1, 2008
- *Universal Serial Bus Specification*, Revision 2.0, USB-IF, April 27, 2000
- *Errata for "USB Revision 2.0 April 27 2000" as of May 28, 2002*, USB-IF
- *High-Speed Inter-Chip USB Electrical Specification, Version 1.0*, USB-IF, September 23, 2007
- *Inter-Chip Supplement to the USB Specification, Version 1.0*, USB-IF, March 13, 2006
<http://www.usb.org/developers/docs>
- *Battery Charging Specification*, Revision 1.2, Nov 2, 2010
- *Inter-Chip Supplement to the USB Revision 3.0 Specification*, Revision 1.02, May 19, 2014
<http://www.usb.org>

USB Protocol (other)

- *Universal Serial Bus PC Legacy Compatibility Specification*, 0.9 Draft Revision, May 30, 1996
http://www.windev.synopsys.com/~dr/products/usb30/uploads/Main.ExternalSpecs/usb_le9.pdf

PHYs and Transceivers

- *PHY Interface for the PCI Express™, SATA, and USB 3.0 Architectures*, Version 4.00, Intel Corp.
<http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/phy-interface-pci-express-sata-usb30-architectures.pdf>
- *USB 2.0 Transceiver Macrocell Interface (UTMI) Specification*, Revision 1.05, Intel Corp., March 29, 2001
http://www.intel.com/technology/usb/download/2_0_Xcvr_macrocell_1_05.pdf
- *UTMI+ Specification*, Revision 1.0, ULPI Working Group, February 25, 2004
<http://www.ulpi.org/> (May require registration)
- *UTMI+ Low Pin Interface (ULPI) Specification*, Revision 1.1, ULPI Working Group, October 20, 2004
<http://www.ulpi.org/> (May require registration)
- *PDIUSBP11A Universal Serial Bus Transceiver Product Specification*, Philips Semiconductor, June 12, 2001
http://www.semiconductors.philips.com/acrobat_download/datasheets/PDIUSBP11A_3.pdf
- *OTG CoreKit Transceiver Specification*, CEA-2011, Revision 0.61, May 8. 2004
<http://global.ihs.com/> (Available for purchase)

- *Mini USB Analog Carkit Interface Specification*, CEA-936, Revision 2, December 2002
<http://global.ihs.com/> (Available for purchase)

SoC Buses

- *AMBA Specification*, Revision 2.0, ARM Ltd, ARM IHI 0001, 1999 (May require registration)
http://www.arm.com/products/solutions/AMBA_Spec.html
- *AMBA AXI(3) Protocol Specification*, v2.0, ARM Ltd, ARM IHI 0022C, 2003-2010 (May require registration)
http://www.arm.com/products/solutions/axi_spec.html

Host Controllers

- *eXtensible Host Controller Interface for Universal Serial Bus (xHCI)*, Revision 1.1 with errata to 12/20/13, Intel Corp., December 20, 2013 (Request from USB-IF)
<http://www.intel.com/technology/usb/xhcispec.htm>

Design Methodology

- *Reuse Methodology Manual: For System-On-A-Chip Designs*, Third Edition, Kluwer Academic Publishers, 2002
<http://www.springer.com/engineering/circuits+%26+systems/book/978-0-387-74098-0>

Customer Support

To obtain support for your product, choose one of the following:

- First, prepare the following debug information, if applicable:
 - For environment setup problems or failures with configuration, simulation, or synthesis that occur within coreConsultant or coreAssembler, use the following menu entry:

File > Build Debug Tar-file

Check all the boxes in the dialog box that apply to your issue. This menu entry gathers all the Synopsys product data needed to begin debugging an issue and writes it to the file `<core tool startup directory>/debug.tar.gz`.

- For simulation issues outside of coreConsultant or coreAssembler:
 - Create a waveforms file (such as VPD or VCD)
 - Identify the hierarchy path to the DesignWare instance
 - Identify the timestamp of any signals or locations in the waveforms that are not understood
- Then, contact the Support Center, with a description of your question and supply the above information, using one of the following methods:
 - For fastest response, use the SolvNet website. If you fill in your information as explained below, your issue is automatically routed to a support engineer who is experienced with your product. The **Sub Product** entry is critical for correct routing.

Go to http://solvnet.synopsys.com/support/open_case.action. Provide the requested information, including:

- Product: DesignWare Cores
- Sub Product: USB 3.0 Device, USB 3.0 Host, USB 3.0 Hub, USB 3.0 DRD
- Version: 3.30b
- Problem Type
- Priority
- Title: Provide a short summary of the issue or list the error message you have encountered
- Description: For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood

After creating the case, attach any debug files you created in the previous step.

- Or, send an e-mail message to support_center@synopsys.com (your email is queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
 - Include the Product name, Sub Product name, Product version, and Tool Version number in your e-mail (as identified above) so it can be routed correctly.
 - For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood
 - Attach any debug files you created in the previous step.
- Or, telephone your local support center:
 - North America:
Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
 - All other countries: <https://www.synopsys.com/support/global-support-centers.html>

Product Overview

This chapter provides an overview of the features and system interfaces, architecture, and verification environment of the DesignWare® Cores USB 3.0 Controller.

This chapter includes the following sections:

- “[General Product Description](#)” on page [58](#)
- “[System Overview](#)” on page [61](#)
- “[Hardware and Software Partition](#)” on page [63](#)
- “[DWC_usb3 Block Diagrams](#)” on page [64](#)
- “[Features List](#)” on page [65](#)
- “[Related Products](#)” on page [76](#)
- “[Speed and Clock Requirements](#)” on page [76](#)
- “[Area](#)” on page [77](#)
- “[Known Issues](#)” on page [78](#)

1.1 General Product Description

The Synopsys DesignWare Cores USB 3.0 Controller (DWC_usb3) is based on the *USB 3.0 Specification* from the USB Implementor Forum. The DWC_usb3 controller includes a verification environment that provides an SoC Integration Example Testbench to help you verify the DWC_usb3 controller with your specific configuration before integrating the controller into your SoC design. Synopsys also provides the coreConsultant tool for automated configuration, simulation, and synthesis of the DWC_usb3 controller.

Depending on the license(s) you purchase, you can configure the USB 3.0 Controller as any of the following:

Table 1-1 USB 3.0 Configurations

USB 3.0 Configuration	Description
USB 3.0 Device Controller	SuperSpeed, High-Speed, and Full-Speed
USB 3.0 Host Controller	SuperSpeed, High-Speed, Full-Speed, and Low-Speed Compatible with the xHCI specification from Intel Corporation
USB 3.0 Static Dual-Role Device (DRD) Controller	SuperSpeed, High-Speed and Full-Speed Low speed in Host mode only Supports either device or host operation separately, not simultaneously.
USB 3.0 Hub Controller	SuperSpeed only

[Figure 1-1](#) and [Table 1-2](#) on page 59 show the Device, Host, and DRD mode usage of the DWC_usb3 controller.

Figure 1-1 DWC_usb3 Controller Usage

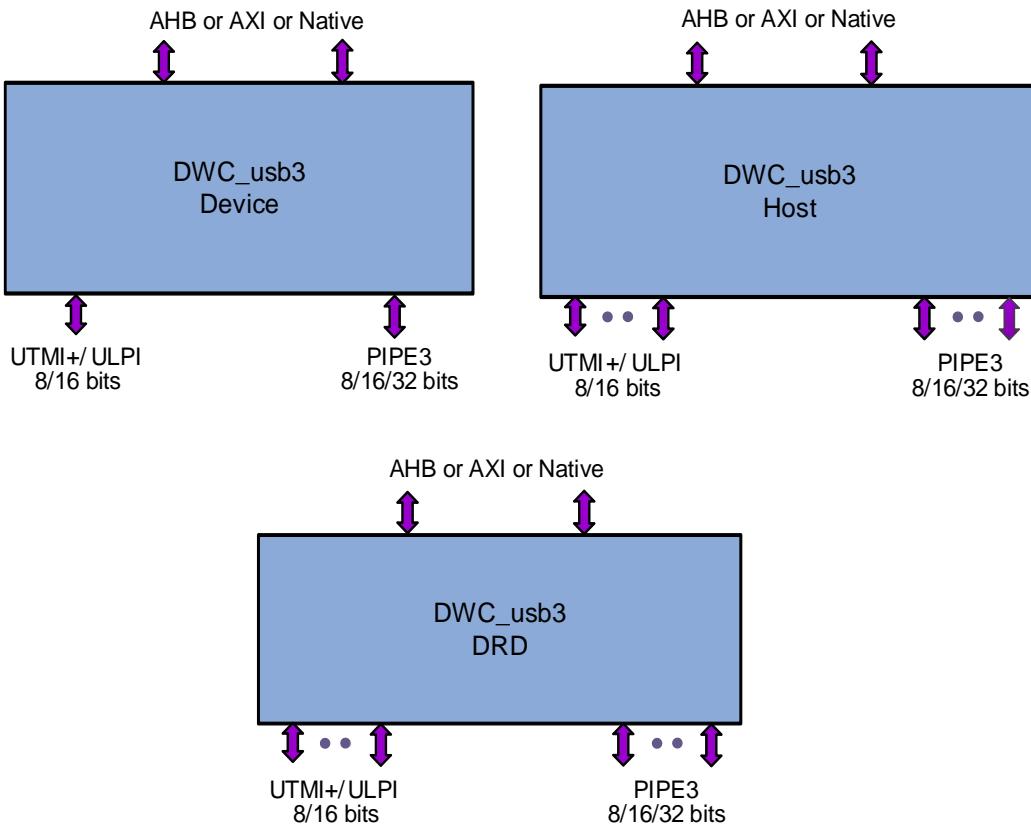
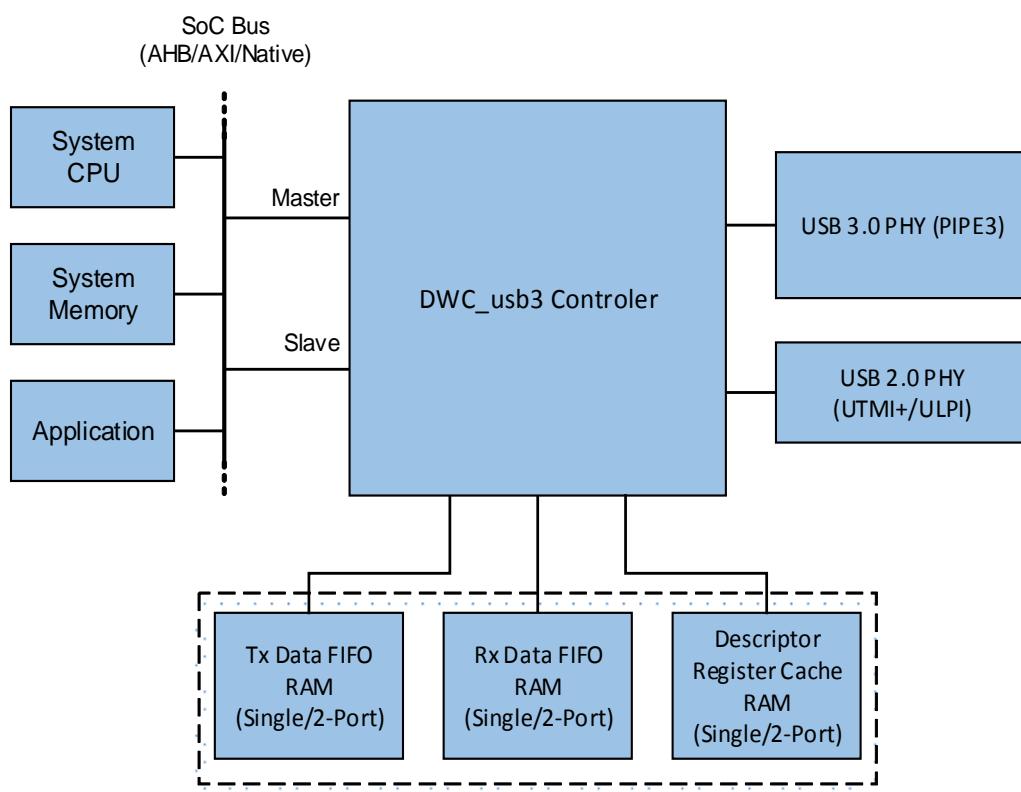


Table 1-2 DWC_usb3 Controller Usage

Mode	Usage
Device	<ul style="list-style-type: none"> ■ Supports only one port.
Host	<ul style="list-style-type: none"> ■ Supports m USB 2.0 ports and n USB 3.0 ports, where, n = 1 to 16, and m = 1 to 16. ■ The number of USB 2.0 ports (m) need not be equal to the number of USB 3.0 ports (n).
DRD	<ul style="list-style-type: none"> ■ Supports Device only on Port-1. ■ Functions either as a Device or a Host at a given time. For example, you cannot use Port-1 as a device at the same time Port-2 is used as a host. ■ For concurrent Device and Host operation, instantiate two DWC_usb3 controllers.

As Figure 1-2 shows, the DWC_usb3 controller bridges the SoC bus and USB 3.0/2.0 PHYs. The DWC_usb3 controller also includes RAM interfaces for data storage, descriptor caching, and register caching.

Figure 1-2 System-Level Block Diagram

The DWC_usb3 controller has the following interfaces:

- AHB, AXI,
- AHB, AXI,
- USB 3.0 PIPE3 PHY interface
- USB 2.0 UTMI+ (L3) or ULPI PHY interface

- Single- or 2-port interfaces for Tx data prefetching, Rx data buffering, and descriptor and register caching
- For SuperSpeed mode, three RAMs are recommended. For High-Speed mode, one or two RAMs (with one RAM for Tx/Rx and the other for caching) are sufficient.

For a description of Hub interfaces that the DWC_usb3 controller supports, refer to Chapter 8, “[Hub](#)”.

1.1.1 Applications

The DWC_usb3 controller is optimized for the following applications and systems:

- Portable electronic devices
- High-bandwidth applications

1.1.2 Compatibility and Quality

Standards Compatibility

The Synopsys DWC_usb3 controller is compatible with the standards listed in the “Reference Documentation” section of the *DWC SuperSpeed USB 3.0 Controller User Guide*.

Coding and Design Guidelines and Exceptions

The DWC_usb3 controller fully complies with the RTL coding practices specified in the *Reuse Methodology Manual*, Third Edition.

Testing and Quality Metrics

The DWC_usb3 controller development adhered to the following quality processes:

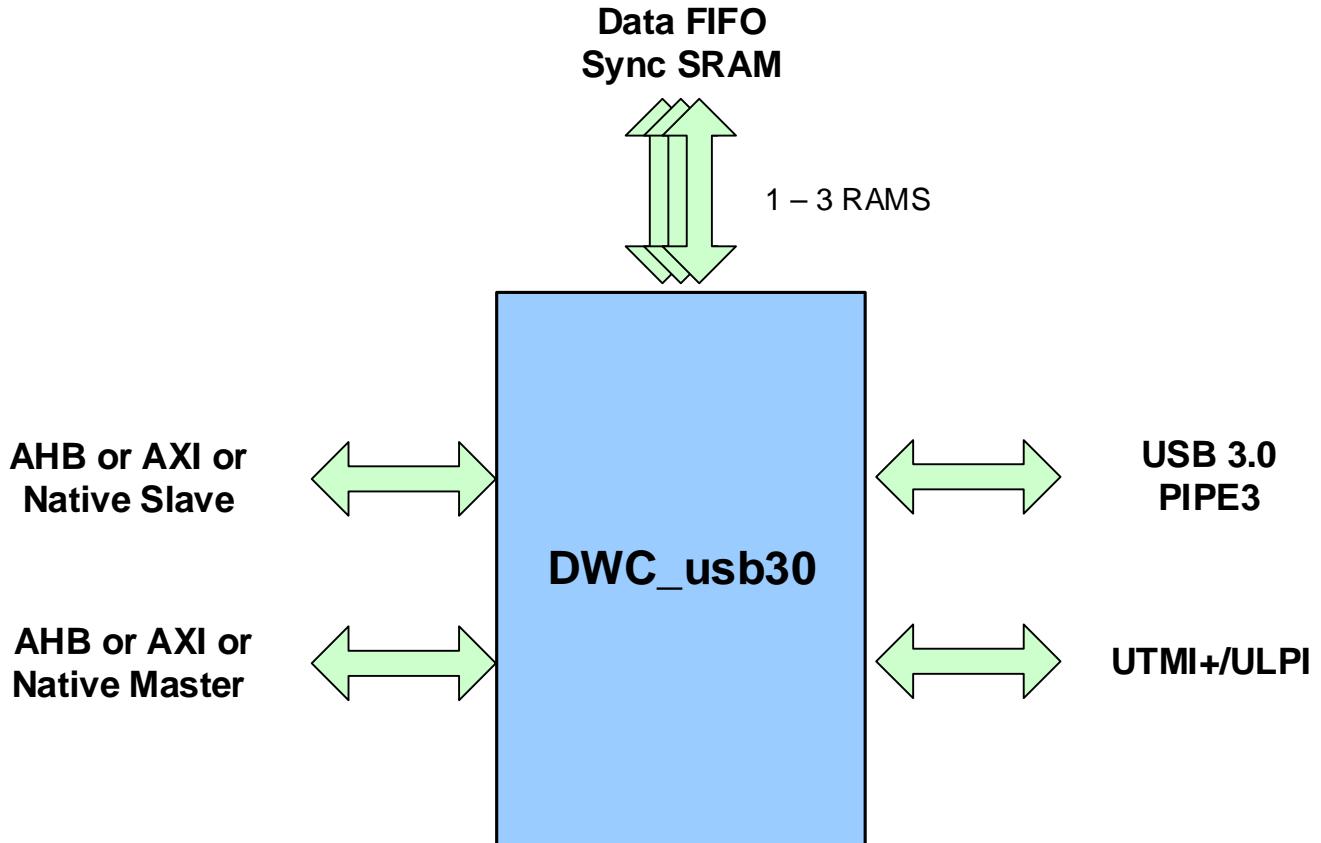
- Randomized multi-configuration verification
- Automated functional coverage
- Structured clock crossing implementation and checking
- Coverage and checker RTL assertion
- Self-documented clock domain crossing RTL coding style
- Well-defined inter-module interfaces

1.2 System Overview

Figure 1-3 shows the DWC_usb3 controller's standard and optional system interfaces, which are summarized in Table 1-3 on page 62.

- For information on DWC_usb3's block interfaces, see Chapter 2, “Architecture” .
- For information on connecting DWC_usb3 interfaces, see the “Integrating the Controller” chapter in the *DWC SuperSpeed USB 3.0 Controller User Guide*.

Figure 1-3 Interface Level Block Diagram



[Table 1-3](#) lists and describes all the standard interfaces present in all configurations of the DWC_usb3 controller.

Table 1-3 Standard Interfaces

Interface Name	Description
AHB/AXI/Native Master Interface	This interface allows the controller to act as a master to perform DMA operations.
AHB/AXI/Native Slave Interface	This interface provides the system CPU with read and write access to the controller's control and status registers (CSRs), and debug access to the RAMs.
RAM Interface(s)	<p>This interface connects the controller's FIFO controllers to an external 2-Port or Single-Port Synchronous Static RAM (2Port-RAM, SPRAM) used for storing receive and transmit transaction data and queues, and descriptor caching.</p> <ul style="list-style-type: none"> ■ 3-RAMs configuration: Rx Data FIFO RAM, Tx Data FIFO RAM, Descriptor/Register Cache RAM ■ 2-RAMs configuration: Rx/Tx Data FIFO RAM, Descriptor Cache ■ 1-RAM configuration: Rx/Tx/Descriptor/Register Cache Data RAM <p>The total RAM sizes for each configuration are configurable.</p>
USB 3.0 SuperSpeed PIPE3 Interface	This is the interface to USB SuperSpeed PHY.
USB 2.0 High-Speed UTMI+/ULPI Interface	This is the interface to HS, FS, and LS PHYs.
General Purpose I/O (GPIO)	You can use this interface for application-specific purposes or debugging.

Optional Interfaces

[Table 1-4](#) lists the interfaces that are built only if they are selected during configuration with the coreConsultant tool. For more information about configuration, see Chapter 4, “Parameter Descriptions” on page [189](#).

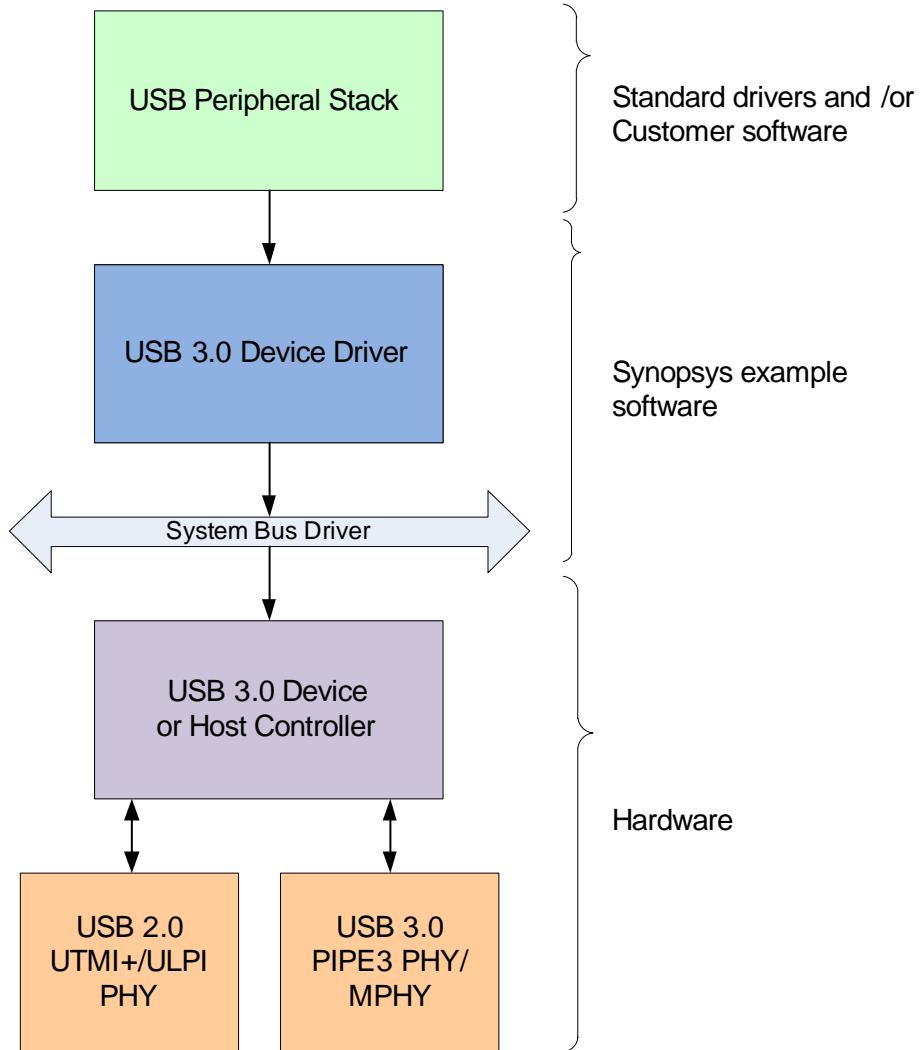
Table 1-4 Optional Interfaces

Interface Name	Description
USB 2.0 High-Speed UTMI+ L3/ULPI Interface	This is an optional vendor control interface for UTMI+.
Hibernation	When this feature is enabled, the controller saves some of its internal state to external modules named “Power Management Units,” or PMUs, when directed by software and the customer-supplied power controller.
External Buffer Control (EBC)	This feature provides an application hardware-level control to throttle the DMA read/write of data at the packet boundaries without involving software control.

1.3 Hardware and Software Partition

Figure 1-4 shows the hardware and software partitions for the DWC_usb3 controller.

Figure 1-4 Hardware/Software Partition



The hardware partition consists of the USB 3.0 Host/Device and the connected PHYs. Host or device drivers that connect the controller to the USB Peripheral stack are available from Synopsys and other third-party sources.

1.4 DWC_usb3 Block Diagrams

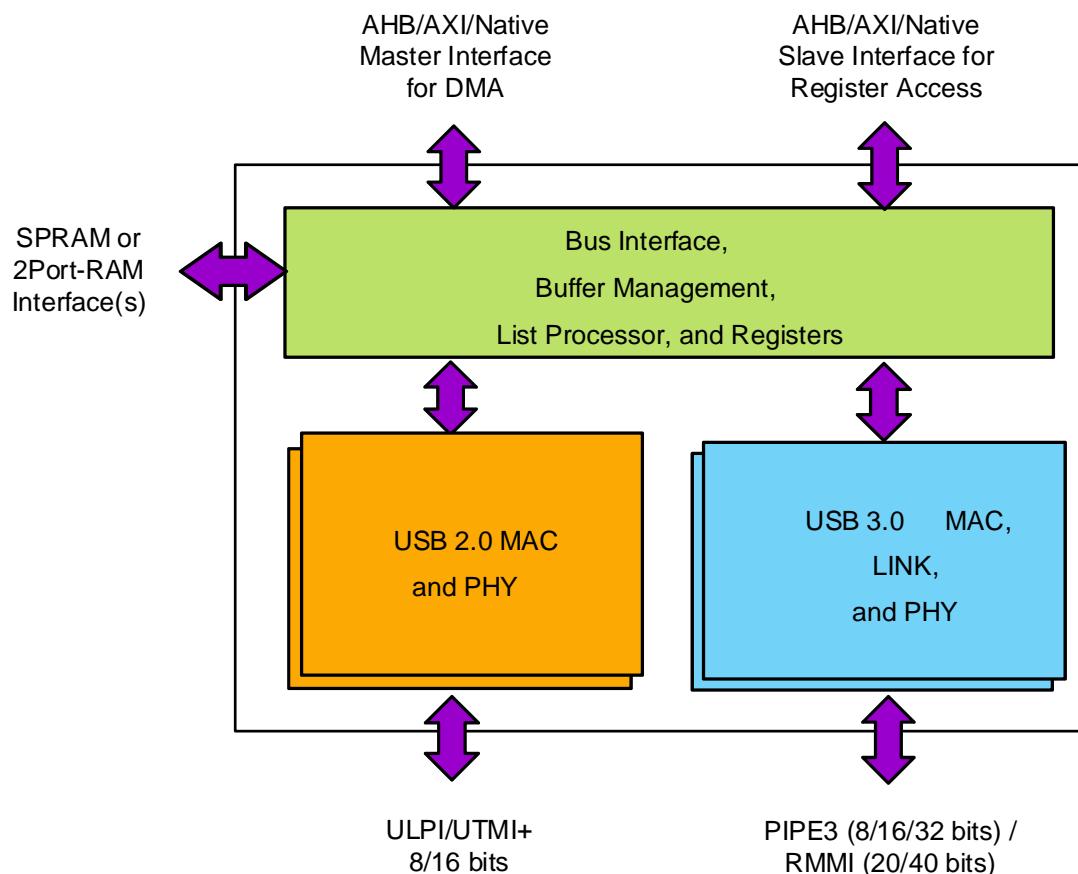
Figure 1-5 shows the main blocks of the DWC_usb3 controller as follows:

- The upper layer is common for USB 2.0 and USB 3.0 operation. This has the Bus Interface, Buffer Management block, List Processor for scheduling, and Control and Status Register (CSR) functions.
- USB 2.0 PHY and MAC layers
- USB 3.0 PHY, LINK, and MAC layers

Because you can connect a device as a USB 2.0 or USB 3.0 device, one of these operations is active at a given time:

- In Host mode, both USB 2.0 and USB 3.0 operations can be simultaneously active to concurrently support USB 2.0 and USB 3.0 devices.
- In multi-port Host mode, multiple instances of the PHY, Link, and MAC layers are instantiated. Buffer management also has separate Rx and Tx buffers for each bus instance.

Figure 1-5 **DWC_usb3 Main Blocks**



1.5 Features List

This section describes the features of the DWC_usb3 controller.

1.5.1 General Features

The DesignWare Cores USB 3.0 Controller supports the following:

- Certified ISO 26262 ASIL B Ready
For more information about the Automotive Safety feature in DWC_usb3, refer to Appendix F, “[DWC_usb3 Automotive Safety](#)” on page [567](#).
- Verilog-2001 RTL source code
- Software configurable architecture
- Design trade-offs for area, performance, and power
- Same programming model for SuperSpeed (SS), High-Speed (HS), Full-Speed (FS), and Low-Speed (LS)
- Well-defined clock domains for ease of integration:
 - PIPE3 PHY (125/250/500 MHz)
 - UTMI+ PHY (30/60 MHz) or ULPI PHY (60 MHz)
 - MAC (nominal 125 MHz for PIPE)
 - BUS clock domain
 - RAM clock domain
- coreConsultant to configure the controller
- Internal DMA controller
- Power-saving features
- LPM protocol in USB 2.0 and U0, U1, U2, and U3 states for USB 3.0
- Hardware-controlled LPM support in host mode
- Hardware-controlled LPM in USB 2.0 host

This is a USB 2.0 host-only feature where the host controller automatically detects that the downstream port is idle for a certain amount of time and autonomously initiates an LPM token to the connected device. If the device accepts, the link is put into the standard USB 2.0 suspend state.

- Dynamic FIFO memory allocation for endpoints
- Endpoint FIFO sizes that are not powers of 2, to allow the use of contiguous memory locations
- Keep-Alive feature in LS mode and (micro-)SOFs in HS/FS modes
- Software controlled standard USB commands (USB SETUP commands detected and forwarded to application for decoding)
- Hardware controlled USB bus level and packet level error handling
- Some registers implemented in the internal RAM to reduce gate count
- Low area design
- High performance
- Low MIPS requirement

- ❑ Driver involved only in setting up transfers and high-level error recovery
- ❑ Hardware handles data packing and routing to a specific pipe
- Descriptor caching and data pre-fetching used to meet system performance in high-latency systems
- Interrupt moderation

1.5.2 Application Interface Features

The DesignWare Cores USB 3.0 Controller supports the following interfaces:

- Application interfaces: AHB, AXI, or Native Bus
 - ❑ Slave interface for CSR and RAM Debug access
 - ❑ Master interface for internal DMA access
- 32-, 64-, or 128-bit data bus
- 32- or 64-bit address bus
- Independent Little or Big Endian (byte invariant) mode selection for both Slave and Master interface
- Independent Endian selection between descriptor and data fetch
- Hardware/software race condition avoidance in systems with bridge
- Concurrent read and write operations to get the best performance of USB 3.0 duplex operation in AXI bus mode
- AHB Slave interface:
 - ❑ Supports all AHB Burst types, including WRAP# supports
 - ❑ Does not generate Split, Retry, or Error responses
- AHB Master interface:
 - ❑ Supports Split, Retry, and Error AHB responses
 - ❑ Configurable burst type (SINGLE, INCR4, INCR8, INCR16, or INCR)
 - ❑ Single and INCR generated at the beginning and end of transfers to align burst boundaries; does not generate WRAP.
 - ❑ Handles fixed burst address alignment. Starting address of a burst is always aligned to burst-size; even though AHB specification does not require this, most memory/cache access require this. For example, if a burst of 8 is selected in a 64-bit wide data bus, the burst starting address will be 0, 64, 128, etc. SINGLE/INCR transfers are used to align the address.
 - Example 1: Consider starting address = 0, transfer size = 512 bytes, bus data width = 64 bits, burst sizes INC4, INC8, and INC16 enabled, and INCR = 0. The DMA transfers will be Address-0 BURST-16, Address-128 BURST-16, Address-256 BURST-16, and Address-384 BURST-16.
 - Example 2: Consider starting address = 8, transfer size = 512 bytes, data bus width = 64 bits, burst sizes INC4, INC8, and INC16 enabled, and INCR = 0. The DMA transfers will be Address-8 INCR (15), Address-128 INC16, Address-256 INC16, Address-384 INC16, and Address-512 SINGLE.
 - Example 3: Consider starting address = 8, transfer size = 520 bytes, data bus width = 64 bits, burst sizes INC4, INC8, and INC16 enabled, and INCR = 0. The DMA transfers will be Address-8 INCR (15), Address-128 INC16, Address-256 INC16, Address-384 INR16, and Address-512 INCR (2).

- Example 4: Consider starting address = 8, transfer size = 520 bytes, data bus width = 64 bits, burst sizes INC4, INC8, and INC16 enabled, and INCR = 1. The DMA transfers will be Address-8 INCR (66).
- Generates AHB BUSY cycles in AHB Master interface, for all RAM configurations
The BUSY cycles are less frequent when the RAM configuration is three 2-Port RAMs, because the RAMs are not shared and there are separate ports for read and write.
- Handles 1KB boundary breakup
- Native interface:
 - Supports transfer pipeline
- AXI Master interface:
 - The software can select the allowable burst lengths (single, burst 4/8/16/32/64/128/256 as applicable) by programming the GSBUSCFG0 register in CSR.
 - Handles fixed burst address alignment. Starting address of a burst is always aligned to burst-size; even though AXI specification does not require this, most memory/cache access require this. For example, if a burst of 8 is selected in a 64-bit wide data bus, the burst starting address will be 0, 64, 128, etc.
 - Unlike AHB, for AXI, the burst size need not be a power of two since AXI bus has un-encoded burst size port support. When INCR = 0, then only bursts of power of two are initiated, and when INCR = 1, then burst size of non power of two are supported.
 - Example 1: Consider starting address = 0, transfer size = 512 bytes, bus data width = 64 bits, burst sizes supported 4, 8, and 16, and INCR = 0 or 1. The DMA transfers will be, Address-0 BURST-16, Address-128 BURST-16, Address-256 BURST-16, and Address-384 BURST-16.
 - Example 2: Consider starting address = 8, transfer size = 520 bytes, bus data width = 64 bits, burst sizes supported 4, 8, and 16, and INCR = 0. The DMA transfers will be, Address-8 BURST 1, Address-16 BURST 2, Address-32 BURST-4, Address-64 BURST-8, Address-128 BURST-16, Address-256 BURST-16, Address-384 BURST-16.
 - Example 3: Consider starting address = 8, transfer size = 504 bytes, bus data width = 64 bits, burst sizes supported 4, 8, and 16, and INCR = 1. The DMA transfers will be Address-8 BURST 15, Address-128 BURST 16, Address-256 BURST-16, and Address-384 BURST-16.
 - Example 4: Consider starting address = 0, transfer size = 120 bytes, bus data width = 64 bits, burst sizes supported 4, 8, and 16, and INCR = 0. The DMA transfers will be, Address-0 BURST-8, Address-64 BURST-4, Address-96 BURST 1, Address-104 BURST 1, and Address-112 BURST 1.
 - Example 5: Consider starting address = 0, transfer size = 120 bytes, bus data width = 64 bits, burst sizes supported 4, 8, and 16, and INCR = 1. The DMA transfers will be Address-0 BURST-15.

- In the case of a non-MDWIDTH aligned access, the controller can do burst starting from the non-aligned address which is compliant with the AXI Specification (refer to Figure 10.1 in AMBA AXI Protocol Specification Version 1.0).

For example, if the MDWIDTH is 32 bits, the controller can do burst read of four from address "0x1". The first byte of the first beat is a don't care, and is ignored by the controller. In case of a write, the byte-enable is inactive for the first byte of the first beat.

If your system requires, all the burst address to be aligned to the MDWIDTH, then you can modify the lower address in your SoC as shown in the following example (for a 128-bit bus):

```
fixed_xm_araddr[3:0] = (xm_arsize == BYTE)? xm_araddr[3:0] :  
                           (xm_arsize == HALF_WORD) {xm_araddr[3:1], 1'b0} :  
                           (xm_arsize == WORD)? {xm_araddr[3:2], 2'b00} :  
                           (xm_arsize == DOUBLE_WORD)? {xm_araddr[3], 3'b000} : 4'h0 ;  
  
final_xm_araddr[31/63:0] = {xm_araddr[31/63:4], fixed_xm_araddr[3:0]};
```

- Software can select the number of outstanding read/write requests (1 to 16) made by the AXI Master Interface by programming the GSBUSCFG1 register in CSR. Read and write channels operate independently and may each have their own outstanding requests limited by the programmed value in GSBUSCFG1.
- Does not support data out-of-order transfers on AXI. For this reason, all transfers use the same ID (default 0) to ensure that the data is in-order.
- Capable of performing zero-wait state data transfers.
- Handles the AXI 4k boundary
 - Transfers are split to prevent crossing of the 4k boundary. Optionally, handles break up transfers on the 1k boundary by utilizing a software programmable option in the GSBUSCFG1 in the CSR.
- Supports cacheable accesses on the AXI Master Interface
- Does not support Atomic and Protected accesses on AXI Master
- AXI Slave interface:
 - Supports all AXI burst types
 - Supports one outstanding request at a time; does not assert arvalid/awvalid until the current transfer is complete implying that data transfer must be in-order
 - Does not support Atomic, Cacheable, and Protected accesses

1.5.3 USB 3.0 Device Features

The DesignWare Cores USB 3.0 Controller supports the following USB 3.0 Device features:

- Up to 16 bidirectional endpoints, including control endpoint 0
 - Software directly handles non-timing-critical and rarely occurring tasks, such as control transactions
 - Flexible endpoint configuration allows a single area optimized configuration meeting multiple applications/USB set-configuration modes
- During coreConsultant configuration, you can configure n number of endpoints, and post-silicon, software can map the USB endpoint to an endpoint resource number, even if the USB endpoint numbers are not contiguous.
- Dynamic mappable TxFIFOs to support more Tx-endpoints than the physical FIFOs
 - Simultaneous IN and OUT transfer support
 - 4 Gbps IN and 4 Gbps OUT bandwidth (interpacket delays and protocol overhead included)
 - Descriptor caching and data pre-fetching for predictable performance in high-latency systems
 - Hardware handles ERDY and burst
 - Hardware handles all data transfers
- Capability to set up multiple transfers without interrupting the host processor on every transfer
- Stream-based bulk endpoints with controller automatically initiating data movement
 - Isochronous endpoints with isochronous data in data buffers or external hardware FIFOs
 - Dual power rail designs with the hibernation feature
 - External Buffer Control (EBC) feature allows transfers to be setup in external FIFOs
 - Flexible Descriptor with rich set of features to support buffer interrupt moderation, multiple transfers, isochronous, control, and scattered buffering support
 - Multiple interrupt support – An endpoint interrupt can be mapped to a selected interrupt line during the endpoint configuration

1.5.4 USB Class-Specific Device Features

The DesignWare Cores USB 3.0 Controller supports the following USB Device features:

- Stream support for UASP application
- Gathering of scattered packet to support Ethernet Over USB. For example, multiple 64-byte scattered data gets packed in to a 1024 byte USB packet by hardware.
- Scheduling of multiple Ethernet packets without interrupt (multiple 1500-byte Ethernet packets can be scheduled without short packets causing interrupts). For example, you can instruct hardware to interrupt only after every five 1500-byte Ethernet packets.
- Variable FIFO buffer allocation for each endpoint allows you to optimize internal RAM. For example, you can allocate 64 bytes for a modem endpoint and 2 KB for a stream endpoint.
- For isochronous applications, scheduling of variable-length payloads for each microframe.
- Microframe precise scheduling for isochronous applications.
- Configurable endpoint type selection and dynamic FIFO allocation to facilitate multi-function/composite device implementation. During set-config or alternate-setting, device resources are reconfigured to meet the configuration or alternate setting requirements.
- Clock gating to optimize power for mobile devices.

1.5.5 USB 3.0 xHCI Host Features

The DesignWare Cores USB 3.0 Controller supports the following Host features:

- Up to 127 devices
- Up to 1024 interrupters
- Up to 15 USB 2.0 ports and 15 Super Speed ports
- Up to four SS bus instances, four HS bus instances, and one FS/LS bus instances
- xHCI1.1 compatible
- Standard or open-source xHCI and class drivers
- xHCI features:
 - Aggressive power management
 - Clean software and hardware interface
 - Memory access optimization
 - Interrupt Moderation
- Descriptor caching for predictable performance in high latency systems
- Concurrent IN and OUT transfers to get the full 8 Gbps duplex throughput (interpacket delays and protocol overhead included)
- Concurrent USB 3.0/2.0/1.1 traffic:
 - Designed so that USB 2.0 Devices do not degrade the overall throughput
 - Net bandwidth increased to 8.48 Gbps (8 Gbps USB 3.0 bandwidth plus 480 Mbps USB 2.0 bandwidth)
 - Dual power rail designs with hibernation feature
- Concurrent USB 3.0 transfers on each port (multiple bus instances). In a four-SS port and four-SS bus instance configuration, net throughput is 32 Gbps (4 * 4 Gbps IN and 4 * 4 Gbps OUT)
- Supports xHCI Debug Capability (DbC)
 - Event Ring Segment Table: Supports a maximum of 215 entries (that is, DCERST_Max is 15).

1.5.6 USB 3.0 Dual-Role Device (DRD) Features

The DesignWare Cores USB 3.0 Controller supports the following USB 3.0 DRD features:

- Static Device operation
- Static Host operation

1.5.7 USB 3.0 Hub (SuperSpeed Only) Features

The SuperSpeed Hub controller supports:

- SuperSpeed only – It can be integrated with an existing USB 2.0 Hub controller to create an USB 3.0 Hub controller.
- One upstream port
- 1 to 15 configurable number of downstream ports
- One time configurable software Hub descriptors stored in ROM
- Optional Xilinx Serial-Flash Interface, which overrides the controller
This enables the controller to use the descriptors stored in the Xilinx Serial-Flash instead of using the descriptors stored in the ROM.
- Optional JTAG port for debug purpose

For more information on USB 3.0 Hub, see Chapter 8, “[Hub](#)” .

1.5.8 Power Optimization Features

The DWC_usb3 controller implements both static and dynamic power reduction techniques at multiple levels.

- System/Architectural level:
 - Controller automatically puts the PHY into Low Power mode (P1, P2, and P3) during U1, U2, and U3 for dynamic power reduction
 - Area reduction (for static and dynamic power reduction)
 - Registers and queues implemented in RAM
 - Setup packets decoded by software (in device mode)
 - Protocol functions handled by controller instead of microprocessor, thus eliminating large switching activities
- RTL level:
 - Clock gating between U0-> U1/U2/U3 for dynamic power reduction
 - Clean, registered outputs reduce unwanted switching and input/output toggle filters (dynamic power reduction)
 - Chip-enable to RAMs driven inactive when no access to RAM
- Gate/physical level:
 - Clock-gateable flip-flops (80 to 90 percent) (dynamic power reduction) – fine-grained clock gating
 - Built-in module level clock gating – coarse-level clock gating
- PHY clock gating support during USB suspend, LPM, and session-off modes
- Link power management support (clock gating during LPM)

1.5.8.1 Clock Gating

Clock gating is a power saving technique in which the controller turns off clocks to its internal modules when they are not being used, that is, when the controller is idle and the following conditions are met.

- In host mode, the controller turns off clocks to its internal modules when
 - all SuperSpeed ports are in U1, U2, or U3 state;
 - all USB 2.0 ports are in Suspend or L1 Suspend state; and
 - the controller is idle.

When the controller is acting as a device, in SuperSpeed mode, the controller implements clock gating when the SuperSpeed link is in U1, U2, or U3 state and the controller is idle.

In USB 2.0 device mode, the controller implements clock gating when the link is in LPM-L1 or suspend, and the controller is idle.

1.5.8.2 Hibernation (2-Power Rail) Features

- When hibernation is supported, the controller saves some of its internal state to Power Management Units (PMUs), when directed by software and the customer-supplied power controller. This occurs when there is no connection or when the link is in a low power state such as L2 (Suspend) or U3.
- The PMUs may be implemented in a separate power rail (Vaux) than the controller (Vcc). They are small modules (~5K gates) so they consume significantly less power than the controller.
- When the save operation has completed, the controller's power rail may be turned off and the PMUs control the PHY interfaces.
- The PMUs detect wakeup conditions from the PHY and request power be reapplied to the controller.
- When power is reapplied to the controller, the saved state information is restored into the controller and the controller resumes control of the PHY interfaces.



- An additional license key is required for this add-on.
- Hibernation feature is supported only in device, host, and DRD configurations. It is not supported in hub configurations.

1.5.9 High Speed Inter-Chip (HSIC) Features

The DWC_usb3 controller supports HSIC through a UTMI or ULPI interface. If you enable the HSIC feature while configuring the controller, you can select HSIC or non-HSIC (standard USB 2.0 PHY) mode using a configuration port (`if_select_hsic`) or a register bit (`GUSB2PHYCFGn[26]`).

When selecting the HSIC feature, set the host side to HSIC mode first, then set the device mode. The reason for doing this is that if you set the device side to HSIC mode first, and if the host doesn't see a connection in HSIC mode, then you must deselect the device HSIC mode and select it again using the “`if_select_hsic`” setting or the register bit `GUSB2PHYCFGn[26]` to ensure that the device can connect to the host.



Note An additional license key is required for this add-on.

1.5.10 Area Reduction Features

The DWC_usb3 controller implements the following area reduction features:

- Single-Port RAM (SPRAM) or 2-Port RAM (2Port-RAM) option
SPRAMs are smaller than 2Port-RAMs.
- 2-port RAM (one port read only and another port write only) support instead of true dual port RAM.
2-port RAMs are much smaller than dual-port rams.
- Queues and most registers stored in RAM instead of flop-based registers
- Driver CPU handles non-timing-critical and rarely occurring events, such as Setup decodes, reducing the controller area

1.5.11 Performance Features

The DWC_usb3 controller implements the following performance features:

- Descriptor caching
- Packet prefetch – Device mode
- Multiple transfer queuing support
- Interrupt moderation support
- Packet packing to get full available bandwidth – Host mode
- Concurrent features:
 - Rx and Tx operations
 - Port Transfers – Host Mode
 - USB 3.0 and USB 2.0 transfers – Host Mode

1.5.12 MIPS Reduction Features

The DWC_usb3 controller implements the following SoC processor MIPS reduction features:

- Transfer level scheduling handled by hardware for both Non-Periodic and Periodic transfers – no software intervention needed within a USB transfer
- Support for even scheduling multiple USB transfers, avoiding interrupts to software
- Scattered data buffering support
- Scattered packet support (Ethernet over USB application) to avoid software to copying and creating USB packets
- Byte-addressing support to avoid any software double copying requirement
- “Buffer complete” interrupt moderation support
- Hardware-only scheduling for high data rate uncompressed video-type applications

1.5.13 Debug Features

The DWC_usb3 controller implements the following debug features:

- CPU read write access to all RAMS
- Most critical state machines’ states read through register interface
- FIFO status information visible to software
- Support for upper layer loop back test for testing DMA read and write operation and interrupt generation without any USB traffic (in Device Mode)
- 64 bit logic analyzer trace port for observing several internal states at the same time.
- Optional JTAG port (Hub Mode only)

1.5.14 Automotive Safety Features

The DWC_usb3 controller (Device, Host, and DRD only) has been certified as ISO 26262 ASIL B Ready. You must have a valid license to avail of the documentation and features included in the DWC_usb3 Automotive Safety Package. For more information about these license requirements, refer to the *DWC SuperSpeed USB 3.0 Controller Installation Guide*.

The DWC_usb3 includes a configuration parameter that enables ECC generation, correction, and checking of every access in the following RAMs: Cache, Tx FIFO, and Rx FIFO RAMs. When you set this parameter, you can also program ECC registers. For more information about the ECC configuration parameter, refer to the “Enable ECC Generation and Checking for RAMs?” parameter option in Chapter 4, “[Parameter Descriptions](#)”. For more information about the ECC registers, refer to “Register Descriptions” in the *DWC SuperSpeed USB 3.0 Controller Programming Guide*.

For more details about the other deliverables in the DWC_usb3 Automotive Safety Package, refer to Chapter F, “[DWC_usb3 Automotive Safety](#)”.

1.5.15 System Bandwidth Requirements

Unlike USB 2.0, the USB 3.0 SoC bus/system memory access must have low latency. In addition to meeting bandwidth requirements, the controller's buffer size must be configured to hide data access latency. For example, in a USB 2.0 system, a 1 KB packet takes approximately 20 μ s on the USB 2.0 cable. In a USB 3.0 system, it takes only 2.2 μ s on the USB 3.0 cable. If your system latency is large, you must have larger buffers to hide system latency. If all the accesses are high-latency, then eventually larger buffers run out of data and result in flow-control.

Unlike USB 2.0, in USB 3.0 it is recommended that your system-bus access latencies are in clocks rather than in microns. This prevents system latency, reducing the USB 3.0 throughput. In addition, you must not use low-speed buses like SDIO, which do not support DMA capabilities. Their bandwidth is also much lower than USB 3.0.

The FIFO buffers in the controller are only required for latency reasons. The main USB buffers are allocated in system buffers. In USB 2.0, the buffer size depends on the operating system. For example, Microsoft typically allocates 64 KB ping/pong buffers for USB transfer; Linux allocates 128 KB buffers; and MAC-OS allocates 1MB buffers. A 64 KB USB transfer takes 1.3 ms in USB 2.0 mode but only takes 164 μ s in USB 3.0 mode. To avoid too many XferComplete interrupts, larger buffers are recommended in USB 3.0 mode.

Because USB 3.0 supports burst transfers, larger system buffers allow efficient burst usage for performance.

A single port root-hub host that supports 8.48 Gbps duplex operation must support a 8.48 Gbps data rate on its system-bus; hence, a 32-bit AHB bus must run at minimum 265 MHz just for USB traffic.

In the case of a host with a four port root-hub that supports concurrent transfers on all of the four ports, the 32-bit AHB must run at a minimum of 1060 MHz to meet the 32.48 Gbps (four SS Bus Instances and one common HS Bus Instance, i.e., $4 * 8 + 0.48$ Gbps) bandwidth requirement. Or, it must run at a 265 MHz bus in 128-bit mode and a 265 MHz 2-Port RAM to support the bandwidth.

The previous numbers are just the USB 3.0 bandwidth requirement. In addition, the SoC bus and system memory must provide additional bandwidth to the CPU and the rest of the peripherals in the SoC.

1.6 Related Products

The following products are available for the DWC_usb3 controller through separate part numbers and written agreements.

- Device drivers
- Demo boards

For more information, contact your Synopsys sales representative.

1.7 Speed and Clock Requirements

The USB 3.0 provides ten times more bandwidth than USB 2.0. Therefore, the SoC bus and system memory bandwidth are also much higher to meet the USB 3.0 bandwidth.

The bus width, bus clock, RAM clock, and RAM mode (SPRAM or 2-Port RAM) must be chosen to meet the required bandwidth for the USB 3.0.

- A USB 2.0 system with a 32-bit AHB needs to run at a minimum of 15 MHz to meet 480 Mbps. In contrast, a USB 3.0 device system needs to run at a minimum of 125 MHz to support a 4 Gbps IN or 4 Gbps OUT transfer at a given time. The RAM clock needs to be a minimum of 250 MHz for a SPRAM or 125 MHz for a 2-Port RAM configuration.
- If the device supports simultaneous IN and OUT transfers, then a 64-bit AHB needs to run at 125 MHz to meet 8 Gbps throughput. The RAM clock needs to be a minimum of 250 MHz for a SPRAM or 125 MHz for a 2-Port RAM configuration.
- The RAM clock needs to be a minimum of 125 MHz to meet the USB 3.0 turnaround time.
- Both the RAM clock and the bus clock needs to be a minimum of 60 MHz in both the 16-bit UTMI (30 MHz) and 8-bit UTMI/ ULPI (60 MHz) modes to meet the USB HS turnaround time. The reason is that on a token request in the MAC clock domain, the data needs to be fetched from the FIFO in the RAM clock domain.

Note that in the USB 2.0 mode, the RAM clock is always same as the bus clock (that is, GCTL.RAMClkSel = 2'b00).

- In device mode, to operate with a slower bus_clk frequency, based on the connected speed (SS or 2.0), you can program GCTL.RAMClkSel to either pipe_clk, pipe_by_2_clk, or mac2_clk (utmi/ulpi clock). Setting of RAMClkSel to any clock other than bus_clk, decouples the bus_clk frequency from the USB minimum ram_clk frequency requirements for functional correctness.
- The minimum RAM/Bus clock speed to meet the USB turnaround time in FS Mode is 15 MHz.

In the host mode, if GCTL.SOFTITPSYNC is enabled, the DWC_usb3 controller uses the ref_clk to count the frame duration and generates ITP. If GFLADJ.GFLADJ_REFCLK_LPM_SEL is selected, both ITP and SOF counters are generated from ref_clk. The ITP must be delivered accurately with an allowed jitter margin of 32ns. This requires an accurate ref_clk with a period that is an integer multiple. However, with GFLADJ.GFLADJ_REFCLK_LPM_SEL, frequencies 16/17/19.2/24/39.7/40 MHz with a period that is not an integer multiple can be used for ref_clk.

If GCTL.SOFTITPSYNC and GFLADJ.GFLADJ_REFCLK_LPM_SEL are not enabled, the USB 3.0 Controller uses the UTMI(ULPI)_CLK of the first port to count the frame duration and generates ITP. Traditionally, the UTMI(ULPI)_CLK are required to be 500 ppm but, for the ITP based on the UTMI(ULPI)_CLK, this exceeds the jitter margin. To maintain the jitter within the allowed range, the ref_clk (if SOFITPSYNC is enabled) and the UTMI(ULPI)_CLK need to be 50 ppm (100ppm also meets the requirement).

Table 1-5 on page 77 summarizes this behavior.

Table 1-5 DWC_usb3 Behavior Depending on GCTL.SOFITPSYNC and GFLADJ.GFLADJ_REFCLK_LPM_SEL

If...	Then...
GCTL.SOFITPSYNC and GFLADJ.GFLADJ_REFCLK_LPM_SEL are not enabled	The DWC_usb3 uses the UTMI(ULPI)_CLK of the first port for ITP and SOF counters.
GCTL.SOFITPSYNC is enabled	The DWC_usb3 uses ref_clk for ITP counter when all the 2.0 ports are suspended.
GFLADJ.GFLADJ_REFCLK_LPM_SEL is enabled	The DWC_usb3 always uses ref_clk to generate SOF and ITP counters irrespective of the state of the ports.

If the host configuration has multiple USB 2.0 ports, and each PHY on the port generates an asynchronous UTMI(ULPI)_CLK, then the SOF on the ports, other than first port, might have a jitter of 16.66 ns. This will cross the jitter margin of 8.32ns allowed for high speed port's SOF repeatable jitter margin. In order to maintain the jitter within the allowed range, it is recommended that all the USB 2.0 ports have synchronous UTMI(ULPI)_CLK.

1.8 Area

Table 1-6 shows the DWC_usb3 configuration baseline gate size for Device, Host, and Hub modes.

Table 1-6 DWC_usb3 Area

Mode	Area
Device Area – 4 endpoints	160k gates
Host Area – 1 port	410k gates
Hub Area – 4 ports	320k gates

1.9 Known Issues

The following sections list the known features and verification limitations for the current release of the DWC_usb3 controller.

1.9.1 Device

- In device mode, the Device does not support a bInterval setting of 15 or 16 (which equates to a polling interval of 214 or 215 microframes). The ITP frame counter is 14 bits wide, so in general USB devices cannot detect service interval boundaries unless they can count more than 0x3FFF microframes. Host driver software (for example, the USBD) typically polls devices every 32 microframes regardless of whether they request a slower polling interval. In addition, it is not clear that there is an application that requires a polling interval greater than 214 microframes.
- There is no limitation in using a Device in Mass-Storage application for BOT and UASP class, Periodic (ISOC and Interrupt), and ethernet class applications.
- Only control endpoint-0 is supported. Support for multiple control endpoints is not needed for most device applications. Though support for multiple control endpoints is coded, this configuration has not completed the entire control abort error testing.

1.9.2 Host

- The host controller does not check for Isoch Buffer Overrun or Bandwidth Overrun Error conditions. The software driver must not create these programming error conditions (which is the expected functional operation).
- The host controller does not generate Endpoint Not Enabled Error when an invalid doorbell is rung. The RTL ignores all invalid doorbell rings.
- The host controller does not adjust the bus interval automatically based on the received Bus Interval Adjustment Notifications from the devices. A workaround for this limitation is described in the section “Host Mode – Miscellaneous Topics” in the “Integrating the Controller” chapter, of the *DWC SuperSpeed USB 3.0 User Guide*.
- The host controller does not support multiple Event Data TRBs in a single Transfer Descriptor (TD). In a TD, the Event Data TRB must be the last TRB with the Chain bit set to ‘0’. In addition, the host controller has limited support for Event Data TD’s (an Event Data TRB that is not chained at the end of a TD; the TD consists of just one Event Data TRB). In some scenarios, the EDTLA reported from an Event Data TD is incorrect.
- The host controller does not calculate the system bus bandwidth requirement when it handles Configure Endpoint commands. Software must calculate the system bus bandwidth requirement of all the endpoints that it intends to configure. It must also ensure that the system bus bandwidth requirement does not exceed the bandwidth that the system can provide.
- The xHCI Specification requires that software setup the entire TD for an Isochronous IN endpoint before Isochronous Scheduling Threshold (IST). However, it does not require that software always setup MaxPacketSize * (MBS+1) * (Mult+1) bytes of TD. The controller requires both.
- xHCI Extended capabilities:
 - I/O Virtualization is not supported.
 - Local Memory is not supported.
 - Debug Capability can be enabled only when the Master Bus Data Width is greater than 32 bits.
- P3 in U2 (GUCTL1[25]) is not supported if SOFITPSYNC and LPM_SYNC are 0.

- With 3.00a release of the controller, the xHCI Errata, Contiguous Frame ID capability (CFC) is added. With this, a legacy xHCI driver not updated for CFC and setting up ISOC TDs with SIA=0 could result in some Missed Service Error from the xHC. This is a known behavior. With CFC enabled, xHC cannot distinguish between legacy drivers setting SIA=0 and a CFC compliant driver setting SIA=0.
- When using 16-bit UTMI and multiple ports with asynchronous UTMI clocks, the host controller does not meet the USB turnaround time with five Hubs.

1.9.3 Hub

- The controller is a SuperSpeed-only Hub product. You need a companion USB 2.0 Hub.
- JTAG mode is not hardware or software tested.

1.9.4 Unsupported Features

- OTG
- ADP
- SSIC
- IC-USB, 3-pin/6-pin USB1.1 serial interface, I2C, and CarKit functions
- Synchronous reset mode

2

Architecture

This chapter describes the block diagrams, architecture, sub-block details, clock domains, and data flow of the DWC_usb3 controller. This chapter also provides example SoC and PCIe integration scenarios.

- “[DWC_usb3 Block Diagrams](#)” on page [82](#)
- “[Logical Hierarchy, Clock Domains and Data Flow](#)” on page [83](#)
- “[System Memory Descriptor and Data Buffers](#)” on page [97](#)
- “[Configuration-Dependent Flexible Resource Allocation](#)” on page [98](#)
- “[Clock Generation and Clock Tree Synthesis \(CTS\) Requirements](#)” on page [101](#)
- “[USB 3.0 PHY Interface Unit \(U3PIU\)](#)” on page [114](#)
- “[USB 2.0 PHY Interface Unit \(U2PIU\)](#)” on page [120](#)
- “[USB 3.0 Link Power Management](#)” on page [121](#)
- “[USB 3.0 Link \(U3LINK\)](#)” on page [129](#)
- “[USB 3.0 MAC \(U3MAC\)](#)” on page [130](#)
- “[USB 3.0 Protocol Transaction Layer \(U3PTL\)](#)” on page [131](#)
- “[USB 2.0 MAC \(U2MAC\)](#)” on page [134](#)
- “[USB 2.0 ROOT HUB \(U2RHB\)](#)” on page [137](#)
- “[Buffer Management Unit \(BMU\)](#)” on page [139](#)
- “[List Processor \(LSP\)](#)” on page [147](#)

2.1 DWC_usb3 Block Diagrams

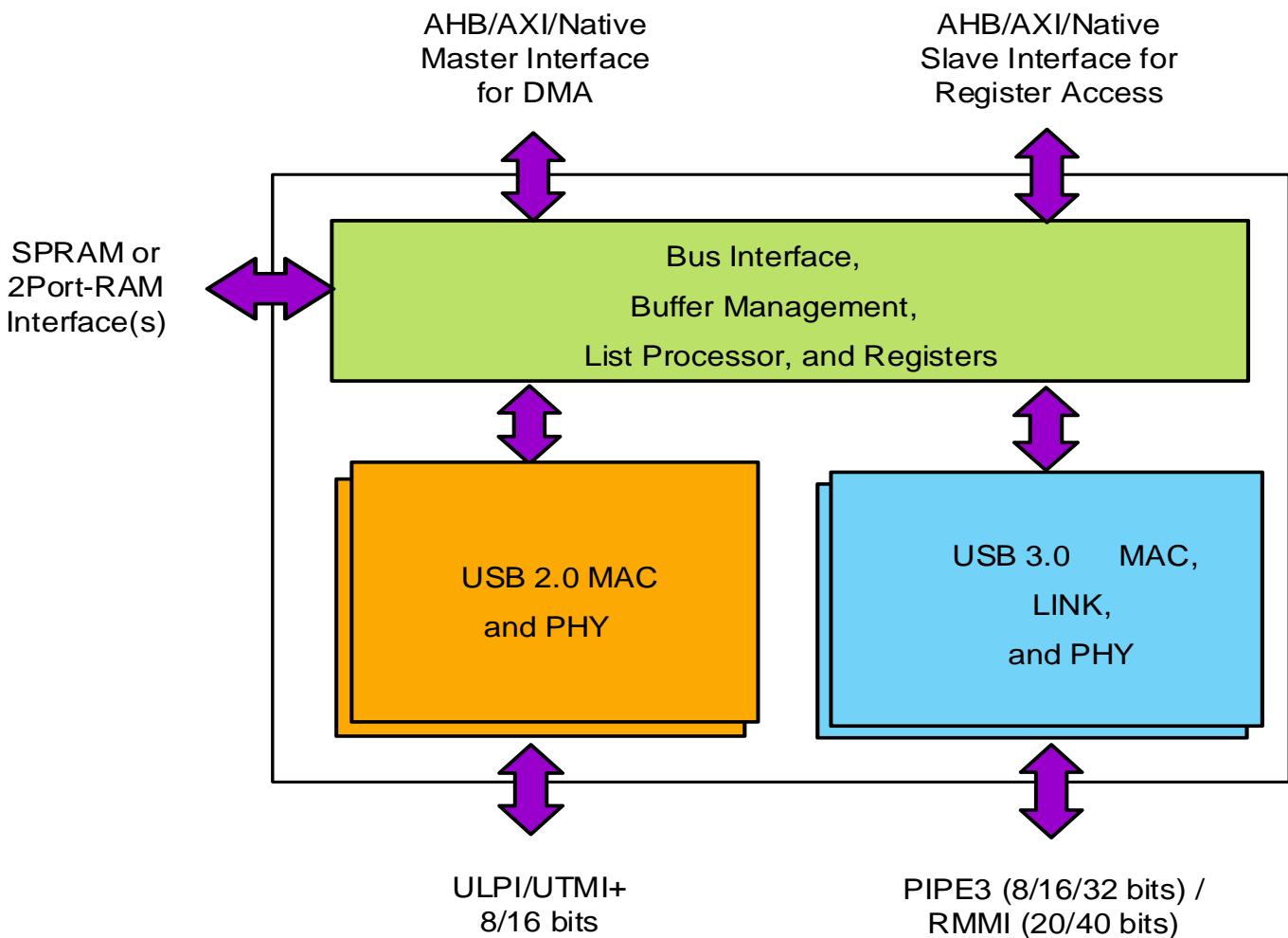
Figure 2-1 shows the main blocks of the DWC_usb3 controller as follows:

- The upper layer is common for USB 2.0 and USB 3.0 operation. This has the Bus Interface, Buffer Management block, List Processor for scheduling, and Control and Status Register (CSR) functions.
- USB 2.0 PHY and MAC layers
- USB 3.0 PHY, LINK, and MAC layers

Because you can connect a device as a USB 2.0 or USB 3.0 device, one of these operations is active at a given time:

- In Host mode, both USB 2.0 and USB 3.0 operations can be simultaneously active to concurrently support USB 2.0 and USB 3.0 devices.
- In multi-port Host mode, multiple instances of the PHY, Link, and MAC layers are instantiated. Buffer management also has separate Rx and Tx buffers for each bus instance.

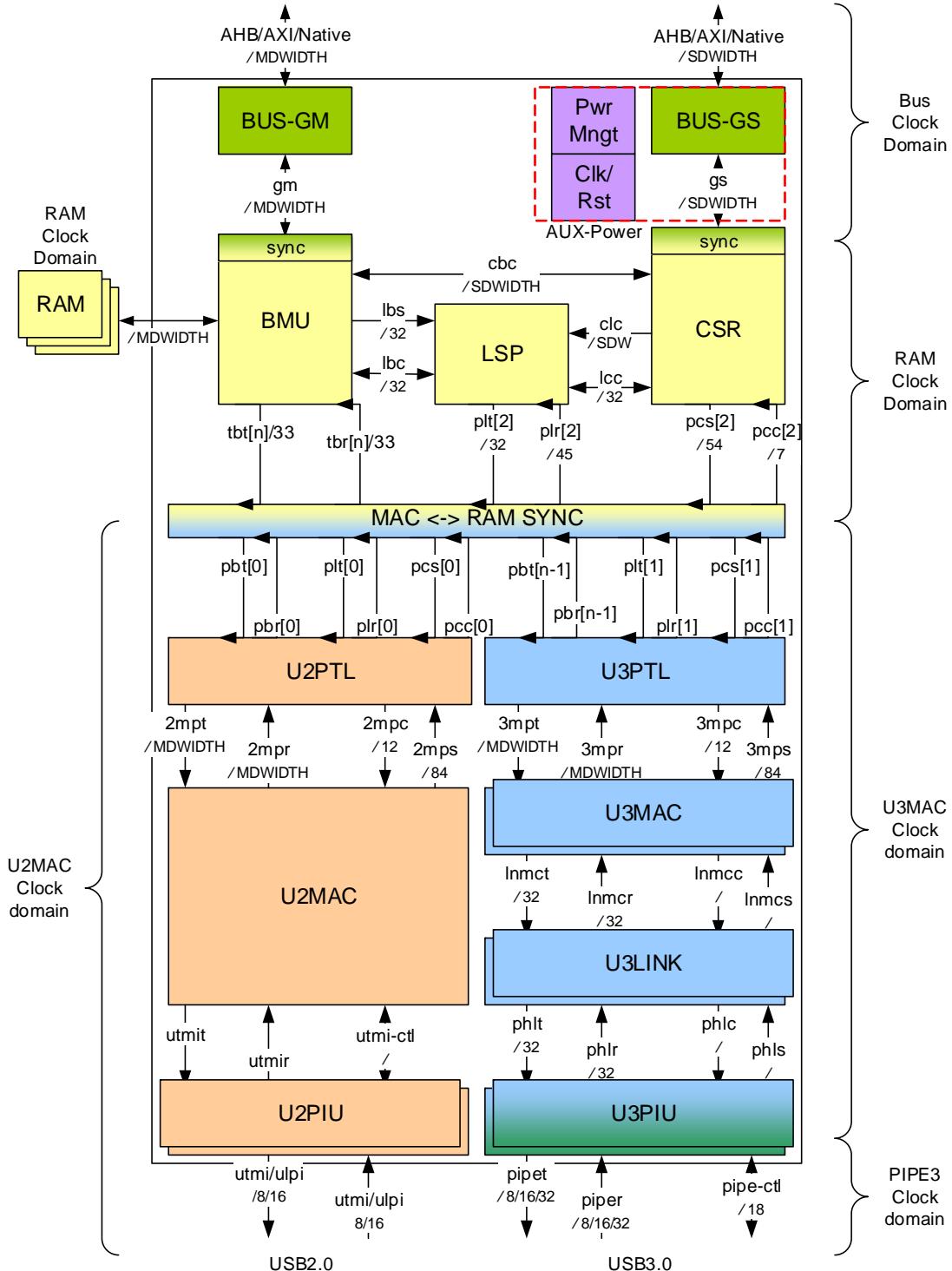
Figure 2-1 **DWC_usb3 Main Blocks**



2.2 Logical Hierarchy, Clock Domains and Data Flow

Figure 2-2 shows the logical hierarchy of the USB 3.0 Controller, clock domains and data flow. The domains illustrated in are described in “[Block Descriptions](#)” on page 84.

Figure 2-2 DWC_usb3 Clock Domains and Data Flow



2.2.1 Block Descriptions

The DWC_usb3 controller has the following major blocks:

Table 2-1 **DWC_usb3 Controller Functional Blocks**

Block Name	Description
Generic BUS Slave Interface (BUS-GS)	The Generic Bus Slave interface module supports AHB, AXI. This interface provides software access to all the internal registers. In addition, the RxFIFO, TxFIFO, and Descriptor Cache RAMs are accessible for debugging.
Generic BUS Master Interface (BUS-GM)	The Generic Bus Master interface module supports AHB, AXI. This interface provides DMA support for the controller to transfer data to and from the system memory.
Clock/Reset and Power Management Units (CLK/RST/PWRMNGT)	The reset module manages hard resets, soft resets, and PHY resets. The clock module manages clock generation for all the modules for LS/FS/HS/SS operation modes, configurations, and PHY selection. The power management unit controls all clock gating and power gating. It also has a few non-powered-down registers. All BC registers are housed here.
Control and Status Registers (CSR)	All the flop-based registers are implemented in this module. It also provides the interface to the Buffer Management Unit (BMU) module to access the registers implemented in the RAM. The LSP also has a dedicated interface to the CSR for register access. In addition, control/status signals are passed across CSR and LSP/BMU/PTL/MAC/LINK/PIU modules.
List Processor (LSP)	The List Processor handles the transfer scheduling in both Device and Host modes. It interprets the transfer descriptor's setup by the software and manages USB transfers. Devices do reactive scheduling to meet host requests. In Host mode, the LSP initiates scheduling and manages multiple devices in the USB tree. It manages data and descriptor access through BMU by issuing commands to the BMU queue.
Buffer Management Unit (BMU)	The BMU manages data and descriptor DMA fetches. It prioritizes DMA fetches from different queues before issuing DMA requests to the BUS-GM. The BMU has one RxFIFO (RxDFIFO) to receive data, one for USB 2.0 in Host mode, and one for each USB 3.0 concurrent port. Each RxFIFO has a corresponding RxQueue to which LSP queues in RxData and RxDescriptor write back commands. The TxFIFOs (TxDFIFO) provide prefetch buffers for the IN endpoints in Device mode and OUT transfers in Host mode. Because devices cannot predict the endpoint from which the host requests data, each active endpoint in Device mode has a TxFIFO. For Host mode to support concurrent USB 2.0 and USB 3.0 transfers, one dedicated TxFIFO is allocated for the USB 2.0 OUT transfers. Because concurrent transfers are supported on each USB 3.0 port, one TxFIFO is provided to each USB 3.0 port. Each TxFIFO has a corresponding TxQueue to which LSP issues Tx fetch requests. There is also one TxINFO FIFO (TxINFO), which holds the packet size, stream ID, and micro-frame information. Descriptor Fetch Queue to hold descriptor fetch requests from LSP. One EventQ per interrupt to which LSP queues events to be sent to the system CPU Descriptor FIFO to hold fetched descriptions

Block Name	Description
Synchronizer (SYNC)	SYNC manages synchronizers for all the clock crossings. Manages static routing of HS and SS traffic from U2PTL and U3PTL in Device mode. Manages dynamic routing of dynamic data from both U2PTL and U3PTL in Host mode.
USB 3.0/USB 2.0 Protocol/Transaction Layer (UxPTL)	<p>The UxPTL is responsible for:</p> <ul style="list-style-type: none"> ■ Managing all the USB protocol details. ■ Handling Toggle Sequence/Retry for USB 2.0 ■ Supporting up to 16 bursts for USB 3.0 ■ Maintaining transaction parameters until transaction finishes ■ Managing packet data routing to/from UxMAC
USB 3.0/2.0-MAC (UxMAC)	<p>The UxMAC is responsible for:</p> <ul style="list-style-type: none"> ■ Packet Rx/Tx Ctrl ■ Host mode <ul style="list-style-type: none"> □ MAC for FS, LS, HS or SS operation by speed detection □ Downstream port state machine □ In concurrent multi-port host, one MAC3 per port is instantiated. ■ Device mode <ul style="list-style-type: none"> □ MAC for FS, LS, HS or SS operation by speed detection □ Upstream port state machine ■ CRC32 generation and checking for USB 3.0 ■ CRC5/CRC16 generation and checking for USB 2.0 ■ Link management ■ Port Router (PRTRTR) <ul style="list-style-type: none"> □ In Host mode, the port is routed to FS/LS MAC □ Supports multiple ports
USB 3.0 Link (U3LINK)	U3LINK provides packet framing, checking, skip and link training, and control over link and power state. In Host and Hub mode, one link per port is instantiated.
USB 3.0 PIU (U3PIU)	<p>U3PIU converts 125 MHz (32-bit), 250 MHz (16-bit), and 500 MHz (8-bit PIPE3) interface into Link data streams of 125 MHz at 32-bits. It performs scrambling. In Host and Hub mode, one U3PIU is instantiated for each port and ring buffer.</p>
USB 2.0 PIU (U2PIU)	U2PIU converts the 8-bit UTMI+ and 8-bit ULPI interfaces into a 16-bit UTMI interface for the MAC. In Host mode, one per port is instantiated.

2.2.2 Clock Domains

The DWC_usb3 has the following clock domains:

- USB 3.0 PIPE3 PHY clock
- USB 2.0 PHY/MAC clock
- 125 MHz MAC3 clock, which is divided PIPE3 clock
- Bus clock
- RAM clock to meet both MAC and BUS domain bandwidth
- Suspend Clock

The following section describes clock usage during SS Only or SS and HS (Host mode) operation:

- USB 3.0 PIPE3 PHY clock
 - This clock is 125 MHz with 32-bit PHY, 250 MHz with 16-bit PHY, or 500 MHz with 8-bit PHY.
 - Separate Rx and Tx clock input pins are provided to make external PHY integration easier. The Tx clock can be an early version of the PHY clock to compensate for output PAD delays.
 - During the lowest SS PHY power-down state (P3), the PHY does not drive this signal; therefore, a suspend clock signal is provided in its place.
- USB 2.0 PHY/MAC clock:

This clock is 30 or 60 MHz in UTMI+ PHY mode, or 60 MHz in ULPI mode

- MAC3 clock:

The MAC3 clock is a divided version of the PIPE3 PHY clock. In Device or single-port Host mode, the MAC3 clock is synchronous to the PIPE3 clock, but may be delayed to tolerate the larger MAC3 clock insertion delay. In multi-port Host and Hub mode, the MAC3 clock and the PIPE3 clock may be asynchronous.

- The bus clock:

The bus clock is system-dependent. The SoC bus bandwidth must be at least the same as USB 3.0 bandwidth. For example, the bus clock needs to be at least 125 MHz with a 64-bit SoC bus to satisfy the device duplex performance requirement of 8 Gbps ($125 \text{ MHz} * 64 = 8 \text{ Gbps}$). For example, in the case of a four-port concurrent host, the bus clock must be at least 250 MHz at 128 bitd to meet 32 Gbps ($250 \text{ MHz} * 128 = 32 \text{ Gbps}$). For more details, refer to “[Minimum Clock Frequencies: bus_clk, ram_clk](#)” on page [515](#).



A system bus interface can have different master and slave types, but they must use the same bus clock.

- RAM clock to meet both MAC and BUS domain bandwidth:

The RAM needs to support twice the USB 3.0 bandwidth: one to service the USB side and the other to service the bus side. In the case of a single-port RAM, the recommended clock speed is at least 250 MHz. In the case of 2-Port mode, it needs to be at least 125 MHz. This is typically connected to PIPE3 or MAC3 clock. If your bus clock is faster than the PIPE3/MAC3 clock, you can connect the bus clock to the RAM. For more details, refer to “[Minimum Clock Frequencies: bus_clk, ram_clk](#)” on page [515](#).

- Suspend clock:

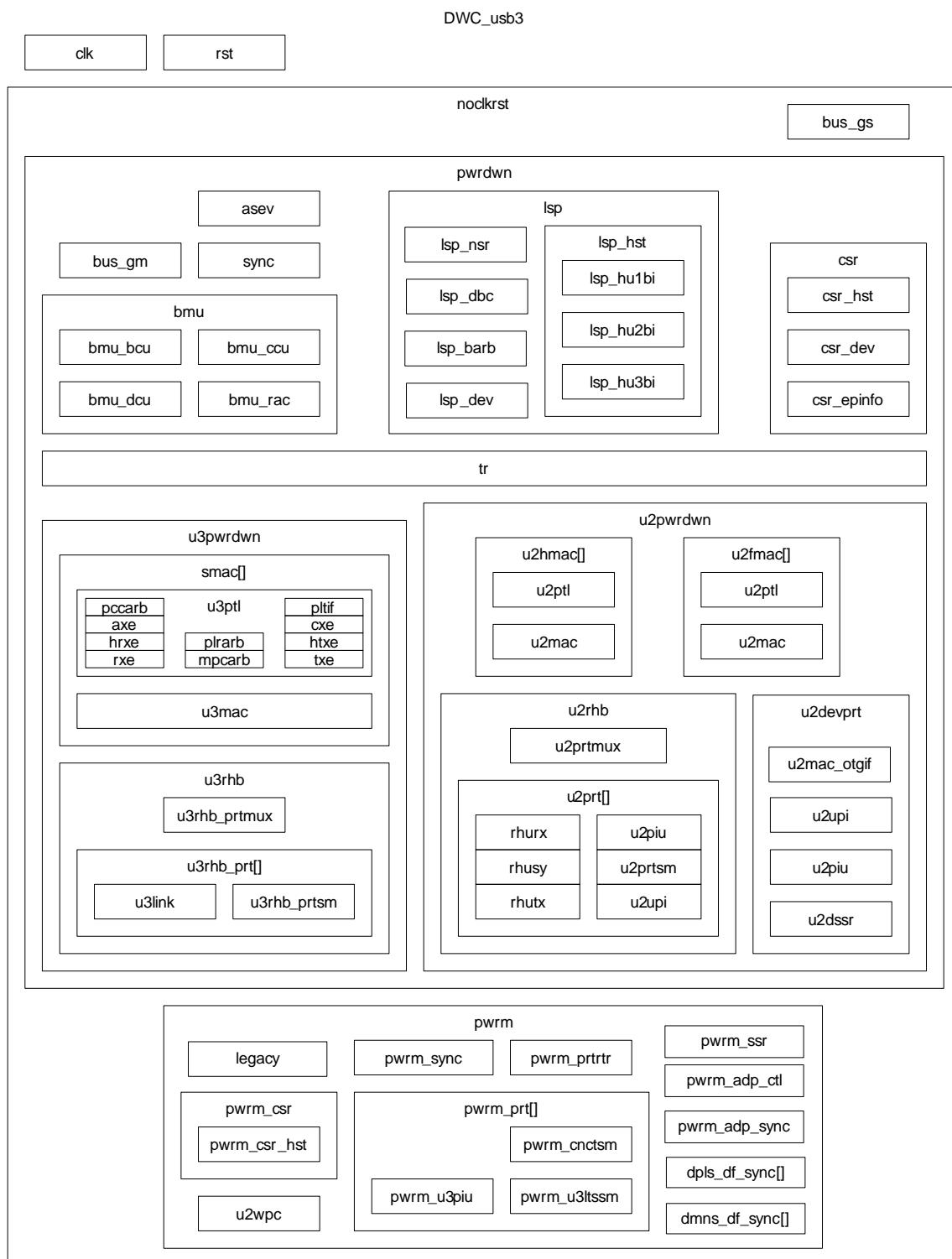
- During suspend, in order to achieve low power, the PIPE3/MAC3/MAC2/MAC clock is switched to a lower frequency variable clock.

The following clock conditions apply during High Speed (High Speed device connection or only High Speed Devices in Host mode)

- The USB 3.0 PIPE3 PHY clock is gated/shut-down.
- The USB 2.0 PHY/MAC clock is 30 or 60 MHz in UTMI+ PHY mode, or 60 MHz in ULPI mode
- The MAC3 clock, which is a divided PIPE3 clock, is gated/shut-down.
- The bus clock is system dependent and should be at least 60 MHz to meet a five-clock USB 2.0 turn-around time.
- The RAM clock, to meet both MAC and BUS domain bandwidth, must support twice the USB 2.0 bandwidth; one to service the USB side and one to service the bus side. Minimum 60 MHz; typically, the bus clock is connected to RAM.

2.2.3 Detailed Hierarchy

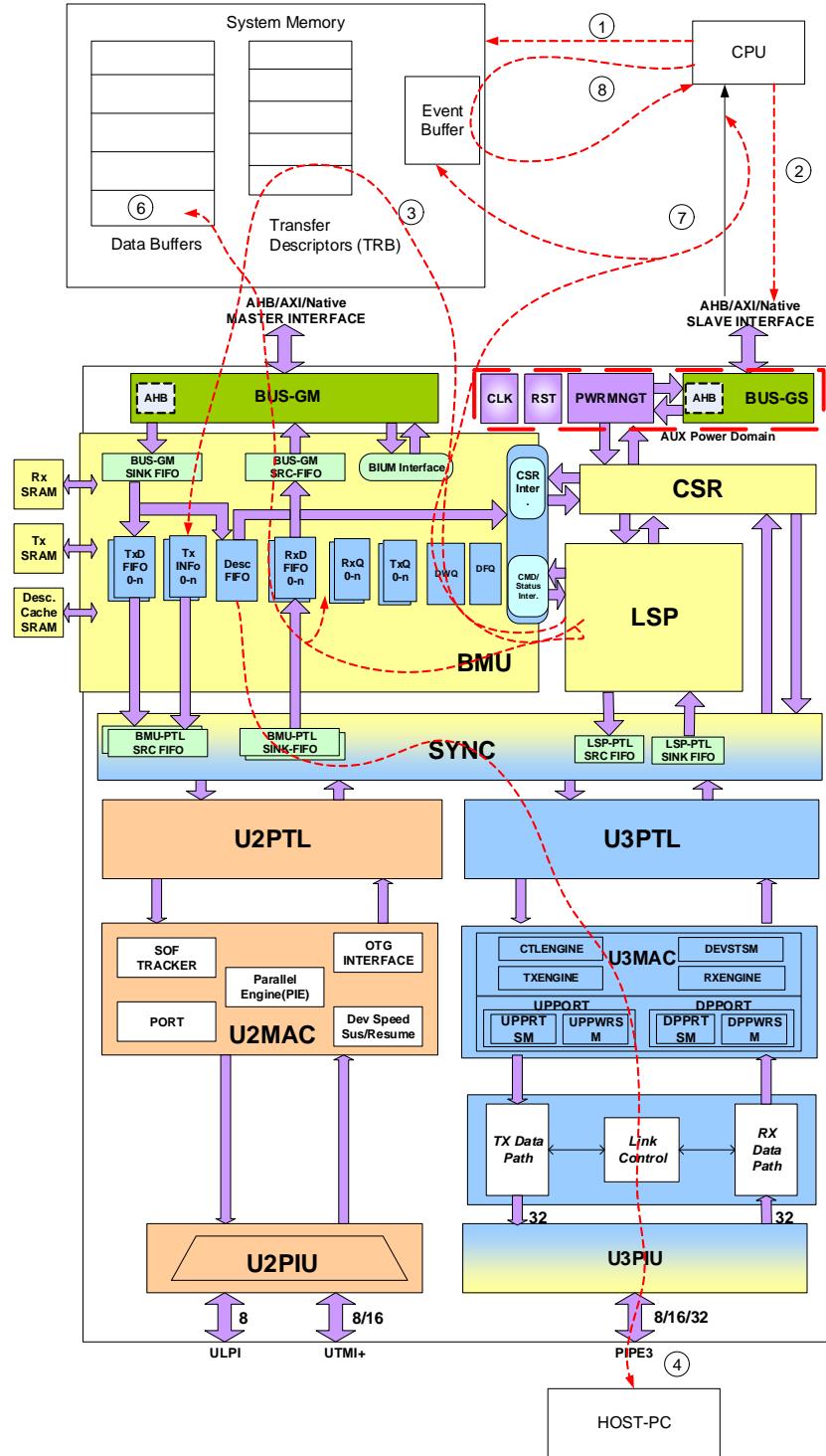
Figure 2-3 on page 88 shows detailed hierarchy of the USB 3.0 controller.

Figure 2-3 DWC_usb3 Detailed Hierarchy

2.2.4 Receiving a Data Packet

Figure 2-4 shows the DWC_usb3 data reception flow, which is described in “Reception Flow Details” on page 90.

Figure 2-4 Data Reception Flow



2.2.5 Reception Flow Details



Note “Stream,” as used in this section, is not the same as defined in the USB 3.0 specification. It refers to a chain of descriptors for one or more transfers to any endpoint type.

The following steps refer to items in [Figure 2-4](#) on page [89](#).

1. Software creates Transfer Descriptors (TRBs) and allocates an event buffer and data buffer in system memory.

For an endpoint, software allocates the data buffers and creates the TRBs that point to the data buffers. In addition, it allocates memory for the event buffer to receive interrupts. It is not necessary for the CPU to allocate TRBs and data buffers at the beginning. It can also do this later, after receiving the “Stream Not Ready” interrupt.

2. Software configures the controller and issues Start Transfer.

Software programs the controller to initialize the endpoint and other registers. Then it programs the Event Buffer pointers and issues a Start Transfer command that indicates where the endpoint’s first TRB is located.

3. The LSP fetches the TRB into the Descriptor Cache RAM.

- a. The LSP, on receiving the Start Transfer command, issues a TRB fetch command to the BMU through the Descriptor Fetch Queue of the BMU.
- b. On receiving the TRBs, the LSP decodes them and keeps them in an internal cache.
- c. Because buffers are allocated for the endpoint, the LSP keeps the endpoint in the “Ready” state.

4. When the host PC sends a packet, it is protocol checked and stored in the RxFIFO.

The host sends a packet to the device.

- U3LINK receives the Data Packet Header and validates CRC5 and CRC16. If passed, it forwards Data Packet Header status to the U3MAC. If there are errors in the Data Packet Header, the header and any associated Data Payload Packet are dropped, and a Link Level response is sent to the link on the host side.
- U3MAC receives the Data Packet Header and presents a summary to the U3PTL.
- U3LINK receives a Data Payload Packet and presents it to the U3MAC along with status. If there are errors, U3MAC prepares ACK with the Retry Data Packet bit set and sends to the U3LINK.
- U3PTL checks the current endpoint, device address and other fields for validity.
- U3PTL checks for the current endpoint state: STALL, NAK, READY (!STALL and !NAK), or Not Active (based on the DALEPENA register).
- If the endpoint is in Not Active state, the transaction times out on the USB.
- If the endpoint is in STALL state, it directs the U3MAC to send a STALL response, and the U3MAC transmits STALL TP to the U3LINK.
- If the endpoint is in NAK state, it directs U3MAC to send NRDY response and U3MAC transmits NRDY TP to U3LINK.
- If the endpoint is ready to accept data, it accepts the Data Packet Payload.
- SS MAC receives the Data Packet Payload.
- U3PTL checks the data sequence.

- U3MAC confirms data to FIFO if U3PTL directs it to receive.
- U3PTL routes the Data to the RxFIFO (in Host mode to port-specific RxFIFO).



Note For bulk endpoints that support stream, starting a stream requires additional PRIME/EPRDY communications.

5. If CRC passes, the PTL informs the LSP of a packet in RxFIFO for an Endpoint. Otherwise, it flushes the packet.
When the packet is received and CRC-checked, the PTL indicates to the LSP that it has received Rx data for an endpoint number with the byte count.
6. Because the LSP has the TRB cache for an EP, the LSP issues an Rx DMA request (which indicates the DMA address and byte count) to the BMU RxQ. The BMU stores (DMA) the packet into the system memory data buffer.
If the status information in the TRBs needs to be updated (for example, transfer count), the LSP also issues a Descriptor Write request through the Descriptor Write Queue (DWQ).
7. When the last packet of a transfer is received (or on TRB IOC), the LSP puts a Stream Complete Event in the BMU's DWQ. The BMU stores it (DMA) into the Event Buffer, and the LSP sets the interrupt.
After all the packets for a transfer are received, the LSP puts the XferComplete event into the BMU DWQ. The BMU then stores (DMA) the event to the event buffer, and the LSP increments the "Event Count" register and sets the interrupt.
The receipt of the last packet is indicated by the last TRB field, receiving a short packet, or Interrupt on Completion (IOC) is set in TRB.
8. Software reads the Event Buffer, which tells it is XferComplete. Software passes data to the application.
On seeing the interrupt, the software reads the "Event Count" number of bytes from the Event Buffer and interprets them. It indicates the XferComplete interrupt. The software passes the data buffers to the application and decrements the "Event Count" register by the number of bytes it has processed. The interrupt line is de-asserted when "Event Count" is zero.

Host-specific details: In the case of the host, between steps 3 and 4, the LSP issues an ACK packet on the USB 3.0 (or Token in USB 2.0) through the PTL/MAY/LINK layers. On seeing this, the device either ends an NRDY or sends a data packet. If the device sends NRDY, the LSP waits until it gets ERDY from the device before sending another ACK again.

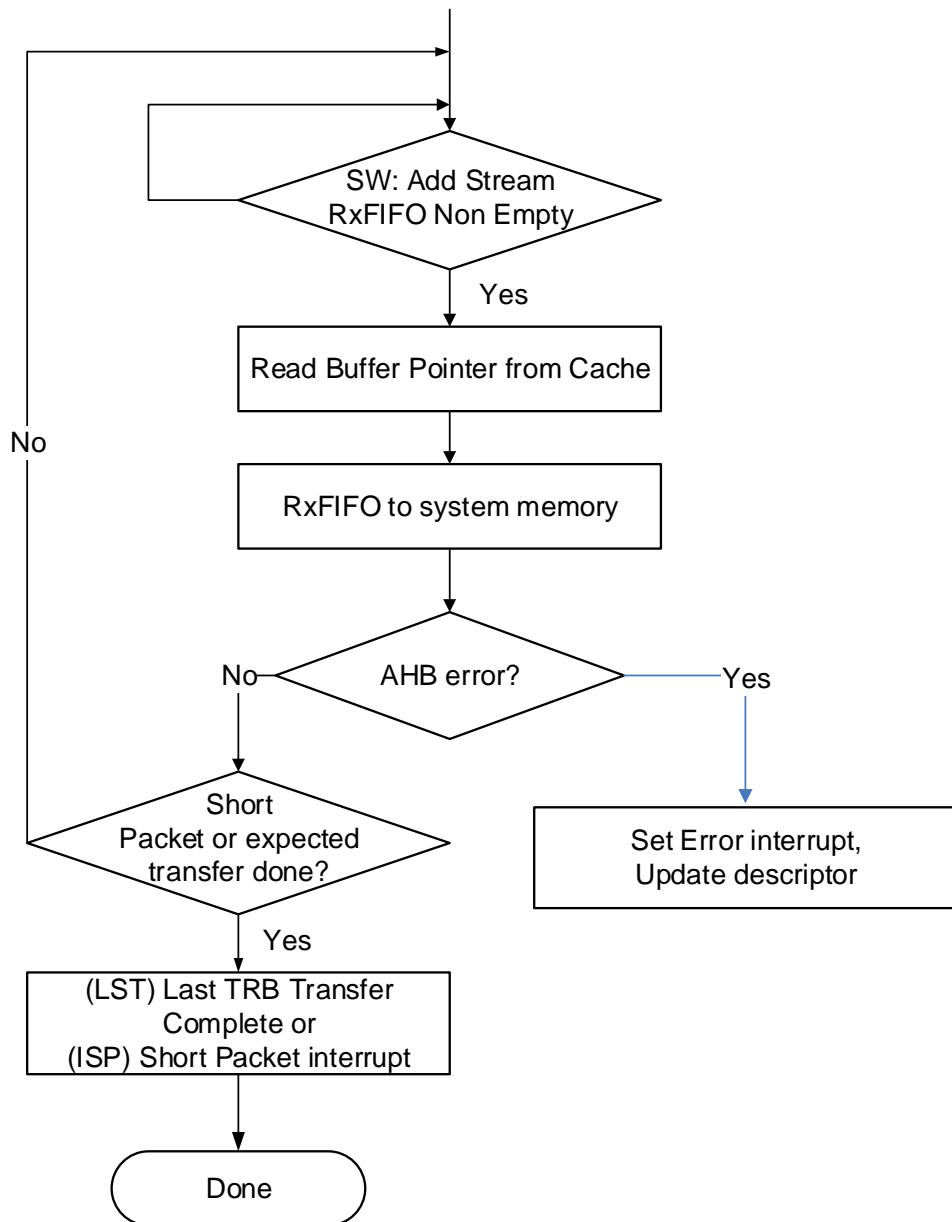
In Host mode, there is a separate Rx FIFO for USB 2.0 port and one each for the concurrent USB 3.0 port. In Host mode, the LSP has more functionality to manage or to schedule traffic to multiple device endpoints.

Device-specific details: The LSP is typically a reactive scheduler. The MAC, LINK, and PIU have port-specific logic not needed in Device mode.

2.2.5.1 Simplified DMA Data Reception Flow

Figure 2-5 shows a simplified DWC_usb3 data reception flow.

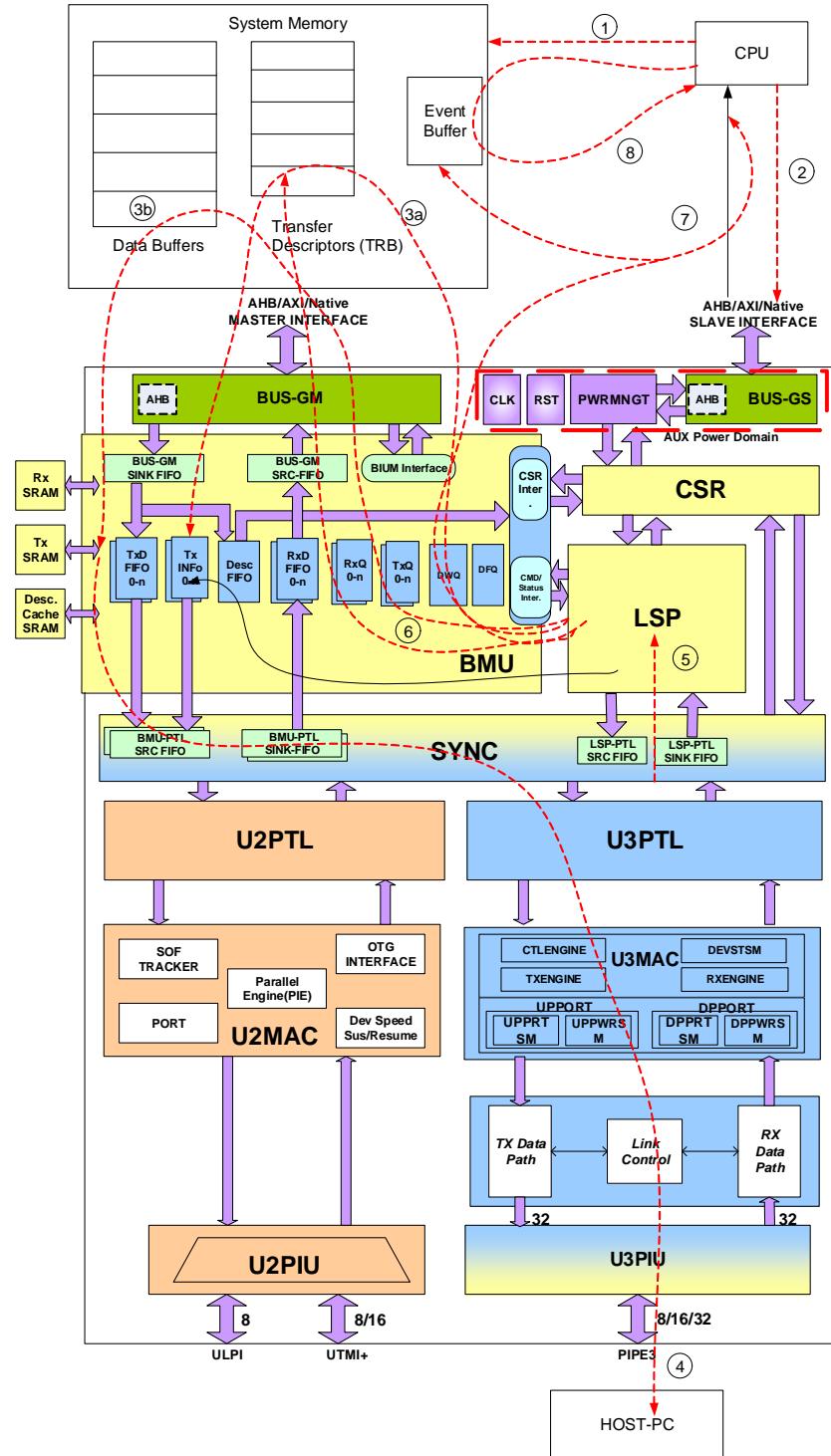
Figure 2-5 Simplified DMA Data Reception Flow



2.2.6 Transmitting a Data Packet

Figure 2-6 shows the DWC_usb3 data transmission flow, as described in “Transmission Flow Details” on page 94.

Figure 2-6 Data Transmission Flow



2.2.7 Transmission Flow Details



"Stream," as used in this section, is not the same as defined in the USB 3.0 specification. It refers to a chain of descriptors for one or more transfers to any endpoint type.

The following steps refer to items in [Figure 2-6](#) on page [93](#).

1. Software creates TRBs, allocates event buffer and data buffers in Event system memory.
For an endpoint, software allocates the data buffers and creates the Transfer Descriptors (TRBs) that point to the data buffers. In addition, it allocates memory for the Event Buffer to receive interrupts. It is not necessary for the CPU to allocate TRBs and Data buffers at the beginning. It can also do this later on receiving the "Stream Not Ready" interrupt.
2. Software configures the controller and issues Start Transfer.
Software programs the controller to initialize the endpoint and other registers. Then it programs the Event Buffer pointers and issues a Start Transfer command that indicate the location of the endpoint's first TRB.
3. The LSP fetches the TRB (through BMU) into the Descriptor Cache RAM.
The LSP, on receiving the Start Transfer command, issues a TRB fetch command to the BMU through the BMU's Descriptor Fetch Queue. On receiving the TRBs, it decodes them and keeps them in an internal cache. Because buffers are allocated for the endpoint, it keeps the endpoint in the "Ready" state. It also issues a Tx Data fetch into the TxQueue for the endpoint so that the BMU can prefetch the data for the endpoint and keep it in the TxFIFO. TxINFO has the packet size and stream-ID (for Stream Bulk only), and microframe number (for isochronous only).
4. The host requests data. The MAC validates the request and the PTL takes data from TxFIFO.

Host-initiated Tx request

- U3LINK receives the ACK packet (IN token) and validates CRC5 and CRC16. If passed, it forwards the ACK packet to the U3MAC. If there are errors in an ACK packet header, it is dropped and a Link Level response is sent to the link on the host side.
- U3MAC receives the ACK transaction packet (IN token) and presents it to the U3PTL with a summary of transaction parameters.
- U3PTL checks the current endpoint, device address, and other fields for validity.
- U3PTL checks for the current endpoint state: STALL, NAK, or READY (!STALL and !NAK)
- If the endpoint is in the STALL state, it directs U3MAC to send a STALL response and the U3MAC transmits a STALL TP to the U3LINK.
- If the endpoint is in the NAK state, it directs the U3MAC to send a NRDY response and the U3MAC transmits a NRDY TP to the U3LINK.
- If the endpoint is ready to send data, the U3MAC sends data (DPH followed by DPP) to the U3LINK.
- U3MAC appends CRC32 and sends it to the U3LINK.
- U3PTL provides the FIFO information for the data.



Note For bulk endpoints that support stream, additional PRIME/EPRDY communications are required to start a stream.

1. On receiving an ACK, the LSP updates transfer count in cache. On retry, it flushes TxFIFO and refetches the packet.

After an ACK for a transmitted packet is received from the host, the LSP updates the transfer count in its TRB cache. While the TxFIFO has space and a transfer is pending, the LSP queues TxPrefetch requests into the BMU's TxQueue. In USB 3.0, if a retry is requested, the LSP flushes the endpoint's TxFIFO and requests the BMU to fetch from where the host is requesting (since USB 3.0 supports burst, the ACK may not come immediately). In USB 2.0, a packet is kept in the TxFIFO until the ACK is received.

2. If TRB needs to be updated, the LSP issues Desc. Writeback request to the BMU. The BMU stores (DMA) the TRB update to system memory.

If status information in the TRBs needs to be updated (transfer count on IOC, for example), the LSP also issues a Descriptor Update Request through the TxQueue.

3. When the last packet's ACK is received (or on TRB IOC), the LSP puts a Stream Complete event in the BMUs Event Queue. The BMU stores it (DMA) to the Event Buffer and raises the interrupt.

After all the packets for a transfer are completed (indicated by the last TRB field or when Interrupt on Completion (IOC) conditions are set in the TRB) and an ACK is received from the host, the LSP puts an XferComplete event into the BMU Event Queue. The BMU stores (DMA) the event into the event buffer, increments the "Event Count" register, and sets the interrupt line.

4. Software reads the Event Buffer, which tells it is XferComplete. Software indicates to the application that transfer has completed.

On seeing the interrupt, software reads "Event Count" number of bytes from the Event Buffer and interprets them. It indicates the XferComplete interrupt. Software indicates to the application that the transfer has finished and it can release the buffers. The LSP decrements the "Event Count" register by the number of bytes it has processed. The interrupt line is de-asserted when "Event Count" is zero.

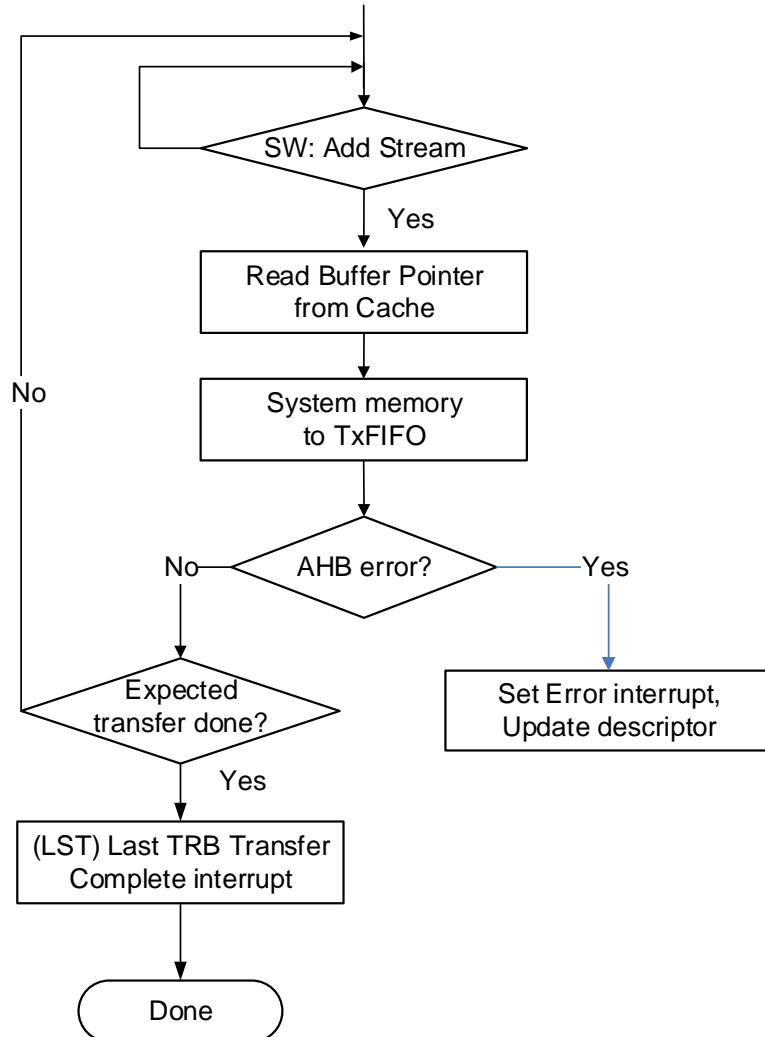
Host Specific Details: In the case of the host, in Step 4, instead of the host request, the LSP issues an OUT request to the PTL/MAC that transfers data to the USB 3.0. If the device accepts the data, the LSP issues the next request to PTL/MAC.

Device Specific Details: If the device retries a packet, the LSP flushes the TxFIFO and fetches the requested data again in USB 3.0 mode. In the case of USB 2.0, a packet is kept in the TxFIFO until an ACK is received.

2.2.7.1 Simplified DMA Transmission Flow

Figure 2-7 shows a simplified DWC_usb3 data transmission flow.

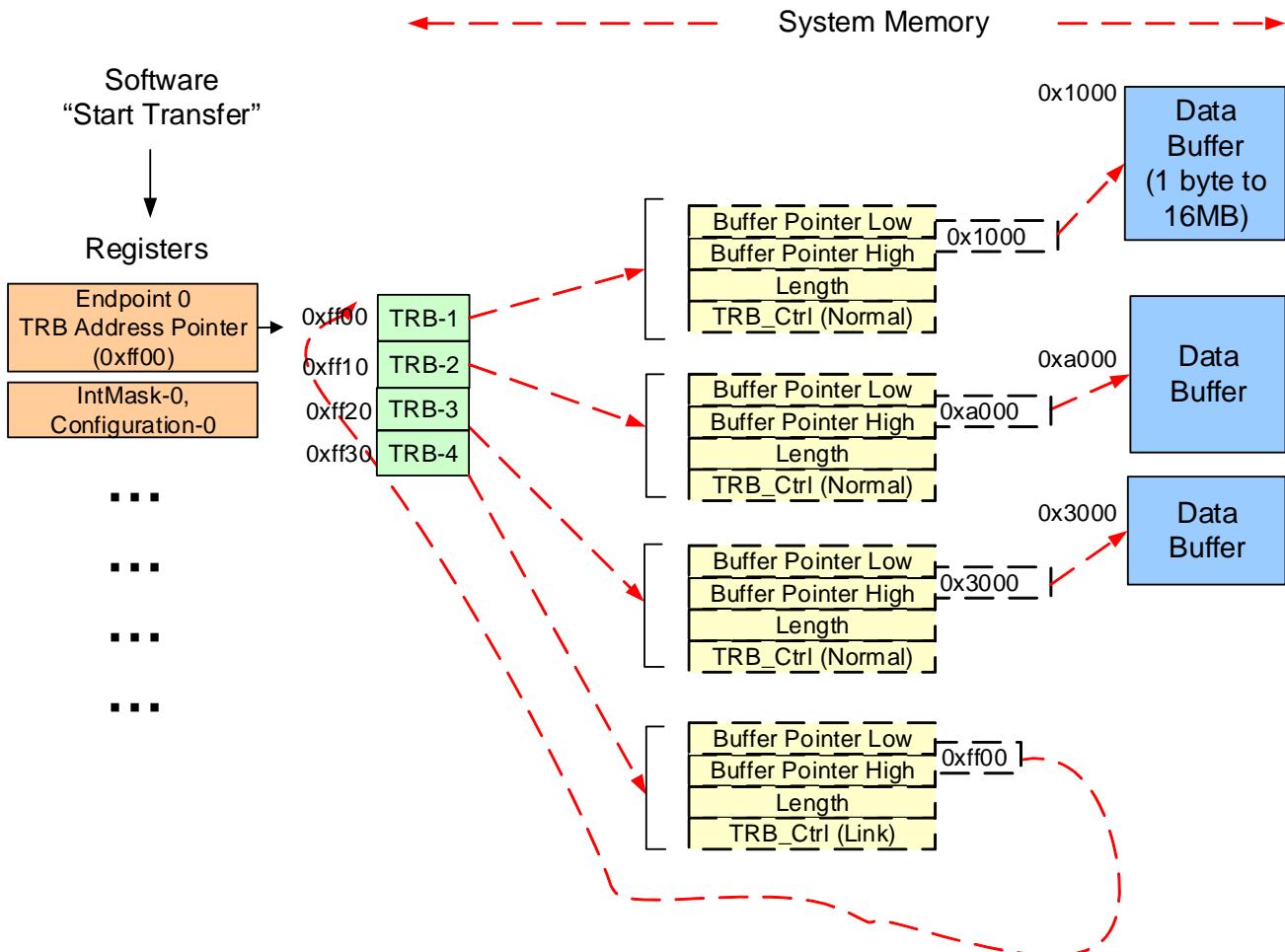
Figure 2-7 Simplified DMA Transmission Flow



2.3 System Memory Descriptor and Data Buffers

Figure 2-8 shows how descriptors and data buffers are allocated in system memory. The software creates TRBs, four DWORDs each, that point to the data buffers. Normally, the TRBs are allocated consecutively in system memory; only the data buffers can be scattered. In the case of a circular buffer, the Link-TRB points to the next TRB. After the TRBs and data buffers are set up in system memory, the software driver issues a Start Transfer command that points to the location of the first TRB in the system memory to start the DMA operation. TRBs, though small (only four DWORDs), provide a rich set of features for the software to schedule transfers, isochronous, control, interrupt moderation, and so on.

Figure 2-8 System Memory Descriptor and Data Buffers



2.4 Configuration-Dependent Flexible Resource Allocation

The DWC_usb3 device controller provides flexible, software-configurable hardware. This allows you to choose a hardware configuration that is optimal for multiple applications. Later, software can configure the controller for each application's requirement. For example, this feature allows you to harden the IP and use it across multiple projects. The software configuration option also gives you flexibility to configure the controller differently than initially planned during the tape-out stage and to avoid some tape-out errors that become obvious only when software drivers are developed.



Noti Software configuration must be done only during initialization or set-configuration.
Hardware must not be re-configured when USB transfers are taking place.

2.4.1 Flexible Endpoint Mapping

Flexible physical endpoint mapping allows a single DWC_usb3 coreConsultant configuration, which is area-optimized for multiple applications/USB set-configuration modes. During coreConsultant configuration, you can configure n number of physical endpoints (register and FIFO resources). Post-silicon, software can map these physical endpoints to USB endpoints, even if the USB endpoints are not contiguous.

In [Figure 2-9](#) on page [99](#), for example, Configuration 1 and Configuration 2 use the following USB endpoints:

Configuration 1	Configuration 2
USB EP0-OUT	USB EP0-OUT
USB EP0-IN	USB EP0-IN
USB EP2-OUT	USB EP4-IN
USB EP2-IN	USB EP15-IN

During coreConsultant configuration, you select four physical endpoints (resources). This implements four sets of endpoint-specific registers.

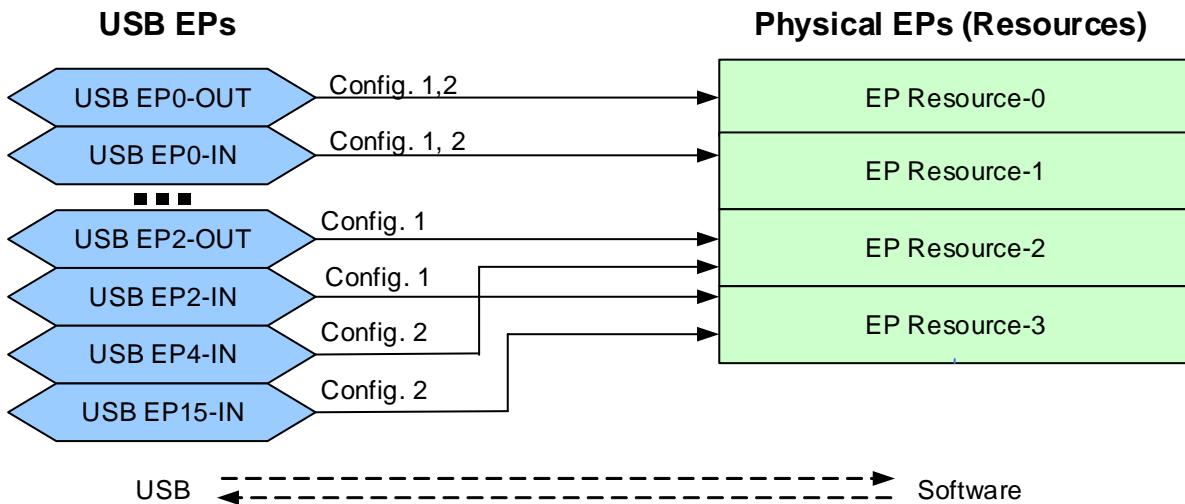
During USB SET_CONFIGURATION-1, the software driver maps:

- Physical endpoint Resource-0 to USB EP0-OUT
- Physical endpoint Resource-1 to USB EP0-IN
- Physical endpoint Resource-2 to USB EP2-OUT
- Physical endpoint Resource-3 to USB EP2-IN

Similarly, during USB SET_CONFIGURATION-2, the software driver maps:

- Physical endpoint Resource-0 to USB EP0-OUT
- Physical endpoint Resource-1 to USB EP0-IN
- Physical endpoint Resource-2 to USB EP4-IN
- Physical endpoint Resource-3 to USB EP15-IN

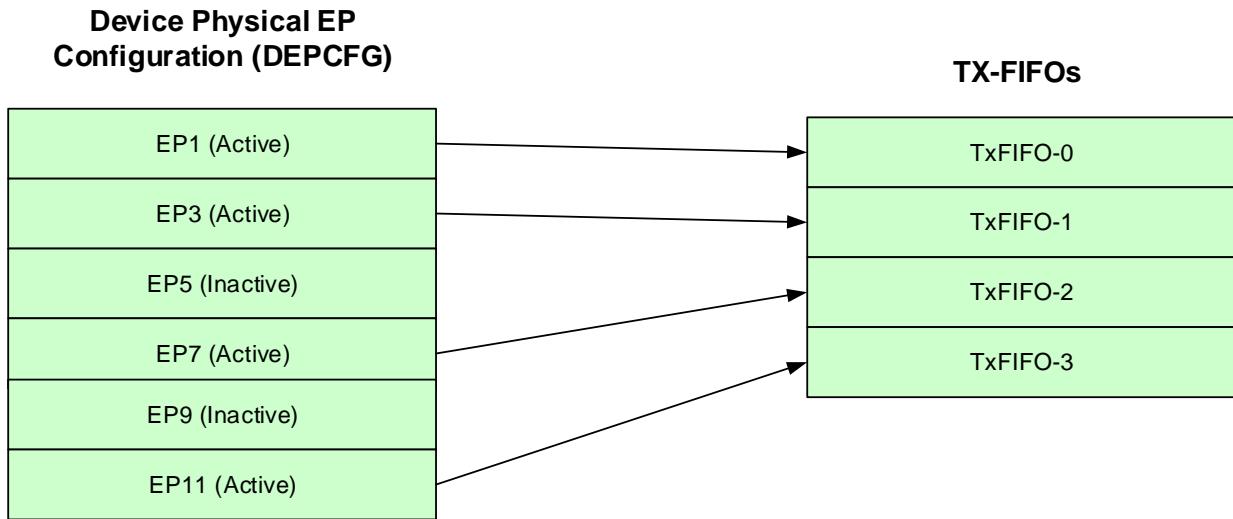
Without this flexibility, the hardware needs 32 Physical endpoints instead of four.

Figure 2-9 Mapping Endpoint Resources with Software

2.4.2 Flexible IN Endpoint TxFIFO Allocation

The USB device design requires each IN endpoint to have its own TxFIFO for data prefetch to meet USB turnaround time. In your application, if you have six IN endpoints, and at a given time only four are active, you can choose four TxFIFOs during coreConsultant configuration and software can map the four TxFIFOs to four currently active endpoints.

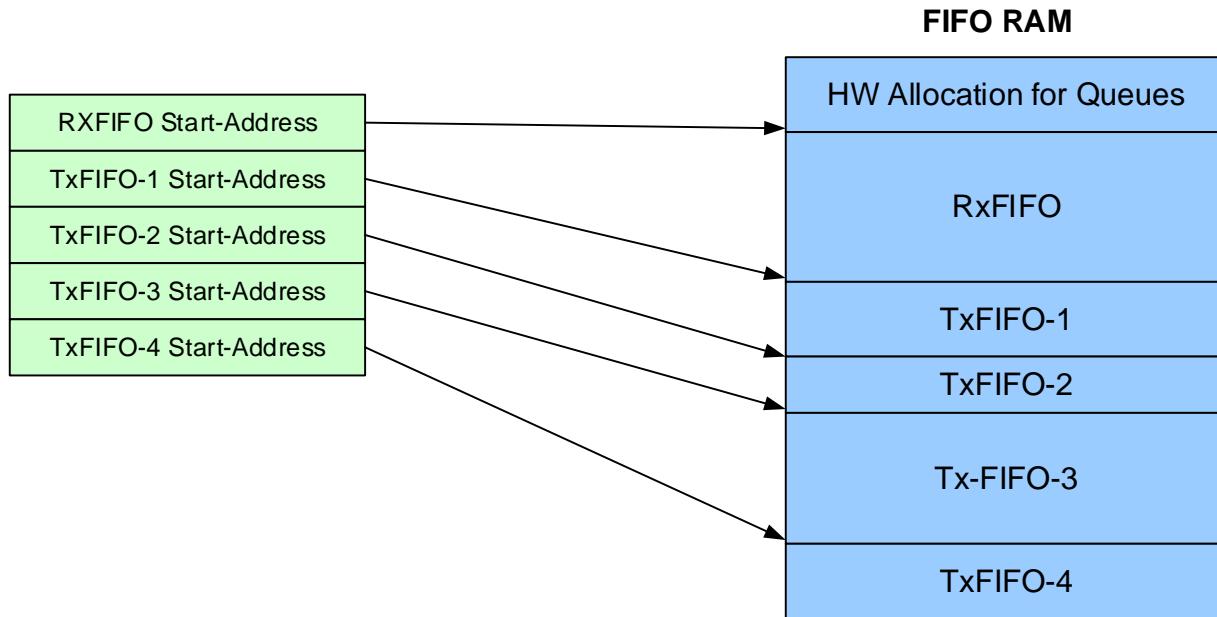
Figure 2-10 shows an example of this software mapping.

Figure 2-10 IN Endpoints and TxFIFOs

2.4.3 Configuration-Dependent RAM Allocation

The software driver can allocate the FIFO RAM of the DWC_usb3 controller to different endpoints, depending on the application configuration requirement. Some portion of the RAM is used for hardware queues, registers, and queues. The remaining memory can be allocated for the RxFIFO and TxFIFOs. In the case of Device mode, the RAM is allocated between different endpoints depending on the endpoints in the device configuration(s), as shown in [Figure 2-11](#). For Host mode, the RAM is allocated between Rx Buffer and Tx-Prefetch buffer.

Figure 2-11 FIFO RAM Allocations



2.4.4 Smaller TxFIFO RAM Requirement

The DWC_usb3 core has separate SoC bus-side Tx fetch Threshold mode which can be used even in Store-and-Forward mode. This allows the TxFIFO RAM size per end point be just 1.25x maximum packet size ($1.25 * 1024$) instead 2x packet size ($2 * 1024$). For more information, see the “Hardware Integration” chapter in the User Guide.

2.4.5 Threshold Mode and Store-and-Forward Mode Support

The DWC_usb3 core supports Threshold mode packet scheduling, where you do not need 1-2 packet size Rx and TxFIFOs. When system memory access is fast and predictable, this mode reduces the overall area of your SoC. If memory access latency is large or unpredictable, then Store-and-Forward (Non-Threshold) mode is recommended.

2.5 Clock Generation and Clock Tree Synthesis (CTS) Requirements

In device, host, or DRD modes, an early offset version of the `bus_clock`, whose edge appears soon enough to match the clock buffering delay, must be provided to the controller so that the clock tree insertion in the controller for `bus_clk` and `bus_clk_gated` can match your SoC clock tree. This avoids unnecessary hold time fixes on SoC bus signals.

2.5.1 ram_clk

In Device, Host, or DRD modes, the `ram_clk` selection is controlled by bits [7:6] in the GCTL register. The RAM clock domain manages the buffering requirements of the controller. In Host mode, it is recommended to select the `bus_clk` as `ram_clk`. However, depending on the bandwidth requirement, you can optionally have the controller dynamically switch the `ram_clk` between `bus_clk`, `pipe_clk`, and `mac2_clk` based on the available 2.0/3.0 clocks. This feature can be enabled by setting the GCTL[7:6] register bits to 3 or 2. For more details, refer to the GCTL register description in the “Register Descriptions” chapter of the *DWC SuperSpeed USB 3.0 Programming Guide*. In Device mode, after Disconnect or USB Reset, the `ram_clk` is automatically attached to the `bus_clk` until a connection is made to the host at either a 2.0 or 3.0 speed. At that time, software may or may not change the `ram_clk` selection based on the following rules:

- When the controller is connected at USB 2.0 speeds (HS/FS/LS), the `ram_clk` cannot be changed and must remain attached to the `bus_clk`, which must be operating at a frequency of at least 60 MHz to meet USB 2.0 timings.
- When the controller is connected at the USB 3.0 speed, the `ram_clk` can be attached to the `bus_clk`, `pipe3_clk`, or `pipe3_clk` divided by 2, but it must be attached to a source that is at least 125 MHz (when using 2Port-RAM) or 250 MHz (when using SPRAM).

2.5.2 mac_clk, mac2_clk, mac3_clk

In Device mode, `mac_clk` is muxed to `mac3_clk` when the device is connected as SuperSpeed, and `mac_clk` is muxed to `mac2_clk` in USB 2.0 mode. The `mac_clk` and `mac3_clk` must have the same clock tree buffer insertion delay, and similarly, `mac_clk` and `mac2_clk` must have the same clock tree buffer insertion delay. There is no clock domain crossing synchronizers between `mac_clk` and `mac3_clk`, and similarly, there is no clock domain crossing synchronizers between `mac_clk` and `mac2_clk`.

In Host mode, the `mac_clk` is assigned to `mac2_clk`. The `mac_clk` and `mac2_clk` must have the same clock tree buffer insertion delay.

In DRD mode, the `mac_clk` and `mac3_clk` must have same clock tree buffer insertion delay, and similarly, `mac_clk` and `mac2_clk` must have same clock tree buffer insertion delay. There is no clock domain crossing synchronizers between `mac_clk` and `mac3_clk`, and similarly, there is no clock domain crossing synchronizers between `mac_clk` and `mac2_clk`.

The clock generation unit for the supported operation modes is illustrated in the following figures:

- Device and DRD modes: [Figure 2-12](#) on page 103
- Host: [Figure 2-13](#) on page 104
- Host Multi-Port SuperSpeed: [Figure 2-14](#) on page 105
- Host USB 2.0: [Figure 2-15](#) on page 106
- Hub SuperSpeed: [Figure 2-16](#) on page 107

Note that in Hub or multi-port Host mode, `mac3_clk` is derived from `pipe3_rx_pcclk` of port-0. The controller requires that both `mac2_clk` and `mac3_clk` be present for proper selection of `mac_clk`.

In Device or single-port Host mode, the `mac3_clk` is treated synchronous to the `pipe3_rx_pclk`. In Hub or multi-port Host mode, the `mac3_clk` is treated synchronous to `pipe3_rx_pclk[0]` and asynchronous to `pipe3_rx_pclk` of other ports.

In Hub mode, the `ram_clk` and `mac3_clk` are the same and must have the same clock tree insertion delay.

In Host and DRD mode, there is an additional clock input `ref_clk`. There is no relationship of this `ref_clk` with other clocks.

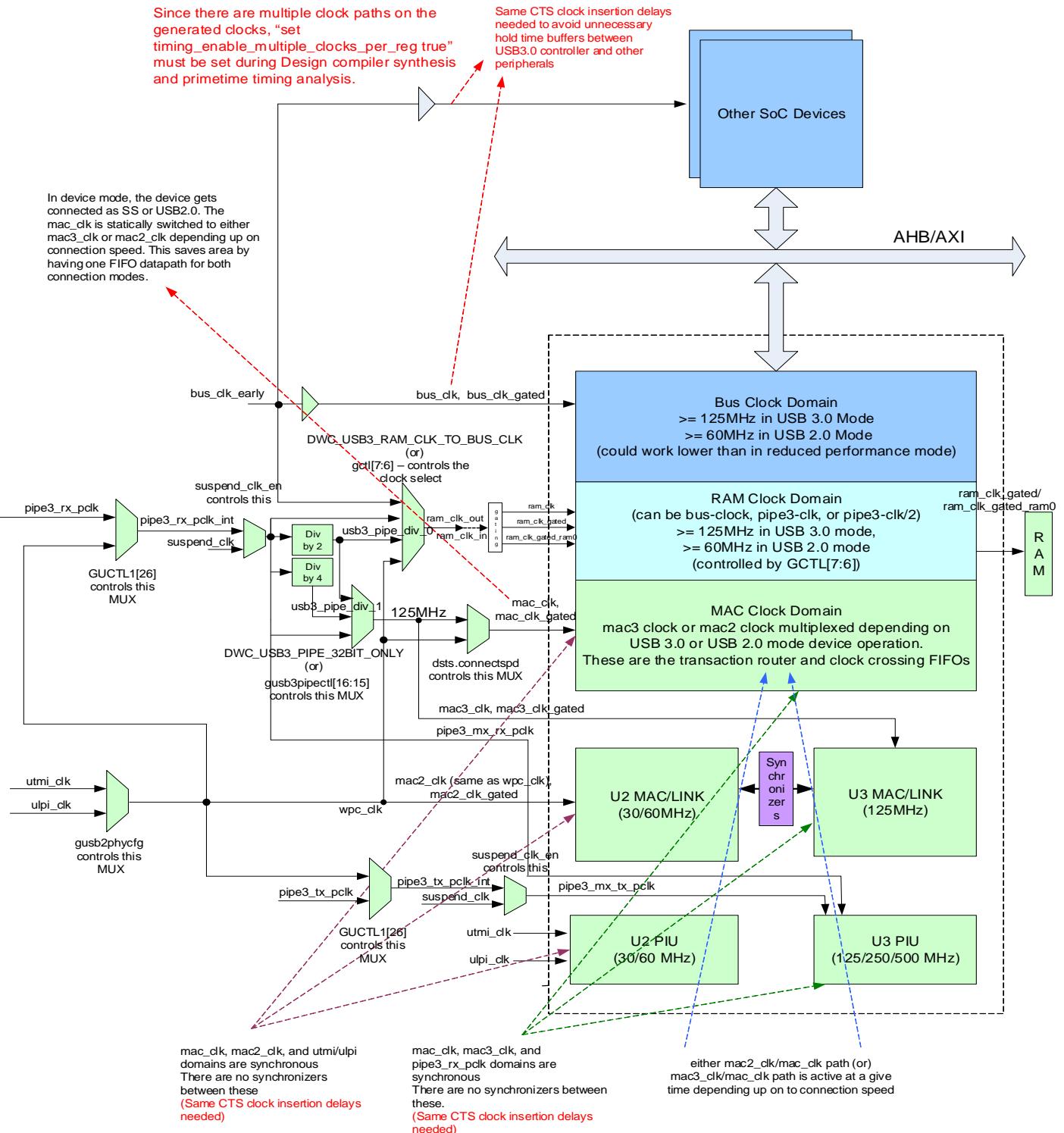
Figure 2-12 Device/DRD Clock Generation and CTS Requirements

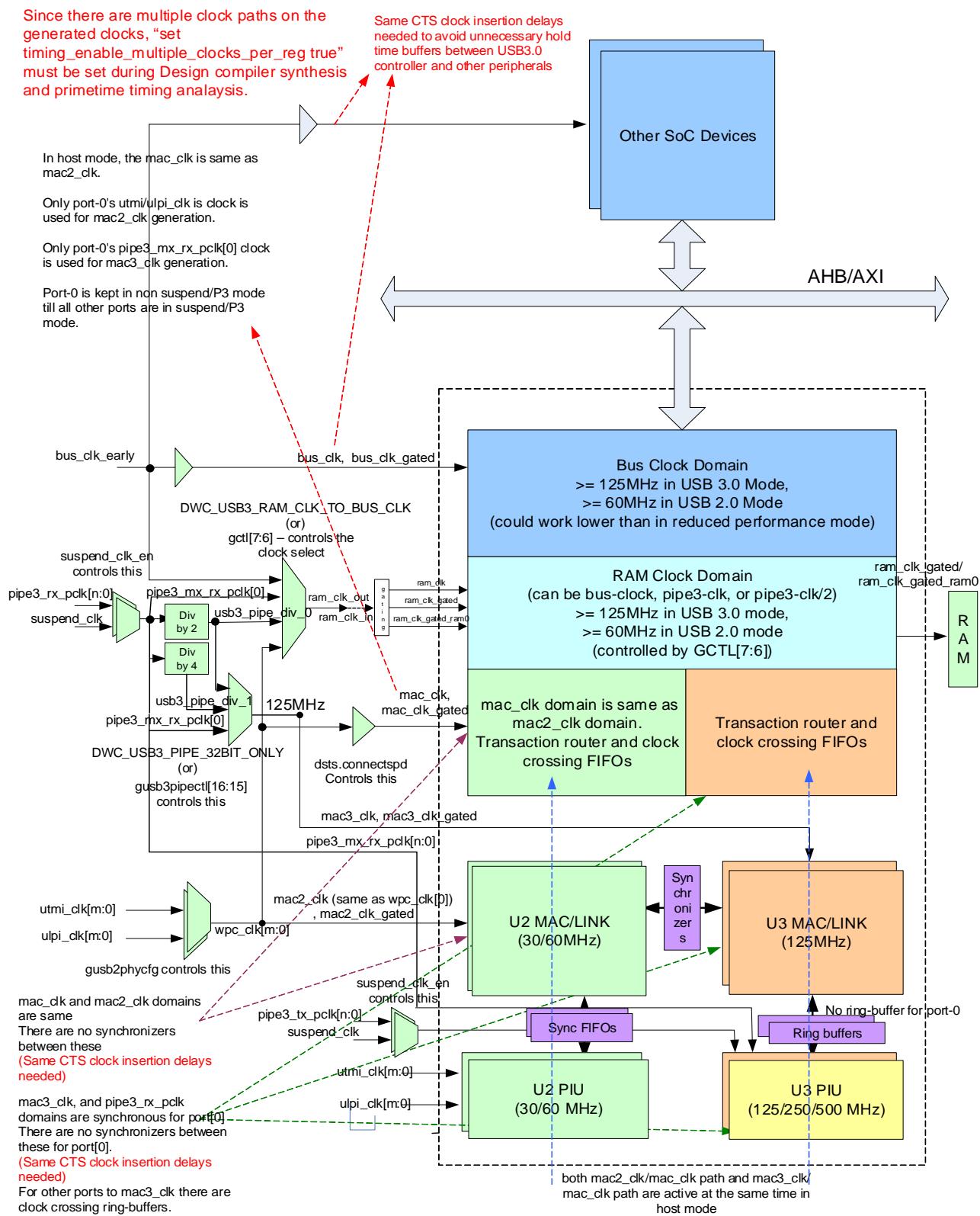
Figure 2-13 Host Clock Generation and CTS Requirements

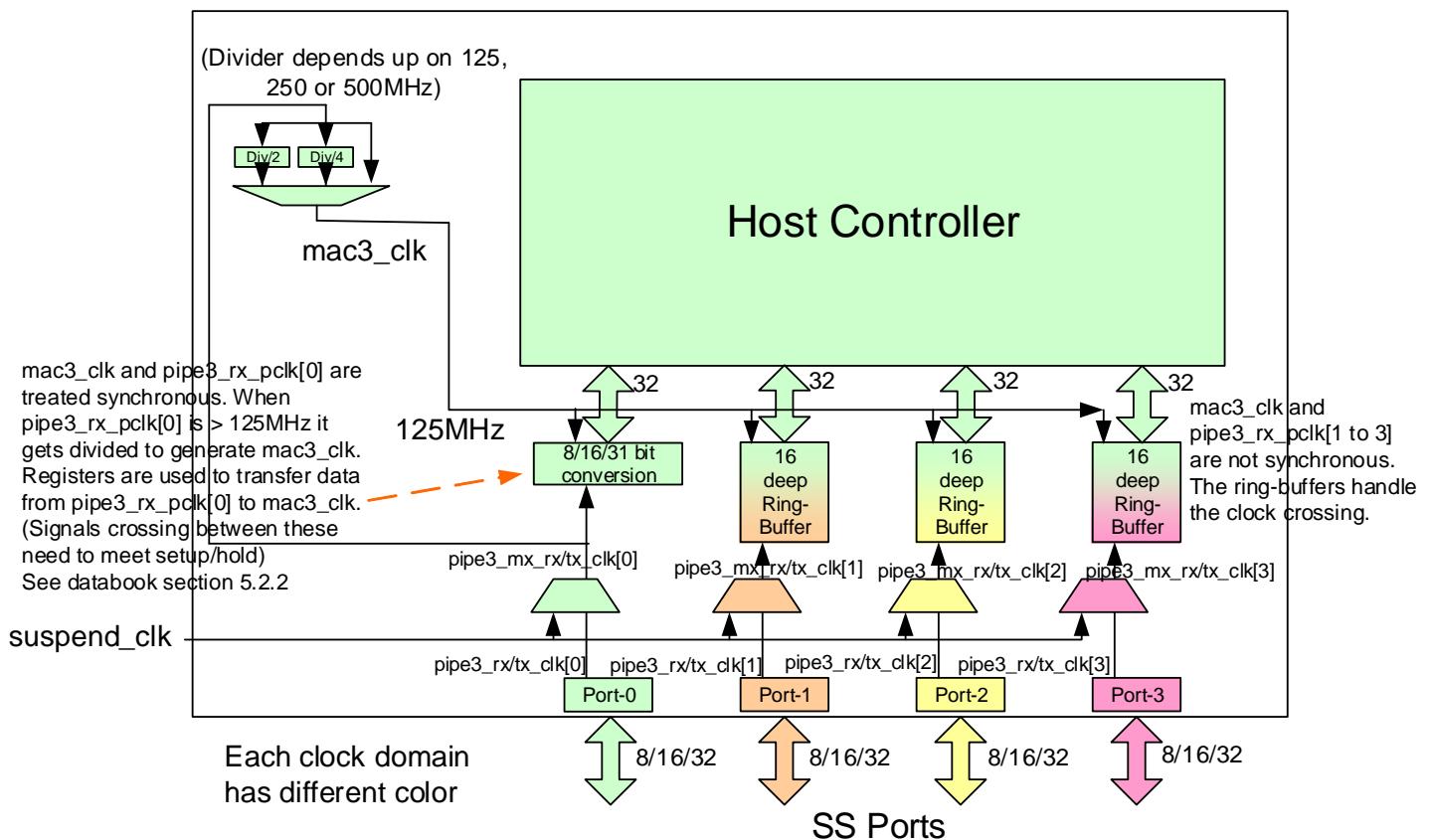
Figure 2-14 Host Multi-Port SS Clock Generation and CTS Requirements

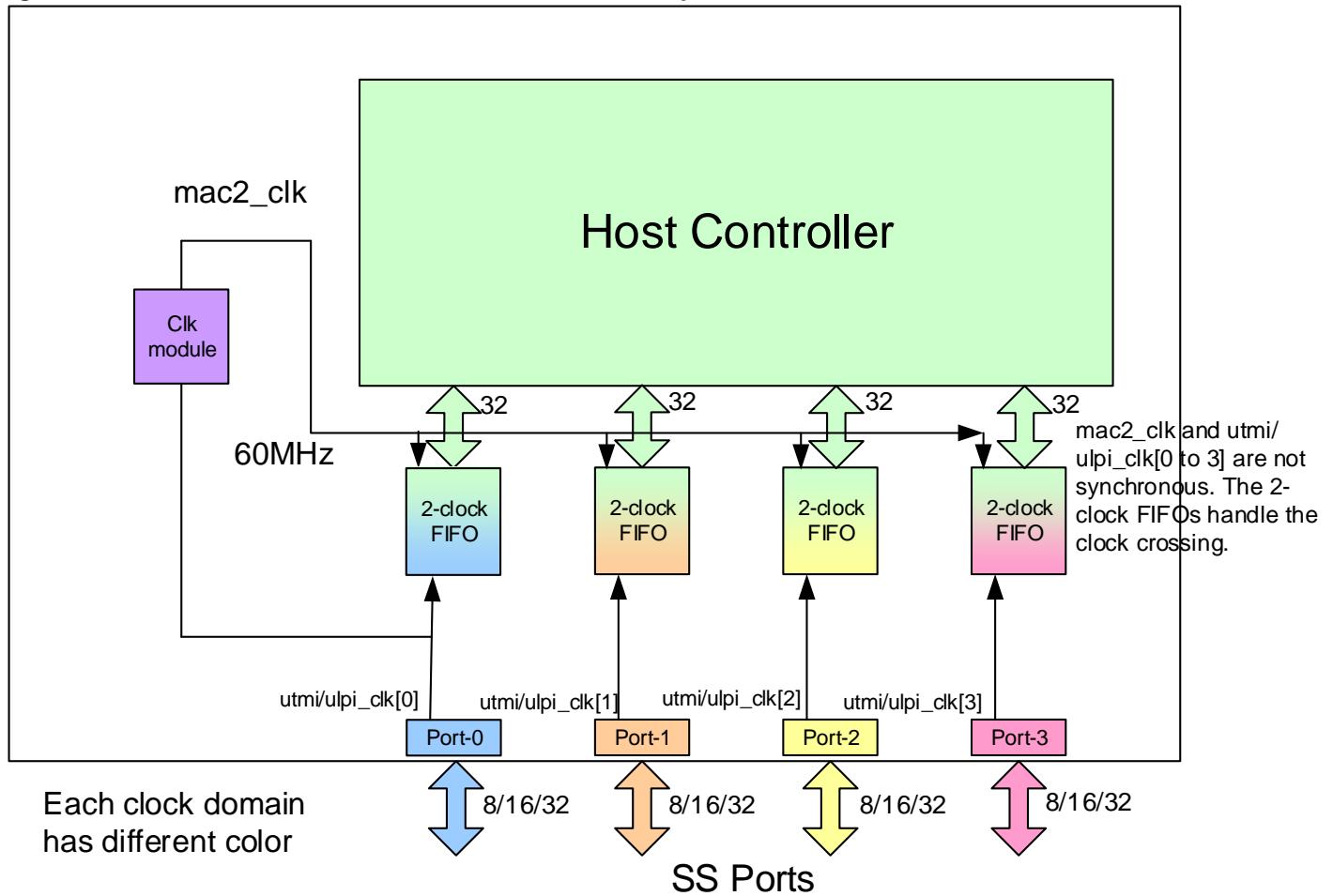
Figure 2-15 Host USB 2.0 Clock Generation and CTS Requirements

Figure 2-16 Hub SS Clock Generation and CTS Requirements

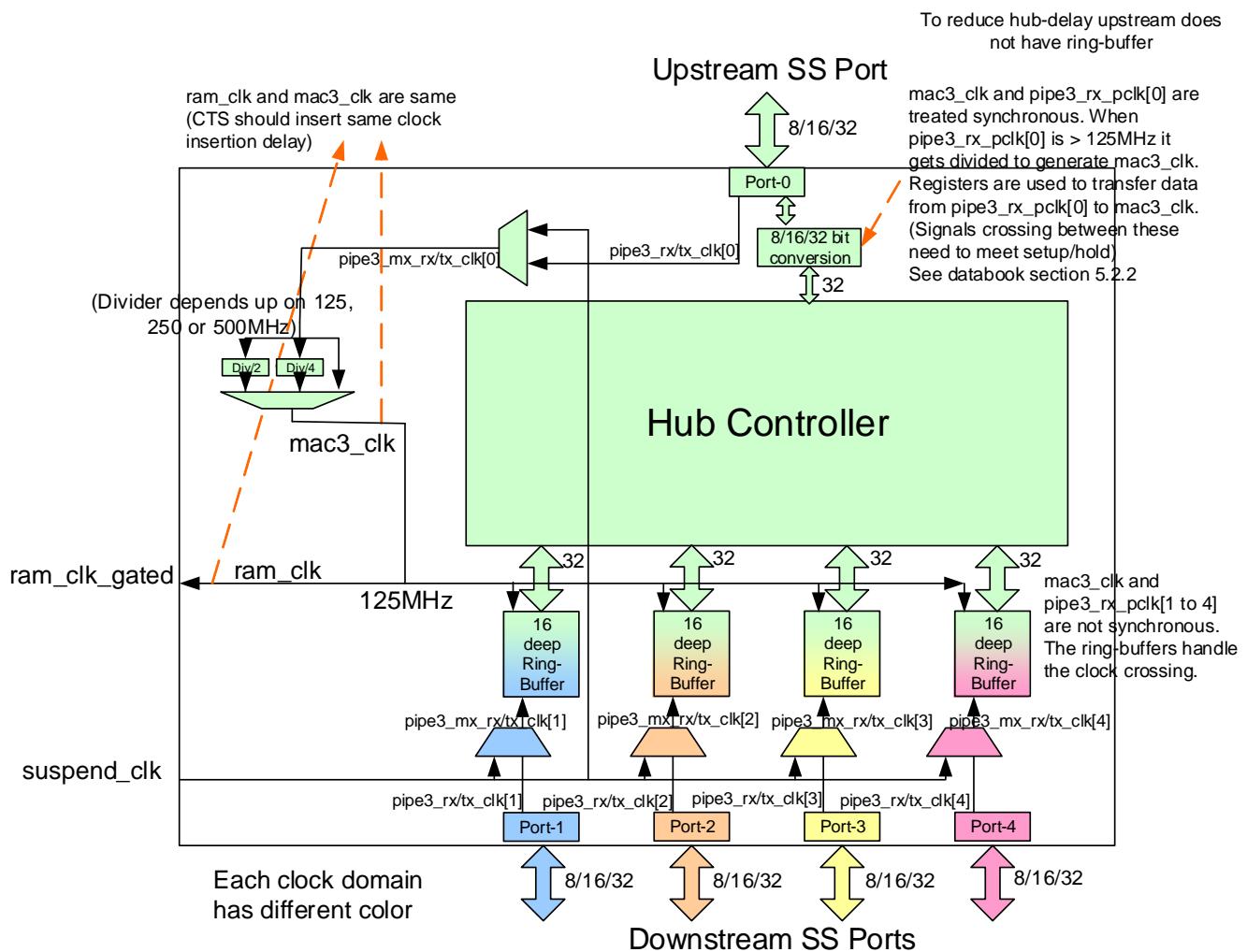
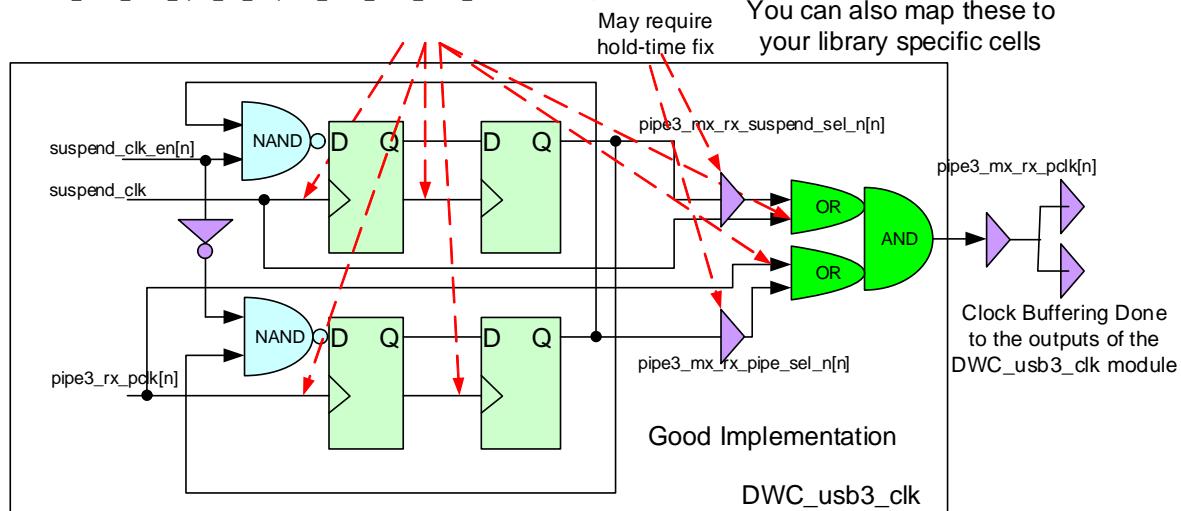


Figure 2-17 Glitch Free Clock MUXing and CTS Requirements

These clock endpoints should not unequal clock buffering/routing delays. In the ICC compiler these flops should be excluded from clock buffering. An example ICC constrain would be:

```
set_clock_tree_exceptions -exclude_pins [U_DWC_usb3_clk/U_DWC_usb3_sync_ctl_pipe3_rx U_DWC_usb3_clk/U_DWC_usb3_sync_ctl_suspend_rx U_DWC_usb3_clk/OR-GATE ...]
```



Example code Segment:

```
pipe3_mx_rx_pclk_int[i] = (pipe3_mx_rx_pipe_sel_n[i] | pipe3_rx_pclk[i]) & (pipe3_mx_rx_suspend_sel_n[i] | suspend_clk);

DWC_usb3_sync_ctl #(1, `DWC_USB3_SYNC_DSYNC_EDGE, `DWC_USB3_SYNC_RAND_TYPE, `DWC_USB3_SYNC_VERIF_EN,
`DWC_USB3_PIPE_CLK_PERIOD, 0, 1)
  U_DWC_usb3_sync_ctl_pipe3_rx(
    .clk (pipe3_rx_pclk[n]),
    .reset_n (pipe3_common_clk_reset_n),
    .clear ('1b0),
    .in_p (~(~suspend_clk_en[n] & pipe3_mx_rx_suspend_sel_n[n])),
    .out_p (pipe3_mx_rx_pipe_sel_n[n]));
  );

DWC_usb3_sync_ctl #(1, `DWC_USB3_SYNC_DSYNC_EDGE, `DWC_USB3_SYNC_RAND_TYPE, `DWC_USB3_SYNC_VERIF_EN,
`DWC_USB3_PIPE_CLK_PERIOD, 1, 1)
  U_DWC_usb3_sync_ctl_suspend_rx(
    .clk (suspend_clk),
    .reset_n (suspend_clk_reset_n),
    .clear ('1b0),
    .in_p (~(~suspend_clk_en[n] & pipe3_mx_rx_pipe_sel_n[n])),
    .out_p (pipe3_mx_rx_suspend_sel_n[n]));
```

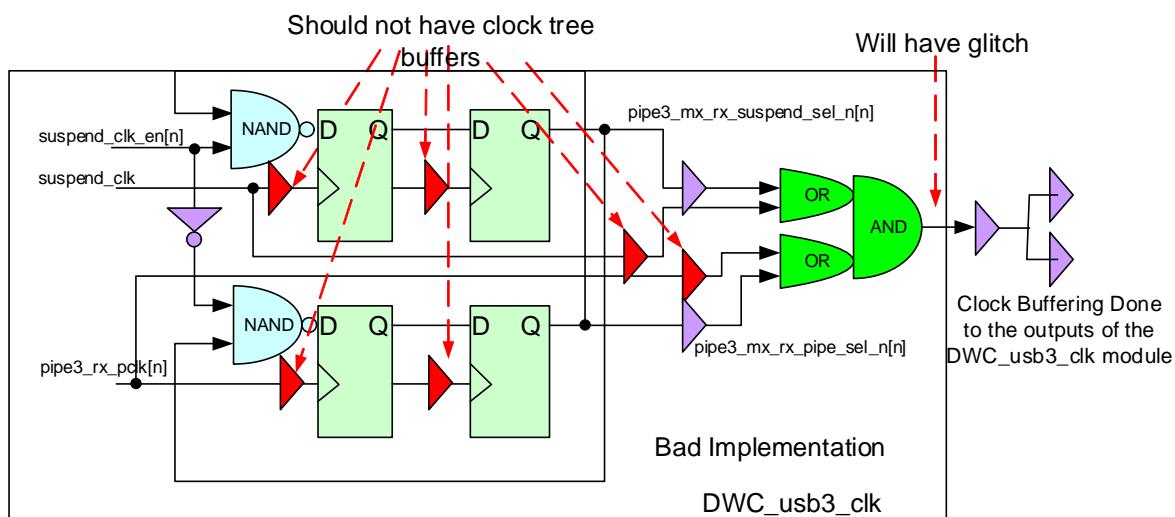
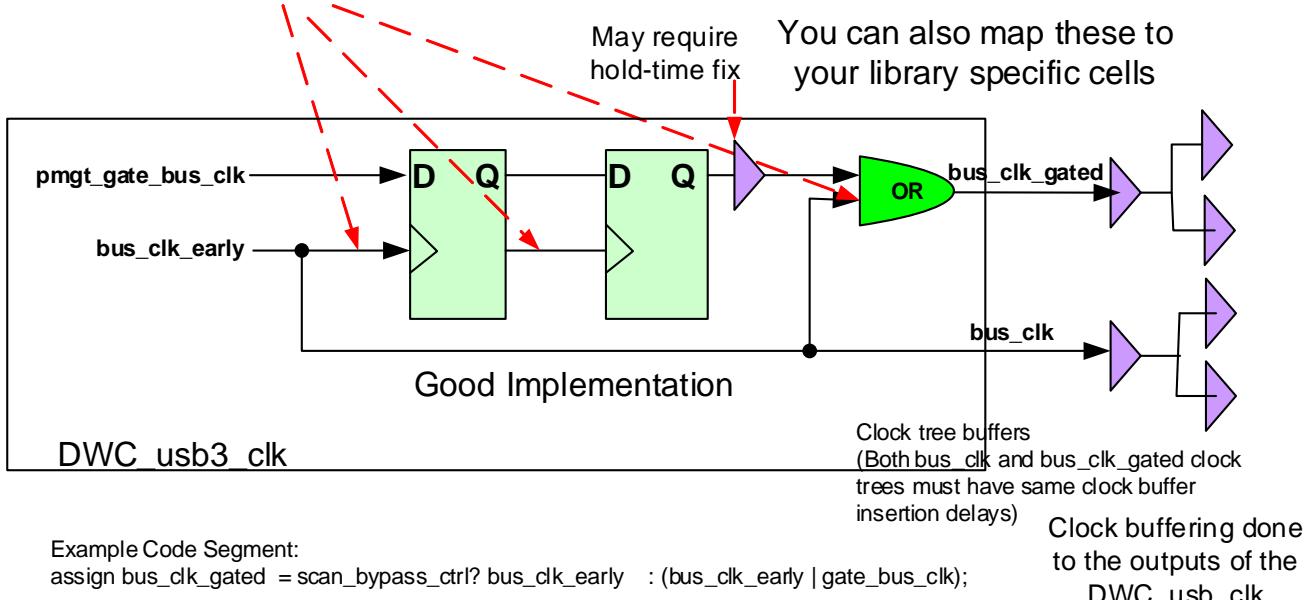


Figure 2-18 Clock Gating and CTS Requirements

These clock endpoints should not have unequal clock buffering/routing delays. In the ICC compiler these flops should be excluded from clock buffering. Else this would result in clock-glitch. An example ICC constrain would be:

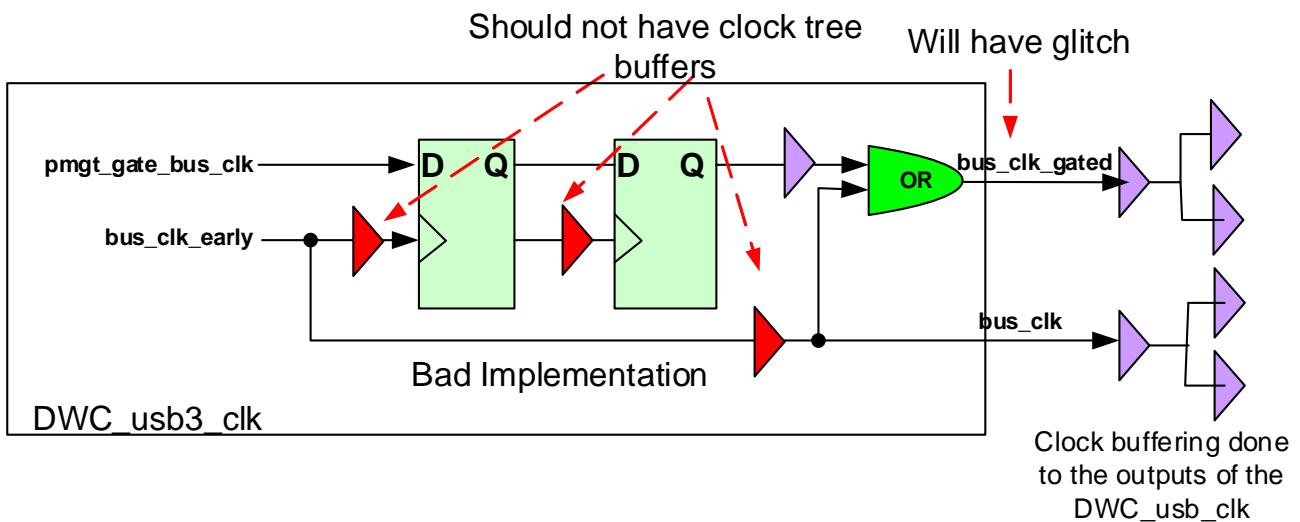
```
set_clock_tree_exceptions -exclude_pins [ U_DWC_usb3_clk/U_DWC_sync_gate_bus U_DWC_usb3_clk/OR-GATE .. ]
```



Example Code Segment:

```
assign bus_clk_gated = scan_bypass_ctrl? bus_clk_early : (bus_clk_early | gate_bus_clk);
```

```
DWC_usb3_sync_ctl #(1, `DWC_USB3_SYNC_DSYNC_EDGE,
`DWC_USB3_SYNC_RAND_TYPE, `DWC_USB3_SYNC_VERIF_EN,
`DWC_USB3_BUS_CLK_PERIOD, 0, 1)
U_DWC_usb3_sync_ctl_gate_bus(
.clk (bus_clk_early),
.reset_n (bus_reset_n),
.clear (1'b0),
.in_p (pmgt_gate_bus_clk),
.out_p (gate_bus_clk));
```



2.6 Reset Generation

The following resets can be controlled by registers (GUSB2PHYCFG, GUSB3PIPECTL, GCTL, and USBCMD), and the signal vcc_reset_n:

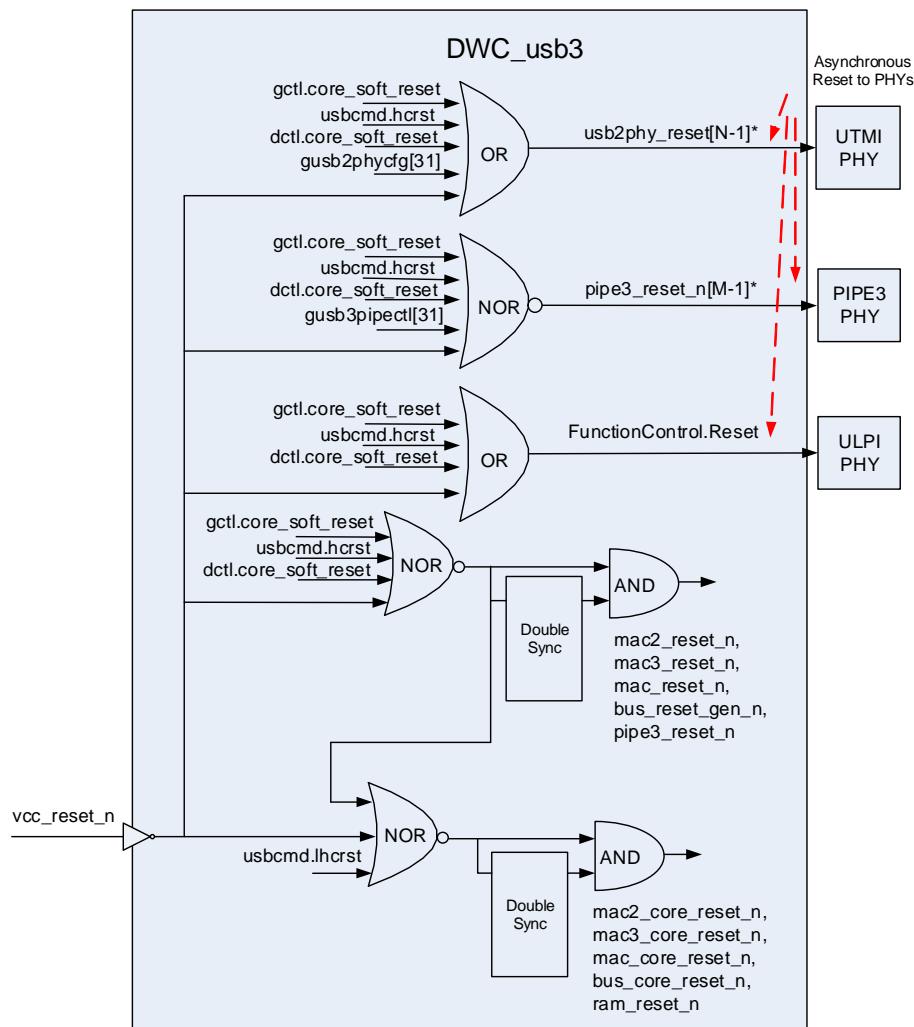
- USB 2.0 PHY reset
- USB 3.0 PHY reset
- Controller internal reset
- Light reset

Figure 2-19 shows the Reset Generation unit.



