



DesignWare Cores Ethernet MAC Universal Databook

DWC Ether MAC 10/100/1000 Universal
DWC Ether MAC 10/100 Universal

Copyright Notice and Proprietary Information

Copyright © 2009 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

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.

Registered Trademarks (®)

Synopsys, AMPS, Cadabra, CATS, CRITIC, CSim, Design Compiler, DesignPower, DesignWare, EPIC, Formality, HSIM, HSPICE, iN-Phase, in-Sync, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Photolynx, Physical Compiler, PrimeTime, SiVL, SNUG, SolvNet, System Compiler, TetraMAX, VCS, and Vera are registered trademarks of Synopsys, Inc.

Trademarks (™)

Active Parasitics, AFGen, Apollo, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanTestchip, AvanWaves, BOA, BRT, ChipPlanner, Circuit Analysis, Columbia, Columbia-CE, Comet 3D, Cosmos, CosmosEnterprise, CosmosLE, CosmosScope, CosmosSE, Cyclelink, DC Expert, DC Professional, DC Ultra, Design Advisor, Design Analyzer, Design Vision, DesignerHDL, DesignTime, Direct RTL, Direct Silicon Access, Discovery, Dynamic-Macromodeling, Dynamic Model Switcher, EDAnavigator, Encore, Encore PQ, Evaccess, ExpressModel, Formal Model Checker, FoundryModel, Frame Compiler, Galaxy, Gatran, HANEX, HDL Advisor, HDL Compiler, Hercules, Hercules-II, Hierarchical Optimization Technology, High Performance Option, HotPlace, HSIM^{plus}, HSPICE-Link, iN-Tandem, Integrator, Interactive Waveform Viewer, i-Virtual Stepper, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, JVXtreme, Liberty, Libra-Passport, Library Compiler, Libra-Visa, Magellan, Mars, Mars-Rail, Mars-Xtalk, Medici, Metacapture, Milkyway, ModelSource, Module Compiler, Nova-ExploreRTL, Nova-Trans, Nova-VeriLint, Orion_ec, Parasitic View, Passport, Planet, Planet-PL, Planet-RTL, Polaris, Power Compiler, PowerCODE, PowerGate, ProFPGA, ProGen, Prospector, Raphael, Raphael-NES, Saturn, ScanBand, Schematic Compiler, Scirocco, Scirocco-i, Shadow Debugger, Silicon Blueprint, Silicon Early Access, SinglePass-SoC, Smart Extraction, SmartLicense, Softwire, Source-Level Design, Star-RCXT, Star-SimXT, Taurus, TimeSlice, TimeTracker, Timing Annotator, TopoPlace, TopoRoute, Trace-On-Demand, True-Hspice, TSUPREM-4, TymeWare, VCS Express, VCSI, Verification Portal, VFormal, VHDL Compiler, VHDL System Simulator, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.

ARM and AMBA are registered trademarks of ARM Limited.

Saber is a registered trademark of SabreMark Limited Partnership and is used under license.

PCI Express is a trademark of PCI-SIG.

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

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

Contents

Revision History	11
Preface	15
Databook Organization	15
Reference Documentation	16
Authority	17
Customer Support	18
Chapter 1	
Introduction	21
1.1 GMAC-UNIV Overview	21
1.2 System Overview	22
1.2.1 Interfaces	23
1.2.2 Transmit and Receive FIFOs	23
1.3 GMAC-UNIV Features	23
1.3.1 GMAC Core Features	24
1.3.2 Transaction Layer (MTL) Features	25
1.3.3 DMA Block Features	26
1.3.4 AMBA AHB Interface Features	27
1.3.5 AMBA AXI Interface Features	27
1.3.6 Audio Video Support Features	28
1.3.7 Monitoring, Testing, and Debugging Support Features	29
1.4 GMAC-UNIV Configuration	29
1.4.1 Configurable Features	29
1.5 Deliverables	30
1.5.1 Licensing Features	31
Chapter 2	
Building and Verifying Your Core	33
2.1 Setup Requirements	33
2.2 Basic Design Flow	33
2.3 Creating a Workspace	34
2.4 Configuration: Creating the RTL	36
2.5 Synthesis: Creating the Gate-Level Netlist	36
2.6 Simulation: Verifying DWC Ether MAC 10/100/1000 Universal	37
2.6.1 Generating a GTECH Model	37
2.6.2 Verifying DWC Ether MAC 10/100/1000 Universal	37
Chapter 3	
Architecture	41
3.1 Introduction	41

3.2 AHB Application Host Interface	42
3.2.1 AHB Master Interface	42
3.2.2 AHB Slave Interface	46
3.3 AXI Application Host Interface	47
3.3.1 AXI Master Interface	47
3.3.2 AXI Slave Interface	52
3.3.3 AXI Low Power Interface	54
3.4 DMA Controller	55
3.4.1 Initialization	56
3.4.2 Transmission	58
3.4.3 Reception	63
3.4.4 Interrupts	67
3.4.5 Error Response to DMA	67
3.4.6 DMA Native Interface	67
3.5 MAC Transaction Layer (MTL)	71
3.5.1 Transmit Path	72
3.5.2 Receive Path	79
3.5.3 Interfacing With External Two-Port RAMs	86
3.6 GMAC Core	89
3.6.1 Transmission	89
3.6.2 MAC Transmit Interface Protocol	93
3.6.3 MAC Transmit Timing	96
3.6.4 Reception	99
3.6.5 MAC Receive Interface Protocol	105
3.6.6 MAC Receive Timing	109
3.6.7 MAC Control Interface Protocol	112

Chapter 4

Optional Modules	115
4.1 IEEE 1588-2002 Timestamps	115
4.1.1 Reference Timing Source	117
4.1.2 System Time Register Module	117
4.1.3 Transmit Path Functions	119
4.1.4 Receive Path Functions	120
4.1.5 Timestamp Error Margin	120
4.1.6 Frequency Range of Reference Timing Clock	120
4.2 IEEE 1588-2008 Advanced Timestamps	121
4.2.1 Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support	121
4.2.2 Clock Types	122
4.2.3 PTP Processing and Control	124
4.2.4 Reference Timing Source	128
4.2.5 Transmit Path Functions	129
4.2.6 Receive Path Functions	129
4.2.7 Auxiliary Snapshot	130
4.3 Audio/Video Bridging (AVB)	130
4.3.1 Block Diagram	131
4.3.2 Transmit Path Functions	132
4.3.3 Receive Path Functions	132
4.3.4 DMA Arbiter Functions	134
4.3.5 Slot Number Function	136

4.3.6 Interrupt Functions	136
4.3.7 Credit-Based Shaper Algorithm Functions	136
4.4 Energy Efficient Ethernet	138
4.4.1 Transmit Path Functions	138
4.4.2 Receive Path Functions	139
4.4.3 LPI Timers	140
4.5 Checksum Offload Engine	141
4.5.1 Transmit Checksum Offload Engine	141
4.5.2 Receive Checksum Offload Engine	143
4.6 MAC Management Counters	144
4.6.1 Address Assignments	144
4.6.2 MMC Register Description	150
4.7 Power Management Block	160
4.7.1 PMT Block Description	160
4.7.2 Remote Wake-Up Frame Detection	162
4.7.3 Magic Packet Detection	163
4.7.4 System Considerations During Power Down	164
4.8 Station Management Agent	164
4.8.1 Functions	165
4.8.2 GMII/MII Management Write Operation	166
4.8.3 GMII/MII Management Read Operation	167
4.9 Physical Coding Sublayer	167
4.9.1 PCS Functions	168
4.10 Reduced Media Independent Interface	169
4.10.1 Block Diagram	170
4.10.2 Block Overview	170
4.10.3 Transmit Bit Ordering	171
4.10.4 RMII Transmit Timing Diagrams	172
4.11 Reverse Media Independent Interface	174
4.11.1 Block Diagram	175
4.11.2 Block Overview	175
4.11.3 RevMII Interrupts	179
4.11.4 RevMII Registers	179
4.12 Serial Media Independent Interface	185
4.12.1 Block Diagram	186
4.12.2 Block Overview	186
4.12.3 SMII Timing	187
4.13 Reduced Gigabit Media Independent Interface	189
4.13.1 Block Diagram	189
4.13.2 Block Overview	190
4.13.3 RGMII Clocks	191
4.13.4 Signal Conversion	191
4.13.5 RGMII Transmit Timing	194
4.13.6 RGMII Receive Timing	194
4.14 Reduced Ten-Bit Interface	194
4.14.1 Block Diagram	195
4.14.2 Block Diagram Overview	195
4.14.3 RTBI Transmit Timing	195
4.15 Serial Gigabit Media Independent Interface	195
4.15.1 Block Diagram	196

4.15.2 Block Overview	196
4.15.3 Block Descriptions	197
4.16 Multiple PHY Interfaces	201
4.17 Interrupts From the GMAC Core	202
Chapter 5	
Signals	205
5.1 Top-Level I/O Diagram	205
5.2 Signal Descriptions	212
5.2.1 System Clock and Reset Signals (Default)	212
5.2.2 PHY Interface Signals (Default)	213
5.2.3 MAC Tx Interface Signals	219
5.2.4 MAC Rx Interface Signals	221
5.2.5 MAC Control Interface Signals	223
5.2.6 Application Clock and Reset Interface Signals	224
5.2.7 Application Transmit Interface (ATI) Signals	225
5.2.8 Application Receive Interface (ARI) Signals	228
5.2.9 MDC Interface Signals	230
5.2.10 AHB Master Interface Signals	233
5.2.11 AHB Slave Interface Signals	236
5.2.12 AXI Master Interface Signals	239
5.2.13 AXI Low Power Interface Signals	246
5.2.14 AXI Slave Interface Signals	247
5.2.15 External DPRAM Interface Signals	253
5.2.16 External DPRAM Interface Signals for Channel 1/Channel 2	257
5.2.17 External DPRAM (Frame Length FIFO) Interface Signals	260
5.2.18 RGMII/RTBI Interface Signals (Optional)	262
5.2.19 RMII Interface Signals (Optional)	263
5.2.20 RevMII Interface Signal (Optional)	264
5.2.21 SMII Interface Signal (Optional)	267
5.2.22 PMT Interrupt Signal (Optional)	267
5.2.23 SGMII/TBI/RTBI Interface Signals (Optional)	267
5.2.24 SGMII Interface Signals (Optional)	269
5.2.25 TBI Interface Signals	270
5.2.26 SMA Interface Signals (Optional)	271
5.2.27 APB Interface Signals (Optional)	272
5.2.28 Sideband Signals (Optional)	273
5.2.29 IEEE 1588 Timestamp Signals (Optional)	276
5.2.30 Energy Efficient Ethernet Signal	278
5.2.31 Test Mode	278
Chapter 6	
Registers	281
6.1 Register Maps	281
6.1.1 DMA Register Map	281
6.1.2 GMAC Register Map	287
6.2 Register Descriptions	294
6.2.1 DMA Register Description	295
6.2.2 GMAC Register Description	321
6.2.3 IEEE 1588 Timestamp Registers	345

Chapter 7	
Parameters	355
7.1 Features	355
7.1.1 System Interface	356
7.1.2 General Features	360
7.1.3 PHY Line Interfaces (RGMII, RMII, SMII, SGMII, TBI, RTBI, and RevMII)	363
7.2 Optional Modules	366
7.2.1 Energy Efficient Ethernet (EEE)	366
7.2.2 AV Feature Support	366
7.2.3 IEEE 1588 Timestamp Block	367
7.2.4 Power Management Block	368
7.2.5 IP Checksum Block	369
7.2.6 GMAC Management (RMON) Counters	369
7.2.7 Station Management Block	395
7.3 Low Power Support	395
Chapter 8	
Descriptors	397
8.1 Normal Descriptor Formats	397
8.1.1 Receive Descriptor	401
8.1.2 Transmit Descriptor	406
8.1.3 Descriptor Format With IEEE 1588 Timestamps Enabled	410
8.2 Alternate or Enhanced Descriptors	414
8.2.1 Transmit Descriptor	414
8.2.2 Receive Descriptor	420
Chapter 9	
Implementation Guidelines	429
9.1 Clocks With GMII/MII	429
9.2 Clocks With TBI	431
9.3 Clocks With SGMII	431
9.4 Clocks With RMII	433
9.5 Clocks With RevMII	434
9.6 Clocks With SMII	436
9.6.1 Non Source Synchronous Mode	436
9.6.2 Source Synchronous Mode	437
9.7 Clocks With RGMII	438
9.8 Clocks With RTBI	439
9.9 Host (System Interface) Clock	440
9.10 Resets	441
9.11 SMA to PHY Interface	441
9.12 Timing Guidelines for Dual Data Rate RGMII/RTBI Outputs	442
9.13 Synthesis Guidelines	448
Chapter 10	
Unified Power Format Support	451
10.1 Introduction	451
10.1.1 Advantages of UPF	452
10.2 Block Diagram	453
10.3 UPF Implementation	453

10.3.1 Isolation Scheme	454
10.3.2 Address Filtering	454
10.3.3 Power-Down and Power-Up Sequence	455
10.3.4 Clocks	456
10.4 UPF Flow and Methodology	457
10.4.1 Simulation	458
10.4.2 Synthesis	460
10.5 Area and Power	461
10.6 UPF Flow Files	462
10.6.1 DWC_gmac_syn.upf	462
10.6.2 DWC_gmac_top_power_domains.rpt	464
10.6.3 DWC_gmac_top_isolation_cell.rpt	465
10.6.4 DWC_gmac_top_power_switch.rpt	465
10.6.5 mvcmp.log	466
10.6.6 mvsim.log	466
Appendix A	
Workspace Directory and Files	469
A.1 Workspace File Directory Structure	469
A.1.1 Directory Structures of Files (For Multiple Configurations)	469
A.1.2 Directory Structures for the Simulation Environment	471
A.2 Workspace Directory and File Descriptions	475
Appendix B	
GMAC-AHB Verification Environment Using AMBA and Ethernet VIPs	479
B.1 VTB Overview	479
B.2 Verification Flow	481
B.2.1 Transmission Verification Flow	481
B.2.2 Receive Verification Flow	485
B.3 Directory Structure	489
B.4 GPIO	490
B.4.1 GPIO_OUT (Tx)	490
B.4.2 GPIO_IN(Tx)	490
B.4.3 GPIO_OUT (Rx)	491
B.4.4 GPIO_IN (Rx)	492
B.5 SoC-Level Verification Guidelines	492
B.6 Running Simulations	493
B.7 Simulation PASS/FAIL	493
Appendix C	
GMAC-AHB Verification Environment	495
C.1 VTB Overview	495
C.2 Initialization	496
C.3 Using VTB	497
C.4 Tool and Setup Requirements	499
C.5 Running Simulations	499
C.6 Simulation PASS/FAIL	500
Appendix D	
GMAC-DMA Verification Environment	501
D.1 Testbench Overview	501

D.2 Test Flow	502
D.2.1 Frame Receive Flow	503
D.2.2 Frame Transmit Flow	503
D.3 Directory Structure	503
D.4 Running Simulations	505
D.5 Simulation PASS/FAIL	506
Appendix E	
GMAC-MTL Verification Environment	507
E.1 Testbench Overview	507
E.2 Test Flow	509
E.3 Directory Structure	509
E.4 Running Simulations	511
E.5 Simulation PASS/FAIL	512
Appendix F	
GMAC-Core Verification Environment	513
F.1 Testbench Overview	513
F.2 Test Flow	514
F.3 Directory Structure	515
F.4 Running Simulations	517
F.5 Simulation PASS/FAIL	518
Appendix G	
GMAC-AXI Verification Environment	521
G.1 VTB Overview	521
G.2 Verification Flow	523
G.2.1 Transmission Verification Flow	524
G.2.2 Receive Verification Flow	527
G.2.3 Transmit-Receive Verification Flow	531
G.3 Directory Structure	536
G.4 GPIO	537
G.4.1 GPIO_OUT (Tx)	537
G.4.2 GPIO_IN (Tx)	537
G.4.3 GPIO_OUT (Rx)	538
G.4.4 GPIO_IN (Rx)	538
G.5 SoC-Level Verification Guidelines	538
G.6 Running Simulations	539
G.7 Simulation PASS/FAIL	540
Appendix H	
Area and Power	541
H.1 Area	541
H.1.1 Gate Count Summary	541
H.1.2 Area Differences Due to Scan Ready and Clock Gating	543
H.2 Power	544
Appendix I	
Programming Guide	547
I.1 Initializing DMA	547
I.2 Initializing MAC	548

I.3 Performing Normal Receive and Transmit Operation	549
I.4 Stopping and Starting Transmission	550
I.5 Programming Guidelines for GMII Link Transitions	550
I.5.1 Transmit and Receive Clocks are Running when the Link is Down	550
I.5.2 Transmit and Receive Clocks are Stopped when the Link is Down	550
I.6 Programming Guidelines for IEEE 1588 Time Stamping	551
I.6.1 Initialization Guideline for System Time Generation	551
I.6.2 System Time Correction	551
I.7 Programming Guidelines for AV Feature	552
I.7.1 Initializing the DMA	552
I.7.2 Enabling Slot Number Checking	553
I.7.3 Enabling Average Bits Per Slot Reporting	553
I.8 Programming Guidelines for Energy Efficient Ethernet	554
 Appendix J	
Synchronizer Methods	555
J.1 Synchronizer 1: Simple Double Register Synchronizer (DWC_gmac_bcm21.v)	555
J.2 Synchronizer 2: Pulse Synchronizer (DWC_gmac_bcm22.v)	559
J.3 Synchronizer 3: Pulse Synchronizer with Acknowledge (DWC_gmac_bcm23.v)	559
J.4 Synchronizer 4: Gray Coded Synchronizer (DWC_gmac_bcm24.v)	561
 Appendix K	
Endian Support	563
K.1 Introduction	563
K.1.1 Little-Endian Mode	563
K.1.2 Big-Endian Mode	563
K.1.3 Byte-Invariant Big-Endian Mode	563
K.2 32-bit Data Bus Transfers	564
K.2.1 Master Interface	564
K.2.2 Slave Interface	565
K.3 64-bit Data Bus Transfers	566
K.3.1 Master Interface	567
K.3.2 Slave Interface	567
K.4 128-bit Data Bus Transfers	570
K.4.1 Master Interface	570
K.4.2 Slave Interface	571

Revision History

Date	Version	Description
October, 2009	3.60a	<ul style="list-style-type: none"> • Updated “GMAC-UNIV Features” on page 23 • Added “Audio Video Support Features” on page 28 • Added “Audio/Video Bridging (AVB)” on page 130 • Added “Energy Efficient Ethernet” on page 138 • Added “Reverse Media Independent Interface” on page 174 • Updated “Signals” on page 205 • Updated “Registers” on page 281 • Added “AV Feature Support” on page 366 • Added “Energy Efficient Ethernet (EEE)” on page 366 • Updated “Alternate or Enhanced Descriptors” on page 414 • Added “Unified Power Format Support” on page 451 • Updated “Area” on page 541 • Added “Programming Guidelines for AV Feature” on page 552 • Added “Programming Guidelines for Energy Efficient Ethernet” on page 554 • Added “Endian Support” on page 563 • Restructured the Architecture chapter into the following two chapters: Architecture and Optional Modules. • Fixed Documentation Errata
February, 2009	3.50a	<ul style="list-style-type: none"> • Added support for Serial Media Independent Interface (SMII) • Added support for AMBA Advanced eXtensible Interface Bus (AXI) Application Host Interface • Updated support for IEEE 1588 standard • Fixed documentation errata <ul style="list-style-type: none"> - Updated “Preface” on page 15 - Modified “Introduction” on page 21 - Added “Auxiliary Snapshot” on page 130 - Added “AXI Application Host Interface” on page 47 - Added “AXI Low Power Interface” on page 54 - Added “ARI with Advanced Time Stamping” on page 83 - Added “Serial Gigabit Media Independent Interface” on page 195 - Added GMAC-AXI (GMII) Top-Level I/O Diagram Figure 5-5 on page 210 and AXI Master and Slave signal descriptions (Table 5-13 on page 239 and Table 5-15 on page 247, respectively)

Date	Version	Description
(Continued)		<ul style="list-style-type: none"> - Added SMII signal to Figure 5-6 on page 211 and description to Table 5-22 on page 267 - Added AXI Bus Mode Register (modified “DMA Register Map” on page 281 and added “Register 10 (AXI Bus Mode Register)” on page 313) - Added SMII and AXI configuration parameters to “System Interface” on page 356 - Added SMII register 54 to “DMA Register Map” on page 281 - Added SMII register to “Register 54 (SGMII/RGMII/SMII Status Register)” on page 345 - Modified Timestamp register descriptions (“IEEE 1588 Timestamp Registers” on page 345) - Added SMII configuration parameters to “PHY Line Interfaces (RGMII, RMII, SMII, SGMII, TBI, RTBI, and RevMII)” on page 363 - Added IEEE 1588 Timestamp configuration parameters to Table 7-17 on page 367 - Modified Timestamp descriptor descriptions “Descriptor Format With IEEE 1588 Timestamps Enabled” on page 410 - Updated descriptor descriptions in “Alternate or Enhanced Descriptors” on page 414 - Added Implementation Guidelines for “Clocks With SMII” on page 436 - Added Implementation Guidelines for “Host (System Interface) Clock” on page 440 - Added Implementation Guidelines for “Synthesis Guidelines” on page 448 - Updated “Workspace Directory and Files” on page 469 - Added “GMAC-AXI Verification Environment” on page 521 - Added “Programming Guide” on page 547 - Added “Synchronizer Methods” on page 555
June 30, 2008	3.42a	Errata correction
February 7, 2008	3.41a	Errata correction
November 21, 2007	3.40a	<ul style="list-style-type: none"> • Added IEEE 1588 support • Added SolvNet article citations • Added Appendix B, “GMAC-AHB Verification Environment Using AMBA and Ethernet VIPs”
March 22, 2007	3.30a	<ul style="list-style-type: none"> • Added alternative descriptor structure • Added optional Transmit and Receive Checksum Offload engines and related RMON counters and interrupts registers • SMA-to-PHY interface updated and documented • Updated databook for minor new features • Fixed documentation errata • Updated gate count, area in “Area and Power” on page 541
August 2006	3.20a	<ul style="list-style-type: none"> • Fixed documentation errata • Edited selected text • Revised documentation of database file structures and user interface elements • Updated gate count, area in “Area and Power” on page 541

Date	Version	Description
October 2005	3.10a	<ul style="list-style-type: none"> • Fixed documentation errata • Expanded architecture interface descriptions, adding timing diagrams to <ul style="list-style-type: none"> - “AHB Application Host Interface” on page 42 - “DMA Controller” on page 55 - “MAC Transaction Layer (MTL)” on page 71 - “GMAC Core” on page 89 • Added GMAC-DMA (GMII subsystem top-level I/O diagram (Figure 5-3) to “Top-Level I/O Diagram” on page 205 and modified “Signal Descriptions” on page 212 accordingly (added Table 5-10 and Table 5-18) • Added Appendixes B through F to describe the verification environments for the GMAC AHB, DMA, MTL, and Core interfaces.
November 2005	3.10a	<ul style="list-style-type: none"> • Added Revision History • Fixed documentation errata • Added notes for 10/100 Universal product applicability and operation
June, 2005	3.0	First release

Preface

This document describes Synopsys's DesignWare Cores Ethernet MAC Universal, release 3.60a. This product enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3 specification. Ethernet MAC Universal (or GMAC-UNIV for short) is provided as two separate licenses, as follows:

- ❖ DWC Ether MAC 10/100/1000 Universal is the Media Access Controller for 10/100/1000 Ethernet.
- ❖ DWC Ether MAC 10/100 Universal is the Media Access Controller for 10/100 Ethernet.

Ethernet MAC Universal corresponds to either DWC Ether MAC 10/100/1000 Universal or DWC Ether MAC 10/100 Universal in the SolvNet database.

Databook Organization

The chapters of this databook are organized as follows:

[Chapter 1, "Introduction,"](#) describes GMAC-UNIV features and provides an overview of the architecture.

[Chapter 2, "Building and Verifying Your Core,"](#) provides an overview of setting up, configuring, and verifying the core.

[Chapter 3, "Architecture,"](#) describes the GMAC-UNIV architecture and its main blocks.

[Chapter 4, "Optional Modules,"](#) describes the optional modules and features of GMAC-UNIV.

[Chapter 5, "Signals,"](#) describes the top-level signals of GMAC-UNIV.

[Chapter 6, "Registers,"](#) maps and describes the function of the GMAC-UNIV's control and status registers.

[Chapter 7, "Parameters,"](#) describes the GMAC-UNIV configuration options and parameters.

[Chapter 8, "Descriptors,"](#) identifies and describes the GMAC UNIV descriptors.

[Chapter 9, "Implementation Guidelines,"](#) provides information about clocks and resets, scan, synthesis, and layout.

[Chapter 10, "Unified Power Format Support,"](#) provides information about how GMAC-UNIV supports the IEEE 1801 standard.

[Appendix A, "Workspace Directory and Files,"](#) describes the database file structure of GMAC-UNIV.

[Appendix B, "GMAC-AHB Verification Environment Using AMBA and Ethernet VIPs"](#) describes effective use of AMBA and Ethernet Verification IP for verifying the GMAC-AHB.

[Appendix C, "GMAC-AHB Verification Environment,"](#) describes the Verilog Testbench (VTB).

[Appendix D, "GMAC-DMA Verification Environment,"](#) describes the example verification environment for the GMAC-DMA configuration.

[Appendix E, "GMAC-MTL Verification Environment,"](#) describes the example verification environment for the GMAC-MTL configuration.

Appendix F, “GMAC-Core Verification Environment,” describes the example verification environment for the GMAC-CORE configuration.

Appendix G, “GMAC-AXI Verification Environment ,” describes effective use of AMBA and Ethernet Verification IP for verifying the GMAC-AXI.

Appendix H, “Area and Power,” summarizes area in terms of gate count, area differences due to scan ready and clock gating, and power dissipation estimates with and without clock gating.

Appendix I, “Programming Guide ,” provides instructions for initializing the DMA/MAC registers in the proper sequence.

Appendix J, “Synchronizer Methods,” documents the synchronizer methods (blocks of synchronizer functionality) used in DesignWare Ethernet Mac Universal IP to cross clock boundaries.

Appendix K, “Endian Support,” describes the endian support for 32-bit, 64-bit, and 128-bit data bus.

Reference Documentation

The following references contain useful information concerning the protocols addressed by this IP core:

- ❖ *Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, Standard 802.3-2005, Institute of Electrical and Electronics Engineers (IEEE)
- ❖ *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Standard 1588-2008, IEEE
- ❖ *Standards for Audio Video (AV) traffic*, Standard 802.1-AS, version D6.0 and Standard 802.1-Qav, version D6.0, IEEE
- ❖ *Standard for Energy Efficient Ethernet (EEE)*, Standard 802.3-az, version D2.0, IEEE
- ❖ *AMBA Specification*, Revision 2.0, ARM Ltd. for AHB, APB interface
- ❖ *AMBA 3 Specification*, ARM Ltd for AXI interface
- ❖ *Reduced Gigabit Media Independent Interface (RGMII)*, Version 2.0, Broadcom Corp., Hewlett-Packard Co., Marvell Technology Group, Ltd.
- ❖ *Serial-GMII (SGMII) Specification*, Revision 1.8, Cisco Systems
- ❖ *Serial-MII (SMII) Specification*, Revision 2.1, Cisco Systems
- ❖ *RMII Specification*, Revision 1.2, RMII Consortium
- ❖ *Reverse Media Independent Interface (RevMII) Block Architecture*, Dmitriy Gusev
- ❖ *RFC 768, User Datagram Protocol (UDP)*, DARPA Internet Program
- ❖ *RFC 791, Protocol Specification* (Internet Protocol, Version 4 (IPv4) Specification), DARPA Internet Program
- ❖ *RFC 792, Internet Control Message Protocol Specification*, DARPA Internet Program
- ❖ *RFC 793, Transmission Control Protocol (TCP)*, DARPA Internet Program
- ❖ *RFC 1071, Computing the Internet Checksum (memo)*, Network Working Group
- ❖ *RFC 2460, Internet Protocol, Version 6 (IPv6) Specification*, The Internet Society, Network Working Group
- ❖ *RFC 2819, Remote Network Monitoring Management Information Base*, The Internet Society, Network Working Group
- ❖ *RFC 2965, HTTP State Management Mechanism*, The Internet Society, Network Working Group

- ❖ [RFC 4443, Internet Control Message Protocol \(ICMPv6\) for the Internet Protocol Version 6 \(IPv6\) Specification](#), The Internet Society, Network Working Group
- ❖ 1588 Main Web Site, National Institute of Standards and Technology, <http://ieee1588.nist.gov/>
- ❖ [Hardware-Assisted IEEE 1588* Implementation in the Intel® IXP46X Product Line](#) (white paper), Intel, Inc., March 2005
- ❖ [A Frequency Compensated Clock for Precision Synchronization using IEEE 1588 Protocol and its Application to Ethernet](#), Sivaram Balasubramanian, Kendal R. Harris, and Anatoly Moldovansky, Rockwell Automation, November, 2003
- ❖ [Synopsys Low-Power Flow User Guide](#), Version C-2009.06, June 2009 (SolvNet ID required)
- ❖ [Synopsys Low Power Verification Tools Suite User Guide](#), Version 2009.06, June 2009 (SolvNet ID required)

Authority

This databook is cited in the following SolvNet articles:

SolvNet Article No.	Title
017017	“VCI/PVCI Interface for 3.x releases of the DWC Ethernet MAC 10/100/1000-Universal and DWC Ethernet 10/100 MAC - Universal DMA Subsystem”
017601	“FIFO size estimation for DWC Ether MAC 10100/1000 Universal IP”
017998	“DWC Ethernet MAC Universal: Behavior During a Transmit Underflow Event”
018112	“RMII, Transmit MII, and Receive MII Clock Balancing for GMAC-Universal 3.x IP”
019407	“DWC Ethernet MAC Universal: Updating GMAC Configuration Register in RGMII, SMII, or SGMII Configuration”
019417	“DWC Ethernet MAC Universal: Auto-Negotiation Support in PCS Mode”
019523	“DWC Ether MAC 10/100/1000 Mbps -Universal IP 3.x - Software Driver”
019825	“DesignWare Cores Ethernet MAC 10/100/1000 Mbps Universal Core: Setting and Clearing the Own Bit Functionality in Transmit and Receive Descriptors”
021418	“Hash filtering in the GMAC core”
023328	“Two consecutive writes to GMAC register”
023329	“BYTE addressability on GMAC”
023331	“MAC addresses in GMAC core”
023332	“Clock gating during power-down mode in GMAC core”
024530	“DWC GMAC 10/100/1000 Mbps: Ideal throughput with GMAC IP”
024534	“DWC GMAC 10/100/1000 Mbps: Type 1 and Type 2 Checksum Offload Engines”
025442	“PTP reference clock constraints in GMAC 10/100/1000”

SolvNet Article No.	Title
025589	"Jumbo frame size in DWC GMAC 10/100/1000 Mbps"
025590	"Error Status Summary in DWC GMAC 10/100/1000Mbps"
025592	"Condition for preamble size in the DWC GMAC 10/100/1000 Mbps Configurations"
025595	"Is the poll demand ignored in GMAC 10/100/1000 Mbps configuration?"
025598	"Late collision before 64bytes in GMAC 10/100Mbps?"
025600	"Does Timestamping Apply Only to MII/GMII in GMAC 10/100/1000 Mbps Configurations"
025602	"Speed control for MII in GMAC 10/100/1000Mbps"
026181	"Register contents in MAC 10/100/1000Mbps that cross clock domain."

Customer Support

For customer support:

- ❖ Enter a call through SolvNet.
 - ◆ Go to <https://solvnet.synopsys.com/ManageCase?ccf=1> and provide the requested information, including:
 - ✧ Product: DesignWare Cores
 - ✧ Sub Product: Select 10/100 Ethernet or Gigabit Ethernet.
 - ✧ Version: 3.60a
 - ✧ Subject: Ethernet MAC Universal
- ❖ Send an e-mail message to support_center@synopsys.com.
 - ◆ Include the Product name, Sub Product name, and Version (product release number) in your e-mail so it can be routed correctly.
 - ◆ Provide the following additional information, if applicable:
 - ✧ For environment setup problems, run the debug_info command in the coreConsultant GUI and include the text file generated.
 - ✧ For configuration failures, include the error messages that appear in the coreConsultant GUI console pane.
 - ✧ For simulation failures, include a text file with your specific configuration. Generate this text file using the coreConsultant GUI's write_batch_script command. Also include the log file from the <workspace>/simulation/ directory, the log file for the specific test, and a VPD/VCD waveform dump file.
 - ✧ For synthesis failures, include the log file from the <workspace>/syn/ directory.
- ❖ Telephone your local support center:
 - ◆ United States:
Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
 - ◆ Canada:
Call 1-650-584-4200 from 7 AM to 5:30 PM Pacific time, Monday through Friday.

- ◆ All other countries:

<http://www.synopsys.com/Support/GlobalSupportCenters/Pages/default.aspx>

Introduction

This chapter provides an overview of the DesignWare Cores Ethernet MAC Universal (GMAC-UNIV) and describes its components and features.

This chapter contains the following sections:

- ❖ “[GMAC-UNIV Overview](#)” on page [21](#)
- ❖ “[System Overview](#)” on page [22](#)
- ❖ “[GMAC-UNIV Features](#)” on page [23](#)
- ❖ “[GMAC-UNIV Configuration](#)” on page [29](#)
- ❖ “[Deliverables](#)” on page [30](#)

1.1 GMAC-UNIV Overview

The GMAC-UNIV enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2005 standard. The GMAC-UNIV is a configurable product that can be optimized for gate count and latency to meet the application requirements. You can use the GMAC-UNIV in number of applications such as switches and network interface cards.

The GMAC-UNIV can have the following five major configurations:

- ❖ GMAC core with native interface (GMAC-CORE)
- ❖ GMAC with transaction layer (GMAC-MTL)
- ❖ GMAC with native DMA (GMAC-DMA)
- ❖ GMAC with AHB-interfaced DMA (GMAC-AHB)
- ❖ GMAC with AXI-interfaced DMA (GMAC-AXI)

In addition to the default interfaces defined in the IEEE 802.3 specifications, GMAC-UNIV supports several industry standard interfaces to the PHY. GMAC-UNIV is compliant with the following standards:

- ❖ IEEE 802.3-2005 for Ethernet MAC, Gigabit Media Independent Interface (GMII)/Media Independent Interface (MII), Ten Bit Interface (TBI), and Reverse Media Independent Interface (RevMII).
- ❖ IEEE 1588-2008 standard for precision networked clock synchronization
- ❖ IEEE 802.1-AS, version D6.0 and 802.1-Qav, version D6.0 for Audio Video (AV) traffic
- ❖ IEEE 802.3-az, version D2.0 for Energy Efficient Ethernet (EEE)

- ❖ AMBA 2.0 for AHB Master/Slave ports
- ❖ AMBA 3.0 for AXI Master/Slave ports
- ❖ RGMII specification version 2.0 from HP/Marvell
- ❖ SGMII specification version 1.8 from Cisco/Marvell
- ❖ RMII specification version 1.2 from RMII consortium
- ❖ SMII specification version 2.1 from Cisco for SMII



The information in this databook is also applicable to DWC Ether MAC 10/100 Universal. The DWC Ether MAC 10/100 Universal users can ignore information specific to Gigabit (1000 Mbps) operations. This mainly includes the RGMII, TBI, RTBI, and SGMII PHY interfaces, and Gigabit functions such as frame bursting and carrier extension.

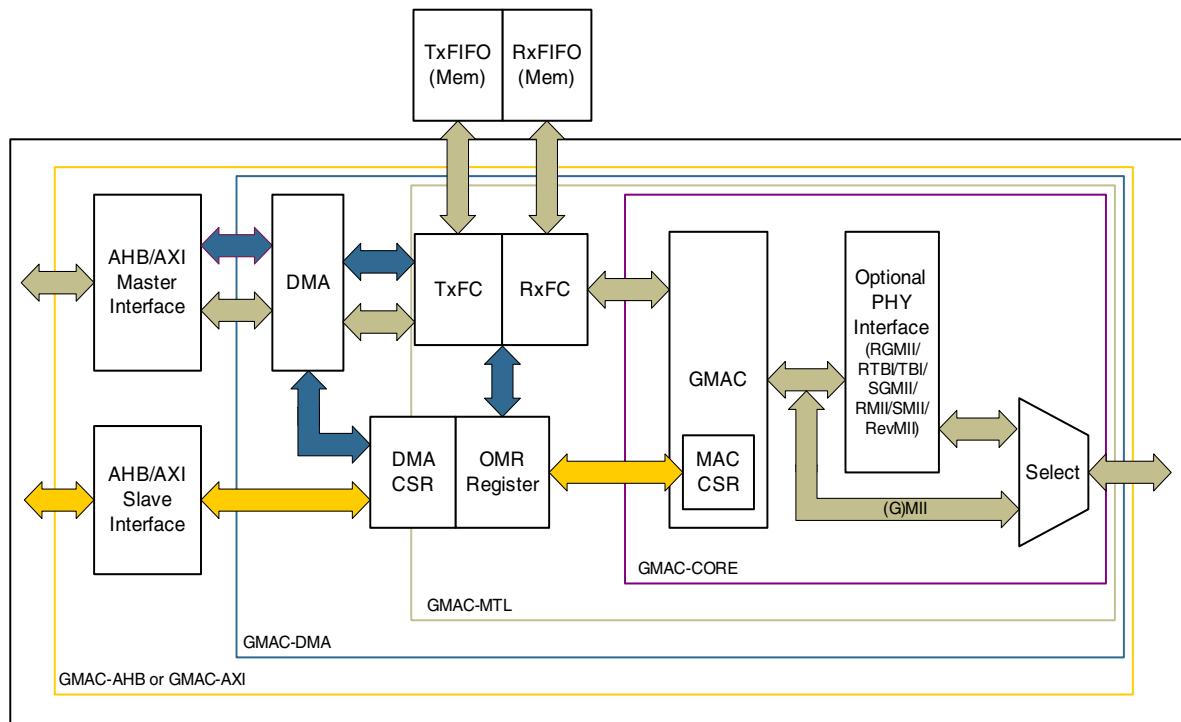
The DWC Ether MAC 10/100 Universal users can ignore the following sections related to Gigabit operations:

- Frame bursting and carrier extension description in [Sections 3.6.1.3](#) and [3.6.4.1](#)
- [Sections 4.9, 4.13, 4.14](#), and [4.15](#)
- [Sections 5.2.18, 5.2.23, 5.2.24](#), and [5.2.25](#)
- [Sections 6.2.2.19](#) through [6.2.2.25](#)

1.2 System Overview

[Figure 1-1](#) shows a system-level block diagram of GMAC-UNIV.

Figure 1-1 GMAC-UNIV Block Diagram





- When you configure the core for a single PHY interface, the MUX logic is removed and the corresponding PHY interface is connected directly to the ports.
- When you enable the AV feature, the block diagram of the GMAC-UNIV changes as shown in [Figure 4-4](#) on page 131.

1.2.1 Interfaces

You can select GMAC-AHB or GMAC-AXI based on whether the system is designed with an AHB or AXI on-chip bus. The GMAC-AHB is designed to integrate with AMBA High-Performance Bus (AHB) whereas the GMAC-AXI is designed to integrate with AMBA Advanced extensible Interface Bus (AXI) on the application side.

The GMAC-AHB transfers data to system memory through the AHB master interface. This interface can be removed for a non-AHB system, and the subsystem can have a direct native FIFO-type interface. The host CPU uses the default 32-bit AHB Slave interface to access the GMAC subsystem's Control and Status registers (CSRs), when GMAC-AHB is selected. There is an option to select an APB port for CSR access instead of the AHB Slave port. For non-AHB systems, the AHB/APB slave modules can be removed and the native 32-bit Read/Write bus is provided for CPU accesses.

The GMAC-AXI transfers data to system memory through the AXI master interface. The host CPU uses the AXI Slave interface to access the GMAC subsystem's Control and Status registers (CSRs), when GMAC-AXI is selected. There is an option to select an APB port or AHB Slave port for CSR access instead of the AXI Slave port.

The GMAC-UNIV supports any one or a combination of the following PHY interfaces:

- ❖ Gigabit Media Independent Interface (GMII)/Media Independent Interface (MII) [Default]
- ❖ Reduced GMII (RGMII)
- ❖ Serial GMII (SGMII)
- ❖ Ten Bit Interface (TBI)
- ❖ Reduced MII (RMII)
- ❖ Serial MII (SMII)
- ❖ Reduced TBI (RTBI)
- ❖ Reverse MII (RevMII)

1.2.2 Transmit and Receive FIFOs

The Transmit FIFO (Tx FIFO) buffers data read from system memory by the DMA before transmission by the GMAC Core. Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until they are transferred to system memory by the DMA. These are asynchronous FIFOs, as they also transfer the data between the application clock and the GMAC line clocks. Both FIFOs are required to be two-ported RAM of configurable width (35/68/133 bits wide for 32/64/128 data bus widths).

In the GMAC-MTL configuration, an additional two-port RAM (width configurable from 12 to 15 bits) is needed to store the frame lengths of received frames in the Rx FIFO. The depth of this frame-length FIFO depends on the maximum number of frames that can be stored in the Rx FIFO. This frame-length FIFO is optional, and is disabled by default.

1.3 GMAC-UNIV Features

The GMAC-UNIV features are divided into the following categories:

- ❖ GMAC Core Features
- ❖ DMA Block Features
- ❖ Transaction Layer (MTL) Features
- ❖ AMBA AHB Interface Features
- ❖ AMBA AXI Interface Features
- ❖ Audio Video Support Features
- ❖ Monitoring, Testing, and Debugging Support Features

1.3.1 GMAC Core Features

The GMAC core has the following features:

- ❖ Supports 10/100/1000 Mbps data transfer rates with the following PHY interfaces:
 - ◆ IEEE 802.3-compliant GMII/MII (default) interface to communicate with an external Gigabit/Fast Ethernet PHY.
 - ◆ IEEE 802.3z-compliant TBI, (optional), with auto-negotiation to communicate with an external PHY.
 - ◆ RGMII/RTBI interface (optional) to communicate with an external gigabit PHY.
 - ◆ SGMII interface (optional) to communicate with an external gigabit PHY.
 - ◆ SMII/RMII interface (optional) to communicate with an external Fast Ethernet PHY.
 - ◆ RevMII interface (optional) to directly communicate with a remote MAC.
- ❖ Provides the following features for full-duplex operation:
 - ◆ IEEE 802.3x flow control automatic transmission of zero-quanta pause frame on flow control input de-assertion.
 - ◆ Optional forwarding of received pause control frames to the user application.
- ❖ Provides the following features for half-duplex operation:
 - ◆ CSMA/CD Protocol support
 - ◆ Flow control using back-pressure support
 - ◆ Frame bursting and frame extension in 1000 Mbps half-duplex operation
- ❖ Supports preamble and start-of-frame data (SFD) insertion in Transmit path. In the Receive path, the GMAC core supports preamble and SFD deletion.
- ❖ Supports Automatic CRC and pad generation controllable on a per-frame basis.
- ❖ Provides options for Automatic Pad/CRC Stripping on receive frames.
- ❖ Supports a variety of flexible address filtering modes such as:
 - ◆ Up to 31 additional 48-bit perfect (DA) address filters with masks for each byte.
 - ◆ Up to 31 48-bit SA address comparison check with masks for each byte.
 - ◆ 64-bit Hash filter (optional) for multicast and unicast (DA) addresses.
 - ◆ Option to pass all multicast addressed frames.
 - ◆ Promiscuous mode support to pass all frames without any filtering for network monitoring.

- ◆ Passes all incoming packets (as per filter) with a status report.
- ❖ Supports programmable frame length to support Standard or Jumbo Ethernet frames with up to 16 KB of size.
- ❖ Supports programmable Interframe Gap (IFG) (40-96 bit times in steps of 8).
- ❖ Supports separate 32-bit status for transmit and receive packets.
- ❖ Supports IEEE 802.1Q VLAN tag detection for reception frames.
- ❖ Supports separate transmission, reception, and control interfaces to the Application.
- ❖ Supports configurable big-endian and little-endian for the Transmit and Receive paths.
- ❖ Supports 32/64/128-bit data transfer interface on the system-side.
- ❖ Provides the support for network statistics (optional) with RMON/MIB Counters (RFC2819/RFC2665).
- ❖ Provides an optional module that you can use for detection of LAN wake-up frames and AMD Magic Packet frames.
- ❖ Provides an optional Receive module for checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame (Type 1).
- ❖ Provides an optional Enhanced Receive module for checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams (Type 2).
- ❖ Provides an optional module to support Ethernet frame time stamping as described in IEEE 1588-2002 and IEEE 1588-2008. Sixty-four-bit time stamps are given in the transmit or receive status of each frame.
- ❖ Supports MDIO Master interface (optional) for PHY device configuration and management.
- ❖ Supports the standard IEEE P802.3az, version D2.0 for Energy Efficient Ethernet.

1.3.2 Transaction Layer (MTL) Features

The MTL block consists of two sets of FIFOs: a Transmit FIFO with programmable threshold capability, and a Receive FIFO with a programmable threshold (default of 64 bytes).

The MTL block has the following features:

- ❖ 32-bit, 64-bit, or 128-bit Transaction Layer block provides a bridge between the application and the GMAC-CORE.
- ❖ Provides single-channel Transmit and Receive engines.
- ❖ Supports data transfers executed using simple FIFO-protocol.
- ❖ Supports synchronization for all clocks in the design (Transmit, Receive, and system clocks).
- ❖ Supports optimization for packet-oriented transfers with frame delimiters.
- ❖ Provides four separate ports for system-side and GMAC-CORE-side transmission and reception.
- ❖ Provides two 2-port RAM-based asynchronous FIFOs with synchronous/asynchronous Read and Write operation with respect to the Read and Write clocks (one for transmission and one for reception).
- ❖ Supports FIFO instantiation outside the top-level module to facilitate memory testing/instantiation.
- ❖ Supports 128-byte, 256-byte, or 512-byte, or 1-, 2-, 4-, 8-, 16-, or 32-KB receive FIFO sizes on reception.

- ❖ Provides an optional interface to indicate the length of a received frame at the top of the MTL Rx FIFO in the GMAC-MTL configuration.
- ❖ Supports programmable burst-length for starting a burst up to half the size of the MTL Rx and Tx FIFO in the GMAC-MTL configuration.
- ❖ Supports Receive Status vectors insertion into the Receive FIFO after the EOF transfer. This enables multiple-frame storage in the Receive FIFO without requiring another FIFO to store those frames' Receive Status.
- ❖ Supports configurable Receive FIFO threshold (default fixed at 64 bytes) in Cut-Through or threshold mode.
- ❖ Provides an option to filter all error frames on reception and not forward them to the application in Store-and-Forward mode.
- ❖ Provides an option to forward under-sized good frames.
- ❖ Supports statistics by generating pulses for frames dropped or corrupted (due to overflow) in the Receive FIFO.
- ❖ Supports 256-byte or 512-byte, or 1-, 2-, 4-, 8-, or 16-KB FIFO sizes on transmission.
- ❖ Supports Store and Forward mechanism for transmission to the GMAC core.
- ❖ Supports threshold control for transmit buffer management.
- ❖ Supports configurable number of frames to be stored in FIFO at any time. The default is 2 frames (fixed) with internal DMA, and up to 8 frames in GMAC-MTL configuration.
- ❖ Supports automatic generation of PAUSE frame control or backpressure signal to the GMAC core based on Receive FIFO-fill (threshold configurable) level.
- ❖ Handles automatic retransmission of Collision frames for transmission.
- ❖ Discards frames on late collision, excessive collisions, excessive deferral, and under-run conditions.
- ❖ Provides software control to flush Tx FIFO.
- ❖ Data FIFO RAM chip-select disabled when inactive, to reduce power consumption.
- ❖ Provides an optional module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum in frames transmitted in Store-and-Forward mode.

1.3.3 DMA Block Features

The DMA block exchanges data between the MTL block and host memory. The host can use a set of registers (DMA CSR) to control the DMA operations.

The DMA block has the following features:

- ❖ Supports 32/64/128-bit data transfers.
- ❖ Supports single-channel Transmit and Receive engines.
- ❖ Provides fully synchronous design operating on a single system clock (except for CSR module, when a separate CSR clock is configured).
- ❖ Provides optimization for packet-oriented DMA transfers with frame delimiters.
- ❖ Supports byte-aligned addressing for data buffer support.
- ❖ Supports dual-buffer (ring) or linked-list (chained) descriptor chaining.

- ❖ Supports descriptor architecture that allows large blocks of data transfer with minimum CPU intervention; each descriptor can transfer up to 8 KB of data.
- ❖ Supports comprehensive status reporting for normal operation and transfers with errors.
- ❖ Supports individual programmable burst size for Transmit and Receive DMA Engines for optimal host bus utilization.
- ❖ Supports programmable interrupt options for different operational conditions.
- ❖ Provides per-frame Transmit/Receive complete interrupt control.
- ❖ Supports round-robin or fixed-priority arbitration between Receive and Transmit engines.
- ❖ Supports Start/Stop modes.
- ❖ Provides separate ports for host CSR access and host data interface.

1.3.4 AMBA AHB Interface Features

The AMBA AHB interface features can be divided into the following categories:

- ❖ [AHB Master Interface Features](#)
- ❖ [AHB Slave Interface Features](#)

1.3.4.1 AHB Master Interface Features

The AHB Master Interface has the following features:

- ❖ Interfaces with the application through AHB.
- ❖ Supports little-endian and big-endian modes.
- ❖ Supports 32-bit, 64-bit, or 128-bit data on the AHB Master port.
- ❖ Provides option to select address-aligned bursts from AHB Master port.
- ❖ Supports Split, Retry, and Error AHB responses.
- ❖ Handles the AHB 1K boundary burst splitting.
- ❖ Does not generate wrap burst.
- ❖ Software can select the type of AHB burst (fixed burst, indefinite burst, or mix of both).

1.3.4.2 AHB Slave Interface Features

The AHB Slave Interface has the following features:

- ❖ Interfaces with the application through AHB.
- ❖ Supports little-endian and big-endian modes.
- ❖ AHB Slave interface (32-bit, 64-bit, or 128-bit) for CSR access, in which only 32-bit or less (byte, half-word) accesses are possible.
- ❖ Provides option for a 32-bit APB port for CSR access instead of an AHB Slave port.
- ❖ Supports all AHB burst types.
- ❖ Does not generate Split, Retry, or Error responses.

1.3.5 AMBA AXI Interface Features

The AMBA AXI interface features can be divided into the following categories:

- ❖ AXI Master Interface Features
- ❖ AXI Slave Interface Features

1.3.5.1 AXI Master Interface Features

The AXI Master Interface has the following features:

- ❖ Interfaces with the application through AXI.
- ❖ Supports 32-bit address width.
- ❖ Supports little-endian and byte-invariant big-endian modes.
- ❖ Supports AXI low-power interface.
- ❖ Supports 32-bit, 64-bit, or 128-bit data.
- ❖ Supports OKAY, SLVERR, and DECERR responses.
- ❖ Software can select the type of AXI burst (fixed and variable length burst) in the AXI Master interface. Provides an option to select extended length fixed bursts of 32, 64, 128, and 256.
- ❖ Provides an option to select address-aligned bursts.
- ❖ Handles the AXI 4K boundary burst splitting.
- ❖ Supports up to 16 read and write outstanding transactions.
- ❖ Supports using posted writes from the AXI Master interface to maximize the bus utilization.
- ❖ Does not support burst interleaving and reordering on AXI Master Write Channel.
- ❖ Does not support Atomic, Locked, Cache, and Protected accesses.

1.3.5.2 AXI Slave Interface Features

The AXI Slave Interface has the following features:

- ❖ Interfaces with the application through AXI.
- ❖ AXI Slave interface (32-bit, 64-bit, or 128-bit) for CSR access.
- ❖ Supports 32-bit address width.
- ❖ Supports little-endian and byte-invariant big-endian modes.
- ❖ Provides option for a 32-bit APB port for CSR access instead of an AXI Slave port.
- ❖ Supports narrow burst and FIXED/INCR burst.
- ❖ Does not support Exclusive access.
- ❖ Does not support Atomic, Locked, Cache, and Protected accesses.

1.3.6 Audio Video Support Features

The GMAC-UNIV supports the following Audio Video (AV) features:

- ❖ Supports separate channels or queues for AV data transfer in 100 Mbps and 1000 Mbps modes.
- ❖ Supports configuring up to two additional channels (Channel 1 and Channel 2) on the transmit and receive paths for AV traffic. The channel 0 is available by default and carries the legacy best-effort Ethernet traffic on the transmit side.
- ❖ Supports IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for the additional transmit channels.

- ❖ Provides separate DMA, TxFIFO, and RxFIFO (MTL) for each additional channel. The system-side interface (AHB, AXI, or native) remains the same.

1.3.7 Monitoring, Testing, and Debugging Support Features

The GMAC-UNIV has the following features that you can use for monitoring, testing, and debugging:

- ❖ Supports internal loopback on the GMII/MII for debugging.
- ❖ Provides DMA states (Tx and Rx) as status bits.
- ❖ Provides Debug status register that gives status of FSMs in Transmit and Receive data-paths and FIFO fill-levels.
- ❖ Application Abort status bits.
- ❖ MMC (RMON) module in the GMAC core.
- ❖ Current Tx/Rx Buffer pointer as status registers.
- ❖ Current Tx/Rx Descriptor pointer as status registers.
- ❖ Statistical counters that help in calculating the bandwidth served by each transmit channel when AV support is enabled.

1.4 GMAC-UNIV Configuration

The GMAC-UNIV can be configured to any of the following major architectures using coreConsultant:

- ❖ GMAC Subsystem with DMA and AHB interface (default)
- ❖ GMAC Subsystem with DMA and AXI interface
- ❖ GMAC Subsystem with DMA and native interface (without AHB and AXI)
- ❖ GMAC Subsystem with transaction (FIFO) layer only (without DMA) and with native FIFO interface
- ❖ GMAC core with native FIFO interface (without DMA, FIFO layer, AHB and AXI)

In addition, the PHY can be configured to have any of the following interfaces (including an option to select them at reset), if multiple interfaces are configured:

- ❖ GMII/MII
- ❖ TBI
- ❖ RGMII
- ❖ RTBI
- ❖ SGMII
- ❖ RMII (for 10/100 Mbps operations only)
- ❖ SMII (for 10/100 Mbps operations only)
- ❖ RevMII

1.4.1 Configurable Features

You can configure the following features in coreConsultant for an optimized design with respect to area and timing:

- ❖ 10/100 Mbps operation only
- ❖ 1000 Mbps operation only

- ❖ 10/100/1000 Mbps operation
- ❖ Full-duplex operation only
- ❖ Depth of Receive FIFO in transaction layer
- ❖ Depth of Transmit FIFO in transaction layer
- ❖ Bus width of AHB/AXI (or application) interface to 32/64/128
- ❖ Address filtering option (Hash filter function)
- ❖ Configurable number of MAC Address registers (up to 32) for filtering
- ❖ RMON Counters (MMC) with individual registers selection
- ❖ Power Management Module (PMT) with remote wake-up and magic packet frame processing options
- ❖ TCP, UDP, or ICMP over IP Checksum Offload engine
- ❖ IEEE 1588 Ethernet frame timestamp support
- ❖ AV traffic queuing and forwarding support
- ❖ Endianness for all application-side data paths
- ❖ Station Management Agent (SMA) – MDIO module
- ❖ APB port instead of AHB/AXI Slave port
- ❖ Separate application clocks option for AHB/APB slave port

1.5 Deliverables

The GMAC-UNIV is packaged as a dwc_ethernet_3.60a.run file. The GMAC-UNIV requires the Synopsys coreConsultant tool (described in detail in “[Building and Verifying Your Core](#)” on page 33) to install, unpack, and configure the core. The license file required to install, unpack, and configure GMAC-UNIV is delivered separately.

The GMAC-UNIV image includes the following:

- ❖ Verilog RTL source code
- ❖ Synthesis script generation for Synopsys Design Compiler and Synplify Pro.
- ❖ Verification testbenches and test configurations derived from your parameter choices
 - ◆ Verilog-based system-level testbench environment and test cases for GMAC-AHB configuration. The Vera-based DesignWare AMBA Verification IP (VIP) is used as the verification model for the AHB bus in this testbench.

For more information about the Verilog Testbench (VTB) used to test GMAC-AHB, see “[GMAC-AHB Verification Environment](#)” on page 495.

- ◆ Verilog-based testbench environment for GMAC-AHB and GMAC-AXI configuration using DW AMBA and DW Ethernet VIPs. You can use this testbench, along with the provided test cases, as a reference to build a verification environment for an SoC in which GMAC-AHB or GMAC-AXI integration can be tested.

For more information about the VTB used to test the GMAC-AHB using AMBA Verification IP (VIP) and Ethernet Verification IP, see “[GMAC-AHB Verification Environment Using AMBA and Ethernet VIPs](#)” on page 479.

- ◆ Verilog-based testbench environment and test cases for GMAC-DMA, GMAC-MTL, and GMAC-CORE configurations.
- ◆ Support for creating gate-level Device Under Test (DUT) models using a GTECH library.
- ◆ Verification scripts for running with three simulators: Synopsys VCS, ModelSim, and NC-Verilog simulators.
- ❖ IP-XACT component representation of the IP (XML format). The coreConsultant supports the generation of configured core integration files that meet the IP-XACT specifications. The coreConsultant also supports the file generation for IP-XACT-compliant software viewing (Memorymap.tcl) of the configured core.
- ❖ Useful example code, scripts, and documents, including:
 - ◆ Implementation flow guidelines and scripts
 - ◆ A configuration for attaching a PVCI interface to GMAC-DMA
 - ◆ C code for calculating CRC-32 and CRC-16



Note The software driver is available on request. Contact your Synopsys representative for more information.

1.5.1 Licensing Features

The GMAC-UNIV license file includes the following licenses:

- ❖ One DW license (DWC-ETHERNET) for installation and configuration of the GMAC-UNIV
- ❖ Two licenses for the following Verification IPs (VIP) to run the simulations:
 - ◆ DesignWare Ethernet VIP (DesignWare-Ethernet-VIP)
 - ◆ DesignWare AMBA VIP (DesignWare-AMBA-VIP)



Note New features such as the AXI system interface, IEEE 1588-2008 Advanced Time Stamping, Energy Efficient Ethernet, and AV support are available only when you have the license for the DWC Ether MAC 10/100/1000 Universal product. With this license, you can still generate a 10/100 Mbps core with the specified features.

In addition, you need the following licenses:

- ❖ For Synopsys synthesis tools:
 - ◆ Design Compiler or Synplify Pro
 - ◆ HDL-Compiler
- ❖ For simulation tools:
 - ◆ Synopsys VCS Simulator, or
 - ◆ Any other simulator such as ModelSim-Verilog or NC-Verilog

2

Building and Verifying Your Core

This chapter describes how to set up, configure, and verify the core. It contains the following sections:

- ❖ “Setup Requirements” on page 33
- ❖ “Basic Design Flow” on page 33
- ❖ “Creating a Workspace” on page 34
- ❖ “Configuration: Creating the RTL” on page 36
- ❖ “Synthesis: Creating the Gate-Level Netlist” on page 36
- ❖ “Simulation: Verifying DWC Ether MAC 10/100/1000 Universal” on page 37

2.1 Setup Requirements

Before performing the procedures in this chapter, make sure that you have installed the DWC Ether MAC 10/100/1000 Universal database and set up your environment. In addition, it is important that you use the required versions of coreConsultant, Design Compiler, Vera, and your preferred simulation tool, and that all associated paths and environment variables are set correctly.



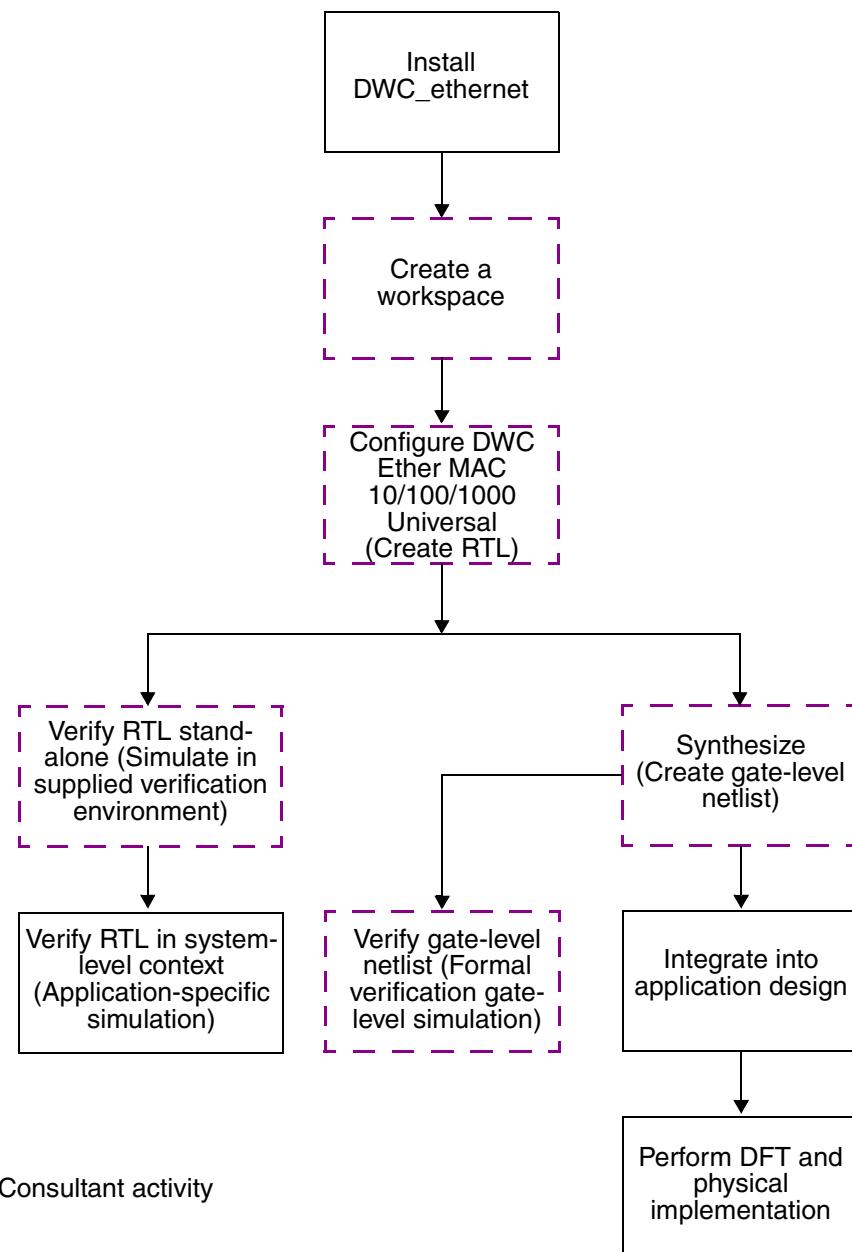
Note In this chapter, the DWC Ether MAC 10/100/1000 Universal product is used as an example. The information in this chapter is also applicable to the DWC Ether MAC 10/100 Universal product.



Attention IP core products are subject to frequent enhancements and fixes. To ensure you have the latest DesignWare Cores Ethernet MAC Universal release, go to www.designware.com and enter Ethernet MAC Universal in the “Search for IP” box.

2.2 Basic Design Flow

You use coreConsultant to configure, verify, and synthesize the DWC Ether MAC 10/100/1000 Universal. [Figure 2-1](#) shows the basic sequence of tasks.

Figure 2-1 Design Flow

2.3 Creating a Workspace

A workspace is a local copy of your DWC Ether MAC 10/100/1000 Universal installation in which you can configure, verify, and synthesize your own implementation of the DWC Ether MAC 10/100/1000 Universal. After installing DWC Ether MAC 10/100/1000 Universal, you must create a workspace to begin working. You can create several workspaces so that you can experiment with different design alternatives, as shown in [Figure 2-2](#).

To create a workspace:

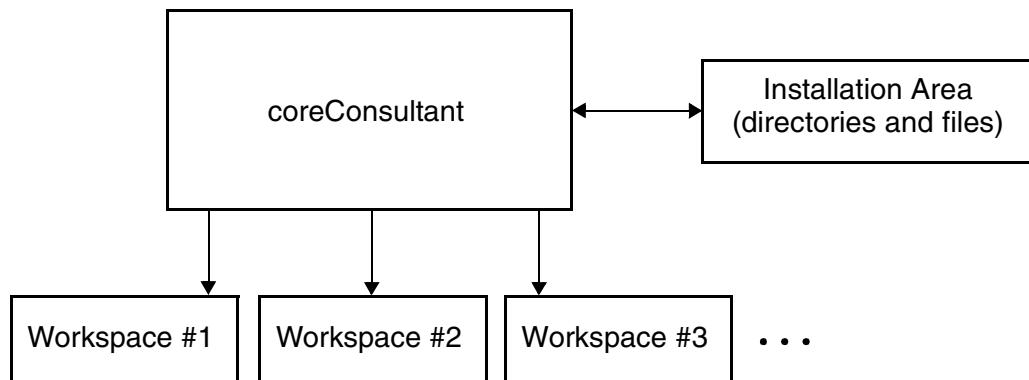
1. Invoke coreConsultant:

```
% coreConsultant
```

2. Select File > New Workspace in the coreConsultant console, and then enter the requested information in the New Workspace dialog box.

For more information about the New Workspace dialog box options, refer to the coreConsultant online help.

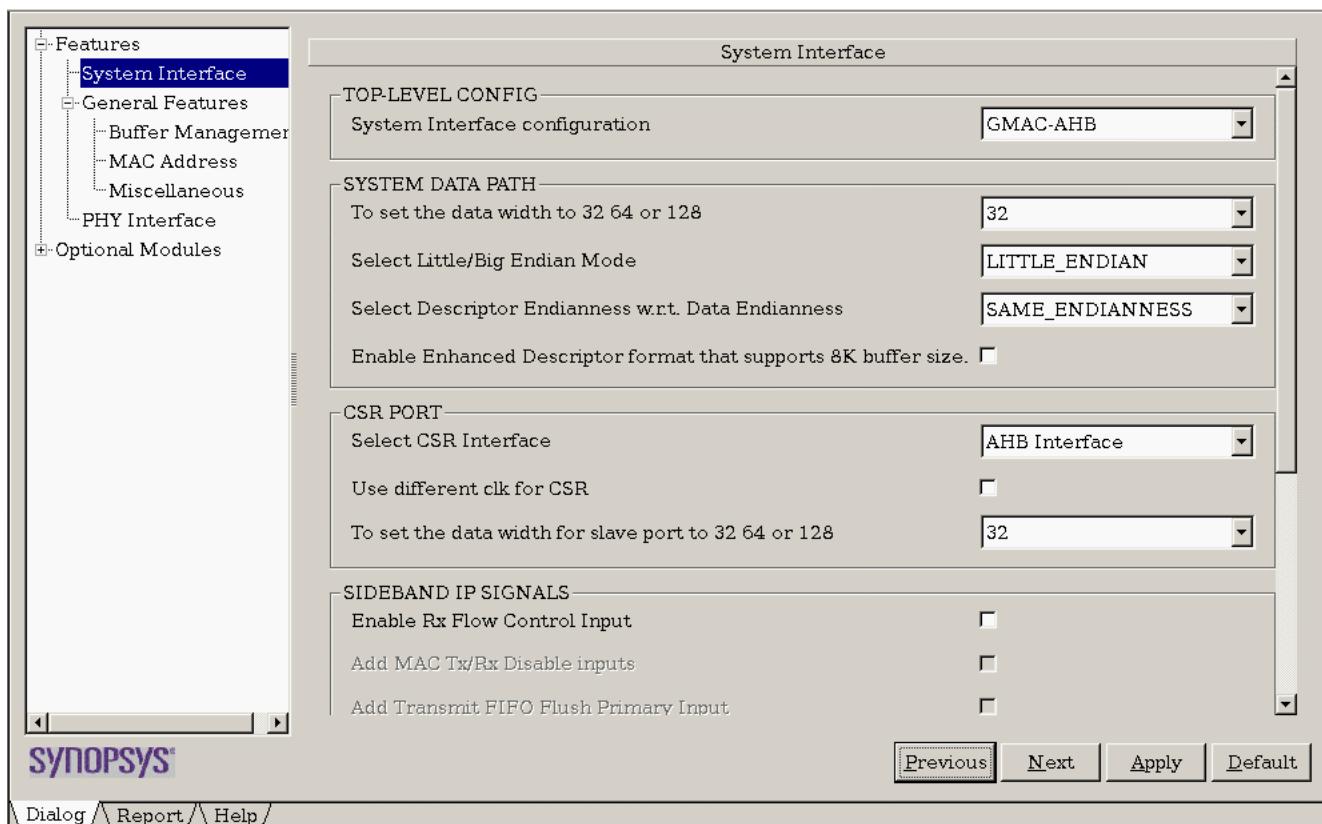
Figure 2-2 Installation and Workspaces



After the workspace creation, coreConsultant displays the Activity List ([Figure 2-3](#)). For DWC Ether MAC 10/100/1000 Universal, the first activity to execute is Specify Configuration. This activity is described in [“Configuration: Creating the RTL”](#) on page 36.



Note Use the Set Design Prefix activity if you plan to instantiate DWC Ether MAC 10/100/1000 Universal more than once in your design. The default Set Design Prefix state is completed because it is not a required activity for most DWC Ether MAC 10/100/1000 Universal users.

Figure 2-3 coreConsultant Specify Configuration Dialog Box

2.4 Configuration: Creating the RTL

To configure DWC Ether MAC 10/100/1000 Universal interactively, go the Specify Configuration dialog box and select your configuration options. There are options available to enable or disable certain features, select memory sizes and data bus width, and select the application and PHY interfaces. You can use the default values for an initial simulation and synthesis trial. Otherwise, you must select or enter the values required for your design.

Refer to “[Parameters](#)” on page [355](#) for detailed descriptions of all configuration options. After you have specified values for all configuration options, click **Apply** to generate the configured RTL code (with `DWC_gmac_top.v` as the top-level file). The coreConsultant checks your parameter values, generates configured RTL code, and displays a configuration report.

You can reconfigure DWC Ether MAC 10/100/1000 Universal (create new RTL) at any time. If you do so, you need to re-complete any downstream activities.

2.5 Synthesis: Creating the Gate-Level Netlist

coreConsultant is your interface to Synopsys synthesis tools for synthesizing DWC Ether MAC 10/100/1000 Universal. Follow the sequence of activities under the Create Gate-Level Netlist group in the coreConsultant GUI to choose your synthesis options, run the synthesis job, and view the results. For more information about specific dialog box items, refer to the coreConsultant online help.

2.6 Simulation: Verifying DWC Ether MAC 10/100/1000 Universal

You can perform two types of verification from within coreConsultant:

- ❖ **Formal Verification:** Runs a formal verification (using the Synopsys Formality tool) to compare the configured RTL with your post-synthesis netlist.
 - ◆ To run a formal verification from coreConsultant, ensure that SVF is enabled and that a proper source and target are selected. By default, the latest synthesis database is taken for comparison.
 - ◆ To run a formal verification outside coreConsultant, look for the formal verification script, named xxxx_fm.tcl (incr2_fm.tcl, for example), in the <workspace>/syn/ directory. This script can run in the Formality shell as a standalone.
- ❖ **Simulate:** Simulates DWC Ether MAC 10/100/1000 Universal in the supplied test environment. You can simulate your configured RTL version of DWC Ether MAC 10/100/1000 Universal or your synthesized gate-level netlist.

For RTL simulation, if you are using an evaluation (encrypted RTL) version of DWC Ether MAC 10/100/1000 Universal, you may need to create a GTECH model for simulation:

- ◆ If you are using the Synopsys VCS simulator with an evaluation version of DWC Ether MAC 10/100/1000 Universal, you can set up and run your simulation directly on the encrypted RTL.
- ◆ If you are using another simulator with an evaluation version of DWC Ether MAC 10/100/1000 Universal, you must first generate a GTECH netlist of the RTL to use in your simulation.

If you are using the paid (source-code) version of DWC Ether MAC 10/100/1000 Universal, you can set up and run your simulation directly on the RTL using any of the supported simulators.



Note When using ModelSim, make sure that the LD_LIBRARY_PATH variable points to SystemC gcc 3.3.6.

2.6.1 Generating a GTECH Model

You only need to generate a GTECH simulation model if you are using a non-Synopsys simulator with an encrypted RTL version of DWC Ether MAC 10/100/1000 Universal. To generate a GTECH simulation model:

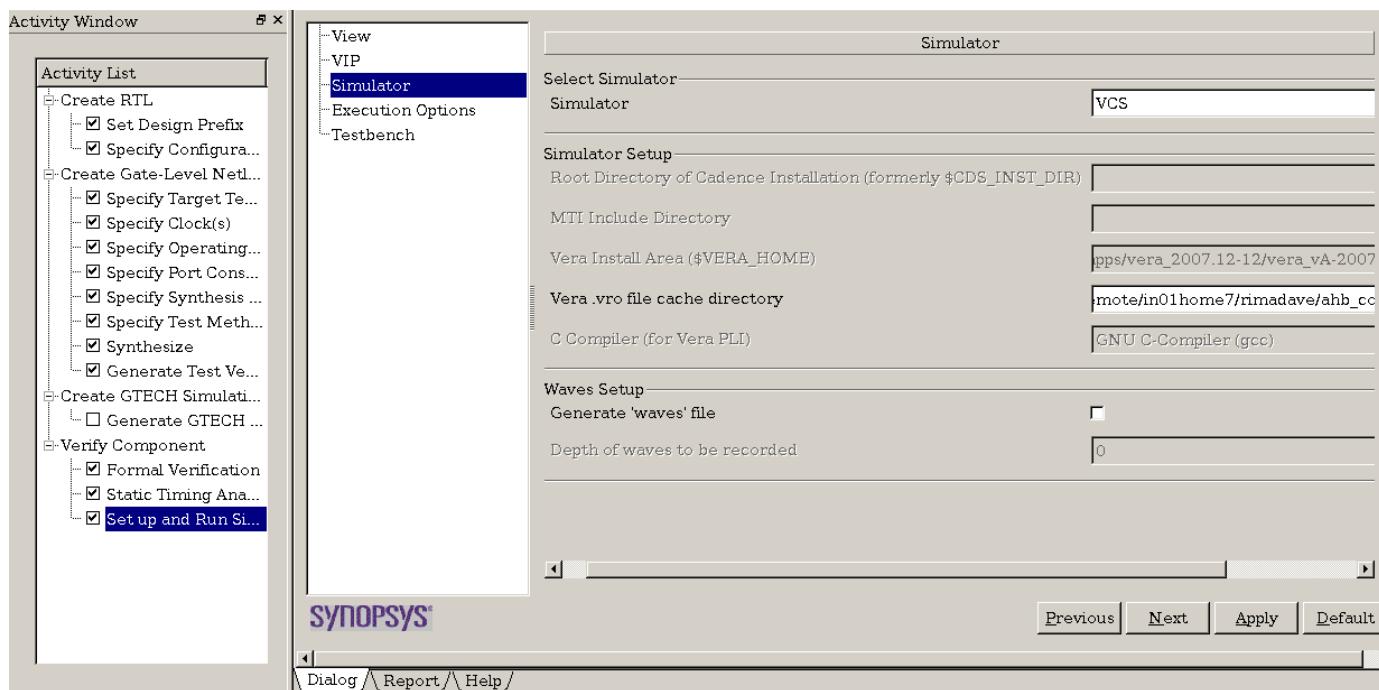
1. Select your options for the GTECH synthesis job. For information about individual options, refer to the coreConsultant online help.
2. Click Apply to launch the GTECH synthesis job.
3. Go to the Generate GTECH Model activity.
4. When the GTECH model activity is complete, you can select the GTECH simulation model as the “Design view to simulate” in the View page of the Simulate dialog box.

2.6.2 Verifying DWC Ether MAC 10/100/1000 Universal

The Set up and Run Simulations activity ([Figure 2-4](#)) is where you select your simulation options and run the simulation. When the simulation is complete, the results are available in the Report tab of this activity.



Note Do not click Apply until you have selected all of your simulation options on all pages of the Simulate dialog box. Clicking Apply in any page of the Simulate dialog box applies all of your simulation options (on all pages) and starts the simulation.

Figure 2-4 Set Up and Run Simulations Dialog Box

To run the simulation through coreConsultant, go to the Set up and Run Simulations dialog box (Figure 2-4) and select from the following simulation options:

- ❖ **Simulator:** Options to select and set up your simulation tool and to generate waveforms. Use coreConsultant's online help to learn more about individual options.
- ❖ **View:** Options that select which view of DWC Ether MAC 10/100/1000 Universal you want to simulate. Choose RTL to simulate your configured RTL model of DWC Ether MAC 10/100/1000 Universal or GTECH to simulate the generated GTECH model or GateLevel to simulate your technology-specific gate-level netlist.



You can run GateLevel simulations only with a limited subset of testcases.

- ❖ **Execution Options:** Options related to simulation launch and execution. Use the coreConsultant online help for more information about individual options.
- ❖ **Testbench:** A list of tests that you can select to execute or not execute. By default, all tests applicable to the selected top-level configuration are selected. Test suites that do not apply to the selected configuration are grayed out. You can run a limited subset of the test suite for sanity check by selecting the Run Quick List Tests option.

After selecting your simulation options and tests, click Apply to launch the simulation. If you select the Do Not Launch Simulation option, coreConsultant does not invoke the simulation; go to your <workspace>/sim/ directory and execute run.scr to run the simulation.

When the simulation is complete, use the Report tab to view the simulation log.

If any test fails, contact support_center@synopsys.com after checking the Known Problems list in the release notes.



3

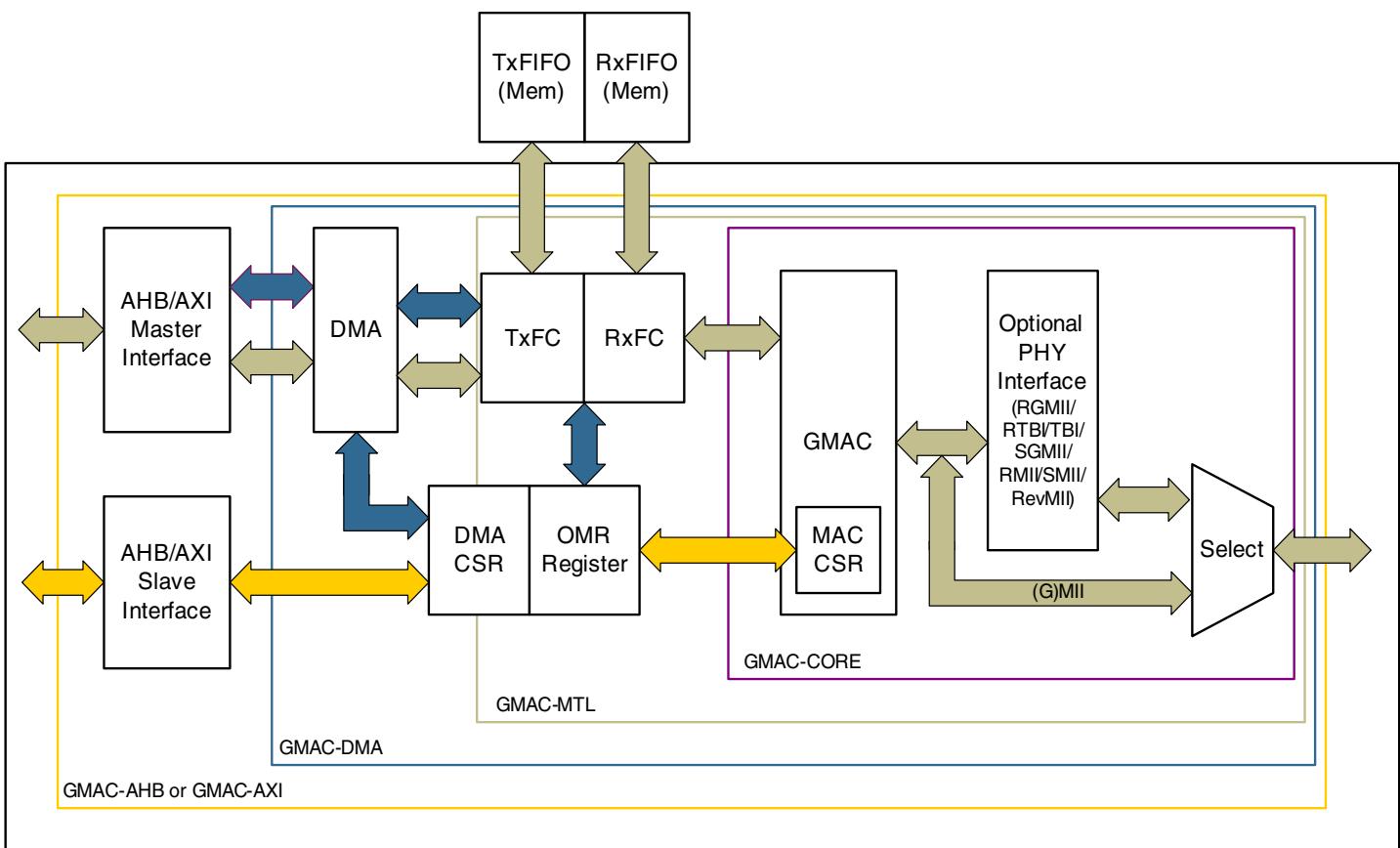
Architecture

This chapter describes the GMAC-UNIV interfaces, protocols, functionality, and implementation. It contains the following sections:

- ❖ “[Introduction](#)” on page [41](#)
- ❖ “[AHB Application Host Interface](#)” on page [42](#)
- ❖ “[AXI Application Host Interface](#)” on page [47](#)
- ❖ “[DMA Controller](#)” on page [55](#)
- ❖ “[MAC Transaction Layer \(MTL\)](#)” on page [71](#)
- ❖ “[GMAC Core](#)” on page [89](#)

3.1 Introduction

The GMAC-UNIV is a configurable product with multiple interfaces to the system side or the PHY side. Major modules and functions are only selected when required for your design, ensuring that your design is optimized for area. A block diagram of the GMAC-UNIV’s five major system configurations is provided in [Figure 3-1](#).

Figure 3-1 GMAC-UNIV Block Diagram

3.2 AHB Application Host Interface

In the GMAC-AHB core, the DMA Controller interfaces with the Host through the AMBA AHB Interface. The AHB Master Interface controls data transfers while the AHB Slave interface accesses CSR space. The DMA can be used in 32/64/128-bit embedded applications where DMA is required to optimize data transfer between the GMAC and system memory. For information about the data transfers in little-endian and big-endian modes, see “[Endian Support](#)” on page [563](#).

3.2.1 AHB Master Interface

The AHB Master interface converts the internal DMA request cycles into AHB cycles. The AHB Master interface has the following characteristics:

- ❖ The AHB Master interface is AMBA 2.0-compliant AHB Master with no restrictions.
- ❖ The AHB Master interface supports the following burst length modes:
 - ◆ Fixed Burst Length

In fixed burst length mode, the AHB master always initiates a burst with SINGLE, INCR4, INCR8, or INCR16 type. But when such a burst is responded with SPLIT/RETRY/early burst termination, the AHB master re-initiates the pending transfers of the burst with INCR or SINGLE burst-length type. The AHB master terminates such INCR bursts when the original requested fixed-burst is transferred.

In fixed burst length mode, if the DMA requests a burst transfer that is not equal to INCR4/8/16, the AHB Host interface splits the transfer into multiple burst transactions. For example, if the DMA requests a 15-beat burst transfer, the AHB interface splits it into multiple transfers of INCR8 and INCR4, and 3 SINGLE transactions.

- ◆ Unspecified burst length

In unspecified burst mode, the AHB master always initiates a transfer with INCR and complete the DMA requested burst in one go.

- ◆ Mixed burst mode

In mixed burst mode, the AHB master always initiates the bursts with fixed-size (INCRx) when the DMA requests transfers of size less than or equal to 16 beats. When DMA requests bursts of length more than 16, the AHB master initiates such transfers with INCR and complete it in one go.

You can choose the burst modes by programming the FB or MB bits in [Register 0 \(Bus Mode Register\)](#).

Note: The AHB master does not generate WRAP bursts.

- ❖ The AHB slaves interfaced to the GMAC-AHB support SINGLE and INCR transfers.
- ❖ The AHB Master interface can handle the AHB SPLIT, RETRY, and ERROR conditions. Any ERROR response halts all further transactions for that DMA, and indicate the error as fatal through the CSR and interrupt. The application must give a hard or soft reset to the module to restart the operation.
- ❖ The AHB Master interface can handle the AHB 1K boundary breaking.
- ❖ The AHB Master interface can handle all 32-bit, 64-bit, or 128-bit data transfers (according to the data bus width configuration), except for Descriptor Status Write accesses (which are always 32-bit). In any burst data transfer, the address bus value is always aligned to the data bus width and need not be aligned to the beat size.
- ❖ The AHB Master interface can align all AHB burst transfers to an address value. You can enable this by using the bit AAL in the DMA Bus Mode register. If both the FB and AAL bits are set to 1, the AHB interface and the DMA together ensure that all initiated beats are aligned to the address, completing the frame transfer in the minimum number of required beats. For example, in 32-bit data bus mode, if a data buffer transfer's start address is 0xF000_0008 and the DMA is configured for a maximum beat size of 16, the AHB transfers occur in the following sequence:
 - ◆ 2 SINGLE transfers at addresses 0xF000_0008 and 0xF000_000C
 - ◆ 1 INCR4 transfer at address 0xF000_0010
 - ◆ 1 INCR8 transfer at address 0xF000_0020
 - ◆ All subsequent beats are INCR16 transfers starting from address 0xF000_0040. For an address-aligned INCR16 transfer, the 6 least-significant bits of the address must be 0.

Similarly, the AHB and DMA split such unaligned address beats for a 64-bit (or 128-bit) data bus and initiate INCR16 beats only when the address's seven (or eight) least-significant bits have a zero value.

- ❖ The DMA Controller requests an AHB Burst Read transfer only when it can accept the received burst data completely. Data read from the AHB is always pushed into the DMA without any delay or BUSY cycles.

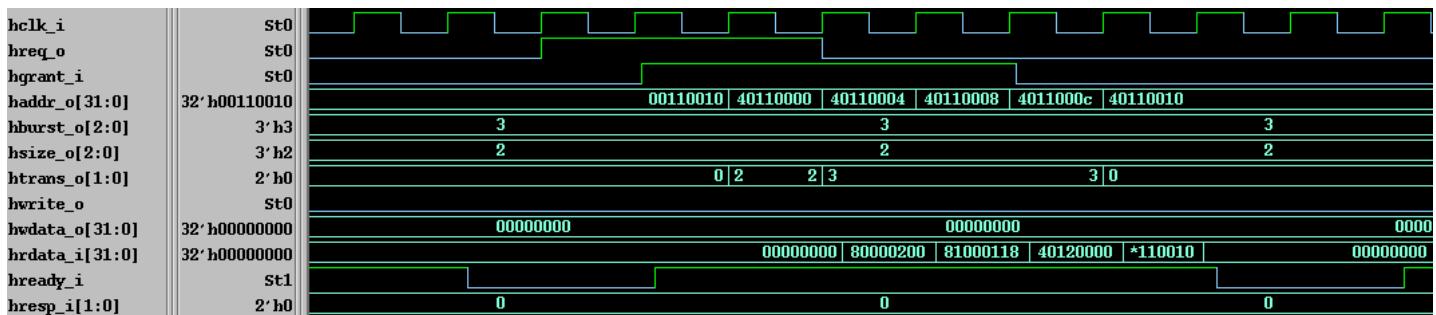
- The DMA requests an AHB Burst Write transfer only when it has the sufficient data to transfer the burst completely. The AHB interface always assumes that it has data available to push into the AHB bus. However, the DMA can prematurely indicate end-of-valid data (because of the transfer of end-of-frame of an Ethernet frame) during the burst.

In Fixed Burst Length mode, the AHB Master interface continues the burst with dummy data until the specified length is completed. In INCR mode, it takes steps to end the burst transfer prematurely.

3.2.1.1 AHB Memory Read

[Figure 3-2](#) shows the GMAC-AHB performing a memory read starting from address 0x40110000. The AHB master port uses the INCR4 burst on this occasion.

Figure 3-2 AHB Memory Read Timing

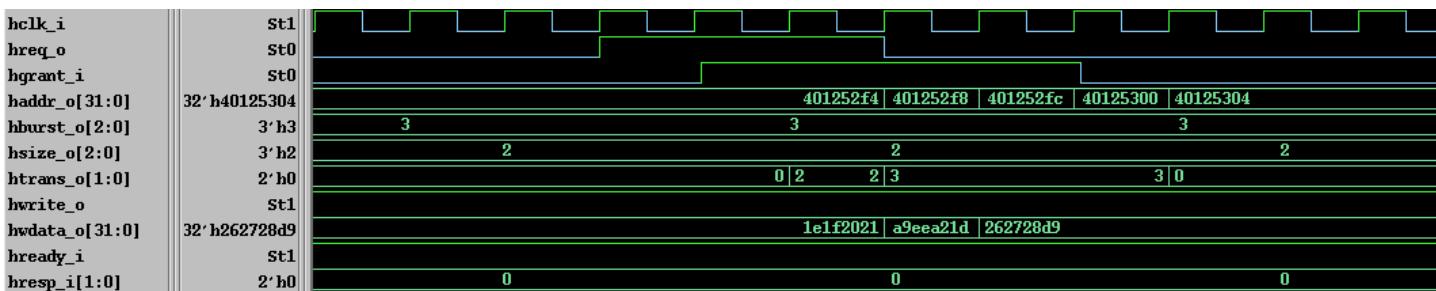


3.2.1.2 AHB Memory Write

[Figure 3-3](#) shows the GMAC-AHB performing a burst memory write transaction, starting from address 0x401252F4. This data transfer corresponds to an Ethernet frame being written to system memory by the Receive DMA. The AHB master is using the INCR4 burst. Notice that the data bus does not change values for the last two beats in the INCR4 transaction. This is because the EOF of the frame was given, with the second beat transfer, by the DMA to the AHB master port internally. As the AHB Master had started a fixed INCR4 burst, it completes the burst with dummy data (same data) for the last 2 beats.

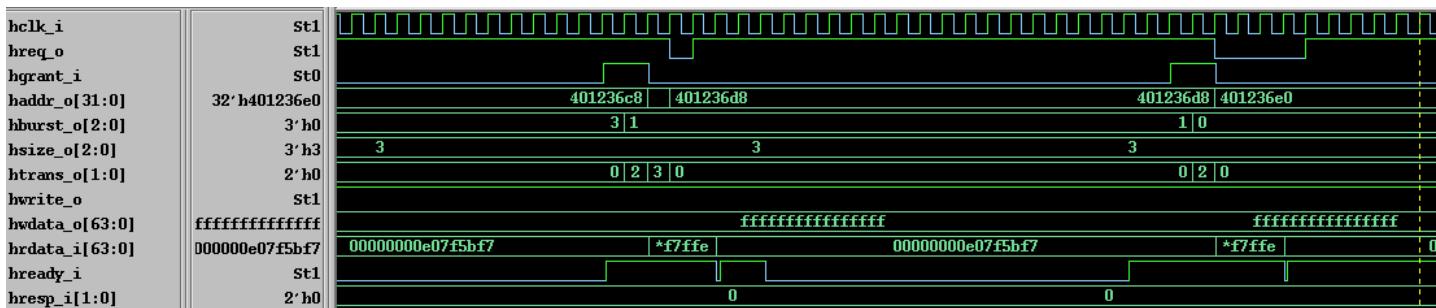
For more information about DMA operations, see “[DMA Controller](#)” on page [55](#).

Figure 3-3 AHB Memory Write Timing



3.2.1.3 Early Burst Termination

[Figure 3-4](#) shows an early burst termination occurring when the GMAC-AHB carries out an INCR (Write) transaction. This self-explanatory figure shows how the AHB master re-requests the bus grant and completes the transfer with a SINGLE transaction.

Figure 3-4 Early Burst Termination Timing

3.2.1.4 Error Response and Fatal Bus Error Interrupt

Figure 3-5 shows the behavior of the GMAC-AHB when an error response is received. Here, the DUT attempts to carry out the INCR4 transaction. However, it receives a two-cycle error response during the data phase of the first beat. Interrupt signal sbd_intr_o is asserted as a result of this response.

For more information about the interrupt functions, see “[Interrupts](#)” on page [67](#).

Figure 3-5 Error Response and Fatal Bus Error Interrupt Timing

3.2.1.5 Split/Retry Response

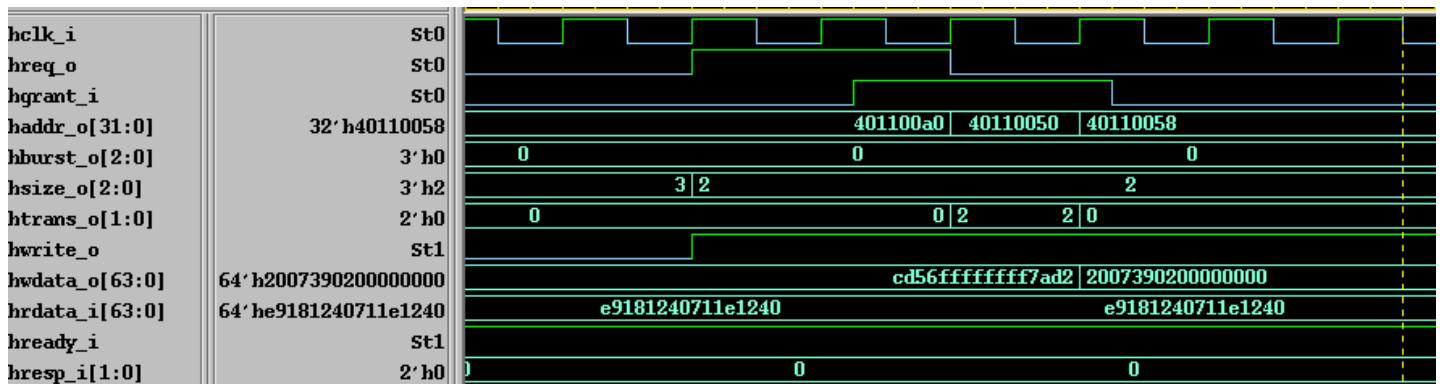
Figure 3-6 shows how the AHB Master port reacts to a split/retry response (to address 0x00120560) during a transaction. In this case, the GMAC-AHB receives a two-cycle retry response. Notice that the AHB Master immediately reconstructs the transaction and requests for bus by asserting hreq_o again. hreq_o is deasserted for only 1 cycle when the retry/split response is sampled in the first cycle of the 2-cycle response.

In case of a SPLIT response, the behavior remains the same, but the second request (hreq_o) from this master is masked in the AHB arbiter until the corresponding AHB slave is ready with the data.

Figure 3-6 Split/Retry Response Timing

3.2.1.6 Descriptor Write Back

Figure 3-7 shows a descriptor being closed on a 64-bit AHB at address 0x40110050 (See “[DMA Controller](#)” on page 55 and “[Descriptors](#)” on page 397 for descriptor operations). Note, that only 32-bit status is written back to memory and this appears on upper 32-bit of the hwdata_o bus, as the AHB is configured as big-endian. For little-endian configuration, the 32-bit descriptor data appears on the lower 32 bits of the 64-bit AHB bus.

Figure 3-7 Descriptor Write-Back Timing

3.2.2 AHB Slave Interface

The AHB 32-bit Slave interface provides access to the DMA and GMAC CSR space. The AHB Slave interface has the following characteristics:

- ❖ Fully AMBA 2.0 Compliant AHB Slave – no restrictions.
- ❖ Supports single and all burst transfers.
- ❖ Supports busy and early terminations.
- ❖ Supports 32-bit, 16-bit, and 8-bit write/read transfers to the CSR; 32-bit access to the CSR are recommended to avoid any SW synchronization problems.
- ❖ Generates OKAY only response; does not generate SPLIT, RETRY, or ERROR responses.

To interface the 32-bit AHB Slave port to a wider AMBA bus, extra logic described in Section 3-15 of the AMBA 2.0 specification is required. You can include this logic in the core at the time of RTL configuration (see “[Configuration: Creating the RTL](#)” on page 36). In such configurations, even though the AHB slave port indicates a 64-bit or 128-bit data bus, all slave port accesses must still be 32 bits or less. If a 64-bit or 128-bit access is performed, the AHB slave port completes a 32-bit access internally and gives an OKAY response.

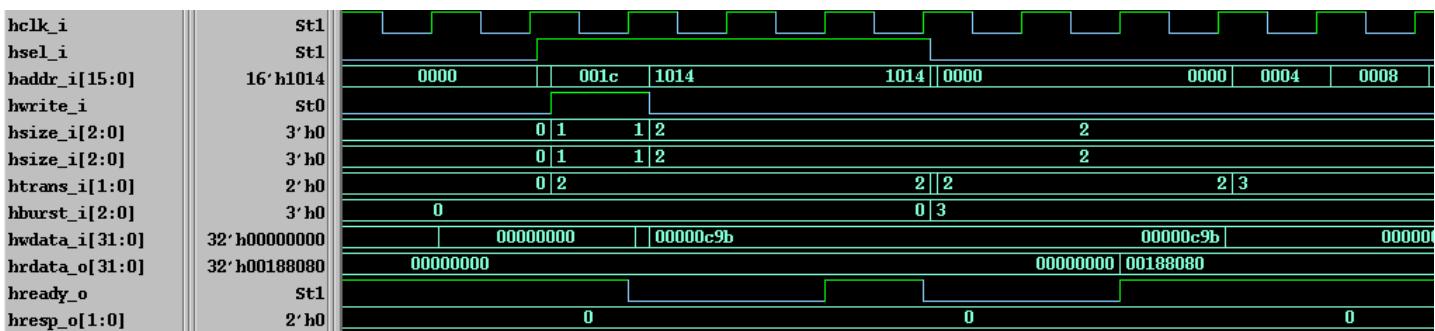
The data on the higher lanes is ignored during write operations and the lower 32-bit lane is duplicated during AHB Slave Read transfers.

You can also enable an APB port to access DMA and GMAC CSR space. All register accesses are completed in 2 clock cycles to support an APB port. The GMAC-AHB default and optional signals are described in “[Signals](#)” on page [205](#).

3.2.2.1 AHB Slave Port Timing

[Figure 3-8](#) shows a register write access (to address 0x001C) followed by a register read access (to address 0x1014) on the AHB Slave port. The waveform is as per the AMBA 2.0 specifications. The AHB slave port completes all read and write transfers in 3 clocks and therefore, hready_o is de-asserted for 2 clocks for any register access. The AHB slave port supports burst operations and the timing response remains the same (3 clocks for completion of any read/write cycle).

Figure 3-8 Register Write Access Followed by a Register Read Access on the AHB Slave Port



3.3 AXI Application Host Interface

In the GMAC-AXI core, the DMA Controller interfaces with the Host through the AMBA 3 AXI Interface. The AMBA 3 AXI bus interface provides the characteristics to support highly effective data traffic throughput. The System bus utilization is maximized by allowing simultaneous Read and Write transactions.

3.3.1 AXI Master Interface

The AXI Master interface allows multiple numbers of burst requests to be queued for Read/Write operation. System bus utilization can be further enhanced through by supporting Requests Re-Ordering and Data-Interleaving which allows Master the flexibility to select multiple slaves dynamically and independently to handle Priority Requests and Slow peripheral Slaves. Requests Re-ordering and Data Interleaving in AXI Write channel is not supported by this AXI Master in this release. It also supports low power interface defined in the AXI specifications.

3.3.1.1 Burst Splitting and Burst Selection

The GMAC-AXI splits the DMA requests into multiple bursts on the AXI System Bus. The splitting is based on the DMA Count, software-controllable burst enables bits (UNDEF, BLEN256, BLEN128, BLEN64, BLEN32, BLEN16, BLEN8, BLEN4), and software-controllable burst types (INCR and INCR_ALIGNED). The GMAC-AXI also takes care of splitting the transfers that cross 4KB address boundary. The Burst Length Select Priority is in the following sequence: UNDEF, BLEN256, BLEN128, BLEN64, BLEN32, BLEN16, BLEN8, and BLEN4.

3.3.1.2 INCR Burst Type

If the UNDEF is enabled, then GMAC-AXI always chooses the maximum allowed burst length based on bits BLEN256, BLEN128, BLEN64, and BLEN32 (possible maximum burst lengths are 256, 128, 64, 32 or 16).

In cases where the DMA requests are not multiples of the maximum allowed burst length or because of "4K address boundary crossing" condition, the AXI chooses a burst-length of any value less than the maximum enabled burst-length (all lesser burst-length enables are redundant). For example, when only BLEN64, BLEN16, and BLEN4 are enabled and the DMA requests a burst transfer of size 102 beats, then the AXI splits it into two bursts of size 64 and 38 beats respectively.

If UNDEF is not enabled, then the burst length is based on the priority of the enabled bits in the following order – BLEN256, BLEN128, BLEN64, BLEN32, BLEN16, BLEN8, and BLEN4. When the DMA requests to transfer a burst, the AXI interface splits the requested bursts into multiple transfers by using only the enabled burst lengths. This splitting can occur either because the requested burst is not a multiple of the maximum enabled burst or because of 4K address boundary crossing. If AXI interface cannot choose any of the enabled burst lengths, then it selects the burst length as 1. For example, when only BLEN64, BLEN16, and BLEN4 are enabled and the DMA requests a burst transfer of size 102 beats, then the AXI splits it into multiple bursts of size 64, 16, 16, 4, 1 and 1 beats respectively. The sequence is in decreasing burst sizes.

3.3.1.3 INCR_ALIGNED Burst Type

When the Address-aligned burst-type is enabled, then in addition to the burst splitting conditions explained in [INCR Burst Type](#), the AXI interface splits the DMA requested bursts such that each burst-size is aligned to the start address least significant bits. The AXI interface initially generates smaller bursts so that the remaining transfers can be transferred with the maximum possible (enabled) fixed burst lengths.

For example, in the same setting as explained above for UNDEF (BLEN64, BLEN16 and BLEN4 are enabled), DMA requests a burst of size 102 beats at the start address of 0x0000_0164 (32-bit bus). The AXI starts the first transfer with size 23 such that the address of the next burst is aligned (0x0000_0400) for a burst of 64. Therefore, the sequence of bursts would be 23, 64, and 17 respectively.

When UNDEF is not set, then in same scenario, the sequence of burst transfers would be 1, 1, 1, 4, 16, 64, 16, and 1 respectively. The sequence of smaller bursts at the beginning is to align the address to the next higher enabled burst-lengths programmed in the register.

3.3.1.4 Outstanding Transactions

The GMAC-AXI supports up to four, eight, or sixteen (user-configurable in coreConsultant) Read/Write outstanding requests on the AXI bus. You can also control these outstanding requests through software by programming the WR_OSR_LMT/RD_OSR_LMT bits in the AXI Bus Mode Register. This is a useful work around for any system-level issues with the handling of multiple requests.



The maximum number of outstanding requests is divided equally between the requests generated by the two internal masters (Rx DMA and Tx DMA). This means that when you select 8 outstanding AXI requests during RTL configuration, then each DMA is limited to a maximum of 4 outstanding requests in each read or write channel. When you reduce the outstanding request, for example to 4, by programming the respective control bits, then all the outstanding requests can be from the same DMA.

3.3.1.5 Priority of AXI Requests

The Descriptor transfers have higher priority than the Data transfers. Therefore, if there are two requests, Rx Descriptor Read and TX Data Read, then Rx Descriptor Read is given higher priority so that the next Rx Data Write (subsequent to Rx Descriptor Read) need not wait for the Tx Data Read transfer to complete. If

there are descriptor read requests from both DMAs, then these requests are serviced based on a first-come first-serve basis. Rx DMA has higher priority if descriptor read requests are generated from both DMAs in the same clock. Similarly, in the write channel, descriptor writes from any DMA have higher priority than the data-write transfers for the Rx DMA.

When you enable the AV feature, the DMA contains the multiple channels. If the requests are generated from different channels simultaneously in the same clock, then the request priority among the channels is: Channel 2 (if present), Channel 1, and Channel 0.

3.3.1.6 Bursts Reordering and Data Interleaving

The AXI protocol allows re-ordering of data transfers with different AXI-IDs with respect to the sequence of requests. It also allows data interleaving between transfers of different AXI-IDs for both Read and Write channels.

In GMAC-AXI, requests from each internal master (RxDMA and TxDMA) are generated with different AXI-ID. Therefore, reordering and data interleaving can be performed as the two DMA operate independently and are allocated different address-space on the slave memory.

The GMAC-AXI supports reordering and interleaving from the AXI slaves on the Read channel. Each DMA internally ensures that a read request and write request to the same address (can occur for only descriptor accesses) is never generated at the same time. The Tx DMA ensures that Tx Descriptor read, Tx Data read, and Tx Descriptor write-back requests are generated only after the previous requests are completed. And since the Rx DMA generates only Descriptor reads on the read channel with a different ID, data re-ordering, and interleaving on the read channel does not cause any problems in the core.

On the write channel, the data re-ordering or interleaving can happen between Tx Descriptor write-back and Rx Data write requests or between Tx Descriptor write-back and Rx Descriptor write-back requests.



Note In the current release of the GMAC-AXI, write requests reordering and data-interleaving on the write channel are not supported.

3.3.1.7 AXI-ID Translation

The GMC-AXI selects unique IDs for each DMA channel and also for each Read/Write DMA in a channel. The GMC-AXI selects the IDs as follows:

- ❖ GMAC-AXI Master ID [0] = 0
Rx DMA access for Read and Write.
- ❖ GMAC-AXI Master ID[0] = 1
Tx DMA access for Read and Write.
- ❖ GMAC-AXI Master ID[2:1] = 0, 1, or 2
Indicates the DMA channel number when multiple DMA channels are present, that is, when you select the AV feature. Otherwise, it is hardwired to zeros.



The AXI system bus ID width is configurable. The Master default width is 4-bit. The unused bits are hardwired to 0s. For the AXI Slave interface, the default ID width is 8.

3.3.1.8 Endianess Support

The GMAC-AXI supports little-endian mode and byte-invariant big-endian mode on the AXI master ports. The AXI master does not support narrow transfers and always performs full data-width transfers. For more information about the supported endian modes, see “[Endian Support](#)” on page [563](#).

3.3.1.9 Posted Writes

In posted writes, the AXI master write channel transfers an OKAY response to the DMA as soon as the last cycle or beat of data is accepted by the AXI interconnect. The AXI master sends this response without waiting for the response from the target AXI slave on the write response channel. In non-posted writes, the AXI master interface transfers the response received from the AXI slave channel to the DMA.

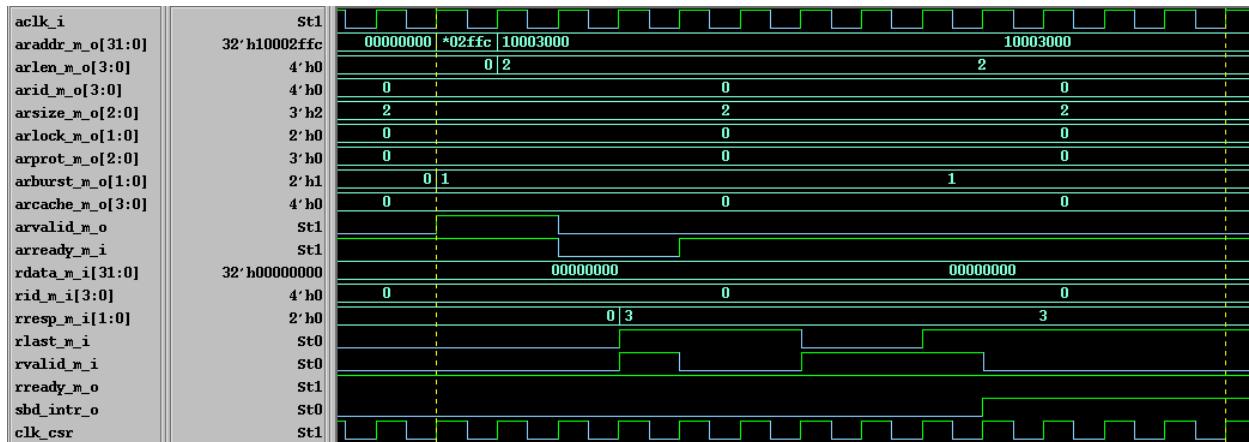
For transferring the Ethernet data to the system memory, the RxDMA always issues posted write requests to the AXI master interface. This enables pipelining of the data requests without delays. GMAC-AXI does not support out-of-order write transfers. However, the RxDMA ensures that the sequence of descriptor writes, descriptor reads, and data transfers during a frame is maintained.

For writing descriptors (status or timestamp), the DMA always issues non-posted write requests. This is needed because the “transfer complete interrupt” generation is based on the descriptor writes for which it needs completion response from the memory slave. This guarantees no race condition, interrupt is guaranteed to be generated only after the data and descriptor is written in to the slave memory.

3.3.1.10 Error Response Handling

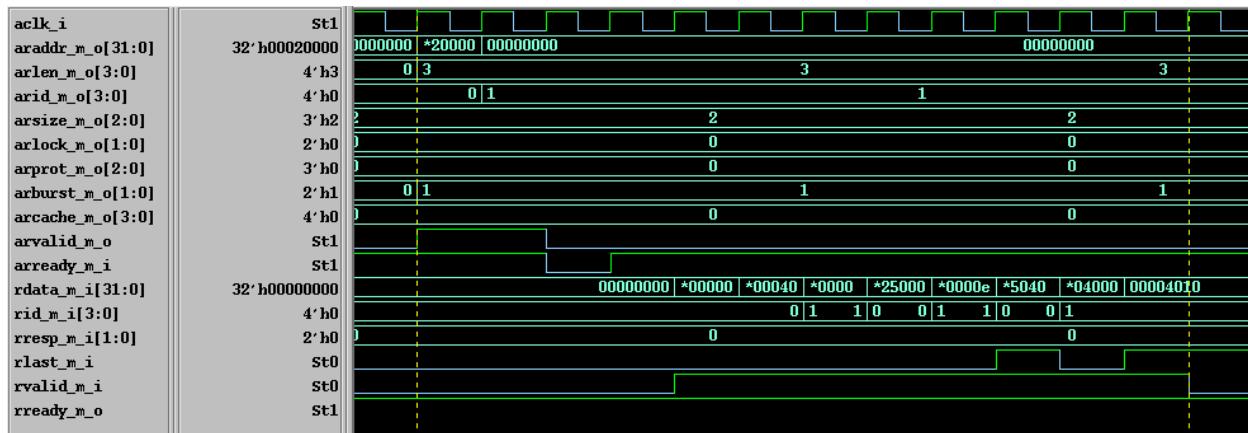
Whenever there is an error response from the AXI Slave (as shown in [Figure 3-9](#) with $rresp_m_i[1:0] = 3$), the respective DMA channel which generated the request gets disabled. The GMAC-AXI asserts the interrupt (sbd_intr_o in [Figure 3-9](#)) when the corresponding “Fatal Bus Error” interrupt is enabled. The host has to reset the core with hard-reset or soft-reset to restart the DMA.

Figure 3-9 Error Response Timing



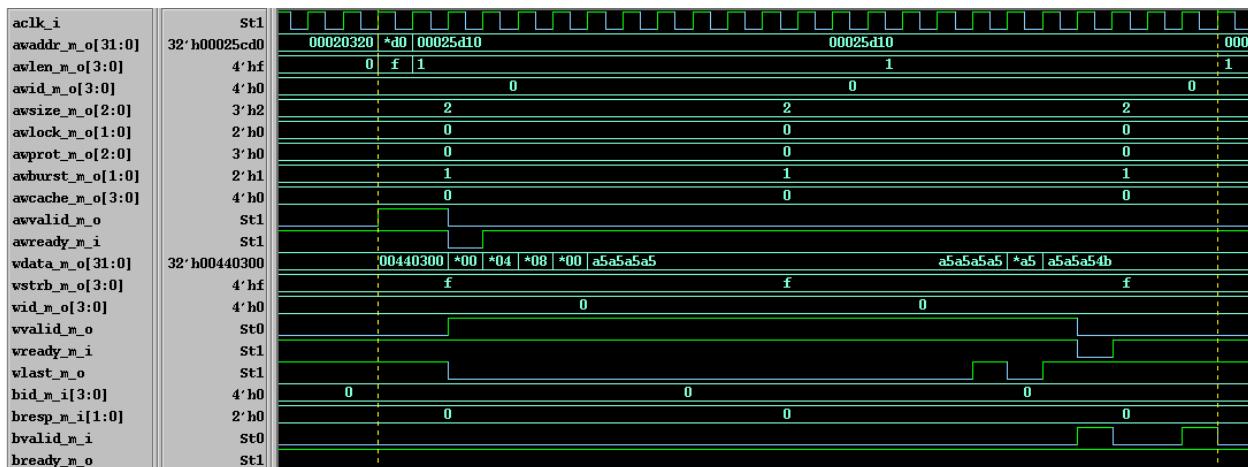
3.3.1.11 AXI Memory Read

[Figure 3-10](#) shows the AXI Master read timings. Note that there are two requests placed for ID 0 and ID 1. The read data is interleaved by the interconnect (by way of ID 0 and ID 1) and received by the GMAC-AXI. As the TxDMA in GMAC-AXI always ensures that it has enough space in its FIFO to accept the requested burst of data, the $r_ready_m_o$ response is always high.

Figure 3-10 AXI Memory Read Timing

3.3.1.12 AXI Memory Write

Figure 3-11 shows the AXI write interface diagram. There are two requests for the ID 0 and are issued with burst lengths of 16 and 2 respectively. The latency between the first write request and the corresponding first data is two clocks. As the RxDMA ensures that it always has the request burst of data available in its FIFO before the request is generated, there is no further latencies or delays in the data transfer. As you can see in the diagram, the complete 18 beats of write data of the two requests is transferred in 18 clocks (provided there is no delay in the acceptance of that data by the AXI slave).

Figure 3-11 AXI Memory Write Timing

3.3.1.13 AXI Early Burst Termination

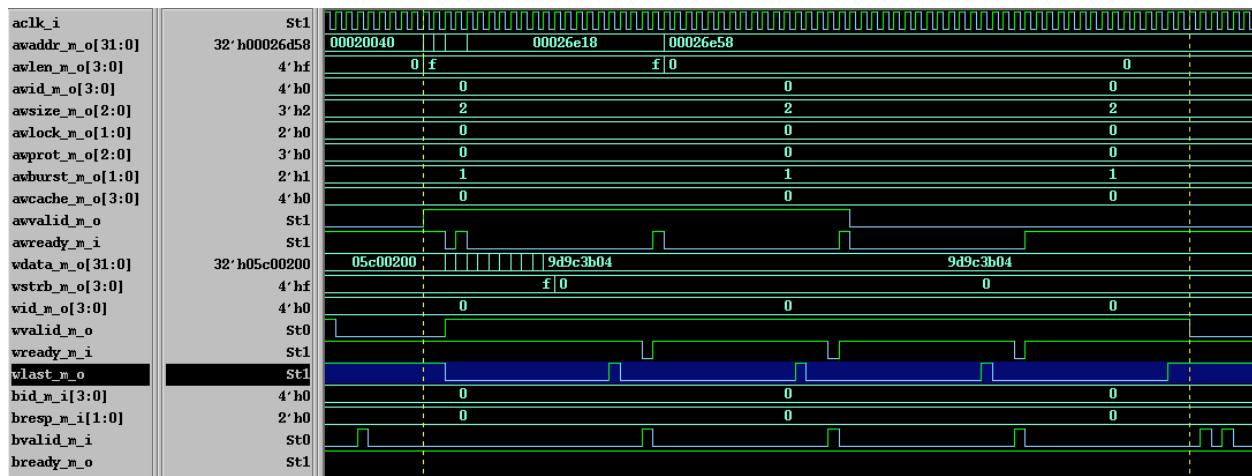
Figure 3-12 shows the early burst termination on the write channel. This can happen during data transfer when the end-of-frame is read out of the RxFIFO before the requested burst transfer is completed. There are 5 requests issued but the EOF gets transferred in middle of the first burst. The GMAC-AXI still continues the data phases of the outstanding requested bursts but does not allow further data to be written to the slave memory by de-asserting the write-strobes (as indicated by wstrb_m_o[3:0] = 0).

You can observe that there are two extra request commands (@address 0x00026e18 and 0x00026e58) active after the write-strobes are de-asserted after EOF transfer. The first one had been already issued to slave and hence is not changed. The second one is a dummy request with single beat required to generate a last data transfer (with wlast high).



Considering the fact that there can be lot of wastage of AXI bandwidth with dummy writes after transfer of EOF data to host memory, it is recommended to have maximum of 2 or less outstanding requests on AXI master write channel by programming the AXI Bus mode register appropriately.

Figure 3-12 AXI Early Burst Termination Timing



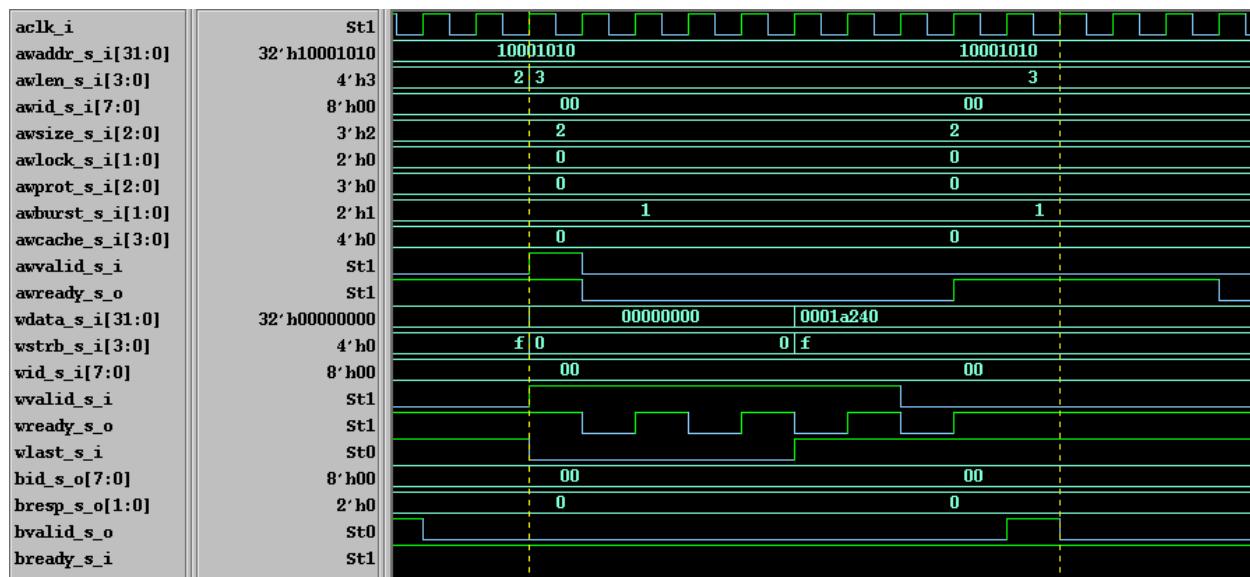
3.3.2 AXI Slave Interface

The GMAC-AXI supports an AXI-Slave Interface to access the CSR register sets. The features supported for the AXI slave interface are listed in “[AXI Slave Interface Features](#)” on page 28.

The slave Interface has the following known limitations

- ❖ No support for data reordering.
- ❖ No support for data interleaving.
- ❖ No support for WRAP burst transfers.
- ❖ No support for awcache/awprot or arcache/arprot.
- ❖ No support for atomic accesses.
- ❖ No support for Exclusive Access.

[Figure 3-13](#) shows the timing relationship on the AXI Slave interface for a burst-write transfer.

Figure 3-13 32-bit AXI Slave Interface's Write Channel Timing

The AXI-slave interface receives a write burst of length 4 which gets accepted in 7 clocks. The respective write response takes an additional 3 clocks. Similarly a single write transfer gets completed in 4 clocks as shown in [Figure 3-14](#).

The AXI-slave interface supports all types of byte/half-word/word burst-size transfers. The latencies may vary depending on the address offset and data-bus width, as shown in [Figure 3-15](#) for a 64-bit AXI Slave interface.

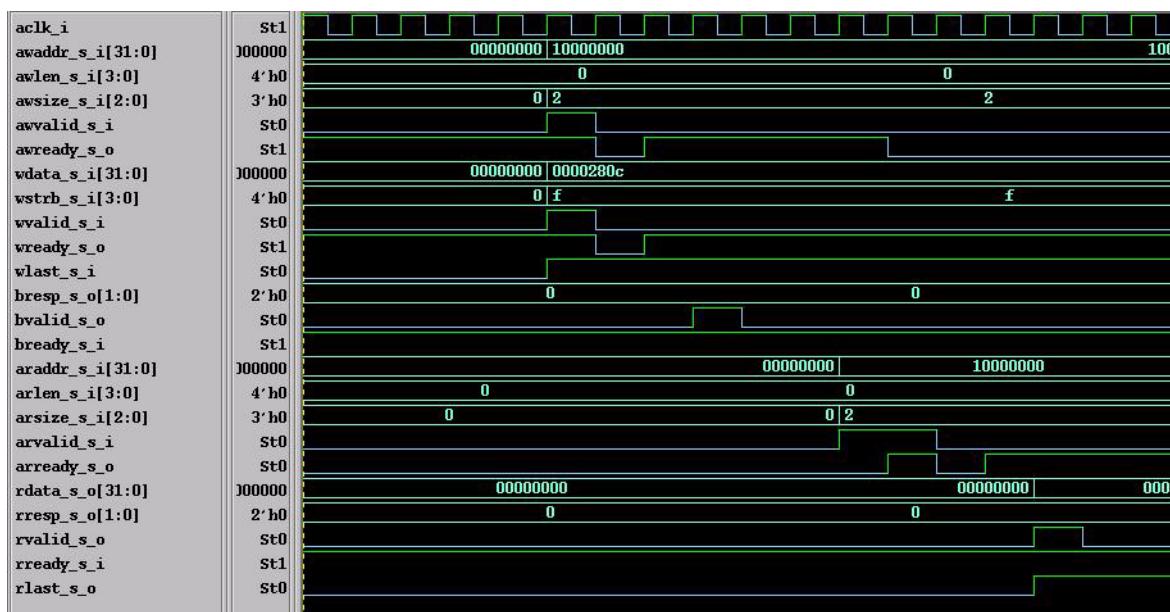
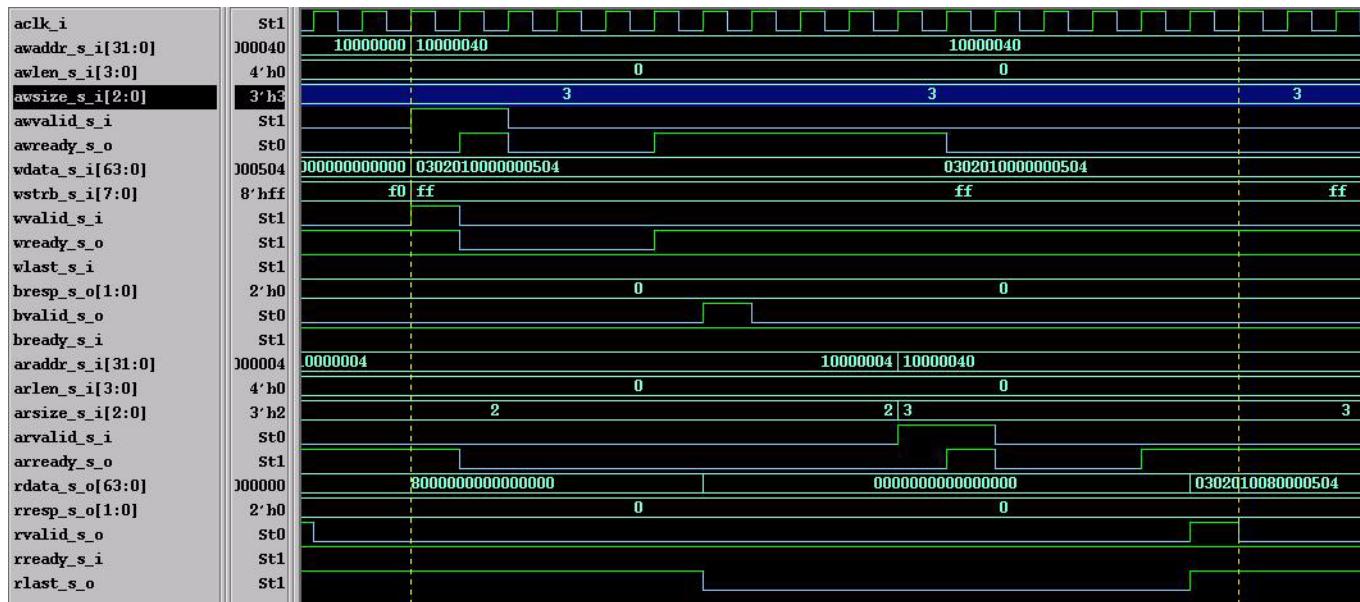
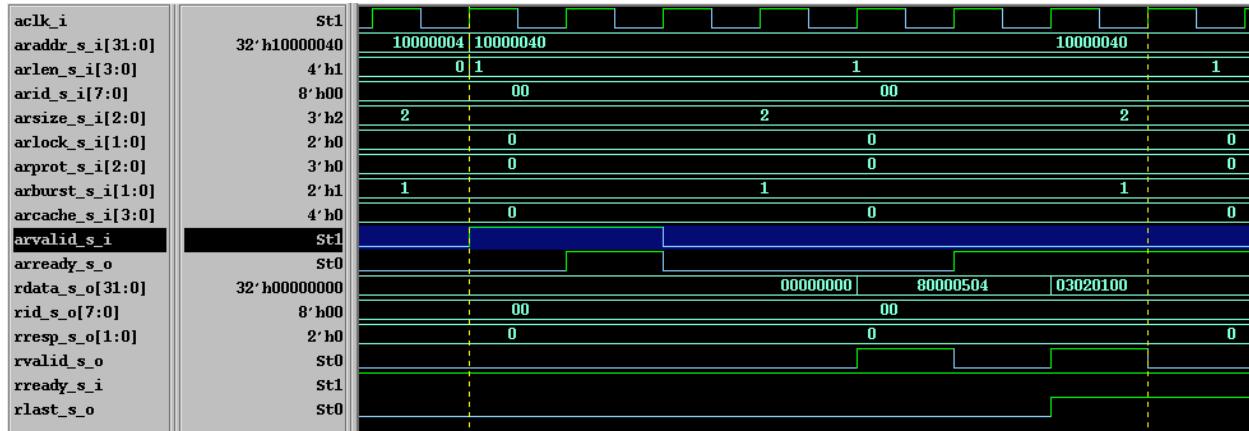
Figure 3-14 Single Write and Read Transfer

Figure 3-15 Latencies for a 64-bit AXI Slave Interface

In Figure 3-16, the AXI slave receives a read request of burst length 2. The latency for the first data to be output on the AXI bus is 4 clocks while the remaining beats are output 2 clocks after the previous beat is accepted by the AXI master. In the AXI-read channel, the latencies depend on the burst-size, data-bus width and the address offset.

Figure 3-16 AXI Slave Receives a Read Request of Burst Length 2

3.3.2.1 Endianess Support

The GMAC-AXI supports little-endian mode and byte-invariant big-endian mode on the AXI slave ports. For more information about the supported endian modes, see “[Endian Support](#)” on page 563.

3.3.3 AXI Low Power Interface

The GMAC-AXI core support the low-power interface defined by AXI specifications. The bits[31:30] of the AXI Bus Mode Register (see “[Register 10 \(AXI Bus Mode Register\)](#)” on page 313) control the behavior of the GMAC-AXI in the low-power mode. When the EN_LPI bit is enabled and the host requests a low-power initiation, the GMAC-AXI checks for the IDLE status of the data-paths before giving the response. In other

words, when both the MAC transmitter and Receiver are in IDLE, both DMA engines are in idle and there is no data in the Transmit or Receive data-path/FIFOs for a continuous period of 16 clock-cycles (aclk_i), then it accepts the low-power request and goes into low-power mode in which the host can remove the AXI-clock. If the GMAC goes to an inactive state within 15 clocks after the request, then the GMAC-AXI waits for confirmation of the inactive state for a further period of 16 clocks before giving the response. If some activity is found in the GMAC during this period of 16 clock cycles, it responds by denying the low-power request.

In the low-power mode, GMAC-AXI can initiate a request to come out of low-power mode when any packet is being received by the MAC Receiver or when a magic packet/remote-wake-up packet is received (as controlled by Bit[30] of the AXI Bus mode register). This request signal assertion (cactive_o) is synchronous to clk_rx_i in such cases, as aclk_i is assumed to be absent.



Note The timing and behavior of the GMAC-AXI low-power interface signals is according to the Figure 12-2 of the [AMBA 3 Specification](#).

3.4 DMA Controller

The DMA has independent Transmit and Receive engines, and a CSR space. The Transmit engine transfers data from system memory to the device port (MTL), while the Receive engine transfers data from the device port to the system memory. The controller uses descriptors to efficiently move data from source to destination with minimal Host CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the Host CPU for situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions.

The DMA and the Host driver communicate through two data structures:

- ❖ Control and Status registers (CSR)
- ❖ Descriptor lists and data buffers

Control and Status registers are described in detail in “[Registers](#)” on page [281](#). Descriptors are described in detail in “[Descriptors](#)” on page [397](#).



Starting with Release 3.30a, you can select an alternative descriptor structure during RTL configuration. The control bits in this descriptor structure are reassigned so that the application can use a larger buffer size (8 KB). A detailed bit map of this alternative descriptor structure is provided in “[Descriptors](#)” on page [397](#). All descriptions in [Section 3.4](#) refer to the default descriptor structure, not this new alternative. If you are using the alternate descriptor structure, ignore the descriptor-specific mapping in [Section 3.4](#) and refer to the alternate descriptor-specific bit maps.

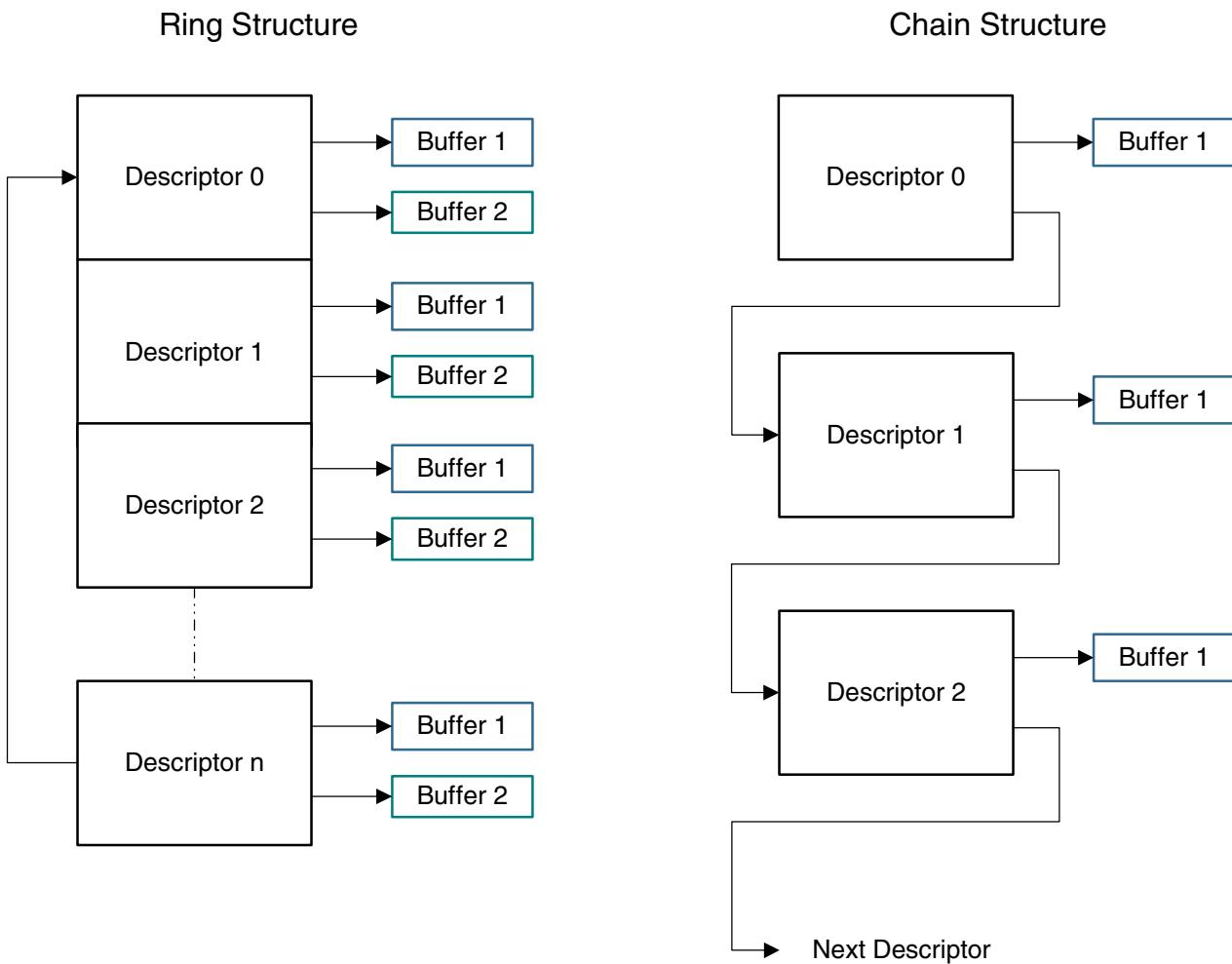
The DMA transfers data frames received by the core to the Receive Buffer in the Host memory, and Transmits data frames from the Transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers.

There are two descriptor lists; one for reception, and one for transmission. The base address of each list is written into DMA Registers 3 and 4, respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both Receive and Transmit descriptors (RDES1[24] and TDES1[24]). The descriptor lists resides in the Host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the Host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA skips to the next frame buffer when end-of-frame is detected. Data chaining can be enabled or disabled.

The descriptor ring and chain structure is shown in [Figure 3-17](#).

Figure 3-17 Descriptor Ring and Chain Structure



3.4.1 Initialization

Initialization for the GMAC is as follows.

1. Write to DMA Register 0 to set Host bus access parameters.
2. Write to DMA Register 7 to mask unnecessary interrupt causes.
3. The software driver creates the Transmit and Receive descriptor lists. Then it writes to both DMA Register 3 and DMA Register 4, providing the DMA with the starting address of each list.
4. Write to GMAC Registers 1, 2, and 3 for desired filtering options.

5. Write to GMAC Register 0 to configure the operating mode and enable the transmit operation (bit 3: Transmitter Enable). The PS and DM bits are set based on the auto-negotiation result (read from the PHY).
6. Write to DMA Register 6 to set bits 13 and 1 to start transmission and reception.
7. Write to GMAC Register 0 to enable the Receive operation (bit 2: Receiver Enable).

The Transmit and Receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The Receive and Transmit engines then begin processing Receive and Transmit operations. The Transmit and Receive processes are independent of each other and can be started or stopped separately.

3.4.1.1 Host Bus Burst Access

The DMA attempts to execute fixed-length Burst transfers on the AHB/AXI Master interface if configured to do so (FB bit of DMA Register 0). The maximum Burst length is indicated and limited by the PBL field (DMA Register 0[13:8]). The Receive and Transmit descriptors are always accessed in the maximum possible (limited by PBL or $16 * 8/\text{bus width}$) burst-size for the 16-bytes to be read.

The Transmit DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in MTL Transmit FIFO or the number of bytes till the end of frame (when it is less than the configured burst-length). The DMA indicates the start address and the number of transfers required to the AHB/AXI Master Interface. When the AHB/AXI Interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise (no fixed-length burst), it transfers data using INCR (undefined length) and SINGLE transactions.

The Receive DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in MTL Receive FIFO or when the end of frame (when it is less than the configured burst-length) is detected in the Receive FIFO. The DMA indicates the start address and the number of transfers required to the AHB/AXI Master Interface. When the AHB/AXI Interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. If the end-of frame is reached before the fixed-burst ends on the AHB/AXI interface, then dummy transfers are performed in-order to complete the fixed-burst. Otherwise (FB bit of DMA Register 0 is reset), it transfers data using INCR (undefined length) and SINGLE transactions.

When the AHB/AXI interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer the AHB/AXI initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for $\text{PBL} > 16$), because the AHB/AXI interface does not support more than INCR16.

3.4.1.2 Host Data Buffer Alignment

The Transmit and Receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers with address aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame.

Example 3-1 Buffer Read

If the Transmit buffer address is 32'h00000FF2 (for 32-bit data bus), and 15 bytes need to be transferred, then the DMA reads five full words from address 32'h00000FF0, but when transferring data to the MTL Transmit FIFO, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last

transfer are also ignored. The DMA always ensures it transfers a full 32-bit data to the MTL Transmit FIFO, unless it is the end-of-frame.

Example 3-2 Buffer Write

If the Receive buffer address is 32'h0000FF2 (for 64-bit data bus) and 16 bytes of a received frame need to be transferred, then the DMA writes 3 full words from address 32'h00000FF0. But the first 2 bytes of first transfer and the last 6 bytes of the third transfer have dummy data.

3.4.1.3 Buffer Size Calculations

The DMA does not update the size fields in the Transmit and Receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES1) towards the GMAC core. If a descriptor is marked as first (FS bit of TDES1 is set), then the DMA marks the first transfer from the buffer as the start of frame. If a descriptor is marked as last (LS bit of TDES1), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL.

The Receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0[29:16]) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The Receive DMA always transfers the start of next frame with a new descriptor.



Note Even when the start address of a receive buffer is not aligned to the system bus's data width, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1,024-byte (1 KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the Receive descriptor to have a 0x1002 offset. The Receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual useful space in this buffer is 1,022 bytes, even though the buffer size is programmed as 1,024 bytes, because of the start address offset.

3.4.1.4 DMA Arbiter for GMAC-DMA and GMAC-AHB Cores

The arbiter inside the DMA module performs the arbitration between the Transmit and Receive channel accesses to the AHB Master interface. Two types of arbitrations are possible: round-robin, and fixed-priority.

When round-robin arbitration is selected (DA bit of [Register 0 \(Bus Mode Register\)](#) is reset), the arbiter allocates the data bus in the ratio set by the PR bits of DMA Register 0, when both Transmit and Receive DMAs are requesting for access simultaneously. When the DA bit is set, the Receive DMA always gets priority over the Transmit DMA for data access by default. When the TXPR bit (bit 27 of DMA register 0) is also set, then the Transmit DMA gets priority over the Receive DMA as explained in [Table 4-12](#) on page [135](#).

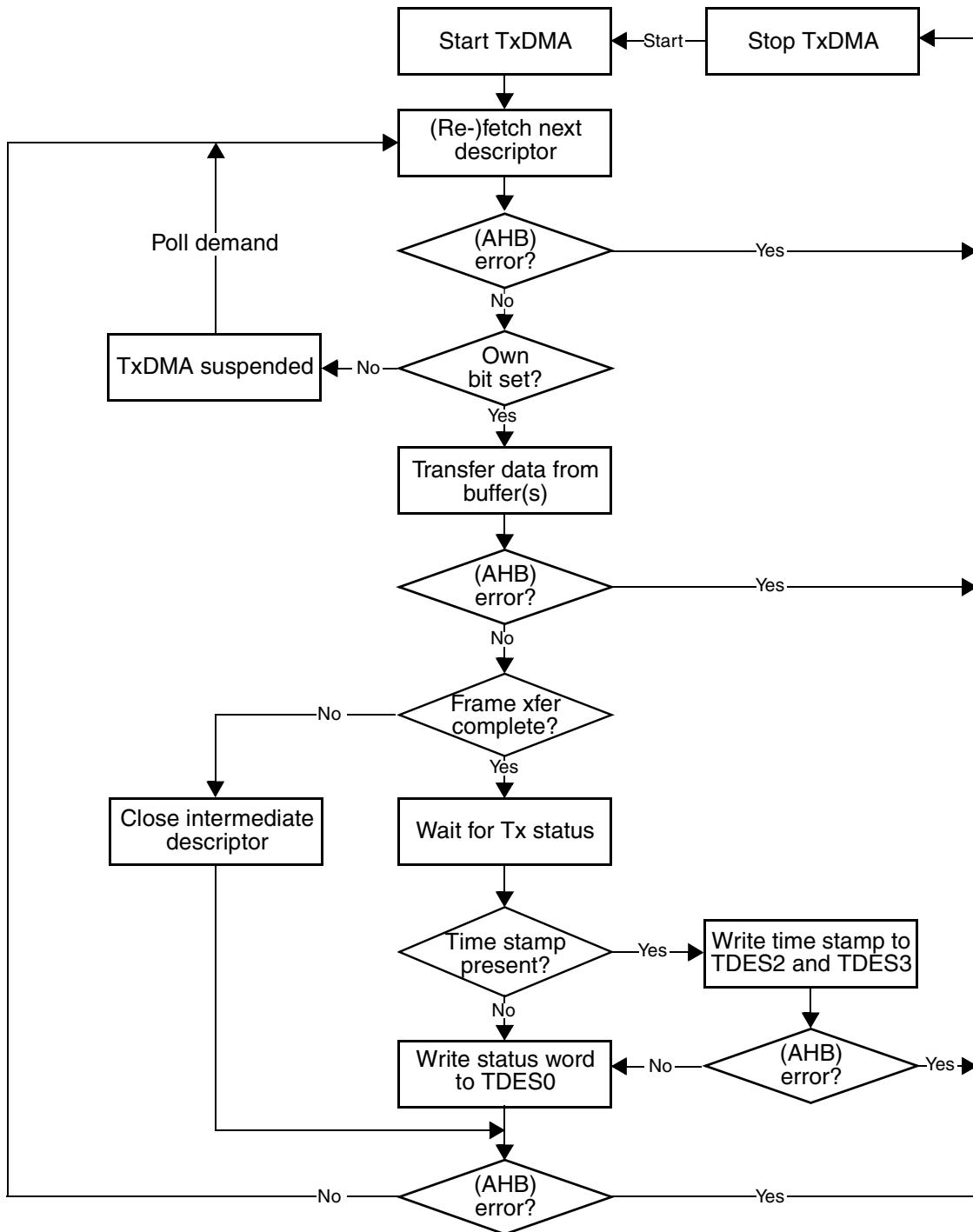
3.4.2 Transmission

3.4.2.1 TxDMA Operation: Default (Non-OSF) Mode

The Transmit DMA engine in default mode proceeds as follows:

1. The Host sets up the transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Frame data.
2. Once the ST bit (DMA Register 6[13]) is set, the DMA enters the Run state.
3. While in the Run state, the DMA polls the Transmit Descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the Host, or if an error condition occurs, transmission is suspended and both the Transmit Buffer Unavailable (DMA Register 5[2]) and Normal Interrupt Summary (DMA Register 5[16]) bits are set. The Transmit Engine proceeds to [Step 9](#).
4. If the acquired descriptor is flagged as owned by DMA (TDES0[31] = 1'b1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
5. The DMA fetches the Transmit data from the Host memory and transfers the data to the MTL for transmission.
6. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps [3](#), [4](#), and [5](#) are repeated until the end-of-Ethernet-frame data is transferred to the MTL.
7. When frame transmission is complete, if IEEE 1588 time stamping was enabled for the frame (as indicated in the transmit status) the timestamp value obtained from MTL is written to the transmit descriptor (TDES2 and TDES3) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES0). Because the Own bit is cleared during this step, the Host now owns this descriptor. If time stamping was not enabled for this frame, the DMA does not alter the contents of TDES2 and TDES3.
8. Transmit Interrupt (DMA Register 5[0]) is set after completing transmission of a frame that has Interrupt on Completion (TDES1[31]) set in its Last Descriptor. The DMA engine then returns to [Step 3](#).
9. In the Suspend state, the DMA tries to re-acquire the descriptor (and thereby return to [Step 3](#)) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared.

The TxDMA transmission flow in default mode is shown in [Figure 3-18](#).

Figure 3-18 TxDMA Operation in Default Mode

3.4.2.2 TxDMA Operation: OSF Mode

While in the Run state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first (if the OSF bit is set in DMA Register 6[2]). As the transmit process finishes transferring the first frame, it immediately polls the Transmit Descriptor list for the second frame. If the

second frame is valid, the transmit process transfers this frame before writing the first frame's status information.

In OSF mode, the Run state Transmit DMA operates in the following sequence:

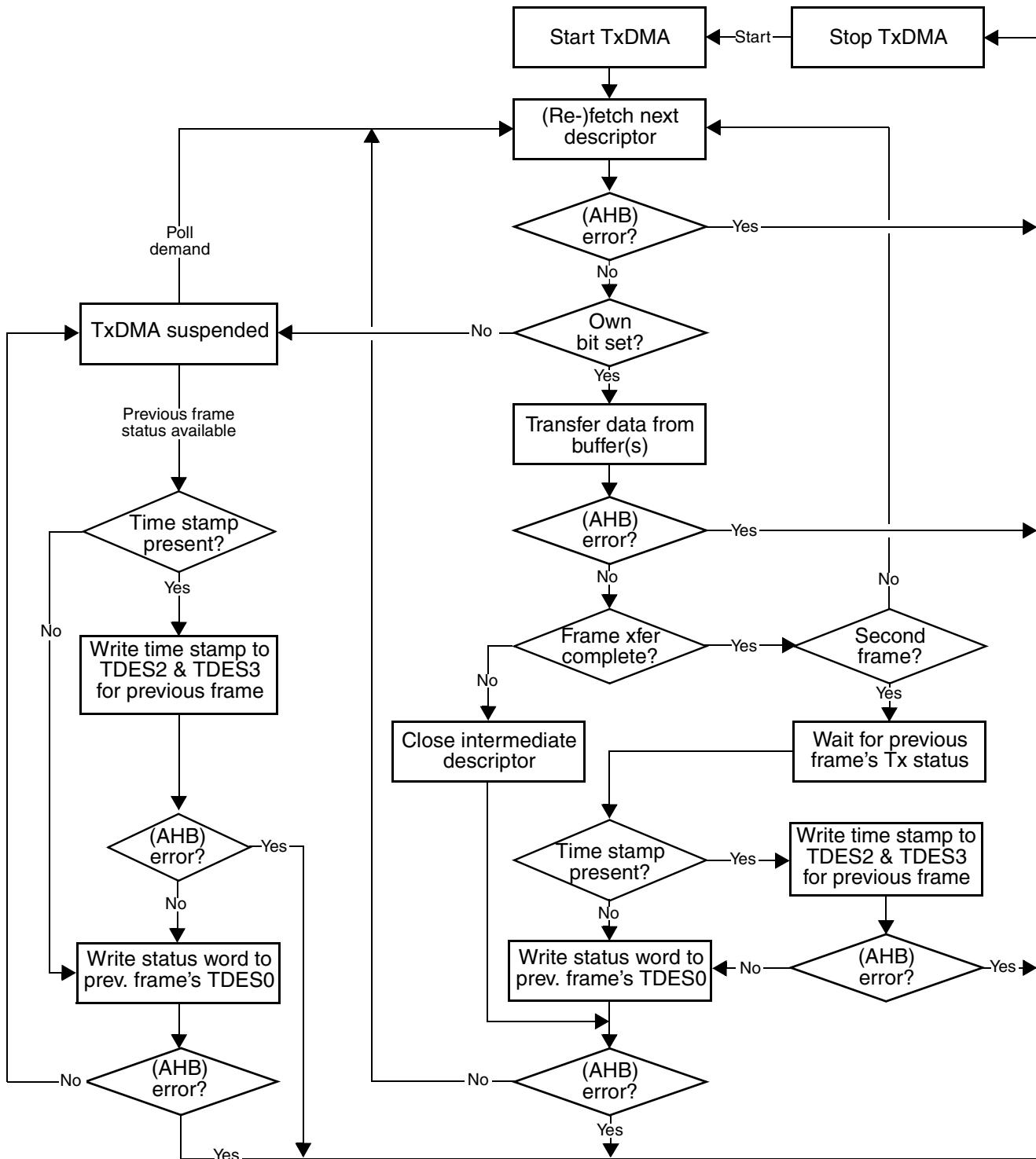
1. The DMA operates as described in [steps 1–6](#) of the TxDMA (default mode).
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and skips to [Step 7](#).
4. The DMA fetches the Transmit frame from the Host memory and transfers the frame to the MTL until the End-of-Frame data is transferred, closing the intermediate descriptors if this frame is split across multiple descriptors.
5. The DMA waits for the previous frame's frame transmission status and time stamp. Once the status is available, the DMA writes the time stamp to TDES2 and TDES3, if such time stamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared Own bit, to the corresponding TDES0, thus closing the descriptor. If time stamping was not enabled for the previous frame, the DMA does not alter the contents of TDES2 and TDES3.
6. If enabled, the Transmit interrupt is set, the DMA fetches the next descriptor, then proceeds to [Step 3](#) (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode ([Step 7](#)).
7. In Suspend mode, if a pending status and time stamp are received from the MTL, the DMA writes the time stamp (if enabled for the current frame) to TDES2 and TDES3, then writes the status to the corresponding TDES0. It then sets relevant interrupts and returns to Suspend mode.
8. The DMA can exit Suspend mode and enter the Run state (go to [Step 1](#) or [Step 2](#) depending on pending status) only after receiving a Transmit Poll demand (DMA Register 1).



Note As the DMA fetches the next descriptor in advance before closing the current descriptor, the descriptor chain should have more than 2 different descriptors for correct and proper operation.

The basic flow is described in [Figure 3-19](#).

Figure 3-19 TxDMA Operation in OSF Mode



3.4.2.3 Transmit Frame Processing

The Transmit DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and FCS fields. The DA, SA, and Type/Len fields contain valid data. If the Transmit Descriptor

indicates that the MAC core must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes.

Frames can be data-chained and can span several buffers. Frames must be delimited by the First Descriptor (TDES1[29]) and the Last Descriptor (TDES1[30]), respectively.

As transmission starts, the First Descriptor must have (TDES1[29]) set. When this occurs, frame data transfers from the Host buffer to the MTL Transmit FIFO. Concurrently, if the current frame has the Last Descriptor (TDES1[30]) clear, the Transmit Process attempts to acquire the Next Descriptor. The Transmit Process expects this descriptor to have TDES1[29] clear. If TDES1[30] is clear, it indicates an intermediary buffer. If TDES1[30] is set, it indicates the last buffer of the frame.

After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the Transmit Descriptor 0 (TDES0) word of the descriptor that has the last segment set in Transmit Descriptor 1 (TDES1[30]). At this time, if Interrupt on Completion (TDES1[31]) was set, Transmit Interrupt (DMA Register 5[0]) is set, the Next Descriptor is fetched, and the process repeats.

The actual frame transmission begins after the MTL Transmit FIFO has reached either a programmable transmit threshold (DMA Register 6[16:14]), or a full frame is contained in the FIFO. There is also an option for Store and Forward Mode (DMA Register 6[21]). Descriptors are released (Own bit TDES0[31] clears) when the DMA finishes transferring the frame.



Note To ensure proper transmission of a frame and the next frame, you must specify a non-zero buffer size for the transmit descriptor that has the Last Descriptor (TDES1[30]) set.

3.4.2.4 Transmit Polling Suspended

Transmit polling can be suspended by either of the following conditions:

- ❖ The DMA detects a descriptor owned by the Host (TDES0[31]=0). To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command.
- ❖ A frame transmission is aborted when a transmit error because of underflow is detected. The appropriate Transmit Descriptor 0 (TDES0) bit is set.

If the second condition occur, both Abnormal Interrupt Summary (DMA Register 5[15]) and Transmit Underflow bits (DMA Register 5 [5]) are set, and the information is written to Transmit Descriptor 0, causing the suspension. If the DMA goes into SUSPEND state because of the first condition, then both Normal Interrupt Summary (DMA Register 5 [16]) and Transmit Buffer Unavailable (DMA Register 5 [2]) are set.

In both cases, the position in the Transmit List is retained. The retained position is that of the descriptor following the Last Descriptor closed by the DMA.

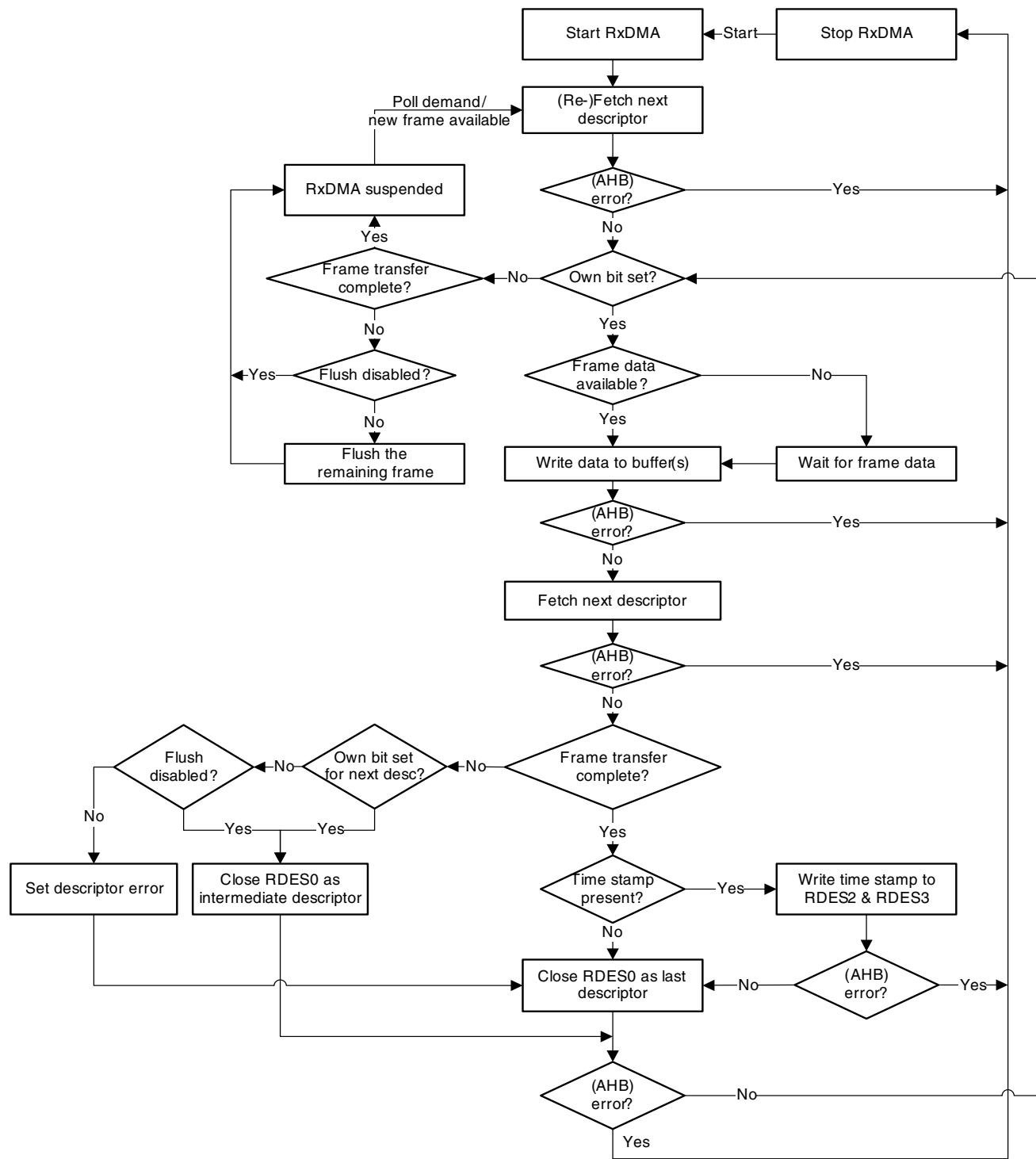
The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension cause.

3.4.3 Reception

The Receive DMA engine's reception sequence is depicted in [Figure 3-20](#) and proceeds as follows:

1. The host sets up Receive descriptors (RDES0-RDES3) and sets the Own bit (RDES0[31]).
2. Once the SR (DMA Register 6[1]) bit is set, the DMA enters the Run state. While in the Run state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the host), the DMA enters the Suspend state and jumps to [Step 9](#).
3. The DMA decodes the receive data buffer address from the acquired descriptors.

4. Incoming frames are processed and placed in the acquired descriptor's data buffers.
5. When the buffer is full or the frame transfer is complete, the Receive engine fetches the next descriptor.
6. If the current frame transfer is complete, the DMA proceeds to [Step 7](#). If the DMA does not own the next fetched descriptor and the frame transfer is not complete (EOF is not yet transferred), the DMA sets the Descriptor Error bit in the RDES0 (unless flushing is disabled). The DMA closes the current descriptor (clears the Own bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value (marks it as Last Descriptor if flushing is not disabled), then proceeds to [Step 8](#). If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to [Step 4](#).
7. If IEEE 1588 time stamping is enabled, the DMA writes the timestamp (if available) to the current descriptor's RDES2 and RDES3. It then takes the receive frame's status from the MTL and writes the status word to the current descriptor's RDES0, with the Own bit cleared and the Last Segment bit set.
8. The Receive engine checks the latest descriptor's Own bit. If the host owns the descriptor (Own bit is 1'b0) the Receive Buffer Unavailable bit (Register 5[7]) is set and the DMA Receive engine enters the Suspended state ([Step 9](#)). If the DMA owns the descriptor, the engine returns to [Step 4](#) and awaits the next frame.
9. Before the Receive engine enters the Suspend state, partial frames are flushed from the Receive FIFO (You can control flushing using Bit 24 of DMA Register 6).
10. The Receive DMA exits the Suspend state when a Receive Poll demand is given or the start of next frame is available from the MTL's Receive FIFO. The engine proceeds to [Step 2](#) and refetches the next descriptor.