When Hibernation is enabled, PHY reset behavior is enabled only when both vcc_reset_n and vaux_reset_n are asserted.

Figure 2-19 Reset Generation



Because there are multiple clock domains in the design, and toggle interfaces are used between clock domain, reset generation requires the following conditions to be met:

- All internal resets are active simultaneously for some period to initialize all the flops. The reset-module (DWC_usb3_RST.v) of the controller takes care of this by asynchronously propagating vcc_reset_n to all the internal resets in the design.
- The external USB 3.0 and USB 2.0 PHY reset requirements must be met. The controller provides pipe3_reset_n and usb2phy_reset signals to achieve these requirements.

2.6.1 Asynchronous Resets

In asynchronous reset mode, the controller generates the PHY resets. You must ensure that vcc_reset_n is active to meet the PHY reset timing. Asynchronous reset mode requires the following conditions to be met:

- Connect pipe3_reset_n to USB 3.0 PHY.
- Connect usb2phy_reset to USB 2.0 PHY.
- Assert the vcc_reset_n signal for a minimum period necessary to reset the USB 3.0 and USB 2.0 PHYs.

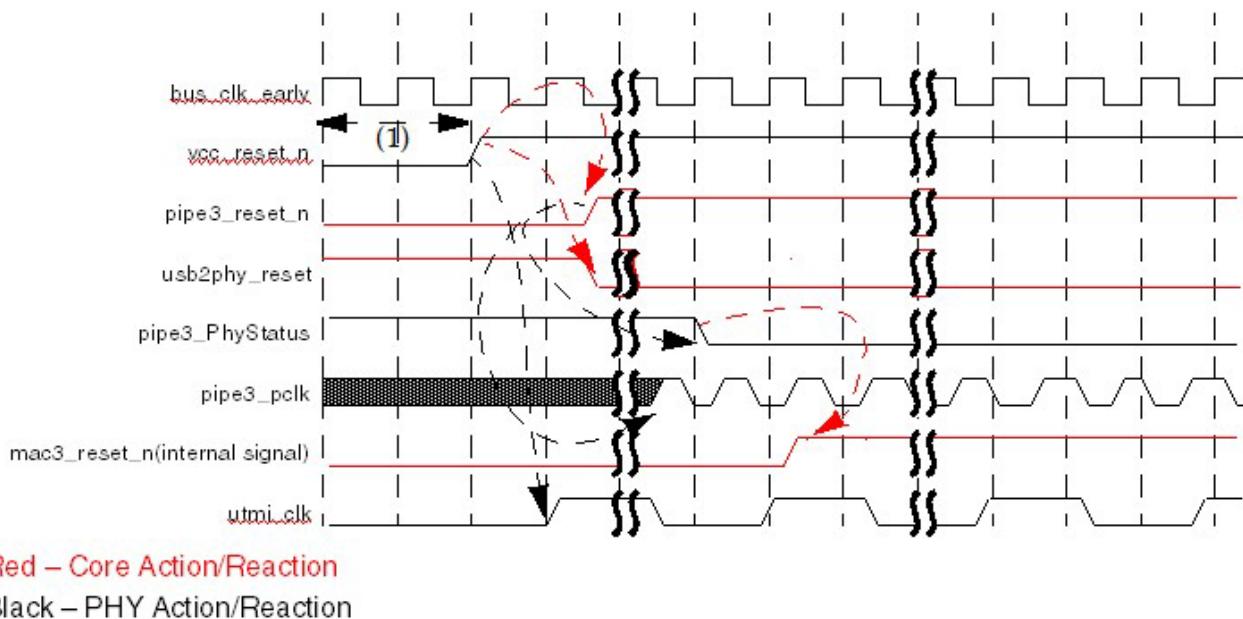
Sequence of Asynchronous Resets

Asynchronous resets must be generated in the following sequence:

1. Drive vcc_reset_n for at least a minimum period required to reset USB 3.0 and USB 2.0 PHYs.
2. The controller drives the pipe3_reset_n and usb2phy_reset outputs active as long as vcc_reset_n input is active. In addition, the controller also drives all the internal resets in the controller.
3. When vcc_reset_n is de-asserted, the controller also de-asserts pipe3_reset_n and usb2phy_reset outputs. In addition, the controller also removes the internal bus and RAM clock domain resets. However, the controller keeps the internal MAC3 clock domain reset.
4. When pipe3_reset_n is de-asserted, the USB 3.0 PHY drives pipe3_Phystatus to "0" once the PHY is reset and the pipe3_pc1k clock is stable. The controller waits until pipe3_Phystatus turns "0" before removing the internal reset to the PHY and MAC domains (mac3_reset_n). This ensures that the controller starts Rx detection only after the PHY clock is stable.
5. In the case of the USB 2.0 PHY, after usb2phy_reset is de-asserted, the USB 2.0 PHY drives a stable valid clock based on the PHY specification.

In addition to power-on reset, you can use the software reset feature discussed in “[Software Resets](#)” on page [112](#) section at any time.

[Figure 2-20](#) on page [112](#) shows the asynchronous reset and PHY clock sequencing and requirements.

Figure 2-20 DWC_usb3 Clock Domains and Data Flow

2.6.2 Software Resets

Software Reset is an optional feature that allows the software to reset the controller and the PHY whenever it is required. For example, while developing drivers you can use this feature to reset the controller and PHY before you load modules into the driver. This ensures that you do not have to do a hardware reset of your development platform if the driver stops functioning, thereby saving some time.

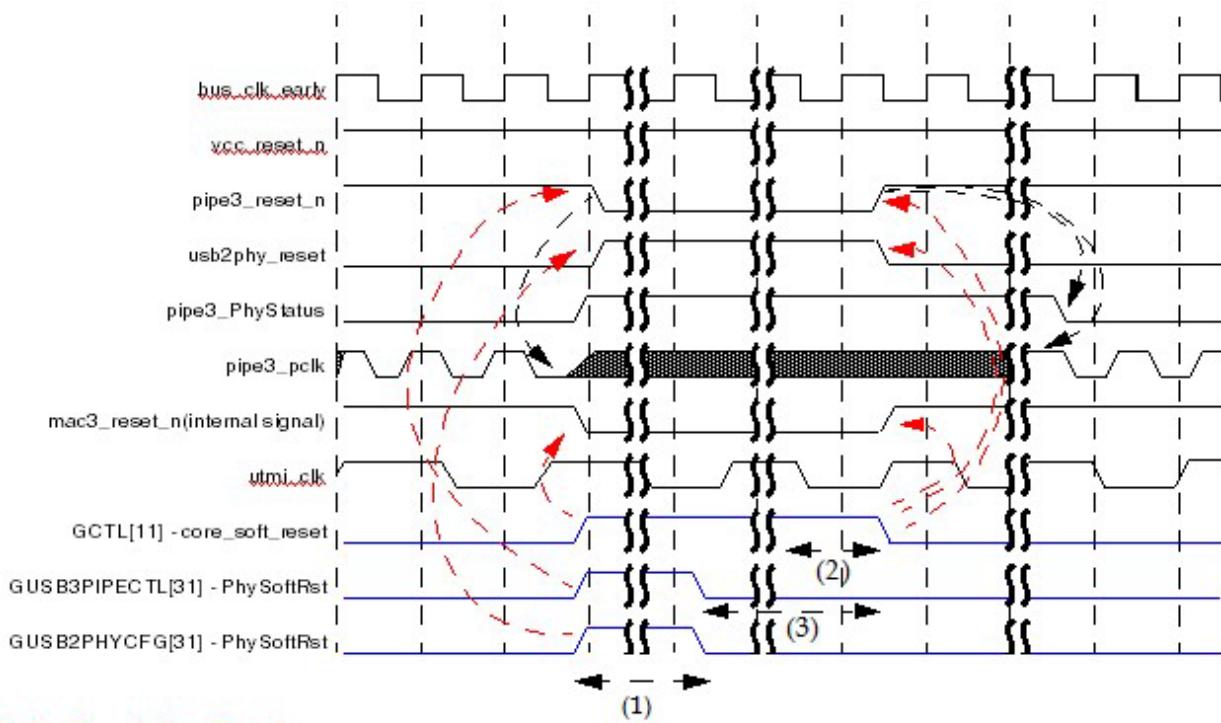


The PHYs can be reset only if you are using the `pipe3_reset_n` and `usb2phy_reset` signals from the controller.

The GUSB2PHYCFG, GUSB3PIPECTL, and GCTL registers control the USB 2.0 PHY reset, USB 3.0 PHY reset, and internal reset of the controller. Software resets must be generated in the following sequence:

1. Set GUSB2PHYCFG[31], GUSB3PIPECTL[31], and GCTL[11] to reset the PHYs and keep the controller in a reset state.
2. Reset GUSB2PHYCFG[31] and GUSB3PIPECTL[31] after they meet PHY reset duration. This removes the reset to the PHYs.
3. Wait for the PHY clock to stabilize and reset GCTL[11] bit. This ensures that the reset to all the internal blocks are asserted when all the clocks are stable.

Figure 2-21 on page 113 shows the software reset and PHY clock sequencing and requirements.

Figure 2-21 Software Resets and PHY Clock Sequencing and Requirements

- (1) – PhySoftRst assertion timing must meet the minimum USB 3.0 and USB 2.0 PHY reset timing.
- (2) – The core_soft_reset signal must be de-asserted only after at least 16 clock cycles of the slowest clock; if the suspend_clk is the slowest clock, then use the suspend_clk.
- (3) – The core_soft_reset de-assertion after the PhySoftRst de-assertion must be at least after 16 clocks of the slowest clock; if the suspend_clk is the slowest clock, then use the suspend_clk.

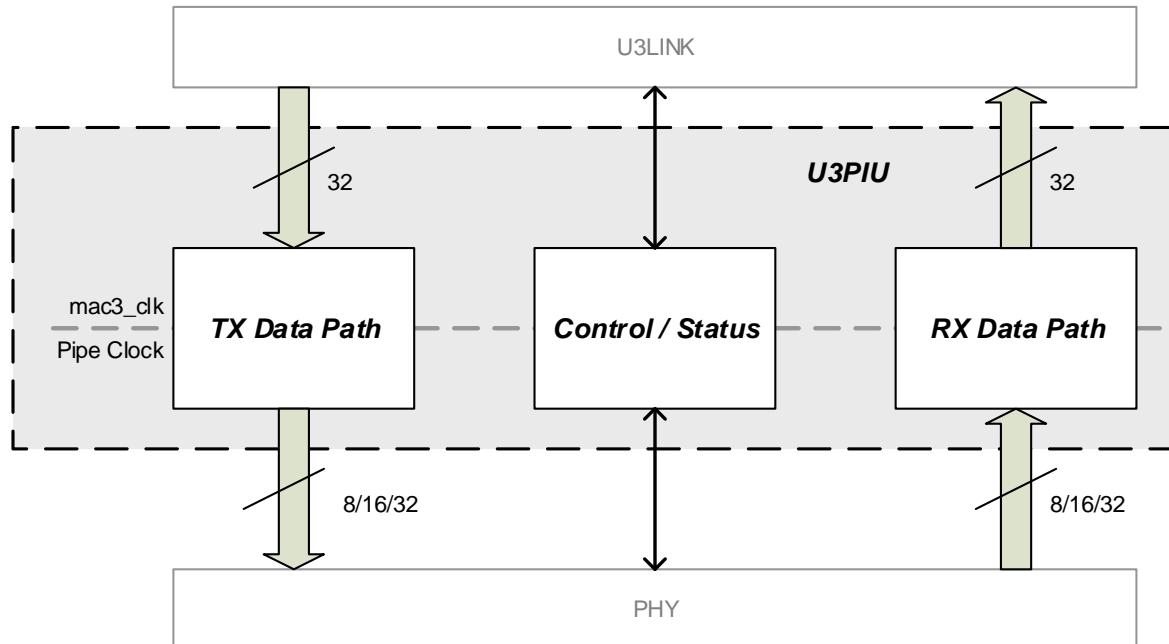
If you change PHY selection or PHY configuration (register-based PHY parameters, such as PHY Data width, ULPI or UTMI, and so on), then you must follow the soft-reset sequence for correct operation of your controller.

2.7 USB 3.0 PHY Interface Unit (U3PIU)

The USB 3.0 PHY Interface Block (U3PIU) provides an interface between the PHY `pipe3` (PIPE) clock domain and the U3LINK and U3LTSSM `mac3_clk` domain, as shown in [Figure 2-2](#) on page [83](#).

[Figure 2-22](#) shows an overview of the U3PIU.

Figure 2-22 U3PIU Overview



The U3PIU block performs the following functions:

- Transfers received and transmitted data symbols, control, and status information between the `pipe3` clock and `mac3_clk` domains.

The U3PIU block supports 8-bit/500-MHz, 16-bit/250-MHz, and 32-bit/125-MHz PHY data interfaces. In Device mode, all transfers are synchronous. In Hub and multi-port Host mode ring buffers are used for data transfers, and multi-bit synchronizers are used for PHY command and status signals.

- Performs scrambling, compliance pattern, and training set generation.
- Drives the PHY with PIPE3 specification required control signal values while PHY reset is asserted.
- Includes input and output registers for all PIPE3 signals for easier timing closure with off-chip PHYs.

The `mac3_clk` side of the U3PIU always operates at 125 MHz and is four symbols wide. To simplify integration, the PHY side of the U3PIU allows the `pipe3` clock to be skewed separately for data and control sent to the PHY (Tx), and data and status received from the PHY (Rx).

The U3PIU design functions well for a wide range of on-chip and off-chip PHY configurations, and accommodates a range of `mac3_clk` tree delays relative to `pc1k`, as well as asynchronous PIPE3 and MAC3 clocks in multi-port devices.

2.7.1 U3 PIU Clocks

The U3PIU has the following clock inputs:

- mac3_clk
- pipe3_rx_pclk
- pipe3_tx_pclk

The pipe3_rx_pclk and pipe3_tx_pclk inputs are normally driven directly from the PHY. The two pipe3 clocks can have their own individual clock trees with delay optimized for PHY integration. The pipe3_rx_pclk and pipe3_tx_pclk clocks may be skewed with respect to each other up to one half of the pclk cycle time.

Refer to “[SS Tx Synchronous Data Transfer](#)” and “[SS Rx Synchronous Data Transfer](#)” on page 117 for clock skew restrictions between the three U3PIU input clocks. In addition to the noted restrictions, the maximum skew between pipe3_rx_pclk and pipe3_tx_pclk is (pclk period /2).

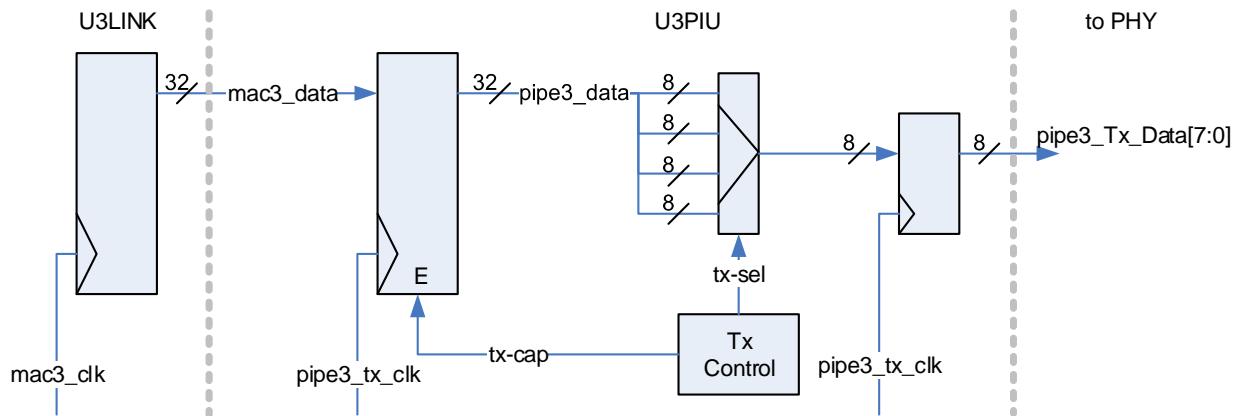
2.7.2 SS Tx Synchronous Data Transfer

In this mode, data from the U3LINK module is transferred synchronously from the mac3_clk to pipe3_tx_pclk. After the transfer, the 32 bits of data are sent through the data output MUX and data output registers as 8, 16, or 32 bits of data to the PHY. One K-bit is also transferred for each 8-bits of data.

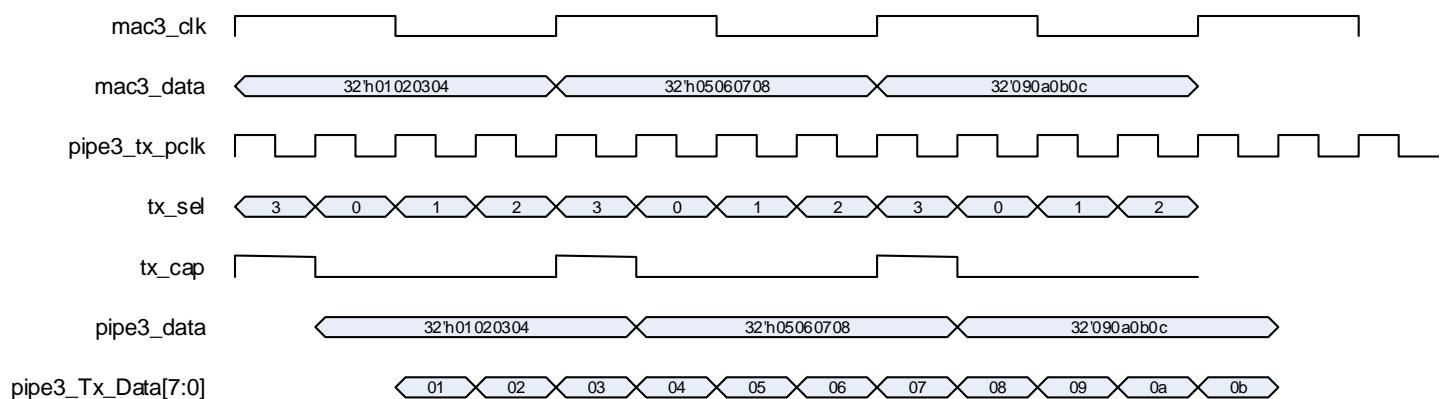
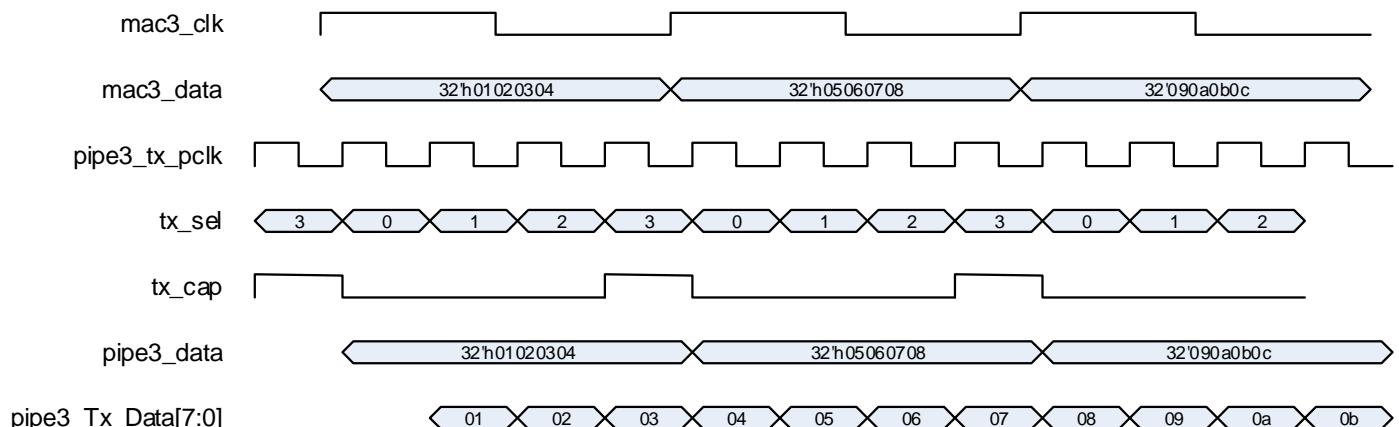
Device, upstream hub ports, and the host port 0 are configured for a synchronous data transfer.

[Figure 2-23](#) shows the data transfer logic for an 8-bit PHY.

Figure 2-23 SS Tx Data Transfer Logic (8 bit PHY)



For an 8-bit PHY, the pipe3_tx_pclk rising edge may occur at the same time or up to about 1.5 ns earlier than the mac3_clk rising edge. See [Figure 2-24](#) on page 116 and [Figure 2-25](#) on page 116.

Figure 2-24 SS Tx Data Transfer Timing (8-bit PHY; no mac3_clk delay)**Figure 2-25 SS Tx Data Transfer Timing (8-bit PHY; 1.5 ns mac3_clk delay)**

For a 16-bit PHY, the pipe3_tx_pclk rising edge may occur at the same time or up to about 3.5 ns earlier than the mac3_clk rising edge. For a 32-bit PHY, there is no capture register, because the two clocks are identical (except for the clock tree delay). In this scenario, the pipe3_tx_pclk rising edge may be up to 8 ns earlier than the mac3_clk rising edge.

2.7.3 SS Tx Asynchronous Data Transfer

In asynchronous transfer mode, a ring buffer is added between the first two registers. This accommodates long-term clock differences of up to 5600 ppm between the MAC3 and PIPE clock logic. When necessary, SKP and/or idle symbols are dropped or inserted in the data stream.

Downstream hub ports and all host ports except 0 are configured for asynchronous data transfer.

2.7.4 SS PHY Control Transfer

PHY controls are transferred in one of two ways:

- In synchronous mode, PHY controls are transferred similarly to data, but without the data width conversion.
- For other cases, PHY controls are transferred with a multi-bit synchronizer (bus synchronizer).

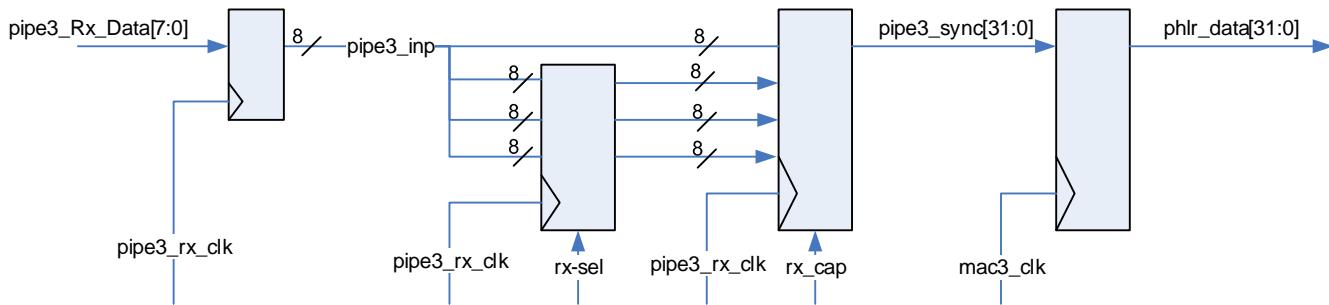
2.7.5 SS Rx Synchronous Data Transfer

Data from the SS PHY is collected into a 32-bit DWORD using `pipe3_rx_pcclk`. The 32 bits of data are then transferred synchronously at a safe time to the `mac3_clk` domain.

Device, upstream hub ports, and host port 0 are configured for synchronous data transfer.

[Figure 2-26](#) shows the data transfer logic for an 8-bit PHY.

Figure 2-26 SS Rx Data Transfer Logic (8-bit PHY)



The `rx_sel` and `rx_cap` signals (derived from the `pipe3_rx_pcclk` divider) are used to determine the transfer time. The `mac3_clk` may be delayed by up to 4 ns with respect to the `pipe3_rx_pcclk`. For more information, see [Figure 2-27](#) and [Figure 2-28](#) on page 118.

Figure 2-27 SS Rx Data Transfer (8-bit PHY; no mac3_clk delay)

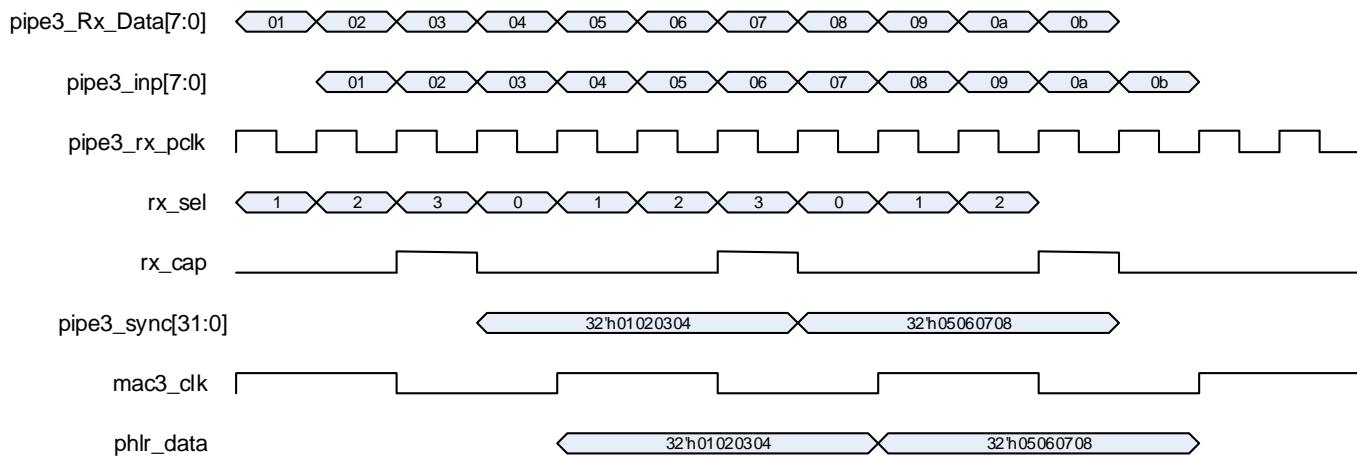
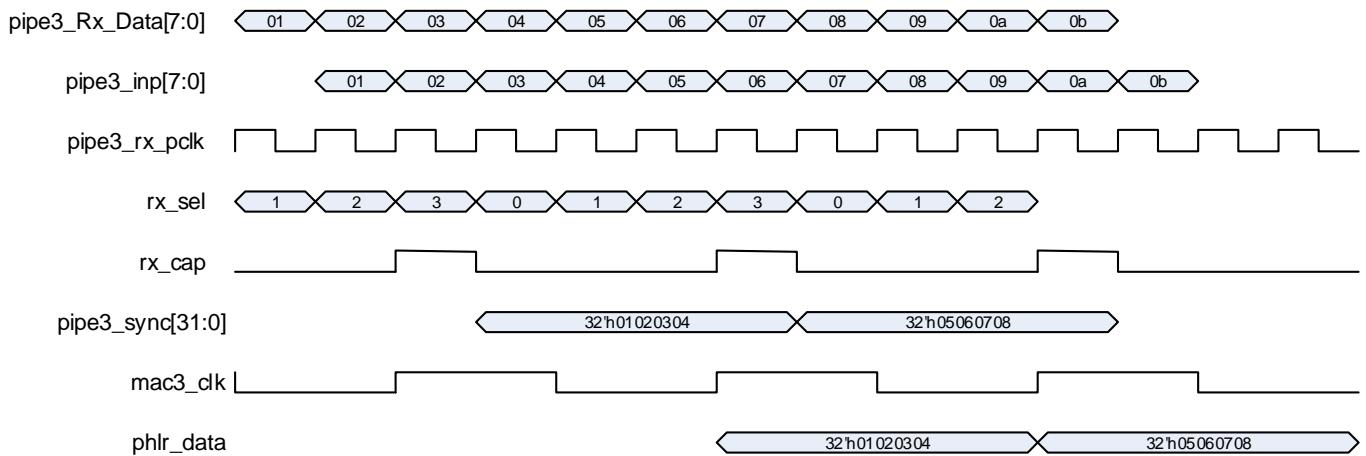


Figure 2-28 SS Rx Data Transfer (8-bit PHY; 4 nS mac3_clk delay)

2.7.6 SS Rx Asynchronous Data Transfer

In asynchronous transfer mode, a ring buffer is added between the last two registers. This accommodates long-term clock differences of up to 5600 ppm between the `mac3_clk` and `pipe3_clk` logic. When necessary, SKP symbols are inserted in the data stream.

Downstream Hub ports and all Host ports except 0 are configured for asynchronous data transfer.

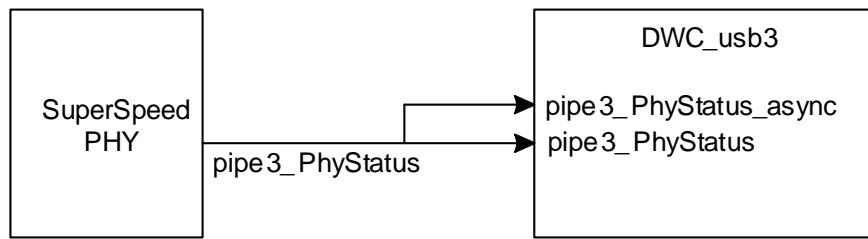
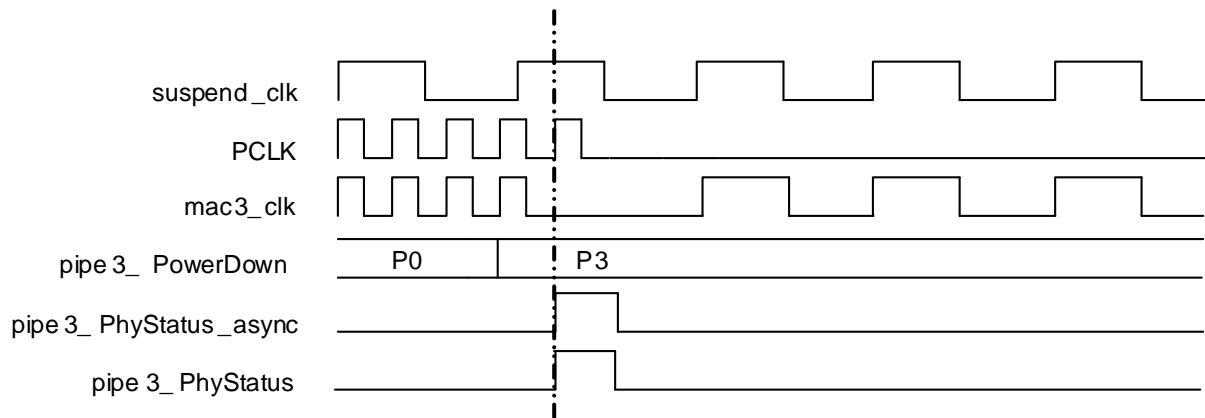
2.7.7 SS PHY Status Transfer

PHY status bits are transferred from `pipe3_rx_clk` to `mac3_clk` using a multi-bit bus synchronizer. In addition, the PHY status signals are processed in the following manner:

- The `pipe3_RxValid` signal must be asserted for four `pipe3_clk` (8-bit PHY), or two `pipe3_clk` (16-bit PHY) before an asserted value is transferred to `mac3_clk`.
- The signals `pipe3_PhysStatus` and `pipe3_RxStatus` are only valid for one `clk` cycle. When `pipe3_PhysStatus` is asserted, it and the corresponding `pipe3_RxStatus` bits are stretched before getting transferred to `mac3_clk`. In addition, before stretching, the two input signals status signals are passed through two stage synchronizers, because they are asynchronous when the PHY is in the P3 power state.
- If the LFPS filter bit in the `GUSB3PIPE3CTL` register is set, then the U3PIU provides an additional filtering operation on the detection of LFPS. LFPS reception in the P0 state is only considered valid if `phy_RxValid` is de-asserted, in addition to the normal condition of `phy_ElecIdle` de-asserted.

Capturing of PhyStatus for P3 Entry or Exit

In the PIPE3 Specification, the width of `pipe3_PhysStatus` is not defined for P0-to-P3, and P3-to-P0 transitions. It may be smaller than `suspend_clk` period. To overcome this, the controller uses `pipe3_PhysStatus_async` for P0-to-P3, and P3-to-P0 transitions. [Figure 2-29 on page 119](#) and [Figure 2-30 on page 119](#) show the top-level connections and timing diagram for PhyStatus.

Figure 2-29 Top-Level Connections for PhyStatus**Figure 2-30 Timing Diagram for PhyStatus**

2.7.8 SS PHY Hard Wired Controls

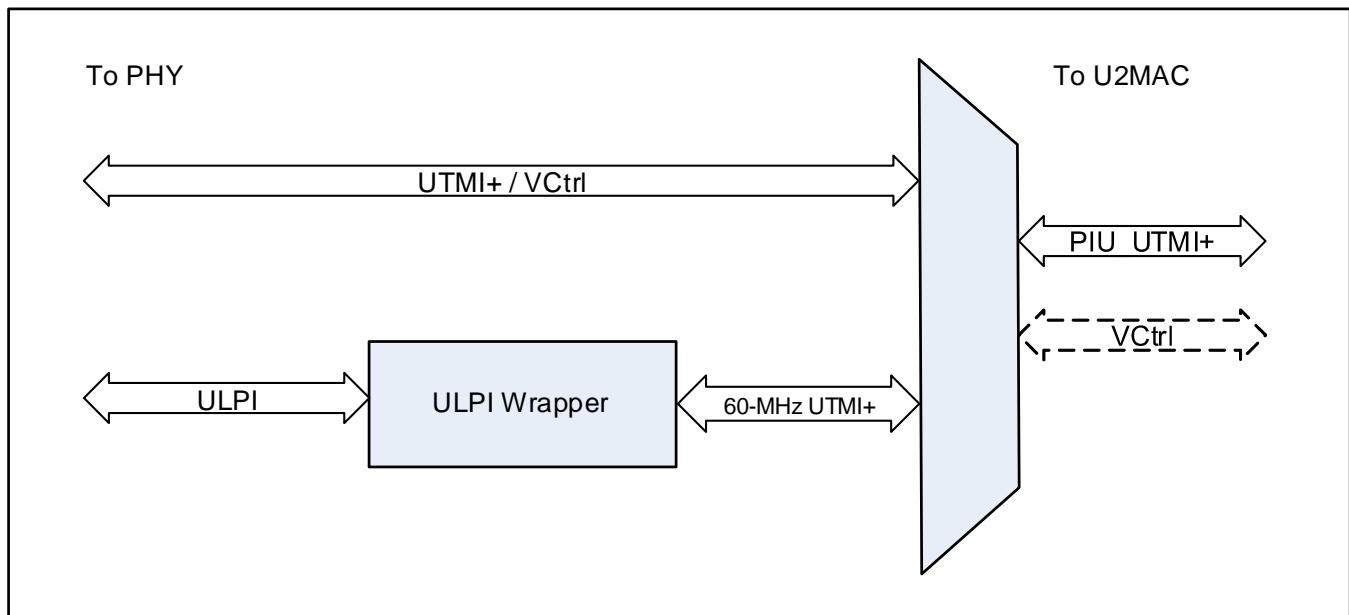
The following PHY output controls are hard-wired to USB 3.0 values:

- `pipe3_compliance = 0`
- `pipe3_PHY_Mode = 2'h1`
- `pipe3_Rate = 1`

2.8 USB 2.0 PHY Interface Unit (U2PIU)

Figure 2-2 shows the USB 2.0 PIU placement with the controller architecture. Figure 2-31 shows the overview of USB 2.0 PIU.

Figure 2-31 U2PIU Overview



2.9 USB 3.0 Link Power Management

When the USB is in sleep, suspend, or Vbus-off mode (for USB 2.0) or when the link is in the U1, U2, U3, SS.Inactive, SS.Rxdetect, or SS.Disabled state (for USB 3.0), the controller can be put into low power mode to save power.

2.9.1 LPM Functions



Note The terms LPM, L1, and sleep are used interchangeably in this document.

LPM functions are classified as either device-only or host-only. The device-only functions are discussed in the following sections. For information on low power mode operation in Host mode, see the xHCI Specification.

2.9.1.1 Detailed Device-Only Functions

The device supports USB Link Power Management states (see Table 1.1 of the USB 2.0 Link Power Management Addendum) by adding the new L1 (Sleep) state. If the host signals a reset in the Sleep (L1) state, the device enters the ON (L0) state.

2.9.1.2 Functions to Support L1 Entry

USB 2.0 Extension Transaction to Support LPM

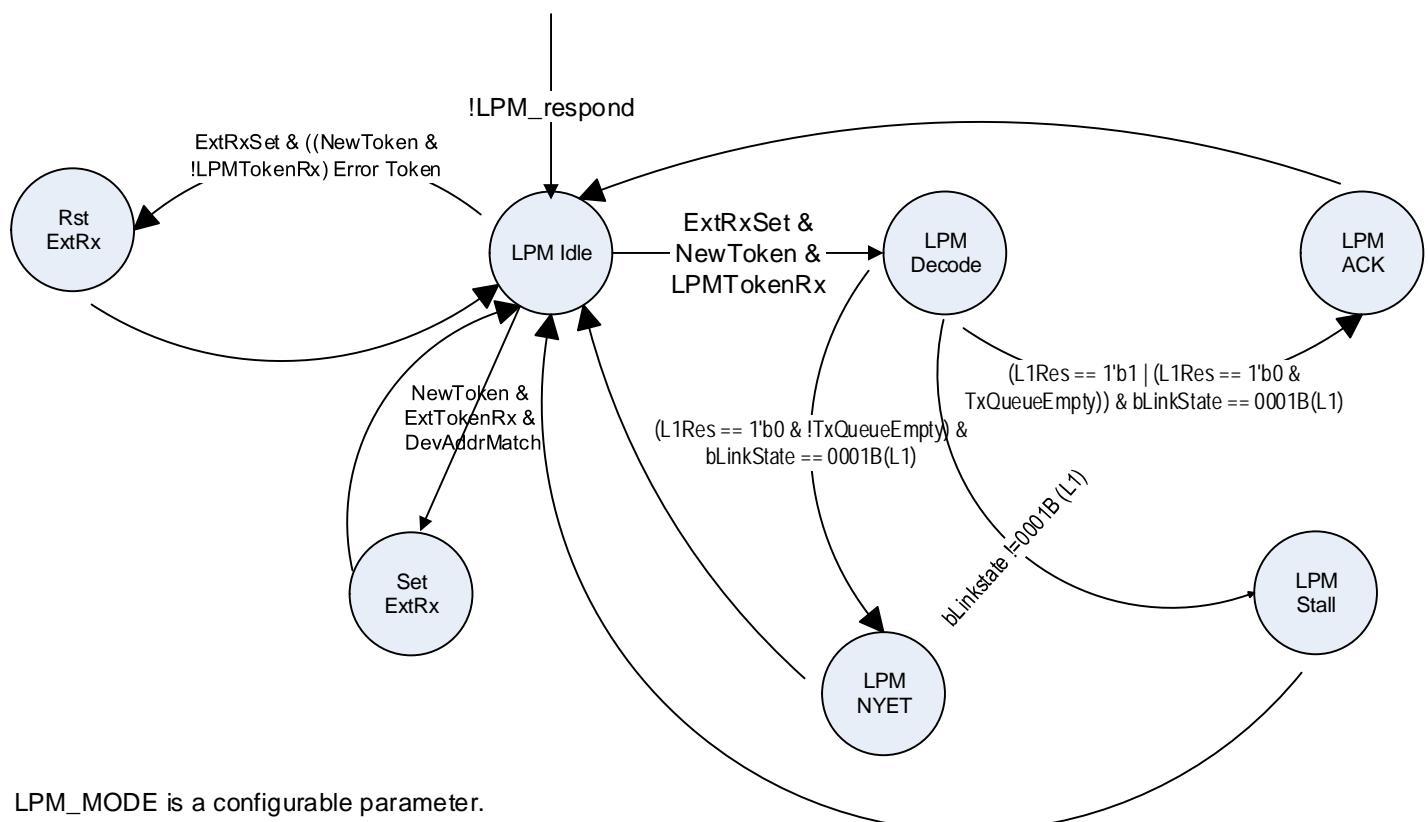
The controller responds to USB 2.0 extension transactions only if the application software sets the LPMCap bit in the DCFG. When the application sets the LPMCap bit, it can also enable low-power options by setting GUSB2PHYCFGn.EnblSlpM (enabling Sleep mode), or DCTL.HIRD_Thres (enabling Shallow Low-Power mode). The handshake response to LPM transaction is pre-programmed by the device application software using AppL1Res in the DCTL to ACK or NYET.

If LPM Errata is enabled, then the response is controlled by the LPM_NYET_thres field (DCTL[23:20]). AppL1Res is not applicable in this case, and LPM_NYET_thres then controls the response to the LPM token. When the HIRD value received in the LPM token is greater than LPM_NYET_thres value, then a NYET handshake is send. A NYET response is sent by the device controller irrespective of the BESL values for an error free LPM token (if any data pending in the Transmit FIFO or RxFIFO is not empty). If there is any error in the EXT or the LPM token received, it results in a timeout error.

The device response in each of these scenarios is described in the following sections.

Controller Handshake Response to LPM Transaction

The device response flowchart is shown in [Figure 2-32](#) on page [122](#).

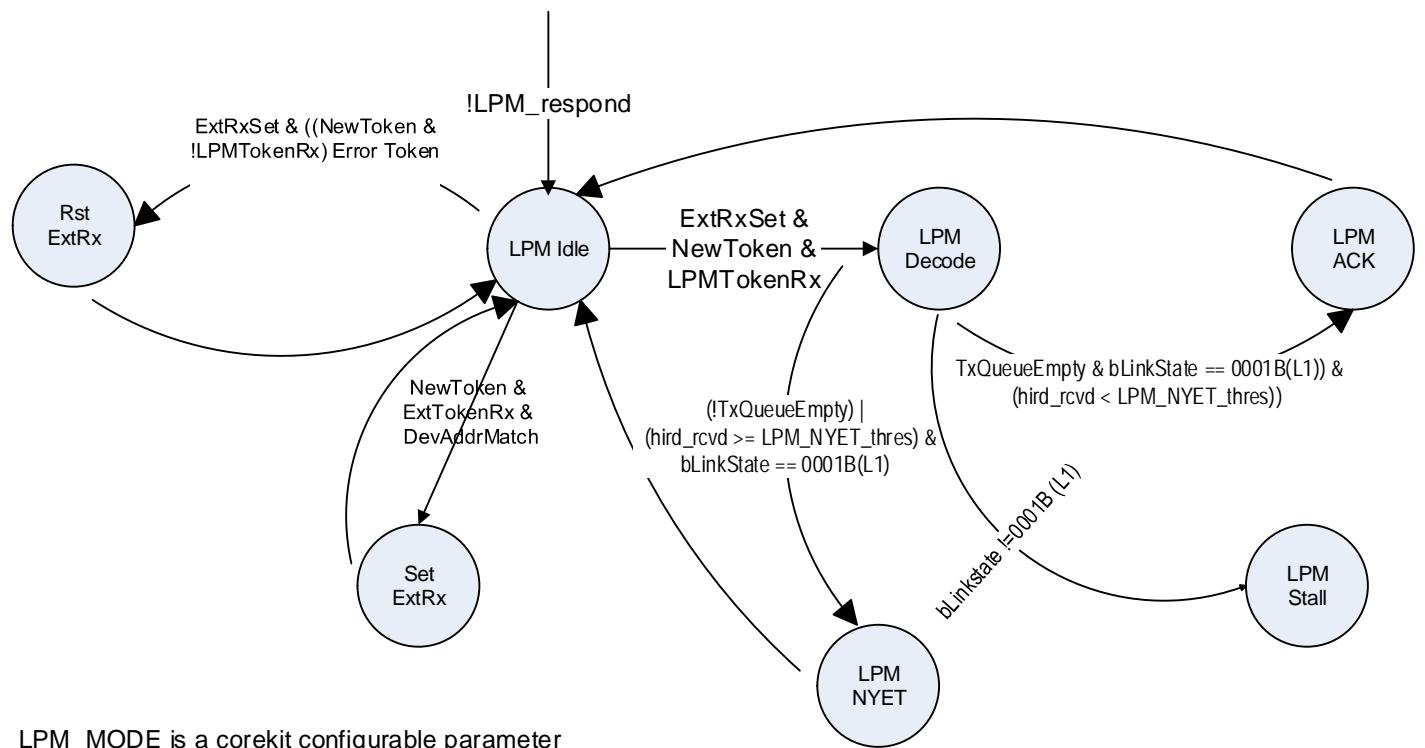
Figure 2-32 LPM Transaction: Device Response Flowchart

LPM_MODE is a configurable parameter.

LPMCap is a CR bit set by application software.

L1Res is a Handshake response to LPM token pre-programmed by the application software.

$LPM_respond = LPM_MODE \& LPMCap \& DeviceInL0$

Figure 2-33 LPM Transaction: Device Response Flowchart when LPM Errata is Enabled

LPM_MODE is a corekit configurable parameter

LPMCap is a CR bit set by application software

L1Res is a Handshake response to LPM token pre-programmed by the application software

LPM_respond=LPM_MODE & LPMCap & DeviceInL0

Hird_rcvd = This is the HIRD value in the received LPM token.

The LPM transaction consists of the following three packets:

1. Token packet with EXT PID 0000B from the host
2. Extended token packet with SubPID 0011B (LPM token) from the host
3. Handshake packet from the device

The LPM Transaction Response Unit is active (LPM_Respond = 1) and the ExtRxSet flag is reset when the following conditions are false. The remainder of the LPM transaction is ignored and no response (ERROR response) is sent either on the USB or to the application.

- There is a reception error.
 - PID error
 - CRC5 error
- The LPM-capable bit (DCFG.LPMCap) is not set.
- The device is in Suspend mode (L2).
- The device is not yet reset or enumerated (L3).
- The device is already in Sleep mode (L1).

If none of the previous conditions is true, the device infers that a new token has been received and responds to the EXT PID token or the LPM token.

Device Response to Token Packet With EXT PID 0000B

When the device receives a token packet, the device parses the token packet, compares the received PID with EXT PID, and compares the received ADDR field with the device ID programmed during enumeration.

- If there is an address and PID match and ExtRxSet is 0, the device sets the ExtRxSet flag and waits for the next token.
- If there is no address or PID match, the device resets the ExtRxSet flag. The device ignores the remainder of the LPM transaction and sends no response, either on the USB or to the application.

Device Response to Extended Token Packet With SubPID 0011B (LPM Token)

The device responds to an LPM token only if the ExtRxSet flag is set. Setting the ExtRxSet flag indicates that the EXT PID token was successfully received, that no other packet was received afterward, and that no error conditions (as defined in “[Device Response to Token Packet With EXT PID 0000B](#)”) are present.

The device thus parses the current packet received after a token packet with EXT PID, then takes one of the following actions, depending on the type of token received:

- If the EXT PID packet is repeated (ERROR)
 - If the next packet following the token packet with EXT PID is not an LPM token packet with LPM subPID, the device ignores the LPM transaction and sends no response.
 - If the next packet following the token packet with EXT PID is an EXT PID token packet, the controller considers it a new LPM transaction and sets the ExtRxSet flag.
 - After managing the ERROR response, the LPM Transaction Response Unit reverts to the Idle state, where it is ready to receive a new LPM transaction.
- Token with EXT PID followed by another token packet other than LPM/EXT token

Because the received packet is not an EXT PID token, the controller resets the internal ExtRxSet flag, and the LPM transaction unit takes no other action. If the received packet is a valid non-LPM or non-EXT token packet, the controller responds to it normally; otherwise, the device infers that an LPM token was received after the EXT token. It then sets a LPMTokenRx flag and decodes the LPM token. After decoding, the response can be STALL, NYET or ACK.

STALL Response

When there is no ERROR response, the device gives a STALL response under the following condition:

- The received LPM token’s bLinkState is anything other than Sleep (L1), in other words, not 0001B.

The device sends a STALL handshake packet as a response on the USB. After catering to the STALL response, the LPM Transaction Response Unit reverts to the Idle state, where it is ready to receive a new LPM transaction.

NYET Response

With LPM Errata Enabled: When there is no ERROR or STALL response, the device gives a NYET response if one of the following is true:

- If the BESL value received is greater than LPM_NYET_thres(DCTL[23:20] value).
- Conditions in “[DWC_usb3 Device LPM Support](#)” on page 126 for NYET response are met.

With LPM Errata Disabled: When there is no ERROR or STALL response, the device gives a NYET response if AppL1Res is programmed as 1'b0, and one or more transmit queues are not empty. A NYET handshake

packet is sent as response on the USB. After handling the NYET response, the LPM Transaction Response Unit reverts to the Idle state, where it is ready to receive a new LPM transaction.

ACK Response

When there is no ERROR, STALL, or NYET response, and AppL1Res is programmed as ACK, the device provides an ACK response under the following conditions:

Even though DCTL.AppL1Res to ACK is pre-programmed, the controller responds with an ACK only on a successful LPM transaction.

The LPM transaction is successful if:

- There are no PID/CRC5 errors in either the EXT token or the LPM token (else ERROR).
- A valid bLinkState = 0001B (L1) is received in the LPM transaction (else STALL).

After handling the ACK response, the LPM Transaction Response Unit triggers the L0-to-L1 Transition Unit. The LPM Transaction Response Unit reverts to the Idle state, where it is ready to receive a new LPM transaction.

2.9.2 Transition from L0 to L1

After sending an ACK response, the device transitions into the Sleep state. If the device's ACK response does not reach the host or hub, the host or hub may retry the LPM transaction to the device. The device must wait for this retry before it transitions to L1. The transition to the Sleep state is as follows:

1. Wait for Token Retry timeout

If an ACK is sent as response, the device waits for TL1TokenRetry = 8 μ s + 0.5 μ s (Here, 0.5 μ s is an additional buffer provided to accommodate delay variation in the host's retry response, while 8 μ s is the LPM specification value)

- If there is any transaction from the host on the USB to this device address before the TL1TokenRetry counter expires, the device resets the counter and does not transition to the L1 state.
- If the TL1TokenRetry counter expires, the device continues transitioning to L1.

2. Initiate Sleep state status update

3. Transition to SLEEP State

- The device controller changes UTMI PHY control signals to transition the bus into the L1 Sleep state. If in HS mode, the controller switches to FS mode as follows:
 - Change XcvrSelect from HS_XVCR to FS_XCVR.
 - Set TermSelect from HS_TERM to FS_TERM.
- The device controller asserts utmi_l1_suspend_n (1'b0) to the UTMI PHY (Automatic PHY Deep Low-Power control from the controller), if the HIRD value received from the host is greater than or equal to DCTL.HIRD_Thres[3:0] with DCTL.HIRD_Thres[4] and GUSB2PHYCFGn.EnblSlpM set.
- If utmi_l1_suspend_n cannot be asserted, the controller asserts utmi_sleep_n (1'b0) to the UTMI PHY (Automatic PHY Shallow Low-Power control from the controller) if GUSB2PHYCFGn.EnblSlpM is set.
- The device must not take more than TL1TransitionDev = 1 μ s (margin of 0.5 μ s = 10 - (8 + 1 + 0.5) over the LPM ECN specification value) to transition into the L1 sleep state.

2.9.3 L1 State Events

A (non-user) parameterizable down-counter is started ($\text{TL1Residency} = 50 \mu\text{s}$). When this counter expires in the L1 state, the L1ResumeOK status bit is set.

The TL1Residency time is essentially the PHY line states' settling time budget. The line states are valid only after this delay and the controller only looks for host-initiated resumes after it as well. Similarly, the device cannot initiate a resume until this counter has expired, because the host also must set its line states on the port after receiving an ACK response.

The device must come out of L1 whenever a reset or a resume signal is detected.

2.9.4 DWC_usb3 Device LPM Support

[Table 2-2](#) summarizes the device LPM behavior. It lists the conditions when the device accepts LPM (by sending ACK) during normal transfers with specific EP types, and the conditions when it does remote wakeup.

Table 2-2 Device LPM Support

Transfer Type	Accept LPM	Remote Wakeup
Bulk IN	When LSP is not ready	When LSP is ready and in L1 state
Bulk OUT	Always accept	When LSP is ready, EP in flow control state (last response was NYET/NAK), and in L1 state
Control IN/OUT	Not accept LPM during control transfer (handled in lower layer)	No remote wakeup
Interrupt IN	When LSP is not ready	When LSP is ready and in L1 state
Interrupt OUT	Always accept	No remote wakeup
Isoc IN	Always accept	No remote wakeup
Isoc OUT	Always accept	No remote wakeup

The device controller does not accept LPM and sends NYET when the following internal conditions occur:

- PSQ is full
- RxFIFO space is less than 1 mps for OUTs
- SPR (short packet received) is set



These are transient conditions and will change when the internal processing is done, after which the device accepts LPM if the conditions listed in the [Table 2-2](#) are met and the host retries on that endpoint. The device sends NAK responses to the normal transactions during the transient window, and NYET for LPM.

2.9.5 Device-Initiated L1 Exit

As long as the device is in the L1 state, it can initiate L1 exit if the device application software writes to USB State Change Request in DCTL. The device application software must not set the DCTL Link state Change to the wakeup bit unless the device controller is in an LPM suspended state (DSTS. USBLnkSt = Sleep State) after the previously received LPM transaction token. The device controller immediately changes the UTMI PHY control signals to initiate an exit from the Sleep state.



Note The bRemoteWake field in the LPM token controls the device-initiated L1 wakeup support. Depending on this field setting, the controller automatically initiates (or does not initiate) the remote wakeup signaling to the host. The bRemoteWake field is not exposed to the application interface.

Automatic Exit for L1

The controller supports automatic hardware L1 exit when transfers are getting ready in the device. This feature is controlled by a CSR bit in the GUCTL1 register.

When GUCTL1.DEV_L1_EXIT_BY_HW is enabled, the controller automatically initiates Remote Wake signaling when it becomes ready for sending/accepting new data. There is no need for the driver to check that the controller is in L1 and initiate RemoteWake.

- This feature is applicable only to bulk and interrupt transfers, and not for Isoch/Control.
- When control transfers are in progress, the LPM is rejected (NYET response). Only after control transfers are completed (either with ACK/STALL), is LPM accepted.
- For Isoch transfers, the host needs to generate the wake-up and start the transfer. The device controller does not do the remote-wakeup when Isoch endpoints get ready. The device software needs to keep the GUSB2PHYCFG.EnblSlpM reset to keep the PHY clock running to keep track of SOF intervals.
- When L1 hibernation is enabled, the controller does not automatically exit hibernation requests through L1.

2.9.6 Host/Hub-Initiated L1 Exit

In the L1 state, after the TL1Residency = 50 µs timer expires, the device controller listens to the line state. If a J-to-K transition is valid for a duration of Jresume:

- The device controller changes the UTMI PHY control signals to initiate an exit from the Sleep state.

2.9.7 Reset Signaling-Initiated L1 Exit

The device controller listens to the line state while in L1. Reset detection is performed even during the TL1Residency = 50 µs time period. If there is a J-to-SE0 transition valid for a time Jreset (2.5 µs), DEVT.USBRst is set for reset-initiated L1 exit.



- PHY power savings: The controller puts the PHY into Deep and Shallow Low-Power modes by automatically asserting utmi_l1_suspend_n or utmi_sleep_n, respectively, while in the L1 state.
- If utmi_l1_suspend_n cannot be asserted, the controller performs power-saving by blocking the PHY clock.

2.9.8 Compile Time Configuration Options

For more details on LPM compile-time configuration options, see Chapter 4, “Parameter Descriptions”.

2.9.9 Low Power Mode Operation in Host Mode

For information on low power mode operation in Host mode, see the XHCI specification.

2.9.10 LPM Errata Support

You can enable the LPM Errata mentioned in the xHCI BESL Errata Dated 10/19/2011 by enabling the Enable LPM Errata configuration parameter in coreConsultant.

Impact when the DWC_usb3 controller is in Host Mode

Enabling this feature has the following effects when the DWC_usb3 controller is operating in Host mode:

- In the xHCI capability registers, bit 20 of the USB 2.0 protocol defined field is set to 1 if LPM Errata is enabled. Setting this bit informs the xHCI driver that the host controller supports BESL decodings.
- The duration of Resume is based on the BESL values as compared to the HIRD values.

Impact when DWC_usb3 Controllers in Device Mode

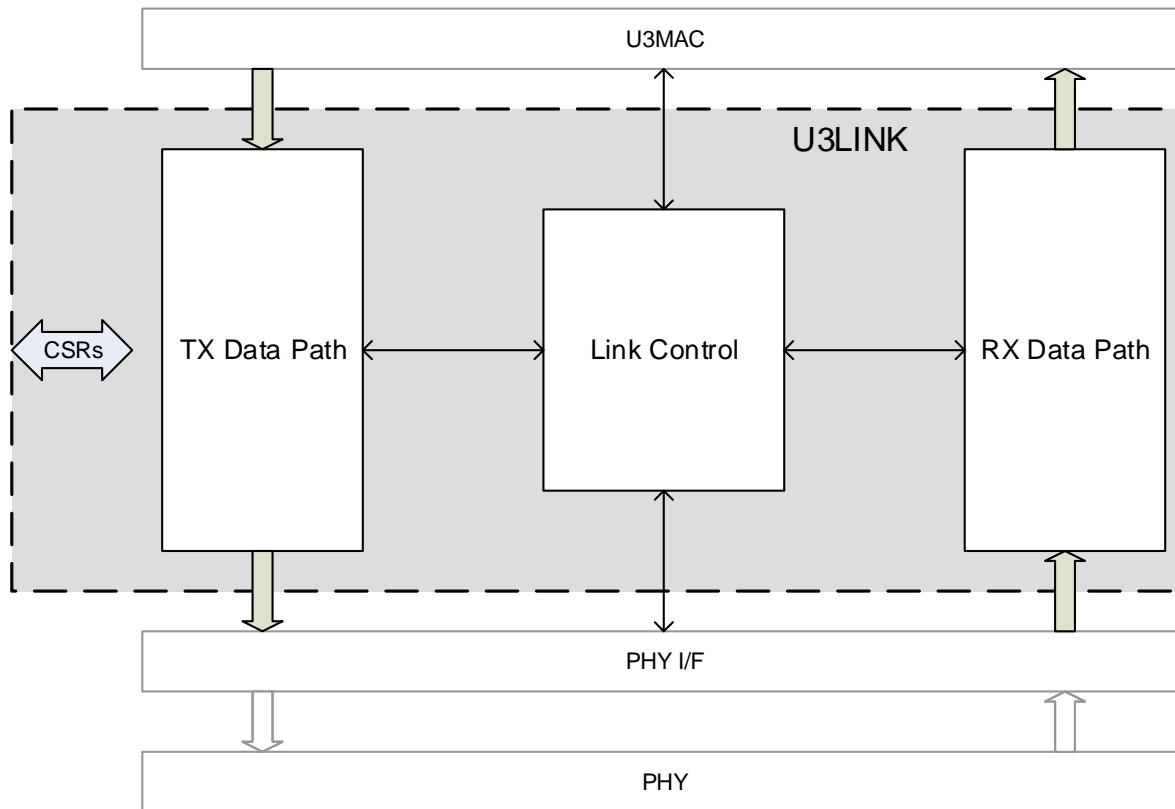
Enabling this feature has the following effects when the DWC_usb3 controller is operating in Device mode:

- A new field in the DCTL register is added that controls the ACK response from the device in response to an LPM token. {DCTL[23:20]} is used for this purpose. If the HIRD value received in the LPM token is greater than the register field value, then the device controller responds with a NYET response. If the HIRD value received in the LPM token is less than or equal to the register field, then the device controller sends an ACK response.
- Device driver for the controller must ensure that bits 15:2 of BmAttribute field in the USB 2.0 Extension Descriptor is set so that it can communicate its optimized power savings design points to the host. The driver must also be able to program the LPM_NYET_thres field (DCTL[23:20]) to control the NYET response if required.

2.10 USB 3.0 Link (U3LINK)

U3LINK provides packet framing and checking, as well as link and power state management. It also includes the physical layer skip, scrambling, and link training functions.

Figure 2-34 U3LINK Overview



The Tx data path includes transmit arbitration, insertion of link commands, transfer of header and data from the U3MAC, packet framing, header packet CRC generation, scrambling, and idle symbol, training set, and skip set insertion.

The Rx data path provides skip set removal, descrambling, training set and link command extraction, and received packet alignment. In addition, it checks packet framing and length, and, for header packets, checks CRC, sequence numbers, and credits.

Link Control functions include the link training state machine, link power state control, and header replay buffer.

The U3LINK internal datapath width is always 36 bits (four symbols including K/D bits). The interface to the U3MAC is always four symbols (32 bits) wide.

The Tx and Rx pipelines are actually wider than 36 bits because some control signals also have to be passed along.

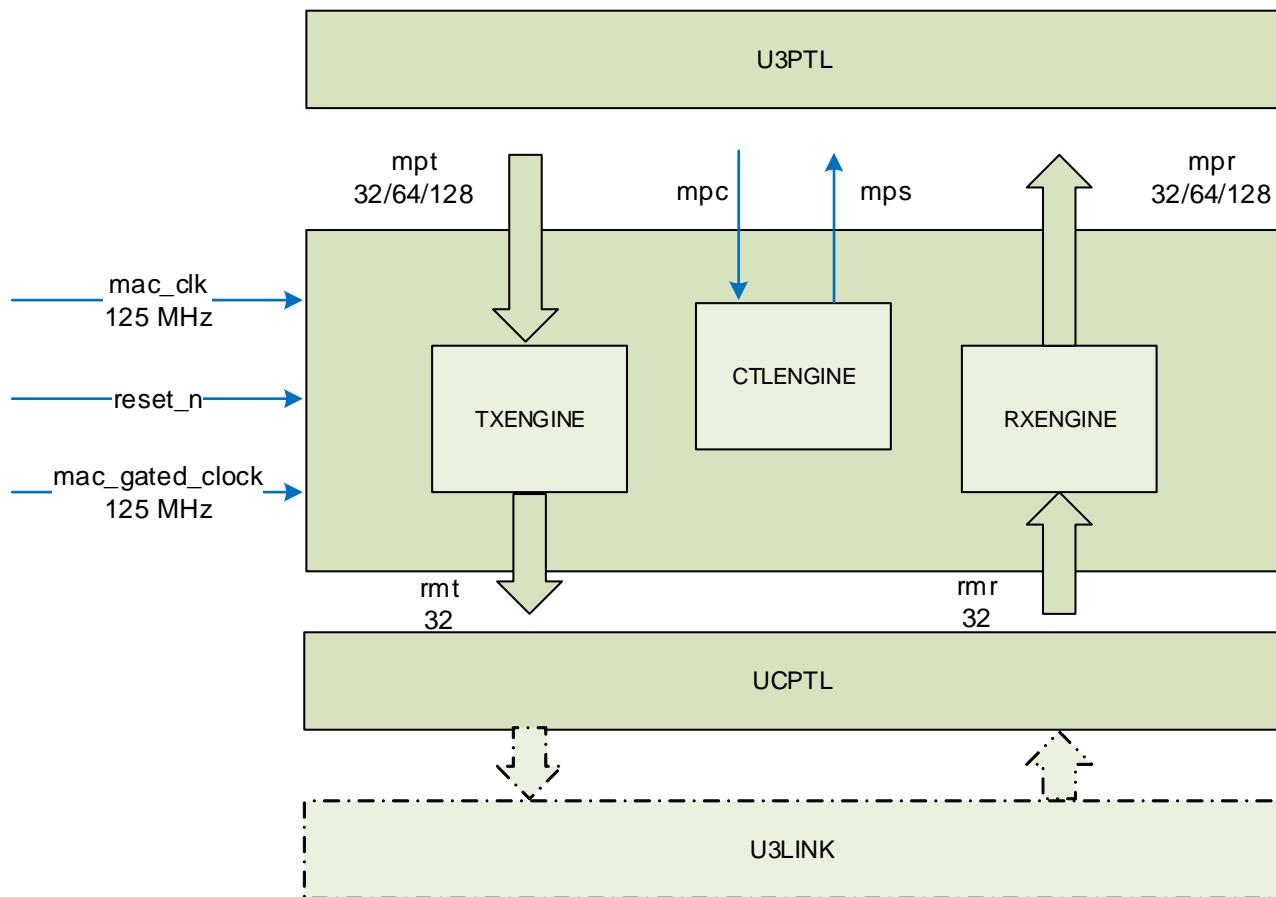


All required “between packet” states are in the Link control blocks (L^*). This is for convenience when implementing power down, retention flops, clock shut-off, and so on.

2.11 USB 3.0 MAC (U3MAC)

Figure 2-35 illustrates the block diagram of the USB 3.0 MAC block.

Figure 2-35 U3MAC Block Diagram



The module handles token transmit requests from U3PTL in Host mode.

In Device mode the module handles the token responses as follows:

- Build header packets (TP) and send to U3LINK.
- Build data payload packets and send to U3LINK.
- Decode responses to the transaction and provide it to U3PTL.
- Must avoid retransmitting data packet if the U3LINK failed to get LGOOD_* for data packet header.
- In a burst, must detect missed/error handshake as early as possible at header packet boundaries and should not abort the transfer. If the packet is data packet, then it can be aborted. Sending one additional header packet though previous header is corrupted should not impact performance severely.
- Must create corrupted data packet only if it is a true under-run or the previous header needs replay and currently this data packet is being transmitted.

2.12 USB 3.0 Protocol Transaction Layer (U3PTL)

The USB 3.0 Protocol and Transaction Layer handles the SuperSpeed transactions and works closely with the U3MAC to implement these SuperSpeed functions. The functionality of U3PTL and U3MAC is so closely intertwined that they can be considered as one entity known as the SMAC.

The U3PTL is solely responsible for tracking the IN and OUT burst transactions with the upstream or the downstream entity that it is connected to. According to the Section 4.4 of the USB 3.0 specification, “the SuperSpeed USB transaction protocol...allows more than one OUT ‘bus transaction’ as well as at most one IN ‘bus transaction’ to be active on the bus at the same time.” It is the U3PTL which keeps track of the state and context of these transactions.

Responsibilities of the U3PTL include:

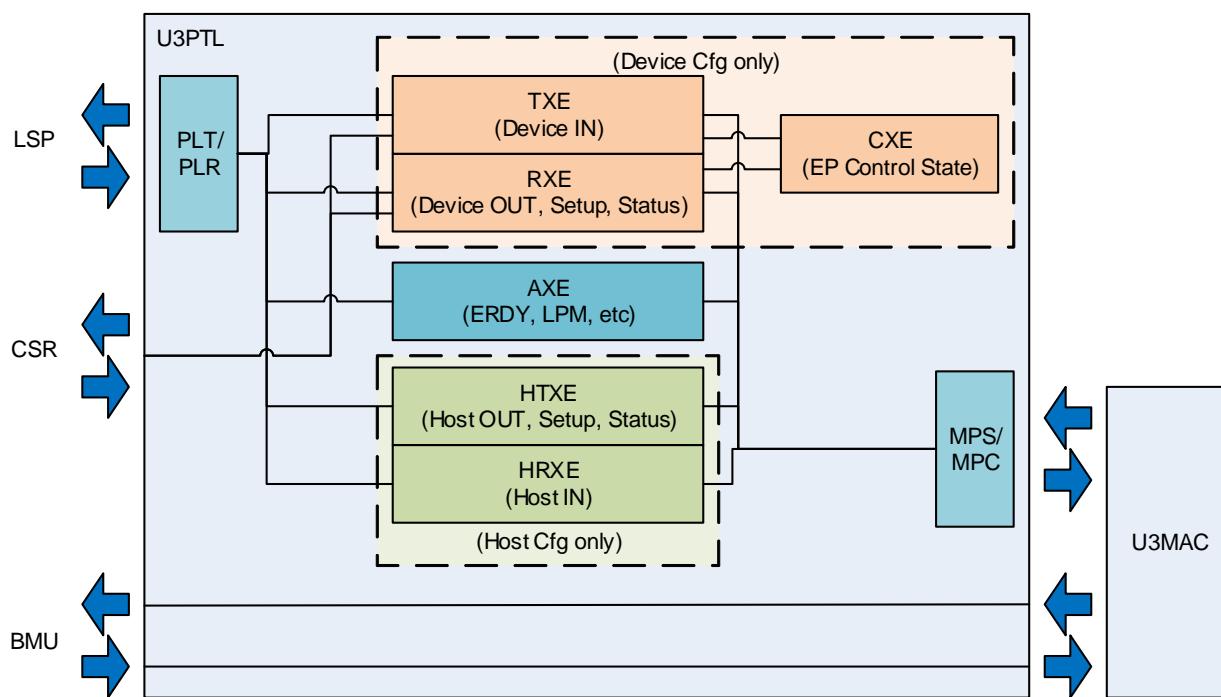
- Generating the packet requests to the U3MAC to USB (Data/Ack/NRdy/ERdy/Stall/Etc)
- Tracking data sequencing for all transactions
- Tracking control transfer state for all endpoints
- Abstracting USB protocol away from the upper layer (LSP) by generating non-USB protocol messages to the upper layer of the controller (LSP) for data tracking and EP state tracking
- Updating endpoint state context to the CSR
- Acting as a data path wrapper to the BMU data path interfaces
- Providing DRD support at the protocol layer



Note Only one of the modes (host or device) can be active at a time.

2.12.1 Block Diagram

The block diagram of U3PTL is shown in [Figure 2-36](#) on page [132](#).

Figure 2-36 U3PTL Block Diagram

2.12.2 Device OUT Transaction Engine (RXE)

In Device mode, the RXE handles all OUT transactions for all EP types, including SETUP, DATA WRITE, and STATUS phases of the control transfers. The RXE is active on a per-packet basis, which still supports burst transactions.

When the host initiates a new OUT transfer by sending a Data Packet (DP), the PTL decodes the Data Packet Header (DPH) and checks for matching device address, active EP, and status of the relevant resources. If all the conditions are met, the PTL generates the TP ACK response to the U3MAC to be transmitted back to the host. Otherwise, an appropriate response is generated (that is, NRDY, STALL, and so on). Context for the EP is updated to the CSR (that is, sequence number, flow control state, and so on). After this, the RXE goes idle and is ready for the next packet.

The upper layer (LSP) is kept informed of the success or failure of each packet transfer through messages pushed to the Protocol Status Queue (PSQ) in the BMU.

OUT bursts are automatically handled by this information because the EP context is always updated after every packet. In addition, once the response on the USB is generated, the protocol requirement is completed for the current packet, allowing the RXE to go idle and be ready for the next incoming packet.

2.12.3 Device IN Transaction Engine (TXE)

In Device mode, the TXE handles all IN transactions for all EP types, including DATA READ phases of the control transfers. The USB 3.0 specification requires that at most one IN transaction to be active at a time on the USB. For this reason, only one TXE resource is needed to support all IN transactions. The TXE must be active for the entire IN transaction, including the burst transactions. This is because of the duplex nature of the SuperSpeed USB. The device cannot predict when a response is received for a data packet it sends.

upstream. Often the response for a previous packet arrives while the device is simultaneously transmitting the next data packet.

When the Host initiates a new IN transfer by sending a Transaction Packet (TP), the PTL decodes the Data Packet Header (DPH) and checks for matching device address, active EP, and the status of the relevant resources. If all conditions are ok, then the PTL commences data packet transmission; keeping track of the appropriate sequence of data packets in the current burst transaction. Otherwise, an appropriate response would be generated (that is, NRDY, STALL, etc). When the entire transaction is complete, context for the EP is updated to the CSR (that is, sequence number, flow control state, etc). After this, the TXE goes idle and is ready for the next IN transaction.

The upper layer (LSP) is kept informed of the success or failure of each packet transfer through messages pushed to the Protocol Status Queue (PSQ) in the BMU.

2.12.4 Host OUT Transaction Engine (HTXE)

In Host mode, the HTXE handles all OUT transactions, including SETUP, DATA WRITE, and STATUS phases of the control transfers. The HTXE is active for one OUT burst transaction at a time.

As a host, it is the xHC LSP which initiates an OUT transfer by requesting to the U3PTL. For OUT transactions, the HTXE initiates transmission of the data packets. As the responses to the packets are received by the host controller, the U3PTL generates non-USB protocol messages to the LSP through the PSQ in the BMU to communicate successful or non-successful transfers, endpoint flow control, stalls, deferred packets, etc. The HTXE tracks the context of each transfer, including the data sequence of the burst. Context for the transfer is integrated into the messages to the LSP. There is no CSR interaction for Host mode U3PTL.

2.12.5 Host IN Transaction Engine (HRXE)

In Host mode, the HRXE handles all IN transactions, including DATA READ phases of the control transfers. The HRXE is active for one IN burst transaction at a time.

As a host, it is the xHC LSP that initiates the IN transfer by requesting to the U3PTL. For IN transactions, the HRXE initiates transmission of a TP ACK to the device. As responses are received by the host controller, the U3PTL generates non-USB protocol messages to the LSP via the PSQ in the BMU to communicate successful/non-successful transfers, endpoint flow control, stalls, deferred packets, etc. The HRXE tracks the context of each transfer, including the data sequence of the burst. Context for the transfer is integrated into the messages to the LSP. There is no CSR interaction for Host mode U3PTL.

2.12.6 Asynchronous Transaction Engine (AXE)

The AXE, active in both device and host modes, is used for non-data transaction packets such as ERDY, LMP, PING, PINGresponse, etc. These packets are always generated by the upper layer LSP and the transmit request is made to the U3MAC. The AXE waits for the transmission to complete, then goes idle and is ready for the next packet to be transferred.

2.12.7 Control State Engine (CXE)

The CXE maintains the control phase information for all endpoints and device address.

2.13 USB 2.0 MAC (U2MAC)

The U2MAC performs the USB 2.0 transaction level operations of token responses (Device mode) and generating token transactions (Host mode).

Device mode:

- Decode token packets and determine End Point.
- Check if the addressed End Point is in NAK mode, and if so, generate a NAK response without involving the U2PTL.
- If the addressed End Point is not in NAK mode, then inform U2PTL that a token was received and wait for the U2PTL to indicate that the appropriate FIFO is ready.
- Perform the movement of data for the transaction between Link layer and BMU.
- Report status of the transaction back to the U2PTL upon completion of the transaction.

Host mode:

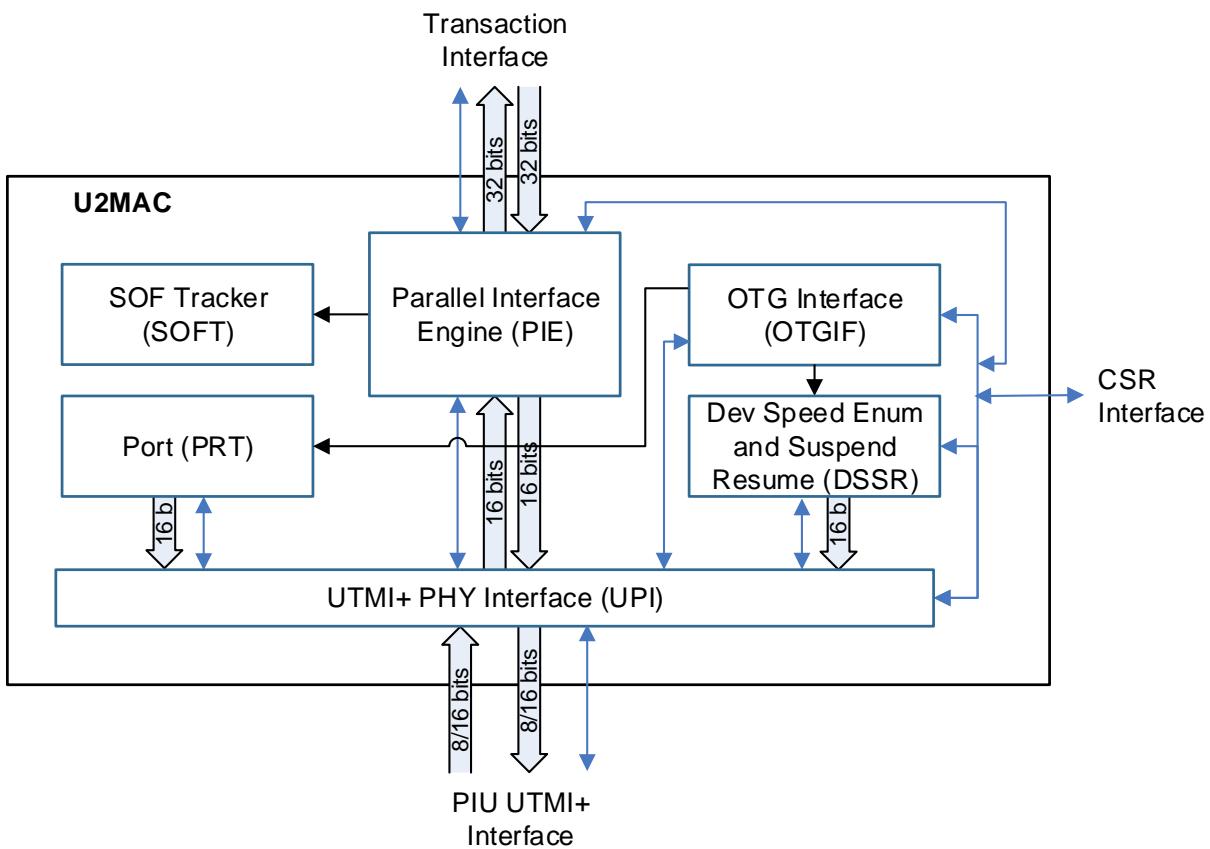
- When initiated by the U2PTL, build token packets and send to Link.
- Perform the movement of data for the transaction between Link layer and BMU.
- Report status of the transaction back to the U2PTL when the transaction completes.

2.13.1 U2MAC Block Description

The U2 MAC is based on the MAC of the DWC_otg controller to use a proven technology and to reduce development time required for this backwards compatibility block. The MAC consists of these major functional parts:

- PIE - Parallel Interface Engine
- DSSR - Device Suspend and Resume Controller
- PIU - PHY Interface Unit
- SOFT - SOF Tracker

Integration of the MAC components of the OTG controller requires interface translation to the U3's CSR. This is handled by the U2PTL.

Figure 2-37 U2MAC Block Diagram

2.13.1.1 Parallel Interface Engine (PIE)

The PIE handles the USB 2.0 bus transactions and packets including generation and transmission (Host mode) of the token packet, data packet, and handshake packets. In Device mode, the PIE identifies and decodes the received token, data, and handshake packets, generates CRC5 and CRC16, checks packet integrity, checks timeout and maintains interpacket delay, reads/ writes packet payload from/ to FIFOs, decides appropriate response to the host/ device based on CSR information and FIFO status, and updates transaction status through the transaction interface.

2.13.1.2 Device Suspend and Resume Controller (DSSR)

The DSSR, only used in Device mode, handles the USB reset sequence and determines the operating speed, handles Device mode suspend/resume/remote wakeup, and SRP (in Device mode).

2.13.1.3 OTG Interface Controller (OTGIF)

The OTGIF handles the Host Negotiation Protocol (HNP) to switch the role (Device mode or Host mode) of the controller. It also handles the Session Request Protocol (SRP) to generate the session request in Device mode and detect session request in the Host mode.

2.13.1.4 PHY interface Unit (PIU)

The PIU is the PHY U2 interface to the controller. Depending upon configuration, this supports the UTMI+, ULPI, and LS Serial interfaces. The U2 MAC communicates with the PIU through the UTMI interface.

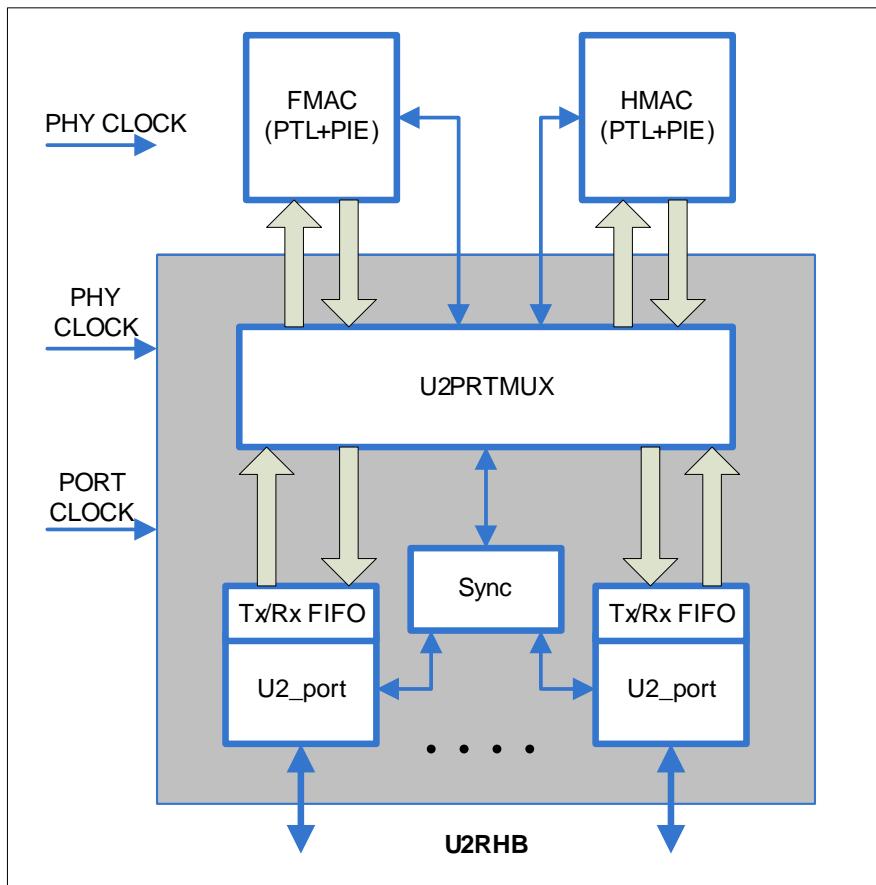
2.13.1.5 SOF Tracker (SOFT)

The SOFT is only active in Device mode. It tracks SOF packets and generates the SOF interrupt to the CSR in the Device mode. It handles missing SOF and delayed SOF to keep the frame number synchronization between the host and the device.

2.14 USB 2.0 ROOT HUB (U2RHB)

The U2RHB routes the USB packets and control signals between the U2MAC and U2 Root Ports with proper synchronization. This module is used in Host mode operation only. In USB 3.0, because the broadcasting of packets to all the ports is not necessary, the packets are routed to the required ports only. [Figure 2-38](#) shows the block diagram of this module.

Figure 2-38 U2RHB Block Diagram



This module consists of the following sub-blocks:

- Port MUX (U2PRTMUX)
- Synchronizers
- Port Modules (U2PORT)

2.14.1 Port MUX (U2PRTMUX)

This module multiplexes the signals from multiple MAC instantiations to the required root port based on the current port mapping information and the connected device speed in the OUT direction. Similarly, in the IN directions, the signals from multiple root ports are routed to the required MAC based on the port mapping information and the connected device speed.

2.14.2 Synchronizers

The MAC works on a common 30/60 MHz clock and the individual ports work on their own 30/60 MHz port clocks. Because they are asynchronous clocks, there must be synchronizers in between the ports and the MACs. The U2SYNC and the FIFO modules RHURX and RHUTX work as synchronizers.

2.14.3 Port Modules (U2PORT)

The port modules work as HS, FS or LS ports based on the connected device speed. They also support the UTMI or ULPI interface based on the configuration used.

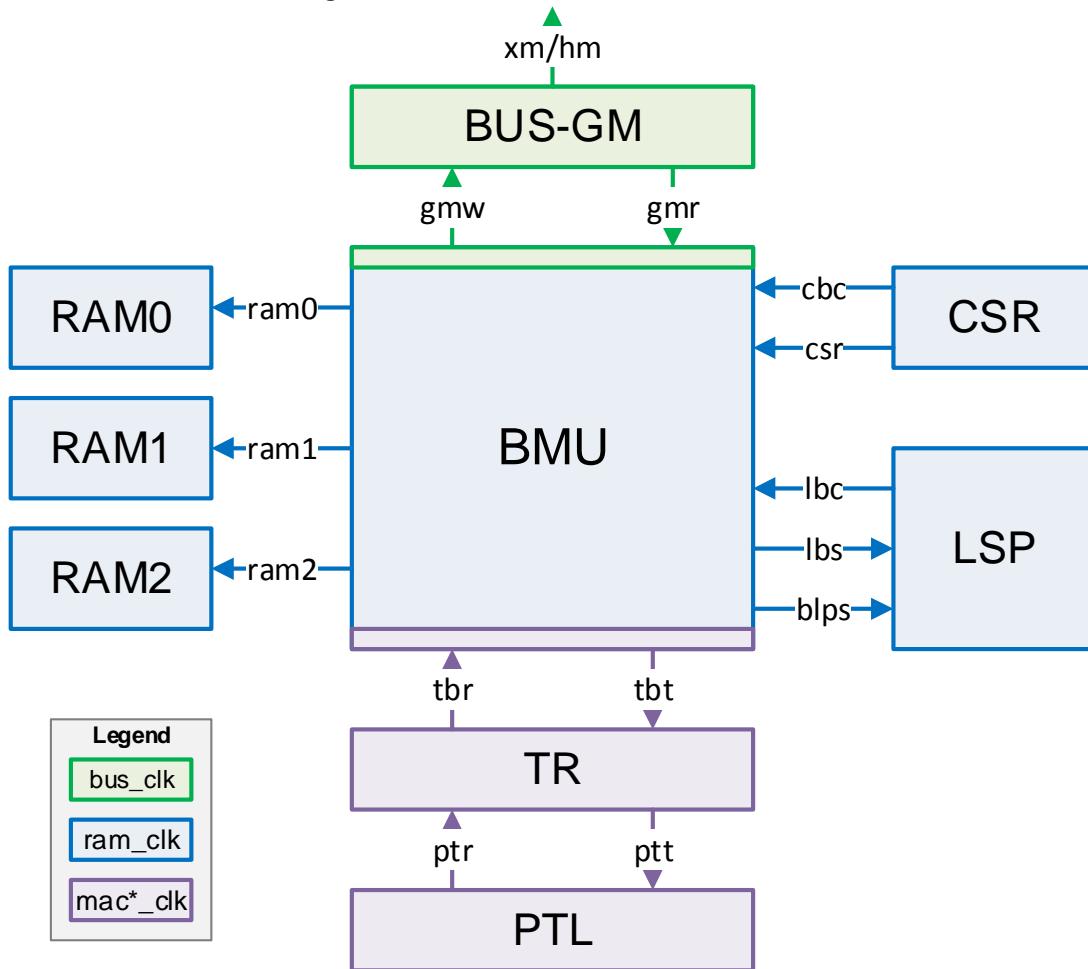
2.15 Buffer Management Unit (BMU)

The DWC_usb3 Buffer Management Unit (BMU) handles the buffering requirements between the system bus and USB bus instances. It executes DMA operations and USB transfers concurrently, while handling packet error conditions that require selective packet flush or FIFO flush. It prioritizes DMA requests based on USB activity and software programmed registers.

2.15.1 Interfaces

The BMU interfaces to the LSP, BUS-GM / native master interface, internal RAMs (one to three single-port or 2-port RAMs), CSR, and multiple instances of the 2.0/3.0 host/device PTLs via the transaction router—exact number of PTLs depends on the number of configured SS/HS/FSLS USB bus instances along with other device, host, and DBC parameters.

Figure 2-39 BMU Interfaces Block Diagram



2.15.2 High-level Flow

LSP issues DMA commands to manage the descriptor cache and transfer packet payloads. Device LSP issues a TXFIFO flush upon receiving an ACK-RETRY or End Transfer command (device IN endpoints each have a dedicated TXFIFO). Host LSP issues a TX packet flush after each TX endpoint burst boundary to clear any packets that were not sent due to system bus delays (host OUT endpoints have a shared TXFIFO).

Each PTL instance manages USB burst transactions, pushes protocol messages to the corresponding BMU PSQ, pushes RX data into the RXFIFOs, and requests TX data from the TXFIFOs. Device PTL requests TX data upon receiving an ACK-TP. Host PTL requests TX data only after the LSP instructs it to do so, which ensures that a threshold packet amount of TX data is in the TXFIFO before starting a TX-direction USB burst transaction.

BMU conducts full-duplex DMAs (including pipelined reads) of descriptors as well as data payloads to/from the internal cache (RAM0) and data payload FIFOs (RAM0/1/2).

DMA

The BMU contains a main DMA FSM, which handles concurrent system bus read and write operations; and, if enabled, up to two outstanding DMA read transactions of data payloads (that is, up to two 1KB packets). Such packet-level DMA pipelining increases bus performance when using AXI or the native interface by reducing the effect of initial access latencies when using PCIe or other system busses with latency. Furthermore, the AXI and AHB gaskets break each packet-level DMA request into bursts of programmable size (for example, a 1 KB packet may consist of 8 AXI transactions of 16 beats each when MDWIDTH=64).

Queue RAM Allocation

Depending on the mode (device, host, DRD), number of device IN EPs or Host BIs, params.v allocates the internal RAM0 while the bmu_rac instantiates the required number and size of TxFIFOs (TxInfo dword header precedes each packet in the TxFIFOs), RxFIFOs (plus separate RxInfo queue FIFOs), and DMA Request Queues (TXQ, RXQ, DFQ, DWQ, and EVQ). The FIFO sizes and priorities are auto-programmed and can be overridden by software – this can be particularly useful for isochronous endpoints when the system bus bandwidth is limited relative to the aggregate USB bandwidth. Because the BMU operates primarily on the ram_clk domain, but interfaces to the bus_clk and mac_clk/mac2_clk/mac3_clk domains, it instantiates the appropriate number of flop-based clock crossing FIFOs and handles toggle-based control signals locally.

LSP Queue/FIFO Operation

LSP/FIFO Queue	Operation
DFQ	LSP issues descriptor fetch DMA requests to cache TRBs.
DWQ	LSP issues descriptor write DMA requests when transfers described by TRBs are complete.
PSQ	LSP reads protocol status information from a RAM-based PSQ (device or DBC mode) or directly from flop-based PSQs per BI (Host mode).
RXQ	Upon reading a DPH from the PSQ, the LSP issues an RX DMA request (consisting of one or more segments depending on TRB setup) or RX packet flush request (upon error condition). If the RX status indicates an error condition (for example, CRC), the BMU auto-flushes the packet based on the status buffered in the RxInfo queue (RIQ).

LSP/FIFO Queue	Operation
TXQ	After receiving a Start transfer command or fetching TRBs for the TX direction, LSP issues TX DMA requests (one per packet) for the entire burst (Host mode) or however many requests the TXQ can accommodate (Device mode).
TXFIFO Threshold	In Host mode, when a TXFIFO packet count exceeds the CSR's TX threshold level, LSP instructs the PTL/MAC to start a burst. (In Device mode, the PTL/MAC automatically requests the required TXFIFO upon receiving an ACK-TP.)
TXFIFO Flush or TX Packet Flush	In Device mode, if the host requests a packet retry, the LSP issues a TXFIFO Flush command to clear the entire TXFIFO, DMA, and TXQ; and then re-fetches the required packets. In Host mode, if the DMA fetch does not complete in time or the downstream device requests a packet retry, the LSP issues a TX Packet Flush to clear the indicated TXFIFO, DMA, TXQ of packets for that specific endpoint. (In Host mode, each BI per speed has its own TXFIFO shared among all OUT endpoints, whereas, in Device mode, each IN endpoint has its own separate TXFIFO.)

MAC/PTL Operation

MAC/PTL	Operation
tbr_msg	The MAC/PTL pushes DPH, RX status (errors include CRC error, RX overflow, and others), and TX status information to the BMU through the tbr_msg interface (per BI in Host mode).
tbr_data	PTL pushes RX data payloads to the RXFIFO through the tbr_data interface.
tbt_cmd	MAC/PTL issues a TxRequest on the tbt_cmd interface to start each packet transmission.
tbt_txinfo	In response to a TxRequest, the BMU indicates the status and packet header information through the tbt_txinfo interface. The txinfo may indicate packet ready or not ready and, in Device mode, a StreamID mismatch for ISOC IN EPs.
tbt_data	In addition to the tbt_txinfo, the BMU supplies TX payload data (if any) through the tbt_data interface.
tbt_cmd TxAck	Normally, the MAC/PTL issues a TxAck (which signals to the BMU that this packet has completed and to prepare for the next packet if available) after accepting all the data from the tbt_data interface. However, if there are not enough link credits available, the MAC interface may issue a premature terminating TxAck before accepting all the data from the tbt_data interface. In such a case, the BMU self-flushes the tbt_data 2clkfifo source buffer and expects the Host LSP to issue a TX packet flush to eliminate the remainder of the packet and any others that may be in the DMA pipeline or the device LSP to issue a TXFIFO Flush.
tbt_cmd TxRewind	Upon receiving a NAK from the host, the U2 device PTL issues a TxRewind request, which permits subsequent re-transmitting of the most recent packet from the TXFIFO rather than having to re-fetch the packet from the system bus. Device U3PTL and host PTLs always issue TxAck and not TxRewind.

CSR

The CSR has access to RAM-based registers in RAM0 as well as debug write and read access to all RAMs through the cbc interface, which passes on requests from the BUS-GS slave interface.

2.15.3 Block Description

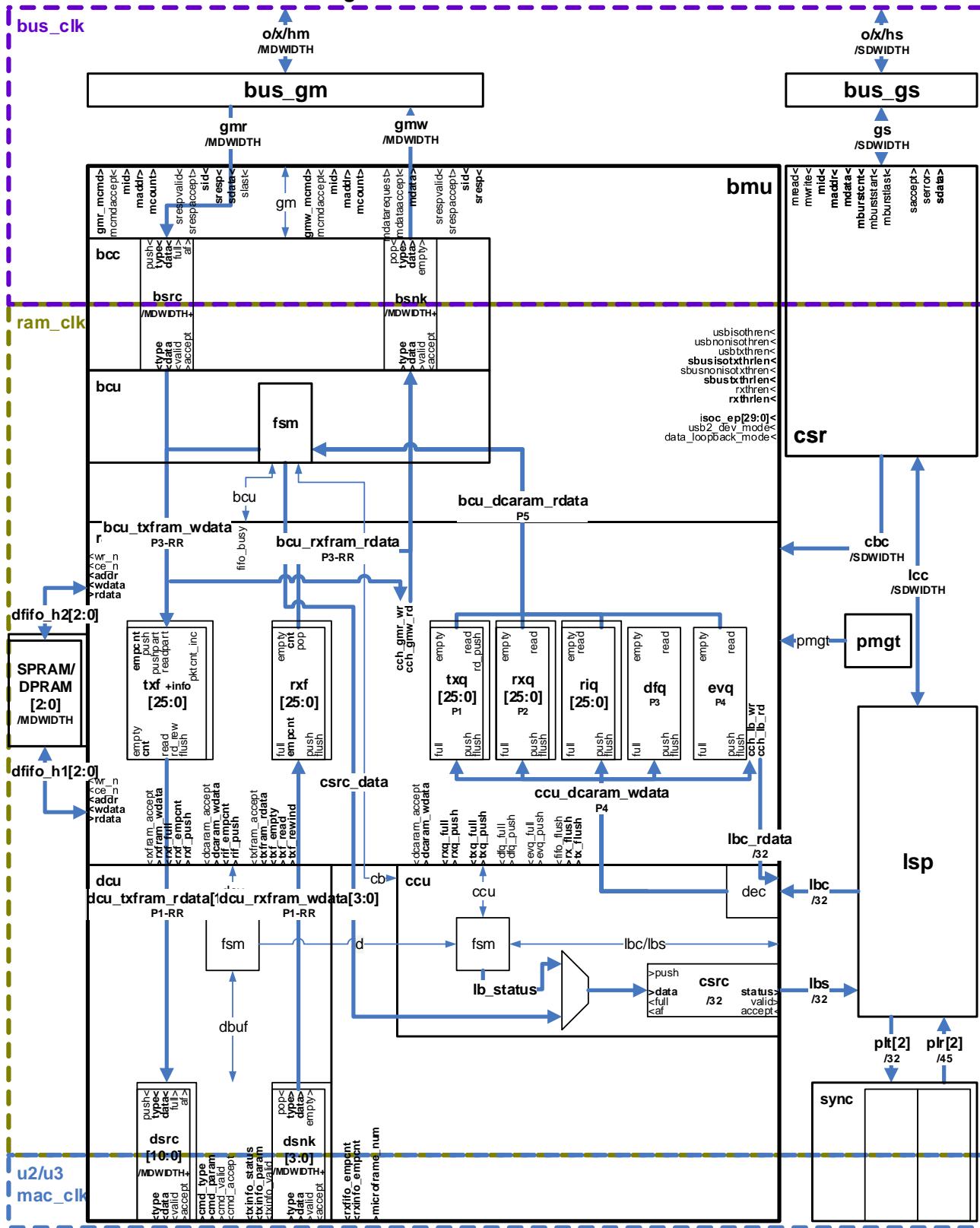
[Table 2-3](#) describes the BMU sub-modules.

Table 2-3 BMU Sub-Modules

Module	Function
RAC	<p>RAM Access Controller</p> <p>Arbitration of RAM interfaces (configurations: 1, 2, or 3 RAMs; single-port or 2-port). Contains all RAM-based programmable FIFO controllers for data buffering and fixed-depth FIFO controllers for DMA request queues. Arbitrates RAM access among multiple agents in a fixed priority order. For example, MAC traffic has highest priority, while LSP cache access has lowest priority. Also, multiple MACs may be receiving or transmitting simultaneously at different data rates; the RAC ensures that the bandwidth is fairly divided on a FIFO-word basis through a weighted-round robin scheme.</p> <p>Note:</p> <p>A similar scheme on a packet basis is used in the BCU for selection of TX and RX DMA requests among competing USB bus instances.</p>

Module	Function
BCU	<p>System Bus Control Unit</p> <p>The main DMA FSM handles concurrent write and read DMAs. If NUM_OUTSTANDING_TXDMA is 2, it also handles up to two (2) outstanding DMA read requests for data payloads of the same or different endpoints on one or more USB bus instances. It initiates DMA based on the presence of DMA requests in the queues, corresponding FIFO data/space availability, and a complex priority order.</p> <p>RXQ, DWQ, DFQ implicit priority requirements -- For both device and Host mode, RX data DMA has priority over pending descriptor write DMAs since the LSP may queue a TRB writeback after receiving the final packet for that buffer. Similarly, descriptor write DMAs have priority over pending descriptor fetch DMAs; this ensures that any subsequently issued descriptor fetch does not skip ahead of a delayed descriptor write DMA -- in addition, the BCU waits for the DMA write response before starting a DMA read in this scenario.</p> <p>RXQ, DWQ, DFQ priority -- Note that DMA writes and DMA reads occur concurrently; moreover, the implicit ordering requirements described above are BI-specific. Therefore, we have the following DMA queue priority order (highest to lowest):</p> <ol style="list-style-type: none"> 1. Descriptor write DMA queues of BI's with empty corresponding RX DMA queues. 2. RX DMA queues. 3. Descriptor fetch DMA queues of BI's with empty corresponding descriptor write DMA queues. 4. TX DMA queues. <p>In this way, descriptor writebacks have the best opportunity to be handled and avoid blocking descriptor fetches; while keeping both directions of the system bus used if there are pending RX and TX DMA requests.</p> <p>Normally, the RXQs and TXQs are effectively handled concurrently (one after another, approximately a dozen clock cycles) because the system bus is typically full-duplex. However, a non-full-duplex bus like AHB causes the BUS-GM to alternate access between the write and read directions.</p> <p>Separate descriptor queues—Given the above implicit queue ordering restrictions and the reality that USB bandwidth can exceed system bus bandwidth, it is useful to separate the descriptor queues on a BI-basis to enable prioritization of one BI over another. This separate descriptor queue architecture is enabled automatically for MDWIDTH=64/128 3.0 host-enabled configurations with 8 or fewer interrupts; internally, this architecture is selected by DWC_USB3_EN_SEPARATE_DESC_QUEUES and means that each BI's RXFIFO/TXFIFO is paired with its own descriptor fetch queue and descriptor write queue; in addition, one auxiliary event queue per interrupt is instantiated to handle the event buffer DMA address and cycle bit.</p> <p>Global DMA priority registers (for QoS)—Regardless of whether separate descriptor queues are instantiated per BI, software may prioritize individual TXFIFOs or RXFIFOs. The arbitration scheme consists of two round-robin arbiters per direction per mode (device and host) with the result of the high-priority round-robin arbiter trumping the result of the low-priority round-robin arbiter. This scheme allows for assigning device TXFIFOs or host TXFIFOs and RXFIFOs containing ISOC traffic a high priority while BULK traffic gets assigned a low priority.</p>
CCU	<p>Control Path (LSP) Control Unit</p> <p>Pushes DMA requests into the proper queues, writes/reads the Cache RAM, and handles FIFO and packet flush commands. Provides status response to the LSP when the command completes.</p>
DCU	<p>Data Path (PTL) Control Unit</p> <p>Delivers requested TxInfo / TxData, writes RX data to RAM, rewinds 2.0 device TX data, handles host TX packet flush commands.</p>

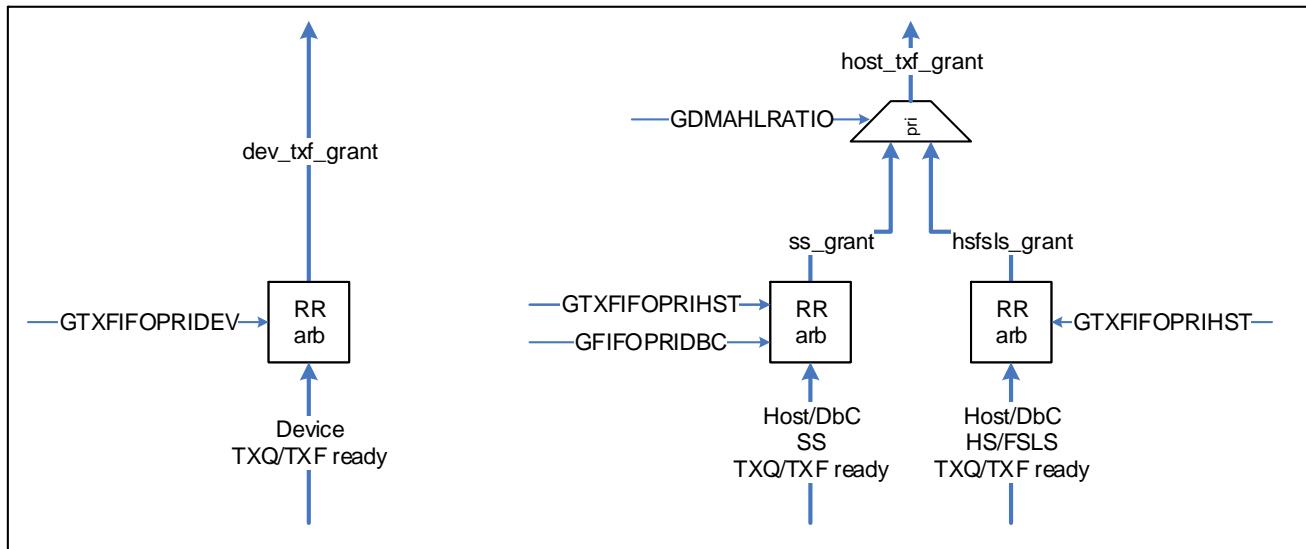
Figure 2-40 BMU Micro-Architecture Diagram



2.15.4 TxFIFO Arbiters

Figure 2-41 shows the device and Host mode TxFIFO arbiters with the programmable priority/ratio registers. The Host mode also has an equivalent RXFIFO arbiter.

Figure 2-41 Device and Host TxFIFO Programmable Priority Round-Robin Arbitration



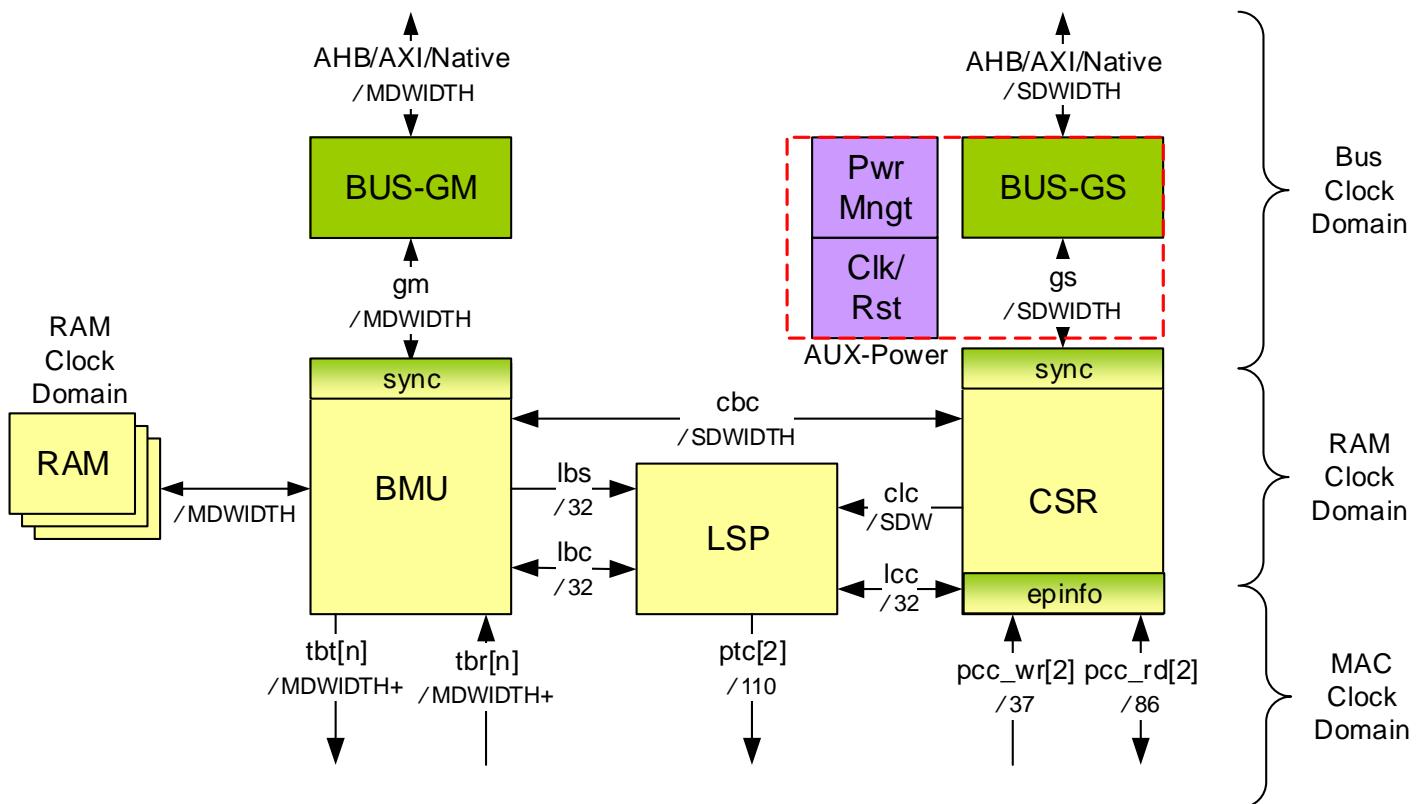
2.16 List Processor (LSP)

The List Processor module manages interaction between the USB bus, the system bus, and the driver.

- USB bus: Receive and transmit transaction packets (TP), data packets (DP), and Link Management Packets (LMP)
- System bus: Request descriptor fetch and write back
- System bus: Request DMA reads into TX FIFOs and DMA writes from RX FIFOs
- Driver: Handle generic commands and side-effects of CSR writes
- Driver: Initiate interrupts and write out events to be acted upon by the driver

The PTL, LSP, CSR, and BMU blocks make up the Protocol Layer of the USB 3.0 controller.

Figure 2-42 Protocol Layer Diagram

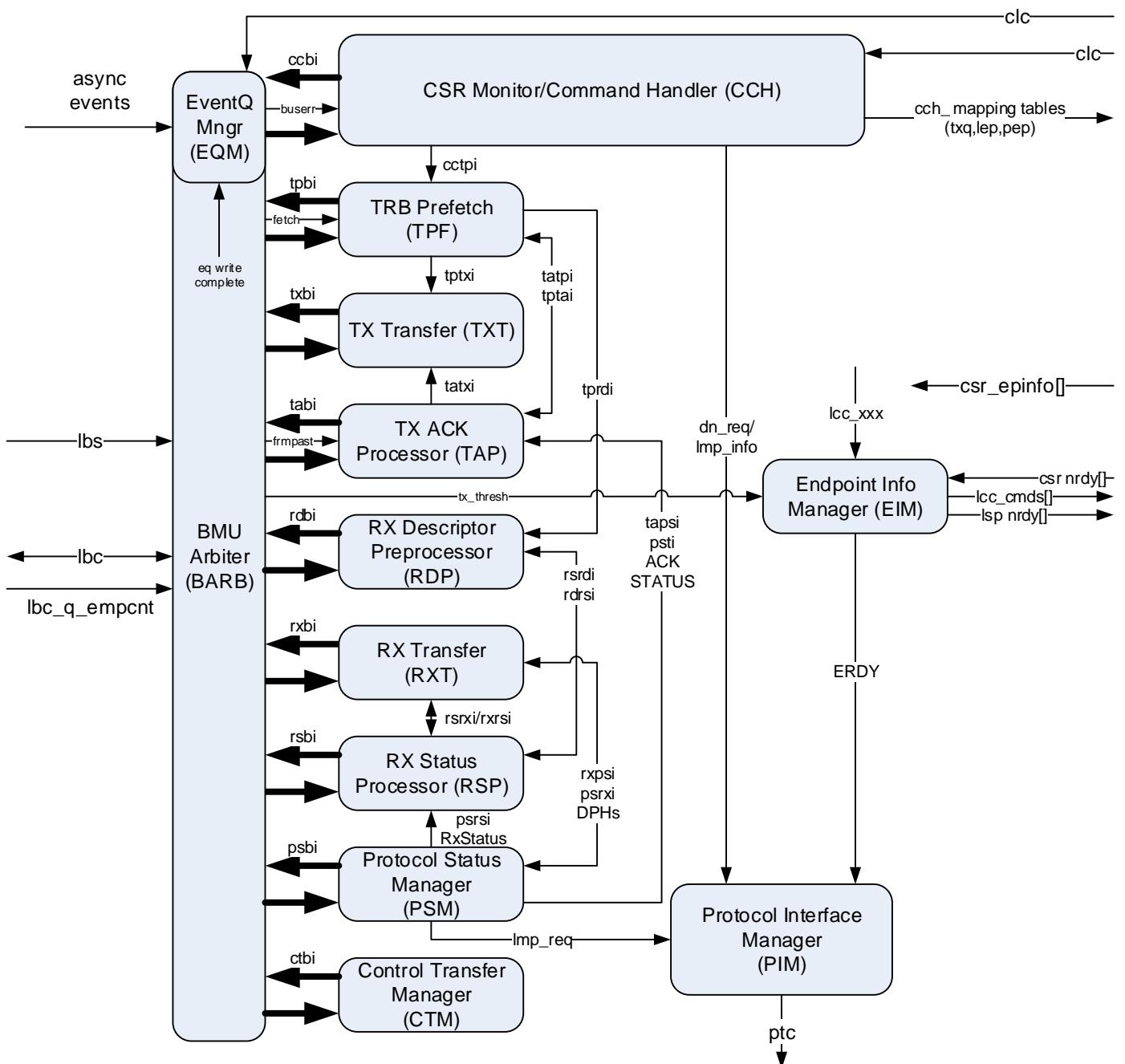


2.16.1 Device List Processor

In Device mode, the List Processor contains 12 modules:

- CSR Monitor/Command Handler (CCH): Handles device and endpoint generic commands.
- TRB Prefetch State Machine (TPF): Issues descriptor fetch operations to the BMU when valid descriptors are present in system memory that need to be cached into the internal RAM.
- TX Transfer (TXT): Issues TX DMA commands to the BMU to begin master reads on the system bus. IN transfers are handled in this module.
- TX ACK Processor (TAP): Handles ACKs returned for transmitted packets by updating the descriptors and issuing events.
- RX Descriptor Preprocessor (RDP): Processes OUT transfer descriptors as they are fetched to determine if there is enough buffer space to receive a packet.
- RX Transfer (RXT): When the packet header is received, manages the receive state machines by issuing RX DMA commands to the BMU and handling their responses. OUT transfers are handled in this module.
- RX Status Processor (RSP): After a packet is received, this module updates descriptors and issues events.
- Protocol Status Manager (PSM): This module receives an indication that activity occurred on the USB bus (packet headers, ACKs, end of packet status, and so on) and reads that activity information from the BMU. It then routes the information to the module that handles the activity information.
- Control Transfer Manager (CTM): This module manages all control endpoints, ensuring that the software interface is consistent, regardless of whether any errors occur during the stages of control transfers.
- BMU Arbiter/EventQ Manager (BARB): Allows arbitration from the above modules to the BMU. Arbitration is done on a cycle-by-cycle basis. In addition, the EventQ Manager in this module generates the correct EventQ offsets when a module requests an EventQ write.
- EP Info Manager (EIM): Maintains endpoint information that is only needed by the LSP, such as flow control state and active stream index. Also triggers an ERDY generation to the PTL block when necessary. For an OUT endpoint, the device sends ERDY if the endpoint is in flow control and at least one packet of the TRB is available for servicing and at least one packet space is available in the receive FIFO, or the application stalls the endpoint. For IN endpoints, the device sends ERDY if the endpoint is in flow control and a threshold amount of data is available in the TxFIFO, or the application stalls the endpoint.
- Protocol Interface Manager (PIM): Receives requests for transmitting LMP's, device notification TPs, and ERDY packets and forwards them to the PTL.

Figure 2-43 on page 149 shows the internal LSP architecture.

Figure 2-43 Device-Mode LSP Block Diagram

2.16.2 Device Functional Description

The device functionality consists of the following features:

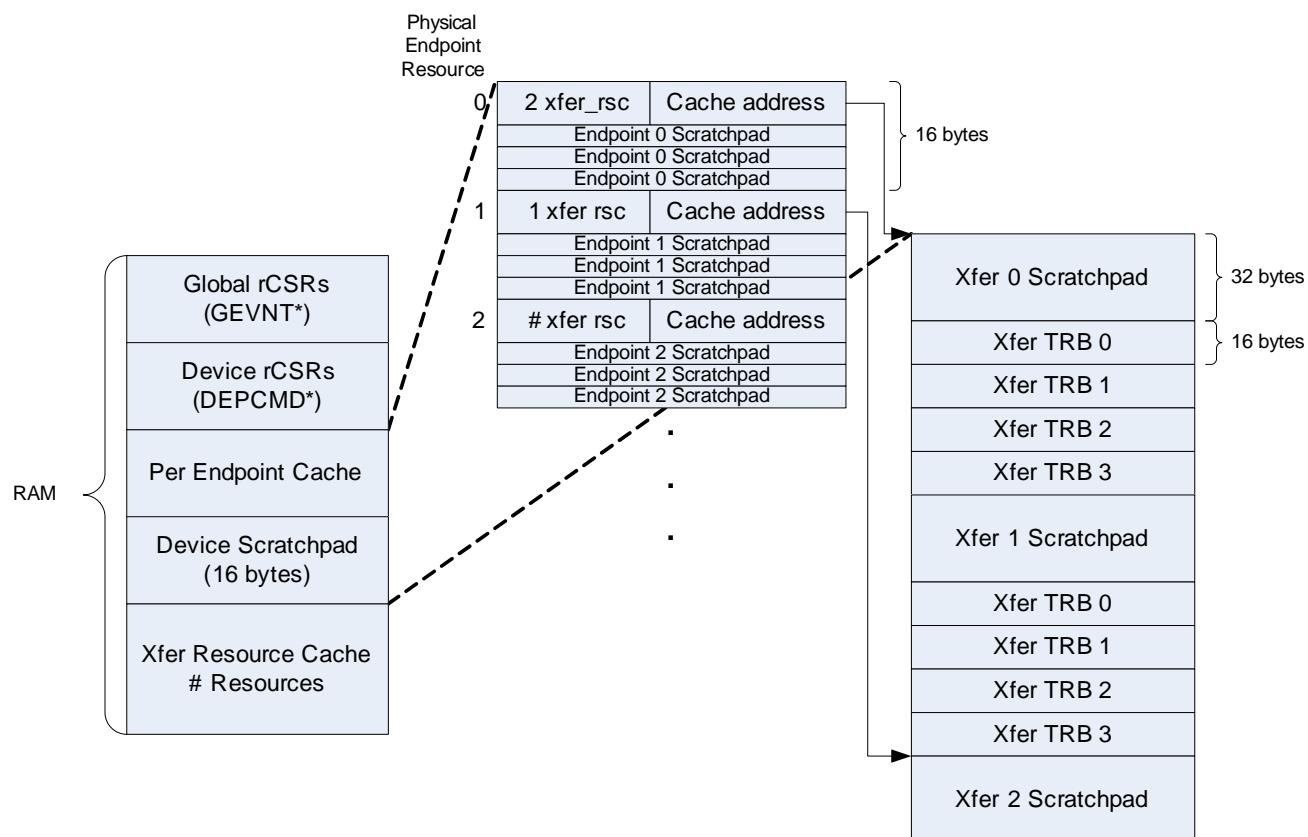
- TRB Management (fetching, caching, interpreting, updating, writing back). This involves primarily interaction with the BMU, and is initiated by the CSR.
- Transfer Management (RX DMA, TX DMA, transfer error handling). This involves interaction with the BMU and PTL.
- CSR and Command Processing. This involves interaction from the CSR and may have PTL/BMU side-effects.
- Interrupt and Event Generation. This involves interaction with the BMU and CSR blocks.

2.16.3 Device Cache Memory Layout

The LSP uses an area of RAM accessible through the BMU for several purposes.

- Device scratchpad: Storage of variables that relate to the device as a whole. There is only one device scratchpad area.
- Endpoint scratchpad: Storage of endpoint configuration parameters, endpoint variables, and a pointer to the transfer resource of the endpoint. There is one endpoint scratchpad area per physical endpoint.
- Transfer scratchpad: Storage of variables, pointers, etc. that relate to a single transfer within an endpoint. There is one transfer scratchpad area per transfer resource, and its address is related to the endpoint look-up table and the transfer index within the endpoint.
- Transfer Resources: When the controller is configured, a fixed number of transfer resources are allocated in the memory, and each transfer resource has a fixed number of TRBs that can be cached for that resource.

Figure 2-44 shows how the cache memory is laid out.

Figure 2-44 Memory Map of Internal RAM that is LSP-Accessible

2.16.4 Host List Processor

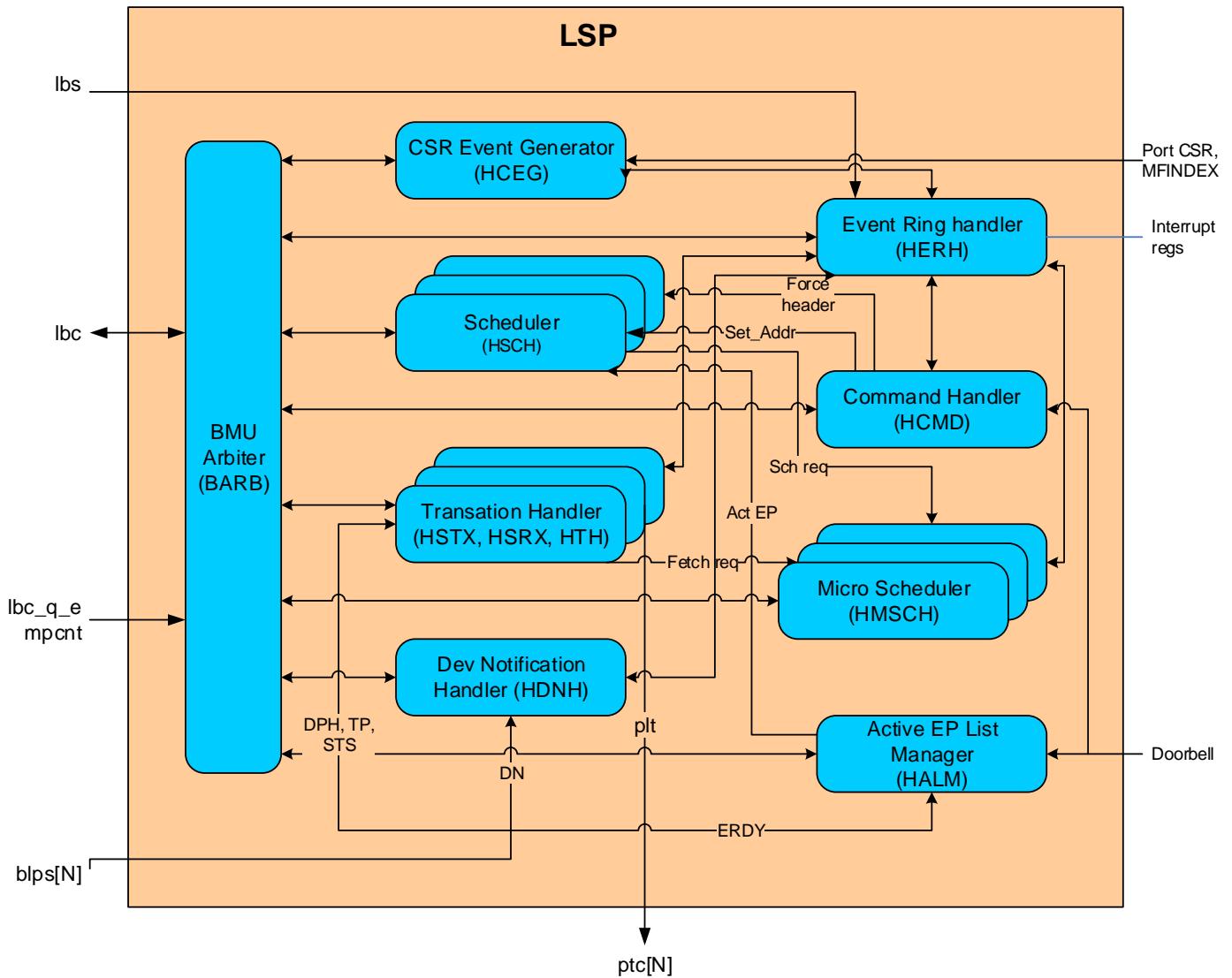
The following sections describe the Host List Processor.

2.16.4.1 Host LSP Overview

The host LSP module performs the following functions.

- Manage XHCI command ring.
 - Monitors doorbell.
 - Reads and reacts to XHCI commands.
 - Generates command complete events.
- Manage XHCI event rings – generates events and interrupts.
- Manage XHCI transfer rings – schedules USB transfers.
 - Monitors doorbell.
 - Maintain active EP lists.
 - Fetch TRBs and Slot/EP context.
 - Schedule asynchronous and periodic traffic based on EP characteristics and bandwidth availability.
 - Handles USB transfers including error cases.
 - Handles TX data fetch and RX data DMA.
 - Generates transfer related events.
- Handle device notifications and generate events.
- Monitor port CSR changes and generate events

[Figure 2-45](#) on page [153](#) shows a host LSP block diagram.

Figure 2-45 Host LSP block diagram

The Host LSP consists of the following modules.

- Command Handler (HCMD)
 - Manages the command ring.
 - Assists other modules in handling all xHCI commands.
- Event Ring Handler (HERH)
 - Manages the event ring.
 - Provides event address for other modules.
- CSR Event Generator (HCEG)
 - Monitors port CSR changes and generates events.

- Scheduler (HSCH)
 - Schedules transaction for active endpoints.
 - Manages endpoint cache.
- Micro-Scheduler (HMSCH)
 - Determines the number of packets to schedule for a given endpoint for each service opportunity.
 - Manages TRB cache.
- SS TX Transaction Handler (HSTX)
 - Handles super speed OUT transactions.
 - Issues TX command to the lower layer, receives and processes response and status.
 - Generates transfer event and updates internal transfer states.
- SS RX Transaction Handler (HSRX)
 - Handles super speed IN transactions.
 - Issues RX command to the lower layer, receives and processes response and status.
 - Generates transfer event and updates internal transfer states.
- HS/FS/LS Transaction Handler (HTH)
 - Handles high speed, full speed and low speed transactions.
 - Issues command to the lower layer, receives and processes response and status.
 - Generates transfer event and updates internal transfer states.
- Active EP List Manager (HALM)
 - Maintains active endpoint lists used by the schedulers to schedule endpoints.
 - High-level periodic endpoint scheduling when periodic endpoints are added.
- Device Notification Handler (HDNH)
 - Receives device notifications.
 - Updates internal variables and generates corresponding events.
- BMU Arbiter (BARB)
 - Arbitrates BMU RAM access requests from LSP modules.

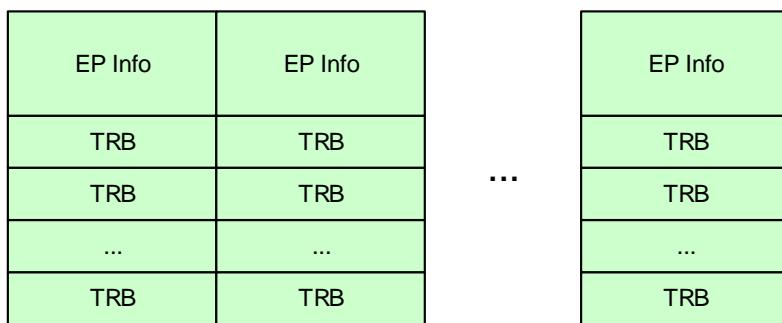
2.16.4.2 Host Endpoint and TRB Caching

The host controller caches slot information, endpoint information and TRBs.

The number of endpoint caches for a USB instance is configurable. Three coreConsultant parameters specify the number of endpoint caches for each SS, HS or FS/LS USB instance, respectively. Each endpoint cache consists of two parts: EP information part (stores Slot and EP context and internal intermediate information) and TRB part (stores cached TRBs). The number of TRB caches for each endpoint cache is configurable. Three coreConsultant parameters specify the number of TRB caches for each cached SS, HS or FS/LS endpoint, respectively.

[Figure 2-46](#) shows the endpoint and TRB cache structure for one USB instance. The same data structure is duplicated multiple times depending on the number of USB instances.

Figure 2-46 Host Endpoint and TRB Cache Structure



When scheduling transactions for an endpoint, the scheduler checks if that endpoint is cached. If the endpoint is not cached, the scheduler checks for available EP cache. If all EP caches are occupied, the scheduler selects an endpoint that the transaction handler is not currently working on and removes that endpoint. When the scheduler finds an available EP cache, it issues fetch commands to BMU to obtain slot context, EP context and build the EP Information part of the EP cache.

After the EP Information part is built by the scheduler, it issues a command to the micro-scheduler. The micro-scheduler checks if for cached TRBs for that endpoint. If there are no cached TRBs, the micro-scheduler issues fetch commands to BMU to fetch TRBs. If there are cached TRBs, the micro-scheduler processes the TRBs and determines the number of packets to schedule for this service opportunity. If it runs out of cached TRBs during scheduling, the micro-scheduler issues commands to fetch more TRBs. The micro-scheduler also prefetches TRBs for cached endpoints if it is not working on any scheduler requests.

The endpoint is removed from cache if it is flow controlled, stopped or stalled. This ensures endpoint caches are not occupied by inactive endpoints. The cached TRB is removed from cache after the corresponding transactions are finished or when the endpoint cache is removed.

2.16.4.3 Asynchronous Transfer Scheduling

The host controller maintains an asynchronous endpoint list that keeps track of the state of each endpoint and indicates when an endpoint has available data buffer and if an endpoint is flow controlled. The scheduler uses the endpoint state information to decide whether to schedule traffic to that endpoint. The scheduler uses a round-robin algorithm to schedule asynchronous endpoints. Each asynchronous endpoint is given one service opportunity per pass through the asynchronous schedule.

Each USB instance has a dedicated scheduler. The endpoints belonging to different USB instances are scheduled independently.

2.16.4.4 Periodic Transfer Scheduling

The host controller constructs and maintains a periodic endpoint list that specifies in which frames/micro-frames the endpoint is scheduled, along with if and how the transfer is split into multiple bursts. The scheduler uses information in the periodic endpoint list to decide when to request the micro-schedule into fetch TRBs and determine how many packets to send/receive for each service opportunity. Within a frame/micro-frame, periodic endpoints are scheduled before any asynchronous endpoints.

The remainder of this section provides details on constructing and updating the periodic endpoint list:

- The host controller maintains a sorted link list for all the periodic endpoints. The list is updated upon a Configure Endpoint Command.
- When the Configure Endpoint command is to add a new periodic EP, the host controller checks the resource and bandwidth availability.
 - If the required resource and bandwidth are available, the host:
 - Schedules the EP and creates a link list entry.
 - Compares the Interval value of the new EP with the Interval value of the existing periodic EP to find the right position for insertion (The link list is sorted based on the Interval).
 - Inserts the entry into the link list.
 - Figures out frame/micro frame number to start the transaction for the EP.
 - Figures out how the Max ESIT Payload is split across bursts and microframes.
 - Subtracts resource and bandwidth from the Bandwidth Available and Resource Available variables.
 - If the required resource or bandwidth is not available, the host controller rejects the EP configure command.
- When the Configure Endpoint command is to remove an existing periodic endpoint, the host controller performs the following:
 - a. Removes the EP from the linked list.
 - b. Adds the released resource and bandwidth back to the Bandwidth Available and Resource Available variables.
- When the host controller schedules a periodic EP, it considers Multi, Max Burst Size, Max packet size, Max ESIT Payload, and Max Exit Latency of the EP. The host controller maintains the following internal variables that help the scheduling. When an EP is added or removed, these variables are updated.
 - Resource Available – Indicates if the linked list is full.
 - Bandwidth Available – Indicates the amount of bandwidth available for the periodic EPs.

Microframe utilization table – Each individual uF bandwidth usage need to be recorded because the used bandwidth is not evenly distributed among all different micro frames. The host controller ensures that no uF exceeds 100% usage.

2.16.5 Debug Capability List Processor

The Debug Capability List Processor performs the following functions:

- Manages all control endpoint (EP0) transfers
 - Responds automatically to SetAddress, SetConfig, GetDescriptor, GetStatus, SetFeature, and all the other device requests (as described in Section 9 of the USB 3.0 Specification) from the Debug Host
 - Responds with STALL to unsupported device requests and requests received in the wrong device state
 - Maintains the device state as described in Section 9.1 of the USB 3.0 Specification
- Manages DbC Event Ring
 - Generates Port Status Change events for the debug port as well as transfer events for the two endpoints
- Manages DbC Transfer Rings
 - The OUT Transfer Ring corresponds to the IN endpoint, and the IN Transfer Ring corresponds to the OUT endpoint
 - Monitors doorbells
 - Fetches TRBs and Endpoint Contexts
 - Pre-fetches IN endpoint data to respond quickly to the debug host
 - Handles USB transfers with Mastered DMA to and from system memory
 - Handles software-controlled endpoint halt requests

The DbC LSP contains various modules. The following subsections describe each module.