Figure 3-20 Receive DMA Operation

The DMA does not acknowledge accepting the status from the MTL until it has completed the time stamp write-back and is ready to perform status write-back to the descriptor.

If software has enabled time stamping through CSR, when a valid time stamp value is not available for the frame (for example, because the receive FIFO was full before the time stamp could be written to it), the

DMA writes all-ones to RDES2 and RDES3. Otherwise (that is, if time stamping is not enabled), the RDES2 and RDES3 remain unchanged.

3.4.3.1 Receive Descriptor Acquisition

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- ❖ The receive Start/Stop bit (Register 6[1]) has been set immediately after being placed in the Run state.
- ❖ The data buffer of current descriptor is full before the frame ends for the current transfer.
- ❖ The controller has completed frame reception, but the current Receive Descriptor is not yet closed.
- ❖ The receive process has been suspended because of a host-owned buffer (RDES0[31] = 0) and a new frame is received.
- ❖ A Receive poll demand has been issued.

3.4.3.2 Receive Frame Processing

The GMAC transfers the received frames to the Host memory only when the frame passes the address filter and frame size is greater than or equal to configurable threshold bytes set for the Receive FIFO of MTL, or when the complete frame is written to the FIFO in Store-and-Forward mode.

If the frame fails the address filtering, it is dropped in the GMAC block itself (unless Receive All GMAC Register 1[31] bit is set). Frames that are shorter than 64 bytes, because of collision or premature termination, can be purged from the MTL Receive FIFO.

After 64 (configurable threshold) bytes have been received, the MTL block requests the DMA block to begin transferring the frame data to the Receive Buffer pointed to by the current descriptor. The DMA sets First Descriptor (RDES0[9]) after the DMA Host Interface (AHB/AXI or MDC) becomes ready to receive a data transfer (if DMA is not fetching transmit data from the host), to delimit the frame. The descriptors are released when the Own (RDES[31]) bit is reset to 1'b0, either as the Data buffer fills up or as the last segment of the frame is transferred to the Receive buffer. If the frame is contained in a single descriptor, both Last Descriptor (RDES[8]) and First Descriptor (RDES[9]) are set.

The DMA fetches the next descriptor, sets the Last Descriptor (RDES[8]) bit, and releases the RDES0 status bits in the previous frame descriptor. Then the DMA sets Receive Interrupt (Register 5[6]). The same process repeats unless the DMA encounters a descriptor flagged as being owned by the host. If this occurs, the Receive Process sets Receive Buffer Unavailable (Register 5[7]) and then enters the Suspend state. The position in the receive list is retained.

3.4.3.3 Receive Process Suspended

If a new Receive frame arrives while the Receive Process is in Suspend state, the DMA refetches the current descriptor in the Host memory. If the descriptor is now owned by the DMA, the Receive Process re-enters the Run state and starts frame reception. If the descriptor is still owned by the host, by default, the DMA discards the current frame at the top of the MTL Rx FIFO and increments the missed frame counter. If more than one frame is stored in the MTL Rx FIFO, the process repeats.

The discarding or flushing of the frame at the top of the MTL Rx FIFO can be avoided by setting Operation Mode register bit 24 (DFF). In such conditions, the receive process sets the Receive Buffer Unavailable status and returns to the Suspend state.

3.4.4 Interrupts

Interrupts can be generated as a result of various events. The DMA Register 5 (Status Register) contains all the bits that might cause an interrupt. Register 7 (Interrupt Enable Register) contains an enable bit for each of the events that can cause an interrupt.

There are two groups of interrupts, Normal and Abnormal, as described in Register 5 (Status Register). Interrupts are cleared by writing a 1'b1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal sbd_intr_o is de-asserted. If the GMAC core is the cause for assertion of the interrupt, then any of the GLI, GMI, or GPI bits of DMA Register 5 (Status Register) are set high.



Note The Register 5 (Status Register) is the (interrupt) status register. The interrupt pin (sbd_intr_o) is asserted because of any event in this status register only if the corresponding interrupt enable bit is set in Register 7 (Interrupt Enable Register).

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt (bit 6 of Register 5 (Status Register)) indicates that one or more frames were transferred to the Host buffer. The driver must scan all descriptors, from the last recorded position to the first one owned by the DMA.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the Register 5 (Status Register) for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs, after the driver has cleared the appropriate bit in Register 5 (Status Register). For example, the controller generates a Receive interrupt (bit 6 of Register 5 (Status Register)) and the driver begins reading Register 5 (Status Register). Next, Receive Buffer Unavailable (bit 7 of Register 5 (Status Register)) occurs. The driver clears the Receive interrupt. Even then, the sbd_intr_o signal is not de-asserted, because of the active or pending Receive Buffer Unavailable interrupt.

An interrupt timer RIWT (bits 7:0 in Register 9 (Receive Interrupt Watchdog Timer Register)) is given for flexible control of Receive Interrupt. When this Interrupt timer is programmed with a non-zero value, it gets activated as soon as the RxDMA completes a transfer of a received frame to system memory without asserting the Receive Interrupt because it is not enabled in the corresponding Receive Descriptor (RDES1[31] in Table 8-3). When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RI is enabled in Register 7 (Interrupt Enable Register). This timer gets disabled before it runs out, when a frame is transferred to memory and the RI is set because it is enabled for that descriptor.

3.4.5 Error Response to DMA

For any data transfer initiated by a DMA channel, if the slave replies with an error response, that DMA stops all operations and updates the error bits and the Fatal Bus Error bit in the Status register (DMA Register 5). That DMA controller can resume operation only after soft resetting or hard resetting the core and re-initializing the DMA. This DMA behavior is true for non-AHB/AXI interfaced DMAs ("Data Transfer With Error" on page 71) that receive an error response through the mdc_error_i signal.

3.4.6 DMA Native Interface

The Subsystem can be configured (GMAC-DMA) to have DMA with a native FIFO-like interface on the application side and an MCI (Optionally APB) interface for CSR port. Figure 5-3 provides the top-level I/O diagram for this GMAC-DMA configuration. All functions and features of the DMA remain the same as described above.

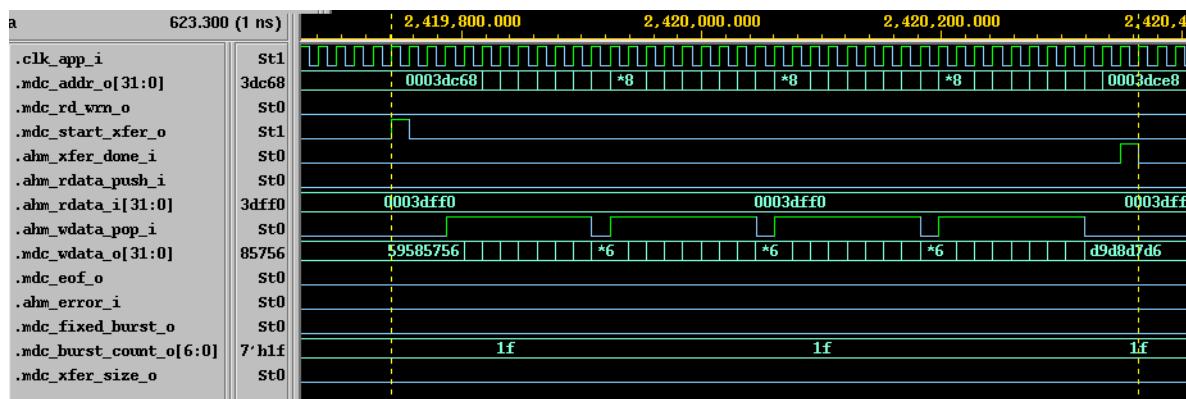
On the native interface, the DMA initiates data or descriptor transfers using a simple hardware protocol. The DMA always starts a transaction with the proper values on the address bus, transfer-size, burst-length and type of transfer (whether read or write). If it is a read transfer, then the application can push valid data by asserting signal `mdc_rdata_push_i`. For write transfers, the application can accept the data from DMA by asserting signal `mdc_wdata_pop_i`. The application can pause the transfers by de-asserting the respective push or pop control signals.

Note

The values on the address bus, the burst-length, and the transfer size are not updated immediately on the receipt of push or pop signals. The address, burst-length, and transfer size are valid only at the start of the transfer.

In certain types of access, the burst-length may not even change during the transfers as shown in [Figure 3-21](#). The application must assert the `mdc_rdata_push_i` or `mdc_wdata_pop_i` for clocks equal to the burst length requested at the start of transfer (unless a premature write transfer is indicated by DMA as explained in next section) before asserting the `mdc_xfer_done_i` signal for proper transfer of data.

Figure 3-21 mdc_burst_count_o



After transferring all of the requested data, application must terminate the DMA transaction with signal `mdc_xfer_done_i`. The application can indicate an error by activating signal `mdc_error_i`, and the respective DMA (Rx or Tx) goes to the Fatal-error state and all operations stop. The application must provide a software-reset to restart that DMA's operation.

3.4.6.1 Write Data Transfer

[Figure 3-22](#) shows the transfer sequence of a data buffer being written to application memory.

1. The subsystem initiates the transaction by asserting `mdc_start_xfer_o` (pulse), along with the host memory address (`mdc_addr_o`), read/write control signal (`mdc_rd_wrn_o = 0`), number of beats (`mdc_burst_count_o`), and the transfer size (`mdc_xfer_size_o`). It also drives the first data beat of the burst on the `mdc_wdata_o` bus.
2. The application asserts signal `mdc_wdata_pop_i` to accept the data.
3. If the application is not ready to handle the data, it can de-assert `mdc_wdata_pop_i` and delay the data transfer. The DMA holds the values on the data bus, when the "pop" control is de-asserted.
4. After all of the data has been transferred, as indicated by `mdc_burst_count_o`, the application must end the DMA transfer by asserting signal `mdc_xfer_done_i` (pulse).
5. The DMA can prematurely end the burst transfer because of an End-of-Frame being transferred to the application. The DMA marks the End-of-Frame data on `mdc_wdata_o` with the assertion of

mdc_eof_o. The application must then complete the transaction by asserting signal mdc_xfer_done_i after accepting the End-of-Frame data even if the initial burst-length given at start of transaction is not completed.

Figure 3-22 Write Data Transfer Timing (1 of 3)

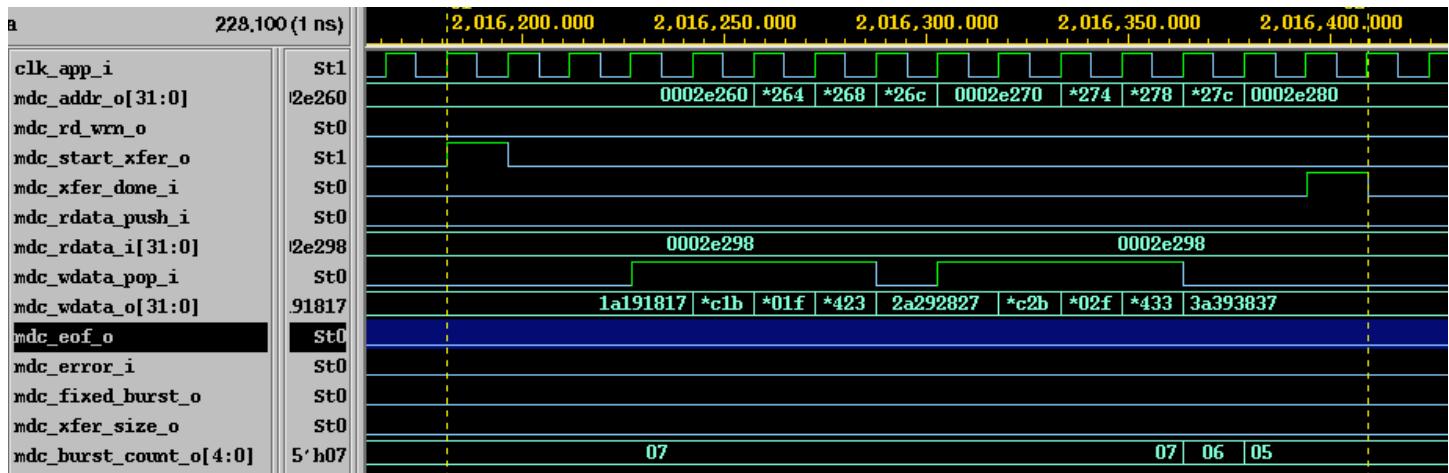
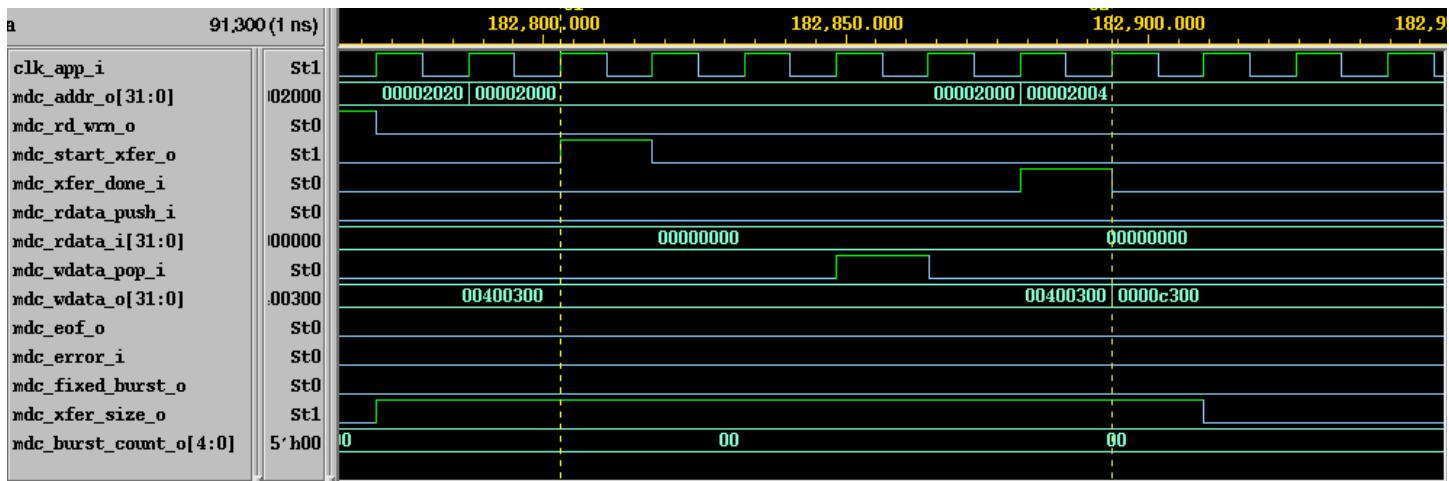


Figure 3-23 shows the transfer sequence when a data buffer is written to application memory with the burst transfer ending prematurely because of the assertion of mdc_eof_o. You can observe that the DMA terminates the burst by asserting mdc_eof_o after completing 3 cycles of the requested 4-cycle burst. The host has to then respond with the assertion of mdc_xfer_done_i without asserting any more mdc_wdata_pop_i.

Figure 3-23 Write Data Transfer Timing (2 of 3)



Figure 3-24 shows the transfer sequence when a descriptor is written to application memory.

Figure 3-24 Write Data Transfer Timing (3 of 3)

3.4.6.2 Read Data Transfer

Figure 3-25 shows the transfer sequence during data buffer being read from application memory.

1. The subsystem initiates the transaction by asserting mdc_start_xfer_o (pulse), along with the host memory address (mdc_addr_o), read/write control signal (mdc_rd_wrn_o = 1), number of beats (mdc_burst_count_o), and the transfer size (mdc_xfer_size_o).
2. The DMA accepts the data on the mdc_rdata_i bus whenever the application asserts signal mdc_rdata_push_i.
3. If the application is not ready with the data, it can de-assert mdc_rdata_push_i and delay the data transfer.
4. After all of the data has been transferred as indicated by mdc_burst_count_o, the application must end the DMA transfer by asserting signal mdc_xfer_done_i (pulse).

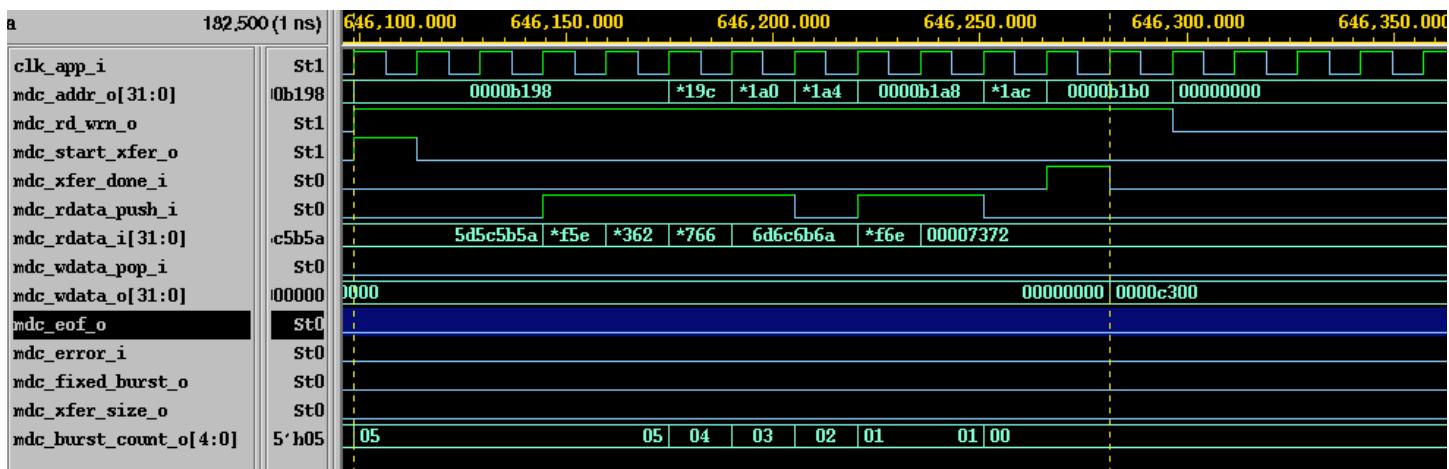
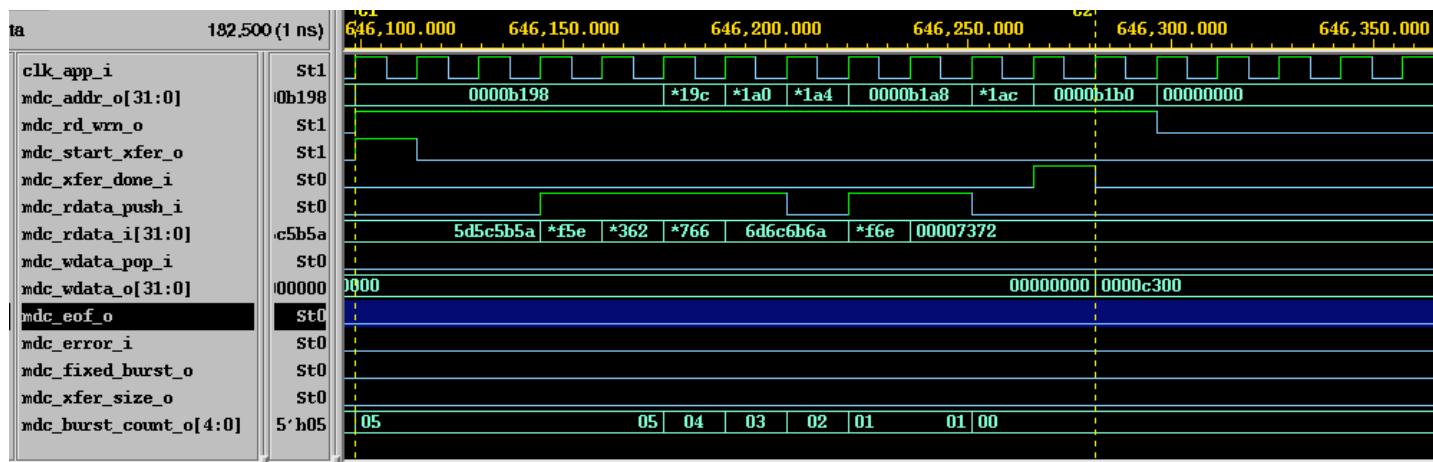
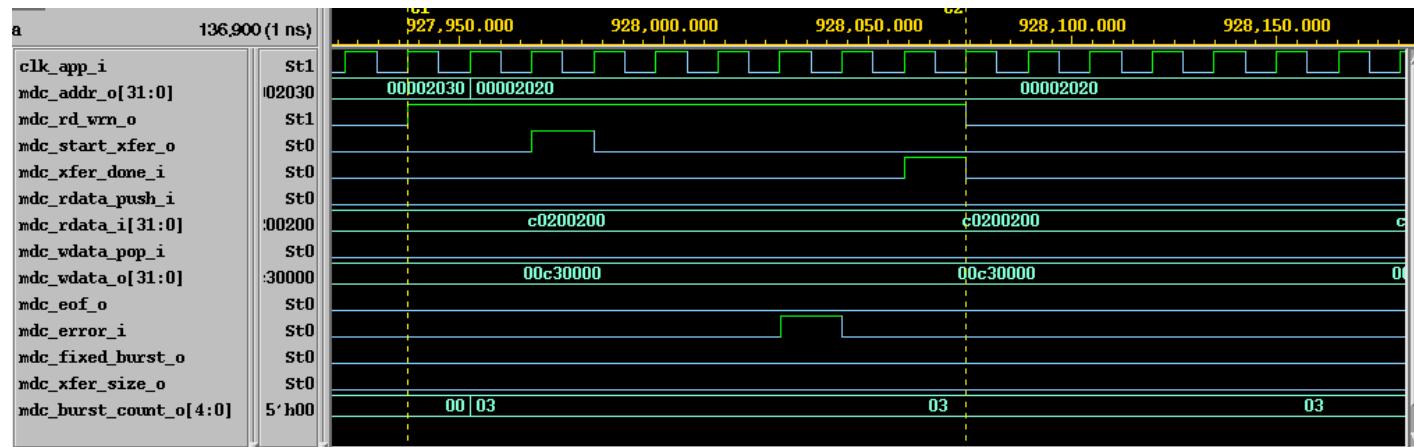
Figure 3-25 Read Data Transfer Timing (1 of 2)

Figure 3-26 shows the transfer sequence during a descriptor being read from application memory.

Figure 3-26 Read Data Transfer Timing (2 of 2)

3.4.6.3 Data Transfer With Error

For any write or read transfer started by the DMA, the application can assert mdc_error_i if it cannot complete the transaction (see [Figure 3-27](#)). The application should also complete the transfer by asserting mdc_xfer_done_i in a different clock cycle. If the application asserts both the signals together, the DMA may restart the transfer to the same address and the behavior of the DMA is not predictable.

Figure 3-27 Data Transfer With Error Timing

3.5 MAC Transaction Layer (MTL)

The MAC Transaction Layer provides FIFO memory to buffer and regulate the frames between the application system memory and the GMAC core. It also enables the data to be transferred between the application clock domain and the GMAC clock domains. The MTL layer has 2 data paths, namely the Transmit path and the Receive Path. The data path for both directions is 32/64/128-bit wide and operates with a simple FIFO protocol.

The GMAC-MTL communicates with the application side with the Application Transmit Interface (ATI), Application Receive Interface (ARI), and the MAC Control Interface (MCI). The default and optional I/O signals are described in “[Signals](#)” on page [205](#)

3.5.1 Transmit Path

In GMAC-AHB/AXI and GMAC-DMA configurations, the DMA controls all transactions for the transmit path through the ATI. Ethernet frames read from the system memory is pushed into the FIFO by the DMA. The frame is then popped out and transferred to the GMAC core when triggered. When the end-of-frame is transferred, the status of the transmission is taken from the GMAC core and transferred back to the DMA.

The Transmit FIFO has a default depth of 2K bytes. FIFO-fill level is indicated to the DMA so that it can initiate a data fetch in required bursts from the system memory, using the AHB/AXI interface. The data from the AHB/AXI Master interface is pushed into the FIFO with the appropriate byte lanes qualified by the DMA. The DMA also indicates the start-of-frame (SOF) and end-of-frame (EOF) transfers along with a few sideband signals controlling the pad-insertion/CRC generation for that frame in the GMAC core.

The per-frame control bits, such as Automatic Pad/CRC Stripping disable, timestamp capture, and so forth are taken as sideband control inputs on the ATI, stored in a separate register FIFO, and passed on to the core transmitter when the corresponding frame data is read from the Transmit FIFO.

There are two modes of operation for popping data towards the GMAC core. In Threshold mode, as soon as the number of bytes in the FIFO crosses the configured threshold level (or when the end-of-frame is written before the threshold is crossed), the data is ready to be popped out and forwarded to the GMAC core. The threshold level is configured using the TTC bits of DMA Register 0. In store-and-forward mode, the MTL pops the frame towards the GMAC core only when one or more of the following conditions are true:

- ❖ When a complete frame is stored in the FIFO
- ❖ When the TX FIFO becomes almost full
- ❖ When the ATI watermark becomes low. The watermark becomes low when the requested FIFO does not have space to accommodate the requested burst-length on the ATI.

Therefore, the MTL never stops in the store-and-forward mode even if the Ethernet frame length is bigger than the Tx FIFO depth.

The application can flush the Transmit FIFO of all contents by setting the FTF (DMA Register 6[20]) bit. This bit is self-clearing and initializes the FIFO pointers to the default state. If the FTF bit is set during a frame transfer from the MTL to the GMAC core, then the MTL stops further transfer as the FIFO is considered to be empty. Hence an underflow event occurs at the GMAC transmitter and the corresponding Status word is forwarded to the DMA.

For information about the initialization and transmit operations for the MTL Layer, see [Sections 3.5.1.1](#) through [Section 3.5.1.7](#). The timing diagrams are provided in “[Transmit Path Timing](#)” on page [74](#).

3.5.1.1 Initialization

Upon reset, the MTL is ready to manage the flow of data to and from the DMA and the GMAC.

There are no requirements for enabling the MTL. However, the GMAC block and the DMA controller must be enabled individually through their respective CSRs.

3.5.1.2 Single-Packet Transmit Operation

During a transmit operation, the MTL block is slaved to the DMA controller. The general sequence of events for a transmit operation is as follows.

1. If the system has data to be transferred, the DMA controller, if enabled, fetches data from the Host through the AHB/AXI Master interface and starts forwarding it to the MTL. The MTL pushes the data received from the DMA into the FIFO. It continues to receive the data until the end-of frame of the frame is transferred.

2. The data is taken out of the FIFO and sent to the MAC by the FIFO controller engine. When the threshold level is crossed or a full packet of data is received into the FIFO, the MTL pops out the frame data and drives them to the GMAC core. The engine continues to transfer data from the FIFO until a complete packet has been transferred to the MAC. Upon completion of the frame, the MTL receives the Status from the GMAC and then notifies the DMA controller

3.5.1.3 Transmit Operation—Two Packets in the Buffer

1. Because the DMA must update the descriptor status before releasing it to the Host, there can be at the most two frames inside a transmit FIFO. The second frame is fetched by the DMA and put into the FIFO only if the OSF (Operate on Second Frame bit is set). If this bit is not set, the next frame is fetched from the memory only after the MAC has completely processed the frame and the DMA has released the descriptors.
2. If the OSF bit is set, the DMA starts fetching the second frame immediately after completing the transfer of the first frame to the FIFO. It does not wait for the status to be updated. The MTL, in the meantime, receives the second frame into the FIFO while transmitting the first frame. As soon as the first frame has been transferred and the status is received from the MAC, the MTL pushes it to the DMA. If the DMA has already completed sending the second packet to the MTL, it must wait for the status of the first packet before proceeding to the next frame.

3.5.1.4 Transmit Operation—Multiple Packets in Buffer

In GMAC-MTL configuration, the transmit FIFO can be configured to accept more than 2 packets at a time. This option limits the number of status words that can be stored in the MTL before it is transferred to the DMA/host. By default, this number is limited to 2 but can be configured for 4 or 8 as well. Once the MTL FIFO accepts the number of frames equal to the status FIFO depth, it stops accepting further frames unless the transmit Status that is given out and accepted by the host/DMA thus freeing up the space in this small FIFO.

3.5.1.5 Retransmission During Collision

While a frame is being transferred from the MTL to the GMAC, a collision event occurs on the GMAC line interface in half-duplex mode. The GMAC then indicates a retry attempt to the MTL by giving the status even before the end-of-frame is transferred from MTL. Then the MTL enables the retransmission by popping out the frame again from the FIFO.

After more than 96 bytes (or 548 bytes in 1000-Mbps mode) are popped towards the GMAC core, the FIFO controller frees up that space and makes it available to the DMA to push in more data. This means that the retransmission is not possible after this threshold is crossed or when the GMAC core indicates a late-collision event.

When a frame transmission is aborted because of underflow and a collision event immediately follows (initiating a retry), then retry has higher priority than abort as described in [Table 3-4](#).

3.5.1.6 Transmit FIFO Flush Operation

The GMAC provides a control to the software to flush the Transmit FIFO in the MTL layer through the use of Bit 20 of the Operation Mode register (see “[Register 6 \(Operation Mode Register\)](#)” on page [305](#)). The Flush operation is immediate and the MTL clears the Tx FIFO and the corresponding pointers to the initial state even if it is in the middle of transferring a frame to the GMAC Core. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow as TxFIFO does not complete the transfer of rest of the frame. As in all underflow conditions, a runt frame is transmitted and observed on the line. The status of such a frame is marked with both Underflow and Frame Flush events (TDES0 bits 13 and 1).

The MTL layer also stops accepting any data from the application (DMA) during the Flush operation. It generates and transfers Transmit Status Words to the application for the number of frames that is flushed inside the MTL (including partial frames). Frames that are completely flushed in the MTL have the Frame Flush Status bit (TDES0[13]) set. The MTL completes the Flush operation when the application (DMA) accepts all of the Status Words for the frames that were flushed, and then clears the Transmit FIFO Flush control register bit. At this point, the MTL starts accepting new frames from the application (DMA).

In GMAC-MTL configuration, an option is provided to have a sideband signal (`ati_txfifoflush_i`) initiate Flush operation. Whenever this input is sampled high, the `ati_rdy_o` signal is de-asserted and the MTL Tx FIFO is flushed. When the Flush operation is complete, `ati_rdy_o` is asserted, indicating the MTL is ready to accept new frames. All data presented to the MTL after a Flush operation is discarded unless it starts with an SOF marker.

3.5.1.7 Transmit Status Word

At the end of transfer of the Ethernet frame to the GMAC core and after the core completes the transmission of the frame, the MTL outputs the transmit status (`ati_txstatus_o[23:0]`) to the application. This is indicated by an active `ati_txstatus_val_o` signal. The detailed description of the Transmit Status is the same as for bits [23:0] of TDES0, given in [Table 8-6](#).

If IEEE 1588 time stamping is enabled, the MTL returns specific frame's 64-bit time stamp, along with the ATI's transmit status. A valid time stamp is indicated by an active high on the `ati_txstatus_o[17]` bit.

3.5.1.8 Transmit Path Timing

Timing specifications for the MAC Transaction layer (MTL) interface signals in the transmit path are illustrated in detail, in the following sections.

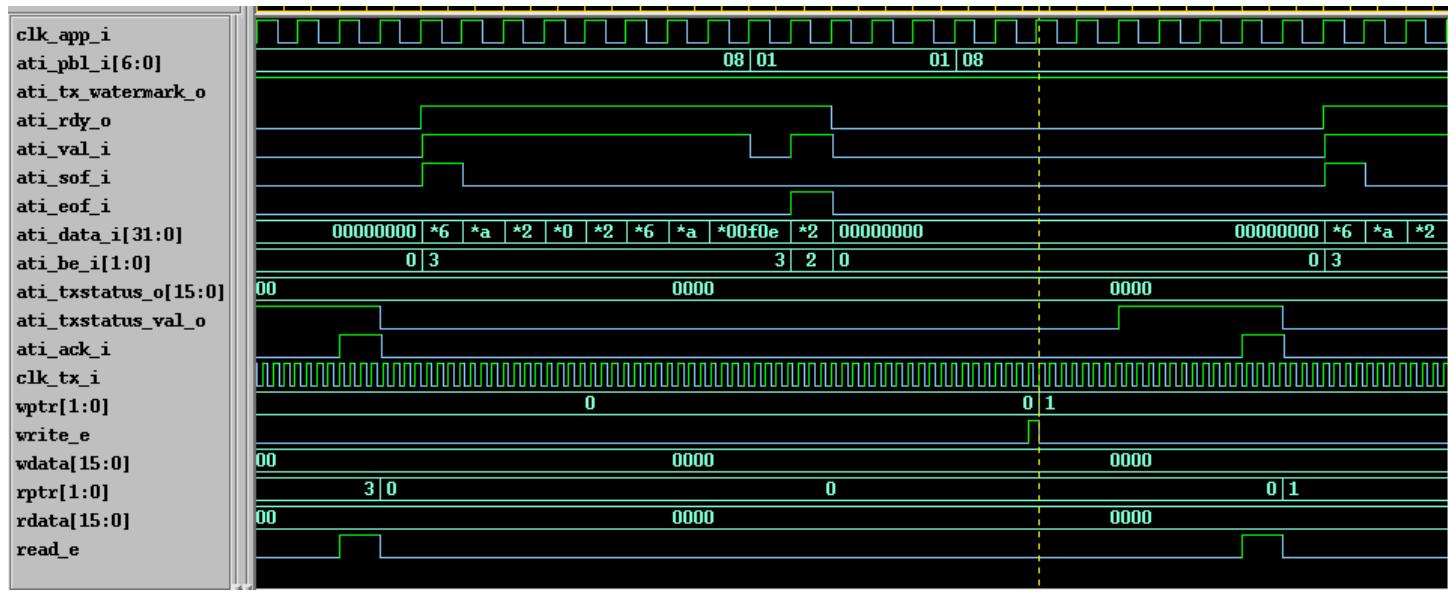
3.5.1.8.1 Application Transmit Interface (ATI)

[Figure 3-28](#) illustrates the timing specifications for the Application Transmit Interface signals. The handshake mechanism involved in frame transmission is as follows:

1. The application, before starting a burst-transfer of data to ATI, should check whether the MTL Tx FIFO has space to accommodate the burst. The length of the proposed burst transfer is put on `ati_pbl_i`.
2. In response, the MTL asserts or de-asserts `ati_txwatermark_o` one clock later, indicating whether it can accommodate the burst data transfer or not.
3. At any point during the transfer, the MTL can alert the application to stop the transfer by de-asserting `ati_rdy_o`. Signal `ati_rdy_o` is de-asserted only when FIFO is almost full, or when the Tx FIFO is being flushed. The Tx FIFO-full event does not occur if the application ensures that enough space is available before the start of data transfer, as described in [steps 1](#) and [2](#).
4. The application starts a frame transfer by asserting `ati_val_i` and `ati_sof_i`. The MTL accepts the data whenever `ati_rdy_o` and `ati_val_i` are asserted.
5. When the application wants to transfer the last bytes of a frame, `ati_eof_i` should be asserted along with the byte enables (`ati_be_i`), indicating the valid number of data bytes on the bus. It is mandatory that all data transfers except the EOF transfer have the byte-enables as all-ones.
6. After transferring the frame onto the Ethernet PHY, the GMAC gives the transmission status to the MTL (not shown in the diagram).
7. The MTL gives the transmit status of the frame back to the application through `ati_txstatus_o`, by asserting `ati_txstatus_val_o`. The application accepts the status by asserting `ati_ack_i`.

8. The application need not wait for steps 6 and 7 to start the transfer of next frame on the ATI. If ati_rdy_o is asserted, the application can transfer the next frame. In the figure, wptra, rptr, write_e, and read_e are the internal signals of the 2-deep asynchronous FIFO where the transmit status is stored. It can be seen that the status written in location 0x0 (wptra = 0) corresponds to the previous transmitted frame. The application reads that status and starts the transfer of next frame on the ATI.

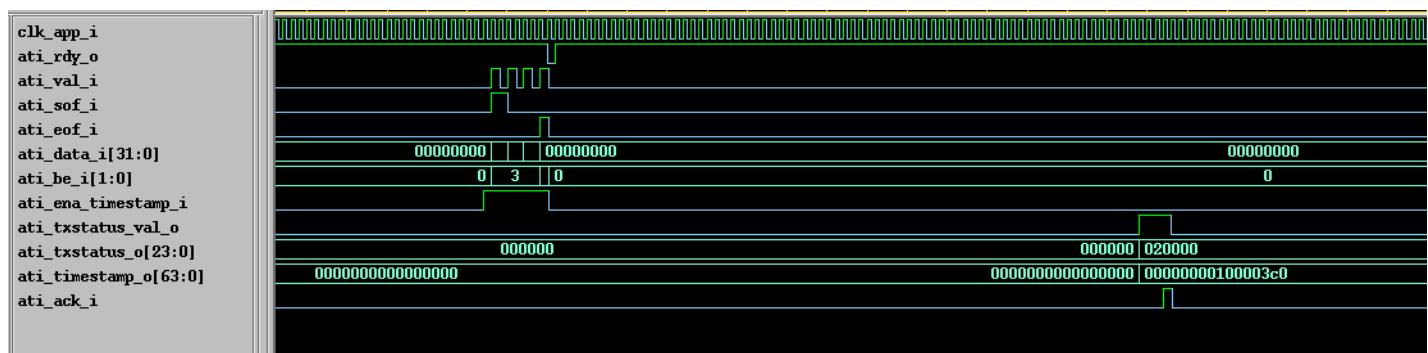
Figure 3-28 Application Transmit Interface (ATI) Timing



3.5.1.8.2 ATI With Time Stamping

Figure 3-29 shows the timing of the ATI when IEEE 1588 time stamping is enabled. The ati_ena_timestamp_i signal is sampled along with the ati_sof_i for an input frame. When that frame's status is given on ati_txstatus_o, the time stamp captured for that frame is output onto ati_timestamp_o. The time stamp's validity is indicated by the assertion of ati_txstatus_o[17] when the ati_txstatus_val_o is high.

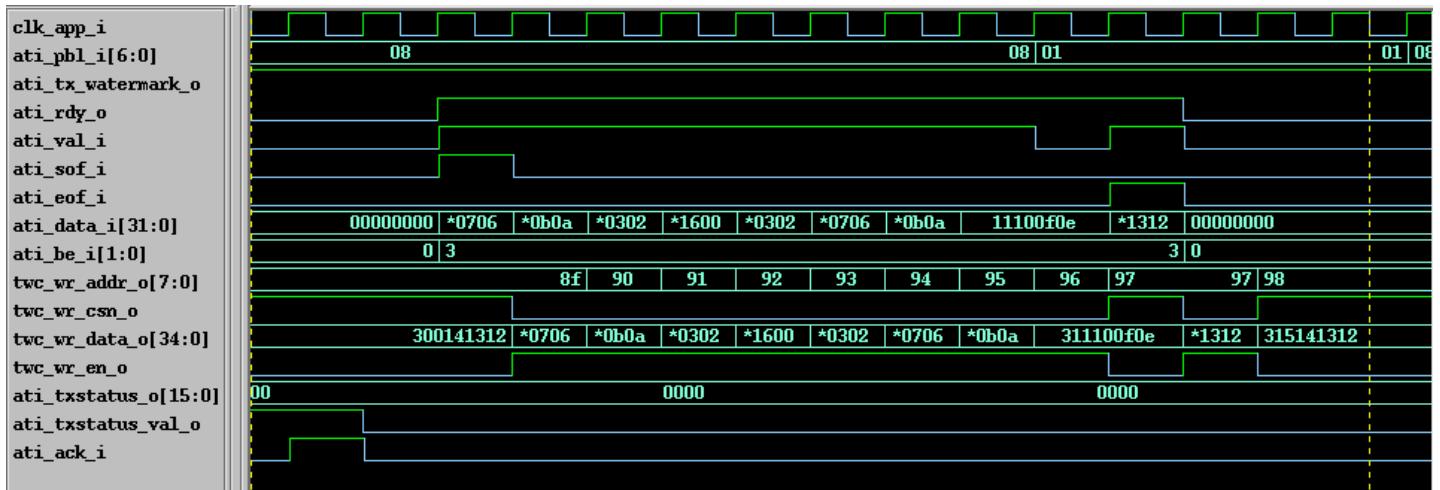
Figure 3-29 ATI Time Stamp Timing



3.5.1.8.3 Transmit FIFO Write Interface

The MTL Transmit FIFO controller transfers the data accepted by the ATI to the Tx FIFO implemented as external memory. Timing specifications for the memory interface signals (`tvc_wr_*` signals) are illustrated in [Figure 3-30](#), which also shows the timing relationship of the memory interface signals with respect to the ATI signals.

Figure 3-30 Transmit FIFO Write Interface Timing



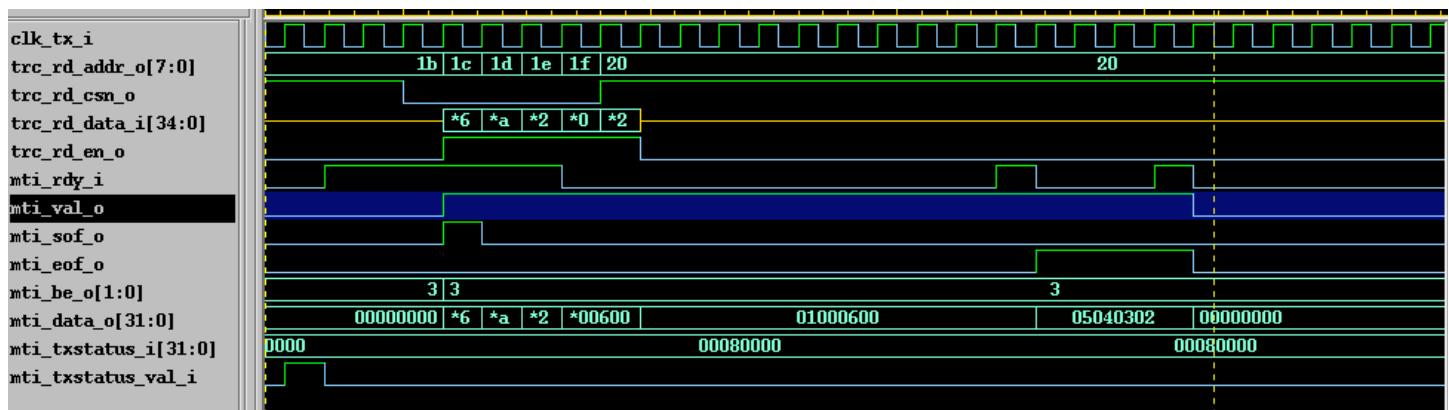
The external memory used for storing frame can be synchronous or asynchronous two-port RAM. The Transmit FIFO write interface timing shown in [Figure 3-30](#) holds good for both synchronous and asynchronous two-port RAM.

3.5.1.8.4 Transmit FIFO Read Interface

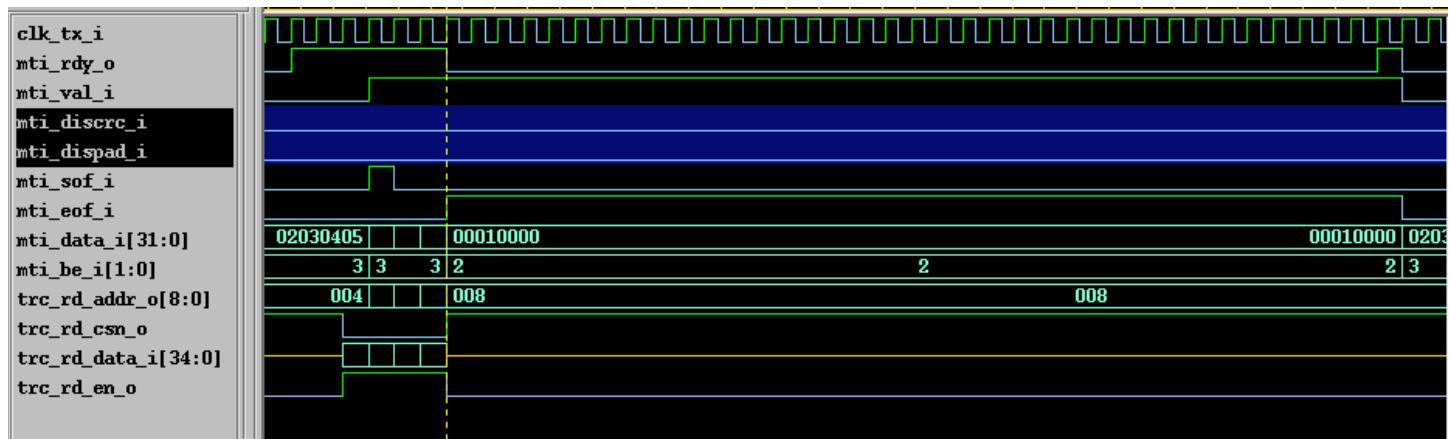
The Transmit FIFO Read controller reads the data from the Tx FIFO and transfers it to the GMAC Core on the MAC Transmit Interface (MTI). The timing relationship of the MTI signals with the Transmit FIFO Read Interface signals are shown in [Figure 3-31](#). The external memory of the Rx FIFO is configured as synchronous RAM in the figure.

The sequence of events during the transfer of a frame from Tx FIFO to the MTI is as follows:

1. When the Tx FIFO becomes non-empty (not shown in the diagram), the Read Controller starts the read transfers by asserting signals `trc_rd_addr_o` and `trc_rd_csn_o`. The assertion of `trc_rd_en_o` is delayed by 1 clock cycle as per synchronous RAM timing requirements.
2. The data is read on the `trc_rd_data_i` bus in the next clock cycle. As this is directly transferred to the `mti_data` bus to the GMAC Core (MTI), `mti_val_i` is also asserted. Signal `mti_sof_i` is also asserted along with `mti_val_i` in the figure, as the data corresponds to start of frame.
3. The Tx FIFO Read Controller can pre-fetch data from three locations (two in Asynchronous RAM port) so that it can sustain continuous transfers to the MTI interface without any delay.
4. If the MAC core accepts the data (active `mti_rdy_o`), more data is read from the FIFO. If the MAC de-asserts `mti_rdy_o`, then the read operations are suspended in the next clock cycle and `trc_rd_csn_o` is de-asserted.
5. Signals `mti_dispad_i` and `mti_discrc_i` are the sideband control signals used to direct the MAC to add pad-bytes and/or CRC. These signals are transferred from the ATI interface (`ati_dispad_i` and `ati_discrc_i`) to the MTI interface when `mti_sof_i` is active, through an internal asynchronous register FIFO.

Figure 3-31 Transmit FIFO Read Interface Timing (1 of 2)

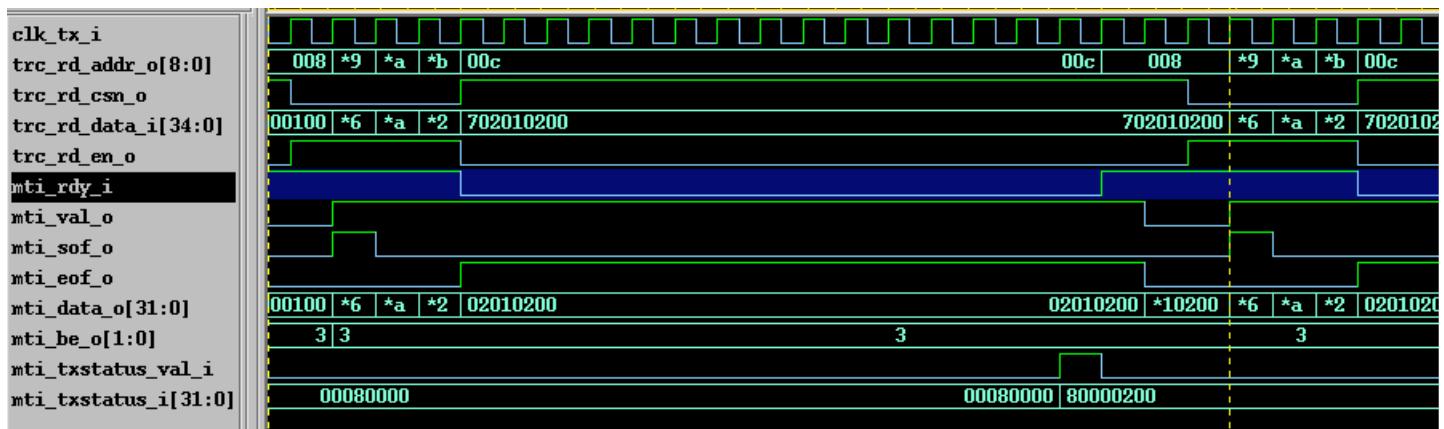
The asynchronous two-port RAM Tx FIFO read interface timing has subtle changes with respect to the synchronous RAM, and is shown in [Figure 3-32](#). In this mode, the read data (trc_rd_data_i) is given out by the Tx FIFO on the assertion of the chip select and read enable signals (trc_rd_csn_o and trc_rd_en_o) in the same clock cycle. This data is registered and is given to the MTI interface in the next clock cycle.

Figure 3-32 Transmit FIFO Read Interface Timing (2 of 2)

3.5.1.8.5 Transmit Frame Retransmission

The MAC transmitter may abort the transmission of a frame because of collision, underflow, loss of carrier, and several other conditions. When frame transmission is aborted because of collision, the MAC requests retransmission of the frame. This scenario is illustrated with the following sequence of events, shown in [Figure 3-33](#).

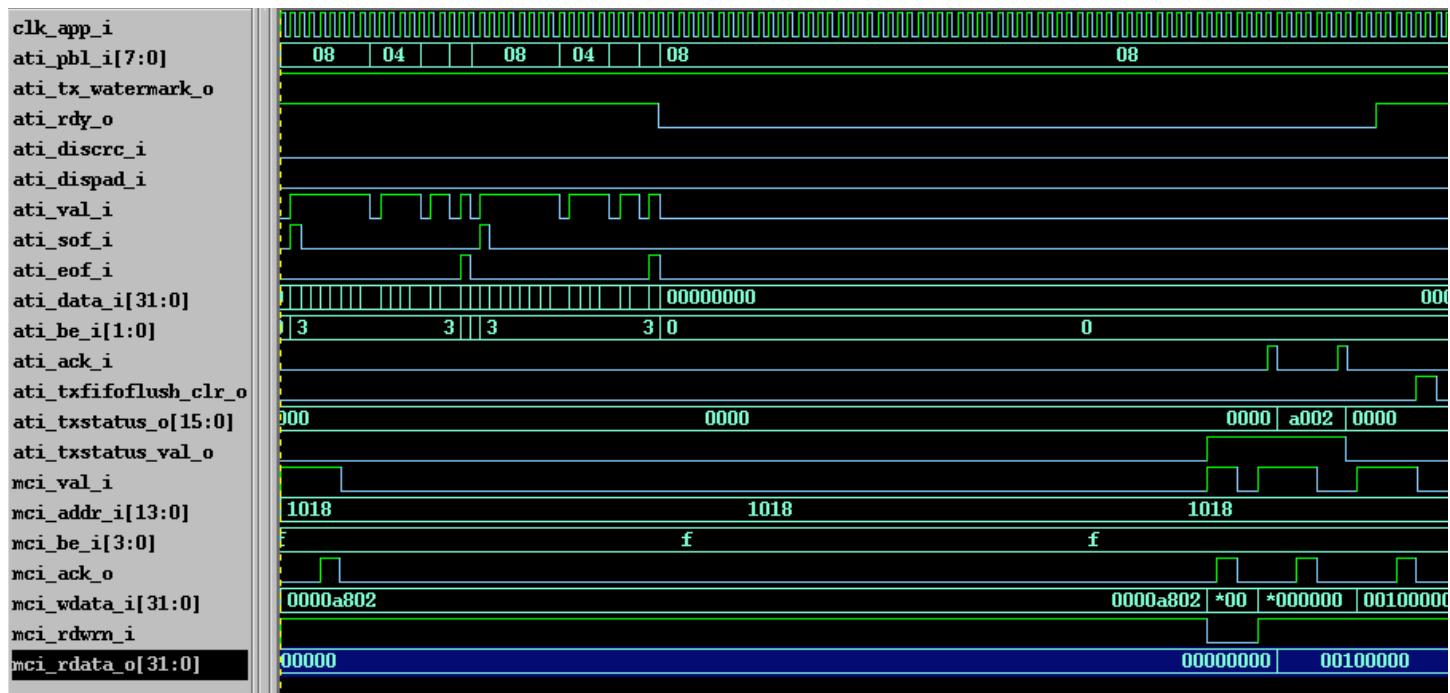
1. The MTL starts the transfer of a frame to the MAC by asserting mti_val_i and mti_sof_i.
2. Subsequently, the remaining data is sent upon the assertion of mti_rdy_o by the MAC.
3. When the MAC experiences a collision on the Ethernet bus (not shown in diagram), frame transmission is suspended first by keeping mti_rdy_o de-asserted.
4. The MAC transfers the transmission status on mti_txstatus, and requests retransmission by asserting Bit 31 of mti_txstatus.
5. The MTL retransmits the frame. The Tx FIFO read address (trc_rd_addr) rewinds back to the start of frame address 0x008 for the retransmission.

Figure 3-33 Transmit Frame Retransmission Timing

3.5.1.8.6 Transmit FIFO Flush Operation

The MTL allows the Tx FIFO to be flushed at any instance, through a CSR bit (Operation Mode Register[20]). [Figure 3-34](#) shows the scheme of things that occur after the FIFO Flush request from the application.

1. In the figure, two frames are pushed by the application into the Tx FIFO on the ATI interface.
2. The application performs a write to the Operation mode register (address = 1018) and sets the Flush Transmit FIFO bit.
3. The application reads back the same register to check the status of the Flush operation. mci_rdata_o[20] is asserted high during CSR read access, indicating the progress of the Flush operation.
4. Since the first frame is already read and transmitted by the MAC by this time, the MTL gives the transmit status of the first frame as normal. This is seen by the value of 0x0000 on the ati_txstatus read with the first ati_ack_i pulse.
5. The MTL now flushes the second frame in the FIFO and subsequently, gives a Flush Frame status (0xa002) to the application on the ati_txstatus bus.
6. The MTL indicates the completion of the Flush operation by asserting ati_txifoflush_clr (internal), which clears the Flush transmit FIFO bit in the OMR.

Figure 3-34 Transmit FIFO Flush Timing

3.5.2 Receive Path

This module receives the frames given out by the GMAC core and pushes them into the Rx FIFO. The status (fill level) of this FIFO is indicated to the DMA once it crosses the configured Receive threshold (RTC of DMA Register 6). The MTL also indicates the FIFO fill level so that the DMA can initiate pre-configured burst transfers towards the AHB interface.

[“Receive Operation” on page 79](#) through [“Receive Status Word” on page 81](#) detail receive operations for the MTL Layer. Timing diagrams are provided in [“Receive Path Timing Diagrams” on page 81](#).

3.5.2.1 Receive Operation

During an Rx operation, the MTL is slaved to the GMAC. The general sequence of Receive operation events is as follows:

- When the GMAC receives a frame, it pushes in data along with byte enables. The GMAC also indicates the SOF and EOF. The MTL accepts the data and pushes it into the Rx FIFO. After the EOF is transferred, the GMAC drives the status word, which is also pushed into the same Rx FIFO by the MTL.
- When IEEE 1588 time stamping is enabled and the 64-bit time stamp is available along with the receive status, it is appended to the frame received from the GMAC and is pushed into the Rx FIFO before the corresponding receive status word is written. Thus, in 32-bit data bus mode, two additional locations per frame are taken for storing the time stamp in the Rx FIFO, while in 64-/128-bit mode, one additional location is taken.
- The MTL_RX engine takes the data out of the FIFO and sends it to the DMA. In the default Cut-Through mode, when 64 bytes (configured with RTC bits of DMA Register 6) or a full packet of data are received into the FIFO, the MTL_RX engine pops out the data and indicates its availability to the DMA. Once the DMA initiates the transfer to the AHB/AI interface, the MTL_RX engine continues to transfer data from the FIFO until a complete packet has been transferred. Upon completion of the

EOF frame transfer, the MTL pops out the status word and sends it to the DMA controller. If the 64-bit time stamp is read out before the receive status, it is marked by the assertion of the `ari_timestamp_val_o` signal on the ARI. Otherwise, the status is given after the EOF data.



Note In 32-bit data bus mode, the timestamp transfer takes two clock cycles and the lower 32-bit of the timestamp is given out first. The status also may be extended to two cycles when Advanced Timestamp feature is enabled.

4. In Rx FIFO Store-and-Forward mode (configured by the RSF bit of DMA Register 6), a frame is read out only after being written completely into the Receive FIFO. In this mode, all error frames are dropped (if the core is configured to do so) such that only valid frames are read out and forwarded to the application. In Cut-Through mode, some error frames are not dropped, because the error status is received at the end-of-frame, by which time the start of that frame has already been read out of the FIFO.

3.5.2.2 Receive Operation Multiframe Handling

Since the status is available immediately following the data, the MTL is capable of storing any number of frames into the FIFO, as long as it is not full.

3.5.2.3 GMAC Flow Control

The flow control operation of the GMAC can also be enabled (EFC bit of DMA Register 6) by the Rx FIFO control logic. The flow control signal to the GMAC is asserted whenever the Rx FIFO fill level crosses the configured threshold (RFA bits of DMA Register 6). This flow control signal is de-asserted once the FIFO fill-level falls below the configured threshold (RFD bits). This operation is independent of whether the GMAC is configured for full-duplex or half-duplex.

The hardware flow control operation is applicable only when the Rx FIFO is 4,096 or more bytes deep. A separate sideband flow control signal (`sbd_flowctrl_i`) is optionally provided at the top-level I/O for all GMAC-AHB, GMAC-AXI, GMAC-DMA and GMAC-MTL configurations. For 4,096-byte or larger Rx FIFOs, this sideband signal is logically OR'ed with the hardware flow control signal. Thus, flow control is enabled when either of these signals is asserted and disabled when both these signals are de-asserted.

3.5.2.4 Error Handling

If the MTL Rx FIFO is full before it receives the EOF data from the GMAC, an overflow is declared, the whole frame (including the status word) is dropped, and the overflow counter in the DMA (Register 8) is incremented. This is true even if the Forward Error Frame (FEF bit of DMA Register 6) is set. If the start address of such a frame has already been transferred to the Read Controller, the rest of the frame is dropped and a dummy EOF is written to the FIFO along with the status word. The status indicates a partial frame because of overflow. In such frames, the Frame Length field is invalid.

The MTL Rx Control logic can filter error and undersized frames, if enabled (using the DMA Register 6 FEF and FUF bits). If the start address of such a frame has already been transferred to the Rx FIFO Read Controller, that frame is not filtered. The start address of the frame is transferred to the Read Controller after the frame crosses the receive threshold (set by the DMA Register 6 RTC bits).

If the MTL Receive FIFO is configured to operate in Store-and-Forward mode, all error frames can be filtered and dropped.

If you enable the Frame Length FIFO in GMAC-MTL configuration (refer to “[Frame Length Interface](#)” on page [81](#)), then error frames can be filtered even after the start address of the frame is transferred to the Read Controller in the default Cut-Through mode. In this configuration, if a frame’s status and length are

available to the Read Controller using the Frame Length FIFO when that frame's SOF is read from the Rx FIFO, then the complete erroneous frame can be dropped indirectly in MTL.

For the DMA to flush the error frame being read from the FIFO, it must assert the ari_frameflush_i signal. The MTL then stops transferring data to the application (DMA). It internally reads out the rest of the frame and drop it. The MTL then starts the transfer of next frame, if it is available.

3.5.2.5 Receive Status Word

At the end of the transfer of the Ethernet frame to the host, the MTL outputs the receive status (ari_data_o[31:0]) to the Application. This is indicated by an active ari_rxstatus_val_o signal. The detailed description of the receive status is the same as for Bits[31:0] of RDES0, given in [Table 8-1](#), except that Bits 31, 14, 9, and 8 are reserved and have a reset of 1'b0 by default. When the status of a partial frame because of overflow is given out, the Frame Length field in the status word is not valid.



Note When Advanced Time Stamp feature is enabled, the status is composed of two parts - normal (default [31:0]), and extended. The extended status[63:32] gives the information about the received ethernet payload when it is carrying PTP packets or TCP/UDP/ICMP over IP packets. In 32-bit data-bus, these are transferred over two clock cycles. The detailed description of the receive status is the same as described in RDES0 and RDES4 in ["Receive Descriptor "](#) on page [420](#), except that bits 31, 14, 9, and 8 of normal status is reserved and have a reset value of 1'b0. When the status of a partial frame because of overflow is given out, the Frame Length field in the status word is not valid.

3.5.2.6 Frame Length Interface

In case of switch applications, data transmission and reception between the application and MTL happens as complete frame transfers. The application layer should be aware of the length of the frames received from the ingress port in-order to transfer the frame to the egress-port. The GMAC-MTL can be configured to provide a separate frame length interface to support this feature.

The GMAC core provides the frame length of each received frame inside the status at the end of each frame transferred to the MTL. The MTL stores the frame length in an asynchronous FIFO (Frame Length FIFO). The width of this FIFO is configurable from 12-15 (including 1 bit for frame-error tag) bits, while the depth of this FIFO depends on the configured size of the MTL Rx FIFO and the minimum size of the frame stored in the Rx FIFO. A two-port RAM is required to implement this asynchronous FIFO. To provide maximum flexibility in choosing the FIFO implementation, the corresponding input/output signals are made available at the top-level ports of the GMAC-MTL.

When this asynchronous FIFO is non-empty, the frame length is read by the MTL Rx FIFO Read controller and given to the application on the ari_frame_len_o bus and the validity is indicated by the assertion of signal ari_frame_len_val_o. An additional signal (ari_rxfifo_frm_cnt_o) indicating the number of frames present in the Rx FIFO is also driven on the ARI interface. When the corresponding frame is read out on the ARI by the application, the frame length of the next frame is read from the FIFO. The application can use this frame length to check if egress port buffer can accommodate the frame.



A frame length value of 0 is given for partial frames written into the Rx FIFO because of overflow.

3.5.2.7 Receive Path Timing Diagrams

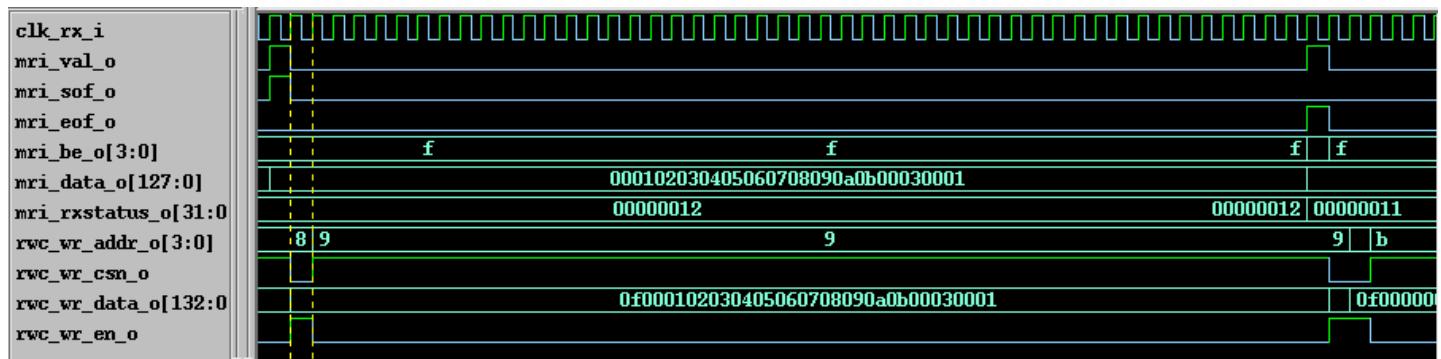
Timing specifications for the MAC Transaction layer (MTL) interface signals in the receive path are illustrated in detail, in the following sections. The timing diagrams are organized to show the sequence of transactions that occur when a received frame is transferred from the MAC to the application.

3.5.2.7.1 Receive FIFO Write Interface

A frame received by the MAC is transferred to the MTL over the MRI interface. The transferred data is accepted and written to the Rx FIFO by the write control logic. (See [Figure 3-35](#).)

1. The MAC starts transferring the received frame to the MTL by asserting `mri_val_o` and `mri_sof_o` along with the corresponding 128-bit data on `mri_data_o`.
2. This is directly transferred to the Rx FIFO in the next clock cycle. This can be seen by the assertion of `rwc_csn_o` and `rwc_wr_en_o` along with the corresponding address and data. Note that the MSB 5 bits of `rwc_wr_data` correspond to the value on `mri_eof` and `mri_be_o` signals accepted by the MTL.
3. Frame transfer is completed by the MAC by the assertion of `mri_eof_o` along with valid data. The receive frame status on `mri_rxstatus_val_o` is also valid with `mri_eof_o`.
4. After writing the EOF data, the Rx FIFO write logic transfers the received status to the external RAM in the next clock cycle.

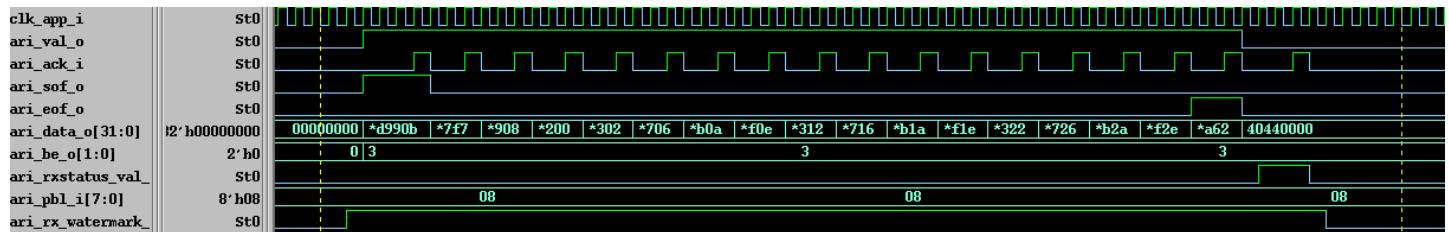
Figure 3-35 Receive FIFO Write Interface Timing



3.5.2.7.2 Application Receive Interface (ARI)

The data received from the MAC and stored in the Rx FIFO is transferred to the Application Receive Interface with the following sequence of events, as shown in [Figure 3-36](#).

1. The ARI indicates the availability of data by asserting signal `ari_val_o` along with the data. The ARI transfers data in every clock cycle as long as data is available in the Rx FIFO.
2. The application can check whether the Rx FIFO has the amount of data (or the complete frame) to support the required burst length on `ari_pbl_i`.
3. The ARI updates `ari_watermark_o` in the next clock cycle. Asserting the watermark indicates that the MTL has enough data to sustain the burst transfer of length given on `ari_pbl_i` (in [Step 2](#)).
4. The application accepts the data from the MTL by asserting `ari_ack_i`. The MTL updates the data in the next clock cycle if it is continuing the burst transfer. The application can pause the transfer by deasserting `ari_ack_i`, as shown in [Figure 3-36](#).
5. The MTL transfers the remaining data by asserting `ari_val_o` and the corresponding data. When the last few bytes of frame (less bytes than the data bus width) are being transferred, `ari_eof_o` is also asserted along with `ari_be_o`, indicating a valid number of bytes in that data phase.
6. The receive frame status (indicated with the assertion of `ari_rxstatus_val_o`) is transferred back-to-back with the end-of-frame data. Frame transfer is complete only after `ari_ack_i` acknowledgement for status.

Figure 3-36 Application Receive Interface (ARI) Timing

3.5.2.7.3 ARI With Time Stamping (Default)

Figure 3-37 shows the ARI timing when IEEE 1588 time stamping is enabled. After the ARI outputs end-of-frame (EOF) data, as indicated by `ari_eof_o` high, it gives the time stamp value on `ari_data_o`. Asserting `ari_timestamp_val_o` validates the time stamp. In this figure, the 64-bit time stamp is output in two cycles, because the data bus is only 32 bits wide. After the application accepts the time stamp (by asserting `ari_ack_i` for two clocks), the receive status is given on the data bus and validated with the assertion of `ari_rxstatus_val_o`.

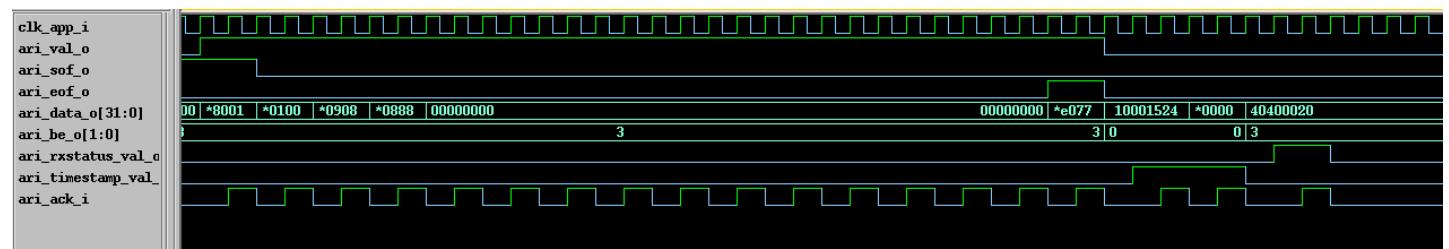
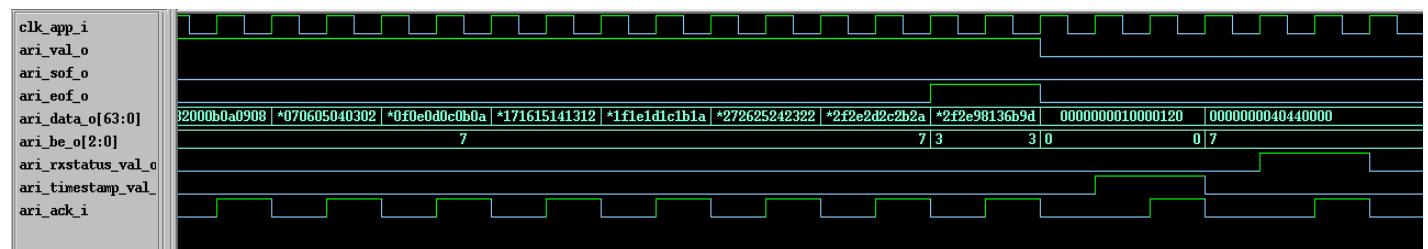
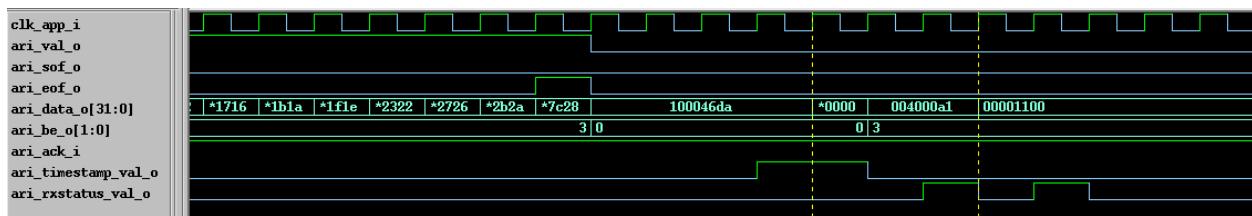
Figure 3-37 ARI Timing With IEEE 1588 Time Stamping Enabled

Figure 3-38 shows the timing of the time stamp transfer when the core is configured for a 64-bit data bus. The 64-bit time stamp is given in one cycle between the EOF data transfer and the Receive Status transfer.

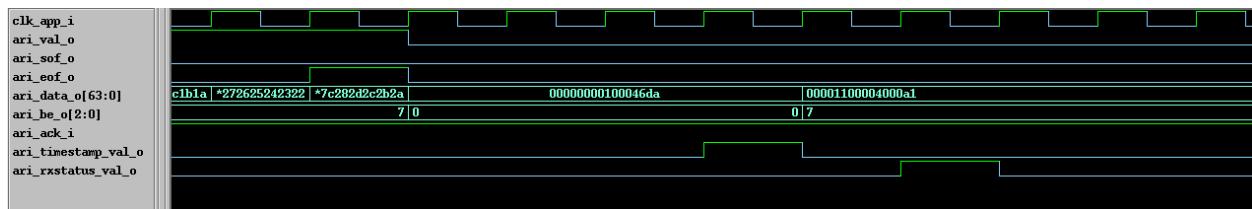
Figure 3-38 ARI Timing for 64-Bit Bus, IEEE 1588 Time Stamping Enabled

3.5.2.8 ARI with Advanced Time Stamping

Figure 3-39 shows the ARI timing when IEEE 1588 Advanced time stamping is enabled for a 32-bit interface. After the time stamp value is read out from the `ari_data_o` port by asserting `ari_ack_i`, the receive status is given on the data bus. The status is divided into two 32-bit status (normal and extended). The normal status is given first on `ari_data_o` validated with the assertion of `ari_rxstatus_val_o` as shown in the figure. When the `ari_ack_i` is asserted for the normal status the extended status is given on `ari_data_o` validated with the assertion of `ari_rxstatus_val_o` as shown in the figure. Bit 0 of the normal status indicates the availability of the extended status.

Figure 3-39 ARI Timing for 32-Bit Interface, IEEE 1588 Advanced Time Stamping Enabled

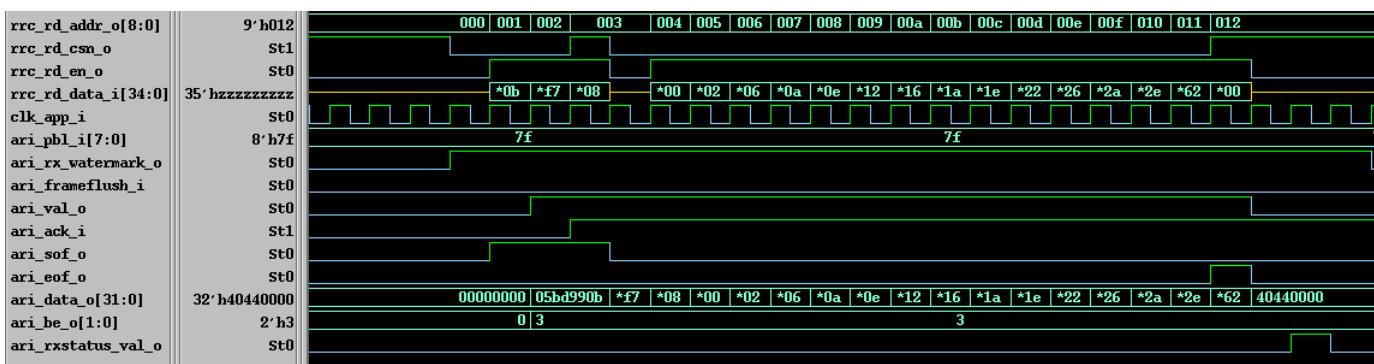
[Figure 3-40](#) shows the ARI timing when IEEE 1588 Advanced time stamping is enabled for 64-bit interface. After the time stamp value is read out from the ari_data_o port in one clock cycle, the receive status is given on the data bus. The normal and the extended status is given together on ari_data_o validated with the assertion of ari_rxstatus_val_o as shown in the figure. The lower 32-bits [31:0] indicate the normal status and the higher 32-bits [63:32] indicate the extended status. Bit 0 indicates the validity of the extended status.

Figure 3-40 ARI Timing for 64-Bit Interface, IEEE 1588 Advanced Time Stamping Enabled

3.5.2.8.1 Receive FIFO Read Interface

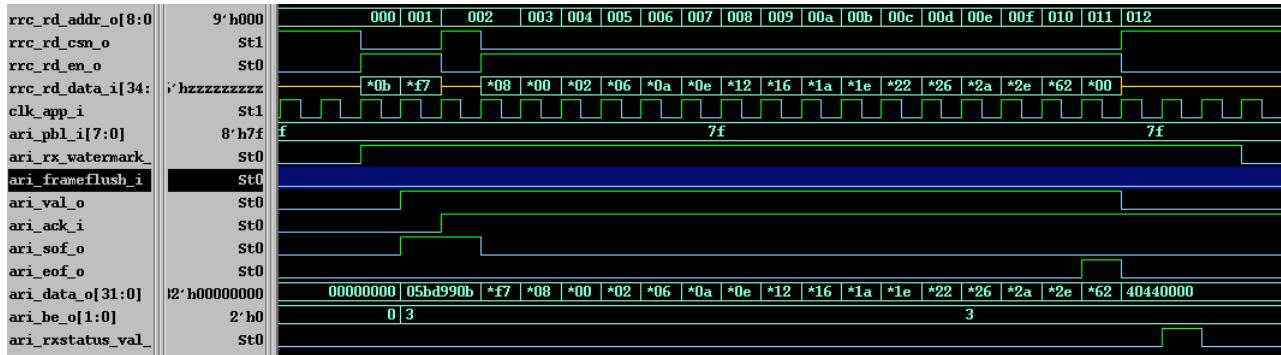
The FIFO read timing diagrams for the synchronous and asynchronous two-port RAM along with the timing relationship with signals on the ARI is shown in this section.

The MTL Rx FIFO read controller pre-fetches data (rrc_rd_* signals) from the Rx FIFO when it finds the Rx FIFO is not empty. The pre-fetching corresponds to maximum of three locations for asynchronous RAM interface and two locations for synchronous RAM interface. This pre-fetching is done so as to sustain burst data transfers without any delay from ARI interface. The (pre-)fetched data is registered and given out on the ARI with the respective ari_* signals. When the application acknowledges the data with ari_ack_i, further data is read from the FIFO. [Figure 3-41](#) shows the FIFO read interface timing for synchronous SRAM. The timing relationship of the rrc_rd_* signals is similar to that of trc_rd_* signals explained in “Transmit FIFO Read Interface” on page 76.

Figure 3-41 Receive FIFO Read Interface Timing (1 of 2)

The timing relationship of the FIFO read interface signals (asynchronous SRAM) with the application receive signals is shown in [Figure 3-42](#). Note that in this figure burst frame transfers are sustained without any break, because of the prefetching of data.