Host Debug Protocol Status Manager (hdpsm)

This module performs the following operations:

- Reads the tokens from the PSQ which contains information about the transactions that happened on the USB, and delivers the token to the appropriate module (hdep0, hdrx, or hdtx)
- Synchronizes an asynchronous event with the PSQ so that all the tokens in the queue prior to the asynchronous event can be read before the asynchronous event is handled

Host Debug Endpoint 0 Manager (hdep0)

This module performs the following operations:

- Handles all control transfers on the EP0, and maintains the Debug Capability device state
- Communicates with the rest of the controller when the host requests a SetFeature or ClearFeature

Host Debug TRB Prefetch Manager (hdtpf)

This module performs the following operations:

- Generates DMA read requests to fetch TRBs for both endpoints into the internal cache
- Tracks the Cycle bit to detect valid TRBs, and automatically re-starts TRB fetching when the corresponding doorbell is rung by the software
- Reports the validity of TRBs in the internal cache to the hdtx and hdrx modules

Host Debug Transmit State Machine (hdtx)

This module handles all IN endpoint (OUT Transfer Ring) transfers. It performs the following operations:

- Generates DMA read requests to prefetch TX data into the TX FIFO
- Handles ACKs from the Debug Host
- Handles re-fetching TX data on retry
- Handles the HOT halt request from the software
- Updates the IN endpoint state and the TR Dequeue Pointer in the Endpoint Context
- Generates transfer events for the completed TRBs

Host Debug Receive State Machine (hdrx)

This module handles all OUT endpoint (IN Transfer Ring) transfers. It performs the following operations:

- Prepares the dbc_csr_epinfo module to receive packets from the Debug Host
- Generates DMA write requests to move the payload from the USB to the system memory
- Flushes the RX packets when they have bad CRC
- Handles the HIT halt request from the software
- Updates the OUT endpoint state and the TR Dequeue Pointer in the Endpoint Context
- Generates transfer events for the completed TRBs

Host Debug Event Ring Handler (herh)

This module handles the Debug Capability Event Ring. It performs the following operations:

- Receives Event Ring requests from the hdtx, hdrx, and hceg modules, and calculates the correct external memory address for the next event.
- Monitors the DCERDP writes from the software to track the number of unacknowledged events in the Event Ring.
- Maintains the DCST.ER bit to communicate to the software when the Event Ring is not empty

Host Debug Port Change Event Generator (hceg)

This module performs the following operations:

- Monitors the DCPORTSC register, and generates a Port Status Change event when necessary
- Communicates with the herh module to retrieve the correct external memory address for the next event

Host Debug Endpoint Information Manager (hdeim)

This module performs the following operations:

- Tracks the flow control state of EP0 IN/OUT and EP1 IN/OUT on the USB and on the system to decide when to transmit an ERDY to the Debug Host
- Tracks the activity of the endpoints, and reports to the lower layer when U1/U2 entry is allowed
- Acts as an arbitrator for the LSP to CSR requests

Memory Requirements

This chapter describes the memory requirements for the DWC_usb3 controller, as detailed in the following topics.

- “[Synchronous Static RAM Requirement](#)” on page [160](#)
- “[Total Device RAM Requirement](#)” on page [162](#)
- “[Total Host RAM Requirement](#)” on page [169](#)
- “[Packet Threshold and Burst Features for High Latency Systems](#)” on page [171](#)
- “[Total DRD RAM Requirement](#)” on page [176](#)
- “[Rx/Tx FIFO Configuration](#)” on page [177](#)
- “[System Bus and RAM Bandwidth Requirements](#)” on page [182](#)
- “[Additional RAM Requirements for xHCI Debug Capability](#)” on page [184](#)
- “[RAM Requirements for USB 2.0-Only Mode](#)” on page [187](#)

3.1 Synchronous Static RAM Requirement

The DWC_usb3 supports one, two, or three synchronous static RAMs, which contain the cache, TxFIFOs, and RxFIFO(s). The RAMs can be single port or two-port. A two-port RAM means that Port1 is read only while Port2 is write only, and each port uses the same clock. You can use a memory compiler to build the RAMs or use pre-existing memory from your ASIC vendor library. The RAMs must be standard synchronous static RAMs. During the write cycle, the write address and write-data are valid on the same clock. During the read cycle, the read data is valid one clock after the address is valid. The address and write-data from the controller arrives late (20% setup) to meet the RAM setup requirement. The read data delay from the RAM should be around 50% of the clock period since there are a few levels of logic inside the controller before being registered.

The following diagrams show the RAM write and read cycles for single- and two-port RAMs.

Figure 3-1 Single-Port RAM Interface (when DWC_USB3_SPRAM_TYP=1)

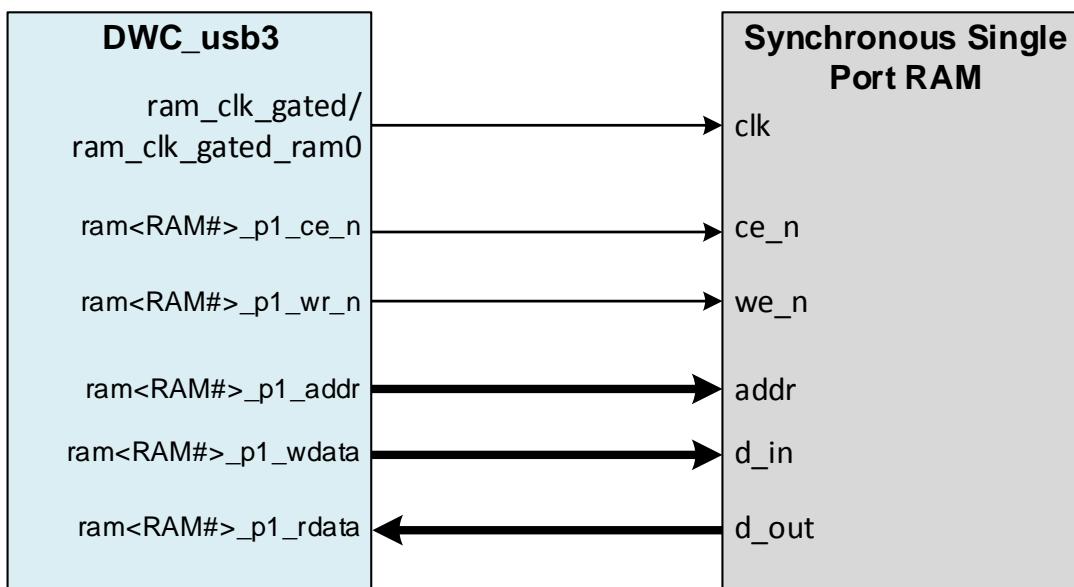
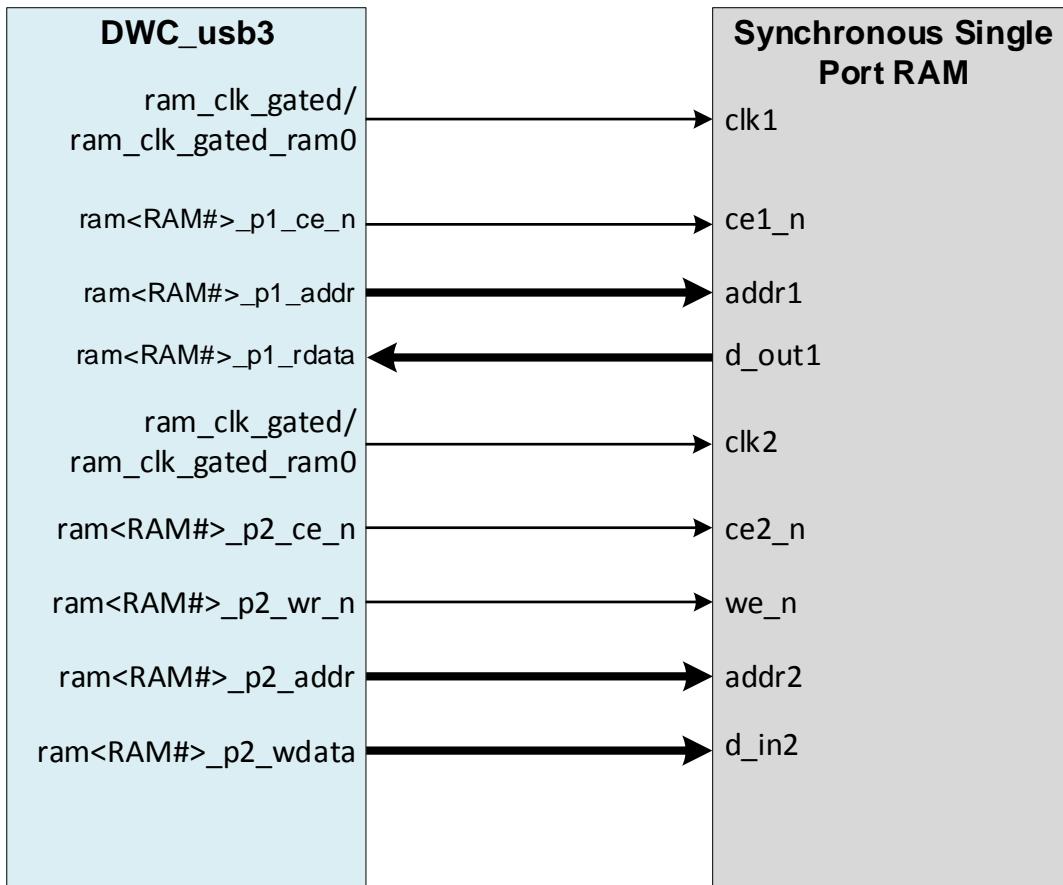
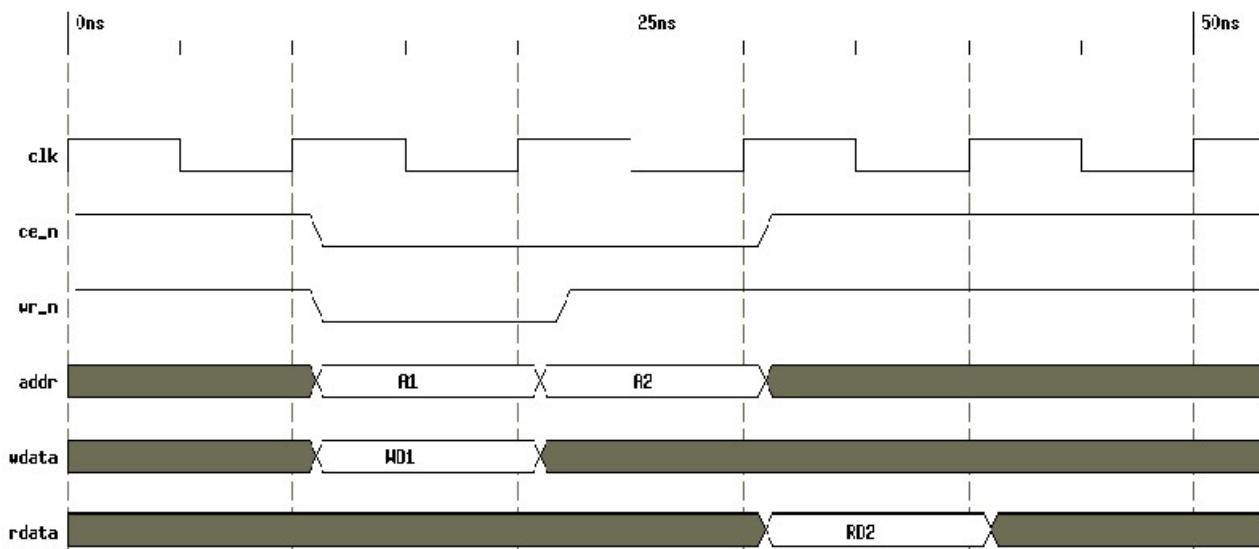


Figure 3-2 Two-Port RAM Interface (when DWC_USB3_SPRAM_TYP=0)

The signal `ram_clk_gated` is connected to RAM1 and RAM2 clock, and `ram_clk_gated_ram0` is connected to RAM0 clock. During U1/U2, the `ram_clk_gated_ram0` will be active once in a while to read the cache information

Figure 3-3 RAM Interface Timing

3.2 Total Device RAM Requirement

The device controller requires 1, 2, or 3 RAMs to store the following:

- Cache
 - Internal Queues - Refer to `src/DWC_usb3_params.v` for details.
 - RAM Registers - Refer to `src/DWC_usb3_params.v` for details.
 - Device TRB Cache - Determined based on `coreConsultant` parameters.
- TxFIFOs (one per IN endpoint)
- RxFIFO (shared among all OUT endpoints)

The total default size of the cache (internal queues, RAM registers, device TRB cache) is automatically derived based on your endpoint configuration. It is shown as the “RAM0 Cache Depth” (`DWC_USB3_DCACHE_DEPTH_INFO`) parameter in `coreConsultant`’s “Advanced Parameters” section. This default value must not be reduced, but may be increased by 256 bytes to provide some margin for future controller designs.

The total default size of the TxFIFOs (RAM1) is automatically (and pessimistically) calculated as if control EP0 must buffer a 512 byte packet and all other IN endpoints are 1024 byte packet burst-capable (each requires a 3 packet buffer). This default value can be changed depending on your requirements.

The total default size of the RxFIFO (RAM2, or RAM0 if `NUM_RAMs < 3`) is automatically derived based on buffering three 1024 byte packets and up to three setup packets. This default value can be changed depending on your requirements.

The default device RAM0 allocation is shown in [Figure 3-4 on page 163](#). Note that this figure is only an illustration and does not show the exact calculation for RAM0 depth.

Figure 3-4 Default Device RAM0 Allocation

RAM Addr	Byte Addr	Contents	Size
000h:	0000h:	txq[0]	128B
010h:	0080h:	txq[1]	128B
020h:	0100h:	txq[2]	128B
030h:	0180h:	txq[3]	128B
040h:	0200h:	rxq[0]	128B
050h:	0280h:	riq[0]	272B
072h:	0390h:	dfq	128B
082h:	0410h:	dwq	256B
0A2h:	0510h:	psq	256B
0C2h:	0610h:	rCSR_global	16B
0C4h:	0620h:	rCSR_device	128B
0D4h:	06A0h:	cache_ep	128B
0E4h:	0720h:	cache_device	128B
0F4h:	0730h:	cache_streams	3,360B
■ ■ ■			
28Ah:	1450h:	rxT[0] (if NUM_RAMs<3)	3,112B
■ ■ ■			

The RAM queue depths are fixed and cannot be changed by the user. The parameters and their fixed values are as follows:

Table 3-1 RAM Queue Sizes

RAM queue	Parameter	Size in MDWIDTH-bytes	
		MDWIDTH = 32	MDWIDTH = 64/128
BMU Tx Request Queue Depth	DWC_USB3_TXQ_FIFO_DEPTH	16	8
BMU Rx Request Queue Depth	DWC_USB3_RXQ_FIFO_DEPTH	16	8
Rx Info Queue Depth	DWC_USB3_RIQ_FIFO_DEPTH	34	26
BMU Descriptor Fetch Request Queue Depth	DWC_USB3_DFQ_FIFO_DEPTH	16	16
BMU Protocol Status Queue Depth	DWC_USB3_PSQ_FIFO_DEPTH	32	32
BMU Descriptor Write Queue	DWC_USB3_DWQ_FIFO_DEPTH	32	32

The RAM-based CSRs and descriptor cache size depend on the values specified by the user for certain parameters as per the following table:

Table 3-2 RAM CSRs and Descriptor Cache Sizes

CSRs/Descriptor Cache Parameters	Assigned Size
DWC_USB3_RCSR_GLOBAL_DEPTH	16 * `DWC_USB3_DEVICE_NUM_INT
DWC_USB3_RCSR_DEVICE_DEPTH	16 * `DWC_USB3_NUM_EPS
DWC_USB3_RCSR_HOST_DEPTH	(12*4 + 32*`DWC_USB3_HOST_NUM_INTERRUPTER_SUPT + 4 * `DWC_USB3_NUM_DEVICE_SUPT)
DWC_USB3_CACHE_EP_DEPTH	16 * `DWC_USB3_NUM_EPS
DWC_USB3_CACHE_DEVICE_DEPTH	16
DWC_USB3_CACHE_STREAMS_DEPTH	(`DWC_USB3_CACHE_TOTAL_XFER_RESOURCES * ((16 * `DWC_USB3_CACHE_TRBS_PER_TRANSFER) + 32))

Here are the formulas for the default RAM sizes:

- RAM0 (Cache RAM) = RAM0 Cache Depth
- RAM1 (Tx RAM) = ((512 + 2*MDWIDTH-Bytes) (Control IN Endpoint) + (Number of IN Endpoints-1) * (3* (1024+ MDWIDTH-Bytes) + MDWIDTH-Bytes) (TxFIFOs))/MDWIDTH-Bytes
- RAM2 (Rx RAM) = (3 * 1024)/MDWIDTH-Bytes (RxFIFO) + (24 bytes (for setup) + 24 bytes (if MDWIDTH=128, since SETUP is not packed) + 16bytes (clock domain crossing tolerance))/MDWIDTH-Bytes

You may reduce the number of packets to buffer per TxFIFO if one or more of your IN endpoints (except for control EP0) does not require burst support of 1024 byte packets, area is limited, and bus latency is low; this reduces the required RAM1 depth. In order for the controller to buffer a packet in a TxFIFO, there must be space for the 1 FIFO-word TXINFO header. The TxFIFO itself also needs a 1 FIFO-word margin to simplify byte-aligned logic calculations.

You may increase the number of packets to buffer per TxFIFO or RxFIFO if your system bus latency is high; this has the effect of increasing the required RAM1 and RAM2 depths as illustrated in the above RAM size equations.



Do not reduce the Cache RAM depth.

The controller does not currently support USB and System Bus sub-packet threshold mode. Therefore, each Tx endpoint that supports bursting requires a minimum TXFIFO depth of three packets.



MDWIDTH-Byte is as follows:

- 4 for 32-bit master-bus
- 8 for 64-bit master bus
- 16 for 128-bit master bus mode

3.2.1 Calculating RAM Size for Example Configurations

Example Configuration 1:

- 32-bit AHB Master
- Control IN and OUT Endpoint (1 IN + 1 OUT)
- Bulk IN and OUT Endpoint For Mass Storage - Supports Burst (1 IN + 1 OUT)
- Bulk IN and Bulk OUT Endpoint for UASP Command and Status (1 IN + 1 OUT)
- ISOC IN and OUT Endpoint - Supports Burst (1 IN + 1 OUT)
- MICE Interrupt EP (1 IN)

==> Number of IN = 5

==> Number of USB Endpoints = 9 (or 10, twice the maximum of all IN or OUT endpoints, if you are not using the re-map feature and doing 1:1 mapping of USB endpoints to Physical endpoints)

Rx RAM Size in Bytes = $3 * 1024 + 24 + 16 = 3112$ bytes

==> Rx FIFO Size Depth = $3112 / 4 = 778$ // Round up the value

Tx FIFO Size in Bytes =

```
( 1 * (512 + 4) + 4 ) + // For control IN endpoint
( 3 * (1024 + 4) + 4 ) // For Bulk IN Burst endpoint
( 2 * (64 + 4) + 4 ) // For UASP Status endpoint
( 3 * (1024 + 4) + 4 ) + // For ISOC IN Burst endpoint
( 2 * (4 + 4) + 4 ) // INTERRUPT IN endpoint
= 6856 bytes
```

==> Tx RAM Depth = $6856 / 4 = 1714$ // Round up the value

Example Configuration 2:

- 64-bit AHB Master
- Control IN and OUT Endpoint (1 IN + 1 OUT)
- Bulk IN and OUT Endpoint For Mass Storage - Supports Burst (1 IN + 1 OUT)
- Bulk IN and Bulk OUT Endpoint for UASP Command and Status (1 IN + 1 OUT)
- ISOC IN and OUT Endpoint - Supports Burst (1 IN + 1 OUT)
- MICE Interrupt EP (1 IN)

\Rightarrow Number of IN = 5

\Rightarrow Number of USB Endpoints = 9 (or 10, twice the maximum of all IN or OUT endpoints, if you are not using the re-map feature and doing 1:1 mapping of USB endpoints to Physical endpoints)

Rx RAM Size in Bytes = $3 * 1024 + 24 + 16 = 3112$ bytes

\Rightarrow Rx FIFO Size Depth = $3112/8 = 389$ // Round up the value

Tx FIFO Size in Bytes =

$(1 * (512 + 8) + 8) +$ // For control IN endpoint

$(3 * (1024 + 8) + 8) +$ // For Bulk IN Burst endpoint

$(2 * (64 + 8) + 8) +$ // For UASP Status endpoint

$(3 * (1024 + 8) + 8) +$ // For ISOC IN Burst endpoint

$(2 * (8 + 8) + 8) +$ // INTERRUPT IN endpoint (If the smallest packet size is less than MDWIDTH-Byte size round it to MDWIDTH-Byte size)

= 6928 bytes

\Rightarrow Tx RAM Depth = $6928/8 = 866$ // Round up the value

Example Configuration 3:

- 128-bit AHB Master
- Control IN and OUT Endpoint (1 IN + 1 OUT)
- Bulk IN and OUT Endpoint For Mass Storage - Supports Burst (1 IN + 1 OUT)
- Bulk IN and Bulk OUT Endpoint for UASP Command and Status (1 IN + 1 OUT)
- ISOC IN and OUT Endpoint - Supports Burst (1 IN + 1 OUT)
- MICE Interrupt EP (1 IN)

\Rightarrow Number of IN = 5

\Rightarrow Number of USB Endpoint = 9 (or 10, twice the maximum of all IN or OUT endpoints, if you are not using the re-map feature and doing 1:1 mapping of USB endpoints to Physical endpoints)

Rx RAM Size in Bytes = $3 * 1024 + 24 + 24 + 16 = 3136$ bytes

\Rightarrow Rx FIFO Size Depth = $3136/16 = 196$ // Round up the value

Tx FIFO Size in Bytes =

$(1 * (512 + 16) + 16) +$ // For control IN endpoint

```

(3 * (1024 + 16) + 16) // Bulk IN Burst Endpoint
(2 * (64 + 16) + 16) // UASP Status Endpoint
(3 * (1024 + 16) + 16) + // ISOC IN Burst Endpoint
(2 * (16 + 16) + 16) // INTERRUPT IN endpoint (If the smallest packet size is less
than MDWIDTH-Byte size round it to MDWIDTH-Byte size)
= 7072 bytes
==> Tx RAM Depth = 7072/16 = 442 // Round up the value

```

3.2.2 Device RxFIFO Data Allocation

Device OUT endpoints share a single RXFIFO.

In USB 3.0, a minimum of 3 times the MaxPacketSize RxFIFO buffer size is recommended plus space to store three setup packets (8 bytes each) and a clock crossing margin of 16 bytes. Because the USB 3.0 maximum packet size is 1024 bytes, the main data payload portion of the RXFIFO needs to be 3072 bytes.

In USB 2.0, if your application has only bulk endpoints, then 3 * 512 byte RxFIFO (+ setup packet and clock margin) is adequate since USB 2.0 non-isochronous endpoints have a maximum packet size of 512 bytes. If your USB 2.0 application supports 1024-byte isochronous endpoints, however, then at least 3 * 1024 byte RxFIFO (+ setup packet and clock margin) is recommended.

If your system memory access latency is large or unpredictable, then you must have a large buffer to meet performance requirements. When interfacing to a PCIe-like bus, a minimum of 3X packet TxFIFO (+ TXINFO and byte alignment margin) is recommended to meet the USB 3.0 burst requirement of setting the EOB flag when the controller cannot guarantee transmission of the next packet.

3.2.3 Device TxFIFO Data Allocation

Device IN endpoints each have their own TXFIFO.

In USB 2.0, a 2X MaxPacketSize TXFIFO buffer size is recommended. The reason that a one packet buffer is insufficient is that the controller must be able to re-transmit the packet from the TXFIFO in the event of an error. By having space to store a second packet, the controller may fetch the second packet while waiting for host acknowledgement of the first packet. In an 8 Tx endpoint configuration, the total TXFIFO size is 8 KB (HS MaxPacketSize is 512 bytes).

If your system memory access latency is large or unpredictable, then you must have a large buffer to meet performance requirements. When interfacing to a PCIe-like bus, a minimum 3X packet TxFIFO size is recommended for an endpoint that supports burst. If a Tx endpoint does not support burst or transfers only a small amount of data per microframe, then a smaller FIFO size is okay.

3.2.4 Number of Transfer Resources in Device

A transfer resource is a set of registers and cache memory needed to setup a transfer. The cost of a transfer resource is (Number of cached TRBs * 16 + 32) bytes of cache memory. There is one transfer resource allocated per endpoint.

3.2.5 Number of Cached TRBs per Transfer in Device Mode

The TRBs are prefetched and stored inside the controller to improve performance. Depending on your application and page size, choose the TRB cache size to maximize burst throughput and ensure correct controller operation:

- The endpoint cache must hold one burst amount of TRBs. For example, if you are doing 16 packet bursts and if your page size is 4 KB, then the recommended number of TRBs in the TRB cache is 4 ($4 * 4\text{KB} = 16$, 1KB packets). Each TRB is 16 bytes (four 32-bit DWORDs). Therefore, you would require 96 bytes ($4 * 16 + 32$) of cache memory per endpoint transfer cache.
- If your application buffer size is less than the packet size, then the TRB cache must have at least one packet amount of TRBs for correct operation. For example, if your Tx buffers are 64-bytes, the TRB cache must hold at least 16 TRBs ($1024 / 64 = 16$).

3.3 Total Host RAM Requirement

The host controller requires 1, 2, or 3 RAMs to store the following:

- Cache
 - Internal Queues – Refer to *src/DWC_usb3_params.v* for details.
 - RAM Registers – Refer to *src/DWC_usb3_params.v* for details.
 - Host Cache – Determined based on coreConsultant parameters.
- TxFIFOs (one per bus instance, minimum of three due to SS, HS, and FS/LS Bus-instance)
- RxFIFOs (one per bus instance, minimum of three due to SS, HS, and FS/LS Bus-instance)

The total default size of the cache (internal queues, RAM registers, device TRB cache) is automatically derived based on your bus instance configuration. It is shown as the “RAM0 Cache Depth” (DWC_USB3_DCACHE_DEPTH_INFO) parameter in coreConsultant’s “Advanced Parameters” section. This default value must not be reduced, but may be increased by 256 bytes to provide some margin for future controller designs.

The total default size of the TxFIFOs (RAM1) is automatically (and pessimistically) calculated as if all non-FS/LS bus instances are 1024 byte packet burst-capable (SS bus instances require a four-packet buffer, while HS bus instances require a two-packet buffer). This default value can be changed depending on your requirements.

The total default size of the RxFIFOs (RAM2, or RAM0 if NUM_RAM<3) is automatically derived based on buffering three, two, or one 1024 byte packets for SS, HS, and FS/LS bus instances, respectively. This default value can be changed depending on your requirements.

The formulas for the default RAM sizes (in terms of MDWIDTH-Bytes) are as follows:

- RAM0 (Cache RAM) = RAM0 Cache Depth
- RAM1 (Tx RAM) = $((4 * (1024 + \text{MDWIDTH-Bytes}) + \text{MDWIDTH-Bytes}) (\text{SS Tx FIFOs}) + (2 * (1024 + \text{MDWIDTH-Bytes}) + \text{MDWIDTH-Bytes}) (\text{HS Tx FIFOs}) + (1 * (1024 + \text{MDWIDTH-Bytes}) + \text{MDWIDTH-Bytes}) (\text{FS/LS Tx FIFOs})) / \text{MDWIDTH-Bytes}$
- RAM2 (Rx RAM) = $((3 * 1024 (\text{SS Rx FIFO}) + 2 * 1024 (\text{HS Rx FIFO}) + 1024 (\text{FS/LS Rx FIFO})) / \text{MDWIDTH-Bytes}) + ((32 / \text{MDWIDTH-Bytes}) * (\text{HS Bus Instance} + \text{FS/LS Bus Instance}))$



Note Do not reduce the Cache RAM depth.

In the two-RAM configuration, Cache RAM and Rx RAM are combined into RAM0. If you have more than one bus instance, then each bus instance has its own cache allocation, RxFIFO, and Tx FIFO.



MDWIDTH-Byte is as follows:

- 4, for 32-bit master bus
- 8, for 64-bit master bus
- 16, for 128-bit master bus mode.

Examples

Table 3-3 shows a few configuration examples:

Table 3-3 RAM CSRs and Descriptor Cache Sizes

Config Example	RAM Size
1-Port, AHB/AXI SoC with latency in < uS range, 32-bit master data-bus	Rx RAM Size in Bytes = $(3 * 1024 \text{ (SS RxFIFO)} + 2 * 1024 \text{ (HS RxFIFO)} + 1024 \text{ (FS/LS RxFIFO)} + 8 * (2+1)) = 6134 \text{ bytes}$ ==> Rx RAM Depth = $6114/4 = 1536$ depth (round up the value) Tx RAM Size in Bytes = $((4 * (1024 + 4) + 4) \text{ (SS Tx FIFOs)} + (2 * (1024 + 4) + 4) \text{ (HS Tx FIFOs)} + (1 * (1024 + 4) + 4) \text{ (FS/LS Tx FIFOs)}) = 6180 \text{ bytes}$ ==> Tx RAM Depth = $6180/4 = 1545$ depth (round up the value)
4-Port, for 4-lane Gen2 PCIe with read latency in ~1.5 range, 128-bit master data-bus	Rx RAM Size in Bytes = $(3 * 1024 \text{ (SS RxFIFO)} + 2 * 1024 \text{ (HS RxFIFO)} + 1024 \text{ (FS/LS RxFIFO)} + 8 * (2+1)) = 6138 \text{ bytes}$ ==> Rx RAM Depth = $6114/16 = 383$ depth (round up the value) Tx RAM Size in Bytes = $((4 * (1024 + 16) + 16) \text{ (SS Tx FIFOs)} + (2 * (1024 + 16) + 16) \text{ (HS Tx FIFOs)} + (1 * (1024 + 16) + 16) \text{ (FS/LS Tx FIFOs)}) = 7328 \text{ bytes}$ => Tx RAM Depth = $7328/16 = 458$ depth (round up the value)
4-Port, for 1-lane Gen2 PCIe with read latency in ~3.2 range, 32-bit master data-bus	Rx RAM Size in Bytes = $(5 * 1024 \text{ (SS RxFIFO)} + 2 * 1024 \text{ (HS RxFIFO)} + 1024 \text{ (FS/LS RxFIFO)} + 8 * (2+1)) = 8216 \text{ bytes}$ ==> Rx RAM Depth = $8192/4 = 2048$ depth (round up the value) Tx RAM Size in Bytes = $((7 * (1024 + 4) + 4) \text{ (SS Tx FIFOs)} + (2 * (1024 + 4) + 4) \text{ (HS Tx FIFOs)} + (1 * (1024 + 4) + 4) \text{ (FS/LS Tx FIFOs)}) = 10292 \text{ bytes}$ ==> Tx RAM Depth = $10292/4 = 2573$ depth (round up the value)
4-Port, 2 SS Bus Instance, for 4-lane Gen2 PCIe with read latency in ~1.5 range, 128-bit data-bus	Rx RAM Size in Bytes = $(3 * 1024 \text{ (SS-1 RxFIFO)} + 3 * 1024 \text{ (SS-2 RxFIFO)} + 2 * 1024 \text{ (HS RxFIFO)} + 1024 \text{ (FS/LS RxFIFO)} + 8 * (2+1)) = 9240 \text{ bytes}$ ==> Rx RAM Depth = $9216/16 = 576$ depth (round up the value) Tx RAM Size in Bytes = $((4 * (1024 + 16) + 16) \text{ (SS-1 Tx FIFOs)} + ((4 * (1024 + 16) + 16) \text{ (SS-2 Tx FIFOs)} + ((2 * 1024 + 16) + 16) \text{ (HS Tx FIFOs)} + (1 * (1024 + 16) + 16) \text{ (FS/LS Tx FIFOs)}) = 11504 \text{ bytes}$ ==> Tx RAM Depth = $11504/16 = 719$ depth (round up the value)

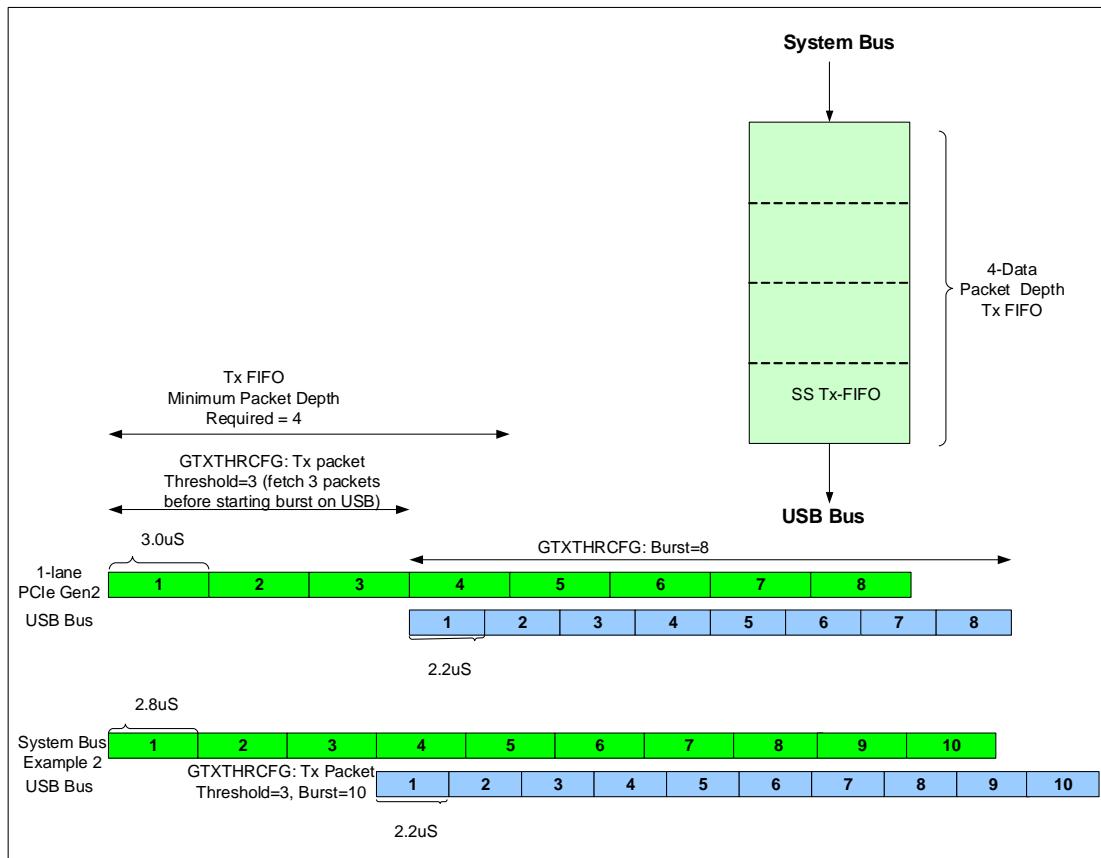
3.4 Packet Threshold and Burst Features for High Latency Systems

In a normal case, a Tx burst starts as soon as one packet is prefetched; an Rx burst starts as soon as 1-packet space is available. This works well as long as the system bus is faster than the USB 3.0 bus (a 1024-bytes packet takes $\sim 2.2\mu s$ on the USB bus in SS mode). If the system bus latency is larger than $2.2\mu s$ to access a 1024-byte packet, then starting a burst on 1-packet condition leads to an early abort of the burst causing unnecessary performance reduction. In order to avoid underrun and overrun during the burst, in a high-latency bus system (like PCIe), threshold and burst size control is provided through GTXTHRCFG and GRXTHRCFG registers. Bit [29] of the GTXTHRCFG and GRXTHRCFG registers enables this feature.

3.4.1 Transmit Path

- GTXTHRCFG.USBTxPktCnt controls the Tx threshold, and GTXTHRCFG.USBMaxTxBurstSize controls the Tx burst size.
- Selecting optimal Tx FIFO size, Tx threshold, and Tx burst size avoids Tx burst aborts due to an underrun if the system bus is slower than USB. Occasionally, an underrun is OK, and there is no functional issue.
- In example-1, the bus access delay for a 1024-byte packet is $3.0\mu s$ and the Tx FIFO size is four packets. A Tx threshold of 3 and a burst size of 8 is optimum.
- In example-2, the bus access delay for a 1024 byte packet is $2.8\mu s$ and the Tx FIFO size is four packets. A Tx threshold of 3 and a burst size of 10 is optimum.
- A larger threshold affects the performance, because the scheduler is idle during this time.

Figure 3-5 Packet Threshold and Burst Size for Transmit Path



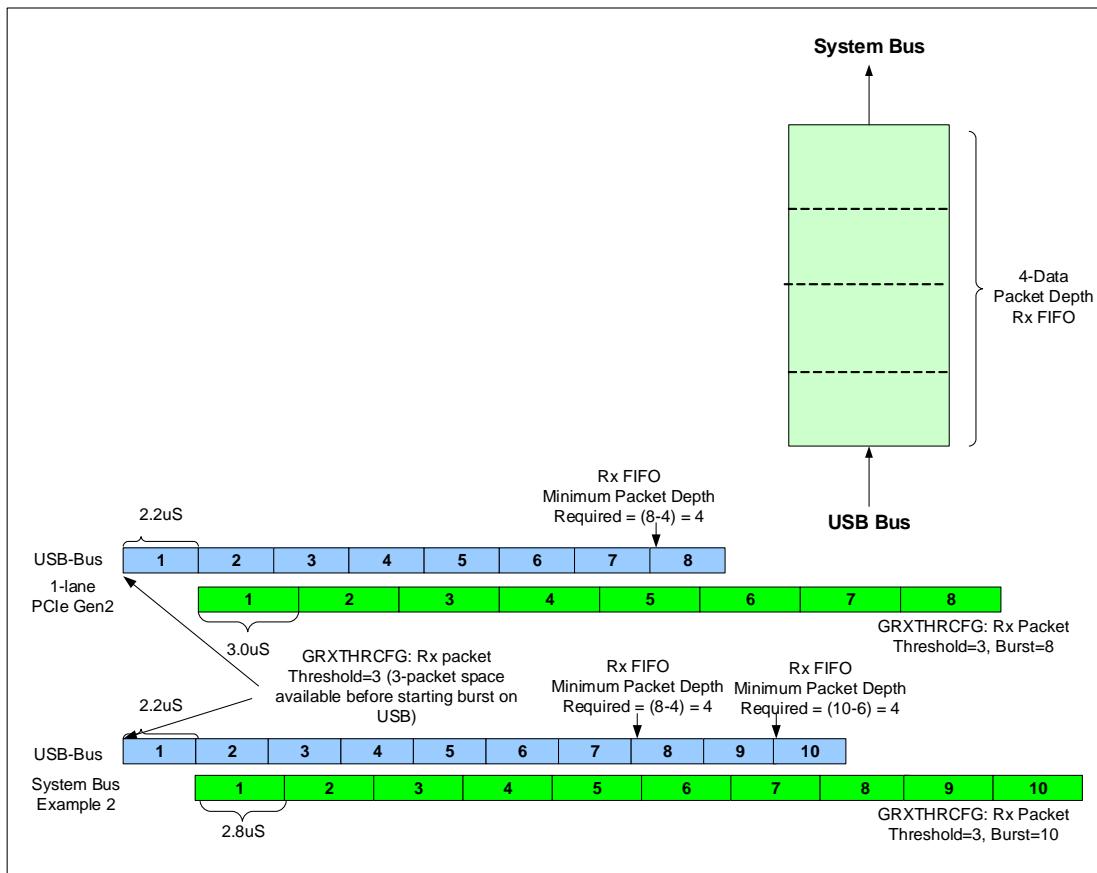
Device-Mode Transmit Path

In Device mode,

- The UsbMaxTxBurstSize is not used.
- The Tx threshold is used for SS bulk and Interrupt IN endpoints.
- The controller tries to honor the Tx threshold whenever possible. Under certain conditions, if the controller does not have the TRBs for the number of packets or if it cannot fetch the TRBs because of high latency or switching between other endpoints, then it does not wait for the threshold number of packets. The threshold number of packets will be honored only when the TRBs are available in the controller for the number of packets before it starts the data fetch.

3.4.2 Receive Path

- The Rx threshold is controlled by `GRXTHRCFG.USBRxPktCnt` and the Rx burst size is controlled by `GRXTHRCFG.USBMaxRxBurstSize`.
- Selecting optimal Rx FIFO size, Rx threshold, and Rx burst size avoids Rx burst aborts due to overrun if the system bus is slower than USB. Once in a while overrun is OK, and there is no functional issue.
- Some devices do not support terminating ACK retry. For these devices, Host cannot set ACK=0 and Retry=0 and do retry later. It must be done immediately. For such devices, minimize retry due to underrun. Setting threshold and burst size guarantees this.
- In example-1, the bus access delay for a 1024-byte packet is 3.0 μ s and the Rx FIFO size is four packets. A Rx threshold of 3 and a burst size of 8 is optimum.
- In example-2, the bus access delay for a 1024-byte packet is 2.8 μ s and the Rx FIFO size is four packets. A Rx threshold of 3 and a burst size of 10 is optimum.
- A larger Rx threshold affects the performance since the scheduler is idle during this time.

Figure 3-6 Packet Threshold and Burst Size for Receive Path

Device-Mode Receive Path

This section discusses the behavior of a USB 3.0 and USB 2.0 device-mode receive path based on the following GRXTHRCFG register field settings:

- **USBRxPktCntSel, USBRxPktCnt and USBMaxRxBurstSize**

In some device systems, if the latency is not predictable, there is a possibility that the device may return NRDY frequently. To optimize the device behavior in such systems, you can use these register fields for managing the reception of packets. The device calculates the NUMP value based on the RX FIFO space available, and returns this value to the host.



- The **USBRxPktCntSel, USBRxPktCnt** and **USBMaxRxBurstSize** register fields are not applicable in USB 2.0 Device mode.
- For Device mode, you can set **GRXTHRCFG.USBRxPktCntSel** to 1 only when you are not using the “on-demand” mode of transfer for any of the SS OUT endpoints. This restriction is because, if the last packet of the transfer ends with an ACK TP (NumP=0), then the USB is in flow control and the host waits for ERDY, but the “on-demand” application does not setup any packet until the host polls, creating a dead-lock situation.

- ResvISOCOUTSpc

When a device is programmed for low-bandwidth ISOC OUT endpoints (for example, audio) along with a large number of high-volume bulk OUT endpoints, the ISOC OUT packets may be dropped because of insufficient RxFIFO space (leading to audio glitches). In such cases, you can use GRXTHRCFG.ResvISOCOUTSpc register field to reserve RX FIFO space for ISOC OUT endpoints to ensure that the ISOC OUT packets are not dropped.

Make note of the following points when you program ResvISOCOUTSpc to a non-zero value:

- a. When ISOC packet(s) is/are already residing in the reserved space and new ISOC packets arrive, the new ISOC packets are written into the remaining space.
- b. Space in RxFIFO for ISOC packets is always reserved from the remaining space. For example: Assume a scenario where the total RxFIFO depth is 6K, and the reserved space for ISOC is 1K. If a 1K ISOC packet arrives, the new RxFIFO space available becomes 5K. Again out of this 5K, 1K is reserved for ISOC packets, and so on.
- c. Space reservation for the ISOC OUT packets is always rounded off to the nearest packet boundary. Therefore, it is always recommended to reserve MPS amount of space. For example, if the reserved space for ISOC OUT is 500 bytes, it will be rounded off to 1024 bytes internally.

USB 3.0 Device-Mode Receive Path

Table 3-4 shows USB 3.0 device-mode receive path behavior based on the GRXTHRCFG register settings.

Table 3-4 USB 3.0 Device Behavior Based on GRXTHRCFG Register Fields

USBRxPktCntSel	ResvISOCOUTSpc	USB 3.0 Device-Mode Receive Path Behavior
0	0	<ul style="list-style-type: none"> ■ The device continues to receive SuperSpeed packets until there is space in the RX FIFO. Whenever a packet is accepted, the NUMP in the ACK TP is the minimum value of (DCFG.NUMP, DEPCFG.BrstSiz). ■ The device returns NRDY if there is no space in the RX FIFO. After returning NRDY, the endpoint enters flow control. ■ When at least one packet space becomes available, the device returns ERDY and the NUMP indicated in the ERDY is the minimum value of (DCFG.NUMP, DEPCFG.BrstSiz). <p>Note: GRXTHRCFG.USBRxPktCnt, and GRXTHRCFG.USBMaxRxBurstSize must be set to zero in this mode.</p>
0	Non-zero	<p>For non-ISOC OUT EPs,</p> <ul style="list-style-type: none"> ■ The device continues to receive SuperSpeed packets until there is space in the RX FIFO after deducting the GRXTHRCFG..ResvISOCOUTSpc value. Whenever a packet is accepted, the NUMP in the ACK TP is the minimum value of (DCFG.NUMP, DEPCFG.BrstSiz). ■ The device returns NRDY if there is no space in the RX FIFO left after deducting GRXTHRCFG.ResvISOCOUTSpc value. After returning NRDY, the endpoint enters flow control. ■ When at least one packet space becomes available after reserving the space as indicated in GRXTHRCFG.ResvISOCOUTSpc, the device returns ERDY and the NUMP indicated in the ERDY is the minimum value of (DCFG.NUMP, DEPCFG.BrstSiz). <p>Note: GRXTHRCFG.USBRxPktCnt, and GRXTHRCFG.USBMaxRxBurstSize must be set to zero in this mode.</p>

USBRxPktCntSel	ResvISOCOUTSpc	USB 3.0 Device-Mode Receive Path Behavior
1	0	<ul style="list-style-type: none"> ■ The device returns NUMP value in the ACK TP as the minimum of (RX FIFO Space, DEPCFG.BrstSiz). ■ If GRXTHRCFG.USBMaxRxBurstSize is, <ul style="list-style-type: none"> □ zero, the device returns NUMP value for ERDY as GRXTHRCFG.USBRxPktCnt. □ a non-zero value, the device returns NUMP value for ERDY as GRXTHRCFG.USBMaxRxBurstSize. ■ If GRXTHRCFG.USBRxPktCnt is, <ul style="list-style-type: none"> □ zero, the device waits for one packet space in RX FIFO before it sends ERDY. □ a non-zero value, the device waits for space in RX FIFO for GRXTHRCFG.USBRxPktCnt packets before it sends ERDY. <p>Note: DCFG.NUMP is used for control endpoints.</p>
1	Non-zero	<p>For non-ISOC OUT EPs,</p> <ul style="list-style-type: none"> ■ The device returns the NUMP value in the ACK TP as the minimum of (RX FIFO Space, DEPCFG.BrstSiz). The RX FIFO Space is computed after deducting the GRXTHRCFG.ResvISOCOUTSpc value. ■ If GRXTHRCFG.USBMaxRxBurstSize is, <ul style="list-style-type: none"> □ zero, the device returns NUMP value for ERDY as GRXTHRCFG.USBRxPktCnt. □ a non-zero value, the device returns NUMP value for ERDY as GRXTHRCFG.USBMaxRxBurstSize. ■ If GRXTHRCFG.USBRxPktCnt is, <ul style="list-style-type: none"> □ zero, the device waits for one packet space in RX FIFO before it sends ERDY. □ a non-zero value, the device waits for space in RX FIFO for GRXTHRCFG.USBRxPktCnt packets before it sends ERDY. If space of ISOC OUT EPs is to be reserved, then program a value after taking into account the amount of packet count that needs to be reserved for ISOC OUT. <p>Note: DCFG.NUMP is used for control endpoints.</p>

USB 2.0 Device-Mode Receive Path

Table 3-5 shows USB 2.0 device-mode receive path behavior based on GRXTHRCFG.ResvISOCOUTSpc register field.

Table 3-5 USB 2.0 Device Behavior Based on GRXTHRCFG.ResvISOCOUTSpc Register Field

ResvISOCOUTSpc	USB 2.0 Device-Mode Receive Path Behavior
0	<p>For non-ISOC OUT EPs,</p> <ul style="list-style-type: none"> ■ the device continues to receive packets until there is space in the RX FIFO. <p>When a packet is accepted, the device returns</p> <ul style="list-style-type: none"> ■ an ACK response to the host indicating that it has accepted wMaxPacketSize data payload. ■ a NYET response to the host indicating that it has accepted the data but does not have space for another wMaxPacketSize data payload. ■ when there is no space in RX FIFO, the device returns a NAK response. The USB Host uses a PING token to check before issuing the next OUT token.
Non-zero	<p>For non-ISOC OUT EPs,</p> <ul style="list-style-type: none"> ■ the device continues to receive packets until there is space in the RX FIFO after reserving GRXTHRCFG.ResvISOCOUTSpc. <p>When a packet is accepted, the device returns</p> <ul style="list-style-type: none"> ■ an ACK response to the host indicating that it has accepted wMaxPacketSize data payload. ■ a NYET response to the host indicating that it has accepted the data but does not have space for another wMaxPacketSize data payload. ■ when there is space in RX FIFO which is equal or less than GRXTHRCFG.ResvISOCOUTspc, the device returns a NAK response. The USB Host uses a PING token to check before issuing the next OUT token.

3.5 Total DRD RAM Requirement

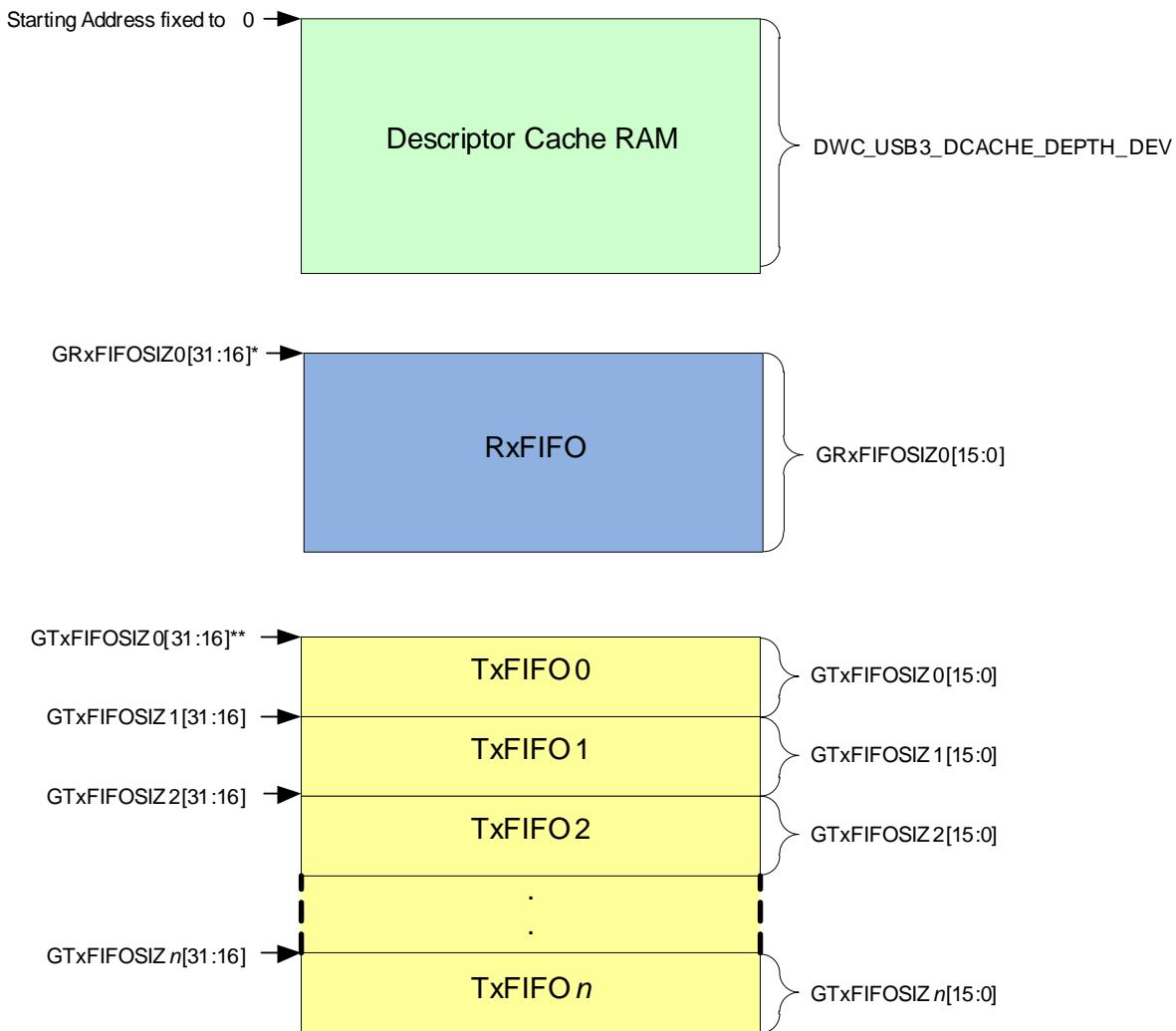
The DRD RAM sizes are determined as the larger of the device and Host RAM0, RAM1, and RAM2 sizes.

The DRD TXFIFO and RXFIFO address and size reset values reflect the device configuration. However, upon switching to Host mode, the controller automatically resets the TxFIFOs and RxFIFOs to reflect the Host configuration.

3.6 Rx/Tx FIFO Configuration

The DWC_usb3 controller provides flexible buffering architecture by allowing software to re-configure the FIFO sizes during controller initialization. [Figure 3-7](#), [Figure 3-8](#), [Figure 3-9](#), [Figure 3-10](#), and [Figure 3-11](#) illustrate the usage of the GTxFIFOSIZn and GRxFIFOSIZn registers for different modes of operation.

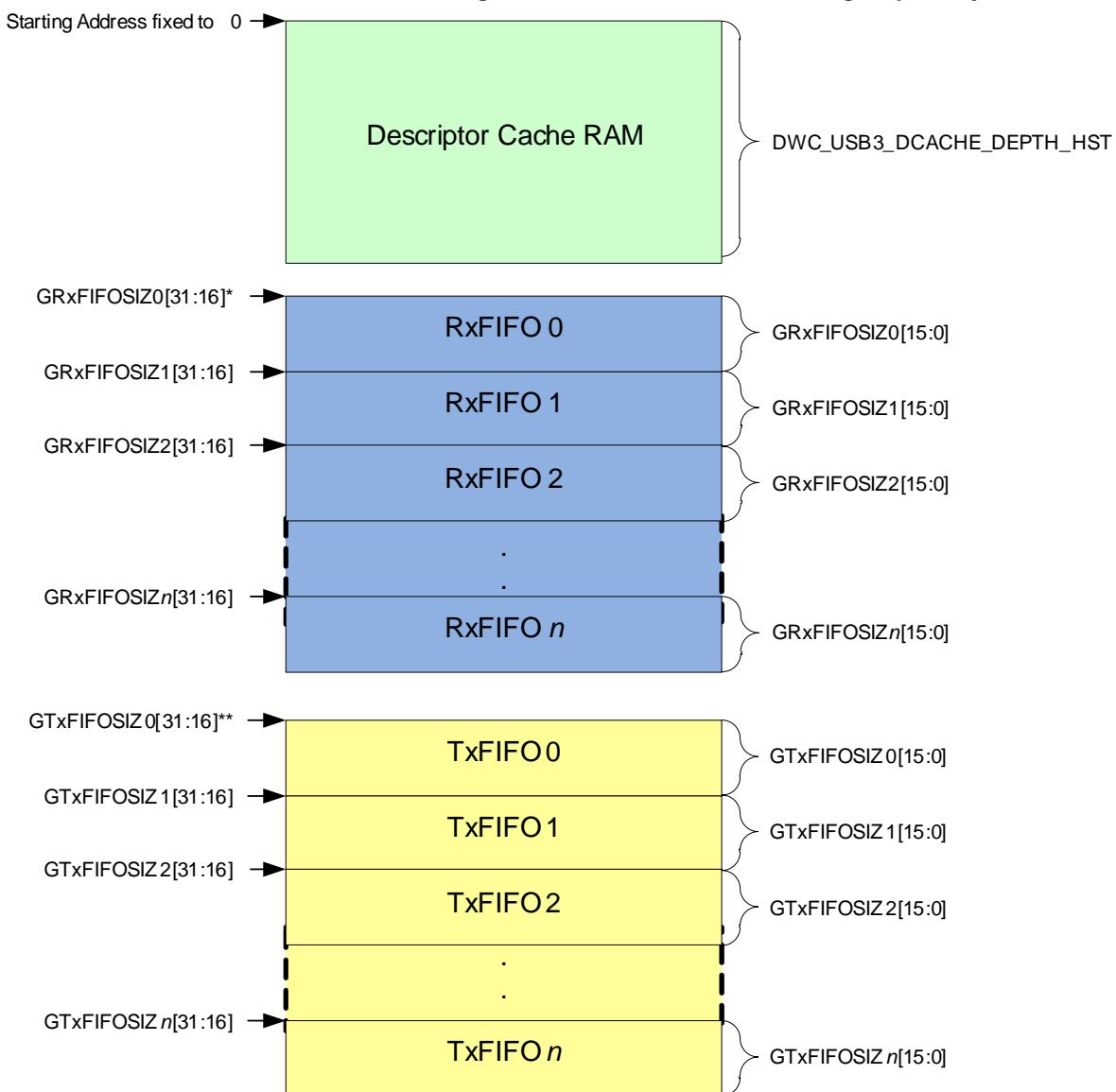
Figure 3-7 GTxFIFOSIZn and GRxFIFOSIZn Usage in Device Mode



n is `DWC_USB3_NUM_TXFIFO-1`.

*If `DWC_USB3_NUM_RAMs` = 1 or 2, the reset value of `GRxFIFOSIZ0[31:16]` is `DWC_USB3_DCACHE_DEPTH_DEV`.

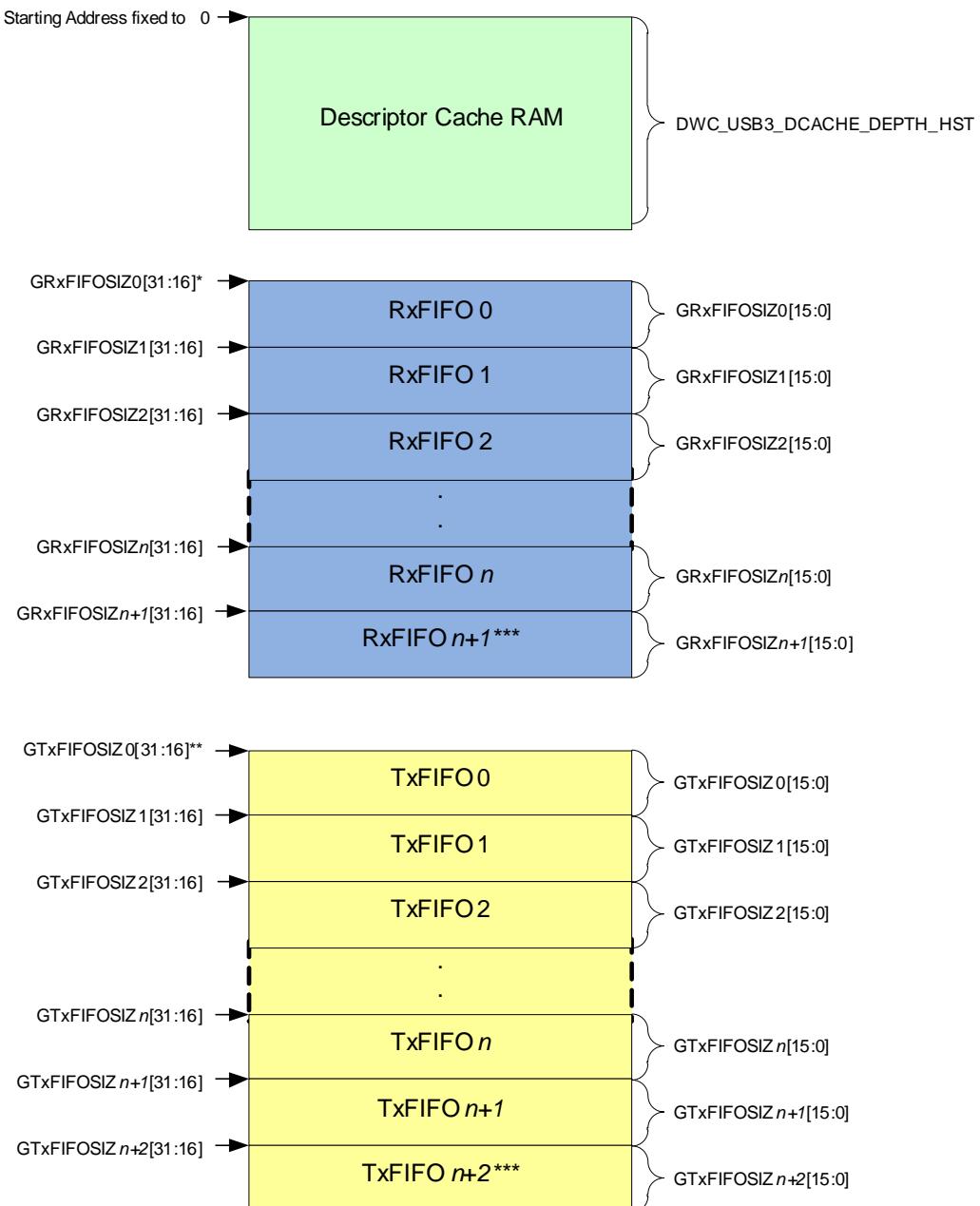
**If `DWC_USB3_NUM_RAMs` = 1, the reset value of `GTxFIFOSIZ0[31:16]` is `DWC_USB3_DCACHE_DEPTH_DEV + DWC_USB3_DEV_RXF_DEPTH`.

Figure 3-8 GTxFIFOSIZn and GRxFIFOSIZn Usage in Host Mode Without Debug Capability

n is `DWC_USB3_NUM_TXFIFO-1`

*If `DWC_USB3_NUM_RAMS` = 1 or 2, the reset value of `GRxFIFOSIZ0[31:16]` is `DWC_USB3_DCACHE_DEPTH_HST`.

**If `DWC_USB3_NUM_RAMS` = 1, the reset value of `GTxFIFOSIZ0[31:16]` is `DWC_USB3_DCACHE_DEPTH_HST + DWC_USB3_HOST_RXF_ALL_DEPTH`.

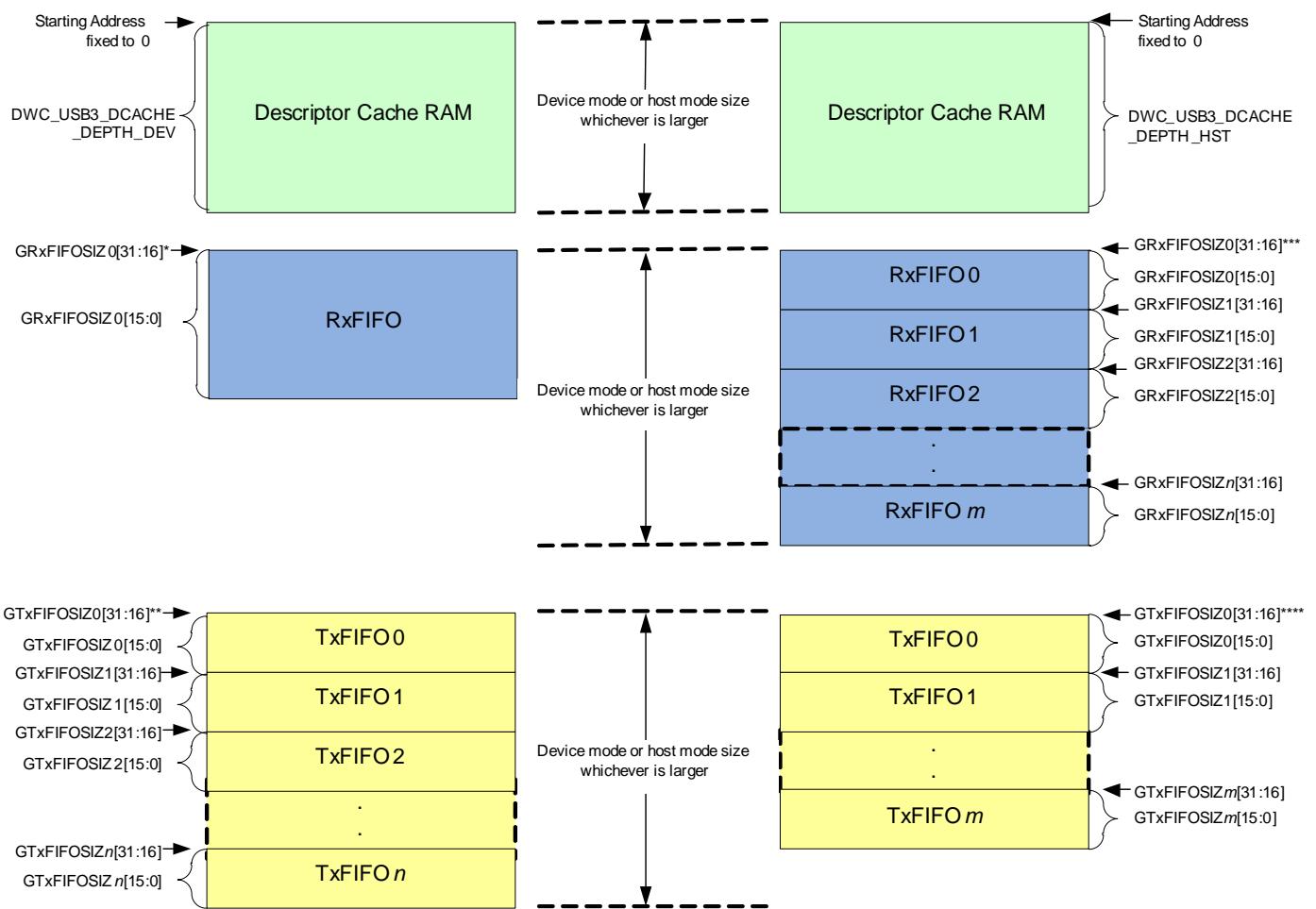
Figure 3-9 GTxFIFOSIZn and GRxFIFOSIZn Usage in Host Mode With Debug Capability

n is DWC_USB3_NUM_TXFIFO-1

*If DWC_USB3_NUM_RAMs = 1 or 2, the reset value of GRxFIFOSIZ0[31:16] is DWC_USB3_DCACHE_DEPTH_HST.

**If DWC_USB3_NUM_RAMs = 1, the reset value of GTxFIFOSIZ0[31:16] is DWC_USB3_DCACHE_DEPTH_HST + DWC_USB3_HOST_RXF_ALL_DEPTH.

*** RxFIFO n+1 and TxFIFO n+2 are present only when
DWC_USB3_NUM_U3_ROOT_PORTS > DWC_USB3_NUM_SS_USB_INSTANCES.

Figure 3-10 GTxFIFOSIZn and GRxFIFOSIZn Usage in DRD Mode Without Debug Capability

n is DWC_USB3_NUM_TXFIFO-1 in Device mode.

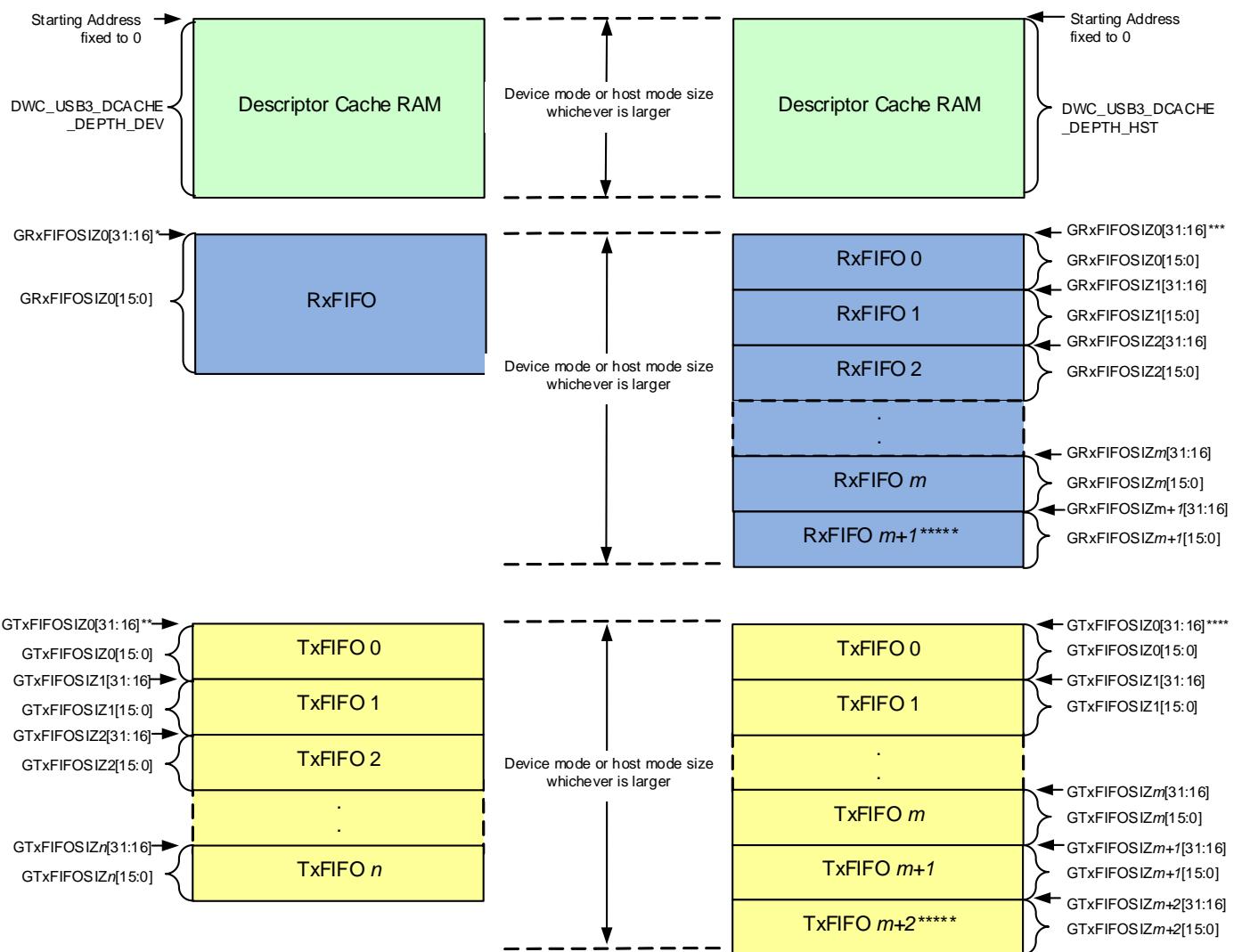
*If DWC_USB3_NUM_RAMs = 1 or 2, the reset value of GRxFIFOSIZ0[31:16] is DWC_USB3_DCACHE_DEPTH_DEV.

**If DWC_USB3_NUM_RAMs = 1, the reset value of GTxFIFOSIZ0[31:16] is DWC_USB3_DCACHE_DEPTH_DEV + DWC_USB3_DEV_RXF_DEPTH.

m is DWC_USB3_NUM_TXFIFO-1 in Host mode.

***If DWC_USB3_NUM_RAMs = 1 or 2, the reset value of GRxFIFOSIZ0[31:16] is DWC_USB3_DCACHE_DEPTH_HST.

****If DWC_USB3_NUM_RAMs = 1, the reset value of GTxFIFOSIZ0[31:16] is DWC_USB3_DCACHE_DEPTH_HST + DWC_USB3_HOST_RXF_ALL_DEPTH.

Figure 3-11 GTxFIFOSIZn and GRxFIFOSIZn Usage in DRD Mode With Debug Capability

n is `DWC_USB3_NUM_IN_EPS-1` in Device mode.

*If `DWC_USB3_NUM_RAMs = 1` or `2`, the reset value of `GRxFIFOSIZ0[31:16]` is `DWC_USB3_DCACHE_DEPTH_DEV`.

**If `DWC_USB3_NUM_RAMs = 1`, the reset value of `GTxFIFOSIZ0[31:16]` is `DWC_USB3_DCACHE_DEPTH_DEV + DWC_USB3_DEV_RXF_DEPTH`.

m is `NUM_SS_USB_INSTANCES + NUM_HS_USB_INSTANCES + NUM_FSLS_USB_INSTANCES + ((NUM_U3_ROOT_PORTS > NUM_SS_USB_INSTANCES) ? 1 : 0) - 1`.

***If `DWC_USB3_NUM_RAMs = 1` or `2`, the reset value of `GRxFIFOSIZ0[31:16]` is `DWC_USB3_DCACHE_DEPTH_HST`.

****If `DWC_USB3_NUM_RAMs = 1`, the reset value of `GTxFIFOSIZ0[31:16]` is `DWC_USB3_DCACHE_DEPTH_HST + DWC_USB3_HOST_RXF_ALL_DEPTH`.

***** Rx FIFO $m+1$ and Tx FIFO $m+2$ are present only when `DWC_USB3_NUM_U3_ROOT_PORTS > DWC_USB3_NUM_SS_USB_INSTANCES`.

3.7 System Bus and RAM Bandwidth Requirements

While USB 2.0 has a maximum throughput of 480 Mbps, USB 3.0 has a maximum combined throughput of 8 Gbps (4 Gbps IN and 4 Gbps OUT). A device that does not support concurrent IN and OUT access would require only 4 Gbps of bandwidth. However, in a multi-port host configuration, each SS bus instance can provide 8 Gbps of throughput, which means that a host with two SS bus instances could have up to 16 Gbps of throughput.

To meet your bandwidth needs, choose appropriate values for: master data bus width, bus and RAM clock frequencies, total number of RAMs, and single port or 2-port RAM options.

For example, to achieve full USB 3.0 performance with concurrent IN and OUT transfers, a 32-bit system bus would need a clock frequency of at least 250 MHz and the controller would need a RAM clock frequency of 250 MHz if it uses 2 or 3 single-port RAMs. While the TxFIFOs and RxFIFO(s) are in separate RAMs in the 2 or 3 RAM configurations, if the RAMs are 2-port instead of single-port, the required RAM clock frequency would be halved to achieve the same data rate. This discussion does not take into account any memory fetch delays. In addition, in a single port RAM configuration the MAC has priority over the system bus access. Depending on the RAM clock frequency and BUS clock frequency, the controller may need to introduce wait states. If you have a CPU and other peripherals sharing the same bus, then this would not meet your overall system bandwidth, and you may need a 64-bit bus or higher bus clock frequency.

For device in 1 or 2 RAM configurations:

- Single-Port RAM throughput = (RAM frequency in MHz * Number of RAMs (1 or 2) * Master Bus Width) > 16.1 Gbs
- 2-Port RAM throughput = (2 * RAM frequency in MHz * Number of RAMs (1 or 2) * Master Bus Width) > 16.1 Gbs

The additional 100 Mbs bandwidth is for the cache and queues access in 1 or 2 RAM configurations.

You can also configure the controller by selecting one of the following clocks for the RAM clock (GCTL[7:6] Register):

- system bus clock
- pipe3-clock
- pipe3-clock/2

For example, if your system bus clock is 150 MHz and pipe3-clock is 250 MHz, you must choose pipe3-clock for the RAMs. If your system bus clock frequency is 400 MHz and pipe3-clock is 250 MHz, then you must choose system clock for your RAM to provide maximum throughput.

3.7.1 Number of RAMs

Table 3-6 describes the RAM-1, RAM-2, and RAM-3 configurations.

Table 3-6 RAM-1, RAM-2, and RAM-3 Configurations

Number of RAMs	Configuration	What It Means
1-RAM	queues/cache, the TxFIFOs and the RxFIFO(s) are together.	The RxFIFO write/read and TxFIFO write/read may occur at the same time on the same RAM interfaces.
2-RAM	queues/cache and RxFIFO(s) are in RAM[0] TxFIFO(s) are in RAM[1]	To support the maximum full-duplex data throughput, two 32-bit wide SPRAMs must each operate at a minimum of 250 MHz + n, where n is the throughput requirements of cache/queue access.
3-RAM	queues/cache are in RAM0, TxFIFOs are in RAM1, and RxFIFO(s) are in RAM2	To support the maximum full-duplex data throughput, three 32-bit wide SPRAMs must each operate at a minimum of 250 MHz.



Because of the high minimum RAM clock frequency to sustain the maximum full-duplex throughput of USB 3.0 SuperSpeed, you may require a 64-bit wide system bus and/or use 2-port RAM buffering.

3.7.2 2-Port or Single-Port RAM

The 2-port RAM option provides the USB side and system bus side access to the RAM without contention from the other. This reduces by half the required RAM clock frequency.

The 2-port RAM configuration is also recommended if your system bus frequency is high (>60 MHz for HS operation and >125 MHz for SS operation) to minimize or prevent wait states introduced on the system bus by the controller. Although the single-port RAM configuration has additional logic to manage arbitration, the total area is smaller than a 2-port RAM configuration of the same size.

Despite the somewhat larger area, the 2-port ram option provides an excellent compromise of performance, clock-frequency, and area.

3.7.3 Host Mode – Number of High-Speed and SuperSpeed Bus Instances

When you have multiport host, you can choose one for the “Number of Ports” parameter of SS Bus instance. When this parameter is set, each root hub port has its own full-duplex SuperSpeed bandwidth. This means that each root hub port requires its own PTL-BMU Sink/Source buffers and Rx/Tx RAM-based FIFOs.

Because the total maximum data throughput through the BMU becomes large (each port supports 4 GBs IN and 4 GBs OUT for a total of 64 GBs for eight concurrent ports), you must be careful while selecting the appropriate RAM configuration and RAM clock frequency, as well as system bus data width and clock frequency.

In the case of multiple HS-Bus Instances, each HS Bus Instance would require 480 MB additional system bandwidth.

3.8 Additional RAM Requirements for xHCI Debug Capability

When the Debug Capability is enabled (that is, when parameter DWC_USB3_EN_DBC is 1), additional internal RAM is needed due to:

- Extra TxFIFOs and RxFIFOs
- Extra space for the DbC LSP Cache
- DbC RAM-based CSRs

The following sections describe the additional internal RAM requirements that are contributed by these three factors.

3.8.1 Additional TxFIFOs and RxFIFOs

In order to support simultaneous host and debug target operation, most configurations require additional RAM and FIFO controllers. One exception is a DRD configuration with a large device TxFIFO RAM requirement.

[Table 3-7](#) indicates the increased RAM requirement, which depends on the number of Host U3 root ports and SS USB instances as specified through coreConsultant. The last column indicates the presence of an additional set of sink/source 2clkfifo buffers. Each set consists of three flop-based FIFOs that are MDWIDTH wide and 8 deep if MDWIDTH is 32 or 64, or 4 deep if MDWIDTH is 128.

Table 3-7 DbC TxFIFO RAM Requirements Based on Configuration

Number of U3 Ports	Number of SS USB Instances	TxFIFO RAM +	TxFIFO Controller +	RxFIFO RAM +		Cache RAM +		Sink/Source Buffer +	
				MDW 32,64	MDW 128	RxFIFO Controller +	MDW 32,64		
1	1	256B+2W ^a	1	0	0	0	64B	128B	0
2	1	3.25KB+6W	2	3KB+40B	3KB+64B	1	4*64B	4*128B	1
2	2	256B+2W	1	0	0	0	64B	128B	0
4	2	3.25KB+6W	2	3KB+40B	3KB+64B	1	4*64B	4*128B	1
4	4	256B+2W	1	0	0	0	64B	128B	0

a. W is the number of bytes in a FIFO word. If MDWIDTH is 32, then W is 4; if MDWIDTH is 64, then W is 8; if MDWIDTH is 128, then W is 16.

In all cases in which an additional TxFIFO controller is necessary, an additional TXQ FIFO controller is also necessary along with additional space in RAM0 (64B if MDWIDTH is 32 or 64, or 128B if MDWIDTH is 128). Similarly, in cases in which an additional Rx FIFO is necessary, an additional RXQ FIFO controller and RIQ FIFO controller are necessary along with RAM0 space as indicated previously.

Examples:

1. In the single-port configuration (see first row in [Table 3-7](#)), one additional TxFIFO controller is necessary along with 256 bytes plus two FIFO words of TxFIFO RAM. If MDWIDTH is 64, then the TxFIFO RAM requires an additional 272 bytes ($256 + 2 * 8$), and the Cache RAM requires 64 bytes for the extra TXQ.
2. In a two-port configuration with one SS USB instance (see second row in [Table 3-7](#)), two additional TxFIFO controllers are necessary along with 3.25KB + 6W TxFIFO RAM, assuming the default three packet buffering requirement for U3 burst operation. One additional RxFIFO is necessary along with 3KB + 40B of RxFIFO RAM, and 4*64B of Cache RAM for the four additional FIFO controllers for the DMA queues (TXQ (2), RXQ, RIQ).
3. In a single-port DRD configuration with four device IN endpoints and a total device TxFIFO RAM requirement larger than the host TxFIFO RAM requirement plus DBC requirement in [Table 3-7](#) (similar to last row but with specified device configuration enabled), no additional FIFO controllers are necessary and no additional FIFO RAM is necessary.

3.8.2 DbC LSP Cache

The DbC LSP uses a part of the internal RAM to track transfers to and from the Debug Host. This is called the *DbC LSP Cache*. The DbC LSP Cache consists of the following sections:

- 12 DWORDs for the DbCIC (String Descriptor addresses and lengths plus padding)
- 4 DWORDs for EP1 OUT Endpoint Context (referred to in the xHCI specification as the “IN Transfer Ring”)
- 8 DWORDs for EP1 OUT Endpoint Scratchpad
- 4*N DWORDs for EP1 OUT TRB cache
- 4 DWORDs for EP1 IN Endpoint Context (referred to in the xHCI specification as the “OUT Transfer Ring”)
- 8 DWORDs for EP1 IN Endpoint Scratchpad
- 4*N DWORDs for EP1 IN TRB cache

The “N” in the previous equations is the number of TRBs per endpoint to be cached. The default value is 16, so the default space required for the DbC LSP Cache is $12+4+8+4*16+4+8+4*16 = 164$ DWORDs, or 656 bytes. This space is added to RAM0.

3.8.3 DbC RAM-Based CSRs

Several of the DbC registers are stored in the internal RAM. These types of CSRs are called “RAM-based CSRs,” and they increase the internal RAM requirement. For DbC, the following registers are RAM-based and increase the internal RAM requirement by 32 bytes:

- DCERSTBA_LO
- DCERSTBA_HI
- DCERDP_LO
- DCERDP_HI
- DCCP_LO
- DCCP_HI
- DCDDI1
- DCDDI2

This space is added to RAM0.

3.9 RAM Requirements for USB 2.0-Only Mode

Table 3-8 summarizes the RAM requirements in USB 2.0-only mode.

Table 3-8 RAM Requirements for USB 2.0-Only Mode

RAM Section	USB 2.0 Device					USB 2.0 Host					USB 2.0 DRD			
	Size			Address		Size			Address		Size		Address	
	Pkts	DW	B	DW	B	Pkts	DW	B	DW	B	DW	B	DW	B
DMA queues	-	196	784	0	0	-	276	1104	0	0	356	1424	0	0
Global CSR	-	4	16	196	784	-	4	16	276	1104	4	16	356	1424
Dev/Host CSR	-	32	128	200	800	-	84	336	280	1120	84	336	360	1440
DCache EP	-	32	128	232	928	-	0	0	364	1456	0	0	444	1776
DCache Device	-	4	16	264	1056	-	0	0	364	1456	0	0	444	1776
DCache Strms (TRBs)	-	192	768	268	1072	-	0	0	364	1456	0	0	444	1776
HCACHE	-	0	0	460	1840	-	1616	6464	364	1456	0	0	444	1776
RXFIFO[0]	2 x 1024B	522	2088	460	1840	1 x 1024B	264	1056	1980	7920	0	0	444	1776
RXFIFO[1]	-	0	0	982	3928	2 x 1024B	520	2080	2244	8976	0	0	444	1776
TXFIFO[0]	1 x 64B	18	72	982	3928	1 x 1024B	258	1032	2764	11056	0	0	444	1776
TXFIFO[1]	2 x 1024B	515	2060	1000	4000	2 x 1024B	515	2060	3022	12088	0	0	444	1776
TXFIFO[2]	2 x 1024B	515	2060	1515	6060	-	0	0	3537	14148	0	0	444	1776
TXFIFO[3]	2 x 512B	259	1036	2030	8120	-	0	0	3537	14148	0	0	444	1776
DRD RAM section	-	0	0	2289	9156	-	0	0	3537	14148	3173	1269 2	444	1776
Total RAM depth	-	-	-	2289	9156	-	-	-	3537	14148	-	-	3617	1446 8

Parameter Descriptions

This chapter details all the configuration parameters. You can use the coreConsultant GUI configuration reports to determine the complete configuration state of the controller. Some expressions might refer to TCL functions or procedures (sometimes identified as <functionof>) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

The parameter descriptions in this chapter include the **Enabled:** attribute which indicates the values required to be set on other parameters before you can change the value of this parameter.

Notes:

- For more details on the parameters other than the coreConsultant parameters that are mentioned in this chapter, refer to *workspace/src/DWC_usb3_params.v* file.
- **USB Class Specifications:** Class specifications are available for many USB applications, which help you determine your implementation requirements (for example, number of endpoints and endpoint types). You can review these specifications before selecting your custom configuration parameters. To download USB-IF approved class specifications for audio, mass storage, printer, smart card, and video devices, go to:
http://www.usb.org/developers/devclass_docs/
- **Register Power-on Initialization Values:** When using standard drivers, such as Microsoft for xHCI controller, the driver does not understand Synopsys controller-specific registers. Depending on your system requirements, the power-on value of these registers can be selected during coreConsultant configuration. Another option is to have an additional software driver which initializes these registers before loading standard drivers such as xHCI driver. If you write your own software driver, then your driver can initialize these registers and you do not have to initialize these registers during coreConsultant configuration.

These tables define all of the user configuration options for this component.

- “[Basic Config Parameters](#)” on page [191](#)
- “[PHY Config Parameters](#)” on page [203](#)
- “[Device Config Parameters](#)” on page [214](#)
- “[Host Config Parameters](#)” on page [220](#)
- “[Hub Config Parameters](#)” on page [229](#)

- “Advanced Config Parameters” on page 233

4.1 Basic Config Parameters

Table 4-1 Basic Config Parameters

Label	Description
Mode of Operation	<p>Selects controller mode of operation. Device, Host, DRD, or Hub configuration selection needs the corresponding license purchase. With a USB 3.0 DRD license, you can configure the controller as a device, host, or DRD (you can select DWC_USB3_MODE as 0, 1, or 2).</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Device (0) ■ Host (1) ■ Device and Host (DRD) (2) ■ Hub (3) <p>Default Value: = ((<DWC-USB30-DEV-SRC feature authorize>==1)? 0 : ((<DWC-USB30-HST-SRC feature authorize>==1)? 1 : (<DWC-USB30-HUB-SRC feature authorize>==1)? 3 : 2))</p> <p>Enabled: Always</p> <p>Parameter Name: DWC_USB3_MODE</p>
Enable USB 2.0-only mode?	<p>Enables USB 2.0 only mode, which removes SuperSpeed gates and RAM requirements.</p> <p>If you need USB 2.0 only support but still want to use USB 3.0 software drivers, this option provides a smaller gate count solution. For example, in your SoC if you need USB 2.0 and USB 3.0 controllers, then using two instances of USB 3.0 controller (one in USB 2.0 only mode) will reduce your IP integration and SoC development time; also the product needs only one SW driver development.</p> <p>In host mode, only Linux and MCCI xHCI drivers support USB 2.0 only mode. Microsoft Windows 8.1 driver does not support USB 2.0 only mode since it expects that both the USB 2.0 and USB 3.0 port extended capabilities be present. Check with Microsoft on the availability of Windows support.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: ((<DWC-USB30-DEV-SRC feature authorize>==1 <DWC-USB30-HST-SRC feature authorize> DWC_USB3_MODE==3) ? 0 : 1)</p> <p>Enabled: DWC_USB3_MODE == 0 DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_EN_USB2_ONLY</p>

Label	Description
Enable Isochronous Support?	<p>Enables isochronous endpoint capability. It is recommended that you always enable this feature in host mode because this is a required feature in a standard xHCI host. There is no area saving by disabling this feature in either device or host mode.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: Yes</p> <p>Enabled: <code>DWC_USB3_MODE == 0 DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</code></p> <p>Parameter Name: <code>DWC_USB3_EN_ISOC_SUPT</code></p>
Power Optimization Mode	<p>Specifies the power optimization mode.</p> <ul style="list-style-type: none"> ■ If Clock Gating Only is selected, BUS and RAM domain module clocks are gated when the controller is inactive during L1, L2, U1, U2, and U3 states. Note that the PHY and MAC clocks are turned off by the PHY during L1 sleep (by Synopsys-PHY), L2, and U3 states. ■ If Hibernation (Two Power Rails) is selected, in addition to clock gating, you can power down the DWC_usb3 controller to minimize leakage power during L1 (device mode only), L2, and U3 states. Only the smaller U2PMU and U3PMU modules are active to detect wakeup conditions. Because the always-on U2PMU/U3PMU modules are much smaller (~5K gates) than the power-gated DWC_usb3 controller (~250K gates in device mode), the leakage power during hibernation is reduced. To use the Hibernation feature, you need to purchase a Hibernation Add-on license. <p>Values:</p> <ul style="list-style-type: none"> ■ No Power Optimization (0) ■ Clock Gating Only (1) ■ Clock Gating and Hibernation (Two Power Rails) (2) <p>Default Value: No Power Optimization</p> <p>Enabled: <code>DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4</code></p> <p>Parameter Name: <code>DWC_USB3_EN_PWROPT</code></p>
Enable Battery Charging Capability?	<p>Enables Battery Charging (ACA) capability. When enabled, the controller supports battery charger and provides support for Accessory Charging Adaptor (ACA) pins. Battery Charger is not supported in the HSIC interface. For more information, refer to "Modes Of Operation" under the "Battery Charger" section of the Databook.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: <code>DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4</code></p> <p>Parameter Name: <code>DWC_USB3_EN_BC</code></p>

Label	Description
Master Bus (DMA Bus) Interface Type	<p>Selects the SoC Master Bus interface type. The Master bus is used for DMA.</p> <ul style="list-style-type: none"> ■ If your SoC is AHB-based, or if you have an AHB to your SoC bus bridge, select AHB. ■ If your SoC is AXI-based, or if you have an AXI to your SoC bus bridge, select AXI. ■ Otherwise, select the Native bus interface, and design a bridge from DWC_usb3's Native interface to your SoC bus. <p>Because the Master Bus is critical for DMA operation/data transfer, selecting the bus type, frequency, and data width of this bus are critical in meeting USB performance. For example, because USB 3.0 is full duplex bus, a full duplex AXI bus would provide better bandwidth than an AHB bus. For more information, refer to the "Minimum Clock Frequencies; bus_clk, ram_clk" section in the Databook.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ AHB (0) ■ AXI (1) ■ Reserved2 (2) ■ Native (3) ■ Reserved (4) <p>Default Value: AHB</p> <p>Enabled: DWC_USB3_MODE !=3</p> <p>Parameter Name: DWC_USB3_MBUS_TYPE</p>
Slave Bus (Register Access Bus) Interface Type	<p>Selects the SoC Slave Bus interface type. The Slave bus is used for register programming.</p> <ul style="list-style-type: none"> ■ If your SoC is AHB-based, or if you have an AHB to your SoC bus bridge, select AHB. ■ If your SoC is AXI-based, or if you have an AXI to your SoC bus bridge, select AXI. ■ Otherwise, select the Native bus interface, and design a bridge from DWC_usb3's Native interface to your SoC bus. <p>Because the Slave Bus is only used for register access (which are less frequent) the performance on this bus is not as critical as in the Master Bus.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ AHB (0) ■ AXI (1) ■ Reserved2 (2) ■ Native (3) <p>Default Value: AHB</p> <p>Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_SBUS_TYPE</p>

Label	Description
Master Bus (DMA Bus) Data Bus Width	<p>Selects data bus width of the Master Bus interface. For USB 3.0 operation, 64-bit or larger data width is recommended to meet USB bandwidth. 33-bit is used only for Hub configuration. For more information, refer to the "Minimum Clock Frequencies; bus_clk, ram_clk" section in the Databook.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 32 bits (32) ■ 33 bits (33) ■ 64 bits (64) ■ 128 bits (128) <p>Default Value: ((DWC_USB3_MODE==3) ? 33 : (DWC_USB3_MBUS_TYPE==4)? 32 : (DWC_USB3_EN_USB2_ONLY==1)? 64: 64)</p> <p>Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_MDWIDTH</p>
Slave Bus (Register Access Bus) Data Bus Width	<p>Selects the data bus width of the Slave Bus interface.</p> <p>Note: The slave data bus width must not be larger than the master data bus width.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 32 bits (32) ■ 64 bits (64) ■ 128 bits (128) <p>Default Value: 32 bits</p> <p>Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_SDWIDTH</p>
Master/Slave Address Bus Width	<p>Selects the address bus width of the master and slave interfaces.</p> <ul style="list-style-type: none"> ■ If your system address bus width is 32 bits, then select 32 bits. ■ If your system address bus width is 64 bits, then select 64 bits. ■ If your system address bus width for example is 40 bits, select 64 and tie the address bits [63:40] of the DWC_usb3 to 0. <p>The address width is common for both master and slave interfaces; and must be set to meet both the DMA master and slave interface requirements. The slave interface uses only address bits [19:0] and for the others, don't care.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 32 bits (32) ■ 64 bits (64) <p>Default Value: 32 bits</p> <p>Enabled: DWC_USB3_MODE !=3</p> <p>Parameter Name: DWC_USB3_AWIDTH</p>

Label	Description
Master ID Port Width	<p>Selects the ID port width of the master bus interface. Because the USB3.0 controller supports up to a maximum of 16 outstanding transfers, using more than 4 master ID widths is not required. The number of registers to track the transfers increases by two to the power of the master ID widths, therefore the larger area overhead when selecting a larger master ID. The number of flops used for tracking transfers are 272, 544, 1066, 2176, and 4352 when master ID widths 4 to 8 are used.</p> <p>Values: 4, 5, 6, 7, 8 Default Value: 4 Enabled: DWC_USB3_MBUS_TYPE!=0 && DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_IDWIDTH</p>
Slave ID Port Width	<p>Selects the ID port width of the slave bus interface. There is no area overhead in using larger slave ID width because the USB 3.0 only registers the input ID and returns it back along with the response.</p> <p>Values: 1, ..., 32 Default Value: 4 Enabled: DWC_USB3_SBUS_TYPE!=0 && DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_SIDWIDTH</p>
Number of RAMs	<p>Selects the number of RAMs.</p> <p>Configuration Recommendations:</p> <ul style="list-style-type: none"> ■ USB 3.0 device and host requires 2 or 3 RAMs (3 recommended); In 3-RAM configuration, RAM0 is used for registers and descriptor cache, RAM2 for Rx-buffering, and RAM1 for Tx prefetch. In a 2-RAM configuration, RAM0 is used for registers, descriptor cache, and Rx-buffering and RAM1 for Tx prefetch. ■ USB 2.0-only mode device and host requires only 1 RAM (USB 2.0 needs less bandwidth); RAM0 is used for registers, descriptor cache, Rx buffering, and Tx prefetch. ■ Hub requires two 2-port RAMs. ■ For USB 3.0 device and host, three 2-Port RAMs provide the best compromise among performance, low clock frequency, and area. <p>Values:</p> <ul style="list-style-type: none"> ■ 1 (1) ■ 2 (2) ■ 3 (3) <p>Default Value: (DWC_USB3_EN_USB2_ONLY==1 ? 1: DWC_USB3_MODE==3 ? 2 : DWC_USB3_MBUS_TYPE==4 ? 2 : DWC_USB3_MODE!=0 ? 3 : 3) Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 && DWC_USB3_EN_USB2_ONLY != 1 Parameter Name: DWC_USB3_NUM_RAMs</p>

Label	Description
Synchronous Static RAM Type	<p>Selects either 2-Port RAM (single-clock, 1 port for read only, and 1 port for write only) or Single-Port RAM (SPRAM). Two-port is preferred for USB 3.0 applications.</p> <ul style="list-style-type: none"> ■ Select 2-port RAM for performance-sensitive applications or if your system bus frequency is very low (for example, below 66 MHz). ■ Select single-port RAM if you do not have 2-port RAM in your library or for USB 2.0 only mode or for area-sensitive applications. <p>Note that the DWC_usb3 controller needs only 2-port RAM which runs on a single clock and one port is read only and one port is write only and not a 2 clock true dual port ram which supports read and write on both ports. Depending on your memory compiler and RAM size, the area of a true dual-port RAM is 1.2-1.8 times larger than a single-port RAM of the same RAM size. The 2-port RAMs are normally a little larger than SPRAM but much smaller than DPRAM.</p> <p>When SPRAM is selected, the same port is shared by both the MAC and SoC Bus; the MAC has priority over the SoC Bus. Depending up on the ram clock frequency, busy cycles may be inserted on the SoC Master interface during simultaneous MAC and SoC accesses to the RAM.</p> <p>For USB 3.0 device and host, three 2-Port RAMs provides the best compromise between performance, lower clock frequency, and area. For additional information, refer to the "Minimum Clock Frequencies; bus_clk, ram_clk" section and "Memory Requirements" sections of the Databook.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 2-Port RAM (0) ■ Single-Port RAM (SPRAM) (1) <p>Default Value: ((DWC_USB3_MODE==3)? 0 : (DWC_USB3_MBUS_TYPE==4)? 0 : (DWC_USB3_MODE!=0)? 0 : 0)</p> <p>Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_SPRAM_TYP</p>
Will UTMI clock PLL be switched off during LPM-L1 (EA Feature)?	<p>This parameter specifies whether you shut down PLL of the UTMI clock during LPM-L1. If this parameter is enabled, then in device mode, SOF tracking is generated with the ref_clk input. This feature can be used to turn off the utmi_clk on LPM when ISOC endpoints are running. This is an EA feature. Contact Synopsys before using this feature.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: Always</p> <p>Parameter Name: DWC_USB3_LPM_SUSP_OFF</p>

Label	Description
Global User ID (GUID) Register's Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the Global User ID (GUID) register's power-on initialization value. After power-up, the software can change this value. This register can be used as either a scratch pad or identification register. For more information, see the "GUID" section of the Databook.</p> <p>Values: 0x0, ..., 0xffffffff Default Value: 0x12345678 Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_USERID</p>
Global User Control (GUCTL) Register's Power-On Initialization Value (0x0-02000010)	<p>Specifies the global User Specific Control Register's power-on initialization value. After power-up, the software can change this value. Choice decides the values of options to improve the host inter-operability with different devices. For more information, see the "GUCTL" section of the Databook.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff Default Value: (DWC_USB3_MODE==0)? 0x10 : 0x02000010 Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_GUCTL</p>
Global User Control 1 (GUCTL1) Register's Power-On Initialization Value (0x0-0004198a)	<p>Specifies the Global User Control 1 Register's power-on initialization value. After power-up, the software can change this value.</p> <ul style="list-style-type: none"> ■ bit[24] - 1'b0: Device L1 exit by hardware disabled; 1'b1: enabled. ■ bit[20] - 1'b0: Device LSP Tail update lock enabled; 1'b1: disabled. <p>For more information, see the "Registers" chapter of the Databook.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff Default Value: = 0x0004198a Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_GUCTL1</p>

Label	Description
Global User Control 2 (GUCTL2) Register's Power-On Initialization Value (0x0-fffffff)	<p>Specifies the Global User Control 2 Register's power-on initialization value. After power-up, the software can change this value.</p> <ul style="list-style-type: none"> ■ bit[4:0]: Indicates the maximum time in units of 8 ns the LTSSM should instruct the PHY to keep transmitting the ping LFPS (For example, a value of 13 indicates 104 ns). ■ bit[10:5]: Indicates the maximum time in units of 8 ns the LTSSM should instruct the PHY to keep receiving the ping LFPS (For example, a value of 32 indicates 256 ns). ■ bit[11]: Disable xHCI Errata Contiguous FrameID capability and Microframe targeting. ■ bit[12]: Enable Evicting Endpoint cache after flow control for bulk endpoints. <p>For more information, see the "Registers" chapter of the Databook. If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff Default Value: = 0x0198040D Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_GUCTL2</p>
Global User Control 3 (GUCTL3) Register's Power-On Initialization Value (0x0-00010000)	<p>Specifies the Global User Control 3 Register's power-on initialization value. After power-up, the software can change this value.</p> <ul style="list-style-type: none"> ■ bit[16]: Enables SuperSpeed Ping Transaction Packet scheduling early in the microframe <p>For more information, see the "Registers" chapter of the Databook. If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0x10000 Default Value: = 0x00010000 Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_GUCTL3</p>
Global Frame Length Adjustment (GFLADJ) Register's Power-On Initialization Value (0x0-00000000)	<p>Specifies the global frame length adjustment Register's power-on initialization value. After power-up, the software can change this value. What you select decides the values of options to program host frame length adjustment. For more information, see the "GFLADJ" section of the Databook. If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff Default Value: = 0x00000000 Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MODE != 0 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_GFLADJ</p>

Label	Description
Does the Master DMA bus latency plus access time exceed the USB MaxPacket transmission time (2.1us for 1KB SS/ ~10 uS for 512Byte HS)?	<p>Configures the Rx FIFO, Tx FIFO, and Cache sizes depending up on the system latency.</p> <p>By setting this parameter, which indicates that the Master DMA bus latency plus access time exceeds the USB MaxPacket transmission time, a larger FIFO size is recommended. For example, when this parameter is set, for SuperSpeed endpoint which supports burst, a 5-packet size FIFO is recommended instead of a 3-packet size FIFO.</p> <p>A 1KB SuperSpeed packet has a transmission time of 2.1 microseconds while a 512B high-speed packet has a transmission time of ~10 microseconds. Therefore, for SS enabled cores, if the DMA bus latency plus access time exceeds 2.1us, then set this parameter. For USB 2.0-only cores, set this parameter if the DMA access time exceeds 10us.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: DWC_USB3_MODE == 0 DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_MDBUS_ACCESS_GT21</p>
Global Transmit Threshold Configuration Register (GTXTHRCFG) Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the Global Transmit Threshold Configuration Register (GTXTHRCFG) power-on initialization value.</p> <p>This register contains threshold enable bits and threshold count fields. To enable USB packet thresholding upon power-on, then you need to update this value. For more information, see the "Memory Requirements" and "GTXTHRCFG" sections of the Databook and "TX/RX Data FIFO Sizes and TX/RX Threshold Control Register Settings" section of User Guide.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller.</p> <p>If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: ((DWC_USB3_MDBUS_ACCESS_GT21 == 0) ? 0x0 : 0x24080000)</p> <p>Enabled: (DWC_USB3_MODE == 0 DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GTXTHRCFG_INIT</p>

Label	Description
Global Receive Threshold Configuration Register (GRXTHRCFG) Power-On Initialization Value (0x0-fffffff)	<p>Specifies the Global Receive Threshold Configuration Register (GRXTHRCFG) power-on initialization value.</p> <p>This register contains threshold enable bits and threshold count fields. To enable USB packet thresholding upon power-on, then you need to update this value. For more information, see the "Memory Requirements" and "GRXTHRCFG" sections of the Databook and "TX/RX Data FIFO Sizes and TX/RX Threshold Control Register Settings" section of User Guide.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller.</p> <p>If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: ((DWC_USB3_MDBUS_ACCESS_GT21 == 0) ? 0x0 : 0x24400000)</p> <p>Enabled: (DWC_USB3_MODE == 0 DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GRXTHRCFG_INIT</p>

Label	Description
Global Control Register's (GCTL) Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the Global Control Register-1's (GCTL) power-on initialization value.</p> <p>bit [31:19]: Suspend Clock Scaledown value (suspend_clk_period/mac3_clk_period)</p> <p>bit [18]: Master filter bypass</p> <p>bit [17]: Bypass SetAddress in Device - For simulation purpose only</p> <p>bit [16]: Attempt 3 SS connection</p> <p>bit [15:14]: SOF Scaledown Value - Simulation purpose only</p> <p>bit [13:12]: Port Capability Direction</p> <ul style="list-style-type: none"> ■ 2'b01: for default Host operation (Power on value allowed only when DWC_USB3_MODE=1) ■ 2'b10: for default Device operation (Power on value allowed only when DWC_USB3_MODE!=1) <p>bit [11]: Core Soft Reset</p> <p>bit [10]: SOFITPSYNC - Core generated ITP from the ref_clk based counter</p> <p>bit [9]: Disable U1/U2 timer Scaledown</p> <p>bit [8]: Debug attach</p> <p>bit [7:6]: RAM Clock Select</p> <ul style="list-style-type: none"> ■ 2'b00: bus clock. ■ 2'b01: pipe clock; only in device mode. ■ 2'b10: pipe/2 clock in device mode; automatic switching between pipe/2 mac2_clk in host mode. ■ 2'b11: mac2_clk in device mode; automatic switching between pipe_clk/mac2_clk in host mode. (supported only when 8-bit UTMI/ULPI is used, and not supported in 16-bit UTMI mode). <p>Note: In host mode, it is recommended to have the ram_clk assigned to bus_clk. However, depending on the bandwidth requirement, you can optionally have the controller dynamically switch the ram_clk between bus_clk, pipe_clk and mac2_clk based on the available USB 2.0/3.0 clocks. For details on when this feature can be enabled in host mode, refer to "Minimum Clock Frequencies: bus_clk, ram_clk" section in the Databook.</p> <p>bit [5:4]: Scaledown Mode</p> <p>bit [3]: 1'b0: Enable Scrambling; 1'b1: Disable Scrambling;</p> <p>bit [2]: Enable 8 us LPFS for U2 exit</p> <p>bit [1]: Global Hibernation Enable</p> <p>bit [0]: 1'b0 - Enable internal clock gating; 1'b1 - Disable internal clock gating; If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register must be configured by your driver.</p> <p>For more information, see the "GCTL" section of the Databook.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: (DWC_USB3_MODE==0 ? 0x30c12004 : DWC_USB3_MODE==1 ? 0x30c01004 : DWC_USB3_MODE==2 ? ((DWC_USB3_EN_OTG_SS==1 DWC_USB3_EN_OTG == 1) ? 0x30c13004 : 0x30c12004) : 0x30c12004)</p> <p>Enabled: DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GCTL_INIT</p>

Label	Description
Enable Automotive ECC Generation and Checking for RAMs?	<p>Enables automotive ECC generation, correction, and checking. When enabled, the controller generates ECC, corrects 1-bit errors, and reports multi-bit errors to the software. The area of ECC logic is approximately "Number of RAMs * 2.5 KGates in 64-bit mode/4.5 Kgates in 128-bit mode". Single-bit error correction is transparent to the software. If a multi-bit error happens, it will be reported to the software through an event and the controller will go in to safe halted state. The only recovery is to reset the controller and re-enumerate.</p> <p>Note: This parameter can only be enabled when you have an automotive package license for USB 3.0 in addition to base licenses for USB 3.0 Host, Device, or DRD products.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: DWC_USB3_MODE == 0 DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_EN_ECC</p>

4.2 PHY Config Parameters

Table 4-2 PHY Config Parameters

Label	Description
Enable Pipelining on the Pipe Interface	<p>Decides the number of additional pipelines to be added in the Rx and Tx of the PIPE interface.</p> <p>This is normally not required because the DWC_usb3 controller registeres the PIPE inputs and outputs.</p> <p>For FPGA validation, this is not needed because DCMs and clock-phase shifts are used for data capture in FPGA mode. In ASIC mode, this is needed only when your SS PHY is placed far from your controller and you have problem in closing timing with PIPE interface. Because most on-chip PHY's PIPE interface runs at only 32bits at 125MHz, timing closer with no additional pipeline should not be an issue. Normally there is no reason to use more than 1 pipeline even if your PHY is placed far from your controller.</p> <p>In the src/pwrm/DWC_usb3_pwrm_u3piu module, the following additional pipeline registers control the pipelining: phy_pipe3_rx_stage1, phy_pipe3_rx_stage2, phy_pipe3_tx_stage1, and phy_pipe3_tx_stage2.</p> <p>Note: The following PIPE signals are not pipelined:</p> <ul style="list-style-type: none"> ■ pipe3_PowerPresent ■ pipe3_PhysStatus_async ■ pipe3_DataBusWidth <p>When you use this feature, these pipeline registers can be moved around during placement to break the routing timing delays. For more details, refer to the "USB 3.0 PHY Interface Unit (U3PIU)" section of the Databook.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No Additional Pipeline (0) ■ 1 stage of Additional Pipeline (1) ■ 2 stages of Additional Pipeline (2) <p>Default Value: No Additional Pipeline</p> <p>Enabled: DWC_USB3_SSPHY_INTERFACE ==1</p> <p>Parameter Name: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE</p>

Label	Description
High-Speed PHY Interface(s)?	<p>Specifies the High-Speed PHY interface(s). Choose both UTMI+ and ULPI if you are not sure whether a UTMI+ or ULPI off-chip PHY will be used in the product, or if you have an on-chip UTMI+ PHY and want to bring out the ULPI interface as a backup in case the on-chip PHY fails. Because the MAC connects to the UTMI+ interface, choosing an ULPI interface adds about 1.5K gates for the ULPI-to-UTMI+ conversion logic. If the UTMI+ and ULPI option is chosen, the software can select either interface.</p> <p>For more details, refer to "Integrating with USB 2.0 PHY" section in the User Guide and "USB 2.0 PHY Interface Unit (U2PIU)" section in the Databook.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Reserved (0) ■ UTMI+ (1) ■ ULPI (2) ■ UTMI+ and ULPI (3) <p>Default Value: (DWC_USB3_MODE==3)? 0 : 1</p> <p>Enabled: DWC_USB3_MODE !=3</p> <p>Parameter Name: DWC_USB3_HSPHY_INTERFACE</p>
PHY supports P3.CPM and P4 (EA Feature)?	<p>This option is for Synopsys PHY that supports P3.CPM and P4. Check whether your PHY supports this feature.</p> <p>To enable P3.CPM and P4, enable this parameter and program LUCTL[28]=1. This is an EA feature. Contact Synopsys before using this feature.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: (DWC_USB3_MODE != 3) && (DWC_USB3_SSOPHY_INTERFACE!=0)</p> <p>Parameter Name: DWC_USB3_SSOPHY_SUPPORT_P3CPM_P4</p>
Will PHY reference clock PLL be switched off (EA Feature)?	<p>Specifies whether you shut down the PLL of the PHY ref_clk during U3/L2. If this parameter is enabled,</p> <ul style="list-style-type: none"> ■ the suspend_clock is multiplexed with the ref_clk to generate the internal ref_clk. ■ the pmgt_ref_clk_off signal indicates when the ref_clk can be turned off. <p>If there are wakeup conditions from the USB, then the DWC_usb3 controller de-asserts the pmgt_ref_clk_off signal. This feature is supported only with Synopsys PHY.</p> <p>This is an EA feature. Contact Synopsys before using this feature.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: DWC_USB3_EN_PWROPT!=0 && DWC_USB3_SSOPHY_SUPPORT_P3CPM_P4==1 && DWC_USB3_EN_USB2_ONLY==0</p> <p>Parameter Name: DWC_USB3_REF_CLK_OFF</p>

Label	Description
Enable HSIC Support for USB 2.0 Ports?	<p>Selects the High Speed Interchip (HSIC) support for the USB 2.0 Ports. When enabled, the HSIC mode of operation or normal UTMI/ULPI mode of operation is selected by register and input port controls.</p> <p>The HSIC feature needs an HSIC Add-on license purchase.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: DWC_USB3_HSPHY_INTERFACE != 0 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_ENABLE_HSIC</p>
Enable UTMI-16bit Fast Turnaround Operation	<p>Selects the USB2.0 FAST Turnaround logic for UTMI 16-bit mode. Enable this parameter only if you require 16-bit UTMI support. In multi-port mode, this parameter can be enabled only if all your USB2.0 port clocks are synchronous to each other. If your USB2.0 port clocks are asynchronous to each other, then enabling this would cause functional failures.</p> <p>The USB specification defines the High-Speed turn around delay to be 192 bit times, which translates into 12 PHY clocks at 30 MHz. Typically, the PHY will take 7 PHY clocks, leaving 5 PHY clocks to the controller. Without this FAST_TAT feature enabled, for IN transactions, when operating at a RAM clock of 60 MHz the USB 3.0 device controller can take up to 7 clocks for its response. In this case, the UTMI PHY must consume no more than 5 clocks for the Receive End Delay and Transmit Start Delay, combined. If the UTMI PHY cannot guarantee that the total time consumed for the Receive End Delay and Transmit Start Delay is 5 PHY clock cycles or less, then the turn-around delay may be higher than 192 HS bit times. As a result, the USB host may timeout (if the device is connected to the 5th hub in a 5-tier hub topology) for a data packet returned by the USB 3.0 device controller for an IN token. Enabling this option optimizes/eliminates some registers in the device mac/ptl path and host mode clock-crossing FIFO, thereby achieving faster turn-around.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ If you select all posedge-clocked synchronizers (DWC_USB3_EN_SYNC_ALL_-POSEDGE=1) and you are not enabling this FAST_TAT feature, then the device mode minimum ram_clk frequency has to be 100 MHz to support the 5-clock turnaround time. If mac2_clk is 60 MHz and FAST_TAT_EN=1, enabling all posedge-clocked synchronizers does not increase the minimum ram_clk frequency of 60 MHz. If you plan to use ULPI mode only or UTMI 8-bit mode only, with ram_clk >= 100MHz, then do not enable this option. Depending on your UTMI PHY time consumption, your ram clock frequency, and your all posedge-clocked synchronizers, select this option. ■ For more details on how to set this parameter, see the "Recommendations for Selecting DWC_USB3_FAST_TAT_EN Value" section of the Databook. <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3</p> <p>Parameter Name: DWC_USB3_FAST_TAT_EN</p>

Label	Description
Enable UTMI PHY Vendor Control Interface?	<p>Specifies whether the UTMI+ PHY Vendor Control interface is enabled. For ULPI PHYs, since there are no additional ports Vendor Control is always enabled. For additional information, refer to "Global USB2 PHY Vendor Control Register(GUSB2PHYACCn)" section in Databook.</p> <p>Values:</p> <ul style="list-style-type: none">■ No (0)■ Yes (1) <p>Default Value: No</p> <p>Enabled: (DWC_USB3_HSPHY_INTERFACE==1 DWC_USB3_HSPHY_INTERFACE == 3) && DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_VENDOR_CTL_INTERFACE</p>

Label	Description
Global USB2 PHY Configuration Register's (GUSB2PHYCFG) Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the Global USB2 PHY Configuration Register's (GUSB2PHYCFG) power-on initialization value.</p> <p>The GUSB2PHYCFG register defines USB2 PHY configuration. The bits other than defined here must be 0. The different fields are:</p> <ul style="list-style-type: none"> bit[31]: UTMI PHY Soft Reset (PHYSoftRst) bit[30]: 1'b0: USB2 FREECLK doesn't exist; 1'b1: It exists; bit[29]: <ul style="list-style-type: none"> ■ 1'b0: LPM tokens to the ULPI PHY have 4 bytes with NOPID and EXTPID ■ 1'b1: LPM tokens to the ULPI PHY have 3 bytes with EXTPID only (opmode based EXTPID check) bit[28:27]: HSIC_CON_WIDTH_ADJ Set the value of this field to 1, 2, or 3, in order to increase the connect duration from 3 to 4, 5, or 6 times the strobe periods. bit[26]: INV_SEL_HSIC The application driver uses this bit to control the HSIC enable/disable function. This bit overrides and functionally inverts the if_select_hsic signal bit[25]: Reserved bit[24:22]: Host LS Rx-to-Tx packet gap (TurnaroundTime) - 0..7 corresponds to 2...5.5 bit times (increments of 0.5) bit[21:19]: Host LS Tx-to-Tx InterpacketGap - 0..7 corresponds to 2...5.5 bit times (increments of 0.5) bit[18]: ULPI External VBUS Indicator (ULPIExtVbusIndicator) bit[17]: ULPI External VBUS Drive (ULPIExtVb) bit[16]: Reserved bit[15]: ULPI Auto Resume (ULPIAutoRes) bit[14]: Reserved bit[13:10]: USB 2.0 Turnaround Time. 4'h5 - 16-bit UTMI+; 4'h9 - 8-bit UTMI+/ULPI; bit[9]: Transceiver Delay (XCVRDLY). 1'b0: No delay between xcvr_sel and tx_vld; 1'b1 - delay between xcvr_sel and tx_vld during chirp; bit[8]: Enable utmi_sleep_n and utmi_l1_suspend_n (EnblSlpM) <p>The application uses this bit to control utmi_sleep_n and utmi_l1_suspend_n assertion to the PHY in the L1 state.</p> <ul style="list-style-type: none"> ■ 1b0: Disable ■ 1b1: Enable <p>Note: When a ULPI interface is configured, enabling this bit results in a write to bit[7] of the ULPI Function Control register as long as the ULPI PHY supports writing to this bit when the SleepM is asserted. The encoding used for Function Control register bits [7:6] is as follows:</p> <p>If {Bit7, Bit6, sleep_n, l1_suspend_n, suspend_n} is,</p> <ul style="list-style-type: none"> ■ 5'b01111: state is normal working ■ 5'b00110: state is L2 Suspend ■ 5'b10101: state is L1 Suspend ■ 5'b11011: state is L1 Sleep <p>In host mode, when the DWC_USB3_FREECLK_USB2_EXIST is 0, the ULPI port0 is not suspended if any of the following conditions are met:</p>

Label	Description
Global USB2 PHY Configuration Register's (GUSB2PHYCFG) Power-On Initialization Value (0x0-ffffffff)...(cont.)	<ul style="list-style-type: none"> ■ At least one of the USB 2.0 ports is not suspended. Because the L1 Sleep requires quick wakeup time from the PHY, if the GUCTL1.OVRLD_L1_SUSP_COM is low, it is not treated as a suspend condition. If GUCTL1.OVRLD_L1_SUSP_COM is high, it is treated as a suspend condition and the clock can be suspended. The setting of this bit depends on how fast the Port0 PHY can recover from L1 sleep. ■ If DWC_USB3_SUSPEND_ON_DISCONNECT_EN is 1, the port disconnect condition with SE0 or SE1 on linestate is treated equivalent to the suspend condition. ■ At least one of the USB 3.0 ports is not in P3 ■ GUSB2PHYCFGn.EnblslpM is 0 for at least one of the ports Hardware LPM is enabled <p>bit[7]: Speed mode. 1'b0 USB2.0 High Speed (UTMI/ULPI); 1'b1 - USB1.1 Full Speed;</p> <p>bit[6]: When set, USB2.0 PHY enters Suspend mode if Suspend conditions are valid. 1'b0 - Disable; 1'b1 - Enable;</p> <p>bit[5]: 1'b0 - 6-Pin Unidirectional FS; 1'b1 - 3-Pin Bidirectional FS;</p> <p>bit[4]: 1'b0 - UTMI; 1'b1 - ULPI;</p> <p>bit[3]: UTMI Mode: 1'b0 8-bit UTMI; 1'b1 - 16-bit UTMI; This bit should be 1'b0 for ULPI Mode.</p> <p>bit[2:0]: HS/FS Timeout Calibration If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register can be configured by your driver. For more information, see the "GUSB2PHYCFG(#n)" section of the Databook.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: (((DWC_USB3_FREECLK_USB2_EXIST == 0) (DWC_USB3_MODE == 0)) ? 0x00102400 : 0x40102400)</p> <p>Enabled: (DWC_USB3_HSPHY_INTERFACE!=0 DWC_USB3_FSPHY_INTERFACE!=0) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GUSB2PHYCFG_INIT</p>

Label	Description
Global USB3 PIPE Control Register's (GUSB3PIPECTL) Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the Global USB3 PIPE Control Register's (GUSB3PIPECTL) power-on initialization value. The GUSB3PIPECTL register defines USB3 PIPE Control. For more details, refer to PIPE3 Specification. The bits other than defined here must be 0. The different fields are:</p> <ul style="list-style-type: none"> bit[31]: Reset SSPHY. After setting this bit to 1, the software needs to clear this bit. Refer to the Databook for how to use this bit. bit[29]: SSInactP3ok; 0: During link state SS.Inactive, put PHY in P2 (Default), 1: During link state SS.Inactive, put PHY in P3 bit[28]: DisRxDetP3 <ul style="list-style-type: none"> ■ 0: If PHY is in P3 and Core needs to perform receiver detection, Core will perform receiver detection in P3 (default) ■ 1: If PHY is in P3 and Core needs to perform receiver detection, Core will change PHY power state to P2 and then perform receiver detection. After receiver detection, Core will change PHY power state to P3. bit[27]: When set to 0, the controller does U1/U2/U3 exit in P0. (Default behavior); When set to 1, the controller does U1/U2/U3 exit in P1/P2/P3 respectively. This bit is added for SSPHY work-around where SSPHY injects glitch on pipe3_RxEleIdle (while receiving Ux exit LFPS) while pipe3_PowerDown change is in progress. This should be '0' for Synopsys-PHY. It is used by third-party SS PHY. bit[26]: When set, Downstream port U1 ping receive timeout becomes 500ms instead of 300ms. Min Ping.LFPS receive duration is 8ns (1 mac3_clk). For downstream port only. This should be '0' for Synopsys-PHY. It is used by third-party SS PHY. bit[25]:

Label	Description
Global USB3 PIPE Control Register's (GUSB3PIPECTL) Power-On Initialization Value (0x0-fffffff)...(cont.)	<p>If the U1/U2 LFPS handshake fails, LTSSM will transition from U1/U2 to Recovery instead of SS.Inactive. If Recovery fails, then the LTSSM can enter SS.Inactive. Enhancement only. Prevents inter-op issue if remote link does not do proper U1/U2 LFPS handshake.</p> <p>bit[24]: When set, the controller will always request PHY power change from P0 to P1/P2/P3 during U0 to U1/U2/U3 transition. If this bit is 0 and immediate Ux exit (remote initiated or Locally initiated) happens, Core may not request P1/P2/P3 power state change. This should be '1' for Synopsys-PHY. For non-Synopsys-PHY, check with your PHY vendor.</p> <p>bit[23]: If DWC_USB3_GUSB3PIPECTL_INIT[22] is set and link is in U3 or Rx.Detect state, the controller will start receiver detection on rising edge of this bit. For Downstream port only. This feature should not be used during normal operation.</p> <p>bit[22]: When set, the controller will not do receiver detection in U3/Rx.Detect. DWC_USB3_GUSB3PIPECTL_INIT[23] should be used to start receiver detection manually. For Downstream port only. This feature should not be enabled for normal operation. Contact Synopsys if have to use this feature.</p> <p>bit[21:19]: Delay P0 to P1/P2/P3 request when entering U1/U2/U3 until (DWC_USB3_GUSB3PIPECTL_INIT[21:19]*8) 8B10B error happens or Pipe3_RxValid drops. DWC_USB3_GUSB3PIPECTL_INIT[18] must be "1" to enable this functionality. This should be 3'h001 for Synopsys-PHY.</p> <p>bit[18]: Delay P1/P2/P3 transition when entering U1/U2/U3 until rxidle is 1 and Rxvalid is 0. 1 - Delay P1/P2/P3 transition; 0 - Do not delay P1/P2/P3 based on rxidle and Rxvalid.</p> <p>bit[17]: Enable USB3.0 SS PHY Suspend.</p> <p>bit[16:15]: PIPE Data Width. Only 32-bit or 16-bit is supported.</p> <ul style="list-style-type: none"> ■ 2'b00: 32 bits ■ 2'b01: 16 bits ■ Others: Reserved

Label	Description
Global USB3 PIPE Control Register's (GUSB3PIPECTL) Power-On Initialization Value (0x0-fffffff)...(cont..)	<p>bit[14]: In link state U2, abort receiver detection if remote partner starts U2 exit. This should be '0' for Synopsys-PHY. Used by 3rd party SSPHY.</p> <p>bit[13]: No Rx Detect If RxEleclidle Low, when set the controller will skip Rx Detection if pipe3_RxEleclidle is low. Skipping means waiting for the appropriate timeout and then repeating the operation.</p> <p>bit[12]: LFPS P0 Align. When set, the controller will deassert LFPS transmission on the same clock edge that it requests Phy power state 0 when exiting U1, U2, or U3 low power states. Otherwise, LFPS transmission will be asserted one clock earlier. In addition, the controller will request symbol transmission two pipe3_rx_pclks after the PHY asserts PhyStatus when the PHY has switched from the P1 or P2 to P0 state.</p> <p>bit[11]: P3 P2 Transitions OK. When set, the controller will transition directly from Phy power states P2 to P3, and from P3 to P2. Otherwise, P0 will always be entered between P2 and P3 as defined in the PIPE3 specification. For Synopsys-PHY this should be "0". Used by 3rd party SSPHY.</p> <p>bit[10]: P3 Exit Signal in P2. When set, the controller will always change the Phy power state to P2, before attempting a U3 exit handshake. For Synopsys-PHY this should be "0". Used by 3rd party SSPHY.</p> <p>bit[9]: LFPS Filter Value. When set, filter LFPS reception with pipe3_RxVaid in PHY power state P0, i.e ignore LFPS reception from the PHY unless both pipe3_Rxeclidle and pipe3_RxValid are deasserted.</p> <p>bit[8]:</p> <ul style="list-style-type: none"> ■ 1'b0 (Default): Enable 400us delay to start Polling LFPS after RX_DETECT to allow VCM offset to settle to proper level. ■ 1'b1: Disable 400us delay to start Polling LFPS after RX_DETECT to allow VCM offset to settle to proper level. <p>bit[7]:</p>

Label	Description
Global USB3 PIPE Control Register's (GUSB3PIPECTL) Power-On Initialization Value (0x0-fffffff)...(cont...)	<p>SSICEEn. This bit is not used.</p> <p>bit[6]: Tx Swing;</p> <ul style="list-style-type: none"> ■ 1'b0: Full Swing ■ 1'b1: Low Swing; <p>bit[5:3]: Tx Margin</p> <ul style="list-style-type: none"> ■ 3'b000: Normal ■ 3'b001: 800-1200mV Full Swing/400-700mV Half Swing ■ 3'b010, 3'b011: Vendor Defined ■ 3'b101 - 3'b111: 200-400mV Full Swing/100-200mV Half Swing <p>bit[2:1]: Tx Deemphasis</p> <ul style="list-style-type: none"> ■ 2'b00: -6DB ■ 2'b01: -3.5DB ■ 2'b10: No Deemphesis ■ 2'b11: Reserved; <p>bit[0]: PHY Elasticity Mode. 1'b0 - Nominal Half Full Buffer; 1'b1 - Nominal Empty Buffer; If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register can be configured by your driver. For more information, see the "GUSB3PIPECTL(#n)" section in the Databook and "Integrating with SuperSpeed PHY" section in the User Guide.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0x10c8002</p> <p>Enabled: (DWC_USB3_SSPHY_INTERFACE==1) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GUSB3PIPECTL_INIT</p>
pipe3_RxTermination signal hardware reset value?	<p>Determines the value of the pipe3_RxTermination signal while hardware reset is asserted.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0 (0) ■ 1 (1) <p>Default Value: 0</p> <p>Enabled: DWC_USB3_SSPHY_INTERFACE==1</p> <p>Parameter Name: DWC_USB3_PIPE_RXTERM_RESET_VAL</p>

Label	Description
Enable Receiver detection in PHY power state P3?	<p>For a USB 3.0 HUB or downstream port of Host, allow receiver detection in PHY power state P3.</p> <p>Check whether your PHY supports this feature. Even though this feature is supported in Synopsys-PHY, some of the existing Synopsys-PHY test chips do not support this. Check with Synopsys whether a particular test chip supports this feature before enabling this feature for FPGA hardware validation.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: (DWC_USB3_SSPhy_INTERFACE==1) && (DWC_USB3_MODE!=0)</p> <p>Parameter Name: DWC_USB3_RXDET_IN_P3_DS</p>
In Host Mode Enable USB2.0 suspend during Disconnect?	<p>For USB 2.0 ports, enable suspend during disconnect. This saves power when no devices are connected to a USB 2.0 port.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 1</p> <p>Enabled: (DWC_USB3_HSPHY_INTERFACE!=0 DWC_USB3_FSPHY_INTERFACE!=0) && (DWC_USB3_MBUS_TYPE!=4) && (DWC_USB3_MODE==1 DWC_USB3_MODE==2)</p> <p>Parameter Name: DWC_USB3_SUSPEND_ON_DISCONNECT_EN</p>
PIPE Interface is 32-bit or 16-bit?	<p>This parameter determines the PIPE DATA WIDTH. The default is 16-bit. If your PIPE interface is always 32-bits @125MHz, then mac3_clk will be set to pipe3_rx_pclk[0]. This avoids the clock dividers and multiplexers on the mac3_clk and simplifies the DFT and clock insertion. If you enable this parameter, make sure the DWC_USB3_GUSB3PIPECTL_INIT parameter bit[16:15] are 2'b00 to match this.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 32BIT (0) ■ 16BIT (1) <p>Default Value: = 0</p> <p>Enabled: DWC_USB3_MBUS_TYPE!=4 && DWC_USB3_SSPhy_INTERFACE ==1</p> <p>Parameter Name: DWC_USB3_PIPE_32BIT_ONLY</p>

4.3 Device Config Parameters

Table 4-3 Device Config Parameters

Label	Description
Basic Device Config	
Number of Device Mode Event Buffers	<p>Selects the number of event buffers in device mode. Each Event Buffer will also have a separate interrupt. If you have one processor in your SoC, then configuring multiple interrupts is not useful. This is used when you have multi-core or multiple processors in your SoC and you plan to do load balancing. There is no performance improvement expected when using multiple interrupts. In a SoC, mapping USB interrupts to one processor and another peripheral like Ethernet/SATA interrupt to another processor is alternate way of load balancing without making the drivers complex.</p> <p>In device mode, you could use multiple interrupts when there are multiple processors in your system. Different endpoint interrupts can be mapped to different processors. For example, the control endpoints can be mapped to one processor and the mass-storage bulk endpoints can be mapped to another endpoint. You still need communication between these two processors. The driver could get complex due to synchronization requirements of the multiple threads between the processors. Another use case in a multi-function device is to route Ethernet related endpoint interrupts to one processor and mass-storage related interrupts to another processor. This interrupt mapping is done static during endpoint configuration. In device mode dynamic mapping of interrupt is not supported. Refer to section "Multiple Device Interrupt Support" in the databook.</p> <p>In device mode, when you are using standard class driver interface (like the Linux BOT gadget driver), single interrupt is better for driver porting. The Synopsys mass-storage BOT and UASP reference driver supports only one interrupt.</p> <p>Values: 1, ..., 32 Default Value: 1 Enabled: DWC_USB3_MODE !=1 && DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_DEVICE_NUM_INT</p>

Label	Description
Number of Device Mode Endpoints (4-32)	<p>Specifies the number of device mode single directional endpoints, including control OUT and IN endpoints 0 which are always present.</p> <p>The DWC_usb3 controller provides flexible endpoint configuration, where an endpoint can be mapped to any USB IN or OUT endpoint. For example, if you need 3 bi-directional endpoints then you need to choose 6 single directional endpoint configuration. Choose the maximum possible number of endpoints that must be supported for all configurations and alternate settings. Specifying additional endpoints is not useful because the controller will not use the excess registers, resulting in increased gate count but no gain in performance.</p> <p>If area is not a concern, and you want a flexible design that can be used in multiple applications/projects, instantiate all 16 IN and 16 OUT endpoints. The default value of 8 endpoints reflects the UASP mass-storage class plus an ISOC application requirement: Control-OUT, Control-IN, Bulk-Data-OUT, Bulk-Data-IN, Bulk-Command-OUT, Bulk-Status-IN, ISOC-OUT, and ISOC-IN. The cost of an OUT/IN endpoint is 2.5/3.5Kgates plus transfer resource cache. For more guidance on configuring the number of device mode endpoints, see "Example Device Endpoint Mapping in Different Applications" section in the Databook.</p> <p>Values: 4, ..., 32 Default Value: 8 Enabled: DWC_USB3_MODE != 1 && DWC_USB3_MODE != 3 Parameter Name: DWC_USB3_NUM_EPS</p>
Number of Device Mode Active IN Endpoints (2-16)	<p>Specifies the maximum number of Device mode IN endpoints active at any time, including control IN endpoint 0, which is always present.</p> <p>This parameter determines the number of Device mode TxFIFOs to be instantiated and Tx RAM allocated. The default value of 4 IN endpoints reflects the UASP mass-storage class plus an ISOC application requirement: Control-IN, Bulk-Data-IN, Bulk-Status-IN, and ISOC-IN. For more guidance on configuring the number of device mode endpoints, see "Example Device Endpoint Mapping in Different Applications" section in the Databook.</p> <p>Values: 2, ..., 16 Default Value: 4 Enabled: DWC_USB3_MODE != 1 && DWC_USB3_MODE != 3 Parameter Name: DWC_USB3_NUM_IN_EPS</p>

Label	Description
Number of cached TRBs per Transfer(2-32)	<p>Selects the number of TRBs per transfer that can be cached within the controller. The cost of TRB cache is, 16 * Number of cached TRBs bytes.</p> <p>If your application performs scatter-gather operations where a single packet is split across more than two TRBs, then the TRB cache must also have at least one packet amount of TRBs (including Link TRB) for correct operation. For example, if a single 1 KB packet is scattered in 8 data buffers of each 128 bytes, then you would need a minimum of 9 TRBs cache (8 normal TRBs plus 1 Link TRB).</p> <p>For performance, it is recommended that the endpoint cache be able to hold one USB SS burst amount of TRBs. For example, if your system page size is 4KB and since the USB SS burst is 16, then 5 TRB cache is required for performance (4 normal TRBs plus 1 Link TRB). For ISOC applications, if your system bus latency is high, then it is recommended to have one BIinterval data worth of TRB cache.</p> <p>In addition, it is always recommended to enable the "Support Device/DbC Scatter-Gather Packets of 8 to 15 TRBs (DWC_USB3_DEV_EN_SCATTER_PACKETS_OF_8_TO_15_TRBS)" parameter in the "Advanced Configuration" section.</p> <p>Values: 2, ..., 32</p> <p>Default Value: 4</p> <p>Enabled: DWC_USB3_MODE != 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4</p> <p>Parameter Name: DWC_USB3_CACHE_TRBS_PER_TRANSFER</p>
Enable Device External Buffer Control?	<p>Enables Device External Buffer Control Sideband Controls.</p> <p>When enabled, dev_usb_outep_pkt_buff_avail and dev_usb_inep_pkt_buff_avail signals are used to indicate space/packet availability in the external Rx and Tx FIFOs for each OUT and IN endpoints, respectively. Note that the External Buffer Control is mainly used for debug endpoints and has usage restrictions. For more details, refer to "External Buffer Control" on page 557.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: ((DWC_USB3_MODE == 0) (DWC_USB3_MODE == 2)) && (DWC_USB3_MBUS_TYPE != 4)</p> <p>Parameter Name: DWC_USB3_EXT_BUFF_CONTROL</p>

Label	Description
Rx FIFO FIFO Depth	
MaxPacket Size in Bytes(1 to 1024)	<p>Selects Rx FIFO MaxPacket size in bytes. Since all OUT Endpoints use the same Rx FIFO, choose the largest packet size of the supported endpoint types.</p> <ul style="list-style-type: none"> ■ The MaxPacket size of a Bulk endpoint is 1024 bytes for SS and 512 bytes for HS. ■ The MaxPacket size of an Isoc endpoint is 1024 bytes when more than 1 packet is scheduled per Micro-Frame, otherwise this is application-specific. ■ The MaxPacket of an Interrupt endpoint is application-specific. For example, a mouse-like Human Interface Device (HID) uses a packet size of 4 bytes. <p>Values: 1, ..., 1024 Default Value: (DWC_USB3_EN_USB2_ONLY==1 && DWC_USB3_EN_ISOC_SUPT==0 ? 512: 1024) Enabled: ((DWC_USB3_MODE == 0) (DWC_USB3_MODE == 2)) Parameter Name: DWC_USB3_DEV_RXF_MAX_PACKET_SIZE</p>
Rx FIFO Size in Number of MaxPackets(1 to 16)	<p>Selects Number of MaxPackets for Rx FIFO All Device OUT Endpoints use the same Rx FIFO. Burst support requires a minimum 3 MaxPacket FIFO size; however, for SuperSpeed, 5 MaxPackets is recommended if the master DMA bus latency plus access time for a 1KB packet is larger than 2.1 microseconds. The default value is 3. However, for USB 2.0 only mode, the default is 2 in a low latency system (< 10 μs), 3 for ISOC, and 4 for non-ISOC in high-latency system.</p> <p>Values: 1, ..., 16 Default Value: (DWC_USB3_EN_USB2_ONLY==1 ? ((DWC_USB3_EN_ISOC_SUPT==1 && DWC_USB3_MDBUS_ACCESS_GT21==1) ? 3: (DWC_USB3_MDBUS_ACCESS_GT21==1 ? 4: 2)): DWC_USB3_MDBUS_ACCESS_GT21==1 ? 5: 3) Enabled: ((DWC_USB3_MODE == 0) (DWC_USB3_MODE == 2)) Parameter Name: DWC_USB3_DEV_RXF_NUM_MAX_PACKETS</p>
Rx FIFO Depth(0 to 8192)	<p>Selects Device Rx FIFO depth in MDWIDTH-bit words. Not recommended to configure less than 2 MaxPacket size. The default expression used is: (DWC_USB3_DEV_RXF_NUM_MAX_PACKETS * ((DWC_USB3_DEV_RXF_MAX_PACKET_SIZE + DWC_USB3_MBYTES - 1) / DWC_USB3_MBYTES) + DWC_USB3_DEV_RXFIFO_OK_SPACE_MARGIN * 4 / DWC_USB3_MBYTES) In addition to the packet storage, the Rx FIFO also stores up to three 8-byte setup packets and requires a 16-byte synchronization allowance. You can override this value depending upon your application requirements. Refer to "Memory Requirement" section of the Databook for more information.</p> <p>Values: ((4*DWC_USB3_DEV_RXFIFO_OK_SPACE) / DWC_USB3_MBYTES), ..., 8192 Default Value: $(DWC_USB3_DEV_RXF_NUM_MAX_PACKETS * ((DWC_USB3_DEV_RXF_MAX_PACKET_SIZE + DWC_USB3_MBYTES - 1) / DWC_USB3_MBYTES)) + ((DWC_USB3_DEV_RXFIFO_OK_SPACE_MARGIN * 4) / DWC_USB3_MBYTES)$ Enabled: ((DWC_USB3_MODE == 0) (DWC_USB3_MODE == 2)) Parameter Name: DWC_USB3_DEV_RXF_DEPTH</p>

Label	Description
IN Endpoint-0 FIFO Depth	
TxFIFO Depth(0 to 8192)	<p>Selects TxFIFO 0 depth in MDWIDTH-bit words The recommended value is: $(1 * (512 / \text{DWC_USB3_MBYTES} + \text{DWC_USB3_NPI_N}) + \text{DWC_USB3_NPI_N})$</p> <p>Values: $(1 * (64 / \text{DWC_USB3_MBYTES} + \text{DWC_USB3_NPI_N}) + \text{DWC_USB3_NPI_N}, \dots, 8192)$</p> <p>Default Value: $((\text{DWC_USB3_EN_USB2_ONLY} == 1) ? ((64 / \text{DWC_USB3_MBYTES} + \text{DWC_USB3_NPI_N}) + \text{DWC_USB3_NPI_N}) : ((512 / \text{DWC_USB3_MBYTES} + \text{DWC_USB3_NPI_N}) + \text{DWC_USB3_NPI_N}))$</p> <p>Enabled: $((\text{DWC_USB3_MODE} == 0) \text{ } (\text{DWC_USB3_MODE} == 2))$</p> <p>Parameter Name: <code>DWC_USB3_DEV_TXF0_DEPTH</code></p>
IN Endpoint-1 FIFO Depth	
MaxPacket Size in Bytes for TxFIFO-n (1 to 1024) (for n = 1; n <= 15)	<p>Selects TxFIFO 1 MaxPacket size in bytes</p> <ul style="list-style-type: none"> The MaxPacket of a Bulk endpoint is 1024 bytes (or 512 bytes in 2.0-only mode). The MaxPacket of a ISOC endpoint is 1024 bytes when more than 1 packet is scheduled per Micro-Frame, otherwise this is application specific. The MaxPacket of an Interrupt endpoint is application specific. For example, a mouse-like Human Interface Device (HID) uses a packet size of 4 bytes. <p>Values: 1, ..., 1024</p> <p>Default Value: $(\text{DWC_USB3_EN_USB2_ONLY} == 1 \& \& \text{DWC_USB3_DEV_TXF1_BURST_EN} == 0 ? 512 : 1024)$</p> <p>Enabled: $((\text{DWC_USB3_MODE} == 0) \text{ } (\text{DWC_USB3_MODE} == 2))$</p> <p>Parameter Name: <code>DWC_USB3_DEV_TXFn_MAX_PACKET_SIZE</code></p>
Burst Supported? (for n = 1; n <= 15)	<p>Selects whether TxFIFO 1 supports burst transfers (Note: In USB 2.0-only mode, applies to ISOC endpoints only).</p> <ul style="list-style-type: none"> In SS mode, burst-capable endpoints require a minimum 3 packet size FIFO. The reason is that the device must decide shortly after sending packet-1 whether to set the End of Burst (EOB) flag in packet-2's header, which depends on the presence of packet-3. For non-burst endpoints, a 2 MaxPacket size FIFO allocation is recommended. <p>Values:</p> <ul style="list-style-type: none"> No (0) Yes (1) <p>Default Value: $(\text{DWC_USB3_EN_USB2_ONLY} == 1 \& \& \text{DWC_USB3_EN_ISOC_SUPT} == 0 ? 0 : 1)$</p> <p>Enabled: $((\text{DWC_USB3_MODE} == 0) \text{ } (\text{DWC_USB3_MODE} == 2))$</p> <p>Parameter Name: <code>DWC_USB3_DEV_TXFn_BURST_EN</code></p>

Label	Description
TxFIFO-n Depth in Number of MaxPackets (1 to 16) (for n = 1; n <= 15)	<p>Selects Number of MaxPackets for TxFIFO 1 All Device IN Endpoints have a dedicated TxFIFO.</p> <ul style="list-style-type: none"> ■ Burst support requires a minimum 3 MaxPacket FIFO size; however, for Super-Speed, 5 MaxPackets is recommended if the master DMA bus latency plus access time for a 1KB packet is larger than 2.1 microseconds. ■ While the recommended value is 3 for burstable endpoints, the recommended value is 2 for non-burstable endpoints (no more than 1 packet scheduled per micro-frame). ■ However, for USB 2.0 only mode, the default is 2 in low latency system (< 10 uS), 3 for ISOC, and 4 for non-ISOC high-latency system. <p>Values: 1, ..., 16</p> <p>Default Value: (DWC_USB3_EN_USB2_ONLY==1 ? ((DWC_USB3_DEV_TXF1_BURST_EN==1 && DWC_USB3_MDBUS_ACCESS_GT21==1) ? 3: (DWC_USB3_MDBUS_ACCESS_GT21==1 ? 4: 2)): (DWC_USB3_DEV_TXF1_BURST_EN==0 ? 2: (DWC_USB3_MDBUS_ACCESS_GT21==1 ? 5: 3)))</p> <p>Enabled: ((DWC_USB3_MODE == 0) (DWC_USB3_MODE == 2))</p> <p>Parameter Name: DWC_USB3_DEV_TXFn_NUM_MAX_PACKETS</p>
TxFIFO-n Depth (0 to 8192) (for n = 1; n <= 15)	<p>Selects TxFIFO 1 depth in MDWIDTH-bit words The default expression used is: DWC_USB3_DEV_TXF1_NUM_MAX_PACKETS * ((DWC_USB3_DEV_TXF1_MAX_PACKET_SIZE + DWC_USB3_MBYTES - 1) / DWC_USB3_MBYTES + DWC_USB3_NPI_N) + DWC_USB3_NPI_N In addition to the packet storage, the TxFIFO also stores additional DWC_USB3_MDWIDTH bytes status for each packet and DWC_USB3_MDWIDTH bytes status for each endpoint. You can override this value depending upon your application requirements.</p> <p>Values: 0, ..., 8192</p> <p>Default Value: (DWC_USB3_DEV_TXF1_NUM_MAX_PACKETS*((((DWC_USB3_DEV_TXF1_MAX_PACKET_SIZE+DWC_USB3_MBYTES)-1)/DWC_USB3_MBYTES)+DWC_USB3_NPI_N))+DWC_USB3_NPI_N</p> <p>Enabled: ((DWC_USB3_MODE == 0) (DWC_USB3_MODE == 2))</p> <p>Parameter Name: DWC_USB3_DEV_TXFn_DEPTH</p>