Figure 3-42 Receive FIFO Read Interface Timing (2 of 2)

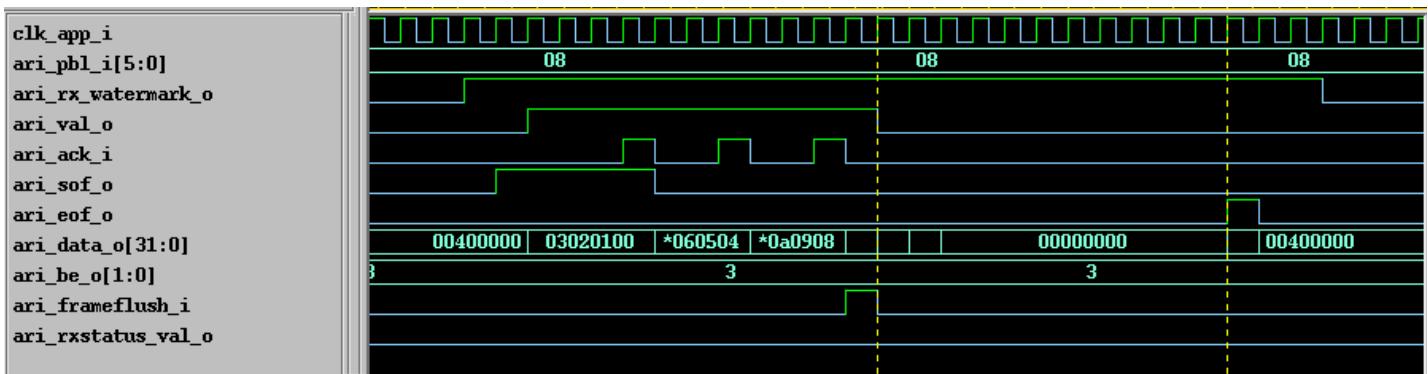


3.5.2.8.2 Receive Frame Flush

When the application decides to drop the frame (at the top of the Rx FIFO) during the start, middle, or end of a frame transfer (`ari_val_o` is asserted), it can assert `ari_frameflush_i` for one clock. the MTL response to this is shown in [Figure 3-43](#).

The MTL first de-asserts `ari_val_o` as soon as it sees an active pulse on `ari_frameflush_i`. It continues the read transfer from the Rx FIFO, but does not transfer it to the ARI. It thus flushes the entire frame, including the status. The MTL is now ready to transfer the next frame.

Figure 3-43 Receive Frame Flush Timing



3.5.2.8.3 Receive Frame Length Interface (Optional)

[Figure 3-44](#) corresponds to a synchronous two-port RAM implementation for frame length FIFO. The sequence of events that occurs during the frame length transfer from the GMAC core to the application is as follows:

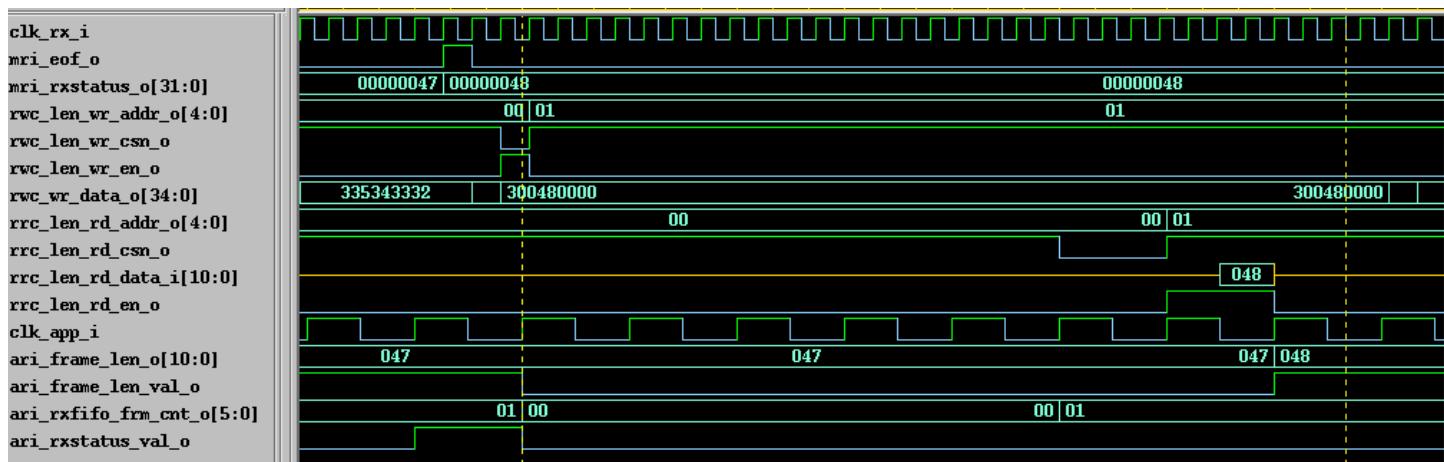
1. GMAC core issues the frame length of the received frame through `mri_rxstatus_o`.
2. Frame length is written with `rwc_wr_data_o[FIFO_WIDTH+16-1:16]` where `FIFO_WIDTH` is configured to any value between 12 to 15. `rwc_wr_data_o` is the data written to the Rx FIFO. The `rwc_err_frame_o` signal is written to the MSB of this frame-length FIFO.

3. When the Frame Length FIFO is not empty, the frame length is read out and issued to the application through ari_frame_len_o and an active ari_frame_len_val_o signal.
4. After a complete frame is read out (i.e. after the assertion of ari_rxstatusval_o), ari_frame_len_val_o is de-asserted or held high, based on the availability of the next frame's length.



Signal ari_rx fifo frm cnt o indicates the number of frames whose EOF is present in the Rx FIFO.

Figure 3-44 Receive Frame Length Interface Timing



The write timing of the asynchronous two-port RAM is similar to that of the synchronous two-port RAM. The difference in the read timing is that the frame length from the FIFO (rrc_len_rd_data_i) is provided in the same clock as the asserted chip select (rrc_len_rd_csn_o).

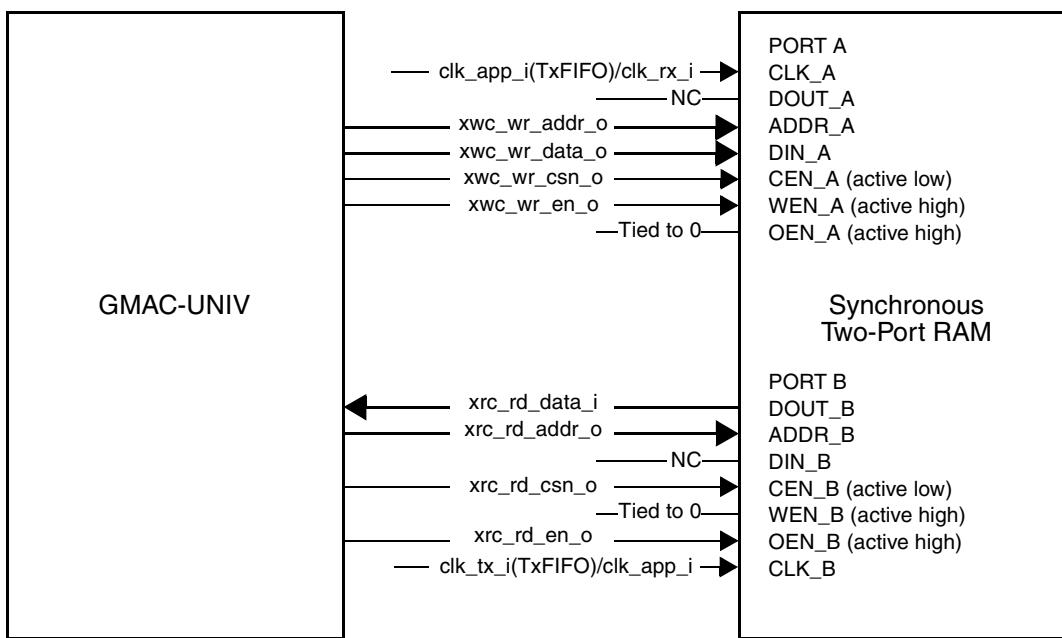
3.5.3 Interfacing With External Two-Port RAMs

The GMAC-UNIV core provides a generic memory interface for easy interconnection to any vendor-specific, two-port RAM for implementing the Tx FIFO and Rx FIFO. This section describes how GMAC-UNIV can be interconnected with different types of two-port RAMs.

The two memory interfaces in GMAC-UNIV provided for connectivity with Transmit and Receive two-port RAMs are identical. Hence, the connections with the Tx FIFO and Rx FIFO RAMs are illustrated in the following sections in a single diagram for each type of RAM. The “x” variable in the signal names can be replaced with a “t” for the Tx FIFO interface and an “r” for the Rx FIFO interface.

3.5.3.1 Low-Power, Synchronous Two-Port SRAM Connection

The synchronous, two-port SRAM shown in [Figure 3-45](#) consists of two ports, Port A and Port B, which can be used for both read and write operations. The GMAC-UNIV (configured for synchronous RAM interface) uses this SRAM as an asynchronous FIFO for transferring data from one clock domain to another clock domain. Thus, a port (Port A, for example) is used to write the frame data and the other port is used to read the frame data, with the writes and reads happening in different clock domains. (Consequently, the Read data (DOUT_A) from Port A and write data (DIN_B) to Port B are left unconnected in [Figure 3-45](#).)

Figure 3-45 Low Power, Synchronous Two-Port SRAM Connection

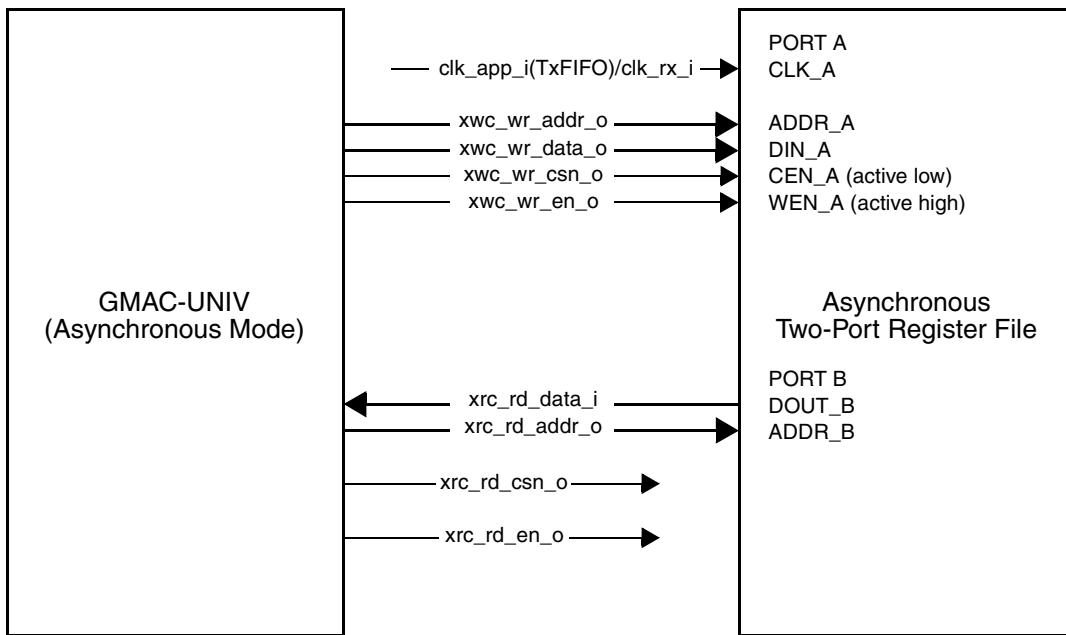
The SRAM chip enable (CEN) and write enable (WEN) are connected to chip select (*_csn_o) and write enable (*_wr_en_o) signals from the GMAC-UNIV core. Based on the active states of CEN and WEN, GMAC-UNIV memory interface signals are directly connected or have inverted polarity. Connecting the chip select signal ensures that the SRAM I/O drivers are turned off when read/write operation is not scheduled and thereby reduces power consumption.

If the memory module (such as Xilinx FPGA Block Select RAM) has a single enable for the read port (No Output Enable), xrc_rd_csn_o must be connected to the enable, with polarity taken into consideration.

3.5.3.2 Low Power, Asynchronous Two-Port Register File Connection

As shown in [Figure 3-46](#), an asynchronous, two-port register file has two ports: Port A for write operation and Port B for read operation. The write operation takes effect with regard to the Port A clock, but the read operation is asynchronous and governed only by the read address.

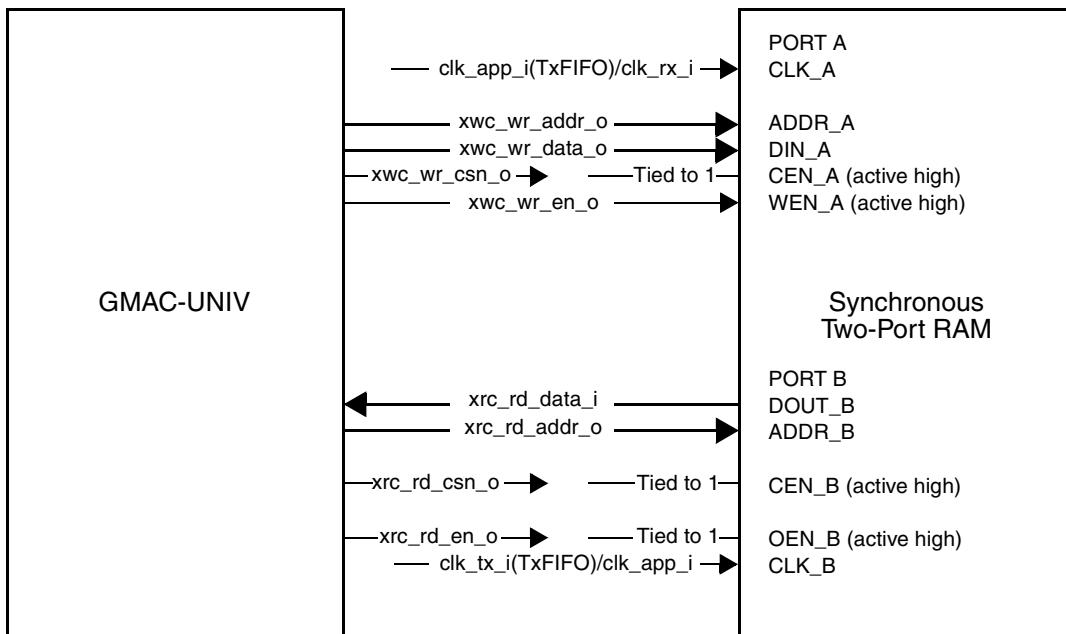
Figure 3-46 Low Power, Asynchronous Two-Port Register File Connection



3.5.3.3 High-Speed, Synchronous Two-Port SRAM Connection

In applications where power consumption is not a factor and memory access times are too critical, the chip enable and read enable signals can always be tied to active high inputs. In such applications, the generic memory interface signals can be connected to synchronous SRAM as shown in Figure 3-47. The unused outputs of the generic memory interface are also shown as floating outputs.

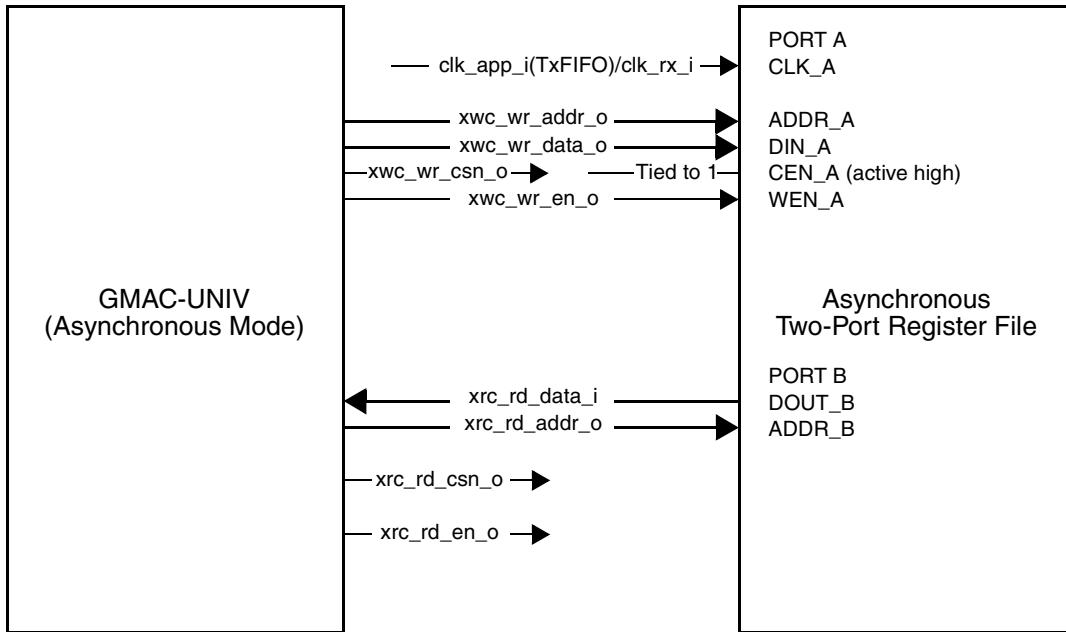
Figure 3-47 High-Speed, Synchronous Two-Port SRAM Connection



3.5.3.4 High-Speed, Asynchronous Two-Port RAM Connection

Figure 3-48 shows the interconnection for achieving high-speed access times for two-port asynchronous RAMs/register files.

Figure 3-48 High-Speed, Asynchronous Two-Port RAM Connection



3.6 GMAC Core

The GMAC core supports many interfaces towards the PHY chip. The PHY interface can be selected only once after reset. The GMAC core communicates with the application side with the MAC Transmit Interface (MTI), MAC Receive Interface (MRI) and the MAC Control Interface (MCI).

3.6.1 Transmission

Transmission is initiated when the MTL Application pushes in data with the SOF (mti_sof_i) signal asserted. When the SOF signal is detected, the GMAC accepts the data and begins transmitting to the GMII/MII. The time required to transmit the frame data to the GMII/MII after the Application initiates transmission varies, depending on delay factors like IFG delay, time to transmit preamble/SFD, and any back-off delays for Half-Duplex mode. Until then, the GMAC does not accept the data received from MTL by de-asserting the mti_rdy_o signal.

After the EOF (mti_eof_i) is transferred to the GMAC Core, the core completes the normal transmission and gives the Status of Transmission to the MTL. If a normal collision (in half-duplex mode) occurs during transmission, the GMAC core makes valid the Transmit Status to the MTL. It then accepts and drops all further data until the next SOF is received. The MTL block should retransmit the same frame from SOF on observing a Retry request (in the Status) from the GMAC.

The GMAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. During the normal transfer of a frame from MTL, if the GMAC receives a SOF without getting an EOF for the previous frame, then it ignores the SOF and considers the new frame as continuation of the previous frame.

The following six modules constitute the transmission function of the GMAC:

- ❖ Transmit Bus Interface Module
- ❖ Transmit Frame Controller Module
- ❖ Transmit Protocol Engine Module
- ❖ Transmit Scheduler Module
- ❖ Transmit CRC Generator Module
- ❖ Transmit Flow Control Module

3.6.1.1 Transmit Bus Interface Module

The Transmit Bus Interface (TBU) module interfaces the transmit path of the GMAC core with the external frame with a FIFO interface. It accepts data in 32/64/128-bit-wide bus and runs on the clock clk_tx_i of the GMII interface. The TBU module performs the following functions:

- ❖ Outputs the (32-bit) Transmit Status to the application at the end of normal transmission or collision.
- ❖ Outputs the Transmit Snapshot register value to the mti_timestamp_o signal and asserts the mti_status_valid signal.
- ❖ Performs the Endian conversion of data bus by swapping the byte lanes and corresponding byte enables.

3.6.1.2 Transmit Frame Controller Module

The Transmit Frame Controller (TFC) module consists of two registers to hold data, byte enables, and the last data control received from the TBU. The register provides a buffer between the Application and the Transmit Protocol Engine (TPE) to regulate data flow as well as converts the input data into an 8-bit bus towards the TPE.

When the number of bytes received from the Application falls below 60 (DA+SA+LT+DATA), the state machine that interfaces with the TBU automatically appends zeros to the transmitting frame to make the data length exactly 46 bytes (provided mti_dispad_i is LOW) to meet the minimum data field requirement of IEEE 802.3. The GMAC can be programmed not to append any padding through the sideband signal, mti_dispad_i, from the MTI.

The cyclic redundancy check (CRC) for the Frame Check Sequence (FCS) field is calculated before transmission to the TPE module. This value is computed by CTX module. The TFC module receives the computed CRC and appends it to the data being transmitted to the TPE module. When the GMAC is programmed (through a sideband signal mti_discrc_i from the Application interface) to not append the CRC value to the end of Ethernet frames, the TFC module ignores the computed CRC and transmits only the data received from the TBU module to the TPE module. An exception to this rule is that when the GMAC is programmed to append pads for frames (DA+SA+LT+DATA) less than 60 bytes sent by the TBU module, the TFC module appends the CRC at the end of padded frame irrespective of the value on the mti_discrc_i signal.

The TFC converts the data received on the 32/64/128-bit interface from the TBU into 8-bit data for the TPE module.

3.6.1.3 Transmit Protocol Engine Module

The Transmit Protocol Engine (TPE) module consists of a transmit state machine that controls the operation of Ethernet frame transmission. The transmit state machine of this module contains the following features to meet the IEEE 802.3/802.3z specifications.

- ❖ Generates preamble and SFD

- ❖ Generates jam pattern in Half-Duplex mode
- ❖ Generates carrier extension in Half-Duplex (only in GMII mode)
- ❖ Supports frame bursting in Half-Duplex (only in GMII mode)
- ❖ Supports jabber timeout
- ❖ Supports flow control for half-duplex mode (back pressure)
- ❖ Generates transmit frame status
- ❖ Contains timestamp snapshot logic for IEEE 1588 support

When a new frame transmission from the TFC is requested, the transmit state machine sends out the preamble and SFD, followed by the data received. The preamble is defined as 7 bytes of 8'b10101010 pattern, and the SFD is defined as 1 byte of 8b'10101011 pattern.

The collision window is defined as 1 slot time (512-bit times for 10/100 Mbps Ethernet and 4,096 bit times for 1,000 Mbps Ethernet). The jam pattern generation is applicable only to half-duplex mode, not to full-duplex mode. In full-duplex mode, the transmit state machine ignores the phy_col_i signal from the PHY.

In MII mode, if a collision occurs any time from the beginning of the frame to the end of the CRC field, the transmit state machine sends a 32-bit jam pattern of 32'h55555555 on the MII to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, the transmit state machine completes the transmission of preamble and SFD, and then sends the jam pattern.

In GMII mode, if a collision occurs any time between the beginning of the frame and the end of the extension field, the transmit state machine sends a 32-bit jam pattern of 32'h55555555 on the GMII to inform all other stations of the collision. If the collision is seen during the preamble transmission phase, the transmit state machine completes the transmission of preamble and SFD, and then sends the jam pattern.

If the collision occurs after the collision window and before the end of the FCS field (or the end of Burst if the Frame Burst mode is enabled), the transmit state machine sends a 32-bit jam pattern and sets the late collision bit in the transmit frame status.



Note At the GMII/MII interface, collision signal (phy_col_i) being asynchronous is checked by the transmitter after it is double-synchronized to clk_tx domain. This additional latency delays the recognition of collision or late-collision event. When the output of phy_col_i synchronizer is asserted after the complete frame is transmitted, it is not recognized as collision even if the COL signal is high at the GMII interface before the end of transmission. Similarly, an assertion of COL signal at the GMII input at the last byte of normal collision window might be identified as late collision because of the synchronizer delay.

In GMII half-duplex mode (1,000 Mbps), the Transmit state machine ensures that all valid carrier events exceed a slot time of 4,096 bit times. To accomplish this, any transmit frame shorter than 512 bytes from the TFC module is extended using a carrier extension. On GMII, this is signaled to the PHY by asserting phy_txer_o, de-asserting phy_txen_o, and setting phy_txd_o[7:0] to 8'0F.

When the Frame Burst mode is enabled, only the first frame of the burst is carrier extended, as necessary. The carrier extension is not applicable for MII half-duplex and GMII/MII full-duplex modes. When the Frame Burst mode is enabled, the transmit state machine transmits a burst of frames (as long as frames are available from the TFC module) without releasing the carrier of the PHY. To accomplish this, the state machine inserts the carrier extension for a minimum IFG period (96 bit times) between the frames. The transmit state machine continues to burst frames as long as additional frames are available from the TFC module and a burst limit of 8192 byte times has not been exceeded. If the additional frame is not available at

the end of the IFG period in the middle of the burst, the transmit state machine releases the carrier by deasserting the phy_txer_o and phy_txen_o signals on the GMII.

Frame bursting is applicable only for GMII half-duplex mode and is not applicable in the MII and GMII/MII full-duplex modes.

The TPE module maintains a jabber timer (only in 10/100-Mbps mode) to stop the transmission of Ethernet frames if the TFC module transfers more than 2,048 (default) bytes. The time-out is changed to 10,240 bytes when the Jumbo frame is enabled.

The Transmit state machine uses the deferral mechanism for the flow control (Back Pressure) in half-duplex mode. When the Application requests to stop receiving frames, the Transmit state machine sends a JAM pattern of 32 bytes whenever it senses a reception of a frame, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. The Application requests the flow control through a sideband signal mti_flowctrl_i (or by setting BPA bit of Register6). If the application requests a frame to be transmitted, then it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur) then the remote stations abort their transmissions because of excessive collisions.

If IEEE 1588 time stamping is enabled for the transmit frame, this block takes a snapshot of the system time when the SFD is put onto the transmit GMII/MII bus. The system time source is either an external input or internally generated, according to the configuration selected.

3.6.1.4 Transmit Scheduler Module

The Transmit Scheduler (STX) module is responsible for scheduling the frame transmission on GMII/MII. It provides an enable signal to the TPE module after satisfying the IFG and back-off delays. The STX module performs the following functions:

- ❖ Maintains the inter-frame gap between two transmitted frames

The STX module maintains an idle period of the configured inter-frame gap (IFG bits of Register 0) between any two transmitted frames. If frames from the TFC arrive at the TPE module sooner than the configured IFG time, the TPE module waits for the enable signal from the STX module before starting the transmission on the GMII/MII. The STX module starts its IFG counter as soon as the carrier signal of the GMII/MII goes inactive. At the end of programmed IFG value, the module issues an enable signal to the TPE module in full-duplex mode.

In half-duplex mode and when IFG is configured for 96 bit times, the STX module follows the rule of deference specified in the IEEE 802.3 specification, *Section 4.2.3.2.1*. The module resets its IFG counter if a carrier is detected during the first two-thirds (64-bit times for all IFG values) of the IFG interval. If the carrier is detected during the final one third of the IFG interval, the STX module continues the IFG count and enables the transmitter after the IFG interval.

- ❖ Implements the Truncated Binary Exponential Back-off algorithm

The STX module implements the Truncated Binary Exponential Back-off algorithm when it operates in the half-duplex mode.

3.6.1.5 Transmit CRC Generator Module

The Transmit CRC Generator (CTX) module interfaces with the TFC module to generate CRC for the FCS field of the Ethernet frame. The TFC module sends the frame data and any necessary padding to the CTX module through an 8-bit interface.

The CTX module calculates the 32-bit CRC for the FCS field of the Ethernet frame. The encoding is defined by the following generating polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The CTX module gets the Ethernet frame's byte data from the TFC module (DA + SA + LT + DATA + PAD) qualified with a Data Valid signal. The TFC also indicates to the CTX when to reset the previously calculated CRC and to start the new CRC calculation for the coming frame. The TFC module issues the start command before sending the new frame data for calculation. The calculated CRC is valid on the next clock after the data is received.

In the GMII mode, the Data Valid signal is valid for every clock, from the first data byte through the last data byte. In MII mode, this signal is valid every alternate clock.

3.6.1.6 Transmit Flow Control Module

The Transmit Flow Control (FTX) module generates Pause frames and transmit them to the TFC module as necessary, in full-duplex mode. The TFC module receives the Pause frame from the FTX module, appends the calculated CRC, and sends the frame to the TPE module. The Application can request the FTX module to send a Pause frame by using either of the following ways:

- ❖ By setting the FCB bit in the [Register 6 \(Flow Control Register\)](#).

If the Application has requested the flow control by setting the bit 0 (FCB) of Register 6 (Flow Control Register), the FTX module generates and transmits a single Pause frame to the TFC module. The value of the Pause Time in the generated frame contains the programmed Pause Time value in Register 6 (Flow Control Register). To extend the pause or end the pause prior to the time specified in the previously transmitted Pause frame, the application must request another Pause frame transmission after programming the Pause Time register with appropriate value.

- ❖ By asserting the mti_flowctrl_i signal in response to the receive FIFO full conditions (packet buffer).

If the Application has requested the flow control by asserting the mti_flowctrl_i signal, the FTX module generates and transmits a Pause frame to the TFC module. The value of the Pause Time in the generated frame contains the programmed Pause Time value in the Register 6 (Flow Control Register). The FTX module monitors the mti_flowctrl_i signal. If it remains asserted at a configurable number of slot-times (bits [5:4] PLT of Register 6 (Flow Control Register)) before this Pause-time runs-out, a second Pause frame is transmitted to the TFC module. The process is repeated as long as the mti_flowctrl signal remains asserted.

If the mti_flowctrl_i signal goes inactive prior to the sampling time, the FTX module transmits a Pause frame with zero Pause Time to indicate to the remote end that the receive buffer is ready to receive new data frames.

3.6.2 MAC Transmit Interface Protocol

The MAC Transmit Interface (MTI) connects the application (MTL module in the GMAC) with the GMAC to provide the Ethernet data for transmission.

The application initiates the Ethernet frame transmission by writing the first data (mti_sof_i) of the frame to the GMAC, provided the GMAC is ready to accept data (indicated by mti_rdy_o). Each valid data transfer is indicated by an active mti_valid_i signal. The Application can push-in data as long as the GMAC core is ready to accept it. Thus the Application can assume a successful data transfer if the mti_rdy_o signal is asserted.

The application indicates the last data of the frame by asserting the end-of-packet mti_eof_i signal, along with the last data and byte enables (mti_be_i). The number of valid byte lanes for the last transfer is decoded with the mti_be_i signal (See [Table 3-1](#)). At the end of normal transmission of the Ethernet frame, the GMAC

Core outputs the transmit status (mti_txstatus_o, described in [Table 3-4](#)) to the application. This is indicated by an active mti_txstatus_val_o signal.

Table 3-1 mti_be_i Function for 128-Bit Bus

mti_be_i	Valid byte lanes (Little-Endian)	Valid byte lanes (Big-Endian)
0000	mti_data_i[7:0] – Byte lane 0	mti_data_i[127:120] – Byte lane 15
0001	mti_data_i[15:0] – Byte lane 0-1	mti_data_i[127:112] – Byte lane 15-14
0010	mti_data_i[23:0] – Byte lane 0-2	mti_data_i[127:104] – Byte lane 15-13
...		
1110	mti_data_i[119:0] – Byte lane 0-14	mti_data_i[127:8] – Byte lane 15-1
1111	mti_data_i[127:0] – Byte lane 0-15	mti_data_i[127:0] – Byte lane 15-0

Table 3-2 mti_be_i Function for 64-Bit Bus

mti_be_i	Valid byte lanes (Little-Endian)	Valid byte lanes (Big-Endian)
000	mti_data_i[7:0] – Byte lane 0	mti_data_i[63:56] – Byte lane 7
001	mti_data_i[15:0] – Byte lane 0-1	mti_data_i[63:48] – Byte lane 7-6
010	mti_data_i[23:0] – Byte lane 0-2	mti_data_i[63:40] – Byte lane 7-5
...		
110	mti_data_i[55:0] – Byte lane 0-6	mti_data_i[63:8] – Byte lane 7-1
111	mti_data_i[63:0] – Byte lane 0-7	mti_data_i[63:0] – Byte lane 7-0

Table 3-3 mti_be_i Function for 32-Bit Bus

mti_be_i	Valid byte lanes (Little-Endian)	Valid byte lanes (Big-Endian)
00	mti_data_i[7:0] – Byte lane 0	mti_data_i[31:24] – Byte lane 3
01	mti_data_i[15:0] – Byte lane 0-1	mti_data_i[31:16] – Byte lane 3-2
10	mti_data_i[23:0] – Byte lane 0-2	mti_data_i[31:8] – Byte lane 3-1
11	mti_data_i[31:0] – Byte lane 0-3	mti_data_i[31:0] – Byte lane 3-0

If the frame transmission is not successful (because of underflow, collision, jabber timeout, or excessive deferral events), the GMAC core asserts the transmit status even before the EOF is received. The Application takes the appropriate action as per the status. The GMAC drops all further data input to it until the next SOF.

Table 3-4 Transmit Status at the GMAC Core Interface

Bit	Description
31	Frame Retry Requested When set, this bit indicates a request from the MAC transmitter for retransmission of the entire frame from the Application. This bit has higher priority than the Frame Aborted (Bit 0). The GMAC core expects a retransmission of the frame when this bit is set irrespective of the value of Frame Aborted bit.
30	TSS: Timestamp Status When set, this bit indicates that the MAC transmitter has captured the transmitted frame's IEEE1588 time stamp. The captured timestamp is available, along with the transmit status, on the mti_timestamp_o[63:0] output. This bit is enabled only when IEEE1588 time stamping is enabled. Otherwise, it is reserved.
29	VLAN Frame When set, this bit indicates to the Application that the transmitted frame is a VLAN tagged frame (Type field equals 16'h8100).
28:27	Address Type This field indicates the type of destination address transmitted. <ul style="list-style-type: none"> • 2'b00: Unicast • 2'b01: Multicast • 2'b10: Reserved • 2'b11: Broadcast
26:13	Transmit Byte Count This 14-bit counter indicates the number of bytes transmitted.
12:9	Collision Count This 4-bit counter indicates the number of collisions that occurred before the frame was transmitted. The count is not valid when the Excessive Collisions bit is set.
8	Deferred When set, this bit indicates that the GMAC defers before transmission because of the presence of carrier.
7	Underflow When set, this bit indicates that the GMAC aborted the transmission because of data underflow. The Underflow error indicates that no more data is available for transmission in the middle of transmission.
6	Excessive Collisions When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the Configuration register is set, this bit is set after the first collision and the transmission of the frame is aborted.
5	Late Collision When set, this bit indicates that the frame transmission was aborted because of a collision occurring after the collision window (64 bytes including Preamble in MII mode and 512 bytes including Preamble and Carrier Extension in GMII mode). Not valid if the Underflow error is set.

Table 3-4 Transmit Status at the GMAC Core Interface (Continued)

Bit	Description
4	Excessive Deferral When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 in 1000Mbps mode or if Jumbo-frame is enabled), if the deferral check (DC) bit is set high in the Control register.
3	Loss of Carrier When set, this bit indicates that the loss of carrier occurred during the frame's transmission (i.e., the gmii_crs_i signal was inactive for one or more transmission clock periods during frame transmission). This is valid only for the frames transmitted without collision and when the GMAC operates in Half-Duplex mode.
2	No Carrier When set, this bit indicates that the carrier signal from the PHY was not present at the end of preamble transmission, and is valid only when the GMAC operates in Half-Duplex mode.
1	Jabber Timeout When set, this bit indicates the MAC transmitter has experienced a jabber timeout. This bit is be set only when the JB bit of the Configuration register is not disabled (de-asserted).
0	Frame Aborted When set, this bit indicates that the transmission of the current frame has been aborted. The reason for the abort could be because of one or more of the following conditions. <ul style="list-style-type: none"> • Jabber Timeout • No Carrier • Loss of Carrier • Excessive Deferral • Late Collision • Retry Count exceeds the attempt limit • Data underrun When reset, this bit indicates that the current frame was successfully transmitted on the GMII/MII (provided Bit 31 is zero).

3.6.3 MAC Transmit Timing

Timing specifications for the MAC Core (GMAC-CORE) interface signals in the transmit path are described in detail in the following sections:

- ❖ [MAC Transmit Interface \(MTI\)](#)
- ❖ [MTI With Time Stamping](#)
- ❖ [Collision During Transmission](#)
- ❖ [Underflow During Transmission](#)

3.6.3.1 MAC Transmit Interface (MTI)

[Figure 3-49](#) shows the timing specifications for the MAC Transmit Interface signals. The frame transmission handshake mechanism is as follows:

1. The application can initiate a frame transfer by asserting mti_val_i along with valid data on mti_data_i. Signal mti_sof_i should be asserted when the start of frame is being transferred.

2. The MAC core accepts the data as long as mti_rdy_o is active. The application should update the data (or de-assert mti_val_i) on observing an active mti_rdy_o signal to transfer data in bursts.
 3. The MAC starts transmitting the frame on the GMII as soon as it gets hold of the channel. At any point during the transfer, the MAC can alert the application to stop the transfer by de-asserting mti_rdy_o. This occurs when the MAC's internal FIFO is full.
 4. When the application wants to transfer the last bytes of a frame, mti_eof_i should be asserted along with the byte enables (mti_be_i), indicating the valid number of data bytes on the bus. It is mandatory that all data transfers except the EOF transfer have the byte enables as all-ones.
 5. After transferring the EOF, the application should wait until the MAC gives the transmission status of that frame. The MAC also de-asserts mti_rdy_o after receiving the EOF, so that the application cannot transfer the next frame.
 6. The MTI gives the transmit status of the frame back to the application through mti_txstatus_o by asserting mti_txstatus_val_o (see [Figure 3-50](#)). The application has to accept it immediately.
 7. The application can start the transmission of the next frame.

[Figure 3-50](#) also shows the timing of the GMII signals during the transmission of a frame, illustrating how the GMAC meets the IFG timing on the GMII during frame-burst transmission.

Figure 3-49 MAC Transmit Interface (MTI) Timing (1 of 2)

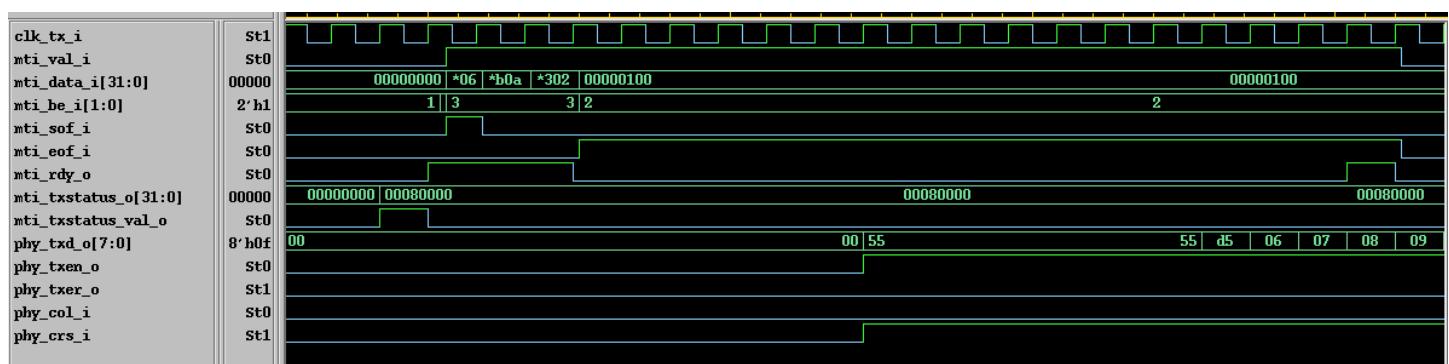
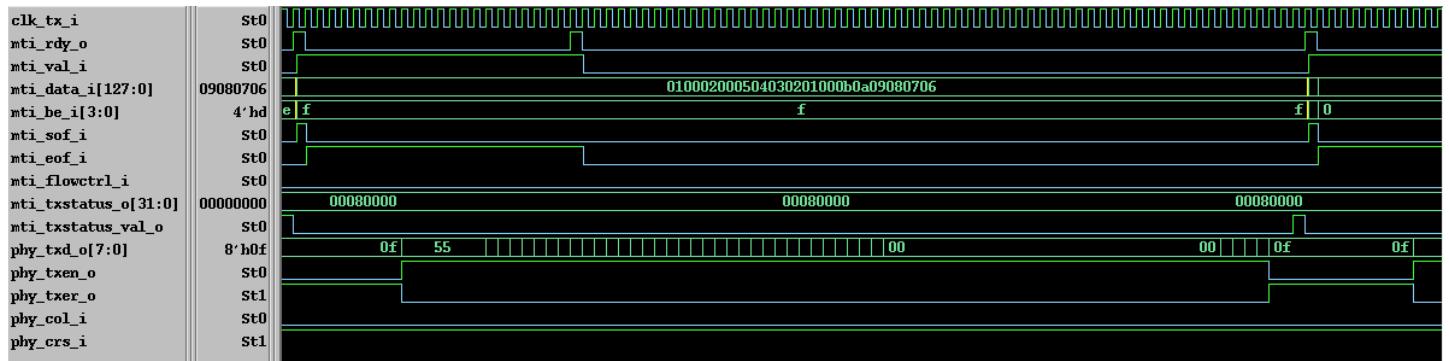


Figure 3-50 MAC Transmit Interface (MTI) Timing (2 of 2)

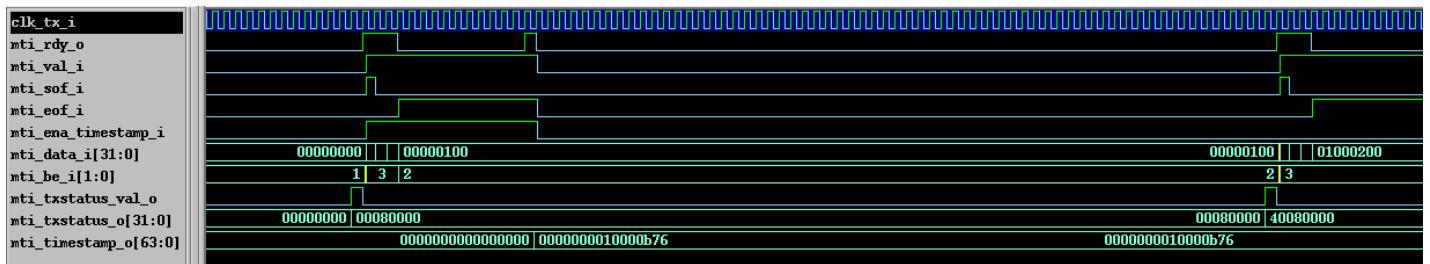


3.6.3.2 MTI With Time Stamping

Figure 3-51 shows the MTI's timing when IEEE 1588 time stamping is enabled. The `mti_ena_timestamp_i` signal is sampled, along with the `mti_sof_i`, for an input frame. When that frame's status (0x40080000) is

given on mti_txstatus_o, the timestamp captured for that frame is also output on mti_timestamp_o. The time stamp's validity is indicated by mti_txstatus_o[30] being asserted when mti_txstatus_val_o is high.

Figure 3-51 MTI Timing With IEEE 1588 Time Stamping Enabled



3.6.3.3 Collision During Transmission

[Figure 3-52](#) and [Figure 3-53](#) show how the MAC core behaves because of collisions occurring on the GMII interface.

1. The MAC accepts data from the application on the MTI and starts transmission in the half-duplex mode.
2. When a collision is recognized, the MAC de-asserts mti_rdy_o in order to stop taking in more data from the application on the MTI. In the figure, this signal is already low because the internal FIFOs are full.
3. After the transmission of the JAM pattern (4 bytes of 0x55), the MAC gives the transmission status (mti_txstatus_o) to the application by asserting mti_txstatus_val_o. It requests the application to retry the frame transfer by setting bit 31 of mti_txstatus_o.
4. In the next clock cycle, the MAC asserts mti_rdy_o, indicating that it is ready to accept the next frame for transmission. The MAC (MTI) ignores or drops all data given after the assertion of mti_txstatus_val_o until the application indicates a start of frame transfer by asserting mti_sof_i.

[Figure 3-53](#) shows how the MAC transmitter starts a JAM pattern only after the transmission of the SFD even though the collision was recognized well in advance during the preamble transmission.

Figure 3-52 Collision During Transmission (1 of 2)

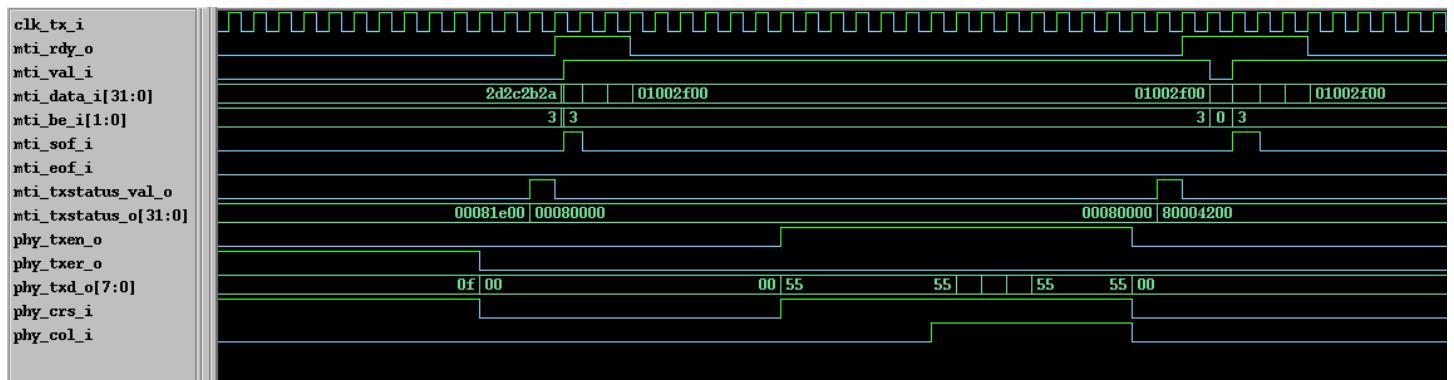
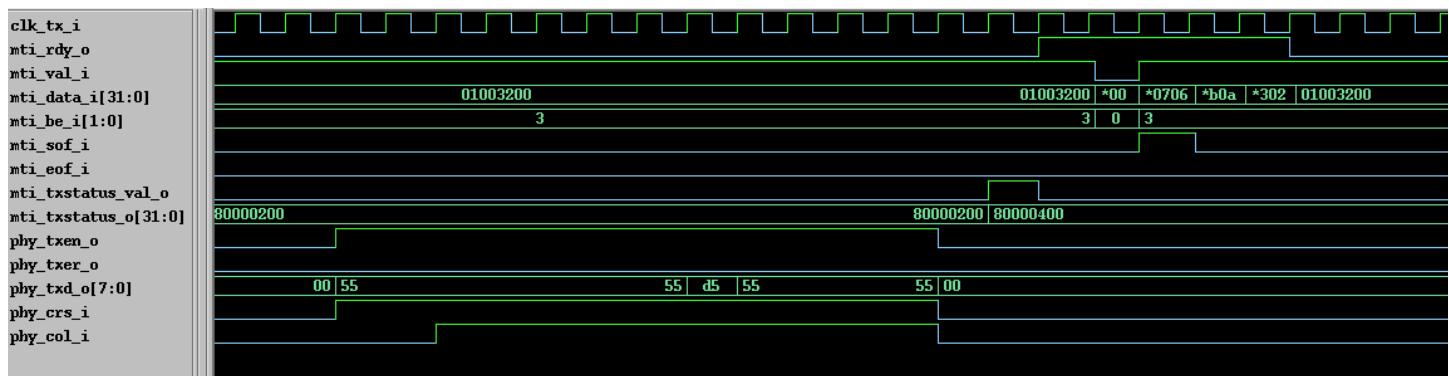
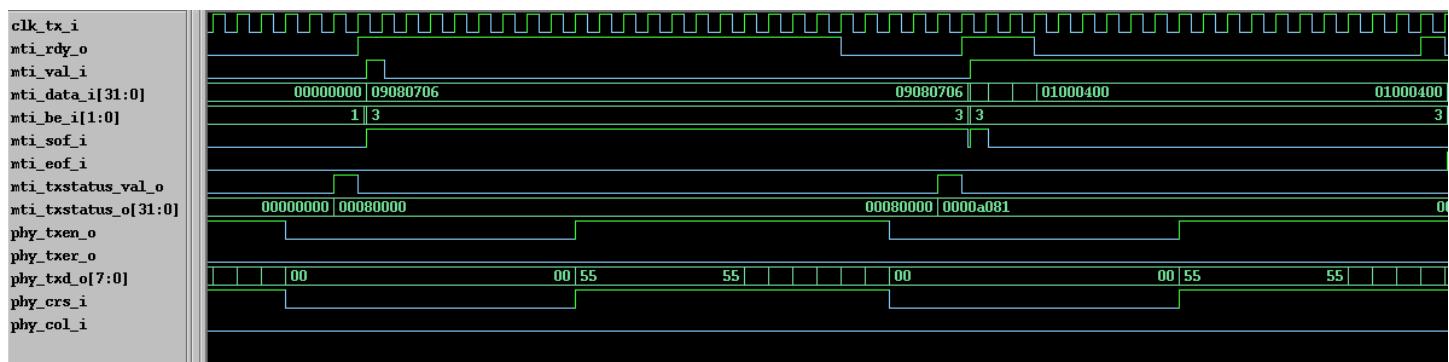


Figure 3-53 Collision During Transmission (2 of 2)

3.6.3.4 Underflow During Transmission

Figure 3-54 shows the timing diagram for a frame abort because of insufficient data being input to the MAC.

1. The MAC schedules and starts a frame transmission as soon it receives the start of the frame on the MTI.
2. The MAC finds that it does not have the end-of-frame data because the application did not input data on the MTI after the start-of-frame and it de-asserts **mti_rdy_o**.
3. The MAC aborts transmission after 5 bytes (the fifth byte is a dummy byte) in the figure.
4. The MAC gives the transmit abort status (0x0000a081) to the application.
5. The MAC indicates that it is ready to accept new data in the next clock cycle by asserting **mti_rdy_o**.

Figure 3-54 Underflow During Transmission Timing

3.6.4 Reception

A receive operation is initiated when the GMAC detects an SFD on the GMII/MII. The core strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC for the frame. The received frame is stored in a shallow buffer until the address filtering is performed. The frame is dropped in the core if it fails the address filter.

The following are the functional blocks in the Receive path of the GMAC core.

- ❖ Receive Protocol Engine Module (RPE)
- ❖ Receive CRC Module (CRX)
- ❖ Receive Frame Controller Module (RFC)

- ❖ Receive Flow Control Module (FRX)
- ❖ Receive IP Checksum checker (IPC)
- ❖ Receive Bus Interface Unit Module (RBU)
- ❖ Address Filtering Module (AFM)

3.6.4.1 Receive Protocol Engine Module

The RPE consists of the receive state machine which strips the preamble, SFD and carrier extension of the received Ethernet frame (in half-duplex 1000-Mbps mode). Once the phy_rxrdv_i signal of the GMII/MII becomes active, the RPE's receive state machine begins hunting for the SFD field from the receive modifier logic. Until then, the state machine drops the receiving preambles. Once the SFD is detected, the state machine begins sending the data of the Ethernet frame to the RFC module, beginning with the first byte following the SFD (destination address).

If IEEE 1588 time stamping is enabled, the RPE takes a snapshot of the system time when any frame's SFD is detected on the GMII/MII. Unless the MAC filters out and drops the frame, this time stamp is passed on to the application.

In MII mode, the RPE converts the received nibble data into bytes, then forwards the valid frame data to the RFC module.

The receive state machine of the RPE module decodes the Length/Type field of the receiving Ethernet frame. If the Length/Type field is less than 600 (hex) and if the MAC is programmed for the auto crc/pad stripping option, the state machine sends the data of the frame up to the count specified in the Length/Type field, then starts dropping bytes (including the FCS field). The state machine of the RPE module decodes the Length/Type field and checks for the Length interpretation.



Note In Audio Video (AV) mode, when you select additional Rx channels, the frames that are less than or equal to 16 bytes in length after pad stripping, always get dropped inside the MAC receiver. This happens even if the frames have passed the address filter and have no CRC error.

If the Length/Type field is greater than or equal to 600 (hex), the RPE module sends all received Ethernet frame data to the RFC module, irrespective of the value on the programmed auto CRC stripping option.

As a default, the GMAC is programmed for watchdog timer to be enabled, that is, frames above 2,048 (10,240 if Jumbo Frame is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. This feature can be disabled by programming the GMAC Configuration register, Watchdog Disable. However, even if the watchdog timer is disabled, frames greater than 16 KB in size are cut off and a watchdog time-out status is given.

The GMAC supports loopback of transmitted frames onto its receiver. As a default, the GMAC loopback function is disabled, but this feature can be enabled by programming the GMAC Configuration register, Loopback bit. The transmit and receive clocks can have an asynchronous timing relationship, so an asynchronous FIFO is used to make the loopback path of the phy_txd_o data onto the receive path. The asynchronous FIFO is 10 bits (6 bits in 10/100-Mbps mode) wide to accommodate phy_txd_o, phy_txen_o, and phy_txer_o. The FIFO is five deep in 1000-Mbps mode (nine deep in 10/100-Mbps mode) and free-running to write on the write clock (clk_tx_i) and read on every read clock (clk_rx_i).

The write and read pointers gets re-initialized to have an offset of 2 (4 in 10/100-Mbps mode) at the start of each frame read out of the FIFO. This helps to avoid overflow/underflow during the transfer of a frame, and ensures that the overflow/underflow occurs only during the IFG period between the frames. Please note that the FIFO depth of five/nine is sufficient to prevent data corruption for frame sizes up to 9,022

bytes with a difference of 200 ppm between the (G)MII Transmit and Receive clock frequencies. Hence, bigger frames should not be looped back, as they may get corrupted in this loopback FIFO.

At the end of every received frame, the RPE module generates received frame status and sends it to the RFC module. Control, missed frame, and filter fail status are added to the receive status in the RFC module.



Note In half-duplex mode, the first frame in a burst should be at least slot time in length. If the first frame is smaller than the slot time, the MAC considers the bytes received up to the slot time as first frame, which results in CRC error for the frame. The receive descriptor is closed with CRC error status.

3.6.4.2 Receive CRC Module

The Receive CRC (CRX) interfaces to the RPE module to check for any CRC error in the receiving frame. This module calculates the 32-bit CRC for the received frame that includes the Destination address field through the FCS field. The encoding is defined by the following generating polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The module gets the data from the RPE module (DA+SA+LT+DATA+PAD+FCS). The RPE module also sends a control signal that indicates the validity of the data. Irrespective of the auto pad/CRC strip, the CRX module receives the entire frame to compute the CRC check for received frame. As a note on the auto pad/CRC strip settings, the entire frame is not transferred between the RPE and RFC 8-bit interface.

3.6.4.3 Receive Frame Controller Module

The Receive Frame Controller (RFC) receives the Ethernet frame data and status from the RPE module. The RFC module consists of a FIFO of parameterized depth (default set to 4 deep and 37 bits wide) and two state machines for writing and reading the FIFO. The FIFO holds the received Ethernet frame data and byte enables, along with a control bit to indicate the last data. The state machines manage the FIFO and provide a frame buffering for the receiving Ethernet frame from the RPE module. The main functions of the RFC module are:

- ❖ Data path conversion, which converts the 8-bit data to 32-bit data to the RBU module.
- ❖ Frame filtering
- ❖ Attaching the calculated IP Checksum input from IPC.
- ❖ Update the Receive Status and forward to RBU.

If the RA bit of the GMAC CSR Frame Filter register is set, the RFC module initiates the data transfer to the RBU module as soon as 4 bytes of Ethernet data are received from the RPE module. At the end of the data transfer, the RFC module sends out the received frame status that includes the frame filter bits (SA Filterfail and DAFilterfail) and status from the RFC module. These bits are generated based on the filter-fail signals from the AFM module. This status bit indicates to the Application whether the received frame has passed the filter controls (both address filter and Frame Filter controls from CSR). The RFC module does not drop any frame on its own in this mode.

If the RA bit is reset, the RFC module performs frame filtering based on the destination/source address (the Application still needs to perform another level of filtering if it decides not to receive any bad frames like runt, CRC error frames, etc. The RFC module waits to receive the first 14 bytes of received data (type field) from the RPE module. Until then, the module does not initiate any transfers to the RBU module. After receiving the destination/source address bytes, the RFC checks the filter-fail signal from the AFM module for an address match. On detecting a filter-fail from AFB, the frame is dropped at the RFC module and not transferred to the Application.

On a delayed filter response from the AFM (this can only occur if you change the AFM logic), the RFC module waits until the FIFO is full, and then proceeds with the frame transfer to the RBU module. However, it still takes the delayed response from the AFM module and if it is a (DA/SA) filter failure, then it drops the rest of the frame and sends the Rx Status Word (with zero frame-length, CRC Error and Runt Error bits set) immediately indicating the filter-fail. If there is no response from the AFM until the end of frame is transmitted, the filter fail status in the Rx Status Word is updated accordingly.

When the optional PMT module is present and configured for power-down mode, all received frames are dropped by this block, and are not forwarded to the application.

3.6.4.4 Receive Flow Control Module

The Receive Flow Controller (FRX) detects the receiving Pause frame and pauses the frame transmission for the delay specified within the received Pause frame. The FRX module is enabled only in Full-Duplex mode. The Pause frame detection function can be enabled or disabled with the RFE bit of GMAC CSR Register 6.

Once the receive flow control is enabled, the FRX module begins monitoring the received frame destination address for any match with the multicast address of the control frame (48'h0180C2000001). If a match is detected, the FRX module indicates to the RFC module, that the destination address of the received frame matches the reserved control frame destination address. The RFC module then decides whether or not to transfer the received control frame to the Application, based on the (PCF) bit setting of GMAC CSR Register 1 (Filter register).

The FRX module also decodes the Type, Op-code, and Pause Timer field of the receiving control frame. At the end of received frame, the FRX module gets the received frame status from RPE. If the byte count of the status indicates 64 bytes, and if there is no CRC error, the FRX module requests the MAC transmitter to pause the transmission of any data frame for the duration of the decoded Pause Time value, multiplied by the slot time (64 byte times for both 10/100-Mbps mode and 1000-Mbps mode). Meanwhile, if another Pause frame is detected with a zero Pause Time value, the FRX module resets the Pause Time and gives another pause request to the Transmitter. If the received control frame matches neither the Type field (16'h8808), Opcode (16'h00001), nor byte length (64 bytes), or if there is a CRC error, the FRX module does not generate a Pause request to Transmitter.

In the case of a pause frame with a multicast destination address, the RFC filters the frame based on the address match from the FRX module. For a pause frame with a unicast destination address, the filtering in the FRX module depends on whether the DA matched the contents of the MAC Address Register 0 and the UP Bit of GMAC Core Register 6 is set (detecting a pause frame even with a unicast destination address). The PCF register bits (Bit [7:6] of GMAC Register 1) controls the filtering for control frames in addition to the Address filter module.

3.6.4.5 Receive Bus Interface Unit Module

The Receive Bus Interface Unit (RBU) converts the 32-bit data received from the RFC module into a 32/64/128-bit FIFO protocol on the Application side. The RBU module interfaces with the Application through the MAC receive interface (MRI). This block also performs the endian conversion if the GMAC-CORE is configured for big-endian mode.

If IEEE 1588 time stamping is enabled, the RBU also outputs the time stamp captured from the received frame, along with the status on the mri_timestamp_o bus in GMAC-CORE configuration. The value on this bus is valid when the mri_eof_o signal is asserted.

3.6.4.6 Address Filtering Module

The Address Filtering (AFM) module performs the destination and source address checking function on all received frames and reports the address filtering status to the RFC module. The address checking is based

on different parameters (Frame Filter register) chosen by the Application. These parameters are inputs to the AFM module as control signals, and the AFM module reports the status of the address filtering based on the combination of these inputs. The AFM module does not filter the receive frames by itself, but reports the status of the address filtering (whether to drop the frame or not) to the RFC module. The AFM module also reports whether the receiving frame is a multicast frame or a broadcast frame, as well as the address filter status.

The AFM module probes the 8-bit receive data path between the RPE module and the RFC module and checks the destination and source address field of each incoming packet. In GMII mode, the module takes 8/14 clocks (from the start of frame) to compare the destination/source address of the receiving frame. Similarly, in MII mode the module takes 14/26 clocks (from the start of frame) to compare the destination/source address of the receiving frame. The AFM module gets the station's physical (MAC) address and the Multicast Hash table from CSR module for address checking. The CSR module provides the Frame Filter register parameters to AFM.

Unicast Destination Address Filter

The AFM supports up to 32 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of Frame Filter register is reset), the AFM compares all 48 bits of the received unicast address with the programmed MAC address for any match. Default MacAddr0 is always enabled, other addresses MacAddr1–MacAddr31 are selected with an individual enable bit. Each byte of these other addresses (MacAddr1–MacAddr31) can be masked during comparison with the corresponding received DA byte by setting the corresponding Mask Byte Control bit in the register. This helps group address filtering for the DA.

In Hash filtering mode (When HUC bit is set), the AFM performs imperfect filtering for unicast addresses using a 64-bit Hash table. For hash filtering, the AFM uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 000000 selects Bit 0 of the selected register, and a value of 111111 selects Bit 63 of the Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast frame is said to have passed the Hash filter; otherwise, the frame has failed the Hash filter.



Note A sample C routine that generates the 6-bit hash for an input DA is provided in your workspace's resources/ directory.

Multicast Destination Address Filter

The GMAC can be programmed to pass all multicast frames by setting the PM bit in the Frame Filter register. If the PM bit is reset, the AFM performs the filtering for multicast addresses based on the HMC bit of Frame Filter register. In Perfect Filtering mode, the multicast address is compared with the programmed MAC Destination Address registers (1–31). Group address filtering is also supported.

In Hash filtering mode, the AFM performs imperfect filtering using a 64-bit Hash table. For hash filtering, the AFM uses the upper 6 bits CRC of the received multicast address to index the content of the Hash table. A value of 000000 selects Bit 0 of the selected register and a value of 111111 selects Bit 63 of the Hash Table register.

If the corresponding bit is set to 1, then the multicast frame is said to have passed the Hash filter; otherwise, the frame has failed the Hash filter.

Hash or Perfect Address Filter

The DA filter can be configured to pass a frame when its DA matches either the Hash filter or the Perfect filter by setting the HPF bit of the Frame Filter register and setting the corresponding HUC or HMC bits. This configuration applies to both unicast and multicast frames. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to the received frame.

Broadcast Address Filter

The AFM does not filter any broadcast frames in the default mode. However, if the GMAC is programmed to reject all broadcast frames by setting the DBF bit in the Frame Filter register, the DAF module asserts the Filter fail signal to RFC, whenever a broadcast frame is received. This tells the RFC module to drop the frame.

Unicast Source Address Filter

The GMAC can also perform a perfect filtering based on the source address field of the received frames. By default, the AFM compares the SA field with the values programmed in the SA registers. The MAC Address registers [1:31] can be configured to contain SA instead of DA for comparison, by setting Bit 30 of the corresponding Register. Group filtering with SA is also supported. The frames that fail the SA Filter are dropped by the GMAC if the SAF bit of Frame Filter register is set. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word (see [Table 3-7](#)).

When SAF bit is set, the result of SA Filter and DA filter is AND'ed to decide whether the frame needs to be forwarded. This means that either of the filter fail result drops the frame and both filters have to pass in-order to forward the frame to the application.

Inverse Filtering Operation

For both Destination and Source address filtering, there is an option to invert the filter-match result at the final output. These are controlled by the DAIF and SAIF bits of the Frame Filter register respectively. The DAIF bit is applicable for both Unicast and Multicast DA frames. The result of the unicast/multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

[Tables 3-5](#) and [3-6](#) summarize the Destination and Source Address filtering based on the type of frames received.

Table 3-5 Destination Address Filtering Table

Frame Type	PR	HPF	HUC	DAIF	HMC	PM	DB	DA Filter Operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all frames.
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match.
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match.

Table 3-5 Destination Address Filtering Table (Continued)

Frame Type	PR	HPF	HUC	DAIF	HMC	PM	DB	DA Filter Operation
	0	0	1	0	X	X	X	Pass on Hash filter match.
	0	0	1	1	X	X	X	Fail on Hash filter match.
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match.
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match.
Multicast	1	X	X	X	X	X	X	Pass all frames.
	X	X	X	X	X	1	X	Pass all frames.
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.
	0	0	X	0	1	0	X	Pass on Hash filter match and drop PAUSE control frames if PCF = 0x.
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.
	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.
	0	0	X	1	1	0	X	Fail on Hash filter match and drop PAUSE control frames if PCF = 0x.
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.

Table 3-6 Source Address Filtering Table

Frame Type	PR	SAIF	SAF	SA Filter Operation
Unicast	1	X	X	Pass all frames.
	0	0	0	Pass status on Perfect/Group filter match but do not drop frames that fail.
	0	1	0	Fail status on Perfect/Group filter match but do not drop frame.
	0	0	1	Pass on Perfect/Group filter match and drop frames that fail.
	0	1	1	Fail on Perfect/Group filter match and drop frames that fail.

3.6.5 MAC Receive Interface Protocol

The MRI interface connects the application to the receive data path of the GMAC core with a simple FIFO-protocol interface. The RBU initiates an Ethernet frame transfer by asserting the mri_sof_o along with the data (mri_data_o). All valid data transfers are indicated by an active high on the mri_val_o signal. The RBU module transfers the last data of the frame by driving the signal mri_eof_o along with the data. The number of byte lanes having valid data in the last transfer (EOF) is indicated by the mri_be_o[3:0] signal. The value

of mri_be_o is the same as for mti_be_i as given in [Table 3-1](#), [Table 3-2](#), and [Table 3-3](#). The 32-bit Receive status (mri_rxstatus_o, described in [Table 3-7](#)) is also valid along with the EOF data transfer.

The GMAC always assumes that the application accepts the data output by it in one clock cycle. Hence it does not have any acknowledgement from the application before performing the next data transfer.

 **Note** Bits 47-32 of mri_rxstatus_o described in [Table 3-7](#) are valid and applicable only when Advanced Timestamping feature is enabled.

Table 3-7 Receive Status at the GMAC Core Interface

Bit	Description
47-46	Reserved
45	<p>PTP Version</p> <p>When set indicates the received PTP message is having the IEEE 1588 version 2 format in the version field. When reset it has the version 1 format. This is valid only if the Message Type is non-zero. This bit is available only when you select the Advance Timestamp feature.</p>
44	<p>PTP Frame Type</p> <p>When set, this bit indicates that the PTP message is sent directly over Ethernet. This bit is available only when you select the Advance Timestamp feature.</p>
43:40	<p>Message Type</p> <p>These bits are encoded to give the type of the message received.</p> <ul style="list-style-type: none"> • 0000: No PTP message received • 0001: SYNC (all clock types) • 0010: Follow_Up (all clock types) • 0011: Delay_Req (all clock types) • 0100: Delay_Resp (all clock types) • 0101: Pdelay_Req (in peer-to-peer transparent clock) • 0110: Pdelay_Resp (in peer-to-peer transparent clock) • 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 1000: Announce • 1001: Management • 1010: Signaling • 1011-1110: Reserved • 1111: PTP packet with Reserved message type <p>These bits are valid only when you select the Advance Timestamp feature.</p> <p>Note: Values 1000, 1001, and 1010 are not backward compatible with release 3.50a.</p>
39	<p>IPv6 Packet Received</p> <p>When set, this bit indicates that the received packet is an IPv6 packet. This bit is available only when you select the Advance Timestamp feature.</p>
38	<p>IPv4 Packet Received</p> <p>When set, this bit indicates that the received packet is an IPv4 packet. This bit is available only when you select the Advance Timestamp feature.</p>

Table 3-7 Receive Status at the GMAC Core Interface

Bit	Description
37	IP Checksum Bypassed When set, this bit indicates that the checksum offload engine is bypassed. This bit is available only when you select the Advance Timestamp feature.
36	IP Payload Error When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the core calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. This bit is available only when you select the Advance Timestamp feature.
35	IP Header Error When set, this bit indicates either that the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or that the IP datagram version is not consistent with the Ethernet Type value. This bit is available only when you select the Advance Timestamp feature.
34:32	IP Payload Type These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE). The COE also sets these bits to 2'b00 if it does not process the IP datagram's payload because of an IP header error or fragmented IP. 3'b000: Unknown or did not process IP payload 3'b001: UDP 3'b010: TCP 3'b011: ICMP 3'b1xx: Reserved This bit is available only when you select the Advance Timestamp feature.
31:28	Rx MAC Address or Payload Checksum Error When the Advanced Time Stamp feature is enabled, these bits, along with bit 27, indicate which of the 0-31 MAC Address Registers matches the DA field. If more than one register matches the DA field, the lower register number is given. When Advanced Timestamp feature and Full Checksum Offload Engine (Type 2) are not present, these bits indicate which Rx MAC Address register (0–15) values matched the frame's DA field. If more than one register matches the DA field, the lower register number is given. For matches with MAC Address registers 16–31, the value is 0. When Advanced Timestamp feature is not selected and Full Checksum Offload Engine (Type 2) is enabled, Bits[31:29] are reserved. When Bit 28 is set, it indicates the TCP, UDP, or ICMP checksum calculated by the core does not match the received encapsulated TCP, UDP, or ICMP segment's Checksum field. Bit 28 is also set when the received number of payload bytes does not tally to the value indicated by the Length field of the encapsulated IPv4/IPv6 datagram in the received Ethernet frame. See Table 3-8 for more detail.
27	Rx MAC Address or IP Header Checksum Error When the Advanced Time Stamp feature is enabled, this bit is used for MAC Address match as described in bits 31-28. Otherwise, when this bit is set, it indicates that the 16-bit IP Header checksum calculated by the core did not match the received Header Checksum bytes. When Full Checksum Offload Engine (Type 2) is enabled, this bit indicates an error in the IPv4 or IPv6 header. This error can be because of: inconsistent values between the Ethernet Type and IP Header Version fields, an IPv4 header checksum mismatch, or an Ethernet frame that does not have the expected number of IP header bytes. See Table 3-8 for more detail.

Table 3-7 Receive Status at the GMAC Core Interface

Bit	Description
26	Control Frame When this bit is set, the control frame is received.
25	SA Filter Fail When this bit is set, the Source Address filter failed. This means that this frame did not clear the SA filter.
24	DAfilter Fail When this bit is set, the Destination Address filter failed. This means that this frame did not clear the DA filter.
23	Length Error When set, this bit indicates that the current frame Length value is inconsistent with the total number of bytes received in the current frame. This is valid when the Frame Type is set to '0' (802.3z Frame) and Frame Length value is less than 1500. Length error bit is invalid when CRC error is present in the status.
22	VLAN Frame When this bit is set, the current reception is tagged with a VLAN ID. The 13th and 14th bytes at the frame are compared with 16'h8100, and the following 2-byte data is compared with the VLAN Tag register. This bit is set in response to a match.
21	CRC Error When set, this bit indicates that a cyclic redundancy check (CRC) error occurred on the received frame
20	Dribbling Bit When set, this bit indicates that the frame contained a non-integer multiple of eight bits (valid only in MII mode). This bit is not valid if a collision is seen or if runt frame bits are set. If this bit is set and the CRC error is reset, then the packet is valid.
19	GMII Error When set, this bit indicates that the gmii_rxer_i was asserted during the reception of this frame. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error (rxd = 1f) during extension.
18	Frame Type When set, this bit indicates that the frame is an Ethernet-type frame (frame length field is greater than or equal to 0x600). When clear, it indicates that the frame is an IEEE 802.3z frame. This bit is not valid for runt frames of less than 14 bytes. See Table 3-8 for more detail.
17	Late Collision Seen When this bit is set, it indicates that the frame may be damaged by a late collision (active gmii_col_i signal seen after 64 bytes after SFD in Half-Duplex mode) during the reception of the frame.
16	Giant Frame When set, this bit indicates that the frame length exceeds the maximum Ethernet specified size of 1,518/1,522 bytes (9,018/9,022 bytes if jumbo frame enable is set). Giant frame is only a frame length indication, and it does not cause any frame truncation.
15	Runt Frame When this bit is set, it indicates that the GMAC has received frames of less than 64 bytes or a collision was observed before the receipt of 64th byte.

Table 3-7 Receive Status at the GMAC Core Interface

Bit	Description
14	Watchdog Timeout When set, this bit indicates that the RPE module has received a frame with byte count greater than 2,048 (10,240 if Jumbo frame is enabled) bytes (DA + SA + LT + DATA + PAD + FCS). This bit is not set if the watchdog timer is disabled in the Configuration register. However, even if the watchdog timer is disabled, this bit is set when the frame is greater than 16 KB in size.
13:0	Frame Length Indicates the length, in bytes, of the received frame (including pad if applicable), and/or the FCS field when the Auto Pad/CRC Strip bit is de-asserted. Otherwise, it indicates the length without pad and/or FCS field. It also includes the 2-byte IP Checksum appended after the FCS when the IP Checksum module (Type 1) is present and enabled by setting the IPC (Bit 10 of GMAC Register 0), and if the received frame is not a MAC control frame.

As mentioned in [Table 3-7](#), when Full Checksum Offload Engine (Type 2) is enabled, the meaning of certain register bits change. Their significance is mapped in [Table 3-8](#).

Table 3-8 Frame Status with Full Checksum Offload Engine Enabled and Advanced Timestamping not Enabled

Bit 18: Ethernet Frame	Bit 27: Header Checksum Error	Bit 28: Payload Checksum Error	Frame Status
0	0	0	The frame is an IEEE 802.3 frame (Length field value is less than 0x0600).
1	0	0	IPv4/IPv6 Type frame in which no checksum error is detected.
1	0	1	IPv4/IPv6 Type frame in which a payload checksum error (as described for PCE) is detected.
1	1	0	IPv4/IPv6 Type frame in which IP header checksum error (as described for IPC HCE) is detected.
1	1	1	IPv4/IPv6 Type frame in which both PCE and IPC HCE is detected.
0	0	1	IPv4/IPv6 Type frame in which there is no IP HCE and the payload check is bypassed because of unsupported payload.
0	1	1	Type frame which is neither IPv4 or IPv6 (COE bypasses the checksum check completely)
0	1	0	Reserved



Note The first five conditions are backward-compatible to versions 3.30a and previous. The last two conditions (001, 011), which are reserved, are not backward-compatible, because the FT bit is reset during bypasses, even for valid Type frames.

3.6.6 MAC Receive Timing

Timing specifications for the MAC Core (GMAC-CORE) interface signals in the receive path are illustrated in detail, in the following sections.

3.6.6.1 MAC Receive Interface (MRI)

The timing diagrams in this section illustrate the timing specifications for the MAC receive interface signals and the GMII receive interface. The sequence of events is as follows:

1. The MAC receives frames on the (G)MII interface.
2. It stores the frame until it makes the decision to filter the frame, if configured so. When the frame is passed by the filter, the MAC puts the data on mri_data_o and indicates the validity by asserting mri_val_o. It also asserts mri_sof_o, if the data corresponds to the start of a frame.
3. The application must accept the data as soon as the MAC presents it because the MAC may output the next data (if available) in the next clock cycle itself.
4. The MAC indicates transfer of the end-of-frame data with the assertion of mri_eof_o. Signal mri_be_o may also have a non all-one value during the EOF, showing how many lanes on the data bus have valid data.
5. The MAC updates the receive status bits on mri_rxstatus_o during the transfer of the EOF.

Figure 3-55 MAC Receive Interface Timing (1 of 4)

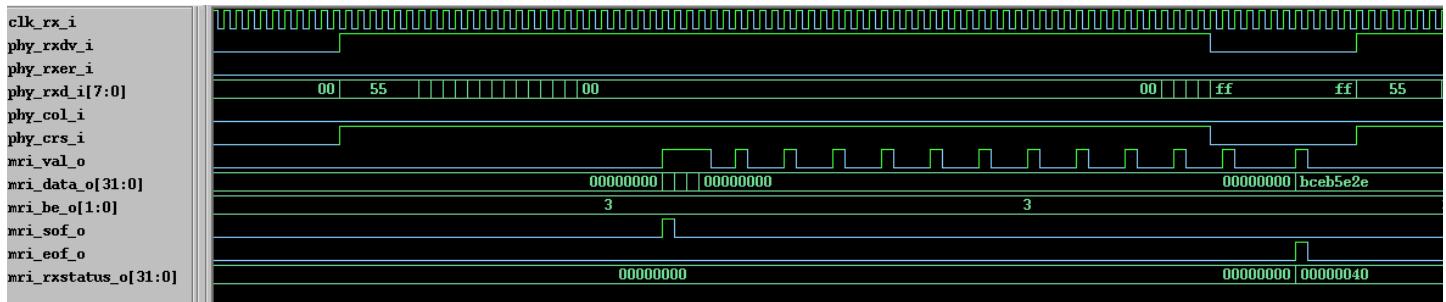
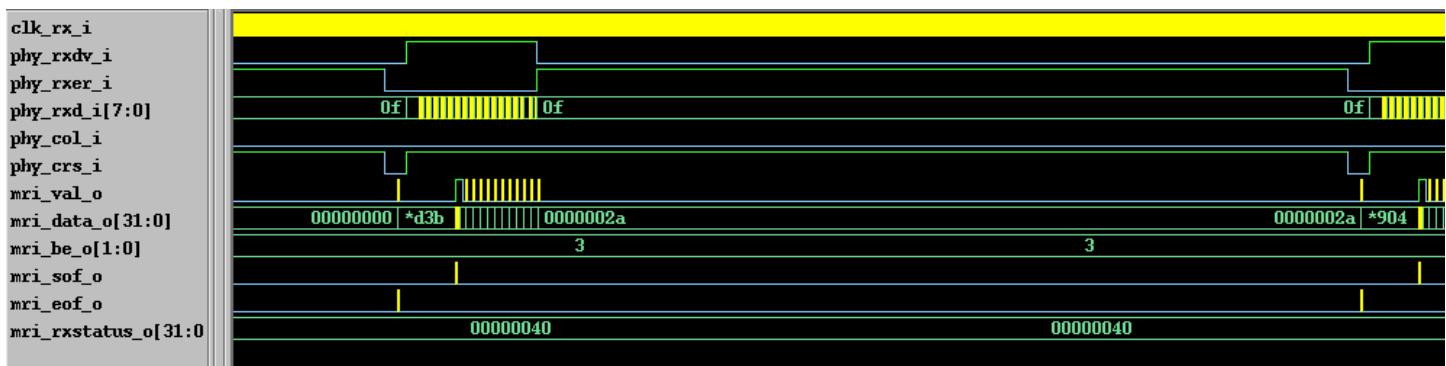
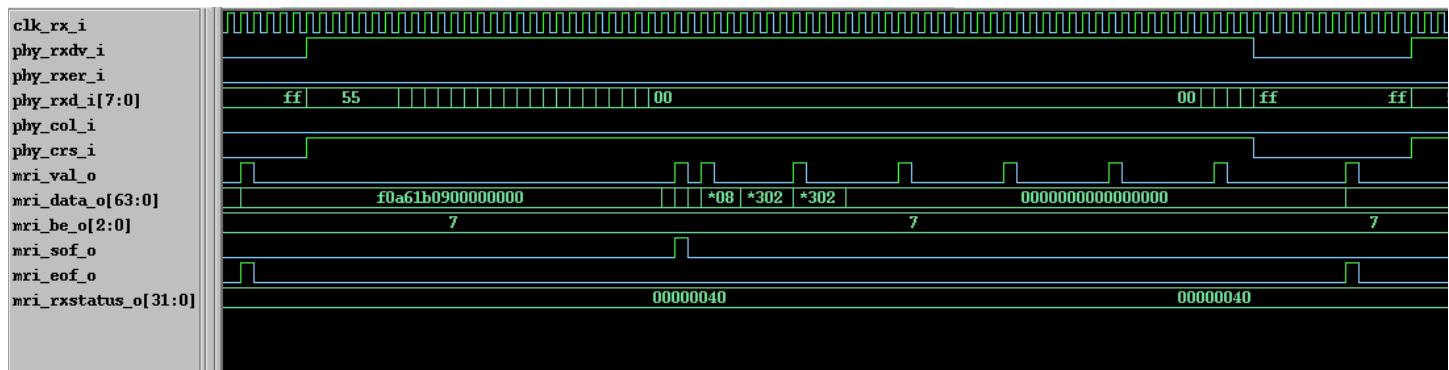
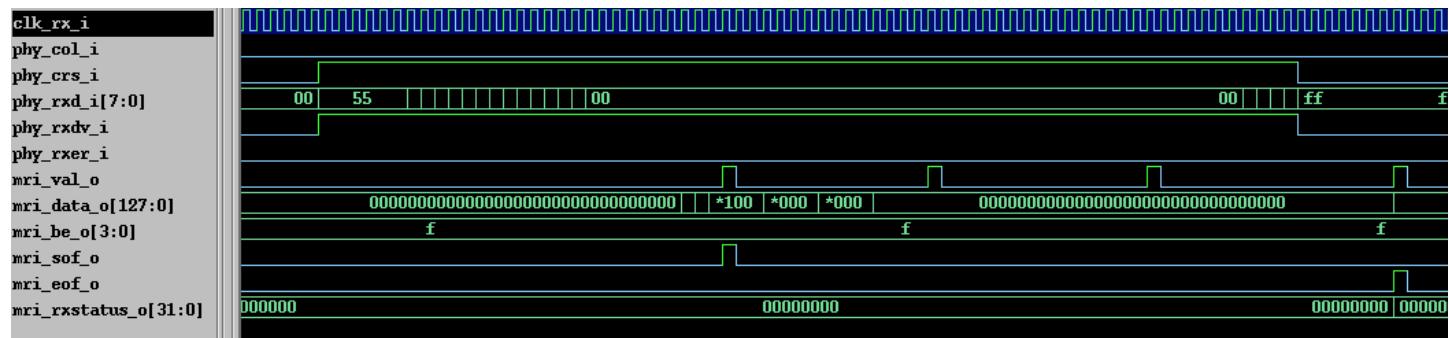


Figure 3-56 illustrates the reception and transfer of a frame in half-duplex mode/gigabit mode. Note that the EOF is not transferred until the completion of the frame-extension so that valid receive status can be output together.

Figure 3-56 MAC Receive Interface Timing (2 of 4)

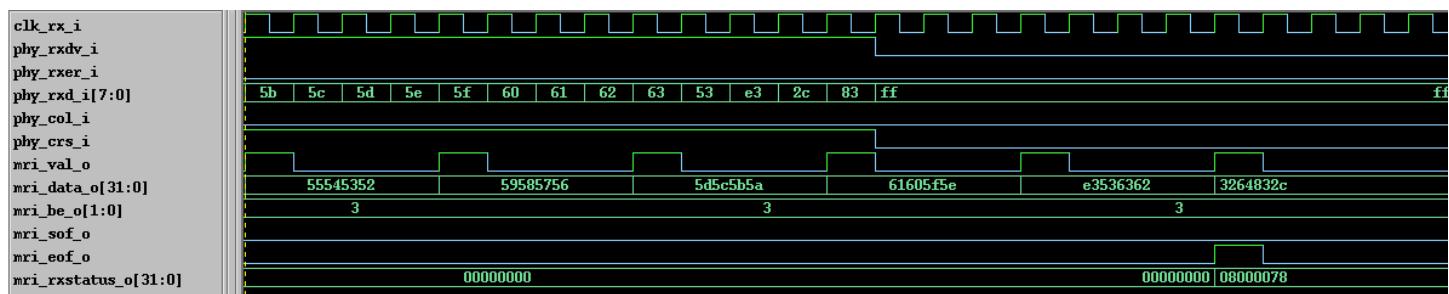


Figures 3-57 and 3-58 show the timing of the MRI, with 64-bit and 128-bit data bus configurations.

Figure 3-57 MAC Receive Interface Timing (3 of 4)**Figure 3-58 MAC Receive Interface Timing (4 of 4)**

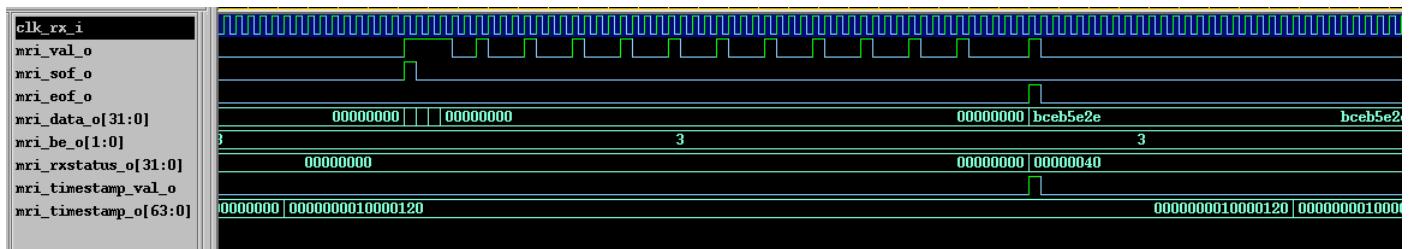
3.6.6.2 Reception of Frame With IP Payload Checksum (Type 1)

Figure 3-59 shows how the MAC receiver adds the 16-bit IP payload checksum to the frame transferred on the MRI. The normal end of a frame byte is 0x83 while the 16-bit IP payload checksum 0x6432 is attached to the frame data during the EOF. If the frame is such that the IP payload checksum spills over to the next transaction, then the EOF is indicated at that point.

Figure 3-59 Reception of Frame Along With IP Payload Checksum Timing

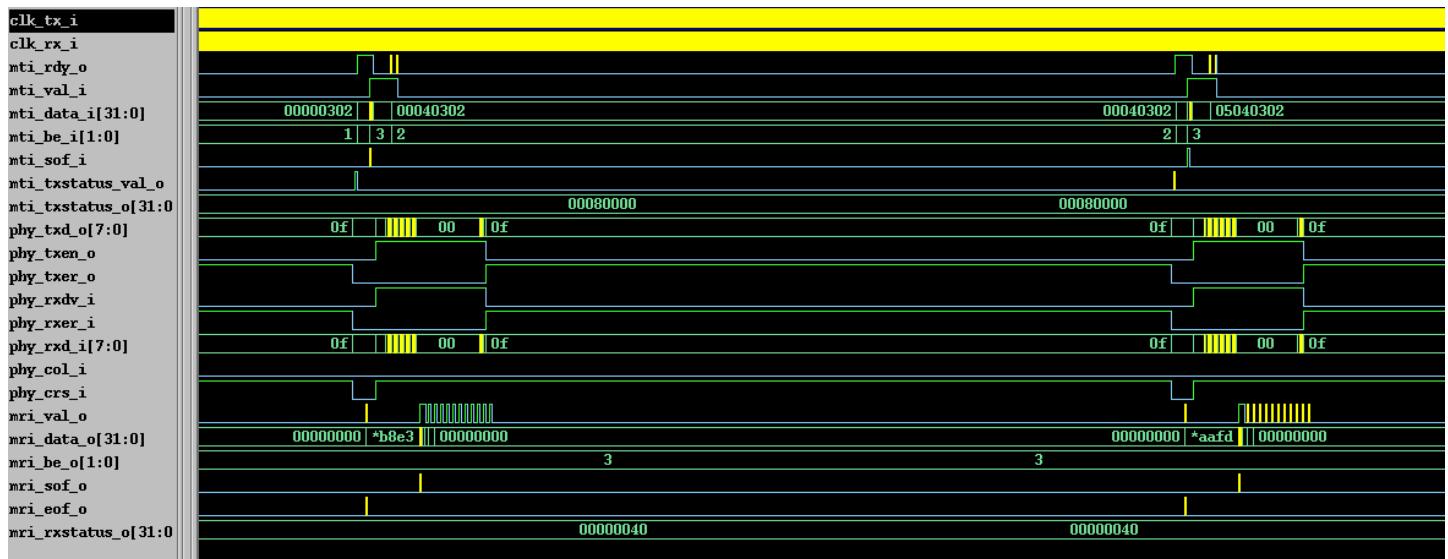
3.6.6.3 Frame Reception With Time-Stamping Enabled

Figure 3-60 shows the MRI timing when IEEE 1588 time stamping is enabled. When the MRI outputs the EOF data, as marked by `mri_eof_o` high, the time stamp value is also given on `mri_timestamp_o`. The time stamp is validated with the assertion of `mri_timestamp_val_o`. When Advanced Timestamp feature is selected, the width of the status is 48-bits.

Figure 3-60 MRI Timing with IEEE 1588 Time Stamping Enabled

3.6.6.4 Frame Transmission and Reception

Figure 3-61 shows both the MTI and MRI interface signal timing along with the GMII signal timing for a frame transmission that is looped back to the GMII receiver in the testbench environment.

Figure 3-61 Frame Transmission and Reception

3.6.7 MAC Control Interface Protocol

This interface connects the Application with the Control and Status register (CSR) module of the Media Access Controller to provide the control path for programming the MAC registers. The Application initiates a Register Write/Read by asserting the command valid signal mci_val_i (with the valid data mci_wdata_i[31:0] and valid byte enables mci_be_i[3:0]), along with the register address on mci_addr_i[12:0]. The control signal mci_rdwn_i indicates a Write (1'b0) or Read (1'b1) operation. The Application assumes a successful data transfer when it receives an acknowledgement (mci_ack_o) asserted for 1 clock cycle. The CSR module drives the Read data on mci_rdata[31:0]. The Application can de-assert the command valid once the acknowledgement is received.

If the Application continues to assert the command valid signal (on receiving an acknowledgement) then the GMAC considers it as the next access and responds to it by asserting the mci_ack_o after 1 clock cycle. Accesses to reserved registers are completed by the GMAC with dummy read or write operations. In other words, writes to reserved addresses have no effect and read operations are completed with an all-zero data output on the mci_rdata_o bus.

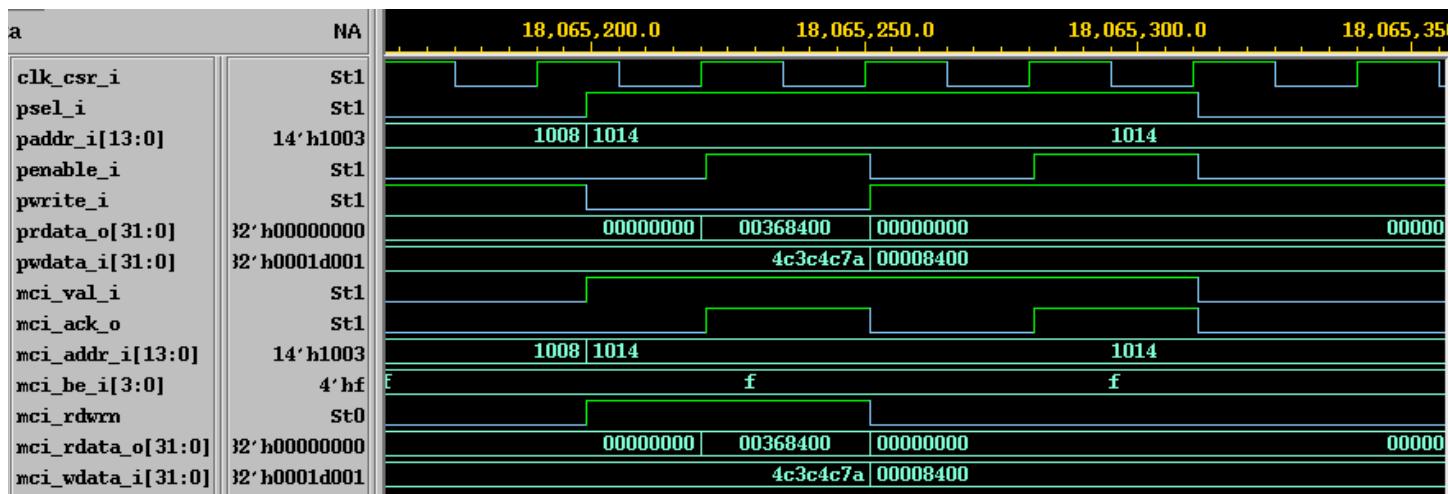
All read and write accesses to the CSR completes in 2 clock cycles, that is, mac_ack_o is asserted after a delay of 1 clock cycle after mci_val_i goes high. This enables the user to map the MCI signals to an APB port signals with minimal glue logic. For non-APB/AHB/AXI Slave interface configuration for CSR, mci_rdata_o is registered and hence there is an additional delay of 1 clock for the assertion of mci_ack_o. Thus, each read access completes in 3 clock cycles for such configurations.

3.6.7.1 APB Port Timing

Figure 3-62 shows a register read access followed by a register write access on the APB port. The flow is as follows:

1. The APB port signals behave as per the AMBA 2.0 standard specifications.
2. The mci_* signals shown in the figure are the internal MCI interface signals. Note that signals psel_i, paddr_i, prdata_o, and pwdata_i are the same as mci_val_i, mci_addr_i, mci_rdata_o, and mci_wdata_i, respectively. Internally, the APB port signals are directly connected to the corresponding MCI signals. Signal mci_rdwrn_i is the invert of pwrite_i. In APB port configuration, the internal mci_ack_o signal is an input tied to penable_i.

Figure 3-62 APB Port Timing



3.6.7.2 MCI Port Timing

Figure 3-63 shows read and write accesses on the MCI port. The flow is as follows:

1. The first access to address 0x0004 is a single read cycle which completes in 3 clock cycles. Signal mci_val_i and the corresponding address and control signals should be kept asserted for the whole duration of the transaction.
2. The second and third access are single write operation cycles to address 0x0004 and 0x0000, respectively. Signal mci_ack_o is asserted 1 clock after mci_val_i is asserted. Thus the write operations are completed in 2 clock cycles.
3. The following accesses are burst write and burst read operations starting from address 0x0040. Note that each individual write/read transaction in the burst is identical to the “single” write/read transactions, with respect to timing.

Figure 3-63 MCI Port Timing

4

Optional Modules

This chapter provides information about the optional modules of GMAC-UNIV. It contains the following sections:

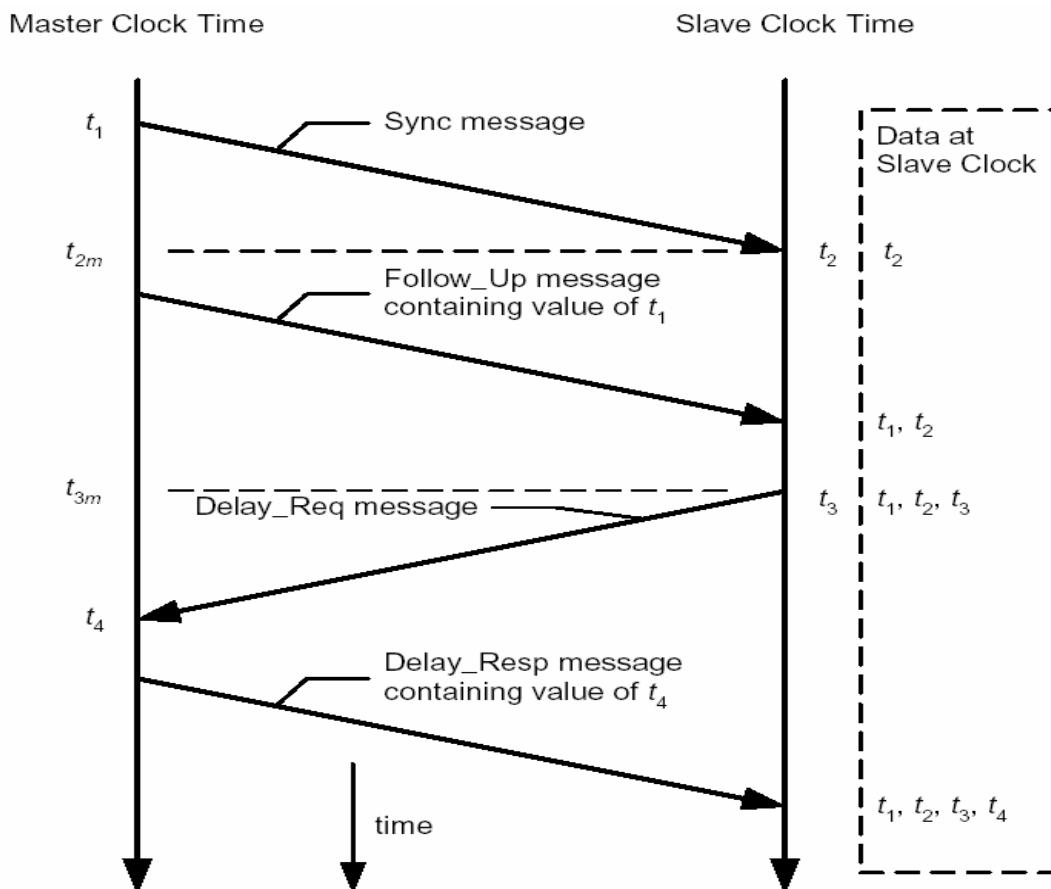
- ❖ “[IEEE 1588-2002 Timestamps](#)” on page [115](#)
- ❖ “[IEEE 1588-2008 Advanced Timestamps](#)” on page [121](#)
- ❖ “[Audio/Video Bridging \(AVB\)](#)” on page [130](#)
- ❖ “[Energy Efficient Ethernet](#)” on page [138](#)
- ❖ “[MAC Management Counters](#)” on page [144](#)
- ❖ “[Power Management Block](#)” on page [160](#)
- ❖ “[Station Management Agent](#)” on page [164](#)
- ❖ “[Physical Coding Sublayer](#)” on page [167](#)
- ❖ “[Reduced Media Independent Interface](#)” on page [169](#)
- ❖ “[Reverse Media Independent Interface](#)” on page [174](#)
- ❖ “[Serial Media Independent Interface](#)” on page [185](#)
- ❖ “[Reduced Gigabit Media Independent Interface](#)” on page [189](#)
- ❖ “[Reduced Ten-Bit Interface](#)” on page [194](#)
- ❖ “[Serial Gigabit Media Independent Interface](#)” on page [195](#)
- ❖ “[Multiple PHY Interfaces](#)” on page [201](#)
- ❖ “[Interrupts From the GMAC Core](#)” on page [202](#)

4.1 IEEE 1588-2002 Timestamps

The IEEE 1588-2002 standard defines a protocol, Precision Time Protocol (PTP), that enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The PTP is transported over UDP/IP. The system or network is classified into Master and Slave nodes for distributing the timing/clock information. Figure 4-1 shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

Figure 4-1 Networked Time Synchronization



As shown in Figure 4-1, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the master's reference time information. The time at which this message leaves the master's system is t_1 . This time must be captured, for Ethernet ports, at GMII/MII.
2. The slave receives the Sync message and also captures the exact time, t_2 , using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t_1 information for later use.
4. The slave sends a Delay_Req message to the master, noting the exact time, t_3 , at which this frame leaves the GMII/MII.
5. The master receives the message, capturing the exact time, t_4 , at which it enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1 , t_2 , t_3 , and t_4 to synchronize its local timing reference to the master's timing reference.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at

the GMII/MII. This timing information must be captured and returned to the software for the proper implementation of PTP with high accuracy.

4.1.1 Reference Timing Source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588 specification. The GMAC-UNIV provides the following two options for using the reference timing source in a node:

- ❖ External Timestamp Input Option

This option takes an external 64-bit timing reference and its clock as input. The clock input is used to synchronize the timing reference to the MAC clock domain. The 64-bit timing reference is split into the following two 32-bit signals:

- ◆ Upper 32 bits providing the time in seconds
- ◆ Lower 32 bits providing the time in nanoseconds

This timing reference is used to timestamp the frames.

- ❖ Internal Reference Time Option

This option takes only the reference clock input and uses it to generate the Reference time (also called the System Time) internally and capture timestamps. The generation, update, and modification of the System Time are described in “[System Time Register Module](#)” on page 117.

You can choose either of these options as described in “[Parameters](#)” on page 355.

4.1.2 System Time Register Module

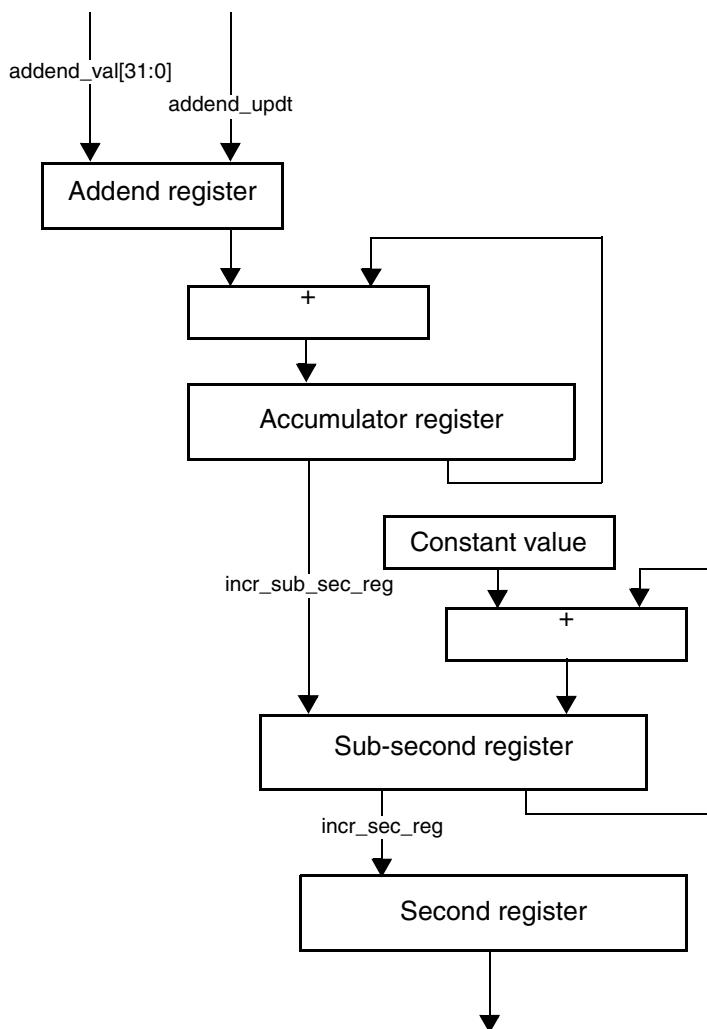
The System Time Generator module is optional and is not available if external time updating is enabled.

The 64-bit time is maintained in this module and updated using the input reference clock (`clk_ptp_ref_i`). This time is the source for taking snapshots (timestamps) of Ethernet frames being transmitted or received at the GMII.

The System Time counter can be initialized or corrected using the *coarse correction method*. In this method, the initial value or the offset value is written to the Timestamp Update register (“[IEEE 1588 Timestamp Registers](#)” on page 345). For initialization, the System Time counter is written with the value in the Timestamp Update registers, while for system time correction, the offset value is added to or subtracted from the system time.

In the *fine correction method*, a slave clock’s (`clk_ptp_ref_i`) frequency drift with respect to the master clock (as defined in IEEE 1588) is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register, as shown in [Figure 4-2](#). The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency multiplier or divider.

This algorithm is depicted in [Figure 4-2](#):

Figure 4-2 System Time Update Using Fine Method

The System Time Update logic requires a 50-MHz clock frequency to achieve 20-ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. Hence, if the reference clock (clk_ptp_ref_i) is, for example, 66 MHz, this ratio is calculated as $66 \text{ MHz} / 50 \text{ MHz} = 1.32$. Hence, the default addend value to be set in the register is $2^{32} / 1.32$, 0xC1F07C1F.

If the reference clock drifts lower, to 65 MHz for example, the ratio is $65 / 50$, or 1.3 and the value to set in the addend register is $2^{32} / 1.30$, or 0xC4EC4EC4. If the clock drifts higher, to 67 MHz for example, the addend register must be set to 0xBF0B7672. When the clock drift is nil, the default addend value of 0xC1F07C1F ($2^{32} / 1.32$) must be programmed.

In Figure 4-2, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20-ns steps). The optional System Time module is unavailable when External Time Update is enabled. Two different methods are used to update the System Time register, depending on which configuration you choose (See “[Parameters](#)” on page [355](#)).

The software must calculate the drift in frequency based on the Sync messages and update the Addend register accordingly.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = 2^{32} / \text{FreqDivisionRatio}$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm described below must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

- ❖ At time MasterSyncTime_n the master sends the slave clock a Sync message. The slave receives this message when its local clock is SlaveClockTime_n and computes MasterClockTime_n as:

$$\text{MasterClockTime}_n = \text{MasterSyncTime}_n + \text{MasterToSlaveDelay}_n$$

- ❖ The master clock count for current Sync cycle, $\text{MasterClockCount}_n$ is given by:

$$\text{MasterClockCount}_n = \text{MasterClockTime}_n - \text{MasterClockTime}_{n-1} \text{ (assuming that MasterToSlaveDelay is the same for Sync cycles } n \text{ and } n-1\text{)}$$

- ❖ The slave clock count for current Sync cycle, SlaveClockCount_n is given by:

$$\text{SlaveClockCount}_n = \text{SlaveClockTime}_n - \text{SlaveClockTime}_{n-1}$$

- ❖ The difference between master and slave clock counts for current Sync cycle, ClockDiffCount_n is given by:

$$\text{ClockDiffCount}_n = \text{MasterClockCount}_n - \text{SlaveClockCount}_n$$

- ❖ The frequency-scaling factor for slave clock, FreqScaleFactor_n is given by:

$$\text{FreqScaleFactor}_n = (\text{MasterClockCount}_n + \text{ClockDiffCount}_n) / \text{SlaveClockCount}_n$$

- ❖ The frequency compensation value for Addend register, $\text{FreqCompensationValue}_n$ is given by:

$$\text{FreqCompensationValue}_n = \text{FreqScaleFactor}_n * \text{FreqCompensationValue}_{n-1}$$

In theory, this algorithm achieves lock in one Sync cycle; however, it may take several cycles, because of changing network propagation delays and operating conditions.

This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm corrects it at the cost of more Sync cycles.

4.1.3 Transmit Path Functions

The MAC captures a timestamp when the Start Frame Delimiter (SFD) of a frame is sent on GMII/MII. The frames for which you want to capture timestamps are controllable on a per-frame basis. In other words, each transmit frame can be marked to indicate whether a timestamp should be captured for that frame.

The MAC does not process the transmitted frames to identify the PTP frames. You need to specify the frames for which you want to capture timestamps. In GMAC-CORE and GMAC-MTL configurations, you can do this through input signals (see “[IEEE 1588 Timestamp Signals \(Optional\)](#)” on page [276](#)). To return the captured timestamps to the application, the MAC uses a separate 64-bit bus. The MAC gives the timestamp, along with the Transmit status of the frame, on this bus.

In GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations, you can use the control bits in the transmit descriptor (see “[Descriptor Format With IEEE 1588 Timestamps Enabled](#)” on page [410](#)). The MAC returns the timestamp to the software inside the corresponding transmit descriptor, thus connecting the timestamp automatically to the specific PTP frame. The 64-bit timestamp information is written to the TDES2 and TDES3 fields. The TDES2 field holds the 32 least significant bits of the timestamp, except as described in “[Transmit Timestamp Field](#)” on page [413](#).



Note When the alternate (enhanced) descriptor is selected, the MAC writes the 64-bit timestamp in TDES6 and TDES7, respectively.

4.1.4 Receive Path Functions

The MAC captures the timestamp of all frames received on the GMII/MII. The MAC does not process the received frames to identify the PTP frames in the default mode, that is, when the Advanced Timestamp feature is not selected.

In GMAC-CORE configuration, the MAC gives the timestamp and the corresponding status on the MAC Receive Interface (MRI) along with the EOF data.

In GMAC-MTL configuration, the MTL provides the timestamp on the data bus ([ari_data_o\[N - 1:0\]](#)) after the EOF data has been transferred. The MTL sends a separate signal ([ari_timestamp_val_o](#)) to validate the timestamp and to indicate the availability of the timestamp. Once the timestamp is transferred, the Status Valid signal ([ari_rxstatus_val_o](#)) is asserted as soon as status data is present on the data bus.

In GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations, the DMA returns the timestamp to the software in the corresponding receive descriptor. The 64-bit timestamp information is written back to the RDES2 and RDES3 fields. The RDES2 holds the 32 least significant bits of the timestamp, except as mentioned in “[Receive Timestamp](#)” on page [411](#). The timestamp is written only to that receive descriptor for which the Last Descriptor status field has been set to 1 (the EOF marker). When the timestamp is not available (for example, because of an RxFIFO overflow), an all-ones pattern is written to the descriptors (RDES2 and RDES3), indicating that timestamp is not correct. If the software uses a control register bit to disable time stamping, the DMA does not alter RDES2 or RDES3.



Note When the alternate (enhanced) descriptor is selected, the 64-bit time-stamp is written in RDES6 and RDES7, respectively. RDES0[7] indicates whether the timestamp is updated in RDES6 and RDES7 or not.

4.1.5 Timestamp Error Margin

According to the IEEE 1588 specifications, a timestamp must be captured at the SFD of the transmitted and received frames at the GMII/MII interface. Because the reference timing source (the PTP clock, [clk_ptp_ref_i](#)) is different from the GMII/MII clocks, a small error margin is introduced, because of the transfer of information across asynchronous clock domains.

In the transmit path, the captured and reported timestamp has a maximum error margin of 2 PTP clocks. It means that the captured timestamp has the reference timing source value that is given within 2 clocks after the SFD has been transmitted on the GMII.

Similarly, in the receive path, the error margin is 3 GMII/MII clocks, plus up to 2 PTP clocks. You can ignore the error margin because of the 3 GMII/MII clocks by assuming that this constant delay is present in the system (or link) before the SFD data reaches the GMAC’s GMII/MII interface.

4.1.6 Frequency Range of Reference Timing Clock

The timestamp information is transferred across asynchronous clock domains, that is, from MAC clock domain to application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is 4 clock cycles of GMII/MII and 3 clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second frame.

The maximum PTP clock frequency is limited by the maximum resolution of the reference time (1 ns resulting in 1 GHz) and the timing constraints achievable for logic operating on the PTP clock. In addition, the resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes. Because the GMII/MII clock frequency is fixed by IEEE specification, the minimum PTP clock frequency required for proper operation depends upon the operating mode and operating speed of the MAC as shown in [Table 4-1](#).

Table 4-1 Minimum PTP Clock Frequency Example

Mode	Minimum Gap Between Two SFDs	Minimum PTP Frequency
100-Mbps full-duplex operation	168 MII clocks (128 clocks for a 64-byte frame + 24 clocks of min IFG + 16 clocks of preamble)	$(3 * \text{PTP}) + (4 * \text{MII}) \leq 168 * \text{MII}$ that is, ~0.5 MHz $((168 - 4) * 40 \text{ ns} \div 3 = 2,180 \text{ ns period})$
1000-Mbps half-duplex operation	24 GMII clocks (4 for a JAM pattern sent just after SFD because of collision + 12 IFG + 8 preamble)	$3 * \text{PTP} + 4 * \text{GMII} \leq 24 * \text{GMII}$ that is, 18.75 MHz

4.2 IEEE 1588-2008 Advanced Timestamps

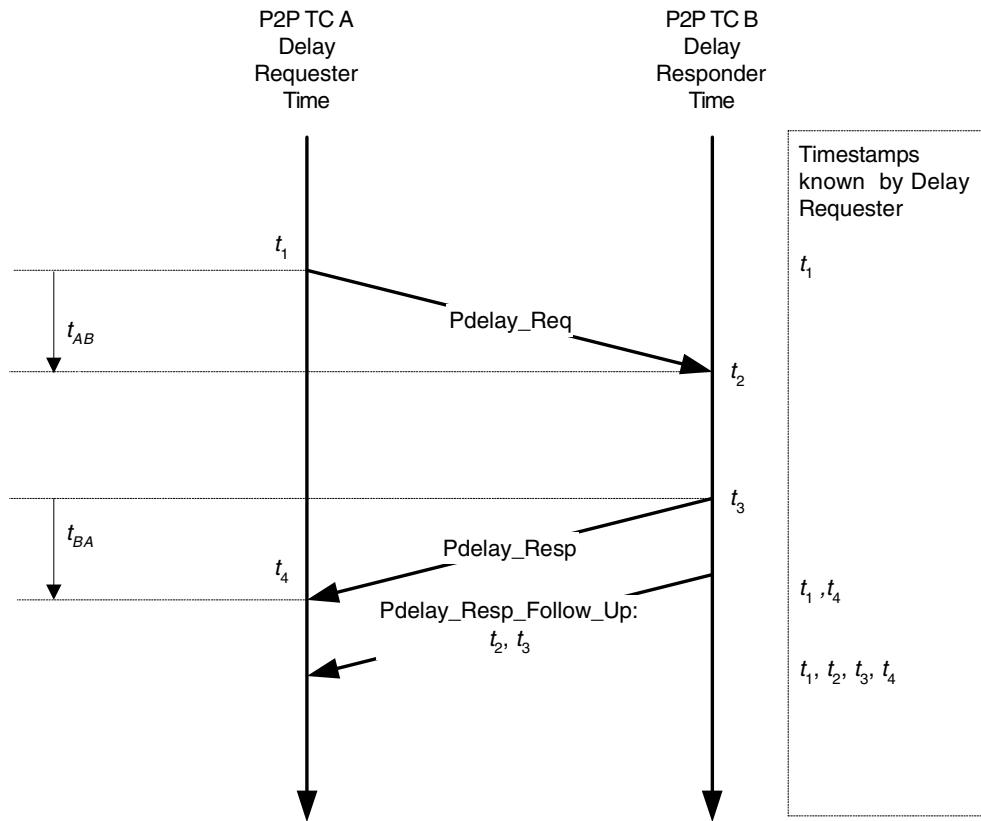
In addition to the basic timestamp features mentioned in [IEEE 1588-2002 Timestamps](#), the GMAC-UNIV supports the following advanced timestamp features defined in the IEEE 1588-2008 standard.

- ❖ Supports the IEEE 1588-2008 (version 2) timestamp format.
- ❖ Provides an option to take snapshot of all frames or only PTP type frames.
- ❖ Provides an option to take snapshot of only event messages.
- ❖ Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end, and peer-to-peer.
- ❖ Provides an option to select the node to be a Master or Slave for ordinary and boundary clock.
- ❖ Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- ❖ Provides an option to measure sub-second time in digital or binary format.

4.2.1 Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 version supports Peer-to-Peer PTP (Pdelay) message in addition to SYNC, Delay Request, Follow-up, and Delay Response messages. [Figure 4-3](#) shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 4-3 Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction



As shown in [Figure 4-3](#), the propagation delay is calculated in the following way:

1. Port-1 issues a **Pdelay_Req** message and generates a timestamp, t_1 , for the **Pdelay_Req** message.
2. Port-2 receives the **Pdelay_Req** message and generates a timestamp, t_2 , for this message.
3. Port-2 returns a **Pdelay_Resp** message and generates a timestamp, t_3 , for this message.

To minimize errors because of any frequency offset between the two ports, Port-2 returns the **Pdelay_Resp** message as quickly as possible after the receipt of the **Pdelay_Req** message. The Port-2 returns any one of the following:

- ◆ The difference between the timestamps t_2 and t_3 in the **Pdelay_Resp** message.
- ◆ The difference between the timestamps t_2 and t_3 in the **Pdelay_Resp_Follow_Up** message.
- ◆ The timestamps t_2 and t_3 in the **Pdelay_Resp** and **Pdelay_Resp_Follow_Up** messages respectively.

4. Port-1 generates a timestamp, t_4 , on receiving the **Pdelay_Resp** message.
5. Port-1 uses all four timestamps to compute the mean link delay.

4.2.2 Clock Types

The GMAC-UNIV supports the following clock types defined in the IEEE 1588-2008 standard:

- ❖ [Ordinary Clock](#)

- ❖ Boundary Clock
- ❖ End-to-End Transparent Clock
- ❖ Peer-to-Peer Transparent Clock

4.2.2.1 Ordinary Clock

The ordinary clock in a domain supports a single copy of the protocol. The ordinary clock has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

The ordinary clock can be a grandmaster or a slave clock. The ordinary clock supports the following features:

- ❖ Sends and receives PTP messages. The timestamp snapshot can be controlled as described in [“Register 448 \(Timestamp Control Register\)”](#) on page [345](#).
- ❖ Maintains the data sets such as timestamp values.

[Table 4-2](#) shows the messages for which you can take the timestamp snapshot on the receive side for Master and Slave nodes.

Table 4-2 Ordinary Clock: PTP Messages for Snapshot

Master	Slave
Delay_Req	SYNC

For an ordinary clock, you can take the snapshot of either one of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the control bit (TSVER2ENA) and selecting the snapshot mode in [Register 448 \(Timestamp Control Register\)](#).

4.2.2.2 Boundary Clock

The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy, and signaling terminate in the protocol engine of the boundary clock and are not forwarded. The PTP message type status given by the core (see [“Receive Path Functions”](#) on page [129](#)) helps you to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

- ❖ The clock data sets are common to all ports of the boundary clock
- ❖ The local clock is common to all ports of the boundary clock.

Therefore, the features of the ordinary clock are also applicable to the boundary clock.

4.2.2.3 End-to-End Transparent Clock

The end-to-end transparent clock supports the end-to-end delay measurement mechanism between slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the Ingress port to the Egress port.

The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a

Delay_Req packet inside the end-to-end transparent clock is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both Ingress and Egress ports only for the messages mentioned in [Table 4-3](#). You can take the snapshot by setting the snapshot select bits (SNAPTYPESEL) to 10 in [Register 448 \(Timestamp Control Register\)](#).

Table 4-3 End to End Transparent Clock: PTP Messages for Snapshot

PTP Messages
SYNC
Delay_Req

4.2.2.4 Peer-to-Peer Transparent Clock

The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other respects, it is identical to the end-to-end transparent clock.

In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer. The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdelay_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in [Table 4-4](#).

Table 4-4 Peer-to-Peer Transparent Clock: PTP Messages for Snapshot

PTP Messages
SYNC
Pdelay_Req
Pdelay_Resp

You can take the snapshot by setting the snapshot select bits (SNAPTYPESEL) to 11 in [Register 448 \(Timestamp Control Register\)](#).

4.2.3 PTP Processing and Control

[Table 4-5](#) shows the common message header for the PTP messages. This format is taken from IEEE standard 1588-2008 (Revision of IEEE Std. 1588-2002).

Table 4-5 Message Format Defined in IEEE 1588-2008

BITS								OCTETS	OFFSET					
7	6	5	4	3	2	1	0							
transportSpecific			messageType			1		0						
Reserved			versionPTP			1		1						
messageLength								2						
domainNumber								1						

Table 4-5 Message Format Defined in IEEE 1588-2008

BITS								OCTETS	OFFSET
7	6	5	4	3	2	1	0		
								1	5
								2	6
								8	8
								4	16
								10	20
								2	30
								1	32
								1	33

(*) – controlField is used in version 1. In version 2, messageType field is used for detecting different message types.

There are some fields in the Ethernet payload that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP frames:

- ❖ [PTP Frames Over IPv4](#)
- ❖ [PTP Frames Over IPv6](#)
- ❖ [PTP Frames Over Ethernet](#)

4.2.3.1 PTP Frames Over IPv4

[Table 4-6](#) provides information about the fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of IEEE 1588-2008 standard and the message format defined in [Table 4-5](#).

Table 4-6 IPv4-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame Type	12, 13	0x0800	IPv4 datagram
IP version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed. 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast Address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP-Primary multicast address: 224.0.1.129 PTP-Pdelay multicast address: 224.0.0.107

Table 4-6 IPv4-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
UDP Destination Port	36, 37	0x013F, 0x0140	0x013F – PTP event message (*) 0x0140 – PTP general messages
PTP Control Field (IEEE version 1)	74	0x00/0x01/0x02/0x03/0x04	0x00 – SYNC, 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management
PTP Message Type Field (IEEE version 2)	42 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xB /0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP Version	43 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

(*) PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only).

4.2.3.2 PTP Frames Over IPv6

Table 4-7 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of IEEE 1588-2008 standard and the message format defined in Table 4-5.

Table 4-7 IPv6-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame Type	12, 13	0x86DD	IP datagram
IP Version	14(bits [7:4])	0x6	IP version is IPv6
Layer 4 Protocol	20 (*)	0x11	UDP
PTP Multicast Address	38 – 53	FF0x:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:6B (Hex)	PTP – Primary multicast address: FF0x:0:0:0:0:0:181 (Hex) PTP – Pdelay multicast address: FF02:0:0:0:0:0:6B (Hex)

Table 4-7 IPv6-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
UDP Destination Port	56, 57 (*)	0x013F, 0x140	0x013F – PTP event message 0x0140 – PTP general messages
PTP Control Field (IEEE 1588 version 1)	93 (*)	0x00/0x01/0x02/0x03/0x04	0x00 – SYNC 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management (version1)
PTP Message Type Field (IEEE version 2)	74 (*) (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xB /0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP Version	75 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

(*) The Extension Header is not defined for PTP packets.

4.2.3.3 PTP Frames Over Ethernet

[Table 4-8](#) provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on *Annex D* of the IEEE 1588-2008 standard and the message format defined in [Table 4-5](#).

Table 4-8 Ethernet PTP Frame Fields Required for Control And Status

Field Matched	Octet Position	Matched Value	Description
MAC Destination Multicast Address ^a	0-5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses ^b : <ul style="list-style-type: none"> • 01-1B-19-00-00-00 • 01-80-C2-00-00-0E^c
MAC Frame Type	12, 13	0x88F7	PTP Ethernet frame
PTP Control Field (IEEE version 1)	45	0x00/0x01/0x02/0x03/0x04	0x00 – SYNC 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management

Table 4-8 Ethernet PTP Frame Fields Required for Control And Status

Field Matched	Octet Position	Matched Value	Description
PTP Message Type Field (IEEE version 2)	14 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xB /0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD – Management
PTP Version	15 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

- a. The address match of destination addresses (DA) programmed in MAC address 1 to 31 is used if the control bit 18 (TSENMACADDR: Enable MAC address for PTP frame filtering) of the Timestamp Control register is set.
- b. IEEE standard 1588-2008, *Annex F*
- c. The GMAC-UNIV does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

4.2.4 Reference Timing Source

The GMAC-UNIV supports the following reference timing source features defined in the IEEE 1588-2008 standard:

- ❖ 48-Bit Seconds Field
- ❖ Pulse-Per-Second Output
- ❖ Auxiliary Snapshots with External Events

4.2.4.1 48-Bit Seconds Field

The GMAC-UNIV supports 80-bit timestamp. The timestamp has the following fields:

- ❖ UIInteger48 secondsField

The seconds field is the integer portion of the timestamp in units of seconds and is 48-bits wide. For example, 2.000000001 seconds are represented as secondsField = 0x0000_0000_0002.

- ❖ UIInteger32 nanosecondsField

The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 nanoseconds are represented as nanoSeconds = 0x0000_0001.

The nanoseconds field supports the following two modes:

- ◆ Digital rollover mode: In digital rollover mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds.
- ◆ Binary rollover mode: In binary rollover mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit.

You can set these modes by using the bit 9 (TSCTRLSSR) of [Register 448 \(Timestamp Control Register\)](#).

When you select the advanced timestamp feature, the timestamp maintained in the core is still 64-bit wide. The overflow to the upper 16-bits of seconds register happens once in 130 years. You can read the values of the upper 16-bits of the seconds field from the CSR register. This feature is optional. You can use this feature by enabling the parameter [IEEE1588 Higher Word Register Enable](#).

4.2.4.2 Pulse-Per-Second Output

The GMAC-UNIV supports the pulse-per-second (PPS) output that is given to indicate 1 second interval (default). You can change the frequency of the PPS output by setting the bits[3:0] in [Register 459 \(PPS Control Register\)](#).

4.2.4.3 Auxiliary Snapshots with External Events

The GMAC-UNIV supports taking auxiliary timestamp snapshots with an external event. For detailed information, see "[Auxiliary Snapshot](#)" on page [130](#).

4.2.5 Transmit Path Functions

In GMAC-CORE and GMAC-MTL configurations, there is no change in the transmit path functions for the Advanced timestamp feature.

In GMAC-AXI, GMAC-AHB, and GMAC-DMA configuration, the structure of the descriptor changes when you enable the advanced timestamp feature. The advanced timestamp feature is supported only through Alternate (Enhanced) descriptors format. The descriptor is 32-bytes long (8 DWORDS) and the snapshot of the timestamp is written in descriptor TDES6 and TDES7. For detailed information about descriptors, see "[Alternate or Enhanced Descriptors](#)" on page [414](#).

4.2.6 Receive Path Functions

When you select the advanced timestamp feature, the MAC processes the received frames to identify valid PTP frames. You can control the snapshot of the time, to be sent to the application, by using the following options of [Register 448 \(Timestamp Control Register\)](#).

- ❖ Enable snapshot for all frames.
- ❖ Enable snapshot for IEEE 1588 version 2 or version 1 timestamp.
- ❖ Enable snapshot for PTP frames transmitted directly over Ethernet or UDP-IP-Ethernet.
- ❖ Enable timestamp snapshot for the received frame for IPv4 or IPv6.
- ❖ Enable timestamp snapshot for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ or PDELAY_RESP) only.
- ❖ Enable the node to be a Master or Slave and select the snapshot type. This controls the type of messages for which snapshots are taken.



The GMAC-UNIV also supports the PTP messages over VLAN frames.

In GMAC-CORE configuration, the MAC provides the timestamp, along with EOF, on a 64-bit bus (`mri_timestamp_o`) on MRI. An additional signal (`mri_timestamp_val_o`) validates the presence of timestamp for the receive frame. The additional status is provided when you enable the advanced timestamp feature in coreConsultant.

In GMAC-MTL configuration, the MTL provides the timestamp on the data bus after the EOF data has been transferred. An additional signal (`ari_timestamp_val_o`) validates the timestamp. This signal is asserted to indicate the availability of timestamp. Once the timestamp is transferred, the MTL sends the receive status to the application. In 32-bit datawidth mode, the MTL provides the additional status related to the timestamp and IP checksum offload (IPC) on the data bus after the normal status is read. The additional status is provided only when the bit 0 of the normal status is set and is validated by the assertion of (`ari_rxstatus_val_o`) signal. In 64-bit and 128-bit datawidth mode, the MTL provides the additional status by using the bits 55:32 of the normal status (as in case of GMAC-CORE).

In GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations, the DMA returns the timestamp to the software inside the corresponding Transmit and Receive Descriptor. The advanced timestamp feature is supported only with the 32-bytes long Alternate (Enhanced) descriptor. The extended status, containing the timestamp message status and the IPC status, is written in descriptor RDES4 and the snapshot of the timestamp is written in descriptors RDES6 and RDES7. For detailed information about descriptors, see “[Alternate or Enhanced Descriptors](#)” on page [414](#)

4.2.7 Auxiliary Snapshot

The auxiliary snapshot feature allows you to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the sideband signal `ptp_aux_ts_trig_i`. This feature is independent of whether the system time is generated internally or given as input (on `ptp_timestamp_i[63:0]` bus). You can use this feature by enabling the [IEEE 1588 Auxiliary Snapshot Enable](#) parameter. This feature is available only when the Advanced Timestamp feature is selected.

The MAC stores these snapshots in a 4-deep FIFO. Only 64-bits of the timestamp are stored in the FIFO. You can read the upper 16-bits of seconds from [Register 457 \(System Time - Higher Word Seconds Register\)](#) when it is present. When a snapshot is stored, the MAC indicates this to the host with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, then a snapshot trigger-missed status(ATSSTM) is set in [Register 458 \(Timestamp Status Register\)](#). This indicates that the latest auxiliary snapshot of the timestamp was not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When a host reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the bit 19 (ATSFC) in [Register 448 \(Timestamp Control Register\)](#). When multiple snapshots are present in the FIFO, the count is indicated in bits [27:25] (ATSNS) of [Register 458 \(Timestamp Status Register\)](#).



Note The asynchronous input signal (`ptp_aux_ts_trig_i`) is first synchronized to `clk_ptp_ref_i` domain, then the snapshot is taken and transferred to the `clk_csr` domain. Because of this latency, the minimum gap between two such events on "`ptp_aux_ts_trig_i`" must have at least 4 cycles of `clk_ptp_i` plus 3 cycles of `clk_csr`. Otherwise, the rising-edge of the trigger is missed by the logic.

4.3 Audio/Video Bridging (AVB)

The Audio/Video Bridging (AVB) feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define the various aspects of AVB implementation:

- ❖ IEEE 802.1Qav/D6.0: Allows the bridges to provide time-sensitive and loss-sensitive real-time audio video (AV) data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.
- ❖ IEEE 802.1Qat/D2.1: Allows the network resources to be reserved for specific traffic streams traversing a bridged local area network.

- IEEE 802.1AS/D6.0: Specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications, such as audio and video and across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

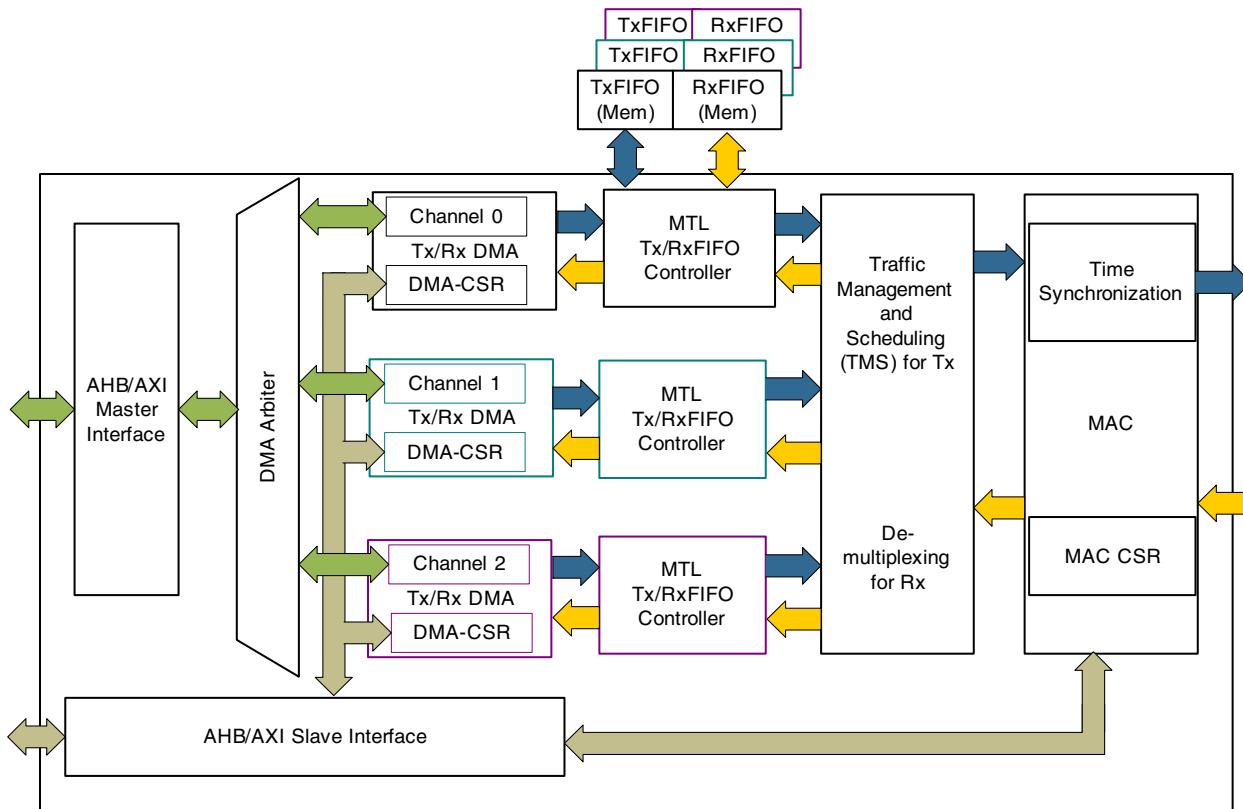
4.3.1 Block Diagram

The GMAC-UNIV supports the AV data transfer in 100 Mbps and 1000 Mbps modes. You can enable the AV feature by using the [Select AV Feature](#) parameter in coreConsultant. [Figure 4-4](#) displays the GMAC-AHB configuration when you enable the AV feature.



- In the current release of GMAC-UNIV, the AV feature is not supported with the GMAC-MTL configuration. In the GMAC-CORE configuration, the AVB function is limited to identifying the AV Type payloads in the Ethernet packet and indicating it on the MAC Receive Interface (MRI) through mri_ch_select signal along with the SOF. For more information, see “[Receive Path Functions](#)” on page [132](#).
- The Transmit Checksum Offload Engine is not present in AV channel 1 and channel 2.

Figure 4-4 Top-Level Block Diagram of AV Support



As shown in [Figure 4-4](#), one AHB/AXI master interface is connected to three DMA channels (channel 0, channel 1, and channel 2). The DMA arbiter helps in arbitration of all the paths (transmit and receive) in channel 0, channel 1, and channel 2. Each channel has a separate Control and Status register (CSR) for managing the transmit and receive functions, descriptor handling, and interrupt handling.

4.3.2 Transmit Path Functions

When you select the AV feature, the transmit paths of channel 0 and channel 1 are enabled by default. You can enable the transmit path of channel 2 by selecting the value 2 for the “[Select AV Transmit Channel\(s\)](#)” parameter.

The transmit path of channel 0 supports strict priority algorithm and is used for best-effort traffic. For a channel, the strict priority algorithm determines that a frame is available for transmission if the channel contains one or more frames. When the threshold mode for MTL Tx FIFO is enabled, the strict priority algorithm determines that a frame is available for transmission if the channel contains a partial frame of size equal to the programmed threshold limit.

The transmit paths of channel 1 and channel 2 support traffic management by using the credit-based shaper algorithm. For a channel, the credit-based shaper algorithm determines that a frame is available for transmission if the following conditions are true:

- ❖ The channel contains one or more frames.
- ❖ The credit for the channel is positive as per the algorithm.

You can disable the credit-based shaper algorithm for all channels or lower-priority channels. This means that you can either disable the credit-based shaper algorithm for channel 1 and channel 2 or only for channel 1. You should not disable the credit-based shaper algorithm for channel 2 and enable it for channel 1. When you disable the credit-based shaper algorithm for a channel, then the channel uses the default strict priority algorithm.

Each transmit DMA has a separate descriptor chain for fetching the transmit data. The transmit channel that gets the access to the system bus depends on the DMA arbiter. For information about the DMA arbiter, see “[DMA Arbiter Functions](#)” on page [134](#).

The transmit path has separate FIFOs (MTL layer) for each channel, as shown in [Figure 4-4](#). The data fetched by the DMA is put in the respective FIFO. The traffic management and scheduler unit (TMS) controls which FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for a channel (1 or 2), then the corresponding channel is selected for transmission if the following conditions are true:

- ❖ If the frame is available in the channel and has a positive or zero credit.
- ❖ If the higher priority channel has no frame waiting in the FIFO.

If the credit-based shaper algorithm is disabled for all channels, then the frame to be transmitted from a channel is selected based on the priority scheme described in [Table 4-10](#).

4.3.3 Receive Path Functions

When you select the AV feature, the receive path of channel 0 is enabled by default. All traffic is received on this channel. You can also enable the receive paths of channel 1 and channel 2 by using the [Select AV Receive Channel\(s\)](#) parameter in coreConsultant. By enabling the receive paths of channel 1 and channel 2, you can demultiplex the received data to send the packets into separate receive channels.

When receive path of only one channel is enabled, then to differentiate between the AV and non-AV traffic, the GMAC core provides a status that indicates if it is an AV packet and its corresponding VLAN Priority tag value. This status is updated in the Extended Status field of the receive descriptor as explained in “[Receive Descriptor](#)” on page [420](#). You can program the bits [15:0] in [Register 462 \(AV MAC Control Register\)](#) at offset 0x0738 to specify a value that is compared with the EtherType field of the incoming Ethernet frame to detect an AV packet. The AV packets can be of the following two types:

- ❖ AV data packets: The AV data packets are always tagged. The tagged AV control packets are received based on the programmed priority value. You can program the bits [18:16] (AVP) in [Register 462 \(AV MAC Control Register\)](#) to specify the channel to which an AV packet with a given priority must be sent.
- ❖ AV control packets: The AV control packets can be either tagged or untagged. The untagged AV control packets are received on channel 0 by default. To receive these packets on channel 1 or channel 2, you can program the bits 25:24 (AVCH) of [Register 462 \(AV MAC Control Register\)](#) (offset 0x0738). Similar to the AV data packets, the tagged AV control packets are received based on the programmed priority value.

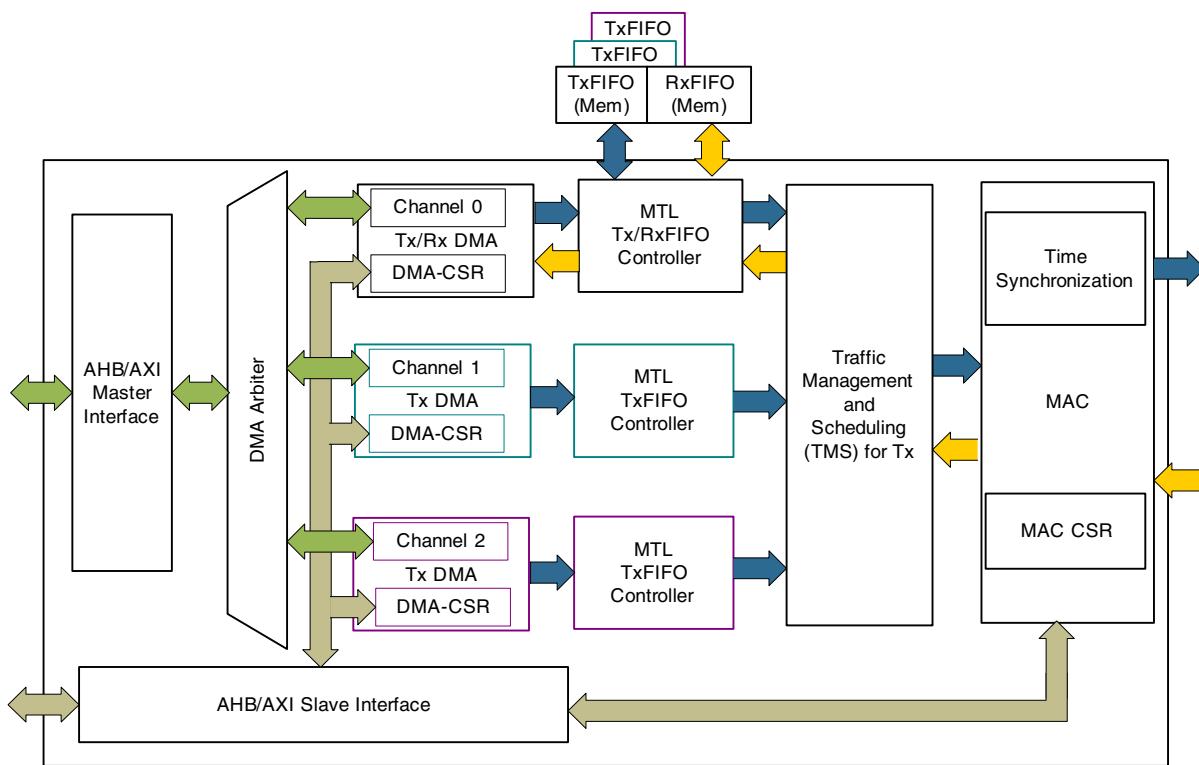
[Table 4-9](#) describes how tagged AV control packets and AV data packets are sent to a channel.

Table 4-9 Tagged AV Control Packets Scenarios

Scenario	Description
Receive path of only channel 1 is enabled	The AV packets with the priority value greater than or equal to the value programmed in the Register 462 (AV MAC Control Register) are received on this channel. All other packets are received on channel 0.
Receive paths of channel 1 and channel 2 are enabled	All AV packets with the priority value greater than or equal to the value programmed in Register 462 (AV MAC Control Register) are received on channel 2. The AV packets with priority value less than the value programmed are received on channel 1. All other packets are received on channel 0.
Priority value of 1 is programmed in Register 462 (AV MAC Control Register)	All AV packets with priority value from 1 to 7 are received on channel 2. The AV packets with priority value of 0 are received on channel 1. All other packets are received on channel 0.

In addition to the AV packets, you can receive the untagged PTP packets on channel 1 or channel 2. The PTP packets (tagged or untagged) are received on channel 0 by default. To receive these packets on channel 1 or channel 2, you need to program the bits [20:19] of the [Register 462 \(AV MAC Control Register\)](#).

You can remove the receive paths of channel 1 and channel 2 by setting the value of “Select AV Receive Channel(s)” parameter to zero. [Figure 4-5](#) displays the GMAC-AHB configuration without the receive paths of channel 1 and channel 2.

Figure 4-5 GMAC-AHB Top-Level Diagram Without Channel 1 and Channel 2 Receive Paths

4.3.4 DMA Arbiter Functions

The DMA arbiter performs the arbitration between the transmit and receive paths of DMA channel 0, channel 1, and channel 2 for accessing descriptors and data buffers.

The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The bit 1 ([DA: DMA Arbitration Scheme](#)) of [Register 0 \(Bus Mode Register\)](#) specifies the arbitration scheme (fixed or weighted round-robin) between the transmit and receive DMA of a channel. The fixed priority scheme is the default priority scheme for the DMA channels. In fixed priority scheme, the highest priority channel (channel 2) always wins the arbitration whenever it requests the bus. [Table 4-10](#) provides information about the priority levels of the DMA channels.

Table 4-10 Fixed Priority Scheme for DMA Channels

Priority Level	Channel
0 (low)	Channel 0
1	Channel 1
2(high)	Channel 2

4.3.4.1 Channel Weights

The bits [29:28] ([PRWG: Channel Priority Weights](#)) of [Register 0 \(Bus Mode Register\)](#), at offset 0x1000 for channel 0, 0x1100 for channel 1, and 0x1200 for channel 2, provide the weight for each channel as shown in [Table 4-11](#).

Table 4-11 Weights for Different DMA Channels

Bits [29:28]	Channel Weight
00	1
01	2
10	3
11	4

The fixed priority arbitration is enabled when the priority weight for all channels is 1, that is, value "00" for bits [29:28]. Otherwise, the arbitration among the channels is weighted round-robin. For example, when the bits [29:28] of each channel are programmed as: "00" for channel 0, "10" for channel 1, and "11" for channel 2, then the weighted round-robin priority is 1:3:4 for the channel 0, channel 1, and channel 2 respectively.

4.3.4.2 Priority Scheme

If you have enabled the transmit (Tx) DMA and receive (Rx) DMA of a channel, you can specify which DMA gets the bus when the channel gets the control of the bus. You can set the priority between the corresponding Tx DMA and Rx DMA by using the bit 27 ([TXPR: Transmit Priority](#)) of [Register 0 \(Bus Mode Register\)](#). For round-robin arbitration, you can use the bits [15:14] ([PR: Priority Ratio](#)) of the Bus Mode Register to specify the weighted priority between the Tx DMA and Rx DMA. [Table 4-12](#) provides information about the priority scheme between Tx DMA and Rx DMA.

Table 4-12 Priority Scheme for Tx DMA and Rx DMA

Bit 27	Bit 15	Bit 14	Bit 1	Priority Scheme
0	x	x	1	Rx always has priority over Tx
0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.
0	0	1	0	Rx has priority over Tx in the ratio 2:1.
0	1	0	0	Rx has priority over Tx in the ratio 3:1.
0	1	1	0	Rx has priority over Tx in the ratio 4:1.
1	x	x	1	Tx always has priority over Rx.
1	0	0	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.
1	0	1	0	Tx has priority over Rx in the ratio 2:1.
1	1	0	0	Tx has priority over Rx in the ratio 3:1.
1	1	1	0	Tx has priority over Rx in the ratio 4:1.



Bits 27, 15, 14, and 1 are not valid if one of the paths (transmit or receive) is not present in channel 1 or channel 2. The transmit and receive paths are always present in channel 0.

4.3.5 Slot Number Function

You can use the slot number function to schedule the data fetching by DMA from the system memory. This feature is useful when the source AV data needs to be transmitted at specific intervals. You can program the slot number at which the DMA should fetch the data from system memory in the [Transmit Descriptor Word 0 \(TDES0\)](#). This 4-bit field allows the host to schedule data up to 16 slots of 125 micro-second each. This field is applicable only for the AV channels (channel 1 and channel 2).

When DMA fetches a transmit descriptor, it compares the slot number of the transmit descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated every 125usec of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the seconds field of the system time is incremented, that is, the sub-second counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state till there is a match.

You can also program the bit 1 ([ASC: Advance Slot Check](#)) of the Slot Function Control and Status register (offset 0x1030) to enable the DMA to fetch the data only if it matches the current slot or the next two slots.



If the slot number in the descriptor is less than the reference slot number, the DMA takes it as a future slot.

You can enable the check for slot number by setting the bit 0 ([ESC: Enable Slot Comparison](#)) of the Slot Function Control and Status register (offset 0x1030). When this check is not enabled, the packets are fetched immediately after the descriptor is read. In addition, bits [19:16] ([RSN: Reference Slot Number](#)) indicate the value of the reference slot number in DMA.

4.3.6 Interrupt Functions

The interrupts from different DMA channels are combined by using the OR function to generate a single interrupt signal sbd_intr_o. Therefore, the software needs to read the Interrupt Status Registers of all DMA channels to get the source of interrupt. The MAC interrupt status (bits [29:26]) is updated in the interrupt status register of channel 0 only.

4.3.7 Credit-Based Shaper Algorithm Functions

The Traffic Manager and Scheduler (TMS) block (shown in [Figure 4-5](#)) uses the credit-based shaper algorithm to arbitrate the AV traffic in all channels and the legacy Ethernet traffic in channel 0. You can program the channel 1 and channel 2 to use the credit-based shaper algorithm.

The following sections provide information about how you can implement the credit-based shaper algorithm:

- ❖ [Credit Value](#)
- ❖ [idleSlopeCredit and sendSlopeCredit Values](#)
- ❖ [Bandwidth Status](#)

4.3.7.1 Credit Value

The credit value is accumulated every transmit clock cycle, that is, 40ns for 100 Mbps and 8ns for 1000 Mbps. The credit to be added or subtracted per cycle can be fractional, based on the required idleSlope and sendSlope values as described in [Table 4-13](#).

Table 4-13 Credit Value Per Transmit Cycle Example

Mode	Values	Description
100 Mbps	<ul style="list-style-type: none"> portTransmitRate = 100 Mbps idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) sendSlope = 30 Mbps 	<ul style="list-style-type: none"> credit = 2.8 bits accumulates per cycle (40ns) for the higher priority traffic class when best-effort frame is being transmitted. credit = 1.2 bits drains per cycle (40ns) when higher priority traffic class frame is being transmitted.

The DMA stores the channel traffic in the respective TxFIFO based on the slot number in the transmit descriptor (if enabled) or depending on the bandwidth availability on the AMBA AHB application bus.

The credit for a channel builds up only when the frame is available but it cannot be transmitted because the MAC is sending a frame from another channel. The GMAC core supports another mode in which the credit can build up in advance for a channel in which no frame is available in its FIFO. This enables sending a burst of high priority traffic in a channel as soon as data is available. You can enable this mode with bit 1 ([CC: Credit Control](#)) of the Channel 1 and Channel 2 CBS Control registers.

When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit and there is no frame to transmit in a channel. The credit does not accumulate when there is no frame waiting in a channel and other channels are transmitting. When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no frame to transmit in a channel. The credit accumulates even when there is no frame waiting in a channel and other channels are transmitting.

4.3.7.2 idleSlopeCredit and sendSlopeCredit Values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1024, such as, $2.8 \times 1024 = 2867$ and $1.2 \times 1024 = 1229$. In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12000 bits and 3036 bits respectively, then the values to be programmed in the hiCredit and loCredit registers of the corresponding channel are 12000×1024 bits and 2's complement of 3036×1024 respectively.

4.3.7.3 Bandwidth Status

The hardware maintains the status of the actual bandwidth consumed by each higher priority channel (channel 1 and channel 2) in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register gives the average number of bits transmitted during the previous programmed slot interval (1, 2, 4, 8, or 16 slots of 125us) in a channel. The status register is updated even if the credit-based shaper algorithm is not enabled for a channel. The number of slots over which the average bits transmitted per slot are computed is programmed in bits[6:4] of the CBS control register of the respective channel. For example, if you have programmed two slots, then the average bits are computed over slot numbers 0-1, 2-3, 4-5, and so on.

The value programmed in the idleSlopeCredit register of a channel is proportional to the bandwidth reserved for the channel. The software can allocate any bandwidth that is not used by the higher priority channel to the reserved bandwidth of the lower priority channel.

A lower priority channel, using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority channel that is using the credit-based shaper algorithm. However, a lower

priority channel that is using the strict-priority algorithm can use the unused reserved bandwidth of any higher priority channel that uses the credit-based shaper algorithm. For example, channel 1 and channel 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25% respectively) and channel 0 uses the strict-priority algorithm. If channel 1 uses only 40% of the reservedBandwidth, then the remaining 10% is used by channel 0. The channel 2 cannot exceed the reserved bandwidth of 25%.

4.4 Energy Efficient Ethernet

Energy Efficient Ethernet (EEE) is an optional operational mode that enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of Physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps, 1000 Mbps, and 10 Gbps.

The LPI mode allows power saving by switching off parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities and save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported and then select the set of parameters that common to both devices.



Note The EEE feature is not supported when the core is configured to use the TBI, RTBI, SMII, RMII, or SGMII single PHY interface. Even if the core supports multiple PHY interfaces, you should activate the EEE mode only when the core is operating with GMII, MII, or RGMII interface.

4.4.1 Transmit Path Functions

In the transmit path, the software must set the bit 16 (LPIEN) of [Register 12 \(LPI Control and Status Register\)](#) at offset 0x0030 to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and then starts transmitting the LPI pattern instead of the IDLE pattern during Interframe gap (IFG).

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN.
2. Asserts TX_ER.
3. Sets TXD[3:0] to 0x1 (for 100 Mbps) or TXD[7:0] to 0x01 (for 1000 Mbps).

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

4. Updates the status (bit 0 of [Register 12 \(LPI Control and Status Register\)](#) at offset 0x0030) and generates an interrupt.

To bring the PHY out of the LPI state, that is, when the software resets the LPIEN bit, the MAC performs the following tasks:

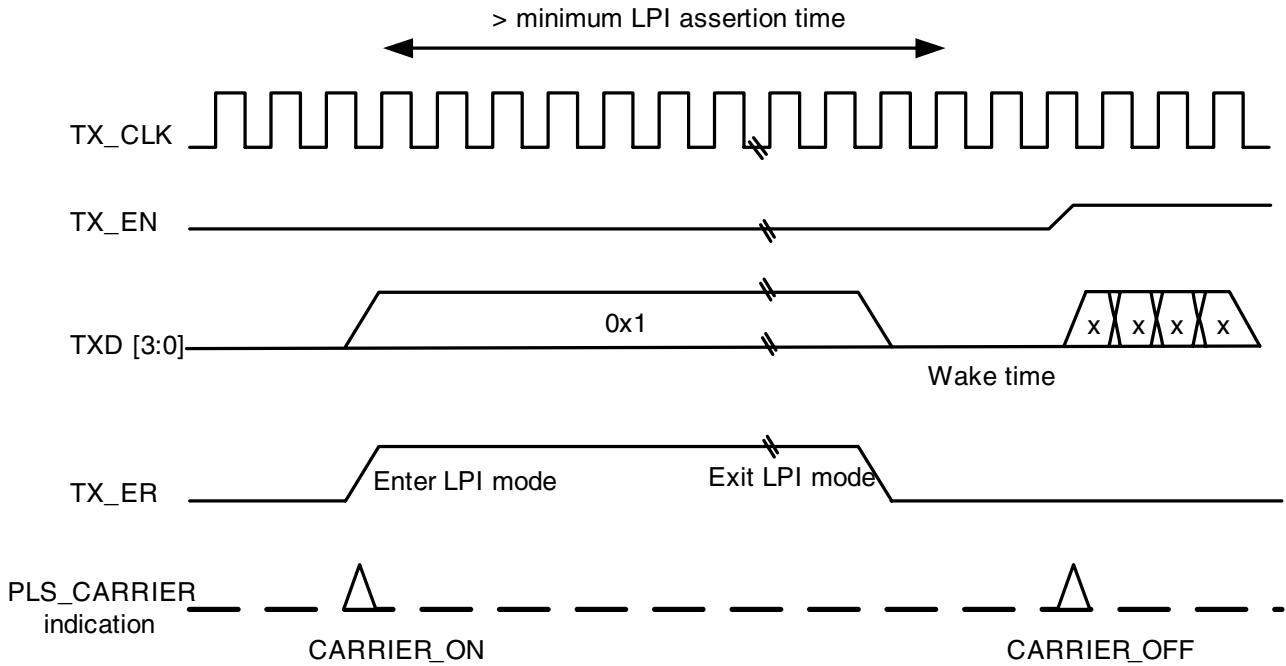
1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER.

The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in bits [15:0] ([TWT: LPI TW TIMER](#)) of [Register 13 \(LPI Timers Control Register\)](#) at offset 0x0034.

3. Updates the LPI exit status (bit 1 of [Register 12 \(LPI Control and Status Register\)](#) at 0x0030) and generates an interrupt.

[Figure 4-6](#) shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Figure 4-6 LPI Transitions (Transmit)



4.4.2 Receive Path Functions

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[7:0] to 0x01.
3. The PHY de-asserts RX_DV.

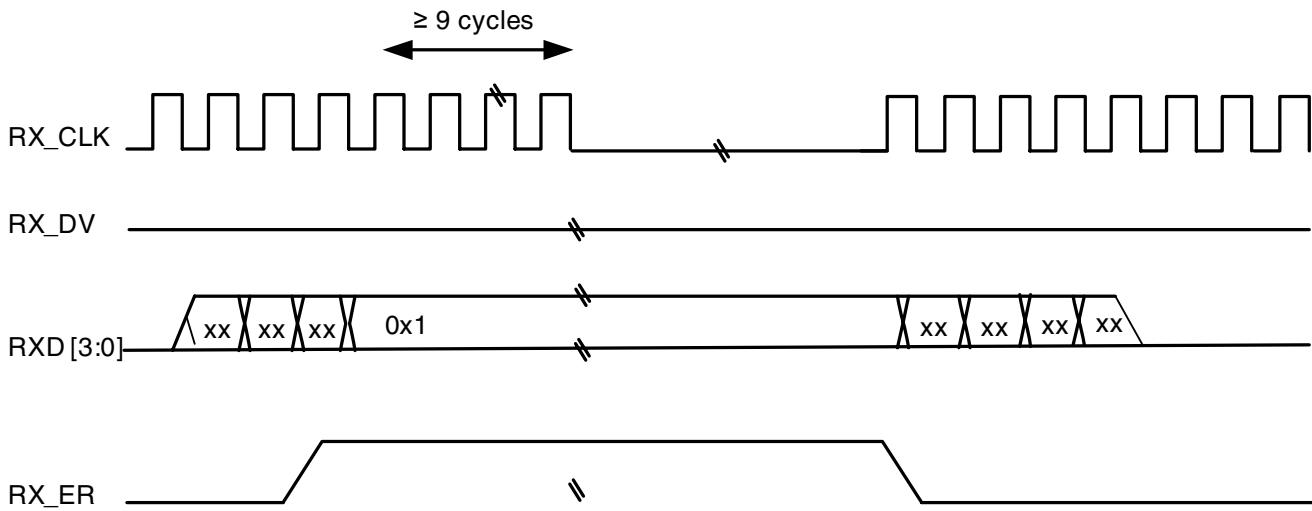
Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

4. The MAC updates the bit 2 (RLPIEN) of [Register 12 \(LPI Control and Status Register\)](#) at offset 0x0030 and generates an interrupt immediately.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-frame state.
2. The MAC updates the bit 3 (RLPIEX) of [Register 12 \(LPI Control and Status Register\)](#) at 0x0030 and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

[Figure 4-7](#) shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 4-7 LPI Transitions (Receive)

Note If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, then the PHY may halt the RX_CLK at any time more than 9 clock cycles after the start of the LPI state as shown in [Figure 4-7](#).

4.4.3 LPI Timers

The transmitter maintains the following two timers that are loaded with the respective values (from [Register 13 \(LPI Timers Control Register\)](#) at offset 0x0034):

- ❖ **LPI LS TIMER**

The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. The link status is indicated by the value of LS bit in [Register 12 \(LPI Control and Status Register\)](#) or the link status bit of [Register 54 \(SGMII/RGMII/SMII Status Register\)](#).



Note The Transmit Checksum Offload Engine is present only in Channel 0 even when you have enabled additional channels in AV mode.

You can enable the monitoring of bit 3 (link status bit) of Register 54 (SGMII/RGMII/SMII Status Register) by setting the bit 18 (PLSEN) of Register 12 (LPI Control and Status Register). This timer is cleared every time the link goes down and is incremented when the link is up again and the terminal count as programmed by the software is reached. The GMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE standard 802.3-az, version D2.0. The LPI LS TIMER is 10-bit wide. Therefore, the software can program up to 1023 milliseconds.

- ❖ **LPI TW TIMER**

The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. The MAC supports the LPI TW

TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 us.

4.5 Checksum Offload Engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. Because the most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams, the GMAC-UNIV has an optional Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path. This engine is not present by default and is included when you select this module in coreConsulatnt configuration.

4.5.1 Transmit Checksum Offload Engine

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each frame by setting the CIC bits (Bits 28:27 of TDES1, described in ["Transmit Descriptor 1 \(TDES1\)" on page 409](#)) in GMAC-AHB or GMAC-DMA configurations. In GMAC-MTL configuration, the ati_chksum_ctrl_i input pins ([Table 5-29](#)) controls the operation.