4.4 Host Config Parameters

Table 4-4 Host Config Parameters

Label	Description
Enable xHCI Debug Capability?	<p>Enables xHCI Debug Capability</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: ((DWC_USB3_MODE == 1) (DWC_USB3_MODE == 2)) && (DWC_USB3_MDWIDTH != 32) && (DWC_USB3_EN_USB2_ONLY == 0)</p> <p>Parameter Name: DWC_USB3_EN_DBC</p>
Number of Host Interrupters (1-16)	<p>Specifies the number of supported interrupters, each of which requires 100 bytes of internal RAM; in addition, depending on configuration, increasing the number of interrupters may cause the number of required scratchpad buffers to increase from 2 to 3. It is recommended not to select more than 8 interrupts; the future releases will remove support for more than 8 interrupts.</p> <p>If you have one processor in your SoC, then configuring multiple interrupts is not useful. This is used when you have multi-core or multiple processors in your SoC and you plan to do load balancing. There is no performance improvement expected when using multiple interrupts. In a SoC, mapping USB interrupts to one processor and another peripheral like Ethernet/SATA interrupt to another processor is an alternate way of load balancing without increasing complexity of the drivers.</p> <p>In host mode, this feature is intended for multi-core processors in the PC. For an embedded system, single interrupt is recommended. The open-source Linux driver does enable multiple interrupts when HW supports this feature, but likely does not actively use them.</p> <p>If the "Separate Descriptor Queues" feature is enabled, the maximum number of interrupts supported is 8 since each interrupt needs an additional Event Queue FIFO which increases area.</p> <p>Values: 1, ..., 16</p> <p>Default Value: 1</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_HOST_NUM_INTERRUPTER_SUPT</p>

Label	Description
Enable Separate Descriptor Queues	<p>Specifies whether to implement separate descriptor fetch and writeback queues for SS, HS, and FS/LS bus instances.</p> <p>It is recommended to enable this parameter; option to not enable this feature will be removed in future releases.</p> <p>When system bus latencies are unpredictable and less than the total bandwidth of the USB SS, HS, and FS/LS bandwidth, this feature provides fairness to the HS and FS/LS traffic and will improve HS and FS/LS periodic endpoints QoS. Even though there are separate Rx and Tx Data FIFOs for SS, and HS, and FS/LS bus instances, high traffic in SS bus instance could impact HS and FS/LS QoS due to the common descriptor queue and due to the fixed DMA arbitration priority of RXQ, DWQ, then DFQ queues. With this enhancement, since each bus instance has its own RxQ, DWQ, and DFQ, the SS traffic does not affect other bus instances QoS.</p> <p>This feature, which requires additional RAM, is available for 3.0 Host/DRD with MDWIDTH=64 or 128 and at most 8 interrupts. Note that when this feature is enabled, in addition to the descriptor queues each Event (interrupt) also needs an Event Queue FIFO in order to queue the events from different bus instances in the order received.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: ((DWC_USB3_MODE==1 DWC_USB3_MODE==2) && DWC_USB3_MBUS_TYPE!=4 && DWC_USB3_MDWIDTH>=64 && DWC_USB3_EN_USB2_ONLY==0 && DWC_USB3_HOST_NUM_INTERRUPTER_SUPT<=8) ? 1: 0</p> <p>Enabled: (DWC_USB3_MODE==1 DWC_USB3_MODE==2) && DWC_USB3_MBUS_TYPE!=4 && DWC_USB3_MDWIDTH>=64 && DWC_USB3_EN_USB2_ONLY==0 && DWC_USB3_HOST_NUM_INTERRUPTER_SUPT<=8</p> <p>Parameter Name: DWC_USB3_EN_SEPARATE_DESC_QUEUES</p>
Number of USB2.0 (HS/FS/LS) Root Hub Ports (1-15)	<p>Specifies the number of USB2 Root Hub ports.</p> <p>The area of a USB 2.0 port is about 12K gates.</p> <p>Values: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15</p> <p>Default Value: 1</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_HOST_NUM_U2_ROOT_PORTS</p>
Number of USB3 SS Root Hub Ports (1-15)	<p>Specifies the number of USB 3.0 Root Hub ports.</p> <p>The area of a USB3 port is about 52K gates.</p> <p>Values: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15</p> <p>Default Value: 1</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && (DWC_USB3_EN_USB2_ONLY==0)</p> <p>Parameter Name: DWC_USB3_HOST_NUM_U3_ROOT_PORTS</p>

Label	Description
Number of Devices Supported (64-127)	<p>Specifies the number of devices supported.</p> <p>The value you choose does not increase the DWC_usb3's gate count, but the memory requirements increase to store context information. Thirty-two (32) bytes of internal RAM is required for each device supported. For example, the Gold Tree testing used at USB-IF Host compliance has 9 Hubs, 2 WebCams, 1 Printer, 2 keyboards, 3 mass storage devices, 1 mouse, 1 headset. This configuration has a total of 19 devices.</p> <p>For more information, see "Area, Speed, Power, DFT, and Performance" appendix in the Databook.</p> <p>Values: 64, 127</p> <p>Default Value: ((DWC_USB3_EN_USB2_ONLY == 0) ? 64 : 64)</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_NUM_DEVICE_SUPT</p>
Number of Periodic Endpoints Supported (32-510)	<p>Specifies the number of periodic endpoints for all USB Bus-Instance (SS, HS, and FS/LS combined).</p> <p>Twenty (20) bytes of internal RAM are required for each periodic endpoint. Increasing the value of this parameter may increase the number of scratchpad buffers needed by the hibernation feature. For example, if DWC_USB3_HOST_NUM_PERIODIC_EP is set to its maximum value (510), the number of scratchpad buffers required may be as much as 4, whereas if DWC_USB3_HOST_NUM_PERIODIC_EP is set to 64, the number of scratchpad buffers should not exceed 2.</p> <p>Most Periodic device like WebCam, Keyboard, Speaker, Hub, Mice, Game Controller, etc, use one to two periodic endpoints. On the other hand, some game controllers may use 7 to 15 periodic endpoints. For example, the Gold Tree testing used at USB-IF Host compliance has 9 Hubs, 2 WebCams, 1 Printer, 2 keyboards, 3 mass storage devices, 1 mice, and 1 headset. This configuration has a total of 20 periodic endpoints.</p> <p>It limits the maximum number of periodic endpoints that can be configured by the application using the EP configuration command. When the application adds new periodic endpoints in the EP configuration command, the Host Controller allocates an unused entry to each endpoint. If there is no more free space, the Host Controller returns a "Resource Error" event TRB. Note, even if the application drops the endpoint in the same EP configuration command as "add EP", the Host Controller does not remove the endpoint information for the dropped endpoint immediately. This is because the command can get rejected because of bandwidth issues. The new added endpoint still needs a free cache slot and not the slot from a "dropped EP" in the same command.</p> <p>Values: 32, ..., 510</p> <p>Default Value: ((DWC_USB3_EN_USB2_ONLY == 0) ? 32 : 32)</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_HOST_NUM_PERIODIC_EP</p>

Label	Description
Does your USB 2.0 PHY Provides a free running PHY clock?	<p>Specifies whether your USB 2.0 PHY provides a free running PHY clock which is active when the clock control input is active.</p> <p>If your USB 2.0 PHY provides the free running PHY clock, it should be connected to "utmi_clk[0]" input. Rest of the "utmi_clk[n]" would be getting the respective port clocks. The Port-0 clock will be used by the controller for generating internal mac2 clock.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: (((DWC_USB3_MODE==1) (DWC_USB3_MODE==2)))? 1 : 0</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_FREECLK_USB2_EXIST</p>
Number of SuperSpeed USB Bus-Instances (1-4)	<p>Specifies the number of SuperSpeed USB Bus-Instances.</p> <p>This is recommended only if your SoC master-bus/memory bandwidth can sustain the USB3.0 peek bandwidth for all the USB3.0 SS Bus-Instances (DWC_USB3_NUM_SS_USB_INSTANCES * 8Gbs) and has less bus access latency. The USB 3.0 controller pipelines only up to 2 packets reads and in high latency system the system bus bandwidth may not meet the USB bandwidth required due to the pipelining limit. In such case, it is recommended to instantiate multiple single port controllers.</p> <p>Number of SuperSpeed USB Bus-Instances cannot be greater than USB 3.0 SS Root Hub ports (DWC_USB3_NUM_SS_USB_INSTANCES <= DWC_USB3_HOST_NUM_U3_ROOT_PORTS).</p> <p>When the number of SuperSpeed USB Bus-Instances is equal to the number of USB 3.0 ports, you can do concurrent USB 3.0 transfers on each port.</p> <p>For example, if you have 4 ports and 4 SuperSpeed USB Bus-Instances, then in concurrent mode your effective data rate is 32 Gbps (8 Gbps * 4). The area overhead of a SuperSpeed USB Bus-Instance is about 145K gates. When compared to the area of the PHY, which is about 400K gates per port, this is small, but the performance increase is multifold. For example, if you have 4 ports, 1 port can be connected to a USB 3.0 drive, another port can be connected to a display, and 2 more ports can be used as general purpose I/O interface. Here, each port has a guaranteed throughput of 8 Gbps (or 4 Gbps simplex).</p> <p>Values: 1, 2, 3, 4</p> <p>Default Value: 1</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_EN_USB2_ONLY==0 && DWC_USB3_HOST_NUM_U3_ROOT_PORTS != 1 && DWC_USB3_MDBUS_ACCESS_GT21 == 0</p> <p>Parameter Name: DWC_USB3_NUM_SS_USB_INSTANCES</p>

Label	Description
Number of High-Speed USB Bus-Instances (1-4)	<p>Specifies the number of High-Speed USB Bus-Instances. When the number of High-Speed USB Bus-Instances is equal to the number of root ports, you can do concurrent High-Speed transfers on each port. For example, if you have 4 ports and 4 High-Speed USB Bus-Instances, then in concurrent mode your effective data rate is 1.92 Gbps (480 Mbps * 4). The area overhead of a High-Speed USB Bus-Instance is about 55K gates.</p> <p>Number of high-speed USB Bus-Instances cannot be greater than USB 2.0 Root Hub ports (DWC_USB3_NUM_HS_USB_INSTANCES <= DWC_USB3_HOST_NUM_U2_ROOT_PORTS).</p> <p>Values: 1, 2, 3, 4 Default Value: 1 Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_HOST_NUM_U2_ROOT_PORTS != 1 Parameter Name: DWC_USB3_NUM_HS_USB_INSTANCES</p>
Number of Cached TRBs for Each Debug Capability Endpoint (16-126)	<p>It is recommended not to reduce the size from default value.</p> <p>Specifies the number of TRBs that can be cached in the controller for each cached endpoint in the Debug Capability.</p> <p>16 bytes of internal RAM are allocated for each cached TRB. The TRB cache must be large enough to hold TRBs (including Link TRBs) for at least one maximum-packet-size packet. For better performance or when system latency is large, TRB cache for maximum-burst-size of packets is recommended.</p> <p>Values: 16, ..., 126 Default Value: 16 Enabled: DWC_USB3_EN_DBC == 1 Parameter Name: DWC_USB3_DBC_TRBS_PER_TRANSFER</p>
Number of Cached Endpoints for Each SuperSpeed USB Instance (1-32)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the number of SuperSpeed endpoints that can be cached for each SuperSpeed USB Bus-Instance. (136 + DWC_USB3_HOST_NUM_CACHE_TRB_PER_SS_EP * 16) bytes of internal RAM are allocated for each super speed endpoint cache.</p> <p>Values: 1, ..., 32 Default Value: ((DWC_USB3_MDBUS_ACCESS_GT21 == 0) ? 8 : 10) Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && (DWC_USB3_EN_USB2_ONLY==0) Parameter Name: DWC_USB3_HOST_NUM_CACHE_EP_PER_SS_INSTANCE</p>

Label	Description
Number of Cached TRBs for Each Cached SuperSpeed Endpoint (2-126)	<p>Recommended not to reduce the size from default value. Specifies the number of TRBs that can be cached in the controller for each cached SuperSpeed endpoint. 16 bytes of internal RAM are allocated for each cached TRB.</p> <p>The TRB cache must be large enough to hold TRBs (including Link TRBs and Event Data TRBs) for at least one maximum burst size of packets. For example, if your system page size is 4 KB (one TRB holds up to four 1 KB packets) and since USB 3.0 burst size is 16 packets, you need at least a cache size of 6 TRBs (4 data TRBs, 1 Link TRB, and 1 Event Data TRB). If your application performs scatter-gather operations where a single packet is split across more than two TRBs then the cache size should account for this. For example, if a single 1 KB packet is scattered in 4 data buffers of each 256 bytes each then you would need a minimum of 66 TRB cache (64 normal TRB, 1 Link TRB, 1 Event Data TRB). If periodic endpoints exit, the TRB cache must be large enough to hold TRBs (including Link TRBs and Event Data TRBs) for at least one service interval.</p> <p>Values: 2, ..., 126</p> <p>Default Value: 16</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && (DWC_USB3_EN_USB2_ONLY==0)</p> <p>Parameter Name: DWC_USB3_HOST_NUM_CACHE_TRB_PER_SS_EP</p>
Number of Cached Endpoints for Each High-Speed USB Instance (1-32)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the number of High-Speed endpoints that can be cached for each High-Speed USB Bus-Instance. (120 + DWC_USB3_HOST_NUM_CACHE_TRB_PER_HS_EP * 16) bytes of internal RAM are allocated for each high speed endpoint cache. If SS is enabled and if the system bus latency is more than 2.1uS, then at least 20 endpoint cache and the following "Number of Cached TRBs for Each Cached High-Speed Endpoint" parameter size can be reduced to 10 from 20 for optimized RAM size.</p> <p>Values: 1, ..., 32</p> <p>Default Value: ((DWC_USB3_EN_USB2_ONLY == 1) ? 8 : (DWC_USB3_MDBUS_ACCESS_GT21 == 0) ? 10 : 20)</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_HOST_NUM_CACHE_EP_PER_HS_INSTANCE</p>

Label	Description
Number of Cached TRBs for Each Cached High-Speed Endpoint (2-126)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the number of TRBs that can be cached in the controller for each cached High-Speed endpoint. 16 bytes of internal RAM are allocated for each cached TRB. The TRB cache must be large enough to hold TRBs (including Link TRBs and Event Data TRBs) for at least one maximum burst size of packets. For example, if your system page size is 4 KB (one TRB holds up to four 1 KB packets) and since USB 2.0 burst size is 3 packets, you need at least a cache size of 3 TRBs (1 data TRBs, 1 Link TRB, and 1 Event Data TRB). If your application performs scatter-gather operations where a single packet is split across more than two TRBs then the cache size should account for this. For example, if a single 1 KB packet is scattered in 4 data buffers of each 256 bytes each then you would need a minimum of 14 TRB cache ($3 * 4$ normal TRB, 1 Link TRB, 1 Event Data TRB). If periodic endpoints exit, The TRB cache must be large enough to hold TRBs (including Link TRBs and Event Data TRBs) for at least one service interval.</p> <p>Values: 2, ..., 126</p> <p>Default Value: ((DWC_USB3_EN_USB2_ONLY == 1) ? 8 : (DWC_USB3 MDBUS_ACCESS_GT21 == 0) ? 8 : 10)</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_HOST_NUM_CACHE_TRB_PER_HS_EP</p>
Number of Cached Endpoints for Each Full/Low-Speed USB Instance (1-32)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the number of Full/Low-Speed endpoints that can be cached for each Full/Low-Speed USB Bus-Instance. (120 + $DWC_USB3_HOST_NUM_CACHE_TRB_PER_FSLS_EP * 16$) bytes of internal RAM are allocated for each full/low speed endpoint cache.</p> <p>Values: 1, ..., 32</p> <p>Default Value: ((DWC_USB3_EN_USB2_ONLY == 1) ? 2 : (DWC_USB3 MDBUS_ACCESS_GT21 == 0) ? 2 : 4)</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_HOST_NUM_CACHE_EP_PER_FSLS_INSTANCE</p>
Number of Cached TRBs for Each Cached Full-Low-Speed Endpoint (2-126)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the number of TRBs that can be cached in the controller for each cached Full/Low-Speed endpoint. 16 bytes of internal RAM are allocated for each cached TRB.</p> <p>The TRB cache must be large enough to hold TRBs (including Link TRBs and Event Data TRBs) for at least one maximum packet size of packets. If your application performs scatter-gather operations where a single packet is split across more than two TRBs then the cache size should account for this. For example, if a single 1 KB packet is scattered in 4 data buffers of each 256 bytes each then you would need a minimum of 6 TRB cache (4 normal TRB, 1 Link TRB, 1 Event Data TRB). If periodic endpoints exit, The TRB cache must be large enough to hold TRBs (including Link TRBs and Event Data TRBs) for at least one service interval.</p> <p>Values: 2, ..., 126</p> <p>Default Value: ((DWC_USB3_EN_USB2_ONLY == 1) ? 8 : (DWC_USB3 MDBUS_ACCESS_GT21 == 0) ? 8 : 10)</p> <p>Enabled: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Parameter Name: DWC_USB3_HOST_NUM_CACHE_TRB_PER_FSLS_EP</p>

Label	Description
Size of the SuperSpeed RxFIFO in Number of 1024-byte Packets - Per SS Bus-Instance (1-32)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the size of the SuperSpeed RxFIFO in number of 1024-byte packets per SS Bus-Instance.</p> <p>Each SuperSpeed USB Bus-Instance would require one RxFIFO. For example, if you have 2 SS Bus-Instances and chose 3 for this parameter, then 6144 bytes of RAM is allocated for SS RxFIFO. Each 1024 bytes provides a 2.1us latency tolerance. If the system bus latency is more than 2.1uS, then at least a 5 to 6 packet buffer is recommended. For more information, see the "Memory Requirements" section in the Databook.</p> <p>Values: 1, ..., 32</p> <p>Default Value: ((DWC_USB3_MBUSB_ACCESS_GT21 == 0) ? 3 : 5)</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUSB_TYPE!=4 && (DWC_USB3_EN_USB2_ONLY==0)</p> <p>Parameter Name: DWC_USB3_NUM_RXF_SS_PKTS</p>
Size of the High-Speed RxFIFO in Number of 1024-byte Packets - Per HS Bus-Instance (1-16)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the size of the High-Speed RxFIFO in number of 1024-byte packets per HS Bus-Instance.</p> <p>Each High-Speed USB Bus-Instance requires one RxFIFO. For example, if you have 2 HS Bus-Instances and chose 2 for this parameter, then 4096 bytes of RAM is allocated for HS RxFIFO. Since HS ISOC max packet size is 1024, at least 2048 bytes should be allocated for HS RxFIFO. Each 1024 bytes provides 20us latency tolerance.</p> <p>For more information, see the "Memory Requirements" section of the Databook.</p> <p>Values: 1, ..., 16</p> <p>Default Value: ((DWC_USB3_MBUSB_ACCESS_GT21 == 0) ? 2 : 3)</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUSB_TYPE!=4</p> <p>Parameter Name: DWC_USB3_NUM_RXF_HS_PKTS</p>
Size of the Full-Speed/Low-Speed RxFIFO in Number of 1024-byte Packets - Per FS/LS Bus-Instance (1-4)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the size of the Full-Speed/Low-Speed RxFIFO in number of 1024-byte packets per FS/LS Bus-Instance.</p> <p>Each Full-Speed/Low-Speed USB Bus-Instance requires one RxFIFO. For example, if you have 1 FS/LS Bus-Instances and chose 1 for this parameter, then 1024 bytes of RAM is allocated for FS/LS RxFIFO. Since FS ISOC max payload size is 1023, at least 1024 bytes must be allocated for FS/LS RxFIFO.</p> <p>For more information, see the "Memory Requirements" section of the Databook.</p> <p>Values: 1, ..., 4</p> <p>Default Value: 1</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUSB_TYPE!=4</p> <p>Parameter Name: DWC_USB3_NUM_RXF_FSLS_PKTS</p>

Label	Description
Size of the SuperSpeed TxFIFO in Number of 1024-byte Packets - Per SS Bus-Instance (1-32)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the size of the SuperSpeed TxFIFO in number of 1024-byte packets per SS Bus-Instance.</p> <p>Each SuperSpeed USB Bus-Instance requires one TxFIFO. For example, if you have 2 SS Bus-Instances and chose 4 for this parameter, then 8182 bytes of RAM is allocated for SS TxFIFO. In a PCIe application, the read access has more latency than write. It is recommended to allocate more TxFIFO than RxFIFO which gives better area and performance trade off. Each 1024byte provides a 2.1us latency tolerance. If the system bus latency is more than 2.1uS, then at least a 5 to 6 packet buffer is recommended.</p> <p>For more information, see the "Memory Requirements" section of the Databook.</p> <p>Values: 1, ..., 32</p> <p>Default Value: ((DWC_USB3_MDBUS_ACCESS_GT21 == 0) ? 4 : 5)</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4 && (DWC_USB3_EN_USB2_ONLY==0)</p> <p>Parameter Name: DWC_USB3_NUM_TXF_SS_PKTS</p>
Size of the High-Speed TxFIFO in Number of 1024-byte Packets - Per HS Bus-Instance (1-16)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the size of the High-Speed TxFIFO in number of 1024-byte packets per HS Bus-Instance.</p> <p>Each High-Speed USB Bus-Instance would require one TxFIFO. For example, if you have 2 HS Bus-Instances and chose 2 for this parameter, then 4096 bytes of RAM is allocated for HS TxFIFO. Because HS ISOC max packet size is 1024, at least 2048 bytes must be allocated for HS TxFIFO. Each 1024 bytes provides 20us latency tolerance.</p> <p>For more information, see the "Memory Requirements" section of the Databook.</p> <p>Values: 1, ..., 16</p> <p>Default Value: ((DWC_USB3_MDBUS_ACCESS_GT21 == 0) ? 2 : 3)</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_NUM_TXF_HS_PKTS</p>
Size of the Full-Speed/Low-Speed TxFIFO in Number of 1024-byte Packets - Per FS/LS Bus-Instance (1-4)	<p>Recommended not to reduce the size from default value.</p> <p>Specifies the size of the Full-Speed/Low-Speed TxFIFO in number of 1024-byte packets per FS/LS Bus-Instance.</p> <p>Each Full-Speed/Low-Speed USB Bus-Instance requires one TxFIFO. For example, if you have 1 FS/LS Bus-Instances and chose 1 for this parameter, then 1024 bytes of RAM is allocated for FS/LS TxFIFO. Since FS ISOC max payload size is 1023, at least 1024 bytes should be allocated for FS/LS TxFIFO.</p> <p>For more information, see the "Memory Requirements" section of the Databook.</p> <p>Values: 1, ..., 4</p> <p>Default Value: 1</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_NUM_TXF_FSLS_PKTS</p>

4.5 Hub Config Parameters

Table 4-5 Hub Config Parameters

Label	Description
Number of USB 3.0 Capable Hub Ports (1-15)	<p>Specifies the number of USB 3.0 capable down-stream Hub ports.</p> <p>Values: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15</p> <p>Default Value: 4</p> <p>Enabled: DWC_USB3_MODE == 3</p> <p>Parameter Name: DWC_USB3_HUB_NUM_U3_PORTS</p>
Enable Xilinx Serial Flash-RAM/PROM Support?	<p>Specifies whether Xilinx Serial Flash-RAM/PROM support is required to load Hub Descriptors.</p> <p>The memory requirements for the Hub descriptor is 256bytes (2048bits). Refer to Xilinx website for a Serial Flash/PROM that meets your requirements. Most Xilinx Serial Flash/PROM are much larger in size than the required 2048bits.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: DWC_USB3_MODE == 3</p> <p>Parameter Name: DWC_USB3_EN_HUB_SFLASH_SUPT</p>
Mac Clock (125MHz) to Xilinx Serial Flash-RAM/PROM Clock Divide Factor?	<p>Specifies the MAC clock(125MHz) to Serial Flash/PROM clock divide factor. The typical Xilinx Serial Flash/PROM memory operating frequency range is 2.5 to 20 MHz. Refer to the Xilinx website for a Serial Flash/PROM that meets your requirements.</p> <p>Values: 4, ..., 128</p> <p>Default Value: 16</p> <p>Enabled: DWC_USB3_MODE == 3</p> <p>Parameter Name: DWC_USB3_HUB_SFLASH_DIVFACTOR</p>
Enable JTAG Debug support?	<p>Specifies whether JTAG debug interface is enabled.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: (DWC_USB3_MODE==3)? 1 : 0</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_JTAG_INTERFACE</p>
Number of Headers in Each Rx Header Buffer?	<p>Specifies the number of Headers in each Rx Header Buffers.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 4 (4) ■ 8 (8) <p>Default Value: 4</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_HUB_NUM_RXHEADERS</p>

Label	Description
Number of Headers in Each Tx Header Buffer?	<p>Specifies the number of Headers in each Tx Header Buffers</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 4 (4) ■ 8 (8) <p>Default Value: 4</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_HUB_NUM_TXHEADERS</p>
Number of Packets in Upstream Rx Data Packet Buffer?	<p>Specifies the number of packets in Upstream Rx Data Packet Buffer. Each packet requires 1024 bytes of space.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1 (1) ■ 2 (2) ■ 3 (3) ■ 4 (4) ■ 5 (5) ■ 6 (6) ■ 7 (7) ■ 8 (8) <p>Default Value: 1</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_HUB_NUM_UPSTRMRX_DATAPKTS</p>
Number of Packets in Upstream Tx Data Packet Buffer?	<p>Specifies the number Packets in Upstream Tx Data Packet Buffer. Each packet needs 1024 bytes of space.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1 (1) ■ 2 (2) ■ 3 (3) ■ 4 (4) ■ 5 (5) ■ 6 (6) ■ 7 (7) ■ 8 (8) <p>Default Value: 1</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_HUB_NUM_UPSTRMTX_DATAPKTS</p>

Label	Description
Depth of the Descriptor ROM or Serial-Flash (in 32-bit Dwords)?	<p>Specifies the depth of the Descriptor ROM or Serial-Flash in 32-bit Dwords. For example, a value of 64 represents a 256byte (64 * 4) ROM.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 64 (64) ■ 128 (128) ■ 256 (256) <p>Default Value: 64</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_HUB_DESC_DEPTH</p>
Starting Depth of the SS Descriptors in the ROM or Serial-Flash (in 32-bit Dwords)?	<p>Specifies the starting depth of the Descriptor ROM or Serial-Flash in 32-bit Dwords used for DWC_usb3 controller.</p> <p>If you share the same ROM or Serial-Flash between USB2.0 hub and USB3.0 SS Hub, the lower portion of the ROM or Serial-Flash has data for the USB2.0 Hub and the upper part has data for the SS Hub. The DWC_usb3 controller reads the entire ROM or Serial-Flash but uses only the data from this depth. The DWC_usb3 controller provides the USB2.0 descriptors to the USB2.0 controller through the "hub_usb2_desc" interface.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0 (0) ■ 64 (64) ■ 128 (128) <p>Default Value: 0</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_HUB_DESC_SS_START_DEPTH</p>
Enable Vendor Control Command Interface	<p>Controls the enabling of Vendor Control Interface for decoding Vendor Control Commands for hub control endpoint (Ep0).</p> <p>When enabled, the hub will not STALL vendor control commands and passes these commands externally through the Vendor Control Interface (VCI) for external decoding. When disabled (default) hub will STALL any Vendor Control commands.</p> <ul style="list-style-type: none"> ■ 0: Vendor Control Interface does not exit and hub STALLS Vendor Control Commands. ■ 1: Vendor Control commands are not stalled and are passed externally for decoding through the VCI. <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: = 0</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_HUB_ENABLE_VCI</p>

Label	Description
Scrambling Control For Simulation	<p>This option only affects the simulation. This controls if scrambling is enabled/disabled when running hub tests in coreConsultant.</p> <p>This directly controls the hub_enable_scrambling input pin of the Hub.</p> <ul style="list-style-type: none">■ 0: A value of 0 will be assigned to "hub_enable_scrambling" input to disable scrambling on all ports.■ 1: A value of 1 will be assigned to "hub_enable_scrambling" input to enable scrambling on all ports. <p>Values:</p> <ul style="list-style-type: none">■ Disable (0)■ Enable (1) <p>Default Value: = 0</p> <p>Enabled: DWC_USB3_MODE ==3</p> <p>Parameter Name: DWC_USB3_HUB_ENABLE_SCRAMBLING</p>

4.6 Advanced Config Parameters

Table 4-6 Advanced Config Parameters

Label	Description
Global System Bus Configuration Register-0's (GSBUSCFG0) Power-On Initialization Value (0x0-fffffff)	<p>Specifies the Global System Bus Configuration Register-0's (GSBUSCFG0) power-on initialization value. The GSBUSCFG0 register defines the system bus configuration. The bits other than defined here must be 0. The different fields are:</p> <ul style="list-style-type: none"> ■ bit[31:28]: AHB-prot/AXI-cache/OCP-ReqInfo for Data Read ■ bit[27:24]: AHB-prot/AXI-cache/OCP-ReqInfo for Descriptor Read ■ bit[23:20]: AHB-prot/AXI-cache/OCP-ReqInfo for Data Write ■ bit[19:16]: AHB-prot/AXI-cache/OCP-ReqInfo for Descriptor Write ■ bit[15:12]: Reserved ■ bit[11]: 1'b1 - Data Access is Big Endian ■ bit[10]: 1'b1 - Descriptor Access is Big Endian ■ bit[9]: 1'b1 - Data Write is Posted ■ bit[8]: 1'b1 - Descriptor Write is Posted ■ bit[7]: 1'b1 - Enable INCR256 Burst Type ■ bit[6]: 1'b1 - Enable INCR128 Burst Type ■ bit[5]: 1'b1 - Enable INCR64 Burst Type ■ bit[4]: 1'b1 - Enable INCR32 Burst Type ■ bit[3]: 1'b1 - Enable INCR16 Burst Type ■ bit[2]: 1'b1 - Enable INCR8 Burst Type ■ bit[1]: 1'b1 - Enable INCR4 Burst Type ■ bit[0]: 1'b1 - Enable Undefined Length INCR Burst Type <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register must be configured by your driver. Refer to "GSBUSCFG0" section of the Databook for more details.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0x1</p> <p>Enabled: (DWC_USB3_MODE == 0 DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GSBUSCFG0_INIT</p>

Label	Description
Global System Bus Configuration Register-1's (GSBUSCFG1) Power-On Initialization Value (0x0-fffffff)	<p>Specifies the Global System Bus Configuration Register-1's (GSBUSCFG1) power-on initialization value. The GSBUSCFG1 register defines the system bus configuration. The bits other than defined here must be 0. The different fields are:</p> <ul style="list-style-type: none"> ■ bit[31:13]: Reserved ■ bit[12]: AXI, 1 KB Page Boundary Enable ■ bit[11:8]: AXI Pipelined Transfer Burst Request Limit ■ bit[7:0]: Reserved <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register must be configured by your driver. Refer to "GSBUSCFG1" section of the Databook for more details.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0x300</p> <p>Enabled: (DWC_USB3_MODE == 0 DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GSBUSCFG1_INIT</p>
Global Device TXFIFO DMA Priority Register's (GTXFIFOPRIDEV) Power-On Initialization Value (0x0-fffffff)	<p>Specifies the Global Device TXFIFO DMA Priority Register's (GTXFIFOPRIDEV) power-on initialization value.</p> <p>This register assigns FIFOs to either the high or low round-robin DMA priority groups. Register bit[n] represents the priority level (0: low; 1: high) of FIFO[n].</p> <p>The DMA arbiter grants system bus access on a per-packet basis:</p> <ul style="list-style-type: none"> ■ First priority to the high-priority FIFOs on a round-robin basis ■ Second priority to the low-priority FIFOs on a round-robin basis <p>For more information, see the "GTXFIFOPRIDEV" section in the Databook.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller.</p> <p>If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0x0</p> <p>Enabled: (DWC_USB3_MODE == 0 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GTXFIFOPRIDEV_INIT</p>

Label	Description
Global Host TXFIFO DMA Priority Register's (GTXFIFOPRIHST) Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the Global Host TXFIFO DMA Priority Register's (GTXFIFOPRIHST) power-on initialization value.</p> <p>This register assigns FIFOs to either the high or low round-robin DMA priority groups of each speed group: SS BIs and HS/FSLS BIs. Register bit[n] represents the priority level (0: low; 1: high) of FIFO[n].</p> <p>The DMA arbiter grants system bus access on a per-packet basis for each speed group:</p> <ul style="list-style-type: none"> ■ First priority to the high-priority FIFOs on a round-robin basis ■ Second priority to the low-priority FIFOs on a round-robin basis <p>For more information, see the "GTXFIFOPRIHST" section in the Databook.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller.</p> <p>If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0x0</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GTXFIFOPRIHST_INIT</p>
Global Host RXFIFO DMA Priority Register's (GRXFIFOPRIHST) Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the Global Host RXFIFO DMA Priority Register's (GRXFIFOPRIHST) power-on initialization value.</p> <p>This register assigns FIFOs to either the high or low round-robin DMA priority groups of each speed group: SS BIs and HS/FSLS BIs. Register bit[n] represents the priority level (0: low; 1: high) of FIFO[n].</p> <p>The DMA arbiter grants system bus access on a per-packet basis for each speed group:</p> <ul style="list-style-type: none"> ■ First priority to the high-priority FIFOs on a round-robin basis ■ Second priority to the low-priority FIFOs on a round-robin basis <p>For more information, see the "GRXFIFOPRIHST" section in the Databook.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller.</p> <p>If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0x0</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_GRXFIFOPRIHST_INIT</p>

Label	Description
Global Host DbC FIFO DMA Priority Register's (GFIFOPRIDBC) Power-On Initialization Value (0x0-fffffff)	<p>Specifies the Global Host DbC FIFO DMA Priority Register's (GFIFOPRIDBC) power-on initialization value.</p> <p>This register assigns the priority level of FIFOs belonging to DbC. For more information, see the "GFIFOPRIDBC" section in the Databook.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller.</p> <p>If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0x0</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4 && DWC_USB3_EN_DBC==1</p> <p>Parameter Name: DWC_USB3_GFIFOPRIDBC_INIT</p>
Global Host FIFO DMA High-Low Priority Ratio Register's (GDMAHLRATIO) TX SS:HSFSLS Ratio Power-On Initialization Value (0x0-1f)	<p>Specifies the Global Host FIFO DMA High-Low Priority Ratio Register's (GDMAHLRATIO) TX SS:HSFSLS Ratio power-on initialization value (bit[4:0]).</p> <p>This register specifies the relative priority of the SS FIFOs vs. the HS+FSLS FIFOs. Specifically, the DMA arbiter prioritizes the HS/FSLS round-robin arbiter group for one packet after the specified number of packet grants to the SS round-robin arbiter group. For more information, see the "GDMAHLRATIO" section in the Databook.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller.</p> <p>If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0, ..., 31</p> <p>Default Value: DWC_USB3_RX_SS_PRI_CNT</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4 && (DWC_USB3_EN_USB2_ONLY==0)</p> <p>Parameter Name: DWC_USB3_GDMAHLRATIO_TX_INIT</p>
Global Host FIFO DMA High-Low Priority Ratio Register's (GDMAHLRATIO) RX SS:HSFSLS Ratio Power-On Initialization Value (0x0-1f)	<p>Specifies the Global Host FIFO DMA High-Low Priority Ratio Register's (GDMAHLRATIO) RX SS:HSFSLS Ratio power-on initialization value (bit[12:8]).</p> <p>This register specifies the relative priority of the SS FIFOs vs. the HS+FSLS FIFOs. Specifically, the DMA arbiter prioritizes the HS/FSLS round-robin arbiter group for one packet after the specified number of packet grants to the SS round-robin arbiter group. For more information, see the "GDMAHLRATIO" section in the Databook.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller.</p> <p>If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0, ..., 31</p> <p>Default Value: DWC_USB3_RX_SS_PRI_CNT</p> <p>Enabled: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_MBUS_TYPE!=4 && (DWC_USB3_EN_USB2_ONLY==0)</p> <p>Parameter Name: DWC_USB3_GDMAHLRATIO_RX_INIT</p>

Label	Description
Enable LPM Errata	<p>Enables LPM Errata.</p> <p>Enabling this parameter enables support for the following USB Errata:</p> <ul style="list-style-type: none"> ■ Errata for USB 2.0 ECN: Link Power Management (LPM) 7/2007 ■ xHCI 1_0 Errata 08 BESL Dated 10/19/2011. <p>In host mode, the following changes will take effect.</p> <ul style="list-style-type: none"> ■ In the xHCI capability registers, bit 20 of the USB2.0 protocol defined fields will be set to 1. Setting this bit informs the xHCI driver that the host controller supports BESL decoding. ■ The duration of resume is changed based on the BESL values as compared to the HIRD values. <p>In Device mode the following changes will take effect.</p> <ul style="list-style-type: none"> ■ A new field in the DCTL register has been added which will control the ACK response from the device in response to an LPM token. {DCTL[23:20]} is used for this purpose. If the HIRD value received in the LPM token is greater than the register field value, then the device controller will respond with an NYET response. If the HIRD value received in the LPM token is less than or equal to the register field, then an ACK response is send by the device controller. <p>Note: Device driver for the controller must make sure that bits 15:2 of BmAttribute field in the USB2.0 Extension Descriptor are set accordingly to communicate to the host its optimized power savings design points. The driver must also be able to program the LPM_NYET_thres field (DCTL[23:20]) for controlling the NYET response if required. For more information, see the "LPM Errata Support" section in the Databook.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ LPM Errata Disable (0) ■ LPM Errata Enable (1) <p>Default Value: = 1</p> <p>Enabled: DWC_USB3_MODE !=3</p> <p>Parameter Name: DWC_USB3_EN_LPM_ERRATA</p>

Label	Description
Support Device/DbC Scatter-Gather Packets of 8 to 15 TRBs (9..15 if MDW128)?	<p>Enables Device and Debug Capability mode scatter-gather packets composed of 8 to 15 TRBs (9 to 15 if MDWIDTH==128). It is recommended that you enable this parameter and option to select only up to 7 will be removed in the future releases. Setting this parameter ensures support for scatter-gather packets composed of up to 15 TRBs, or DWC_USB3_CACHE_TRBS_PER_TRANSFER (in device mode), or DWC_USB3_DBC_TRBS_PER_TRANSFER (in DbC mode), whichever is less. Otherwise, Device/DbC mode can support scatter-gather packets composed of up to 7 TRBs (8 TRBs if MDWIDTH==128), or DWC_USB3_CACHE_TRBS_PER_TRANSFER (in device mode), or DWC_USB3_DBC_TRBS_PER_TRANSFER (in DbC mode), whichever is less. This parameter affects the RXQ and TXQ FIFO depths.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: (DWC_USB3_EN_DBC==1) ? 1: 0</p> <p>Enabled: (DWC_USB3_EN_DBC==1) ((DWC_USB3_MODE==0 DWC_USB3_MODE==2) && DWC_USB3_MBUS_TYPE!=4 && DWC_USB3_CACHE_TRBS_PER_TRANSFER > (7 + (DWC_USB3_MDWIDTH==128)))</p> <p>Parameter Name: DWC_USB3_DEV_EN_SCATTER_PACKETS_OF_8_TO_15_TRBS</p>
Information - Device minimum ram_clk frequency (MHz) for full performance	<p>This informational (read-only) value defines the device-mode minimum ram_clk frequency to achieve full bandwidth. This full performance frequency depends on the following parameters:</p> <ul style="list-style-type: none"> ■ SS or HS-only mode ■ MDWIDTH (32, 64, 128) ■ Number of RAMs (2 or 3) ■ RAM Type (single or two-port) <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: (DWC_USB3_DEV_RAM_CLK_FREQ_TURNAROUND>((DWC_USB3_LSP_SHARE_D_RAM_FREQ_PERF+DWC_USB3_DEV_CLK_FREQ_RXDATA)+(((DWC_USB3_SPRAM_TYP==1)&&(DWC_USB3_EN_USB2_ONLY==0))?DWC_USB3_DEV_CLK_FREQ_RXDATA:0)))?DWC_USB3_DEV_RAM_CLK_FREQ_TURNAROUND:((DWC_USB3_LSP_SHARED_RAM_FREQ_PERF+DWC_USB3_DEV_CLK_FREQ_RXDATA)+(((DWC_USB3_SPRAM_TYP==1)&&(DWC_USB3_EN_USB2_ONLY==0))?DWC_USB3_DEV_CLK_FREQ_RXDATA:0))</p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_DEV_RAM_CLK_FREQ_PERF</p>

Label	Description
	<p>Information - Device Minimum ram_clk frequency (MHz) for correct functionality</p> <p>This informational (read-only) value defines the device-mode minimum ram_clk frequency to achieve correct functionality of the controller. However, this clock frequency does not guarantee full performance.</p> <p>This lower clock frequency is useful for FPGA testing or low-power low-performance modes. The controller continues to function as long as the RAM has sufficient bandwidth to service USB traffic since DMA traffic has lower priority for the internal RAM arbitration. If the DMA bandwidth is less than the USB bandwidth, obviously the data throughput is reduced. The minimum ram_clk frequency for correct functionality depends on the master bus data width (MDWIDTH). MDWIDTH=32 requires a minimum ram_clk frequency of 125MHz to service USB 3.0 device traffic, while MDWIDTH=64 requires 62.5MHz and MDWIDTH=128 requires 60MHz.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: $(DWC_USB3_DEV_RAM_CLK_FREQ_TURNAROUND > (DWC_USB3_LSP_SHARE_D_RAM_FREQ_FUNC + DWC_USB3_DEV_CLK_FREQ_RXDATA)) ? DWC_USB3_DEV_RAM_CLK_FREQ_TURNAROUND : (DWC_USB3_LSP_SHARED_RAM_FREQ_FUNC + DWC_USB3_DEV_CLK_FREQ_RXDATA)$ </p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_DEV_RAM_CLK_FREQ_FUNC</p>

Label	Description
Information - Host minimum ram_clk frequency (MHz) for full performance	<p>This informational (read-only) value defines the host-mode minimum ram_clk frequency to achieve full bandwidth on all possible concurrently operating BIs. This full performance frequency depends on the following parameters:</p> <ul style="list-style-type: none"> ■ 1. Number of 3.0 BIs (SS) ■ 2. Number of 2.0 BIs (HS and FSLS) ■ 3. MDWIDTH (32, 64, 128) ■ 4. Number of RAMs (2 or 3) ■ 5. RAM Type (single or two-port) ■ 6. Dedicated DBC BI (if HOST_NUM_U3_ROOT_PORTS > NUM_SS_USB_INSTANCES) <p>Note: If the value of this parameter is less than the frequency of pipe/utmi clock, then you can enable automatic ram_clk selection using GUCTL[7:6] option.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value:</p> <pre>(DWC_USB3_HOST_RAM_CLK_FREQ_FIFOLATENCY>(((DWC_USB3_LSP_SHARED_RAM_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA)>(DWC_USB3_HOST_CLK_FREQ_TXDATA+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0))>(DWC_USB3_HOST_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0))))?((DWC_USB3_LSP_SHARED_RAM_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0)):(DWC_USB3_HOST_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0)):(DWC_USB3_HOST_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0)))))?DWC_USB3_HOST_RAM_CLK_FREQ_FIFOLATENCY:(((DWC_USB3_LSP_SHARED_RAM_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA)>(DWC_USB3_HOST_CLK_FREQ_RXDATA+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0))>(DWC_USB3_HOST_CLK_FREQ_RXDATA+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0))))?((DWC_USB3_LSP_SHARED_RAM_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0)):(DWC_USB3_HOST_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0)):(DWC_USB3_HOST_CLK_FREQ_RXDATA)+((DWC_USB3_HOST_CLK_FREQ_RXDATA:0))))))</pre> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_HOST_RAM_CLK_FREQ_PERF</p>

Label	Description
Information - Host minimum ram_clk frequency (MHz) for correct functionality	<p>This informational (read-only) value defines the host-mode minimum ram_clk frequency to achieve correct functionality of the controller with all possible configured and concurrently operating BIs. However, this clock frequency does not guarantee full performance.</p> <p>This lower clock frequency is useful for FPGA testing or low-power low-performance modes. The controller continues to function as long as the RAM has sufficient bandwidth to service USB traffic since DMA traffic has lower priority for the internal RAM arbitration. If the DMA bandwidth is less than the USB bandwidth, obviously the data throughput is reduced. The minimum ram_clk frequency for correct functionality depends on the master bus data width (MDWIDTH). MDWIDTH=32 requires a minimum ram_clk frequency of 125MHz to service USB 3.0 host traffic with single active SS-BI, while MDWIDTH=64 requires 62.5MHz and MDWIDTH=128 requires 60MHz to ensure adequate multi-BI arbitration (Note: Device mode also requires at least 60MHz).</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: $((DWC_USB3_HOST_RAM_CLK_FREQ_FIFOLATENCY > ((DWC_USB3_LSP_SHARED_RAM_FREQ_FUNC + DWC_USB3_HOST_CLK_FREQ_RXDATA) > DWC_USB3_HOST_RAM_CLK_FREQ_TXDATA)) ? (DWC_USB3_LSP_SHARED_RAM_FREQ_FUNC + DWC_USB3_HOST_CLK_FREQ_RXDATA) : DWC_USB3_HOST_RAM_CLK_FREQ_TXDATA) ? DWC_USB3_HOST_RAM_CLK_FREQ_FIFOLATENCY : ((DWC_USB3_LSP_SHARED_RAM_FREQ_FUNC + DWC_USB3_HOST_CLK_FREQ_RXDATA) > DWC_USB3_HOST_RAM_CLK_FREQ_TXDATA)) ? (DWC_USB3_LSP_SHARED_RAM_FREQ_FUNC + DWC_USB3_HOST_CLK_FREQ_RXDATA) : DWC_USB3_HOST_RAM_CLK_FREQ_TXDATA)$ </p> <p>Enabled: DWC_USB3_MODE==1 DWC_USB3_MODE==2</p> <p>Parameter Name: DWC_USB3_HOST_RAM_CLK_FREQ_FUNC</p>
Information - Default Device RAM0 Cache Depth	<p>This informational (read-only) value defines the minimum Cache requirement for Device.</p> <p>Values: 32, ..., 65536</p> <p>Default Value: $(((DWC_USB3_ADDR_CACHE_STREAMS + DWC_USB3_CACHE_STREAMS_DEPTH) + DWC_USB3_MBYTES) - 1) \gg DWC_USB3_MADDR_LO$ </p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_DCACHE_DEPTH_DEV</p>
Information - Default Host RAM0 Cache Depth	<p>This informational (read-only) value defines the minimum cache requirement for Host.</p> <p>Values: 32, ..., 65536</p> <p>Default Value: $(((DWC_USB3_DBCACHE_ADDR + (DWC_USB3_EN_DBC ? DWC_USB3_DBCACHE_SIZE : 0)) + DWC_USB3_MBYTES) - 1) \gg DWC_USB3_MADDR_LO$ </p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_DCACHE_DEPTH_HST</p>

Label	Description
Information - Default Device/Host/DRD RAM0 Cache Depth	<p>This informational (read-only) value defines the minimum RAM0 requirement for Device/Host/DRD Cache, Register, and internal queues mapped into RAM0. In the case of DRD, it is either the Device or Host Cache requirement, whichever is greater. When RAM0 is shared with RxFIFO (and TxFIFO), you can use this base value and add the RxFIFO (and TxFIFO) RAM requirements needed for your application over this.</p> <p>This value cannot be reduced. It is recommended that you allocate additional 256 bytes to provide margin for future updates. This is done by increasing the "Total RAM0 Depth" by 256/MDWIDTH.</p> <p>Values: 32, ..., (262144 / (DWC_USB3_MDWIDTH / 8))</p> <p>Default Value: DWC_USB3_DCACHE_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_DCACHE_DEPTH_INFO</p>
Information - Default Device RAM0/RAM1 TxFIFO Depth	<p>This informational (read-only) value defines the minimum TxFIFO requirement for Device.</p> <p>Values: 32, ..., 65536</p> <p>Default Value:</p> $(((((((((DWC_USB3_DEV_TXF0_DEPTH + DWC_USB3_DEV_TXF1_DEPTH) + DWC_USB3_DEV_TXF2_DEPTH) + DWC_USB3_DEV_TXF3_DEPTH) + DWC_USB3_DEV_TXF4_DEPTH) + DWC_USB3_DEV_TXF5_DEPTH) + DWC_USB3_DEV_TXF6_DEPTH) + DWC_USB3_DEV_TXF7_DEPTH) + DWC_USB3_DEV_TXF8_DEPTH) + DWC_USB3_DEV_TXF9_DEPTH) + DWC_USB3_DEV_TXF10_DEPTH) + DWC_USB3_DEV_TXF11_DEPTH) + DWC_USB3_DEV_TXF12_DEPTH) + DWC_USB3_DEV_TXF13_DEPTH) + DWC_USB3_DEV_TXF14_DEPTH) + DWC_USB3_DEV_TXF15_DEPTH$ <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_DEV_TXF_ALL_DEPTH</p>
Information - Default Host RAM0/RAM1 TxFIFO Depth	<p>This informational (read-only) value defines the minimum TxFIFO requirement for Host.</p> <p>Values: 32, ..., 65536</p> <p>Default Value:</p> $(((DWC_USB3_NUM_FSLS_USB_INSTANCES * DWC_USB3_HOST_TXFIFO_DEPTH_PER_FSLS_INSTANCE) + (DWC_USB3_NUM_HS_USB_INSTANCES * DWC_USB3_HOST_TXFIFO_DEPTH_PER_HS_INSTANCE)) + (DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * DWC_USB3_HOST_TXFIFO_DEPTH_PER_SS_INSTANCE)) + (DWC_USB3_EN_DBC_SRCSNK * DWC_USB3_DBC_EP1_TXF_DEPTH) + (DWC_USB3_EN_DBC * DWC_USB3_DBC_EP0_TXF_DEPTH)$ <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_HOST_TXF_ALL_DEPTH</p>

Label	Description
Information - Default Device/Host/DRD RAM0/RAM1 TxFIFO Depth	<p>This informational (read-only) value defines the minimum TxFIFO requirement for Device/Host/DRD.</p> <p>In the case of DRD, it is either the Device or Host TxFIFO requirement, whichever is greater.</p> <p>When RAM0 is shared with the TXFIFOs and/or RXFIFOs, take the sum of this base value and the required TXFIFO and/or RXFIFO depths to find the total RAM0 depth.</p> <p>Values: 32, ..., 65536</p> <p>Default Value: $(DWC_USB3_MODE==0)?DWC_USB3_DEV_TXF_ALL_DEPTH:(DWC_USB3_MODE==1)?DWC_USB3_HOST_TXF_ALL_DEPTH:(DWC_USB3_DEV_TXF_ALL_DEPTH>DWC_USB3_HOST_TXF_ALL_DEPTH)?DWC_USB3_DEV_TXF_ALL_DEPTH:DWC_USB3_HOST_TXF_ALL_DEPTH)$</p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_TXF_ALL_DEPTH</p>
Information - Default Device RAM0/RAM2 RxFIFO Depth	<p>This informational (read-only) value defines the minimum RxFIFO requirement for Device.</p> <p>Values: 32, ..., 65536</p> <p>Default Value: DWC_USB3_DEV_RXF_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_DEV_RXF_DEPTH_DEFAULT</p>
Information - Default Host RAM0/RAM2 RxFIFO Depth	<p>This informational (read-only) value defines the minimum RxFIFO requirement for Host.</p> <p>Values: 32, ..., 65536</p> <p>Default Value: $((DWC_USB3_NUM_FSLS_USB_INSTANCES*DWC_USB3_HOST_RXFIFO_DEPTH_PER_FSLS_INSTANCE)+(DWC_USB3_NUM_HS_USB_INSTANCES*DWC_USB3_HOST_RXFIFO_DEPTH_PER_HS_INSTANCE))+(DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE*DWC_USB3_HOST_RXFIFO_DEPTH_PER_SS_INSTANCE))+(DWC_USB3_EN_DBC_SRCNSK*DWC_USB3_DBC_RXF_DEPTH)$</p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_HOST_RXF_ALL_DEPTH</p>
Default Information - Device/Host/DRD RAM0/RAM2 RxFIFO Depth	<p>This informational (read-only) value defines the minimum RxFIFO requirement for Device/Host/DRD. In the case of DRD, it is either the Device or Host RxFIFO requirement, whichever is greater.</p> <p>Values: 32, ..., 65536</p> <p>Default Value: $(DWC_USB3_MODE==0)?DWC_USB3_DEV_RXF_DEPTH:(DWC_USB3_MODE==1)?DWC_USB3_HOST_RXF_ALL_DEPTH:(DWC_USB3_DEV_RXF_DEPTH>DWC_USB3_HOST_RXF_ALL_DEPTH)?DWC_USB3_DEV_RXF_DEPTH:DWC_USB3_HOST_RXF_ALL_DEPTH)$</p> <p>Enabled: 0</p> <p>Parameter Name: DWC_USB3_RXF_ALL_DEPTH</p>

Label	Description
Total RAM0 Depth (32 to 65536)	<p>Specifies the depth of RAM0.</p> <p>It is recommended to not reduce the size from the default value since this value is automatically calculated from your previous parameters selections. It is recommended that you allocate an additional 256 bytes to provide a margin for future Cache size updates. This is done by increasing the RAM0 depth by 256/MDWIDTH.</p> <p>In Host, Device, and DRD configuration, the RAM0 contains:</p> <ul style="list-style-type: none"> ■ 3 RAM Config: RAM0 contains Descriptor Cache. ■ 2 RAM Config: RAM0 contains Descriptor Cache and RxFIFOs. ■ 1 RAM Config: RAM0 contains Descriptor Cache, RxFIFOs, and TxFIFOs. <p>RxFIFO Size:</p> <ul style="list-style-type: none"> ■ In Device mode, the recommended RxFIFO size in number of MaxPacket is "(DWC_USB3_EN_USB2_ONLY==1 ? ((DWC_USB3_EN_ISOC_SUPT==1 && DWC_USB3 MDBUS_ACCESS_GT21==1) ? 3: (DWC_USB3 MDBUS_ACCESS_GT21==1 ? 4: 2)): (DWC_USB3 MDBUS_ACCESS_GT21==1 ? 5: 3)". When system latency is larger than 2.1uS (10 uS in USB 2.0 only mode) larger FIFO size is selected. Note that in addition to the packet space there are a few additional bytes overhead. MDWIDTH/8 bytes per packet are allocated for status information. 40 to 64 bytes allocated in Rx FIFO for receiving 3 back to back SETUP packets in device mode and also to allow margin for the clock-crossing delay. When the Master Bus width is 128 bits, 64 bytes is needed; else, 40 bytes is needed ■ In Host mode, for each SS, HS, and FS/LS instances 3, 2, and 1 1KB buffer is allocated if the system latency is larger than 2.1uS (10 uS in USB 2.0 only mode) else 5, 3, and 1 1KB buffer is recommended. Note that in addition to the packet space there are a few additional bytes overhead. The HS and FS/LS FIFOs need additional 32-bytes for status information. In Host mode, the sum of SS, HS, and FS/LS Rx data buffering is total Rx FIFO size. <p>TxFIFO Size:</p> <ul style="list-style-type: none"> ■ In Device configuration 512 bytes for Control IN endpoint is needed and for other IN endpoints "(DWC_USB3_EN_USB2_ONLY==1 ? ((DWC_USB3_DEV_TXF1_BURST_EN==1 && DWC_USB3 MDBUS_ACCESS_GT21==1) ? 3: (DWC_USB3 MDBUS_ACCESS_GT21==1 ? 4: 2)): (DWC_USB3_DEV_TXF1_BURST_EN==0 ? 2: (DWC_USB3 MDBUS_ACCESS_GT21==1 ? 5: 3)))" number of MaxPackets amount of FIFO is recommended. When system latency is larger than 2.1uS (10 uS in USB 2.0 only mode) larger FIFO size is selected and also larger FIFO for burstable endpoint is needed. Note that in addition to the packet space there are a few additional bytes overhead. MDWIDTH/8 bytes per packet and per endpoint are allocated for status information. In Device mode, sum of all the IN endpoint buffering is total TxFIFO size. ■ In Host mode, for each SS, HS, and FS/LS instances 4, 2, and 1 1KB buffer is allocated if the system latency is larger than 2.1uS (10 uS in USB 2.0 only mode) else 5, 3, and 1 1KB buffer is recommended. Note that in addition to the packet space there are a few additional bytes overhead. The SS, HS and FS/LS TxFIFOs need additional MDWIDTH/8 bytes per packets for status information and one additional MDWIDTH/8 per TxFIFO. In Host mode, the sum of SS, HS, and FS/LS Tx data buffering is total TxFIFO size. <p>In Hub configuration, RAM0 contains Hub descriptors and Upstream Tx Data Packets.</p> <p>Values: 32, ..., 65536</p> <p>Default Value: DWC_USB3_RAM0_DEPTH_DEFAULT</p> <p>Enabled: Always</p> <p>Parameter Name: DWC_USB3_RAM0_DEPTH</p>

Label	Description
Total RAM1 Depth (32 to 65536)	<p>Specifies the depth of RAM1. It is recommended not to reduce the size from default value since this value is automatically calculated from your previous parameters selections.</p> <p>In Host, Device, and DRD configuration RAM1 contains:</p> <ul style="list-style-type: none"> ■ 3 RAM Config: Contains TxFIFOs. ■ 2 RAM Config: Contains TxFIFOs. <p>TxFIFO Size:</p> <ul style="list-style-type: none"> ■ In Device configuration 512 bytes for Control IN endpoint is needed and for other IN endpoints "(DWC_USB3_EN_USB2_ONLY==1 ? ((DWC_USB3_DEV_TXF1_BURST_EN==1 && DWC_USB3 MDBUS_ACCESS_GT21==1) ? 3: (DWC_USB3 MDBUS_ACCESS_GT21==1 ? 4: 2)): (DWC_USB3_DEV_TXF1_BURST_EN==0 ? 2: (DWC_USB3 MDBUS_ACCESS_GT21==1 ? 5: 3)))" number of MaxPackets amount of FIFO is recommended. When system latency is larger than 2.1uS (10 uS in USB 2.0 only mode), a larger FIFO size is selected and also a larger FIFO for burstable endpoint is needed. Note that in addition to the packet space there are a few additional bytes overhead. MDWIDTH/8 bytes per packet are allocated for status information. In Device mode, the sum of all the IN endpoint buffering is total TxFIFO size. ■ In Host mode, for each SS, HS, and FS/LS instances 4, 2, and 1 1KB buffer is allocated if the system latency is larger than 2.1uS (10 uS in USB 2.0 only mode); otherwise, 5, 3, and 1 1KB buffer is recommended. Note that in addition to the packet space there are a few additional bytes overhead. The SS, HS and FS/LS TxFIFOs need additional MDWIDTH/8 bytes per packets for status information and one additional MDWIDTH/8 per TxFIFO. In Host mode, the sum of SS, HS, and FS/LS Tx data buffering is total TxFIFO size. <p>In Hub configuration RAM1 contains the Upstream Rx Data Packets.</p> <p>Values: 0, ..., 65536</p> <p>Default Value: DWC_USB3_RAM1_DEPTH_DEFAULT</p> <p>Enabled: DWC_USB3_NUM_RAMS > 1</p> <p>Parameter Name: DWC_USB3_RAM1_DEPTH</p>

Label	Description
Total RAM2 Depth (32 to 65536)	<p>Specifies the depth of RAM2. It is recommended not to reduce the size from default value since this value is automatically calculated from your previous parameters selections.</p> <p>3 RAM config: Contains RxFIFOs.</p> <p>RxFIFO Size:</p> <ul style="list-style-type: none"> ■ In Device mode, the recommended Rx FIFO size in number of MaxPacket is "(DWC_USB3_EN_USB2_ONLY==1 ? ((DWC_USB3_EN_ISOC_SUPT==1 && DWC_USB3 MDBUS_ACCESS_GT21==1) ? 3: (DWC_USB3 MDBUS_ACCESS_GT21==1 ? 4: 2)): DWC_USB3 MDBUS_ACCESS_GT21==1 ? 5: 3)". When system latency is larger than 2.1uS (10 uS in USB 2.0 only mode) larger FIFO size is selected. Note that in addition to the packet space there are a few additional bytes overhead. MDWDITH/8 bytes per packet are allocated for status information. 40 to 64 bytes allocated in Rx FIFO for receiving 3 back to back SETUP packets in device mode and also to allow margin for the clock-crossing delay. When the Master Bus width is 128 bits, 64 bytes is needed; else, 40 bytes is needed ■ In Host mode, for each SS, HS, and FS/LS instances 3, 2, and 1 1KB buffer is allocated if the system latency is larger than 2.1uS (10 uS in USB 2.0 only mode) else 5, 3, and 1 1KB buffer is recommended. Note that in addition to the packet space there are a few additional bytes overhead. The HS and FS/LS FIFOs need additional 32-bytes for status information. In Host mode, the sum of SS, HS, and FS/LS Rx data buffering is total Tx FIFO size. <p>Values: 0, ..., 65536</p> <p>Default Value: DWC_USB3_RAM2_DEPTH_DEFAULT</p> <p>Enabled: DWC_USB3_NUM_RAMS > 2 && DWC_USB3_MODE !=3</p> <p>Parameter Name: DWC_USB3_RAM2_DEPTH</p>
Preserve logic_analyzer_trace mux select during reset for hardware debug?	<p>When enabled, this parameter will detach the reset signal from the GDBGLSPMUX register so that the mux select for the logic_analyzer_trace top-level output will not lose its value during reset. This is useful when debugging signals across hibernation resets.</p> <p>Note: This will cause synthesis tools to warn about flip-flops with missing reset in DWC_usb3_csr_dev/gdbglspmux_reg[]</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: DWC_USB3_MODE !=3</p> <p>Parameter Name: DWC_USB3_PRESERVE_LOGIC_ANALYZER_SELECT</p>

Label	Description
Enable FPGA Implementation?	<p>This parameter is used at Synopsys for FPGA Device and Host hardware validation of the controller. Enabling this parameter is not recommended for ASIC because this adds 1-clock delay during cache/Rx/Tx memory access.</p> <p>You can use this parameter during driver development with an FPGA platform. Set this parameter to 1 to improve FPGA timing; timing is improved by:</p> <ul style="list-style-type: none"> ■ adding registers for a pipeline stage on RX data path in the PTL ■ adding registers for a pipeline stage on cache/DMA request in LSP BARB ■ adding registers for a pipeline stage on address generation in LSP HCMD ■ adding registers for a pipeline in the MAC ■ connect the pipe3_TxData and pipe3_TxDataK outputs directly to the external PHY instead of through the U3PMU (only for DWC_USB3_EN_PWR0PT==2) <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: DWC_USB3_MODE !=3</p> <p>Parameter Name: DWC_USB3_EN_FPGA</p>

Label	Description
	<p>Enable USB Logical Endpoint to Physical Endpoint Mapping?</p> <p>This parameter is used at Synopsys for FPGA Device hardware validation of the controller. This is not recommended for the ASIC; since this will removed flexible endpoint configurability by software.</p> <p>You can use this parameter during driver development with an FPGA platform. Set this parameter to 0 to improve FPGA timing. Selects whether the device controller supports a flexible or fixed logical->physical endpoint mapping. A fixed mapping improves timing and may reduce gate count (5K gate reduction). For example,</p> <p>Flexible EP DISABLE: No SW control</p> <ul style="list-style-type: none"> ■ Physical ep 0 -> Logical ep 0 OUT ■ Physical ep 1 -> Logical ep 0 IN ■ Physical ep 2 -> Logical ep 1 OUT ■ Physical ep 3 -> Logical ep 1 IN ■ Physical ep 4 -> Logical ep 2 OUT ■ Physical ep 5 -> Logical ep 2 IN ■ Physical ep 6 -> Logical ep 3 OUT ■ Physical ep 7 -> Logical ep 3 IN ■ ■ Physical ep 30 -> Logical ep 15 OUT ■ Physical ep 31 -> Logical ep 15 IN <p>Flexible EP ENABLE: SW configurable</p> <ul style="list-style-type: none"> ■ Physical ep 0 -> Logical ep 0 OUT ■ Physical ep 1 -> Logical ep 0 IN ■ Physical ep 2 -> Logical ep 4 IN ■ Physical ep 3 -> Logical ep 15 IN ■ <p>For more details on flexible EP option, refer to "Flexible Endpoint Mapping" section in the Databook.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: Yes</p> <p>Enabled: DWC_USB3_MODE != 1 && DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_EN_LOG_PHYS_EP_SUPT</p>

Label	Description
Enable Additional DFT Control Ports?	<p>This parameter enables additional DFT Control signals for internally generated clocks. This creates additional <code>dft_*</code> ports to control clocks. When this parameter is not enabled then, only the "scan_mode" port controls scan by-pass.</p> <p>This parameter controls only whether additional inputs are used to control clock muxes, clock-gate cell, and reset bypass. The DWC_usb3 controller by default supports At-Speed DFT irrespective of this parameter is enabled or not. All the module are driven with their appropriate clock during scan-mode.</p> <p>Refer to "Synthesis, DFT, and UPF Flow" chapter in the User Guide for more details.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: <code>DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</code></p> <p>Parameter Name: <code>DWC_USB3_ATSPEED_DFT</code></p>
Enable Filter for VBUS and ID related control inputs from the PHY?	<p>Enables Bus Filters for the following UTMI/ULPI signals and PIPE3PowerControl input signal from the PHY.</p> <ul style="list-style-type: none"> ■ <code>utmiotg_vbusvalid</code> ■ <code>utmisrp_bvalid</code> ■ <code>pipe3_PowerPresent</code> <p>Each filter is implemented in the <code>DWC_usb3_filter.v</code> module as a 5ms counter that works on the <code>mac_clk</code>. If your PHY already has a filter de-bounce, then it is not necessary to enable the one in the <code>DWC_usb3</code> controller. For example, Synopsys PHY does not have the filters. In the case of UTMI PHY, the <code>utmi*</code> signals arrive from the PHY and in the case of ULPI PHY, these signals are internally generated in the ULPI wrapper of the <code>DWC_usb3</code> controller. These filters can be individually bypassed by the "<code>bus_filter_bypass[3:0]</code>" port or all these filters can be disabled by the <code>GCTL.MASTERFILTBYPASS</code> register bit.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: <code>DWC_USB3_MODE !=3</code></p> <p>Parameter Name: <code>DWC_USB3_EN_BUS_FILTERS</code></p>

Label	Description
Enable Filters for UTMI and Overcurrent related inputs from the PHY in PMU module?	<p>Enables simple filters for the following UTMI signals and overcurrent input signal in the PMU module. These filters are used only during hibernation when DWC_usb3 is powered off.</p> <ul style="list-style-type: none"> ■ phy_utmi_linestate ■ phy_utmiotg_vbusvalid ■ phy_utmisrp_bvalid ■ phy_hub_port_overcurrent <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: = 0</p> <p>Enabled: DWC_USB3_EN_PWROPT ==2</p> <p>Parameter Name: DWC_USB3_EN_PMU_FILTER</p>
Power switch polarity	<p>Determines the polarity of the power control switch when DWC_USB3_EN_PWROPT=2. This parameter is used to specify the polarity of the power switch when used along with UPF.</p> <ul style="list-style-type: none"> ■ When a value of "Low" is chosen, it implies the external power controller should drive 0 to keep the power ON for Vcc domain. Driving a 1 will turn OFF the power for Vcc domain. ■ When a value of "High" is chosen, it implies the external power controller should drive 1 to keep the power ON for Vcc domain. Driving a 0 will turn OFF the power for Vcc domain. <p>Values:</p> <ul style="list-style-type: none"> ■ Low (0) ■ High (1) <p>Default Value: = 0</p> <p>Enabled: DWC_USB3_EN_PWROPT ==2</p> <p>Parameter Name: DWC_USB3_PWR_SWITCH_POLARITY</p>

Label	Description
Always Assign ram_clk to bus_clk_early?	<p>This parameter enables you to always connect bus_clk_early to ram_clk. Normally, it is not recommended that you select this parameter because this removes the flexibility in lowering bus_clk in low power modes, and also could cause functional failures if the system bus_clk does not meet the minimum ram clock frequency requirement.</p> <p>The bus_clk_early frequency must satisfy the following minimum ram_clk frequency requirement.</p> <p>For a USB 2.0 device, the minimum ram_clk frequency is 60 MHz.</p> <p>For a SS device and host, the minimum ram_clk frequency depends on the configuration and mode of operation. For details, refer to "Minimum Clock Frequencies: bus_clk, ram_clk" section in the "Area, Speed, Power, DFT, and Performance" chapter of the Databook.</p> <p>Note: Functional failure occurs if the ram_clk frequency does not meet the requirements. This parameter avoids the clock dividers and multiplexers on the ram_clk and simplifies the DFT and clock insertion (the clock domain synchronizers between bus_clk and ram_clk are still present and are not optimized away).</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_RAM_CLK_TO_BUS_CLK</p>

Label	Description
Always use posedge clock synchronizers?	<p>Selects whether to use posedge clocked synchronizers instead of first-stage negedge, second-stage posedge synchronizers. Note that given design requirements, certain synchronizers will still remain with a negedge first-stage and certain clock-generation logic requires negedge clocking. For a list of negedge flops, see workspace/syn/final/report/neg*.rpt. Negedge first-stage synchronizers are required in the DWC_usb3_pwrn_u3piu module to meet the USB 3.0 Specification requirement that SuperSpeed signaling must start within 20us of LFPS transmission. These synchronizers also minimize PIPE3 signal compression and expansion through the following synchronizers:</p> <ul style="list-style-type: none"> ■ U_DWC_usb3_pwrn_prt_U_DWC_usb3_pwrn_u3piu_U_DWC_usb3_bussync_rx_status_U_DWC_usb3_sync_ctl_s2dl_in_p_1d_reg_0 ■ U_DWC_usb3_pwrn_prt_U_DWC_usb3_pwrn_u3piu_U_DWC_usb3_bussync_rx_status_U_DWC_usb3_sync_ctl_d2sl_in_p_1d_reg_0 ■ U_DWC_usb3_pwrn_prt_U_DWC_usb3_pwrn_u3piu_U_DWC_usb3_bussync_lnkstate_U_DWC_usb3_sync_ctl_d2sl_in_p_1d_reg_0 ■ U_DWC_usb3_pwrn_prt_U_DWC_usb3_pwrn_u3piu_U_DWC_usb3_bussync_lnkstate_U_DWC_usb3_sync_ctl_s2dl_in_p_1d_reg_0 ■ U_DWC_usb3_pwrn_prt_U_DWC_usb3_pwrn_u3piu PIPE3_sync_9b_32_reg_35 <p>If you select all posedge-clocked synchronizers with either FAST_TAT_EN set to zero or mac2_clk set to 30 MHz (16-bit UTMI mode), the device-mode minimum ram_clk frequency increases from 60 MHz to 100 MHz to support the 5-clock turnaround time. If the mac2_clk is 60 MHz and FAST_TAT_EN is 1, enabling all posedge-clocked synchronizers does not increase the minimum ram_clk frequency of 60 MHz. For more details, refer to "Negedge-Sampled Signals and SDF Annotated Timing Check OFF flops" section of the Databook. In host mode, if you are using 16-bit UTMI interface and you have DWC_USB3_FAST_TAT disabled, you cannot enable this parameter.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: DWC_USB3_MODE !=3</p> <p>Parameter Name: DWC_USB3_EN_SYNC_ALL_POSEdge</p>

Label	Description
Remove pipe_clk mux for 2.0 mode?	<p>This parameter removes additional clock muxes on pipe3_clk for working with USB 2.0 PHY only.</p> <p>In v2.90a, a new feature was added to support USB 2.0 speeds only even when the controller is configured for USB 3.0 (that is, non USB 2.0-only) mode and for some reason USB 3.0 PHY is not connected or not working.</p> <p>The feature can be enabled by setting GUCTL1[DEV_FORCE_20_CLK_FOR_30_CLK] to 1'b1.</p> <p>This mode adds muxes on pipe3_clk input to route the USB 2.0 clocks internally. If this mode is not used or the muxes are implemented outside the controller, then the muxes inside the controller are not needed and can be removed by enabling this parameter.</p> <p>Enabling the parameter removes the muxes and the related logic to support GUCTL1[DEV_FORCE_20_CLK_FOR_30_CLK] functionality.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: DWC_USB3_MODE !=3 && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_REMOVE_PIPE_CLK_MUX_FOR_20_MODE</p>
Use set_false_path instead of set_max_delay between clocks?	<p>This parameter enables either setting max_delay or setting false_path between all top-level clocks during synthesis. The max_delay constraint avoids scenic routing during backend and makes sure the data signals passed between the clock domains do not exceed 2-clock max delay which is needed for correct operation when the data signals are passed with the control signals between two clock domains.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ SET_MAX_DELAY (0) ■ SET_FALSE_PATH (1) <p>Default Value: SET_MAX_DELAY</p> <p>Enabled: Always</p> <p>Parameter Name: DWC_USB3_CLK_SET_FALSE_PATH</p>
LINK_SETTINGS Power-On Initialization Value (0x0-ffffffff)	<p>This parameter specifies Link Settings.</p> <p>The different fields are as follows:</p> <ul style="list-style-type: none"> ■ DWC_USB3_LINK_SETTINGS[23:20] pm_entry_timer_us: This field specifies entry PM_LC_TIMER in us ■ DWC_USB3_LINK_SETTINGS[26:24] pm_lc_timer_us: This field specifies PM_LC_TIMER in us ■ DWC_USB3_LINK_SETTINGS[30:28] u1_residency_timer_us: This field enables the use of U1_RESIDENCY_TIMER in us <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: = 0x45900000</p> <p>Enabled: (DWC_USB3_SSPHY_INTERFACE!=0) && DWC_USB3_MBUS_TYPE!=4</p> <p>Parameter Name: DWC_USB3_LINK_SETTINGS_INIT</p>

Label	Description
Link User Control (LUCTL) Register's Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the SS Link Layer User Control Register's power-on initialization value.</p> <ul style="list-style-type: none"> ■ LUCTL[29]: P4 power gate mode is enabled ■ LUCTL[28]: support P3.CPM and P4 ■ LUCTL[23]: DisRxDet_LTSSM_Timer_Ovrrd; When DisRxDetU3RxDet is asserted in Polling or U1, the timeout expires immediately ■ LUCTL[12]: U2P3CPMok; enable P3CPM in U2 ■ LUCTL[11]: en_reset_pipe_after_phy_mux; controller issues 3.0 PHY reset after DisRxDetU3RxDet is de-asserted. ■ LUCTL[7]: mask_pipe_reset; If this bit is set, controller will block pipe_reset_n from going to PHY when DisRxDetU3RxDet_ack=1 ■ LUCTL[5]: no_ux_exit_p0_trans; Link LTSSM detects Ux_exit lfps when P0 transition is on-going by default. If this bit is set, Link LTSSM may miss Ux_exit lfps when P0 transition is happening. Set this bit if PHY has problem with LFPS detection during PHY power state transition. <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own device or xHCI-host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0x8000080</p> <p>Enabled: (DWC_USB3_SSPHY_INTERFACE!=0) && (DWC_USB3_MBUS_TYPE!=4)</p> <p>Parameter Name: DWC_USB3_LINK_LUCTL_INIT</p>
Link PTM_DATAPATH_DELAY (PTM_DATAPATH_DELAY) Register's Power-On Initialization Value (0x0-ffffffff)	<p>Specifies the SS Link Layer PTM_DATAPATH_DELAY Register's power-on initialization value. LINK_PTM_DATAPATH_DELAY[21:10]: p3cpmp4_residency timer in terms of number of suspend_clk periods. That is, minimum number of suspend_clk periods that the PHY needs to stay in P3.CPM or P4 before exit P3.CPM or P4.</p> <p>If you use a standard driver, such as the xHCI driver from Microsoft, you must initialize this register to meet your system requirements before synthesizing the controller. If you are developing your own xHCI host driver, then this register can be configured by your driver.</p> <p>Values: 0x0, ..., 0xffffffff</p> <p>Default Value: 0xc00</p> <p>Enabled: (DWC_USB3_SSPHY_INTERFACE!=0) && (DWC_USB3_MBUS_TYPE!=4)</p> <p>Parameter Name: DWC_USB3_LINK_LPTMDPDELAY_INIT</p>

Label	Description
U1/U2 LFPS RX TIMER Power-On Initialization Value (0x0-ffffffff)	<p>This parameter specifies Link U1/U2 LFPS RX TIMER reset values The different fields are as follows:</p> <ul style="list-style-type: none"> ■ DWC_USB3_LU1LFPSRXTIM[7:0] u1u2_exit_rsp_rx_clk: U1/U2 EXIT_RESPONSE_RX_CLKS ■ DWC_USB3_LU1LFPSRXTIM[15:8] u1u2_lfps_exit_rx_clk: U1/U2 LFPS_EXIT_RX_CLKS <p>Values: 0x0, ..., 0xffffffff Default Value: = 0x00001F1F Enabled: (DWC_USB3_SSPHY_INTERFACE!=0) && DWC_USB3_MBUS_TYPE!=4 Parameter Name: DWC_USB3_LU1LFPSRXTIM_INIT</p>

Additional Configuration Details

This chapter provides additional details for various configuration parameters, as provided in the following topics:

- “[Recommendations for Selecting DWC_USB3_FAST_TAT_EN Value](#)” on page [258](#)
- “[Example Device Endpoint Mapping in Different Applications](#)” on page [259](#)

5.1 Recommendations for Selecting DWC_USB3_FAST_TAT_EN Value

[Table 5-1](#) and [Table 5-2](#) summarize the recommendation for selecting the DWC_USB3_FAST_TAT_EN value for device and host mode, respectively.

Table 5-1 Recommendation for DWC_USB3_FAST_TAT_EN Value in Device Mode

Configuration			Recommendation		
8/16-Bit Mode	Single/Multi-Port	DWC_USB3_EN_SYNC_ALL_POSEDGE	DWC_USB3_FAST_TAT_EN	Minimum RAM_CLK_FREQ	Notes
8	Single	X	X	60MHz	-
16	Single	1	1	100MHz	Recommend enabling FAST_TAT when in 16-bit mode
16	Single	0	1	60MHz	Recommend enabling FAST_TAT when in 16-bit mode

Table 5-2 Recommendation for DWC_USB3_FAST_TAT_EN Value in Host Mode

Configuration				Recommendation	
8/16-Bit Mode	Single/Multi-Port	Clocks	DWC_USB3_EN_SYNC_ALL_POSEDGE	DWC_USB3_FAST_TAT_EN	Notes
8	Single	N/A	X	X	-
8	Multi-port	Sync clocks	X	X	-
8	Multi-port	Async clocks	X	0	-
16	Single	N/A	X	1	Recommend enabling FAST_TAT when in 16-bit mode
16	Multi-port	Sync clocks	X	1	Recommend enabling FAST_TAT when in 16-bit mode
16	Multi-port	Async clocks	1	0	Violates turnaround (FAST_TAT must be zero for async clocks, and ALL_POSEDGE cannot be 1). Recommend using 8-bit mode.
16	Multi-port	Async clocks	0	0	Meets turnaround (FAST_TAT must be zero for async clocks, if ALL_POSEDGE=0, then it may meet turnaround if PHY delays are smaller).

5.2 Example Device Endpoint Mapping in Different Applications

[Table 5-3](#) provides some examples of endpoint mapping for different applications. You can also connect a USB device to a USB port of a PC running on Windows and find out the device endpoint configuration using USBView application. USBView is a part of Windows Driver Kit (WDK) and can be downloaded from Microsoft. For more details, see the following link:

[http://msdn.microsoft.com/en-us/library/windows/hardware/ff560019\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/hardware/ff560019(v=vs.85).aspx)

Table 5-3 Example Device Endpoint Mapping in Different Applications

Application Type	Device	Number of Endpoints	Endpoint Type and Max Packet Size
Mass storage	SuperSpeed, Flash Drive, Example-1	4	EP-0: Control (IN, OUT), 512 Bytes EP-3: Bulk OUT, 1024 Bytes EP-4: Bulk IN, 1024 Bytes
	SuperSpeed, SATA Drive, Example-2	4	EP-0: Control (IN, OUT), 512 Bytes EP-3: Bulk OUT, 1024 Bytes EP-4: Bulk IN, 1024 Bytes
	SuperSpeed, Synopsys Mass-Storage Demo RAM-Disk Device	4	EP-0: Control (IN, OUT), 512 Bytes EP-1: Bulk IN, OUT, 1024 Bytes
	High Speed, USB2.0 Flash Drive, Example-3	4	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, 512 Bytes EP-2: Bulk OUT, 512 Bytes
	High Speed, USB 2.0 Storage RAID Array, Example-4	4	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, 512 Bytes EP-2: Bulk OUT, 512 Bytes
	High Speed, Hard Drive, Example-5	4	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, 512 Bytes EP-2: Bulk OUT, 512 Bytes
DVD Burner	High Speed, DVD Burner	4	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, 512 Bytes EP-2: Bulk OUT, 512 Bytes
Card Reader	High Speed, Card Reader	5	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, 512 Bytes EP-2: Bulk OUT, 512 Bytes EP-3: Bulk OUT, 512 Bytes
Flatbed Scanner	High Speed, USB 2.0 Interface Flatbed Scanner	4	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, 512 Bytes EP-2: Bulk OUT, 512 Bytes

Application Type	Device	Number of Endpoints	Endpoint Type and Max Packet Size
Game Controller	Full Speed, Wireless Game Controller	6	EP-0: Control (IN, OUT), 8 Bytes EP-1: Interrupt IN, 32 Bytes EP-2: Interrupt IN, 32 Bytes EP-3: Interrupt OUT, 32 Bytes EP-4: Interrupt IN, 32 Bytes
Headset	Full Speed, Headset	6	EP-0: Control (IN, OUT), 64 Bytes EP-1: Interrupt IN, 2 Bytes EP-2: ISO IN, 96 Bytes EP-3: ISO IN, 3 Bytes EP-5: ISO OUT, 784 Bytes
Wireless Network Adapter	High Speed, Wireless Adapter	11	EP-0: Control (IN, OUT), 64 Bytes EP-3: Bulk IN, 512 Bytes EP-4: Bulk OUT, 512 Bytes EP-5: Bulk OUT, 512 Bytes EP-6: Bulk OUT, 512 Bytes EP-7: Bulk OUT, 512 Bytes EP-9: Bulk IN, 512 Bytes EP-10: Bulk OUT, Bytes EP-11: Bulk IN, Bytes EP-12: Bulk OUT, Bytes
Barcode Scanner	Full Speed, USB Barcode Scanner	3	EP-0: Control (IN, OUT), 8 Bytes EP-1: Interrupt IN, 8 Bytes
Biometric Scanner	Full Speed, Biometric Scanner	5	EP-0: Control (IN, OUT), 8 Bytes EP-1: Bulk IN, 64 Bytes EP-2: Bulk OUT, 64 Bytes EP-3: Interrupt IN, 4 Bytes
Bluetooth Adapter	Full Speed, Bluetooth Adapter	7	EP-0: Control (IN, OUT), 64 Bytes EP-1: Interrupt IN, 16 Bytes EP-2: Bulk IN, Bulk OUT, 64 Bytes EP-3: ISO IN, 49 Bytes EP-4: Bulk IN, 32 Bytes
Webcam	High Speed, Webcam with Microphone	5	EP-0: Control (IN, OUT), 64 Bytes EP-1: Interrupt IN, 10 Bytes EP-2: ISO IN, 1024 Bytes EP-3: ISO IN, 98 Bytes
Speaker	Full Speed, Speaker System	4	EP-0: Control (IN, OUT), 8 Bytes EP-1: ISO OUT, 300 Bytes EP-2: Interrupt IN, 2 Bytes

Application Type	Device	Number of Endpoints	Endpoint Type and Max Packet Size
TV Tuner	High Speed, TV Tuner	5	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, Bulk OUT, 512 Bytes EP-2: ISO IN, 512 Bytes
Sound Card	High Speed, USB Interface Sound Card	5	EP-0: Control (IN, OUT), 8 Bytes EP-1: Interrupt, 3 Bytes EP-5: ISO IN, 200 Bytes EP-6: ISO OUT, 584 Bytes
Printer	High Speed, Laser Printer	4	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk OUT, Bulk IN, 512 Bytes
Camera	High Speed, Camera	5	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, Bulk OUT, 512 Bytes EP-2: Bulk IN, 512 Bytes
	Full Speed, Camera	4	EP-0: Control (IN, OUT), 64 Bytes EP-2: Bulk IN, Bulk OUT, 512 Bytes
Display Adapter	High Speed, Multi Monitor Video Adapter	4	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, 512 Bytes EP-2: Bulk OUT, 512 Byte
Keyboard	Full Speed, Keyboard	4	EP-0: Control (IN, OUT), 8 Bytes EP-1: Interrupt IN, 8 Bytes EP-2: Interrupt IN, 8 Bytes
Mouse	Low Speed, USB Mouse	3	EP-0: Control (IN, OUT), 8 Bytes EP-1: Interrupt IN, 8 Bytes
MP3 Player	High Speed, Digital Multimedia Device	5	EP-0: Control (IN, OUT), 64 Bytes EP-2: Bulk IN, Bulk OUT, 512 Bytes EP-3: Bulk IN, 8 Byte

Application Type	Device	Number of Endpoints	Endpoint Type and Max Packet Size
Phone	High Speed, Phone-1	13	0: Control (IN, OUT), 64 Bytes 1: Bulk IN, 512 Bytes 2: Interrupt IN, 64 Bytes 3: Bulk IN, 512 Bytes 4: Interrupt, 64 Bytes 5: Bulk IN, 512 Bytes 6: Bulk IN, 512 Bytes 7: Bulk IN, 512 Bytes 8: Bulk IN, 512 Bytes 9: Interrupt IN, 64-Bytes 10: Bulk IN, 512 Bytes 11: Bulk OUT, 512 Bytes
	Full Speed, Phone-2	12	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk OUT, 64 Bytes EP-2: Bulk OUT, 64 Bytes EP-3: Bulk OUT, 64 Bytes EP-4: Bulk OUT, 64 Bytes EP-5: Interrupt IN, 64 Bytes EP-6: Bulk IN, 64 Bytes EP-7: Interrupt IN, 8 Bytes EP-8: Interrupt IN, 16 Bytes EP-9: Interrupt IN, 16 Bytes EP-10: Interrupt IN, 16 Bytes
	High Speed, Phone-3	5	EP-0: Control (IN, OUT), 64 Bytes EP-1: Bulk IN, 512 Bytes EP-2: Bulk OUT, 512 Bytes EP-3: Interrupt IN, 64 Bytes

6

Signal Interfaces

This chapter describes the sub-block details of the DWC_usb3 controller. This chapter discusses the following topics:

- “[System Bus Interface Overview](#)” on page [264](#)
- “[AXI Master Interface](#)” on page [266](#)
- “[AXI Slave Interface](#)” on page [273](#)
- “[Native Master Interface – Timing](#)” on page [277](#)
- “[Native Slave Interface – Timing](#)” on page [279](#)
- “[Native Slave Interface – Combination Request](#)” on page [280](#)
- “[Description of Synopsys Test Environment Interface Signals](#)” on page [281](#)

6.1 System Bus Interface Overview

This section describes the master and slave interface of a System Bus.

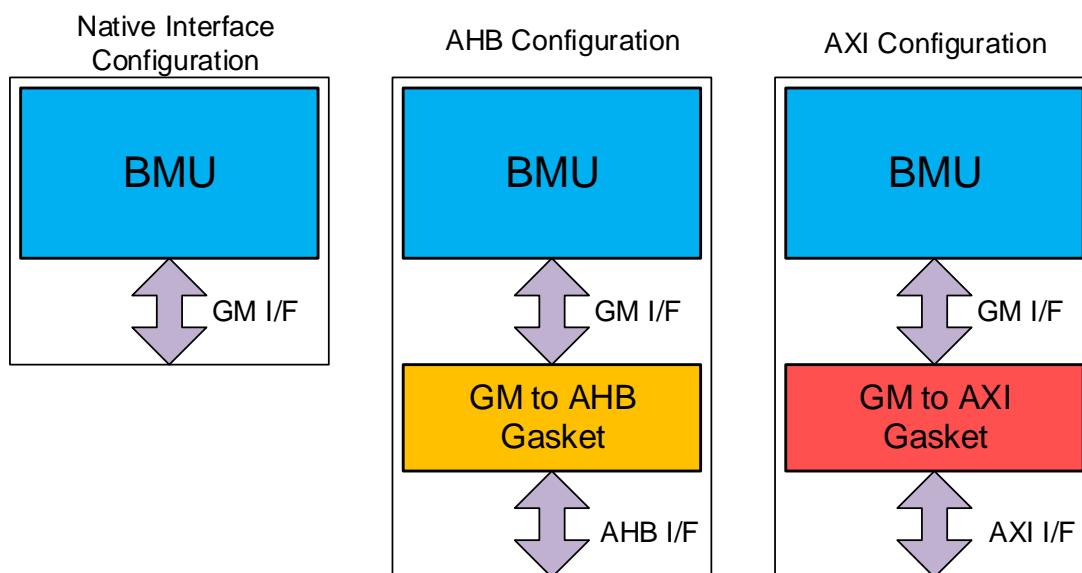
6.1.1 Master Interface

The USB 3.0 controller utilizes the System Bus Interface to initiate data traffic with the system memory. These DMA transfers are requests initiated by the List Processor (LSP) to the Buffer Management Unit (BMU). The BMU then initiates these DMA transfers via a Native Master Interface (GM). The DMA transfers are used to read/write descriptor data and read/write packet data.

Interface Configurations

The USB 3.0 controller has a configurable master interface: Native master interface (GM), AHB, and AXI. AHB and AXI utilize interface gaskets to convert from the GM interface to AHB/AXI as shown in [Figure 6-1](#).

Figure 6-1 Master Interface



The GM interface operates on a packet/descriptor level transfer. That is, each request represents a descriptor or packet size transfer. In contrast, the AHB and AXI gaskets break the GM request into multiple AHB/AXI transfers as appropriate for the interface. For example, the AHB gasket splits a 1,024 byte GM read request into multiple INCR8/INCR4 bursts depending on the burst settings.

Software can program the GSBUSCFG0 register to set the cache type settings (Cacheable, Bufferable/Posted) of the descriptor and data write/read transfers. For more details, refer to “Register Descriptions” chapter in the *DWC SuperSpeed USB 3.0 Programming Guide*.

The controller's DMA master does not insert wait cycles on the system bus write and read data paths as long as both the following conditions are met:

- The RAM bandwidth is at least as much as the system bus bandwidth.
- The master interface is AHB.

Wait cycles may appear on the master read data path in the AXI master and Native master configurations. This happens if the BMU FSM is busy when the last word of one of the two outstanding DMA read packets arrives. If the BMU FSM is busy servicing a different request and the two outstanding read packets have zero or little gap on the read data path, then wait cycles may be introduced. Since AHB does not support multiple outstanding DMA read requests, wait states are not introduced on AHB.

6.1.2 Slave Interface

The System Bus Slave Interface is used to access the CSR of the USB 3.0 controller. The USB 3.0 controller can be configured to use one of three slave bus interfaces: Native Slave (GS), AHB, and AXI.

6.2 AXI Master Interface

The AXI Master Interface is a gasket between the Native Master Interface (GM) and the AXI bus. DMA read and write accesses are initiated by the DWC_usb3 controller via the AXI master interface to read/write Descriptors, and read/write USB transfer data. The read and write channels are independent and can be active simultaneously. However, if a read access is dependent on a write access completing first, the controller does not issue the DMA read until it receives the DMA write response from AXI.

6.2.1 Burst Type

The AXI master interface only utilizes AXI burst type INCR (ARBURST/AWBURST = 2'b01).

Internally, the DWC_usb3 controller initiates up to two read and one write DMA transfers simultaneously on the Native master interface. If the AXI master interface is utilized, the DMA requests on the GM are broken up into smaller AXI burst transfers based upon the settings in the CSR register GSBUSCFG0[7:0]. For GSBUSCFG0 register definition, refer to the “Register Descriptions” chapter in the *DWC SuperSpeed USB 3.0 Programming Guide*.

There are two primary modes; Undefined Length INCR Burst Type Enabled and Disabled. These modes are defined in the following sections.

Undefined Length INCR Burst Type Enabled

When this mode is enabled, the AXI master attempts to do the largest enabled INCR burst length that is not greater than the CSR settings: INCR32/64/128/256. If the remaining bytes to transfer results in 16 beat or less, then the burst length is based on the remaining number of bytes (i.e., the CSR settings for INCR4/8/16 do not have any effect).

Note that if the number of remaining bytes results in crossing the page boundary, only the bytes remaining in the current page are accessed in the current transfer. A subsequent transfer is initiated on a new page to access the remaining bytes.

For example, for a 1k transfer in a 64b data system, if GSBUSCFG0 is 0x0000003f, and the starting address is 56 bytes before the page boundary, then the following transfers are initiated by the AXI master:

1. 7-beat burst ($64b * 7 / 8 = 56$ bytes) on the current page; remaining bytes = 72
2. 9-beat burst ($64b * 9 / 8 = 72$ bytes) on the next page to complete the transfer

Undefined Length INCR Burst Type Disabled

When this mode is disabled, then the AXI master attempts to do the largest enabled INCR burst length allowed by the CSR settings: INCR4/8/16/32/64/128/256. It does not do odd beat transfers such as INCR3/5/7/9/10/11/etc. The intent of this mode is to adhere to the cacheline size of the application.

In addition, the AXI master chooses the burst length only on burst-aligned intervals.

Note that if the number of remaining bytes results in crossing the page boundary, then only the bytes remaining in the current page are accessed in the current transfer. A subsequent transfer is initiated on the new page to access the remaining bytes.

For example, for a 1k transfer in a 64b data system, if GSBUSCFG0 is 0x0000003e, and the starting address is 56 bytes before the page boundary, then the following transfers are initiated by the AXI master:

1. 1-beat burst; remaining bytes in the transfer = 120, remaining bytes on the page = 48
2. (Aligned to 128b boundary) 2-beat burst; remaining bytes in the transfer = 104, remaining bytes on the page = 32.

3. (Aligned to 256b boundary) 4-beat burst; remaining bytes in the transfer = 72, remaining bytes on the page = 0.
4. 8-beat burst on the next page; remaining bytes in the transfer = 8.
5. 1-beat burst to complete the transfer

6.2.2 Page Boundary

The AXI specifies page boundaries at 4k, and by default the AXI master utilizes 4k page boundaries. Optionally, the controller may also be configured (CSR: GSBUSCFG1[12]) to utilize 1k page boundaries.

The AXI interface will never execute burst transfers which span across the page boundary. If the DMA transfer for packet data (as specified in the TRB) would result in the crossing of a page boundary, the transfer will be broken up into multiple AXI transfers such that no transfer will span a page boundary.

6.2.3 Outstanding Request Limit

Each read and write channels can queue up multiple address phase requests on the AXI before actually completing the associated data phases. The number of outstanding requests can be limited by programming the PipeTransLimit in the GSBUSCFG1. By default, the number of outstanding requests is set to 4. The allowable range is 1 to 16 outstanding requests.

For example, if the limit is set to 4, then up to four read and four write requests can be pushed onto the AXI bus. As the data phases are completed, then additional address phase requests are performed to maintain the maximum allowed number of outstanding requests. This process continues until the DMA transfer is completed.

6.2.4 Transaction ID

The DWC_usb3 controller uses transaction ID 0 only.

The DWC_usb3 controller does not currently support out-of-order data transfers. For this reason, the controller always uses ID=0 for all the transfers in order to guarantee in-order data transfers. No other ID values will be utilized by the AXI master interface.

The width of the ID signals is 4, by default. This can be changed by coreConsultant parameter DWC_USB3_IDWIDTH with legal values ranging from 4 to 8. Note that there is an area impact when increasing the width of the ID signal.

[Table 6-1](#) lists the number of registers generated for a default device AXI configuration where DWC_USB3_IDWIDTH is varied from 4 to 8 (default 4).

Table 6-1 Relationship Between DWC_USB3_IDWIDTH and Number of Registers

DWC_USB3_IDWIDTH	4	5	6	7	8
Number of registers	272	544	1088	2176	4352

[Table 6-2](#) demonstrates an example of the area impact of DWC_USB3_IDWIDTH on a default device AXI configuration.

Table 6-2 Impact of DWC_USB3_IDWIDTH on Area

DWC_USB3_IDWIDTH	4	8
AXI master gasket area	27,504.4	75,307.5
Total controller area	311,228	358,954
% of area	8.8%	21.0%

6.2.5 AXI Low-Power Interface

The AXI specification allows optional usage of the Low-Power Clock Control (`xm_csysreq`, `xm_csysack`, `xm_cactive`). The DWC_usb3 controller does not support this interface and the outputs are driven to 0.

6.2.6 Bus Error Address

The USB 3.0 controller provides bus address capture as a debugging aid when a bug error occurs. On the AXI write channel, if the BRESP from the AXI slave is not OKAY, or on the AXI read channel if the RRESP is not OKAY; then the `GSTS[4]` bit will be set and the first address associated with the DMA transfer that received the error will be reflected in the `GBUSERRADDR` register. Only the first error encountered captures an address.

6.2.7 AXI Master – Waveform Examples

This section provides waveform examples of the AXI Master interface.

AXI Master Write Channel

Figure 6-2 shows an example of an AXI Master Write channel transaction.

Assuming the AXI write channel is idle at the beginning of the waveform, the USB 3.0 controller initiates the beginning of a DMA write transfer (in this case packet data) by asserting the `xm_awvalid` signal. The AXI slave accepts the first address phase for a burst of length 16 (`xm_awlen == 8'h0F`) for a bus width of 64 bits (`xm_awsize == 3'h3`). Three more address phases burst transfers are requested by the USB 3.0 controller on the following three clock cycles for a total of four outstanding write burst requests. The USB 3.0 controller performs these AXI transactions for the following reasons:

- The DMA transfer is a 1k packet data transfer (not visible on AXI bus).
 - CSR register GSBUSCFG0 is set to 32'h0000_000E which enables burst transfers of length 4, 8, and 16, but does not enable bursts of size 32 or larger.
 - CSR register GSBUSCFG1 is set to 32'h0000_0300 which sets the BREQLIMIT = 4'h3 (4 outstanding).

After the first burst is requested, the USB 3.0 controller asserts the `xm_wvalid` line to indicate that the first data burst is ready to transfer. The IDs of all transactions are 0 (not shown in waveform) because the USB 3.0 controller does not utilize other IDs in order to force in-order data transfers. 16 beats of data transfers take place and the `xm_wlast` is asserted for the last beat. The data burst for the 2nd burst request immediately begins to transfer.

When the AXI slave asserts the `xm_bvalid` signal with a OKAY response, then the first burst transfer is considered completed. This reduces the current count of outstanding write bursts to 3. In response, the USB 3.0 controller pushes a fifth burst request onto the AXI write address channel by asserting `xm_awvalid`. In this way, the USB 3.0 controller tries to maximize the number of outstanding burst request on the AXI bus for maximum utilization.

Figure 6-2 Example of an AXI Master Write Channel Transaction

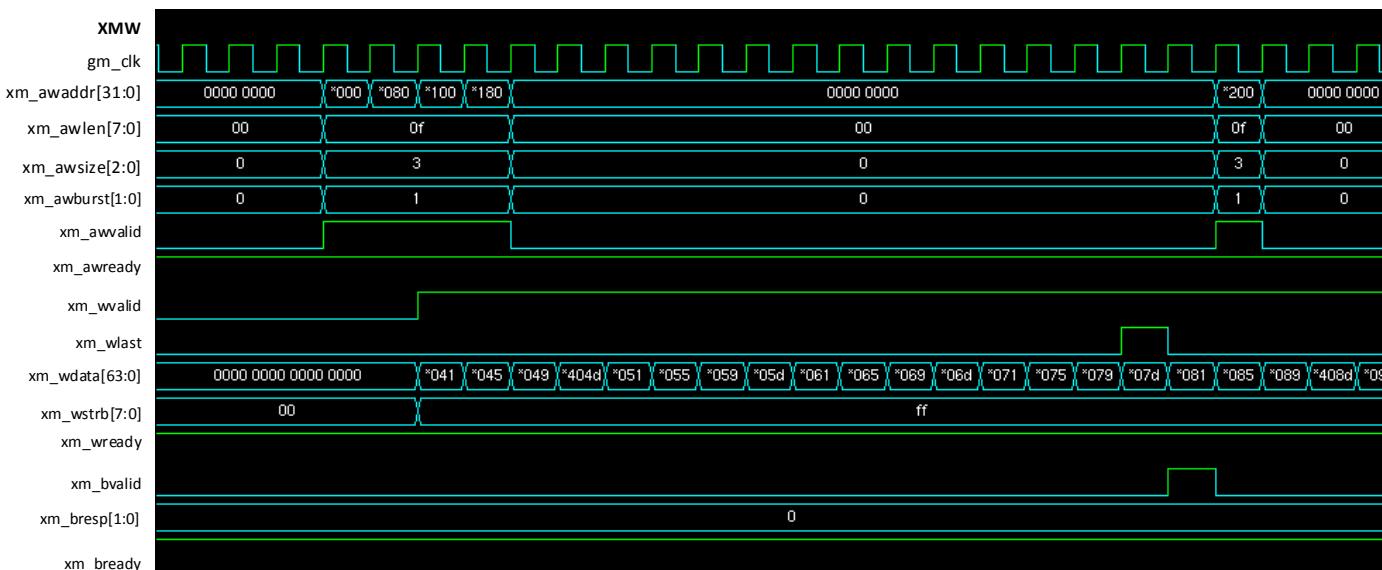
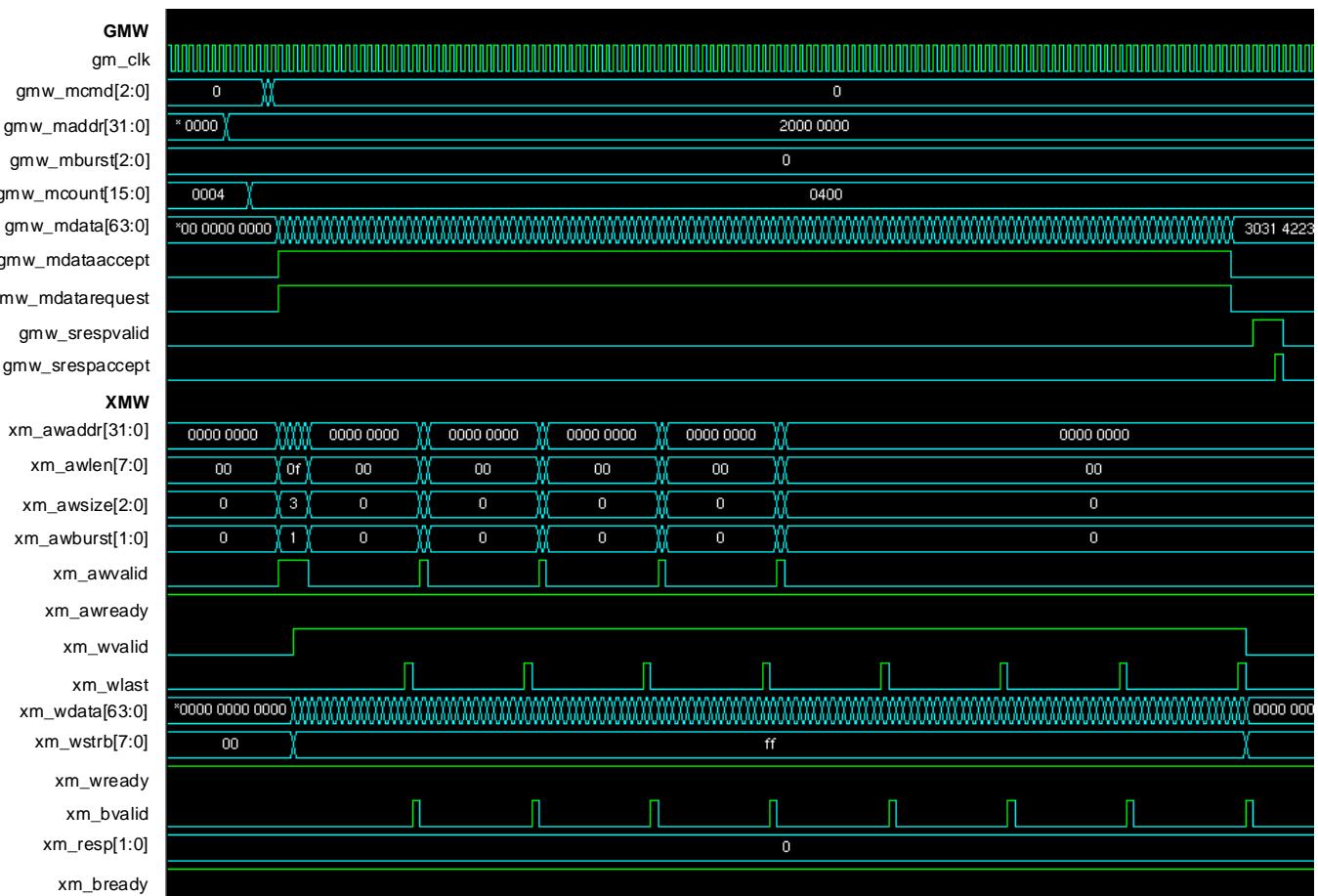


Figure 6-3 shows an example of the (hidden) Native Master Write interface (GMW) relative to the AXI Master Write channel for the same transaction as previously described.

In the beginning of the waveform, the DMA write is initiated on the GMW interface with the `gmw_mcmd` signal and the `gmw_mcount == 'h400` (1k bytes). This DMA transfer of 1k bytes is broken up into smaller AXI transactions. In this case, it is broken up into 8 16-beat bursts. After the eighth and final AXI burst is requested, the USB 3.0 controller waits for all of the data associated with the DMA to complete. The USB 3.0 controller will service one outstanding DMA transfer at a time, which can result into multiple AXI transactions as determined by CSR settings, buffer address offset, and bus conditions.

Figure 6-3 Example of a Native Master Write Interface Transaction



AXI Master Read Channel

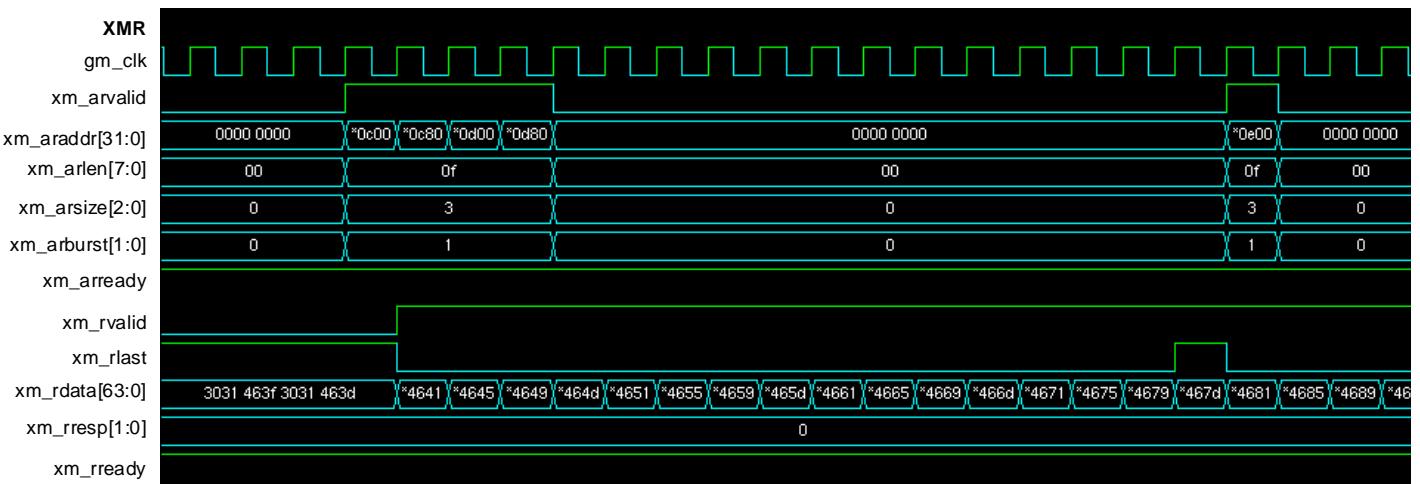
Figure 6-4 shows an example of an AXI Master Read channel transaction.

Assuming the AXI read channel is idle at the beginning of the waveform, the USB 3.0 controller initiates the beginning of a DMA read transfer (in this case packet data) by asserting the `xm_arvalid` signal. The AXI slave accepts the first address phase for a burst of length 16 (`xm_arlen == 8'h0F`) for a bus width of 64-bits (`xm_rwsiz == 3'h3`). Three more address phases burst transfers are requested by the USB 3.0 controller on the following three clock cycles for a total of four outstanding read burst requests. The USB 3.0 controller performs these AXI transactions for the following reasons:

- The DMA transfer is a 1k packet data transfer (not visible on AXI bus).
- CSR register `GSBUSCFG0` is set to `32'h0000_000E` which enables burst transfers of length 4, 8, and 16, but does not enable bursts of size 32 or larger.
- CSR register `GSBUSCFG1` is set to `32'h0000_0300` which sets the `BREQLIMIT = 4'h3` (4 outstanding).

Eventually the AXI slave returns the first burst of read data by asserting `xm_rvalid`. On the last beat of the burst, the `xm_rlast` is asserted and the `xm_rresp` indicates OKAY. This marks the completion of the first read burst, reducing the current outstanding read bursts to 3. In response, the USB 3.0 controller pushes a fifth read burst request onto the AXI read channel to maintain the `BREQLIMIT` of 4 outstanding. This process continues until the DMA transfer is fulfilled.

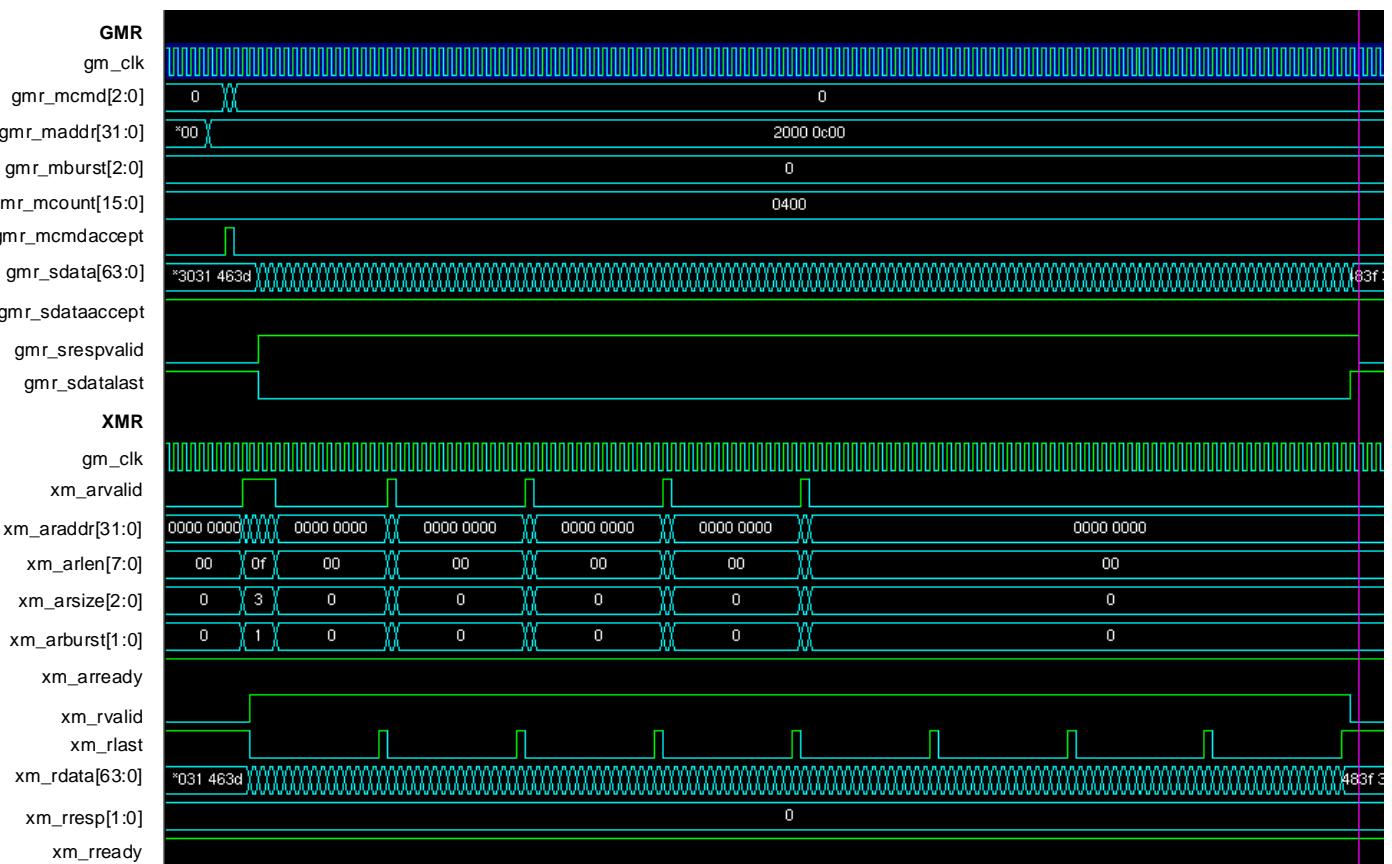
Figure 6-4 Example of an AXI Master Read Channel Transaction



[Figure 6-5](#) shows an example of the (hidden) Native Master Read interface relative to the AXI Master Read channel for the same transaction as previously described.

In the beginning of the waveform, the DMA read is initiated on the GMR interface with the gmr_mcmand signal and the gmr_mcount == 'h400 (1k bytes). This DMA transfer of 1k bytes is broken up into smaller AXI transactions. In this case, it is broken up into eight 16-beat bursts. After the eighth and final AXI burst is requested, the USB 3.0 controller waits for all of the data associated with the DMA to complete. The USB 3.0 controller services one or two outstanding DMA transfer at a time (depending upon controller configuration), which can result into multiple AXI transactions as determined by CSR settings, buffer address offset, and bus conditions.

Figure 6-5 Example of a Native Master Read Interface Transaction



6.3 AXI Slave Interface

The purpose of the AXI Slave Interface is to access the CSRs of the DWC_usb3 controller. It is assumed to be a low bandwidth, and a low gate count interface. Internally, the AXI slave interface is a gasket to the Native Slave Interface (GS) of the controller and it performs the translation from AXI to GS.

6.3.1 Transfer ID

By default, the width of the ID signals (`xs_arid`, `xs_rid`, `xs_awid`, `xs_wid`, `xs_bid`) is 4 bits, in accordance with the AXI3 specification. This width can be customized using the coreConsultant parameter `DWC_USB3_SDWIDTH`, and has a valid range from 0 to 16 bits. This parameter is independent of the ID width parameter for the AXI Master Interface and may be changed regardless of the ID width on the AXI master interface.

The AXI slave interface operates on one transaction at a time. Hence out-of-order transfers are not relevant to the AXI slave. All ID values are supported, but only one transfer for all IDs will be active at a time.

Note that there is no significant area impact when increasing the width of the ID signal. The actual impact is 1 register of width `DWC_USB3_SDWIDTH`. No additional data registers or FIFOs are changed as a result of changing the ID width.

6.3.2 Burst Support

The AXI slave supports burst transfers for compatibility. This is not a high bandwidth interface, as each beat of the burst transfer is translated into single transfers internally on the Native Slave Interface (GS).

The AXI slave interface operates on one transaction at a time. From idle, once the address phase of a transfer starts, the AXI slave will not accept any more address phase transfers on read or write channels until the data and response phases complete for the current transfer.

6.3.3 Read Vs. Write Channel

The AXI specification allows independent read and write channels. The AXI slave interface is defined to be a low resource, low bandwidth gasket between the AXI bus and the Native slave interface. For this reason, the AXI slave interface performs only one transaction (either read or write) at a time. That is, if a read (or a write) transfer is active, then the subsequent read and write transfers are blocked until the active read transfer completes. When both read and write access are requested on the same cycle and there are no currently active read or write CSR transactions, then the write access takes priority.

6.3.4 Slave Responses

The AXI slave utilizes the following possible responses on the BRESP signals:

- OKAY – The transfer has completed successfully
- EXOKAY – Not used
- SLVERR – The data phase ID does not match with the address phase ID. This is a fatal error and is not recoverable. Because the AXI slave interface only operates on one transfer at a time, it implies that there is a system problem external to the DWC_usb3 controller.
- DECERR – Not used

6.3.5 AXI Low-Power Interface

The AXI specification allows for optional usage of the Low-Power Clock Control (`xs_csysreq`, `xs_csysack`, `xs_active`). This interface is not supported, and the outputs are driven to 0.

6.3.6 Miscellaneous Info Signals

The Miscellaneous Info signals (`xs_armisc_info`, `xs_awmisc_info`, `xs_rwmisc_info`, `xs_bwmisc_info`) are not currently supported but are reserved for potential future use. All miscellaneous info outputs are currently driven to 0.

6.3.7 AXI Slave – Waveform Examples

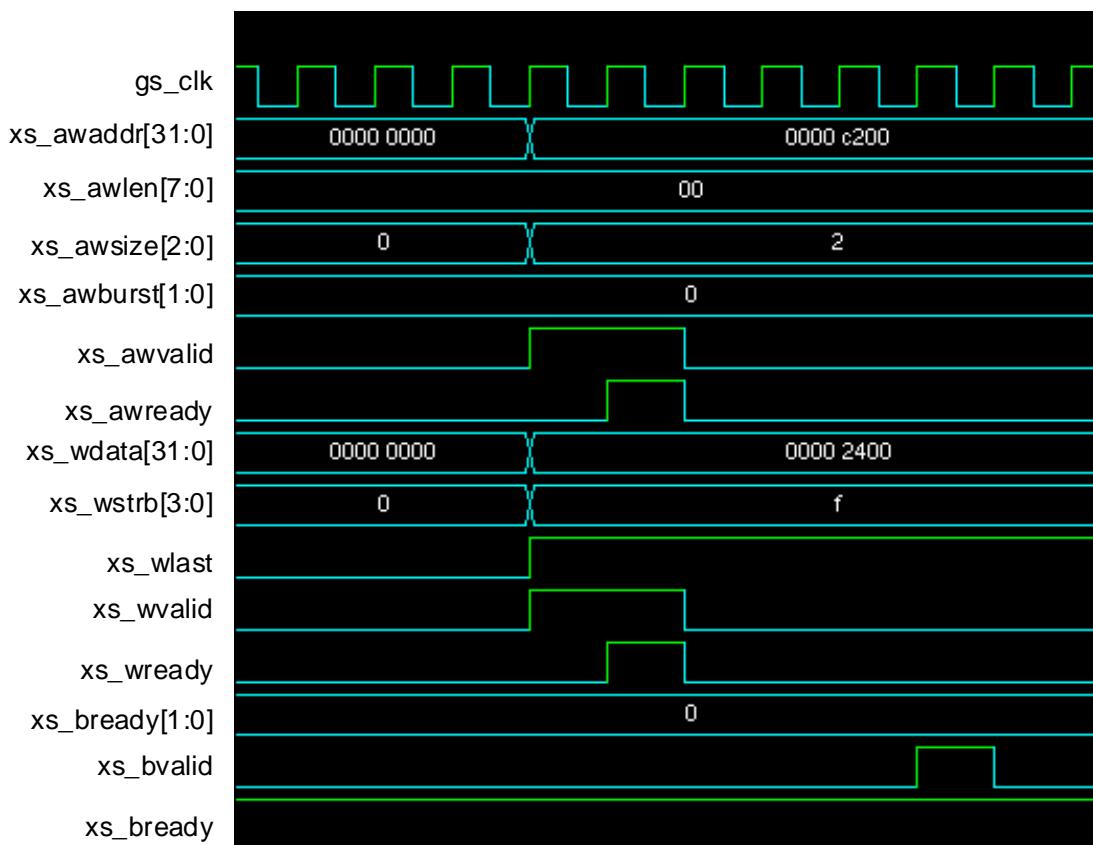
This section provides waveform examples of the AXI Slave interface.

AXI Slave Write Channel

[Figure 6-6](#) on page [275](#) is an example of a CSR write operation by the application to the USB 3.0 controller.

The application initiates the CSR write by asserting the `xs_awvalid` signal. A cycle later, the USB 3.0 controller AXI slave interface accepts the request by assertion the `xs_awready` signal, and then immediately de-asserts it. In this example, the application has simultaneously asserted the `xs_wvalid` signal and the USB 3.0 controller accepts the data on the next cycle. Internally, the write data is written to the appropriate internal CSR location. When the process completes, the USB 3.0 controller responds with the `xs_bvalid` and `xs_bresp` of OKAY to complete the transaction.

Because the CSR accesses are intended to be a low bandwidth bus, the USB 3.0 controller only services one request at a time including both Read and Write AXI slave channels. That means during the time that this write transaction takes place, the USB 3.0 controller will keep the `xs_awready` and `xs_arready` signals deasserted until the current active transfer completes.

Figure 6-6 Example of a CSR Write Operation

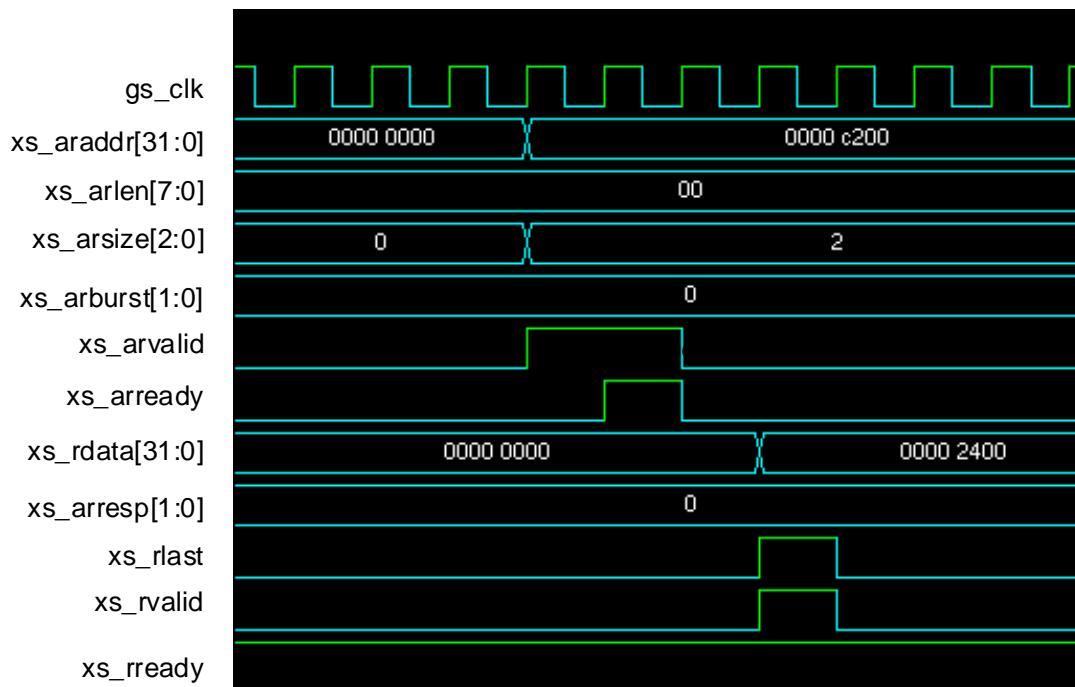
AXI Slave Read Channel

Figure 6-7 is an example of a CSR read operation by the application to the USB 3.0 controller.

The application initiates the CSR read by asserting the `xs_arvalid` signal. A cycle later, the USB 3.0 controller AXI slave interface accepts the request by assertion the `xs_arready` signal, and then immediately de-asserts it. The USB 3.0 controller then fetches the CSR read data and presents it on the `xs_rdata` bus to complete the transfer.

Because the CSR accesses are intended to be a low bandwidth bus, the USB 3.0 controller only services one request at a time including both Read and Write AXI slave channels. That means during the time that this write transaction takes place, the USB 3.0 controller keeps the `xs_awready` and `xs_arready` signals deasserted until the current active transfer completes.

Figure 6-7 Example of a CSR Read Operation

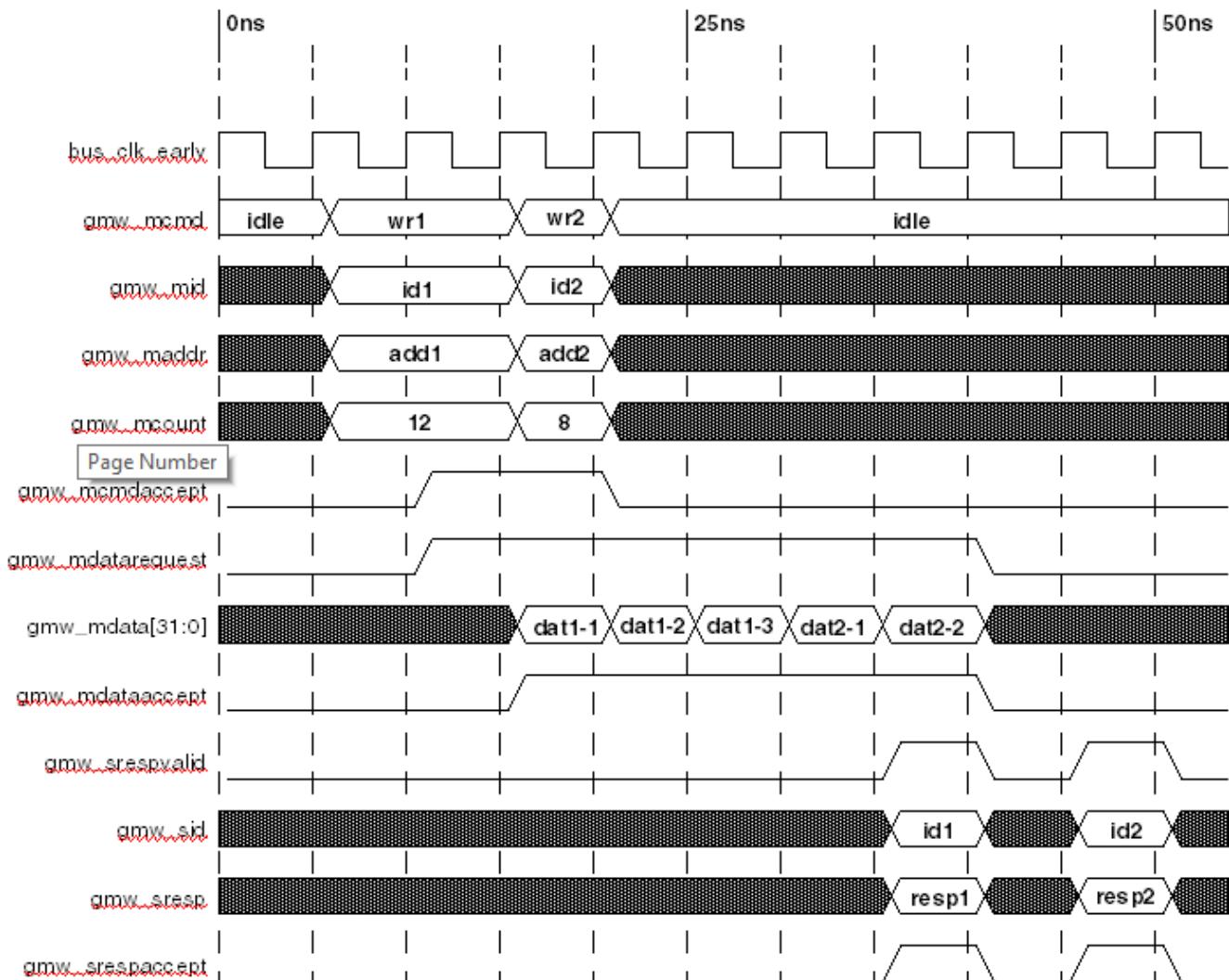


6.4 Native Master Interface – Timing

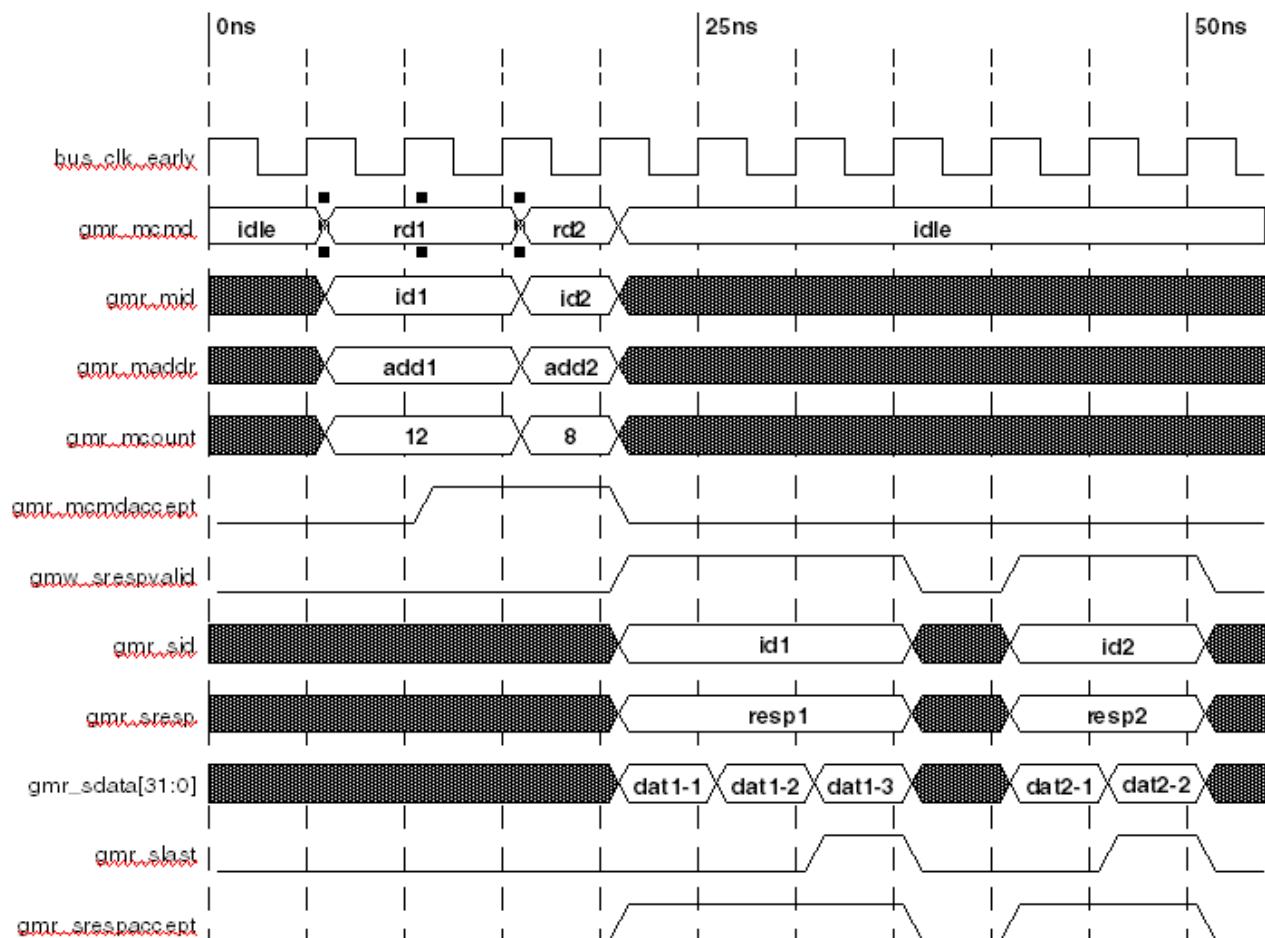
Each Native Master Interface (GM-IF) channel is a point-to-point connection from a single source to a single sink. The channels offer two-way flow control, similar to the valid/ready handshake on the AXI bus.

[Figure 6-8](#) shows two write requests of three and four burst sizes on the GM-IF Write Request/Response Channels.

Figure 6-8 GM Interface Write Access



[Figure 6-9](#) on page 278 shows two read requests of three and four burst sizes on the GM-IF Read Request/Response Channels.

Figure 6-9 GM Interface Read Access

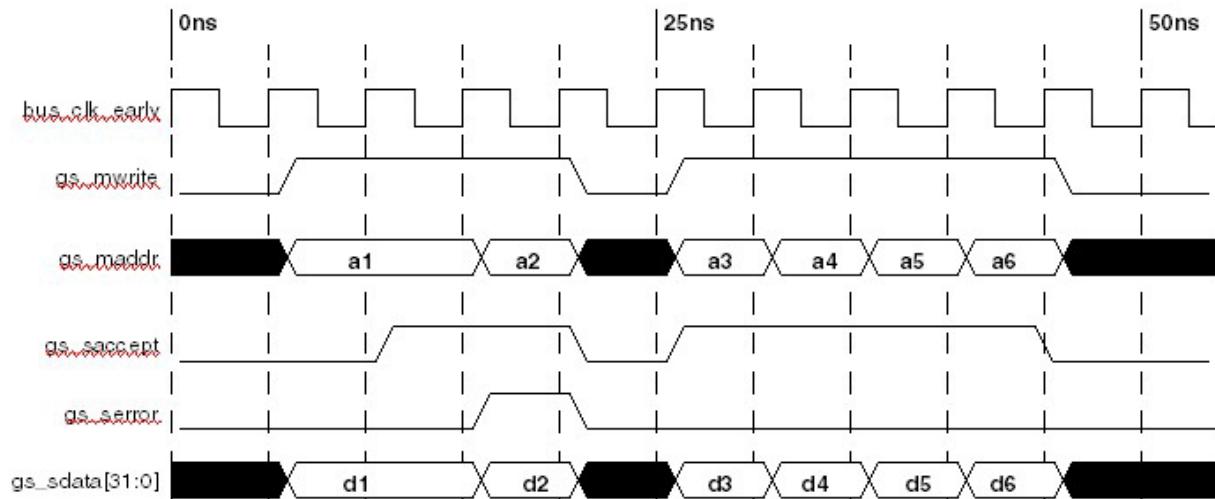
6.5 Native Slave Interface – Timing

The Native Slave Interface (GS-IF) channels are point-to-point connection from a single source to a single sink. The channels offer two-way flow control, similar to the valid/ready handshake on the AXI bus.

DWC_usb3_bus_gs provides address and byte-enables for each data access, thus creating simpler register access interface. The write or read access can be extended by a slave by driving the gs_saccept signal 0. The data and address are valid together in the same cycle and are nor pipelined as in AHB.

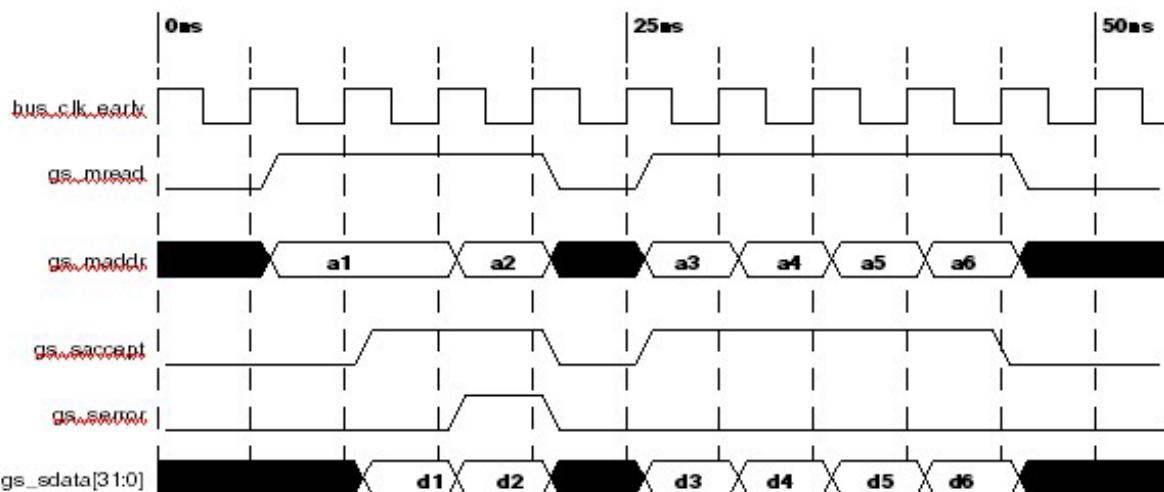
[Figure 6-10](#) shows two write requests of 3 and 4 burst size on the GS-IF Write Request/Response Channels.

Figure 6-10 GS-IF Write Access



[Figure 6-11](#) shows two read requests of 3 and 4 burst size on the GS-IF Read Request/Response Channels.

Figure 6-11 GS-IF Read Access



6.6 Native Slave Interface – Combination Request

If an application has less logic in the request path, it can choose combination-request interface. This avoids one clock latency.

6.7 Description of Synopsys Test Environment Interface Signals

The top-level debug port has information on the internal state of the controller, which can be used for debug purposes. The debug signal is defined as shown below:

```
wire [USB3_DEBUG_WIDTH-1:0] debug = {
    ssic_suspend_clk_en,      // DWC_USB3_NUM_SSIC_PORTS, Valid only for SSIC. Indicates
    suspend-enabled condition
    sb2m3l_ssic_gear,        // Valid only for SSIC.
    (sb2m3l_ssic_gear[DWC_USB3_NUM_U3_ROOT_PORTS*2-1:0])           // Indicates current operating HS-GEAR for each U3 port.
    // 2'b00- HS-G1_only, 2'b01- HS-G1, 2'b10- HS-G2, 2'b11- HS-G3
    rxdet_poll_fail_us,       // RX_DET failed for hub upstream port.
    hub_desc_rd_done,         // Hub only. Indicates descriptor read from ROM done.
    mphy_cmd_state,          // Valid for SSIC only, MPHY state as seen by controller
    dbg_ram_clk_gated,       // Indicates ram_clk is actually being gated internally (1 bit)
    pmgt_gate_bus_clk,        // pmgt_captive (1 bit)
    pmgt_gate_bus_clk_ext,   // pll off for bus_clk
    suspend_clk_en_out,       // DWC_USB3_NUM_U3_ROOT_PORTS?
    bmu_bus_idle,             // BMU idle (1 bit)
    gm_idle,                  // GM/AXIM idle (1 bit)
    //USB2 states
    b2rl_cur_mode,            // 1'b0 - Host, 1'b1 - Device
    utmi_suspend_n,           // DWC_USB3_NUM_U2_ROOT_PORTS, Active low on suspend
    utmi_suspend_com_n,       // Active low, goes inactive on all ports' suspend
    utmi_l1_suspend_com_n,    // Active low, goes inactive on all ports' L1 or mixed L1/L2 suspend
    usb2_enumspeed,           // Device enumerated speed (HS = 2'b00,
    // FS = 2'b01,
    // LS = 2'b10,
    // FS48 = 2'b11)
    u2_dssr_state,             // [3:0] Device DSSR port state
    u2mac_txrx_state,          // 5*DWC_USB3_NUM_U2TIS, USB2 MAC state
    u2_prt_state,              // 5*DWC_USB3_NUM_U2_ROOT_PORTS, Current USB2 port state
    //USB3 states
    gsts_buserraddvld,         // 1b (DWC_USB3_DEBUGIO_DEV_BUS_ERR)
    debug_mclk_usof_number,    // [18:0] (DWC_USB3_DEBUGIO_SOF_NUM)
    lldb_link_state,            // 4*DWC_USB3_NUM_U3_ROOT_PORTS (DWC_USB3_DEBUGIO_LINK_STATE)
    lldb_sub_state              // 4*DWC_USB3_NUM_U3_ROOT_PORTS (DWC_USB3_DEBUGIO_LINK_SUB_STATE)
};
```

The width of the debug signal is equal to the sum of the individual signal widths. The formula used to calculate the width of the debug signal is:

```
DWC_USB3_DEBUG_WIDTH= ( `DWC_USB3_DEBUGIO_SSIC_SUSPEND_CLK_EN_W +
`DWC_USB3_DEBUGIO_SSIC_HS_GEAR_W + `DWC_USB3_DEBUGIO_RXDET_POLL_FAIL_US_W +
`DWC_USB3_DEBUGIO_HUB_DESC_RD_DONE_W + `DWC_USB3_DEBUGIO_MPHY_CMD_STATE_W +1+ 1 +
1 + `DWC_USB3_NUM_U3_ROOT_PORTS + 1 + 1 + 1 + `DWC_USB3_NUM_U2_ROOT_PORTS + 1 + 1
+ 2 + 4 + 5*DWC_USB3_NUM_U2TIS + 5*DWC_USB3_NUM_U2_ROOT_PORTS + 1 + 19 +
```

```
4 * `DWC_USB3_NUM_U3_ROOT_PORTS + 4 * `DWC_USB3_NUM_U3_ROOT_PORTS)
```

Where,

```
DWC_USB3_DEBUGIO_SSIC_SUSPEND_CLK_EN_W = ((`DWC_USB3_NUM_SSIC_PORTS != 0) ?
```

```
1 * `DWC_USB3_NUM_SSIC_PORTS : 1)
```

```
DWC_USB3_DEBUGIO_SSIC_HS_GEAR_W = (2 * `DWC_USB3_NUM_U3_ROOT_PORTS)
```

```
DWC_USB3_DEBUGIO_HUB_DESC_RD_DONE_W = 1
```

```
DWC_USB3_DEBUGIO_RXDET_POLL_FAIL_US_W = 1
```

```
DWC_USB3_DEBUGIO_MPHY_CMD_STATE_W = ((`DWC_USB3_NUM_SSIC_PORTS != 0) ?
```

```
3 * `DWC_USB3_NUM_SSIC_PORTS : 3)
```

```
DWC_USB3_NUM_U2TIS = (DWC_USB3_MODE == 0) ? 1 : (DWC_USB3_NUM_HS_USB_INSTANCES +  
DWC_USB3_NUM_FSLS_USB_INSTANCES)
```

Table 6-3 shows examples of debug width calculation.

Table 6-3 Examples for Debug Width Calculation

Item Number	Signal Name	Signal Width	Examples	
			Single-port Device Configuration	Single-port Host, DRD, or OTG Configuration
1	ssic_suspend_clk_en	((DWC_USB3_NUM_SSIC_PORTS != 0) ? 1 * `DWC_USB3_NUM_SSIC_PORTS : 1)	1	1
1	sb2m3l_ssic_gear	(2 * `DWC_USB3_NUM_U3_ROOT_PORTS)	2	2
2	rxdet_poll_fail_us	1	1	1
3	hub_desc_rd_done	1	1	1
4	mpphy_cmd_state[2:0]	((DWC_USB3_NUM_SSIC_PORTS != 0) ? 3 * `DWC_USB3_NUM_SSIC_PORTS : 3)	3	3
5	dbg_ram_clk_gated	1	1	1
6	pmgt_gate_bus_clk	1	1	1
7	pmgt_gate_bus_clk_ext	1	1	1
8	suspend_clk_en_out	DWC_USB3_NUM_U3_ROOT_PORTS	1	1
9	bmu_bus_idle	1	1	1
10	gm_idle	1	1	1
11	b2rl_cur_mode	1	1	1
12	utmi_suspend_n	DWC_USB3_NUM_U2_ROOT_PORTS	1	1
13	utmi_suspend_com_n	1	1	1

Item Number	Signal Name	Signal Width	Examples	
			Single-port Device Configuration	Single-port Host, DRD, or OTG Configuration
14	utmi_l1_suspend_com_n	1	1	1
15	usb2_enumspeed	2	2	2
16	u2_dssr_state	4	4	4
17	u2mac_txrx_state	5*DWC_USB3_NUM_U2TIS	5	10
18	u2_prt_state	5*DWC_USB3_NUM_U2_ROOT_PORTS	5	5
19	gsts_buserraddvld	1	1	1
20	debug_mclk_usof_number	19	19	19
21	ltdb_link_state	4*DWC_USB3_NUM_U3_ROOT_PORTS	4	4
22	ltdb_sub_state	4*DWC_USB3_NUM_U3_ROOT_PORTS	4	4
Total	debug	DWC_USB3_DEBUG_WIDTH ^a	62	67

a. The width of the debug signal is equal to the sum of the individual signal widths as shown in the previously-mentioned formula.

Table 6-4 describes the Synopsys Test Environment Interface signals.

Table 6-4 Debug Port Detail

Signal Name	I/O	Description
ssic_suspend_clk_en	O	<p>Function: SSIC Suspend Clock Enable This signal reflects the suspend enable condition for SSIC. Note: This signal is valid only when DWC_USB3_NUM_SSIC_PORTS != 0.</p> <p>Active State: High Synchronous to: ssic_soc_pa_clk Registered: Yes</p>

Signal Name	I/O	Description
sb2m3l_ssic_gear [DWC_USB3_NUM_U3_ROOT_PORTS*2:1:0]	O	<p>Function: Valid only for SSIC. It indicates current operating HS-GEAR for each U3 port. The width of the port is represented by (2*DWC_USB3_NUM_U3_ROOT_PORTS).</p> <p>Active State: N/A</p> <p>Valid Values: 2'b00- HS-G1_only 2'b01- HS-G1 2'b10- HS-G2 2'b11- HS-G3</p> <p>Synchronous to: mac3_clk</p> <p>Registered: Yes?</p>
hub_desc_rd_done	O	<p>Function: This bit is valid only in Hub mode. It indicates the descriptor read from the ROM is completed during initialization.</p> <p>Active State: High</p> <p>Synchronous to: mac3_clk</p> <p>Registered: Yes</p>
rxdet_poll_fail_us	O	<p>Function: This bit is valid only in Hub mode. It indicates either of the following:</p> <ul style="list-style-type: none"> ■ The upstream port receiver detection failed after maximum number of attempts. ■ A polling timeout occurred. <p>Active State: High</p> <p>Synchronous to: mac3_clk</p> <p>Registered: Yes</p>
mphy_cmd_state[2:0]	O	
dbg_ram_clk_gated	O	<p>Indicates that the internal ram_clk is being gated. It indicates whether ram_clk_gated is turned on or off. This signal is used by the test environment to verify low power operation.</p>
pmgt_gate_bus_clk	O	<p>Indicates that the conditions are appropriate for some internal modules to receive a gated bus_clk. This signal is used by the test environment to verify low power operation.</p>
pmgt_gate_bus_clk_ext	O	<p>Indicates that the conditions are appropriate for bus_clk pll to be turned off. This signal is asserted when the USB2 port is in disconnect/L2/L1 Suspend and SS port is in Disconnect/U3. This signal is used by the test environment to verify low power operation.</p>

Signal Name	I/O	Description
suspend_clk_en_out[DWC_USB3_NUM_U3_ROOT_PORTS-1:0]	O	<p>Internal Suspend Clock Enable</p> <p>Function: This signal is brought out for debug purposes. It indicates when the RTL is switching to suspend clock.</p> <p>Active State: High</p> <p>Synchronous to: mac_clk</p> <p>Registered: Yes</p>
bmu_bus_idle	O	<p>USB 3.0 BMU GM Bus Interface Idle Indicator</p> <p>Function: Indicates whether the BMU has no currently active transactions to perform on the system bus (GM/AHB Master/AXI Master). This signal is deasserted just before any activity is seen on the system bus, and is asserted after the operation on the system bus is completed.</p> <p>Active State: High</p> <p>Synchronous to: bus_clk</p> <p>Registered: Yes</p>
gm_idle	O	<p>USB 3.0 GM/AHB Master/AXI Master Bus Interface Idle Indicator</p> <p>Function: This signal is most useful when the core is configured with either AHB or AXI gasket. This signal indicates when the system bus gasket of the core is not involved in any current transactions on the system bus. This signal is deasserted once the BMU initiates a transaction on the (internal) GM bus, and is asserted when all the transactions are completed on the system bus and the internal GM bus.</p> <p>Active State: High</p> <p>Synchronous to: bus_clk</p> <p>Registered: Yes</p>
b2rl_cur_mode	O	<p>Current Mode</p> <p>Function: Current Mode</p> <ul style="list-style-type: none"> ■ 1'b0: Host ■ 1'b1: Device <p>Active State: N/A</p> <p>Synchronous to: bus_clk_early</p> <p>Registered: Yes</p>
utmi_suspend_n[DWC_USB3_NUM_U2_ROOT_PORTS-1:0]	O	<p>USB 2.0 Port Suspend</p> <p>Function: USB 2.0 PHY Reset</p> <p>Used by USB 2.0</p> <p>Active State: Low</p> <p>Synchronous to: No</p> <p>Registered: No</p>

Signal Name	I/O	Description
utmi_suspend_com_n	O	<p>Common suspend</p> <p>Function: For device mode, this signal is same as inverted utmi_suspend_n[0]. For OTG mode of operation, it is same as for host mode operation, but depends only on Port0 of SS and Port0 of non-SS connection. For host mode, this signal is asserted (low) if either of the following conditions is met:</p> <ul style="list-style-type: none"> ■ Any of the host ports (either 3.0 or 2.0 ports) is not suspended. (In this case, for the USB 3.0 port, suspend is equivalent to P3; for the USB 2.0 port, it is either L2 suspend or disconnected state if DWC_USB3_SUSPEND_ON_DISCONNECT_EN is high). ■ Hardware LPM is enabled <p>For host mode, this signal is high if all of the following conditions are met:</p> <ul style="list-style-type: none"> ■ All the ports are suspended (or disconnected if `DWC_USB3_SUSPEND_ON_DISCONNECT_EN is 1) ■ GUSB2PHYCFGn.SusPHY is 1 for all the ports ■ All the SS PHYs are in P3 ■ Hardware LPM is not enabled <p>Note:</p> <ul style="list-style-type: none"> ■ The polarity of this signal is different than the utmi_suspend_n signal. ■ In OTG and host mode, for details on the exceptions to the dependency on SS port and USB 2.0 HW LPM, based on GCTL.SOFIT-PSYNC, see GCTL register description in the “Register Descriptions” chapter of the Programming Guide. <p>Active State: N/A</p> <p>Synchronous to: Asynchronous</p> <p>Registered: No</p>

Signal Name	I/O	Description
utmi_l1_suspend_com_n	O	<p>Common L1 suspend</p> <p>Function: For device mode, this signal is same as inverted utmi_l1_suspend_n[0]. For OTG mode of operation, it is same as for host mode operation, but depends only on Port0 of SS and Port0 of non-SS connection. For host mode, this signal is asserted (low) if any of the following conditions is met:</p> <ul style="list-style-type: none"> ■ Any of the host ports (either 3.0 or 2.0 ports) is not suspended. (In this case, for the USB 3.0 port, suspend is equivalent to P3; for the USB 2.0 port, it is L1/L2 Suspend). ■ Hardware LPM is enabled <p>For host mode, this signal is high if all of the following conditions are met:</p> <ul style="list-style-type: none"> ■ All the USB 2.0 ports are in L1/L2 suspend (or in disconnected state if `DWC_USB3_SUSPEND_ON_DISCONNECT_EN is high) with at least one port in L1 suspend/sleep state ■ GUSB2PHYCFGn.EnblSlpM is 1 for all the ports ■ All the SS PHYs are in P3 ■ Hardware LPM is not enabled <p>Note:</p> <ul style="list-style-type: none"> ■ The polarity of this signal is different than the utmi_l1_suspend_n signal. ■ In OTG and host mode, for details on the exceptions to the dependency on SS port and USB 2.0 HW LPM, based on GCTL.SOFIT-PSYNC and GFLADJ.GFLADJ_REF_CLK_LPM_SEL, see 'SOFITPSYNC' and 'GFLADJ_REFCLK_LPM_SEL' description. <p>Active State: N/A</p> <p>Synchronous to: Asynchronous</p> <p>Registered: No</p>
usb2_enumspeed[1:0]	O	<p>Device Enumerated Speed</p> <p>Function:</p> <ul style="list-style-type: none"> ■ 2'b00: HS ■ 2'b01: FS ■ 2'b10: LS ■ 2'b11: FS48 (unused) <p>Active State:</p> <p>Synchronous to: mac2_clk</p> <p>Registered: No</p>

Signal Name	I/O	Description
u2_dssr_state[3:0]	O	<p>Device DSSR Port Status</p> <p>Function: Device DSSR Port Status</p> <ul style="list-style-type: none"> ■ 4'b0000: DEV_INIT ■ 4'b0001: DEV_IDLE ■ 4'b0010: PULLUP_EN ■ 4'b0011: CHK_BUS ■ 4'b0100: SEND_CHIRP ■ 4'b0101: WAIT_HST_CHIRP ■ 4'b0110: SUSPENDED ■ 4'b0111: RESUMING ■ 4'b1000: FLS_RESET ■ 4'b1001: DLINE_PULSING ■ 4'b1010: PULSING_DONE ■ 4'b1011: DEV_CHIRP_END ■ 4'b1100: DEV_CHIRP_DONE ■ 4'b1101: REMOTE_WAKEUP ■ 4'b1110: RMWAKEUP_DONE <p>Active State: High</p> <p>Synchronous to: mac2_clk</p> <p>Registered: Yes</p>

Signal Name	I/O	Description
u2mac_txrx_state[5* -DWC_USB3_NUM_U2TIS-1:0]	O	<p>USB2.0 MAC Status</p> <p>Function: 5-bits per port.</p> <p>DWC_USB3_NUM_U2TIS is 1 for device only configuration.</p> <p>For Host/DRD/OTG configuration, DWC_USB3_NUM_U2TIS is sum of all the HS and FS/LS bus instances (DWC_USB3_NUM_HS_USB_INSTANCES + DWC_USB3_NUM_FSLS_USB_INSTANCES).</p> <p>For single-port configuration, this is 2 in Host/DRD/OTG configuration. The lower bits are for FS/LS MAC(s) and upper bits are for HS MAC(s) ({5 * Number of HS-Instance, 5 * Number of FS/LS Instance}).</p> <p>USB 2.0 MAC Status:</p> <ul style="list-style-type: none"> ■ 5'b00000: IDLE ■ 5'b00001: SEND_SPLT(Host mode only) ■ 5'b00010: TKN2TKN(Host mode only) ■ 5'b00011: SEND_CRC5(Host mode only) ■ 5'b00100: TKN2DAT(Host mode only) ■ 5'b00101: WAIT_DPKT(Device mode only) ■ 5'b00110: RECV_ERR ■ 5'b00111: RECV_DATA ■ 5'b01000: SEND_HSHK ■ 5'b01001: SEND_EOP ■ 5'b01010: SEND_DATA ■ 5'b01011: WAIT_HSHK ■ 5'b01100: CHK_CRC16 ■ 5'b01101: WAIT_DATA ■ 5'b01110: TESTMOD(Device mode only) ■ 5'b01111: WAIT_BUSIDLE ■ 5'b10000: WAIT_TURNAROUND(Device mode only) ■ 5'b10001: DATA_TO_STS ■ 5'b10010: WAIT_EOP(Host mode only) <p>Active State: N/A</p> <p>Synchronous to: mac2_clk</p> <p>Registered: Yes</p>

Signal Name	I/O	Description
u2_prt_state[5*'DWC_USB3_NUM_U2_ROOT_PORTS-1:0]	O	<p>USB2.0 Port Status</p> <p>Function: 5-bits per port.</p> <p>USB2.0 Port Status:</p> <ul style="list-style-type: none"> ■ 5'b00000: PRT_DISCON ■ 5'b00001: DISABLE ■ 5'b00010: FSLS_RESET ■ 5'b00011: WAIT_CHIRP ■ 5'b00101: DEV_CHIRP ■ 5'b00110: HST_CHIRP_J ■ 5'b00111: HST_CHIRP_K ■ 5'b01000: HST_CHIRP_END ■ 5'b01001: ENABLE ■ 5'b01010: AEOF ■ 5'b01011: SOF ■ 5'b01100: SUSPEND ■ 5'b01101: RESUME ■ 5'b01110: RESUME_END ■ 5'b01111: RESUME_DONE ■ 5'b10000: TEST_PKT ■ 5'b10001: PRE_ENABLE ■ 5'b10010: SOF_END ■ 5'b10011: TEST ■ 5'b10101: HST_CHIRP_DONE ■ 5'b10110: HST_CHIRP_SWITCH_OPMODE <p>Active State: N/A</p> <p>Synchronous to: mac2_clk</p> <p>Registered: Yes</p>
gsts_buserraddvld	O	<p>Bus error</p> <p>Function: Bus Error</p> <p>Active State: Low</p> <p>Synchronous to: bus_clk_early</p> <p>Registered: Yes</p>

Signal Name	I/O	Description
debug_mcclk_usof_number[18:0]	O	<p>SOF Number</p> <p>A mac_clk cycle-accurate reference, which could be used to for instance, lock another PLL to the SOF rate.</p> <p>Device Mode:</p> <p>Operating in SS mode, the signal is incremented when the ITP is received or ITP is not observed with in 125128 ns. The clock reference is 125 MHz mac3_clk. If the ITP received has its delta offset non-zero and if it is not marked with delay (DL) and deferred (DF) bits, then the delay offset from the ITP timestamp is used to adjust the next micro-frame duration.</p> <p>Operating in HS mode, the signal is incremented when the uSOF is received or uSOF is not observed with in 125200ns. The clock reference is 30/60MHz utmi/ulpi clock.</p> <p>Operating in FS mode, the signal is incremented by 8 when the SOF is received SOF is not observed with in 128500ns. the clock reference is 30/60MHz utmi/ulpi clock.</p> <p>When mac_clk is off, the counter freezes. There will not be changes to this signal during this state. If the core is in suspended state for example mac_clk can stop. If the device was operating in super-speed mode, then mac_clk switches to suspend_clk. In this case, the signal changes when 15641*suspend_clk period duration elapses.</p> <p>During the usb reset/resume completion the number jumps when the uSOF changes from 7->0. This is because when the core starts synchronizing after these events, until the uSOF number indicates a new frame, the internal counter may be running in different micro-frame. Once the counter locks to the host frame number, there will not be any jumps.</p> <p>Host Mode:</p> <p>The signal is incremented every 125 us. The clock reference is 30/60MHz utmi/ulpi clock. When mac_clk is off, the counter freezes. There will not be changes to this signal during this state. If the core is in suspended state for example mac_clk can stop.</p> <p>If the configuration has hibernation enabled (DWC_USB3_EN_PWROPT=2), then this signal changes after every 7500*suspend_clk period (or 3750*suspend_clk period if the UTMI is in word mode) duration elapses. If the GCTL.SOFITPSYNC or GFLADJ.GFLADJ_REFCLK_LPM_SEL is enabled, then this signal continues to generate SOF reference every 125us even if all the ports are suspended.</p> <p>Function: Current SOF Value Active State: Low Synchronous to: mac_clk Registered: Yes</p>
ltdb_link_state[4*DWC_USB3_NUM_U3_ROOT_PORTS-1:0]	O	<p>USB3.0 Link State</p> <p>Function: 4-bits per U3 port</p> <p>Active State: N/A</p> <p>Synchronous to: mac3_clk</p> <p>Registered: Yes</p>

Signal Name	I/O	Description
ltdb_sub_state[4*DWC_USB3_NUM_U3_ROOT_PORTS-1:0]	O	<p>USB3.0 Link Sub state</p> <p>Function: 4-bits per U3 port</p> <p>Native Master and Slave, all signals are sampled on rising edge of clock.</p> <p>Active State: N/A</p> <p>Synchronous to: mac3_clk</p> <p>Registered: Yes</p>

Table 6-5 describes the different link states.

Table 6-5 Link States

State	Name	Description
4'h0	U0	Normal operational state where packets can be transmitted and received
4'h1	U1	No packets are to be transmitted, PHY power state P1
4'h2	U2	No packets are to be transmitted, PHY power state P2
4'h3	U3	No packets are to be transmitted, PHY power state P3
4'h4	SS_DIS	SuperSpeed connectivity is disabled
4'h5	RX_DET	Warm reset, Receiver detection
4'h6	SS_INACT	Link has failed SuperSpeed operation
4'h7	POLL	SuperSpeed Training with Receiver equalization
4'h8	RECOV	Retrain SuperSpeed link, Perform Hot reset, Switch to Loop back mode
4'h9	HRESET	Hot reset using Training sets
4'ha	CMPLY	Test the transmitter for compliance to voltage and timing specifications
4'hb	LPBK	For test and fault isolation

The Link sub-states are LTSSM SUB-STATE. Table 6-6 describes the different link sub-states.

Table 6-6 Link Sub-states

State	Name	Description
U0		
No sub-states		
U1		
4'h0	U1_POWER	PHY Power state P0 -> P1 request
4'h1	U1_POWER_A	Wait for PHY power change done
4'h2	U1_ACTIVE	Wait for remote/local U1 exit
4'h3	U1_EXIT_LOC_RESP	Start U1 exit and wait for min response from remote partner

State	Name	Description
4'h4	U1_EXIT_LOC_FIN	Locally initiated U1 exit, Send min required U1 exit
4'h5	U1_EXIT_Rem	Remote initiated U1 exit, Send min required U1 exit
4'h6	U1_EXIT_P0	PHY Power state P1 -> P0 request
4'h7	U1_EXIT_P0_A	Wait for PHY power change P1 -> P0 done
4'h8	U1_EXIT_DONE	U1 exit successful. U1 -> Recovery
U2		
4'h0	U2_POWER	PHY Power state P0 -> P2 request
4'h1	U2_POWER2	Wait for PHY power change done
4'h2	U2_ACTIVE	Wait for remote/local U2 exit
4'h3	U2_ACTIVE_0	Request to start receiver detection
4'h4	U2_ACTIVE_1	Wait for Receiver detection response
4'h5	U2_EXIT_LOC_RESP	Start U2 exit and wait for min response from remote partner
4'h6	U2_EXIT_LOC_FIN	Locally initiated U2 exit, Send min required U2 exit
4'h7	U2_EXIT_Rem	Remote initiated U2 exit, Send min required U2 exit
4'h8	U2_EXIT_P0	PHY Power state P2 -> P0 request
4'h9	U2_EXIT_P0_A	Wait for PHY power change P1 -> P0 done
4'ha	U2_EXIT_DONE	U2 exit successful. U2 -> Recovery
U3		
4'h0	U3_POWER	PHY Power state P0 -> P2/P3 request
4'h1	U3_POWER3	Wait for PHY power change done
4'h2	U3_ACTIVE	Wait for remote/local U3 exit
4'h3	U3_ACTIVE_0	Request to start receiver detection
4'h4	U3_ACTIVE_1	Wait for Receiver detection response
4'h5	U3_EXIT_LOC_POW	PHY Power state P3 -> P0 -> P2 request
4'h6	U3_EXIT_LOC_POW_A	Wait for PHY power change done
4'h7	U3_EXIT_LOC_RESP	Start U3 exit and wait for min response from remote partner
4'h8	U3_EXIT_LOC_FIN	Locally initiated U3 exit, Send min required U3 exit
4'h9	U3_EXIT_Rem_POW	PHY Power state P3 -> P0 -> P2 request
4'ha	U3_EXIT_Rem_POW_A	Wait for PHY power change done

State	Name	Description
4'hb	U3_EXIT_Rem	Remote initiated U3 exit, Send min required U3 exit
4'hc	U3_EXIT_P0	PHY Power state P2/P3 -> P0 request
4'hd	U3_EXIT_P0_A	Wait for PHY power change P2/P3 -> P0 done
4'he	U3_EXIT	Received remote/local U3 exit
4'hf	U3_EXIT_DONE	U3 exit successful. U3 -> Recovery
SS_DIS		
4'h0	SSD_POWER3	PHY Power state P2/P3 request
4'h1	SSD_POWER3A	Wait for PHY power change done
4'h2	SSD_MAIN	Wait for RUN/STOP bit
RX_DET		
4'h0	RXD_INIT	PHY Power state P2 request
4'h1	RXD_POWER2	Wait for PHY power change done
4'h2	RXD_RESET	Warm reset
4'h3	RXD_ACTIVE0	Request to start receiver detection
4'h4	RXD_ACTIVE1	Wait for Receiver detection response
4'h5	RXD QUIET	Wait for 12ms timer timeout
SS_INACT		
4'h0	SSI_RESET	PHY Power state P0 -> P2 request
4'h1	SSI_POWER2	Wait for PHY power change done
4'h2	SSI QUIET0	Start 12-ms timer
4'h3	SSI QUIET1	Wait for 12-ms timer timeout
4'h4	SSI_DIS_DET0	Request to start receiver detection
4'h5	SSI_DIS_DET1	Wait for Receiver detection response
POLL		
4'h0	POLL_RESET	PHY Power state P0 request
4'h1	POLL_POWER0	Wait for PHY power change done
4'h2	POLL_LFPS	Send/Receive Poll.LFPS
4'h3	POLL_RXEQ	Send/Receive TSEQ Training set
4'h4	POLL_ACTIVE	Send/Receive TS1 Training set

State	Name	Description
4'h5	POLL_CONFIG	Send/Receive TS2 Training set
4'h6	POLL_IDLE	Send/Receive Logical IDLE Symbols
RECOV		
4'h0	RECOV_RESET	PHY Power state P0 request
4'h1	RECOV_POWER0	Wait for PHY power change done
4'h2	RECOV_ACTIVE	Send/Receive TS1 Training set
4'h3	RECOV_CONFIG	Send/Receive TS2 Training set
4'h4	RECOV_IDLE	Send/Receive Logical IDLE Symbols
HRESET		
4'h0	HRESET_RST	Request TCRRRL to send TS2
4'h1	HRESET_GO	Start 12-ms Timer
4'h2	HRESET_ACT1	Send/Receive TS1 Training set with reset bit
4'h3	HRESET_ACT2	Send/Receive TS1 Training set without reset bit
4'h4	HRESET_EXIT	Send/Receive Logical IDLE Symbols
LPBK		
4'h0	LPBK_IDLE	Wait for Loop back start request
4'h1	LPBK_MASTER	Master mode: Wait for Loopback Exit request
4'h2	LPBK_SLAVE	Slave mode: Wait for Loopback Exit request
4'h3	LPBK_EXIT_LOC_RESP	Start Loopback exit and wait for min response from remote partner
4'h4	LPBK_EXIT_LOC_FIN	Locally initiated Loopback exit, Send min required LFPS
4'h5	LPBK_EXIT_Rem	Remote initiated Loopback exit, Send min required LFPS

Signal Descriptions

This chapter details all possible I/O signals in the controller. For configurable IP titles, your actual configuration might not contain all of these signals.

Inputs are on the left of the signal diagrams; outputs are on the right.

Attention: For configurable IP titles, do not use this document to determine the exact I/O footprint of the controller. It is for reference purposes only.

When you configure the controller in coreConsultant, you must access the I/O signals for your actual configuration at workspace/report/IO.html or workspace/report/IO.xml after you have completed the report creation activity. That report comes from the exact same source as this chapter but removes all the I/O signals that are not in your actual configuration. This does not apply to non-configurable IP titles. In addition, all parameter expressions are evaluated to actual values. Therefore, the widths might change depending on your actual configuration.

Some expressions might refer to TCL functions or procedures (sometimes identified as <functionof>) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

In addition to describing the function of each signal, the signal descriptions in this chapter include the following information:

- **Active State:** Indicates whether the signal is active high or active low. When a signal is not intended to be used in a particular application, then this signal needs to be tied or driven to the inactive state (opposite of the active state).
- **Registered:** Indicates whether or not the signal is registered directly inside the IP boundary without intervening logic (excluding simple buffers). A value of No does not imply that the signal is not synchronous, only that there is some combinatorial logic between the signal's origin or destination register and the boundary of the controller. A value of N/A indicates that this information is not provided for this IP title.
- **Synchronous to:** Indicates which clock(s) in the IP sample this input (drive for an output) when considering all possible configurations. A particular configuration might not have all of the clocks listed. This clock might not be the same as the clock that your application logic should use to clock (sample/drive) this pin. For more details, consult the clock section in the databook.
- **Exists:** Name of configuration parameter that populates this signal in your configuration.

- **Validated by:** Assertion or de-assertion of signal(s) that validates the signal being described.

The I/O signals are grouped as follows:

- “System Clock, Reset, and Control Signals” on page 299
- “AHB Master Interface Signals” on page 311
- “AHB Slave Interface Signals” on page 315
- “Native Master Interface Signals” on page 319
- “Native Slave Interface Signals” on page 332
- “AXI Master Interface Signals” on page 335
- “AXI Slave Interface Signals” on page 346
- “RAM Interface Signals” on page 357
- “USB 3.0 PIPE3 PHY Interface Signals” on page 361
- “USB 2.0 UTMI+ Parallel Interface Signals” on page 370
- “USB 2.0 UTMI+ Vendor Control Interface Signals” on page 381
- “USB 2.0 UTMI+ OTG Interface Signals” on page 382
- “USB 2.0 ULPI PHY Interface Signals” on page 383
- “HSIC Interface Signals” on page 387
- “LPM Interface Signals” on page 388
- “Power Controller Interface Signals” on page 390
- “Host Interface Signals” on page 395
- “Hub Interface Signals” on page 403
- “ACA Interface Signals” on page 418
- “Miscellaneous Signal Interface Signals” on page 421
- “JTAG Interface Signals” on page 428
- “Synopsys Test Environment Interface Signals” on page 430
- “Type-C Support Signals for Synopsys PHY” on page 431

7.1 System Clock, Reset, and Control Signals

vcc_reset_n	-
bus_clk_early	-
bus_clken_gs	-
bus_clken_gm	-
bigendian_gs	-
ram_clk_in	-
suspend_clk	-
ref_clk	-
pmgt_ref_clk_ok	-
pmgt_ref_clk_req_n	-
fladj_30mhz_reg	-
dev_usb_outep_pkt_buff_avail	-
dev_usb_inep_pkt_buff_avail	-
vaux_reset_n	-

Note:

- PHY-side clocks are described with their signal interfaces.
- In USB 2.0-only mode, the suspend_clk signal is not present, if the hibernation and ADP features are disabled.

Table 7-1 System Clock, Reset, and Control Signals

Port Name	I/O	Description
ram_clk_gated	O	<p>RAM Clock Gated (for RAM1 and RAM2). This is the input clock for RAM1 and RAM2. If clock gating is enabled, this clock is gated when the USB is in low power mode and the controller (except the host-mode periodic scheduling logic) is idle.</p> <p>In the device mode, ram_clk_gated is identical to ram_clk_gated_ram0. In the host mode, if there are periodic EPs, sometimes ram_clk_gated is gated but ram_clk_gated_ram0 is active so that the controller can access the cache RAM to decide when to schedule the next periodic transaction.</p> <p>Exists: (DWC_USB3_MBUS_TYPE != 4) Synchronous To: DWC_USB3_MODE==3 ? "pipe3_rx_pclk[0],suspend_clk" : (DWC_USB3_EN_PWR0PT==0 ? "None" : "ram_clk_in") Registered: N/A Power Domain: Vcc Active State: N/A</p>

Port Name	I/O	Description
ram_clk_gated_ram0	O	<p>RAM Clock Gated (for RAM0). This is the input clock for RAM0. If clock gating is enabled, this clock is gated when the USB is in low power mode and the controller is idle.</p> <p>Exists: (DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_MODE==3 ? "pipe3_rx_pclk[0],suspend_clk" : (DWC_USB3_EN_PWROPT==0 ? "None" : "ram_clk_in")</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
ram_clk_out	O	<p>RAM Clock out. This is the output RAM clock. This clock should be connected to ram_clk_in.</p> <p>Exists: (DWC_USB3_MBUS_TYPE != 4 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: (DWC_USB3_MODE==1 DWC_USB3_MODE==2) & DWC_USB3_EN_PWROPT!=0 ? "None" : "bus_clk_early,pipe3_rx_pclk[0],suspend_clk,utmi_clk[0]"</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
pmgt_bus_clk_ref_off	O	<p>Indicates that the bus_clk input may be turned off. If this signal is '1', then external logic can safely gate the bus_clk input. The external logic must ungate the bus_clk before any slave CSR accesses.</p> <p>Exists: (DWC_USB3_BUS_CLK_REF_OFF==1)</p> <p>Synchronous To: bus_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pmgt_ref_clk_off	O	<p>Indicates that the ref_clk input may be turned off. If this signal is '1', then external logic can safely gate the ref_clk input.</p> <p>Exists: (DWC_USB3_REF_CLK_OFF==1)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
usb2phy_reset[(DWC_USB3_NUM_U2_R OOT_PORTS-1):0]	O	<p>USB 2.0 PHY Reset. USB 2.0 PHY Reset Used by USB 2.0</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 2 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
dev_usb_outep_pkt_received[15:0]	O	<p>Device Mode USB Logic OUT Endpoint Packet Received. This signal indicates that an OUT packet has been successfully received by the DWC_usb3 controller. This is an early indication that a DMA write for this packet will occur.</p> <p>For every received packet, there is only one pulse of this signal. In addition, there is at least one deassertion cycle after the pulse. Only one bit of this vector that corresponds to one endpoint of the DWC_usb3 controller, pulses at any given clock cycle.</p> <p>Note that the DWC_usb3 controller only pulses this signal when an OUT packet is successfully received. The pulse occurs before the DMA write is initiated.</p> <p>This signal is present only in all device and DRD configurations, and when EBC is enabled.</p> <p>For more details, refer to “External Buffer Control” on page 557.</p> <p>Default Value: 16'h0000</p> <p>Exists: (DWC_USB3_EXT_BUFF_CONTROL == 1)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT!=2 ? "bus_clk_early" : "bus_clk_early,suspend_clk"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dev_usb_inep_pkt_ack_received[15:0]	O	<p>Device Mode USB Logic IN Endpoint ACK Received. This signal indicates that the DWC_usb3 controller has received an ACK from the host, and the application can consider that the packet is transferred. The ACK pointer can then be advanced to the next packet.</p> <p>For every received packet, there is only one pulse of this signal. In addition, there is at least one deassertion cycle after the pulse. Only one bit of this vector that corresponds to one Endpoint of the DWC_usb3 controller, pulses at any given clock cycle.</p> <p>This signal is present only in all device and DRD configurations, and when EBC is enabled.</p> <p>For more details, refer to “External Buffer Control” on page 557.</p> <p>Default Value: 16'h0000</p> <p>Exists: (DWC_USB3_EXT_BUFF_CONTROL == 1)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT!=2 ? "bus_clk_early" : "bus_clk_early,suspend_clk"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
dev_usb_inep_pkt_retry_received[15:0]	O	<p>Device Mode USB Logic IN Endpoint Retry Received. This signal indicates that the DWC_usb3 controller has received a retry from the host, completed currently active DMA reads, and flushed its internal TxFIFO to re-synchronize to the retry packet. The application can use this pulse to rewind the read pointer back to the ACK pointer position. For every retry, there will be only one pulse of this signal. In addition, there will be at least one deassertion cycle after the pulse. Only one bit of this vector that corresponds to one Endpoint of the DWC_usb3 controller, will pulse at any given clock cycle. During the retry pulse, bus activity for the corresponding EP will not be active. If it is active, the DWC_usb3 controller continues the current DMA until completion, and then signal the retry received. This signal is present only in all device and DRD configurations, and when EBC is enabled. For more details, refer to “External Buffer Control” on page 557.</p> <p>Default Value: 16'0000</p> <p>Exists: (DWC_USB3_EXT_BUFF_CONTROL == 1)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT!=2 ? "bus_clk_early" : "bus_clk_early,suspend_clk"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
vcc_reset_n	I	<p>Active Low Core Reset (AHB/AXI/Native). Active low reset for the DWC_usb3 controller which:</p> <ul style="list-style-type: none"> ■ clears the poll_lfps_once flag in the LTSSM which is used to detect whether a SS polling failure leads the link to enter compliance mode or not; ■ applies reset to the 2.0 and 3.0 PHYs; ■ clears the logic analyzer MUX register. <p>It is asynchronously asserted and deasserted, and internally synchronized.</p> <p>In a two power-rail configuration (hibernation feature enabled), this is the reset for the Vcc power domain, similar to the PERST# in PCIe. The PMUs restore the sticky state when this signal is deasserted.</p> <p>vcc_reset_n signal has two behaviors depending whether Hibernation is enabled or not.</p> <ul style="list-style-type: none"> ■ Hibernation Off: vcc_reset_n resets DWC_usb3 controller and PHYs and Vaux_reset_n is not used. ■ Hibernation On: vcc_reset_n resets DWC_usb3 controller and vaux_reset_n (along with vcc_reset_n) resets DWC_usb3 controller and PHYs. <p>For more information vaux_reset_n, refer to “Hibernation” on page 481.</p> <p><i>When Hibernation is enabled and vaux_reset_n is de-asserted:</i></p> <p>This signal is asserted asynchronously by the power controller when the DWC_usb3 controller enters D3. It is de-asserted asynchronously by the power controller when the DWC_usb3 controller exits D3, either due to a PME or due to software-initiated wakeup, and Vcc is valid. Sticky bits are retained by the PMU and the PHYs are not reset. The power controller must assert this reset at least one suspend_clk cycle after the DWC_usb3 controller enters D3 to allow the PMUs time to start controlling the PHYs.</p> <p><i>When Hibernation is not enabled, or when Hibernation is enabled and vaux_reset_n is asserted:</i></p> <p>This signal is driven to the same value as vaux_reset_n to perform a cold reset, which resets the entire controller including the sticky bits and the PHYs. It may be asserted and de-asserted asynchronously.</p> <p>Exists: Always</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>

Port Name	I/O	Description
bus_clk_early	I	<p>SoC Bus Clock (AHB/AXI/Native). Clock from application bus. In host mode, depending on the host bandwidth required for your configuration, you can use a bus clock frequency lower than the ram_clk requirements by programming GCTL[7:6]. For more details, refer to “Minimum Clock Frequencies: bus_clk, ram_clk” on page 515. The minimum bus_clk frequency that has been tested is 60MHz.</p> <p>In device mode, when UTMI is running at 16-bit mode, the minimum frequency is 60 MHz and when UTMI/ULPI is running in 8-bit mode, then the minimum frequency can be up to 1 MHz. The maximum frequency depends on the technology.</p> <p>In device mode, ram_clk must be connected to pipe_clk when running the bus_clk lower than 125 MHz in SS mode or to ULPI/UTMI clock when running bus_clk lower than 60 MHz in USB 2.0 mode.</p> <p>The nominal frequency must meet the USB 3.0 maximum data rate of 4Gbs IN and OUT direction for concurrent IN and OUT operation. For example, in device mode this needs to be a 125 MHz@32-bit AXI-bus when using 2-port RAM and 250 MHz when using single port RAM.</p> <p>In the case of a 4-port host with four SuperSpeed bus instances, this needs to be a 125 MHz@128-bit AXI-bus when using 2-port RAM and 250 MHz when using single port RAM. This nominal frequency recommendation is not taking into consideration other bus masters in your SoC.</p> <p>This signal is used by USB 2.0/3.0.</p> <p>Note: For native master and slave, all signals are sampled on the rising edge of clock.</p> <p><i>When Hibernation is enabled:</i></p> <p>This signal is also the primary clock used by the non-sticky modules after reset. The controller applies its own reset to the non-sticky modules asynchronously with vcc_reset_n and deasserts it synchronously with the first rising edge of bus_clk.</p> <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
bus_clken_gs	I	<p>Slave Interface Clock Enable (AHB/AXI/Native). Allows slower operation on the bus. Typically useful in CPU applications, where the CPU, for example, can run at 400 MHz and the bus can run at 200 MHz.</p> <p>This pin is reserved for future use. Tie to 1'b1.</p> <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>

Port Name	I/O	Description
bus_clken_gm	I	<p>Master Interface Clock Enable (AHB/AXI/Native). Allows slower operation on the bus. This signal is typically useful in CPU applications, where the CPU, for example, can run at 400 MHz and the bus can run at 200 MHz.</p> <p>This pin is reserved for future use. Tie to 1'b1.</p> <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
big endian_gs	I	<p>Slave Big Endian Select (AHB/AXI/Native). Selects Big Endian mode for the SoC bus Slave.</p> <ul style="list-style-type: none"> ■ 1'b0: Little Endian ■ 1'b1: Big Endian <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: DWC_USB3_EN_PWROPT==2 ? "Yes" : ((DWC_USB3_SBUS_TYPE==3 DWC_USB3_SBUS_TYPE==1) ? "No" : "Yes")</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
ram_clk_in	I	<p>RAM Clock IN. This is the input RAM clock. This clock should be connected to ram_clk_out.</p> <p>Exists: (DWC_USB3_MBUS_TYPE != 4 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>

Port Name	I/O	Description
suspend_clk	I	<p>Suspend Clock</p> <p>The USB3 suspend_clk input replaces pipe3_rx_pclk as the clock source to a small part of the USB3 controller that operates when the SS PHY is in its lowest power state (P3) and therefore not providing a clock. The suspend_clk drives the mac3_clk, pipe3_mx_rx_pclk, pipe3_mx_tx_pclk during suspend. The maximum frequency of the internally divided suspend clock is 125 MHz. The minimum frequency is 32 KHz. For suspend clock accuracy requirement, refer to bits [31:19] in GCTL register.</p> <p>Note that this signal is not present in USB 2.0-only mode, if the hibernation and ADP features are disabled.</p> <p><i>When Hibernation is enabled:</i></p> <p>This is a minimum 32 kHz, maximum 200 kHz clock that is provided to the PMU and the controller. In the PMU, it is used to filter out glitches from the PHY, time PIPE3 receiver detection, and to set PORTSC bits while in hibernation. In the controller, it is used to allow the sticky state to be restored before the bus_clk, USB 2.0 PHY clock, and USB 3.0 PHY clock are available.</p> <p>Note: When Hibernation is enabled,</p> <ul style="list-style-type: none"> ■ Suspend clock must be always running (even in USB 2.0 mode). ■ When PMU filters are enabled (DWC_USB3_EN_PMU_FILTER=1), the maximum suspend_clk frequency is only 200kHz because the suspend_clk is used to filter glitches from the PHY interfaces. A pulse is considered to be a glitch if it is less than 2.5us long (derived from USB 2.0). Therefore, sampling a signal on two consecutive rising edges of a 200kHz clock guarantees that the signal is stable for more than 2.5us. ■ When PMU filters are disabled (DWC_USB3_EN_PMU_FILTER=0), the suspend_clk frequency may be faster (up to 30MHz). <p>Exists: ((DWC_USB3_EN_USB2_ONLY == 0) (DWC_USB3_EN_PWROPT == 2) (DWC_USB3_EN_ADP==1))</p> <p>Synchronous To: None</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

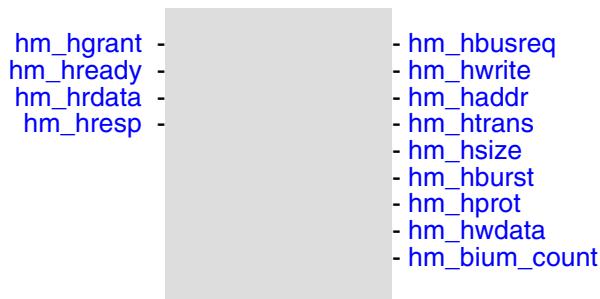
Port Name	I/O	Description
ref_clk	I	<p>Reference Clock.</p> <p>The ref_clk input generates the ITP counter when the UTMI/ULPI PHY are in the suspended state. This clock exists as an input in the Host and DRD configuration. The maximum frequency of this clock is 125 MHz and the minimum frequency is 16.129 MHz. This clock must meet the accuracy requirement of 50 ppm (100 ppm also meets the requirement).</p> <p>Note:</p> <ul style="list-style-type: none"> ■ For more details on how to program the controller with ref_clk frequency, refer to REFCLKPER in GUCTL section of the Programming Guide. ■ For more details on how to enable the ITP generation based the ref_clk counter, refer to SOFITPSYNC in GCTL section of the Programming Guide. ■ If GFLADJ.GFLADJ_REFCLK_LPM_SEL is selected, the SOF and ITP counters are generated off of ref_clk. In this mode of operation, the ref_clk frequencies supported are 16/17/19.2/20/24/39.7/40MHz frequencies. For more information on enabling this mode, see GFLADJ section of the Programming Guide. ■ If you never use the GCTL.SOFITPSYNC or the GFLADJ.GFLADJ_REFCLK_LPM_SEL feature, the minimum frequency for the ref_clk can be as low as 32KHz. <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2 (DWC_USB3_MODE == 0 && DWC_USB3_LPM_SUSP_OFF == 1) DWC_USB3_NUM_SSIC_PORTS != 0)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
pmgt_ref_clk_ok	I	<p>This signal indicates that the ref_clk input is stable. If this signal is '1', then the external logic guarantees the ref_clk input to be accurate.</p> <p>Exists: (DWC_USB3_REF_CLK_OFF==1)</p> <p>Synchronous To: ref_clk</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pmgt_ref_clk_req_n[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>This signal indicates that the ref_clk input to the PIPE PHY may be turned off. If this signal is '1', then PIPE PHY does not need the ref_clk input.</p> <p>Note: If you are using the Synopsys SS PHY, connect the pipe_ref_clk_req_n output signal to this signal.</p> <p>Exists: (DWC_USB3_REF_CLK_OFF==1)</p> <p>Synchronous To: ref_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
fladj_30mhz_reg[5:0]	I	<p>HS Jitter Adjustment. Indicates the correction required to accommodate mac3 clock and utmi clock jitter to measure 125's duration. With fladj_30mhz_reg tied to zero, the high speed 125us micro-frame is counted for 123933ns. You must program the value in terms of high speed bit times in a 30 MHz cycle. The default value that must be driven is 32 (assuming 30 MHz perfect clock).</p> <p>fladj_30mhz_reg connects to the FLADJ register defined in the xHCI spec in the PCI configuration space. Each count is equal to 16 high speed bit times. By default, when this register is set to 32, it gives a 125us interval.</p> <p>Now, based the clock accuracy you can decrement the count or increment the count to get the 125 us uSOF window. For non-PCI systems, it is recommended that this strap is connected to a register that can be controlled by software. This strap is used in device mode also. If device only mode is implemented, it is recommended to tie this input to 'd32.</p> <p>You can override this value using the register GFLADJ in case this input signal is tied to a hardcoded value and must be changed post-silicon. For details how to program this register, refer to the GFLADJ section of the Programming Guide.</p> <p>Default Value: Strap Value</p> <p>Exists: DWC_USB3_MODE != 3</p> <p>Synchronous To: DWC_USB3_EN_PWROPT!=2 ? "bus_clk_early" : "bus_clk_early,suspend_clk"</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: No</p>

Port Name	I/O	Description
dev_usb_outep_pkt_buff_avail[15:0]	I	<p>Device Mode USB Logic OUT Endpoint Buffer Available. This signal indicates that space for at least two maximum-sized packets is available in the application RxFIFO for DMA write. Because the controller can receive two back-to-back packets during a burst, two packet slots prevent buffer overflow.</p> <p>This signal is asserted when the application reads from its RxFIFO (AXI/AHB data width), and space for a maximum-size packet is released. This signal is deasserted in response to the dev_usb_outep_pkt_received signal and when the available space is less than two maximum-sized packets.</p> <p>This signal is present only in all device and DRD configurations. To disable the External Buffer Control (EBC) feature, this input must be tied to 1.</p> <p>For more details, refer to “External Buffer Control” on page 557.</p> <p>Note: This signal must be registered with respect to bus_clk_early. It cannot be combinational because it is re-synchronized in another clock domain, and does cause errors in design.</p> <p>Default Value: 16'hffff</p> <p>Exists: (DWC_USB3_EXT_BUFF_CONTROL == 1)</p> <p>Synchronous To: DWC_USB3_HSPHY_INTERFACE==1 ? (DWC_USB3_NUM_U3_ROOT_PORTS!=0 ? "0:8=bus_clk_early,pipe3_rx_pclk[0],ram_clk_in,suspend_clk,utmi_clk[0];9:15=bus_clk_early" : "0:8=bus_clk_early,ram_clk_in,suspend_clk,utmi_clk[0];9:15=bus_clk_early") : (DWC_USB3_HSPHY_INTERFACE==2 ? (DWC_USB3_NUM_U3_ROOT_PORTS!=0 ? "0:8=bus_clk_early,pipe3_rx_pclk[0],ram_clk_in,suspend_clk,ulpi_clk[0];9:15=bus_clk_early" : "0:8=bus_clk_early,ram_clk_in,suspend_clk,ulpi_clk[0];9:15=bus_clk_early") : (DWC_USB3_NUM_U3_ROOT_PORTS!=0 ? "0:8=bus_clk_early,pipe3_rx_pclk[0],ram_clk_in,suspend_clk,utmi_clk[0],ulpi_clk[0];9:15=bus_clk_early" : "0:8=bus_clk_early,ram_clk_in,suspend_clk,utmi_clk[0],ulpi_clk[0];9:15=bus_clk_early"))</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
dev_usb_inep_pkt_buff_avail[15:0]	I	<p>Device Mode USB Logic IN Endpoint Buffer Available. This signal indicates that at least one packet is available for DMA read for IN EP transmit. The DWC_usb3 controller initiates read DMAs only if this signal is asserted. This signal is asserted when the application accumulates at least one packet of data in the external buffer. This signal is deasserted when the data available in the external buffer is below the one packet of data threshold.</p> <p>This signal is present only in all device and DRD configurations, and when EBC is enabled. For more details, refer to “External Buffer Control” on page 557.</p> <p>Note: This signal must be registered with respect to bus_clk_early. It cannot be combinational because it is re-synchronized in another clock domain, and causes errors in design.</p> <p>Default Value: 16'hffff</p> <p>Exists: (DWC_USB3_EXT_BUFF_CONTROL == 1)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT!=2 ? "bus_clk_early,ram_clk_in" : "bus_clk_early,ram_clk_in,suspend_clk"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vaux_reset_n	I	<p>Active low reset for the DWC_usb3_u2pmu and DWC_usb3_u3pmu modules which are powered by Vaux. This signal is asynchronously asserted and deasserted, and internally synchronized. It must be asserted at the same time as vcc_reset_n to apply a cold reset to the controller which resets the controller, sticky bits, and PHYs.</p> <p>Note: This signal is valid only when Hibernation is enabled.</p> <p>Exists: (DWC_USB3_EN_PWROPT==2)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: Vaux</p> <p>Active State: Low</p>

7.2 AHB Master Interface Signals



The AHB Master interface connects the DWC_usb3 controller as a master on the AHB.

The AHB Master interface is in the bus_clk_early clock domain. This interface does not generate locked transfers or wrapped-burst transfers.

Note: For more information on this interface, see *AMBA Specification 2.0*.

Table 7-2 AHB Master Interface Signals

Port Name	I/O	Description
hm_hbusreq	O	<p>AHB Master Bus Request. Asserted by the controller to request AHB grant for memory access.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_hwrite	O	<p>AHB Master Transfer Direction. Indicates whether the current transaction is a read or write request.</p> <ul style="list-style-type: none"> ■ 1'b0: Read ■ 1'b1: Write <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_haddr[(DWC_USB3_AWIDTH-1):0]	O	<p>AHB Master Address Bus. Provides the address that the controller reads or writes.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hm_htrans[1:0]	O	<p>AHB Master Transfer Type. Indicates the data transfer type.</p> <ul style="list-style-type: none"> ■ 2'b00: IDLE ■ 2'b01: BUSY ■ 2'b10: NONSEQ ■ 2'b11: SEQ <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_hsize[2:0]	O	<p>AHB Master Transfer Size. Indicates the data transfer size.</p> <ul style="list-style-type: none"> ■ 3'b000: 8 bits ■ 3'b001: 16 bits ■ 3'b010: 32 bits ■ 3'b011: 64 bits ■ 3'b100: 128 bits No other values are supported. The controller uses same hsize as the data bus width to optimize the read. It discards unnecessary bytes to improve the bus efficiency. <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_hburst[2:0]	O	<p>AHB Master Burst Type. Indicates the bursting mode used by the AHB Master during a burst transaction. This value is set in the Global SoC Bus Configuration register 0 (GSBUSCFG0).</p> <ul style="list-style-type: none"> ■ 3'b000: SINGLE ■ 3'b001: INCR ■ 3'b011: INCR4 ■ 3'b101: INCR8 ■ 3'b111: INCR16 ■ No other values are supported <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hm_hprot[3:0]	O	<p>AHB Master Protection Control. For cache type bit assignments, see "Cache Type Bit Assignments" table in the Programming Guide.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_hwdata[(DWC_USB3_MDWIDTH-1):0]	O	<p>AHB Master Write Data Bus. Data bus for controller writes.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_bium_count[12:0]	O	<p>AHB Master Request Count. Master request count in bytes. This signal is the same as gmw/gmr_mcount the native interface. This signal is not a standard AHB signal. You can use this sideband signal in an AHB-to-SoC bus gasket design.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_hgrant	I	<p>AHB Bus Grant. Asserted in response to hm_hbusreq, indicating that the DWC_usb3 controller has taken ownership of the AHB.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_hready	I	<p>AHB Master Transfer Done. Asserted by the AHB Slave (system memory) to indicate completion of a data transfer.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hm_hrdata[(DWC_USB3_MDWIDTH-1):0]	I	<p>AHB Master Read Data Bus. Application read data bus.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hm_hresp[1:0]	I	<p>AHB Master Transfer Response. Indicates the response of the AHB Slave to the data transfer. This signal is qualified by hm_hready.</p> <ul style="list-style-type: none">■ 2'b00: OKAY■ 2'b01: ERROR■ 2'b10: RETRY■ 2'b11: SPLIT <p>Exists: (DWC_USB3_MBUS_TYPE == 0 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.3 AHB Slave Interface Signals

hs_haddr	-	hs_hready_resp
hs_hsel	-	hs_hresp
hs_hwrite	-	hs_hrdtata
hs_htrans	-	
hs_hsizze	-	
hs_hburst	-	
hs_hready	-	
hs_hwdata	-	

This interface connects the DWC_usb3 controller as a slave on the AHB and is used by the system CPU (AHB Master) to read from and write to the CSRs.

The AHB Slave interface is in the bus_clk_early clock domain.

Features supported on this interface include:

- Single and burst transfers
- 32-bit transfers (Non-32-bit writes corrupt registers; reads return invalid data.)
- OKAY response

Note: For more information on this interface, see *AMBA Specification 2.0*.

Table 7-3 AHB Slave Interface Signals

Port Name	I/O	Description
hs_hready_resp	O	<p>AHB Slave Transfer Done: Out. Asserted by the DWC_usb3 controller completion of a data transfer.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hs_hresp[1:0]	O	<p>AHB Slave Transfer Response. Indicates transfer response.</p> <ul style="list-style-type: none"> ■ 2'b00: OKAY <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hs_hrdata[(DWC_USB3_SDWIDTH-1):0]	O	<p>AHB Slave Read Data Bus. Application read data bus.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWR0PT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hs_haddr[(DWC_USB3_AWIDTH-1):0]	I	<p>AHB Slave Address Bus. Provides the address that the controller reads or writes. Bits [19:0] are used by the controller.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWR0PT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hs_hsel	I	<p>AHB Slave Slave Select. Asserted when the transaction address falls within the memory address range allocated in the DWC_usb3 controller.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWR0PT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hs_hwrite	I	<p>AHB Slave Transfer Direction. Indicates whether the current transaction is a read or write.</p> <ul style="list-style-type: none"> ■ 1'b0: Read ■ 1'b1: Write <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWR0PT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hs_htrans[1:0]	I	<p>AHB Slave Transfer Type. Indicates AHB transfer type.</p> <ul style="list-style-type: none"> ■ 2'b00: IDLE ■ 2'b01: BUSY ■ 2'b10: NONSEQ ■ 2'b11: SEQ <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hs_hsize[2:0]	I	<p>AHB Slave Transfer Size. Indicates AHB transfer size.</p> <ul style="list-style-type: none"> ■ 3'b010: 32 bits ■ 3'b011: 64 bits <p>No other values are supported by DWC_usb3. For device mode, it is recommended that the software uses 32-bit access.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hs_hburst[2:0]	I	<p>AHB Slave Burst Type (not used, but all burst sizes are supported by breaking them into single accesses).</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hs_hready	I	<p>AHB Slave Transfer Done - In. The HREADY signal from the currently selected AHB slave. It is asserted to indicate a completed data transfer.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hs_hwdata[(DWC_USB3_SDWIDTH-1):0]	I	<p>AHB Slave Write Data Bus. Application write data bus.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 0 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.4 Native Master Interface Signals

gmw_mcmandaccept	- gm_enburst
gmw_mdatarequest	- gm_storeandforward
gmw_mdatarequestid	- gm_onerequestonly
gmw_mdatalast	- gmw_mbig endian
gmw_sabortdone	- gmw_mslnum
gmw_sid	- gmw_mid
gmw_srespvalid	- gmw_maddr
gmw_slast	- gmw_mcmand
gmw_sresp	- gmw_mburst
gmw_srespinfo	- gmw_mcount
gmr_mcmandaccept	- gmw_mpriority
gmr_sdatalast	- gmw_mcache_reqinfo
gmr_sdata	- gmw_mprot_addrspace
gmr_sabortdone	- gmw_mshift
gmr_sid	- gmw_mlock
gmr_srespvalid	- gmw_mabort
gmr_sresp	- gmw_mabortid
gmr_srespinfo	- gmw_mdata
	- gmw_mdataaccept
	- gmw_srespaccept
	- gmr_mbig endian
	- gmr_mslnum
	- gmr_mid
	- gmr_maddr
	- gmr_mcmand
	- gmr_mburst
	- gmr_mcount
	- gmr_mpriority
	- gmr_mcache_reqinfo
	- gmr_mprot_addrspace
	- gmr_mshift
	- gmr_mlock
	- gmr_mabort
	- gmr_mabortid
	- gmr_sdataaccept

This interface connects the DWC_usb3 controller as a master on the SoC bus.

The Native Master interface is in the bus_clk_early clock domain. If a read access is dependent on a write access completing first, the controller does not issue the DMA read until it receives the DMA write response.

Table 7-4 Native Master Interface Signals

Port Name	I/O	Description
gm_enburst[7:0]	O	<p>Native Master Burst Size Enable. One enable bit for each burst size. This is a static signal, and it is not recommended that it change dynamically. This signal reflects the value of the register bits GSBUSCFG0[7:0].</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gm_storeandforward	O	<p>Native Master Store-and-Forward Mode. This signal must not be used.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
gm_onerequestonly	O	<p>One request at a time. Queue only one transfer at a time on the SoC bus. This is a static signal, and it is not recommended that it change dynamically.</p> <p>This signal is not used; it reflects the value of the register bit GSBUSCFG0[14].</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mbig endian	O	<p>Write Request Big Endian. Indicates that the data for this DMA write request is big endian. Depending whether this DMA write is for a descriptor or data, this port reflects the corresponding endian mode selection bit in GSBUSCFG[12] or [11], respectively.</p> <p>Because the native interface is inherently little endian (that is, the byte for address 0 is always bits [7:0] regardless of endian selection), a gasket may need to perform an endian transform on the data. For AHB, the gasket does the required byte order reversal for big endian transfers. For AXI, no endian transform is required. For native interface applications, the appropriate endian transform, if any, depends on the requirement of the particular system bus. For more details on endianness, refer to "Little-Endian and Big-Endian" section in the User Guide.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mslnum	O	<p>Write Request Slave Number. Not used. Used for out-of-order completion support. This indicates the destination slave number, and uses an unused write buffer. In a USB application, descriptor access could be mapped to one slave and data access could be mapped to another slave.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmw_mid[(DWC_USB3_IDWIDTH-1):0]	O	<p>Write Request ID. Not used. This signal has no direct relation to the AXI Bus ID signals.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
gmw_maddr[(DWC_USB3_AWIDTH-1):0]	O	<p>Write request address.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mcmd[2:0]	O	<p>Write request command.</p> <ul style="list-style-type: none"> ■ 000: IDLE ■ 001: WRITE ■ 010: Write-NonPosted ■ 011: WRConditional - not used ■ 100: Broadcast - not used ■ Others: N/A - not used <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: 0:1=bus_clk_early;2:2=None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mburst[2:0]	O	<p>Write request burst type.</p> <ul style="list-style-type: none"> ■ 000: INC-ALIGNED ■ 001: INCR - not used ■ 010: WRAP - not used ■ 011: STRM - not used ■ Others: N/A - not used <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmw_mcount[15:0]	O	<p>Write Request Length Count (in Bytes). The GM-IF transfer can be broken into multiple smaller bursts on the system bus.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: 0:10=ram_clk_in;11:15=None</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mpriority	O	<p>Write Request Priority Over Read. Not used.</p> <ul style="list-style-type: none"> ■ bit[0]: GM-IF write priority over read. When a single read/write channel AHB system bus is used, this is indicated to DWC_usb3_bus_gm to give priority to a write request over a read request. ■ bit[1]: Give priority to this write request over the previous write request queued in. Useful when a larger packet transfer is going on, to a get short descriptor write done in between the bursts for the large packet. <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mcache_reqinfo[(DWC_USB3_REQ1_NFOWIDTH-1):0]	O	<p>Write Request Cache/Info. Gets mapped to AXI's awcache signal. For cache type bit assignments, see "Cache Type Bit Assignments" table in the Programming Guide.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mprot_addrspace[(DWC_USB3_AS_PACEWIDTH-1):0]	O	<p>Write Request Address Space. Not used. This signal gets mapped to AXI's awprot signal.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mshift[(DWC_USB3_MDLOG-1):0]	O	<p>Write Request Data Shift Count. Not used. Indicates the number of bytes the data is shifted before being passed onto the system bus.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmw_mlock[1:0]	O	<p>Write Request Lock. Not used. Indicates a locked transfers.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mabort[2:0]	O	<p>Write Request Abort. Not used.</p> <p>An application can abort a previously sent request. If the SoC bus supports early termination, an ongoing cycle is properly terminated as soon as possible; otherwise, the transfer is completed properly. The signaling requires one cycle.</p> <p>The abort done is indicated through the response interface. When an abort is asserted, the gasket does not ask for more data from the Generic Master (GM) interface. It sends dummy data (0) for the rest of a burst in AHB and keeps the strobe/byte-enables inactive in AXI. Any uncommitted request from the bus is terminated within the gasket.</p> <p>Only one abort request can be issued at a time.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mabortionid[(DWC_USB3_IDWIDTH-1):0]	O	<p>Write request abort ID</p> <p>Note: This signal is driven to zero.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
gmw_mdata[(DWC_USB3_MDWIDTH-1):0]	O	<p>Write Data. Data is taken on the clock when both gmw_mdatarequest and gmw_mdataaccept are valid.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mdataaccept	O	<p>Write data accept. Application indicates when valid data is present.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmw_srespaccept	O	<p>Write Response Accept. When the application is not ready to accept a response, it drives this low. A request is accepted only when both gmw_srespvalid and gmw_srespaccept are high.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mbig endian	O	<p>Read Request Big Endian. Indicates that the data for this DMA Read Request is big endian. Depending whether this DMA read is for a descriptor or data, this port reflects the corresponding endian mode selection bit in GSBUSCFG[12] or [11], respectively. Because the native interface is inherently little endian (that is, the byte for address 0 is always bits [7:0] regardless of endian selection), a gasket may need to perform an endian transform on the data.</p> <p>For AHB, the gasket does the required byte order reversal for big endian transfers. For AXI, no endian transform is required. For native interface applications, the appropriate endian transform, if any, depends the requirement of the particular system bus.</p> <p>For more details on endianness, refer to "Little-Endian and Big-Endian" section in the User Guide.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mslnum	O	<p>Read Request Slave Number. Not used.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mid[(DWC_USB3_IDWIDTH-1):0]	O	<p>Read Request ID. Not used. This signal has no direct relation to the AXI Bus ID signals.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmr_maddr[(DWC_USB3_AWIDTH-1):0]	O	<p>Read Request Address.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mcmd[2:0]	O	<p>Read Request Command.</p> <ul style="list-style-type: none"> ■ 000: IDLE ■ 001: Read ■ 010: Read Exclusive: not used ■ 011: Read Linked: not used ■ Others: N/A: not used <p>Note: Only read is supported now.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: 0:0=bus_clk_early;1:2=None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mburst[2:0]	O	<p>Read Request Burst Type.</p> <ul style="list-style-type: none"> ■ 000: INC-ALIGNED ■ 001: INCR: not used ■ 010: WRAP: not used ■ 011: STRM: not used ■ Others: N/A: not used <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mcount[15:0]	O	<p>Read Request Length Count (in Bytes). The GM-IF transfer can be broken into multiple smaller bursts on the system bus.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: 0:13=ram_clk_in;14:15=None</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmr_mpriority	O	<p>Read Request Priority Over Write. Not used.</p> <ul style="list-style-type: none"> ■ bit[0]: GM-IF read priority over write. When a single read/write channel AHB system bus is used, this is indicated to DWC_usb3_bus_gm to give priority to read request instead of write. ■ bit[1]: Gives priority to this request over the previous read request queued in. Useful to get a descriptor fetch done in between bursts of a previous larger data packet read. <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mcache_reqinfo[(DWC_USB3_REQI_NFOWIDTH-1):0]	O	<p>Read Request Cache Information When reqinfo[3:0] is:</p> <ul style="list-style-type: none"> ■ 4'b0000: Transfer Type is Single/final segment TX Data. ■ 4'b0001: Transfer Type is Multi-segment TX Data. ■ 4'b0010: Transfer Type is Desc Fetch. ■ 4'b0100: Transfer Type is Event Writeback. <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
gmr_mprot_addrspace[(DWC_USB3_ASPECTWIDTH-1):0]	O	<p>Read Request Address Space</p> <p>Note: This signal is driven to 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
gmr_mshift[(DWC_USB3_MDLOG-1):0]	O	<p>Read Request Data Shift. Not used. Indicated the number of bytes the data is shifted before being passed onto the GM-IF Bus.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmr_mlock[1:0]	O	<p>Read Request Lock. Indicates locked transfers. Asserted for descriptor fetches.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: 0:0=ram_clk_in;1:1=None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mabort[2:0]	O	<p>Read Request Abort. Not used. An application can abort a previously sent request in DWC_usb3_bus_gm. If the system bus supports early termination, an ongoing cycle is properly terminated as soon as possible. Otherwise, the transfer is completed properly through cycle signaling. The abort done is indicated through the response interface. When an abort is asserted, the gasket does not send any more data to GM for that request. Any uncommitted request on the bus is terminated within the gasket. Only one abort request can be issued at a time.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_mabortid[(DWC_USB3_IDWIDTH-1):0]	O	<p>Read Request Abort ID</p> <p>Note: This signal is tied to 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
gmr_sdataaccept	O	<p>Read Data Accept.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mc mdaccept	I	<p>Write request command accept. Accepts current request by driving gmw_mc mdaccept high.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

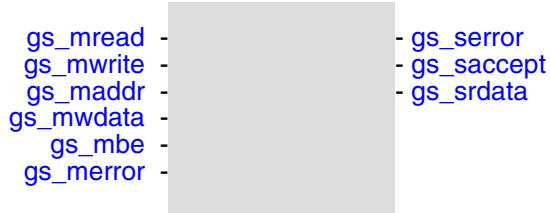
Port Name	I/O	Description
gmw_mdatarequest	I	<p>Write Data Request. Can get asserted combinationally with the same clock as gmw_mcmd.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_mdatarequestid[(DWC_USB3_IDWI_DTH-1):0]	I	<p>Write Data Request ID. Not Used.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
gmw_mdatalast	I	<p>Write Data Last. Indicates the last data.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_saborteddone	I	<p>Write Abort Done. Not used.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_sid[(DWC_USB3_IDWIDTH-1):0]	I	<p>Write Response ID. Not used. The identification tag of responses. This matches one of the earlier request IDs. This ID is not related to AXI IDs.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_srespvalid	I	<p>Write Response Valid.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmw_slast	I	<p>Write Response Last.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_sresp[2:0]	I	<p>Write Response. Encoding is as follows:</p> <ul style="list-style-type: none"> ■ h'0: OK; read transaction finished successfully ■ h'1: SLVERR; AHB/AXI addressed slave generated error ■ h'2: DECERR; AXI decode error from interconnect ■ h'3: Response Error ■ h'4: Abort Done - Not used ■ Others: N/A <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmw_srespinfo[(DWC_USB3_REQINFOW-1):0]	I	<p>Write Response Information.</p> <p>Note: This signal is not used.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
gmr_mcmandaccept	I	<p>Read Request Command Accept. Accepts current request by driving gmr_mcmandaccept high.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_sdatalast	I	<p>Read Data Last.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmr_sdata[(DWC_USB3_MDWIDTH-1):0]	I	<p>Native Master Read Data.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_saborteddone	I	<p>Read Abort Done. Not used.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_sid[(DWC_USB3_IDWIDTH-1):0]	I	<p>GM-IF response ID. Not used. The identification tag of responses. This matches one of the earlier request IDs. This ID is not related to AXI IDs.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_srespvalid	I	<p>Read Response Valid.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gmr_sresp[2:0]	I	<p>Read Response. Encoding is as follows:</p> <ul style="list-style-type: none"> ■ h'0: OK: read transaction finished successfully ■ h'1: SLVERR: AHB/AXI addressed slave generated error ■ h'2: DECERR: AXI decode error from interconnect ■ h'3: Response Error ■ h'4: Abort Done - not used ■ Others: N/A <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gmr_srespinfo[(DWC_USB3_REQINFO1_DTH-1):0]	I	<p>Read Response Information.</p> <p>Note: This signal is not used.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 3 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>

7.5 Native Slave Interface Signals



This interface connects the DWC_usb3 controller as a slave on the SoC bus.

The Native Slave interface is in the bus_clk_early clock domain.

Table 7-5 Native Slave Interface Signals

Port Name	I/O	Description
gs_serror	O	<p>Slave Error. Either gs_error or gm_saccept indicates an accept.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE == 3 ? "None" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gs_saccept	O	<p>Slave Accept.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gs_srdata[(DWC_USB3_SDWIDTH-1):0]	O	<p>Native Slave Read Data.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE == 3) ? ("0:9=bus_clk_early,ram_clk_in;10:11=bus_clk_early,ram_clk_in,utmi_ck[0];12:31=bus_clk_early,ram_clk_in") : "bus_clk_early,ram_clk_in"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gs_mread	I	<p>Read Request.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gs_mwrite	I	<p>Write Request.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE == 3) ? "bus_clk_early,ram_clk_in" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gs_maddr[(DWC_USB3_AWIDTH-1):0]	I	<p>Address. This is valid for every beat.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early,ram_clk_in"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gs_mwdata[(DWC_USB3_SDWIDTH-1):0]	I	<p>Write Data. Data is taken at the clock when both gs_mwrite and gs_gaccept are valid.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early,ram_clk_in"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
gs_mbe[(DWC_USB3_SDLOG-1):0]	I	<p>Byte Enables. This is valid for every beat.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early,ram_clk_in</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
gs_merror	I	<p>Write Request/Data Error. Indicates error data. The slave must still return either gs_serror or gm_saccept.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 3 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE == 3 ? "None" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.6 AXI Master Interface Signals

xm_awready	- xm_awid
xm_wready	- xm_awaddr
xm_bid	- xm_awlen
xm_bvalid	- xm_awsize
xm_bresp	- xm_awburst
xm_bmisc_info	- xm_awlock
xm_arready	- xm_awcache
xm_rid	- xm_awprot
xm_rvalid	- xm_awvalid
xm_rlast	- xm_awmisc_info
xm_rdata	- xm_wid
xm_rmisc_info	- xm_wvalid
xm_rresp	- xm_wlast
xm_csysreq	- xm_wdata
	- xm_wstrb
	- xm_bready
	- xm_arid
	- xm_arvalid
	- xm_araddr
	- xm_arlen
	- xm_arsize
	- xm_arburst
	- xm_arlock
	- xm_arcache
	- xm_arprot
	- xm_armisc_info
	- xm_rready
	- xm_csysack
	- xm_cactive

This interface connects the DWC_usb3 controller as a master on the AXI.

The AXI Master interface is in the bus_clk_early clock domain.

Features supported on this interface include:

- Support for burst-based transactions with only start address issued
- Support for separate read and write channels to enable low-cost Direct Memory Access (DMA)
- Support for separate address (control), and data phases

Note: For more information on this interface, see *AMBA AXI(3) Protocol v2.0 Specification*.

Table 7-6 AXI Master Interface Signals

Port Name	I/O	Description
xm_awid[(DWC_USB3_IDWIDTH-1):0]	O	<p>AXI Master write address identification. Gives identification tag for write address signals.</p> <p>Note: Currently, the AXI Master only requests ID 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xm_awaddr[(DWC_USB3_AWIDTH-1):0]	O	<p>AXI Master write address. Specifies address of first transfer in write burst transaction. Associated control signals used to determine addresses of remaining transfers in burst.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_awlen[(DWC_USB3_BURSTWIDTH-1):0]	O	<p>AXI Master write burst length. Specifies exact number of transfers in burst; determines number of data transfers associated with address.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_awsize[2:0]	O	<p>AXI Master write burst size. Indicates size of each transfer in burst. Byte lane strobes indicate exactly which byte lanes to update.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: (DWC_USB3_MODE!=3 && DWC_USB3_MBUS_TYPE==3) ? "None" : (DWC_USB3_MBUS_TYPE != 1 DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 1 ? ("0:0=None;1:1=bus_clk_early;2:2=None") : ("0:1=bus_clk_early;2:2=None"))</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_awburst[1:0]	O	<p>AXI Master write burst. Combined with size information, shows how address for each transfer within burst is calculated.</p> <p>Note: Currently only the INCR burst is used.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_awlock[1:0]	O	<p>AXI Master write lock. Provides additional information about characteristics of transfer.</p> <p>Note: Currently, the USB 3.0 Controller does not do atomic access. This output will always be 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xm_awcache[3:0]	O	<p>AXI Master write cache. For cache type bit assignments, see "Cache Type Bit Assignments" table in the Programming Guide.</p> <p>Note: All cache type bits are programmable using GSBUSCFG0 register (see GSBUSCFG0 section of the Programming Guide).</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_awprot[2:0]	O	<p>AXI Master write protection. Indicates normal, privileged, or secure protection level of transaction and whether transaction is data access or instruction access.</p> <p>Note: Currently only normal, non-secure, and data accesses are indicated. This output signal is always 3'b010.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_awvalid	O	<p>AXI Master write address valid. Indicates valid write address and control information are available. Address and control information remain stable until a ready signal is high.</p> <ul style="list-style-type: none"> ■ 0: Address and Control information not available ■ 1: Address and Control information available <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_awmisc_info[(DWC_USB3_MISCWID TH-1):0]	O	<p>AXI Master Write Request Miscellaneous Information. This signal is not currently used. You can leave this signal unconnected.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
xm_wid[(DWC_USB3_IDWIDTH-1):0]	O	<p>AXI Master write identification tag of write data transfer.</p> <p>Note: Currently, the AXI Master only requests ID 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xm_wvalid	O	<p>AXI Master write valid. Indicates valid write data and strobes are available.</p> <ul style="list-style-type: none"> ■ 0: Write data and strobes not available ■ 1: Write data and strobes available <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: Yes Power Domain: Vcc Active State: High</p>
xm_wlast	O	<p>AXI Master write last. Indicates last transfer in write burst.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: Yes Power Domain: Vcc Active State: High</p>
xm_wdata[(DWC_USB3_MDWIDTH-1):0]	O	<p>AXI Master write data.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: Yes Power Domain: Vcc Active State: High</p>
xm_wstrb[(DWC_USB3_MDLOG-1):0]	O	<p>AXI Master write strobes. Indicates which byte lanes to update in memory. One write strobe for each eight bits of write data bus. WSTRB[n] is associated with the following byte of MDATA: MDATA[8n+7: 8n]</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: Yes Power Domain: Vcc Active State: High</p>
xm_bready	O	<p>AXI Master response ready. Indicates master can accept response information.</p> <ul style="list-style-type: none"> ■ 0: Master not ready ■ 1: Master ready <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: Yes Power Domain: Vcc Active State: High</p>

Port Name	I/O	Description
xm_arid[(DWC_USB3_IDWIDTH-1):0]	O	<p>AXI Master read address identification tag for read address signals.</p> <p>Note: Currently, the AXI Master only requests ID 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_arvalid	O	<p>AXI Master read address valid. When high, indicates read address and control information is valid and remains stable until already is high.</p> <ul style="list-style-type: none"> ■ 0: Address and Control information not valid ■ 1: Address and Control information valid <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_araddr[(DWC_USB3_AWIDHT-1):0]	O	<p>AXI Master read address. Gives initial address of read burst transaction. Only start address of burst is provided, and control signals issued alongside address show how address is calculated for remaining transfers in burst.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_arlen[(DWC_USB3_BURSTWIDTH-1):0]	O	<p>AXI Master read burst length. Gives exact number of transfers in burst; determines number of data transfers associated with the address.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_arsize[2:0]	O	<p>AXI Master read burst size. Indicates size of each transfer in burst.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: (DWC_USB3_MODE!=3 && DWC_USB3_MBUS_TYPE==3) ? "None" : (DWC_USB3_MBUS_TYPE != 1 DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 1 ? ("0:0=None;1:1=bus_clk_early;2:2=None") : ("0:1=bus_clk_early;2:2=None"))</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xm_arburst[1:0]	O	<p>AXI Master read burst. Combined with size information, shows how address for each transfer within burst is calculated.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_arlock[1:0]	O	<p>AXI Master read lock. Encoded value indicates whether the transfer is a normal, exclusive, or locked access.</p> <p>Note: Currently the USB 3.0 Controller does not do atomic access. This output is always 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_arcache[3:0]	O	<p>AXI Master read cache. For cache type bit assignments, see "Cache Type Bit Assignments" table in the Programming Guide.</p> <p>Note: All cache type bits are programmable using the GSBUSCFG0 register (see GSBUSCFG0 section of the Programming Guide).</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_arprot[2:0]	O	<p>AXI Master read protection. Indicates normal, privileged, or secure protection level of transaction and whether the transaction is data access or instruction access.</p> <p>Note: Currently only normal, non-secure, and data accesses are indicated. This output signal is always 3'b010.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_armisc_info[(DWC_USB3_MISCWIDT H-1):0]	O	<p>AXI Master Read Request Miscellaneous Information. This signal is not currently used. You can leave this signal unconnected.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>

Port Name	I/O	Description
xm_rready	O	<p>AXI Master read ready. Indicates that the master can accept read data and response information.</p> <ul style="list-style-type: none"> ■ 0: Master not ready ■ 1: Master ready <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_csysack	O	<p>AXI Master Low-power request acknowledgement. Indicates acknowledgement from a peripheral of a system low-power request.</p> <p>Note: This feature is currently not supported, and this output is hard-wired to 1.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_cactive	O	<p>AXI Master Clock active. Indicates that peripheral requires clock signal:</p> <ul style="list-style-type: none"> ■ 1: Peripheral clock required ■ 0: Peripheral clock not required <p>Note: This feature is currently not supported, and this output is hard-wired to 1.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_awready	I	<p>AXI Master write address ready. Indicates that slave is ready to accept address and associated control signals.</p> <ul style="list-style-type: none"> ■ 0: Slave not ready ■ 1: Slave ready <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xm_wready	I	<p>AXI Master write ready. Indicates that slave can accept write data.</p> <ul style="list-style-type: none"> ■ 0: Slave not ready ■ 1: Slave ready <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>
xm_bid[(DWC_USB3_IDWIDTH-1):0]	I	<p>AXI Master write response identification tag.</p> <p>Note: Currently, the AXI Master only requests ID 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>
xm_bvalid	I	<p>AXI Master write response valid. Indicates that valid write response is available.</p> <ul style="list-style-type: none"> ■ 0: Write response not available ■ 1: Write response available <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>
xm_bresp[1:0]	I	<p>AXI Master write response. Indicates status of write transaction.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>
xm_bmisc_info[(DWC_USB3_MISCWIDTH-1):0]	I	<p>AXI Master Write Response Miscellaneous Information. This signal is not currently used. Tie this signal to '0'.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: N/A Registered: N/A Power Domain: Vcc Active State: N/A</p>

Port Name	I/O	Description
xm_already	I	<p>AXI Master read address ready. Indicates that the slave is ready to accept address and associated control signals.</p> <ul style="list-style-type: none"> ■ 0: Slave not ready ■ 1: Slave ready <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>
xm_rid[(DWC_USB3_IDWIDTH-1):0]	I	<p>AXI Master Read identification tag.</p> <p>Note: Currently, the AXI Master only requests ID 0.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>
xm_rvalid	I	<p>AXI Master read valid. Indicates that required read data is available and read transfer can complete.</p> <ul style="list-style-type: none"> ■ 0: Read data not available ■ 1: Read data available <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>
xm_rlast	I	<p>AXI Master read last. Indicates last transfer in read burst.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>
xm_rdata[(DWC_USB3_MDWIDTH-1):0]	I	<p>AXI Master read data.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3) Synchronous To: bus_clk_early Registered: No Power Domain: Vcc Active State: High</p>

Port Name	I/O	Description
xm_rmisc_info[(DWC_USB3_MISCWIDTH -1):0]	I	<p>AXI Master Read Response Miscellaneous Information. This signal is not currently used. Tie this signal to '0'.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
xm_rresp[1:0]	I	<p>AXI Master read response. Indicates status of read transaction.</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xm_csysreq	I	<p>AXI Master System low-power request from system clock controller for peripheral to enter low-power state.</p> <p>Note: This feature is currently not supported and this input must be hard-wired to 0. (The value of 1 means non-active.)</p> <p>Exists: (DWC_USB3_MBUS_TYPE == 1 && DWC_USB3_MODE != 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>

7.7 AXI Slave Interface Signals

xs_awid	- xs_awready
xs_awaddr	- xs_awaddr_parerr
xs_awlen	- xs_wready
xs_awsize	- xs_bid
xs_awburst	- xs_bresp
xs_awlock	- xs_bvalid
xs_awcache	- xs_bmisc_info
xs_awprot	- xs_arready
xs_awmisc_info	- xs_rid
xs_awvalid	- xs_rdata
xs_wid	- xs_rresp
xs_wdata	- xs_rlast
xs_wstrb	- xs_rvalid
xs_wlast	- xs_rmisc_info
xs_wvalid	- xs_csysack
xs_bready	- xs_cactive
xs_arid	
xs_araddr	
xs_arlen	
xs_arsize	
xs_arburst	
xs_arlock	
xs_arcache	
xs_arprot	
xs_armisc_info	
xs_arvalid	
xs_rready	
xs_csysreq	

This interface connects the DWC_usb3 controller as a slave on the AXI for access to the CSR of the DWC_usb3 controller.

The AXI Slave interface is in the bus_clk_early clock domain.

Features supported on this interface include:

- Single and burst transfers
- 32-bit transfers (Non-32-bit writes corrupt registers; reads return invalid data)
- Support for all possible values of ID
- Support for burst-based transactions with only start address issued

Note: For more information on this interface, see *AMBA AXI(3) Protocol v2.0 Specification*.

Table 7-7 AXI Slave Interface Signals

Port Name	I/O	Description
xs_awready	O	<p>AXI Slave write address ready. Indicates that slave is ready to accept address and associated control signals.</p> <ul style="list-style-type: none"> ■ 0: Slave not ready ■ 1: Slave ready <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_awaddr_parerr	O	<p>AXI Slave write address parity error.</p> <p>Note: Currently, this signal is hardwired to 0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
xs_wready	O	<p>AXI Slave write ready. Indicates that slave can accept write data.</p> <ul style="list-style-type: none"> ■ 0: Slave not ready ■ 1: Slave ready <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_bid[(DWC_USB3_SIDWIDTH-1):0]	O	<p>AXI Slave write response identification tag.</p> <p>Note: The AXI Slave processes only one ID request at a time. Only after completing the current ID data transfer, the AXI Slave processes the next ID request.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_bresp[1:0]	O	<p>AXI Slave write response. Indicates status of write transaction. This signal indicates an error if the data phase ID does not match the address phase ID. For example, if the AWID = 0 and the WID != 0, then the controller responds with bresp slave error.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_bvalid	O	<p>AXI Slave write response valid. Indicates that valid write response is available.</p> <ul style="list-style-type: none"> ■ 0: Write response not available ■ 1: Write response available <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_bmisc_info[(DWC_USB3_MISCWIDTH -1):0]	O	<p>Write response miscellaneous information</p> <p>Note: This signal is driven to zero.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
xs_arready	O	<p>AXI Slave read address ready. Indicates that slave is ready to accept address and associated control signals.</p> <ul style="list-style-type: none"> ■ 0: Slave not ready ■ 1: Slave ready <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_rid[(DWC_USB3_SIDWIDTH-1):0]	O	<p>AXI Slave Read identification tag.</p> <p>Note: The AXI Slave processes only one ID request at a time. Only after completing the current ID data transfer, the AXI Slave will process the next ID request.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_rdata[(DWC_USB3_SDWIDTH-1):0]	O	<p>AXI Slave read data.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_rresp[1:0]	O	<p>AXI Slave read response. Indicates status of read transaction.</p> <p>Note: Currently, the controller always responds with xs_rresp = 0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_rlast	O	<p>AXI Slave read last. Indicates last transfer in read burst.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_rvalid	O	<p>AXI Slave read valid. Indicates that required read data is available and read transfer can complete.</p> <ul style="list-style-type: none"> ■ 0: Read data not available ■ 1: Read data available <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_rmisc_info[(DWC_USB3_MISCWIDTH-1):0]	O	<p>Read response miscellaneous information</p> <p>Note: This signal is driven to zero.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
xs_csysack	O	<p>AXI Slave Low-power request acknowledgment. Indicates acknowledgment from peripheral of system low-power request.</p> <p>Note: This feature is currently not supported and this output is hard-wired to 0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_cactive	O	<p>AXI Slave Clock active. Indicates that peripheral requires clock signal:</p> <ul style="list-style-type: none"> ■ 1: Peripheral clock required ■ 0: Peripheral clock not required <p>Note: This feature is currently not supported and this output is hard-wired to 0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_awid[(DWC_USB3_SIDWIDTH-1):0]	I	<p>AXI Slave write address identification. Gives identification tag for write address signals.</p> <p>Note: The AXI Slave processes only one ID request at a time. Only after completing the current ID data transfer, the AXI Slave processes the next ID request.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_awaddr[(DWC_USB3_AWIDTH-1):0]	I	<p>AXI Slave write address. Specifies address of first transfer in write burst transaction. Associated control signals used to determine addresses of remaining transfers in burst.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_awlen[(DWC_USB3_BURSTWIDTH-1):0]	I	<p>AXI Slave write burst length. Specifies exact number of transfers in burst; determines number of data transfers associated with address.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_awsize[2:0]	I	<p>AXI Slave write burst size. Indicates size of each transfer in burst. Byte lane strobes indicate exactly which byte lanes to update.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_awburst[1:0]	I	<p>AXI Slave write burst. Combined with size information, shows how address for each transfer within burst is calculated.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_awlock[1:0]	I	<p>AXI Slave write lock. Provides additional information about characteristics of transfer. This feature is not supported. This input must be driven to 2'b00 (normal access).</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_awcache[3:0]	I	<p>AXI Slave write cache. Indicates bufferable, cacheable, write-through, and write-back attributes of transaction. This feature is not supported. This input must be driven to 4'h0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_awprot[2:0]	I	<p>AXI Slave write protection. Indicates normal, privileged, or secure protection level of transaction and whether transaction is data access or instruction access. This feature is not supported. This input must be driven to 3'h0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_awmisc_info[(DWC_USB3_MISCWIDT H-1):0]	I	<p>Write request miscellaneous information</p> <p>Note: This signal is not used.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
xs_awvalid	I	<p>AXI Slave write address valid. Indicates valid write address and control information are available. Address and control information remain stable until awready signal is high.</p> <ul style="list-style-type: none"> ■ 0: Address and control information not available ■ 1: Address and control information available <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_wid[(DWC_USB3_SIDWIDTH-1):0]	I	<p>AXI Slave write identification tag of write data transfer.</p> <p>Note: The AXI Slave processes only one ID request at a time. Only after completing the current ID data transfer, the AXI Slave processes the next ID request.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_wdata[(DWC_USB3_SDWIDTH-1):0]	I	<p>AXI Slave write data.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_wstrb[(DWC_USB3_SDLOG-1):0]	I	<p>AXI Slave write strobes. Indicates which byte lanes to update in memory. One write strobe for each eight bits of write data bus. WSTRB[n] is associated with the following byte of SDATA: SDATA[8n+7: 8n]</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_wlast	I	<p>AXI Slave write last. Indicates last transfer in write burst.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_wvalid	I	<p>AXI Slave write valid. Indicates valid write data and strobes are available.</p> <ul style="list-style-type: none"> ■ 0: Write data and strobes not available ■ 1: Write data and strobes available <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_bready	I	<p>AXI Slave response ready. Indicates that the master can accept response information.</p> <ul style="list-style-type: none"> ■ 0: Master not ready ■ 1: Master ready <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_arid[(DWC_USB3_SIWIDTH-1):0]	I	<p>AXI Slave read address identification tag for read address signals.</p> <p>Note: The AXI Slave processes only one ID request at a time. Only after completing the current ID data transfer, the AXI Slave processes the next ID request.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_araddr[(DWC_USB3_AWIDTH-1):0]	I	<p>AXI Slave read address. Gives initial address of read burst transaction. Only the start address of the burst is provided, and control signals issued alongside address show how the address is calculated for remaining transfers in the burst.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_arlen[(DWC_USB3_BURSTWIDTH-1):0]	I	<p>AXI Slave read burst length. Gives exact number of transfers in burst; determines number of data transfers associated with the address.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_arsize[2:0]	I	<p>AXI Slave read burst size. Indicates size of each transfer in burst.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_arburst[1:0]	I	<p>AXI Slave read burst. Combined with size information, shows how address for each transfer within burst is calculated.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_arlock[1:0]	I	<p>AXI Slave read lock. Encoded value indicates whether the transfer is a normal, exclusive, or locked access. This feature is not supported. This input must be driven to 2'h0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_arcache[3:0]	I	<p>AXI Slave read cache. Indicates bufferable, cacheable, read-through, and read-back attributes of transaction. This feature is not supported. This input must be driven to 4'h0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_arprot[2:0]	I	<p>AXI Slave read protection. Indicates normal, privileged, or secure protection level of transaction and whether transaction is data access or instruction access. This feature is not supported. This input must be driven to 3'h0.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_armisc_info[(DWC_USB3_MISCWIDT_H-1):0]	I	<p>Read request miscellaneous information</p> <p>Note: This signal is not used.</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
xs_arvalid	I	<p>AXI Slave read address valid. When high, indicates read address and control information is valid and remains stable until already is high.</p> <ul style="list-style-type: none"> ■ 0: Address and Control information not valid ■ 1: Address and Control information valid <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
xs_rready	I	<p>AXI Slave read ready. Indicates master can accept read data and response information.</p> <ul style="list-style-type: none"> ■ 0: Master not ready ■ 1: Master ready <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: bus_clk_early</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
xs_csysreq	I	<p>AXI Slave System low-power request from system clock controller for peripheral to enter low-power state.</p> <p>Note: This feature is currently not supported and this input must be driven to 1. (A value of 1 means non-active.)</p> <p>Exists: (DWC_USB3_SBUS_TYPE == 1 && DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>

7.8 RAM Interface Signals

`ramN_p1_rdata` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1) -

- `ramN_p1_wr_n` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)
- `ramN_p1_ce_n` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)
- `ramN_p1_addr` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)
- `ramN_p1_wdata` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)
- `ramN_p2_wr_n` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)
- `ramN_p2_ce_n` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)
- `ramN_p2_addr` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)
- `ramN_p2_wdata` (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)

This interface connects the DWC_usb3 controller with one, two, or three Single- or 2-Port Synchronous Static RAMs.

The RAM0 interface is in the ram_clk_gated_ram0 clock domain. The RAM1 and RAM2 interfaces are in the ram_clk_gated clock domain.

For the Hub, always use two 2-Port RAMs of 33-bit data width. The Port-1 is always read, and Port-2 is always written in a Hub configuration.

Table 7-8 RAM Interface Signals

Port Name	I/O	Description
<code>ramN_p1_wr_n[(DWC_USB3_RAM0_P1_WE_W-1):0]</code> (for N = 0; N <= DWC_USB3_NUM_RAMPS-1)	O	<p>RAM<N> Port1 Write Enable. Qualifies <code>ramN_p1_addr</code> and <code>ramN_p1_addr</code>.</p> <p>Note: For a Hub configuration, Port-1 write is not used.</p> <p>Exists: (DWC_USB3_NUM_RAMPS > N && DWC_USB3_MBUS_TYPE != 4 && DWC_USB3_SPRAM_TYP == 1)</p> <p>Synchronous To: (DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : (DWC_USB3_SPRAM_TYP==1) ? ((DWC_USB3_EN_USB2_ONLY==1) ? ((DWC_USB3_HSPHY_INTERFACE==3) ? "ram_clk_in,utmi_clk[0],ulp_i_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2) ? "ram_clk_in,ulp_i_clk[0]" : "ram_clk_in,utmi_clk[0]") : "ram_clk_in") : "None"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>

Port Name	I/O	Description
ramN_p1_ce_n (for N = 0; N <= DWC_USB3_NUM_RAMs-1)	O	<p>RAM<N> Port1 Chip Select. Indicates that the RAM<N> is being accessed for a read or a write.</p> <p>Exists: (DWC_USB3_NUM_RAMs > N && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: (DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : ((DWC_USB3_EN_USB2_ONLY==1) ? ((DWC_USB3_HSPHY_INTERFACE==3) ? "ram_clk_in,utmi_clk[0],ulpi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2) ? "ram_clk_in,ulpi_clk[0]" : "ram_clk_in,utmi_clk[0]" : "ram_clk_in")</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>
ramN_p1_addr[(DWC_USB3_RAM0_A-1):0] (for N = 0; N <= DWC_USB3_NUM_RAMs-1)	O	<p>RAM<N> Port1 RAM Address Bus. Provides the address for a read or write operation.</p> <p>Exists: (DWC_USB3_NUM_RAMs > N && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: (DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : ((DWC_USB3_EN_USB2_ONLY==1) ? ((DWC_USB3_HSPHY_INTERFACE==3) ? "ram_clk_in,utmi_clk[0],ulpi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2) ? "ram_clk_in,ulpi_clk[0]" : "ram_clk_in,utmi_clk[0]" : "ram_clk_in")</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
ramN_p1_wdata[(DWC_USB3_RDWRTH-1):0] (for N = 0; N <= DWC_USB3_NUM_RAMs-1)	O	<p>RAM<N> Port1 Write Data. Provides data to be written to the RAM<N> address specified by ramN_p1_addr.</p> <p>Note: For a Hub configuration,</p> <ul style="list-style-type: none"> ■ 33-bit data width RAM0 and RAM1 are used. ■ Port-1 Write data is not used. <p>Exists: (DWC_USB3_NUM_RAMs > N && DWC_USB3_MBUS_TYPE != 4 && DWC_USB3_SPRAM_TYP == 1)</p> <p>Synchronous To: (DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : (DWC_USB3_SPRAM_TYP==1) ? ((DWC_USB3_EN_USB2_ONLY==1) ? ((DWC_USB3_HSPHY_INTERFACE==3) ? "ram_clk_in,utmi_clk[0],ulpi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2) ? "ram_clk_in,ulpi_clk[0]" : "ram_clk_in,utmi_clk[0]" : "ram_clk_in") : "None"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
ramN_p2_wr_n[(DWC_USB3_RAM0_P1_WE_W-1):0] (for N = 0; N <= DWC_USB3_NUM_RAMS-1)	O	<p>RAM<N> Port2 Write Enable. Qualifies ramN_p2_addr and ramN_p2_addr.</p> <p>Note: For a Hub configuration, Port-1 write is not used.</p> <p>Exists: (DWC_USB3_NUM_RAMS > N && DWC_USB3_MBUS_TYPE != 4 && DWC_USB3_SPRAM_TYP == 0)</p> <p>Synchronous To: (DWC_USB3_SPRAM_TYP==1) ? "None" : ((DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : ((DWC_USB3_EN_ECC==1) ? ((DWC_USB3_EN_USB2_ONLY==1) ? ((DWC_USB3_HSPHY_INTERFACE==3) ? "ram_clk_in,utmi_clk[0],ulpi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2) ? "ram_clk_in,ulpi_clk[0]" : "ram_clk_in,utmi_clk[0]") : "ram_clk_in") : "None"))</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>
ramN_p2_ce_n (for N = 0; N <= DWC_USB3_NUM_RAMS-1)	O	<p>RAM<N> Port2 Chip Select. Indicates that the RAM<N> is being accessed for a read or a write.</p> <p>Exists: (DWC_USB3_NUM_RAMS > N && DWC_USB3_MBUS_TYPE != 4 && DWC_USB3_SPRAM_TYP == 0)</p> <p>Synchronous To: (DWC_USB3_SPRAM_TYP==1) ? "None" : ((DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : ((DWC_USB3_EN_USB2_ONLY==1) ? ((DWC_USB3_HSPHY_INTERFACE==3) ? "ram_clk_in,utmi_clk[0],ulpi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2) ? "ram_clk_in,ulpi_clk[0]" : "ram_clk_in,utmi_clk[0]") : "ram_clk_in"))</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>
ramN_p2_addr[(DWC_USB3_RAM0_A-1):0] (for N = 0; N <= DWC_USB3_NUM_RAMS-1)	O	<p>RAM<N> Port2 RAM Address Bus. Provides the address for a read or write operation.</p> <p>Exists: (DWC_USB3_NUM_RAMS > N && DWC_USB3_MBUS_TYPE != 4 && DWC_USB3_SPRAM_TYP == 0)</p> <p>Synchronous To: (DWC_USB3_SPRAM_TYP==1) ? "None" : ((DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : ((DWC_USB3_EN_USB2_ONLY==1) ? ((DWC_USB3_HSPHY_INTERFACE==3) ? "ram_clk_in,utmi_clk[0],ulpi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2) ? "ram_clk_in,ulpi_clk[0]" : "ram_clk_in,utmi_clk[0]") : "ram_clk_in"))</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
ramN_p2_wdata[(DWC_USB3_RDWRTH-1):0] (for N = 0; N <= DWC_USB3_NUM_RAMs-1)	O	<p>RAM<N> Port2 Write Data. Provides data to be written to the RAM<N> address specified by ramn_p2_addr.</p> <p>Note: For a Hub configuration,</p> <ul style="list-style-type: none"> ■ 33-bit data width RAM0 and RAM1 are used. ■ Port-1 Write data is not used. <p>Exists: (DWC_USB3_NUM_RAMs > N && DWC_USB3_MBUS_TYPE != 4 && DWC_USB3_SPRAM_TYP == 0)</p> <p>Synchronous To: (DWC_USB3_SPRAM_TYP == 1) ? "None" : ((DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : ((DWC_USB3_EN_USB2_ONLY==1) ? ((DWC_USB3_HSPHY_INTERFACE==3) ? "ram_clk_in,utmi_clk[0],ulpi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2) ? "ram_clk_in,ulpi_clk[0]" : "ram_clk_in,utmi_clk[0]" : "ram_clk_in"))</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
ramN_p1_rdata[(DWC_USB3_RDWRTH-1):0] (for N = 0; N <= DWC_USB3_NUM_RAMs-1)	I	<p>RAM<N> Port1 Read Data. Provides read data from the RAM address specified by ramN_p1_addr.</p> <p>Note: For a Hub configuration,</p> <ul style="list-style-type: none"> ■ 33-bit data width RAM0 and RAM1 are used. ■ Port-2 read data is not used. <p>Exists: (DWC_USB3_NUM_RAMs > N && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: (DWC_USB3_MODE==3) ? "pipe3_rx_pclk[0],suspend_clk" : "ram_clk_in"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.9 USB 3.0 PIPE3 PHY Interface Signals

pipe3_rx_pclk	- pipe3_reset_n
pipe3_tx_pclk	- pipe3_TxData
pipe3_RxValid	- pipe3_TxDataK
pipe3_RxData	- pipe3_PHY_Mode
pipe3_RxDataK	- pipe3_RxEqTrain
pipe3_PhyStatus_async	- pipe3_ElasBufferMode
pipe3_PhyStatus	- pipe3_TxDetectRxLoopbk
pipe3_RxElecIdle	- pipe3_TxElecIdle
pipe3_RxStatus	- pipe3_compliance
pipe3_PowerPresent	- pipe3_RxPolarity
pipe3_DataBusWidth	- pipe3_PowerDown
	- pipe3_Rate
	- pipe3_TxDeemph
	- pipe3_TxMargin
	- pipe3_TxSwing
	- pipe3_RxTermination
	- pipe3_TxOnesZeros

Table 7-9 USB 3.0 PIPE3 PHY Interface Signals

Port Name	I/O	Description
pipe3_reset_n[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 Asynchronous Reset. Only deassertion is synchronous.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: DWC_USB3_MODE == 3 ? "None" : (DWC_USB3_HSPHY_INTERFACE==1 ? "bus_clk_early,suspend_clk,pipe3_rx_pclk[0],utmi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2 ? "bus_clk_early,suspend_clk,pipe3_rx_pclk[0],ulp_i_clk[0]" : "bus_clk_early,suspend_clk,pipe3_rx_pclk[0],utmi_clk[0],ulp_i_clk[0]"))</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>
pipe3_TxData[((32*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	O	<p>PIPE3 Transmit Data.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
pipe3_TxDataK[((4*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	O	<p>PIPE3 Transmit DataK.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_PHY_Mode[((2*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	O	<p>PIPE3 PHY Mode. Hard wire to USB3 mode.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_RxEqTrain[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 equalization training.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] or pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_ElasBufferMode[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 Elasticity Buffer Mode.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: Half-Full mode ■ 1'b1: Empty mode <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] or pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
pipe3_TxDetectRxLoopbk[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 Receive Detect/LoopBack.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_TxEleclidle[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 Transmit Electrical Idle.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] or pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_compliance[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 PCIe Compliance Pattern. Hard wired to 1'b0.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_RxPolarity[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 Receive Polarity.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
pipe3_PowerDown[((4*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	O	<p>PIPE3 Power Down Transceiver.</p> <p>Values: 4'b0000 - P0 4'b0001 - P1 4'b0010 - P2 4'b0011 - P3 4'b0100 - P3CPM 4'b1100 - P4 Supports new PHY power states P3.CPM and P4. If Your phy supports P0,P1,P2 and P3 states only, then connect lower 2 bits to your phy and leave the upper 2 bits unconnected.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_EN_PWROPT == 2 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_Rate[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 Data Rate. Hard wired to 1'b1.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: 2.5GHz ■ 1'b1: 5GHz <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_TxDeemph[((2*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	O	<p>PIPE3 Transmit De-emphasis.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_TxMargin[((3*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	O	<p>PIPE3 Transmit Voltage Level.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
pipe3_TxSwing[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 Voltage Swing. Values:</p> <ul style="list-style-type: none"> ■ 1'b0: Full swing ■ 1'b1: Low swing <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1 Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report. Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes" Power Domain: Vcc Active State: High</p>
pipe3_RxTermination[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 Receiver Termination. Exists: DWC_USB3_SSPHY_INTERFACE == 1 Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report. Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes" Power Domain: Vcc Active State: High</p>
pipe3_TxOnesZeros[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>PIPE3 USB3 Transmit Compliance. Exists: DWC_USB3_SSPHY_INTERFACE == 1 Synchronous To: This port uses pipe3_tx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report. Registered: DWC_USB3_SSPHY_INTERFACE_NUM_PIPE == 0 ? "No" : "Yes" Power Domain: Vcc Active State: High</p>
pipe3_rx_pclk[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>PIPE3 Rx Port Clock. Pipe 3 output clock from PHY. Capture and logic clock for PIPE3 inputs to controller. Frequency depends PHY data width (32 bit -> 125 MHz, 16 bit -> 250 MHz, 8-bit -> 500 MHz). Exists: DWC_USB3_SSPHY_INTERFACE == 1 Synchronous To: None Registered: N/A Power Domain: Vcc Active State: N/A</p>

Port Name	I/O	Description
pipe3_tx_pclk[(DWC_USB3_NUM_U3_R_OOT_PORTS-1):0]	I	<p>PIPE3 Tx Port Clock. Pipe 3 output clock from PHY. Drive and logic clock for PIPE3 outputs from controller to PHY. Frequency depends on PHY data width (32 bit -> 125 MHz, 16 bit -> 250 MHz, 8-bit -> 500 MHz). The pipe3_tx_pclk and the pipe3_rx_pclk must be connected to the same PHY clock output, and must have the same clock insertion delay for layout purposes.</p> <p>For FPGA testing, the pipe3_tx_pclk can be a delayed version of the pipe3_rx_pclk clock, but cannot be an asynchronous version. The setup and hold timing must be met between the pipe3_rx_clk and the pipe3_tx_clk.</p> <p>In the case of Port-0, the setup and hold timing between the pipe3_rx_clk, the pipe3_tx_clk, and the internal mac3_clk must be met. For more details, refer to “U3 PIU Clocks” on page 115.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
pipe3_RxValid[(DWC_USB3_NUM_U3_R_OOT_PORTS-1):0]	I	<p>PIPE3 Receive Data Valid.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_MODE!=3 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_RxData[((32*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	I	<p>PIPE3 Receive Data.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_MODE!=3 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_RxDataK[((4*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	I	<p>PIPE3 Receive DataK.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_MODE!=3 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
pipe3_Phystatus_async[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>PIPE3 PHY Status Asynchronous. Connect this signal to the same pipe3_Phystatus signal from the PHY. The reason for implementing this duplicate signal is to achieve a clean FPGA implementation that meets timing. The pipe3_Phystatus signal is synchronous to the pipe3_pclk during a non-P3 state, and asynchronous to the pipe3_pclk during P3 because there is no pipe3_pclk during P3.</p> <p>All of the PIPE3 input signal sampling is required to be inferred in the FPGA I/O ring flops to meet timing. If a signal is sampled by two different clocks, the FPGA synthesis tools do not use the FPGA I/O flops to sample the signal and instead they use two normal flops in a CLB. This can cause a large routing delay of these critical signals from the external PHY (which runs at 250MHz) and result in not meeting the timing requirements.</p> <p>By duplicating the signal, the FPGA synthesis tool can use the I/O ring flops to register pipe3_Phystatus with pipe3_pclk which has critical timing.</p> <p>For more details PhyStatus, refer to the following sections:</p> <ul style="list-style-type: none"> ■ "SS PHY Status Transfer" on page 118 ■ "Inputs from PHY" section in the User Guide <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_Phystatus[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>PIPE3 PHY Status. For details on PhyStatus, refer to the following sections:</p> <ul style="list-style-type: none"> ■ "SS PHY Status Transfer" on page 118 ■ "Inputs from PHY" section in the User Guide, and ■ Description of pipe3_Phystatus_async signal in this table <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
pipe3_RxEleIdle[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>PIPE3 PHY Electrical Idle.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, ram_clk_in, utmi_clk[0] or ulpi_clk[0], where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_MODE!=3 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_RxStatus[((3*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	I	<p>PIPE3 PHY Receive Status.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, utmi_clk[0] or ulpi_clk[0], where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: DWC_USB3_MODE!=3 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_PowerPresent[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>PIPE3 PHY Power Present. This must be a filtered signal and must be the same as utmiotg_vbusvalid. In Host-only mode, this signal must be connected to logic high value (1'b1). For complete usage information, refer to section "Integrating the Controller with the PHY" in the User Guide.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
pipe3_DataBusWidth[((DWC_USB3_NUM_U3_ROOT_PORTS*2)-1):0]	I	<p>PIPE3 Data Bus Width.</p> <ul style="list-style-type: none"> ■ 00: 32-bit ■ 01: 16-bits ■ 10: Reserved ■ 11: Reserved <p>Sampled once after the PHY reset is released.</p> <p>Exists: DWC_USB3_SSPHY_INTERFACE == 1</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.10 USB 2.0 UTMI+ Parallel Interface Signals

utmi_clk	- utmi_suspend_com_n
utmi_txready	- utmi_l1_suspend_com_n
utmi_rx_data	- utmi_tx_data
utmi_rxvalidh	- utmi_txvalidh
utmi_rxvalid	- utmi_txvalid
utmi_rxactive	- utmi_opmode
utmi_rxerror	- utmi_suspend_n
utmi_linenstate	- utmi_termselect
utmi_hostdisconnect	- utmi_xcvrselect
utmisrp_bvalid	- utmi_word_if
utmiotg_vbusvalid	- utmi_fsls_low_power
	- utmi_fslsserialmode

The DWC_usb3 controller supports the UTMI+ Level 3 (L3) protocol, and this interface connects the controller to any USB 2.0 PHY that complies with UTMI+ L3 or UTMI.

Table 7-10 USB 2.0 UTMI+ Parallel Interface Signals

Port Name	I/O	Description
utmi_suspend_com_n	O	<p>UTMI common Suspend. For device mode of operation, this signal is same as inverted utmi_suspend_n[0]. For host mode, this signal is asserted (low) if either of the following conditions is met:</p> <ul style="list-style-type: none"> ■ Any of the host ports (either 3.0 or 2.0 ports) is not suspended (in this case, for the USB 3.0 port, suspend is equivalent to P3; for the USB 2.0 port, it is either L2 suspend or disconnected state if DWC_USB3_SUSPEND_ON_DISCONNECT_EN is high). The dependency on the USB 3.0 ports is nullified if the GCTL[10] bit or GFLADJ[23] bit is set. ■ Hardware LPM is enabled. The dependency hardware LPM is nullified if the GFLADJ[23] bit is set. <p>For host mode, this signal is high if all of the following conditions are met:</p> <ul style="list-style-type: none"> ■ All the ports are L2 suspended (or disconnected if DWC_USB3_SUSPEND_ON_DISCONNECT_EN is 1). ■ GUSB2PHYCFGn.SusPHY is 1 for all the ports. ■ All the SS PHYs are in P3. The dependency of the USB 3.0 ports is nullified if the GCTL[10] or GFLADJ[23] bit is set. ■ Hardware LPM is not enabled. The dependency hardware LPM is nullified if the GFLADJ[23] bit is set. <p>This signal must be used with a third-party PHY that does not have a free-running UTMI clock available. In Host/DRD mode, it must be inverted and connected to the SUSPENDM input of the port0 PHY. In Device mode, utmi_suspend_n[0] can also be connected to the SUSPENDM input of the PHY directly.</p> <p>When using the Synopsys PHY, you can leave this signal unconnected. For more information on how to use this signal, refer to "Integrating with USB 2.0 PHY" section of the User Guide.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ If the selected PHY has only SUSPENDM input, it can be connected to (~utmi_suspend_com_n & ~utmi_l1_suspend_com_n). In these cases, the PHY recovery time (from suspend) must meet the LPM HIRD duration. ■ The overloading of the utmi_l1_suspend_com_n signal with the utmi_sleep_n signal is done only if the GUCTL1[1] bit is set high. For the description of the GUCTL1[1] bit, refer to GUCTL1 section of the Programming Guide. <p>If some of the ports are in L2 suspend and the others are in L1 suspend or L1 sleep (in other words, each port is either in L2 suspend, L1 suspend or L1 sleep with at least one in L1 suspend or sleep) then the following occurs:</p>

Port Name	I/O	Description
utmi_suspend_com_n...(cont.)	O.	<ul style="list-style-type: none"> ■ utmi_suspend_com_n signal is not asserted high ■ the utmi_l1_suspend_com_n signal is asserted high <p>If the utmi_l1_suspend_com_n signal is used to control the Port0 PHY suspend, the Port0 PHY must have a resume recovery time that meets the least LPM HIRD duration used.</p> <p>Exists: ((DWC_USB3_FSPHY_INTERFACE == 1 DWC_USB3_FSPHY_INTERFACE == 3 DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3) && (DWC_USB3_HOST_NUM_ROOT_PORTS > 1) && (DWC_USB3_MODE != 3))</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>

Port Name	I/O	Description
utmi_l1_suspend_com_n	O	<p>Common L1 suspend. For device mode, this signal is the same as inverted utmi_l1_suspend_n[0]. For host mode, this signal is asserted (low) if any of the following conditions is met:</p> <ul style="list-style-type: none"> ■ Any of the host ports (either 3.0 or 2.0 ports) is not suspended. (In this case, for the USB 3.0 port, suspend is equivalent to P3; for the USB 2.0 port, it is L1/L2 Suspend). ■ Hardware LPM is enabled. <p>For host mode, this signal is high if all of the following conditions are met:</p> <ul style="list-style-type: none"> ■ All the USB 2.0 ports are in L1/L2 suspend (or in disconnected state if `DWC_USB3_SUSPEND_ON_DISCONNECT_EN is high) with at least one port in L1 suspend/sleep state ■ GUSB2PHYCFGn.EnblSlpM is 1 for all the ports ■ All the SS PHYs are in P3 ■ Hardware LPM is not enabled <p>Note:</p> <ul style="list-style-type: none"> ■ The polarity of this signal is different than the utmi_l1_suspend_n signal. ■ In host mode, for details the exceptions to the dependency SS port and USB 2.0 HW LPM, based GCTL.SOFITPSYNC and GFLADJ.GFLADJ_REF_CLK_LPM_SEL, see 'SOFITPSYNC' and 'GFLADJ_REFCLK_LPM_SEL'. <p>Exists: ((DWC_USB3_FSPHY_INTERFACE == 1 DWC_USB3_FSPHY_INTERFACE == 3 DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3) && (DWC_USB3_HOST_NUM_ROOT_PORTS > 1) && (DWC_USB3_MODE != 3))</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
utmi_tx_data[((16*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0]	O	<p>UTMI Data Output Bus. 8- or 16-bit data transmitted to the PHY. The low and high bytes are asserted valid by utmi_rxvalid and utmi_rxvalidh, respectively.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, utmi_clk[0], suspend_clk, ulpi_clk[0] or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
utmi_txvalidh[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>UTMI Transmit Data Valid, High Byte. Indicates that utmi_tx_data[15:8] contains valid data.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, utmi_clk[0], suspend_clk, ulpi_clk[0] or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_txvalid[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>UTMI Transmit Data Valid, Low Byte. Indicates that utmi_tx_data[7:0] contains valid data.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, utmi_clk[0], suspend_clk, ulpi_clk[0] or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_opmode[((2*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0]	O	<p>UTMI Operating Mode. The DWC_usb3 controller drives this signal to select the UTMI mode.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 2'b00: Normal operation ■ 2'b01: Non-driving ■ 2'b10: Disable bit stuffing and NRZI encoding ■ No other values are supported <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise utmi_clk[0], ulpi_clk[0] or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
utmi_suspend_n[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>UTMI Suspend. Places the PHY in Suspend mode, drawing minimal power. In this mode, the PHY shuts down all internal circuits that are not necessary for suspend/resume. The PHY also disables the clock. Whenever the USB 2.0 PHY is requested to exit from the suspended state (for example, when utmi_suspend_n is set back to 1'b1 requesting to turn the utmi clock), the DWC_usb3 controller assumes that the utmi/ulpi_clk is switched within 990 us.</p> <p>Exists: (DWC_USB3_FSPHY_INTERFACE == 1 DWC_USB3_FSPHY_INTERFACE == 3 DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, utmi_clk[0], ulpi_clk[0] or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
utmi_termselect[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>Termination Select. Selects HS or FS termination the PHY.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: HS termination enabled ■ 1'b1: FS termination enabled <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise utmi_clk[0], ulpi_clk[0] or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
utmi_xcvrselect[((2*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0]	O	<p>Transceiver Select. Selects a HS, FS, or LS transceiver the PHY.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 2'b00: HS transceiver enabled ■ 2'b01: FS transceiver enabled ■ 2'b10: LS transceiver enabled ■ 2'b11: Send a LS packet a FS bus or receive a LS packet. <p>If xcrvselect is 2'b11, the transceiver sends a preamble packet at FS before sending the LS packet. In receive mode, the transceiver waits to receive an LS packet with the LS transceiver enabled. The transceiver must send all data (both FS preamble packet and the LS data) with FS signaling (fast rise and fall times, and opposite polarity).</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise utmi_clk[0], ulpi_clk[0] or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_word_if[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>UTMI Data Bus Width and Clock Select. Indicates whether the PHY is operating in 8- or 16-bit mode.</p> <p>Note: This is not an actual UTMI+ Level 3 signal.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: 8-bit interface ■ 1'b1: 16-bit interface <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: (DWC_USB3_MODE == 2 && DWC_USB3_EN_USB2_ONLY != 1) ? (DWC_USB3_HSPHY_INTERFACE==3 ? "bus_clk_early,ulpi_clk[0],utmi_clk[0]" : "bus_clk_early,utmi_clk[0]") : (DWC_USB3_EN_USB2_ONLY == 1) ? "bus_clk_early,utmi_clk[0]" : "bus_clk_early"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
utmi_fsls_low_power[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>PHY Low-Power Clock Select. Selects either 480- or 48-MHz Low-Power mode for the PHY. Typically in FS and LS modes, the PHY can operate a 48-MHz clock to save power.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: 480-MHz Internal PLL clock ■ 1'b1: 48-MHz External clock <p>Clocks:</p> <ul style="list-style-type: none"> ■ Using a 48-MHz clock, the UTMI interface operates at 48 MHz in FS and LS modes. ■ Using a 480-MHz clock, the UTMI interface operates at either 60 MHz for 8-bit data mode, or 30 MHz for 16-bit data mode. <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_fslsserialmode[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>PHY Interface Mode Select. Indicates which PHY interface is used to transfer the FS and LS packets.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: FS and LS packets are transferred through the parallel interface ■ 1'b1: FS and LS packets are transferred through the serial interface. <p>This signal is tied low to select the parallel interface.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_clk[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	I	<p>UTMI Clock. Receives the 30-, 60-, or 48-MHz clock supplied by the High Speed UTMI+ PHY. You must enable this clock when DWC_USB3_HSPHY_INTERFACE is set to a value of 1 or 3.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>

Port Name	I/O	Description
utmi_txready[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	I	<p>UTMI Transmit Data Ready. Indicates that the PHY accepted the current transmit data and is ready for the next packet the utmi_tx_data bus.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, pipe3_rx_pclk[0], suspend_clk or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_rx_data[((16*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0]	I	<p>UTMI Receive Data. 8- or 16-bit data received from the PHY. The low and high bytes are asserted valid by the utmi_rxvalid and utmi_rxvalidh signals, respectively.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, pipe3_rx_pclk[0], suspend_clk or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_rxvalidh[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	I	<p>UTMI Receive Data Valid, High Byte. Indicates that utmi_rx_data[15:8] contains valid data.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, pipe3_rx_pclk[0], suspend_clk or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
utmi_rxvalid[(DWC_USB3_NUM_U2_PORTS-1):0]	I	<p>UTMI Receive Data Valid, Low Byte. Indicates that utmi_rx_data[7:0] contains valid data.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, pipe3_rx_pclk[0], suspend_clk or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_rxactive[(DWC_USB3_NUM_U2_PORTS-1):0]	I	<p>UTMI Receive Active. Indicates that the PHY has detected SYNC and is active. This signal is deasserted after a bit stuff error or when an EOP is detected.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, pipe3_rx_pclk[0], suspend_clk or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_rxerror[(DWC_USB3_NUM_U2_PORTS-1):0]	I	<p>UTMI Receive Error. Indicates that the PHY has detected a receive error.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, pipe3_rx_pclk[0], suspend_clk or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
utmi_linestate[((2*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0]	I	<p>Line State Indicator. Indicates the current state of the two USB data signals:</p> <ul style="list-style-type: none"> ■ D+ (utmi_line_state[0]). ■ D' (utmi_line_state[1]). <p>This signal is combinational when the clock is not available (in Suspend state), but otherwise is a registered signal.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 2'b00: SE0 ■ 2'b01: J ■ 2'b10: K ■ 2'b11: SE1 <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early or ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmi_hostdisconnect[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	I	<p>Peripheral Disconnect Indicator to Host. Indicates that the USB transceiver has detected a disconnect condition the cable.</p> <ul style="list-style-type: none"> ■ This signal is valid only when utmiotg_dppulldown and utmiotg_dmpulldown are sampled asserted. ■ When utmiotg_dppulldown and utmiotg_dmpulldown are not sampled asserted, then the behavior of utmi_hostdisconnect is undefined. <p>Connection:</p> <ul style="list-style-type: none"> ■ No peripheral connected: signal remains asserted. ■ Peripheral connected: signal de-asserted. <p>Exists: ((DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3) && DWC_USB3_MODE !=0)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
utmisrp_bvalid	I	<p>BVALID: Session valid for Peripheral. Indicates if the voltage Vbus is valid for peripheral and B-Device session. The comparator thresholds are:</p> <ul style="list-style-type: none"> ■ 1'b0: Vbus < 0.8 V ■ 1'b1: Vbus >= 4 V <p>This signal must be a filtered signal if DWC_USB3_EN_BUS_FILTERS is chosen as 0.</p> <p>This input must be driven in all peripheral (non-host) modes when DWC_USB3_HSPHY_INTERFACE is chosen as UTMI.</p> <p>In Device-only mode, this is used as power-present.</p> <p>Note: For DWC_usb3 version 1.72a and earlier, the utmisrp_bvalid input was not present.</p> <p>For complete usage, refer to "Integrating the Controller with the PHY" section in the User Guide.</p> <p>Exists: (DWC_USB3_MODE == 0 DWC_USB3_MODE == 2) && (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses utmi_clk[0] to utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, suspend_clk, pipe3_rx_pclk[0], ulpi_clk[0] to ulpi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmiotg_vbusvalid[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	I	<p>Vbus Valid. Indicates if the voltage Vbus is valid for Host and A-Device operation. The comparator thresholds are:</p> <ul style="list-style-type: none"> ■ 1'b0: Vbus < 4.4 V ■ 1'b1: Vbus > 4.75 V <p>This signal must be a filtered signal if DWC_USB3_EN_BUS_FILTERS is chosen as 0. It must be the same as pipe3_PowerPresent. This input must be driven in all non-peripheral (host) modes if DWC_USB3_HSPHY_INTERFACE is chosen as UTMI.</p> <p>In Host-only mode, this signal must be connected to logic high value of 1'b1.</p> <p>For complete usage, refer to "Integrating the Controller with the PHY" section in the User Guide.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.11 USB 2.0 UTMI+ Vendor Control Interface Signals

utmi_vstatus - - utmi_vcontrolload_n
- utmi_vcontrol

This interface connects the DWC_usb3 controller to UTMI+ PHY internal registers.

Table 7-11 USB 2.0 UTMI+ Vendor Control Interface Signals

Port Name	I/O	Description
utmi_vcontrolload_n[$(DWC_USB3_NUM_U2_ROOT_PORTS-1):0$]	O	<p>Vendor Control Load. Loads the vendor control register in the PHY. Values:</p> <ul style="list-style-type: none"> ■ 1'b0: Load vendor control register ■ 1'b1: No operation <p>Exists: (DWC_USB3_VENDOR_CTL_INTERFACE==1) Synchronous To: utmi_clk Registered: DWC_USB3_VENDOR_CTL_INTERFACE ? "No" : "Yes" Power Domain: Vcc Active State: Low</p>
utmi_vcontrol[$((4*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0$]	O	<p>Vendor Control. Vendor defined parallel output bus. Exists: (DWC_USB3_VENDOR_CTL_INTERFACE==1) Synchronous To: utmi_clk Registered: DWC_USB3_VENDOR_CTL_INTERFACE ? "No" : "Yes" Power Domain: Vcc Active State: High</p>
utmi_vstatus[$((8*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0$]	I	<p>Vendor Status. Vendor-defined parallel input bus. Exists: (DWC_USB3_VENDOR_CTL_INTERFACE==1) Synchronous To: DWC_USB3_VENDOR_CTL_INTERFACE ? "ram_clk_in" : "None" Registered: No Power Domain: Vcc Active State: High</p>

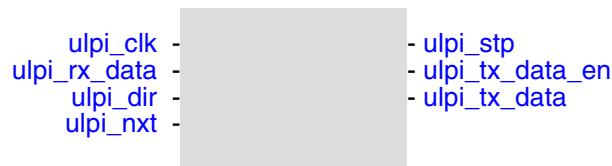
7.12 USB 2.0 UTMI+ OTG Interface Signals

- utmiotg_dppulldown
- utmiotg_dmpulldown

Table 7-12 USB 2.0 UTMI+ OTG Interface Signals

Port Name	I/O	Description
utmiotg_dppulldown	O	<p>D+ Pull-down Resistor Enable. Enables the 15 KW pull-down resistor on the D+ line.</p> <p>Exists: ((DWC_USB3_MODE == 2) && (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3 (DWC_USB3_FSPHY_INTERFACE != 0 && DWC_USB3_I2C_INTERFACE != 1)))</p> <p>Synchronous To: DWC_USB3_HSPHY_INTERFACE==1 ? "utmi_clk[0]" : "utmi_clk[0],ulpi_clk[0]"</p> <p>Registered: DWC_USB3_EN_PWROPT==2 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
utmiotg_dmpulldown	O	<p>D' Pull-down Resistor Enable. Enables the 15 KW pull-down resistor on the D' line.</p> <p>Exists: ((DWC_USB3_MODE == 2) && (DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3 (DWC_USB3_FSPHY_INTERFACE != 0 && DWC_USB3_I2C_INTERFACE != 1)))</p> <p>Synchronous To: DWC_USB3_HSPHY_INTERFACE==1 ? "utmi_clk[0]" : "utmi_clk[0],ulpi_clk[0]"</p> <p>Registered: DWC_USB3_EN_PWROPT==2 ? "No" : "Yes"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.13 USB 2.0 ULPI PHY Interface Signals



This interface connects the DWC_usb3 controller to any USB PHY with a ULPI interface that complies with UTMI+ L3 or UTMI.

Table 7-13 USB 2.0 ULPI PHY Interface Signals

Port Name	I/O	Description
ulpi_stp[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>Stop Output Control. The DWC_usb3 controller asserts this signal for one clock cycle to indicate the end of a USB transmit packet or a register write operation and, optionally, to stop packet reception. Assertion occurs after the last data byte is presented to the bus.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 2 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses ulpi_clk[0] to ulpi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early, suspend_clk, pipe3_tx_pclk[0] or utmi_clk[0] to utmi_clk[i], where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
ulpi_tx_data_en[((8*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0]	O	<p>Data Output Enable. ULPI data output enable is driven by the controller to the pad enables of drivers.</p> <p>Note: This signal is not defined in the ULPI specification. The specification says that DIR (ulpi_dir) can be used for the tri-state control of both Controller and PHY, but depending on the delay in the buffer enables, there could be a short duration of Controller and PHY driver clash. To avoid this clash, the controller has ulpi_tx_data_en output which is delayed by a clock to DIR de-assertion. If you use ulpi_tx_data_en, make sure that the ulpi_tx_data_en delay + output buffer delay in your library still meets the ULPI requirement of 6 ns setup delay (Tsd) for the PHY.</p> <p>ULPI Interface:</p> <ul style="list-style-type: none"> ■ Bit 0 connects to the pad enables of ulpi_tx_data[0] ■ Bit 1 connects to the pad enables of ulpi_tx_data[1] ■ Bit 2 connects to the pad enables of ulpi_tx_data[2] ■ Bit 3 connects to the pad enables of ulpi_tx_data[3] ■ Bit 4 connects to the pad enables of ulpi_tx_data[4] ■ Bit 5 connects to the pad enables of ulpi_tx_data[5] ■ Bit 6 connects to the pad enables of ulpi_tx_data[6] ■ Bit 7 connects to the pad enables of ulpi_tx_data[7] <p>Exists: ((DWC_USB3_HSPHY_INTERFACE == 2 DWC_USB3_HSPHY_INTERFACE == 3))</p> <p>Synchronous To: This port uses ulpi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise utmi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
ulpi_tx_data[((8*DWC_USB3_NUM_U2_ROOT_PORTS)-1):0]	O	<p>ULPI Data Output. ULPI data output bus driven by the DWC_usb3 controller.</p> <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 2 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses ulpi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise utmi_clk[i], where 0<=i<=14 depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
ulpi_clk[$(\text{DWC_USB3_NUM_U2_ROOT_PORTS}-1):0$]	I	<p>ULPI Clock. Receives the 60-MHz clock supplied by the High Speed ULPI PHY. The negative edge is also used in DDR mode. This clock must be enabled when <code>DWC_USB3_HSPHY_INTERFACE</code> is chosen to be 2 or 3.</p> <p>Exists: (<code>DWC_USB3_HSPHY_INTERFACE == 2</code> <code>DWC_USB3_HSPHY_INTERFACE == 3</code>)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
ulpi_rx_data[$((8*\text{DWC_USB3_NUM_U2_ROOT_PORTS})-1):0$]	I	<p>ULPI Data Input. ULPI data input to the controller driven by the PHY.</p> <p>Exists: (<code>DWC_USB3_HSPHY_INTERFACE == 2</code> <code>DWC_USB3_HSPHY_INTERFACE == 3</code>)</p> <p>Synchronous To: This port uses <code>ulpi_clk[i]</code> during USB 2.0 (HS/FS/LS) operational mode, otherwise <code>bus_clk_early</code> or <code>utmi_clk[i]</code>, where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
ulpi_dir[$(\text{DWC_USB3_NUM_U2_ROOT_PORTS}-1):0$]	I	<p>Data Bus Control. Controls the direction of the data bus.</p> <ul style="list-style-type: none"> ■ When the PHY has data to transfer to the DWC_usb3 controller, it drives <code>ulpi_dir</code> high to take ownership of the bus. ■ When the PHY has no data to transfer, it drives <code>ulpi_dir</code> low and monitors the bus. <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: DWC_usb3 is the driver. ■ 1'b1: ULPI PHY is the driver. <p>Exists: (<code>DWC_USB3_HSPHY_INTERFACE == 2</code> <code>DWC_USB3_HSPHY_INTERFACE == 3</code>)</p> <p>Synchronous To: This port uses <code>ulpi_clk[i]</code> during USB 2.0 (HS/FS/LS) operational mode, otherwise <code>bus_clk_early</code> or <code>utmi_clk[i]</code>, where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
ulpi_nxt[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	I	<p>Next Data Control.</p> <ul style="list-style-type: none"> ■ When the DWC_usb3 controller is sending data to the PHY, this signal indicates when the current byte is accepted by the PHY. The DWC_usb3 controller places the next byte on the data bus in the following clock cycle. ■ When the PHY is sending data to the DWC_usb3 controller, this signal indicates when a new byte is available for the DWC_usb3 controller. <p>Exists: (DWC_USB3_HSPHY_INTERFACE == 2 DWC_USB3_HSPHY_INTERFACE == 3)</p> <p>Synchronous To: This port uses ulpi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise utmi_clk[i], where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.14 HSIC Interface Signals



This interface connects the DWC_usb3 controller to any USB 2.0 PHY that complies with the High Speed Inter-Chip (HSIC) USB protocol.

For details on HSIC timing diagrams, see “[High Speed Inter-Chip \(HSIC\) Feature](#)” on page [535](#).

Table 7-14 HSIC Interface Signals

Port Name	I/O	Description
hsic_if_mode[(DWC_USB3_NUM_U2_RO OT_PORTS-1):0]	O	<p>HSIC Interface Mode Active. Indicates that the HSIC mode is active. This signal is set high when any of the following conditions is true:</p> <ul style="list-style-type: none"> ■ the if_select_hsic[n] signal is high and GUSB2PHYCFGn[26]=0 ■ the if_select_hsic[n] strap is low and GUSB2PHYCFGn[26]=1 <p>Exists: (DWC_USB3_ENABLE_HSIC==1) Synchronous To: utmi_clk[n],ulpi_clk[n] Registered: Yes Power Domain: Vcc Active State: High</p>
if_select_hsic[(DWC_USB3_NUM_U2_RO OT_PORTS-1):0]	I	<p>HSIC Interface Select. Selects the HSIC mode of operation. The HSIC mode for a given port is active when one of the following conditions is true:</p> <ul style="list-style-type: none"> ■ This strap is high and GUSB2PHYCFGn[26]=0 ■ This strap is low and GUSB2PHYCFGn[26]=1 <p>Each bit in this strap corresponds to each HS port. When selecting the HSIC feature, set the host side to HSIC mode first, then set the device mode side. If the device side is set to HSIC mode first, and if the host doesn't see a connection in HSIC mode, then you must de-select the device HSIC mode and select it again using the 'if_select_hsic' or register bit GUSB2PHYCFGn[26] to ensure that the device can connect to the host.</p> <p>Exists: (DWC_USB3_ENABLE_HSIC==1) Synchronous To: bus_clk Registered: N/A Power Domain: Vcc Active State: High</p>

7.15 LPM Interface Signals

- [utmi_sleep_n](#)
- [utmi_l1_suspend_n](#)

This interface controls the PHY Shallow and Deep Low-Power states in the LPM Sleep state.

Table 7-15 LPM Interface Signals

Port Name	I/O	Description
utmi_sleep_n[(DWC_USB3_NUM_U2_RO OT_PORTS-1):0]	O	<p>UTMI Sleep. The controller asserts this active-low signal to indicate that the controller is in the Sleep (L1) state, and the PHY can use this signal to enter Shallow Low-Power mode in the Sleep state. In Shallow Low-Power mode, the PHY can switch off some of its internal logic but still provide a PHY clock output by keeping the oscillator circuitry alive. Note that in the Host and DRD configurations, the controller assumes that when it asserts the utmi_sleep_n signal, the UTMI PHY shuts down some of its internal blocks but still provides the PHY clock. If the PHY turns off the clocks also, then it is recommended that you connect '1' to the utmi_sleep_n signal of the PHY and leave the controller's output unconnected. For the Synopsys PHY, in the Host mode operation, avoid assertion of this signal by setting the L1_SUSP_THRLD_EN_FOR_HOST=1'b1 and HIRD/BESL>=L1_SUSP_THRLD_FOR_HOST. When this signal is asserted low, the UTMI PHY transitions to Shallow Low-Power mode by powering down necessary blocks. When this signal is deasserted (high), the PHY returns to its normal operating state. The controller asserts this signal when it transitions to L1 (SLEEP) power state after a successful LPM transaction and is waiting for a minimum T(L1token retry time) of 50us. This signal is de-asserted when the power state exits the SLEEP (L1) state, due either to host-initiated resume or device-initiated remote wakeup.</p> <p>Note: In device mode of operation, if a PHY does not support LPM, the normal utmi_suspend_n and utmi_l1_suspend_n can be ANDed and connected to the PHY's SUSPENDM as long as the PHY's wake up time is less than the HIRD_Thres maximum value, and the utmi_sleep_n can be ignored. This signal is always present because LPM support is enabled by default. It goes to de-asserted state (driven high) when LPM Capability is not enabled in DCFG[22]. For details on the connections in the host mode, see 'OVRLD_L1_SUSP_COM' register field.</p> <p>Exists: ((DWC_USB3_FSPHY_INTERFACE == 1 DWC_USB3_FSPHY_INTERFACE == 3 DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3) && DWC_USB3_ENABLE_LPM==1)</p> <p>Synchronous To:</p>

Port Name	I/O	Description
utmi_sleep_n[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]...(cont.)	O.	<p>This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early or ulpi_clk[i], where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>
utmi_l1_suspend_n[(DWC_USB3_NUM_U2_ROOT_PORTS-1):0]	O	<p>UTMI Suspend in L1. This active low signal is asserted from the controller indicating that the controller is in Sleep (L1) state. In sleep state, the PHY can use this signal to enter deep low power mode. Deep low power mode is where the PHY is able to switch off its internal logic as well as the oscillator circuitry. The PHY clock to controller is switched off.</p> <p>When this signal is asserted low, the UTMI PHY transitions to Deep Low-Power mode by powering down its internal logic as well as the oscillator circuitry. If this signal is de-asserted (high), the PHY returns to its normal operating state.</p> <p>The controller asserts this signal when it transitions to L1 (SLEEP) power state after a successful LPM transaction and is waiting for a minimum TL1 token retry time of 50us. This signal is deasserted when the power state exits the SLEEP (L1) state, whether due to host-initiated resume or device-initiated remote wakeup.</p> <p>Note: In device mode of operation, if a PHY does not support LPM, the normal utmi_suspend_n and utmi_l1_suspend_n can be ANDed and connected to the PHY's SUSPENDM as long as the PHY's wake up time is less than the HIRD_Thres maximum value, and the utmi_sleep_n can be ignored. This signal is always present because LPM support is enabled by default. It goes to de-asserted state (driven high) when the LPM capability is not enabled in DCFG[22]. For details on the connections in the host mode, see 'OVRLD_L1_SUSP_COM' register field.</p> <p>Exists: ((DWC_USB3_FSPHY_INTERFACE == 1 DWC_USB3_FSPHY_INTERFACE == 3 DWC_USB3_HSPHY_INTERFACE == 1 DWC_USB3_HSPHY_INTERFACE == 3) && DWC_USB3_ENABLE_LPM==1)</p> <p>Synchronous To: This port uses utmi_clk[i] during USB 2.0 (HS/FS/LS) operational mode, otherwise bus_clk_early or ulpi_clk[i], where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>

7.16 Power Controller Interface Signals

pm_power_state_request pm_pmu_config_strap power_switch_control	- operational_mode - clk_gate_ctrl - current_power_state_u2pmu - pme_generation_u2pmu - connect_state_u2pmu - debug_u2pmu - current_power_state_u3pmu - pme_generation_u3pmu - connect_state_u3pmu - debug_u3pmu
---	---

This interface connects the DWC_usb3 controller to the Power Controller.

Table 7-16 Power Controller Interface Signals

Port Name	I/O	Description
operational_mode[1:0]	O	<p>Port capability direction</p> <p>This signal defines the mode of operation of the controller. It directly reflects the value programmed in the bits [13:12] of the GCTL register.</p> <p>Exists: (DWC_USB3_MODE == 2)</p> <p>Synchronous To: DWC_USB3_EN_PWR0PT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
clk_gate_ctrl[2:0]	O	<p>Clock gating control</p> <p>This signal indicates the clock gating controls used for gating RAM and bus clocks.</p> <p>Bit[0] is gating control for bus_clk, bit[1] and bit[2] is for ram_clk_in.</p> <p>This output must be used only for debug purposes.</p> <p>Exists: (DWC_USB3_EN_PWR0PT == 1 DWC_USB3_EN_PWR0PT == 2)</p> <p>Synchronous To: 0:0=bus_clk_early;1:2=ram_clk_in</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
current_power_state_u2pmu[1:0]	O	<p>The current power state of the controller. When equal to '3', the U2PMU is controlling the PHY, and when equal to '0', the controller is controlling the PHY. When both inputs from the PMUs are equal to '3', the controller is prepared to lose Vcc. When both inputs are equal to '0', software may access the CSRs.</p> <p>Exists: (DWC_USB3_EN_PWR0PT==2) Synchronous To: suspend_clk Registered: Yes Power Domain: Vaux Active State: High</p>
pme_generation_u2pmu	O	<p>Wakeup request from the 2.0 PMU which is a request to restore Vcc and put the controller into the D0 state. For details on when this signal is asserted, see the tables in the "PHY-Initiated Hibernation Exit Event for Host and Device Modes" section in the Programming Guide. This signal stays asserted until the PMU enters D0. You can use the rising edge of this signal to generate WAKE# and the PCI Express PME message.</p> <p>Exists: (DWC_USB3_EN_PWR0PT==2) Synchronous To: suspend_clk Registered: Yes Power Domain: Vaux Active State: High</p>
connect_state_u2pmu	O	<p>When '1', indicates the 2.0 PMU is maintaining at least one connection to the host or a device (the link is in L1, L2, or U3). When '0', indicates the PMU has no connection to the host or any device.</p> <p>Note: This signal is valid only when the 2.0 PMU is in D3 (current_power_state_u2pmu = 3).</p> <p>Exists: (DWC_USB3_EN_PWR0PT==2) Synchronous To: Asynchronous Registered: No Power Domain: Vaux Active State: N/A</p>

Port Name	I/O	Description
debug_u2pmu[((DWC_USB3_NUM_U2_ROOT_PORTS*54)+14)-1]:0]	O	<p>Debug output bus for 2.0 PMU Bit [DWC_USB3_NUM_U2_ROOT_PORTS*54] indicates whether hibernation is disabled. The same information is available to software which sets the GCTL.GblHibernationEn field, but it is replicated here for convenience.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ This signal is valid only when the current_power_state_u2pmu output is equal to '3'. ■ When hibernation is disabled, vcc_reset_n must not be asserted in D3, so the system can use these signals to determine whether to assert vcc_reset_n in D3 or not. <p>Exists: (DWC_USB3_EN_PWRROPT==2) Synchronous To: Asynchronous Registered: No Power Domain: Vaux Active State: N/A</p>
current_power_state_u3pmu[1:0]	O	<p>The current power state of the controller. When equal to '3', the U3PMU is controlling the PHY, and when equal to '0', the controller is controlling the PHY. When both inputs from the PMUs are equal to '3', the controller is prepared to lose Vcc. When both inputs are equal to '0', software may access the CSRs.</p> <p>Exists: (DWC_USB3_EN_PWRROPT==2 && DWC_USB3_EN_USB2_ONLY!=1) Synchronous To: suspend_clk Registered: Yes Power Domain: Vaux Active State: High</p>
pme_generation_u3pmu	O	<p>Wakeup request from the 3.0 PMU which is a request to restore Vcc and put the controller into the D0 state. For details on when this signal is asserted, see the tables in the "PHY-Initiated Hibernation Exit Event for Host and Device Modes" section in the Programming Guide. This signal stays asserted until the PMU enters D0. You can use the rising edge of this signal to generate WAKE# and the PCI Express PME message.</p> <p>Exists: (DWC_USB3_EN_PWRROPT==2 && DWC_USB3_EN_USB2_ONLY!=1) Synchronous To: suspend_clk Registered: Yes Power Domain: Vaux Active State: High</p>

Port Name	I/O	Description
connect_state_u3pmu	O	<p>When '1', indicates the 3.0 PMU is maintaining at least one connection to the host or a device (the link is in L1, L2, or U3). When '0', indicates the PMU has no connection to the host or any device.</p> <p>Note: This signal is valid only when the 3.0 PMU is in D3 (current_power_state_u3pmu = 3).</p> <p>Exists: (DWC_USB3_EN_PWR0PT==2 && DWC_USB3_EN_USB2_ONLY!=1)</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vaux</p> <p>Active State: N/A</p>
debug_u3pmu[((DWC_USB3_NUM_U3_ROOT_PORTS*25)+14)-1]:0]	O	<p>Debug output bus for 3.0 PMU</p> <p>Bit [0] indicates if hibernation is disabled. The same information is available to software which sets the GCTL.GblHibernationEn field, but it is replicated here for convenience.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ This signal is valid only when the current_power_state_u3pmu output is equal to '3'. ■ When hibernation is disabled, vcc_reset_n must not be asserted in D3, so the system can use these signals to determine whether to assert vcc_reset_n in D3 or not. <p>Exists: (DWC_USB3_EN_PWR0PT==2 && DWC_USB3_EN_USB2_ONLY!=1)</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vaux</p> <p>Active State: N/A</p>
pm_power_state_request[1:0]	I	<p>Power state request to the controller. This port defines the PCI power management state requested by the software. The power controller sets this to '3' and waits for current_power_state_pmu from both PMUs to reflect '3' before removing Vcc and clocks.</p> <p>After restoring Vcc, the power controller sets this to '0'. Only bit [0] of this 2-bit bus is synchronized and used. Therefore, no values other than 0 and 3 are allowed.</p> <p>If this signal is connected to the PMCSR, the D1 and D2 states must be mapped to '0'.</p> <p>Valid states:</p> <ul style="list-style-type: none"> ■ 00: D0 ■ 11: D3 <p>Exists: (DWC_USB3_EN_PWR0PT == 2)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
pm_pmu_config_strap[(DWC_USB3_PMU_CFG_W-1):0]	I	<p>This is an array of fixed values that indicates which (if any) PHY signal may be used to detect connect/disconnect when the controller is acting as a device. In a host-only configuration, the entire bus can be set to '0'. [0]: Vbusvalid_valid. When '1', indicates that the phy_utmiotg_vbusvalid OR phy_utmisrp_bvalid signal can be used to detect connect/disconnect.[14:1]: Reserved. Tie to '0' [15]: PowerPresent_valid. When '1', indicates the phy_pipe3_PowerPresent signal can be used to detect connect/disconnect. [29:16]: Reserved. Tie to '0'. In an ULPI configuration where the ULPI interrupt reports the connect/disconnect, it is safe to set both Vbusvalid_valid and PowerPresent_valid to '0'. It is not legal to set both Vbusvalid_valid and PowerPresent_valid to '1'. Note: For an HSIC device configuration, tie pm_pmu_config_strap[0] and pm_pmu_config_strap[15] inputs to '0'. Because HSIC devices are permanently connected, the PMUs need not detect connect and disconnect for these devices. Because the HSIC selection can be based on strap or CSR register, it is good to have a mux control to select the right value for this strap instead of hard coding.</p> <p>Exists: (DWC_USB3_EN_PWROPT==2) Synchronous To: N/A Registered: No Power Domain: Vaux Active State: High</p>
power_switch_control	I	<p>Indicates that the power controller has turned off the power to the DWC_usb3 controller. This will enable isolation.</p> <p>Exists: (DWC_USB3_EN_PWROPT==2) Synchronous To: suspend_clk Registered: No Power Domain: Vcc Active State: DWC_USB3_PWR_SWITCH_POLARITY=1? High : Low</p>

7.17 Host Interface Signals

host_legacy_irq12_in	-	- host_legacy_irq12_out
host_legacy_irq1_in	-	- host_legacy_irq1_out
hub_port_overcurrent	-	- host_legacy_emulation_interrupt
hub_port_perm_attach	-	- host_current_belt
host_num_u2_port	-	- host_system_err
host_num_u3_port	-	- hub_vbus_ctrl
host_u2_port_disable	-	- host_legacy_smi_interrupt
host_u3_port_disable	-	
host_port_power_control_present	-	
host_msi_enable	-	
host_legacy_smi_pci_cmd_reg_wr	-	
host_legacy_smi_bar_wr	-	
xhc_bme	-	

This interface is present when the DWC_usb3 controller is configured as a Host.