Note The checksum for TCP, UDP, or ICMP is calculated over a complete frame, then inserted into its corresponding header field. Because of this requirement, this function is enabled only when the Transmit FIFO is configured for Store-and-Forward mode (that is, when the TSF bit is set in DMA Register 6). If the core is configured for Threshold (cut-through) mode, the Transmit COE is bypassed.



Note You must make sure that the Transmit FIFO is deep enough to store a complete frame before that frame is transferred to the GMAC Core transmitter. The reason being that when space is not available to accept the programmed burst-length of data, then the MTL TxFIFO starts reading to avoid dead-lock. Once reading starts, the COE fails and consequently all succeeding frames may get corrupted because of improper recovery. Therefore, you must enable the checksum insertion only in the frames that are less than the following number of bytes in size (even in the store-and-forward mode):

FIFO Depth – PBL – 3 FIFO Locations

The PBL is the programmed burst-length in DMA Bus mode register or ati_pbl_i in GMAC-MTL.



Hint See IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

4.5.1.1 IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Ethernet frame's Type field has the value 0x0800 and the IP datagram's Version field has the value 0x4. The input frame's checksum field is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 16 in [Table 8-6](#)). This status bit is set whenever the values of the Ethernet Type field and

the IP header's Version field are not consistent, or when the Ethernet frame does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- ❖ For IPv4 datagrams:
 - ◆ The received Ethernet type is 0x0800, but the IP header's Version field is not equal to 0x4.
 - ◆ The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
 - ◆ The total frame length is less than the value given in the IPv4 Header Length field.
- ❖ For IPv6 datagrams:
 - ◆ The Ethernet type is 0x86dd but the IP header Version field is not equal to 0x6.
 - ◆ The frame ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

4.5.1.2 TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP.



- For non-TCP/UDP/ICMP/ICMPv6 payloads, this checksum engine is bypassed and nothing further is modified in the frame.
- Fragmented IP frames (IPv4 or IPv6), IP frames with security features (such as an encapsulated security payload), and IPv6 frames with routing headers are not processed by this engine. The checksum engine bypasses the checksum insertion for such frames even if the checksum insertion is enabled.

The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. This engine can work in the following two modes:

- ❖ The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input frame. This engine includes the Checksum field in the checksum calculation, then replaces the Checksum field with the final calculated checksum.
- ❖ The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.



For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16'h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (Bit 12 in [Table 8-6](#)). This engine sets the Payload Checksum Error status bit when it detects that the frame has been forwarded to the MAC Transmitter engine in the Store-and-Forward mode without the end-of-frame (EOF) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

4.5.2 Receive Checksum Offload Engine

There are two types of Receive Checksum Offload engine available for configuration. When you select only Enable IP Checksum for Received frames, the Type 1 engine is instantiated in the core. When you select the second option, Full Checksum Offload, the Type 2 engine is instantiated. The first configuration was made available in the initial 3.0 release of the core, while the second is available as of release 3.30a. Type 1 configuration is retained for backward compatibility, as the Type 2 engine is generally preferable.

4.5.2.1 Type 1

The application can enable IP header checksum checking and TCP/UDP checksum offload by setting the bit 10 (IPC) of [Register 0 \(MAC Configuration Register\)](#). This module calculates the 16-bit ones' complement of the Ethernet frame's payload data's (DATA field) ones' complement sum. The payload data is assumed to start from byte 15 (19 for a VLAN-tagged frame) of the received Ethernet frame. This module only processes IPv4 datagrams, bypassing and not processing all other types (such as IPv6). The receiver identifies a IPv4 frame by checking if the first nibble of the Ethernet payload (IP header) contains value 0x4.

This module also compares the calculated IP checksum with the received frame's IPv4 header checksum. Bytes 25 and 26 of the received Ethernet frame (29 and 30 for a VLAN-tagged frame) are taken as the IP header checksum. The header checksum is calculated against the header length field (20 bytes minimum). The result of the comparison (pass or fail) is given to the RFC, which sets the appropriate bit in the receive status word. If the Header Length field value is less than 5 or if the IP Version field does not equal 4, an error is indicated for the IP header checksum.

The ones' complement sum of the IP datagram's 16-bit payload is also calculated. The start of the payload is considered to be the data after the IP header. If the data payload ends with a non-aligned halfword, then a pad byte is added for the sum calculation. The 16-bit ones' complement of the resultant sum is forwarded to the RFC module, which inserts it into the data stream (towards the application) right after the FCS bytes (MS byte first) of the Ethernet payload. This 16-bit sum helps the software check the TCP/UDP header checksums faster. This 16-bit sum (which is always appended to the Ethernet frame in this mode) is invalid when the IP header checksum bit shows an Error status.

4.5.2.2 Type 2

In this mode, both IPv4 and IPv6 frames in the received Ethernet frames are detected and processed for data integrity. Like Type 1, you can enable this module by setting the bit 10 (IPC) of [Register 0 \(MAC Configuration Register\)](#). The GMAC receiver identifies IPv4 or IPv6 frames by checking for value 0x0800 or 0x86DD, respectively, in the received Ethernet frames' Type field. This identification applies to VLAN-tagged frames as well.

The Receive Checksum Offload engine calculates IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received frame does not have enough bytes, as indicated by the IPv4 header's Length field (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP/UDP/ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

As mentioned in “[TCP/UDP/ICMP Checksum Engine](#)” on page 142, this engine bypasses the payload of fragmented IP datagrams, IP datagrams with security features, IPv6 routing headers, and payloads other than TCP, UDP or ICMP. This information (whether the checksum engine is bypassed or not) is given in the receive status, as described in [Table 3-7](#) on page 106.

In this configuration, the core does not append any payload checksum bytes to the received Ethernet frames (as the Type 1 engine does).

4.6 MAC Management Counters

The counters in the MAC Management Counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted frames. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the MAC Control Interface (MCI). Each register is 32-bits wide. The write data is qualified with the corresponding mci_be_i signals. Therefore, non-32-bit accesses are allowed as long as the address is word-aligned.

The organization of these registers is shown in [Table 4-14](#). The MMCs are accessed using transactions, in the same way the CSR address space is accessed. The following sections in the chapter describe the various counters and list the address for each of the statistics counters. This address is used for Read/Write accesses to the desired transmit/receive counter.

The Receive MMC counters are updated for frames that are passed by the Address Filter (AFM) block. Statistics of frames that are dropped by the AFM module are not updated unless they are runt frames of less than 6 bytes (DA bytes are not received fully).

The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet frames. This gathering is only enabled when you select the Full Checksum Offload Engine in coreConsultant. The address map of the corresponding registers, 0x0200–0x02FC, is given in [Table 4-14](#).

You can select the MMC counters individually during configuration. The addresses for counters that are not selected become reserved. The width of each MMC counter is 32-bit by default. You can individually change the width to 16-bit for each selected MMC counter during configuration.

4.6.1 Address Assignments

The MMC register naming conventions are as follows.

- ❖ “tx” as a prefix or suffix indicates counters associated with transmission.
- ❖ “rx” as a prefix or suffix indicates counters associated with reception.
- ❖ “_g” as a suffix indicates registers that count good frames only.
- ❖ “_gb” as a suffix indicates registers that count frames regardless of whether they are good or bad.

The following definitions define the terminology used in [Tables 4-14](#) through [4-21](#).

- ❖ Transmitted frames are considered “good” if transmitted successfully. In other words, a transmitted frame is good if the frame transmission is not aborted because of any of the following errors:
 - ◆ Jabber Timeout
 - ◆ No Carrier/Loss of Carrier
 - ◆ Late Collision
 - ◆ Frame Underflow

- ◆ Excessive Deferral
- ◆ Excessive Collision
- ❖ Received frames are considered “good” if none of the following errors exists:
 - ◆ CRC error
 - ◆ Runt Frame (shorter than 64 bytes)
 - ◆ Alignment error (in 10/100 Mbps only)
 - ◆ Length error (non-Type frames only)
 - ◆ Out of Range (non-Type frames only, longer than maximum size)
 - ◆ GMII_RXER Input error
- ❖ The maximum frame size depends on the frame type, as follows:
 - ◆ Untagged frame maxsize = 1518
 - ◆ VLAN Frame maxsize = 1522
 - ◆ Jumbo Frame maxsize = 9018
 - ◆ JumboVLAN Frame maxsize = 9022

Table 4-14 MMC Register Map

GMAC CSR Register No.	Address Offset	Register Name	Register Description
64	0x0100	mmc_cntrl	MMC Control establishes the operating mode of MMC. For more details, see Table 4-15 .
65	0x0104	mmc_intr_rx	MMC Receive Interrupt maintains the interrupt generated from all of the receive statistic counters. For more details, see Table 4-16 .
66	0x0108	mmc_intr_tx	MMC Transmit Interrupt maintains the interrupt generated from all of the transmit statistic counters. For more details, see Table 4-17 .
67	0x010C	mmc_intr_mask_rx	MMC Receive Interrupt mask maintains the mask for the interrupt generated from all of the receive statistic counters. For more details, see Table 4-18 .
68	0x0110	mmc_intr_mask_tx	MMC Transmit Interrupt Mask maintains the mask for the interrupt generated from all of the transmit statistic counters. For more details, see Table 4-19 .
69	0x0114	txoctetcount_gb	Number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad frames.
70	0x0118	txframecount_gb	Number of good and bad frames transmitted, exclusive of retried frames.
71	0x011C	txbroadcastframes_g	Number of good broadcast frames transmitted.
72	0x0120	txmulticastframes_g	Number of good multicast frames transmitted.
73	0x0124	tx64octets_gb	Number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames.

Table 4-14 MMC Register Map (Continued)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
74	0x0128	tx65to127octets_gb	Number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.
75	0x012C	tx128to255octets_gb	Number of good and bad frames transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.
76	0x0130	tx256to511octets_gb	Number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.
77	0x0134	tx512to1023octets_gb	Number of good and bad frames transmitted with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.
78	0x0138	tx1024tomaxoctets_gb	Number of good and bad frames transmitted with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.
79	0x013C	txunicastframes_gb	Number of good and bad unicast frames transmitted.
80	0x0140	txmulticastframes_gb	Number of good and bad multicast frames transmitted.
81	0x0144	txbroadcastframes_gb	Number of good and bad broadcast frames transmitted.
82	0x0148	txunderflowerror	Number of frames aborted because of frame underflow error.
83	0x014C	txsinglecol_g	Number of successfully transmitted frames after a single collision in Half-duplex mode.
84	0x0150	txmulticol_g	Number of successfully transmitted frames after more than a single collision in Half-duplex mode.
85	0x0154	txdeferred	Number of successfully transmitted frames after a deferral in Half-duplex mode.
86	0x0158	txlatecol	Number of frames aborted because of late collision error.
87	0x015C	txexesscol	Number of frames aborted because of excessive (16) collision errors.
88	0x0160	txcarriererror	Number of frames aborted because of carrier sense error (no carrier or loss of carrier).
89	0x0164	txoctetcount_g	Number of bytes transmitted, exclusive of preamble, in good frames only.
90	0x0168	txframecount_g	Number of good frames transmitted.
91	0x016C	txexcessdef	Number of frames aborted because of excessive deferral error (deferred for more than two max-sized frame times).

Table 4-14 MMC Register Map (Continued)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
92	0x0170	txpauseframes	Number of good PAUSE frames transmitted.
93	0x0174	txvlanframes_g	Number of good VLAN frames transmitted, exclusive of retried frames.
94	0x0178	Reserved	N/A
95	0x017C	Reserved	N/A
96	0x0180	rxframecount_gb	Number of good and bad frames received.
97	0x0184	rxoctetcount_gb	Number of bytes received, exclusive of preamble, in good and bad frames.
98	0x0188	rxoctetcount_g	Number of bytes received, exclusive of preamble, only in good frames.
99	0x018C	rbroadcastframes_g	Number of good broadcast frames received.
100	0x0190	rmulticastframes_g	Number of good multicast frames received.
101	0x0194	rxcrcerror	Number of frames received with CRC error.
102	0x0198	rxalignmenterror	Number of frames received with alignment (dribble) error. Valid only in 10/100 mode.
103	0x019C	rxrunterror	Number of frames received with runt (<64 bytes and CRC error) error.
104	0x01A0	rxjabbererror	Number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.
105	0x01A4	rxundersize_g	Number of frames received with length less than 64 bytes, without any errors.
106	0x01A8	rxoversize_g	Number of frames received with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames), without errors.
107	0x01AC	rx64octets_gb	Number of good and bad frames received with length 64 bytes, exclusive of preamble.
108	0x01B0	rx65to127octets_gb	Number of good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.
109	0x01B4	rx128to255octets_gb	Number of good and bad frames received with length between 128 and 255 (inclusive) bytes, exclusive of preamble.
110	0x01B8	rx256to511octets_gb	Number of good and bad frames received with length between 256 and 511 (inclusive) bytes, exclusive of preamble.

Table 4-14 MMC Register Map (Continued)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
111	0x01BC	rx512to1023octets_gb	Number of good and bad frames received with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.
112	0x01C0	rx1024tomaxoctets_gb	Number of good and bad frames received with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.
113	0x01C4	rxunicastframes_g	Number of good unicast frames received.
114	0x01C8	rxlengtherror	Number of frames received with length error (Length type field ≠ frame size), for all frames with valid length field.
115	0x01CC	rxoutoffrange_type	Number of frames received with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).
116	0x01D0	rxpauseframes	Number of good and valid PAUSE frames received.
117	0x01D4	rxfifooverflow	Number of missed received frames because of FIFO overflow. This counter is not present in the GMAC-CORE configuration.
118	0x01D8	rxvlanframes_gb	Number of good and bad VLAN frames received.
119	0x01DC	rxwatchdogerror	Number of frames received with error because of watchdog timeout error (frames with a data load larger than 2,048 bytes).
120:127	0x01E0–0x01FC	Reserved	
128	0x0200	mmc_ipc_intr_mask_rx	MMC IPC Receive Checksum Offload Interrupt Mask maintains the mask for the interrupt generated from the receive IPC statistic counters. See Table 4-20 for further detail.
129	0x0204	Reserved	
130	0x0208	mmc_ipc_intr_rx	MMC Receive Checksum Offload Interrupt maintains the interrupt that the receive IPC statistic counters generate. See Table 4-21 for further detail.
131	0x020C	Reserved	
132	0x0210	rxipv4_gd_frms	Number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload
133	0x0214	rxipv4_hdrerr_frms	Number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors
134	0x0218	rxipv4_nopay_frms	Number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine
135	0x021C	rxipv4_frag_frms	Number of good IPv4 datagrams with fragmentation
136	0x0220	rxipv4_udsbl_frms	Number of good IPv4 datagrams received that had a UDP payload with checksum disabled

Table 4-14 MMC Register Map (Continued)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
137	0x0224	rxipv6_gd_frms	Number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads
138	0x0228	rxipv6_hdrerr_frms	Number of IPv6 datagrams received with header errors (length or version mismatch)
139	0x022C	rxipv6_nopay_frms	Number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers
140	0x0230	rxudp_gd_frms	Number of good IP datagrams with a good UDP payload. This counter is not updated when the rxipv4_udsbl_frms counter is incremented.
141	0x0234	rxudp_err_frms	Number of good IP datagrams whose UDP payload has a checksum error
142	0x0238	rxtcp_gd_frms	Number of good IP datagrams with a good TCP payload
143	0x023C	rxtcp_err_frms	Number of good IP datagrams whose TCP payload has a checksum error
144	0x0240	rxicmp_gd_frms	Number of good IP datagrams with a good ICMP payload
145	0x0244	rxicmp_err_frms	Number of good IP datagrams whose ICMP payload has a checksum error
146:147	0x0248–0x024C	Reserved	
148	0x0250	rxipv4_gd_octets	Number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter or in the octet counters listed below).
149	0x0254	rxipv4_hdrerr_octets	Number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter.
150	0x0258	rxipv4_nopay_octets	Number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 header's Length field is used to update this counter.
151	0x025C	rxipv4_frag_octets	Number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 header's Length field is used to update this counter.
152	0x0260	rxipv4_udsbl_octets	Number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes.
153	0x0264	rxipv6_gd_octets	Number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data

Table 4-14 MMC Register Map (Continued)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
154	0x0268	rxipv6_hdrerr_octets	Number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the IPv6 header's Length field is used to update this counter.
155	0x026C	rxipv6_nopay_octets	Number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 header's Length field is used to update this counter.
156	0x0270	rxudp_gd_octets	Number of bytes received in a good UDP segment. This counter (and the counters below) does not count IP header bytes.
157	0x0274	rxudp_err_octets	Number of bytes received in a UDP segment that had checksum errors
158	0x0278	rxtcp_gd_octets	Number of bytes received in a good TCP segment
159	0x027C	rxtcp_err_octets	Number of bytes received in a TCP segment with checksum errors
160	0x0280	rxicmp_gd_octets	Number of bytes received in a good ICMP segment
161	0x0284	rxicmp_err_octets	Number of bytes received in an ICMP segment with checksum errors
162:191	0x0288– 0x02FC	Reserved	

4.6.2 MMC Register Description

4.6.2.1 MMC Control Register

The MMC Control register establishes the operating mode of the management counters.

Table 4-15 MMC Control Register

Bit	Access Type	Reset Value	Description
31:6	R	0000_000H	Reserved
5	R/W	0	<p>Full-Half preset</p> <p>When low and bit4 is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0xFFFF_F800 (half - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (half - 16).</p> <p>When high and bit4 is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (full - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (full - 16).</p> <p>For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and frame counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.</p>

Table 4-15 MMC Control Register (Continued)

Bit	Access Type	Reset Value	Description
4	R_W_SC	0	Counters Preset When set, all counters are initialized or preset to almost full or almost half as per Bit 5. This bit is cleared automatically after 1 clock cycle. This bit along with bit5 is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.
3	R/W	0	MMC Counter Freeze When set, this bit freezes all the MMC counters to their current value. (None of the MMC counters are updated because of any transmitted or received frame until this bit is reset to 0. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.)
2	R_W	0	Reset on Read When set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read.
1	R_W	0	Counter Stop Rollover When set, counter after reaching maximum value does not roll over to zero.
0	R_W_SC	0	Counters Reset When set, all counters are reset. This bit is cleared automatically after 1 clock cycle



When Bit 0 and Bit 4 are set in the same cycle, Counters get preset immediately after getting cleared.

4.6.2.2 MMC Receive Interrupt Register

The MMC Receive Interrupt register maintains the interrupts that are generated when the receive statistic counters reach half their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Table 4-16 MMC Receive Interrupt Register

Bit	Access Type	Reset Value	Description
31:24	RO	00H	Reserved
23	R_SS_RC	0	The bit is set when the rxwatchdog error counter reaches half the maximum value, and also when it reaches the maximum value.
22	R_SS_RC	0	The bit is set when the rxvlanframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-16 MMC Receive Interrupt Register (Continued)

Bit	Access Type	Reset Value	Description
21	R_SS_RC	0	The bit is set when the rx fifo overflow counter reaches half the maximum value, and also when it reaches the maximum value.
20	R_SS_RC	0	The bit is set when the rx pause frames counter reaches half the maximum value, and also when it reaches the maximum value.
19	R_SS_RC	0	The bit is set when the rx out of range type counter reaches half the maximum value, and also when it reaches the maximum value.
18	R_SS_RC	0	The bit is set when the rx length error counter reaches half the maximum value, and also when it reaches the maximum value.
17	R_SS_RC	0	The bit is set when the rx unicast frames_g counter reaches half the maximum value, and also when it reaches the maximum value.
16	R_SS_RC	0	The bit is set when the rx 1024 to max octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
15	R_SS_RC	0	The bit is set when the rx 512 to 1023 octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
14	R_SS_RC	0	The bit is set when the rx 256 to 511 octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
13	R_SS_RC	0	The bit is set when the rx 128 to 255 octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
12	R_SS_RC	0	The bit is set when the rx 65 to 127 octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
11	R_SS_RC	0	The bit is set when the rx 64 octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
10	R_SS_RC	0	The bit is set when the rx oversize_g counter reaches half the maximum value, and also when it reaches the maximum value.
9	R_SS_RC	0	The bit is set when the rx undersize_g counter reaches half the maximum value, and also when it reaches the maximum value.
8	R_SS_RC	0	The bit is set when the rx jabber error counter reaches half the maximum value, and also when it reaches the maximum value.
7	R_SS_RC	0	The bit is set when the rx runt error counter reaches half the maximum value, and also when it reaches the maximum value.
6	R_SS_RC	0	The bit is set when the rx alignment error counter reaches half the maximum value, and also when it reaches the maximum value.
5	R_SS_RC	0	The bit is set when the rx CRC error counter reaches half the maximum value, and also when it reaches the maximum value.
4	R_SS_RC	0	The bit is set when the rx multicast frames_g counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-16 MMC Receive Interrupt Register (Continued)

Bit	Access Type	Reset Value	Description
3	R_SS_RC	0	The bit is set when the rxbroadcastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
2	R_SS_RC	0	The bit is set when the rxoctetcount_g counter reaches half the maximum value, and also when it reaches the maximum value.
1	R_SS_RC	0	The bit is set when the rxoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	R_SS_RC	0	The bit is set when the rxframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.



Note R_SS_RC means that this register bit is set internally and is cleared when the Counter register is read.

4.6.2.3 MMC Transmit Interrupt Register

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Table 4-17 MMC Transmit Interrupt Register

Bit	Access Type	Reset Value	Description
31:25	RO	00H	Reserved
24	R_SS_RC	0	The bit is set when the txvlanframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
23	R_SS_RC	0	The bit is set when the txpauseframes error counter reaches half the maximum value, and also when it reaches the maximum value.
22	R_SS_RC	0	The bit is set when the txoexcessdef counter reaches half the maximum value, and also when it reaches the maximum value.
21	R_SS_RC	0	The bit is set when the txframecount_g counter reaches half the maximum value, and also when it reaches the maximum value.
20	R_SS_RC	0	The bit is set when the txoctetcount_g counter reaches half the maximum value, and also when it reaches the maximum value.
19	R_SS_RC	0	The bit is set when the txcarriererror counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-17 MMC Transmit Interrupt Register (Continued)

Bit	Access Type	Reset Value	Description
18	R_SS_RC	0	The bit is set when the txexcesscol counter reaches half the maximum value, and also when it reaches the maximum value.
17	R_SS_RC	0	The bit is set when the txlatecol counter reaches half the maximum value, and also when it reaches the maximum value.
16	R_SS_RC	0	The bit is set when the txdeferred counter reaches half the maximum value, and also when it reaches the maximum value.
15	R_SS_RC	0	The bit is set when the txmulticol_g counter reaches half the maximum value, and also when it reaches the maximum value.
14	R_SS_RC	0	The bit is set when the txsinglecol_g counter reaches half the maximum value, and also when it reaches the maximum value.
13	R_SS_RC	0	The bit is set when the txunderflowerror counter reaches half the maximum value, and also when it reaches the maximum value.
12	R_SS_RC	0	The bit is set when the txbroadcastframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.
11	R_SS_RC	0	The bit is set when the txmulticastframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.
10	R_SS_RC	0	The bit is set when the txunicastframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.
9	R_SS_RC	0	The bit is set when the tx1024tomaxoctets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
8	R_SS_RC	0	The bit is set when the tx512to1023octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
7	R_SS_RC	0	The bit is set when the tx256to511octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
6	R_SS_RC	0	The bit is set when the tx128to255octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
5	R_SS_RC	0	The bit is set when the tx65to127octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
4	R_SS_RC	0	The bit is set when the tx64octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
3	R_SS_RC	0	The bit is set when the txmulticastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
2	R_SS_RC	0	The bit is set when the txbroadcastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
1	R_SS_RC	0	The bit is set when the txframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-17 MMC Transmit Interrupt Register (Continued)

Bit	Access Type	Reset Value	Description
0	R_SS_RC	0	The bit is set when the txoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

4.6.2.4 MMC Receive Interrupt Mask Register

The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when receive statistic counters reach half their maximum value, and when they reach their maximum values. This register is 32 bits wide.

Table 4-18 MMC Receive Interrupt Mask Register

Bit	Access Type	Reset Value	Description
31:24	RO	00H	Reserved
23	R_W	0	Setting this bit masks the interrupt when the rxwatchdog counter reaches half the maximum value, and also when it reaches the maximum value.
22	R_W	0	Setting this bit masks the interrupt when the rxvlanframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.
21	R_W	0	Setting this bit masks the interrupt when the rxfifooverflow counter reaches half the maximum value, and also when it reaches the maximum value.
20	R_W	0	Setting this bit masks the interrupt when the rxpauseframes counter reaches half the maximum value, and also when it reaches the maximum value.
....	R_W Same as explained in the previous rows for corresponding counters in MMC Receive Interrupt Register
1	R_W	0	Setting this bit masks the interrupt when the rxoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	R_W	0	Setting this bit masks the interrupt when the rxframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

4.6.2.5 MMC Transmit Interrupt Mask Register

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when transmit statistic counters reach half their maximum value, and when they reach their maximum values. This register is 32 bits wide.

Table 4-19 MMC Transmit Interrupt Mask Register

Bit	Access Type	Reset Value	Description
31:25	RO	00H	Reserved
24	R_W	0	Setting this bit masks the interrupt when the txvlanframes_g counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-19 MMC Transmit Interrupt Mask Register (Continued)

Bit	Access Type	Reset Value	Description
23	R_W	0	Setting this bit masks the interrupt when the txpauseframes counter reaches half the maximum value, and also when it reaches the maximum value.
22	R_W	0	Setting this bit masks the interrupt when the txoexcessdef counter reaches half the maximum value, and also when it reaches the maximum value.
21	R_W	0	Setting this bit masks the interrupt when the txframecount_g counter reaches half the maximum value, and also when it reaches the maximum value.
....	R_W Same as above for corresponding counters in MMC Transmit Interrupt Register
1	R_W	0	Setting this bit masks the interrupt when the txframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	R_W	0	Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

4.6.2.6 MMC Receive Checksum Offload Interrupt Mask Register

The MMC Receive Checksum Offload Interrupt Mask register maintains the masks for the interrupts generated when the receive IPC (Checksum Offload) statistic counters reach half their maximum value, and when they reach their maximum values. This register is 32 bits wide.

Table 4-20 MMC Receive Checksum Offload Interrupt Mask Register

Bit	Access Type	Reset Value	Description
31:30	RO	00	Reserved
29	R_W	0	Setting this bit masks the interrupt when the rxic平_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
28	R_W	0	Setting this bit masks the interrupt when the rxic平_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
27	R_W	0	Setting this bit masks the interrupt when the rxtcp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
26	R_W	0	Setting this bit masks the interrupt when the rxtcp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
25	R_W	0	Setting this bit masks the interrupt when the rxudp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
24	R_W	0	Setting this bit masks the interrupt when the rxudp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
23	R_W	0	Setting this bit masks the interrupt when the rxipv6_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.
22	R_W	0	Setting this bit masks the interrupt when the rxipv6_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-20 MMC Receive Checksum Offload Interrupt Mask Register (Continued)

Bit	Access Type	Reset Value	Description
21	R_W	0	Setting this bit masks the interrupt when the rxipv6_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
20	R_W	0	Setting this bit masks the interrupt when the rxipv4_udslsbl_octets counter reaches half the maximum value, and also when it reaches the maximum value.
19	R_W	0	Setting this bit masks the interrupt when the rxipv4_frag_octets counter reaches half the maximum value, and also when it reaches the maximum value.
18	R_W	0	Setting this bit masks the interrupt when the rxipv4_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.
17	R_W	0	Setting this bit masks the interrupt when the rxipv4_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
16	R_W	0	Setting this bit masks the interrupt when the rxipv4_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
15:14	RO	0	Reserved
13	R_W	0	Setting this bit masks the interrupt when the rxicmp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
12	R_W	0	Setting this bit masks the interrupt when the rxicmp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
11	R_W	0	Setting this bit masks the interrupt when the rxtcp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
10	R_W	0	Setting this bit masks the interrupt when the rxtcp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
9	R_W	0	Setting this bit masks the interrupt when the rxudp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
8	R_W	0	Setting this bit masks the interrupt when the rxudp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
7	R_W	0	Setting this bit masks the interrupt when the rxipv6_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.
6	R_W	0	Setting this bit masks the interrupt when the rxipv6_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
5	R_W	0	Setting this bit masks the interrupt when the rxipv6_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
4	R_W	0	Setting this bit masks the interrupt when the rxipv4_udslsbl_frms counter reaches half the maximum value, and also when it reaches the maximum value.
3	R_W	0	Setting this bit masks the interrupt when the rxipv4_frag_frms counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-20 MMC Receive Checksum Offload Interrupt Mask Register (Continued)

Bit	Access Type	Reset Value	Description
2	R_W	0	Setting this bit masks the interrupt when the rxipv4_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.
1	R_W	0	Setting this bit masks the interrupt when the rxipv4_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
0	R_W	0	Setting this bit masks the interrupt when the rxipv4_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.

4.6.2.7 MMC Receive Checksum Offload Interrupt Register

The MMC Receive Checksum Offload Interrupt register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Checksum Offload Interrupt register is 32-bits wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The counter's least-significant byte lane (bits[7:0]) must be read to clear the interrupt bit.

Table 4-21 MMC Receive Checksum Offload Interrupt Register

Bit	Access Type	Reset Value	Description
31:30	RO	00	Reserved
29	R_SS_RC	0	The bit is set when the rxicmp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
28	R_SS_RC	0	The bit is set when the rxicmp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
27	R_SS_RC	0	The bit is set when the rxtcp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
26	R_SS_RC	0	The bit is set when the rxtcp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
25	R_SS_RC	0	The bit is set when the rxudp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
24	R_SS_RC	0	The bit is set when the rxudp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
23	R_SS_RC	0	The bit is set when the rxipv6_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.
22	R_SS_RC	0	The bit is set when the rxipv6_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
21	R_SS_RC	0	The bit is set when the rxipv6_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-21 MMC Receive Checksum Offload Interrupt Register (Continued)

Bit	Access Type	Reset Value	Description
20	R_SS_RC	0	The bit is set when the rxipv4_udslbl_octets counter reaches half the maximum value, and also when it reaches the maximum value.
19	R_SS_RC	0	The bit is set when the rxipv4_frag_octets counter reaches half the maximum value, and also when it reaches the maximum value.
18	R_SS_RC	0	The bit is set when the rxipv4_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.
17	R_SS_RC	0	The bit is set when the rxipv4_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
16	R_SS_RC	0	The bit is set when the rxipv4_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
15:14	RO	00	Reserved
13	R_SS_RC	0	The bit is set when the rxicmp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
12	R_SS_RC	0	The bit is set when the rxicmp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
11	R_SS_RC	0	The bit is set when the rxtcp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
10	R_SS_RC	0	The bit is set when the rxtcp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
9	R_SS_RC	0	The bit is set when the rxudp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
8	R_SS_RC	0	The bit is set when the rxudp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
7	R_SS_RC	0	The bit is set when the rxipv6_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.
6	R_SS_RC	0	The bit is set when the rxipv6_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
5	R_SS_RC	0	The bit is set when the rxipv6_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
4	R_SS_RC	0	The bit is set when the rxipv4_udslbl_frms counter reaches half the maximum value, and also when it reaches the maximum value.
3	R_SS_RC	0	The bit is set when the rxipv4_frag_frms counter reaches half the maximum value, and also when it reaches the maximum value.
2	R_SS_RC	0	The bit is set when the rxipv4_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.

Table 4-21 MMC Receive Checksum Offload Interrupt Register (Continued)

Bit	Access Type	Reset Value	Description
1	R_SS_RC	0	The bit is set when the rxipv4_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
0	R_SS_RC	0	The bit is set when the rxipv4_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.

4.7 Power Management Block

The power management (PMT) block supports the reception of network (remote) wake-up frames and magic packet frames. The PMT block does not perform the clock gate function, but generates interrupts for wake-up frames and magic packets that the MAC receives.

When you enable the power-down mode in PMT block, then the MAC drops all received frames and does not forward these frames to the application. The MAC comes out of the power-down mode only when a magic packet or a remote wake-up frame is received and the corresponding detection is enabled.

The PMT block is available in the receive path of GMAC-UNIV. You can enable the PMT block by selecting the [Enable Power Management block \(PMT\)](#) option in coreConsultant. You can select both types of power management frames (remote wake-up and magic packet). You can use the [Wake-Up Frame Enable](#) and [Magic Packet Enable](#) bits of the [PMT Control and Status Register](#) to generate power management events. The application should program these bits. You can access the PMT registers in the similar manner as you access the GMAC CSR registers. For mapping information, see registers 10 and 11 in [Table 6-4](#).

4.7.1 PMT Block Description

4.7.1.1 PMT Control and Status Register

The PMT CSR programs the request wake-up events and monitors the wake-up events.

Table 4-22 PMT Control and Status Register

Bit	Access Type	Reset Value	Description
31	R_WS_SC	0	Wake-Up Frame Filter Register Pointer Reset When set, resets the remote wake-up frame filter register pointer to 3'b000. It is automatically cleared after 1 clock cycle.
30:10	R	0000_00H	Reserved
9	R_W	00	Global Unicast When set, enables any unicast packet filtered by the GMAC (DAF) address recognition to be a wake-up frame.
8:7	R	0	Reserved
6	R_SS_RC	0	Wake-Up Frame Received When set, this bit indicates the power management event is generated because of the reception of a wake-up frame. This bit is cleared by a Read into this register.

Table 4-22 PMT Control and Status Register (Continued)

Bit	Access Type	Reset Value	Description
5	R_SS_RC	0	Magic Packet Received When set, this bit indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared by a Read into this register.
4:3	R	00	Reserved
2	R_W	0	Wake-Up Frame Enable When set, enables generation of a power management event because of wake-up frame reception.
1	R_W	0	Magic Packet Enable When set, enables generation of a power management event because of magic packet reception.
0	R_WS_SC	0	Power Down When set, all received frames are dropped. This bit is cleared automatically when a magic packet or wake-up frame is received, and power-down mode is disabled. Frames, that are received after this bit is cleared, are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Wake-Up Frame Enable bit is set high.

4.7.1.2 Remote Wake-Up Frame Filter Register

The register wkupfmfilter_reg, at address (028H), loads the Wake-up Frame Filter register. To load values in a Wake-up Frame Filter register, the entire register (wkupfmfilter_reg) must be written. The wkupfmfilter_reg register is loaded by sequentially loading the eight register values in address (028) for wkupfmfilter_reg0, wkupfmfilter_reg1, ... wkupfmfilter_reg7, respectively. The wkupfmfilter_reg register is read in a similar way.



Note If you are accessing these registers in byte or half-word mode, then the internal counter, to access the appropriate wkupfmfilter_reg, is incremented when CPU accesses the lane3 (or lane 0 in big-endian mode).

Figure 4-8 Wake-Up Frame Filter Register

wkupfmfilter_reg0	Filter 0 Byte Mask											
wkupfmfilter_reg1	Filter 1 Byte Mask											
wkupfmfilter_reg2	Filter 2 Byte Mask											
wkupfmfilter_reg3	Filter 3 Byte Mask											
wkupfmfilter_reg4	RSVD	Filter 3 Command	RSVD	Filter 2 Command	RSVD	Filter 1 Command	RSVD	Filter 0 Command				
wkupfmfilter_reg5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset					
wkupfmfilter_reg6	Filter 1 CRC - 16				Filter 0 CRC - 16							
wkupfmfilter_reg7	Filter 3 CRC - 16				Filter 2 CRC - 16							

4.7.1.2.1 Filter *i* Byte Mask

The Filter *i* Byte Mask register defines the bytes of the frame that are examined by filter *i* (0, 1, 2, and 3) in order to determine whether or not a frame is a wake-up frame. The MSB (thirty-first bit) must be zero.

The bit *j* [30:0] is the Byte Mask. If bit *j* (byte number) of the Byte Mask is set, then the CRC block processes the Filter *i* Offset + *j* of the incoming frame; otherwise Filter *i* Offset + *j* is ignored.

4.7.1.2.2 Filter *i* Command

The 4-bit Filter *i* Command controls the filter *i* operation. The bit 3 specifies the address type, defining the destination address type of the pattern.

When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame. Bit 2 and Bit 1 are reserved. Bit 0 is the enable for filter *i*. If bit 0 is not set, filter *i* is disabled.

4.7.1.2.3 Filter *i* Offset

This Filter *i* Offset register defines the offset (within the frame) from which the filter *i* examines the frames. This 8-bit pattern-offset is the offset for the filter *i* first byte to be examined. The minimum allowed offset is 12, which refers to the 13th byte of the frame. The offset value 0 refers to the first byte of the frame.

4.7.1.2.4 Filter *i* CRC-16

This Filter *i* CRC-16 register contains the CRC_16 value calculated from the pattern, as well as the byte mask programmed to the wake-up filter register block.

4.7.2 Remote Wake-Up Frame Detection

When the MAC is in sleep mode and the remote wake-up bit is enabled in the [PMT Control and Status Register](#) (002C), the normal operation is resumed after a remote wake-up frame is received. The Application writes all eight wake-up filter registers, by performing a sequential Write to address (0028). The Application

enables remote wake-up by writing a 1 to bit 2 ([Wake-Up Frame Enable](#)) of the PMT Control and Status Register.

The PMT block supports four programmable filters that allow support of different receive frame patterns. If the incoming frame passes the address filtering of Filter Command, and if Filter CRC-16 matches the CRC of the incoming pattern, then the MAC identifies the frame as wake-up frame.

The Filter Offset (see [Filter i Offset](#)) determines the offset from which the frame is to be examined. The Filter Byte Mask (see [Filter i Byte Mask](#)) determines which bytes of the frame must be examined. The thirty-first bit of Byte Mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The wake-up frame is checked only for length error, FCS error, dribble bit error, GMII error, collision, and to ensure that it is not a runt frame. Even if the wake-up frame is more than 512 bytes long, if the frame has a valid CRC value, it is considered valid. The wake-up frame detection is updated in the PMT Control and Status register for every remote wake-up frame received. A PMT interrupt to the Application triggers a Read to the PMT Control and Status register to determine reception of a wake-up frame.



Note A sample C routine that generates CRC-16 for a sequence of data has been provided in the resources/ directory of your workspace.

4.7.3 Magic Packet Detection

The magic packet frame is based on a method that uses Advanced Micro Device's Magic Packet technology to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network.

The MAC checks only those magic packets that are addressed to the MAC or a broadcast address to determine whether these packets meet the wake-up requirements. The magic packets that pass the address filtering (unicast or broadcast) are checked to determine whether they meet the remote Wake-on-LAN data format of 6 bytes of all ones followed by a GMAC Address appearing 16 times.

The application enables the magic packet wake-up by writing 1 to bit 1 ([Magic Packet Enable](#)) of the [PMT Control and Status Register](#). The PMT block constantly monitors each frame addressed to the node for a specific magic packet pattern. Each frame received is checked for a 48'hFF_FF_FF_FF_FF_FF pattern following the destination and source address field. The PMT block then checks the frame for 16 repetitions of the GMAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the PMT block again scans the 48'hFF_FF_FF_FF_FF_FF pattern in the incoming frame. The 16 repetitions can be anywhere in the frame, but must be preceded by the synchronization stream (48'hFF_FF_FF_FF_FF_FF). The device can also accept a multicast frame, as long as the 16 duplications of the GMAC address are detected.

If the MAC address of a node is 48'h00_11_22_33_44_55, then the GMAC scans for the following data sequence:

```

Destination Address Source Address ..... FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
...CRC

```

The magic packet detection is updated in the PMT Control and Status Register for the received magic packet. A PMT interrupt to the Application triggers a read to the PMT Control and Status Register to determine whether a magic packet frame has been received.

4.7.4 System Considerations During Power Down

The GMAC-UNIV neither gates nor stops clocks when the power-down mode is enabled. Power saving by clock gating must be done outside the core by the application. The receive data path must be clocked with clk_rx_i during the power-down mode, because it is involved in the magic packet/wake-on-LAN frame detection. However, the transmit path and the application path clocks can be gated off during the power-down mode.

The pmt_intr_o signal is asserted when a valid wake-up frame is received. This signal is generated in the clk_rx_i domain. The pmt_intr_o signal is separated from the sbd_intr_o interrupt because during the power-down mode, only the Rx clock needs to be active. All other clocks can be stopped.

The recommended power-down and wake-up sequence is as follows.

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. These transmissions can be detected when Transmit Interrupt (TI-DMA Register 5[0]) is received.
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the MAC Configuration register.
3. Wait until the Receive DMA empties all the frames from the Rx FIFO (a software timer may be required).
4. Enable Power-Down mode by appropriately configuring the PMT registers.
5. Enable the MAC Receiver and enter the Power-Down mode.
6. Gate the application and transmit clock inputs to the core (and other relevant clocks in the system) to reduce power and enter Sleep mode.
7. On receiving a valid wake-up frame, the GMAC-UNIV asserts the pmt_intr_o signal and exits the Power-Down mode.
8. On receiving the interrupt, the system must enable the application and transmit clock inputs to the core.
9. Read the PMT Status register to clear the interrupt, then enable the other modules in the system and resume normal operation.



Note The pmt_intr_o is generated in the clk_rx_i domain. Therefore, it is not cleared immediately when the PMT Status register is read. The resultant clear signal has to cross to the clk_rx_i domain and then clear the interrupt source. This delay is at least 4 clock cycles of clk_rx and can be significant when the core is operating in the 10 Mbps mode.

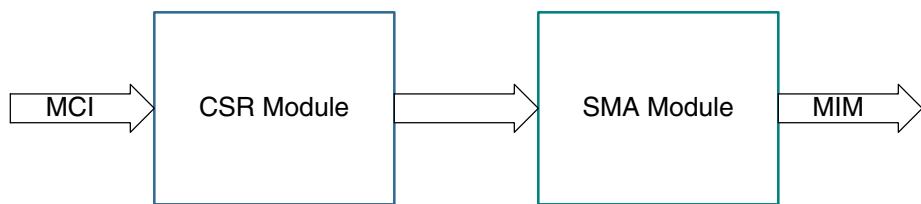
The pmt_intr_o signal is generated in the clk_rx_i domain so that the application clock (csr clock) can be stopped during the power-down mode. Therefore, if the interrupt is generated on the application clock then switching off the application clock is not possible.

4.8 Station Management Agent

The Station Management Agent (SMA) module allows the Application to access any PHY registers through a 2-wire Station Management interface (MIM). The interface supports accessing up to 32 PHYs.

The application can select one of the 32 PHYs and one of the 32 registers within any PHY and send control data or receive status information. Only one register in one PHY can be addressed at any given time. For more details on the communication from the Application to the PHYs, refer to the Reconciliation Sublayer and Media Independent Interface Specifications section of the IEEE 802.3z specification, 1000BASE Ethernet. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in [Figure 4-9](#).

Figure 4-9 SMA Interface Block



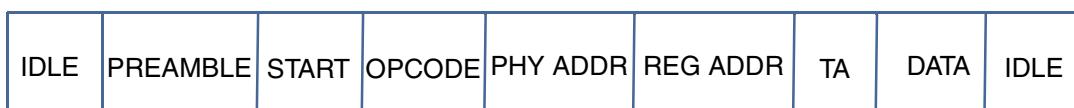
4.8.1 Functions

The MAC initiates the Management Write/Read operation. The clock gmii_mdc_o is a divided clock from the Application clock clk_csr_i. The divide factor depends on the clock range setting in the GMII Address register. Clock range is set as follows:

Selection	clk_csr_i	MDC Clock
0000	60-100 MHz	clk_csr_i/42
0001	100-150 MHz	clk_csr_i/62
0010	20-35 MHz	clk_csr_i/16
0011	35-60 MHz	clk_csr_i/26
0100	150-250 MHz	clk_csr_i/102
0101	250-300 MHz	clk_csr_i/124
0110, 0111	Reserved	

The gmii_mdc_o is the derivative of the Application clock clk_csr_i. The Management operation is performed through the gmii_mdi_i, gmii_mdo_o, and gmii_mdo_o_e signals. You need to implement a three-state buffer outside the MAC to interface with an external PHY. For more information about connecting the MAC to an external PHY, see [“SMA to PHY Interface”](#) on page 441.

The following figure displays the structure of a frame on the MDIO line:



[Table 4-23](#) provides information about the frame fields.

Table 4-23 MDIO Frame Structure

Field	Description
IDLE	The mdio line is three-state; there is no clock on gmii_mdc_o.
PREAMBLE	32 continuous bits of value 1
START	Start-of-frame is 2'01
OPCODE	2'b10 for Read and 2'b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround is 2'bZ0 for Read and 2'b10 for Write
DATA	Any 16-bit value. In a Write operation, the GMAC drives mdio; in a Read operation, PHY drives it.

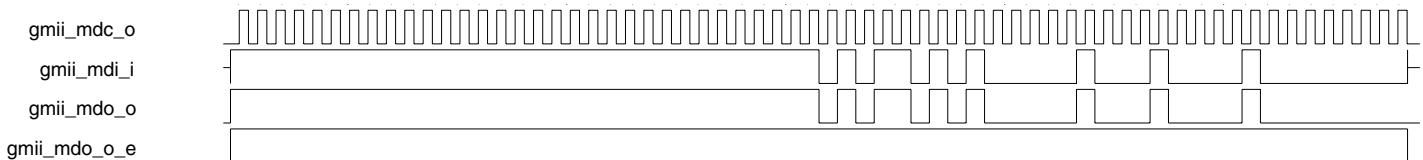
4.8.2 GMII/MII Management Write Operation

When you set the GMII Write bit and GMII Busy bit in [Register 4 \(GMII Address Register\)](#), the GMAC CSR module transfers the PHY address, the register address in PHY, and the write data (GMII Data Register) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the GMII Management Interface using the Management Frame Format specified in the GMII specifications (as per IEEE Standard 802.3-2002, *Section 22.2.4.5*). The application should not change the content of the GMII Address register or GMII Data register while the transaction is going on. The MAC ignores the Write operations performed to the GMII Address register or the GMII Data Register during this period (the Busy bit is high), and the transaction is completed without any error on the MCI interface.

When the Write operation is complete, the SMA indicates this to the CSR which then resets the Busy bit. The SMA module divides the CSR (Application) clock with the clock divider programmed (CR bits of GMII Address Register) to generate the MDC clock for this interface. The GMAC drives the MDIO line for the complete duration of the frame. The frame format for the Write operation is as follows:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111...11	01	01	AAAAAA	RRRRR	10	DDD . . . DDD	Z

[Figure 4-10](#) is a reference for the Write operation.

Figure 4-10 Management Write Operation

4.8.3 GMII/MII Management Read Operation

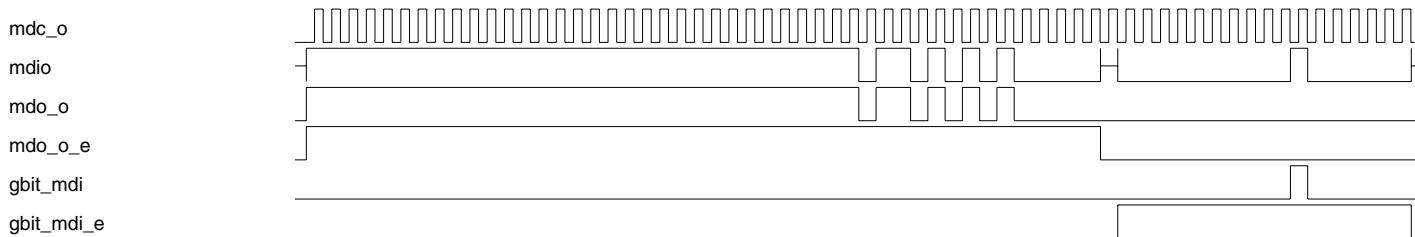
When you set the GMII Busy bit and GMII Write bit to 0 in [Register 4 \(GMII Address Register\)](#), the GMAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the GMII Management Interface using the Management Frame Format specified in the GMII specifications (as per IEEE Standard 802.3-2002, *Section 22.2.4.5*). The application should not change the contents of the GMII Address register or GMII Data register while the transaction is going on. The MAC ignores the Write operations to the GMII Address register or GMII Data Register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface.

When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the GMII Data register with the data read from the PHY. The SMA module divides the CSR (Application) clock with the clock divider programmed (CR bits of GMII Address Register) to generate the MDC clock for this interface. The GMAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. The frame format for the Read operation is as follows:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111...11	01	10	AAAAA	RRRRR	Z0	DDD DDD	Z

[Figure 4-11](#) is a reference for the read operation.

Figure 4-11 Management Read Operation



In verification testbench (VTB), the PHY model physical address is fixed to 0x15. If physical address is programmed different from 0x15, then the PHY model does not drive any value on the input data. When no value is driven, the tristate Z is connected to the input data in the test bench.

In real system environment, gbit_mdi is pulled high during the turnaround. Therefore, high value is present on the gbit_mdi.

4.9 Physical Coding Sublayer

The GMAC-UNIV provides an IEEE 802.3z compliant 10-bit Physical Coding Sublayer (PCS) interface that you can use when the GMAC-UNIV is configured for TBI/RTBI/SGMII PHY interface. The PCS interface transmits 10-bit code groups with a 125-MHz transmit clock. In addition, the interface can receive 10-bit code groups with two out-of-phase 62.5-MHz clocks in TBI interface mode and a single 125 MHz clock in SGMII/RTBI interface mode.

In PCS interface, the auto-negotiation process is implemented as described in IEEE 802.3z specification, [Figure 37-6](#). The auto-negotiation provides an automatic process for two MACs to share configuration information. The MACs use this information to operate at the best of their common abilities. You can also

use manual configuration by using the GMII Control register and Advertisement register and disabling the auto-negotiation in the GMII Control register.

You can include the optional PCS module in the GMAC controller by selecting the TBI/RTBI/SGMII interface option in the coreConsultant during configuration.

4.9.1 PCS Functions

The PCS interface performs the following tasks:

- ❖ Encapsulation in the transmit path.
- ❖ Decapsulation in the receive path.
- ❖ Synchronization.
- ❖ Auto-Negotiation.
- ❖ Start-of-packet (SOP) and end-of-packet (EOP) detection and generation
- ❖ Collision and Carrier sense generation

4.9.1.1 Transmit

During the transmit process, the MAC performs the encapsulation function according to the following rules.

- ❖ The first byte of the preamble is replaced with the /S/ code group.
- ❖ All data in the frame are encoded according to 8B/10B encoding standard.
- ❖ The /T/R/R/ or /T/R/ code group is inserted after the last byte of FCS.
- ❖ An idle code group /I/ is transmitted between frames.
- ❖ Collision and Carrier sense are generated.
- ❖ /V/ Error insertion is also supported.

4.9.1.2 Receive

During the receive process, the MAC performs the decapsulation function according to the following rules.

- ❖ An /I/S/ code group sequence results in assertion of the internal data valid signal.
- ❖ The /S/ code group is replaced by a preamble byte.
- ❖ The data of the received frame is decoded according to 10B/8B decoding standard.
- ❖ The /T/R/R/ or /T/R/ code group sequence results in assertion of the internal data valid signal.

4.9.1.3 Synchronization

The synchronization process determines whether the underlying receive channel is ready for operation. The MAC implements synchronization of the received code groups according to the IEEE 802.3z standard. The MAC acquires the synchronization when the PHY achieves bit synchronization and the MAC detects three consecutive valid /COMMA/D/ patterns sent by the PHY in the even code group position. Once MAC acquires the synchronization, it is possible to begin auto-negotiation.

If MAC receives four consecutive invalid code groups or commas in odd clock after acquiring synchronization, the synchronization process restarts automatically. A built-in hysteresis prevents any restarting of the synchronization process happening because of invalid code groups reception.

4.9.1.4 Auto-Negotiation

A subset of the management registers is supported (Register 48-Register 53) in the GMAC. These registers can be accessed through MCI. The GMAC supports only base page exchange and does not support any next page exchanges.

Once auto-negotiation (AN) is enabled, any one the following events triggers the AN process.

- ❖ Request from another device (transmitting configuration code groups)
- ❖ Loss of synchronization for more than 10ms
- ❖ Error condition detected while receiving /C/ or /I/ ordered set
- ❖ Auto-negotiation restart bit is enabled

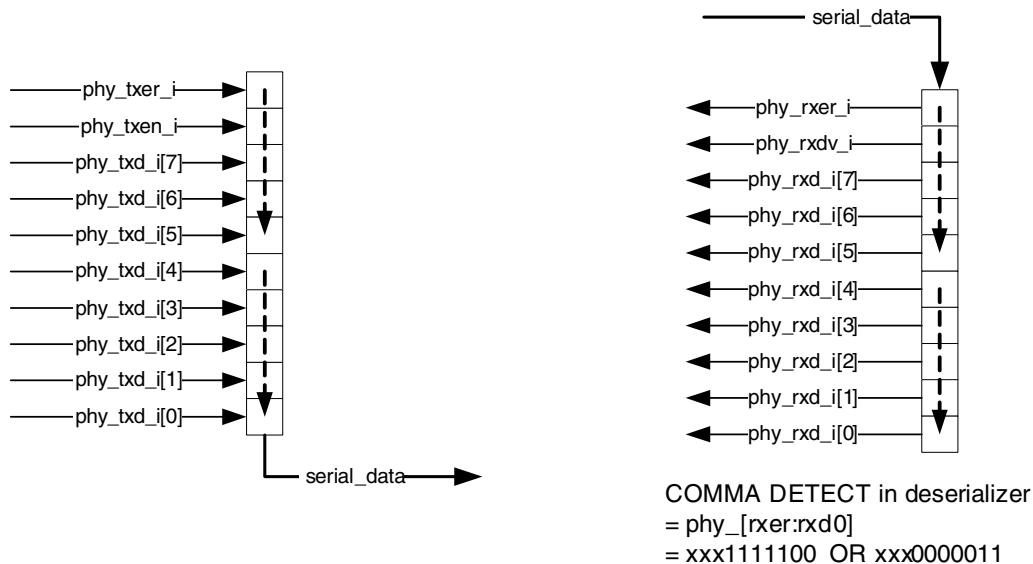
Once AN has been initiated, the GMAC initially sends configuration words with all zeros for a duration determined by the value in the link timer (10 ms). After the link timer duration expires, the contents of the AN Advertisement register are transmitted within the configuration words. The AN Advertisement register should be loaded prior to AN being enabled.

The AN process is completed after the base page exchange. Upon completion of the base page exchange, the ANC bit in the GMII Status register is set, and GMAC is programmed automatically for the Duplex and Flow Control Modes, based on the priority resolution. Once the AN is completed, the GMAC can transmit frames.

4.9.1.5 PMA SerDes

The 10-bit TBI signals from the PCS block are connected to the PMA layer of the PHY chip. The connection of the 10-bit transmit data signals with the serializer and the 10-bit receiver data signals from the de-serializer of the PMA are shown in [Figure 4-12](#).

Figure 4-12 PMA Serializer and De-Serializer



4.10 Reduced Media Independent Interface

The Reduced Media Independent Interface (RMII) specification reduces the pin count between Ethernet PHYs and Switch ASICs (only in 10/100 mode). According to the IEEE 802.3u standard, an MII contains 16 pins for data and control. In devices incorporating multiple MAC or PHY interfaces (such as switches), the

number of pins adds significant cost with increase in port count. The RMII specification addresses this problem by reducing the pin count to 7 for each port – a 62.5% decrease in pin count.

The RMII module is instantiated between the MAC and the PHY. This helps in translating MAC's MII into the RMII. The RMII block has the following characteristics:

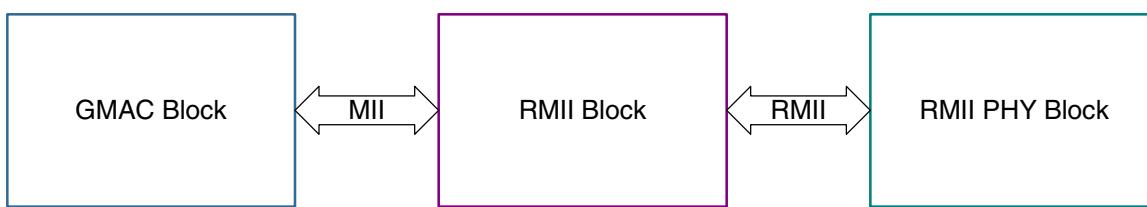
- ❖ Supports 10-Mbps and 100-Mbps operating rates. It does not support the 1000-Mbps operation.
- ❖ Provides independent, 2-bit wide transmit and receive paths by sourcing two clock references externally. The guidelines for clock connections and phase differences when the GMAC operates with an RMII are provided in "[Clocks With RMII](#)" on page 433.

The RMII interface can be included by selecting the RMII interface in the coreConsultant during the configuration.

4.10.1 Block Diagram

[Figure 4-13](#) shows the position of the RMII block relative to the DWC Ether MAC 10/100/1000 Universal and RMII PHY. The RMII block is placed in front of the DWC Ether MAC 10/100/1000 Universal to translate the MII signals to RMII signals.

Figure 4-13 RMII Block Diagram

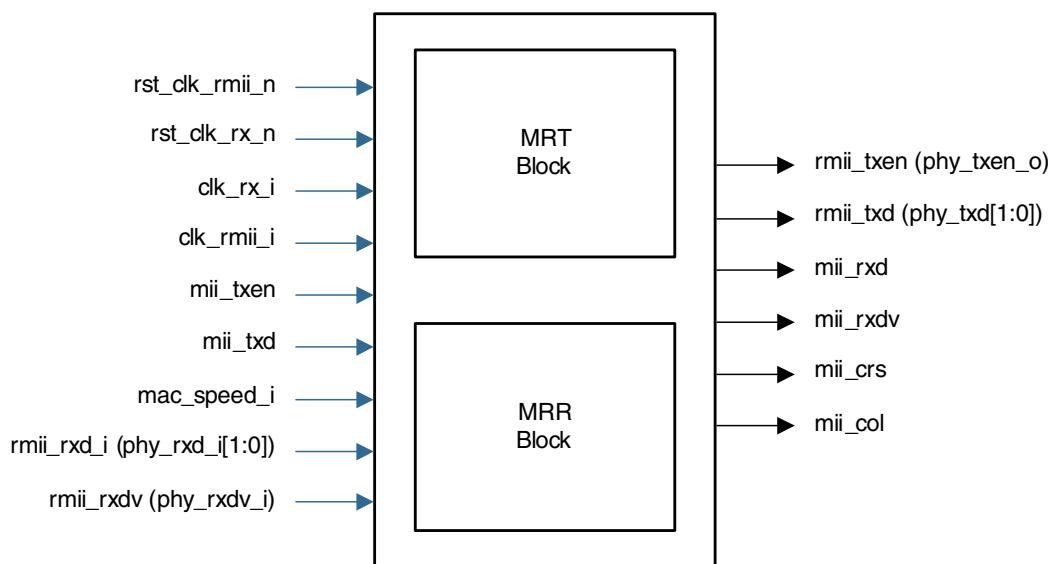


4.10.2 Block Overview

The following list describes the RMII's hardware components, which are shown in [Figure 4-14](#). Each of these blocks is briefly described in the following sections.

MII-RMII Transmit (MRT) Block: This block translates all MII transmit signals to RMII transmit signals. All RMII signals are synchronous to clk_rmii_i.

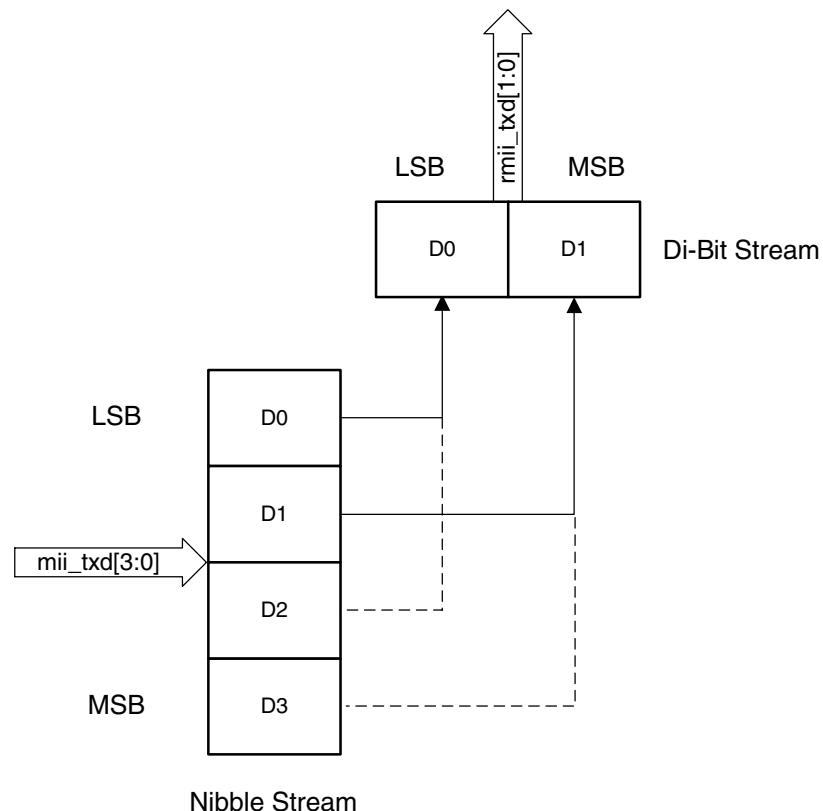
MII-RMII Receive (MRR) Block: This block translates all RMII receive signals to MII receive signals. All MII signals are synchronous to clk_rx_i. You must ensure that the same clock is connected to both clk_tx_i and clk_rx_i clock input ports in RMII mode. This is required because clock-MUXing logic is avoided inside the GMAC core.

Figure 4-14 RMII Pinout

Note The mac_speed_i signal configures the RMII to operate at 10 Mbps or 100 Mbps. This signal is either taken directly as a core input or driven from the MAC Configuration register's FES bit. See [Section 9.4](#) for details.

4.10.3 Transmit Bit Ordering

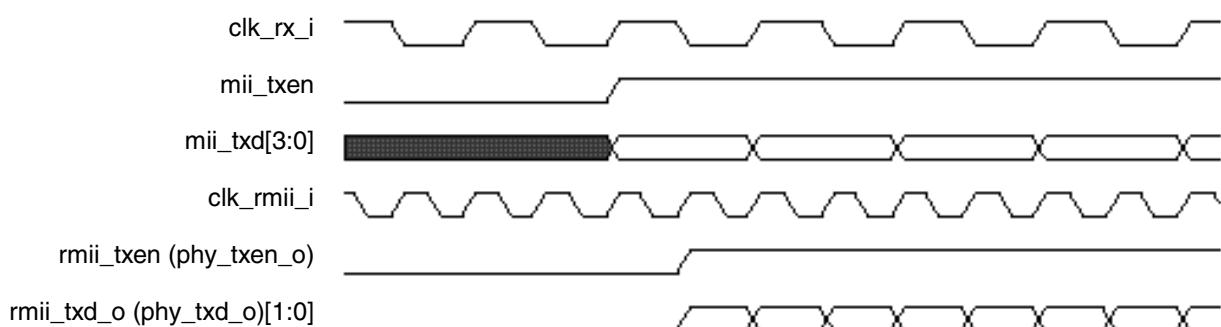
Each nibble from the MII must be transmitted on the RMII a di-bit at a time with the order of di-bit transmission shown in [Figure 4-15](#). The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

Figure 4-15 Transmission Bit Ordering

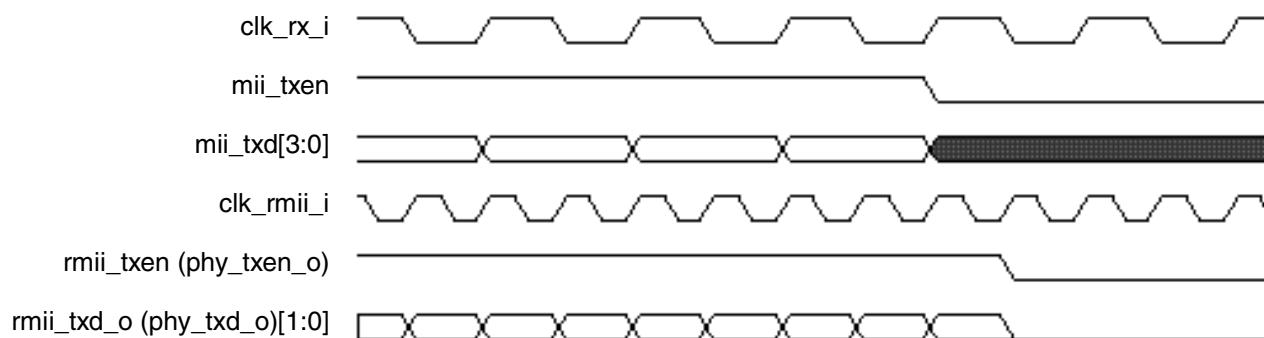
4.10.4 RMII Transmit Timing Diagrams

[Figures 4-16 through 4-19](#) show MII-to-RMII transaction timing.

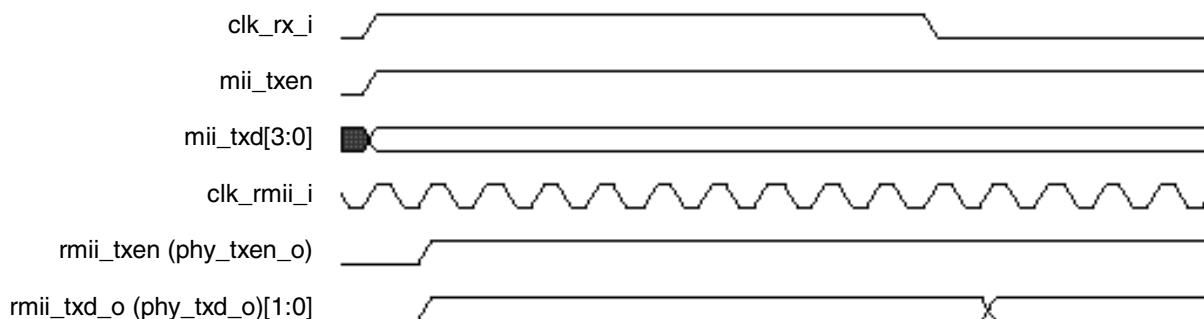
[Figure 4-16](#) shows the start of MII transmission and the following RMII transmission in 100-Mbps mode.

Figure 4-16 Start of MII and RMII Transmission in 100-Mbps Mode

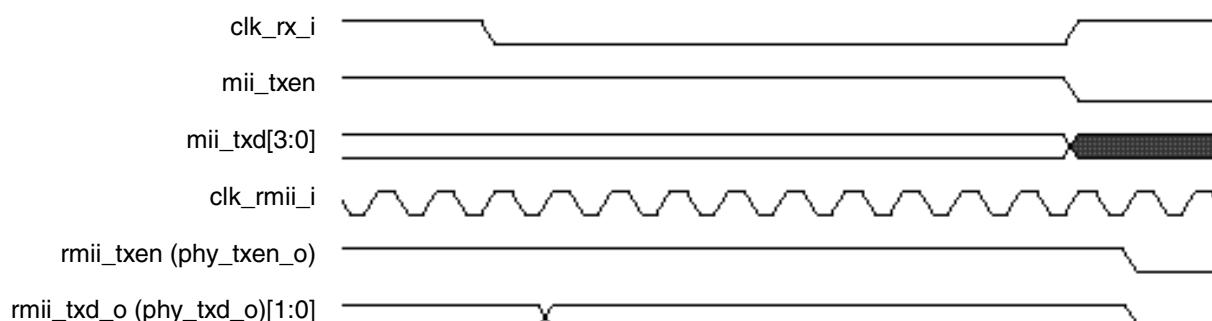
[Figure 4-17](#) shows the end of frame transmission for MII and RMII in 100-Mbps mode.

Figure 4-17 End of MII and RMII Transmission in 100-Mbps Mode

[Figure 4-18](#) shows the start of MII transmission and the following RMII transmission in 10-Mbps mode.

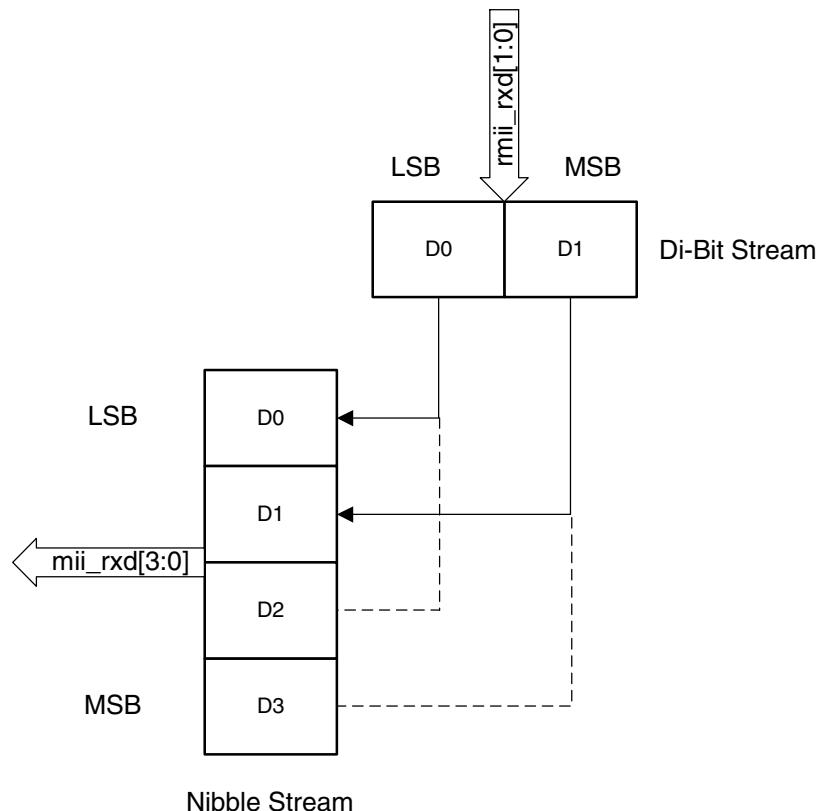
Figure 4-18 Start of MII and RMII Transmission in 10-Mbps Mode

[Figure 4-19](#) shows the end of MII transmission and RMII transmission in 10-Mbps mode.

Figure 4-19 End of MII and RMII Transmission in 10-Mbps Mode

Receive Bit Ordering

Each nibble is transmitted to the MII from the di-bit received from the RMII in the nibble transmission order shown in [Figure 4-20](#). The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 4-20 Receive Bit Ordering

4.11 Reverse Media Independent Interface

The Reverse Media Independent Interface (RevMII) connects two Ethernet MACs with a point-to-point link. The RevMII link provides a simple and low-cost alternative to using the Ethernet PHY in the system.

The RevMII has the following features:

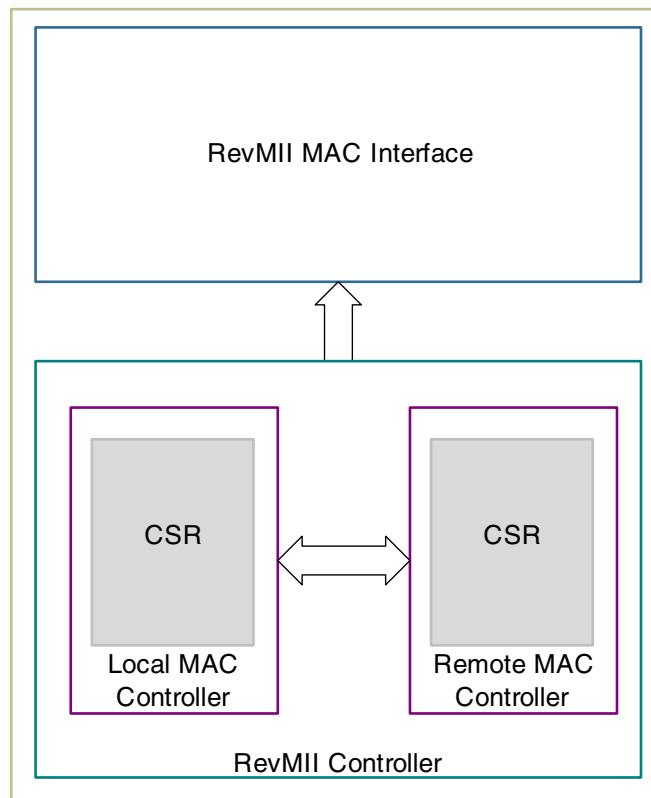
- ❖ Connects two Ethernet MACs directly as defined in the IEEE 802.3u standard.
- ❖ Supports configuration through Serial Management Interface (SMI).
- ❖ Supports basic register set. This includes the Control and Status registers.
- ❖ Supports extended register set.
- ❖ Supports the following MII test modes:
 - ◆ Loopback
 - ◆ Isolate
 - ◆ Collision test
 - ◆ Power down
- ❖ Supports the half-duplex or full-duplex mode.
- ❖ Supports 10 Mbps, 100 Mbps, and 1000 Mbps modes.
- ❖ Supports carrier sense and collision detection.

The RevMII does not support auto-negotiation and jabber detection.

4.11.1 Block Diagram

Figure 4-21 shows the block diagram of RevMII.

Figure 4-21 RevMII Block Diagram



4.11.2 Block Overview

The RevMII block consists of the following sub-blocks:

- ❖ RevMII MAC Interface
- ❖ RevMII Controller

4.11.2.1 RevMII MAC Interface

The RevMII MAC interface block connects two Ethernet MACs (MAC and Remote MAC). The RevMII MAC interface uses the following GMII/MII signals to connect two MACs:

- ❖ Transmit/Receive Data (TXD/RXD)
- ❖ Transmit/Receive Enable (TXEN/RXDV)
- ❖ Transmit/Receive Error (TXER/RXER)
- ❖ Carrier Sense (CRS)
- ❖ Collision Detection (COL)

The transmit and receive data paths (TXD/RXD) are 8-bit wide. If the MAC and remote MAC are working in the 10/100 Mbps mode, then the lower 4 bits of the data path contain valid data. The upper 4 bits are tied to zero. Similarly, if MAC and remote MAC are working in full-duplex mode, then the CRS and COL signals are set to zero unless you have selected the collision test mode.



- The Tx Bus (TXD, TXEN, and TXER) from the Remote MAC and the Rx Bus (RXD, RXDV, and RXER) to the Remote MAC are registered at the IOs to meet the timing constraints.
- The link between the MAC and remote MAC is up only when both operate at the same speed and in same mode. If the speed or mode are not same, then the link goes down. In addition, the link goes down if MAC or remote MAC is operating in any of test modes specified in [RevMII MAC Interface Modes](#).

4.11.2.1.1 RevMII MAC Interface Modes

The MAC interface supports the following data transfer modes:

❖ Normal Mode

In normal mode, the data MUXes are switched so that the Tx bus from the MAC goes to the Rx bus input of the Remote MAC and the Tx bus from the Remote MAC goes to the Rx bus input of the MAC. This is the default mode of operation. This mode is valid when no other mode is selected.

In half-duplex mode, when a MAC transmits the data to a remote MAC, the CRS signal becomes high on both sides. If MAC and remote MAC are in the half-duplex mode and both transmit at the same time, then there will be a collision. The COL signal is set to high to indicate a collision. When COL signal is high, the CRS signal also remains high.

❖ Power Down Mode

In the power down mode, the link between the MAC and remote MAC is shut down by switching the data multiplexers to port 2 so that the Rx buses of both MACs read all zeros. The RevMII controller remains active during this mode. The link status is set to 0 (bit[2] of the Status register in RevMII Controller block) during the power down mode. In addition, when you enable the power down mode, the CRS and COL signals are set to low.

You can select the power down mode by setting the bit[11] (Power down bit) to 1 in either of the RevMII Control registers.

❖ Isolate Mode

In RevMII, the isolate mode is similar to the power down mode in functionality. You can select the isolate mode by setting the bit[10] (Isolate bit) to 1 in either of the RevMII Control registers.

❖ Loopback Mode

In loopback mode, the link between the MAC and remote MAC is shut down. The MAC going to the loopback mode switches the Tx bus to its Rx bus and the other MAC stops receiving any data. The RevMII controller sets the link status to 0 (bit[2] of the Status register in RevMII Controller block) during this mode. When the TX_EN signal of a MAC is high, the CRS signal to that MAC is asserted high. The COL signal remains low throughout the process, unless you have selected the collision test mode.

You can select the loopback mode by setting the bit[14] (loopback mode bit) to 1 in either of the RevMII Control registers.

❖ Collision Test Mode

In collision test mode, the COL signal is asserted in response to the assertion of corresponding TX_EN signal. The COL signal is set to low after the TX_EN signal goes low. The link status is set to 0 (bit[2] of Status register in RevMII Controller block) to indicate that the link is down.

You can select the collision test mode by setting the bit[7] (Collision Test bit) to 1 in either of the RevMII Control registers.

4.11.2.2 RevMII Controller

The RevMII controller block handles the station management functionality and controls the RevMII MAC interface. The RevMII Controller block consists of the registers to control and monitor the RevMII bus. In addition, it provides the serial interface (SMI) to the remote MAC and a parallel interface to the MAC to access these registers. The RevMII controller block consists of the following components:

- ❖ Local MAC Controller
- ❖ Remote MAC Controller

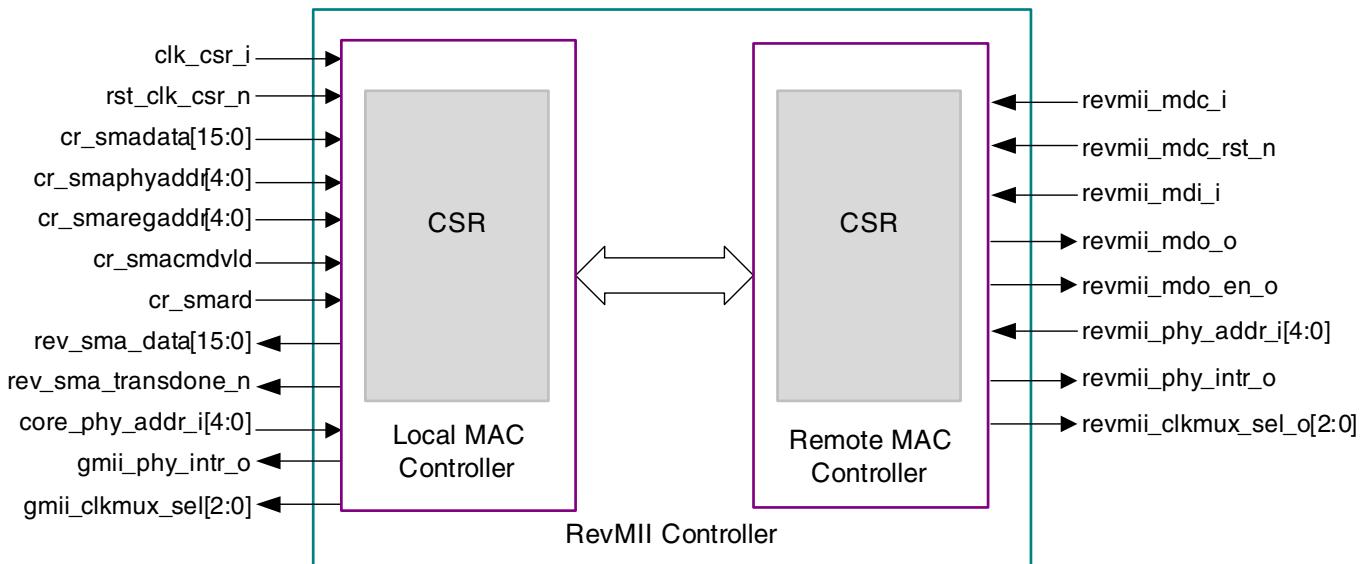
4.11.2.2.1 Local MAC Controller

The RevMII controller provides a parallel control interface with the MAC. The local MAC controller controls the dataflow between the MAC and RevMII MAC interface. It performs the following tasks:

- ❖ Verifies the PHY address and Register address
- ❖ Writes the data into the register
- ❖ Returns the register read-back data along with transaction completion signal.