Table 7-17 Host Interface Signals

Port Name	I/O	Description
host_legacy_irq12_out	O	<p>External Keyboard IRQ out. Exists: ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_LEGACY_SUPT==1) Synchronous To: Asynchronous Registered: No Power Domain: Vcc Active State: bus_clk</p>
host_legacy_irq1_out	O	<p>External Mice IRQ out. Exists: ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_LEGACY_SUPT==1) Synchronous To: bus_clk Registered: No Power Domain: Vcc Active State: High</p>
host_legacy_emulation_interrupt	O	<p>Emulation interrupt out. Exists: ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_LEGACY_SUPT==1) Synchronous To: bus_clk Registered: No Power Domain: Vcc Active State: High</p>

Port Name	I/O	Description
host_current_belt[11:0]	O	<p>Current BELT Value. This signal indicates the minimum value of all received BELT values and the BELT that is set by the Set LTV command. This signal is valid only in Host mode.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>
host_system_err	O	<p>Host System Error. This signal indicates that a Host System Error has occurred as reflected in the USBSTS.HSE field. This signal is asserted only if the USBCMD.HSEE field is set to '1'. It can occur when the host controller encounters an 'Error' response in the AHB, the AXI, or the Native Master Bus. When the USBSTS.HSE field is cleared by software, this signal is deasserted unless the master continues to assert its bus error output. The typical software response to an HSE is to reset the controller. For more details, refer to section 4.10.2.6 of the xHCI 1.0 specification.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2)</p> <p>Synchronous To: ram_clk_in</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_vbus_ctrl[(DWC_USB3_HOST_NUM_ROOT_PORTS_PIN-1):0]	O	<p>Port Power control for each downstream port.</p> <ul style="list-style-type: none"> ■ 0: VBUS OFF ■ 1: VBUS ON <p>The lower bits are for USB 2.0 ports and the higher bits are for USB 3.0 SS ports:</p> <pre>hub_vbus_ctrl[N-1:0] = { hub_vbus_ctrl['DWC_USB3_NUM_U3_ROOT_PORTS-1:0], hub_vbus_ctrl['DWC_USB3_NUM_U2_ROOT_PORTS-1:0]}</pre> <p>Note: In USB 2.0-only mode, only the bits corresponding to USB 2.0 ports are present.</p> <p><i>For hub configurations:</i></p> <p>Port Power control for each Downstream port. hub_vbus_ctrl[0] (upstream port's bit) must be unused.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2 DWC_USB3_MODE == 3)</p> <p>Synchronous To: DWC_USB3_MODE==3 ? "pipe3_rx_pclk[0],suspend_clk" : "N/A"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
host_legacy_smi_interrupt	O	<p>SMI interrupt: active high. The PCIe interface need to pass this SMI interrupt output. No connect, if you do not need Legacy support</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2)</p> <p>Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early"</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
host_legacy_irq12_in	I	<p>Asynchronous external Keyboard IRQ in.</p> <p>Exists: ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_LEGACY_SUPT==1)</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
host_legacy_irq1_in	I	<p>Asynchronous external Mice IRQ in.</p> <p>Exists: ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && DWC_USB3_LEGACY_SUPT==1)</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_port_overcurrent[(DWC_USB3_HC_MAXPORTS_PIN-1):0]	I	<p>This is the per port Overcurrent indication of the root-hub ports:</p> <ul style="list-style-type: none"> ■ 0: No Overcurrent ■ 1: Overcurrent <p>The lower bits are for USB 2.0 ports and the higher bits are for USB 3.0 SS ports:</p> <pre>hub_port_overcurrent[N-1:0] = { hub_port_overcurrent[DWC_USB3_NUM_U3_ROOT_PORTS-1:0], hub_port_overcurrent[DWC_USB3_NUM_U2_ROOT_PORTS-1:0]}</pre> <p>The minimum required overcurrent duration is 3 suspend clock periods.</p> <p>Note: In USB 2.0-only mode, only the bits corresponding to USB 2.0 ports are present. <i>For hub configurations:</i> The hub_port_overcurrent[0] (upstream port's bit) bit must be tied to '0'. In hub level overcurrent mode (hub_overcurrent_ctrl=1), any overcurrent must be tied to all overcurrent bits except the bit 0.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2 DWC_USB3_MODE == 3)</p> <p>Synchronous To: Asynchronous</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hub_port_perm_attach[(DWC_USB3_HC_MAXPORTS_PIN-1):0]	I	<p>Indicates if the device attached to a downstream port is permanently attached or not.</p> <ul style="list-style-type: none"> ■ 0: Not permanently attached ■ 1: Permanently attached <p>The lower bits are for USB 2.0 ports and the higher bits are for USB 3.0 SS ports:</p> <pre>hub_port_perm_attach[N-1:0] = { hub_port_perm_attach['DWC_USB3_NUM_U3_ROOT_PORTS-1:0], [hub_port_perm_attach['DWC_USB3_NUM_U2_ROOT_PORTS-1:0]}</pre> <p><i>For hub configurations:</i></p> <p>This signal indicates if the device attached to a downstream port is permanently attached or not. The hub_port_perm_attach[0] (upstream port's bit) must be tied to '0'.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2 DWC_USB3_MODE == 3)</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
host_num_u2_port[3:0]	I	<p>Number of USB 2.0 Ports.</p> <p>This signal overrides the 'DWC_USB3_NUM_U2_ROOT_PORTS synthesis configuration parameter.</p> <p>This allows you, for example, to develop a chip with a 4-Port USB 2.0 host and package it as only two USB 2.0 ports in a board. The upper two ports are not enabled. The 'Number of Ports' field of the HCSPARAMS1 register is controlled by this port. The 'Number of Ports' indicates 'host_num_u3_ports + host_num_u2_port' value.</p> <p>If you do not require the override feature, assign this port as follows:</p> <pre>assign host_num_u2_ports[3:0] = 'DWC_USB3_NUM_U2_ROOT_PORTS;</pre> <p>Note:</p> <ul style="list-style-type: none"> ■ The valid value of host_num_u2_port is from 1 to `DWC_USB3_NUM_U2_ROOT_PORTS. Make sure that this signal is not set to 0. It must be set to have at least one USB 2.0 port. ■ If the controller is operated as a Device (PrtCapDir = 2'b10), then only Port 0 (such as, one u2 port and one u3 port) is valid. Internal to the controller, the host_num_u2_port/host_num_u3_port are forced to 1 when operating in Device mode. <p>Exists: DWC_USB3_MODE == 1 DWC_USB3_MODE == 2</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
host_num_u3_port[3:0]	I	<p>Number of USB 3.0 SS Ports. This signal overrides the 'DWC_USB3_NUM_U3_ROOT_PORTS' synthesis configuration parameter. This allows you, for example, to develop a chip with a 4-Port USB 3.0 SS host and package it as only two USB 3.0 SS ports in a board. The upper two ports are not enabled. The 'Number of Ports' field of the HCSPARAMS1 register is controlled by this port. The 'Number of Ports' indicates 'host_num_u3_ports + host_num_u2_port' value. If you do not require the override feature, assign this port as follows: assign host_num_u3_port[3:0] = 'DWC_USB3_NUM_U3_ROOT_PORTS; Note: <ul style="list-style-type: none"> ■ The valid value of host_num_u3_port is from 1 to `DWC_USB3_NUM_U3_ROOT_PORTS. Make sure that this signal is not set to 0. It must be set to have at least one USB3.0 SS port. ■ If the controller is operated as a Device (PrtCapDir = 2'b10), then only Port0 (for instance, one u2 port and one u3 port) is valid. Internal to the controller, the host_num_u2_port/ host_num_u3_port will be forced to 1 when operating in Device mode. ■ This signal is not present in USB 2.0-only mode. Exists: ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && (DWC_USB3_EN_USB2_ONLY == 0)) Synchronous To: N/A Registered: No Power Domain: Vcc Active State: High </p>
host_u2_port_disable[(DWC_USB3_HOS_T_NUM_U2_ROOT_PORTS-1):0]	I	<p>USB 2.0 Port Disable control.</p> <ul style="list-style-type: none"> ■ 0: Port Enabled ■ 1: Port Disabled When '1', this signal stops reporting connect/disconnect events the port and keeps the port in disabled state. <p>This could be used for security reasons where hardware can disable a port irrespective of whether xHCI driver enables a port or not. The 'Number of Ports' field of the HCSPARAMS1 register is not affected by this signal. This signal should be either static (should not change during operation), or change only once from 0 to 1 during operation and stay at 1 after that. Note: Port-0 (the first U2 port) must not be disabled if any other U2 port is operational. Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) Synchronous To: N/A Registered: No Power Domain: Vcc Active State: High </p>

Port Name	I/O	Description
host_u3_port_disable[(DWC_USB3_HOST_NUM_U3_ROOT_PORTS-1):0]	I	<p>USB 3.0 SS Port Disable control.</p> <ul style="list-style-type: none"> ■ 0: Port Enabled ■ 1: Port Disabled <p>This signal, when '1', stops reporting connect/disconnect events the port and keeps the port in disabled state. This could be used for security reasons where hardware can disable a port irrespective of whether xHCI driver enables a port or not.</p> <p>The 'Number of Ports' field of the HCSPARAMS1 register is not affected by this signal.</p> <p>This signal should be either static (should not change during operation), or change only once from 0 to 1 during operation and stay at 1 after that.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ This signal is not present in USB 2.0-only mode. ■ The SuperSpeed Port0 (first SS port) must not be disabled if the other SuperSpeed ports are operational. <p>Exists: ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) && (DWC_USB3_EN_USB2_ONLY == 0))</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
host_port_power_control_present	I	<p>This port defines the bit [3] of Capability Parameters (HCCPARAMS). Change the PPC value through the pin Port Power Control (PPC). This indicates whether the host controller implementation includes port power control.</p> <ul style="list-style-type: none"> ■ 0: Indicates that the port does not have port power switches. ■ 1: Indicates that the port has port power switches. <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2)</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
host_msi_enable	I	<p>This enables the pulse type interrupt signal (one bus clock cycle) 'interrupt' port instead of level-sensitive interrupt. When interfacing to PCIe, this allows you to easily map 'interrupt' to MSI in the PCIe controller.</p> <p>MSI can only be enabled in the Host mode. There is no MSI support in the Device mode yet. So if the controller is configured in the DRD mode, it can only use wire interrupt which is a level signal.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2)</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
host_legacy_smi_pci_cmd_reg_wr	I	<p>PCI command register write: one clock pulse. The PCIe interface needs to generate one clock pulse during PCIe command register write. Tie this low if you are not using Legacy support.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) Synchronous To: DWC_USB3_EN_PWR0PT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early" Registered: No Power Domain: Vcc Active State: High</p>
host_legacy_smi_bar_wr	I	<p>PCI Base Address register (BAR) write: one clock pulse. The PCIe interface need to generate one clock pulse during PCIe Base Address register (BAR) write. Tie this low, if you are not using legacy support.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) Synchronous To: DWC_USB3_EN_PWR0PT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early" Registered: No Power Domain: Vcc Active State: High</p>
xhc_bme	I	<p>This signal is used to disable the bus mastering capability of the xHC. In a PCI system, it comes from the Bus Master Enable (BME) bit of the Device Control Register in the PCI Configuration register space.</p> <ul style="list-style-type: none"> ■ 1'b0: Bus mastering capability is disabled. The host controller cannot use the bus master interface. ■ 1'b1: Bus mastering capability is enabled. <p>Note: In Host mode, <ul style="list-style-type: none"> ■ For a non-PCI system, the xhc_bme is always tied to 1'b1 for the controller master to work. ■ For a PCI system, connect the BME register bit of the PCI to this xhc_bme input. In Device mode, the xhc_bme can be any value, but it is recommended to tie this signal to 1.</p> <p>Exists: (DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) Synchronous To: ram_clk_in Registered: No Power Domain: Vcc Active State: High</p>

7.18 Hub Interface Signals

hub_sf_use_flash	-	- hub_sf_ce_n
hub_sf_din	-	- hub_sf_clk
hub_rom_data	-	- hub_sf_reset_n
hub_rom_data_rdy	-	- hub_rom_ce_n
vci_rx_space_avail	-	- hub_rom_addr
vci_tx_pkt_status	-	- hub_usb2_desc_wr_n
vci_tx_pkt_count	-	- hub_usb2_desc_addr
vci_tx_data	-	- hub_usb2_desc_wdata
vci_stall	-	- vci_rx_cmd_vld
hub_busPowered	-	- vci_rx_cmd_param
hub_local_power_lost	-	- vci_rx_data_push
hub_overcurrent_ctl	-	- vci_rx_data
hub_desc_override_params	-	- vci_rx_status_vld
hub_desc_override_ctrl	-	- vci_rx_status
hub_port_dev_speed	-	- vci_tx_ack_vld
hub_port_unused	-	- vci_tx_ack_status
hub_scaledown_mode	-	- vci_tx_data_pop
hub_pwrdnscl_whubdel	-	- vci_tx_data_pop_last
hub_enable_scrambling	-	- vci_hub_us_reset
		- vci_hub_configured
		- vci_ctrl_xfer_active
		- hub_link_states
		- hub_dsport_state

This interface is present when the DWC_usb3 controller is configured as a Hub.

For description of the following signals, refer to “[Host Interface Signals](#)” on page [395](#):

- hub_port_overcurrent[N-1:0]
- hub_port_perm_attach[N-1:0]
- hub_vbus_ctrl[N-1:0]

The vendor command interface (VCI) has the following sets of signals:

- Receive Command interface signals (vci_rx_cmd*): These signals indicate the data packets (DPH) that are received by the control endpoint.
- Receive Status Interface signals (vci_rx_sts*): These signals indicate the status of the data packet payload (DPP) received for the control endpoint.
- Receive Data Interface (vci_rx_data*): These signals indicate the data payload received by the control endpoint.
- Transmit ACK Interface (vci_tx_sts*): These signals indicate the ACK TP received for the control IN endpoint.
- Transmit Data Interface (vci_tx_data*): These signals are used to move transmit data for the Control Read Data Phase of vendor commands.
- Stall control interface (vci_stall): These signals indicate the external vendor command interface to STALL a non-supported vendor command from the host.
- Hub internal state interface (vci_hub*, vci_ctrl*): These signals indicate the internal state of the hub controller related to the vendor command interface.

Table 7-18 Hub Interface Signals

Port Name	I/O	Description
hub_sf_ce_n	O	<p>Hub Serial Flash Chip Enable.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_EN_HUB_SFLASH_SUPT == 1)</p> <p>Synchronous To: ram_clk_gated</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>
hub_sf_clk	O	<p>Hub Serial Flash Clock.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_EN_HUB_SFLASH_SUPT == 1)</p> <p>Synchronous To: ram_clk_gated</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_sf_reset_n	O	<p>Hub Serial Flash Reset.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_EN_HUB_SFLASH_SUPT == 1)</p> <p>Synchronous To: ram_clk_gated</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>
hub_rom_ce_n	O	<p>Hub ROM Chip Enable.</p> <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: pipe3_rx_pclk[0],suspend_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_rom_addr[(DWC_USB3_ROM_A-1):0]	O	<p>Hub ROM Address</p> <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: pipe3_rx_pclk[0],suspend_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_usb2_desc_wr_n	O	<p>Hub USB2 Descriptor Write Enable: active low.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_DESC_SS_START_DEPTH > 0)</p> <p>Synchronous To: ram_clk_gated</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: Low</p>

Port Name	I/O	Description
hub_usb2_desc_addr[(DWC_USB3_ROM_A-1):0]	O	<p>Hub USB2.0 Descriptor Address.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_DESC_SS_START_DEPTH > 0)</p> <p>Synchronous To: ram_clk_gated</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_usb2_desc_wdata[(DWC_USB3_MDWIDTH-2):0]	O	<p>Hub USB 2.0 Descriptor Data.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_DESC_SS_START_DEPTH > 0)</p> <p>Synchronous To: ram_clk_gated</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_rx_cmd_vld	O	<p>This single-pulse signal indicates the reception of a data packet header (DPH) from the host.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_rx_cmd_param[29:0]	O	<p>Valid with the vci_rx_cmd_vld.</p> <p>This signal specifies the parameters corresponding to the received DPH.</p> <ul style="list-style-type: none"> ■ [29:25]: Sequence Number ■ [24]: Deferred Packet ■ [23]: Reserved ■ [22]: NRDY send to host for this DPH ■ [21]: Packet Pending ■ [20]: SETUP packet ■ [19:16]: Endpoint Number ■ [15:0]: Reserved <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
vci_rx_data_push	O	<p>The Hub asserts this signal to indicate that the data received during the SETUP phase of the control transfer and also during the data phase of a 3-stage control write transfer. The corresponding data is available through the output vci_rx_data. This is HIGH during the SETUP phase to push the SETUP data, and also during the data phase of a control wrtite transfer.</p> <p>Note that this signal is valid even for non-Vendor control commands. During the Control write data phase, the application uses this push signal only after decoding that the SETUP packet was a Vendor command and ignores the following push otherwise. If the vci_rx_cmd_vld/vci_rx_cmd_param indicate a SETUP, then the following vci_rx_data_push is valid. The application uses vci_ctrl_xfer_active as a qualifier for the vci_rx_data_push during the data phase, but not during the SETUP phase.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1) Synchronous To: mac3_clk Registered: Yes Power Domain: Vcc Active State: High</p>
vci_rx_data[31:0]	O	<p>This signal is data corresponding to the SETUP packet or data stage of a control write transfer. Valid with the vci_rx_data_push signal.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1) Synchronous To: mac3_clk Registered: Yes Power Domain: Vcc Active State: High</p>
vci_rx_status_vld	O	<p>This is a valid signal indicating the validity of the data packet received during the SETUP or data stage of the control OUT transfer. This signal is asserted for one clock after all the data been pushed. The status itself is available the output vcs_rx_status[15:0]. This signal is valid for non-vendor commands also. For the SETUP phase, application uses this signal. For data phase, the application optionally qualifies this with vci_ctrl_xfer_active, or ignores it for non-vendor commands.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1) Synchronous To: mac3_clk Registered: Yes Power Domain: Vcc Active State: High</p>

Port Name	I/O	Description
vci_rx_status[21:0]	O	<p>Indicates the status of the SETUP packet or the Control write data packet reception. This is valid with the output vci_rx_status_vld.</p> <p>[21:14]: Reserved</p> <p>[13:3]: Packet Byte Count. Indicates the number of data bytes pushed in for the packet. '0' indicates no data was pushed because space is not available or the data length is zero.</p> <p>[2:0]: Status</p> <ul style="list-style-type: none"> ■ 0: OK; ■ 1: DPP Error (both On CRC and timeout: DPH but no DPP); ■ 2: BABBLE; ■ Others: Reserved. <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_tx_ack_vld	O	<p>This signal is the qualifier for the vci_tx_ack_status output. Asserted for one clock pulse. This indicates that the controller has received a TP corresponding to the Data/Status of the Vendor control transfer. This signal is active for non-vendor commands also. The application optionally qualifies this signal with the vci_ctrl_xfer_active or ignores this signal for non-vendor commands.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
vci_tx_ack_status[32:0]	O	<p>This signal indicates the details of the ACK TP received from the host corresponding to the vci_tx_ack_vld.</p> <p>[31:29]: SubType</p> <ul style="list-style-type: none"> ■ 1: ACK; ■ 2: Status; ■ Others: Reserved <p>[28:24]: Sequence Number</p> <p>[23]: Reserved</p> <p>[22]: Packet Pending</p> <p>[21]: ACK Packet. '0' indicates requesting an ACK TP. '1' indicates completing an ACK TP; this indicates that the host has successfully received a packet.</p> <p>[20]: Retry. Indicates a host retry because the previous packet was not received. On seeing retry, the application rewinds the packet.</p> <p>[19:16]: Endpoint Number</p> <p>[15:0]: Reserved</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_tx_data_pop	O	<p>Asserted by the hub during the data phase of a control transfer. This signal is used to transfer data from the application to the USB as part of the Vendor command control in the data phase. The application uses this only if the SETUP packet has been decoded to be a vendor command and ignore this otherwise. The application optionally qualifies this signal with the vci_ctrl_xfer_active output signal from the controller.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_tx_data_pop_last	O	<p>Indicates a hub vendor control Tx packet's last data pop.</p> <p>Active with the last pop for a packet. The applicable must sample this signal with vci_tx_data_pop.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
vci_hub_us_reset	O	<p>This is a pulse signal valid for a single clock cycle, indicating that the hub upstream has received a USB reset (Hot/warm). This signal is also asserted when the hub upstream link transitions from a Receiver detect state to Polling state.</p> <p>Exists: (DWC_USB3_MODE == 3 & DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_hub_configured	O	<p>This is a active HIGH signal indicating that the hub is current in a configured state.</p> <p>Exists: (DWC_USB3_MODE == 3 & DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_ctrl_xfer_active	O	<p>This is active HIGH signal indicating that a vendor control access is active. This signal will go HIGH when the hub receives a vendor control SETUP packet and remains HIGH until the hub receives a non-vendor control SETUP packet.</p> <p>Exists: (DWC_USB3_MODE == 3 & DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_link_states[((4*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	O	<p>This signal represent the link states and is only for debug purpose. There are 4 bits representing each link state of each port, including the hub upstream port.</p> <ul style="list-style-type: none"> ■ Bits [3:0] represent the upstream link state ■ Bits [7:4] represent the Downstream port-1 link state ■ Bits [11:8] represent the Downstream port-2 link state, and so on. <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>

Port Name	I/O	Description
hub_dsport_state[((4*DWC_USB3_NUM_U3_ROOT_PORTS)-1):0]	O	<p>This signal represents the hub downstream port state machine (4 bits per port). Bits [3:0] are for the upstream port, and can be ignored. The decoding of the hub downstream port state machine is as follows:</p> <ul style="list-style-type: none"> ■ 0: PRT_POWEREDOFF ■ 1: PRT_DISCONNECTED ■ 2: PRT_DISABLED ■ 3: PRT_POLLING ■ 4: PRT_RESET ■ 5: PRT_ENABLED ■ 6: PRT_ERROR ■ 7: PRT_LPBK ■ 8: PRT_COMPLIANCE ■ 9: PRT_POWEREDOFF_DET ■ 10: PRT_POWEREDOFF_RST <p>Exists: DWC_USB3_MODE == 3 Synchronous To: N/A Registered: N/A Power Domain: Vcc Active State: N/A</p>
hub_sf_use_flash	I	<p>Use Serial Flash.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_EN_HUB_SFLASH_SUPT == 1)</p> <p>Synchronous To: None Registered: No Power Domain: Vcc Active State: High</p>
hub_sf_din	I	<p>Serial Flash Data In.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_EN_HUB_SFLASH_SUPT == 1)</p> <p>Synchronous To: ram_clk_gated Registered: No Power Domain: Vcc Active State: High</p>
hub_rom_data[(DWC_USB3_MDWIDTH-2):0]	I	<p>Hub ROM Read Data.</p> <p>Exists: DWC_USB3_MODE == 3 Synchronous To: pipe3_rx_pclk[0],suspend_clk Registered: No Power Domain: Vcc Active State: High</p>

Port Name	I/O	Description
hub_rom_data_rdy	I	<p>Hub ROM Read Data Ready.</p> <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: pipe3_rx_pclk[0],suspend_clk</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_rx_space_avail[13:0]	I	<p>Space available in terms of 32-bit DWORDS. The VCI application asserts this signal, indicating the availability of space for holding Control Write Data stage packets. SETUP packets are always accepted. The VCI application must make sure that it can always accept the SETUP packet. The Hub controller uses this signal for flow control. The Hub controller samples this signal when it receives DPH for the control write transfer data stage. The VCI application updates the space available after receiving the SETUP packet in the case of the first packet of the data stage or at the end of a packet reception for the following packets in the data stage. When the controller samples the signal, it initiates flow control if the space_avail is less than the size of the received packet. The Hub controller sends ERDY when the space available goes beyond the packet size and if the endpoint is in flow control.</p> <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: MISSING-FIELD-ERROR</p>
vci_tx_pkt_status[12:0]	I	<p>Indicates Tx packet size and last packet flag for the next transmitted packet.</p> <ul style="list-style-type: none"> ■ [10:0]: Size. Indicates the packet size in Bytes. ■ [11]: Last Packet. Last packet of the current transfer. ■ [12]: Data/Status stage. 0: Data Stage. 1: Status Stage <p>Exists: (DWC_USB3_MODE == 3 && DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
vci_tx_pkt_count[1:0]	I	<p>Indicates the number of packets available for transfer. The Hub controller uses this information to insert flow control during the status stage of any vendor control transfer or during the data stage of a control read vendor control transfer.</p> <p>Exists: (DWC_USB3_MODE == 3 & DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_tx_data[31:0]	I	<p>32-bit data valid used for the control in the data phase of the vendor command. The next 32-bit data is presented with the vci_tx_data_pop signal.</p> <p>Exists: (DWC_USB3_MODE == 3 & DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
vci_stall[1:0]	I	<p>The application uses this signal to STALL a vendor command control transfer. Bit[0] is used for stalling Control IN endpoint0 and Bit[1] is used for stalling control OUT endpoint 0. After seeing STALL, the application makes sure that it is cleared with the following SETUP packet.</p> <p>Exists: (DWC_USB3_MODE == 3 & DWC_USB3_HUB_ENABLE_VCI == 1)</p> <p>Synchronous To: mac3_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_busPowered	I	<p>Indicates if the hub is local or bus powered.</p> <ul style="list-style-type: none"> ■ 0: Self Powered ■ 1: Bus Powered. <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: pipe3_rx_pclk[0],suspend_clk</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hub_local_power_lost	I	<p>Indicates if the Local Power is good or lost.</p> <ul style="list-style-type: none"> ■ 0: Local Power good ■ 1: Local Power lost <p>Exists: DWC_USB3_MODE == 3 Synchronous To: Asynchronous Registered: No Power Domain: Vcc Active State: High</p>
hub_overcurrent_ctl	I	<p>This input specifies the Overcurrent protection mode to be used in the hub.</p> <ul style="list-style-type: none"> ■ 0: Individual port Overcurrent Control ■ 1: Global overcurrent protection <p>Exists: DWC_USB3_MODE == 3 Synchronous To: pipe3_rx_pclk[0],suspend_clk Registered: No Power Domain: Vcc Active State: High</p>

Port Name	I/O	Description
hub_desc_override_params[(DWC_USB3__HUB_OVERRIDE_PARAMS_WIDTH-1):0]	I	<p>This defines the descriptor override values.</p> <p>[127:0]: Container ID (GUID); Specifies the value to be given in the ContainerID field in the container ID Descriptor.</p> <p>[143:128]: Vendor ID; Specifies the value to be given in the idVendor field of Device Descriptor</p> <p>[159:144]: Product ID. Specifies the value to be given in the idProduct field of Device Descriptor.</p> <p>[160]:</p> <ul style="list-style-type: none"> ■ 1'b1: Individual Power Switch ■ 1'b0: Gang Power Switch <p>[167:161]: Size of the SerialId String Descriptor in Bytes. Value of 0 indicates that this value should be ignored and the length must then be taken as specified in the ROM. A non-zero value will force the SerialNumber field in the Device Descriptor to indicate String Number 1.</p> <p>[679:168]: Override values for SerialId string descriptor.</p> <p>[686:680]: Size of the MfgId String descriptor in Bytes. Value of 0 indicates that this value should be ignored and the length should then be taken as specified in the ROM. A non-zero value will force the Manufacturer field in the Device Descriptor to indicate String Number 3.</p> <p>[1198:687]: Override values for MfgId String Descriptor.</p> <p>[1205:1199]: Size of the ProductId String descriptor. Value of 0 indicates that this value should be ignored and the length should then be taken as specified in the ROM. A non-zero value will force the Product field in the Device Descriptor to indicate String Number 2.</p> <p>[1717:1206]: Override values for ProductId String Descriptor.</p> <p>[1733:1718]: Override values for LangId for String Descriptor-0. Value of 0 indicates that this value should be ignored and the length should then be taken as specified in the ROM.</p> <p>[1734]: Hub downstream port overcurrent protection/detection and power switching support.</p> <ul style="list-style-type: none"> ■ 1'b0: support is disabled. Power settings are based on Hub Bus Power setting (see 'hub_busPowered'). bPwron2PwrGood is set to 0, indicating no power switching ■ 1'b1: support is enabled. The controller overrides bPwron2PwrGood field in the hub descriptor and changes its value to 50 indicating max power switching time of 100 ms. <p>[1735]:</p> <ul style="list-style-type: none"> ■ 1'b0: U1/U2 entry is enabled as per USB specification.

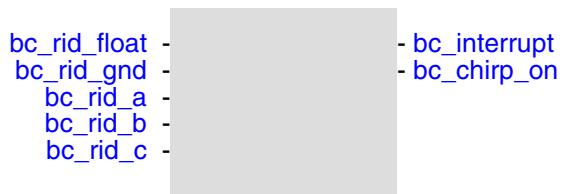
Port Name	I/O	Description
hub_desc_override_params[(DWC_USB3_HUB_OVERRIDE_PARAMS_WIDTH-1):0]...(cont.)	I.	<ul style="list-style-type: none"> ■ 1'b1: the hub upstream and downstream ports do not request a low power entry (U1/U2). They also reject any low power entry request from their link partners. The downstream port still sends the U2 timer LMPs and the upstream port still accepts the SetFeature U1/U2 enables if set by the host, but the hub does not request or accept a U1/U2 entry. However, if the hub upstream receives a ForceLinkPMAccept LMP, or if the hub controller receives a SetPortFeature(ForceLinkPmAccept) command to any of the downstream ports, then this feature is overridden and the hub continues to enable low power entry until the upstream port is disconnected or a hardware reset occurs. <p>[1736]:</p> <ul style="list-style-type: none"> ■ 1'b0: power down mode P3 is attempted for U3, SS_DIS and RX_DETECT (if enabled through coreConsultant). ■ 1'b1: power down mode P3 is never attempted, even in U3. In this case, when in U3, the power down mode is P2. <p>[1737]:</p> <ul style="list-style-type: none"> ■ 1'b1: The U1/U2 timers of the downstream port will be forced to FF. ■ 1'b0: The U1/U2 timers of the downstream port will follow normal operation. <p>[1738]:</p> <ul style="list-style-type: none"> ■ 1'b1: The downstream port U1 ping receive timeout is 500 ms instead of 300 ms. Minimum Ping.LFPS receive duration is 8 ns in the mac3_clk domain. ■ 1'b0: The downstream port U1 ping receive timeout is 300ms. <p>[1739]:</p> <ul style="list-style-type: none"> ■ 1'b1: On a U1/U2 LFPS exit fail, the link transitions to recovery instead of SS_INACTIVE. ■ 1'b0: On a U1/U2 LFPS exit fail, the link transitions to SS_INACTIVE. <p>[1740]:</p> <ul style="list-style-type: none"> ■ 1'b1: The hub has to be enabled for remote wakeup to propagate the downstream wakeup. ■ 1'b0: Propagating the downstream wakeup does not depend the hub remote wakeup. <p>[1741]:</p> <ul style="list-style-type: none"> ■ 1'b0: If a propagated reset ends up as a warm reset the downstream port, the CCS goes low. ■ 1'b1: If a propagated reset ends up as warm reset the downstream port, the CCS remains high. This behavior matches with the USB 3.0 Specification which requires the CCS to be 1 in the DSPort.resetting state. <p>[1742]:</p> <ul style="list-style-type: none"> ■ 1'b1: Does not STALL SetfeatureHalt(EP0). ■ 1'b0: Does STALL SetfeatureHalt(EP0). <p>[1743]:</p> <ul style="list-style-type: none"> ■ 1'b0: The controller does U1/U2/U3 exit in P0.

Port Name	I/O	Description
hub_desc_override_params[(DWC_USB3_HUB_OVERRIDE_PARAMS_WIDTH-1):0]...(cont..)	I..	<ul style="list-style-type: none"> ■ 1'b1: The controller does U1/U2/U3 exit in P1/P2/P3 respectively. <p>[1759-1744]: Disables P3 in RX detect Active state (Per Port). Bit 1744 is for Upstream port. Bits 1745 to 1759 control a maximum of 15 Downstream ports including the unused ones. Bit 1745 controls DSPort-1, and bit 1759 controls DSPort-15. If DWC_USB3_RXDET_IN_P3_DS =0, then these bits have no effect.</p> <ul style="list-style-type: none"> ■ 1'b0: P3 for RxDetect.Active, Quiet states ■ 1'b1: P2 for RxDetect.Active. P3 for RxDetect. Quiet state. <p>[1760]: Specifies the minimum LFPS reception from the remote link to consider Ux exit handshake successful for a locally-initiated U2 exit</p> <ul style="list-style-type: none"> ■ 1'b0: 256ns (default) ■ 1'b1: 80ms <p>[1761]: Enable check for LFPS overlap during remote Ux Exit</p> <ul style="list-style-type: none"> ■ 1'b1: The SuperSpeed link when exiting U1/U2/U3 waits for either the remote link LFPS or TS1/TS2 training symbols before it confirms that the LFPS handshake is complete. This is done to handle the case where the LFPS glitch causes the link to start exiting from the low power state. Looking for the LFPS overlap makes sure that the link partner also sees the LFPS. ■ 1'b0: When the link exists U1/U2/U3 because of a remote exit, it does not look for an LFPS overlap. <p>[1762]: Enable P3 in U2.</p> <ul style="list-style-type: none"> ■ 1'b0 - Do not attempt P3 in U2. ■ 1'b1 - Attempt P3 in U2. <p>Note: If this bit is set, then bit [1743] must be 0 to allow U2 exit in P0.</p> <p>[1763]: Control for tRxDetectQuietTimeoutDFP</p> <ul style="list-style-type: none"> ■ 1'b0 - Use 12ms timeout value ■ 1'b1 - Use 114 ms to 118 ms timeout value. Range depends on suspend clock. <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: 0:1744=pipe3_rx_pclk[0],suspend_clk;1745:1759=N/A;1760:1763=pipe3_rx_pclk[0],suspend_clk</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hub_desc_override_ctrl[1:0]	I	<p>Controls overriding of descriptor values.</p> <p>bit[0]: Controls overriding of Container ID (hub_desc_override_params[127:0])</p> <ul style="list-style-type: none"> ■ 0: Do Not Override the Descriptor Values ■ 1: Override the Descriptor value <p>bit[1]: Controls overriding of Vendor ID, Product ID, and Power Switching Control (hub_desc_override_params[160:128])</p> <ul style="list-style-type: none"> ■ 0: Do Not Override the Descriptor Values ■ 1: Override the Descriptor value <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: pipe3_rx_pclk[0],suspend_clk</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_port_dev_speed[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>Unused input. Will be removed in later release.</p> <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
hub_port_unused[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>Indicates if a particular port must not be used as a downstream port. For example, you can develop a 4 port Hub chip and package it as only a 2-port hub product. You can disable multiple downstream ports in any way you want; there are no restrictions that only the upper ports can be unused. hub_port_unused[0] (upstream port's bit) should be tied to '0'.</p> <ul style="list-style-type: none"> ■ 0: Port used ■ 1: Port unused <p>Exists: DWC_USB3_MODE == 3</p> <p>Synchronous To: This port uses pipe3_rx_pclk[i] during USB 3.0 (SS) operational mode, otherwise suspend_clk, where $0 \leq i \leq 14$ depending on your configuration. For clock(s) specific to your configuration, see coreConsultant IO report.</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
hub_scaledown_mode[1:0]	I	<p>When Scale-Down mode is enabled for simulation, the controller uses scaled-down timing values, resulting in faster simulations. When Scale-Down mode is disabled, actual timing values are used. This is required for hardware operation.</p> <ul style="list-style-type: none"> ■ 2 b00: Disables all scale-downs. Actual timing values are used. ■ 2 b01: Enables scaled down SS timing and repeat values including Number of TxEq training sequences reduced to 8, LFPS polling burst time reduced to 100 nS, LFPS warm reset receive reduced to 30 uS ■ 2 b10: No TxEq training sequences are sent. Overrides Bit 4. ■ 2'b11: Enables bit 0 and bit 1 scale-down timing values. <p>Exists: DWC_USB3_MODE == 3 Synchronous To: N/A Registered: N/A Power Domain: Vcc Active State: N/A</p>
hub_pwrdnscl_whubdel[31:0]	I	<p>This input controls the Power Down scale value and wHubDelay.</p> <ul style="list-style-type: none"> ■ [15:0]: Specifies the wHubDelay in the hub descriptor. If these bits are 0, then a default value of 253ns is specified. ■ [31:16]: Specifies the Power Down scale value. <p>The USB3 suspend_clk input replaces pipe3_rx_clk as a clock source to part of the USB3 hub controller that operates when the SS PHY is in the lowest power (P3) state and not capable of providing a clock. Suspend clock is divided down internally to derive mac3_clk.</p> <p>The Power down scale field specifies how many mac3_clks(divided suspend_clk) periods fit into the period of a 16 KHz clock. For details on calculating the value, refer to PwrDnScale field in the GCTL section of the Programming Guide.</p> <p>Exists: DWC_USB3_MODE == 3 Synchronous To: N/A Registered: N/A Power Domain: Vcc Active State: N/A</p>
hub_enable_scrambling	I	<p>Scrambling Enable/Disable control for upstream and all downstream Ports.</p> <ul style="list-style-type: none"> ■ 1: Enables scrambling. ■ 0: Disable scrambling. <p>Exists: DWC_USB3_MODE == 3 Synchronous To: N/A Registered: N/A Power Domain: Vcc Active State: N/A</p>

7.19 ACA Interface Signals



The bc_rid_* signals must be filtered outside the DWC_usb3 controller because they are not filtered within the controller.

Table 7-19 ACA Interface Signals

Port Name	I/O	Description
bc_interrupt	O	<p>BC Interrupt Line. Interrupts the application to indicate the occurrence of a BC event needing application intervention. This signal is high as long as any event is set in BC Event (BCEVT) register.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: No pending BC Events ■ 1'b1: At least one pending BC Event <p>Exists: (DWC_USB3_EN_BC == 1)</p> <p>Synchronous To: bus_clk_early,suspend_clk</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
bc_chirp_on	O	<p>When asserted indicates an imminent chirp signal. Values: 0,1</p> <p>Exists: (DWC_USB3_EN_BC == 1)</p> <p>Synchronous To: PHY clock</p> <p>Registered: Yes</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
bc_rid_float	I	<p>Measured RID_A resistance of the IDDIG pin.</p> <p>Values: 0, 1</p> <p>Exists: (DWC_USB3_EN_BC == 1)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
bc_rid_gnd	I	<p>Measured RID_GND resistance of the IDDIG pin. If this value is 1, it indicates a A-device mode of operation.</p> <p>Values: 0,1</p> <p>Exists: (DWC_USB3_EN_BC == 1)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
bc_rid_a	I	<p>Measured RID_A resistance of the IDDIG pin. If this value is 1, it indicates a A-device mode of operation.</p> <p>Values: 0,1</p> <p>Exists: (DWC_USB3_EN_BC == 1)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
bc_rid_b	I	<p>Measured RID_B resistance of the IDDIG pin. Values: 0,1</p> <p>Exists: (DWC_USB3_EN_BC == 1)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
bc_rid_c	I	<p>Measured RID_C resistance of the IDDIG pin. Values: 0,1</p> <p>Exists: (DWC_USB3_EN_BC == 1)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.20 Miscellaneous Signal Interface Signals

gp_in	- logic_analyzer_trace
scan_mode	- interrupt
dft_reset_mux_ctl	- gp_out
dft_clk_mux_ctl	- pme_generation
dft_clk_mux_static_ctl	
dft_clk_mux_ctlval_mac2_wpc_clk	
dft_clk_mux_ctlval_rx_mx_pclk	
dft_clk_mux_ctlval_tx_mx_pclk	
dft_clk_mux_ctlval_ram_clk	
dft_clk_mux_ctlval_mac3_clk	
dft_clk_mux_ctlval_mac_clk	
dft_clk_mux_ctlval_bus_clk	
dft_clk_mux_ctlval_ref_clk	
dft_clk_mux_ctlval_pa_cfgclk	
dft_en_bus_clk_gated	
dft_en_ram_clk_gated	
pme_en	
bus_filter_bypass	

This interface contains signals that do not belong to any other interface.

Table 7-20 Miscellaneous Signal Interface Signals

Port Name	I/O	Description
logic_analyzer_trace[63:0]	O	<p>Logic Analyzer Trace. These are internal design signals that you can use for debug purposes. Some of the signals are interface signals like PIPE, UTMI, ULPI, AXI, and AHB, and others are internal state machines and status information signals. During chip bring-up, to debug functional issues, you can probe the interface signals for additional visibility.</p> <p>Chip Debug Recommendation:</p> <ul style="list-style-type: none"> ■ Check USB trace, driver log, register dump, and system interface (PCIe/AXI/AHB) signals to isolate the problem. ■ Check the additional interface-related logic_analyzer_trace signals for further debugging. ■ For further assistance, open a SolvNet case and send the USB trace, driver log, register dump, and PCIe analyzer trace (if your system is PCIe) to Synopsys. Synopsys will analyze the information, and if needed, will request that you log additional logic_analyzer_trace signals. <p>For details on how to control this port to bring out different debug information, refer to GDBGLSPMUX_DEV and GDBGLSPMUX_HST of the Programming Guide.</p> <p>For details on the selected signals, see the section "assign logic_analyzer_trace = " of the DWC_usb3.v file.</p> <p>Note: The mux select register loses its value when vcc_reset_n is asserted. To change this behavior, set the parameter DWC_USB3_PRESERVE_LOGIC_ANALYZER_SELECT to 1. This detaches the mux register from vcc_reset_n which allows debugging across board resets and hibernation resets.</p> <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: N/A</p>

Port Name	I/O	Description
interrupt[(DWC_USB3_NUM_INT-1):0]	O	<p>Interrupt Line This signal interrupts the application to indicate the occurrence of an event needing application intervention. This signal is used in both the device mode and the host mode. In the host mode, it is a one-bus clock cycle pulse when host_msi_enable is 1, or a level signal when host_msi_enable is 0. In the device mode, it is always a level signal. In the DRD mode, host_msi_enable must be tied to 0, and it is a level signal.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: Event interrupt disabled ■ 1'b1: Event interrupt enabled <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4) Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early" Registered: No Power Domain: Vcc Active State: High</p>
gp_out[15:0]	O	<p>General Purpose Output Port; can be used as general purpose outputs. Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4) Synchronous To: DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early" Registered: Yes Power Domain: Vcc Active State: High</p>
pme_generation	O	<p>PME# Generation. This signal is used to generate a PME# (Power Management Event). When the Run/Stop bit of the USB Command Register is cleared during USB suspend mode, the controller cannot generate an event and cannot assert a regular interrupt. In this case, the controller asserts a pme_generation signal to report any wakeup condition if pme_en is high. If the system does not support PCI-like PME interface, then it must not clear the Run/Stop bit during USB suspend mode. In this case, the controller generates an event and asserts an interrupt when there is any wakeup event. If the pme interface is not used, connect pme_en to zero and keep pme_generation unconnected. This is a level signal that gets cleared when the software clears CSC, OCC and PLC bits in all the PORTSC registers. In device mode, connect pme_en to zero and keep pme_generation unconnected.</p> <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4) Synchronous To: DWC_USB3_MODE==0 ? "None" : "bus_clk_early" Registered: No Power Domain: Vcc Active State: High</p>

Port Name	I/O	Description
gp_in[15:0]	I	<p>General Purpose Input Port; can be used as general purpose inputs.</p> <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_RM_OPT_FEATURES==1 ? "None" : (!!(DWC_USB3_MODE != 3 && DWC_USB3_SBUS_TYPE == 3) ? (DWC_USB3_EN_PWROPT==2 ? "bus_clk_early,suspend_clk" : "bus_clk_early") : "None")</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
scan_mode	I	<p>Scan Mode Port.</p> <p>This signal enables scan-mode bypass to improve scan coverage.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: Scan mode bypass disabled ■ 1'b1: Scan mode bypass enabled Controls the internally generated clocks and reset during scan mode for clean DFT. <p>When additional At-Speed DFT controls are enabled, it only controls a few flops which have internally generated reset/clock. During scan-mode, this should be '1'.</p> <p>Exists: Always</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dft_reset_mux_ctl	I	<p>At-Speed DFT control port that controls the internally generated reset bypass.</p> <p>This signal is '0' by default.</p> <p>Exists: DWC_USB3_ATSPEED_DFT == 1</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dft_clk_mux_ctl	I	<p>At-Speed DFT control port; a dynamic control signal which enables external clock MUX and clock gate control.</p> <p>This signal is '0' by default.</p> <p>Exists: DWC_USB3_ATSPEED_DFT == 1</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

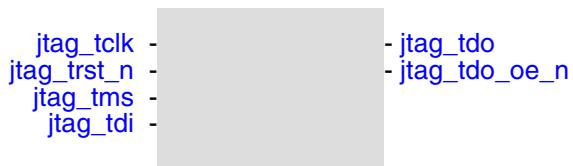
Port Name	I/O	Description
dft_clk_mux_static_ctl	I	<p>At-Speed DFT control port; a static dft signal which controls clock bypass.</p> <p>This signal is '0' by default. It must be '1' in test mode.</p> <p>Exists: DWC_USB3_ATSPEED_DFT == 1</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dft_clk_mux_ctlval_mac2_wpc_clk[2:0]	I	<p>At-Speed DFT port that controls mac2_clk/wpc_clk selection.</p> <p>This signal is '0' by default.</p> <p>Note: To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_mac2_wpc_clk[2:0]. For example, 3'b001, 3'b010, etc.</p> <p>Exists: DWC_USB3_ATSPEED_DFT == 1</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dft_clk_mux_ctlval_rx_mx_pclk[1:0]	I	<p>At-Speed DFT port that controls pipe3_mx_rx_pclk.</p> <p>This signal is '0' by default.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_rx_mx_pclk[2:0]. For example, 2'b01, or 2'b10. ■ In USB 2.0-only mode, this signal is not present. <p>Exists: ((DWC_USB3_ATSPEED_DFT == 1) && (DWC_USB3_SSPhy_INTERFACE == 1))</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dft_clk_mux_ctlval_tx_mx_pclk[1:0]	I	<p>At-Speed DFT port that controls pipe3_mx_tx_pclk.</p> <p>This signal is '0' by default.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_tx_mx_pclk[2:0]. For example, 2'b01, or 2'b10. ■ In USB 2.0-only mode, this signal is not present. <p>Exists: ((DWC_USB3_ATSPEED_DFT == 1) && (DWC_USB3_SSPhy_INTERFACE == 1))</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
dft_clk_mux_ctlval_ram_clk[3:0]	I	<p>At-Speed DFT port that controls ram_clk. This signal is '0' by default.</p> <p>Note: To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_ram_clk[2:0]. For example, 4'b0001, 4'b0010, etc.</p> <p>Exists: DWC_USB3_ATSPEED_DFT == 1</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dft_clk_mux_ctlval_mac3_clk[2:0]	I	<p>At-Speed DFT port that controls mac3_clk. This signal is '0' by default.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_mac3_pclk[2:0]. For example, 3'b001, 3'b010, etc. ■ In USB 2.0-only mode, this signal is not present. <p>Exists: ((DWC_USB3_ATSPEED_DFT == 1) && (DWC_USB3_SSPHY_INTERFACE == 1))</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dft_clk_mux_ctlval_mac_clk[2:0]	I	<p>At-Speed DFT port that controls mac_clk. This signal is '0' by default.</p> <p>Note: To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_mac_clk[2:0]. For example, 3'b001, 3'b010, etc.</p> <p>Exists: DWC_USB3_ATSPEED_DFT == 1</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
dft_clk_mux_ctlval_bus_clk[1:0]	I	<p>At-Speed DFT port that controls bus_clk. This signal is '0' by default.</p> <p>Note: To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_bus_clk[2:0]. For example, 2'b01, or 2'b10.</p> <p>Exists: DWC_USB3_ATSPEED_DFT == 1</p> <p>Synchronous To: N/A</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
dft_clk_mux_ctlval_ref_clk[1:0]	I	<p>This signal is the At-Speed DFT port that controls ref_clk. This signal is '0' by default.</p> <p>Note: To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_ref_clk[1:0]. For example, 2'b01, or 2'b10.</p> <p>Exists: DWC_USB3_ATSPEED_DFT == 1 Synchronous To: N/A Registered: No Power Domain: Vcc Active State: High</p>
dft_clk_mux_ctlval_pa_cfgclk[1:0]	I	<p>At-speed DFT control input that controls the pa_cfg_clk. This signal is '0' by default.</p> <p>Note: To select a clock for the scan mode, set only one bit enable for dft_clk_mux_ctlval_pa_cfgclk [1:0]. For example, 2'b01, 2'b10, etc.</p> <p>Exists: ((DWC_USB3_NUM_SSIC_PORTS != 0) && (DWC_USB3_ATSPEED_DFT == 1) && (DWC_USB3_REF_CLK_SHUTDOWN ==1)) Synchronous To: N/A Registered: No Power Domain: Vcc Active State: High</p>
dft_en_bus_clk_gated	I	<p>At-Speed DFT control port that controls bus_clk_gated. This signal is '0' by default.</p> <p>Exists: Always Synchronous To: N/A Registered: No Power Domain: Vcc Active State: High</p>
dft_en_ram_clk_gated	I	<p>At-Speed DFT port that controls ram_clk_gated and ram_clk_gated_ram0. This signal is '0' by default.</p> <p>Exists: Always Synchronous To: N/A Registered: No Power Domain: Vcc Active State: High</p>

Port Name	I/O	Description
pme_en	I	<p>Enable signal for the pme_generation. Enable the controller to assert pme_generation.</p> <p>Refer to the description of pme_generation signal for details.</p> <p>Exists: (DWC_USB3_MODE != 3 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_MODE==0 ? "None" : "bus_clk_early"</p> <p>Registered: (DWC_USB3_MODE==1 DWC_USB3_MODE==2) ? "Yes" : "No"</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
bus_filter_bypass[3:0]	I	<p>Bus Filter Bypass. Disables the internal bus filters that are enabled by DWC_USB3_EN_BUS_FILTERS coreConsultant parameter. This static signal is present only when DWC_USB3_EN_BUS_FILTERS is 1. It is expected that this signal is set or reset at power-on reset and is not changed during the normal operation of the controller. The function of each bit is:</p> <ul style="list-style-type: none"> ■ bus_filter_bypass[3]: Reserved ■ bus_filter_bypass[2]: Bypass the filter for utmisrp_bvalid ■ bus_filter_bypass[1]: Bypass the filter for pipe3_PowerPresent all U3 ports ■ bus_filter_bypass[0]: Bypass the filter for utmiotg_vbusvalid all U2 ports <p>In Host-only mode, internal bus filters are not needed. Therefore, bus_filter_bypass[2:0] must be connected to logic high value (3'b111). The reserved bit can be tied to 0 or 1, but should not be floating.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: Bus filter(s) enabled ■ 1'b1: Bus filter(s) disabled (bypassed) <p>Exists: (DWC_USB3_EN_BUS_FILTERS==1)</p> <p>Synchronous To: N/A</p> <p>Registered: N/A</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.21 JTAG Interface Signals



The DWC_usb3 controller provides standard JTAG ports and a TAP controller to control debug operation. If the JTAG ports are not used, tie all the JTAG inputs (jtag_tclk, jtag_tdi, jtag_trst_n, and jtag_tms) to 0.

Table 7-21 JTAG Interface Signals

Port Name	I/O	Description
jtag_tdo	O	JTAG Test Data Output (negedge-driven). Test data out. Driven out the negedge of jtag_tclk Exists: DWC_USB3_JTAG_INTERFACE == 1 Synchronous To: jtag_tclk Registered: Yes Power Domain: Vcc Active State: High
jtag_tdo_oe_n	O	JTAG Test Data Output Enable (negedge-driven). Buffer enable for the jtag_tdo tri-state buffer; active low. Driven out the negedge of jtag_tck. Exists: DWC_USB3_JTAG_INTERFACE == 1 Synchronous To: jtag_tclk Registered: Yes Power Domain: Vcc Active State: Low
jtag_tclk	I	JTAG Test Clock. This clock frequency must be one fourth or less of the APP clock (mac_clk) speed. Exists: DWC_USB3_JTAG_INTERFACE == 1 Synchronous To: None Registered: No Power Domain: Vcc Active State: High
jtag_trst_n	I	JTAG Test Reset. This port must have a weak pull-up board. Exists: DWC_USB3_JTAG_INTERFACE == 1 Synchronous To: Asynchronous Registered: N/A Power Domain: Vcc Active State: Low

Port Name	I/O	Description
jtag_tms	I	<p>JTAG Test Mode Select (posedge-sampled). If jtag_tms is kept high for 5 consecutive clocks, the tap controller enters a Test Logic Reset state. This port must have a weak pull-up board.</p> <p>Exists: DWC_USB3_JTAG_INTERFACE == 1</p> <p>Synchronous To: jtag_tclk</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
jtag_tdi	I	<p>JTAG Test Mode Select (posedge-sampled). Test data in, captured the posedge of jtag_tclk. This port must have a weak pull-up board.</p> <p>Exists: DWC_USB3_JTAG_INTERFACE == 1</p> <p>Synchronous To: jtag_tclk</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

7.22 Synopsys Test Environment Interface Signals

- debug

These debug signals are used in Synopsys internal testbench. You can also use them for debug purposes.

Table 7-22 **Synopsys Test Environment Interface Signals**

Port Name	I/O	Description
debug[(DWC_USB3_DEBUG_WIDTH-1):0]	O	<p>Debug Port. For more details, see “Description of Synopsys Test Environment Interface Signals” on page 281.</p> <p>Exists: Always Synchronous To: N/A Registered: N/A Power Domain: Vcc Active State: N/A</p>

7.23 Type-C Support Signals for Synopsys PHY



The signals in this section provide support for Type-C implementation with a Synopsys PHY. This interface allows the controller to release the PIPE interface to the PHY so that the PHY can perform lane switching.

Table 7-23 Type-C Support Signals for Synopsys PHY

Port Name	I/O	Description
DisRxDetU3RxDet_ack[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	O	<p>DisRxDetU3RxDet_ack of USB 3.0 SS Ports This signal is an acknowledgment of DisRxDetU3RxDet. When this signal is 1'b1, the controller releases the ownership of the the PIPE interface and does not issue PHY commands to the PIPE interface.</p> <p>Exists: (DWC_USB3_EN_USB2_ONLY == 0 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_MODE == 3 ? "suspend_clk,pipe3_rx_pclk[0]" : (DWC_USB3_HSPHY_INTERFACE==1 ? "suspend_clk,pipe3_rx_pclk[0],utmi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2 ? "suspend_clk,pipe3_rx_pclk[0],ulpi_clk[0]" : "suspend_clk,pipe3_rx_pclk[0],utmi_clk[0],ulpi_clk[0]"))</p> <p>Registered: DWC_USB3_NUM_SSIC_PORTS!=0 ? "Yes" : (DWC_USB3_EN_PWROPT==2 ? "No" : "Yes")</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>
StartRxDetU3RxDet[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>StartRxdetU3RxDet of USB 3.0 SS Ports This signal is the Start Receiver Detection in U3/Rx.Detect (StartRxdetU3RxDet). If DisRxDetU3RxDet is set and the link is in either U3 or Rx.Detect state, the controller starts receiver detection on the rising edge of this bit. This signal can only be used for downstream ports. It must be set to '0' for upstream ports.</p> <p>Exists: (DWC_USB3_EN_USB2_ONLY == 0 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_MODE == 3 ? "None" : "suspend_clk"</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

Port Name	I/O	Description
DisRxDetU3RxDet[(DWC_USB3_NUM_U3_ROOT_PORTS-1):0]	I	<p>DisRxDetU3RxDet of USB 3.0 SS Ports This input signal requests the controller to stop issuing more PHY commands and release the PIPE ownership. The controller acknowledges this request by asserting the DisRxDetU3RxDet_ack signal. This signal is for Type-C support.</p> <p>Exists: (DWC_USB3_EN_USB2_ONLY == 0 && DWC_USB3_MBUS_TYPE != 4)</p> <p>Synchronous To: DWC_USB3_MODE==3 ? "None" : (DWC_USB3_HSPHY_INTERFACE==1 ? "suspend_clk,pipe3_rx_pclk[0],utmi_clk[0]" : (DWC_USB3_HSPHY_INTERFACE==2 ? "suspend_clk,pipe3_rx_pclk[0],ulp_i_clk[0]" : "suspend_clk,pipe3_rx_pclk[0],utmi_clk[0],ulp_i_clk[0]"))</p> <p>Registered: No</p> <p>Power Domain: Vcc</p> <p>Active State: High</p>

8

Hub

This chapter describes the function of the DesignWare Cores USB 3.0 Controller when it has been configured as a Hub.

This chapter includes the following sections:

- “General Product Description” on page 434
- “Hub Software Storage Requirement and Data Structure” on page 437
- “Configuring the Hub” on page 441
- “Hub Descriptor Storage and Initialization” on page 441
- “Sharing ROM or Serial-Flash/PROM between USB 2.0 Hub and SS Hub” on page 444
- “Overriding Descriptor Data through Input Ports” on page 445
- “Hub RAM Requirements” on page 445
- “Debug JTAG” on page 446
- “Hub Controller Clock Scheme” on page 452
- “Hub Controller Timing” on page 453
- “Vendor Control Interface” on page 454

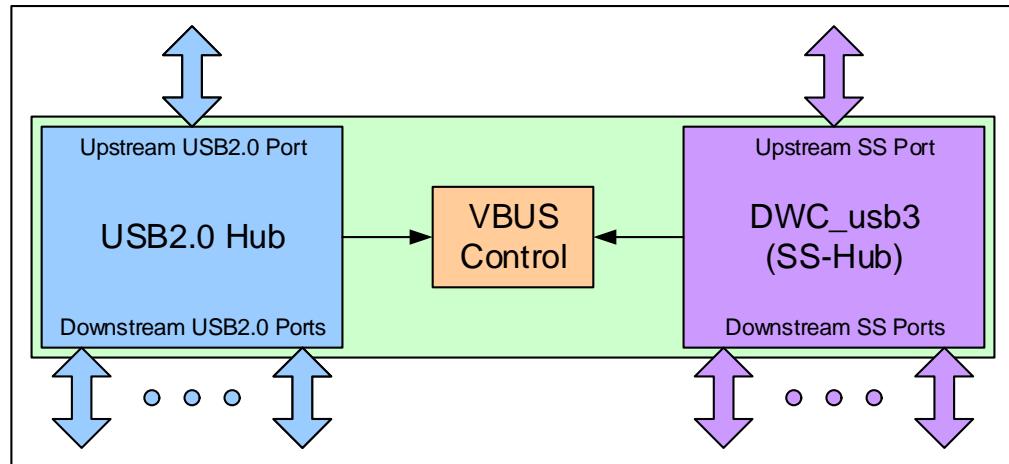
Related Information

- Parameters
 - “Basic Config Parameters” on page 191
 - “Hub Config Parameters” on page 229
- Signals:
 - “Hub Interface Signals” on page 402
 - “JTAG Interface Signals” on page 428

8.1 General Product Description

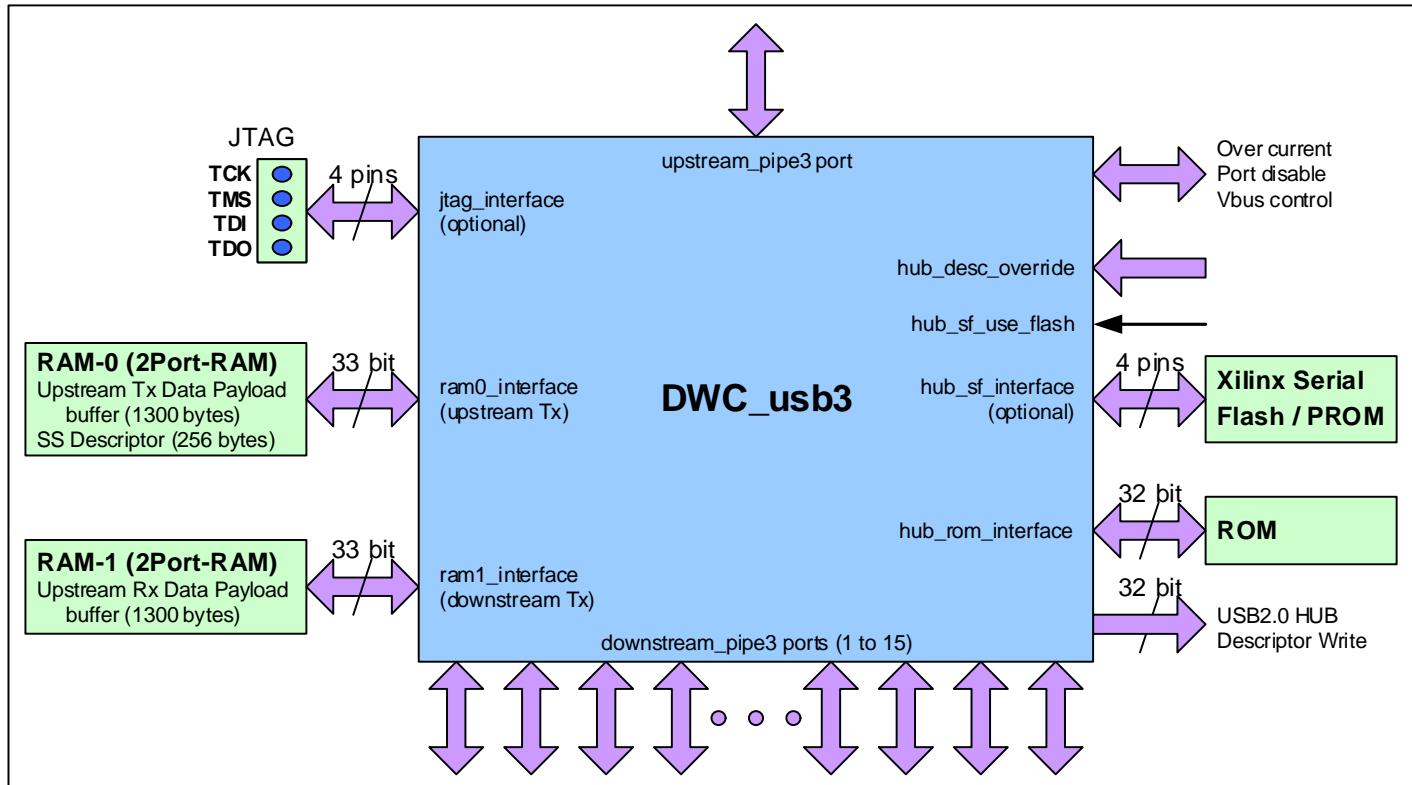
The DesignWare Cores USB 3.0 Controller supports only a SuperSpeed (SS) Hub. You can integrate the SuperSpeed Hub controller with your USB 2.0 Hub controller to build a USB 3.0 Hub controller. The SS Hub controller and USB 2.0 Hub controller functions are independent although both control the VBUS. [Figure 8-1](#) illustrates how to create a USB 3.0 Hub by combining a USB 2.0 Hub and a SS Hub.

Figure 8-1 USB 3.0 Hub



[Figure 8-2](#) shows the DWC_usb3 SS Hub controller interfaces.

Figure 8-2 USB 3.0 Hub Controller Interfaces



The SS Hub controller supports the following features:

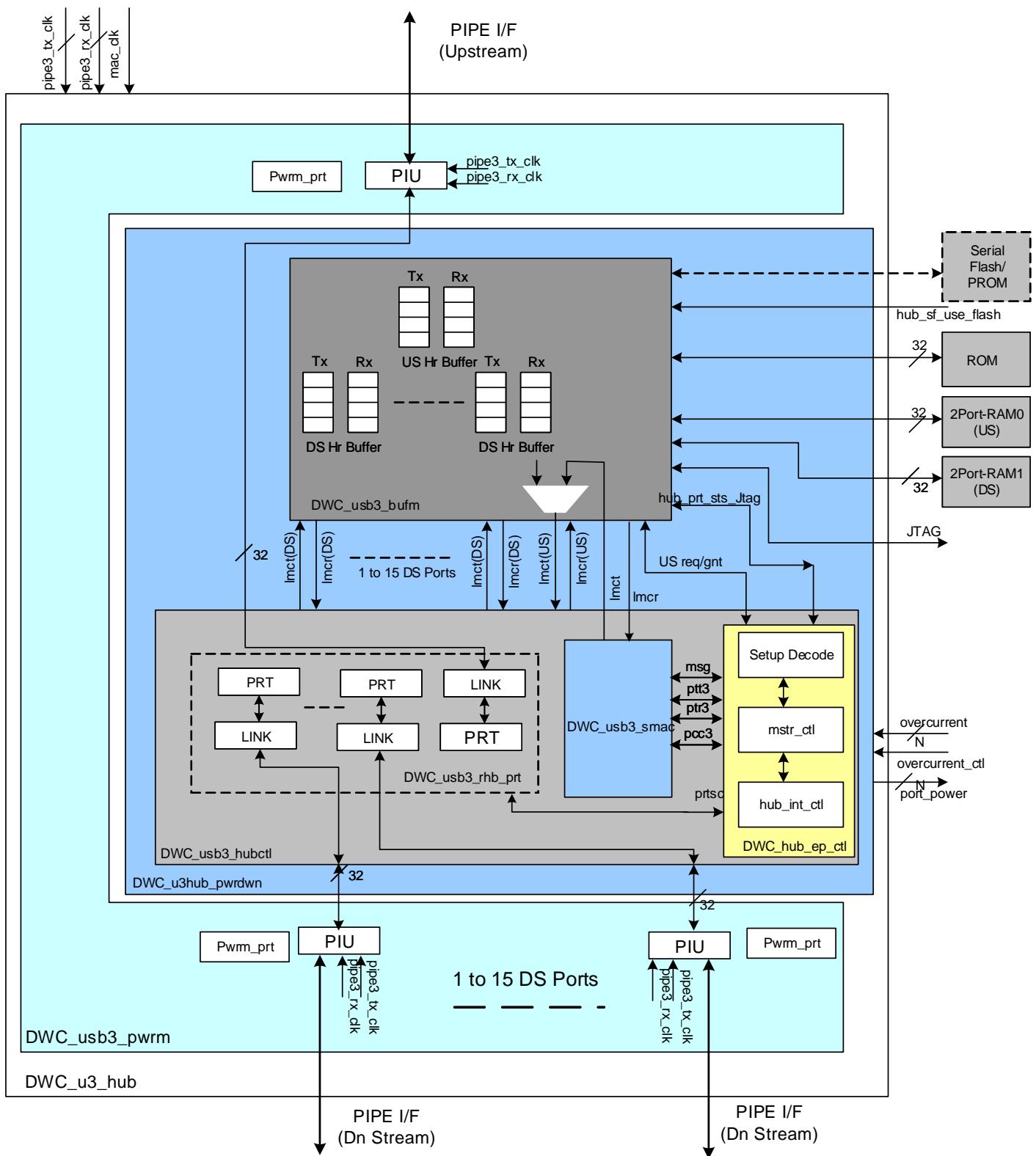
- One upstream port
- 1 to 15 configurable downstream ports
- ROM for storing configurable Hub descriptors

The ROM is created with a ROM compiler or synthesized using Design Compiler or a similar tool (*src/hub/DWC_usb3_hub_rom.v*, *DWC_usb3_hub_rom_data_inc.v*). Some descriptors can be edited through configuration pins.

- Optional Xilinx Serial-Flash/PROM Interface for controller override and alternative descriptor storage location (instead of ROM)
The Controller uses the descriptors from Serial-Flash PROM when it is present if `hub_sf_use_flash` is asserted.
- Sharing the ROM or Serial-Flash/PROM with USB 2.0 Hub controller
- Option to override some descriptors (such as GUID, VendorID, ProductID, power control) with input ports
- Two 2-port RAMs of 33-bit data width (one port read only, one port write only):
 - RAM-0: Upstream Tx Data Payload buffer (1300 bytes minimum) and SS Descriptor storage (256 bytes)
 - RAM-1: Upstream Rx Data Payload buffer (1300 bytes minimum)
- Optional JTAG port for debug
- Permanently attached devices
- Over current protection
- Vbus control
- Option to disable any port

Example: In a port-port hub, when ports 1 and 3 are disabled, the chip behaves as a 2-port hub with port-2 and port-4 remapped as port-1 and port-2.

[Figure 8-3](#) on page [436](#) illustrates the internal block diagram of the hub.

Figure 8-3 Internal Block Diagram of the Hub

8.2 Hub Software Storage Requirement and Data Structure

Figure 8-4 illustrates the hub software descriptor storage format.

- The first 16 DWORDS (32 bit words) specify the size and offset of the descriptors in the ROM.
- The device, configuration, BOS, Hub and string descriptors must start at a DWORD boundary.
- Descriptors read by the host as a part of a single get_descriptor command must be contiguous, such as, config, interface, endpoint, and endpoint companion descriptors.
- Four variable-length string descriptors are supported; they can be expanded to support additional string descriptors.

Figure 8-4 Descriptor ROM Storage Format

Address	Byte3	Byte2	Byte1	Byte0
x00	DeviceDescriptorOffset[15:0]		DeviceDescriptorSize[15:0]	
x04	ConfigDescriptorOffset[15:0]		ConfigDescriptorSize[15:0]	
x08	BosDescriptorOffset[15:0]		BosDescriptorSize[15:0]	
x0C	HubDescriptorOffset[15:0]		HubDescriptorSize[15:0]	
x10	StringDescriptor-0Offset[15:0]		StringDescriptor-0Size[15:0]	
x14	StringDescriptor-1Offset[15:0]		StringDescriptor-1Size[15:0]	
x18	StringDescriptor-2Offset[15:0]		StringDescriptor-2Size[15:0]	
x1C	StringDescriptor-3Offset[15:0]		StringDescriptor-3Size[15:0]	
x20	StringDescriptor-4Offset[15:0]		StringDescriptor-4Size[15:0]	
x24		Reserved		
x28		Reserved		
x2C		Reserved		
x30		Reserved		
x34		Reserved		
x38		Reserved		
x3C		Reserved		
x40	bcdUSB[15:8]	bcdUSB[7:0]	bDescriptorType	bLength
	bMaxPacketSize0	bDeviceProtocol	bDeviceSubclass	bDeviceClass
	idProduct[15:8]	idProduct[7:0]	idVendor[15:8]	idVendor[7:0]
	iProduct	iManufacturer	bcdDevice[15:8]	bcdDevice[7:0]
	Reserved	Reserved	bNumConfigurations	iSerialNumber
	wTotalLength[15:8]	wTotalLength[7:0]	bDescriptorType	bLength
	bmAttributes	iConfiguration	bConfigurationValue	bNumInterfaces
	blInterfaceNumber	bDescriptorType	bLength	bMaxPower
	blInterfaceSubClass	blInterfaceClass	bNumEndpoints	bAlternateSetting
	bDescriptorType	bLength	ilInterface	blInterfaceProtocol
	wMaxPacketSize[15:0]	wMaxPacketSize[7:0]	bmAttributes	bEndpointAddress
	bMaxBurst	bDescriptorType	bLength	blInterval
	Reserved	Reserved	wBytesPerInterval	bmAttributes
	wTotalLength[15:8]	wTotalLength[7:0]	bDescriptorType	bLength
	bDevCapabilityType(2)	bDescriptorType	bLength	bNumDeviceCaps
	bmAttributes[31:24]	bmAttributes[23:16]	bmAttributes[15:8]	bmAttributes[7:0]
	bmAttributes	bDevCapabilityType	bDescriptorType	bLength
	bu1DevExitLat	bFunctionalitySupport	wSpeedsSupported[15:8]	wSpeedsSupported[7:0]
	bDescriptorType	bLength	wU2DevExitLat[15:8]	wU2DevExitLat[7:0]
	ContainerID[15:8]	ContainerID[7:0]	bReserved	bDevCapabilityType
	ContainerID[47:40]	ContainerID[39:32]	ContainerID[31:24]	ContainerID[23:16]
	ContainerID[79:72]	ContainerID[71:64]	ContainerID[63:56]	ContainerID[55:48]
	ContainerID[111:104]	ContainerID[103:96]	ContainerID[95:88]	ContainerID[87:80]
	Reserved	Reserved	ContainerID[127:120]	ContainerID[119:112]
	wHubCharacteristics[7:0]	bNbrPorts	bDescriptorType	bDescLength
	bHubHdrDeclLat	bHubContrCurrent	bPwrOn2PwrGood	wHubCharacteristics[15:8]
	DeviceRemovable[15:8]	DeviceRemovable[7:0]	wHubDelay[15:8]	wHubDelay[7:0]
	wLANGID[0][15:8]	wLANGID[0][7:0]	bDescriptorType	bLength
	wLANGID[N][N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType	bLength
	bString[N:8]			
	bString[15:8]	bString[7:0]	bDescriptorType</	

Table 8-1 lists the descriptors that need to be stored in the ROM/serial flash along with the total byte count.

Table 8-1 Descriptors Stored in ROM/Serial-Flash

Descriptor	Total Byte Count
Standard Descriptors	
Device Descriptor	18 Bytes
Configuration Descriptor	9 Bytes
Interface Descriptor	9 Bytes
Endpoint	7 Bytes (only one Interrupt endpoint)
Endpoint Companion	6 Bytes
BOS Descriptor	5 Bytes
Dev-Cap (2.0 Ext)	7 Bytes
Dev-Cap (SS)	10 Bytes
Dev-Cap (ConID)	20 Bytes
Hub Class Specific Descriptors	12 Bytes
String Descriptor	Four string descriptors, approximately 20-25 Bytes each (variable and vendor specific)

8.2.1 Overriding the Hub Descriptor

The controller also has the option to override the descriptor information read from the ROM, based on the values of some primary inputs. The following table lists the specific fields in the descriptors that are overridden and provides details on how they are overridden.

Table 8-2 Overriding the Hub Descriptor Information

Descriptor	Descriptor Field	Controlling Input	Details
Config Descriptor	bmAttributes	hub_busPowered	The Self Powered field in the bmAttributes is modified based on the hub_busPowered input.
	bMaxPower	hub_busPowered	<ul style="list-style-type: none"> ■ If hub_busPowered ==1, then a value of h'70 is returned. ■ If hub_busPowered ==0, then a value of 0 is returned.
Device descriptor	idVendor	hub_desc_override_params[143:128], hub_desc_override_ctrl	If hub_desc_override_ctrl is 1 then overrides VendorId with hub_desc_override_params[143:128].
	idProduct	hub_desc_override_params[159:144], hub_desc_override_ctrl	If hub_desc_override_ctrl is 1 then overrides VendorId with hub_desc_override_params[159:144].

Descriptor	Descriptor Field	Controlling Input	Details
	iManufacturer	hub_desc_override_params[686:680]	If the value of hub_desc_override_params[686:680] > 0 then a value of 3 is written to the iManufacturer string id.
	iProduct	hub_desc_override_params[1205:1199]	If the value of hub_desc_override_params[1205:1199] > 0 then a value of 2 is written to the iProduct string id.
	iSerialNumber	hub_desc_override_params[167:161]	If the value of hub_desc_override_params[167:161] > 0 then a value of 1 is written to the iSerialNumber string id.
Bos Descriptor	containerID	hub_desc_override_params[127:0], hub_desc_override_ctrl[0]	If hub_desc_override_ctrl[0] is 1 then containerID field is overwritten with the value of hub_desc_override_params[127:0].
Hub Descriptor	bnbrPorts	hub_port_unused	Based on the hub_port_unused bits the number of ports reported is updated.
	wHubCharacteristics - D0	hub_desc_override_ctrl[1], hub_desc_override_params[160]	If hub_desc_override_ctrl[1] is 1 then the D0 field is set to 1.
	wHubCharacteristics - D2	hub_port_perm_attach, hub_port_unused	D2 (Compound device) is set to 1 if any bit in the hub_port_perm_attach is set to 1 and the corresponding port is NOT marked as unused.
	wHubCharacteristics - D3	hub_busPowered, hub_desc_override_params[1734], hub_overcurrent_ctrl	The equation used is (!hub_busPowered hub_desc_override_params[1734]) ? !hub_override_ctrl : 0.
	wHubCharacteristics - D4	hub_busPowered, hub_desc_override_params[1734]	The equation used is (hub_busPowered !hub_desc_override_params[1734]) ? 1 : 0.
	bPwrOn2PwrGood	hub_busPowered, hub_desc_override_params[1734]	If hub_busPowered =1 Or hub_desc_override_params[1734]=1 then a value of 50 is written. Otherwise a value of 0 is written.
	wHubDelay	hubpwrwnscl_whubdel[15:0]	If hubpwrwnscl_whubdel[15:0] is a non-zero value, then this value will be written.
	Device Removable	hub_port_perm_attach	The value of hub_port_perm_attach is used to override this.

Descriptor	Descriptor Field	Controlling Input	Details
StringDescriptor1	bLength	hub_desc_override_params[167:161]	if hub_desc_override_params [167:161] is > 0 then this value is written.
String Descriptor 2	bLength	hub_desc_override_params[1205:1199]	if hub_desc_override_params [1205:1199] is > 0 then this value is written.
String Descriptor 3	bLength	hub_desc_override_params[686:680]	if hub_desc_override_params [686:680] is > 0 then this value is written.

8.3 Configuring the Hub

To enable the DWC_usb3 controller to operate as a Hub, set the “Mode of Operation” parameter (DWC_USB3_MODE) to Hub (3). You can then specify more details about the Hub by configuring the following parameter options:

- Number of USB 3.0 capable Hub ports
- Xilinx serial flash-RAM/PROM support
- MAC clock to serial flash/PROM clock divide factor
- JTAG debug support
- Number of header in each Rx/Tx header buffer
- Number of packets in upstream Rx/Tx data packet buffer
- Depth of descriptor ROM or serial-flash
- Starting depth of SS descriptors in ROM or serial-flash
- Vendor control interface
- Scrambling control for simulation

For more information on hub parameters, see “[Hub Config Parameters](#)” on page 229.

8.4 Hub Descriptor Storage and Initialization

[Figure 8-5](#) illustrates how the Hub descriptors are stored. On power on, the DWC_usb3 controller reads the descriptors from the ROM (in ROM only configuration) or from Serial-Flash/PROM (when Serial-Flash/PROM interface is present and hub_sf_use_f1sh port is 1) and stores them to the SRAM. The DWC_usb3 controller uses the descriptors stored in the SRAM during Hub enumeration. After initialization, the ROM and Serial-Flash/PROM are not used and can be shut down.

A 32-bit synchronous ROM interface is provided in the controller. If your configuration is fixed, you can synthesize the `src/hub/DWC_usb3_hub_rom.v` and `DWC_usb3_hub_rom_data_inc.v` ROM files using DesignCompiler. The ROM interface has a `hub_rom_data_rdy` pin that allows you to add a wrapper outside the controller and interface to any ROM of your choice. The ROM does not have to be a 32-bit synchronous ROM.



The hub_rom_data_rdy signal is for the data phase and does not control the address phase. For example, if hub_rom_data_rdy is low from a power-on reset, the address '0' stays valid only for one clock cycle after reset. Then the address increments to '1' and stays at '1' until hub_rom_data_rdy is asserted high.

Figure 8-5 Firmware Code Initialization Options

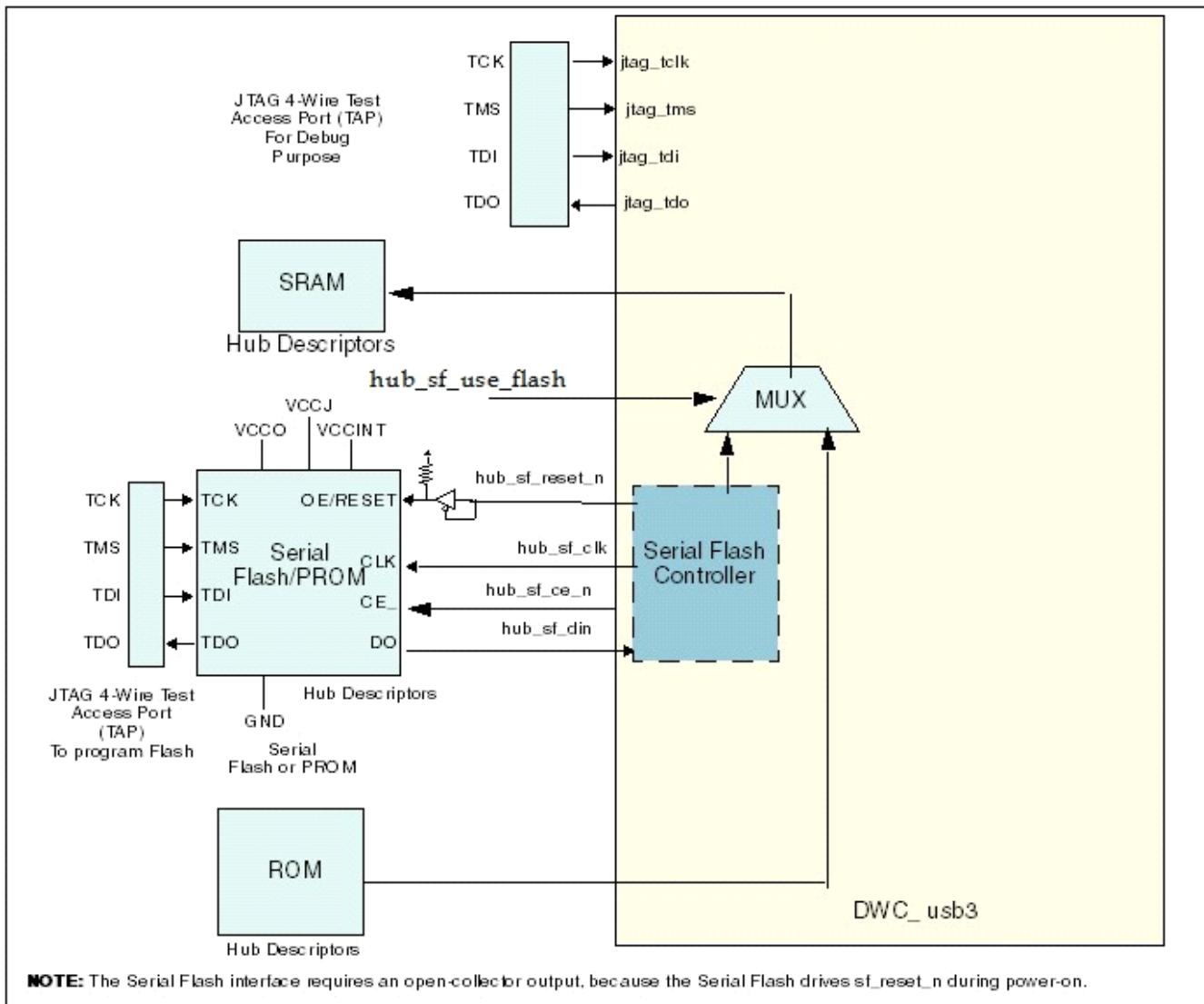


Figure 8-5 also illustrates how the Serial-Flash interfaces with the DWC_usb3 controller. The DWC_usb3 controller provides the interface for Xilinx Serial Flash/PROM memories. The Serial Flash/PROM size must be greater than or equal to 256 bytes. The Serial-Flash/PROM controller in the DWC_usb3 acts as master to read the data from the Flash/PROM.

When Serial Flash/PROM mode is chosen (Serial-Flash/PROM interface present and hub_sf_use_flash is 1), the Serial-Flash/PROM Controller uses the Serial-Flash/PROM clock divider count parameter value to

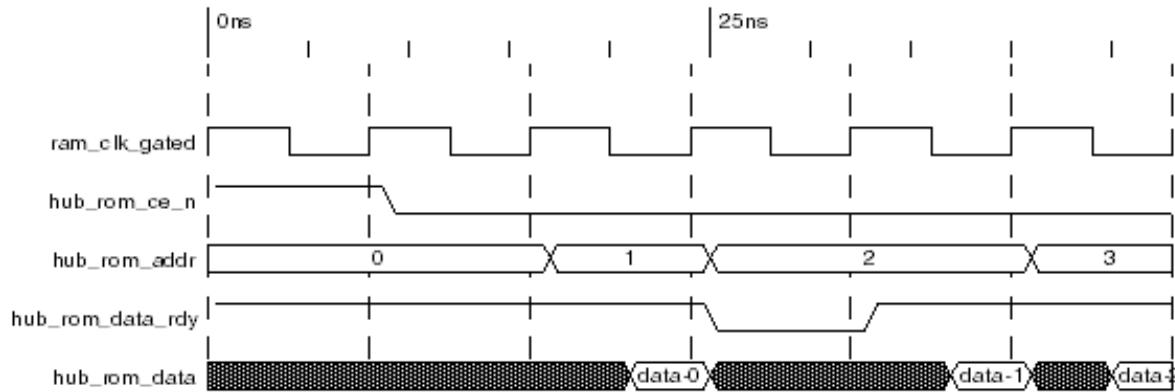
divide the 125-MHz mac3_clk and generate the serial flash clock. Typical Xilinx Flash/PROMs operate at 2.5 MHz to 20 MHz. When the reset is removed, the controller drives sf_reset_n, sf_ce_n, and sf_clk signals and reads in serial data from sf_din.

If Serial-Flash is used, then the JTAG TAP port can be used to program the Serial-Flash during manufacturing or during a field upgrade.

For more information, refer to the *Xilinx Platform Flash In-System Programmable Configuration PROMS Datasheet*.

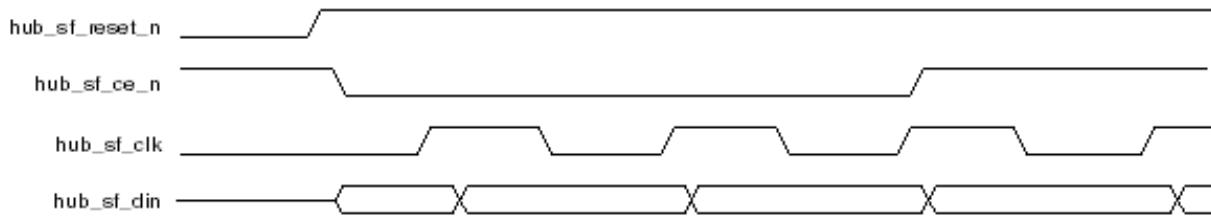
[Figure 8-6](#) illustrates the Hub ROM interface waveform of the DWC_usb3 controller.

Figure 8-6 Hub ROM Interface



[Figure 8-7](#) illustrates the Hub Serial Flash/PROM interface waveform of DWC_usb3 controller.

Figure 8-7 Hub Serial-Flash/PROM Interface



8.5 Sharing ROM or Serial-Flash/PROM between USB 2.0 Hub and SS Hub

The SS Hub and the USB 2.0 Hub can share the ROM or the Serial-Flash/PROM.

The following parameters assign the Hub Device access to memory:

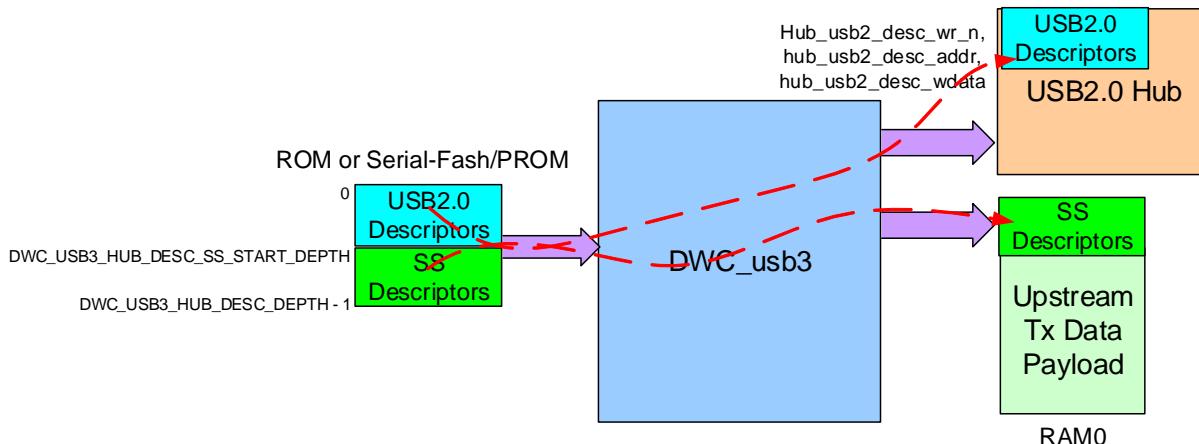
- DWC_USB3_HUB_DESC_DEPTH defines the depth of the ROM or the Serial-Flash/PROM.
On power-on, this parameter specifies the number of memory locations accessed by the hub devices.
- DWC_USB3_HUB_DESC_SS_START_DEPTH defines the SS-descriptor starting address.
All memory locations smaller than this parameter value are accessed by the USB 2.0 hub. Sharing memory between the SS Hub and USB 2.0 Hub is enabled when this parameter is not 0.

Figure 8-8 displays a graphical representation of memory sharing between the USB 2.0 Hub and SS Hub.

Example: If $DWC_USB3_HUB_DESC_DEPTH=128$ and $DWC_USB3_HUB_DESC_SS_START_DEPTH=64$:

- Locations 0-127 are reserved for the Hubs
 - Locations 64-127 are reserved for the SS Hub.
 - Remaining locations (0-63) are reserved for the USB 2.0 Hub.
- On power-on, the DWC_usb3 controller reads all addresses from 0 to 127.
 - For addresses 0 to 63, it asserts hub_usb2_desc_wr_n, allowing a USB 2.0 controller to capture hub_usb2_desc_addr and hub_usb2_desc_wdata signals; DWC_usb3 ignores data on these addresses.
 - For addresses 64 to 127, data is written to RAM0 for Host descriptor access; hub_usb2_desc_wr_n is not asserted.

Figure 8-8 ROM Sharing between USB 2.0 Hub and SS Hub



8.6 Overriding Descriptor Data through Input Ports

When the Descriptor content is read and stored in to the internal RAM (from ROM or Serial Flash/PROM), the controller overwrites some of the data in the RAM, depending on the `hub_desc_override_params[160:0]` and the `hub_desc_override_ctrl[1:0]` ports. The `hub_desc_override_ctrl` fields are as follows:

- `hub_desc_override_ctrl[127:0] = GUID[127:0]`
- `hub_desc_override_ctrl[143:128] = VendorID[15:0]`
- `hub_desc_override_ctrl[159:144] = ProductID[15:0]`
- `hub_desc_override_ctrl[16] = PowerControl (1'b0 - Individual Power Switch, 1'b1 - Gang Power Switch)`

When `hub_desc_override_ctrl[0] == 1'b1`, the GUID field from the `hub_desc_override_ctrl` port is used instead of the data from the ROM/Serial-Flash/PROM.

When `hub_desc_override_ctrl[1] == 1'b1`, the VendorID, ProductID, and PowerControl fields from the `hub_desc_override_ctrl` port is used instead of the data from the ROM/Serial-Flash/PROM.

8.7 Hub RAM Requirements

The DWC_usb3 hub requires two synchronous 2-port RAMs of each 33-bit data width running at 125 MHz. The port-1 of RAM0 and RAM1 is a read-only port, and port-2 is a write-only port. If your PHY has a 250-MHz clock, you can also choose to create 2-port RAMs using single port RAMs.

The RAM0 (upstream Tx Data Payload RAM) size is minimum 1300 + 256 bytes. The first 256 bytes are reserved in the RAM0 for storing SS descriptors. The descriptors are loaded from Serial-Flash/PROM or from ROM to RAM0 after reset. The RAM1 size is 1300 bytes minimum. Although USB 3.0 maximum packet size is only 1024, 1300 bytes are needed for optimal performance.

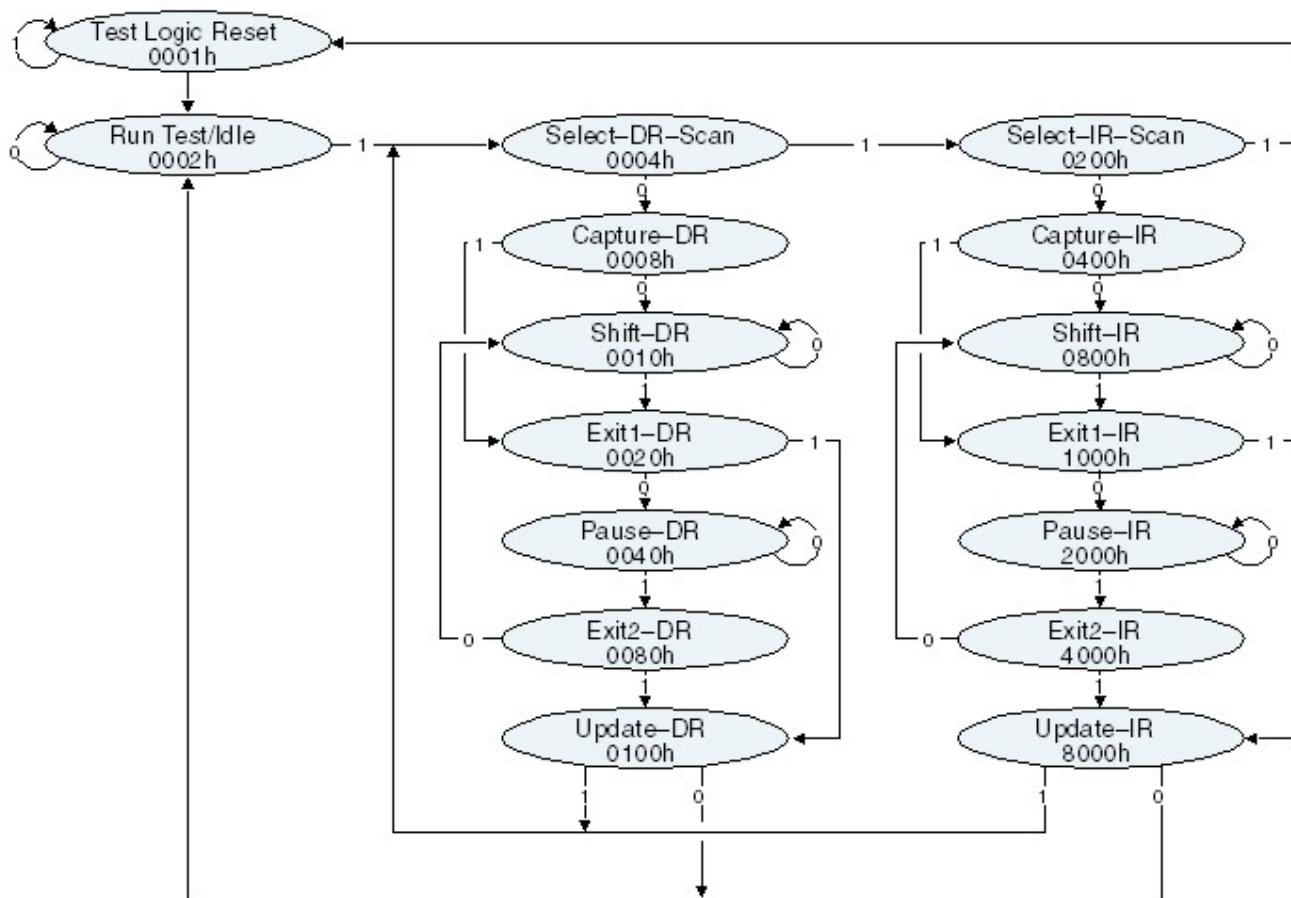
Having a larger Rx and Tx buffer may not improve the throughput because the hub forwards a packet as soon as it arrives. If there are frequent recoveries, the buffer can hold the data until recovery is completed. In the hardware, recoveries are very rare events. Because the RAM size is much smaller when compared to the area of multiple PHYs used in a hub design, it is recommended to have 2048 packet Rx and Tx buffers.

8.8 Debug JTAG

This section describes how the control and status bits specific to hub upstream and downstream ports and the hub controller endpoints are accessed using the JTAG interface for debug purposes. For descriptions of the ports in JTAG interface, refer to “[JTAG Interface Signals](#)” on page [428](#).

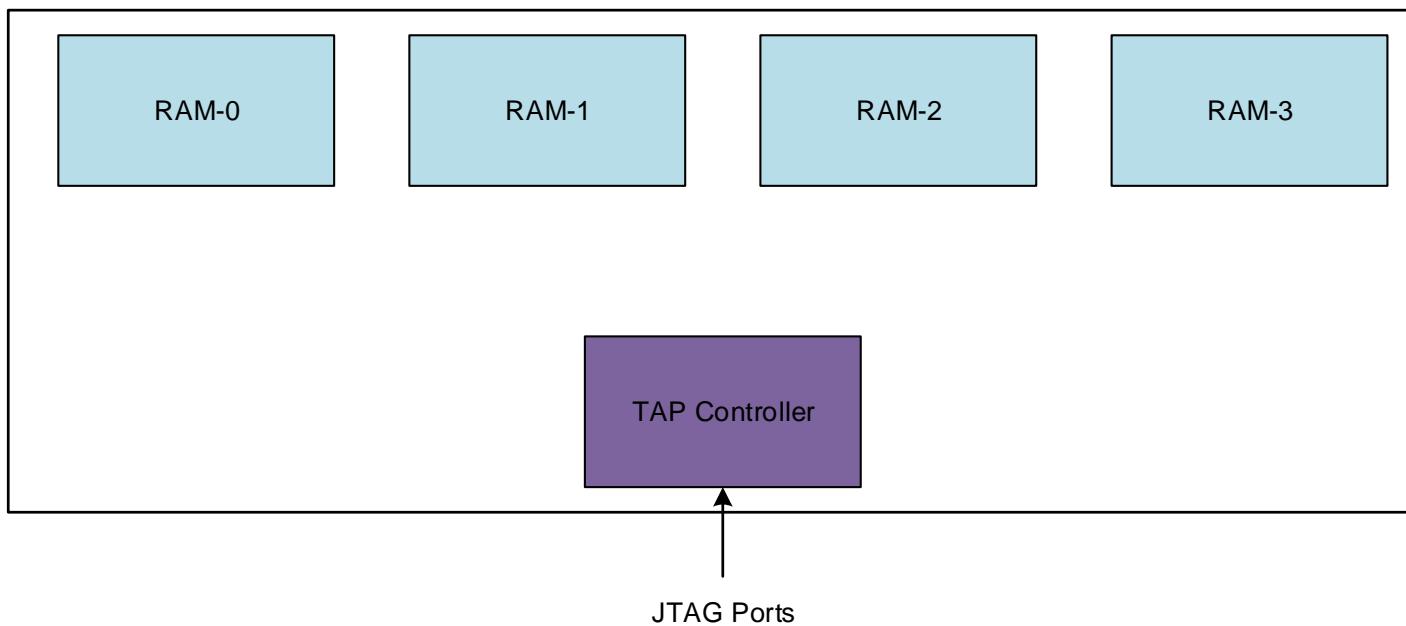
Figure 8-9 shows the TAP (Test Access Port) states and state transitions. The TAP state machine uses one-hot encoding.

Figure 8-9 TAP States and State Transitions



For more information on JTAG, see *IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Std. 1149.1. Figure 8-10 shows how the JTAG accesses all the resources. The JTAG can access the RAMS and internal registers at any time. However, the JTAG must only access the registers, Rx memory and Tx memory for debug purpose and not when Hub is in normal operation.

The ICE (In-Circuit Emulator) can explicitly set the `ice_stall_req` bit in the `DGBCTLSTS` register to stall the APP at any time, or it can program the debugger to stall the APP after a watchpoint condition, or once a trace is done.

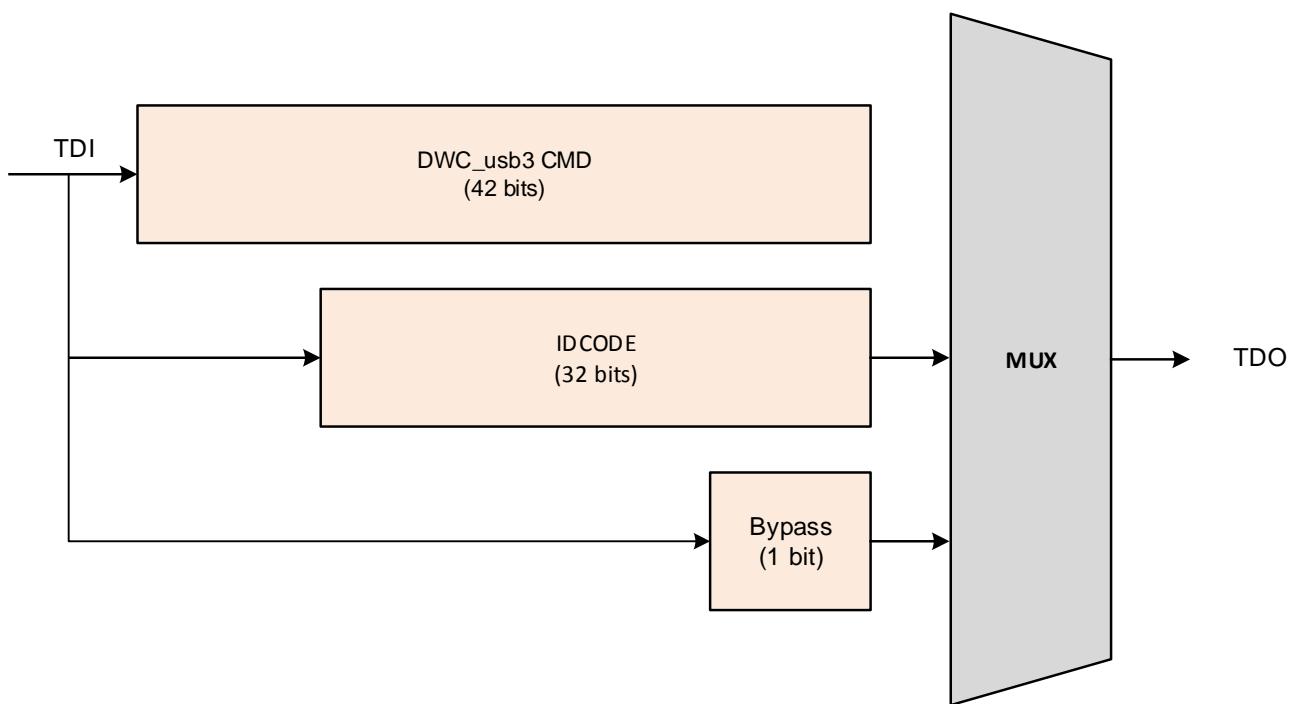
Figure 8-10 JTAG Resource Access

The TAP controller instructions are 3 bits wide. [Table 8-3](#) shows the TAP instructions used.

Table 8-3 TAP Instructions Used

Value	Description
3'b000	DWC_usb3 specific command
3'b001	IDCODE: Returns the TAP ID code (32'h14150_000) <ul style="list-style-type: none"> ■ Version: 4'h1 ■ Part: 16'h4150 ■ Manufacturer: 11'h0 ■ Valid: 1'b0 Default power-on reset command
3'b010	SAMPLE: Not used, will result in BYPASS
3'b111	BYPASS
Others	Not used, will result in Bypass

At power-on, the IDCODE is selected. During Capture-IR, a value of 3'b010 is loaded into the IR shift register. [Figure 8-11](#) on page 448 shows all test data registers, including the bypass.

Figure 8-11 Test Data Registers and Bypass

The DWC_usb3-specific command is a 42-bit register, through which writes and reads to different registers and memories can be achieved. DWC_usb3-specific SET_WRITE, SET_READ and EXECUTE_CMD commands are implemented.

[Table 8-4](#) shows the DWC_usb3 CMD Test Data Register fields

Table 8-4 DWC_usb3 Command Test Data Register

Field	Description
41:36	TAP Function Select[5:0] <ul style="list-style-type: none"> ■ Bit 0: RAM/Register Access ■ Bit 4 -1: Not Used
35:4	TAP Data or Address
3:0	TAP Command: <ul style="list-style-type: none"> ■ 3'b000: Execute command ■ 3'b001: Set write ■ 3'b010: Set read ■ Others: Unused

Writing to and Reading from Data RAM

To write to the data RAM, you must do the following:

1. Send a SET_WRITE command (bit[3:0]=3'b001) with the data RAM address you want to write to (bits[35:4]=Address), and select the data RAM (bits[41:36]=6'b000001).
2. Send an EXECUTE_CMD command with bits[35:4] containing the data to be written, which causes the actual write.

After the write, the address is auto-incremented by 4, enabling you to send another EXECUTE_CMD without having to send another SET_WRITE command.

Similarly, for reads, you must do the following:

1. Send SET_READ once.
2. You can then send multiple EXECUTE_CMD signals to read from the next memory/register location. Because the memories/registers are 32-bit, the address is auto-incremented by 4. The lower 2 bits of the address are always 0.

During the Capture-DR, bit[41:36] contains the previous tap_function_select, bits[35:4] contain the read data, and bits[3:0] contain the previous set_read or set_write command issued.

In Hub mode, only address bits [19:2] are used to access the internal resource. Address bit [31:20] must be 0. The workspace/sim/SOC_sim/vtb/hub/Test20_jtag.v test shows how to access different resources in the hub. The JTAG address map for accessing hub resources is given as follows:

- Address[19:16] == 0 && Address[15] == 0: Hub controller Registers listed in [Table 8-5 on page 450](#)
- Address[19:16] == 0 && Address[15] == 1: Hub Buffer-management Resources for debug purpose, listed at the end of src/hub/DWC_usb3_hub_bufm.v
- Address[19:16] >= 4 && Address[19:16] <= 5: RAM-0 bits [31:0]
- Address[19:16] >= 6 && Address[19:16] <= 7: RAM-0 bits {31'h0, bit[32]}
- Address[19:16] >= 8 && Address[19:16] <= 9: RAM-1 bits [31:0]
- Address[19:16] >= A && Address[19:16] <= B: RAM-1 bits {31'h0, bit[32]}

The JTAG can be used for the following:

- Setting the Link/PHY in loopback mode
- Setting the Link/PHY in compliance modes
- Probing the internal registers
- Probing the internal RAMs

[Table 8-5 on page 450](#) explains how the control and status bits specific to hub upstream and downstream ports and the hub controller endpoints are accessed using the JTAG interface for debug purpose.

Table 8-5 JTAG CSR Interface Mapping

Address Offset	Register	Bit mappings
0x00	Endpoint 0 Status	<ul style="list-style-type: none"> ■ 0 - IN Endpoint Active ■ 1 - IN Endpoint Nrdy ■ 2 - IN Endpoint Stall ■ 13:3 - IN MaxPktSize ■ 15:14 - IN DataSequenceNum ■ 16 - OUT Endpoint Active ■ 17 - OUT Endpoint Nrdy ■ 18 - OUT Endpoint Stall ■ 29:19 - OUT MaxPktSize ■ 31:30 - OUT DataSequenceNum.
0x04	Endpoint 1 status	<ul style="list-style-type: none"> ■ 0 - IN Endpoint Active ■ 1 - IN Endpoint Nrdy ■ 2 - IN Endpoint Stall ■ 13:3 - IN MaxPktSize ■ 17:14 - IN DataSequenceNum ■ 31:18 - Reserved
0x08	Hub Controller Params	<ul style="list-style-type: none"> ■ 0 - U1_Enable ■ 1 - U2_Enable ■ 2 - LTM_Enable ■ 4:3 - Functionsuspend ■ 7:5 - Reserved ■ 15:8 - RemoteWakeUpMask ■ 20:16 - missing_dpp_cnt ■ 31:21 - Reserved.
0x0C	Hub Status	<ul style="list-style-type: none"> ■ 0 - HUB_LOCAL_POWER ■ 1 - HUB_OVER_CURRENT ■ 15:2 - Reserved ■ 16 - C_HUB_LOCAL_POWER ■ 17 - C_HUB_OVER_CURRENT ■ 31:18 - Reserved

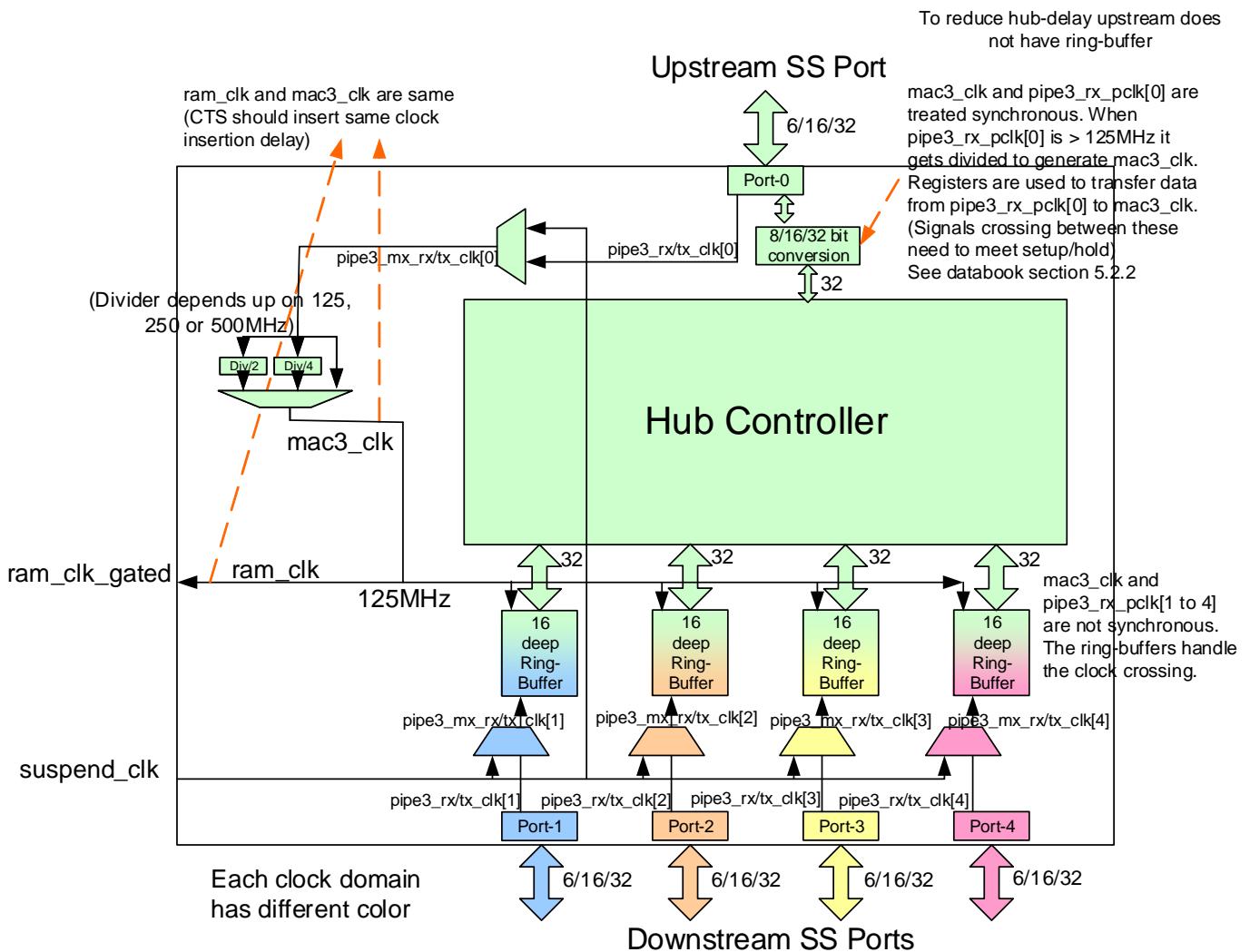
Address Offset	Register	Bit mappings
0x10	Downstream Port1 Status	
0x14	Downstream Port2 Status	
0x18	Downstream Port3 Status	
0x1C	Downstream Port4 Status	
0x20	Downstream Port5 Status	
0x24	Downstream Port6 Status	
0x28	Downstream Port7 Status	
0x2C	Downstream Port8 Status	
0x30	Downstream Port9 Status	
0x34	Downstream Port10 Status	
0x38	Downstream Port11 Status	
0x3C	Downstream Port12 Status	
0x40	Downstream Port13 Status	
0x44	Downstream Port14 Status	
0x48	Downstream Port15 Status	
0x50-0x88	Downstream Port1-15 PowerManagement and Control	<ul style="list-style-type: none"> ■ 0 - PORT_CONNECTION ■ 1 - PORT_ENABLE ■ 2 - Reserved ■ 3 - PORT_OVER_CURRENT ■ 4 - PORT_RESET ■ 8:5 - PORT_LINK_STATE ■ 9 - PORT_POWER ■ 12:10 - PORT_SPEED ■ 15:13 - RESERVED ■ 16 - C_PORT_CONNECTION ■ 18:17 - RESERVED ■ 19 - C_PORT_OVER_CURRENT ■ 20 - C_PORT_RESET ■ 21 - C_BH_PORT_RESET ■ 22 - C_PORT_LINK_STATE ■ 23 - C_PORT_CONFIG_ERROR ■ 31:24 - RemoteWakeupMask
		<ul style="list-style-type: none"> ■ 7:0 - U1_Timeout ■ 15:8 - U2_Timeout ■ 16 - ForceLinkPM accept ■ 31:17 Reserved.

8.9 Hub Controller Clock Scheme

The following diagram shows the Hub Controller clock scheme. The upstream clock is used as the main clock in the design. When all downstream ports go in to P3 and the upstream is expected to go in to P3, the upstream clock is always on when any one of the downstream ports is on.

Each downstream port is treated asynchronous to the hub controller and a free running ring-buffer on each port takes care of the clock crossing between the downstream ports and hub controller. This allows us to validate the hub with multiple single-port hubs during hardware validation and still meet the maximum Hub delay requirement.

Figure 8-12 USB 3.0 Hub



8.10 Hub Controller Timing

- Hub Delay: 400 ns requirement

The Hub delay = Hub Digital Controller Delay + PHY Rx Delay + PHY Tx Delay = ~200 ns (with ring-buffer) + PHY Delays

The USB Specification limit is 400 ns and because the typical combined PHY Rx and Tx delays are in the range of 50 ns to 100 ns, this meets the specification.

- Hub Latency tolerance: ±12 ns requirement

In the Hub's multi-port PHY design, it is recommended to use common PLL or spread-spectrum clock for the upstream and all the downstream ports.

The expected Latency tolerance with Hub controller Ring-buffer = PHY Rx jitter + PHY Tx jitter + Hub controller Ring-buffer jitter

= 1 ns (Typical combined PHY Rx and Tx jitter tolerance is less than 1 ns) + 2 SKIP symbol timing (4 ns) rounded to mac_clk period

= 1 ns + 8 ns = ~9 ns.

Cross check whether multiple single port PHY meets the 12 ns latency tolerance or you would need a multi-port PHY which has common PLL or spread-spectrum clock to all ports.

- Hub Header Packet Decode Delay: 400 ns requirement

- a. When the bit 12 of GUSB3PIPECTL_INIT is set to 0 (Default behavior to meet 20 ns LFPS to SS delay requirement)

- Best Case (exit happens immediately and when in the process of P0-> P1/P2 change) - 350 ns
- Worst Case (exit happens when already in P1) - 600 ns (This is outside the specification limit)
- Worst Case (exit happens when already in P2) - 900 ns (This is outside the specification limit)

- b. When bit 12 of GUSB3PIPECTL_INIT is set to 1

- Best Case (exit happens immediately and when in the process of p0-> P1/P2 change) - 100 ns
- Worst Case (exit happens when already in P1) - 396 ns
- Worst Case (exit happens when already in P2) - 548 ns

The above timings also depend on the PHY used and on the different modes of the PHY. The above numbers are based on the Synopsys PHY.

8.11 Vendor Control Interface

To enable Vendor Control Interface (VCI), select “Enable Vendor Control Command Interface” in coreConsultant (parameter DWC_USB3_HUB_ENABLE_VCI = 1). When VCI is enabled, the hub does not STALL vendor control commands. It passes these commands through the VCI for external decoding. For details on vendor command interface signals, refer to the “Hub Interface Signals” section in the Signals chapter.

This section discusses the DWC_usb3 Hub controller’s support for decoding the vendor commands. It describes the following:

- “Control Transfer Phases”
- “Control Transfer Scenarios” on page [455](#)
- “Handling Vendor Command Control Transfers” on page [462](#)
- “Timing Diagrams” on page [463](#)

8.11.1 Control Transfer Phases

This section discusses the various control transfer phases.

8.11.1.1 SETUP Phase

The application must allocate buffers to handle 8 bytes of SETUP packet at any time. SETUP packet is always ACK’ed by the controller. On receiving a VCI setup packet, internal NAK bits for IN and OUT directions are set by the controller internally. The application re-evaluates the vci_rx_space_avail and npi_tx_pkt_count on receiving a SETUP packet. The npi_tx_pkt_count is made 0 with the reception of a SETUP packet. This gives the application the flexibility to STALL the Status phase later if it receives a wrong vendor command.

8.11.1.2 Control Write Data Phase (3-Stage Control Transfer)

On receiving a VCI SETUP packet, the application evaluates the vci_rx_space_avail and update the value. Based on the value of vci_rx_space_avail, controller puts the endpoint in flow control. If there is not sufficient space, the data is discarded, there is no vci_rx_data_push, and the vci_rx_status corresponding to vci_rx_status_vld indicates NOT_OK status. If vci_rx_space_avail indicates sufficient space, then vci_rx_data_push is asserted and the vci_rx_status corresponding to vci_rx_status_vld indicates the status of the packet received. If the status is NOT_OK, then the application must flush the received packet.

8.11.1.3 Control Read Data Phase (3-Stage Control Transfer)

After receiving a VCI SETUP packet for 3stage control read, the application prepares the packet and updates vci_tx_pkt_count and vci_tx_pkt_status. Once this is done, the application waits for vci_tx_ack_vld with vci_tx_ack_status[21] indicating the completion ACK. For transmitting a zero length packet, the packet count is set to 1, and the size (vci_tx_pkt_status[10:0] set to 0. The application rewinds the fifo pointers if it receives a vci_tx_ack_vld with vci_tx_ack_status[20] set, indicating that the host is retrying the packet.

8.11.1.4 Control Status Phase

For confirming a status phase, the application sets the vci_tx_pkt_count to 1 with the vci_tx_pkt_status[12] set to 1.



- The external logic (outside DWC_usb3 hub), interfacing to VC, deals with any unsupported vendor commands, and STALLs them appropriately.
- The external logic takes care of STALLing any abnormal control transfers. An example of abnormal control transfer is host sending more DP than what is specified in the wLength field of the SETUP packet.

8.11.2 Control Transfer Scenarios

This section explains in detail how the DWC_usb3 controller handles the various vendor command control transfer scenarios. The various columns in the table are as follows.

- Host: This column specifies the action from the USB Host or the Hub downstream port, to which the upstream port of the hub is connected to.
- DUT: This column specifies the actions of the hub to the vendor command interface, and also within the main sub modules within the hub.
- PTL: This column specifies the actions of the PTL sub module of the HUB.
- VCI Interface: This column specifies the actions of the vendor command logic that is part of the hub controller.
- VC Application: This column specifies the actions of the module that is external to the DWC_usb3, and connected to the VC Interface.

8.11.2.1 3-Stage Control Write Without Flow Control

Table 8-6 details the 3-stage control write without flow control.

Table 8-6 3-Stage Control Write Without Flow Control

Step	Host	DUT		VC Application
		PTL	VC Interface	
1	Host sends SETUP packet DPH	Asserts ptr3_msg_valid, ptr3_msg for DPH	Asserts vci_rx_cmd_valid, vci_rx_cmd_param for DPH indicating SETUP (bit 20)	Application knows that it is a SETUP packet
2	Host sends DPP for SETUP	Asserts ptr3_data_valid, ptr3_data for data for the SETUP packet.	Asserts vci_rx_data_push, vci_rx_data for the SETUP data	Application takes in the SETUP packet
3		Sends ACK TP to the host acknowledging SETUP packet. Asserts ptr3_msg_valid, ptr3_msg	Asserts vci_rx_status_vld and vci_rx_status indicating the validity of the SETUP packet	Application decodes the SETUP, ignores the non-VC SETUP commands. After decoding SETUP packet, updates rx-fifo_space

Step	Host	DUT		VC Application
		PTL	VC Interface	
4	Host Sends DPH with SETUP =0	PTL asserts ptr3_msg_valid, mtr3_msg	DUT asserts vci_rx_cmd_valid, vci_rx_cmd_param to application	Application knows from this that the host is starting the control write data phase.
5	Host follows DPH with DPP for data packet	If Rx fifo space is available (vci_rx_fifo_space) PTL asserts ptr3_data_valid, ptr3_data	DUT asserts vci_rx_data_push, vci_rx_data provided RXFIFO space was available. Sets out_csr_nrdy, and in_csr_nrdy. These will be cleared based on non-zero values of rx_fifo_space and tx_pkt_count	Application receives the Data packet
6		If the CRC was good, send out ACK on the USB. Also asserts status as OK on the ptr3_msg_valid and ptr3_msg	Asserts vci_rx_sts_valid, vci_rx_status indicating that the received packet is OK	Application uses this information to take the packet (ACK) or flush it otherwise. If the received size of the packet is more than the wlength filed in the SETUP, then application should set the stall
7	If the Data phase has more than one packets, then Steps 4,5,6 will repeat			
8	Host sends TP indicating STATUS stage	Asserts ptr3_msg_valid,ptr3_ms g indicating status stage	Asserts vci_tx_ack_valid, vci_tx_status indicating STATUS	Application knows it is the STATUS phase. Updates vci_tx_pkt_count with 1. Also updates vci_tx_pkt_status indicating status data. If the application need to STALL, then set the stall input
9		Asserts ptt3_cmd_request,cmd_type, since the NRDY bit is already cleared		
10			Asserts ptt3_cmd_done and ptt3_txinfo to PTL. Asserts vci_tx_data pop to the application	
11		Sends out ACK to USB		

8.11.2.2 3-Stage Control Write With Flow Control

[Table 8-7](#) details the 3-stage control write with flow control.

Table 8-7 3-Stage Control Write With Flow Control

Step	Host	DUT		VC Application
		PTL	VC Interface	
		ptr3_msg_valid, ptr3_msg for DPH	Asserts vci_rx_cmd_valid, vci_rx_cmd_param for DPH indicating SETUP	
1	Host Sends SETUP packet DPH	Asserts ptr3_data_valid, ptr3_data for data for the SETUP packet	Asserts vci_rx_data_push, vci_rx_data for the SETUP data	Application takes in the SETUP packet
2	Host sends DPP for SETUP	Sends ACK TP to the host acknowledging SETUP packet. Asserts ptr3_msg_valid, ptr3_msg	Asserts vci_rx_status_vld and vci_rx_status indicating the validity of the SETUP packet. Sets out_csr_nrdy, and in_csr_nrdy. These will be cleared based on non-zero values of rx_fifo_space and tx_pkt_count	Application decodes the SETUP, ignores the non-VC SETUP commands. After decoding SETUP packet, updates rx-fifo_space to be 0 bytes (no space available)
3	Host Sends DPH with SETUP =0	PTL asserts ptr3_msg_valid, mtr3_msg indicating that NRDY will be send to host for this DPH	DUT asserts vci_rx_cmd_valid, vci_rx_cmd_param to application. Sets out_csr_nrdy, and in_csr_nrdy. These will be cleared based on non-zero values of rx_fifo_space and tx_pkt_count	
4	Host follows DPH with DPP			
5		Sends out NRDY on the USB to the host. Asserts ptr3_msg_valid, ptr3_ms g indication transaction status (NRDY). Writes CSR interface to set CSR NAK	Sets out_csr_nrdy. Asserts vci_rx_status_valid, vci_rx_status indicating Non ACK	Application checks to see if there is enough fifo space
6	Time elapses.			
7				Application updates vci_rx_fifo_space to be >512
8				

Step	Host	DUT		VC Application
		PTL	VC Interface	
9			On seeing out_csr_nrdy HIGH and vci_rx_fifo_space >512, asserts ptc3 request to PTL to send ERDY	
10	Host Sends DPH with SETUP =0	PTL asserts ptr3_msg_valid, mtr3_msg	DUT asserts vci_rx_cmd_valid, vci_rx_cmd_param to application	Application knows from this that the host is continuing with the data phase.
11	Host Sends DPH with SETUP =0	PTL asserts ptr3_msg_valid, mtr3_msg	DUT asserts vci_rx_cmd_valid, vci_rx_cmd_param to application	Application knows from this that the host is continuing with the data phase
Rest of the steps are similar to steps 5-11 of previous case.				

8.11.2.3 3-Stage Control Transfer Write for Standard Commands

[Table 8-8](#) details the 3-stage control transfer write for standard commands.

Table 8-8 3-Stage Control Transfer Write for Standard Commands

Step	Host	DUT		VC Application
		PTL	VC Interface	
		Asserts ptr3_msg_valid, ptr3_msg for DPH	Asserts vci_rx_cmd_valid, vci_rx_cmd_param for DPH indicating SETUP	
1	Host Sends SETUP packet DPH	Asserts ptr3_data_valid, ptr3_data for data for the SETUP packet	Asserts vci_rx_data_push, vci_rx_data for the SETUP data	Application takes in the SETUP packet
3		Sends ACK TP to the host acknowledging SETUP packet. Asserts ptr3_msg_valid, ptr3_msg	Not a VC command. So VC logic does not do anything	Application decodes the SETUP, finds that it is a Standard command, and waits for the next rx_cmd_valid indicating SETUP
4	Host Sends DPH with SETUP =0	PTL asserts ptr3_msg_valid, mtr3_msg	DUT asserts vci_rx_cmd_valid, vci_rx_cmd_param to application	Application ignores it since it is not a SETUP
5	Host follows DPH with DPP for data packet	PTL asserts ptr3_data_valid, ptr3_data	DUT asserts vci_rx_data_push, vci_rx_data provided Rx FIFO space was available	Application ignores the data push
6		If the CRC was good, send out ACK on the USB. Also asserts status as OK on the ptr3_msg_valid and ptr3_msg	Asserts vci_rx_sts_valid, vci_rx_status indicating that the received packet is OK	Application ignores it
7	If the Data phase has more than one packets, then Steps 4,5,6 will repeat			
8	Host sends TP indicating STATUS stage	Asserts ptr3_msg_valid, ptr3_msg indicating status stage	Asserts vci_tx_ack_valid, vci_tx_status indicating STATUS	Application ignores it
9		Asserts ptt3_cmd_request, cmd_type, since the NRDY bit is already cleared	Ignores the request	
10		Sends out ACK to USB		

8.11.2.4 3-Stage Control Read Without Flow Control

Table 8-9 details the 3-stage control read without flow control.

Table 8-9 3-Stage Control Read Without Flow Control

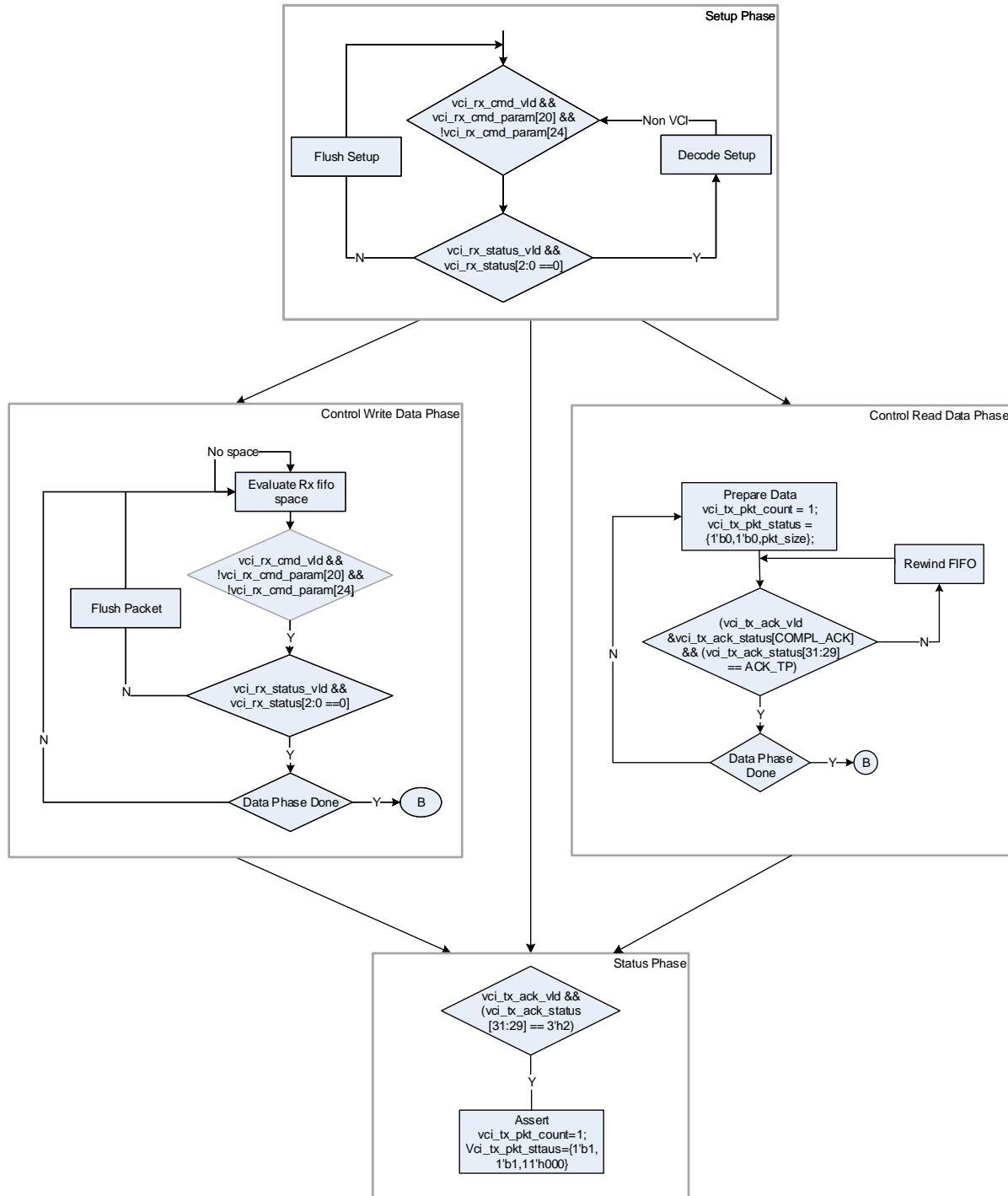
Step	Host	DUT		VC Application
		PTL	VC Interface	
		ptr3_msg_valid, ptr3_msg for DPH	Asserts vci_rx_cmd_valid, vci_rx_cmd_param for DPH indicating SETUP	
1	Host Sends SETUP packet DPH	Asserts ptr3_data_valid, ptr3_data for data for the SETUP packet	Asserts vci_rx_data_push, vci_rx_data for the SETUP data	Application takes in the SETUP packet
2	Host sends DPP for SETUP	Sends ACK TP to the host acknowledging SETUP packet. Asserts ptr3_msg_valid, ptr3_msg	vci_rx_status_vld and vci_rx_status indicating the validity of the SETUP packet. Sets out_csr_nrdy, and in_csr_nrdy. These will be cleared based on non-zero values of rx_fifo_space and tx_pkt_count.	Application decodes the SETUP, ignores the non-VC SETUP commands. After decoding SETUP packet, updates rx-fifo_space, and tx_pkt_count
4	Host sends ACK TP for the Data phase.	Asserts ptr3_msg_valid and ptr3_msg indicating TP-ACK	Asserts vci_tx_sts_valid and vci_tx_ack_status. Bit 21 of vci_tx_ack_status indicates whether it is a requesting/terminating ACK	Application knows the IN data phase has started
5		Asserts ptt3_cmd_request, ptt3_cmd_type indicating data packet request	Waits for 2 clocks and then checks the tx_pkt_count and tx_pkt_status. Asserts ptt3_cmd_done. Asserts txinfo with OK/not_ok status (OK in this case because tx_pkt_count is >0). Also updates pkt_size in the txinfo from tx_pkt_status. Asserts ptt3_data_valid, ptt3_data (ptt3_data is from vci_tx_data)	
6		Asserts ptt3_data_accept. Transmits the packet to USB	Asserts vci_tx_data_pop. This is the same as ptt3_data_accept	Application continues to provide each DWORD for the packet
7	Host sends ACK-TP (with NUMP=0/1)	Asserts ptr3_msg_valid, ptr3_msg indicating ACK (can be terminating ACK)	Asserts vci_tx_sts_valid, vci_tx_status indicating that the host has accepted the packet	Application advances the fifo pointer
8	If there are more data packets pending in the data stage, then steps 4,5,6,7 will repeat.			

Step	Host	DUT		VC Application
		PTL	VC Interface	
9	Host sends TP-STATUS indicating status phase	Asserts ptr3_msg_valid,ptr3_ms g indicating status stage	Asserts vci_tx_ack_valid, vci_tx_status indicating STATUS	Application knows it is the STATUS phase. Updates vci_tx_pkt_count with 1. If the application need to STALL, then set the stall input
10		Sends out TP-ACK for the Status		

8.11.3 Handling Vendor Command Control Transfers

[Figure 8-13](#) gives recommendations on the logic that can be used by the external interface for decoding the vendor commands.

Figure 8-13 Handling Vendor Command Control Transfers Flow Chart



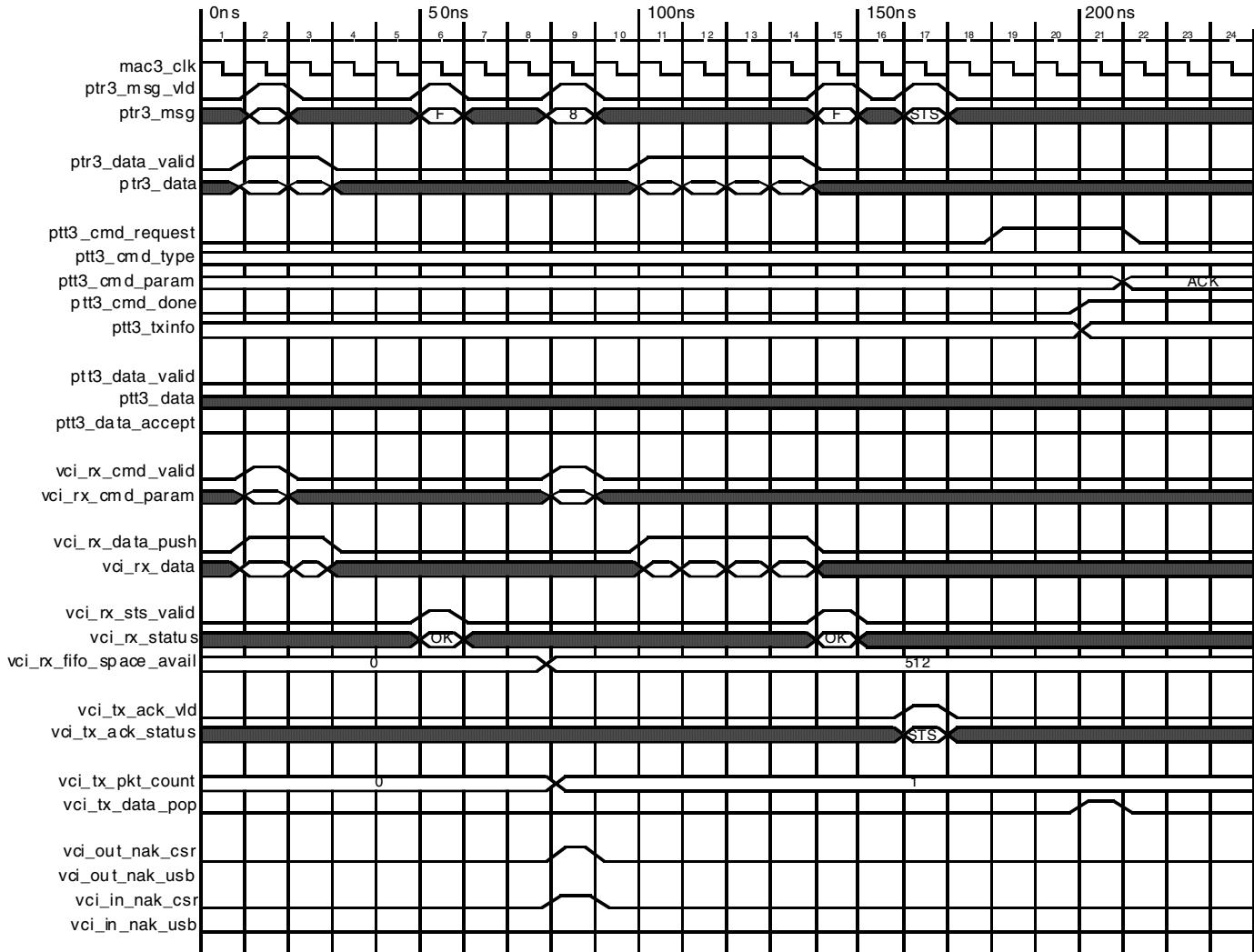
8.11.4 Timing Diagrams

This section gives examples of the interfaces for various scenarios.

8.11.4.1 3-Stage VC Control Write

Figure 8-14 shows the timing diagram for a 3-stage VC control write.

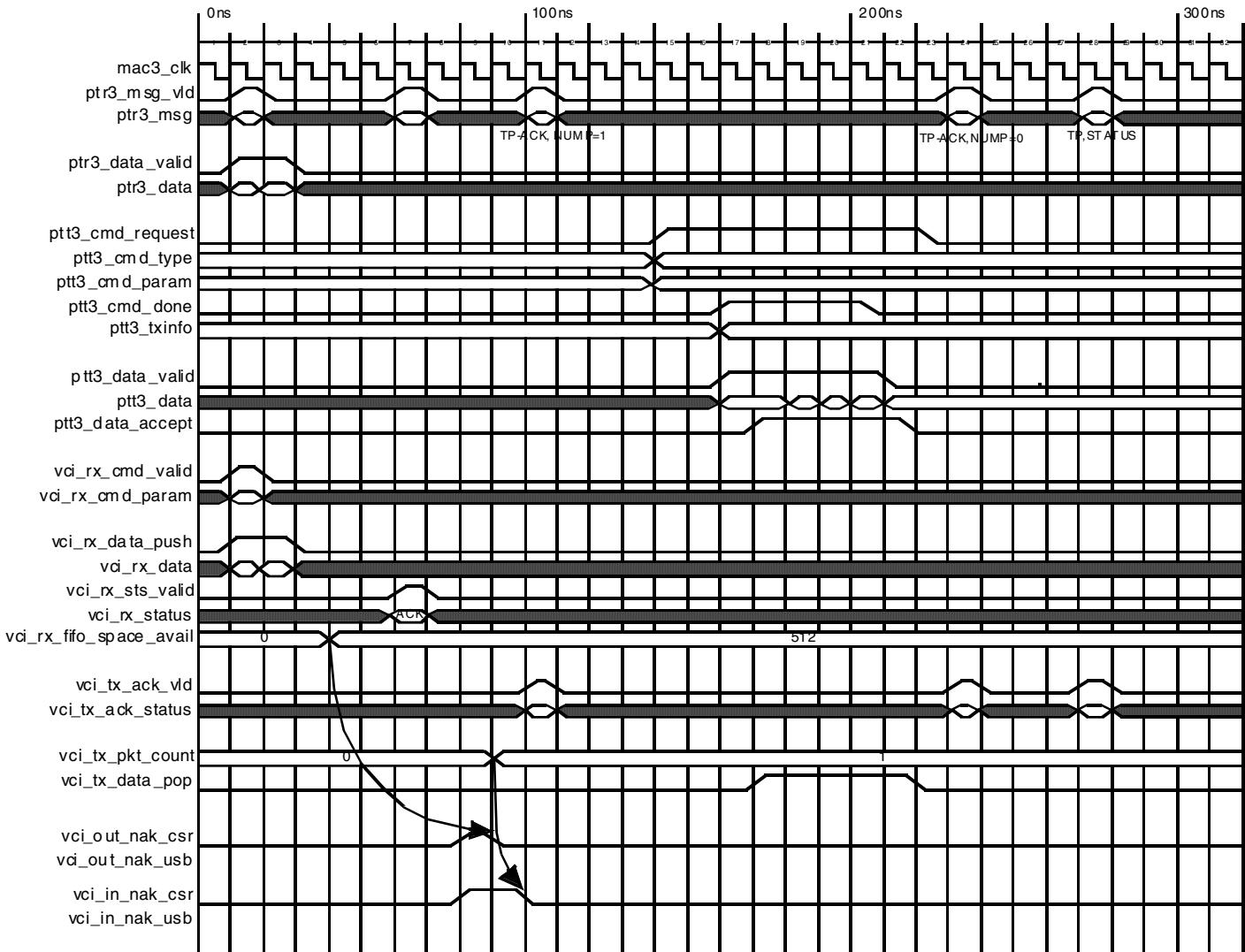
Figure 8-14 3-Stage VC Control Write



8.11.4.2 3-Stage VC Control Read

Figure 8-15 shows the timing diagram for a 3-stage VC control read.

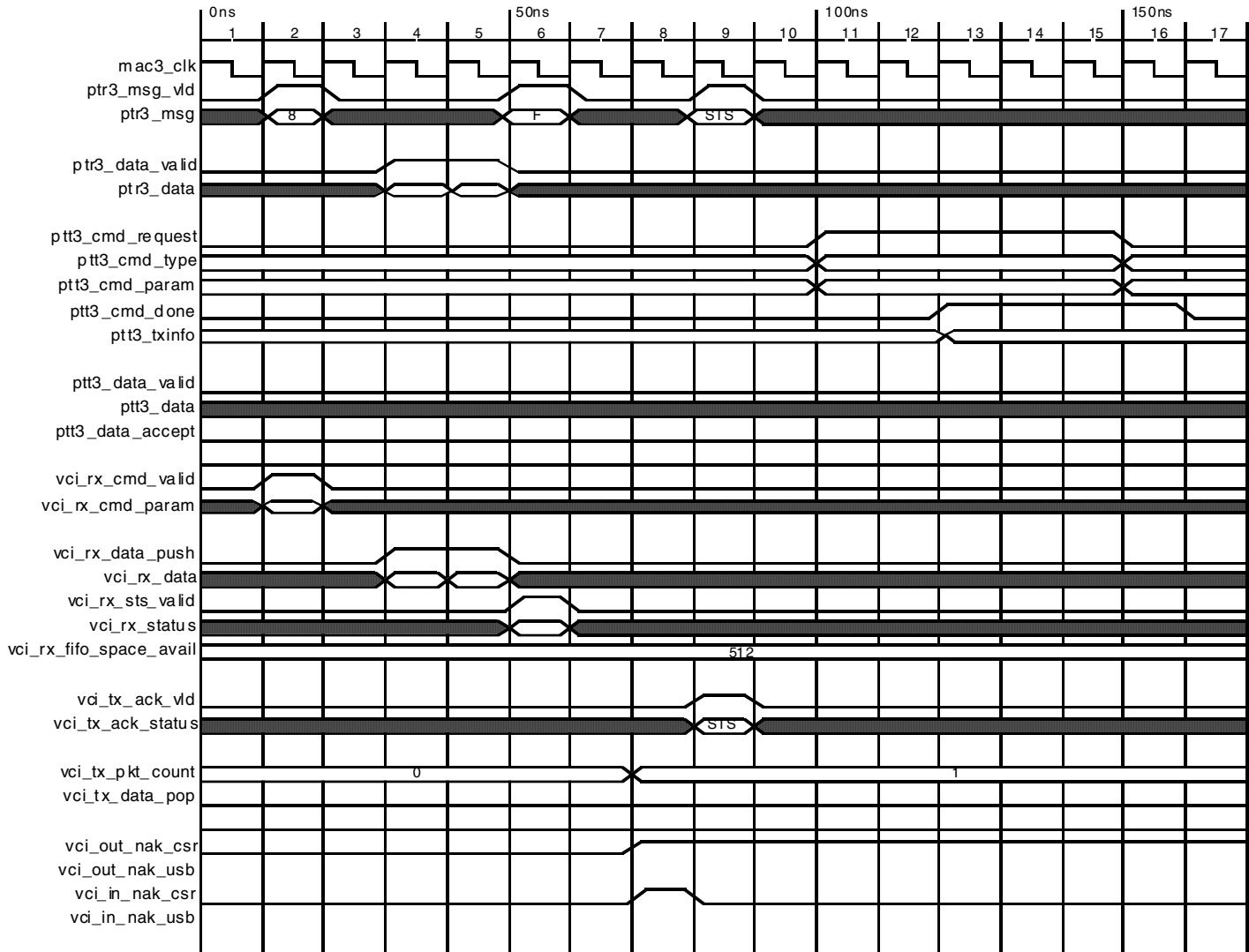
Figure 8-15 3-Stage VC Control Read



8.11.4.3 2-Stage VC Control Transfer

Figure 8-16 shows the timing diagram for a 2-stage VC control transfer.

Figure 8-16 2-Stage VC Control Transfer



Battery Charger

This chapter describes the battery charger functionality and interrupt hierarchy.

- “[Battery Charger](#)” on page [468](#)
- “[Interrupt Hierarchy](#)” on page [478](#)

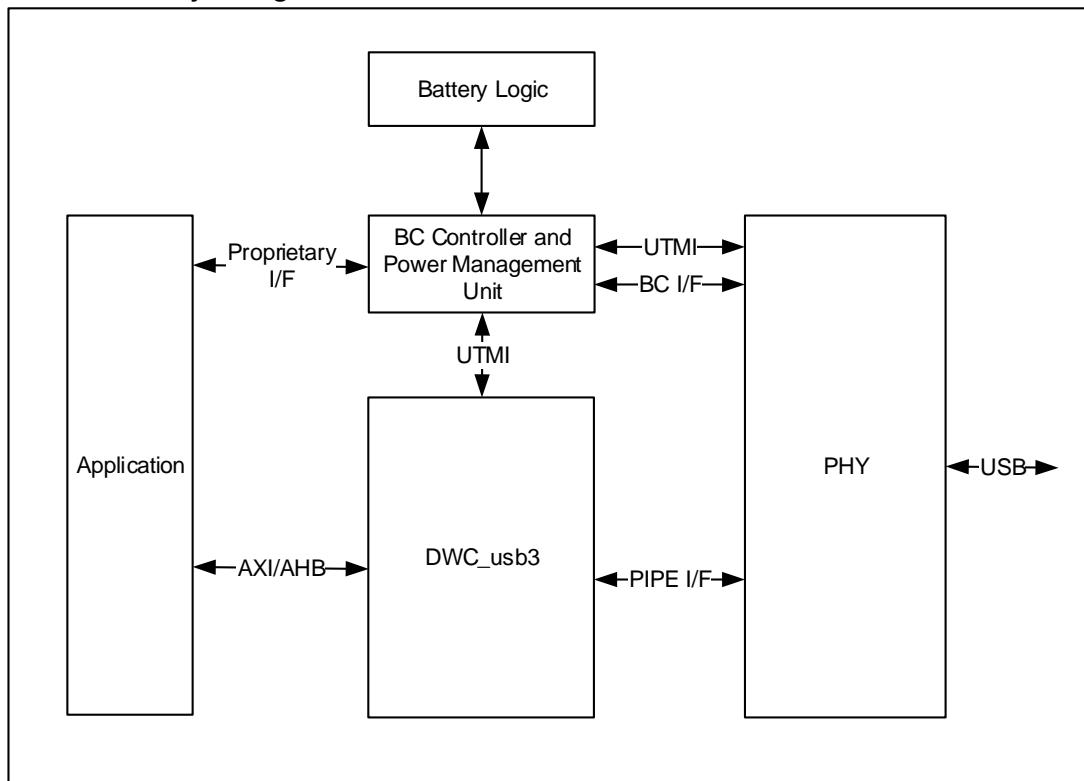
9.1 Battery Charger

This section describes the Battery Charger (BC) functionality of the DWC_usb3 controller.

9.1.1 BC (ACA) Functions

The following figure shows implementation of an external BC control logic.

Figure 9-1 External Battery Charger



A typical Synopsys PHY that supports BC feature has the following BC pins (indicated as BC I/F in the Figure 9-1). The direction of the pins listed in [Table 9-1](#) is with respect to the PHY.

Table 9-1 Sample BC PHY Pin List

Pin Name	Direction	Description
CHRGSEL0	I	BC Source Select
VDATSRCENB0	I	BC Sourcing Select
VDATDETENB0	I	BC Attach/Connect Detection Enable
DCDENB0	I	Data Contact Detection Enable
ACAENB0	I	ACA ID_OTG Pin Resistance Detection Enable
RIDFLOAT0, RIDGND0, RIDA0, RIDB0, RIDC0	O	ACA ID_OTG Pin Resistance Indicator
CHGDET0	O	BC Detection Output

For a Portable Device (PD), a BC controller can implement the following detecting mechanisms in the given sequence in conjunction with PHY.

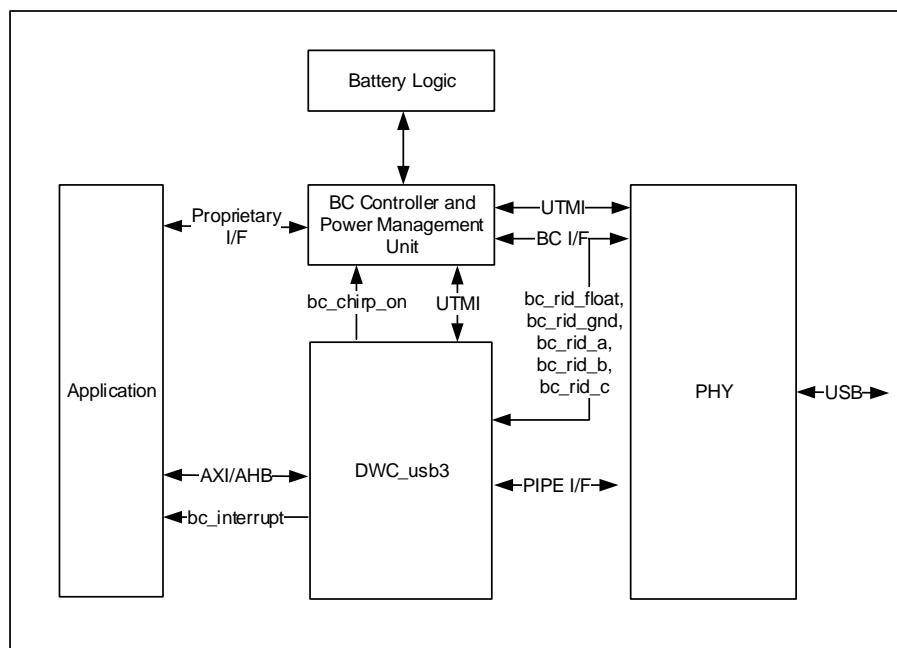
Table 9-2 BC Detection Mechanism

Serial Number	Detection Type	Description
1	VBUS Detect	PD Compares VBUS > VOTG_SESS_VLD
2	Data Contact Detect (DCD)	May or may not be implemented by PD. If DCD is not implemented by PD, the BC bypasses this detection scheme.
3	Primary Detection	This is used to distinguish between Standard Downstream Port (SDP) and different types of charging ports
4	Secondary Detection	This is used to distinguish between Dedicated Charging Port (DCP) and Charging Downstream Port (CDP).
5	ACA Detection	Only PDs having u-AB receptacle can implement this mechanism. PDs that support ACA detections should also implement Good Battery Algorithm.

9.1.2 Battery Charger Operation

Figure 9-2 shows the battery charger interconnections.

Figure 9-2 Battery Charger Interconnections

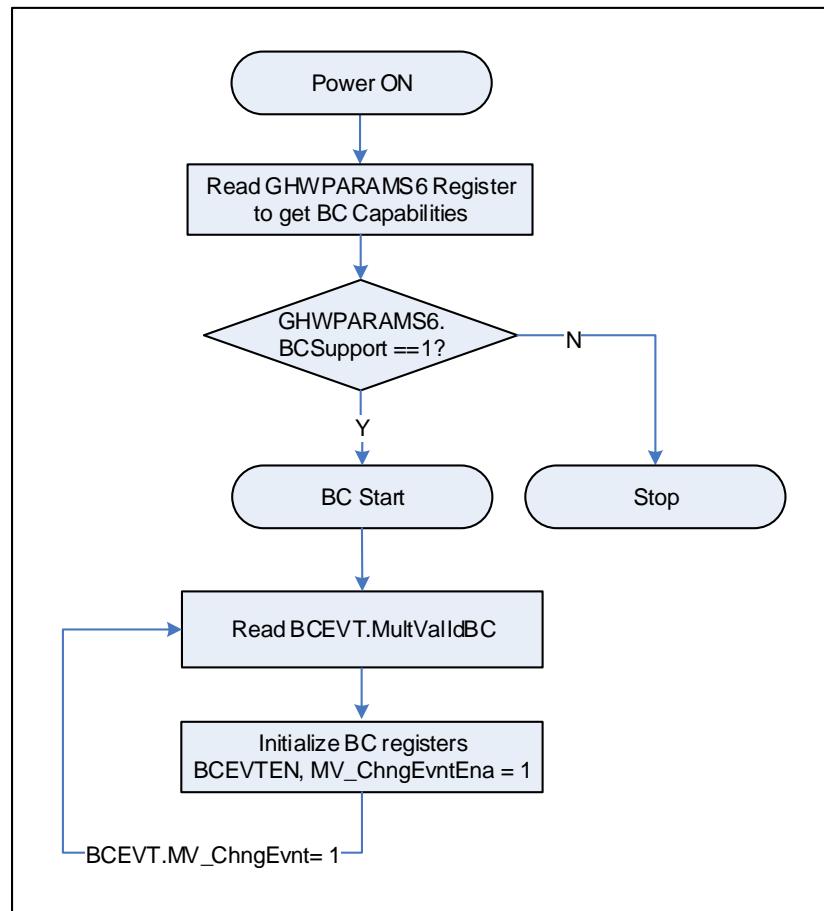


The bc_chirp_on signal shown in Figure 9-2 is output from the DWC_usb3 controller as a device to indicate an imminent chirp. The bc_chirp_on signal is not asserted when the DWC_usb3 controller is functioning as a host. For a detailed behavior of the bc_chirp_on signal, see “[The bc_chirp_on Signal Behavior](#)” on page 473.

The bc_rid_* pins can be optionally passed through DWC_usb3 in case of PD applications when interfaced to ACA. Note that if the BC Controller and Power Management Unit is capable of handling the bc_rid_* pin changes, there is no need to pass the bc_rid_* pins through the DWC_usb3 controller.

If the bc_rid_* pins are passed through the DWC_usb3 controller, the bc_rid_* pin values are passed on as status information to the application via BCEVT register in DWC_usb3 controller. Any change in the bc_rid_* pins is passed on to application as a bc_interrupt. On reception of a bc_interrupt, the application is expected to take necessary action by programming the BC controller and Power Management Unit as shown in [Figure 9-3](#).

Figure 9-3 Flow Diagram for Handling bc_rid_* Changes



9.1.3 BC Support for ULPI Interface

In the discussion so far, the BC controller controls the BC functionality in the PHY through BC interface. But in the case of ULPI PHY, BC functionality can be controlled by the application in either of the following ways:

- Using the proprietary interface through the BC controller and power management unit
- Exclusively through the ULPI interface

9.1.3.1 BC Functionality Controlled Through the BC Controller

In this case, there is an external BC controller. The application controls the BC functionality directly using the proprietary interface through the BC controller and power management unit. The ULPI PHY may implement new vendor specific BC ULPI registers through which the application can know about the status of the BC detection. This register can have read-only access as shown in [Table 9-3](#).

Table 9-3 Sample BC ULPI PHY Status Register Bits

Pin Name	Access	Description
RIDFLOAT0, RIDGND0, RIDA0, RIDB0, RIDC0	RO	ACA ID_OTG Pin Resistance Indicator. Valid only if the ACAENB0 is set.
CHGDET0	RO	BC Detection Output. Valid only when VDATSRCENB0 & VDATDETENB0 are set.

The interconnections are the same as shown in the case of UTMI PHY except that there is no IDDIG pin connected as an input to the controller. The controller uses RX_CMD to obtain the ID value.

9.1.3.2 BC Functionality Controlled Exclusively Through ULPI Interface

In this case, there is no BC controller. The BC functionality is controlled by vendor control access of the vendor-specific BC ULPI registers. Using these registers, BC detection can be enabled or disabled. Note that a new register that can be used to control BC functionality must be implemented in the PHY. To know whether your PHY supports BC register, check with your PHY vendor.

Assumption: If the PHY supports BC, then the PHY will not start the BC detection or any other detection until the BC ULPI PHY register is accessed by the application. [Table 9-4](#) shows an example of BC ULPI PHY register bits.

Table 9-4 Sample BC ULPI PHY Register Bits

Pin Name	Access	Description
CHRGSEL0	R/W	BC Source Select
VDATSRCENB0	R/W	BC Sourcing Select
VDATDETENB0	R/W	BC Attach/Connect Detection Enable
DCDENB0	R/W	Data Contact Detection Enable
ACAENB0	R/W	ACA ID_OTG Pin Resistance Detection Enable

Pin Name	Access	Description
RIDFLOAT0, RIDGND0, RIDA0, RIDB0, RIDC0	RO	ACA ID_OTG Pin Resistance Indicator Valid only if the ACAENB0 is set.
CHGDET0	RO	BC Detection Output Valid only when VDATSRCENB0 & VDATDETENB0 are set.
BCDONE	RO	This bit is set by the PHY when the BC Detection is done.

The BCDONE bit is used to monitor the status of the PHY. This bit when set indicates that the BC detection mechanism is completed by the PHY and the controller can use the PHY now.

The control of the BC functionality is shown in [Table 9-5](#). This example is based on the BC functionality implementation of SNPS UTMI PHY. By controlling three inputs (VDATSRCENB0, VDATDETENB0, and CHRGSEL0) and monitoring an output (CHGDET0), the type of port to which the PD (SDP, CDP, DCP) is connected can be determined. For more details on the description of these signals, refer to Synopsys DesignWare Cores USB 2.0 picoPHY One-Port Databook.

DCD detection can be enabled or disabled by writing to the DCDENB0 bit of the BC ULPI register. Similarly, ACA detection can be enabled or disabled by writing to the ACAENB0 bit of the BC ULPI register. Note that the RID* outputs are valid only if ACA detection is enabled.

Table 9-5 BC Operations

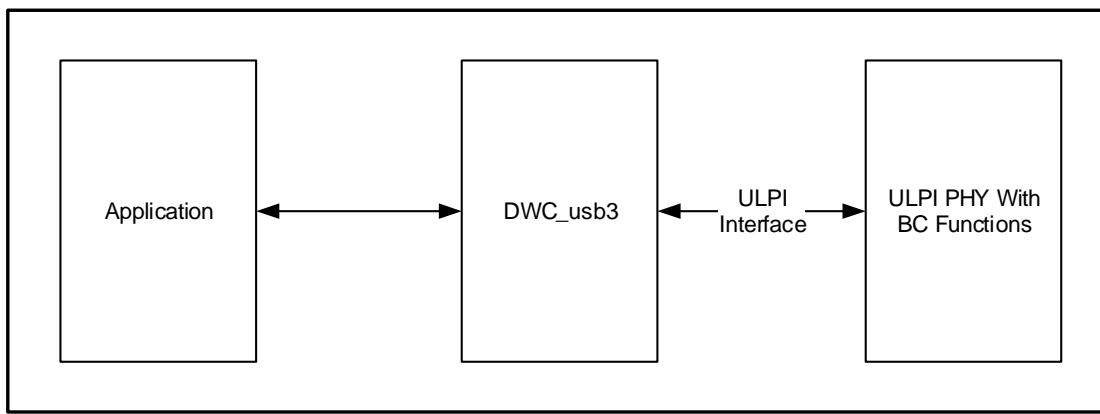
VDATSRCENB0	VDATDETENB0	CHRGSEL0	CHGDET0	Operation ^a
0	0	X	0	BC Detection is Disabled.
1	1	0	0	PHY connected to a standard downstream port.
1	1	0	1	PHY connected to a dedicated charging port or charging downstream port.
1	1	1	0	PHY connected to a charging downstream port.
1	1	1	1	PHY device connected to a dedicated charging port.

a. PHY is operating as a portable device.



As PHY has only ULPI interface, IDDIG pin may not be present at the output of the PHY. In such a case, program BCFG.IDDIG_SEL as 0. The controller uses RX_CMD to obtain the ID value.

The interconnections are shown [Figure 9-4](#). There is no external logic for controlling the BC functions. The BC functions can be controlled by the application through vendor control access of ULPI PHY registers.

Figure 9-4 Interconnections

9.1.4 The bc_chirp_on Signal Behavior

The bc_chirp_on signal can be used to know about an imminent Chirp K signaling when the DWC_usb3 controller is acting as a device. This requirement is from the Battery Charging standard which mandates that during Chirp K, the device must lower its current consumption from IDEV_HCHG_HS (900 mA) to IDEV_HCHG_CHRP (560 mA).

This section describes the behavior of the bc_chirp_on signal in the following scenarios:

- FS Idle to Host Reset
- FS Suspend to Host Reset
- HS Idle to Host Reset
- HS Idle to Host Suspend to Host Reset
- HS IDLE to HS LPM Suspend to Host Reset

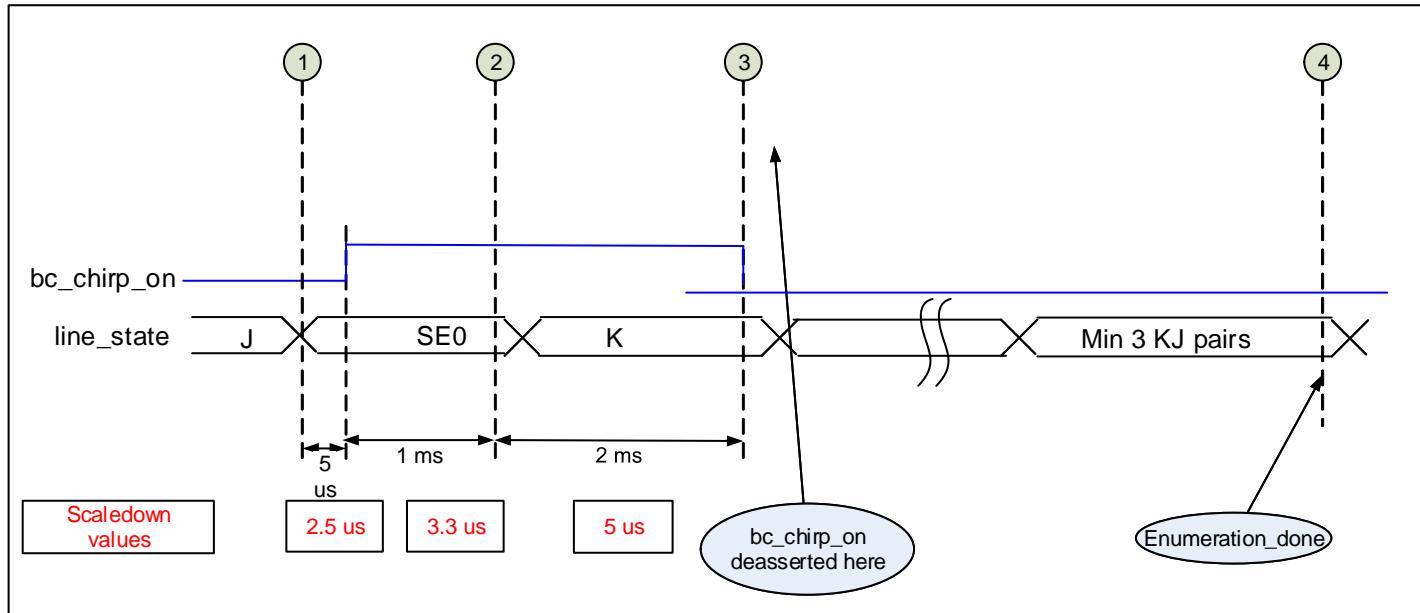


The timing diagrams are not to scale and are illustrative only. In all these scenarios, DWC_usb3 is working as a USB device.

9.1.4.1 FS Idle to Host Reset

In this scenario, USB is in Full Speed Idle state (J) when the USB Host drives an SE0 to signal a USB reset.

Figure 9-5 The bc_chirp_on Signal Behavior During FS Idle to Host Reset



Sequence of operations:

1. The USB host drives an SE0 to indicate a USB reset. About 5 μ s ($\pm 1 \mu$ s) after sensing SE0, the DWC_usb3 controller asserts bc_chirp_on.
2. About 1 ms ($\pm 1 \mu$ s) after asserting bc_chirp_on, the DWC_usb3 controller drives a Chirp K on the USB. The duration of the Chirp K is 2 ms ($\pm 1 \mu$ s).
3. DWC_usb3 controller stops driving Chirp K on USB along with de-assertion of bc_chirp_on.
4. As a consequence of the DWC_usb3 controller stopping driving Chirp K on USB, the USB host may respond with a minimum of three KJ pairs, completing the High Speed reset protocol. In this case, the device enumerates as High Speed or the host may also not respond with Chirp in which case the device enumerates as Full Speed.

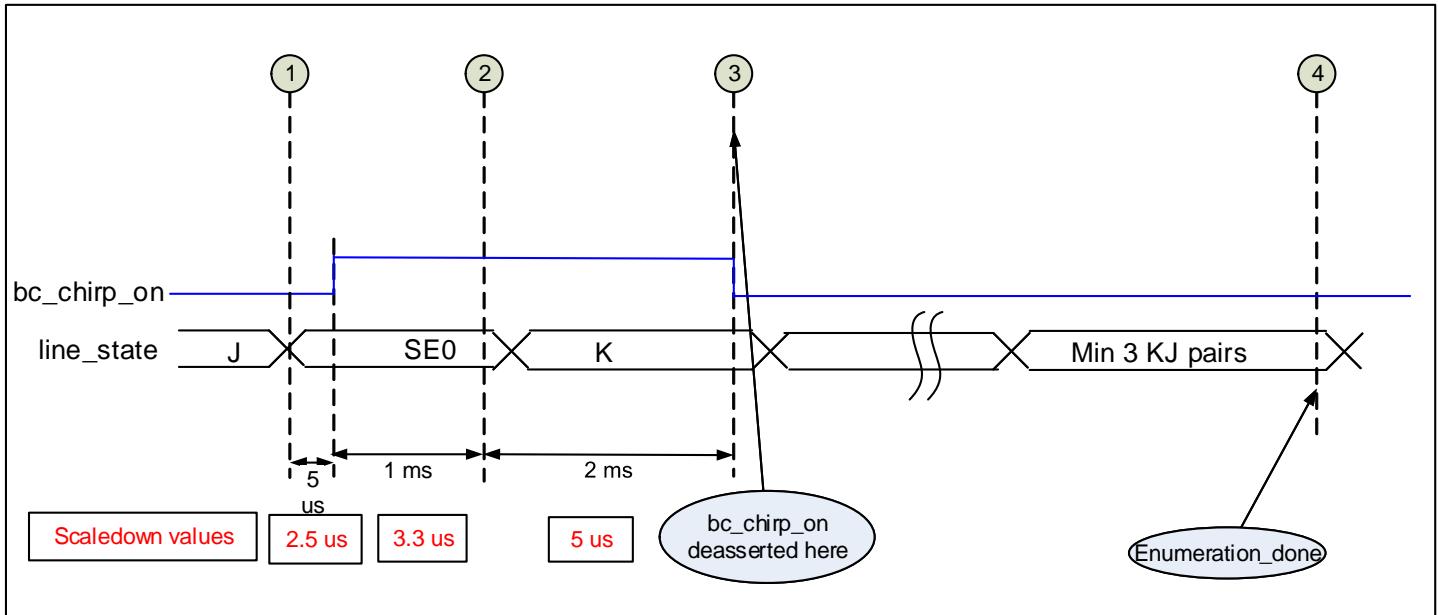


The $\pm 1 \mu$ s timing variance is provided for keeping the design simple and is applicable to all scenarios described in this section.

9.1.4.2 FS Suspend to Host Reset

In this scenario, USB is in Full Speed Suspend state when the USB Host drives a SE0 to signal a USB reset.

Figure 9-6 The bc_chirp_on Signal Behavior During FS Suspend to Host Reset



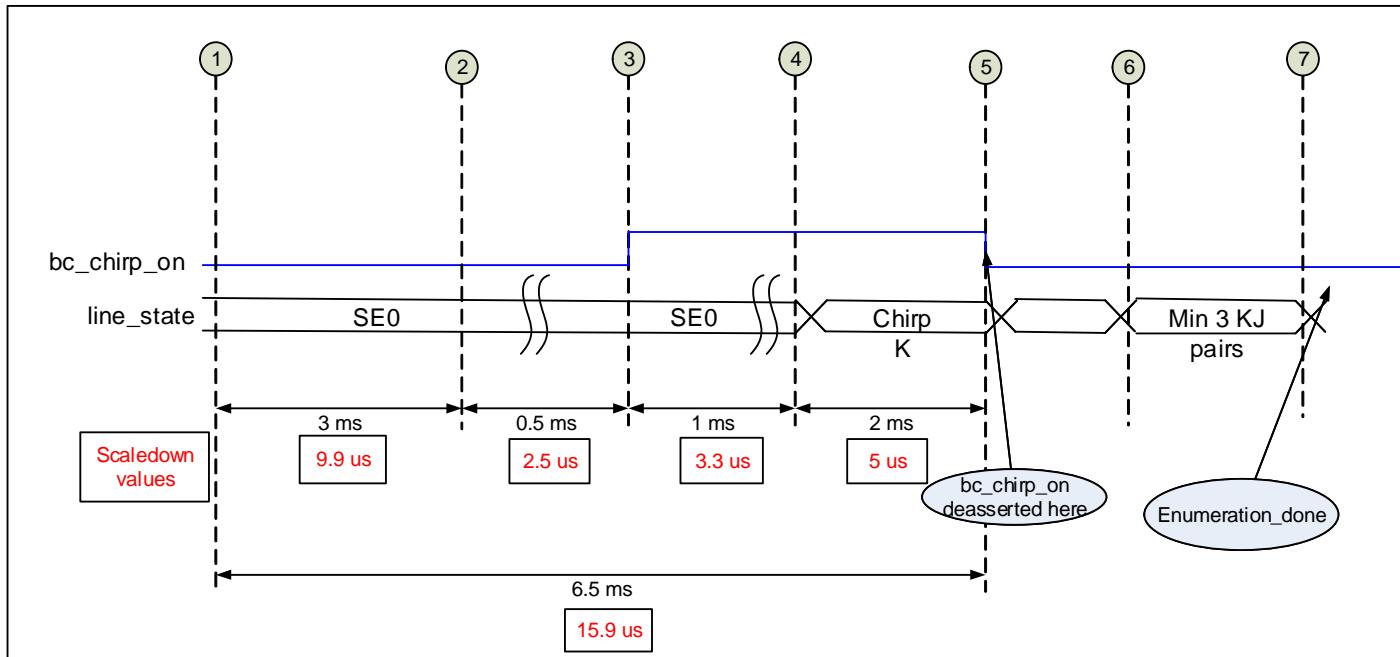
Sequence of operations:

1. The USB host drives an SE0 to indicate a USB reset. About $5 \mu\text{s}$ ($\pm 1 \mu\text{s}$) after sensing SE0, the DWC_usb3 controller drives bc_chirp_on.
2. About 1 ms ($\pm 1 \mu\text{s}$) asserting bc_chirp_on, the DWC_usb3 controller drives a Chirp K on the USB. The duration of the Chirp K is 2 ms .
3. The DWC_usb3 controller stops driving Chirp K on USB along with de-assertion of bc_chirp_on.
4. As a consequence of the DWC_usb3 controller stopping driving Chirp K on USB, the USB host may respond with a minimum of 3 KJ pairs, completing the High Speed reset protocol. In this case, the device enumerates as High speed or the host may also not respond with Chirp in which case the device enumerates as Full Speed.

9.1.4.3 HS Idle to Host Reset

HS Idle state on USB is an SE0 for less than or equal to 3 ms. Beyond this time, the USB Host goes into Suspend state when the DWC_usb3 controller drives a J state. If the Host is driving the SE0 even after 3.5 ms, the DWC_usb3 controller interprets this as a Host Reset and drives a Chirp K.

Figure 9-7 The bc_chirp_on Signal Behavior During HS Idle to Host Reset

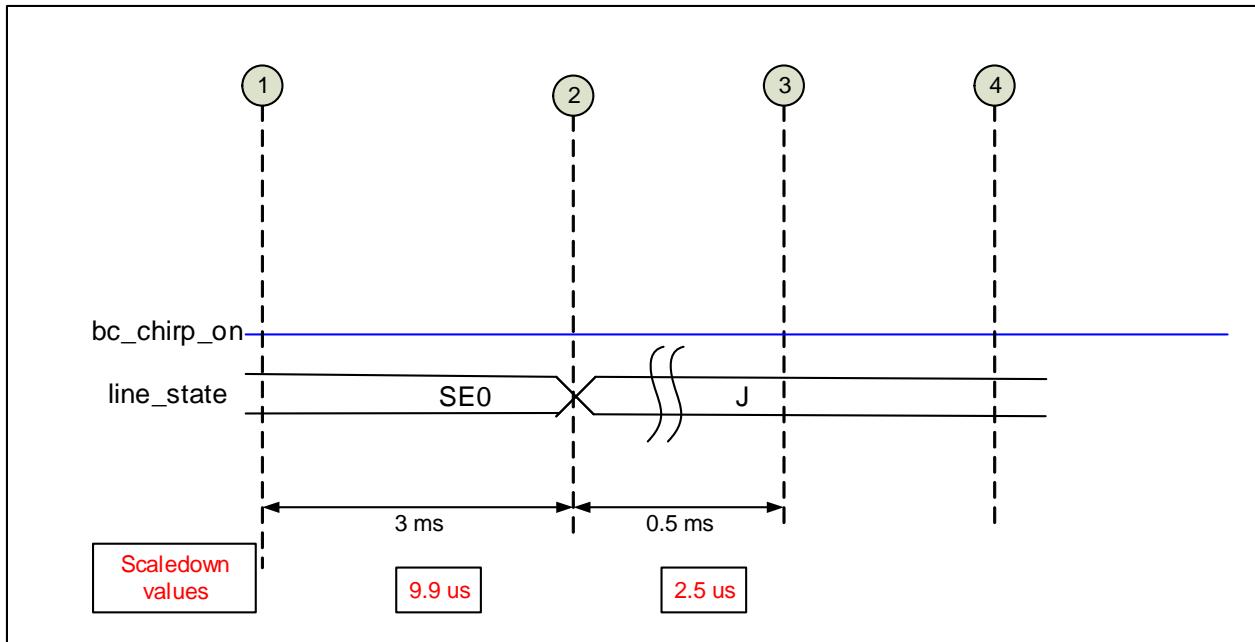


Sequence of operations:

1. The USB host drives SE0 to indicate a USB reset. (At this point of time, the DWC_usb3 controller is unable to determine if the SE0 on USB is due to USB host driving SE0 or because of inactivity on USB).
2. There is no activity on USB for 3 ms. the DWC_usb3 controller pulls up the D+ line to indicate a J on USB. Because the host is driving a strong SE0, it does not get reflected on USB. The DWC_usb3 controller keeps its D+ line pulled up for 0.5 ms.
3. The DWC_usb3 controller dssr_state checks for the USB line state and finds that it is still SE0. Because of this, bc_chirp_on is asserted. At the same time, an 1 ms timer is started by the DWC_usb3 controller.
4. At the end of 1 ms timeout, Chirp K is driven by the DWC_usb3 controller for a duration of 2 ms.
5. At the end of 2 ms duration of driving chirp K, the DWC_usb3 controller de-asserts bc_chirp_on signal.
6. The USB host (if it is HS-capable) recognizes the Chirp K and start to respond with minimum of 3 KJ pairs.
7. The HS/FS enumeration process is complete.

9.1.4.4 HS Idle to Host Suspend to Host Reset

Figure 9-8 The bc_chirp_on Signal Behavior During HS Idle to Host Suspend



Sequence of operations:

1. There is no activity on USB. (At this point of time, the DWC_usb3 controller is unable to determine if the SE0 on USB is because of USB host driving SE0 or because of inactivity on USB).
2. There is no activity on USB for 3 ms. The DWC_usb3 controller pulls up the D+ line to indicate a J state on USB. Because the host is not driving a strong SE0, it gets reflected on USB as a J state. The DWC_usb3 controller keeps its D+ line pulled up for 0.5 ms.
3. The DWC_usb3 controller checks for the USB line state and finds that it is a J state. Because of this, bc_chirp_on is not asserted.
4. Because of J state on USB, the DWC_usb3 controller goes to suspend state.
5. The DWC_usb3 controller then follows the steps listed in section “FS Suspend to Host Reset” in this chapter.

9.2 Interrupt Hierarchy

The host and peripheral operation requires DWC_USB3_NUM_INT number of interrupt lines where,

DWC_USB3_NUM_INT = max {DWC_USB3_HOST_NUM_INTERRUPTER_SUPT,
DWC_USB3_DEVICE_NUM_INT}

In addition, the following separate interrupt line is provided:

- bc_interrupt

The CPU sub-system has a single interrupt handler for all DWC_usb3 interrupts.

In the case of multiple sources of interrupts in a device or host, the priority is determined as per the device or host programming considerations. If there is a bc interrupt, it is given higher priority compared to a host or device interrupt. The interrupt priority is listed as follows:

1. BC
2. Host or Device

To determine the source of the interrupt, the following Interrupt Pending fields in GSTS have to be read:

- BC_IP
- Host_IP
- Device_IP

10

Power Management

This chapter describes the power management capabilities of the DWC_usb3 controller, which include Low Power Mode, Hibernation, and Clock Gating features. Hibernation and clock gating can be configured using coreConsultant. The following topics are discussed:

- “[Managing Power in USB 3.0 by Using Different Power States](#)” on page [480](#)
- “[Hibernation](#)” on page [481](#)
 - “[Overview of Hibernation](#)” on page [481](#)
 - “[Hibernation Architecture](#)” on page [483](#)
 - “[General Operation of Hibernation](#)” on page [485](#)
 - “[Hibernation Memory Requirements](#)” on page [486](#)
 - “[Configuring DWC_usb3 for Hibernation](#)” on page [486](#)
 - “[Programming DWC_usb3 for Hibernation](#)” on page [487](#)
 - “[Integrating DWC_usb3 with Hibernation Enabled](#)” on page [487](#)
 - “[Hibernation Dependencies and Assumptions](#)” on page [494](#)
- “[Clock Gating](#)” on page [496](#)
- “[Reference Clock Turn Off Feature for Power Saving](#)” on page [498](#)

Related Information

- Parameters
 - “[Basic Config Parameters](#)” on page [191](#)
 - “[Advanced Config Parameters](#)” on page [233](#)
- Signals:
 - “[Power Controller Interface Signals](#)” on page [390](#)

This chapter is aimed primarily at system architects and RTL/verification/software engineers who need to understand how power management works so that they can integrate the controller into an SoC.

10.1 Managing Power in USB 3.0 by Using Different Power States

The USB specification defines various power states:

- For SuperSpeed, there are four power states – U0, U1, U2 and U3
- USB 2.0 has L0, L1 and Suspend (L2).

The U0 state is normal data transfer operating state in SuperSpeed mode and, similarly, L0 state is the normal data transfer operating state during USB 2.0 mode. Higher power states increase both power savings and exit latency. For example, U3 has more power savings than U1, and the exit latency for U3 is greater than U1.

Though the specification defines these power states, the actual power saving technique depends on the specific implementation.

[Table 10-1](#) lists the power saving mechanisms that are available in the DWC_usb3 controller.

Table 10-1 Power Savings Support

USB 3.0 State	USB 2.0 State	Device Power Savings	Host Power Savings
U1 (Hardware Initiated)	N/A	Core: Clock gating PHY: P1	Core: Clock Gating PHY: P1
U2 (Hardware Initiated)	LPM-L1 (Hardware initiated)	Core: Clock gating, LPM-L1 Hibernation PHY: P2/Sleep	Core: Clock gating, PHY: P2/Sleep
U3 (Software initiated)	Suspend (Software initiated)	Core: Clock Gating, Hibernation PHY: P3/Suspend	Core: Clock gating, hibernation PHY: P3/Suspend
Rx.Detect (Software Initiated)	None to Disconnect	N/A	Core: Clock gating, Hibernation PHY: P3/Suspend
SS.Disabled (Software Initiated)	None to Disconnect	Core: Clock gating, hibernation PHY: P3/Suspend	None to N/A

According to the USB 3.0 specification:

- U0 is the normal operational state during SuperSpeed operation.
- U1 is a low power state (during SuperSpeed operation) with fast transition time back to the U0. During U1, PHY PLL remains on.
- U2 is the next low power state (during SuperSpeed operation) with slower exit time. PHY PLL can be turned off in this state.
- U3 is a deep power saving state (during SuperSpeed operation) in which portions of hardware, except those required to detect wakeup, can be turned off. Unlike other states, U3 is always initiated by software.
 - Similarly, for USB 2.0, the low power states are LPM_L1 and suspend. In LPM_L1, the controller can go into shallow sleep mode or deep suspend mode depending on the HIRD value.
 - Suspend is a deeper power saving state in USB 2.0 mode, similar to U3 in SS.

10.2 Hibernation

Hibernation is an optional feature that requires a separate add-on license. This section provides an overview of Hibernation, its architecture, and operation model. Configuration, programming, and integration details are also provided.

10.2.1 Overview of Hibernation

The hibernation feature supports the following Hibernation (save/restore) capabilities:

- Device mode: L1 [optional], L2, U3, Disconnected
- Host mode: L2, U3, Disconnected

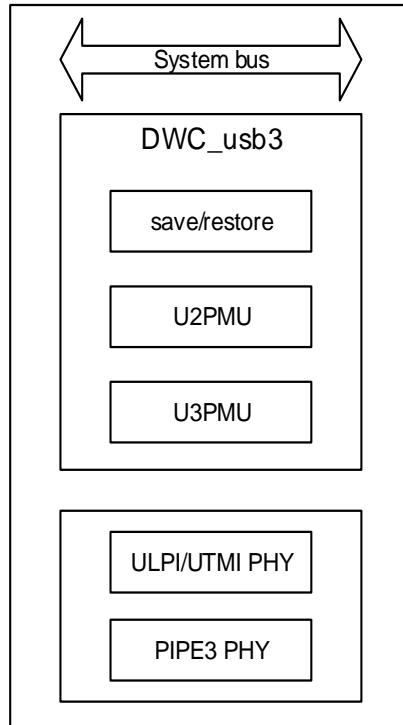


Hibernation is an optional feature; to enable this feature, you need the Hibernation Add-on license.

When hibernation is supported, the controller saves some of its internal state to an external location and allows the DWC_usb3 controller to be powered off. Small modules, namely the Power Management Units (PMUs), detect wakeup conditions and request that power be reapplied to the controller, at which time the external state is brought back into the controller.

[Figure 10-1](#) is a system-level diagram the DWC_usb3 controller with hibernation enabled.

Figure 10-1 Power Management Hibernation Architecture



Hibernation State Diagram

The hibernation feature for both host controller and device uses terminology introduced in the *PCI Bus Power Management Interface Specification, Revision 1.2* (PCI-PM) as a standard way of referring to the different device states and power rails. This includes the concept of Vcc and Vaux, device power states, and the PME signal.

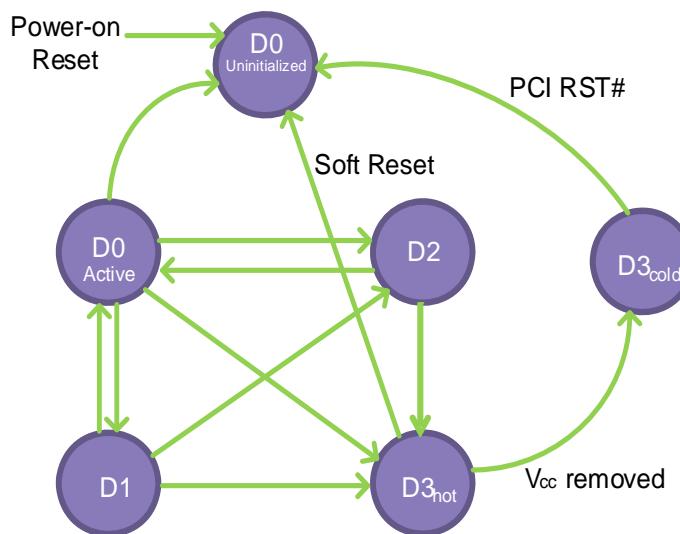
The DWC_usb3 controller has two power supplies:

- Vcc is the power supply for the majority of the controller.
- Vcc is switched on and off by a customer-provided power controller.
- Vaux is the power supply for the PMUs, which is the only logic that is necessary to detect wakeup from the PHY interface and restore state to the sticky modules.

When the controller is implemented as part of a *PCI Express controller*, the *PCI Express Base Specification, Revision 2.1* governs the limits placed on Vaux. For example, the current limit on Vaux is 375 mA when wakeup is enabled and 20 mA when wakeup is not enabled.

[Figure 10-2](#) illustrates a state diagram from the PCI-PM specification. There are six states involved, however, the implementation of the D1 and D2 light sleep states is optional. In the DWC_usb3 controller, hibernation is supported only during the PCIe D3 state.

Figure 10-2 State Diagram for Hibernation



The controller has the following states:

- **D0:** This is the fully operational state. Vcc and Vaux are being provided to the controller.
- **D3hot:** The controller software has already suspended the port(s), stopped the controller, and performed a save operation. Vcc and Vaux are still being provided to the controller, but the controller must be prepared to lose Vcc by switching the PHY interface to the PMU modules.
- **D3cold:** Vcc has been removed and only Vaux is being provided to the controller. All logic powered by Vcc has lost its state.

The states D0_active and D0_uninitialized are combined as D0. D0 is the fully operational state of the controller. Power controller supplies both Vcc and Vaux to the controller.

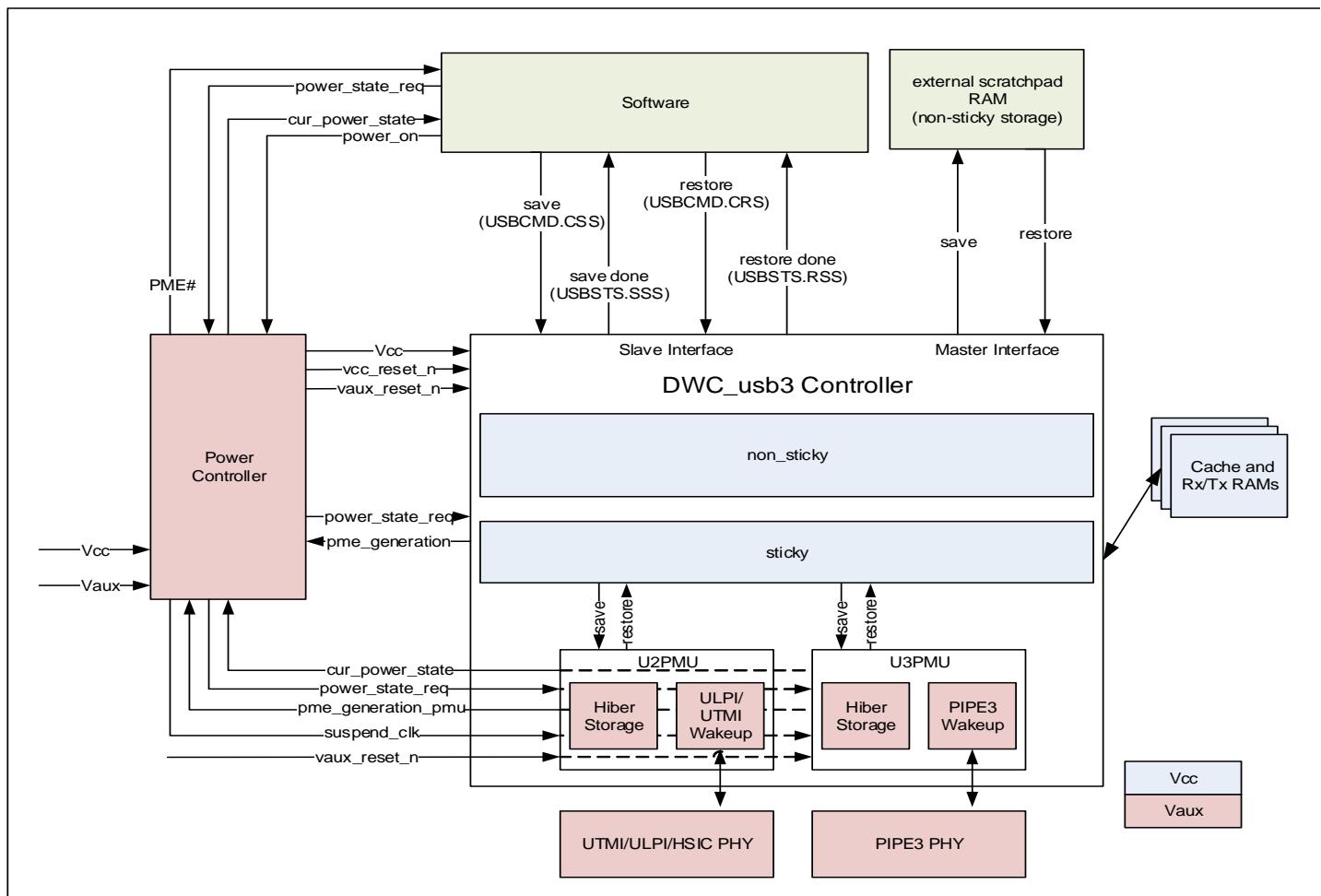
All the timing diagrams shown later in this chapter are with respect to this state diagram. Note that in the state machine implemented in the controller, there is no difference between D3cold and D3hot. Both the states are combined as D3. It is only based on whether the power controller supplies Vcc or not, D3 is differentiated as D3hot and D3cold.

According to [PCI-PM], when the device is in a non-D0 state, software cannot access any of the memory-mapped I/O space, which includes any of the controller's registers. After being programmed into D0, software cannot access the memory-mapped I/O space for 10 ms (including the PORTSC register). While the controller is in D3 state, the power controller asserts a reset to the USB3 controller.

10.2.2 Hibernation Architecture

[Figure 10-3](#) illustrates the top-level architecture of the Hibernation feature for Host mode. For Device mode, you can use the same diagram to understand the behavior of the DWC_usb3, but the software interactions are slightly different.

Figure 10-3 Hibernation Architecture in Host Mode



There following are the four main modules of the architecture:

- Non-sticky modules
- Sticky modules
- PMUs
- Power controller

In addition, to integrate the hibernation blocks requires attention to the isolation cells, level-shifters, and the PHY power supply.

Non-Sticky Modules

Non-sticky modules are reset by a light reset. Then, when a save request is initiated by software, the information in these modules is saved (via DMA) to the external scratchpad memory. The amount of time it takes to restore these modules is not critical to how the USB operates.

Sticky Modules

Sticky modules are modules that are reset by a cold reset or a hibernation reset. When the power state is set to D3, these modules save their state to the sticky storage area of the PMU and they restore their state from the same area. To maintain that the USB is operating correctly, these modules must be restored quickly. Sticky modules are primarily associated with managing the link.

Traditionally, these modules would be powered by Vaux. In the Synopsys implementation, to reduce power further, these are in the Vcc domain, and during hibernation, the hardware saves and restores the required state information in the PMU modules. This is transparent to the software.

The sticky modules produce an output (`pme_generation`) that may also get asserted when the controller is powered up.

Note: This PME is intended for use with application designs that do not use the hibernation feature.

PMUs

The PMUs are the only modules powered by Vaux. They contain wakeup logic for each port (either USB 2.0 or USB 3.0) that detects when the link partner is trying to resume the link, and when a connection or disconnection occurs. The PMU also has 200+ registers (flops) to save the sticky state information.

In SuperSpeed host mode, the U3 PMU periodically performs Receiver Detection during U3 or during the Rx.Detect state.

The PMUs have an output (`pme_generation_pmu`) that indicates that a wakeup event has occurred that requires Vcc to be restored.

Isolation Cells and Level-Shifters

The signals crossing from Vcc to Vaux must have isolation cells. Also, if Vcc and Vaux are different voltages, then you need to place level-shifters (L->H and H-> L) on the signals crossing between the two voltages.

For an illustration of where you should place isolation cells and level shifters, see [Figure 10-5 on page 489](#).

PHYs

The PHY may also use Vaux to power an always-on block, and Vcc to power the rest of the PHY, which is similar to Vcc and Vaux being used in the digital controller.



This is required in a PCI Express application to meet the Vaux current limit.

In a non-PCIe application, the PHY may use one voltage input but you can implement an internal power switch to turn off power to unused logic in P3/Suspend state to save power.

Power Controller

The power controller is a customer-provided piece of logic that interfaces with the controller and the PMU during power state transitions. Even though this is a piece of hardware, it can be controlled by a hardware state machine, or it can be controlled by software.

10.2.3 General Operation of Hibernation

Hibernation is controlled in the following two stages:

- Software – initiates the save and restore of the upper layers (non-sticky modules)
- Power controller – initiates the save and restore of the lower layers (sticky modules)

Entering Hibernation Mode

In general, when the software directs the controller to save its state to the external memory, the controller performs the following sequence of steps:

1. Collects state information from the non-sticky internal modules, and writes this information into a temporary location in RAM0.
2. Performs one or multiple DMA writes on the system bus to copy the saved non-sticky information plus required internal cache state information in RAM0 to the external scratchpad memory.
3. Indicates to the software that the save operation has completed.

When the power controller directs the controller to enter D3 state, the controller performs the following steps:

1. Indicates to the internal modules to stop operations and returns to an idle state if possible.
2. Collects the state information from the sticky internal modules, and writes this information to the sticky storage in the PMUs.
3. Switches control of the PHY interface to the PMUs.
4. Indicates to the power controller that the controller has entered D3hot state, and it is safe to remove Vcc.

Exiting Hibernation Mode

The PMUs detect the exit condition and generate PME signals to the power controller. When the power controller restores Vcc and de-asserts reset, the controller performs this sequence of steps:

1. Begins restoring state information from the sticky storage into the sticky modules.
2. Waits for the power controller to direct the controller to enter the D0 state.
3. After restoration is complete, indicates to the power controller that the controller has entered D0 state.
4. Switches control of the PHY interface to the sticky modules.

When software directs the controller to restore its state, the controller follows this sequence of operations:

1. Performs DMA reads to fetch the contents of the non-sticky and internal cache state information from the external scratchpad memory.
2. Restores the state information to the non-sticky modules.
3. Indicates to internal modules to resume operations.
4. Indicates to software that the restore operation completed.

10.2.4 Hibernation Memory Requirements

The hibernation mode system memory storage requirement (in 32-bit DWORDs) are:

- 5 + `DWC_USB3_HOST_NUM_U3_ROOT_PORTS DWORDs for the internal states of the controller
This requirement is managed by hardware, where the DMA save/restore to the scratchpad buffer when the DCTL.SSS or DCTL.RSS field is set to '1'.
- One DWORD for each active endpoint to store the endpoint state information.
This is managed by software which contains the information received by the "Get Endpoint State" command.
- Transfer related information if any transfer is in progress; this is managed by the software.
 - If the transfers are ISOC only and if the application does not need to save information, then none are required.
 - In a mass storage application, the host can suspend in the middle of a USB transfer and therefore all the unfinished TRBs and corresponding data buffers must be saved and restored by device software.

Example: In a device-only configuration (one port) where data does not need to be saved, for 20 endpoints (10 IN, 10 OUT), the amount of RAM required outside the controller is: $(20 + 6) * 4 = 104$ bytes.

10.2.5 Configuring DWC_usb3 for Hibernation

You can enable the Hibernation feature in DWC_usb3 controller by selecting "Clock Gating and Hibernation (Two Power Rails)" for the configuration parameter "Power Optimization Mode" (DWC_USB3_EN_PWR0PT) in coreConsultant.

- The DWC_usb3 controller does not support the following features in Hibernation mode:
 - BC
 - SRP Signaling
- The SuperSpeed PHY must support receiver detection in the PHY power state P3 to enable the Hibernation feature, and the DWC_USB3_RXDET_IN_P3_DS parameter must be set to “1”.
- In host mode, the DWC_usb3 controller does not support clock gating and hibernation when the debug capability is enabled during runtime (`DCCTRL.DCE=1`).
During the normal host mode of operation, the DWC_usb3 controller supports clock gating and hibernation features when the debug capability is not enabled by the software (`DCCTRL.DCE=0`).
- To meet timing, after receiving a wake-up request from the PMU, the customer-provided power controller hardware must restore power within 100us, provide a stable `bus_clk` within 5ms, and the software driver must re-initialize and finish the restoration process within 6ms. For more details, see the “Timing Dependencies” section in the *DWC Super-Speed USB 3.0 Controller Programming Guide*.



Note

10.2.6 Programming DWC_usb3 for Hibernation

For details, see the “Programming DWC_usb3 for Hibernation” section in the *DWC SuperSpeed USB 3.0 Controller Programming Guide*.

10.2.7 Integrating DWC_usb3 with Hibernation Enabled

This section discusses the following topics:

- [“Integrating DWC_usb3 with the Power Controller Interface”](#)
- [“Integrating DWC_usb3 with the ULPI Interface” on page 490](#)
- [“Integrating with a PCI Express Controller” on page 490](#)
- [“Hardware and Software Integration Requirements for Hibernation Disable Control” on page 491](#)

Integrating DWC_usb3 with the Power Controller Interface

[Figure 10-4](#) shows the interface-level block diagram of the PMU.

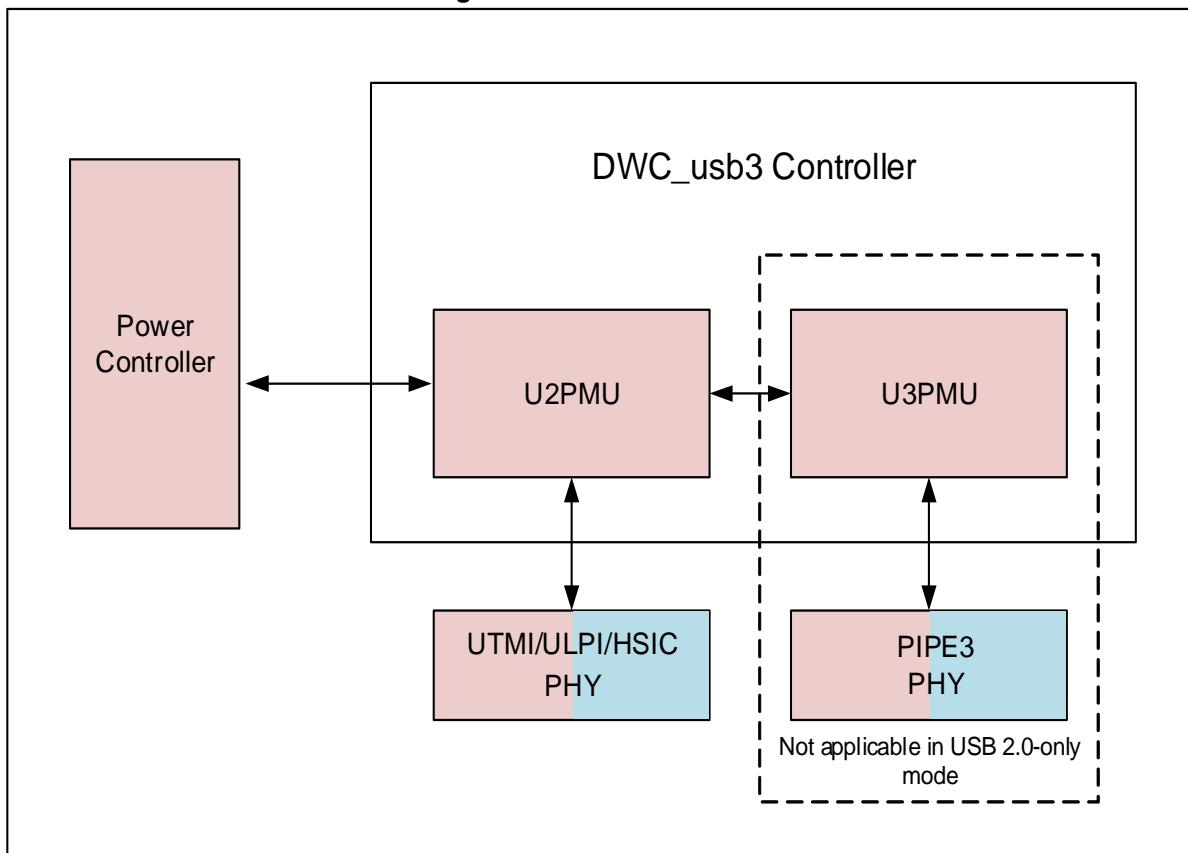
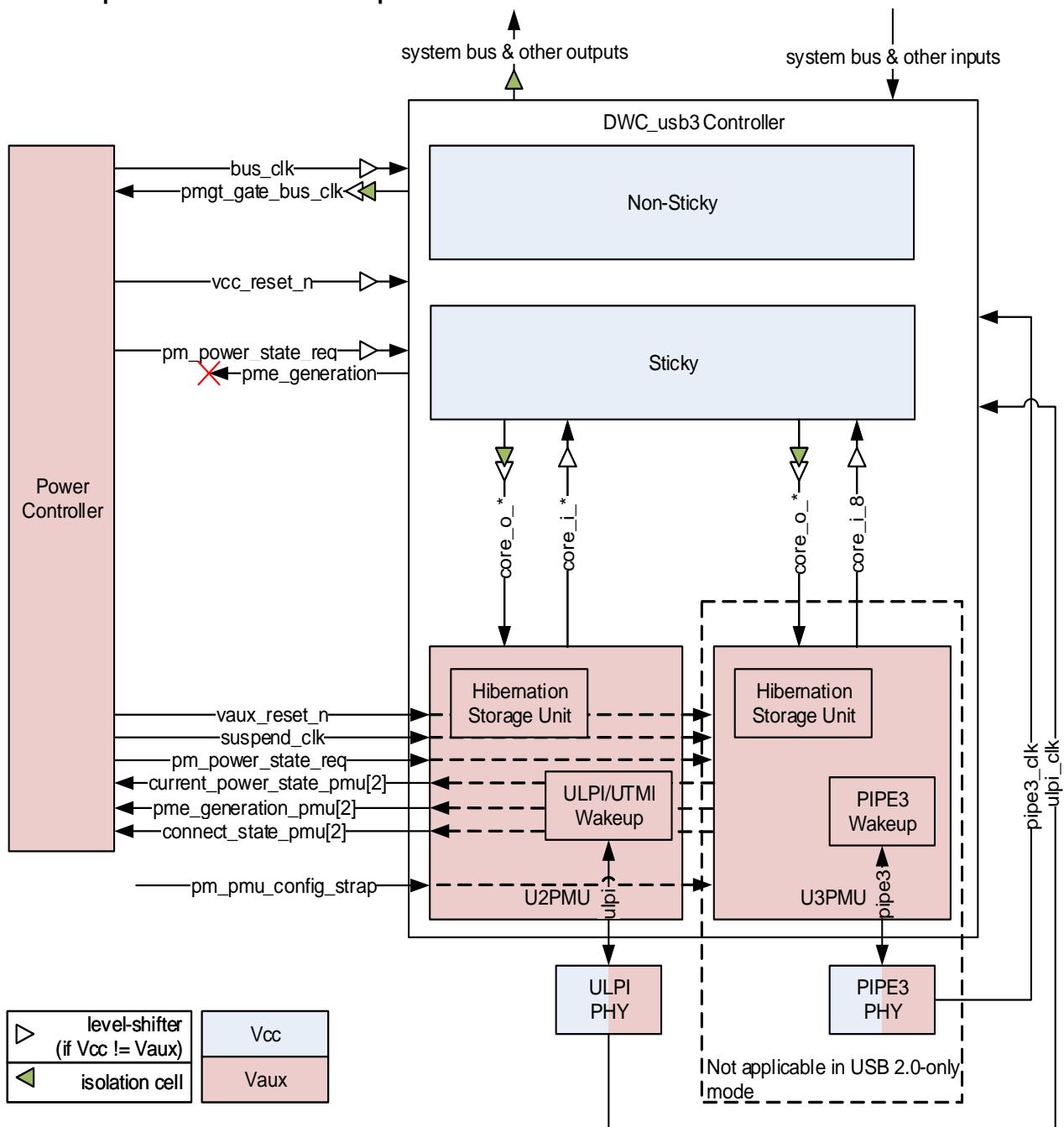
Figure 10-4 PMU – Interface Level Block Diagram

Figure 10-5 shows the top-level interfaces between the controller, power controller, and PHYs. This is only an example, as the clocks and reset may come from a different module.



For description of the power controller interface signals, refer to Chapter 7, “Signal Descriptions” on page 297.

Figure 10-5 Top-Level Interfaces – Example With One ULPI Port and One PIPE3 Port

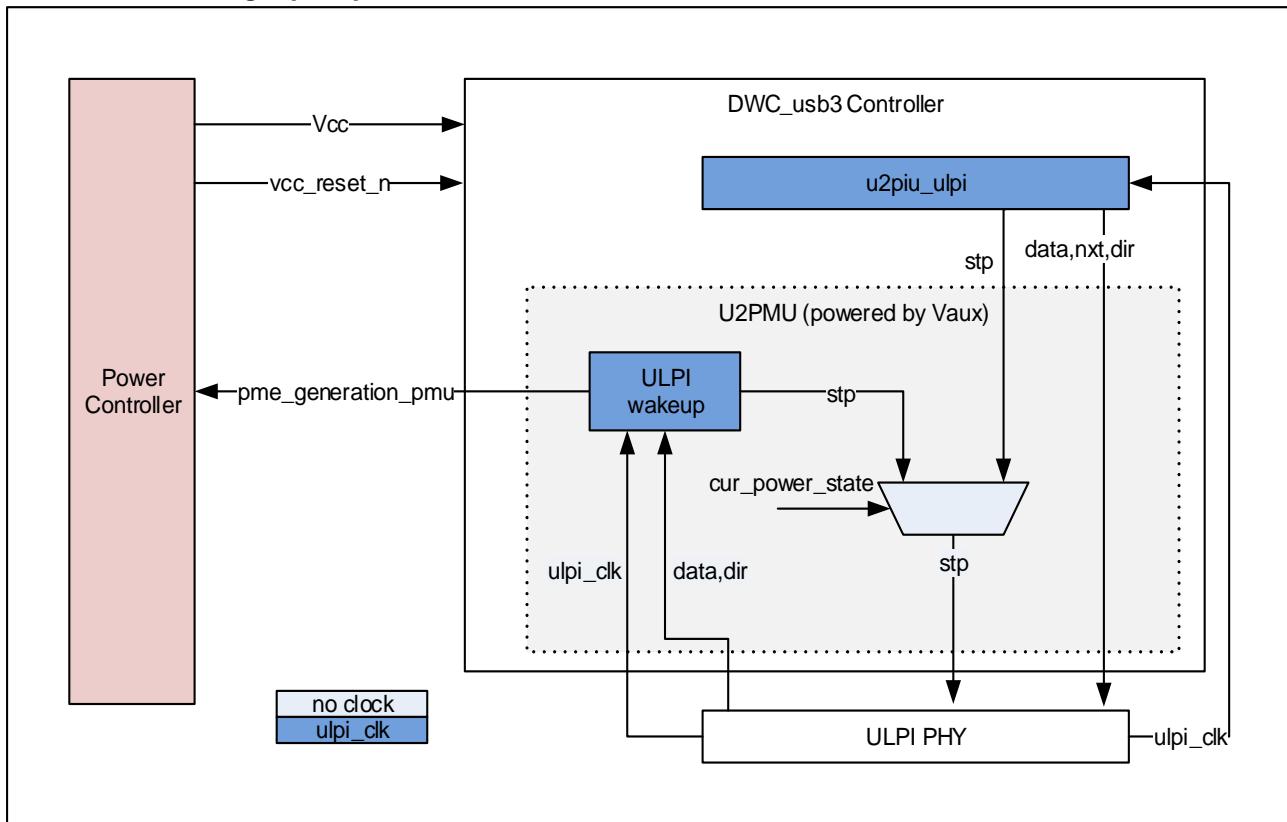
Integrating DWC_usb3 with the ULPI Interface



Disregard the following diagram. This version of the controller drives STP from the controller, not from the PMU. This is still being investigated, so future versions may revert to driving STP from the PMU.

[Figure 10-6](#) illustrates how the PMU controls the ULPI stop signal while it is in the D3 state.

Figure 10-6 PMU Driving ulpi_stp in D3



Each signal shown is unique to the ULPI port that generates it. For example, if there are four ULPI ports, there will be four copies of these signals. The controller and PMU do not support mixing ULPI and UTMI ports.

Integrating with a PCI Express Controller

To integrate the DWC_usb3 controller with the hibernation feature with a PCI Express controller, refer to section “Integrating the Dual Power Rail (Hibernation) Feature” in the *DWC SuperSpeed USB 3.0 Controller User Guide*.

With the suggested connections, when power management software puts the device into the D3hot state, the controller receives this information and starts storing its sticky state. When the controller has finished saving the sticky state, it indicates that it is in D3, which causes the PCIe controller to transmit the PM_Enter_L23 DLLP. Then the host puts the link into L2/L3 Ready state, and finally into the L2 state where Vcc is removed.

When a wakeup condition is detected, the PME from the PMUs triggers the PCIe controller to request Vcc to be restored either in-band (via beacon) or out-of-band (via WAKE#) and then transmit a PM_PME message to the Root Complex. Power management software sets the device state to D0, which causes the controller to restore its sticky state.

In device mode, where the timing requirements are very strict, if the latency of power management software setting the device state to D0 is unknown, you can take steps to increase the predictability of the timing when coming out of hibernation. First of all, the power controller must request the controller and PMUs to enter D0 without waiting for software intervention. In addition, software (or firmware) must initialize the device controller up to the last step in “Software or PHY-Initiated Resume While Connected” without waiting for power management software to put the controller into the D0 state.

Hardware and Software Integration Requirements for Hibernation Disable Control

In some cases, it may be desirable to disable the hibernation feature to work around hardware or software issues that cannot be worked around in any other way. You need to make some special considerations for the controller configuration, power controller design, and device driver design to allow hibernation to be disabled. These are described in this section.

The register bit GCTL[1] (GblHibernationEn) controls the global hibernation enable setting for the controller. When set to ‘1’, hibernation is enabled and when it is ‘0’, hibernation is disabled. Specifically, the controller’s behavior is influenced in the following way when GCTL[1] is set to ‘0’:

- In host mode, if software sets USBCMD.CSS to ‘1’, the controller sets USBSTS.SSS to ‘1’ and then set it to ‘0’ 100 cycles later.
- In host mode, if software sets USBCMD.CRS to ‘1’, the controller sets USBSTS.RSS to ‘1’ and then sets it to ‘0’ 100 cycles later.
 - This is to emulate a save/restore state operation. However, the controller does not perform any DMA to fetch the scratchpad buffer array nor does it save or restore any state from the external scratchpad buffers.
 - If vcc_reset_n is asserted between the CSS and CRS requests, the controller sets USBSTS.SRE (Save/Restore Error) upon completion of the restore request to indicate that the restore failed.
- In device mode, if software sets DCTL.CSS to ‘1’, the controller sets DSTS.SSS to ‘1’ and then immediately set it to ‘0’.
- In device mode, if software sets DCTL.CRS to ‘1’, the controller sets DSTS.RSS to ‘1’ and then immediately sets it to ‘0’.
 - Similar to host mode, the controller will not perform any DMA or save/restore operations.
- If the power controller requests D3 by setting pm_power_state_req to 3, the controller issues a command to the PMUs through the core_hsu interface indicating that hibernation is disabled, and no sticky save operation takes place. Both PMUs then indicate current_power_state=D3.
 - However, in this situation the PMUs are not controlling the PHY interfaces, and the controller is still controlling the PHY interfaces.
 - Because the PMUs are in a “fake” D3 state, they do not assert pme_generation_u2pmu or pme_generation_u3pmu signals.
- If the power controller requests D0 by setting pm_power_state_req to 0, the PMUs immediately indicates that current_power_state=D0. No sticky restore occurs, and the controller continues to control the PHY interfaces.

Therefore, note the following to ensure hibernation can be disabled:

1. In coreConsultant, configure the reset value of GCTL.GblHibernationEn.

Bit 1 of the coreConsultant parameter DWC_USB3_GCTL_INIT controls the reset value of the GblHibernationEn field. Depending on the software, it may be more appropriate to set the reset value to '0' which assumes hibernation is disabled by default, and then only enable it in software when the driver decides that it is safe to enable. The value of GCTL.GblHibernationEn must be valid before software issues the CSS request.

2. In host mode, use the pme_generation output from the controller in addition to the pme_generation outputs from the PMUs.

Because the PMUs do not generate a PME in "fake" D3, the software needs some way of receiving an interrupt when a wakeup condition occurs in host mode if USBCMD.RS is set to '0'. Because the PMUs are disabled, the wakeup condition comes from the controller's pme_generation output. The power controller should OR the pme_generation output of the controller along with the pme_generation outputs of the PMUs.

3. Continue to request D3 entry and wait for the current power state of the PMUs to be equal to D3.

If the power controller knows that hibernation is disabled, it may choose not to request D3 from the controller. However, to keep the power controller design more straightforward, it can request D3 entry as it normally does. The PMUs, knowing that hibernation is disabled, must respond to the power controller that the current state is D3 even though the PMUs are disabled.

4. During D3, do not assert vcc_reset_n.

- If hibernation is disabled and the power controller asserts vcc_reset_n, the controller's state information is lost (because it was never saved). When hibernation is disabled, the power controller must not assert vcc_reset_n in D3.
- In some cases, the power controller knows hibernation is disabled through a vendor-specific register in the power controller. However, if this information is not available, the following bits indicate that the PMUs are disabled:

- debug_u2pmu[DWC_USB3_NUM_U2_ROOT_PORTS*DWC_USB3_DEBUG_U2WAKEUP_W], where, DWC_USB3_DEBUG_U2WAKEUP_W=54
- debug_u3pmu[0]

These bits are valid when the corresponding PMU is in D3. Therefore, the power controller must only assert reset if both PMUs are enabled and in D3. For example:

```
always @(posedge suspend_clk or negedge reset_switch_n) begin      if
(~reset_switch_n) pm_hibernation_en_r <= 1'b0;      else if (pm_hibernation_en_r==0)
begin      if ((pm_power_state_request == 2'h3) && (current_power_state_u3pmu[1:0]
== 2'h3) && (current_power_state_u2pmu[1:0] == 2'h3))      pm_hibernation_en_r
<= 1'b1;      end      else if (pm_hibernation_en_r==1) begin
      if(pm_power_state_request == 2'h0)      pm_hibernation_en_r <=
1'b0;      end      end      assign vcc_reset_n = (~reset_switch_n ||
(!debug_u2pmu[DWC_USB3_NUM_U2_ROOT_PORTS*DWC_USB3_DEBUG_U2WAKEUP_W] &
!debug_u3pmu[0] & pm_hibernation_en_r)) ? 1'b0 : 1'b1;
```

5. In device mode, use the existing device programming model (leave DCTL.RunStop set to '1', and set DCTL.KeepConnect to '0')

When hibernation is disabled, the device controller does not support retaining the connection to the host while DCTL.RunStop=0. The device driver needs to use the existing non-hibernation programming model, which means:

- Keep DCTL.RunStop=1, even when the link is suspended or disconnected. RunStop is only set to '0' if the device does not want to connect to the host (soft disconnect).
- Set DCTL.KeepConnect=0, so that the controller does not generate Hibernation Events and does not go into a locked state when the link goes into suspend. This allows the controller to automatically resume if the host initiates resume.
- If the link is in suspend and the host initiates resume, the controller writes out a Link State Change Event which triggers the controller's interrupt because there is an event present in the Event Buffer.

10.2.8 Hibernation Dependencies and Assumptions

The following dependencies and assumptions apply to the hibernation feature:

- The software is responsible for initiating the save process and the restore process, and it does so through the controller's register interface. It only initiates the save or restore process when the controller's Run/Stop bit is set to 0.

Note: Run/Stop may only be set to 0 when endpoint transfers are stopped. This is a software requirement that is detailed in the programming models in the xHCI Specification (for host) or this databook (for device).

- Before removing power, the software must put the controller and PMUs into the D3 power state. The software may only put the controller into the D3 state if all enabled port(s) are in the U3, Disabled, or RxDetect state.

Note: This requirement is described from Appendix A in the xHCI specification.

- When the controller is in the D3 power state, system software may not access any CSR within the controller.

Note: The slave interface has no power or clock, so reads and writes to the controller memory space time out on the system bus. This requirement is described in the [PCI-PM] specification.

- Vcc cannot be removed from the controller until the output `current_power_state_pmu` from both PMUs is equal to 3 (D3).

Note: After the `power_state_req` signal is set to 3, the controller begins to save sticky state. When it has finished saving sticky state, the `current_power_state_pmu` output is set to 3 to reflect the fact that the controller is prepared to lose power.

- In host mode, the software must not set the Bus Master Enable (BME) bit in the PCI command register to '0' while the controller is performing a save or restore operation (when `USBSTS.SSS` or `USBSTS.RSS` are '1'). If it does, the controller may set the Host Controller Error (HCE) bit and interrupt.

Note: The save and restore requests cause the controller to perform bus mastering even though the `USBCMD.RunStop` bit is '0' and `HCHalted` is '1', so bus mastering cannot be disabled at these times.

- Each PMU signals its own PME from D3, regardless of the D state of the other PMU, and it keeps the PME signal asserted until it enters D0.

- The PMU contains noise filtering on some inputs from the PHY. The filter is very simple, however, and glitches may still cause the PME to be asserted even though a resume, connect, or disconnect did not actually occur. After wakeup, if the software sees no reason to restore the state, it may put the controller back into D3.

- Filtering is in place for the following signals: Overcurrent, UTMI `vbusvalid`, UTMI linestate, PIPE3 RxElecIdle, and PIPE3 PowerPresent.

- When the power controller transitions the controller from D3 to D0, it must assert `vcc_reset_n`, even if Vcc was never removed.

A D3-to-D0 transition causes a light reset to occur. To keep consistent behavior between D3hot-to-D0 and D3cold-to-D0, the power controller must assert `vcc_reset_n` in both situations.

- The `GSBUSCFG0/GSBUSCFG1` registers are not changed from their coreConsultant-configured reset values.

This means that during restore, these two registers come up with their configured reset values. The controller does not save these registers, and the software must program these register values to be restored. If you need to change these registers, these registers must be restored.

- For the controller acting as a host and operating in USB 2.0 mode to detect connect/disconnect, it continues to drive Vbus while in hibernation.
- When hibernation is going to be used, GUSB3PIPECTL[17] and GUSB2PHYCFG[6] (Suspend Enable for the 3.0 and 2.0 PHYs) must be set to '1'.

The U3PMU and controller assume that the PHY is in the P3 power state during hibernation, and if this bit is '0', the PHY is not put into the P3 power state.

- To support Receiver Detection on the PIPE3 in host mode during hibernation, the coreConsultant parameter "Enable Receiver detection in PHY power state P3?" (DWC_USB3_RXDET_IN_P3_DS) must be set to 1.

While the controller is in hibernation, the PIPE3 must be in the P3 state. Therefore, the PHY must support receiver detection in P3.

10.3 Clock Gating

Clock gating is a power saving technique in which the controller turns off clocks to its internal modules when they are not being used, that is, when the controller is idle and the following conditions are met.

- In host mode, the controller turns off clocks to its internal modules when:
 - All SuperSpeed ports are in U1, U2, or U3 state;
 - all USB 2.0 ports are in Suspend or L1 Suspend state; and
 - the controller is idle.

Similarly, when the controller is acting as a device, in SuperSpeed mode, the controller implements clock gating when the SuperSpeed link is in U1, U2, or U3 state and the controller is idle.

In USB 2.0 device mode, the controller implements clock gating when the link is in LPM-L1 or suspend, and the controller is idle. ?

The controller has the following clock domains:

- Bus clock
- RAM clock
- MAC3 clock
- USB 3.0 PIPE3 clock
- USB 2.0 MAC/PHY clock
- Suspend clock

For more details on different clock domains, refer to “[Logical Hierarchy, Clock Domains and Data Flow](#)” on page [83](#).

Additional Power Saving Options

When the clock gating feature is selected, the controller turns off the following clocks:

- RAM clock to all modules in the RAM clock domain;
- BUS clock to all modules in the bus clock domain, except the BUS_GS module that is needed to detect wakeup from the slave interface or software;
- MAC3 and PIPE3 clocks during U3; and
- MAC2 and USB2 PHY clocks during suspend and LPM-L1.

To enable the clock gating feature, select the clock gating option in coreConsultant and also enable it using the GCTL register.

The controller implements clock gating in the following situations:

- In USB 2.0 device mode
 - When the UTMI suspend or l1_suspend signal is asserted and the controller is idle.
 - When the ULPI PHY is suspended via Suspend and the controller is idle.
- In USB 3.0 device mode, when the link is in U1, U2 or U3 state and the controller is idle

In host mode, clock gating is enabled in the following situations:

- When all USB 2.0 ports are in the suspend or L1 Suspend state, and all USB 3.0 ports are in the U1/U2/U3 state, and the controller is idle.

Internal clock gating applies to:

- Modules that use `ram_clk`
- Some modules that use `bus_clk`

Internal clock gating applies to:

- Modules that use `mac3_clk`: When the USB 3.0 PHY is suspended, these modules switch to using `suspend_clk`, so that their power consumption is reduced but the clock is not completely stopped.
- Modules that use `mac2_clk`: When the USB 2.0 PHY is suspended and the PHY (utmi/ulpi) clock is turned off, these modules do not get any clock. Therefore, their power consumption is reduced during suspend.
- The `bus_gs` module, which uses `bus_clk`. This module detects wakeup on the slave interface and therefore needs a non-gated clock.

You can enable or disable clock gating by using the `GCTL` register. You can adjust the default value of the enable/disable signal by configuring the value of the `coreConsultant` parameter. Because the `GCTL` register is “sticky,” it retains its value across soft resets and hibernation.

10.4 Reference Clock Turn Off Feature for Power Saving



Note Reference Clock Turn Off (EA) Feature is an Early Adopter (EA) feature. If you are planning to use this feature, contact Synopsys.

This section describes how to enable/disable the reference clock which is fed to the `ref_clk` input of the controller and the reference clock input of the PHY. It covers the following topics:

- “Reference Clock Turn Off Feature Overview”
- “Configuring the Controller to Enable Reference Clock Turn Off Feature”
- “Integrating the DWC_usb3 with Reference Clock Turn Off Feature Enabled” on page 499
- “Programming the Reference Clock Startup Time” on page 499

10.4.1 Reference Clock Turn Off Feature Overview

The following scenarios summarize when the reference clock can be turned on/off:

Host Mode:

- The reference clock can be turned off when the controller enters low power state with all the ports either in `ss.disable`, `rx.detect`, or `u3`.
- When the host is performing receiver detection, it requests `ref_clk` to be turned on.
- After the receiver detection, if the controller returns to `rx.detect` or `u3`, it requests `ref_clk` to be turned off again.

Device Mode:

- The reference clock can be turned off when the device enters `ss.disable`, `rx.detect`, or `u3`.

In both host and device mode, if a remote wakeup is detected, the controller requests the `ref_clk` to be turned on again. This message is communicated to the SoC through the `pmgt_ref_clk_off` signal. This signal is an asynchronous signal. When the SoC restarts the clock and the clock is stable, this message is communicated back to the controller through the `pmgt_ref_clk_ok` signal.

10.4.2 Configuring the Controller to Enable Reference Clock Turn Off Feature

The `ref_clk` turn off feature can be supported if the controller and the PIPE support power states of P3.CPM/P4 on the `pipe_PowerDown` signal.

You can enable the `ref_clk` turn off feature by selecting "Yes" for the following configuration parameters in coreConsultant:

- "Will ref_clk reference PLL be switched off?" (`DWC_USB3_REF_CLK_OFF=1`)
- "PHY supports P3.CPM and P4 - EA Feature?" (`DWC_USB3_SSPHY_SUPPORT_P3CPM_P4=1`)

During run time, as part of initialization of the controller, you must set the registers `GUCTRL1.DisRefClkGtng=0`, and `LLUCTL.support_p4=1`.

10.4.3 Integrating the DWC_usb3 with Reference Clock Turn Off Feature Enabled

Figure 10-7 is an example showing how to integrate DWC_usb3 controller with Synopsys PHY that supports P3.CPM/P4 states.

Figure 10-7 Integrating the DWC_usb3 Controller with SSPHY – Reference Clock Off Signals

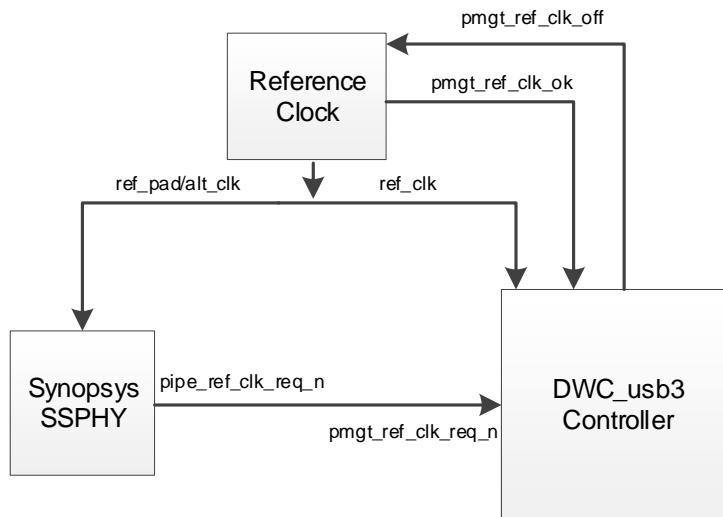
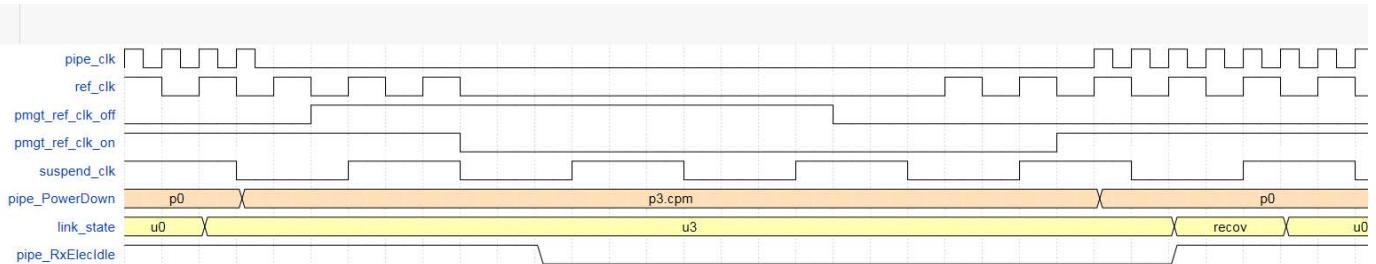


Figure 10-8 shows an example of how to turn off and turn on the reference clock.

Figure 10-8 Reference Clock Turned off and Turned on Again – Reference Timing Diagram



10.4.4 Programming the Reference Clock Startup Time

The time taken for the reference clock to start and run within the jitter margin needs to be programmed in the LPTMDPDELAY.p3cpmp4_residency register field. This timer is programmed in terms of the number of suspend_clk periods. For example, if the suspend_clk is 32kHz, and it takes 64us to restart the reference clock after the controller de-asserts the pmgt_ref_clk_off signal, then the p3cpmp4_residency register field must be set to 2.

A

Area, Speed, and Power

This appendix describes the area, speed, and power requirements for several example configurations of the SuperSpeed USB 3.0 Controller IP.

The topics are as follows:

- “Device Area” on page 502
- “Host Area Numbers” on page 503
- “DRD Area Numbers” on page 504
- “USB 2.0-Only Mode Area Numbers” on page 504
- “Hub Area Numbers” on page 504
- “Area Differences Due to Scan Ready and Clock Gating” on page 505
- “Speed” on page 505
- “Power-Compiler Clock Gating” on page 505
- “Power Consumption” on page 506
- “DFT Coverage” on page 507
- “Performance” on page 507
- “Minimum Clock Frequencies: bus_clk, ram_clk” on page 515

A.1 Device Area

Table A-1 and **Table A-2** show the areas for different hardware architecture parameters with different numbers of bidirectional device endpoints/host channels.



The area numbers are in Kilo-gates (K = 1,000 gates) for industry standard 65LP technology. The total cell area is divided by the area of the smallest NAND cell in the library to get the area numbers.

Area for 32-bit Native Master Data-Width, 32-bit Native Slave Data-Width, 2 Bi-Directional endpoints (1 control OUT, 1 control IN, 1 Bulk OUT, and 1 Bulk OUT), 125-MHz AHB and RAM clock, 125-MHz PIPE clock, and 60-MHz UTMI clock configuration = 160 KGates

Area for 32-bit AHB Master Data-Width, 32-bit AHB Slave Data-Width, 2 Bi-Directional endpoints (1 control OUT, 1 control IN, 1 Bulk OUT, and 1 Bulk OUT), 125MHz AHB and RAM clock, 125 MHz PIPE clock, and 60 MHz UTMI clock configuration = 166.5 KGates.

Table A-1 shows areas for different device bi-directional endpoint configurations at 64-bit AHB Master Interface.

Table A-1 Area for Different Architectures vs. Device Bi-dir Endpoints (K Gates)

Main Parameters: AHB Master Data Width = 64-bits, AHB Slave Data Width = 32-bits, ISOC = enabled, DWC_USB3_BMU_SRC/SNKBUF_TYPE=1 Synthesis AHB clock = 125 MHz, RAM Clock = 125 MHz, PIPE Clock = 250 MHz, UTMI clock = 60 MHz											
BiDir EPs ->	2	3	4	5	6	7	8	10	12	14	16
Area	295K	302K	309K	316K	322K	328K	333K	374K	359K	369K	380K

Table A-2 shows the areas required for DWC_usb3 features other than endpoints.

Table A-2 Areas for DWC_USB3_MODE Features

Feature	Area
ULPI Wrapper	2.5K
Area per Bi-Directional Endpoint	7K
AHB Master Interface	7.5K
32-bit Master Interface (vs) 64-bit Master Interface (Slave Interface Width Change impact is negligible)	24K
64-bit Master Interface (vs) 128-bit Master Interface	30.6K
AXI Master Interface	18K
64-bit Address (controller's internal address width is always 64-bit)	4K
Hibernation	22.8K

Device area estimation for your configuration = 277K + (Num Bi-Dir EPs - 2) * 7.0K

- + ULPI? 2.5K : 0 + (64-bit Address)? 4K : 0
- + Native Interface? (32-bit? 0 : 64-bit? 22.2K : 49K) : 0
- + AHB? (32-bit? 7K : 64-bit? 18K : 62K) : 0
- + AXI? (32-bit? 24K : 64-bit? 50K : 84K) : 0
- + Isoc Support? 5.7K : 0

A.2 Host Area Numbers

Table A-3 shows the DWC_usb3 Host Area using industry standard 16FFC library.

Table A-3 **DWC_usb3 Host Area**

Configuration: 32-bit Master Address, 32-bit Slave Data Width, AHB, 125 MHz BUS/RAM clock, 250 MHz PIPE3 clock, and 60 MHz UTMI clock	Area
32-bit Data Bus, AHB, 1 U2 and 1 U3 Ports	550K
64-bit Data Bus, AHB, 1 U2 and 1 U3 Ports	595K
64-bit Data Bus, AHB, 2 U2 and 1 U3 Ports	613K
64-bit Data Bus, AHB, 1 U2 and 2 U3 Ports	684K
64-bit Data Bus, AHB, 2 U2 and 2 U3 Ports	703K
64-bit Data Bus, AHB, 4 U2 and 4 U3 Ports	894K
64-bit Data Bus, AHB, 8 U2 and 8 U3 Ports	1276K
64-bit Data Bus, AHB, 15 U2 and 8 U3 Ports	1400K
64-bit Data Bus, AHB, 8 U2 and 15 U3 Ports	1821K
64-bit Data Bus, AHB, 15 U2 and 15 U3 Ports	1945K
64-bit Data Bus, AHB, 2 U2 and 2 U3 Ports, 2 SS Bus-Instances	892K
64-bit Data Bus, AHB, U2 and 2 U3 Ports, 2 HS Bus-Instances, 2 SS Bus-Instances	975K
64-bit Data Bus, AHB, 4 U2 and 4 U3 Ports, 2 SS Bus-Instances	1084K
64-bit Data Bus, AHB, 4 U2 and 4 U3 Ports, 4 HS Bus-Instances	1156K
64-bit Data Bus, AHB, 4 U2 and 4 U3 Ports, 4 HS Bus-Instances, 2 SS Bus-Instance,	1344K
64-bit Data Bus, AHB, 4 U2 and 4 U3 Ports, 4 SS Bus-Instances	1457K
64-bit Data Bus, AHB, 4 U2 and 4 U3 Ports, 4 HS Bus-Instances, 4 SS Bus-Instance,	1724K
32-bit Data Bus, AHB, 4 U2 and 4 U3 Ports, 4 HS Bus-Instances, 2 SS Bus-Instance	1264K
32-bit Data Bus, AHB, 64-bit Address, 4 U2 and 4 U3 Ports, 4 HS Bus-Instances, 2 SS Bus-Instance	1268K
32-bit Data Bus, AHB, 4 U2 and 4 U3 Ports, 4 HS Bus-Instances, 2 SS Bus-Instance, FPGA	1278K
32-bit Data Bus, Native-Interface, 4 U2 and 4 U3 Ports, 4 HS Bus-Instances, 2 SS Bus-Instance	1333K
32-bit Data Bus, AHB, 4 U2 and 4 U3 Ports, 4 HS Bus-Instances, 2 SS Bus-Instance, Scan-Insertion	1462K

$$\begin{aligned}
 \text{Host Area} = & 542K + (\text{Num U2 Ports} - 1) * 18K + (\text{Num U3 Ports} - 1) * 89K + (\text{Num HS Bus-Instance} - 1) * 83K \\
 & + (\text{Num SS Bus-Instance} - 1) * 189K \\
 & + (\text{Enable DbC} == 1) ? ((\text{Num SS Bus-Instance} == \text{Num U3 Ports}) ? 52K : 239K) : 0 \\
 & + \text{ULPI? } 3K : 0 + (\text{64-bit Address}) ? 4K : 0 \\
 & + \text{Hibernation? } (29K) : 0 \\
 & + \text{Native Interface? } (32-bit? 0 : 64-bit? 44K : 78K) : 0 \\
 & + \text{AHB? } (32-bit? 8K : 64-bit? 52K : 92K) : 0 \\
 & + \text{AXI? } (32-bit? 25K : 64-bit? 70K : 114K) : 0
 \end{aligned}$$

A.3 DRD Area Numbers

[Table A-4](#) shows the DWC_usb3 DRD Area using industry standard 16ffc library, 64-bit master and 32-bit slave, AHB, 125 MHz AHB

Table A-4 DWC_usb3 DRD Area

Configuration	Area
4 Bi-Dir EPs, 1 U2 and 1 U3 Ports	723K
4-Bi-Dir EPs, 2 U2, 2 U3, 1 SS instance, 2 HS instance	923K
4-Bi-Dir EPs, 2 U2, 2 U3, 2 SS instance, 2 HS instance	1110K
4 Bi-Dir EPs, 4 U2 and 4 U3 Ports	1025K

Approximate DRD area increase over Host configuration = 100K to 110K

A.4 USB 2.0-Only Mode Area Numbers

[Table A-5](#) shows the area numbers for USB 2.0-only mode.

Table A-5 USB 2.0-only Mode – Area/RAM Numbers (Default Configuration, 32-bit AHB)

Mode	Area
Device	209.6 KGates/12 KB RAM
Host	341.3 KGates/15 KB RAM
DRD	450 KGates/15 KB RAM

A.5 Hub Area Numbers

[Table A-6](#) shows the DWC_usb3 Hub Area using industry standard 65LP library, 4 deep header buffers and with JTAG.

Table A-6 DWC_usb3 Hub Area

Configuration	Area
1-Port	200K
2-Port	285.2K
3-Port	370.2K
4-Port	455.4K
15-Port	1392K

Hub Area with 4 deep header buffers = 193K + (Num Ports - 1) * 85K + (JTAG? 7.6K : 0)

4-Port area with 8 deep header buffers = 567K

Area of JTAG = 7.6K

A.6 Area Differences Due to Scan Ready and Clock Gating

[Table A-7](#) shows area differences due to scan ready and clock gating using an example configuration.

Table A-7 Area Differences Due to Scan Ready and Clock Gating

Main Parameters: Default Device Clocks: Synthesis AHB/RAM clock = 125 MHz, PIPE3 Clock = 250 MHz, UTMI clock = 60 MHz	
No Clock Gating, No Scan-Ready	308.6K
Clock Gating, No Scan-Ready	308.8K
No Clock Gating, Scan-Ready	336.9K
Clock Gating, Scan-Ready	336.9K

A.7 Speed

[Table A-8](#) shows the areas for different frequencies and tools.

Table A-8 Area vs. Frequency vs. Tool

Main Parameters: Device Default		
Frequency	Synthesis Tool	Area
AHB/RAM = 125 MHz, PIPE = 250 MHz, UTMI = 60 MHz	Design Compiler	308.6K
AHB/RAM = 250 MHz, PIPE = 250 MHz, UTMI = 60 MHz	Design Compiler	308.6K
AHB/RAM = 400 MHz, PIPE = 250 MHz, UTMI = 60 MHz	Design Compiler	308.6K

The PIPE and UTMI/ULPI clock frequency does not impact the area much since they are local to a small PHY Interface unit. On the other hand, the System Bus clock and RAM clock frequency has impact on the area.

A.8 Power-Compiler Clock Gating

[Table A-9](#) shows the percentage of flops which are clock-gateable using Power Compiler.

Table A-9 Power Compiler Clock Gating Summary

Main Parameters: Default				
Configuration	Number of Clock Gating Elements	Number of Gated Registers	Number of Ungated Registers	Total Number of Registers
Device	880	14836 (81.48%)	3373 (18.52%)	18209
Host 1-Port	1590	27552 (86.04%)	4472 (13.96%)	32024
Hub 4-Port	1150	23091 (86.18%)	3703 (13.82%)	26794

The modules are coded in such a way that 85% of the flops can be clock gated using Power Compiler. The state machine and control signal flops are typically not clock gateable by Power Compiler, because they do not have enable or do not meet the minimum 3-bit common enable requirements. The power-compiler clock gating would save power even during normal operation.

A.9 Power Consumption

Table A-10 shows the dynamic power consumption with and without clock gating for an industry standard 16nm library using RTL simulation waveform with PrimeTime power analysis tool.

Table A-10 Power Consumption

Simulation: AHB/RAM Frequency = 125 MHz, PIPE Frequency = 125 MHz, UTMI Frequency = 60 MHz, suspend_clk Frequency = 62.5 MHz								
Mode	Tool Clock Gating Enabled				Tool Clock Gating Disabled			
	Internal	Switching	Leakage	Total	Internal	Switching	Leakage	Total
Device								
Idle	2.50E-03	1.22E-03	5.95E-07	3.72E-03	7.35E-03	3.16E-03	6.13E-07	1.05E-02
Traffic	2.70E-03	1.33E-03	5.93E-07	4.03E-03	7.53E-03	3.26E-03	6.15E-07	1.08E-02
Clock Gated	1.32E-03	6.11E-04	5.96E-07	1.93E-03	3.32E-03	1.36E-03	6.24E-07	4.69E-03
Hibernation	1.77E-05	3.31E-06	9.02E-09	2.09E-05	4.25E-05	7.03E-07	9.26E-09	4.32E-05
Host								
Idle	3.87E-03	1.83E-03	1.07E-06	5.71E-03	1.21E-02	5.20E-03	1.09E-06	1.73E-02
Traffic	4.04E-03	1.89E-03	1.07E-06	5.94E-03	1.23E-02	5.25E-03	1.09E-06	1.76E-02
Clock Gated	3.02E-03	1.42E-03	1.07E-06	4.45E-03	9.39E-03	3.99E-03	1.10E-06	1.34E-02
Hibernation	1.72E-05	2.91E-06	1.06E-08	2.01E-05	5.07E-05	7.10E-07	1.10E-08	5.14E-05
DRD (Device)								
Idle	4.31E-03	2.05E-03	1.28E-06	6.36E-03	1.41E-02	6.02E-03	1.30E-06	2.01E-02
Traffic	4.95E-03	2.30E-03	1.28E-06	7.25E-03	1.48E-02	6.31E-03	1.30E-06	2.11E-02
Clock Gated	1.35E-03	6.30E-04	1.29E-06	1.98E-03	4.07E-03	1.64E-03	1.33E-06	5.71E-03
Hibernation	1.84E-05	3.36E-06	1.10E-08	2.18E-05	5.44E-05	7.27E-07	1.18E-08	5.51E-05

In the coreConsultant GUI, after creating and synthesizing the RTL, you can do power analysis on your configuration using the Static Timing Analysis activity. For details on how to run power analysis using coreConsultant GUI, see “Performing Power Analysis” section in the User Guide.

A.10 DFT Coverage

[Table A-11](#) shows the DWC_usb3 DFT Coverage

Table A-11 DFT Coverage

Configuration	DFT Coverage
Device	99.4%
Host 1-Port	99.4%
Hub 4-Port	99.4%

The “scan_mode” top level is port is used for DFT scan bypass to improve DFT coverage. During Insert-DFT, this need to be set “1” as:

```
set_dft_signal -view existing_dft -type Constant -active_state 1 -port [list scan_mode]
```

By default at-speed DFT is supported so that all the modules get their default functional clock during scan mode.

If “Enable Additional DFT Control Ports” is selected then, in addition to “scan_mode” port additional DFT signals are used to control all the clock MUXes and reset logic.

A.11 Performance

[Table A-12](#) shows the performance.

Table A-12 Device Performance

Configuration	HW + Driver	Performance
Device	Synopsys Host, MCC1 xHCI Driver, Microsoft BOT (uses 64KB/128KB USB transfers)	170 - 190 MBs
Device	Synopsys Host, MCC1 xHCI Driver, MCC1 BOT (used 1MB USB transfers)	410 MBs Read, 419 MBs Write



This is not the Rx and TX RAMs used by the DWC_usb3 controller. This is the USB transfer buffer allocated by the USB SW driver in the system memory.

A.11.1 Host/Device/Hub Performance

Table A-13 shows the xHCI Host-mode performance numbers that are measured during hardware testing.

Table A-13 xHCI Host-Mode Performance During Hardware Testing

xHCI Host -> Hardware	PCIe Gen2x4 (Latency ^a ~ 1.5μs)	PCIe Gen2x1 (Latency1 ~= 2.8μs)	PCIe Gen1x4 (Latency1 ~= 2.0μs)	PCIe Gen1x1 (Latency1 ~= 3.5μs)
SNPS SS Device read	410 MBs	390 MBs	407 MBs	206 MBs
SNPS SS Device write	419 MBs	273MBs / 392MBs ^b	411 MBs	162MBs / 203MBs ^b
SNPS SS Device read w/Hub	394 MBs	380 MBs	392 MBs	205 MBs
SNPS SS Device write w/Hub	416 MBs	270MBs / 390MBs ^b	408 MBs	162MBs / 200MBs ^b
SNPS HS Device read	49 MBs	49 MBs	49 MBs	49 MBs
SNPS HS Device write	49 MBs	45 MBs	45 MBs	45 MBs
SNPS HS Device read w/Hub	49 MBs	45 MBs	45 MBs	45 MBs
SNPS HS Device write w/Hub	45 MBs	45 MBs	45 MBs	45 MBs

a. PCIe read latency for a 1 KB packet. A 1 KB packet on USB takes 2.2μS, if the system latency to fetch an 1 KB packet is more than 2.2μS, then the pipelining read requests improves the performance.

b. These are performance numbers when two outstanding system memory read requests are enabled ('DWC_USB3_NUM_OUT-STANDING_TDMA = 2').



- The Synopsys Device controller is 4-lane Gen2 PCIe-based.
- The PC motherboard used: Rampagell Gene, CPU I7 2.8GHz, Memory 6GB

Table A-14 shows the xHCI Host-mode super-speed performance numbers that are measured in the VTB simulation.

Table A-14 xHCI Host-Mode Super-Speed Performance in Simulation

Transfer size	Bulk IN Only (VTB Test2_BulkIN)	Bulk Out Only (VTB Test3_BulkOUT)	Concurrent Bulk IN and OUT (VTB Test9)	
			Bulk IN	Bulk OUT
128 KB	448 MBs	451 MBs	369 MBs	380 MBs
256 KB	450 MBs	455 MBs	379 MBs	386 MBs



- The performance numbers are dependent on configuration and system bus bandwidth/latency. The performance numbers in Table A-14 on page 508 are measured with the default host control configuration and high-bandwidth, low-latency system bus.
- The Bulk IN/OUT performance in the concurrent mode is lower than the Bulk IN/OUT only performance. To reduce the host controller area, only one scheduler is implemented for each super-speed bus instance. When there is only a Bulk IN transfer or a Bulk OUT transfer, the scheduler is dedicated to the IN or OUT transfer. When there are concurrent Bulk IN and Bulk OUT transfers, the scheduler is shared by two directions; therefore, the performance of each direction is reduced.

[Figure A-1](#) on page 509 and [Figure A-2](#) on page 510 are sample CrystalDiskMark snapshots that show the read and write performance in terms of Mega Bytes per Second (MBs). These snapshots are taken for the Synopsys FPGA platform on the PCIe Gen2 x4.



In this section, Mega Bytes per Second (MBs) means 1000,000 bytes per second.

[Figure A-3](#) on page 511, [Figure A-4](#) on page 512, [Figure A-6](#) on page 514, and [Figure A-7](#) on page 515 are sample snapshots that show the performance in terms of the number of Maximum packet size packets for SuperSpeed (1KB) and High-Speed (512-bytes) reads and writes. These snapshots are taken from the bus analyzer traces from the Synopsys Host FPGA platform on the PCIe Gen2 x4.

A SuperSpeed mode host receives scheduling (read) more conservatively than the transmit (write), since some SS device misbehave on burst early termination.

In general, high-speed write is slower than high-speed read because, an OUT transaction takes a little longer than an IN transaction due to internal delays, and the device and PHY turnaround times. In addition, in high-speed mode, the GUCTL[21] controls whether to schedule the USB 2.0 packets right after SOF or after a delay. Some of the USB 2.0 devices misbehave if packets are scheduled right after SOF. By default, it is recommended to have a delay for inter-operability purposes. [Figure A-2](#) shows the performance numbers with USB 2.0 packets scheduled right after SOF.

Figure A-1 SuperSpeed Read and Write

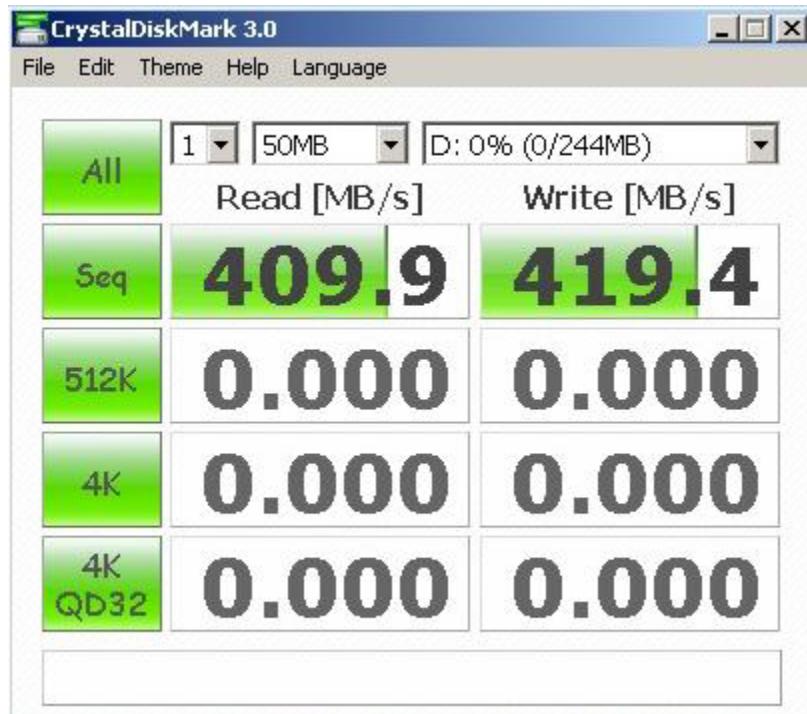


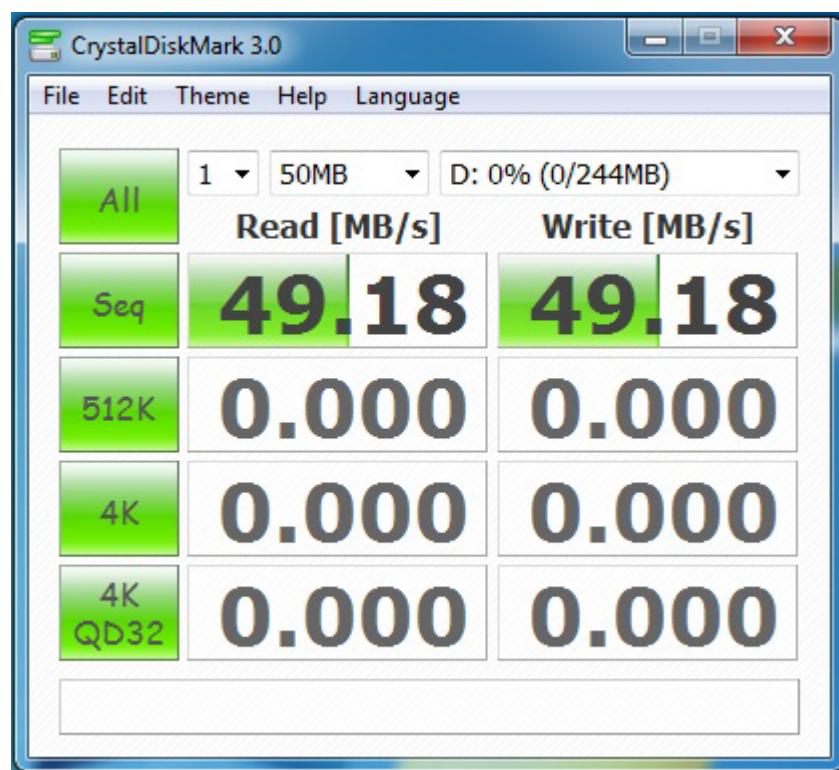
Figure A-2 High-Speed Read and Write

Figure A-3 SuperSpeed Read (53 SS Read Packets Per uframe)

Transaction		S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
3		S		1	1	5120 bytes	5	16.032 µs	4.386 155 808
ITP	Packet	H	I	S	ITP	Bus Interval Counter	Time Delta	Bus Interval Adjustment Control.	LCW
	274					1000	1	0	Hseq:4
53 * 1024 Bytes	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	4	S		1	1	16384 bytes	16	36.728 µs	4.386 172 056
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	5	S		1	1	16384 bytes	16	36.720 µs	4.386 208 784
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	6	S		1	1	16384 bytes	16	36.656 µs	4.386 245 504
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	7	S		1	1	5120 bytes	5	14.744 µs	4.386 282 160
ITP	Packet	H	I	S	ITP	Bus Interval Counter	Time Delta	Bus Interval Adjustment Control.	LCW
	607					1001	1	0	Hseq:6
53 * 1024 Bytes	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	8	S		1	1	16384 bytes	16	36.368 µs	4.386 297 160
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	9	S		1	1	16384 bytes	16	36.328 µs	4.386 333 528
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	10	S		1	1	16384 bytes	16	36.248 µs	4.386 369 856
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	11	S		1	1	5120 bytes	5	15.872 µs	4.386 406 104
ITP	Packet	H	I	S	ITP	Bus Interval Counter	Time Delta	Bus Interval Adjustment Control.	LCW
	940					1002	1	0	Hseq:0
53 * 1024 Bytes	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	12	S		1	1	16384 bytes	16	36.264 µs	4.386 422 208
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	13	S		1	1	16384 bytes	16	36.288 µs	4.386 458 472
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	14	S		1	1	16384 bytes	16	36.176 µs	4.386 494 760
	Transaction	S	IN	ADDR	ENDP	Data	ACK	Time	Time Stamp
	15	S		1	1	5120 bytes	5	16.104 µs	4.386 530 936
ITP	Packet	H	I	S	ITP	Bus Interval Counter	Time Delta	Bus Interval Adjustment Control.	LCW
	1273					1003	1	0	Hseq:2
53 * 1024 Bytes	Time	Time Stamp							
	224.000 ns	4.386 547 040							

Figure A-4 SuperSpeed Write (54 SS Write Packets Per uframe)

Figure A-5 High-Speed Read (12 HS Read Packets Per uframe)

Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
9	S	0x96	1	1	1 512 bytes	0x4B	15.084 µs	4 . 235 231 582
Packet	H	SOF	Frame #	CRC5	Pkt Len			
31	S	0xA5	49.7	0x01	12	2.066 µs	4 . 235 246 656	
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
10	S	0x96	1	1	0 512 bytes	0x4B	9.818 µs	4 . 235 248 732
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
11	S	0x96	1	1	1 512 bytes	0x4B	9.750 µs	4 . 235 258 550
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
12	S	0x96	1	1	0 512 bytes	0x4B	9.782 µs	4 . 235 268 300
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
13	S	0x96	1	1	1 512 bytes	0x4B	10.200 µs	4 . 235 278 082
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
14	S	0x96	1	1	0 512 bytes	0x4B	9.768 µs	4 . 235 288 282
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
15	S	0x96	1	1	1 512 bytes	0x4B	9.782 µs	4 . 235 298 050
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
16	S	0x96	1	1	0 512 bytes	0x4B	9.800 µs	4 . 235 307 832
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
17	S	0x96	1	1	1 512 bytes	0x4B	9.768 µs	4 . 235 317 632
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
18	S	0x96	1	1	0 512 bytes	0x4B	9.782 µs	4 . 235 327 400
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
19	S	0x96	1	1	1 512 bytes	0x4B	9.768 µs	4 . 235 337 182
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
20	S	0x96	1	1	0 512 bytes	0x4B	9.800 µs	4 . 235 346 950
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
21	S	0x96	1	1	1 512 bytes	0x4B	14.966 µs	4 . 235 356 750
Packet	H	SOF	Frame #	CRC5	Pkt Len			
68	S	0xA5	49.7	0x01	12	2.084 µs	4 . 235 371 716	
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
22	S	0x96	1	1	0 512 bytes	0x4B	9.782 µs	4 . 235 373 800
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
23	S	0x96	1	1	1 512 bytes	0x4B	9.784 µs	4 . 235 383 582
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
24	S	0x96	1	1	0 512 bytes	0x4B	9.784 µs	4 . 235 393 366
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
25	S	0x96	1	1	1 512 bytes	0x4B	9.782 µs	4 . 235 403 150
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
26	S	0x96	1	1	0 512 bytes	0x4B	9.800 µs	4 . 235 412 932
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
27	S	0x96	1	1	1 512 bytes	0x4B	9.800 µs	4 . 235 422 732
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
28	S	0x96	1	1	0 512 bytes	0x4B	9.768 µs	4 . 235 432 532
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
29	S	0x96	1	1	1 512 bytes	0x4B	10.200 µs	4 . 235 442 300
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
30	S	0x96	1	1	0 512 bytes	0x4B	9.782 µs	4 . 235 452 500
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
31	S	0x96	1	1	1 512 bytes	0x4B	9.750 µs	4 . 235 462 282
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
32	S	0x96	1	1	0 512 bytes	0x4B	9.818 µs	4 . 235 472 032
Transaction	H	IN	ADDR	ENDP	T ▶ Data	ACK	Time	Time Stamp
33	S	0x96	1	1	1 512 bytes	0x4B	14.900 µs	4 . 235 481 850
Packet	H	SOF	Frame #	CRC5	Pkt Len			
105	S	0xA5	49.7	0x01	12	2.100 µs	4 . 235 496 750	

Figure A-6 High-Speed Write (12 HS Write Packets Per uframe)

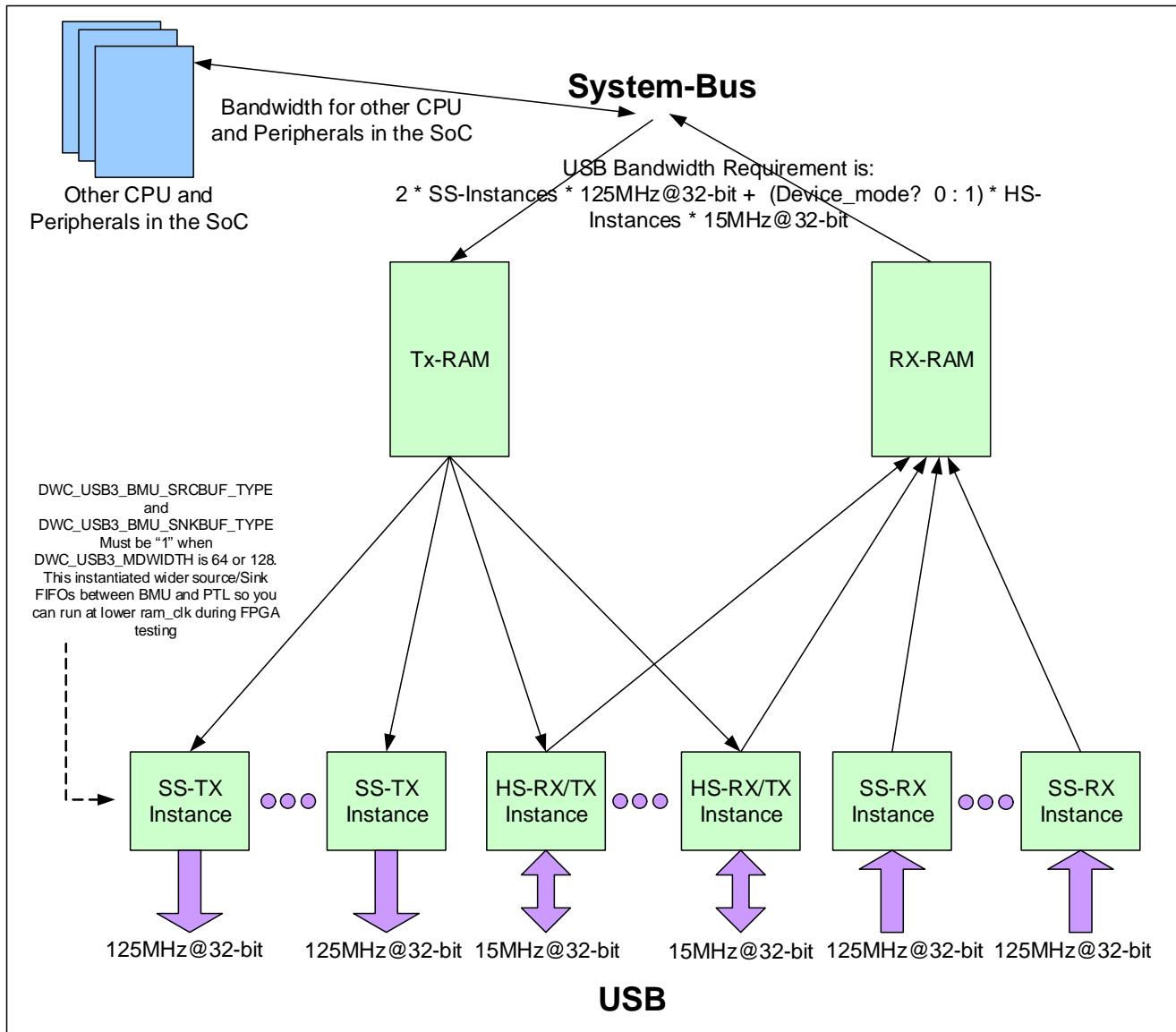
Packet	H	SOF	Frame #	CRC5	Pkt Len	Time	Time Stamp
4	H	SOF	0xA5	334.?	0x09 12	434.000 ns	13.536 314 816
1	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
2	S	0x87	1	1	0 512 bytes	0x4B	9.932 µs 13.536 315 250
3	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
4	S	0x87	1	1	0 512 bytes	0x4B	9.934 µs 13.536 325 182
5	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
6	S	0x87	1	1	0 512 bytes	0x4B	9.932 µs 13.536 355 350
7	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
8	S	0x87	1	1	1 512 bytes	0x4B	9.900 µs 13.536 365 282
9	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
10	S	0x87	1	1	0 512 bytes	0x4B	9.934 µs 13.536 375 182
11	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
12	S	0x87	1	1	0 512 bytes	0x4B	9.934 µs 13.536 385 116
13	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
14	S	0x87	1	1	0 512 bytes	0x4B	9.968 µs 13.536 440 282
15	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
16	S	0x87	1	1	0 512 bytes	0x4B	9.932 µs 13.536 450 250
17	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
18	S	0x87	1	1	0 512 bytes	0x4B	9.934 µs 13.536 460 182
19	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
20	S	0x87	1	1	1 512 bytes	0x4B	9.900 µs 13.536 470 116
21	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
22	S	0x87	1	1	0 512 bytes	0x4B	9.900 µs 13.536 520 150
23	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
24	S	0x87	1	1	0 512 bytes	0x4B	9.900 µs 13.536 540 016
25	H	OUT	ADDR	ENDP	T Data	ACK	Time Time Stamp
26	S	0x87	1	1	1 512 bytes	0x1B	14.968 µs 13.536 619 916
41	H	SOF	Frame #	CRC5	Pkt Len	Time	Time Stamp

A.12 Minimum Clock Frequencies: bus_clk, ram_clk

The controller requires a minimum bus_clk frequency to achieve the targeted USB data throughput of 4 Gbs, 8 Gbs, or more for multi-port hosts. Reducing the bus_clk frequency may decrease USB data throughput but does not cause functional problems.

The controller requires a minimum ram_clk frequency to ensure functional correctness (due to internal design requirements).

Figure A-7 Internal RAM, Bus Bandwidth and Clock Frequency Requirements



A.12.1 Recommended Configuration

1. DWC_USB3_SPRAM: Because USB traffic occurs concurrently with DMA traffic, it is recommended to select the two-port RAM configuration rather than the single-port RAM configuration. Otherwise, the minimum ram_clk frequency would double, and the DMA-side would incur additional wait states because the USB-side would have priority access of the single RAM port.
2. DWC_USB3_MDWIDTH: Due to the high USB 3.0 data rate and clock frequency requirements, it is recommended to select the 64-bit Master bus configuration.
3. DWC_USB3_NUM_RAMs: To achieve a lower minimum ram_clk frequency, select the 3 RAM configuration instead of the 2-RAM configuration. The 2-RAM configuration requires 15 MHz of additional RAM bandwidth to accommodate internal Cache RAM accesses.
4. Clocks: To avoid excess DMA wait states due to slow internal RAM access, ensure that the ram_clk frequency is greater than or equal to the bus_clk frequency.

A.12.2 Device bus_clk Frequency

For device, the bus_clk bandwidth requirement is equivalent to one SS instance. This is equivalent to 125 MHz @ 32-bit IN and 125 MHz @ 32-bit OUT for full-duplex system busses (AXI, Native), or 250 MHz @ 32-bit for half-duplex system buses (AHB).

A.12.3 Host bus_clk Frequency

For host, the bus_clk bandwidth requirement is equivalent to the sum of the bandwidth requirements for each SS and HS USB bus instance:

- SS BIs require 4 Gbs IN and 4 Gbs OUT. This is equivalent to 125 MHz @ 32b-it IN and 125 MHz @ 32-bit OUT for full-duplex system buses, or 250 MHz @ 32-bit for half-duplex system buses.
- HS BIs require 480 MB. This is equivalent to 15 MHz @ 32-bit for both full-duplex and half-duplex system buses.

A.12.4 Device ram_clk Frequency

For device, the minimum ram_clk frequency is 125 MHz, 62.5 MHz, and 60 MHz for MDWIDTH of 32, 64, and 128, respectively.

Though the device controller can function in MDWIDTH=128 configuration with a ram_clk frequency less than 60 MHz, the Tx turnaround time for USB 2.0 instances may not be met. Consequently, it is recommended to maintain a minimum ram_clk frequency of 60 MHz for device mode operation.

A.12.5 Host ram_clk Frequency

For the host, the RAM arbiter implementation grants RAM access to 2.0 instances (HS and FSLS) based on peak internal data rates, which are currently twice the USB 2.0 HS data rates of 480 MB. For this reason, the minimum ram_clk frequency depends on the number of 3.0 BIs (SS) and the number of 2.0 BIs (HS and FSLS), while allocating each 2.0 BI ram_clk domain bandwidth equivalent to one-fourth the bandwidth allocated to a 3.0 BI.

The host controller supports one to four SS instances, one to four HS instances, and one to four FSLS instances, where the number of HS instances is at least the number of FSLS instances. The number of configurations relating to these parameters is $4 * (4 + 3 + 2 + 1) = 40$, as shown in [Table A-15](#) on page [517](#).

Based on the number of the various speed bus instances, [Table A-15](#) shows the minimum bus_clk (AHB, AXI/Native) and ram_clk frequencies (for configurations of three two-port RAMs or three single-port

RAMs) for a master bus data width of 32, 64, or 128 bits in order to meet the full USB data throughput and peak RAM bandwidth requirements, respectively.

The recommended configurations are highlighted.



- If using a two-RAM configuration (instead of three RAMs), add 15 MHz to the indicated ram_clk frequency.
- Multiple SS BI configurations must use MDWIDTH = 64 or MDWIDTH = 128 due to the higher clock frequency requirement. Consequently, invalid configurations are left blank in the table.
- For a 64-bit MDWIDTH and any RAM configuration (one-, two- or three-RAM configuration), to operate in either SS-only or HS/FS/LS-only mode without performance consideration, the minimum ram_clk and bus_clk for achieving just the correct functionality is 62.5 MHz approximately.
- Use [Table A-15](#) to determine whether to enable GCTL.RAMClkSel to be 2/3 and have a lower bus_clk frequency. If the ram_clk is required (based on whether your configuration is less than or equal to the pipe clock available), you can enable RAMClkSel 2/3. Otherwise, always use the default value of ram_clk set to bus_clk; thus, bus_clk should follow the ram_clk requirements or higher.

Table A-15 Host: Minimum clock frequencies: bus_clk, ram_clk, Three RAMs

Number of BIs			Num 2.0 Ports	Minimum Clock Frequency (MHz)												
SS	HS	FSLS		MDW 32				MDW 64				MDW 128				
				AHB	AXI, Nat	3 2P RAM	3 SP RAM	AHB	AXI, Nat	3 2P RAM	3 SP RAM	AHB	AXI, Nat	3 2P RAM	3 SP RAM	
1	1	1	1	282	141	157	298	142	711	79 ^a	150	721	361	601	761	
1	2	1	2	314	157	188	345	158	791	94	173	801	401	601	87	
1	2	2	2	314	157	188	345	158	791	94	173	801	401	601	87	
1	3	1	3	344	172	250	422	172	861	125	211	861	431	631	106	
1	3	2	3	344	172	250	422	172	861	125	211	861	431	631	106	
1	3	3	3	344	172	250	422	172	861	125	211	861	431	631	106	
1	4	1	4	376	188	250	438	188	941	125	219	941	471	631	110	
1	4	2	4	376	188	250	438	188	941	125	219	941	471	631	110	
1	4	3	4	376	188	250	438	188	941	125	219	941	471	631	110	
1	4	4	4	376	188	250	438	188	941	125	219	941	471	63	110	
2	1	1	1					266	133	141	274	134	671	711	138	
2	2	1	2					282	141	157	298	142	711	791	150	
2	2	2	2					282	141	157	298	142	711	791	150	
2	3	1	3					298	149	188	337	150	751	941	169	
2	3	2	3					298	149	188	337	150	751	94	169	

Number of BLs			Num 2.0 Ports	Minimum Clock Frequency (MHz)												
SS	HS	FSLS		MDW 32				MDW 64				MDW 128				
				AHB	AXI, Nat	3 2P RAM	3 SP RAM	AHB	AXI, Nat	3 2P RAM	3 SP RAM	AHB	AXI, Nat	3 2P RAM	3 SP RAM	
2	3	3	3					298	149	188	337	150	751	94	169	
2	4	1	4					314	157	188	345	158	791	94	173	
2	4	2	4					314	157	188	345	158	791	94	173	
2	4	3	4					314	157	188	345	158	791	94	173	
2	4	4	4					314	157	188	345	158	791	94	173	
3	1	1	1						196	204	400	196	981	102	200	
3	2	1	2						204	219	423	204	1021	110	212	
3	2	2	2						204	219	423	204	1021	110	212	
3	3	1	3						211	235	446	212	1061	118	224	
3	3	2	3						211	235	446	212	1061	118	224	
3	3	3	3						211	235	446	212	1061	118	224	
3	4	1	4						219	250	469	220	1101	125	235	
3	4	2	4						219	250	469	220	1101	125	235	
3	4	3	4						219	250	469	220	1101	125	235	
3	4	4	4						219	250	469	220	1101	125	235	
4	1	1	1						258	266			258	129	133	262
4	2	1	2						266	282			266	133	141	274
4	2	2	2						266	282			266	133	141	274
4	3	1	3						274	300			274	137	150	287
4	3	2	3						274	300			274	137	150	287
4	3	3	3						274	300			274	137	150	287
4	4	1	4						282	313			282	141	157	298
4	4	2	4						282	313			282	141	157	298
4	4	3	4						282	313			282	141	157	298
4	4	4	4						282	313			282	141	157	298

a. This is the FPGA minimum clock frequency. For ASIC, the Host-mode's minimum Bus and RAM clock frequencies must be at least 125 MHz. However, if GCTL.RAMClkSel is set to 2/3, the bus_clk can be lower than 125 MHz.



Configurations with bus_clk lower than 60MHz are not verified, and must not be used.

A.12.6 Minimum AHB/AXI and RAM Clock Frequencies for 4Gbs/8Gbs Data Rate

This section contains tables for device and host that show the minimum bus_clk and ram_clk frequencies to achieve aggregate throughput of either 4Gbs or 8Gbs (single or bi-directional SS traffic) depending on the following configuration parameters:

- Device or 1-port Host (1 SS BI, 1 HS BI, and 1 FSLS BI)
- 4 Gbs or 8 Gbs aggregate OUT+IN data throughput
- 2 or 3 RAMs
- Single-port or 2-port RAMs
- MDWIDTH of 32, 64, or 128

Table A-16 Device: Minimum Clock Frequencies: bus_clk, ram_clk. 4Gbs/8Gbs Data Rate

OUT+IN Data Rate (Gbs)	Num RAMs	Num RAM ports	Minimum Clock Frequency (MHz)								
			MDW 32			MDW 64			MDW 128		
			AHB	AXI, Nat	RAM	AHB	AXI, Nat	RAM	AHB	AXI, Nat	RAM
4	2	1	125	125	265	62.5	62.5	140	31.3	31.3	77.5
4	2	2	125	125	140	62.5	62.5	77.5	31.3	31.3	46.3 ^a
4	3	1	125	125	250	62.5	62.5	125	31.3	31.3	62.5
4	3	2	125	125	125	62.5	62.5	62.5	31.3	31.3	31.31
8	2	1	250	125	265	125	62.5	140	62.5	31.3	77.5
8	2	2	250	125	140	125	62.5	77.5	62.5	31.3	46.31
8	3	1	250	125	250	125	62.5	125	62.5	31.3	62.5
8	3	2	250	125	125	125	62.5	62.5	62.5	31.3	31.31

a. This is the minimum clock frequency for FPGA implementation. For ASIC, the minimum device-mode RAM clock frequency is 60MHz.

Table A-17 Host (1 SS, 1 HS, 1 FSLS BI): Minimum Clock Frequencies: bus_clk, ram_clk. 4Gbs/8Gbs Data Rate

OUT+IN Data Rate (Gbs)	Num RAMs	Num RAM ports	Minimum Clock Frequency (MHz)								
			MDW 32			MDW 64			MDW 128		
			AHB	AXI, Nat	RAM	AHB	AXI, Nat	RAM	AHB	AXI, Nat	RAM
4	2	1	141	141	282	711	711	142	361	361	72 ^a
4	2	2	141	141	156	711	711	791	361	361	601
4	3	1	141	141	282	711	711	142	361	361	721
4	3	2	141	141	156	711	711	791	361	361	601
8	2	1	282	141	282	141	711	142	711	361	721
8	2	2	282	141	156	141	711	791	711	361	601

OUT+IN Data Rate (Gbs)	Num RAMs	Num RAM ports	Minimum Clock Frequency (MHz)								
			MDW 32			MDW 64			MDW 128		
			AHB	AXI, Nat	RAM	AHB	AXI, Nat	RAM	AHB	AXI, Nat	RAM
8	3	1	282	141	282	141	711	142	711	361	721
8	3	2	282	141	156	141	711	791	711	361	601

a. This is the minimum clock frequency for FPGA. For ASIC, the Host-mode's minimum bus and RAM clock frequencies must be at least 125 MHz.

A.12.7 Minimum Clock Frequencies (MHz) for USB 2.0-Only Mode

Table A-18 summarizes the minimum clock frequencies (MHz) for USB 2.0-only mode.

Table A-18 Minimum Clock Frequencies (MHz) for USB 2.0-Only Mode

Number of USB 2.0 Ports	MDWIDTH 32				MDWIDTH 64				MDWIDTH 128			
	bus_clk		ram_clk		bus_clk		ram_clk		bus_clk		ram_clk	
	AHB	AXI, Native	2-Port RAM	Single Port RAM	AHB	AXI, Native	2-Port RAM	Single Port RAM	AHB	AXI, Native	2-Port RAM	Single Port RAM
1	60	60	60	60	60	60	60	60	60	60	60	60
2	60	60	60	75	60	60	60	60	60	60	60	60
3	60	60	60	105	60	60	60	60	60	60	60	60
4	60	60	75	135	60	60	60	75	60	60	60	60

B

Clock Domain Crossing

This appendix discusses the types of Clock Domain Crossing (CDC) signals, CDC implementation and validation, and the types of synchronizer modules used in DWC_usb3 controller.

The topics discussed in this appendix are as follows:

- “[Types of Clock-Crossing Signals](#)” on page [522](#)
- “[CDC Implementation and Validation](#)” on page [523](#)
- “[Negedge-Sampled Signals and SDF Annotated Timing Check Off Flops](#)” on page [527](#)
- “[Synchronizer](#)” on page [527](#)

The DWC_usb3 controller has the following clock domains:

- User bus-specific bus_clk
- USB 2.0 utmi_clk and/or ulpi_clk
- SuperSpeed pipe3_rx/tx_pclk
- Low power mode suspend_clk
- Internal RAM domain ram_clk
- ref_clk

B.1 Types of Clock-Crossing Signals

The DWC_usb3 controller has the following types of clock-crossing signals.

1. Single-bit control signals
 - Active high or active low single-bit control signal synchronized using double synchronization.
 - Example: Idle indication from mac clock domain to bus clock domain
 - The *src/com/DWC_usb3_sync_ctl.v* is used for this (double-sync mode) purpose
 - Toggle type signal (low to high or high to low transition indicates valid) synchronized from one clock domain to another using three flops, and the last two stage outputs are XORED to create an active high valid signal in the destination clock domain. This is one of the fastest synchronizer between any two clock domains without any frequency assumption.
 - Example: Interrupt indication from mac clock domain to bus clock domain
 - The *src/com/DWC_usb3_sync_ctl.v* is used for this (toggle mode) purpose
2. Multi-bit data synchronization
 - Gray pointer synchronized from one domain to another domain using double synchronization. This is the fastest multi-bit synchronization. Multi-bit DWC_usb3_sync_ctl is used for this purpose.
 - Example: 2-clock FIFOs use this synchronization
 - Multi-bit *src/com/DWC_usb3_sync_ctl.v* is used for this (double sync mode) purpose
 - Bus synchronizer: On input data change, the data is registered in the source domain and is kept until the data is transferred to the destination domain. The internal handshake signaling guarantees the data is stable when the data is sampled in the destination domain. Due to the handshake nature, this is a slower synchronizer.
 - Example: The “SOF/ITP” counters use this synchronization
 - The *src/com/DWC_usb3_bussync.v* is used for this purpose
3. Passing multi-bit data along with a toggle control signal from one domain to another domain
 - Example: Passing write data along with write control signal from bus clock domain to RAM clock domain.
 - The *src/com/DWC_usb3_sync_toggledata.v* is used for this purpose
4. Pulse synchronizer to pass a single-clock pulse signal from one domain to another domain
 - Example: Passing usb reset pulse from mac to bus clock domain
 - The *src/com/DWC_usb3_sync_pulse.v* is used for this purpose
5. Dummy data synchronizer for static or pseudo-static multi-bit data signals

This synchronizer has delay randomization for simulation purposes and there is no logic associated with it. The pseudo-static signals are expected to be stable when sampled, by design. The different types of these signals are configuration register bits which are one time programmed before normal operation, and the data signals that are passed along with the control signals and are expected to be stable and valid when the control signal is sampled on the other domain.

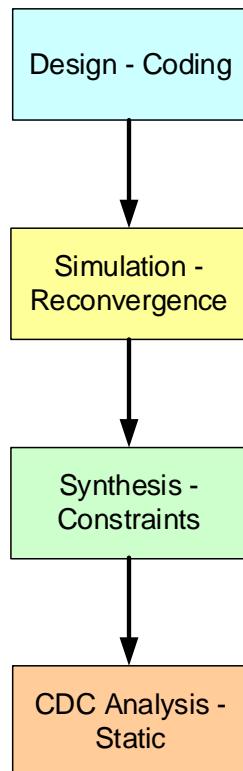
 - Example: GUSB2PHYCFG register bits from RAM clock domain to mac clock domain
 - The *src/com/DWC_usb3_sync_data.v* is used for this purpose

B.2 CDC Implementation and Validation

Careful considerations are made for the following different stages to ensure successful operation of clock domain crossing:

- Design
- Simulation – dynamic delay randomization for re-convergence check
- Synthesis – constraint to meet design requirement and to match the verified simulation convergence delays
- Analysis – static analysis

Figure B-1 CDC Flow

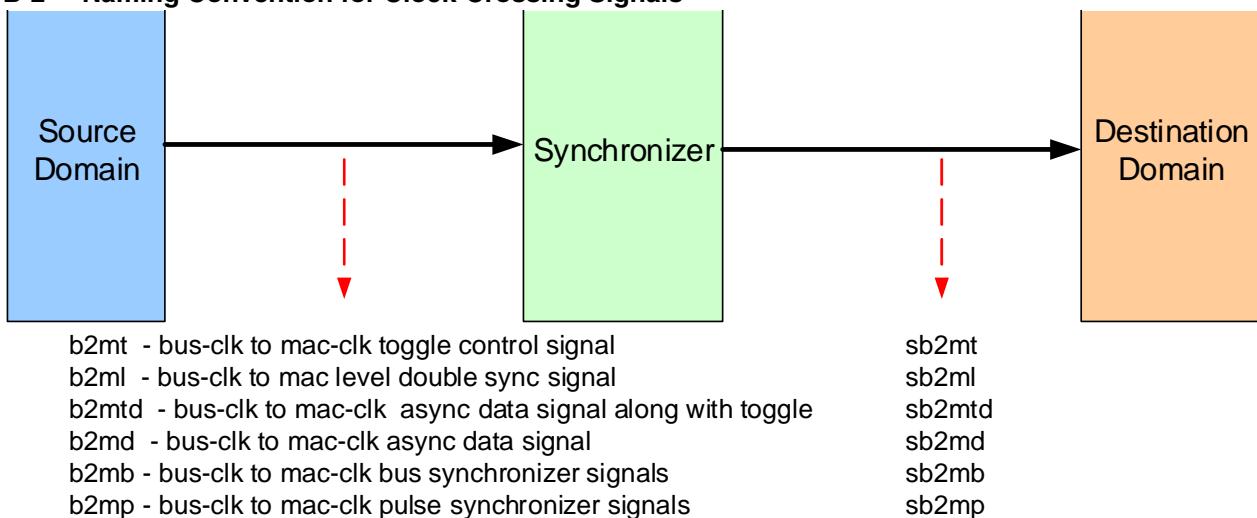


B.2.1 Design Flow

The following design guidelines are used:

1. All the clock-crossing signals go through the standardized synchronization modules. These modules are in the src/com directory. The standard synchronizers:
 - provide consistency
 - avoid design errors and are easy to review
 - have built-in delay randomization for simulation
 - make it easy to identify signals for exception filtering during CDC analysis
2. All the clock-crossing signals have a prefix to identify the signal type. The prefix helps in identifying the signals during design, review, and analysis phase. This also enables exception filtering using the signal prefixes. For example:
 - The source clock domain signal prefixes are:
 - b2mt: bus_clk to mac_clk toggle control signal
 - b2ml: bus_clk to mac level double sync signal
 - b2mtd: bus_clk to mac_clk asynchronous data signal along with toggle
 - b2md: bus_clk to mac_clk asynchronous data signal
 - b2mb: bus_clk to mac_clk bus synchronizer signals
 - b2mp: bus_clk to mac_clk pulse synchronizer signals
 - The synchronized destination clock domain prefixes are:
 - sb2mt: bus_clk to mac_clk toggle control signal - synchronized
 - sb2ml: bus_clk to mac level double sync signal - synchronized
 - sb2mtd: bus_clk to mac_clk asynchronous data signal along with toggle - synchronized
 - sb2md: bus_clk to mac_clk asynchronous data signal - synchronized
 - sb2mb: bus_clk to mac_clk bus synchronizer signals - synchronized
 - sb2mp: bus_clk to mac_clk pulse synchronizer signals - synchronized

Figure B-2 Naming Convention for Clock-Crossing Signals



B.2.2 Simulation Flow

The following verification flow is used to identify CDC re-convergence problems during verification:

1. The clock-crossing signal delays are randomized from 0, 0.5, and 1.0 clock period to find functional and CDC convergence problems. All the bits of a bus signal are randomized independently. This is enabled by setting DWC_USB3_SYNC_VERIF_EN=1 in the src/DWC_usb3_params.v during simulation.
2. In simulation, the clocks are simulated at minimum, maximum, typical, and random frequencies (between minimum and maximum) to find out any functional issues.

For example:

```
// Generation of BUS clock
initial begin
bus_clk_early = 1'b0;
forever begin
#(`BUS_CLK_PERIOD/2)
bus_clk_early = ~bus_clk_early;
#(`BUS_CLK_PERIOD - `BUS_CLK_PERIOD/2)
bus_clk_early = ~bus_clk_early;
end
end
```

3. On the unrelated clocks, the offset of the clocks are randomized to catch any functional issue.

For example:

```
// Generation of PIPE3-125 clock
reg [31:0] pipe3_pclk_125_offset;
initial begin
pipe3_pclk_125 = 1'b0;
pipe3_pclk_125_offset = $random % `PIPE3_CLK_PERIOD_125;
#(pipe3_pclk_125_offset);
forever begin
#(`PIPE3_CLK_PERIOD_125/2)
pipe3_pclk_125 = ~pipe3_pclk_125;
#(`PIPE3_CLK_PERIOD_125 - `PIPE3_CLK_PERIOD_125/2)
pipe3_pclk_125 = ~pipe3_pclk_125;
end
end
```

B.2.3 Synthesis Flow

Earlier, the methodology was to set false paths between the clock domains. This can cause re-convergence issues if two functionally-related signals are crossing from one domain to another domain. In a highly congested ASIC, the backend tools could do scenic routing if there are no constraints on the clock-crossing signals (false path) causing functional failures.

The following synthesis methodology is required for achieving successful functional operation of the controller:

1. Do not set false path between the clock domains. Setting false paths between the clock domains cause scenic routing.
Example: `set_false_path -from bus_clk -to mac_clk`. This is not recommended.
2. Set `max_delay` constraint between the clock domains. This avoids scenic routing. The `max_delay` should match the maximum allowed skew between two related signals by design and also the maximum delays used in the re-convergence simulation.
3. For control and data signals which cross clock domains 1-clock destination clock period is used for `max_delay`. For example, when passing a control and data signal together from one clock domain where the control signal gets double synchronized, this methodology guarantees that the data is still valid even if the data has a larger routing delay than the control signal.

For example:

```
set_max_delay [expr 1.0 * $ram_clk_pe] -from [get_ports U_DWC_usb3_bmu/U_DWC_usb3_2clkfifo/U_DWC_usb3_sync_ctl_0/*U_DWC_usb3_double_sync*/in_p*] -to [all_registers -clock bus_clk]
```

4. For gray pointers, 1-clock source clock period is used. This guarantees that the signals arrive at the destination before the next change and one-hot signaling is valid. In addition, this also makes sure that the 2-clock FIFO storage depth is minimal and is optimally used. If the routing delays are large, then the pointer update will take longer and hence a larger FIFO would be needed in order to hold the data if wait states cannot be tolerated.

For example:

```
set_max_delay [expr 1.0 * $ram_clk_pe] -from [get_ports U_DWC_usb3_bmu/U_DWC_usb3_2clkfifo/U_DWC_usb3_sync_ctl_0/*U_DWC_usb3_double_sync*/in_p*] -to [all_registers -clock bus_clk]
```

5. Additional Static Timing Analysis (STA) considerations

When `max_delay` constraints are set between clock domain crossing signals, the STA tools also add the clock-tree insertion delay for timing analysis. One way to avoid this is to have same clock tree insertion on all the clocks. Another option is to add the difference in the clock-tree insertions to the `max_delay` constraint.

B.2.4 CDC Analysis Flow

CDC analysis is done through Spyglass.

The DesignWare USB 3.0 Controller has different functional modes (Host, USB 2.0, USB 3.0, and so on). Spyglass analysis needs to be done for these functional modes, but only one mode for a given run. There also can be sub-functional modes. For instance, the `ram_clk` can be programmed to be `bus_clk`, `mac2_clk`, or `pipe_clk`. This increases the combinational space for performing the analysis. This is the same as performing `set_case_analysis`.

For details on running Spyglass and the list of analysis modes that are currently supported, refer to the “Running Spyglass Lint” section in the *DWC SuperSpeed USB 3.0 User Guide*.

B.3 Negedge-Sampled Signals and SDF Annotated Timing Check Off Flops

In `DWC_usb3` controller usage, the first stage of the `DWC_usb3_sync_ctl` and `DWC_usb3_sync_toggledata` synchronizers are configured for negedge sampling and the second stage is configured for posedge sampling. This is to minimize the synchronization delays in order to meet the USB 2.0 and USB 3.0 turnaround times and also to reduce the depth of the clock-crossing FIFOs.

1. The negedge sampled first stage synchronization flops are:
 - `src/com/DWC_usb3_sync_ctl.v/*U_bcm41_w_async_rst*/'U_SYNC*/sample_meta_n`
 - `src/com/DWC_usb3_sync_toggledata/in_toggle_1d`
2. The clock-crossing flops are listed below. These would need setup and hold time check exception in a SDF annotated netlist simulation.
 - `src/com/DWC_usb3_sync_ctl.v/*U_bcm41_w_async_rst*/'U_SYNC*/sample_meta_n`
 - `src/com/DWC_usb3_sync_toggledata/* U_bcm21_toggle*/sample_meta_n`
 - `src/com/DWC_usb3_sync_toggledata/out_data_reg`
 - `src/com/DWC_usb3_bussync/d_data`
 - `src/com/DWC_usb3_sync_2edge.v/in_p_1d`
 - `src/pwrm/DWC_usb3_pwrm_u3piu/pipe3_sync_9b_32`

B.4 Synchronizer

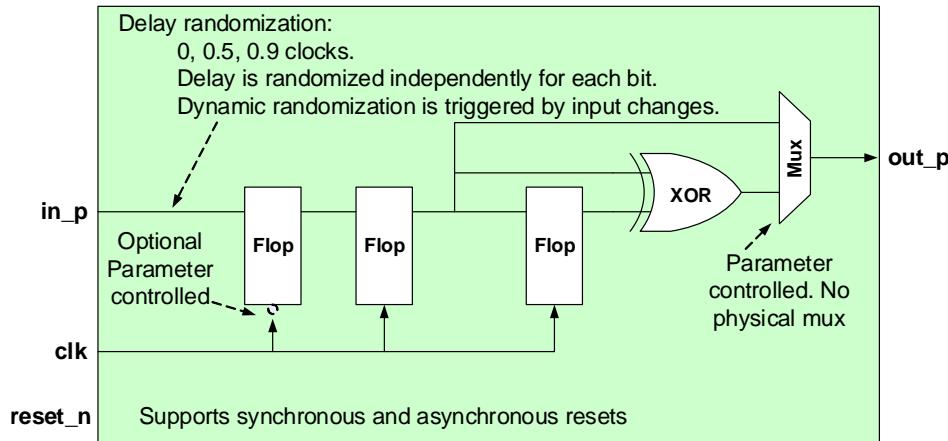
This section discusses the various clock-crossing synchronizer modules used in the `DWC_usb3` design.

You can choose to configure synchronizers to use posedge clocks for both the first and second stages (instead of the default first-stage negedge). However, if you select all posedge clocks, the device mode minimum `ram_clk` frequency increases from 60 MHz to 100 MHz to accommodate the increased turnaround time latency.

B.4.1 Control Synchronizer (src/com/DWC_usb3_sync_ctl.v)

The DWC_usb3_sync_ctl.v is used for double synchronization and toggle synchronization. It is also used in multi-bit mode for gray pointer synchronization.

Figure B-3 Control Synchronizer



Parameters:

- DATA_WIDTH: Data Width
- SYNC_TYPE:
 - 0 – Bypass
 - 1 – Posedge toggle
 - 2 – 1st stage negedge and 2nd stage posedge toggle
 - 3 – Posedge double sync
 - 4 – First stage negedge and second stage posedge double-sync
- RAND_TYPE:
 - 0 – 0, 0.5, 1.0, 1.5, 1.9 clocks (not used)
 - 1 – 0, 0.5, 0.9 clocks
- ENABLE_DELAY: 1 – Enables delay randomization for verification re-convergence check
- CLOCK_PERIOD: Clock period; Timescale dependent (default is 0)
- INIT_VALUE:
 - 0 – Initialization value of 0
 - 1 – Initialization value of {DATA_WIDTH{1'b1}}
- RST_FORCE:
 - 0 – Reset Mode defined by DWC_USB3_CLK_RST_MODE
 - 1 – Force Asynchronous Reset

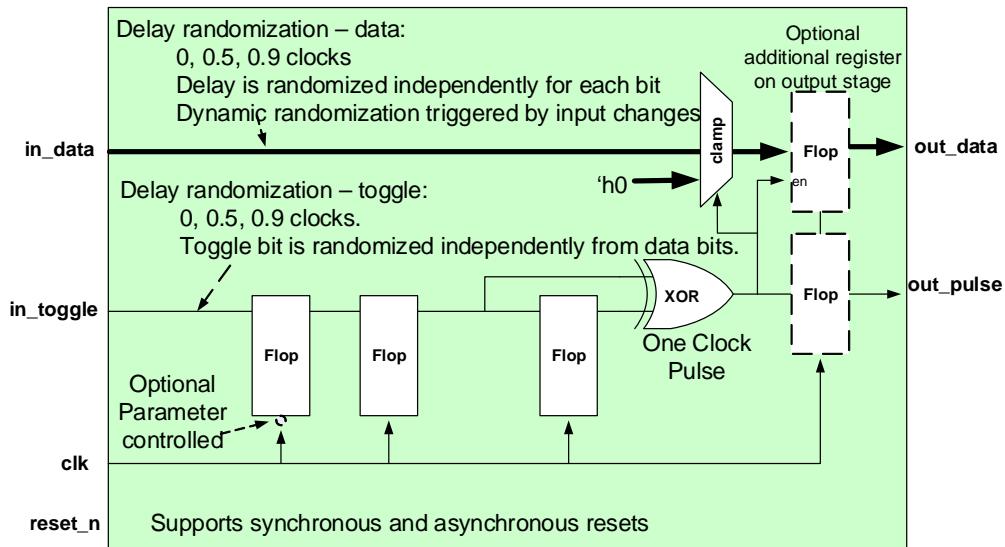
Features:

- The output is available as toggle or double-sync level control signals. Toggle provides the fastest clock crossing method and is independent of source and destination domain frequencies.
- This synchronizer is used to pass control signals between two clock domains without associated data
- Toggle control output is only available as a pulse type. This avoids the multiple usage model where data validity is signaled by pulse or toggle.
- Each control-bit is randomized independently to avoid potential re-convergence problem. This also allows reduction in code size by using one multi-bit instance to synchronize multiple unrelated signals.
- Delay randomization options of 0, 0.5, 0.9 clocks conforms to the 1.0 clock max_delay synthesis constraint.
- Each double flop is instanced as a leaf cell module for easy replacement into tech-specific double-flops.