Figure 4-22 shows the input and output signals of the local and remote MAC controller.

Figure 4-22 RevMII Controller



The CSR module inside the local MAC controller contains the registers described in “RevMII Registers” on page 179. The parallel data transfer occurs in the application clock domain (clk_csr_i). A valid transaction starts when the CSR module of the MAC asserts the cr_smacmdvld signal. Once the transaction is complete, the rev_sma_transdone_n signal goes low. In response, the cr_smacmdvld signal is reset to low, and the sma_transdone_n goes high, completing the handshaking process.

4.11.2.2.2 Remote MAC Controller

The RevMII controller provides a conventional serial interface (SMA Slave port) with the Remote MAC. The serial interface consists of serial management clock (MDC) and serial data input/output (MDIO) signals shown in [Figure 4-22](#). The remote MAC controller performs the following tasks:

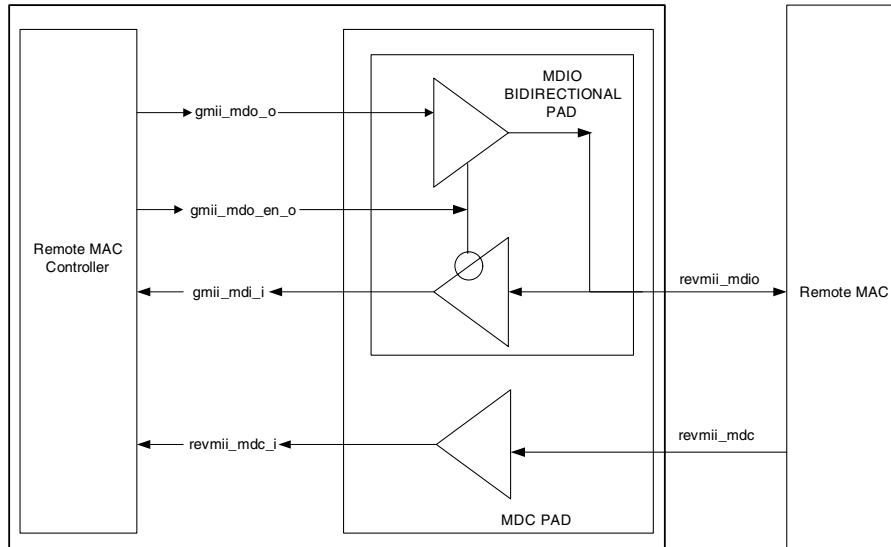
- ❖ Deserializes the input data.
- ❖ Checks the PHY and register address for validity.
- ❖ Writes data to the register.
- ❖ Serializes the register read-back data.

The CSR module inside the remote MAC controller contains the registers described in [“RevMII Registers”](#) on page 179. To connect the bidirectional MDIO signal to the remote MAC, you need to implement a three-state buffer outside the MAC as shown in [Figure 4-23](#).



Note When you select the SMA interface and the RevMII PHY interface during configuration, the MDIO interface signals from the SMA module and RevMII modules are multiplexed internally. This selection is controlled by the phy_intf_sel_i sampled at reset. For the RevMII mode, the phy_intf_sel_i signal has value 3'b111. For SMA mode, the phy_intf_sel_i signal can have any other value specified in “[phy_intf_sel_i\[2:0\]](#)” on page 214.

Figure 4-23 Remote MAC to RevMII MDIO Connectivity



If the remote MAC tries to perform a read transaction to a register in the extended register set that is not implemented by the RevMII, then the RevMII does not drive the MDIO line in response to the read transaction. This means that gmii_mdo_o and gmii_mdo_en_o signals remain low. Similarly, if the remote MAC tries to perform a write transaction to a register in the extended register set that is not implemented by the RevMII, then the RevMII ignores the write transaction.

If a MAC tries to perform a read transaction on a register that is not implemented, then the RevMII reads back a zero value. Similarly, if MAC tries to perform a write transaction, then the RevMII does not write anything and ignores the write transaction.

4.11.3 RevMII Interrupts

The RevMII controller contains two interrupt signals, one for MAC and one for remote MAC. These interrupt signals are asserted high when the link status changes, that is, link goes up or down. After receiving the interrupt, the software should clear the [Register 16 \(RevMII Interrupt Status and Mask Register\)](#) by reading this register. The Link Status Change interrupt is not generated if bit 0 ([LSIM: Link Status Change Interrupt Mask](#)) is set.

4.11.4 RevMII Registers

The CSR module of the local MAC controller and remote MAC controller contains the control, status, and interrupt registers for RevMII. This section provides the address maps of these registers and describes these registers.



Note The application indirectly accesses the RevMII registers by using [Register 4 \(GMII Address Register\)](#) and [Register 5 \(GMII Data Register\)](#).

4.11.4.1 RevMII Register Maps

[Table 4-24](#) provides the address map and high-level summary of the RevMII registers for MAC.

Table 4-24 RevMII Register Map - MAC

Register Number	Address Offset	Register Name and Description
0	5'b00000	Register 0 (RevMII PHY Control Register) Controls the MAC side of the RevMII Controller.
1	5'b00001	Register 1 (RevMII Common Status Register) Shows the status of the RevMII. This is a shared register for both MACs.
2-14	5'b00010-5'b01110	Reserved
15	5'b01111	Register 15 (RevMII Common Extended Status Register) Shows the status of the RevMII supporting 1000 Mbps speed. This is a common register for both MACs.
16	5'b10000	Register 16 (RevMII Interrupt Status and Mask Register) Shows the status of the interrupt generated for the MAC and also provides interrupt masking signal for the generated interrupts.
17	5'b10001	Register 17 (RevMII Remote PHY Status Register) Shows the status of speed and duplex-mode programmed in the remote PHY Control register.
18-31	5'b10010-5'b11111	Reserved

[Table 4-25](#) provides the address map and high-level summary of the RevMII registers of the remote MAC.

Table 4-25 RevMII Register Map - Remote MAC

Register Number	Address Offset	Register Name and Description
0	5'b00000	Register 0 (RevMII Remote PHY Control Register) Controls the remote side of RevMII Controller. This register is similar to Register 0 (RevMII PHY Control Register) .
1	5'b00001	Register 1 (RevMII Common Status Register) Shows the status of the RevMII. This is a common register for both MACs.
2-14	5'b00010-5'b01110	Reserved
15	5'b01111	Register 15 (RevMII Common Extended Status Register) Shows the status of the RevMII supporting 1000 Mbps speed. This is a common register for both MACs.
16	5'b10000	Register 16 (RevMII Remote PHY Interrupt and Mask Register) Shows the status of the interrupt generated for remote MAC and also provides interrupt masking signal for the generated interrupts. This register is similar to Register 16 (RevMII Interrupt Status and Mask Register) .
17	5'b10001	Register 17 (RevMII PHY Status Register) Shows the status of speed and duplex-mode programmed in the RevMII PHY Control register.
18-31	5'b10010-5'b11111	Reserved

4.11.4.2 Register 0 (RevMII PHY Control Register)

This register controls the RevMII operations for the MAC and enables the RevMII modes.

Table 4-26 Register 0 (RevMII PHY Control Register)

Field	Description	Reset	Access
15	REVRST: Reset When set, this bit configures the PHY Control register to its default values. This bit is cleared after the reset operation is complete.	0	R_W_SC
14	REVLPBCK: Loopback When set, this bit enables the loopback mode. When reset, this bit disables the loopback mode.	0	R_W

Table 4-26 Register 0 (RevMII PHY Control Register)

Field	Description			Reset	Access
13	REVSSL: Speed Selection (LSB) This bit along with bit 6 (MSB) indicates the link speed as described in the following table.			0	R_W
	Bit 6 Bit 13 Speed				
	1 1 Reserved				
	1 0 1000 Mbps				
	0 1 100 Mbps				
	0 0 10 Mbps				
	When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 1.				
12	REvanEN: Auto-Negotiation Enable This bit is not used in RevMII.			0	RO
11	REVPWRDN: Power Down When set, this bit enables the power down mode. When reset, this bit disables the power down mode. For more information, see Power Down Mode .			0	R_W
10	REVISOL: Isolate When set, this bit enables the isolate mode. When reset, this bit disables the isolate mode. For more information, see Isolate Mode .			0	R_W
9	REVREAN: Restart Auto-Negotiation This bit is not used in RevMII.			0	RO
8	REVDM: Duplex Mode When set, this bit configures the RevMII PHY for full-duplex operation. When reset, this bit configures the RevMII PHY for half-duplex operation. When you select the FDUPLEX_ONLY parameter in coreConsultant, the reset value of this bit is 1.			0	R_W
7	REVCOLTST: Collision Test When set, this bit enables the collision test mode. When reset, this bit disables the collision test mode. For more information, see Collision Test Mode .			0	R_W
6	REVSSH: Speed Selection (MSB) When set, this bit along with bit 6 (LSB) indicates the link speed. For more information, see REVSSL: Speed Selection (LSB) . When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 0.			1	R_W
5:0	Reserved			0	RO

4.11.4.3 Register 1 (RevMII Common Status Register)

This register provides the RevMII status. This register is common for MAC and remote MAC.

Table 4-27 Register 1 (RevMII Common Status Register)

Field	Description	Reset	Access
15	100T4: 100BASE-T4 When set, this bit indicates that the RevMII PHY can perform the link transmission and reception using the 100BASE-T4 signaling specification.	1	RO
14	100XFD: 100BASE-X Full Duplex When set, this bit indicates that the RevMII PHY can perform the full-duplex link transmission and reception using the 100BASE-X signaling specification.	1	RO
13	100XHD: 100BASE-X Half Duplex When set, this bit indicates that the RevMII PHY can perform half-duplex link transmission and reception using the 100BASE-X signaling specification. When you select the FDUPLEX_ONLY parameter in coreConsultant, the reset value of this bit is 0.	1	RO
12	10FD: 10 Mbps Full Duplex When set, this bit indicates that the RevMII PHY can perform the full-duplex link transmission and reception while operating in 10 Mbps mode.	1	RO
11	10HD: 10 Mbps Half Duplex When set, this bit indicates that the RevMII PHY can perform half-duplex link transmission and reception while operating in 10 Mbps mode. When you select the FDUPLEX_ONLY parameter in coreConsultant, the reset value of this bit is 0.	1	RO
10	100T2FD: 100BASE-T2 Full Duplex When set, this bit indicates that the RevMII PHY can perform full-duplex link transmission and reception using the 100BASE-T2 signaling specification.	1	RO
9	100T2HD: 100BASE-T2 Half Duplex When set, this bit indicates that the RevMII PHY can perform half-duplex link transmission and reception using the 100BASE-T2 signaling specification. When you select the FDUPLEX_ONLY parameter in coreConsultant, the reset value of this bit is 0.	1	RO
8	EXTSTS: Extended Status When set, this bit indicates that the base register status information is extended into Register 15 (RevMII Common Extended Status Register) . This bit should be set for 1000 Mbps mode. When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 0.	1	RO
7	Reserved		RO
6	PRESUP: MF Preamble Suppression This bit is not used in RevMII.	0	RO

Table 4-27 Register 1 (RevMII Common Status Register)

Field	Description	Reset	Access
5	ANC: Auto-Negotiation Complete This bit is not used in RevMII.	0	RO
4	RMTFLT: Remote Fault This bit is not used in RevMII.	0	RO
3	ANA: Auto-Negotiation Ability This bit is not used in RevMII.	0	RO
2	LNKSTS: Link Status When set, this bit indicates that a valid link has been established. When reset, this bit indicates that the link is not valid.	0	R_SS_SC_LLO
1	JABDET: Jabber Detect This bit is not used in RevMII.	0	RO
0	EXTCAP: Extended Capability This bit is always set as RevMII supports extended register capability.	1	RO

4.11.4.4 Register 15 (RevMII Common Extended Status Register)

This register is common for MAC and remote MAC. This register is implemented for RevMII supporting 1000 Mbps speed. It is not present when the MAC supports only 10/100Mbps operations.

Table 4-28 Register 15 (RevMII Common Extended Status Register)

Field	Description	Reset	Access
15	1000XFD: 1000BASE-X Full Duplex When set, this bit indicates that the RevMII PHY can perform the full-duplex link transmission and reception using the 1000BASE-X signaling specification. When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 0.	1	RO
14	1000XHD: 1000BASE-X Half Duplex When set, this bit indicates that the RevMII PHY can perform the half-duplex link transmission and reception using the 1000BASE-X signaling specification. When you select either FDUPLEX_ONLY parameter or 10/100 Mbps Operating Mode (OP_MODE =1) in coreConsultant, the reset value of this bit is 0.	1	RO
13	1000TFD: 1000BASE-T Full Duplex When set, this bit indicates that the RevMII PHY can perform the full-duplex link transmission and reception using the 1000BASE-T signaling specification. When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 0.	1	RO

Table 4-28 Register 15 (RevMII Common Extended Status Register)

Field	Description	Reset	Access
12	1000THD: 1000BASE-T Half Duplex When set, this bit indicates that the RevMII PHY can perform the half-duplex link transmission and reception using the 1000BASE-T signaling specification. When you select either FDUPLEX_ONLY parameter or 10/100 Mbps Operating Mode (OP_MODE =1) in coreConsultant, the reset value of this bit is 0.	1	RO
11:0	Reserved		

4.11.4.5 Register 16 (RevMII Interrupt Status and Mask Register)

This register provides the status of the interrupts and enable you to mask the interrupt signal. The status bits are cleared when this register is read.

Table 4-29 Register 16 (RevMII Interrupt Status and Mask Register)

Field	Description	Reset	Access
15:9	Reserved		
8	LSI: Link Status Change Interrupt When set, this bit indicates that the link status has changed.	0	R_SS_RC
7:1	Reserved		
0	LSIM: Link Status Change Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of bit 8 (LSI: Link Status Change Interrupt).	0	R_W

4.11.4.6 Register 17 (RevMII Remote PHY Status Register)

This register shows the status of speed and duplex-mode programmed in the remote PHY Control register. You can use this register for MAC to MAC handshake when the link between the MAC and remote MAC is down because of the speed or duplex-mode mismatch.

Table 4-30 Register 17 (RevMII Remote PHY Status Register)

Field	Description	Reset	Access
15:3	Reserved		
2	RMACDM: Remote MAC Duplex mode When set, this bit indicates the duplex mode configured in the bit 8 of Register 0 (RevMII Remote PHY Control Register). When you select the FDUPLEX_ONLY parameter in coreConsultant, the reset value of this bit is 1.	0	RO
1	RMACSSH: Remote MAC Speed Select MSB When set, this bit indicates the link speed specified in bit 6 of Register 0 (RevMII Remote PHY Control Register). When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 0.	1	RO

Table 4-30 Register 17 (RevMII Remote PHY Status Register)

Field	Description	Reset	Access
0	RMACSSL: Remote MAC Speed Select LSB When set, this bit indicates the link speed specified in bit 13 of Register 0 (RevMII Remote PHY Control Register). When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 1.	0	RO

Table 4-31 describes the RevMII PHY Status Register of a remote MAC.

Table 4-31 Register 17 (RevMII PHY Status Register)

Field	Description	Reset	Access
15:3	Reserved		
2	MACDM: MAC Duplex mode When set, this bit indicates the duplex mode configured in the bit 8 of Register 0 (RevMII PHY Control Register) . When you select the FDUPLEX_ONLY parameter in coreConsultant, the reset value of this bit is 1.	0	RO
1	MACSSH: MAC Speed Select MSB When set, this bit indicates the link speed specified in bit 6 of Register 0 Register 0 (RevMII PHY Control Register) . When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 0.	1	RO
0	MACSSL: MAC Speed Select LSB When set, this bit indicates the link speed specified in bit 13 of Register 0 (RevMII PHY Control Register) . When you select the Operating Mode as 10/100 Mbps (OP_MODE =1) in coreConsultant, the reset value of this bit is 1.	0	RO

4.12 Serial Media Independent Interface

The SMII block is designed to fully comply with the SMII specification, revision 2.1, from Cisco. The SMII has the following characteristics:

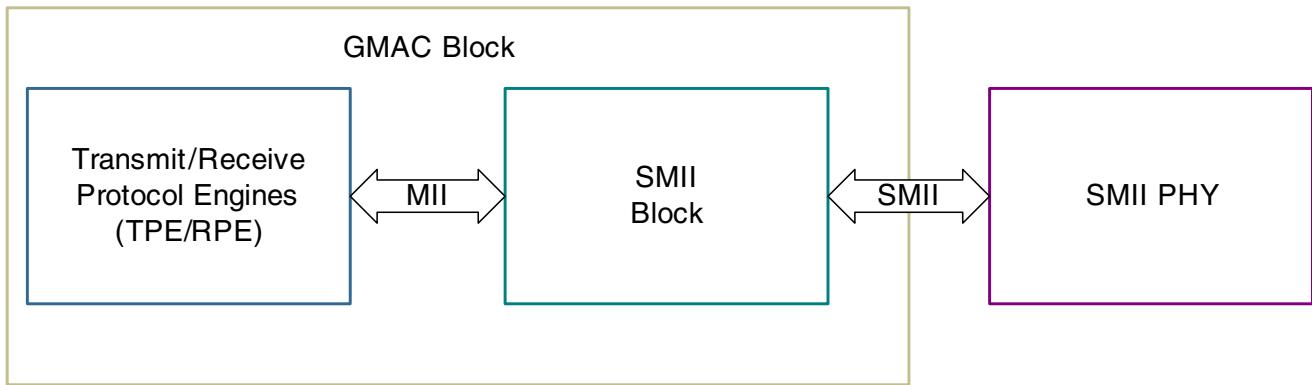
- ❖ Conveys complete MII information between a 10/100 PHY and MAC with two pins (TX and RX) per port, one global 125 MHz System clock (CLOCK) and one global Synchronization signal (SYNC).
For information about clock connections when the GMAC operates with an SMII, see “[Clocks With SMII](#)” on page 436.
- ❖ Allows a multi-port MAC/PHY communication with one system clock.
- ❖ Allows per packet switching between 10 MBit and 100 MBit data rates.
- ❖ Allows direct MAC to MAC communication.
- ❖ Operates in half-duplex and full-duplex modes.
- ❖ Provides optional source synchronous mode in which four signals TX_CLK, TX_SYNC, RX_CLK and RX_SYNC replace SYNC signal.

- ❖ Supports optional selection of TX_SYNC as input in source synchronous mode to synchronize the transmit data.
- ❖ Transmits status in between frames, when enabled. This is useful in MAC to MAC connection.

4.12.1 Block Diagram

Figure 4-24 shows the position of the SMII block relative to the GMAC and SMII PHY. The SMII block is placed between the DWC Ether MAC 10/100/1000 Universal's GMII and the PHY to translate the GMII signals to SMII signals. The pinout is shown in Figure 4-25.

Figure 4-24 SMII Block Diagram



4.12.2 Block Overview

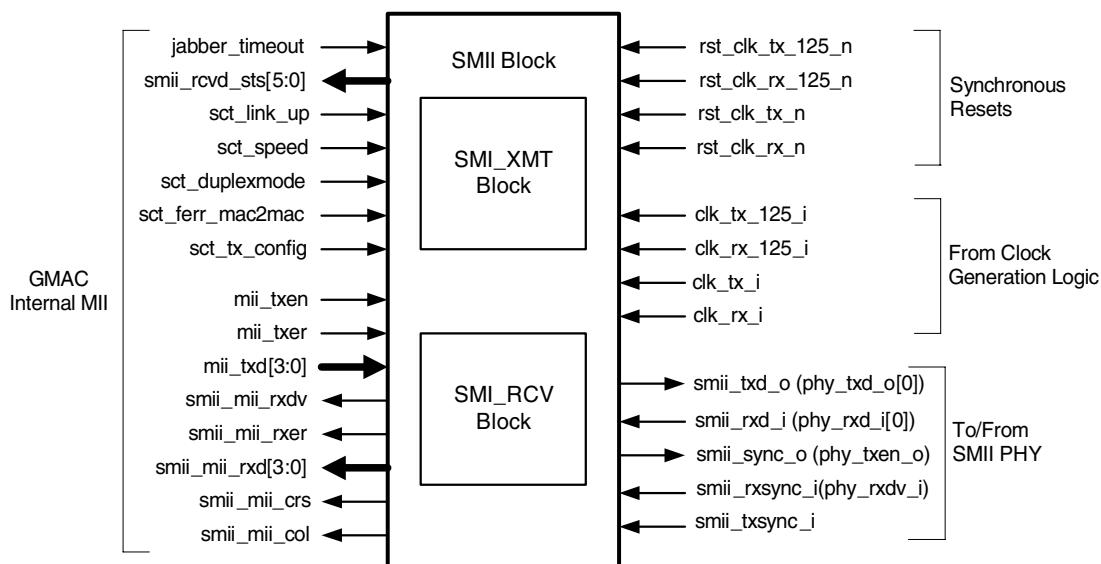
The SMII sub-blocks perform the signal translation.

4.12.2.1 GMII-SMII Transmit Block

The GMII-SMII Transmit (SMI_XMT) block converts the two data nibbles and control signals received (at 2.5 MHz and 25 MHz for 10 Mbps and 100 Mbps respectively) from MII transmit interface into one ten bit segment transmitted serially at 125 MHz on the SMII transmit interface. The single ten bit segment is repeated 10 times in 10 Mbps mode. This block generates the SYNC signal (TX_SYNC signal in source synchronous mode) once every 10 clocks of 125 MHz indicating the beginning of a ten-bit segment. In source synchronous mode with TX_SYNC as input, it synchronizes the transmit data to the received TX_SYNC.

4.12.2.2 GMII-SMII Receive Block

The GMII-SMII Receive (SMI_RCV) block converts the ten-bit segment received serially at 125 MHz from SMII receive interface into two data nibbles and control signals (at 2.5 MHz and 25 MHz for 10 Mbps and 100 Mbps respectively) on the MII receive interface. In 10 Mbps mode, although each ten bit segment is received ten times, only one sample is passed to the MII receive interface.

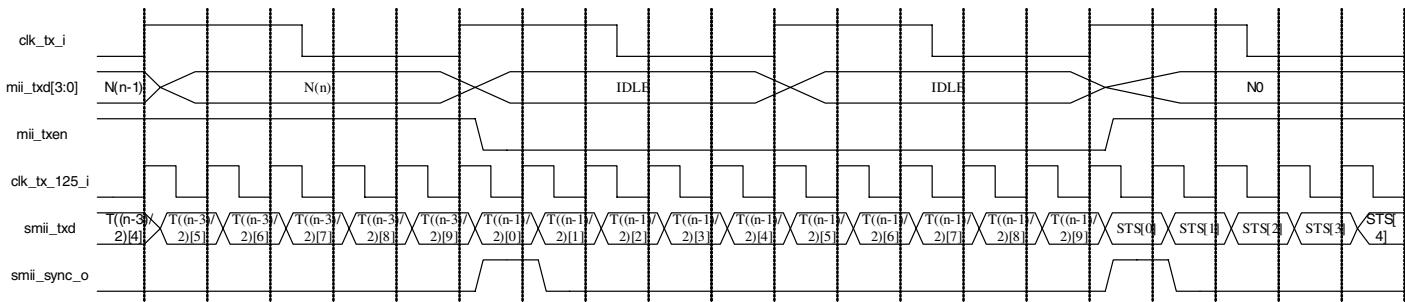
Figure 4-25 SMII Pinout

4.12.3 SMII Timing



Note “T” and “N” in the timing diagrams indicate ten bit segment and nibble, respectively. The relation between the T(m) and N(n) number is given by $m = (n-1)/2$ for a frame, that is, when there are 128 nibbles (minimum sized frame) to be transmitted, then N is from N(0) to N(127). Correspondingly, the T is from T(0) to T(63) in the waveform in [Figure 4-26](#).

The SMII transmit (SMI_XMT) block converts the two data nibbles and control signals received at 25 MHz from 100 Mbps MII transmit interface into one ten bit segment transmitted serially at 125 MHz clock on SMII transmit interface. The SMII transmit block sends ten bit status segments in the interpacket gap as defined in the SMII specification. This status is sent only if enabled by setting bit 24 (Transmit Configuration bit) of the MAC Configuration Register 0. This feature is useful for sending status in MAC to MAC connection only. The status includes the SFTERR, FES, DM and LUD bits from MAC Configuration Register 0 and Jabber timeout error signal from the TPE block.

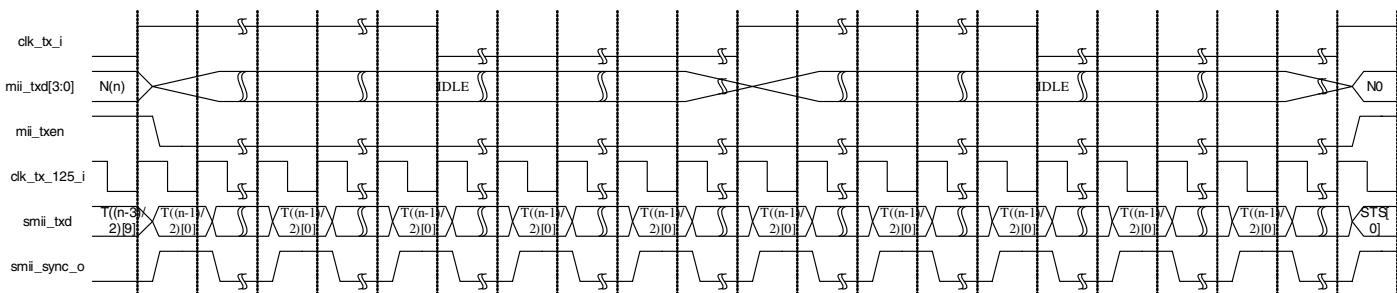
Figure 4-26 SMII transmit in 100 Mbps Mode

The SMII transmit (SMI_XMT) block converts the two data nibbles and control signals received at 2.5 MHz from 10 Mbps MII transmit interface into one ten bit segment transmitted serially at 125 MHz clock on SMII transmit interface. The single ten bit segment is repeated 10 times in 10 Mbps mode. The SMII transmit

block sends ten bit status segments in the interpacket gap as defined in the SMII specification. This status is sent only if enabled by setting bit 24 (Transmit Configuration bit) of the MAC Configuration Register 0. This feature is useful for sending status in MAC to MAC connection only. The status includes the SFTERR, FES, DM and LUD bits from MAC Configuration Register 0 and Jabber timeout error signal from the TPE block.

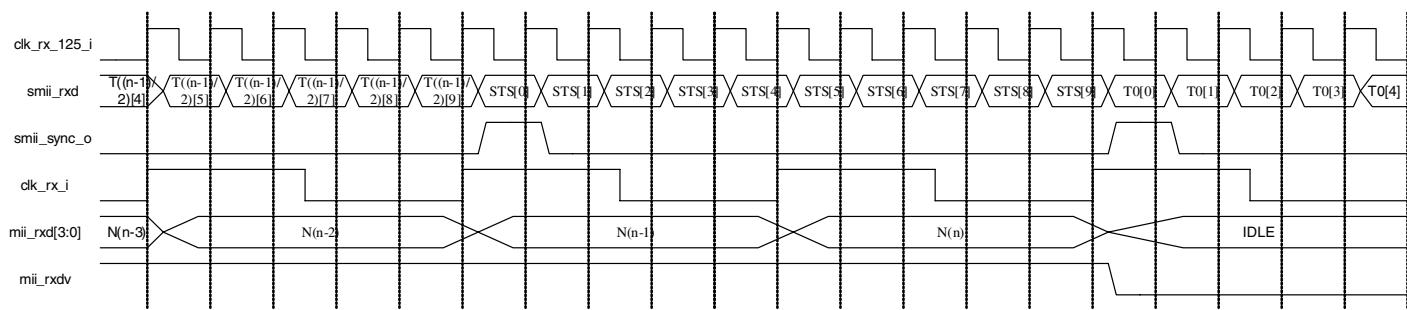
The SMI_XMT block generates the SYNC signal (TX_SYNC signal in source synchronous mode) once every 10 clocks of 125 MHz indicating the beginning of a ten bit segment. In source synchronous mode with TX_SYNC as input, it synchronizes the transmit data to the received TX_SYNC.

Figure 4-27 SMII Transmit Block in 10 Mbps Mode

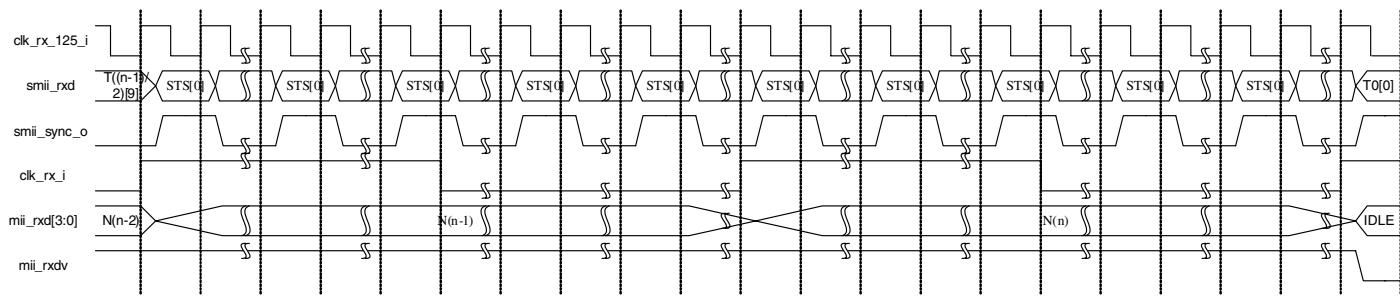


The SMII receive (SMI_RCV) block converts the ten-bit segment received serially at 125 MHz on SMII receive interface into two data nibbles and control signals at 25 MHz in 100 Mbps mode on the MII receive interface. The data nibble given to MII receive interface is delayed by one cycle so that receive error (if indicated in the ten bit status segment after the frame) can be signaled to the MAC with the last nibble.

Figure 4-28 SMII Receive Block in 100 Mbps Mode



The SMII receive (SMI_RCV) block converts the ten bit segment received serially at 125 MHz on SMII receive interface into two data nibbles and control signals at 2.5 MHz in 10 Mbps mode on the MII receive interface. In 10 Mbps mode, although each ten bit segment is received ten times, only one sample is passed to the MII receive interface. The data nibble given to MII receive interface is delayed by one cycle so that receive error (if indicated in the ten bit status segment after the frame) can be signaled to the MAC with the last nibble.

Figure 4-29 SMII Receive Block in 10 Mbps Mode

The SMII uses the RGMII interrupt mechanism as SGMII/RGMII Status Register 54 is reused by the SMII block.

When the TXSYNC signal is selected as input in Source Synchronous Mode, the smi_xmt block uses it to synchronize the ten bit segments being transmitted on the SMII transmit interface.

4.13 Reduced Gigabit Media Independent Interface

The Reduced Gigabit Media Independent Interface (RGMII) specification reduces the pin count of the interconnection between the DWC Ether MAC 10/100/1000 Universal and the PHY for GMII and MII interfaces. To achieve this, the data path and control signals are reduced and multiplexed together with both the edges of the transmit and receive clocks. For gigabit operation the clocks operate at 125 MHz; for 10/100 operation, the clock rates are 2.5 MHz/25 MHz.

In the DWC Ether MAC 10/100/1000 Universal core, the optional RGMII module is instantiated between the GMAC core's GMII and the PHY to translate the control and data signals between the GMII and RGMII protocols.

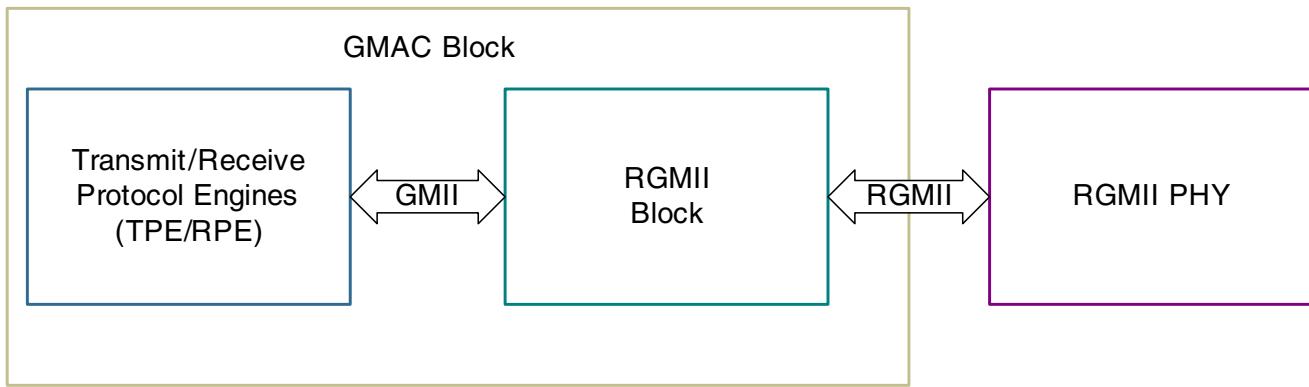
The RGMII block has the following characteristics:

- ❖ Supports 10-Mbps, 100-Mbps, and 1000-Mbps operation rates.
- ❖ Requires no extra clock because both edges of the incoming clocks are used. To simplify back-end implementation of the DWC Ether MAC 10/100/1000 Universal, there are separate clock inputs (180 degrees out-of-phase with the associated transmit/receive clocks) for logic that uses the falling edges.
- For guidelines for clock connections when the GMAC operates with an RGMII, see “[Clocks With RGMII](#)” on page 438.
- ❖ Provides a synthesis option for the DWC Ether MAC 10/100/1000 Universal to inherently delay the RGMII transmit clock with respect to the data and control signals, as described in Version 2.0 of RGMII specification.
- ❖ Extracts the in-band (link speed, link status, and duplex mode) status signals from the PHY and provides them to the GMAC core logic for link detection.

You can include the RGMII module in the GMAC core controller by selecting the RGMII interface in the coreConsultant during configuration.

4.13.1 Block Diagram

[Figure 4-30](#) shows the position of the RGMII block relative to the GMAC and RGMII PHY. The RGMII block is placed between the DWC Ether MAC 10/100/1000 Universal's GMII and the PHY to translate the GMII signals to RGMII signals.

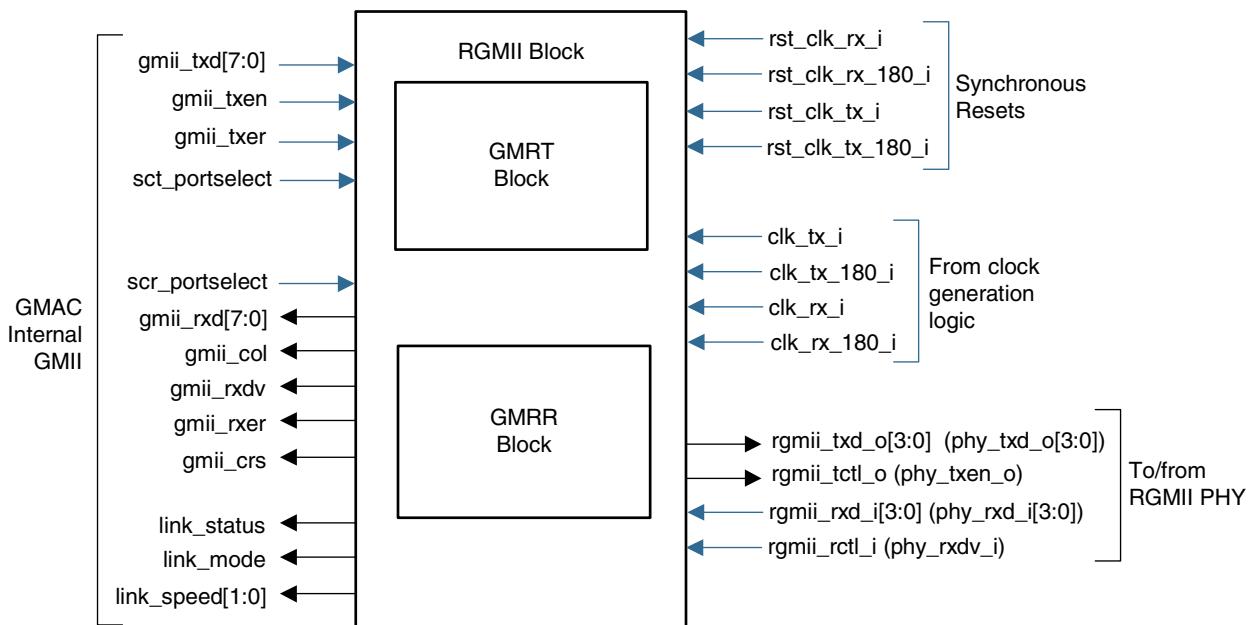
Figure 4-30 RGMII Block Diagram

4.13.2 Block Overview

The RGMII sub-blocks perform the signal translation:

GMII-RGMII Transmit (GMRT) Block: Translates all GMII transmit signals to RGMII signals. The GMRT registers all GMII input signals at the rising edge of `clk_tx_i` and generates all RGMII transmit signals at the rising edge of either `clk_tx_i` or `clk_tx_180_i`.

GMII-RGMII Receive (GMRR) Block: Translates all RGMII receive signals to GMII receive signals. The GMRR registers all RGMII input signals at the rising edge of either `clk_rx_i` or `clk_rx_180_i` and generates all GMII receive signals at the rising edge of `clk_rx_i`.

Figure 4-31 RGMII Block Pinout Diagram

4.13.3 RGMII Clocks

The RGMII operation uses both edges of the RGMII transmit and receive clocks (`clk_tx_i` and `clk_rx_i`), which run at 125 MHz for gigabit operation or 2.5 MHz/25 MHz for 10/100 operation. These clocks come from a separate PLL logic block. To simplify implementation of the DWC Ether MAC 10/100/1000 Universal, the RGMII block provides two additional inputs to drive the logic that would use the falling edges of `clk_tx_i` and `clk_rx_i`:

- ❖ `clk_tx_180_i` must be 180 degrees out-of-phase with respect to `clk_tx_i`
- ❖ `clk_rx_180_i` must be 180 degrees out-of-phase with respect to `clk_rx_i`

Separate synchronous inputs are present to synchronously reset the logic driven by each of the four RGMII clocks.

4.13.4 Signal Conversion

4.13.4.1 Transmit Data Conversion

As defined in the RGMII specification, multiplexing of data and control in gigabit mode is accomplished by using both edges of the transmit clock. The RGMII block accepts the 8-bit GMII transmit data from the DWC Ether MAC 10/100/1000 Universal GMII (`gmii_txd[7:0]`) on the rising edge of `clk_tx_i`. For gigabit operation, the RGMII block then converts `gmii_txd[7:0]` to two 4-bit nibbles and transmits the data on the RGMII transmit data port (`rgmii_txd_o[3:0]`) as follows:

- ❖ At the rising edge of `clk_tx_i`, `rgmii_txd_o[3:0]` contains `gmii_txd[3:0]`
- ❖ At the falling edge of `clk_tx_i` (rising edge of `clk_tx_180_i`), `rgmii_txd_o[3:0]` contains `gmii_txd[7:4]`

For 10/100 mode, the transmit data to the PHY is 4 bits. So, for 10/100 mode, the RGMII block drives the `gmii_txd[3:0]` data to the PHY on `rgmii_txd_o[3:0]` at the rising edge of `clk_tx_i`.

4.13.4.2 Transmit Control Signal Conversion

The RGMII block accepts the GMII transmit control signals (`gmii_txen` and `gmii_txer`) from the DWC Ether MAC 10/100/1000 Universal GMII on the rising edge of `clk_tx_i`, then multiplexes `gmii_txen` and `gmii_txer` onto the `rgmii_tctl_o` output as required by the RGMII specification:

- ❖ At the rising edge of `clk_tx_i`, `rgmii_tctl_o` is `gmii_txen`.
- ❖ At the falling edge of `clk_tx_i` (rising edge of `clk_tx_180_i`), `rgmii_tctl_o` is the logical XOR of `gmii_txen` and `gmii_txer`.

Table 4-32 shows the mapping of the `PLS_DATA.request` parameters from the DWC Ether MAC 10/100/1000 Universal to the corresponding signaling on the GMII and RGMII control and data signals.

Table 4-32 Encoding of gmii_txen, gmii_txer, rgmii_tctl_o, and rgmii_txd_o

gmii_txen	gmii_txer	gmii_txd ^a	rgmii_tctl_o ^b	Description	PLS_DATA.request parameter
0	0	0x00–0xFF	0,0	Normal inter-frame	TRANSMIT_COMPLETE
0	1	0x00–0x0E	0,1	Reserved	
0	1	0x0F	0,1	Carrier extend	EXTEND (8 bits)
0	1	0x10–0x1E	0,1	Reserved	

Table 4-32 Encoding of gmii_txen, gmii_txer, rgmii_tctl_o, and rgmii_txd_o

gmii_txen	gmii_txer	gmii_txd^a	rgmii_tctl_o^b	Description	PLS_DATA.request parameter
0	1	0x1F	0,1	Carrier extend error	EXTEND_ERROR (8 bits)
0	1	0x20–0xFF	0,1	Reserved	
1	0	0x00–0xFF	1,1	Normal data transmission	ZERO, ONE (8 bits)
1	1	0x00–0xFF	1,0	Transmit error propagation	No applicable parameter

a. When the RGMII interface is configured to transmit the configuration during the IFG, then rgmii_txd[3:0] reflects the Duplex Mode, Port Select, Speed (encoded as 00 for 10 Mbps, 01 for 100 Mbps and 10 for 1000 Mbps), and Link Up/Down bits of the MAC Configuration Register, during this period.

b. Values at rising, falling edges of clk_tx_i.

4.13.4.3 Receive Data Conversion

For Gigabit mode, the RGMII block registers the 4-bit data on rgmii_rxd_i[3:0] at both edges of the receive clock (clk_rx_i) and transfers the resulting 8-bit data to the DWC Ether MAC 10/100/1000 Universal GMII on the rising edge of clk_rx_i, as follows:

- ❖ gmii_rxd[3:0] is the value received on rgmii_rxd_i[3:0] at the rising edge of clk_rx_i
- ❖ gmii_rxd[7:4] is the value received on rgmii_rxd_i[3:0] at the falling edge of clk_rx_i (rising edge of clk_rx_180_i)

For 10/100 mode, the RGMII block registers the 4-bit data on rgmii_rxd_i[3:0] only at the rising edge of clk_rx_i and transfers the data to the DWC Ether MAC 10/100/1000 Universal GMII on the rising edge of clk_rx_i, as follows:

- ❖ gmii_rxd[3:0] is the value received on rgmii_rxd_i[3:0] at the rising edge of clk_rx_i.
- ❖ gmii_rxd[7:4] is 0x0.

4.13.4.4 Receive Control Signal Conversion

The RGMII block samples the rgmii_rctl_i signal from the PHY at both edges of clk_rx_i and drives the resulting GMII receive control signals to the DWC Ether MAC 10/100/1000 Universal GMII as follows:

- ❖ gmii_rxdv is the value received on rgmii_rctl_i at the rising edge of clk_rx_i.
- ❖ gmii_rxer is the logical XOR of the following two signals:
 - ◆ The value received on rgmii_rctl_i at the rising edge of clk_rx_i (gmii_rxdv).
 - ◆ The value received on rgmii_rctl_i at the falling edge of clk_rx_i (rising edge of clk_rx_180_i).

4.13.4.5 Receive Status Signal Conversion

To generate the GMII link status signals, the RGMII block decodes the rgmii_rxd_i[3:0] value sampled at the rising edge of clk_rx_i when the value of rgmii_rctl_i is 0 for both the rising and falling edges of clk_rx_i (normal inter-frame value). The decoded status signals are:

- ❖ link_status is the value of rgmii_rxd_i[0].
- ❖ link_speed is the value of rgmii_rxd_i[2:1].

- ❖ link_mode is the value of rgmii_rxd_i[3].

[Table 4-33](#) shows the mapping of the PLS_DATA.indicate parameters from the DWC Ether MAC 10/100/1000 Universal to the corresponding signalling on the GMII and RGMII control and data signals. In addition, the RGMII block asserts gmii_crs to the DWC Ether MAC 10/100/1000 Universal GMII when any of the following conditions is true:

- ❖ The gmii_rxdv is true.
- ❖ The gmii_rxdv is false, gmii_rxer is true, and the value of gmii_rxd is 0xFF.
- ❖ A carrier extend, carrier extend error, or false carrier occurs in gigabit mode (see [Table 4-33](#)).
- ❖ A false carrier occurs in 10/100 mode.
- ❖ The RGMII is transmitting data, carrier extend, or carrier extend error on the RGMII transmit outputs.

The RGMII block asserts gmii_col to the GMAC GMII when both of the following conditions is true:

- ❖ The RGMII is transmitting data, carrier extend, or carrier extend error on the RGMII transmit outputs.
- ❖ The RGMII is asserting either gmii_crs or gmii_rxdv.

Table 4-33 Decoding of rgmii_rctl_i and rgmii_rxd_i

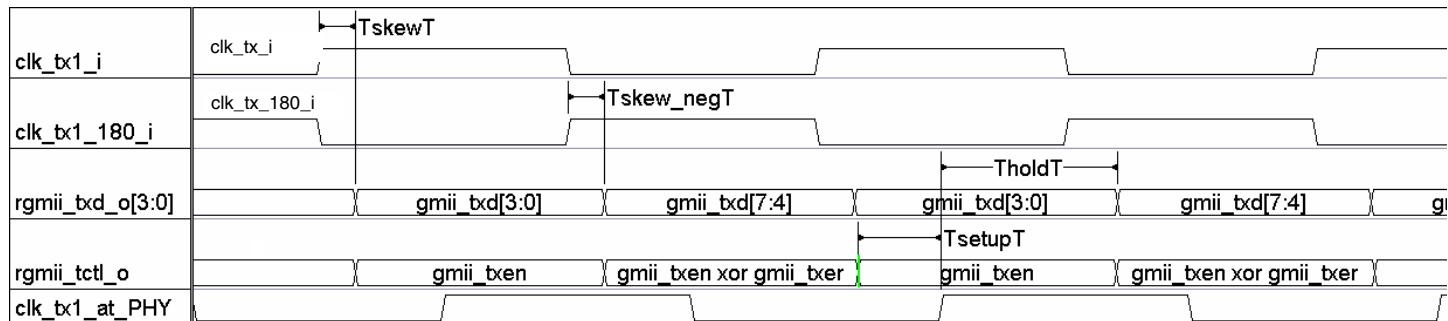
rgmii_rctl_i ^a	gmii_rxd[7:0] ([3:0] in 10/100 mode)	gmii_rxdv	gmii_rxer	Description	PLS_DATA.indicate or PHY_status parameter
0,0	XXXXXXXX0 or XXXXXXXX1	0	0	Normal inter-frame	Indicates link status: <ul style="list-style-type: none"> • 0: down • 1: up
0,0	XXXXX00X or XXXXX01X or XXXXX10X or XXXXX11X	0	0	Normal inter-frame	Indicates receive clock frequency: <ul style="list-style-type: none"> • 00: 2.5 MHz • 01: 25 MHz • 10: 125 MHz • 11: Reserved
0,0	XXXX1XXX or XXXX0XXX	0	0	Normal inter-frame	Indicates duplex status: <ul style="list-style-type: none"> • 0: half-duplex • 1: full-duplex
0,1	0x0E (or 0xE in 10/100 mode)	0	1	False carrier indication	False carrier present
0,1	0x0F	0	1	Carrier extend	EXTEND (8 bits)
0,1	0x1F	0	1	Carrier extend error	ZERO, ONE (8 bits)
0,1	0xFF (or 0xF in 10/100 mode)	0	1	Carrier sense	PLS_Carrier.Indicate
1,1	0x00–0xFF	1	0	Normal data reception	ZERO, ONE (8 bits)
1,0	0x00–0xFF	1	1	Data reception error	ZERO, ONE (8 bits)

- a. Values at rising, falling edges of clk_rx_i.

4.13.5 RGMII Transmit Timing

[Figure 4-32](#) shows the timing for RGMII transmission. The RGMII specification defines a transmit skew (T_{skewT}) of ± 0.5 ns. As shown in [Figure 4-32](#), the trace delay on the PC board for clk_{tx_i} going to a receiving PHY must satisfy the set-up and hold time requirements of the RGMII transmit signals at the receiving PHY.

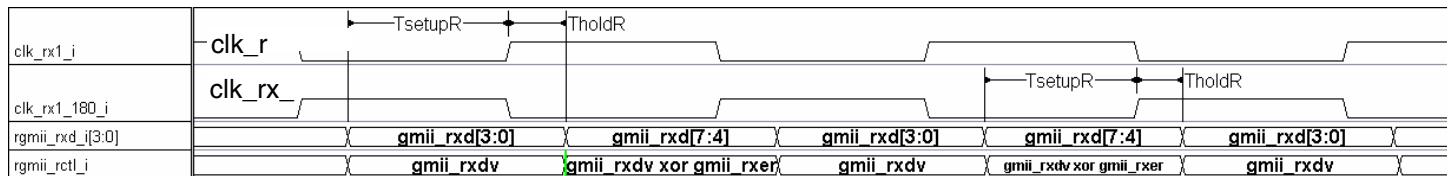
Figure 4-32 RGMII Transmit Timing



4.13.6 RGMII Receive Timing

[Figure 4-33](#) shows the receive timing for RGMII. The T_{setupR} and $TholdR$ values are required to be a minimum of 1 ns.

Figure 4-33 RGMII Receive Timing



4.14 Reduced Ten-Bit Interface

The Reduced Ten-Bit Interface (RTBI) specification reduces the pin count of the interconnection between the GMAC-UNIV and the PHY for TBI interface. To achieve this, the transmit and receive code-groups are multiplexed together with both the edges of the transmit and receive clocks.

In the GMAC-UNIV core, the optional RTBI module is instantiated between the GMAC's PCS module and the PHY to translate the code-group signals between the TBI and RTBI protocols. As the GMAC-UNIV's TBI supports only 1000BASE-X, the RTBI also supports only 1000BASE-X and is not valid for 10/100 Mbps operation.

The RTBI block has the following characteristics:

- ❖ For the RTBI block, no extra clock is required because both the edges of the incoming clocks are used. To simplify back-end implementation of the GMAC-UNIV, there are separate clock inputs (180 degrees out-of-phase with the associated transmit/receive clocks) for logic that uses the falling edges.

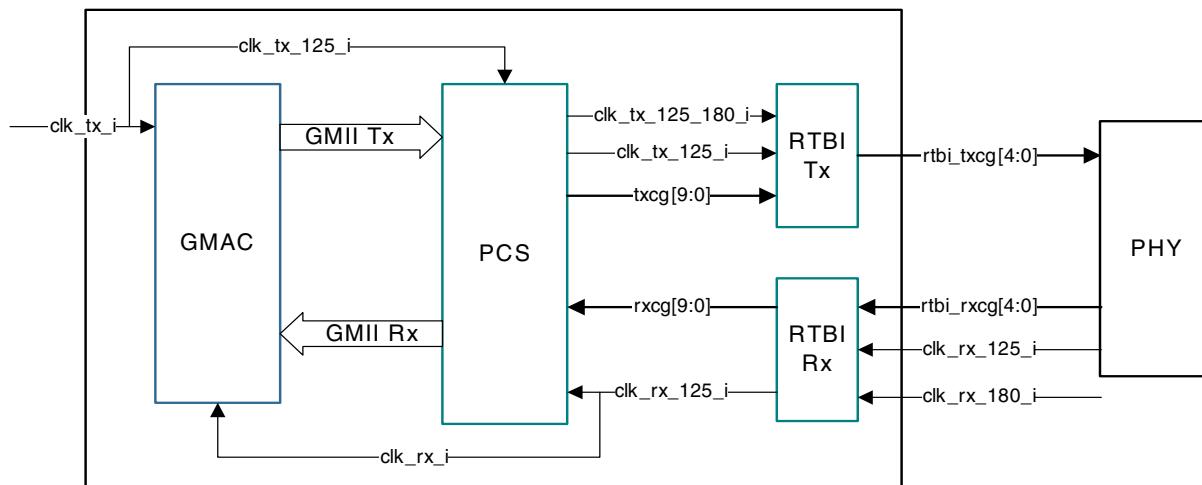
- ❖ Guidelines for clock connections when the GMAC operates with an RTBI are given in “[Clocks With RTBI](#)” on page [439](#).
- ❖ There is a synthesis option for the GMAC-UNIV to inherently delay the RTBI transmit clock with respect to the data and control signals, as described in Version 2.0 of RGMII/RTBI specification.

The RTBI module can be included in the GMAC-UNIV controller by selecting the RTBI interface in the coreConsultant during configuration.

4.14.1 Block Diagram

[Figure 4-34](#) shows the position of the RTBI block and its I/O signals relative to the GMAC and RTBI PHY.

Figure 4-34 RTBI Block Diagram



4.14.2 Block Diagram Overview

Transmit (RTBI Tx) Block: Translates the 10-bit Transmit code-group signal from PCS module to 5-bit transmit code-group. The RTBI Tx registers the input signals at the rising edge of `clk_tx_125_i` and generates the RTBI transmit signals at the rising edge of either `clk_tx_125_i` or `clk_tx_125_180_i`.

Receive (RTBI Rx) Block: Translates the 5-bit code-group signals received from the PHY to 10-bit code-group towards the PCS. The RTBI Rx registers the input signals at the rising edge of either `clk_rx_125_i` or `clk_rx_180_i` and generates the `rxcg[9:0]` signal output at the rising edge of `clk_rx_125_i`.



Note `clk_tx_125_180_i` and `clk_rx_180_i` should be 180 degrees out-of-phase with respect to the `clk_tx_125_i` and `clk_rx_125_i` clocks respectively.

4.14.3 RTBI Transmit Timing

In the default mode, the RTBI transmitter outputs the lower 5 bits of the input `txcg[9:0]` (from PCS) at the rising edge of `clk_tx_125_i` followed by the upper 5 bits of `txcg[9:0]` on the falling edge of `clk_tx_125_i`.

4.15 Serial Gigabit Media Independent Interface

The Serial Gigabit Media Independent Interface (SGMII) defines the interface from the Gigabit MAC to the Gigabit PHY. Using the SGMII lowers the pin count required for the GMII. The interface definition supports

all speed modes (10 Mbps, 100 Mbps, and 1 Gbps). The reduced pin count comes at the cost of higher power consumption, mainly because the SGMII interface maintains a constant 1250-MHz clock rate, regardless of the MAC's operating speed.

The SGMII block has the following characteristics:

- ❖ Supports 10-Mbps, 100-Mbps and 1000-Mbps operations.
- ❖ Provides in-band (link speed, duplex mode and link status) status signals from PHY provided to GMAC for link detection.
- ❖ Requires separate 125-MHz clock inputs for transmit and receive paths. For guidelines for clock connections when the GMAC operates with an SGMII, see “[Clocks With SGMII](#)” on page [431](#).

The Synopsys RTL deliverable does not include the SerDes and high-speed I/O blocks. You must develop these modules to suit the technology and ASIC manufacture process.

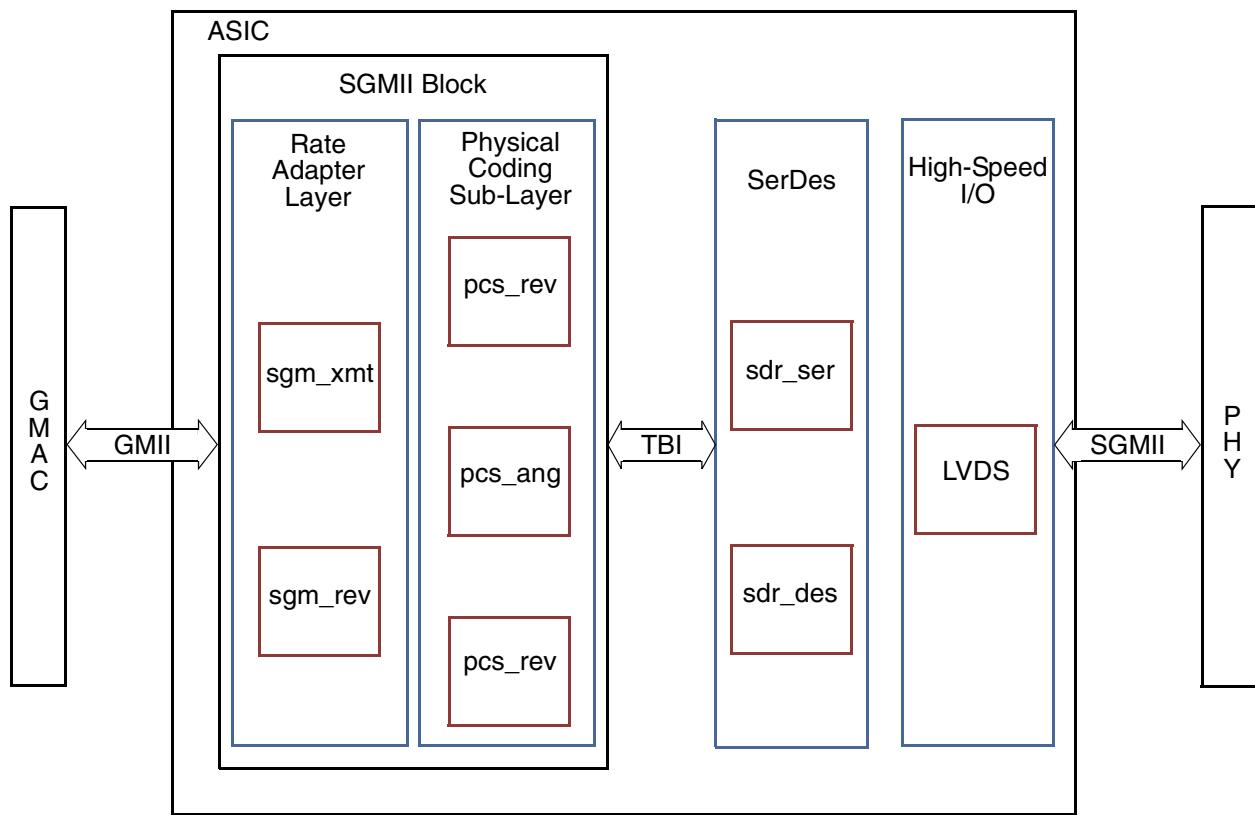
You can include the SGMII module in the DWC Ether MAC 10/100/1000 Universal controller by selecting the SGMII PHY interface in the coreConsultant during configuration.

4.15.1 Block Diagram

[Figure 4-35](#) shows the placement of the SGMII block with respect to the DWC Ether MAC 10/100/1000 Universal and PHY blocks.

4.15.2 Block Overview

The SGMII block comprises the Rate Adapter Layer (RAL) and the Physical Coding Scheme (PCS) modules as shown in [Figure 4-35](#). You must implement the SerDes and high-speed I/O modules to implement the actual SGMII interface.

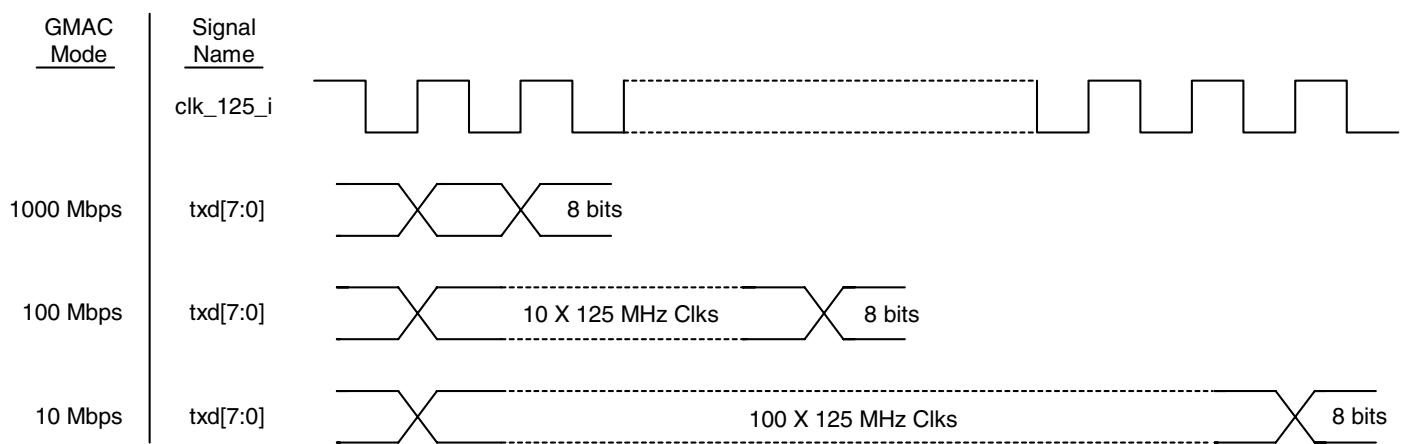
Figure 4-35 SGMII Block Diagram

4.15.3 Block Descriptions

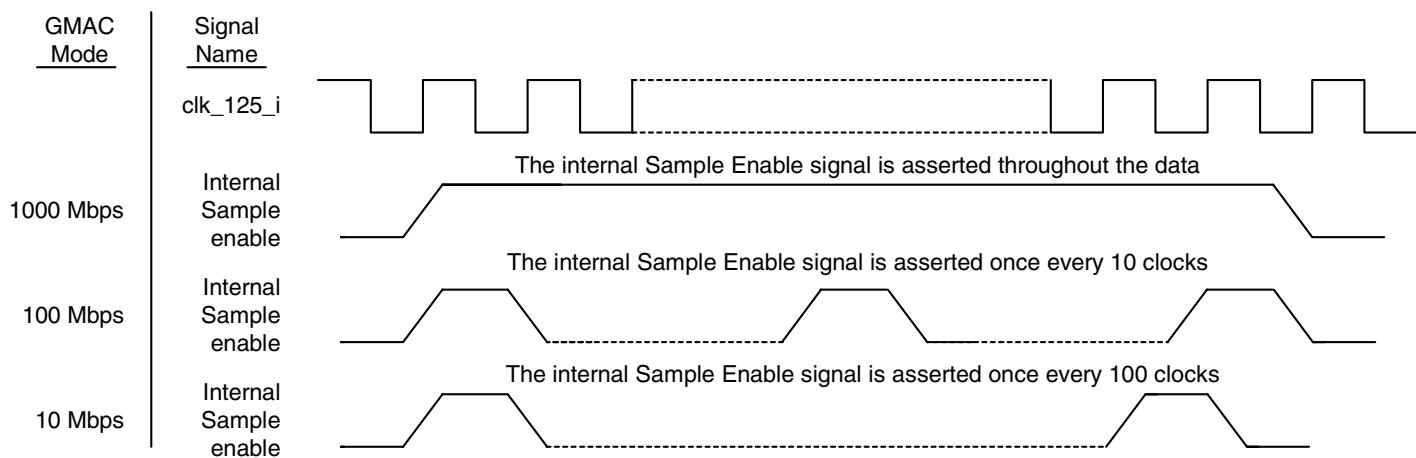
4.15.3.1 Rate Adapter Layer (RAL)

Because the PCS layer runs on a 125-MHz clock while the GMII interface may run on a 2.5-MHz, 25-MHz, or 125-MHz clock, the data to/from the GMII must be stretched in the transmit path, based on the DWC Ether MAC 10/100/1000 Universal data rate. [Figure 4-36](#) shows how a single 8-bit data stream is stretched with respect to the 125-Mhz clock.

If the DWC Ether MAC 10/100/1000 Universal is in the Gigabit mode, then the data is not stretched at all, and takes only one clock cycle. However, if the GMAC is in either the 100 Mbps or 10-Mbps mode, the data is repeated for 10 or 100 consecutive 125-MHz clock cycles, respectively. In [Figure 4-36](#), data is repeated 10 or 100 times in the RAL's transmit path to match the speed mode of the MAC layer. txd[7:0] is an internal signal.

Figure 4-36 Data Stretched in the Tx Path With a 125-MHz Clock

In the receive path, the SGMII PHY stretches the data according to the MAC's speed before it sends it out to the SGMII block. The RAL receives the data repeated 10 or 100 times, depending on what mode the MAC is in. Therefore, the RAL does not have to repeat that data, as in the case of the transmit path. It only samples the data once every 10 or 100 clocks, as shown in [Figure 4-37](#). For additional information, refer to the Serial GMII Specification, Revision 1.7.

Figure 4-37 Data Stretched in the Rx Path With a 125-MHz Clock

4.15.3.2 Physical Coding Sub-Layer (PCS)

In the transmit path, the Physical Coding Sub-Layer takes the 8-bit-wide GMII data from the RAL and encodes it using the 8-bit/10-bit algorithm to supply the SerDes block with a 10-bit-wide data stream. In the receive block, the Physical Coding Sub-Layer takes 10-bit-wide data from the SerDes block and decodes it to supply the RAL block with an 8-bit-wide data stream.

The PCS module is configured with the changes specific to SGMII as given in the SGMII Specification. These changes are with respect to the Link Timer duration (1.6 ms instead of 10 ms) and the Configuration Register codes transmitted during auto-negotiation.

The tx_config_reg[15:0] bits sent by the MAC during Auto-negotiation depend on whether the Transmit Configuration register bit is enabled for the SGMII interface. [Table 4-34](#) describes this in detail.

Table 4-34 tx_config_reg[15:0] Setting When Transmit Configuration Register Bit Is Enabled/Disabled

Transmit Configuration Bit		
Bit	Disabled (Default)	Enabled
15	0	Link Up/Down (Bit 8 of MAC Configuration register)
14	1	1
13	0	0
12	0	Duplex (Bit 11 of MAC Configuration register)
11:10	00	<ul style="list-style-type: none"> • 10 in 1000-Mbps mode • 01 in 100-Mbps mode • 00 in 10-Mbps mode <p>Where the speed is determined by the Port Select and Speed Control bits in the MAC Configuration register</p>
9:1	00H	00H
0	1	1

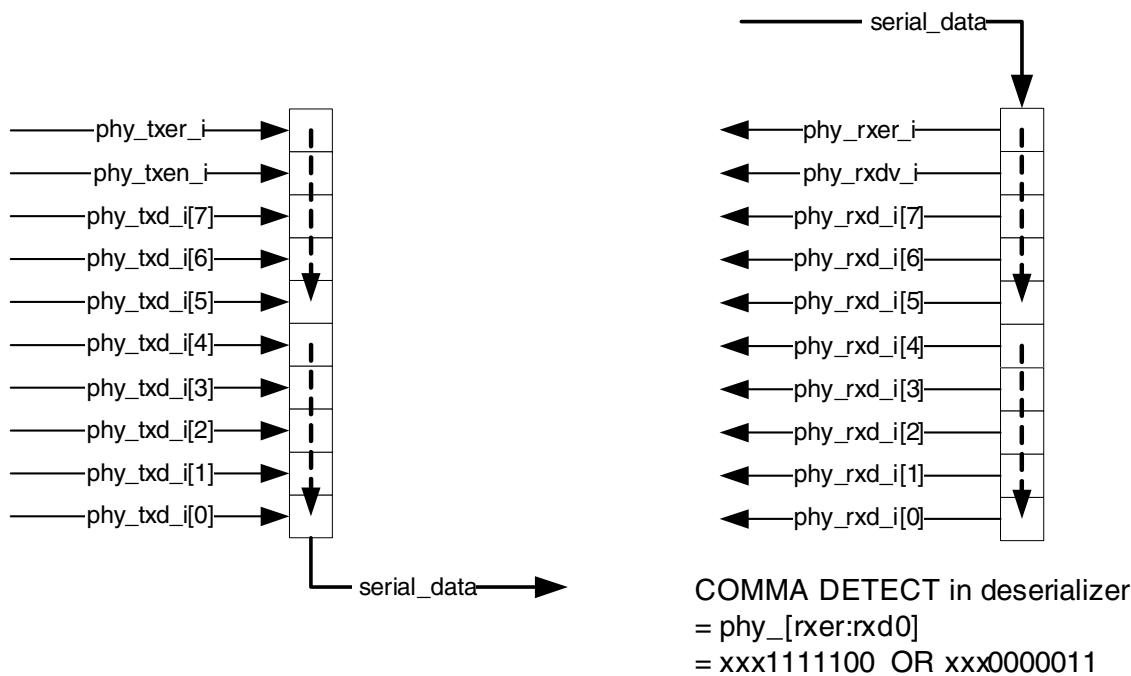
4.15.3.3 SerDes

In the transmit path, the SerDes block takes the 10-bit-wide parallel data from the PCS and converts it to a serial format. In the receive path, the SerDes block takes the serial SGMII input and converts it to a parallel 10-bit-wide data format for the PCS layer.

The connection of the 10-bit transmit data signals with the serializer and the 10-bit receiver data signals from the de-serializer are shown in [Figure 4-38](#).



Note The provided RTL is supplied/used for simulation purposes only and is not to be used for synthesis. The SerDes and LVDS I/O blocks are custom blocks of technology-specific library parts. See [Figure 4-39](#) for a block diagram of the GMAC core (DUT) with support for an SGMII interface.

Figure 4-38 SGMII Serializer and De-Serializer

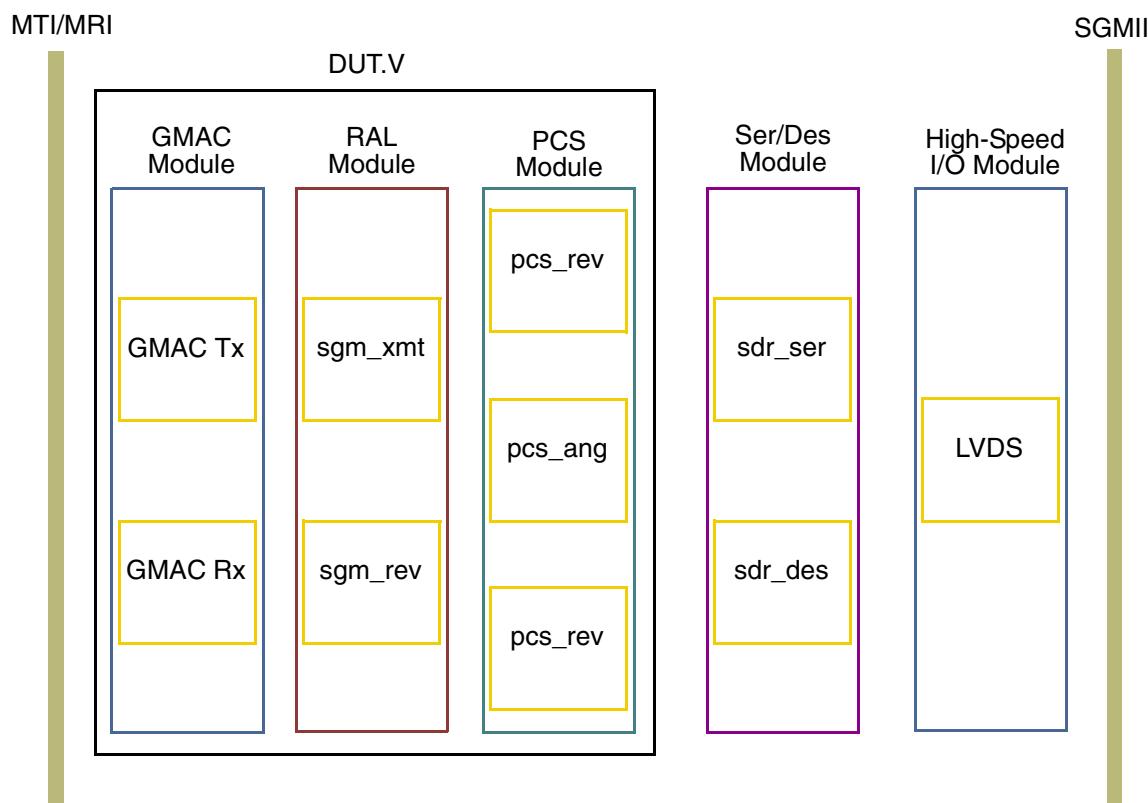
4.15.3.4 High-Speed I/O

The I/O block used in the SGMII is an LVDS type I/O. For additional information, refer to the Serial GMII Specification, Revision 1.7.



Note The provided RTL is supplied/used for simulation purposes only. It should not be used for synthesis.
 The SerDes and LVDS I/O blocks are custom blocks of technology-specific library parts.

Figure 4-39 shows a block diagram of the GMAC core (DUT) with support for an SGMII interface.

Figure 4-39 GMAC Core (DUT) With Support for an SGMII Interface

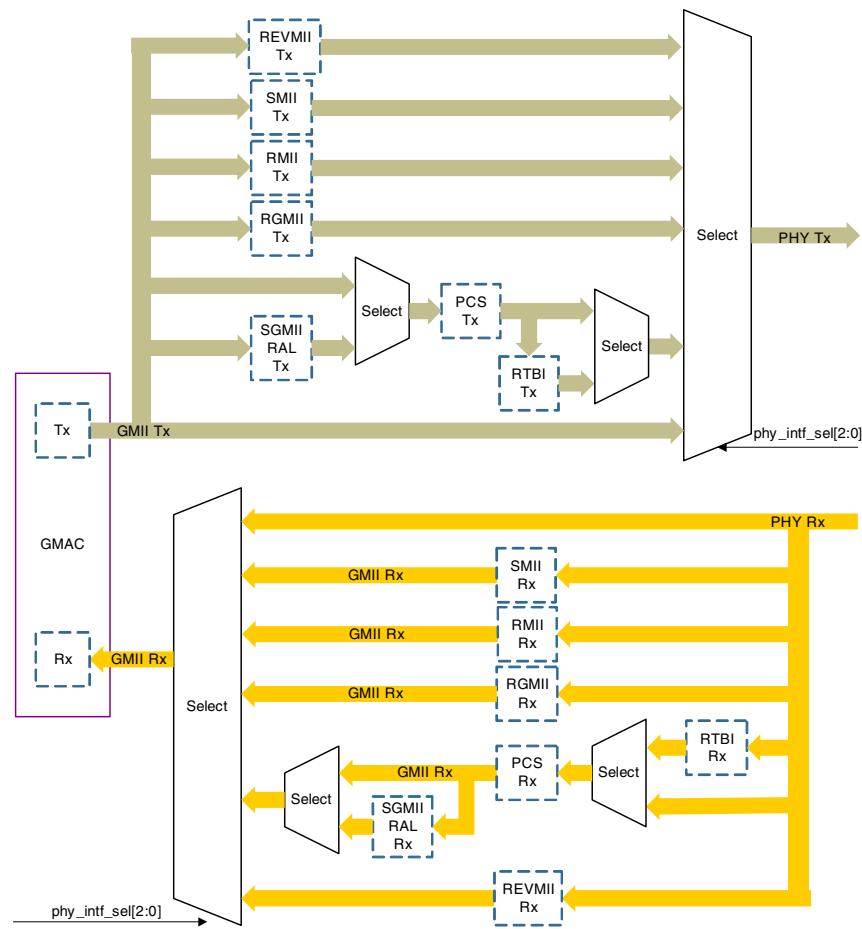
4.16 Multiple PHY Interfaces

Figure 4-40 illustrates the multiplexing logic implemented to support multiple programmable PHY interfaces.



When the core is configured for a single PHY interface, all MUX logic is removed and the corresponding PHY interface is connected directly to the ports.

Figure 4-40 Multiplexing Logic for Multiple Programmable PHY Interfaces



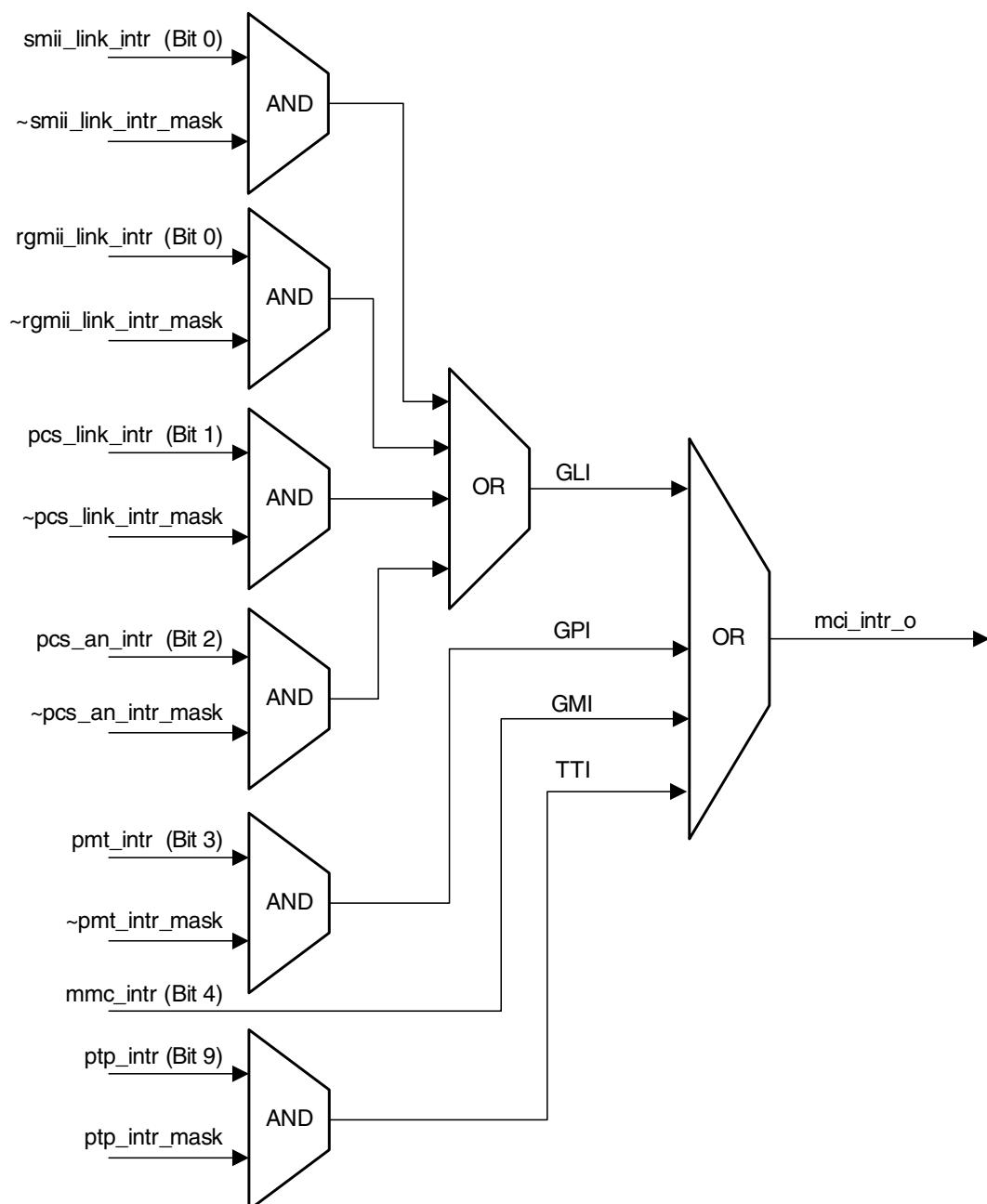
4.17 Interrupts From the GMAC Core

Interrupts can be generated from the GMAC core as a result of various events in the optional modules. `mci_intr_o` is the interrupt signal for the GMAC-CORE and GMAC-MTL configurations, while in other configurations (GMAC-DMA, GMAC-AHB), these interrupt events are combined with the events in the DMA on the `sbd_intr_o` signal.