B.4.2 Toggle Control and Data Synchronizer (src/com/DWC_usb3_sync_toggledata.v)

The DWC_usb3_sync_ctl.v is used for passing multi-bit data along with toggle control signal from one clock to another clock.

Figure B-4 Toggle Control and Data Synchronizer



Parameters:

- DATA_WIDTH: Data Width
- SYNC_TYPE:
 - 0 – Bypass
 - 1 – Posedge toggle
 - 2 – First stage negedge and Second stage posedge
- RAND_TYPE:
 - 0 – 0, 0.5, 1.0, 1.5, 1.9 clocks (not used)

- 1 – 0, 0.5, 0.9 clocks
- ENABLE_DELAY: 1 – Enables delay randomization for verification re-convergence check
- CLOCK_PERIOD: Clock period; Timescale dependent (default is 0)
- REGISTER_OUT: 1 – Add additional register in the output stage
- INIT_VLAUE:
 - 0 – Initialization value of 0
 - 1 – Initialization value of {DATA_WIDTH{1'b1}}
- DEBUG_EN: 1 – Enable debug display message

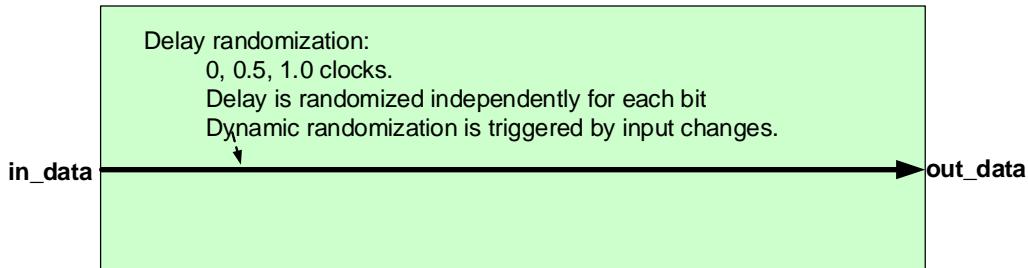
Features:

- Toggle is independent of source and destination domain frequencies.
- Data and Control are contained in the same module for ease of association.
- Control output is only available as a pulse. Pulse only availability avoids the multiple usage model, where data validity is signaled by pulse or toggle.
- In un-registered mode, output data is valid for one clock period. Gating invalid output avoids synthesis dependent glitch propagation that from result from changes to data between control pulses.
- The control-bit and each data-bit is independently randomized to avoid potential re-convergence problems.
- Delay randomization options (0, 0.5, 0.9 clocks) conform to the 1.0 clock max_delay synthesis constraint.
- Leaf cell double sync is not instanced for toggle sync double flops in this release.

B.4.3 Data Synchronizer (src/com/DWC_usb3_sync_data.v)

The DWC_usb3_sync_data.v is used for pseudo data synchronization. There is synthesizable logic associated with this synchronizer. It only introduces random delays in simulation mode.

Figure B-5 Data Synchronizer



Parameters:

- DATA_WIDTH: Data Width
- SYNC_TYPE:
 - 0 – Bypass
 - 1 – Normal
- RAND_TYPE:
 - 0 – 0, 0.5, 1.0, 1.5, 1.9 clocks (not used)
 - 1 – 0, 0.5, 0.9 clocks
- ENABLE_DELAY: 1 – Enables delay randomization for verification re-convergence check
- CLOCK_PERIOD: Clock period Timescale dependent (default is 0)

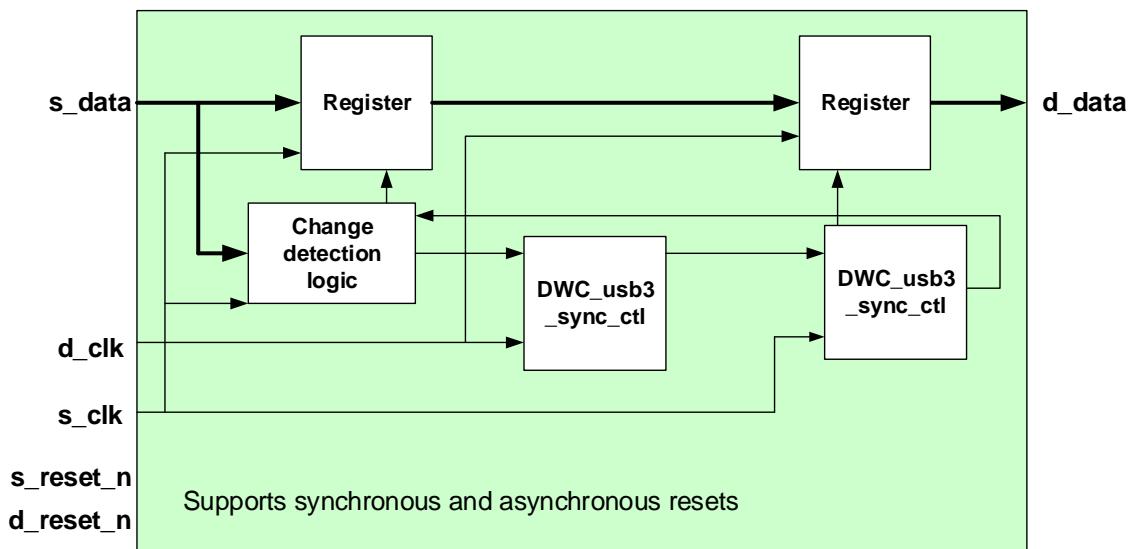
Features:

- Each data bit is randomized independently to avoid potential re-convergence problem.
- Delay randomization options (0, 0.5, 0.9 clocks) conform to the 1.0 clock max_delay synthesis constraint.

B.4.4 Bus Synchronizer (src/com/DWC_usb3_bussync.v)

The DWC_usb3_bussync.v is used for synchronizing multi-bit bus from one domain to another domain. On any input change, the synchronizer registers the input value and passes it to the destination domain register. If the input data changes in the middle of the transfer the synchronizer will complete the previous data transfer before re-synchronizing the new data.

Figure B-6 Bus Synchronizer



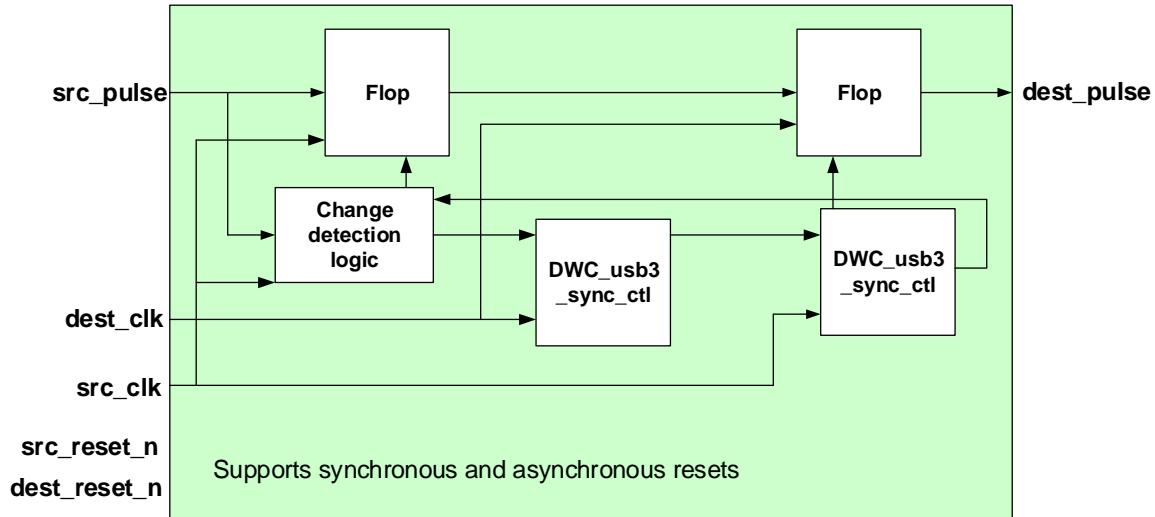
Parameters:

- DATA_WIDTH: Data Width
- SYNC_TYPE:
 - 0 – Bypass
 - 1 – Posedge toggle
 - 2 – First stage negedge and Second stage posedge
- RAND_TYPE:
 - 0 – 0, 0.5, 1.0, 1.5, 1.9 clocks (not used)
 - 1 – 0, 0.5, 0.9 clocks
- ENABLE_DELAY: 1 – Enables delay randomization for verification re-convergence check
- S_CLK_PERIOD: Source clock period; Timescale dependent (default is 0)
- D_CLK_PERIOD: Destination clock period; Timescale dependent (default is 0)
- INIT_VLAUE:
 - 0 – Initialization value of 0
 - 1 – Initialization value of {DATA_WIDTH{1'b1}}

B.4.5 Pulse Synchronizer (src/com/DWC_usb3_sync_pulse.v)

The DWC_usb3_sync_pulse.v is used for synchronizing a single clock pulse from the source domain to a single clock pulse in the destination domain.

Figure B-7 Pulse Synchronizer



Parameters:

- DATA_WIDTH: Data Width
- SYNC_TYPE:
 - 0 – Bypass
 - 1 – Posedge toggle
 - 2 – First stage negedge and second stage posedge
- RAND_TYPE:
 - 0 – 0, 0.5, 1.0, 1.5, 1.9 clocks (not used)
 - 1 – 0, 0.5, 0.9 clocks
- ENABLE_DELAY: 1 – Enables delay randomization for verification re-convergence check
- S_CLK_PERIOD: Source clock period; Timescale dependent (default is 0)
- D_CLK_PERIOD: Destination clock period; Timescale dependent (default is 0)

C

High Speed Inter-Chip (HSIC) Feature

This appendix describes the High Speed Inter-Chip (HSIC) feature supported by the DWC_usb3 controller.



Note An additional license key is required for this add-on.

The topics discussed in this appendix are as follows:

- “[HSIC Operation Model](#)” on page [535](#)
- “[Limitations](#)” on page [541](#)
- “[Operational Restrictions in HSIC Mode](#)” on page [541](#)

C.1 HSIC Operation Model

The timing diagrams described in this section explain the UTMI interface signals with the PHY for the following USB scenarios when the HSIC feature is enabled.

- Attach Sequence
- Suspend/Resume
- Suspend/Remote Wakeup

C.1.1 Attach Sequence

This section describes the attach sequence.

Host Mode

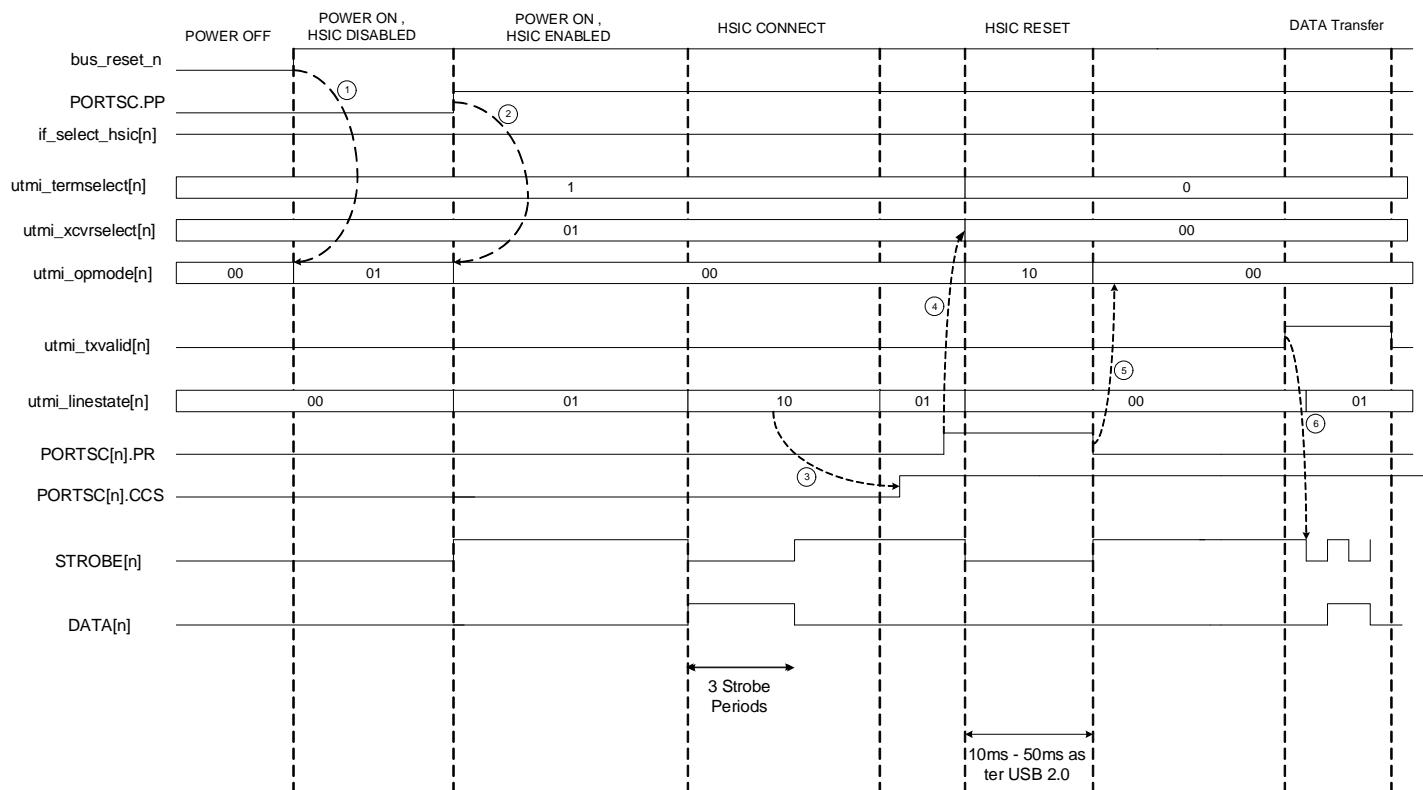
- Enable the HSIC feature by selecting “Yes” for “Enable HSIC Support for USB 2.0 Ports?” configuration parameter. In addition, set the value of GUSB2PHYCFGn[26] bit. The PORTSC.PP is high by default after the power-on reset. After this step, the HSIC port power is ON, and the port is enabled to connect. The HSIC PHY drives the HSIC Idle condition onto the STROBE/DATA lines. Refer to section 3.1.3 of the HSIC Specification, version 1.0.
- When the K-state is detected on the line state, the HSIC connection is established.?

Device Mode

- Enable the HSIC feature by selecting “Yes” for “Enable HSIC Support for USB 2.0 Ports?” configuration parameter. In addition, set the value of GUSB2PHYCFGn[26] bit. After this step, the device waits for the DCTL.Run/Stop to be set before looking for the HSIC Idle condition.
- When the DCTL.Run/Stop is set, the device waits for the HSIC Idle condition on the bus. Once Idle is detected, the device starts driving the K-state for three STROBE periods for a HSIC Connect.

Figure C-1 on page 536 explains the attach sequence.

Figure C-1 Attach Sequence



1. Host: When the power-on reset is done, DWC_usb3 host controller looks at if_select_hsic, INV_SEL_HSIC, HSIC_CONNECT and PORTSC.PP. If HSIC feature is selected ((if_select_hsic[n] ^ INV_SEL_HSIC[n]) = 1) and if the HSIC is not enabled to connect (PORTSC.PP=0), the controller drives utmi_opmode to non-driving mode (2'b01) in order to put the controller into HSIC-disabled state.? Device: When the power-on reset is done, DWC_usb3 device controller looks at if_select_hsic, INV_SEL_HSIC and DCTL.Run/Stop. If HSIC feature is selected ((if_select_hsic[n] ^ INV_SEL_HSIC[n]) = 1) and if the HSIC is not enabled to connect (DCTL.Run/Stop = 0), the controller waits for the DCTL.Run/Stop to be set. It does not connect until the DCTL.Run/Stop is set.

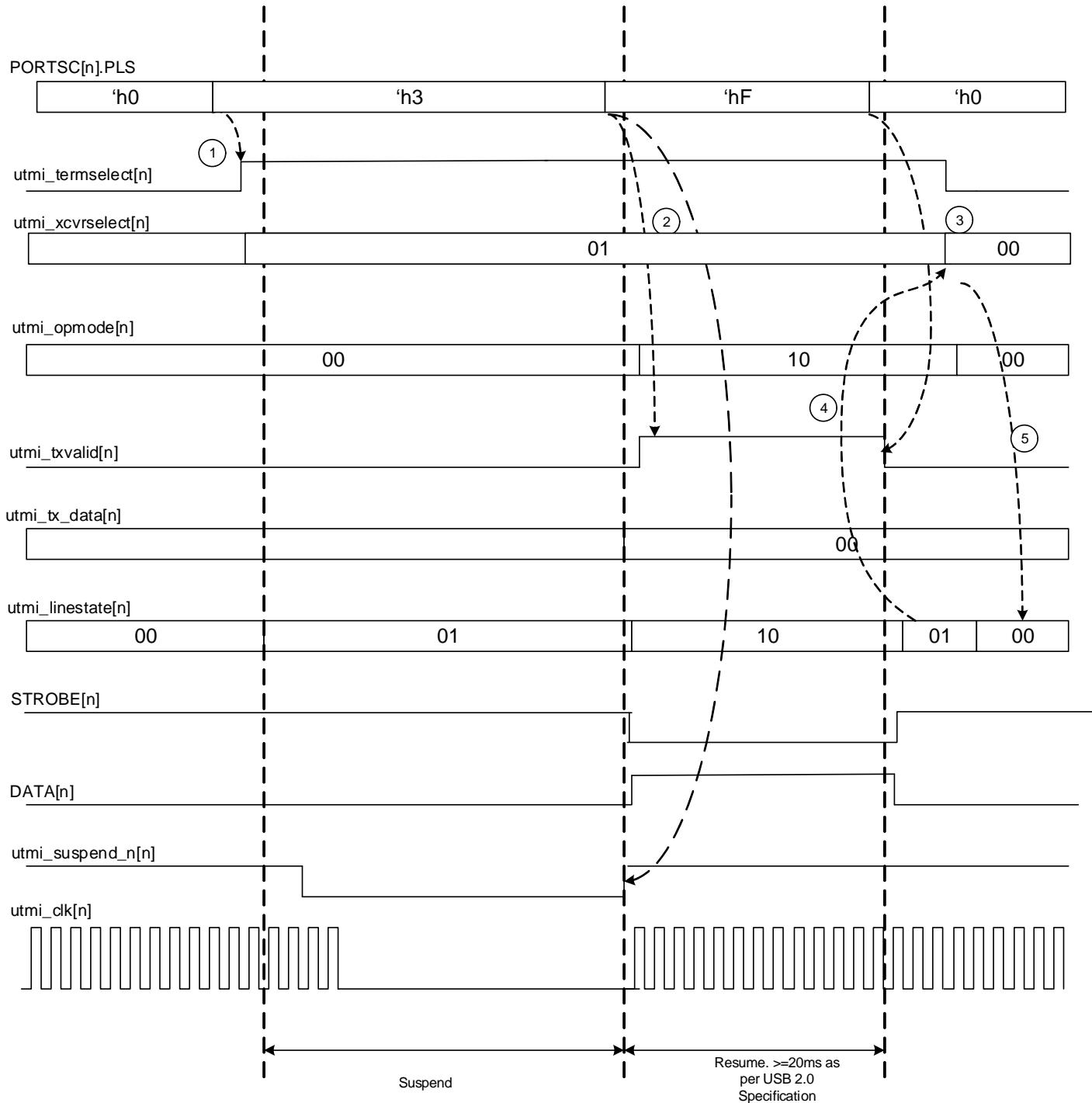
2. Host: When the host controller is enabled to connect (PORTSC.PP is 1), the utmi_opmode is driven to 2'b00 to put the controller into HSIC-enabled state. In this state the controller looks for any connect from the device.? Device: Once the DCTL.Run/Stop is set, the device waits for the HSIC Idle condition before driving the HSIC connect sequence.?

3. Device: Once the HSIC Idle is detected, the device drives the K-state for three strobe periods for the device connect sequence.
Host: On detecting a connect (linestate change from 2'b01 to 2'b10), the controller sets the port connect status bits (PORTSC.CCS/CSC). On this interrupt, the host driver sets the port reset bit (PORTSC.PR).
4. Host: On detecting the port reset bit (PORTSC.PR) set, the host controller drives a reset (SE0) on the bus. The controller drives SE0 until the reset is done. (Setting and Clearing of PORTSC.PP bit must be according to the xHCI Specification 1.0)
Device: The device detects the reset on the HSIC bus and goes to reset state.
5. Host, Device: When the port reset duration reached, the host controller stops driving SE0 by setting the utmi_opmode to 2'b00. The device completes the reset process.
6. Host, Device: The host and device do the normal data transfer.

C.1.2 Suspend/Resume

Figure C-2 shows the suspend and resume sequence.

Figure C-2 Suspend/Resume Sequence

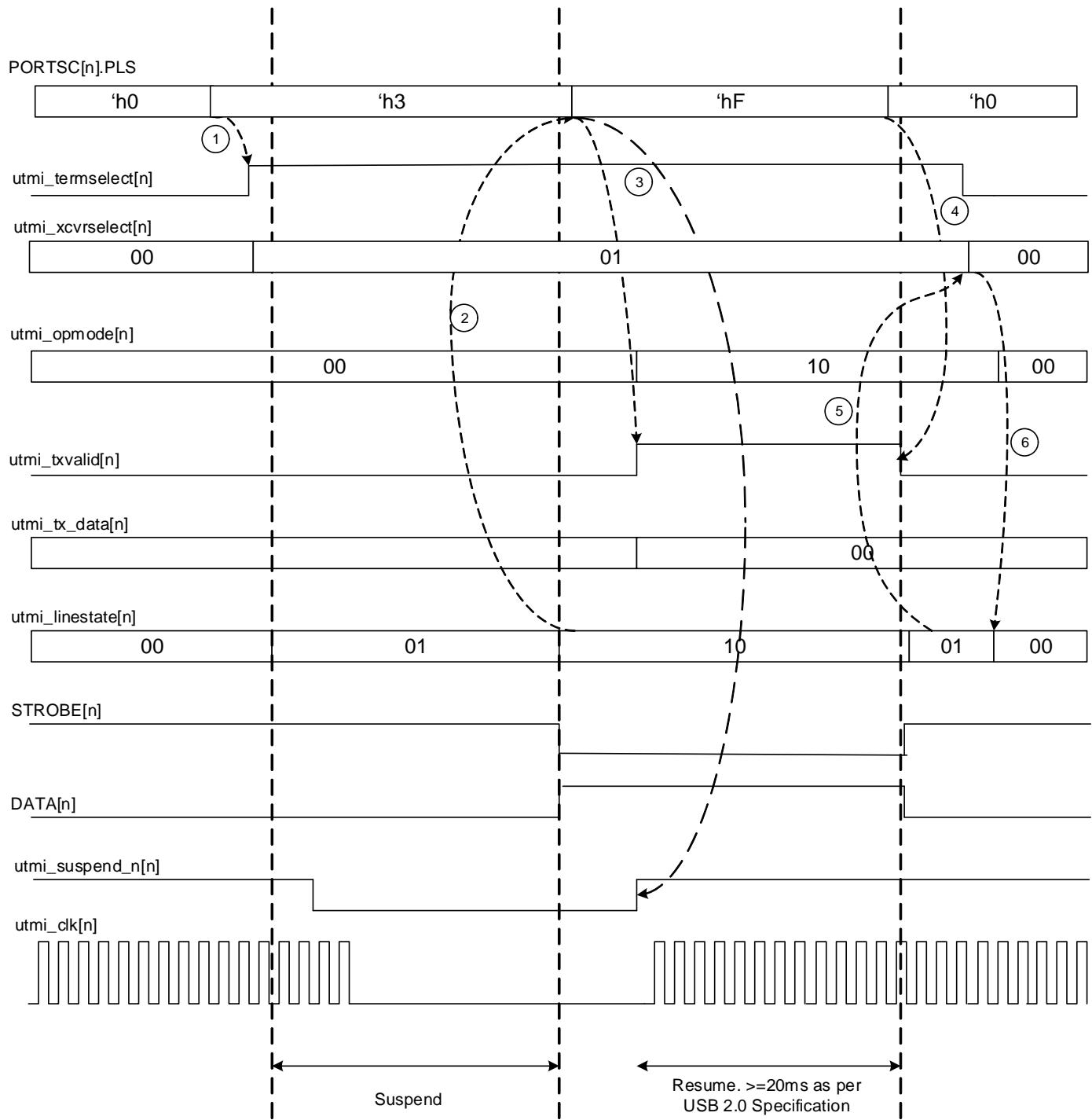


1. When the host driver sets the PORTSC[n].PLS to 3, the controller drives utmi_termselect[n] and utmi_x_cvrselect[n] to 1'b1 and 2'b01, respectively. The utmi_linestate[n] becomes 2'b01 to indicate suspend. The device goes to suspend after 3ms because there is no SOF sent. The host/device controller drives the utmi_suspend_n[n] LOW which puts the PHY into suspend mode drawing suspend current only and shutting off the UTMI clock. The setting of PORTSC[n].PLS is according to the xHCI Specification 1.0.
2. The host driver sets the PORTSC[n].PLS to 15 to initiate the resume. The host de-asserts the utmi_suspend_n[n] to bring the PHY out of suspend condition and start the clock. The controller drives utmi_tx_vaid[n] high with utmi_tx_data[n] as zeros and switches utmi_opmode[n] to 2'b10. The setting of PORTSC[n].PLS is according to the xHCI Specification 1.0.
The device controller detects the resume signaling and de-asserts the utmi_suspend_n to bring the PHY out of suspend.
3. When the host driver sets the PORTSC[n].PLS to 0 (or when it gets cleared internally by the controller in LPM), the controller de-asserts utmi_txvalid[n]. The utmi_opmode[n], utmi_xcvrselect[n] and utmi_termselect[n] change to zeros after 2usec delay.
4. The host controller de-asserting utmi_txvalid[n] (end of resume) causes both host and device PHY to change the utmi_linestate[n] from 2'b10 to 2'b01 or to 2'b00 directly. On detecting linestate as 2'b01 or 2'b00, the host/device controller goes back to HighSpeed USB IDLE state by de-asserting utmi_termselect[n] and utmi_xcvrselect[n].
5. When utmi_termselect[n] and utmi_xcvrselect[n] are de-asserted, utmi_linestate[n] stays in 2'b00 indicating HS Idle.

C.1.3 Suspend/Remote Wakeup

Figure C-3 shows the suspend and remote wakeup sequence.

Figure C-3 Suspend/Remote Wakeup



1. When the host driver sets the PORTSC[n].PLS to 3, the controller drives utmi_termselect[n] and utmi_xcvrselect[n] to 1'b1 and 2'b01 respectively. The utmi_linestate[n] becomes 2'b01 to indicate suspend. The device goes to suspend after 3ms since there is no SOF sent. The host or device controller drives the utmi_suspend_n[n] LOW which puts the PHY into suspend mode drawing suspend current only and shutting off the UTMI clock. The setting of PORTSC[n].PLS is according to the xHCI Specification 1.0.
2. When the device drives the remote wakeup, the utmi_linestate[n] changes from 2'b01 to 2'b10 at the host side.
3. By sensing the utmi_linestate[n], the host controller sets PORTSC[n].PLS to 15 to initiate resume. The host controller drives utmi_suspend_n[n] HIGH to bring the PHY out of the suspend mode so that it draws normal current and bring the UTMI clock up. The host controller also drives utmi_txvalid[n] with utmi_tx_data[n] as zeros to continue resume on the bus.
4. When the host driver sets the PORTSC[n].PLS to 0 (or when it gets cleared internally by the controller in LPM), the controller de-asserts utmi_txvalid[n]. The utmi_opmode[n], utmi_xcvrselect[n] and utmi_termselect[n] changes to zeros after 2usec delay.
5. The host controller de-asserting utmi_txvalid[n] (end of resume) causes both host and device PHY to change the utmi_linestate[n] from 2'b10 to 2'b01 or to 2'b00 directly. On detecting the linestate as 2'b01 or 2'b00, the host or device controller goes back to HighSpeed USB IDLE state by de-asserting utmi_termselect[n] and utmi_xcvrselect[n].
6. When utmi_termselect[n] and utmi_xcvrselect[n] are de-asserted, utmi_linestate[n] stays in 2'b00 indicating HS Idle.

C.2 Limitations

This section discusses the switching between UTMI and ULPI interface when HSIC is enabled. If both UTMI and ULPI interfaces are supported when the HSIC is enabled, the following sequence of operation is recommended.

- Host Mode: The switching between UTMI and ULPI is supported only once. Before switching the PHY selection (using GUSB2PHYCFGn[4]), disable the HSIC feature ((if_select_hsic[n] ^ INV_SEL_HSIC[n]) = 0) for both host and device. Enable the HSIC again after the UTMI/ULPI switching. The host side should be set to HSIC mode first. The new device must be able to connect when the Host HSIC is enabled to connect after the PHY selection.
- Device Mode: Switching between the UTMI and ULPI is not supported when the HSIC is enabled without DCTL[30] soft reset. The soft reset needs to be done after the device UTMI/ULPI selection, while the Host HSIC is selected ((if_select_hsic[n] ^ INV_SEL_HSIC[n]) = 1) but not enabled to connect (that is, PORTSC.PP=0). This assumes two separate instances of DWC_usb3 HSIC controller, one in Host mode and the other in Device mode?

C.3 Operational Restrictions in HSIC Mode

The following are the operational restrictions in HSIC mode (GUSB2PHYCFG.INV_SEL_HSIC XOR if_select_hsic).

- The controller does not support the USB 2.0 test modes when HSIC mode is enabled. The test mode is intended only for cable-based USB 2.0 PHYs.
- After a HSIC connection, the disconnect feature is not supported from the host side and the soft disconnect is not supported from the device side. This is the intended behavior of the HSIC Specification, version 1.0.

D

Fixed ECNs

The USBIF and Intel revise the USB 3.0, xHCI, and USB 2.0 specifications through the ECN/Specification. The DWC_usb3 controller has been enhanced to support these ECNs or specification changes. This appendix describes the released ECNs to date and the status of the controller in terms of whether these ECNs are applied or outstanding.

- “[Fixed USB 3.0 ECNs](#)” on page [544](#)
- “[Fixed USB 2.0 ECNs](#)” on page [547](#)
- “[Fixed xHCI ECNs](#)” on page [555](#)

Table D-1 Fixed USB 3.0 ECNs

Errata Version	Ref/Section/Table/ Page	Errata Brief Description	Affected Layer	Version of RTL Fixed	Comments
Pending_HP_timer	Table 7-7 Section 7.2.4.1.13 Section 7.5.6.1	<p>To address propagation delay of the cable and retimers</p> <ol style="list-style-type: none"> 1. Relax Pending_H-P_Timer from 3us to 10us at link and PHY layers to allow extended propagation delay of the cable and retimer when both host and device are implemented based on this ECR. 2. Add new timing requirement of tDHPResponse on new USB 3.0 and USB 3.1 SSP/SS implementations to constrain HP and LC processing time for maximum legacy compatibility. 	Link	3.20a	Retimer ECN
Ux_LFPS_Exit	Table 6-30 Table 6-29	<ol style="list-style-type: none"> 1. To accommodate for propagated cable delay during U1 exit <ol style="list-style-type: none"> a. Extend (t11-t10) from 900ns (max) to 2us b. Extend (t12-t11) from 900ns (max) to 2us 1. To add additional guard band by extending (t13-t11) min to 900ns for normal Ux LFPS exit 2. To add additional guard band by extending (t12-t11) min to 600ns for simultaneous Ux LFPS exit 	Link	3.20a	Retimer ECN
PM_timer	Table 7-8	Relax PM_LC_TIMER and PM_ENTRY_TIMER	Link	3.20a	Retimer ECN
Q1'09 USB 3.0 Errata (Released 05/15/2009)	D.6.301/Section 6.9.2/Page 6-34	Min U1 exit LFPS duration 600ns	LINK	1.86a	

Errata Version	Ref/Section/Table/ Page	Errata Brief Description	Affected Layer	Version of RTL Fixed	Comments
Q1'09 USB 3.0 Errata (Released 05/15/2009)	D.6.302/Table 6-21/Page 6-33	U1 exit tNoLFPSResponseTimeo ut 2ms	LINK	1.86a	
Q1'09 USB 3.0 Errata (Released 05/15/2009)	D.6.303/Table 6-20/Page 6-35	U1 -> U2 transition is enabled. On remote initiated U1/U2 exit, If local link is in U1, always follow U1 exit timing. If local link is in U2, always follow U2 exit timing.	LINK	1.86a	
Q1'09 USB 3.0 Errata (Released 05/15/2009)			Protocol	N/A	No specific changes were required. Core was following the specification.
Q2'10 USB 3.0 Errata (Released 6/09/2010)			Protocol	N/A	no specific changes were required. Core was following the specification.
Q1'11 USB 3.0 Errata (Released June/2011)			Hub only	1.88a	Mark ITP delayed if more than 32ns delay occurs.
ECN001		LDN command	LINK	0.73a	
ECN002			Cables	N/A	No changes required.
ECN003		Reset propagation	Hub only	1.09a	
ECN004			Connectors	N/A	No changes required.
ECN005			Connectors	N/A	No changes required.
ECN006		Peripheral upstream state m/c	Root Hub	1.20a	

Errata Version	Ref/Section/Table/ Page	Errata Brief Description	Affected Layer	Version of RTL Fixed	Comments
ECN007		efficient ISO/PING	Protocol	not implemented	Targets the host scheduler functionality. This will enhance low power efficiency. Currently if there are more than one ISOC endpoint with each endpoint having different service intervals, device needs to keep the link in U0 for the endpoint having the largest service interval.
ECN008			Connectors	N/A	No changes required.
ECN009			Connectors	N/A	No changes required.
ECN010		DS Port state m/c	Hub only	1.88a	Handles special scenarios for DS PORT of self powered hub to ensure SS devices stay at SS speed.
ECN011		US Port state m/c	Peripheral	Not implemented	Can workaround by enabling GCTL[16].
ECN012		Equalizer change	PHY	N/A	No changes required
LTSSM timings naming errata		Documentation	Specifications	N/A	No changes required
ECN013		USB3.0 Standard A Receptacle	Connectors	N/A	No changes required
ECN014		USB3.0 Standard A Contact Height	Connectors	N/A	No changes required
ECN015		USB3.0 Spread Spectrum Clock Range	PHY	N/A	No changes required
ECN016		USB3.0 Connector Crosstalk Spec	Connectors	N/A	No changes required
ECN017		USB3.0 MicroAB Connector	Connectors	N/A	No changes required
ECN018		USB3.0 Radio Friendly Clock Unit Interval	PHY	N/A	No changes required

Errata Version	Ref/Section/Table/ Page	Errata Brief Description	Affected Layer	Version of RTL Fixed	Comments
ECN019		USB3.0 MicroB Cable Loss Specification	Cables	N/A	No changes required
ECN020	Section 7.5.3.7.1 Section 7.5.3.7.2 Table 7-12	USB 3.0 - Relaxation of tRxDetectQuietTimeout	Link	3.00a	-

Table D-2 Fixed USB 2.0 ECNs

Errata Version	Ref/Section/Table/ Page	Description	Layer Affected	Version of RTL Fixed	Comments	Mode of Checking
Clarification on the Chamfer on USB 2.0 Micro Connectors ECN	This ECN affects the MicroUSB Specification to the USB 2.0 Specification, Revision 1.01.	Modify the USB2.0 micro receptacle definition so that the external chamfer metals are optional as they are in the USB3.0 micro specification. The ECN proposes to add a note "Chamfer metals are optional with no sharp edges." to the Figure 4-9 Micro-AB receptacle interface on the page 24 and to the Figure 4-10 Micro-B receptacle interface on the page 25.	Cables/ Plugs	NA	No Link-Hardw are implication in this ECN	NA
Maximum Unmating Force Value Definition to USB 2.0 Micro Connectors ECN	This ECN affects the MicroUSB Specification to the USB 2.0 Specification, Revision 1.01. There two Changes for Original specification. 1. Section 6.3 Extraction Force 2. Figure 4-8 Micro-A/B Plug Interface	Specify a maximum un-mating force value of 25N for micro series connectors. Also, as a guide to Microseries plug designers, reference dimensions are added for the angle and typical height of the latch feature on the plug. Changes are related to Micro USB Plug.	Cables/ Plugs	NA	No Link-Hardw are implication in this ECN	NA

Errata Version	Ref/Section/Table/ Page	Description	Layer Affected	Version of RTL Fixed	Comments	Mode of Checking
Micro-USB Cables and Connectors ADOPTERS AGREEMENT – Device Class Specification	USB Micro-USB cables and connectors Specification, Revision 1.0a, April 4,2007 There are two ECNS: <ul style="list-style-type: none">■ ECN1.Section 4.5.1 and Section 4.10.1.1.2■ ECN2.Section 4.2 and Figure 4.8	<p>There are two ECNS:</p> <ol style="list-style-type: none"> 1. ECN for Cable R3 2. ECN for Micro-B ID Pin resistance and tolerance stack-up b/n D+ and D- <ul style="list-style-type: none"> ■ ECN#1. Cable R3: A new type of micro-USB cable referred to as quad type micro-USB cable is introduced. The benefits of quad type micro-USB cables is that they have a smaller diameter, more flexible and has good bending performance. They are also economical. ■ ECN#2.Micro-B ID Pin Resistance change: Increases the minimum resistance to ground for the ID pin. The resistance of the ID pin is required for Battery charging applications. 	Cables/ Plugs	NA	No Link-Hardw are implication in this ECN	NA
Micro-USB ECN for Cable R3	USB Micro-USB cables and connectors Specification, Revision 1.01. (Same as that listed as ECN1 under item#7)	<p>Cable R3: A new type of micro-USB cable referred to as quad type micro-USB cable is introduced.</p> <p>The benefits of quad type micro-USB cables is that they have a smaller diameter, more flexible and has good bending performance. They are also economical.</p>	Cables/ Plugs	NA	No Link-Hardw are implication in this ECN	NA

Errata Version	Ref/Section/Table/ Page	Description	Layer Affected	Version of RTL Fixed	Comments	Mode of Checking
MicroUSB ECN	USB Micro-USB cables and connectors Specification, Revision 1.01. (Same as that listed as ECN2 under item#7)	Increase the minimum resistance to ground of the MicroB plug ID pin. Micro-B ID Pin Resistance change: Increases the minimum resistance to ground for the ID pin. The resistance of the ID pin is required for Battery charging applications.	Cables/ Plugs	NA	No Link-Hardw are implication in this ECN	NA
Micro-USB_1_01	USB Micro-USB cables and connectors Specification, Revision 1.0.	New version of USB Micro-USB Cables and connectors released with Editorial corrections and additions to contributor list. Reinserted shell and plug material requirements as section 6.10. Clarified wording on Plating Recommendations.	Cables/ Plugs	NA	No Link-Hardw are implication in this ECN	NA
USB 2.0 Material Change ECN	USB 2.0 Specification Chapter 6.	Change several of the materials in the USB 2.0 specification and allow optional alternatives that are either better materials or equivalent substitutes that ultimately enhance the products. The substitute materials must meet all performance requirements as defined in the USB 2.0 document and Compliance Specification. ECN specifies the change several Materials used for Cables and Connectors.	Cables/ Plugs	NA	No Link-Hardw are implication in this ECN	NA

Errata Version	Ref/Section/Table/ Page	Description	Layer Affected	Version of RTL Fixed	Comments	Mode of Checking
USB 2.0 Connect Timing ECN (April 24, 2009)		ECN Address The Following Issues : dead battery pre-connect current post connect current timing VBUS valid to connect timing Summary impact: Before ECN it was 100nA for 100 ms. After ECN it is 100 mA for 1 sec before connect.	Link	Not Implemented in RTL.	The S/W driver to take care of this.	NA
USB 2.0 Phase-Locked SOFs ECN	USB 2.0 Specification Chapter 7 and Section 7.1.7.7 and section10.2.3.	The SOF tokens sent by a host after resuming a selective suspended port must be still phase-locked with respect to the SOF tokens that were sent before the suspend/resume signaling. Or L1 Entry /Exit Signaling.”	Link/PHY	Supported through PHY	For UTMI /UIPI mode of operations Phy should take care of Sending Phase locked SOF after Coming out L2 or L1. No direct impact on Link RTL.	HW
USB2_LinkPowerManagement(ECN)[final]	LPM features related ECN	Defines a power management feature	Link	1.00a	This ECN was taken care, in the first released version of USB3 controller itself.	CRV, HW
XV-4687C ECN Proposal for Micro USB	USB Micro-USB cables and connectors Specification, Revision 1.01. Changes to Chapter 5.	This ECN specification is for Contact for Connectors.	Cables/Plugs	NA	No Link-Hardware implication in this ECN	NA

Errata Version	Ref/Section/Table/ Page	Description	Layer Affected	Version of RTL Fixed	Comments	Mode of Checking
XV-4687C_Supplement Quad cable R1	USB Micro-USB cables and connectors Specification, Revision 1.01.	Supplement to ECN Quad Cable R1. This Supplement says "Quad type has no impact to current USB2.0 specification and system. So any additional test or specification are not needed.	Cables/ Plugs	NA	No Link-Hardw are implication in this ECN	NA
5V Short Circuit Withstand ECN	This ECN affects the core USB 2.0 specification. ECN discusses the Change in section 7.1.1 USB Driver Characteristic.	"This ECN proposes to remove the 5V short circuit withstand requirement for the USB D+ and/or D- bus traces. A USB transceiver is recommended, but not required, to withstand a continuous short of D+ and/or D- to VBUS for a minimum of 24 hours without degradation.	USB Transceiver/ PHY layer	NA	No Link-Hardw are implication in this ECN	NA
2002_05_28_errata :		This Errata defines multiple clarifications, in the USB 2.0 specifications.	Link / Cable / PHY	1.00a	The Link component is implemented in the controller RTL	CRV, HW
2002_05_28_errata : Modify Cable Attenuation for 5 Meters						
2002_05_28_errata : Clarified Intended Behaviour of NAK'ed OUT						
2002_05_28_errata : Correct use of version 2.0 in bcdUSB						

Errata Version	Ref/Section/Table/ Page	Description	Layer Affected	Version of RTL Fixed	Comments	Mode of Checking
Device Capacitance ECN	USB 2.0 Specification for USB 2.0 Specification. Changes are for Section 7.2.4.2.	This ECN ensures that all USB peripherals have a detectable change in capacitance on VBus when they are connected. This ensures that Embedded Hosts can detect attachment of a peripheral in the absence of VBus. This ECN proposes a change to "A minimum of 1.0uF is recommended for bypass across Vbus." in section 7.2.4.2 of USB 2.0 Specification.	USB Transceiver / Phy layer	NA	No Link-Hardware implication in this ECN	NA
ecn1-usb20-miniB-revD	USB 2.0 Specification Chapter 6.	The ECN specifies the mechanical requirements for the mini-B plug receptacle and cable assembly. The ECN is in the form of a new Chapter 6 with the mini-B connector requirements inserted into the appropriate locations.	Cables and Plugs	NA	No Link-Hardware implication in this ECN	NA
errata-092800-1207001		Fix for typographical errors in "USB revision 2.0 April 27, 2000"	Link	NA	No Link-Hardware implication in this ECN	NA
Inter-Chip USB Supplement to the USB 2.0 Specification_1.0		Defines a new interface to support LS/FS speeds	Link	NA	ICUSB is not yet supported in the controller.	NA
InterfaceAssociationDescriptor_ecn		"This ECN defines a new standard descriptor and interface numbering rules that allow a device to describe which interfaces are associated with the same device function. This allows the operating system to bind all of the appropriate interfaces to the same driver instance."	Link	NA	No Link-Hardware implication in this ECN	NA

Errata Version	Ref/Section/Table/ Page	Description	Layer Affected	Version of RTL Fixed	Comments	Mode of Checking
resistor_ecn	USB 2.0 Specification Chapter 7. No changes Outside Chapter 7 of USB 2.0 Specification.	This ECN changes the range of the pull-up and pull-down resistors used to bias the bus. This ECN shows FULL Speed and Low speed Device resistor Connections.	USB Controller & USB Transceiver / Phy layer	NA	This ECN has implication for FS1.1 interface; This interface is not supported by controller and hence the ECN does not have impact on the controller	NA
rnd_chmfr_ECNFINAL	USB 2.0 Specification ECN #1: Mini-B Connector Date:10/20/2000	New mini-B plugs are recommended to have rounded chamfer design because this results in longer lasting mini-B receptacle (for longer durability). The edges of the mini-B plugs need to be rounded (rather than pointed) so that there is less abrasion while plugging in and out of the mini-B receptacle.	Only Mini-B Plug	NA	No Link-Hardware implication in this ECN	NA

Errata Version	Ref/Section/Table/ Page	Description	Layer Affected	Version of RTL Fixed	Comments	Mode of Checking
Suspend Current ECN	Section 7.2.3 of USB 2.0 standard, April 2000	This errata now establishes no difference b/n high-power and low-power devices. Prior to the errata, only high-power devices were allowed to draw 2.5mA during suspend and low-power devices allowed to draw 500uA during suspend. With this ECN, the maximum suspend current is raised to 2.5mA for all devices except ICUSB devices. ICUSB device suspend current remains at 150 uA.	Battery Charging Hardware, PHY, Host Power-Mgmt Software	NA	No Link-Hardware implication in this ECN	NA
UnicodeECN	Section 9.6.7 of USB 2.0 standard, April 2000	String descriptors must now use UNICODE UTF-16LE encodings. String Descriptrors are referenced by USB device descriptors to provide displayable information describing a descriptor in human-readable form. The inclusion of string descriptors is optional. Currently, there are known cases of devices using an encoding other than UTF-16LE for the string descriptors.	Software / Driver	NA	No Link-Hardware implication in this ECN	NA
USB2_LinkPowerManagement(ECN[new, 9/29/2011])	LPM features enhancement ECN	Defines deeper power states and options to negotiate LPM based on current power state	Link	2.40a	Enabled by selecting LPM Errata configurable option	

Table D-3 Fixed xHCI ECNs

Errata Version	Errata Brief Description	Version of Implementation
Errata 01	Miscellaneous changes to multiple chapters	1.90a
Errata 02	Multiple changes to Save and Restore	1.90a
Errata 03	Multiple changes to USB2 LPM	1.90a
Errata 04	Miscellaneous changes to multiple chapters	1.90a
Errata 05	Miscellaneous changes to multiple chapters	No changes required
Errata 06	Miscellaneous changes to multiple chapters	2.10a
Errata 07	Miscellaneous changes to multiple chapters	2.10a
Errata 08	LPM update for the USB2 LPM BESL Errata	2.40a
Errata 09	Miscellaneous changes to multiple chapters	2.60a
Errata 10	Add a new Stopped-Short Packet Completion Code	2.60a
Errata 12	Miscellaneous changes to multiple chapters	2.60a
Errata 13	Add EDTLA reporting in Stream Contexts	2.60a
Errata 14	Allow Ring Overruns to be generated if LPF is not set, Require that Missed Service Errors reference TRBs	Not implemented. Error case handling, ease of software handling missed service error.
Errata 15	Clarify Interrupter to MSI-X vector mapping	No changes required. Clarifications to the specification.
Errata 16	Add Contiguous Frame ID Capability	Not implemented. Flexibility/recovery of ISOC transfer
Errata 17	FLADJ capability, System bus reset capability with DbC mode	Not implemented. Flexibility in implementation
Errata 18	Clarify BESL LPM Usage, Clarify Coarse vs. Fine Grain Scatter/Gather, Clarify ownership of DbC Transfer Rings	No changes required, clarifications to different sections of the specification.

Errata Version	Errata Brief Description	Version of Implementation
Errata 19	BESL LPM Capability	2.50a
	If software sets the PORTSC PLS field to U3 when the link is in L1, automatic transition from the L1 state through the U0 state to the L2 state	2.60a
	U3 Entry Capability	2.90a
	Enable flag for Maximum Exit Latency Too Large Errors with Config EP Commands	2.90a
	Make CFC support mandatory	3.00a
	Add U3 Entry Capability and make support mandatory in 1.1	3.00a
	Make valid TRB Pointer fields in Overrun/Underrun TRBs required for 1.1	3.00a
	Clarify Missed Service Error handling due to Overrun/Underrun conditions	3.00a
	Clarify PAE and Short Packet IOC handling	3.00a
	Recommend Split Transaction completion	3.00a
	Force Save Context Capability	3.00a
Errata 20	Support software compliance mode enable	2.90a
	Extended Configuration Information (CIC/CIE)	3.00a
Errata 21	Allow xHC transition to D3 if a port is in the Error state	2.90a
	Change CAS Assertion conditions to support more flexible power management	2.90a
	Drop the SPE Flag	2.90a
	SuperSpeed Terminations on overcurrent, etc.	Not supported
	Microframe Targeting	3.00a
	Clarify relationship of Isoch Buffer Overrun to TD Babble	3.00a
	Clarify Ring Overrun/Underrun	3.00a
	Clarify errors on late TD scheduling	3.00a
	Allow Ring Overruns to be generated if LPF is not set	3.00a
	SIA=1 Compatibility	3.00a

E

External Buffer Control

This appendix describes the device External Buffer Control (EBC) interface of the SuperSpeed USB 3.0 Controller controller.

The topics are as follows:

- “[Overview of EBC](#)” on page [558](#)
- “[External Buffer Control for OUT Endpoint](#)” on page [558](#)
- “[External Buffer Control for IN Endpoint](#)” on page [561](#)
- “[External Buffer Control for Trace Applications](#)” on page [566](#)

E.1 Overview of EBC

Device hardware External Buffer Control (EBC) is a minimized hardware-level control interface that allows the application to throttle the DMA read/write of data without involving software intervention. EBC is for low-level debugging interfaces where the debugging endpoint functionality can be restricted to meet the requirements of EBC. It is not for general USB use. Software must setup the TRB structures and initiate the transfers, but once running the hardware handles the transfers.

EBC minimizes traffic to the system memory on the AXI/AHB bus using a “buffer available” and “packet received/acknowledged” sideband interface. The bus transfers occur in maximum-packet size lengths, and are initiated by the DWC_usb3 controller only when the application indicates that there is either a packet-sized space available to be written to, or a packet-sized amount of data is available to be read.

The application must implement an external buffer to utilize the EBC feature. The external buffer recognizes all data transfers associated with the EBC endpoint using the data address and generates the appropriate EBC handshake signals. Refer to the “[System Clock, Reset, and Control Signals](#)” on page [299](#).

To enable EBC, set the coreConsultant parameter `DWC_USB3_EXT_BUFF_CONTROL` to ‘1’. For each EBC endpoint, software must program bit 15 of the DEPCFG Parameter 1 to ‘1’. For more details on DEPCFG, refer to the “Device Physical Endpoint-Specific Command Structure” section in the *DWC SuperSpeed USB 3.0 Controller Programming Guide*.

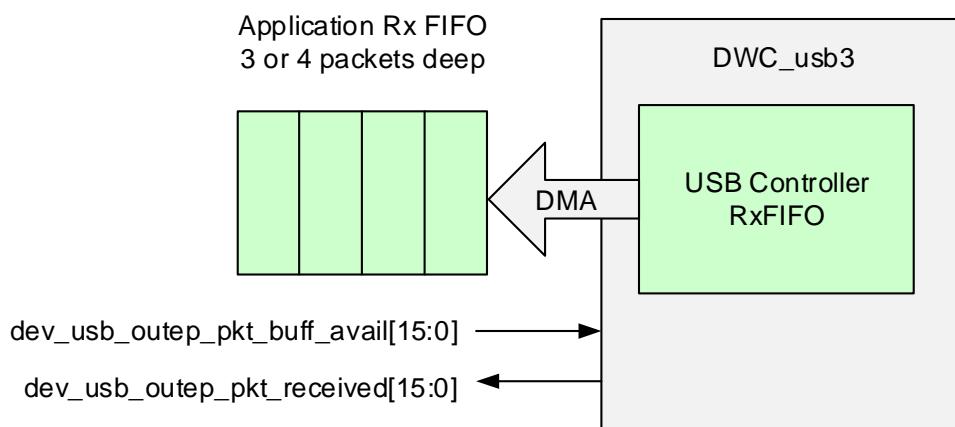
The following requirements must be met to enable EBC:

- The mode of operation is device.
- The endpoint is a Bulk endpoint.
- The endpoint is a SuperSpeed or high-speed endpoint.
- The length of each TRB is aligned. It is also an even multiple of the Maximum Packet Size of the endpoint which is 1024k for SuperSpeed and 512k for high-speed.
- The Maximum Packet Size of the endpoint is programmed to be the USB Bulk Maximum Packet Size.
- No USB short packets or zero-length packets used.
- No streams used.

For efficient operation of EBC, it is recommended that the application uses maximum-sized TRB buffer space (or, the largest possible buffer space).

E.1.1 External Buffer Control for OUT Endpoint

The application uses EBC for communicating to the DWC_usb3 controller that there is sufficient space available for an OUT data packet. This prevents initiation of a DMA write until sufficient space is available in the Application RxFIFO. This happens without software intervention. The Application RxFIFO is 3 or more packets deep.

Figure E-1 External Buffer Control for OUT Endpoints

When an OUT transfer is initiated by the host on the USB, the device DWC_usb3 controller checks the `dev_usb_outep_pkt_buff_avail` input. If this input is asserted (and all the other normal checks pass), it assumes that the application can accept at least two packets of maximum packet size. The controller accepts the packet and sends an ACK TP back to the host. It alerts the application that a good packet is received and intends to perform a DMA write by pulsing the `dev_usb_outep_pkt_received` output. The DMA write to the application RxFIFO is queued to be executed.

When the application RxFIFO has space for less than two maximum-sized packets only, it de-asserts the `dev_usb_outep_pkt_buff_avail` signal. The application must account for both the amount of data present in its RxFIFO and the early indication from the controller that an OUT packet has been received and a DMA write will take place. This allows the application to update the buffer available signal before actual DMA write happens.

If the `dev_usb_outep_pkt_buff_avail` input is de-asserted, the DWC_usb3 controller assumes that no more packets can be received. On the USB, the controller responds with a NRDY/NAK to subsequent packets from the host. Because the check is done at the beginning of a packet, the controller may receive one additional packet when the available buffer de-asserts.

In this manner, the application can give an early indication of the available space, which prevents the system bus from being tied-up due to insufficient buffer space.

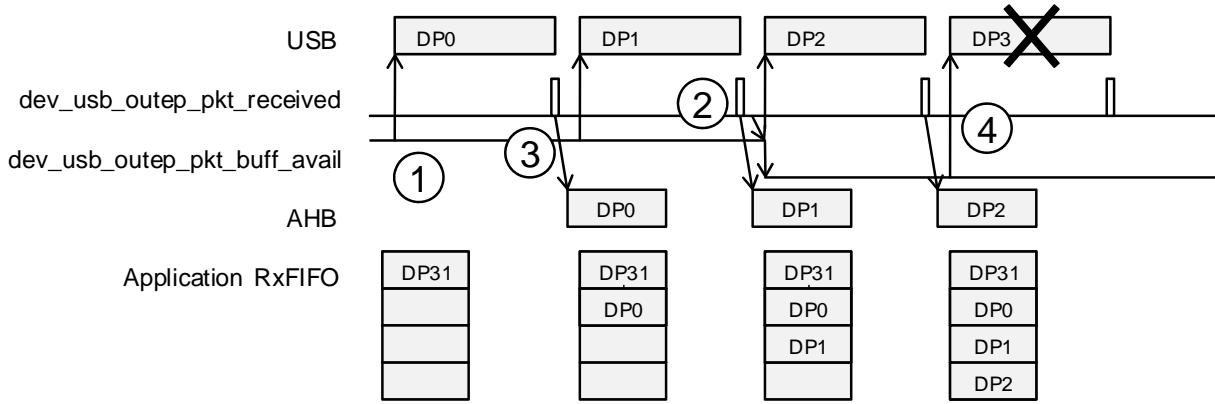


Note When EBC-enabled endpoints encounter USB short packets or zero-length packets, they result in undefined behavior.

E.1.1.1 Waveform Examples

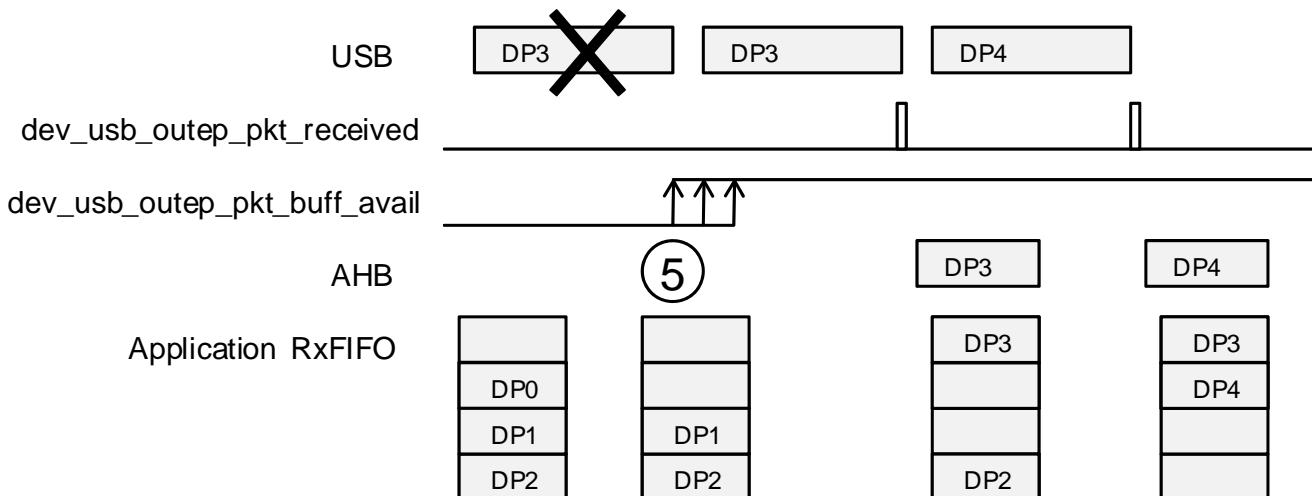
Figure E-2 and Figure E-3 help in better understanding of EBC for OUT Endpoints.

Figure E-2 EBC for OUT Endpoints – Example 1



1. When a packet arrives on the USB, the `dev_usb_outep_pkt_buff_avail` is checked. If the `dev_usb_outep_pkt_buff_avail` is 1, the DWC_usb3 controller accepts the packet.
2. When the accepted packet is confirmed to be error free, the DWC_usb3 controller pulses the `dev_usb_outep_pkt_received` signal. The application RxFIFO reserves a slot for this packet, reducing the available space. After this calculation, if the available space is less than 2, the `dev_usb_outep_pkt_buff_avail` is de-asserted.
3. When a packet is accepted, the DWC_usb3 controller performs a DMA write.
4. Packets received on the USB when the `dev_usb_outep_pkt_buff_avail` is 0 are rejected at the USB.

Figure E-3 EBC for OUT Endpoints – Example 2



5. The `dev_usb_outep_pkt_buff_avail` signal can be asserted before, during, or after evaluation of the incoming packet.

E.1.1.2 Setting the Device Endpoint Configuration Parameters Burst Size and EBC

The EBC functionality adds an additional throttling point to the DWC_usb3 controller, which you need to consider when programming the Device Endpoint Configuration Parameter (see DEPCFG in the *DWC SuperSpeed USB 3.0 Controller Programming Guide*). The DEPCFG.Parameter0.BrstSiz indicates the burst size supported by the endpoint.

The following two register fields in the DWC_usb3 controller determine the value of NumP that is used in the ACK TP.

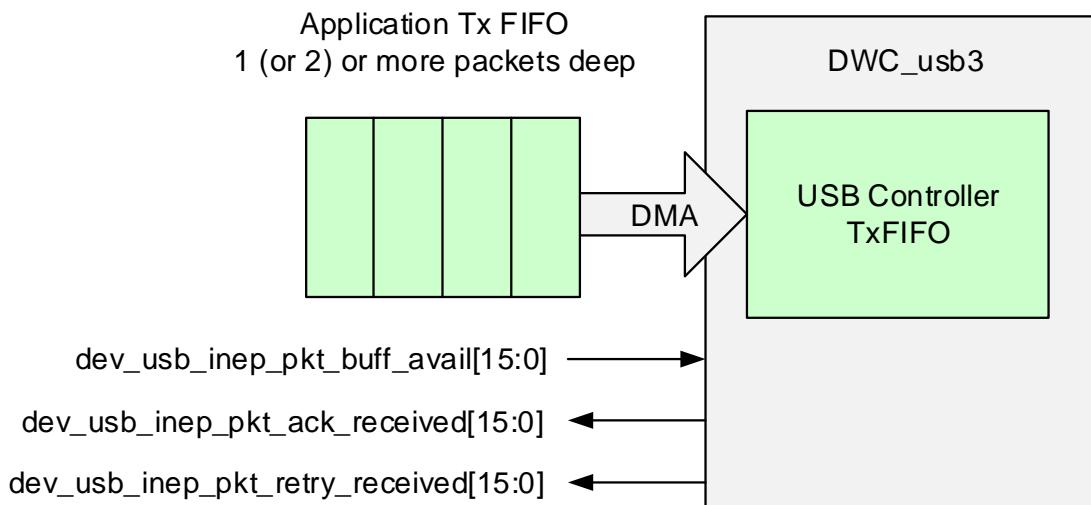
- DCFG.NumP – globally indicates the number of receive buffers available
- DEPCFG.Parameter0.BurstSiz – endpoint-specific parameter

The controller uses the minimum of these two values in the ACK TP. These registers may be used to further optimize throughput for each specific application.

E.1.2 External Buffer Control for IN Endpoint

The application uses EBC for communicating to the DWC_usb3 controller that there is at least one IN data packet in the external buffer. This prevents initiation of a DMA read until sufficient data is available in the application TxFIFO. This happens without software intervention. The application TxFIFO is one, two or more packets in depth. The application must indicate when it has the minimum number of packets necessary. This is configuration dependent because the DWC_usb3 controller can be configured to generate one outstanding DMA packet read requests.

Figure E-4 External Buffer Control for IN Endpoints



When the DWC_usb3 controller processes an IN TRB, the controller will perform DMA reads to preload the IN data into the controller RAM. In this case, the read data comes from the application TxFIFO. The application TxFIFO will cache the data packets, and when at least one packet is available, the application asserts the dev_usb_inep_pkt_buff_avail signal. Until this signal is asserted, the DWC_usb3 controller does not start DMA read for data packets. This prevents the DWC_usb3 controller from initiating DMA transfers on the system bus and occupying the system bus when high latency data is not immediately available. Once asserted, the DMA read proceeds.

The application should de-assert the `dev_usb_inep_pkt_buff_avail` signal when the application TxFIFO contains less than one packet of unread data (that is, after the first pop from the application TxFIFO and the amount of data in the TxFIFO is one maximum packet minus one TxFIFO data word). The DWC_usb3 controller performs the evaluation to initiate the next DMA transfer after the previous DMA transfer completes. De-asserting the buffer available signal as early as possible will ensure that the next DMA transfer will not be initiated when insufficient data is present in the application TxFIFO. Note that unacknowledged data should not count towards the amount of data available to be read.

The application TxFIFO must retain all the data packets until the packet is acknowledged as indicated by the DWC_usb3 controller. When an acknowledgment is signaled, the application TxFIFO can safely discard the corresponding packet, freeing up space for pushing any subsequent packets. In this manner, the application TxFIFO will cache the current packets of data until there is a confirmation that the packet has been transferred successfully on the USB. Packets will always be acknowledged in order. When the DWC_usb3 controller signals an acknowledgment with a `dev_usb_inep_pkt_ack_received` pulse, the application TxFIFO can advance the ACK pointer to the beginning of the next packet, effectively freeing up space in the TxFIFO for new packets.

If a packet transfer on the USB is unsuccessful, the host will request a retry of the packet from the device. To fulfill this, the DWC_usb3 controller signals a “retry received” (`dev_usb_inep_pkt_retry_received`), and then performs a repeat DMA read of the packet. Since the packet being retried will be the first unacknowledged packet in the application TxFIFO, the application TxFIFO must set the read pointer to the value of the ACK pointer. This “rewinding” of the pointer allows the DMA read to access the data packet being retried on the USB. When a retry condition is detected, the DWC_usb3 controller completes any active DMAs belonging to the corresponding endpoint that received the retry before signaling the retry pulse. For any single endpoint, ACK and Retry received pulses never overlap.

Note that for configurations where the `DWC_USB3_NUM_OUTSTANDING_TXDMA` is 2, the endpoints that utilize EBC must operate in `DWC_USB3_NUM_OUTSTANDING_TXDMA = 1` mode. This is specified by bit 15 of the DEPCFG (for more details, refer to the “Command 1: Set Endpoint Configuration Parameters” section in the *DWC SuperSpeed USB 3.0 Controller Programming Guide*). This eliminates the problem with deadlock when only one packet is outstanding, while allowing non-EBC endpoints to function with a higher performance using 2-outstanding mode.



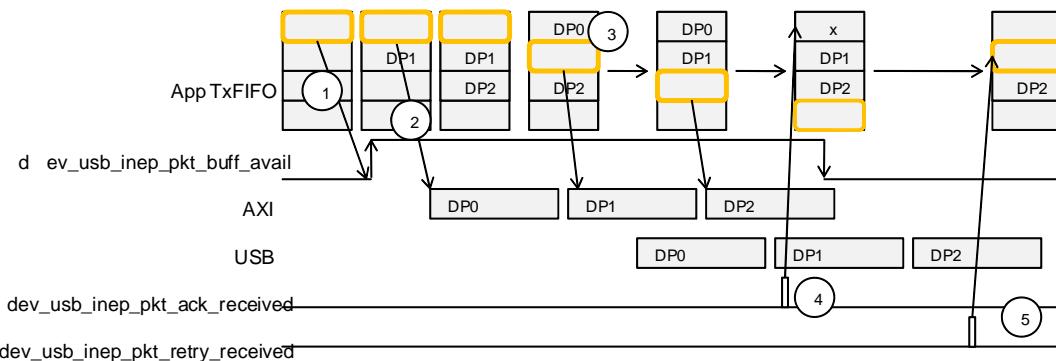
- Multiple IN transfers must not be setup using the Chain (CHN) bit because it results in undefined behavior.
- Short packet and zero-length transfers must not be utilized for EBC-enabled endpoints because it results in undefined behavior.

The following two examples help in better understanding of EBC for IN Endpoints.

E.1.2.1 EBC for IN Endpoints – Example 1

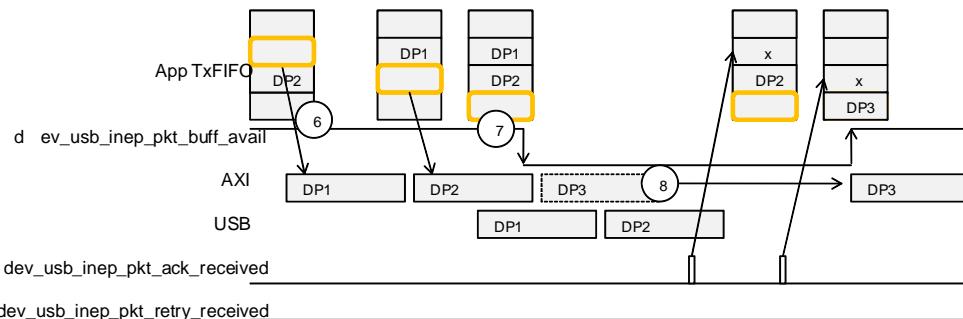
Figure E-2 provides an example of EBC for IN endpoints.

Figure E-5 EBC for IN Endpoints – Example 1



1. The application pushes IN data packets to the TxFIFO; the minimum available data packet is met, and hence the `dev_usb_inep_pkt_buff_avail` signal goes from 0 to 1.
2. Since `dev_usb_inep_pkt_buff_avail` is 1, the controller does a DMA read, and transmits it on the USB when the host requests for data. The `dev_usb_inep_pkt_buff_avail` signal should de-assert after the first data is popped from the TxFIFO, as opposed to after the entire packet has been read out. Since the DWC_usb3 controller evaluates this signal after the current DMA read completes, de-asserting this signal as soon as possible will prevent the next DMA read from getting initiated due to late signaling by the external TxFIFO.
3. The read pointer advances in the buffer, and retains the data packet until acknowledged.
4. The host sends an ACK; the controller indicates that an ACK is received using the `dev_usb_inep_pkt_ack_received` signal, and the application de-allocates the associated packet.
5. The host sends a RETRY; the controller flushes, waits for the AXI to complete, and then signals that a retry is received using `dev_usb_inep_pkt_retry_received` signal; the application reverts the read pointer to the beginning of the first unacknowledged packet.

Figure E-6 EBC for IN Endpoints – Example 1 (Continued)



6. The controller reads the rewind packet again, and continues transfer from there.

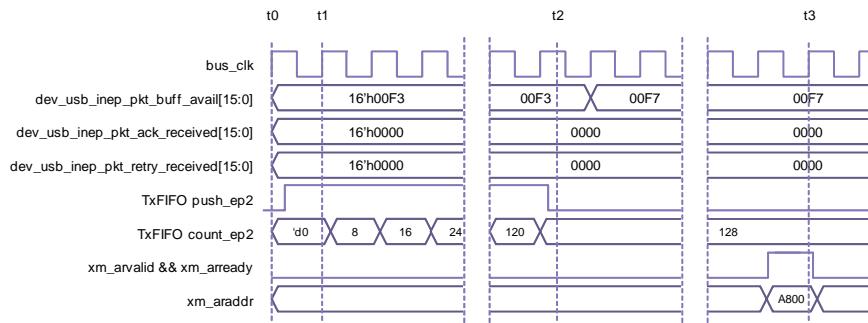
7. The last packet in the application TxFIFO is read and no more packets remain. Hence the dev_usb_inep_pkt_buff_avail signal goes from 1 to 0.
Note: 1->0 should transition as soon as the first WORD of the packet is read out of the application TxFIFO.
8. The controller needs the DP3 packet, but it is unavailable. Hence it does not initiate the DMA until the dev_usb_inep_pkt_buff_avail signal becomes 1.

E.1.2.2 EBC for IN Endpoints – Example 2

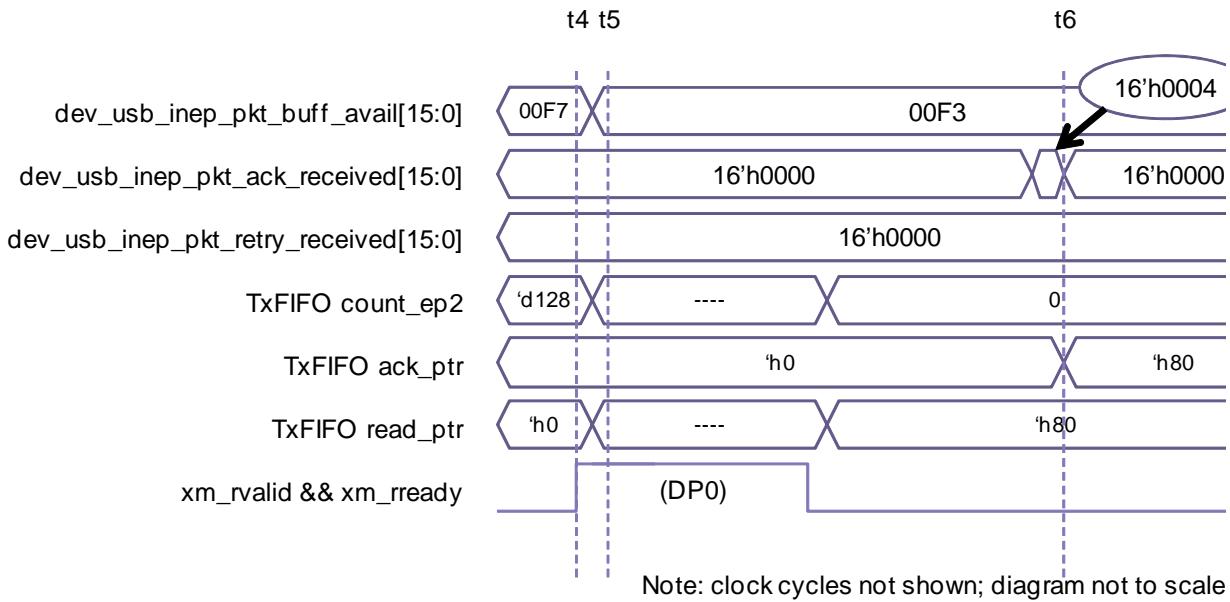
In the following example,

- The bus data width is 64 (i.e., each TxFIFO word is 8 bytes).
- Endpoints 0, 1, 4, 5, 6, 7 are not EBC endpoints. Hence bits [7:4] and [1:0] of the EBC output dev_usb_inep_pkt_buff_avail are tied to 1.
- Endpoints 2 and 3 are EBC endpoints, and each is connected to its corresponding EBC application TxFIFO.
- Endpoints 8 to 15 are unused. So the EBC output dev_usb_inep_pkt_buff_avail [15:8] is tied to 0 (Note that alternatively, they can be tied to 1).

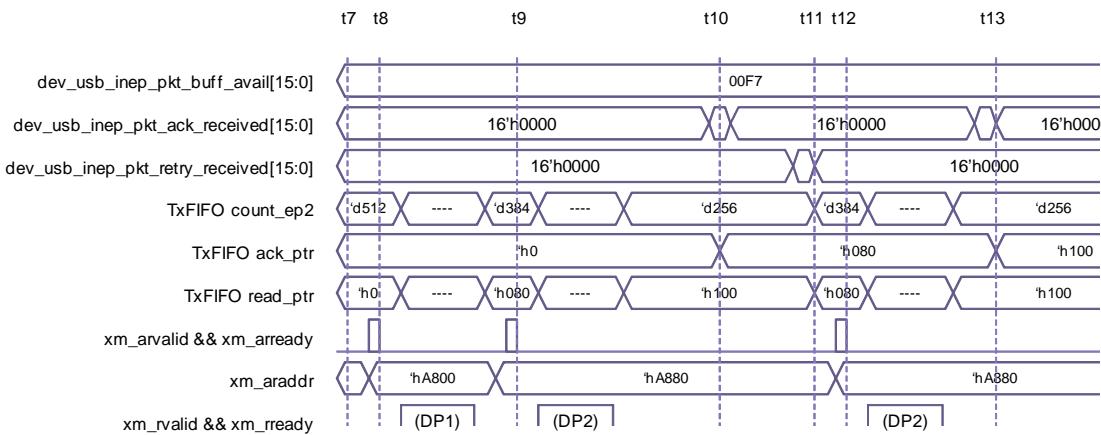
Figure E-7 EBC for IN Endpoints – Example 2



- t0: The Application TxFIFO for EP2 is empty and does not contain even a maximum packet size amount of IN data. For this reason, the dev_usb_inep_pkt_buff_avail[2] signal for EP2 is driven to 0 which prevents the USB controller from initiating a DMA read for this endpoint.
- t1: The application begins to push IN data into the TxFIFO and the byte count of the TxFIFO increments.
- t2: The amount of data in TxFIFO reaches maximum packet size (1024 bytes), and a clock later dev_usb_inep_pkt_buff_avail[2] transitions to 1.
- t3: The USB controller sees that dev_usb_inep_pkt_buff_avail[2] is 1, and initiates the DMA read for EP2.

Figure E-8 EBC for IN Endpoints – Example 2 (Continued)

- t4: The USB controller accepts the read data for the IN data packet 0.
- t5: When the count of the TxFIFO drops below a maximum packet size of data, the `dev_usb_inep_pkt_buff_avail[2]` de-asserts. This does not interrupt the current DMA which is in progress.
- t6: On the USB, the IN data packet is transferred and the host acknowledges the packet. The packet has been successfully transferred on the USB. So the USB controller pulses the `dev_usb_inep_pkt_ack_received[2]`. The TxFIFO advances the `ack_pointer` by a maximum packet size.

Figure E-9 EBC for IN Endpoints – Example 2 (Continued)

- t7: At this time, it is assumed that the application has already pushed four maximum packet-sized packets into the TxFIFO for EP2 (TxFIFO count shows 512 bytes). Since the TxFIFO contains more than one packet (maximum packet-sized), `dev_usb_inep_pkt_buff_avail[2]` is 1.
- t8: The USB controller performs a DMA read for DP1, the TxFIFO count decrements, and the TxFIFO read pointer advances. However, the TxFIFO ack pointer does not advance.

- t9: The USB controller performs a DMA read for DP2.
- t10: On the USB, DP1 is acknowledged and `dev_usb_inep_pkt_ack_received[2]` is pulsed. This causes the TxFIFO `ack_ptr` to advance one maximum packet size.
- t11: On the USB, DP2 is retried and `dev_usb_inep_pkt_retry_received[2]` is pulsed. This causes the TxFIFO `read_ptr` to rewind to the `ack_ptr` value so that DP2 may be read again.
- t12: The USB controller repeats the DMA read for DP2.
- t13: On the USB, DP2 is acknowledged and `dev_usb_inep_pkt_ack_received[2]` is pulsed.

E.1.3 External Buffer Control for Trace Applications

When the controller is used for trace applications in EBC mode of operation, the application prepares a set of TRBs and programs the controller for normal operations. After the TRBs are created, the application may use the same set of TRBs over and over again. The controller keeps writing back the TRBs after it has consumed them. If the buffer size is small, then the controller keeps closing the TRB descriptor(s) by making `HWO=0`, and the application has to keep validating the TRB by making `HWO=1`. This adds to application overhead and can make the trace application slower. In order to prevent the controller from writing back the TRB descriptor, set field [14] of “Command 1: Set Endpoint Configuration Parameters” (for details on the command, refer to the *DWC SuperSpeed USB 3.0 Controller Programming Guide*) to 1.

F

DWC_usb3 Automotive Safety

DWC_usb3 is Certified ISO 26262 ASIL B ready. This appendix describes the automotive safety features supported by the USB 3.0 Controller when you have the corresponding automotive license. The following topics are discussed:

- “[Overview of Automotive Safety Feature](#)” on page [568](#)
- “[Automotive Safety Package Documents](#)” on page [569](#)
- “[Configuring Automotive Safety Feature](#)” on page [569](#)
- “[Programming Automotive Safety Feature](#)” on page [570](#)

F.1 Overview of Automotive Safety Feature

To make an IP better suited for applications in the automotive space, functional safety needs to be considered throughout the IP development process. ISO 26262 addresses functional safety and requires the hardware components to be analyzed with respect to their ability to detect and/or control the effects of random hardware faults. Random hardware faults can occur at any point in time during the life-cycle of the hardware component. They can take the form of permanent faults, or transient faults, and require safety mechanisms to be in place to ensure that severe consequences do not occur as a result of those faults.

The DWC_usb3 includes a broad range of internal safety mechanisms which assist in handling random hardware faults. These complement the mechanisms already defined in the USB 3.0 protocol.

ASIL B Certification

The DWC_usb3 has been analyzed in the scope of ISO 26262 targeting ASIL B for a particular reference configuration.

Reference Configuration

A reference configuration containing the major features such as USB 3.0 DRD mode, ECC, Hibernation, and HSIC (a typical automotive configuration) has been used for ASIL B analysis.

The reference configuration internal safety mechanisms monitor the IP logic and report any detected errors to the application. The application should take an action to handle the error conditions and ensure safe operation of the hardware component.

FMEDA

A Failure Mode, Effects, and Diagnostic Analysis (FMEDA) has also been performed for this reference configuration. The analysis relies on a per-transistor failure rate derived in accordance to IEC TR 62380. This, in conjunction with area information from design synthesis trials, allows to determine the failure rate for the IP which can then be distributed between the modules of the design according to their relative area.

The list of failure modes and their effects is derived for each module and each of the failure modes is assigned a portion of the failure rate of the module they belong to. If the failure mode has the potential to violate a top-level safety requirement (error at a top-level output pin) there is a search for a safety mechanism that can prevent that violation, totally or partially. If a safety mechanism exists, for example, ECC protection for RAMs, a diagnostic coverage (DC) value is assigned representing the percentage of the failure rate associated with the specific failure mode that is prevented from violating a top-level safety requirement. The process is repeated for all failure modes and finally results are accumulated and IP safety metrics are collected in the FMEDA.

F.2 Automotive Safety Package Documents

Table F-1 describes the documents included in the DWC_usb3 safety package. After you have configured the controller in coreConsultant, you can access the automotive documents in your workspace directory at workspace/doc/automotive.



Note You must have a valid license to have access to the automotive safety documents and features in DWC_usb3. For more information on required licensing, refer to the *DesignWare Cores USB 3.0 Controller Installation Guide*.

Table F-1 Description of Documents in Automotive Safety Package

File Name	Description	Contents
DWC_usb3_safety_manual.pdf	Safety Manual	Lists all Synopsys quality process documents; explains IP design flow and tools, verification flows (high-level), and code coverage concepts; describes FMEDA worksheet
DWC_usb3_FMEDA.pdf	FMEDA Worksheet	Calculates FIT rates for each module; enumerates diagnostic capabilities for failure modes
DWC_usb3_certificate.pdf	Automotive Certificate for DWC_usb3 from SGS	N/A
DWC_usb3_automotive_report.pdf	Automotive Certification report for DWC_usb3 from SGS	N/A

F.3 Configuring Automotive Safety Feature

The DWC_usb3 controller includes an Error Checking and Correcting configuration parameter (DWC_USB3_EN_ECC) that when enabled generates ECC, corrects 1-bit errors, and reports multi-bit errors to the software. If you do not have a valid license, this configuration parameter is disabled (grayed out) in coreConsultant. For more information about the ECC parameter, refer to Chapter 4, “Parameter Descriptions” on page 189.

F.4 Programming Automotive Safety Feature

When the DWC_USB3_EN_ECC parameter is enabled, it allows you to program ECC registers. For more information about the ECC registers, refer to the “Register Descriptions” chapter in the *DWC SuperSpeed USB 3.0 Controller Programming Guide*.

F.4.1 ECC Operational Model and Programming Flow

ECC errors are classified as single-bit errors and multiple bit errors. Only Multiple bit errors result in an event to the system because they are not correctable. The controller can be configured to interface up to a maximum of three RAMs; hence, the status reporting is RAM specific.

F.4.1.1 Single-Bit Error

The DWC_usb3 controller automatically detects and corrects single-bit errors when they occur. No interrupt is generated.

When a single bit error occurs,

1. The controller increments the corresponding BRSERRCNT.ram<0,1,2>serrcnt register field.
2. The controller sets the BRMECCERR.ramserr bit along with corresponding BRMECCERR.ramserrvec value. (For example, if a single-bit error occurs on RAM1, the BRMECCERR.ramserrvec reads 3'010.)
3. The software can choose to read the contents of the previously mentioned registers at its own pace and to clear them by writing BRERRCTRLrserrclr=1.



Note Any subsequent single bit errors occurring prior to software read is overwritten with newer ones. Hence software is informed only about the latest status.

F.4.1.2 Multiple Bit Error

The controller generates an interrupt when an uncorrectable ECC error occurs. The event generation is specific to Host and Device modes of operation.

When a multiple bit occurs,

1. The controller increments the value corresponding BRMERRCNT.ram<0,1,2>merrcnt register field.
2. The controller sets the BRMECCERR.rammerr bit along with corresponding BRMECCERR.rammerrvec value. (For example, if multiple bit error occurs on RAM2, the BRMECCERR.rammerrvec reads 3'100.)
3. The controller records the address of the corresponding RAM location on which an ECC error occurs in BRAM<0,1,2>ADDRERR.
4. The software can read the contents of the previously mentioned registers when an event is generated and clear them by writing BRERRCTRLrserrclr=1. Additionally, depending on Host or Device mode of operation, the corresponding flow is applicable.
 - a. Host Mode - The controller generates an interrupt due to USBSTS.HCE and stops all activity. The software response is described as per “[Host Mode Handling](#)”.
 - b. Device Mode - See “[Device Mode Handling](#)” on page [571](#).

Host Mode Handling

According to the xHCI specification, Section 4.24.1,

"The Host Controller Error (HCE) flag is asserted when an internal xHC error is detected that exclusively affects the xHC. When the HCE flag is set to '1' the xHC shall cease all activity. Software response to the assertion of HCE is to reset the xHC (HCRST = '1') and reinitialize it.

Software should implement an algorithm for checking the HCE flag if the xHC is not responding.

Note: HCE may be asserted due to a soft or hard error. An SRAM parity error while accessing an internal data structure is an example of a soft error that may assert HCE. However a hard error shall cause the xHC to reassert HCE immediately after it is reinitialized. In this case, software should employ some heuristics to prevent the case where the xHC is continually in an error-reset-reinitialize loop and report this condition to the user."

Device Mode Handling

An ECC error can occur while the controller is reading the data from the RAMs internally. In such a case, the controller generates an event to the software by indicating a value of 16 in the DEVT register.

To enable the ECC error generation, software should program DEVTEEN.ECCERREN=1.

In response to an ECC error indication, the software driver should issue a soft reset by programming DCTL.Run/Stop=0 and issue a soft reset (DCTL.CSFTRST) before re-initializing the controller.

Internal Parameter Descriptions

Provides a description of the internal parameters that might be indirectly referenced in expressions in the Signals, Parameters, or Registers chapters. These parameters are not visible in the coreConsultant GUI and most of them are derived automatically from visible parameters. **You must not set any of these parameters directly.**

Some expressions might refer to TCL functions or procedures (sometimes identified as **function_of**) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the core in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

Table G-1 Internal Parameters

Parameter Name	Equals To
D0	0
D3	3
DWC_USB3_ADDR_CACHE_BCU	((DWC_USB3_ADDR_PSQ + ((DWC_USB3_MODE ==1 && DWC_USB3_EN_DBC ==0) ? 0: DWC_USB3_PSQ_FIFO_DEPTH)) << DWC_USB3_MADDR_LO)
DWC_USB3_ADDR_CACHE_DEVICE	(DWC_USB3_ADDR_CACHE_EP + DWC_USB3_CACHE_EP_DEPTH)
DWC_USB3_ADDR_CACHE_EP	(DWC_USB3_ADDR_RCSR_DEVICE + DWC_USB3_RCSR_DEVICE_DEPTH)
DWC_USB3_ADDR_CACHE_STREAMS	(DWC_USB3_ADDR_CACHE_DEVICE + DWC_USB3_CACHE_DEVICE_DEPTH)
DWC_USB3_ADDR_DFQ	(DWC_USB3_ADDR_RIQ + DWC_USB3_RIQ_FIFO_ALL_DEPTH)
DWC_USB3_ADDR_DWQ	(DWC_USB3_ADDR_DFQ + DWC_USB3_DFQ_FIFO_ALL_DEPTH)

Parameter Name	Equals To
DWC_USB3_ADDR_EVQ	(DWC_USB3_ADDR_DWQ + DWC_USB3_DWQ_FIFO_ALL_DEPTH)
DWC_USB3_ADDR_PSQ	(DWC_USB3_ADDR_EVQ + DWC_USB3_EVQ_FIFO_ALL_DEPTH)
DWC_USB3_ADDR_RCSR_DEVICE	(DWC_USB3_ADDR_RCSR_GLOBAL + DWC_USB3_RCSR_GLOBAL_DEPTH)
DWC_USB3_ADDR_RCSR_GLOBAL	DWC_USB3_ALIGN_MDW(DWC_USB3_ADDR_CACHE_BCU + DWC_USB3_CACHE_BCU_DEPTH)
DWC_USB3_ADDR_RCSR_HOST	(DWC_USB3_ADDR_RCSR_GLOBAL + DWC_USB3_RCSR_GLOBAL_DEPTH)
DWC_USB3_ADDR_RIQ	(DWC_USB3_ADDR_RXQ + DWC_USB3_RXQ_FIFO_ALL_DEPTH)
DWC_USB3_ADDR_RXQ	(DWC_USB3_ADDR_TXQ + DWC_USB3_TXQ_FIFO_ALL_DEPTH)
DWC_USB3_ADDR_TXQ	0
DWC_USB3_ALL_RXSNK_MBPS_MAX	(DWC_USB3_EN_USB2_ONLY ==1 ? 480 * DWC_USB3_MAX_ACTIVE_U2_INSTANCES: ((8 * DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC < DWC_USB3_MAX_ACTIVE_U2_INSTANCES) ? 4000 * 2 * DWC_USB3_MAX_ACTIVE_U2_INSTANCES / 8 : 4000 * DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC * (DWC_USB3_RX_SS_PRI_CNT + 1) / DWC_USB3_RX_SS_PRI_CNT))
DWC_USB3_ALL_TXSRC_MBPS_MAX	(DWC_USB3_EN_USB2_ONLY ==1 ? 960 * DWC_USB3_MAX_ACTIVE_U2_INSTANCES: ((4 * DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC < DWC_USB3_MAX_ACTIVE_U2_INSTANCES) ? 4000 * 2 * DWC_USB3_MAX_ACTIVE_U2_INSTANCES / 4 : 4000 * DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC * (DWC_USB3_TX_SS_PRI_CNT + 1) / DWC_USB3_TX_SS_PRI_CNT))
DWC_USB3_ASPACEWIDTH	4
DWC_USB3_BURSTWIDTH	8
DWC_USB3_BUS_CLK_REF_OFF	0
DWC_USB3_CACHE_BCU_DEPTH	(DWC_USB3_CACHE_BCU_ENTRY_SIZE * (DWC_USB3_NUM_TXFIFO_ACTIVE + DWC_USB3_NUM_RXFIFO))
DWC_USB3_CACHE_BCU_ENTRY_SIZE	0
DWC_USB3_CACHE_DEVICE_DEPTH	16

Parameter Name	Equals To
DWC_USB3_CACHE_EP_DEPTH	(16 * DWC_USB3_NUM_EPS)
DWC_USB3_CACHE_STREAMS_DEPTH	(DWC_USB3_CACHE_TOTAL_XFER_RESOURCES*((16 * DWC_USB3_CACHE_TRBS_PER_TRANSFER) + 32))
DWC_USB3_CACHE_TOTAL_XFER_RESOURCES	= (DWC_USB3_NUM_EPS)
DWC_USB3_DB CACHE_ADDR	DWC_USB3_ALIGN_MDW(DWC_USB3_HOST_CACHEEN_D_ADDR)
DWC_USB3_DB CACHE_CONTEXT_SIZE	16
DWC_USB3_DB CACHE_DBCIC_SIZE	48
DWC_USB3_DB CACHE_SIZE	(DWC_USB3_DB CACHE_DBCIC_SIZE + ((DWC_USB3_DB_NUM_EPS - 2) * (DWC_USB3_DB CACHE_CONTEXT_SIZE + DWC_USB3_DB CACHE_XFERSP_SIZE + DWC_USB3_DB CACHE_XFERTRB_SIZE)))
DWC_USB3_DB CACHE_XFERSP_SIZE	32
DWC_USB3_DB CACHE_XFERTRB_SIZE	(16 * DWC_USB3_DB TRBS_PER_TRANSFER)
DWC_USB3_DB EP0_TXF_DEPTH	(1 * (256 / DWC_USB3_MBYTES + 1) + 1)
DWC_USB3_DB EP1_TXF_DEPTH	DWC_USB3_HOST_TXFIFO_DEPTH_PER_SS_INSTANCE
DWC_USB3_DB_NUM_EPS	4
DWC_USB3_DB_RXF_DEPTH	DWC_USB3_HOST_RXFIFO_DEPTH_PER_SS_INSTANCE
DWC_USB3_DCACHE_DEPTH	(DWC_USB3_MODE == 0 ? DWC_USB3_DCACHE_DEPTH_DEV : DWC_USB3_DCACHE_DEPTH_HST : DWC_USB3_DCACHE_DEPTH_DRD)
DWC_USB3_DCACHE_DEPTH_DRD	((DWC_USB3_DCACHE_DEPTH_HST > DWC_USB3_DCACHE_DEPTH_DEV) ? DWC_USB3_DCACHE_DEPTH_HST : DWC_USB3_DCACHE_DEPTH_DEV)
DWC_USB3_DEBUGIO_HUB_DESC_RD_DONE_W	1
DWC_USB3_DEBUGIO_MP PHY_CMD_STATE_W	((DWC_USB3_NUM_SSIC_PORTS != 0) ? 3*DWC_USB3_NUM_SSIC_PORTS : 3)
DWC_USB3_DEBUGIO_RXDET_POLL_FAIL_US_W	1
DWC_USB3_DEBUGIO_SSIC_HS_GEAR_W	(2*DWC_USB3_NUM_U3_ROOT_PORTS)
DWC_USB3_DEBUGIO_SSIC_SUSPEND_CLK_EN_W	((DWC_USB3_NUM_SSIC_PORTS != 0) ? 1*DWC_USB3_NUM_SSIC_PORTS : 1)

Parameter Name	Equals To
DWC_USB3_DEBUG_WIDTH	(DWC_USB3_DEBUGIO_SSIC_SUSPEND_CLK_EN_W + DWC_USB3_DEBUGIO_SSIC_HS_GEAR_W + DWC_USB3_DEBUGIO_RXDET_POLL_FAIL_US_W + DWC_USB3_DEBUGIO_HUB_DESC_RD_DONE_W + DWC_USB3_DEBUGIO_MPHY_CMD_STATE_W + 1 + 1 + 1 + DWC_USB3_NUM_U3_ROOT_PORTS + 1 + 1 + 1 + DWC_USB3_NUM_U2_ROOT_PORTS + 1 + 1 + 2 + 4 + 5*DWC_USB3_NUM_U2TIS + 5*DWC_USB3_NUM_U2_ROOT_PORTS + 1 + 19 + 4*DWC_USB3_NUM_U3_ROOT_PORTS + 4*DWC_USB3_NUM_U3_ROOT_PORTS)
DWC_USB3_DEV_CLK_FREQ_RXDATA	DWC_USB3_CEIL_DIV((DWC_USB3_EN_USB2_ONLY ==1 ? 480: 4000), DWC_USB3_MDWIDTH)
DWC_USB3_DEV_RAM_CLK_FREQ_TURNAROUND	60
DWC_USB3_DEV_RXFIFO_OK_SPACE	((DWC_USB3_DEV_RXF_MAX_PACKET_SIZE/4) + DWC_USB3_DEV_RXFIFO_OK_SPACE_MARGIN)
DWC_USB3_DEV_RXFIFO_OK_SPACE_MARGIN	((DWC_USB3_MDWIDTH ==128 ? 12: 6) + 4)
DWC_USB3_DFDMA_CMD_DEPTH	(DWC_USB3_MDWIDTH ==32 ? 3: DWC_USB3_MDWIDTH ==64 ? 2: 1)
DWC_USB3_DFQ_CMDS_DEV	8
DWC_USB3_DFQ_CMDS_HST	8
DWC_USB3_DFQ_CMDS_PER_FSLS_INSTANCE	((DWC_USB3_EN_SEPARATE_DESC_QUEUES ==1 && DWC_USB3_MODE ==2) ? DWC_USB3_MAX_CB(DWC_USB3_DFQ_FIFO_DEPTH_ DEV / DWC_USB3_DFDMA_CMD_DEPTH, DWC_USB3_DFQ_CMDS_PER_FSLS_INSTANCE_INT): DWC_USB3_DFQ_CMDS_PER_FSLS_INSTANCE_INT)
DWC_USB3_DFQ_CMDS_PER_FSLS_INSTANCE_INT	(DWC_USB3_DFQ_CMDS_HST / 2)
DWC_USB3_DFQ_CMDS_PER_HS_INSTANCE	(DWC_USB3_DFQ_CMDS_HST / 2)
DWC_USB3_DFQ_CMDS_PER_SS_INSTANCE	DWC_USB3_DFQ_CMDS_HST
DWC_USB3_DFQ_DEPTH_PER_FSLS_INSTANCE	(DWC_USB3_DFQ_CMDS_PER_FSLS_INSTANCE * DWC_USB3_DFDMA_CMD_DEPTH)
DWC_USB3_DFQ_DEPTH_PER_HS_INSTANCE	(DWC_USB3_DFQ_CMDS_PER_HS_INSTANCE * DWC_USB3_DFDMA_CMD_DEPTH)
DWC_USB3_DFQ_DEPTH_PER_SS_INSTANCE	(DWC_USB3_DFQ_CMDS_PER_SS_INSTANCE * DWC_USB3_DFDMA_CMD_DEPTH)

Parameter Name	Equals To
DWC_USB3_DFQ_FIFO_ALL_DEPTH	(DWC_USB3_MODE ==0 ? DWC_USB3_DFQ_FIFO_DEPTH_DEV: DWC_USB3_EN_SEPARATE_DESC_QUEUES ==0 ? DWC_USB3_DFQ_FIFO_DEPTH_HST_ONEQ: DWC_USB3_NUM_FSLS_USB_INSTANCES * DWC_USB3_DFQ_DEPTH_PER_FSLS_INSTANCE + DWC_USB3_NUM_HS_USB_INSTANCES * DWC_USB3_DFQ_DEPTH_PER_HS_INSTANCE + DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * DWC_USB3_DFQ_DEPTH_PER_SS_INSTANCE + DWC_USB3_EN_DBC_SRCSNK * DWC_USB3_DFQ_FIFO_DEPTH_DEV)
DWC_USB3_DFQ_FIFO_DEPTH_DEV	(DWC_USB3_EN_SCALED_DESC_QUEUE_DEPTHS==0 ? 16: DWC_USB3_DFQ_CMDS_DEV * DWC_USB3_DFDMA_CMD_DEPTH)
DWC_USB3_DFQ_FIFO_DEPTH_HST_ONEQ	(DWC_USB3_EN_SCALED_DESC_QUEUE_DEPTHS==0 ? 16: DWC_USB3_DFQ_CMDS_HST * DWC_USB3_DFDMA_CMD_DEPTH)
DWC_USB3_DWDMA_DEV_CMD_DEPTH	(DWC_USB3_MDWIDTH ==32 ? 5: DWC_USB3_MDWIDTH ==64 ? 3: 2)
DWC_USB3_DWDMA_HST_CMD_DEPTH	(DWC_USB3_MDWIDTH ==32 ? 6: DWC_USB3_MDWIDTH ==64 ? 3: 2)
DWC_USB3_DWQ_CMDS_DEV	10
DWC_USB3_DWQ_CMDS_HST	10
DWC_USB3_DWQ_CMDS_PER_FSLS_INSTANCE	((DWC_USB3_EN_SEPARATE_DESC_QUEUES ==1 && DWC_USB3_MODE ==2) ? DWC_USB3_MAX_CB(DWC_USB3_DWQ_FIFO_DEPTH_ DEV / DWC_USB3_DWDMA_HST_CMD_DEPTH, DWC_USB3_DWQ_CMDS_HST / 2): DWC_USB3_DWQ_CMDS_HST / 2)
DWC_USB3_DWQ_CMDS_PER_HS_INSTANCE	(DWC_USB3_DWQ_CMDS_HST / 2)
DWC_USB3_DWQ_CMDS_PER_SS_INSTANCE	DWC_USB3_DWQ_CMDS_HST
DWC_USB3_DWQ_DEPTH_PER_FSLS_INSTANCE	(DWC_USB3_DWQ_CMDS_PER_FSLS_INSTANCE * DWC_USB3_DWDMA_HST_CMD_DEPTH)
DWC_USB3_DWQ_DEPTH_PER_HS_INSTANCE	(DWC_USB3_DWQ_CMDS_PER_HS_INSTANCE * DWC_USB3_DWDMA_HST_CMD_DEPTH)
DWC_USB3_DWQ_DEPTH_PER_SS_INSTANCE	(DWC_USB3_DWQ_CMDS_PER_SS_INSTANCE * DWC_USB3_DWDMA_HST_CMD_DEPTH)

Parameter Name	Equals To
DWC_USB3_DWQ_FIFO_ALL_DEPTH	(DWC_USB3_MODE ==0 ? DWC_USB3_DWQ_FIFO_DEPTH_DEV: DWC_USB3_EN_SEPARATE_DESC_QUEUES ==0 ? DWC_USB3_DWQ_FIFO_DEPTH_HST_ONEQ: DWC_USB3_NUM_FSLS_USB_INSTANCES * DWC_USB3_DWQ_DEPTH_PER_FSLS_INSTANCE + DWC_USB3_NUM_HS_USB_INSTANCES * DWC_USB3_DWQ_DEPTH_PER_HS_INSTANCE + DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * DWC_USB3_DWQ_DEPTH_PER_SS_INSTANCE + DWC_USB3_EN_DBC_SRCSNK * DWC_USB3_DWQ_FIFO_DEPTH_DEV)
DWC_USB3_DWQ_FIFO_DEPTH_DEV	(DWC_USB3_EN_SCALED_DESC_QUEUE_DEPTHS==0 ? 32: DWC_USB3_DWQ_CMDS_DEV * DWC_USB3_DWDMA_DEV_CMD_DEPTH)
DWC_USB3_DWQ_FIFO_DEPTH_HST_ONEQ	(DWC_USB3_EN_SCALED_DESC_QUEUE_DEPTHS==0 ? 32: DWC_USB3_DWQ_CMDS_HST * DWC_USB3_DWDMA_HST_CMD_DEPTH)
DWC_USB3_ECC_CHK_WIDTH	((DWC_USB3_MDWIDTH == 128) ? 9 : 8)
DWC_USB3_ENABLE_LPM	= (DWC_USB3_MODE ==3) ? 0 : 1
DWC_USB3_EN_ADP	= 0
DWC_USB3_EN_DBC_SRCSNK	((DWC_USB3_EN_DBC ==1 & (DWC_USB3_HOST_NUM_U3_ROOT_PORTS > DWC_USB3_NUM_SS_USB_INSTANCES)) ? 1: 0)
DWC_USB3_EN_OTG	0
DWC_USB3_EN_OTG_SS	((DWC_USB3_EN_OTG == 2) ? 1 : 0)
DWC_USB3_EN_PMU_DEBUG	0
DWC_USB3_EN_SCALED_DESC_QUEUE_DEPTHS	= (DWC_USB3_MDWIDTH<=64 ? 0: 1)
DWC_USB3_EVDMA_CMD_DEPTH	(DWC_USB3_MDWIDTH ==32 ? 2: 1)
DWC_USB3_EVQ_FIFO_ALL_DEPTH	(DWC_USB3_NUM_EVQ_ACTIVE * DWC_USB3_EVQ_FIFO_DEPTH)
DWC_USB3_EVQ_FIFO_DEPTH	(DWC_USB3_EN_SEPARATE_DESC_QUEUES ==0 ? 0: (DWC_USB3_NUM_FSLS_USB_INSTANCES * DWC_USB3_DWQ_CMDS_PER_FSLS_INSTANCE + DWC_USB3_NUM_HS_USB_INSTANCES * DWC_USB3_DWQ_CMDS_PER_HS_INSTANCE + DWC_USB3_NUM_SS_USB_INSTANCES * DWC_USB3_DWQ_CMDS_PER_SS_INSTANCE) * DWC_USB3_EVDMA_CMD_DEPTH)
DWC_USB3_FSPHY_INTERFACE	0

Parameter Name	Equals To
DWC_USB3_HCACHE_BI_BANDWIDTH	((DWC_USB3_HCACHE_DEVICE_ADDRESS_TABLE + DWC_USB3_HCACHE_DEVICE_ADDRESS_TABLE_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_BI_BANDWIDTH_SIZE	(DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE*4) + (DWC_USB3_NUM_HS_USB_INSTANCES*4) + (DWC_USB3_NUM_FSLs_USB_INSTANCES*4)
DWC_USB3_HCACHE_CMD_HANDLER	((DWC_USB3_HCACHE_SS_TMPTRB + DWC_USB3_HCACHE_SS_TMPTRB_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_CMD_RING_POINTER	((DWC_USB3_HCACHE_CMD_HANDLER + DWC_USB3_HCACHE_CMD_STORE_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_CMD_RING_POINTER_SIZE	8
DWC_USB3_HCACHE_CMD_STORE_SIZE	16
DWC_USB3_HCACHE_DEVICE_ADDRESS_TABLE	((DWC_USB3_HCACHE_SLOT_TABLE + DWC_USB3_HCACHE_SLOT_TABLE_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_DEVICE_ADDRESS_TABLE_SIZE	16
DWC_USB3_HCACHE_DEVICE_BASE_ADDR	((DWC_USB3_HCACHE_HERH_DATA + DWC_USB3_HCACHE_HERH_DATA_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_DEVICE_BASE_ADDR_SIZE	(DWC_USB3_NUM_DEVICE_SUPT + 1) * 8
DWC_USB3_HCACHE_DEVICE_CONTEXT_ADDR	((DWC_USB3_HCACHE_CMD_RING_POINTER + DWC_USB3_HCACHE_CMD_RING_POINTER_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_DEVICE_CONTEXT_ADDR_SIZE	8
DWC_USB3_HCACHE_DEVICE_EP_CONTEXT	((DWC_USB3_HCACHE_DEVICE_SLOT_CONTEXT + DWC_USB3_HCACHE_DEVICE_SLOT_CONTEXT_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_DEVICE_EP_CONTEXT_SIZE	36
DWC_USB3_HCACHE_DEVICE_SLOT_CONTEXT	((DWC_USB3_HCACHE_DEVICE_CONTEXT_ADDR + DWC_USB3_HCACHE_DEVICE_CONTEXT_ADDR_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_DEVICE_SLOT_CONTEXT_SIZE	32
DWC_USB3_HCACHE_EPINFO_BITMAP	(DWC_USB3_HCACHE_HUB_CONTEXT_ADDR + DWC_USB3_HCACHE_HUB_CONTEXT_ADDR_SIZE)

Parameter Name	Equals To
DWC_USB3_HCACHE_EPINFO_BITMAP_SIZE	((DWC_USB3_NUM_DEVICE_SUPT + 1) * DWC_USB3_HCACHE_EPINFO_BITMAP_SIZE_PER_SLOT)
DWC_USB3_HCACHE_EPINFO_BITMAP_SIZE_PER_SLOT	16
DWC_USB3_HCACHE_EPTYPE_BITMAP	(DWC_USB3_HCACHE_EPINFO_BITMAP + DWC_USB3_HCACHE_EPINFO_BITMAP_SIZE)
DWC_USB3_HCACHE_EPTYPE_BITMAP_SIZE	((DWC_USB3_NUM_DEVICE_SUPT + 1) * DWC_USB3_HCACHE_EPTYPE_BITMAP_SIZE_PER_SLOT)
DWC_USB3_HCACHE_EPTYPE_BITMAP_SIZE_PER_SLOT	8
DWC_USB3_HCACHE_FSLs_EP	(DWC_USB3_HCACHE_SCRATCH_PAD + DWC_USB3_HCACHE_SCRATCH_PAD_SIZE)
DWC_USB3_HCACHE_FSLs_EPHEADER_SIZE	80
DWC_USB3_HCACHE_FSLs_EP_SIZE	(DWC_USB3_NUM_FSLs_USB_INSTANCES * (DWC_USB3_HOST_NUM_CACHE_EP_PER_FSLs_INSTANCE * (DWC_USB3_HCACHE_FSLs_EPHEADER_SIZE + DWC_USB3_HOST_NUM_CACHE_TRB_PER_FSLs_EP_INT * 16)))
DWC_USB3_HCACHE_FSLs_TMPTRB	(DWC_USB3_HCACHE_FSLs_TRQ + DWC_USB3_HCACHE_FSLs_TRQ_SIZE)
DWC_USB3_HCACHE_FSLs_TMPTRB_SIZE	(DWC_USB3_NUM_FSLs_USB_INSTANCES * 16)
DWC_USB3_HCACHE_FSLs_TRQ	(DWC_USB3_HCACHE_FSLs_EP + DWC_USB3_HCACHE_FSLs_EP_SIZE)
DWC_USB3_HCACHE_FSLs_TRQ_SIZE	(DWC_USB3_NUM_FSLs_USB_INSTANCES * (DWC_USB3_HOST_NUM_CACHE_EP_PER_FSLs_INSTANCE * DWC_USB3_HCACHE_TR_SIZE))
DWC_USB3_HCACHE_HALM_DATA	(DWC_USB3_HCACHE_HUB_SLOT_CONTEXT + DWC_USB3_HCACHE_HUB_SLOT_CONTEXT_SIZE)
DWC_USB3_HCACHE_HALM_DATA_SIZE	((64 + 16)*4)
DWC_USB3_HCACHE_HCMD_TRB	((DWC_USB3_HCACHE_PORT_BANDWIDTH_CONTEXT + DWC_USB3_HCACHE_PORT_BANDWIDTH_CONTEXT_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_HCMD_TRB_SIZE	16

Parameter Name	Equals To
DWC_USB3_HCACHE_HERH_DATA	(DWC_USB3_HCACHE_BI_BANDWIDTH + DWC_USB3_HCACHE_BI_BANDWIDTH_SIZE + 12)
DWC_USB3_HCACHE_HERH_DATA_SIZE	(4 + 2*16 + 2*16)*(DWC_USB3_HOST_NUM_INTERRUPTER_SUPT + DWC_USB3_EN_DBC)
DWC_USB3_HCACHE_HS_EP	(DWC_USB3_HCACHE_FSLS_TMPTRB + DWC_USB3_HCACHE_FSLS_TMPTRB_SIZE)
DWC_USB3_HCACHE_HS_EPHEADER_SIZE	80
DWC_USB3_HCACHE_HS_EP_SIZE	(DWC_USB3_NUM_HS_USB_INSTANCES * (DWC_USB3_HOST_NUM_CACHE_EP_PER_HS_INSTANCE * (DWC_USB3_HCACHE_HS_EPHEADER_SIZE + DWC_USB3_HOST_NUM_CACHE_TRB_PER_HS_EP_IN T * 16)))
DWC_USB3_HCACHE_HS_TMPTRB	(DWC_USB3_HCACHE_HS_TRQ + DWC_USB3_HCACHE_HS_TRQ_SIZE)
DWC_USB3_HCACHE_HS_TMPTRB_SIZE	(DWC_USB3_NUM_HS_USB_INSTANCES * 16)
DWC_USB3_HCACHE_HS_TRQ	(DWC_USB3_HCACHE_HS_EP + DWC_USB3_HCACHE_HS_EP_SIZE)
DWC_USB3_HCACHE_HS_TRQ_SIZE	(DWC_USB3_NUM_HS_USB_INSTANCES * (DWC_USB3_HOST_NUM_CACHE_EP_PER_HS_INSTANCE * DWC_USB3_HCACHE_TR_SIZE))
DWC_USB3_HCACHE_HUB_CONTEXT_ADDR	(DWC_USB3_ADDR_RCSR_HOST + DWC_USB3_RCSR_HOST_DEPTH)
DWC_USB3_HCACHE_HUB_CONTEXT_ADDR_SIZE	8
DWC_USB3_HCACHE_HUB_SLOT_CONTEXT	((DWC_USB3_HCACHE_HCMD_TRB + DWC_USB3_HCACHE_HCMD_TRB_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_HUB_SLOT_CONTEXT_SIZE	16
DWC_USB3_HCACHE_INPUT_CONTROL_CONTEXT	((DWC_USB3_HCACHE_INPUT_EP_CONTEXT + DWC_USB3_HCACHE_INPUT_EP_CONTEXT_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_INPUT_CONTROL_CONTEXT_SIZE	8
DWC_USB3_HCACHE_INPUT_EP_CONTEXT	((DWC_USB3_HCACHE_INPUT_SLOT_CONTEXT + DWC_USB3_HCACHE_INPUT_SLOT_CONTEXT_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_INPUT_EP_CONTEXT_SIZE	20

Parameter Name	Equals To
DWC_USB3_HCACHE_INPUT_SLOT_CONTEXT	$((DWC_USB3_HCACHE_DEVICE_EP_CONTEXT + DWC_USB3_HCACHE_DEVICE_EP_CONTEXT_SIZE + 15) \gg 4) \ll 4$
DWC_USB3_HCACHE_INPUT_SLOT_CONTEXT_SIZE	20
DWC_USB3_HCACHE_PERIODIC_EPINFO_SIZE	20
DWC_USB3_HCACHE_PERIODIC_LIST	$(DWC_USB3_HCACHE_EPTYPE_BITMAP + DWC_USB3_HCACHE_EPTYPE_BITMAP_SIZE)$
DWC_USB3_HCACHE_PERIODIC_LIST_SIZE	$(DWC_USB3_HOST_NUM_PERIODIC_EP * DWC_USB3_HCACHE_PERIODIC_EPINFO_SIZE)$
DWC_USB3_HCACHE_PERIODIC_SLOTINFO_SIZE	4
DWC_USB3_HCACHE_PORT_BANDWIDTH_CONTEXT	$((DWC_USB3_HCACHE_SECONDARY_STREAM_CONTEXT_ADDR + DWC_USB3_HCACHE_SECONDARY_STREAM_CONTEXT_ADDR_SIZE + 15) \gg 4) \ll 4$
DWC_USB3_HCACHE_PRIMARY_STREAM_CONTEXT_ADDR	$((DWC_USB3_HCACHE_SCRATCH_PAD_ADDR + DWC_USB3_HCACHE_SCRATCH_PAD_ADDR_SIZE + 15) \gg 4) \ll 4$
DWC_USB3_HCACHE_PRIMARY_STREAM_CONTEXT_ADDR_SIZE	16
DWC_USB3_HCACHE_SCRATCH_PAD	$(DWC_USB3_HCACHE_SLOT_ID + DWC_USB3_HCACHE_SLOT_ID_SIZE)$
DWC_USB3_HCACHE_SCRATCH_PAD_ADDR	$((DWC_USB3_HCACHE_INPUT_CONTROL_CONTEXT + DWC_USB3_HCACHE_INPUT_CONTROL_CONTEXT_SIZE + 15) \gg 4) \ll 4$
DWC_USB3_HCACHE_SCRATCH_PAD_ADDR_SIZE	16
DWC_USB3_HCACHE_SCRATCH_PAD_SIZE	64
DWC_USB3_HCACHE_SECONDARY_STREAM_CONTEXT_ADDR	$((DWC_USB3_HCACHE_PRIMARY_STREAM_CONTEXT_ADDR + DWC_USB3_HCACHE_PRIMARY_STREAM_CONTEXT_ADDR_SIZE + 15) \gg 4) \ll 4$
DWC_USB3_HCACHE_SECONDARY_STREAM_CONTEXT_ADDR_SIZE	16
DWC_USB3_HCACHE_SLOT_ID	$(DWC_USB3_HCACHE_DEVICE_BASE_ADDR + DWC_USB3_HCACHE_DEVICE_BASE_ADDR_SIZE)$
DWC_USB3_HCACHE_SLOT_ID_SIZE	128
DWC_USB3_HCACHE_SLOT_LIST	$(DWC_USB3_HCACHE_PERIODIC_LIST + DWC_USB3_HCACHE_PERIODIC_LIST_SIZE)$

Parameter Name	Equals To
DWC_USB3_HCACHE_SLOT_LIST_SIZE	((DWC_USB3_NUM_DEVICE_SUPT + 1) * DWC_USB3_HCACHE_PERIODIC_SLOTINFO_SIZE)
DWC_USB3_HCACHE_SLOT_TABLE	((DWC_USB3_HCACHE_SLOT_LIST + DWC_USB3_HCACHE_SLOT_LIST_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_SLOT_TABLE_SIZE	((DWC_USB3_NUM_DEVICE_SUPT/32) + 1) * 4
DWC_USB3_HCACHE_SS_EP	((DWC_USB3_HCACHE_HS_TMPTRB + DWC_USB3_HCACHE_HS_TMPTRB_SIZE + 15) >>4) <<4)
DWC_USB3_HCACHE_SS_EPHEADER_SIZE	96
DWC_USB3_HCACHE_SS_EP_SIZE	(DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * (DWC_USB3_HOST_NUM_CACHE_EP_PER_SS_INSTANCE * (DWC_USB3_HCACHE_SS_EPHEADER_SIZE + DWC_USB3_HOST_NUM_CACHE_TRB_PER_SS_EP_IN T * 16)))
DWC_USB3_HCACHE_SS_TMPTRB	(DWC_USB3_HCACHE_SS_TRQ + DWC_USB3_HCACHE_SS_TRQ_SIZE)
DWC_USB3_HCACHE_SS_TMPTRB_SIZE	(DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * 16)
DWC_USB3_HCACHE_SS_TRQ	(DWC_USB3_HCACHE_SS_EP + DWC_USB3_HCACHE_SS_EP_SIZE)
DWC_USB3_HCACHE_SS_TRQ_SIZE	(DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * DWC_USB3_HOST_NUM_CACHE_EP_PER_SS_INSTANCE * DWC_USB3_HCACHE_TR_SIZE)
DWC_USB3_HCACHE_TR_SIZE	8
DWC_USB3_HC_MAXPORTS_PIN	((DWC_USB3_MODE ==0) ? 1 : ((DWC_USB3_MODE ==1 DWC_USB3_MODE ==2) ? DWC_USB3_HOST_NUM_ROOT_PORTS_PIN : ((DWC_USB3_MODE ==3) ? (DWC_USB3_HUB_NUM_U3_PORTS + 1) : 1)))
DWC_USB3_HOST_20_SNGL_PRT_OPT	= 0
DWC_USB3_HOST_CACHEEND_ADDR	(DWC_USB3_HCACHE_HALM_DATA + DWC_USB3_HCACHE_HALM_DATA_SIZE)
DWC_USB3_HOST_CLK_FREQ_RXDATA	DWC_USB3_CEIL_DIV(DWC_USB3_ALL_RXSNK_MBPS _MAX, DWC_USB3_MDWIDTH)
DWC_USB3_HOST_NUM_CACHE_TRB_PER_FSLS_EP_INT	(DWC_USB3_HOST_NUM_CACHE_TRB_PER_FSLS_EP + 2)
DWC_USB3_HOST_NUM_CACHE_TRB_PER_HS_EP_IN T	(DWC_USB3_HOST_NUM_CACHE_TRB_PER_HS_EP + 2)

Parameter Name	Equals To
DWC_USB3_HOST_NUM_CACHE_TRB_PER_SS_EP_IN_T	(DWC_USB3_HOST_NUM_CACHE_TRB_PER_SS_EP + 2)
DWC_USB3_HOST_NUM_ROOT_PORTS	((DWC_USB3_MODE ==0) ? 1 : (DWC_USB3_HOST_NUM_U2_ROOT_PORTS + DWC_USB3_HOST_NUM_U3_ROOT_PORTS))
DWC_USB3_HOST_NUM_ROOT_PORTS_PIN	((DWC_USB3_MODE ==0) ? 1 : (DWC_USB3_HOST_NUM_U2_ROOT_PORTS_PIN + DWC_USB3_HOST_NUM_U3_ROOT_PORTS_PIN))
DWC_USB3_HOST_NUM_U2_ROOT_PORTS_PIN	DWC_USB3_HOST_NUM_U2_ROOT_PORTS
DWC_USB3_HOST_NUM_U3_ROOT_PORTS_PIN	((DWC_USB3_EN_USB2_ONLY ==1) ? 0: DWC_USB3_HOST_NUM_U3_ROOT_PORTS)
DWC_USB3_HOST_RAM_CLK_FREQ_FIFOLATENCY	60
DWC_USB3_HOST_RAM_CLK_FREQ_TXDATA	DWC_USB3_CEIL_DIV(DWC_USB3_ALL_TXSRC_MBPS_MAX, DWC_USB3_MDWIDTH)
DWC_USB3_HOST_RIQ_DEPTH_PER_FSLS_INSTANCE	((DWC_USB3_MODE ==2 ? (DWC_USB3_HOST_RXQ_DEPTH_PER_FSLS_INSTANCE + (DWC_USB3_PSQ_FIFO_DEPTH/2) +2): (DWC_USB3_HOST_RXQ_DEPTH_PER_FSLS_INSTANCE + 2))
DWC_USB3_HOST_RIQ_DEPTH_PER_HS_INSTANCE	(DWC_USB3_HOST_RXQ_DEPTH_PER_HS_INSTANCE + 2)
DWC_USB3_HOST_RIQ_DEPTH_PER_SS_INSTANCE	(DWC_USB3_HOST_RXQ_DEPTH_PER_SS_INSTANCE + 2)
DWC_USB3_HOST_RXFIFO_DEPTH_PER_FSLS_INSTANCE	((DWC_USB3_NUM_RXF_FSLS_PKTS * DWC_USB3_MPS + 4 * 8) / DWC_USB3_MBYTES)
DWC_USB3_HOST_RXFIFO_DEPTH_PER_HS_INSTANCE	((DWC_USB3_NUM_RXF_HS_PKTS * DWC_USB3_MPS + 4 * 8) / DWC_USB3_MBYTES)
DWC_USB3_HOST_RXFIFO_DEPTH_PER_SS_INSTANCE	((DWC_USB3_NUM_RXF_SS_PKTS * DWC_USB3_MPS) / DWC_USB3_MBYTES)
DWC_USB3_HOST_RXQ_DEPTH_PER_FSLS_INSTANCE	((DWC_USB3_MODE ==2) ? DWC_USB3_MAX_CB(DWC_USB3_HOST_RXQ_DEPTH_PER_FSLS_INSTANCE_HOST, DWC_USB3_RXQ_FIFO_DEPTH): DWC_USB3_HOST_RXQ_DEPTH_PER_FSLS_INSTANCE_HOST)
DWC_USB3_HOST_RXQ_DEPTH_PER_FSLS_INSTANCE_HOST	(2 * (1 * DWC_USB3_RXDMA_CMD_DEPTH + (DWC_USB3_HOST_NUM_CACHE_TRB_PER_FSLS_EP -1) * DWC_USB3_RXDMA_CMD_MSEG_FIFO_DEPTH))

Parameter Name	Equals To
DWC_USB3_HOST_RXQ_DEPTH_PER_HS_INSTANCE	((DWC_USB3_MODE ==2) ? DWC_USB3_MAX_CB(DWC_USB3_HOST_RXQ_DEPTH _PER_HS_INSTANCE_HOST, DWC_USB3_RXQ_FIFO_DEPTH): DWC_USB3_HOST_RXQ_DEPTH_PER_HS_INSTANCE HOST)
DWC_USB3_HOST_RXQ_DEPTH_PER_HS_INSTANCE_HOST	(2 * (3 * DWC_USB3_RXDMA_CMD_DEPTH + (DWC_USB3_HOST_NUM_CACHE_TRB_PER_HS_EP-1) * DWC_USB3_RXDMA_CMD_MSEG_FIFO_DEPTH))
DWC_USB3_HOST_RXQ_DEPTH_PER_SS_INSTANCE	(2 * (16 * DWC_USB3_RXDMA_CMD_DEPTH + (DWC_USB3_HOST_NUM_CACHE_TRB_PER_SS_EP-1) * DWC_USB3_RXDMA_CMD_MSEG_FIFO_DEPTH))
DWC_USB3_HOST_TXFIFO_DEPTH_PER_FSLS_INSTANCE	((DWC_USB3_NUM_TXF_FSLS_PKTS * (DWC_USB3_MPS / DWC_USB3_MBYTES + 1)) + 1)
DWC_USB3_HOST_TXFIFO_DEPTH_PER_HS_INSTANCE	((DWC_USB3_NUM_TXF_HS_PKTS * (DWC_USB3_MPS / DWC_USB3_MBYTES + 1)) + 1)
DWC_USB3_HOST_TXFIFO_DEPTH_PER_SS_INSTANCE	((DWC_USB3_NUM_TXF_SS_PKTS * (DWC_USB3_MPS / DWC_USB3_MBYTES + 1)) + 1)
DWC_USB3_HOST_TXQ_DEPTH_PER_FSLS_INSTANCE	((DWC_USB3_MODE ==2) ? DWC_USB3_MAX_CB(DWC_USB3_HOST_TXQ_DEPTH _PER_FSLS_INSTANCE_HOST, DWC_USB3_TXQ_FIFO_DEPTH): DWC_USB3_HOST_TXQ_DEPTH_PER_FSLS_INSTANCE HOST)
DWC_USB3_HOST_TXQ_DEPTH_PER_FSLS_INSTANCE_HOST	(2 * (1 * DWC_USB3_RXDMA_CMD_DEPTH + (DWC_USB3_HOST_NUM_CACHE_TRB_PER_FSLS_EP -1) * DWC_USB3_RXDMA_CMD_MSEG_FIFO_DEPTH))
DWC_USB3_HOST_TXQ_DEPTH_PER_HS_INSTANCE	((DWC_USB3_MODE ==2) ? DWC_USB3_MAX_CB(DWC_USB3_HOST_TXQ_DEPTH _PER_HS_INSTANCE_HOST, DWC_USB3_TXQ_FIFO_DEPTH): DWC_USB3_HOST_TXQ_DEPTH_PER_HS_INSTANCE HOST)
DWC_USB3_HOST_TXQ_DEPTH_PER_HS_INSTANCE_HOST	(2 * (3 * DWC_USB3_RXDMA_CMD_DEPTH + (DWC_USB3_HOST_NUM_CACHE_TRB_PER_HS_EP-1) * DWC_USB3_RXDMA_CMD_MSEG_FIFO_DEPTH))
DWC_USB3_HOST_TXQ_DEPTH_PER_SS_INSTANCE	((2 + DWC_USB3_EN_ISOC_SUPT) * (16 * DWC_USB3_RXDMA_CMD_DEPTH + (DWC_USB3_HOST_NUM_CACHE_TRB_PER_SS_EP-1) * DWC_USB3_RXDMA_CMD_MSEG_FIFO_DEPTH))
DWC_USB3_HUB_OVERRIDE_PARAMS_WIDTH	1764

Parameter Name	Equals To
DWC_USB3_I2C_INTERFACE	0
DWC_USB3_LEGACY_SUPT	0
DWC_USB3_LSP_SHARED_RAM_FREQ_FUNC	(DWC_USB3_LSP_SHARED_RAM_FREQ_PERF * 1)
DWC_USB3_LSP_SHARED_RAM_FREQ_PERF	(DWC_USB3_NUM_RAMSS <= 2 ? 15: 0)
DWC_USB3_MADDR_LO	((DWC_USB3_MDWIDTH == 32) ? 2: (DWC_USB3_MDWIDTH == 64) ? 3: 4)
DWC_USB3_MAX_ACTIVE_U2_INSTANCES	DWC_USB3_MIN(DWC_USB3_HOST_NUM_U2_ROOT_PORTS, DWC_USB3_NUM_HSFSLS_USB_INSTANCES)
DWC_USB3_MB_BYTES	(DWC_USB3_MDWIDTH / 8)
DWC_USB3_MDLOG	= (DWC_USB3_MDWIDTH/8)
DWC_USB3_MDWORDS	(DWC_USB3_MDWIDTH / 32)
DWC_USB3_MISCWIDTH	4
DWC_USB3_MPS	1024
DWC_USB3_NPI_IN_EP0_FIFO_DEPTH	DWC_USB3_DEV_TXF0_DEPTH
DWC_USB3_NPI_IN_EP10_FIFO_DEPTH	DWC_USB3_DEV_TXF10_DEPTH
DWC_USB3_NPI_IN_EP11_FIFO_DEPTH	DWC_USB3_DEV_TXF11_DEPTH
DWC_USB3_NPI_IN_EP12_FIFO_DEPTH	DWC_USB3_DEV_TXF12_DEPTH
DWC_USB3_NPI_IN_EP13_FIFO_DEPTH	DWC_USB3_DEV_TXF13_DEPTH
DWC_USB3_NPI_IN_EP14_FIFO_DEPTH	DWC_USB3_DEV_TXF14_DEPTH
DWC_USB3_NPI_IN_EP15_FIFO_DEPTH	DWC_USB3_DEV_TXF15_DEPTH
DWC_USB3_NPI_IN_EP1_FIFO_DEPTH	DWC_USB3_DEV_TXF1_DEPTH
DWC_USB3_NPI_IN_EP2_FIFO_DEPTH	DWC_USB3_DEV_TXF2_DEPTH
DWC_USB3_NPI_IN_EP3_FIFO_DEPTH	DWC_USB3_DEV_TXF3_DEPTH
DWC_USB3_NPI_IN_EP4_FIFO_DEPTH	DWC_USB3_DEV_TXF4_DEPTH
DWC_USB3_NPI_IN_EP5_FIFO_DEPTH	DWC_USB3_DEV_TXF5_DEPTH
DWC_USB3_NPI_IN_EP6_FIFO_DEPTH	DWC_USB3_DEV_TXF6_DEPTH
DWC_USB3_NPI_IN_EP7_FIFO_DEPTH	DWC_USB3_DEV_TXF7_DEPTH
DWC_USB3_NPI_IN_EP8_FIFO_DEPTH	DWC_USB3_DEV_TXF8_DEPTH
DWC_USB3_NPI_IN_EP9_FIFO_DEPTH	DWC_USB3_DEV_TXF9_DEPTH
DWC_USB3_NPI_N	(DWC_USB3_MBUS_TYPE != 4)

Parameter Name	Equals To
DWC_USB3_NPI_TX_FIFO_DEPTH	(DWC_USB3_NPI_IN_EP0_FIFO_DEPTH + DWC_USB3_NPI_IN_EP1_FIFO_DEPTH + DWC_USB3_NPI_IN_EP2_FIFO_DEPTH + DWC_USB3_NPI_IN_EP3_FIFO_DEPTH + DWC_USB3_NPI_IN_EP4_FIFO_DEPTH + DWC_USB3_NPI_IN_EP5_FIFO_DEPTH + DWC_USB3_NPI_IN_EP6_FIFO_DEPTH + DWC_USB3_NPI_IN_EP7_FIFO_DEPTH + DWC_USB3_NPI_IN_EP8_FIFO_DEPTH + DWC_USB3_NPI_IN_EP9_FIFO_DEPTH + DWC_USB3_NPI_IN_EP10_FIFO_DEPTH + DWC_USB3_NPI_IN_EP11_FIFO_DEPTH + DWC_USB3_NPI_IN_EP12_FIFO_DEPTH + DWC_USB3_NPI_IN_EP13_FIFO_DEPTH + DWC_USB3_NPI_IN_EP14_FIFO_DEPTH + DWC_USB3_NPI_IN_EP15_FIFO_DEPTH)
DWC_USB3_NUM_EVQ_ACTIVE	(DWC_USB3_EN_SEPARATE_DESC_QUEUES ==1 ? DWC_USB3_HOST_NUM_INTERRUPTER_SUPT: 0)
DWC_USB3_NUM_FSLS_USB_INSTANCES	1
DWC_USB3_NUM_HSFSL_USB_INSTANCES	(DWC_USB3_NUM_HS_USB_INSTANCES + DWC_USB3_NUM_FSLS_USB_INSTANCES)
DWC_USB3_NUM_INT	((DWC_USB3_MODE == 0) ? DWC_USB3_DEVICE_NUM_INT : ((DWC_USB3_MODE == 1) ? DWC_USB3_HOST_NUM_INTERRUPTER_SUPT : ((DWC_USB3_HOST_NUM_INTERRUPTER_SUPT > DWC_USB3_DEVICE_NUM_INT) ? DWC_USB3_HOST_NUM_INTERRUPTER_SUPT : DWC_USB3_DEVICE_NUM_INT)))
DWC_USB3_NUM_RXFIFO	(DWC_USB3_MODE ==0 ? 1: DWC_USB3_NUM_TXFIFO_HOST - DWC_USB3_EN_DBC)
DWC_USB3_NUM_SSIC_PORTS	= 0
DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE	((DWC_USB3_EN_USB2_ONLY ==1) ? 0: DWC_USB3_NUM_SS_USB_INSTANCES)
DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC	(DWC_USB3_EN_DBC_SRCSNK + DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE)
DWC_USB3_NUM_TXFIFO_ACTIVE	(DWC_USB3_MODE ==0 ? DWC_USB3_NUM_IN_EPS: DWC_USB3_MODE ==1 ? DWC_USB3_NUM_TXFIFO_HOST_ACTIVE: DWC_USB3_NUM_TXFIFO_SDRD_ACTIVE)

Parameter Name	Equals To
DWC_USB3_NUM_TXFIFO_HOST	(DWC_USB3_EN_DBC + (DWC_USB3_EN_DBC & (DWC_USB3_HOST_NUM_U3_ROOT_PORTS > DWC_USB3_NUM_SS_USB_INSTANCES)) + DWC_USB3_NUM_SS_USB_INSTANCES + DWC_USB3_NUM_HS_USB_INSTANCES + DWC_USB3_NUM_FSLS_USB_INSTANCES)
DWC_USB3_NUM_TXFIFO_HOST_ACTIVE	(DWC_USB3_EN_USB2_ONLY == 1 ? DWC_USB3_NUM_TXFIFO_HOST-1 : DWC_USB3_NUM_TXFIFO_HOST)
DWC_USB3_NUM_TXFIFO_SDRD_ACTIVE	DWC_USB3_MAX_CB(DWC_USB3_NUM_TXFIFO_HOST_ACTIVE, DWC_USB3_NUM_IN_EPS)
DWC_USB3_NUM_U2_ROOT_PORTS	= ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) ? DWC_USB3_HOST_NUM_U2_ROOT_PORTS : 1)
DWC_USB3_NUM_U2TIS	((DWC_USB3_MODE == 0) ? 1 : ((DWC_USB3_NUM_U2_ROOT_PORTS == 1) && (DWC_USB3_HOST_20_SNGL_PRT_OPT == 1)) ? 1 : (DWC_USB3_NUM_HS_USB_INSTANCES + DWC_USB3_NUM_FSLS_USB_INSTANCES))
DWC_USB3_NUM_U3_ROOT_PORTS	= ((DWC_USB3_MODE == 1 DWC_USB3_MODE == 2) ? DWC_USB3_HOST_NUM_U3_ROOT_PORTS : (DWC_USB3_MODE == 3) ? (DWC_USB3_HUB_NUM_U3_PORTS + 1) : 1)
DWC_USB3_PMUCFG_U2_RFILE	30
DWC_USB3_PMUCFG_U2_RFILE_W	(DWC_USB3_NUM_U3_ROOT_PORTS + (DWC_USB3_NUM_U2_ROOT_PORTS * 58) + 62)
DWC_USB3_PMUCFG_U3_RFILE	(DWC_USB3_PMUCFG_U2_RFILE + DWC_USB3_PMUCFG_U2_RFILE_W)
DWC_USB3_PMUCFG_U3_RFILE_W	((DWC_USB3_NUM_U3_ROOT_PORTS * 52) + 19)
DWC_USB3_PMUCFG_W	(DWC_USB3_EN_PMU_DEBUG ? (DWC_USB3_PMUCFG_U3_RFILE + DWC_USB3_PMUCFG_U3_RFILE_W) : (DWC_USB3_PMUCFG_U3_PWRPRESENT_VALID + DWC_USB3_PMUCFG_U3_PWRPRESENT_VALID_W))
DWC_USB3_RAM0_A	DWC_USB3_LOG2(DWC_USB3_RAM0_DEPTH)

Parameter Name	Equals To
DWC_USB3_RAM0_DEPTH_DEFAULT	(DWC_USB3_MODE ==3 ? (((DWC_USB3_HUB_NUM_UPSTRMRX_DATAPKTS * DWC_USB3 MPS + 276) >> 2) + DWC_USB3_HUB_DESC_DEPTH) : DWC_USB3_MBUS_TYPE ==4 ? 776 : DWC_USB3_MODE ==2 ? DWC_USB3_MAX_CB(DWC_USB3_RAM0_DEPTH_DEV, DWC_USB3_RAM0_DEPTH_HOST) : DWC_USB3_MODE ==1 ? DWC_USB3_RAM0_DEPTH_HOST: DWC_USB3_RAM0_DEPTH_DEV)
DWC_USB3_RAM0_DEPTH_DEV	(DWC_USB3_DCACHE_DEPTH_DEV + (DWC_USB3_NUM_RAMS<3 ? DWC_USB3_DEV_RXF_DEPTH: 0) + (DWC_USB3_NUM_RAMS<2 ? DWC_USB3_DEV_TXF_ALL_DEPTH: 0))
DWC_USB3_RAM0_DEPTH_HOST	(DWC_USB3_DCACHE_DEPTH_HST + (DWC_USB3_NUM_RAMS<3 ? DWC_USB3_HOST_RXF_ALL_DEPTH: 0) + (DWC_USB3_NUM_RAMS<2 ? DWC_USB3_HOST_TXF_ALL_DEPTH: 0))
DWC_USB3_RAM0_DWORD_ENABLE_EN	0
DWC_USB3_RAM0_P1_WE_W	((DWC_USB3_RAM0_DWORD_ENABLE_EN ==0) ? 1 : ((DWC_USB3_SPRAM_TYP ==1) ? DWC_USB3_MDWORDS : 1))
DWC_USB3_RAM1_DEPTH_DEFAULT	(DWC_USB3_NUM_RAMS ==1 ? 0: (DWC_USB3_MODE ==3 ? ((DWC_USB3_HUB_NUM_UPSTRMTX_DATAPKTS * DWC_USB3 MPS + 276) >> 2) : (DWC_USB3_MBUS_TYPE ==4) ? DWC_USB3_NPI_TX_FIFO_DEPTH : DWC_USB3_TXF_ALL_DEPTH))
DWC_USB3_RAM2_DEPTH_DEFAULT	(DWC_USB3_NUM_RAMS<=2 ? 0: (DWC_USB3_MODE ==3 ? 300 : DWC_USB3_RXF_ALL_DEPTH))
DWC_USB3_RCSR_DBC_DEPTH	(DWC_USB3_EN_DBC ? 32 : 0)
DWC_USB3_RCSR_DEVICE_DEPTH	(16 * DWC_USB3_NUM_EPS)
DWC_USB3_RCSR_GLOBAL_DEPTH	(16 * DWC_USB3_DEVICE_NUM_INT)
DWC_USB3_RCSR_HOST_DEPTH	(12*4 + 32*DWC_USB3_HOST_NUM_INTERRUPTER_SUPT + 4 *DWC_USB3_NUM_DEVICE_SUPT + DWC_USB3_RCSR_DBC_DEPTH)
DWC_USB3_RDWIDTH	((DWC_USB3_EN_ECC == 0) ? DWC_USB3_MDWIDTH : (DWC_USB3_MDWIDTH + DWC_USB3_ECC_CHK_WIDTH))

Parameter Name	Equals To
DWC_USB3_REF_CLK_SHUTDOWN	0
DWC_USB3_REQINFOWIDTH	4
DWC_USB3_RIQ_FIFO_ALL_DEPTH	(DWC_USB3_MODE ==0 ? DWC_USB3_NUM_RXFIFO * DWC_USB3_RIQ_FIFO_DEPTH: DWC_USB3_NUM_FSL_S_USB_INSTANCES * DWC_USB3_HOST_RIQ_DEPTH_PER_FSL_S_INSTANCE + DWC_USB3_NUM_HS_USB_INSTANCES * DWC_USB3_HOST_RIQ_DEPTH_PER_HS_INSTANCE + DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * DWC_USB3_HOST_RIQ_DEPTH_PER_SS_INSTANCE + DWC_USB3_EN_DBC_SRCSNK * DWC_USB3_RIQ_FIFO_DEPTH)
DWC_USB3_RIQ_FIFO_DEPTH	(DWC_USB3_RXQ_FIFO_DEPTH + (DWC_USB3_PSQ_FIFO_DEPTH/2) + 2)
DWC_USB3_RM_OPT_FEATURES	= (DWC_USB3_MBUS_TYPE ==4) ? 1 : 0
DWC_USB3_ROM_A	DWC_USB3_LOG2(DWC_USB3_HUB_DESC_DEPTH)
DWC_USB3_RXDMA_CMD_DEPTH	(DWC_USB3_MDWIDTH ==32 ? 2: 1)
DWC_USB3_RXDMA_CMD_MSEG_FIFO_DEPTH	(DWC_USB3_MDWIDTH ==32 ? 2: 1)
DWC_USB3_RXQ_FIFO_ALL_DEPTH	(DWC_USB3_MODE ==0 ? DWC_USB3_NUM_RXFIFO * DWC_USB3_RXQ_FIFO_DEPTH: DWC_USB3_NUM_FSL_S_USB_INSTANCES * DWC_USB3_HOST_RXQ_DEPTH_PER_FSL_S_INSTANCE + DWC_USB3_NUM_HS_USB_INSTANCES * DWC_USB3_HOST_RXQ_DEPTH_PER_HS_INSTANCE + DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * DWC_USB3_HOST_RXQ_DEPTH_PER_SS_INSTANCE + DWC_USB3_EN_DBC_SRCSNK * DWC_USB3_RXQ_FIFO_DEPTH)
DWC_USB3_RXQ_FIFO_DEPTH	((DWC_USB3_DEV_EN_SCATTER_PACKETS_OF_8_TO_15_TRBS ==1) ? ((DWC_USB3_MDWIDTH ==32) ? 32: 16) : ((DWC_USB3_MDWIDTH ==32) ? 16: 8))
DWC_USB3_RX_SS_PRI_CNT	(DWC_USB3_EN_USB2_ONLY ==1 ? 0: ((8 * DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC < DWC_USB3_MAX_ACTIVE_U2_INSTANCES) ? 1: 8 * DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC / DWC_USB3_MAX_ACTIVE_U2_INSTANCES))
DWC_USB3_SDLOG	= (DWC_USB3_SDWIDTH/8)
DWC_USB3_SSPHY_INTERFACE	= (DWC_USB3_EN_USB2_ONLY ==1) ? 0 : 1
DWC_USB3_TXDMA_CMD_DEPTH	(DWC_USB3_MDWIDTH ==32 ? 3: DWC_USB3_MDWIDTH ==64 ? 2: 1)

Parameter Name	Equals To
DWC_USB3_TXDMA_CMD_MSEG_FIFO_DEPTH	(DWC_USB3_MDWIDTH ==32 ? 2: 1)
DWC_USB3_TXQ_FIFO_ALL_DEPTH	(DWC_USB3_MODE ==0 ? DWC_USB3_TXQ_FIFO_DEVICE_DEPTH: DWC_USB3_MODE ==1 ? DWC_USB3_TXQ_FIFO_HOST_DEPTH: DWC_USB3_TXQ_FIFO_DRD_DEPTH)
DWC_USB3_TXQ_FIFO_DEPTH	((DWC_USB3_DEV_EN_SCATTER_PACKETS_OF_8_TO_15_TRBS ==1) ? ((DWC_USB3_MDWIDTH ==32) ? 32: 16) : ((DWC_USB3_MDWIDTH ==32) ? 16: 8))
DWC_USB3_TXQ_FIFO_DEVICE_DEPTH	(DWC_USB3_NUM_IN_EPS * DWC_USB3_TXQ_FIFO_DEPTH)
DWC_USB3_TXQ_FIFO_DRD_DEPTH	(DWC_USB3_TXQ_FIFO_HOST_DEPTH + ((DWC_USB3_NUM_IN_EPS > DWC_USB3_NUM_TXFIFO_HOST_ACTIVE) ? (DWC_USB3_NUM_IN_EPS - DWC_USB3_NUM_TXFIFO_HOST_ACTIVE) * DWC_USB3_TXQ_FIFO_DEPTH: 0))
DWC_USB3_TXQ_FIFO_HOST_DEPTH	(DWC_USB3_NUM_FSL_S_USB_INSTANCES * DWC_USB3_HOST_TXQ_DEPTH_PER_FSL_S_INSTANCE + DWC_USB3_NUM_HS_USB_INSTANCES * DWC_USB3_HOST_TXQ_DEPTH_PER_HS_INSTANCE + DWC_USB3_NUM_SS_USB_INSTANCES_ACTIVE * DWC_USB3_HOST_TXQ_DEPTH_PER_SS_INSTANCE + DWC_USB3_EN_DBC * DWC_USB3_TXQ_FIFO_DEPTH + DWC_USB3_EN_DBC_SRCSNK * DWC_USB3_TXQ_FIFO_DEPTH)
DWC_USB3_TX_SS_PRI_CNT	(DWC_USB3_EN_USB2_ONLY ==1 ? 0: ((4 * DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC < DWC_USB3_MAX_ACTIVE_U2_INSTANCES) ? 1: 4 * DWC_USB3_NUM_SS_USB_INSTANCES_P_DBC / DWC_USB3_MAX_ACTIVE_U2_INSTANCES))
GFLADJ_REF_CLK_LPM_SEL	23
P0	4'b0000
P1	4'b0001
P2	4'b0010
P3	4'b0011
P3CPM	4'b0100
P4	4'b1100
SS_DIS	4'h4

Parameter Name	Equals To
U0	4'h0
U1	4'h1
U2	4'h2
U3	4'h3