The Interrupt Status register in the GMAC Register Map ([Table 6-4](#)) describes the events that can cause an interrupt from the GMAC core. Each event can be prevented from asserting the interrupt on the `mci_intr_o` or `sbd_intr_o` signals by setting the corresponding mask bits in the Interrupt Mask register.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt. For example, Bit 0 of the Interrupt register, set high, indicates that the link status on the RGMII/SMII interface has changed. You must read Register 54 (the RGMII/SMII Status register) to clear this interrupt event.

The interrupts from the RGMII/SMII and PCS blocks are combined (OR'ed) and given as the GLI bit in the DMA Status register. The TTI, GPI, and GMI signals in [Figure 4-41](#) refer to the bits that can be read in the DMA Status register.

Figure 4-41 GMAC Core Interrupt Masking Scheme

5

Signals

This chapter provides information about the top-level signals of GMAC-UNIV. It contains the following sections:

- ❖ “[Top-Level I/O Diagram](#)” on page [205](#)
- ❖ “[Signal Descriptions](#)” on page [212](#)

5.1 Top-Level I/O Diagram

This section provides the top-level diagrams of GMAC-UNIV configurations. Each GMAC-UNIV configuration has a corresponding figure that contains information about signals and signal interface groups. The signal names and the signal interface group names in the figures are cross-referenced to the signal descriptions in [Tables 5-2 to 5-29](#).

As shown in [Table 5-1](#), [Figure 5-1](#) to [Figure 5-5](#) provide information about the default top-level ports of GMAC-UNIV configurations. [Figure 5-6](#) provides information about the optional ports.

Table 5-1 Figure Numbers and Corresponding GMAC-UNIV Configurations

Figure Number	Description
Figure 5-1	Displays top-level ports of the GMAC core only, with native interface (GMAC-CORE).
Figure 5-2	Displays top-level ports of the GMAC-MTL configuration.
Figure 5-3	Displays top-level ports of the GMAC-DMA configuration.
Figure 5-4	Displays top-level ports of the GMAC with AHB interface (GMAC-AHB).
Figure 5-5	Displays top-level ports of the GMAC with AXI interface (GMAC-AXI).
Figure 5-6	Displays top-level ports of the optional interfaces.

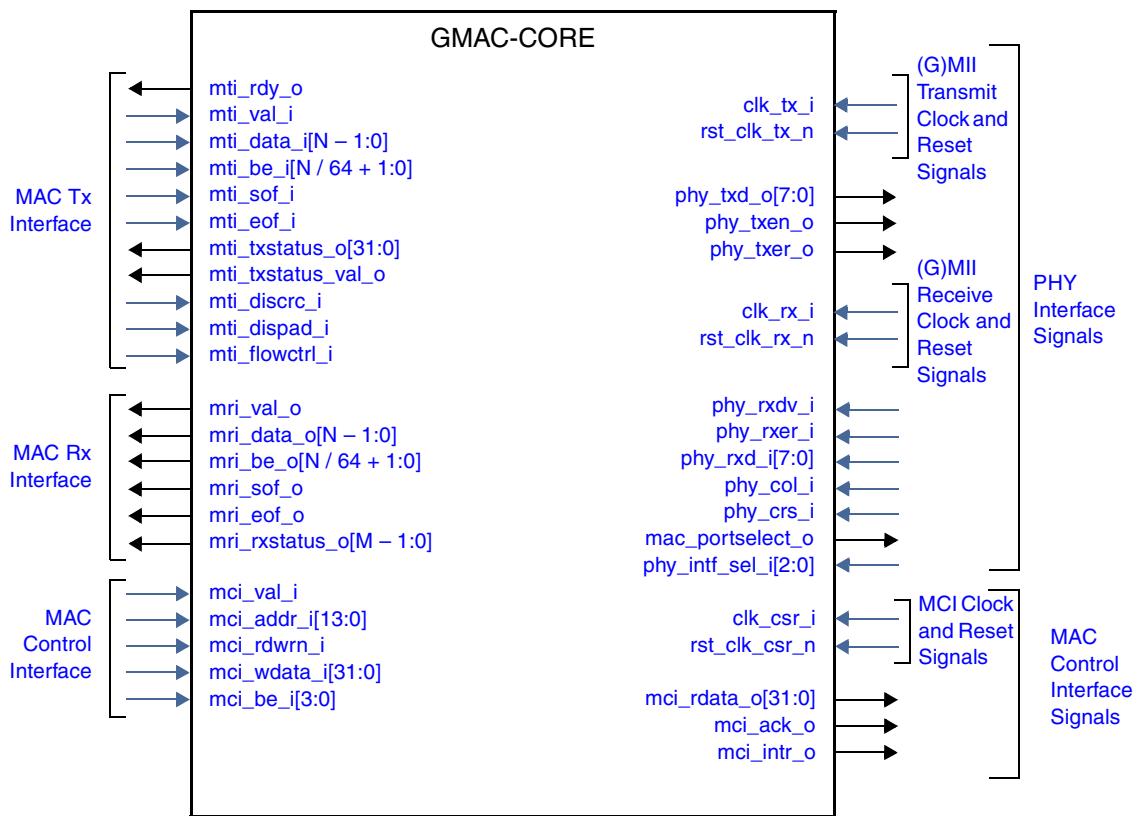
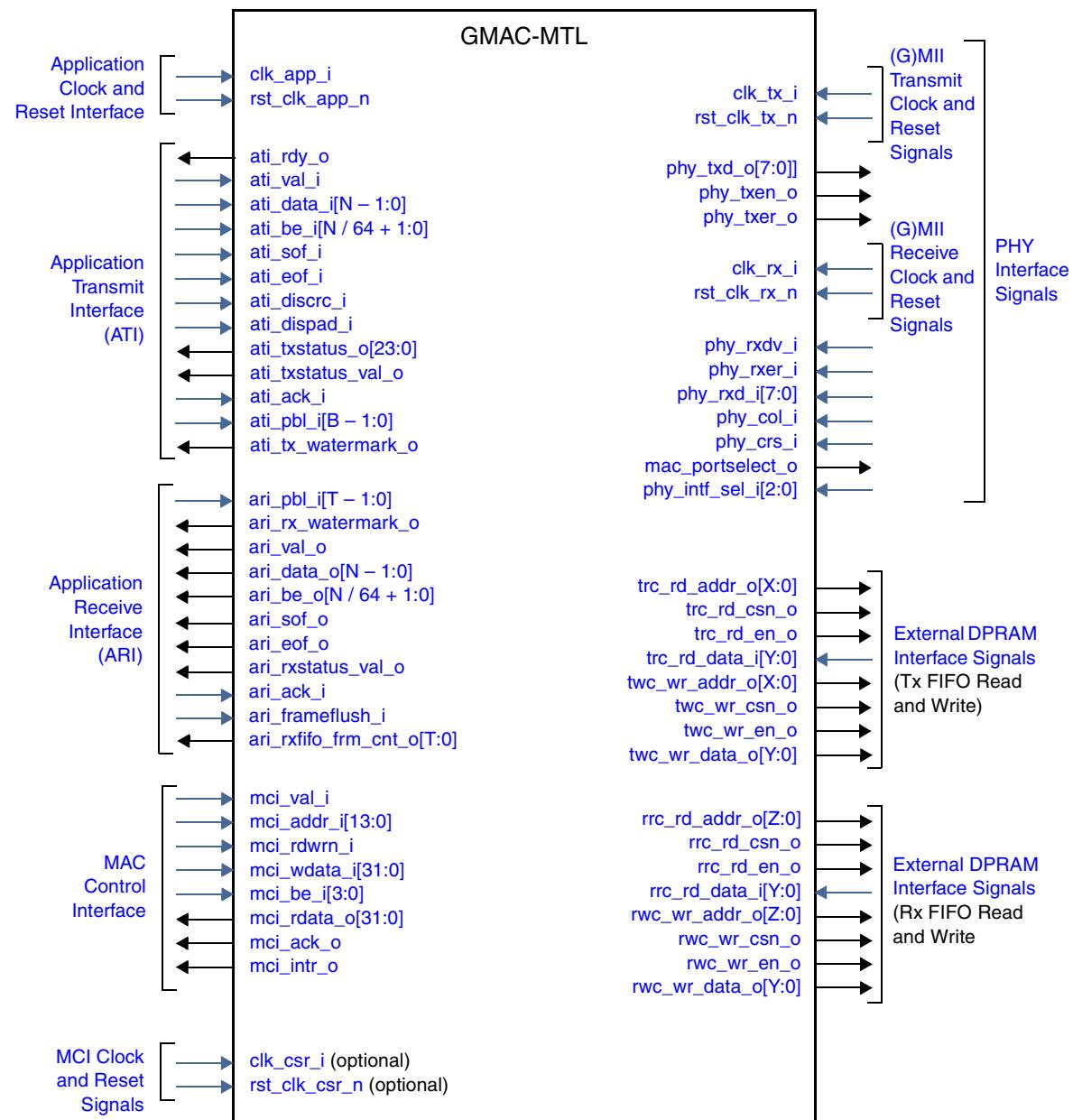
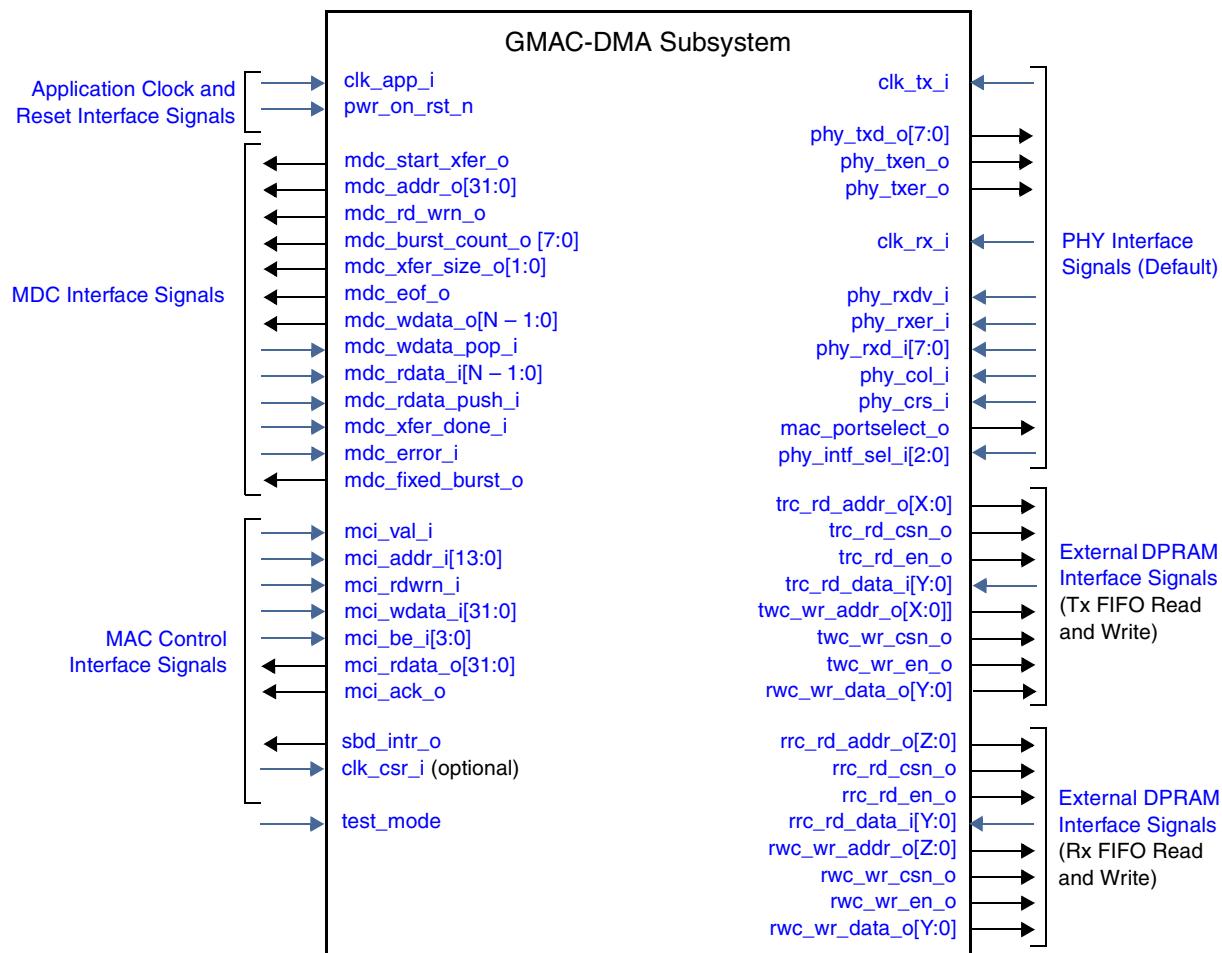
Figure 5-1 GMAC-CORE Interface Top-Level I/O Diagram

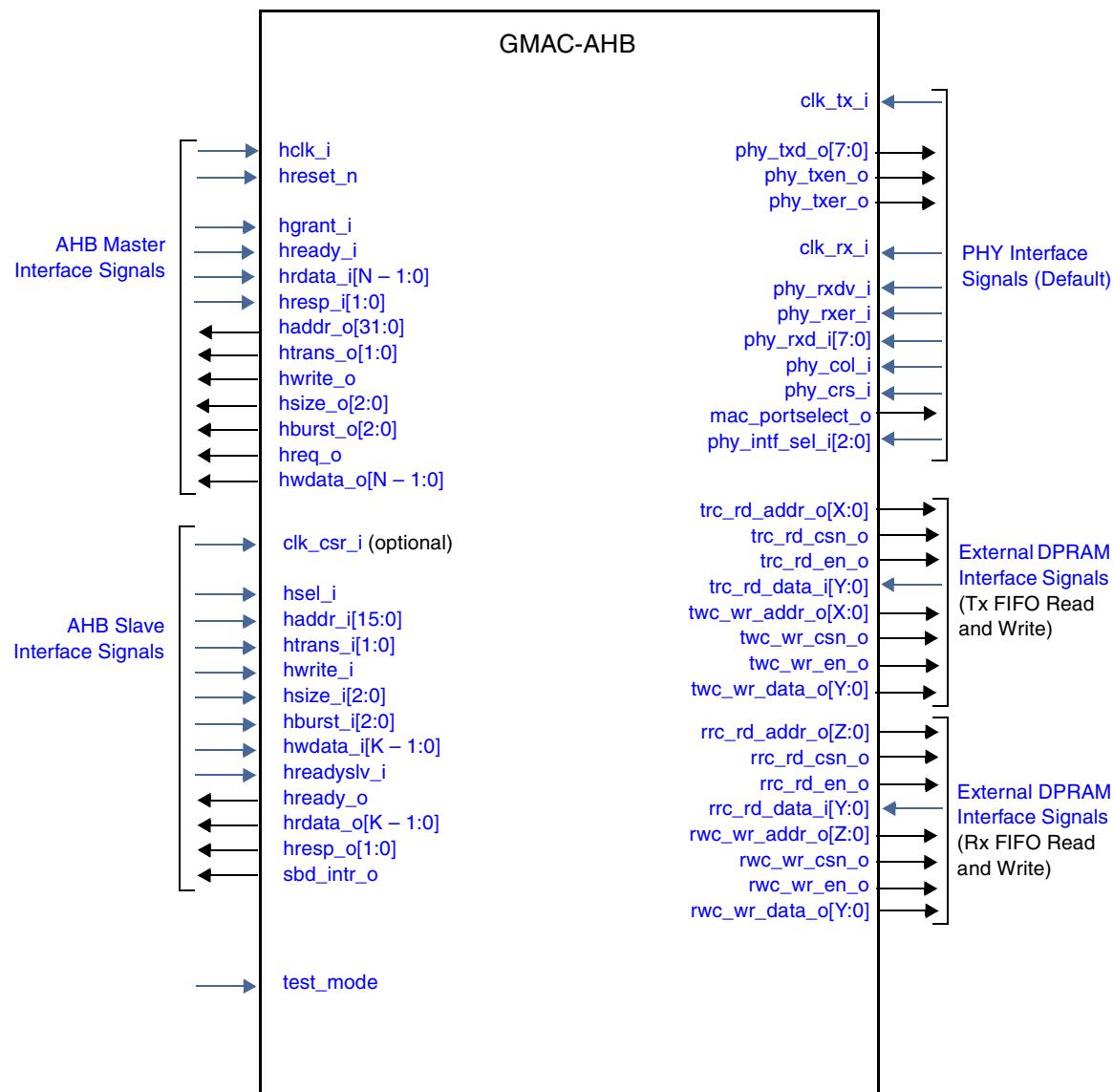
Figure 5-2 GMAC-MTL Interface Top-Level I/O Diagram

The address and data width of the memory interface (External DPRAM) depends on the configuration (FIFO depth and data width).

Figure 5-3 GMAC-DMA (GMII) Configuration Top-Level I/O Diagram

Note

The address and data width of the memory interface (External DPRAM) depends on the configuration (FIFO depth and data width).

Figure 5-4 GMAC-AHB (GMII) Top-Level I/O Diagram

The address and data width of the memory interface (External DPRAM) depends on the configuration (FIFO depth and data width).

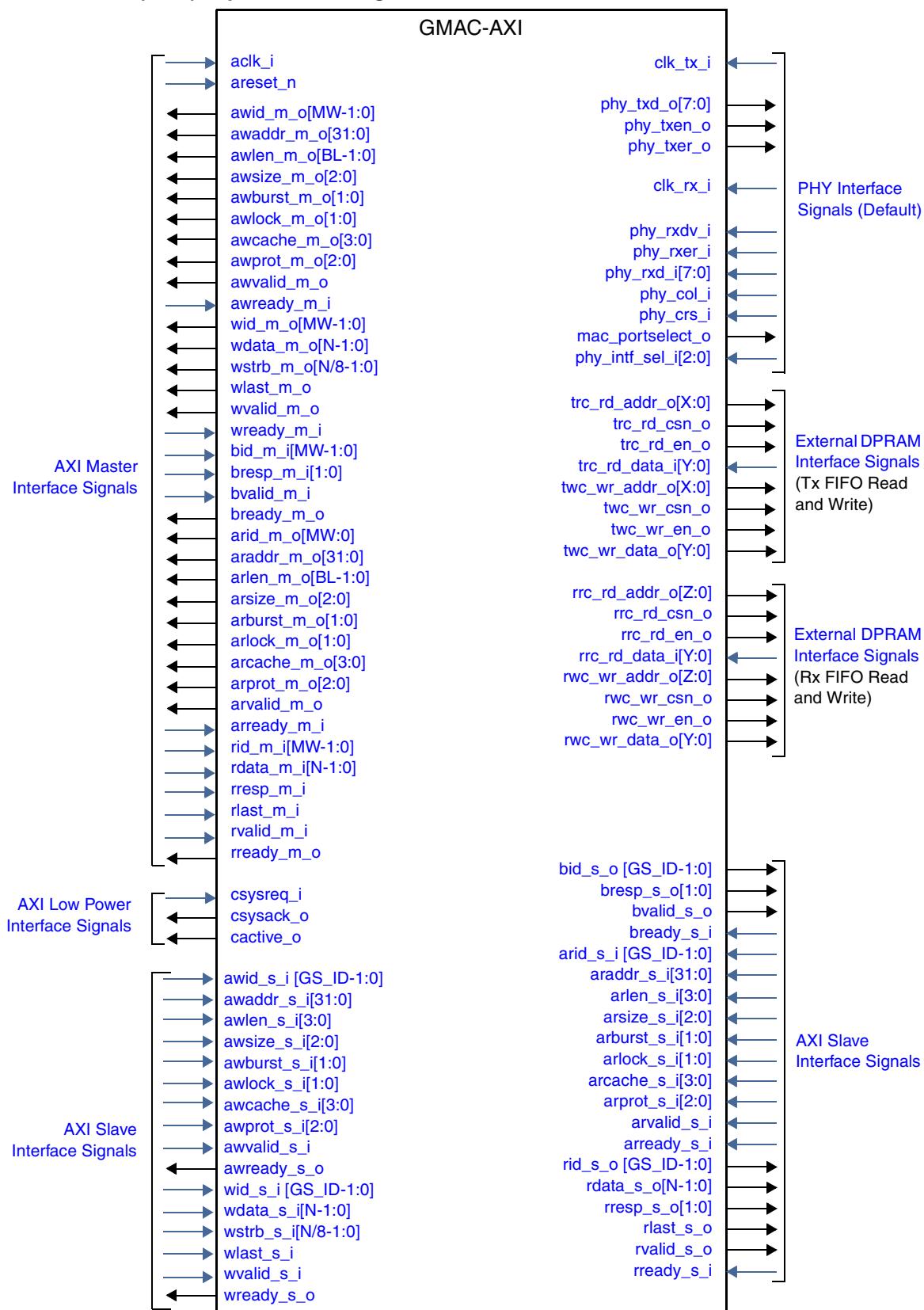
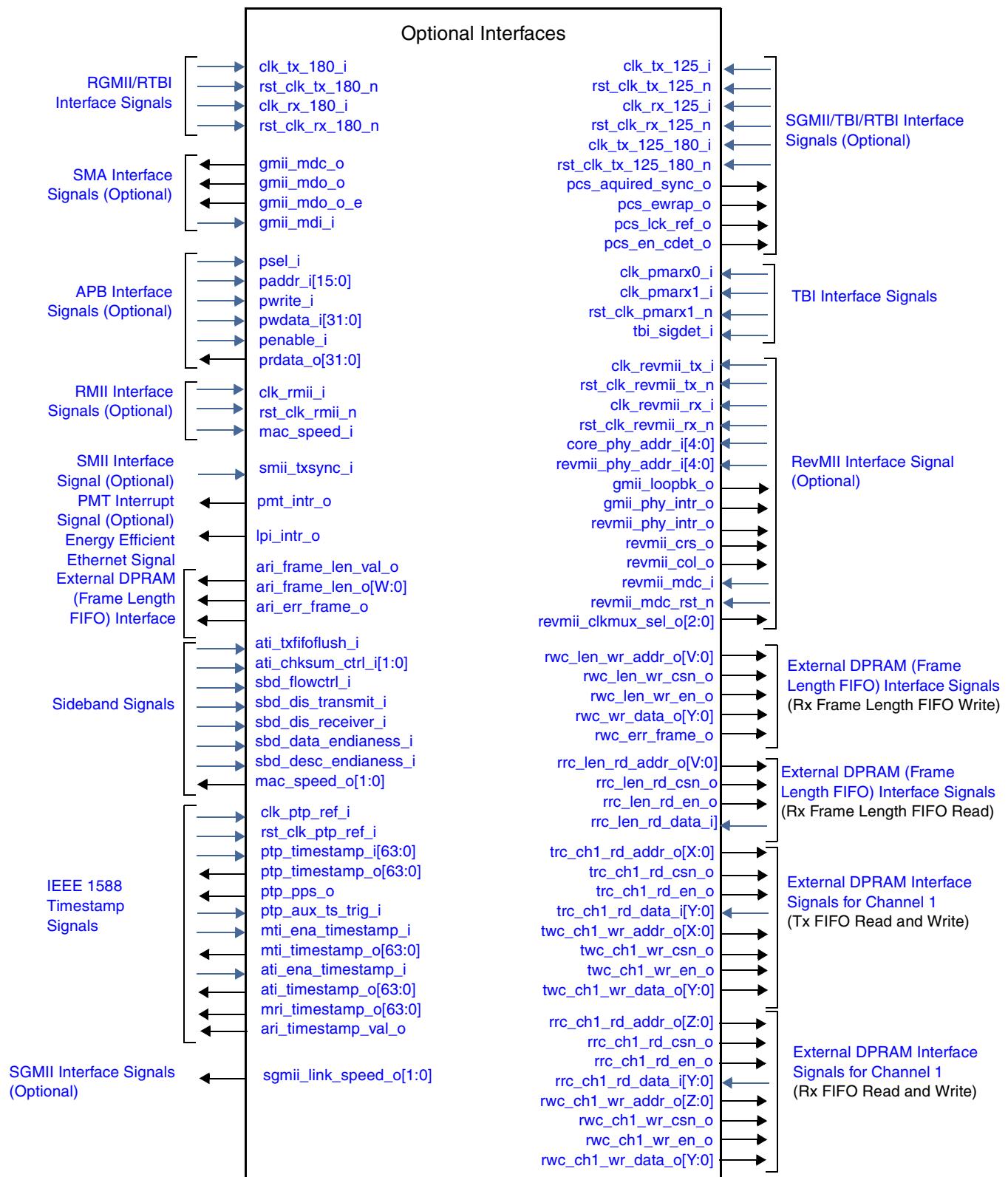
Figure 5-5 GMAC-AXI (GMII) Top-Level I/O Diagram

Figure 5-6 Optional Interfaces Top-Level I/O Diagram



- All reset signals (rst_*) in [Figure 5-6](#) are required only for GMAC-CORE or GMAC-MTL configurations.
- The External DPRAM interface signal names for channel 2 are similar to the signal names for channel 1. For channel 2, the value ch1 in the signal name is replaced by ch2.

5.2 Signal Descriptions

In addition to describing the function of each signal, this section provides the following information about each signal:

- ❖ **I/O:** Indicates whether the signal is an input or an output. A signal with “_i” is an input. A signal with “_o” is an output. For example, signal clk_tx_i is an input.
- ❖ **Registered:** Indicates whether or not the signal is registered directly at the software-core boundary. A “No” value does not mean the signal is not synchronous, only that there is some combinatorial logic between the signal’s origin or destination register and the soft-core boundary. A value of “Optionally” means that the signal is not registered at the core boundary by default, but you can choose to insert an input/output register by selecting the corresponding configuration option.
- ❖ **Synchronous to:** Indicates to which clock (for example, clk_csr_i or clk_tx_i) the signal is synchronous, or if the signal is asynchronous.
- ❖ **Active State:** Indicates the active state (high or low) of the signal. All signals with “_n” are active low. For example, signal hreset_n is an active low signal.

5.2.1 System Clock and Reset Signals (Default)

The system clock and reset signals are described in [Table 5-2](#).

Table 5-2 System Clock and Reset Signals

Signal	I/O	Description
(G)MII Transmit Clock and Reset Signals		
clk_tx_i	In	<p>Transmission Clock</p> <p>Function: This is the transmission clock (125/25/2.5 MHz in 1G/100M/10Mbps) provided by the external PHY/oscillator for the RGMII/GMII/MII. All the RGMII/GMII/MII transmission signals generated by the GMAC-UNIV are synchronous to this clock. This clock input is required even when you select the RMII, SGMII, TBI, RTBI, or SMII interface.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
rst_clk_tx_n	In	<p>Transmit Clock Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>

Table 5-2 System Clock and Reset Signals (Continued)

Signal	I/O	Description
(G)MII Receive Clock and Reset Signals		
clk_rx_i	In	<p>Reception Clock</p> <p>Function: This is the reception clock (125/25/2.5 MHz in 1G/100M/10Mbps) that external PHY provides for RGMII, GMII, MII, and RMII interfaces. All RGMII, GMII, and MII receive signals that GMAC-UNIV receives are synchronous to this clock. This clock input is required even when you select the RMII, SGMII, TBI, RTBI, or SMII interface.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
MCI Clock and Reset Signals		
clk_csr_i	In	<p>CSR Clock</p> <p>Function: This is the free-running Application clock input provided by the Application. The MAC Control Interface, CSR, and Station Management Agent of the GMAC run on this clock. All signals generated by the CSR module are synchronous to clk_csr_i. The valid frequency range of this clock is 20–300 MHz (minimum frequency is 25 MHz when the MMC module is active in 1000-Mbps mode). Frequencies outside this range may result in incorrect operation. By default, this clock port is internally tied to hclk_i in the GMAC-AHB configuration, and clk_app_i in the GMAC-MTL and GMAC-DMA configurations. This input port is present only if you select a separate clock for the CSR port in the GMAC-AHB, GMAC-DMA, or GMAC-MTL configuration. This clock input is required in the GMAC-CORE configuration.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
rst_clk_csr_n	In	<p>CSR Clock Reset</p> <p>Function: This is an active low reset signal. All registers and counters inside the CSR go to their default state when rst_clk_csr_n is asserted. This input is required in the GMAC-CORE configuration by default. It is an optional input for the GMAC-MTL configuration and is not present in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>

5.2.2 PHY Interface Signals (Default)

The default PHY Interface signals are described in [Table 5-3](#).

Table 5-3 PHY Interface Signals

Signal	I/O	Description
phy_intf_sel_i[2:0]	In	<p>PHY Interface Select</p> <p>Function: These pins select one of the multiple PHY interfaces of GMAC. This is sampled only during reset assertion (rst_clk_csr_n for GMAC-CORE, rst_clk_app_n for GMAC-MTL, pwr_on_rst for GMAC-DMA, hreset_n for GMAC-AHB, and areset_n for GMAC-AXI), and ignored after that. This input is removed from the port list when the core is configured for a single PHY interface.</p> <ul style="list-style-type: none"> • 000: GMII/MII • 001: RGMII • 010: SGMII • 011: TBI • 100: RMII • 101: RTBI • 110: SMII • 111: RevMII • All others: Reserved <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>
mac_portselect_o	Out	<p>MAC Port Select</p> <p>Function: This signal mirrors the PS bit of the GMAC Configuration Register. It is low after reset and is used by the external clock circuitry to generate the GMII/MII clocks. A high indicates an MII interface, and a low a GMII interface.</p> <p>This output signal is substituted by mac_speed_o in the port list if the optional mac_speed_o signal output is enabled during the configuration. Therefore, mac_speed_o[1] is equal to mac_portselect_o. For more information, see mac_speed_o[1:0].</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>

Table 5-3 PHY Interface Signals (Continued)

Signal	I/O	Description
phy_txd_o[7:0]	Out	<p>PHY Transmit Data</p> <p>Function: This is a group of eight transmit data signals driven by the GMAC. It has multiple functions depending on which PHY interface is selected, as given below. Unused bits in the RGMII/RTBI/RMII/MII interface configurations are tied to low.</p> <ul style="list-style-type: none"> • GMII: All 8 bits provide the GMII transmit data byte. The validity of the data is qualified with phy_txen_o and phy_txer_o. Synchronous to: clk_tx_i • MII: Bits [3:0] provide the MII transmit data nibble. The validity of the data is qualified with phy_txen_o and phy_txer_o. Synchronous to: clk_tx_i • RGMII: Bits [3:0] provide the RGMII transmit data. The data bus changes with both rising and falling edges of the transmit clock (clk_tx_i). The validity of the data is qualified with phy_txen_o. Synchronous to: clk_tx_i, clk_tx_180_i • RMII: Bits [1:0] provide the RMII transmit data. The validity of the data is qualified with phy_txen_o. Synchronous to: clk_rmii_i • SMII: Bit[0] provides the SMII transmit data. Synchronous to: clk_tx_125_i • SGMII/TBI: The 8 bits correspond to the transmit code group [7:0]. Synchronous to: clk_tx_125_i • RTBI: Bits[3:0] provide the 8 bits (bits[3:0] and bits[8:5]) of the RTBI 10-bit transmit code group. The data bus changes with both edges of the transmit clock. Synchronous to: clk_tx_125_i and clk_tx_125_180_i • RevMII: All 8 bits provide the RevMII transmit data byte. The validity of the data is qualified with phy_txen_o and phy_txer_o. Synchronous to: clk_revmii_rx_i <p>Active State: N/A</p> <p>Registered: This signal is a registered output only when a single PHY interface (except RGMII or RTBI) is selected. When MUX logic is present, it is unregistered.</p> <p>Synchronous to: Depends on the selected PHY interface.</p> <p>Note 1: If you have configured the core for 10/100 Mbps-only operation, phy_txd_o is only 4 bits wide.</p> <p>Note 2: If you have configured the core for Single PHY interface, then the width of the signal corresponds to the specified width such as 4, 2, and 1 for RGMII, RMII, and SMII respectively.</p>

Table 5-3 PHY Interface Signals (Continued)

Signal	I/O	Description
phy_txen_o	Out	<p>PHY Transmit Data Enable</p> <p>Function: This signal is driven by the GMAC and has multiple functions depending on which PHY interface is selected, as given below.</p> <ul style="list-style-type: none"> • GMII/MII: When high, indicates that valid data is being transmitted on the phy_txd bus. Synchronous to: clk_tx_i • RMII: When high, indicates that valid data is being transmitted on the phy_txd bus. Synchronous to: clk_rmii_i • SMII: This is the Global synchronization signal for SMII. This signal is not present/valid when global sync signal is an input in SSSMII configuration. Synchronous to: clk_tx_125_i • RGMII: This signal is the control signal (rgmii_tctl) for the transmit data, and is driven on both edges of the clock. Synchronous to: clk_tx_i, clk_tx_180_i • SGMII/TBI: Contains Bit 8 of the transmit code group. Synchronous to: clk_tx_125_i • RTBI: This signal contains 2 bits (Bit[4] and Bit[9]) of the 10-bit RTBI transmit code-group and is driven on both edges of the clock. Synchronous to: clk_tx_125_i, clk_tx_125_180_i. • RevMII: When high, indicates that valid data is being transmitted on the phy_txd bus. Synchronous to: clk_revmii_rx_i <p>Active State: N/A</p> <p>Registered: This signal is a registered output only when a single PHY interface (except RGMII or RTBI) is selected. When MUX logic is present, it is unregistered.</p> <p>Synchronous to: Depends on the selected PHY interface.</p>
phy_txer_o	Out	<p>PHY Transmit Error</p> <p>Function: This signal is driven by the GMAC and has multiple functions depending on which PHY interface is selected, as explained in the following list:</p> <ul style="list-style-type: none"> • GMII: When high, indicates a transmit error or carrier extension on the phy_txd bus. For more information about the GMII transmit control signals, see Table 4-32. Synchronous to: clk_tx_i • MII, RMII, RGMII, SMII, or RTBI: Not used. Tied low in some configurations; driven low in others. • SGMII/TBI: Contains Bit 9 of the transmit code group. Synchronous to: clk_tx_125_i • RevMII: When high, indicates a transmit error or carrier extension on the phy_txd bus. Synchronous to: clk_revmii_rx_i <p>Active State: N/A</p> <p>Registered: This signal is a registered output only when a single PHY interface is selected. When MUX logic is present, it is unregistered.</p> <p>Synchronous to: Depends on the selected PHY interface.</p> <p>Note: If you have configured the core for Single PHY interface, then this output is present only for the required interface such as only in GMII/SGMII/TBI.</p>

Table 5-3 PHY Interface Signals (Continued)

Signal	I/O	Description
phy_rxd_i[7:0]	In	<p>PHY Receive Data</p> <p>Function: This is a bundle of eight data signals received from the PHY. It has multiple functions depending on which PHY interface is selected, as given below.</p> <ul style="list-style-type: none"> GMII: All 8 bits provide the GMII receive data byte. The validity of the data is qualified with phy_rxrv_i and phy_rxer_i. Synchronous to: clk_rx_i MII: Bits [3:0] provide the MII receive data nibble. The validity of the data is qualified with phy_rxrv_i and phy_rxer_i. Synchronous to: clk_rx_i RGMII: Bits [3:0] provide the RGMII receive data. The data bus is sampled with both rising and falling edges of the receive clock (clk_rx_i). The validity of the data is qualified with phy_rxrv_i. Synchronous to: clk_rx_i, clk_rx_180_i RMII: Bits [1:0] provide the RMII receive data. The validity of the data is qualified with phy_rxrv_o. Synchronous to: clk_rmii_i SMII: Bit[0] provides the SMII receive data. Synchronous to: clk_rx_125_i SGMII: The 8 bits correspond to the receive code group [7:0]. Synchronous to: clk_rx_125_i TBI: The 8 bits correspond to the receive code group [7:0]. Synchronous to: clk_pmarx0_i, clk_pmarx1_i RTBI: Bits [3:0] provides the 8 bits (Bits[3:0] and Bits[8:5]) of the 10-bit receive code-group and is sampled at both edges of the clock. Synchronous to: clk_rx_125_i and clk_rx_180_i RevMII: All 8 bits provide the RevMII receive data byte. The validity of the data is qualified with phy_rxrv_i and phy_rxer_i. Synchronous to: clk_revmii_tx_i <p>Active State: N/A</p> <p>Registered: Yes, except when RTBI is selected along with TBI/SGMII.</p> <p>Synchronous to: Depends on the selected PHY interface.</p> <p>Note 1: If you have configured the core for 10/100 Mbps-only operation, phy_rxd_i is only 4 bits wide.</p> <p>Note 2: If you have configured the core for Single PHY interface, then the width of the signal corresponds to the specified width such as 4, 2, and 1 for RGMII, RMII, and SMII respectively.</p>

Table 5-3 PHY Interface Signals (Continued)

Signal	I/O	Description
phy_rxrv_i	In	<p>PHY Receive Data Valid</p> <p>Function: This signal is driven by PHY and has multiple functions depending on which PHY interface is selected as given below.</p> <ul style="list-style-type: none"> GMII/MII: When high, indicates that the data on the phy_rxd bus is valid. It remains asserted continuously from the first recovered byte/nibble of the frame through the final recovered byte/nibble. Synchronous to: clk_rx_i RGMII: This is the receive control signal used to qualify the data received on phy_rxd. This signal is sampled on both edges of the clock. Synchronous to: clk_rx_i, clk_rx_180_i RMII: Contains the crs and data valid information of the receive interface. Synchronous to: clk_rmii_i SMII: This is the Receive Global synchronization signal input for SSSMII. This signal is not present/valid when SMII is not configured for SSSMII. Synchronous to: clk_rx_125_i SGMII: Contains Bit 8 of the receive code group. Synchronous to: clk_rx_125_i TBI: Contains Bit 8 of the receive code group. Synchronous to: clk_pmarx0_i, clk_pmarx1_i RTBI: This contains the 2 bits (Bit[4] and Bit[9]) of the 10-bit receive code-group and is sampled at both edges of the clock. Synchronous to: clk_rx_125_i and clk_rx_180_i RevMII: When high, indicates that the data on the phy_rxd bus is valid. It remains asserted continuously from the first recovered byte/nibble of the frame through the final recovered byte/nibble. Synchronous to: clk_revmii_tx_i <p>Active State: N/A</p> <p>Registered: Yes, except when RTBI is selected along with TBI/SGMII</p> <p>Synchronous to: Depends on the selected PHY interface.</p>
phy_rxer_i	In	<p>PHY Receive Error</p> <p>Function: This signal is driven by the PHY and has multiple functions depending on which PHY interface is selected as given below.</p> <ul style="list-style-type: none"> GMII/MII: When high, indicates an error or carrier extension (in GMII) in the received frame on the phy_rxd bus. Synchronous to: clk_rx_i RGMII/RMII/SMII/RTBI: Not used. SGMII: Contains Bit 9 of the receive code group. Synchronous to: clk_rx_125_i TBI: Contains Bit 9 of the receive code group. Synchronous to: clk_pmarx0_i, clk_pmarx1_i RevMII: When high, indicates an error or carrier extension (in GMII) in the received frame on the phy_rxd bus. Synchronous to: clk_revmii_tx_i <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: Depends on the selected PHY interface.</p> <p>Note: If you have configured the core for Single PHY interface, then this input is present only for the required interface such as only in GMII/MII/SGMII/TBI.</p>

Table 5-3 PHY Interface Signals (Continued)

Signal	I/O	Description
phy_crs_i	In	<p>PHY CRS</p> <p>Function: This signal, valid only in the GMII/MII mode, is asserted by the PHY when either the transmit or receive medium is not idle. The PHY de-asserts this signal when both transmit and receive medium are idle. This signal is not synchronous to any clock.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: N/A</p> <p>Note: If you have configured the core for Single PHY interface, then this input is present only for GMII or MII and not present in other PHY interfaces.</p>
phy_col_i	In	<p>PHY Collision</p> <p>Function: This signal, valid only in GMII/MII mode, is asserted by the PHY when a collision is detected on the medium. This signal is not synchronous to any clock.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: N/A</p> <p>Note: If you have configured the core for Single PHY interface, then this input is present only for GMII or MII and not present in other PHY interfaces.</p>



The phy_crs_i and phy_col_i signals are ignored in the configurations with only full-duplex mode enabled.

5.2.3 MAC Tx Interface Signals

The MAC Tx Interface (MTI) signals are described in [Table 5-4](#).

Table 5-4 MAC Tx Interface Signals

Signal	I/O	Description
mti_rdy_o	Out	<p>Transmit Data Ready</p> <p>Function: When high, indicates that the GMAC-CORE is ready to accept the data input on the mti_data_i bus.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
mti_val_i	In	<p>Transmit Data Valid</p> <p>Function: This signal is asserted by the Application to indicate to the GMAC-CORE that data is available for transmission on the mti_data_i bus. This signal also qualifies the validity of mti_be_i, mti_sof_i, mti_eof_i, mti_discrc_i, and mti_dispad_i signals.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>

Table 5-4 MAC Tx Interface Signals (Continued)

Signal	I/O	Description
mti_data_i[N – 1:0]	In	<p>Transmit Data Bus</p> <p>Function: The data for transmission is input to the core on this bus. The width of the bus can be configured to (N=) 32, 64, or 128. The GMAC core can be configured to process the data in little-endian or big-endian format.</p> <p>In the little-endian format, the LSB lane mti_data_i[7:0] is the first byte transmitted by the core. In the big-endian format, the MSB lane mti_data_i[N-1:N-8] is the first byte transmitted by the core.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>
mti_be_i[N / 64 + 1:0]	In	<p>Transmit Byte Lanes</p> <p>Function: This signal indicates the number of byte lanes valid on the mti_data_i bus when mti_eof_i is high. The width of the bus depends on the data bus width (N). If N = 32, the width is 2, for N = 64, the width is 3, for N = 128, the width is 4. When mti_eof_i is low, the GMAC core always considers all byte lanes to have valid data.</p> <p>When the width is 2, a value of</p> <ul style="list-style-type: none"> • 11: all 4 byte lanes have data • 10: LS 3 byte lanes have data • 01: LS 2 byte lanes have data • 00: LS 1 byte lane have data <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>
mti_sof_i	In	<p>Transmit Start of Frame</p> <p>Function: This signal indicates that the start of an Ethernet frame is being transferred to the GMAC core.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
mti_eof_i	In	<p>Transmit End of Frame</p> <p>Function: This signal indicates that the end of an Ethernet frame is being transferred to the GMAC core.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
mti_txstatus_o[31:0]	Out	<p>Transmit Status</p> <p>Function: This bus gives the transmission status of the last Ethernet frame sent to the GMAC core on the mti_data_i bus.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>

Table 5-4 MAC Tx Interface Signals (Continued)

Signal	I/O	Description
mti_txstatus_val_o	Out	<p>Transmit Status Valid</p> <p>Function: This signal is set high for 1 clock cycle to indicate that the mti_txstatus_o gives valid status for the last transmitted frame.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
mti_discrc_i	In	<p>Transmit Disable Calculation</p> <p>Function: When set high, this signal instructs the GMAC-CORE to disable the calculation and insertion of the FCS bytes into the frame being transmitted. By asserting this signal, the Application assures the GMAC that the proper CRC is appended prior to sending. The GMAC does not append the CRC for any frames received from the Application with a byte count greater than or equal to 60. However, the GMAC appends both pads and CRC for all frames from the Application with a byte count less than 60 when mti_dispad_i is reset to low, even if mti_discrc_i is set high.</p> <p>This signal is sampled only when the mti_sof_i is high.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
mti_dispad_i	In	<p>Transmit Disable Padding</p> <p>Function: When set high, this signal instructs the GMAC to disable the padding function. By asserting this signal, the Application assures the GMAC that proper padding is appended prior to sending. The GMAC appends only the CRC for the transmitting Ethernet frame. This signal is sampled only when mti_sof_i is high.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>
mti_flowctrl_i	In	<p>Transmit Flow Control</p> <p>Function: When set high, this signal instructs the GMAC to transmit PAUSE Control frames in Full-duplex mode. In Half-duplex mode, the GMAC enables the Back-pressure function until this signal is made low again.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>

5.2.4 MAC Rx Interface Signals

The MAC Rx Interface signals are described in [Table 5-5](#).

Table 5-5 MAC Rx Interface Signals

Signal	I/O	Description
mri_val_o	Out	<p>Receive Data Valid</p> <p>Function: The GMAC-CORE asserts this signal to indicate to the Application (MTL) that received frame data is available on mri_data_o. This signal also qualifies mri_data_o, mri_be_o, mri_sof_o, and mri_eof_o.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
mri_data_o[N – 1:0]	Out	<p>Receive Data Bus</p> <p>Function: This signal carries the received data of the frame from the GMAC-CORE to the MTL. The width of the bus can be configured to (N=) 32, 64, or 128. The validity of mri_data_o is qualified with mri_val_o.</p> <p>In little-endian format, the LSB lane mri_data_o[7:0] is the first byte received by the core. In big-endian format, the MSB lane mri_data_o[N-1:N-8] is the first byte received by the core.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
mri_be_o[N / 64 + 1:0]	Out	<p>Receive Byte Lanes</p> <p>Function: This signal indicates the number of byte lanes valid on mri_data_o when mri_eof_o is high. The width of this bus depends on the data bus width (N). When N = 32, the width is 2; for N = 64, the width is 3, for N = 128, the width is 4. When mri_eof_i is low, the GMAC core always drives all-ones on mri_be_o.</p> <p>When mri_be_o width is 2, a value of</p> <ul style="list-style-type: none"> • 11: all 4 byte lanes have data • 10: LS 3 byte lanes have data • 01: LS 2 byte lanes have data • 00: LS 1 byte lane have data <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
mri_sof_o	Out	<p>Receive Start of Frame</p> <p>Function: This signal indicates that the start of an Ethernet frame is being transferred from the GMAC core.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
mri_eof_o	Out	<p>Receive End of Frame</p> <p>Function: This signal indicates that the end of an Ethernet frame is being transferred from the GMAC core.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>

Table 5-5 MAC Rx Interface Signals (Continued)

Signal	I/O	Description
mri_rxstatus_o[M – 1:0]	Out	<p>Receive Status</p> <p>Function: This bus gives the status of the received Ethernet frame output from the GMAC on mri_data_i. The validity of this signal is qualified with mri_eof_o. The width of the bus(M) can be 32-bit, 48-bit, or 56-bit.</p> <p>The status is 32-bit wide by default. It is 48-bit wide when you select the Advance Timestamp feature and 56-bit wide when you select the AV feature.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>

5.2.5 MAC Control Interface Signals

The MAC Control Interface signals are described in [Table 5-6](#).

Table 5-6 MAC Control Interface Signals

Signal	I/O	Description
mci_val_i	In	<p>Data Valid</p> <p>Function: The application asserts mci_val_i to access the GMAC's CSR module, indicating to the GMAC that the address, Read/Write and, in the case of a Write, that the data are all valid. The assertion of this signal is the start of transaction.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>
mci_addr_i[13:0]	In	<p>Address</p> <p>Function: This signal carries the register address of the CSR module from the Application. The address is valid only when mci_val_i is asserted.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>
mci_rdwrn_i	In	<p>Read/Write</p> <p>Function: This signal alerts the GMAC to a Read/Write operation. The Application writes the data into the GMAC Control registers by de-asserting this signal. Asserting this signal indicates the Read operation.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>

Table 5-6 MAC Control Interface Signals (Continued)

Signal	I/O	Description
mci_wdata_i[31:0]	In	<p>Write Data</p> <p>Function: This signal carries the Write data from the Application for the Control registers of the CSR module. The validity of each byte in mci_wdata_i[31:0] is qualified with the byte enables on mci_be_i[3:0].</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>
mci_be_i[3:0]	In	<p>Bytes Enabled</p> <p>Function: This signal indicates which bytes are valid on the mci_wdata_i[31:0] signal. This is valid when mci_val_i is asserted.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>
mci_ack_o	Out	<p>Data Acknowledge</p> <p>Function: This is an acknowledgement from the MAC Control Interface indicating that write data on the mci_wdata_i bus has been accepted for the CSR write operation and valid read data is available on the mci_rdata_o bus for the CSR read operation.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>
mci_rdata_o[31:0]	Out	<p>Read Data</p> <p>Function: This signal carries the Read data from the GMAC's Control registers to the Application. The GMAC supplies all bytes irrespective of the byte enable signal.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>
mci_intr_o	Out	<p>MCI Interrupt</p> <p>Function: When set high, indicates an interrupt event in the core's optional MMC, PCS, RGMII, or PMT module. This interrupt pin is present only in the GMAC-CORE and GMAC-MTL configurations.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>

5.2.6 Application Clock and Reset Interface Signals

The Application Clock and Reset Interface signals are described in [Table 5-7](#).

Table 5-7 Application Clock and Reset Interface Signals

Signal	I/O	Description
clk_app_i	In	<p>Application Clock</p> <p>Function: A free-running input clock from the application. In the GMAC-DMA configuration, the DMA interface signals are synchronous to this clock.</p> <p>In the GMAC-MTL configuration, the MTL application interface operates synchronous to this clock. The valid frequency range is 25 MHz to 300 MHz (minimum frequency is 25 MHz when the MMC module is active in 1000-Mbps mode). The Transmit Write Control (TWC) and Receive Read Control (RRC) blocks operate with this clock.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
rst_clk_app_n	In	<p>Application Clock Reset</p> <p>Function: This is the reset signal synchronous to the application clock domain in the GMAC-MTL configuration.</p> <p>Active State: Low</p> <p>Registered: N/A</p> <p>Synchronous to: clk_app_i</p>
pwr_on_rst_n	In	<p>Application Clock Reset</p> <p>Function: This is the reset signal synchronous to the application clock domain in the GMAC-DMA configuration.</p> <p>Active State: Low</p> <p>Registered: N/A</p> <p>Synchronous to: clk_app_i</p>

5.2.7 Application Transmit Interface (ATI) Signals

The Application Transmit Interface (ATI) signals are described in [Table 5-8](#).

Table 5-8 Application Transmit Interface (ATI) Signals

Signal	I/O	Description
ati_sof_i	In	<p>Start of frame</p> <p>Function: Indicates that the application is ready to transfer the first data of the frame. This signal is qualified by ati_val_i.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
ati_discrc_i	In	<p>Disable CRC</p> <p>Function: When asserted, disables the addition of FCS to the frame by the GMAC core. This signal value is sampled only when ati_sof_i is active. This signal is qualified by ati_val_i.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>

Table 5-8 Application Transmit Interface (ATI) Signals (Continued)

Signal	I/O	Description								
ati_dispad_i	In	<p>Disable Pad</p> <p>Function: When asserted, disables the addition of pads to the frame by the GMAC core. This signal value is sampled only when ati_sof_i is active. This signal is qualified by ati_val_i.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>								
ati_eof_i	In	<p>End of Frame</p> <p>Function: Indicates that the current data transferred to then MTL is the last data phase. This signal is qualified by ati_val_i.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>								
ati_be_i[N / 64 + 1:0]	In	<p>Number of Valid Byte Lanes</p> <p>Function: Indicates the number of valid byte lanes during the last data phase transfer to MTL.</p> <p>Application data width determines the ati_be_i signal width:</p> <table> <thead> <tr> <th>N</th> <th>(N / 64 + 1)</th> </tr> </thead> <tbody> <tr> <td>32</td> <td>1</td> </tr> <tr> <td>64</td> <td>2</td> </tr> <tr> <td>128</td> <td>3</td> </tr> </tbody> </table> <p>This signal is qualified by ati_val_i and ati_eof_i. When ati_eof_i is low, ati_be_i is taken as all-ones by the MTL.</p> <p>When ati_be_i width is 2, a value of</p> <ul style="list-style-type: none"> • 11: all 4 byte lanes have data • 10: LS 3 byte lanes have data • 01: LS 2 byte lanes have data • 00: LS 1 byte lane have data <p>Active State: N/A</p> <p>Registered: Yes. If Transmit Checksum Engine is selected, then No.</p> <p>Synchronous to: clk_app_i</p>	N	(N / 64 + 1)	32	1	64	2	128	3
N	(N / 64 + 1)									
32	1									
64	2									
128	3									
ati_val_i	In	<p>ATI Valid</p> <p>Function: This signal qualifies all the frame transmission control signals and data on application transmit interface</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>								

Table 5-8 Application Transmit Interface (ATI) Signals (Continued)

Signal	I/O	Description
ati_data_i[N – 1:0]	In	<p>Application Data</p> <p>Function: Frame data from the application. Width of the data bus can be configured to 32-bit, 64-bit, or 28-bit.</p> <p>In little-endian format, the LSB lane ati_data_i[7:0] is the first byte transmitted by the core. In big-endian format, the MSB lane ati_data_i[N-1:N-8] is the first byte transmitted by the core.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ati_rdy_o	Out	<p>ATI Ready</p> <p>Function: When asserted, indicates that the MTL is ready to accept additional data from the application. This signal is de-asserted when Tx FIFO is being flushed or when Tx FIFO is full.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ati_txstatus_val_o	Out	<p>Valid Transmit Status</p> <p>Function: Signal qualifying the status word on ati_txstatus_o. This signal is asserted when a valid status word is available in the status FIFO.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
ati_txstatus_o[23:0]	Out	<p>Transmit Status Word</p> <p>Function: Transmit frame status word transferred to the application. The eight most significant bits are tied low in the default mode. Bit 16 is required when a checksum offload engine is included in the core. Bit 17 is required when IEEE1588 time stamping is included.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
ati_ack_i	In	<p>ATI Acknowledgement</p> <p>Function: Acknowledgement signal from the application indicating that the transmit status word has been accepted.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>

Table 5-8 Application Transmit Interface (ATI) Signals (Continued)

Signal	I/O	Description
ati_tx_watermark_o	Out	<p>Transmit Watermark</p> <p>Function: When asserted, indicates that Tx FIFO in MTL has enough space to accommodate the number of data beats as indicated by ati_pbl_i (in the previous clock cycle).</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ati_pbl_i[B – 1:0]	In	<p>Number of Beats</p> <p>Function: Indicates the space requested by the application in MTL (in terms of number of beats: 4, 8, or 16 bytes, respectively, in 32-bit, 64-bit, or 128-bit data bus mode) Transmit FIFO. The maximum limit of the number of beats is equal to half the Tx FIFO depth. Therefore, the width of this signal depends on the data bus width and Tx FIFO depth (B = 8 by default; Ranges from 3 to 10).</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>

5.2.8 Application Receive Interface (ARI) Signals

The Application Receive Interface (ARI) signals are described in [Table 5-9](#).

Table 5-9 Application Receive Interface (ARI) Signals

Signal	I/O	Description
ari_sof_o	Out	<p>Start of Frame</p> <p>Function: Indicates that the current data transferred to the application corresponds to the first data phase of the frame.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ari_eof_o	Out	<p>End of Frame</p> <p>Function: Indicates that the current data transferred to the application corresponds to the last data phase of the frame.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ari_val_o	Out	<p>ARI Valid</p> <p>Function: When asserted, qualifies the data and other control signals on MTL receive application interface.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>

Table 5-9 Application Receive Interface (ARI) Signals (Continued)

Signal	I/O	Description								
ari_data_o[N – 1:0]	Out	<p>Frame Data</p> <p>Function: Frame data from the Rx FIFO to the application.</p> <p>In little-endian format, the LSB lane ari_data_o[7:0] is the first byte received by the core. In big-endian format, the MSB lane ari_data_o[N-1:N-8] is the first byte received by the core.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>								
ari_be_o[N / 64 + 1:0]	Out	<p>Number of Valid Byte Lanes</p> <p>Function: Indicates the number of valid bytes lanes in the last data phase transfer on the MTL receive application interface.</p> <p>Application data width determines the ari_be_o signal width:</p> <table> <tr> <td>N</td> <td>(N / 64 + 1)</td> </tr> <tr> <td>32</td> <td>1</td> </tr> <tr> <td>64</td> <td>2</td> </tr> <tr> <td>128</td> <td>3</td> </tr> </table> <p>This signal is qualified by ari_val_o and ari_eof_o. When ari_eof_o is low, ari_be_o is always driven with all-ones by the MTL.</p> <p>When ari_be_o width is 2, a value of</p> <ul style="list-style-type: none"> • 11: all 4 byte lanes have data • 10: LS 3 byte lanes have data • 01: LS 2 byte lanes have data • 00: LS 1 byte lane have data <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>	N	(N / 64 + 1)	32	1	64	2	128	3
N	(N / 64 + 1)									
32	1									
64	2									
128	3									
ari_ack_i	In	<p>ARI Acknowledgement</p> <p>Function: Acknowledgement from the application indicating that the current data (ari_data_o) including Receive status has been accepted by the host.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>								
ari_rxstatus_val_o	Out	<p>Valid Receive Status</p> <p>Function: Indicates that the ari_data_o bus contains the receive status to the application.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>								

Table 5-9 Application Receive Interface (ARI) Signals (Continued)

Signal	I/O	Description
ari_rx_watermark_o	Out	<p>Receive Watermark</p> <p>Function: When asserted, indicates that the Rx FIFO in the MTL has enough data to generate the number of beats (one beat is of 4, 8, or 16 bytes size in 32, 64, and 128 bit data bus mode) as requested through ari_pbl_i (in the previous clock cycle).</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ari_rxfifo_frm_cnt_o[T:0]	Out	<p>Rx FIFO Frames Counter</p> <p>Function: This signal provides the number of frames (with status) present in the Rx FIFO. The width of the counter depends on the maximum number of frames that can be stored in the Rx FIFO. For example, for an Rx FIFO depth of 2,048 bytes and a minimum receive frame size of 16 bytes, the maximum number of frames that can be stored in the Rx FIFO is 128. Thus, T = 7 in this example.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ari_pbl_i[T – 1:0]	In	<p>Number of Beats</p> <p>Function: Indicates the number of beats of data requested by application from the MTL Receive FIFO. The maximum limit of the number of beats (one beat is of 4, 8, or 16 bytes size in 32-bit, 64-bit, and 128-bit data bus mode) is equal to half the Rx FIFO depth. Thus the width of this signal depends on the data bus width and Rx FIFO depth (T = 8 by default; Ranges from 2 to 11).</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
ari_frameflush_i	In	<p>Frame Flush Request</p> <p>Function: Frame Flush request from the application to flush the current frame that is being transferred to the application.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>

5.2.9 MDC Interface Signals

The MDC signals are described in [Table 5-10](#).

Table 5-10 MDC Interface Signals

Signal	I/O	Description
mdc_start_xfer_o	Out	<p>Start Transfer</p> <p>Function: This signal, when asserted (for 1 clock), indicates the start of a transaction by the DMA. This signal also indicates the validity of the values on mdc_rd_wrn, mdc_burst_count, mdc_xfer_size, and mdc_fixed_burst signal outputs.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
mdc_addr_o[31:0]	Out	<p>Address</p> <p>Function: These bits give the address of the data transfer. The start address of a burst transfer is given along with the assertion of mdc_start_xfer_o, and the address is incremented at the end of each read/write cycle, indicated by the assertion of mdc_wdata_pop_i or mdc_rdata_push_i.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
mdc_rd_wrn_o	Out	<p>Read/Write</p> <p>Function: When high, indicates a read operation command; when low, requests a write transfer.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
mdc_burst_count_o [7:0]	Out	<p>Burst Count</p> <p>Function: These bits indicate the length of the burst transfer initiated by the DMA.</p> <ul style="list-style-type: none"> • 0000_0000: 1 beat • 0000_0001: 2 beats ... • 1111_1100: 255 beats • 1111_1111: 256 beats <p>Note that each beat is 4, 8, or 16 bytes, respectively, for 32-bit, 64-bit, or 128-bit wide buses.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>

Table 5-10 MDC Interface Signals (Continued)

Signal	I/O	Description
mdc_xfer_size_o[1:0]	Out	<p>Transfer Size</p> <p>Function: This signal is a single bit by default, and 2-bits wide when the core is configured for 128-bit data bus width and IEEE1588 time stamping functions. In such configurations, the signal indicates that the DMA has initiated the following transfer sizes:</p> <ul style="list-style-type: none"> • 2'b00 - 128-bit transfer • 2'b01 - 32-bit transfer • 2'b10 - 64-bit transfer • 2'b11 - Reserved <p>In single-bit configurations, when low, this bit indicates a 32-bit, 64-bit, or 128-bit transfer (same as data bus width) and when high, this bit indicates that a 32-bit transfer must be performed. This signal is used only for 32-bit descriptor write operations in a 64-bit or 128-bit-wide bus configuration.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
mdc_wdata_o[N – 1:0]	Out	<p>Write Data</p> <p>Function: This bus contains the data output for write transfers. The data on this bus changes at the clock cycle at which the mdc_wdata_pop_i is sampled high.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
mdc_wdata_pop_i	In	<p>Write Data Pop</p> <p>Function: This signal is asserted to get the next valid data on the mdc_wdata_o bus in the next clock cycle. The DMA updates the data bus with next valid data when this signal is asserted.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
mdc_rdata_i[N – 1:0]	In	<p>Read Data</p> <p>Function: This bus contains the data read from the application. The DMA samples the read data bus whenever mdc_rdata_push_i is asserted.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
mdc_rdata_push_i	In	<p>Read Data Valid</p> <p>Function: This signal is asserted to indicate that the application has valid data on mdc_rdata_i bus. The DMA starts a burst read transfer only when it is capable of accepting the complete burst data. Therefore, the DMA accepts the read data whenever mdc_rdata_push_i is sampled as high.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>

Table 5-10 MDC Interface Signals (Continued)

Signal	I/O	Description
mdc_xfer_done_i	In	<p>Transfer Done</p> <p>Function: This signal, when asserted (for 1 clock), indicates that the data transfer started with the assertion of mdc_start_xfer_o is complete. This signal should be asserted 1 clock after the last read data is pushed in the case of read transfer. In the case of write transfer, this signal should be asserted after the last data is accepted by the slave on the bus.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
mdc_error_i	In	<p>Transfer Error</p> <p>Function: This signal, when high (for 1 clock), indicates that an error occurred during application data transfer. The application must also terminate the transfer by asserting mdc_xfer_done_i in the next clock cycle.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
mdc_eof_o	Out	<p>End of Frame</p> <p>Function: The signal indicates that the current data on the mdc_wdata bus is an EOF data and the DMA cannot supply more data as per the requested burst length. The application must terminate the burst transfer on the receipt of this signal by asserting the mdc_xfer_done_i signal. The application must not assert mdc_wdata_pop_i after the transaction terminates.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
mdc_fixed_burst_o	Out	<p>Fixed Burst Length</p> <p>Function: This signal reflects the value of the FB (fixed burst) bit of the DMA Bus Mode register.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>

5.2.10 AHB Master Interface Signals

The AHB Master interface signals are described in [Table 5-11](#).

Table 5-11 AHB Master Interface Signals

Signal	I/O	Description
hclk_i	In	<p>AMBA AHB System Clock</p> <p>Function: This is the free-running AHB clock input provided by the Application. The DMA and the MTL modules also operate with this clock. When AHB Slave interface is selected in the GMAC-AXI configuration, hclk_i is used only in the slave CSR port.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
hreset_n	In	<p>AMBA AHB Power On System Reset</p> <p>Function: AMBA AHB power on system reset signal</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p>
hreq_o	Out	<p>AHB Request</p> <p>Function: This signal is set high by the GMAC-AHB subsystem to indicate that it requires the bus. This signal is processed by the arbiter to grant the bus.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>
hgrant_i	In	<p>AHB Grant</p> <p>Function: Indicates that the GMAC-AHB Subsystem is currently the highest priority Master. Ownership of address and control signals changes at the end of a transfer, when hready_i is high. Thus, the Subsystem gets access when both hgrant_i and hready_i are high.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p>
haddr_o[31:0]	Out	<p>AHB Address</p> <p>Function: AHB 32-bit address for the current transaction</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>

Table 5-11 AHB Master Interface Signals (Continued)

Signal	I/O	Description
htrans_o[1:0]	Out	<p>AHB Transfer Type</p> <p>Function:</p> <p>The AHB Master interface uses the following values:</p> <ul style="list-style-type: none"> • 00: IDLE • 10: NONSEQUENTIAL • 11: SEQUENTIAL <p>All other encodings are not used.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>
ysize_o[2:0]	Out	<p>AHB Data Transfer Size</p> <p>Function: Typically values used are:</p> <ul style="list-style-type: none"> • 010: Word (32 bits) • 011: DWord (64 bits) • 100: LWord (128 bits) <p>All other encodings are not used.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>
hwrite_o	Out	<p>AHB Read/Write Signal</p> <p>Function: A high hwrite_o signal indicates a write transfer; a low hwrite_o signal indicates a read transfer.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>
hburst_o[2:0]	Out	<p>AHB Burst</p> <p>Function: Indicates the transfer part of an AHB Burst:</p> <ul style="list-style-type: none"> • 000: SINGLE (Single Transfer) • 001: INCR (Incrementing burst of undefined length) • 011: INCR4 (4-beat incrementing burst) • 101: INCR8 (8-beat incrementing burst) • 111: INCR16 (16-beat incrementing burst) <p>All other encodings are not used.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>

Table 5-11 AHB Master Interface Signals (Continued)

Signal	I/O	Description
hwdata_o[N – 1:0]	Out	AHB Write Data Bus Function: hwdata_o transfers data from the GMAC-AHB subsystem to the AHB Slaves. The width of the data bus is configurable to (N=) 128, 64, or 32 (default). Active State: N/A Registered: Yes Synchronous to: hclk_i
hready_i	In	AHB Slave Ready Function: hready_i indicates that a transfer has finished on the bus. It may be driven low by the Slave being addressed to extend a transfer. Active State: High Registered: No Synchronous to: hclk_i
hresp_i[1:0]	In	AHB Slave Response Function: hresp_i provides information on the transfer status: <ul style="list-style-type: none">• 00: OKAY (Transfer completed OK)• 01: ERROR (Error in current transfer)• 10: RETRY (Slave is busy and wants Master to retry the transfer)• 11: SPLIT (Slave accepted request, but Master shall back off from bus until the Slave is ready to serve) Active State: N/A Registered: No Synchronous to: hclk_i
hrdata_i[N – 1:0]	In	AHB Slave Read Data Function: hrdata_i transfers data from the AHB Slaves to the GMAC-AHB during read operations. The width of the data bus is configurable to (N =) 128, 64, or 32 (default). Active State: N/A Registered: Yes Synchronous to: hclk_i

5.2.11 AHB Slave Interface Signals

The AHB Slave interface signals are described in [Table 5-12](#).



All signals of the AHB Slave interface are synchronous to hclk_i by default. When you select the optional clk_csr_i clock input for the Slave interface, these signals are synchronous to clk_csr_i. In addition, when you select the AHB Slave interface in the GMAC-AXI configuration, the AHB Slave interface signals are always synchronous to the hclk_i input.

Table 5-12 AHB Slave Interface Signals

Signal	I/O	Description
hsel_i	In	<p>AHB Slave Select</p> <p>Function: hsel_i is a combinatorial decode of the address bus, indicating that the GMAC-AHB Subsystem is the target for the current transfer. All reads are completed without split response for this address region.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p>
haddr_i[15:0]	In	<p>AHB Address</p> <p>Function: AHB 16-bit address for the current transaction</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p> <p>Note: The CSR uses only 8K address space, that is, haddr_i[12:0]. You can connect the unused bits of the haddr_i[15:13] to 0.</p>
htrans_i[1:0]	In	<p>AHB Transfer</p> <p>Function: Current AHB transfer type:</p> <ul style="list-style-type: none"> • 00: IDLE • 01: BUSY • 10: NONSEQUENTIAL • 11: SEQUENTIAL <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p>
hsize_i[2:0]	In	<p>AHB Transfer Data Size</p> <p>Function: Acceptable values are:</p> <ul style="list-style-type: none"> • 000: Byte (8 bits) • 001: Halfword (16 bits) • 010: Word (32 bits) <p>All other encodings are considered to be Word.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p>
hwrite_i	In	<p>AHB Read/Write</p> <p>Function: A high hwrite_i signal indicates a write transfer; a low hwrite_i signal indicates a read transfer.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p>

Table 5-12 AHB Slave Interface Signals (Continued)

Signal	I/O	Description
hburst_i[2:0]	In	<p>AHB Burst Length</p> <p>Function: Indicates the transfer part of an AHB burst:</p> <ul style="list-style-type: none"> • 000: SINGLE (Single Transfer) • 001: INCR (Incrementing burst of undefined length) • 010: WRAP4 (4-beat wrapping burst) • 011: INCR4 (4-beat incrementing burst) • 100: WRAP8 (8-beat wrapping burst) • 101: INCR8 (8-beat incrementing burst) • 110: WRAP16 (16-beat wrapping burst) • 111: INCR16 (16-beat incrementing burst) <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p> <p>Note: Wrap burst is not supported with this signal.</p>
hwdata_i[K – 1:0]	In	<p>AHB Write Data</p> <p>Function: AHB write data input to the Slave port. The data bus width is configurable (K = 32, 64 or 128).</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>
hreadyslv_i	In	<p>AHB Master/Slave Ready</p> <p>Function: The hready of the AHB bus input to all master and slave ports</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p>
hready_o	Out	<p>AHB Slave Ready</p> <p>Function: The GMAC-AHB Subsystem Slave indicates that the current transfer has finished.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>
hresp_o[1:0]	Out	<p>AHB Slave Response</p> <p>Function: hresp_o provides transfer status information. The valid response is:</p> <p>00: OKAY: Transfer completed OK</p> <p>All other encodings are not used.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>

Table 5-12 AHB Slave Interface Signals (Continued)

Signal	I/O	Description
hrdata_o[K – 1:0]	Out	<p>AHB Slave Read Data</p> <p>Function: AHB slave read data output from the Slave port. The data bus width is configurable (K = 32, 64 or 128).</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i</p>
sbd_intr_o	Out	<p>GMAC-AHB Subsystem Interrupt</p> <p>Function: This signal, when high, indicates an interrupt event to the application. This signal is available in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: hclk_i</p>

5.2.12 AXI Master Interface Signals

The AHB Master interface signals are described in [Table 5-13](#).

Table 5-13 AXI Master Interface Signals

Signal	I/O	Description
aclk_i	In	<p>AMBA AXI System Clock</p> <p>Function: This is the free-running AXI clock input provided by the Application. The DMA and the MTL modules also operate with this clock.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
areset_n	In	<p>AMBA AXI Power On System Reset</p> <p>Function: AMBA AXI power on system reset signal</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
awid_m_o[MW-1:0]	Out	<p>Master Write Address ID</p> <p>Function: This signal is the identification tag for the write address group of signals. The width is configured during RTL configuration in coreConsultant. The minimum value of MW is 1 and the maximum value is 12.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>

Table 5-13 AXI Master Interface Signals (Continued)

Signal	I/O	Description
awaddr_m_o[31:0]	Out	<p>Master Write Address</p> <p>Function: The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
awlen_m_o[BL-1:0]	Out	<p>Master Burst Length</p> <p>Function: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. The default width is 4 but can be increased to support up to 256-beat transfers during RTL configuration in coreConsultant. The value of all-zeros indicates a single transfer. The minimum value of BL is 4 and the maximum value is 8.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
awszie_m_o[2:0]	Out	<p>Master Burst Size</p> <p>Function: This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update. This is always driven to have 'System Datawidth' transfers. 3'b010 for 32-bit, 3'b011 for 64-bit and 3'b100 for 128-bit interface.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
awburst_m_o[1:0]	Out	<p>Master Burst Type</p> <p>Function: The burst type, along with the size information, specifies how the address for each transfer within the burst is calculated. This is always hard wired to 2'b01 to indicate INCR burst type.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
awlock_m_o[1:0]	Out	<p>Master Lock Type</p> <p>Function: This signal provides additional information about the atomic characteristics of the transfer. This is always hard wired to 2'b00 to indicate normal access.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>

Table 5-13 AXI Master Interface Signals (Continued)

Signal	I/O	Description
awcache_m_o[3:0]	Out	<p>Master Cache Type</p> <p>Function: This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction. This is always hard wired to 4'b0000 to indicate non-cacheable and non-bufferable data access.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
awprot_m_o[2:0]	Out	<p>Master Protection Type</p> <p>Function: This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. This is always hard wired to 3'b000 to indicate normal and non-secure data access.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
awvalid_m_o	Out	<p>Master Write Address Valid</p> <p>Function: This signal indicates that valid write address and control information are available:</p> <ul style="list-style-type: none"> • 1: address and control information available • 0: address and control information not available <p>The address and control information remain stable until the address acknowledge signal, awready, goes high.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
awready_m_i	In	<p>Master Write Address Ready</p> <p>Function: This signal indicates that the AXI slave is ready to accept an address and associated control signals:</p> <ul style="list-style-type: none"> • 1: Slave ready • 0: Slave not ready <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
wid_m_o[MW-1:0]	Out	<p>Master Write ID Tag</p> <p>Function: This signal is the ID tag of the write data transfer. The WID value must match the AWID value of the write transaction. Master Write takes the value of parameter AXI_GM_AXI_ID_WIDTH.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>

Table 5-13 AXI Master Interface Signals (Continued)

Signal	I/O	Description
wdata_m_o[N-1:0]	Out	<p>Master Write Data</p> <p>Function: The write data bus can be 32-bit, 64-bit, or 128-bit wide, and takes the value of parameter "System Data Width" (DATAWIDTH).</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
wstrb_m_o[N/8-1:0]	Out	<p>Master Write Strobes</p> <p>Function: This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
wlast_m_o	Out	<p>Master Write Last</p> <p>Function: This signal indicates the last transfer in a write burst.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
wvalid_m_o	Out	<p>Master Write Valid</p> <p>Function: This signal indicates that valid write data and strobes are available:</p> <ul style="list-style-type: none"> • 1: write data and strobes available • 0: write data and strobes not available <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
wready_m_i	In	<p>Master Write Ready</p> <p>Function: This signal indicates that the Slave accepted the write data:</p> <ul style="list-style-type: none"> • 1: Slave accepted • 0: Slave has not accepted <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
bid_m_i[MW-1:0]	In	<p>Master Response ID</p> <p>Function: The identification tag of the write response. The BID value must match the AWID value of the write transaction to which the slave is responding. The width of this signal is configured during the RTL configuration.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>

Table 5-13 AXI Master Interface Signals (Continued)

Signal	I/O	Description
bresp_m_i[1:0]	In	<p>Master Write Response</p> <p>Function: This signal indicates the status of the write transaction. The valid responses are OKAY(2'b00), EXOKAY(2'b01), SLVERR(2'b10), and DECERR(2'b11).</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
bvalid_m_i	In	<p>Master Write Response Valid</p> <p>Function: This signal indicates that a valid write response is available:</p> <ul style="list-style-type: none"> • 1: write response available • 0: write response not available <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
bready_m_o	Out	<p>Master Response Ready</p> <p>Function: This signal indicates that the master can accept the response information.</p> <ul style="list-style-type: none"> • 1: master ready • 0: master not ready <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
arid_m_o[MW:0]	Out	<p>Master Read Address ID</p> <p>Function: This signal is the identification tag for the read address group of signals. The width of this signal is configured during RTL configuration.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
araddr_m_o[31:0]	Out	<p>Master Read Address</p> <p>Function: The read address bus gives the initial address of a read burst transaction. It provides the start address of the burst and the control signals that are issued with the address. The control signals specify how the address is calculated for the remaining transfers in the burst.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>

Table 5-13 AXI Master Interface Signals (Continued)

Signal	I/O	Description
arlen_m_o[BL-1:0]	Out	<p>Master Burst Length</p> <p>Function: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. The default width is 4 but width of this signal can be increased during RTL configuration to support bigger bursts.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
arsize_m_o[2:0]	Out	<p>Master Burst Size</p> <p>Function: This signal indicates the size of each transfer in the burst. This is always driven to have 'System Datawidth' transfers such as 3'b010 for 32-bit, 3'b011 for 64-bit, and 3'b100 for 128-bit interface.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
arburst_m_o[1:0]	Out	<p>Master Burst Type</p> <p>Function: The burst type, along with the size information, specifies how the address for each transfer within the burst is calculated. This is always hard wired to 2'b01 to indicate INCR burst type.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
arlock_m_o[1:0]	Out	<p>Lock Type</p> <p>Function: This signal provides additional information about the atomic characteristics of the transfer. This is always hard wired to 2'b00 to indicate normal access.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
arcache_m_o[3:0]	Out	<p>Master Cache Type</p> <p>Function: This signal provides additional information about the cacheable characteristics of the transfer. This is always hard wired to 4'b0000 to indicate noncacheable and nonbufferable data access.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
arprot_m_o[2:0]	Out	<p>Master Protection Type</p> <p>Function: This signal provides protection unit information for the transaction. This is always hard wired to 3'b000 to indicate normal and nonsecure data access.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>

Table 5-13 AXI Master Interface Signals (Continued)

Signal	I/O	Description
arvalid_m_o	Out	<p>Master Read Address Valid</p> <p>Function: When HIGH, this signal indicates that the read address and control information is valid. This signal remains stable until the address acknowledge signal ARREADY is high.</p> <ul style="list-style-type: none"> • 1: address and control information valid • 0: address and control information not valid <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
arready_m_i	In	<p>Master Read Ready</p> <p>This signal indicates that the Slave can accept the read request:</p> <ul style="list-style-type: none"> • 1: Slave ready • 0: Slave not ready <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rid_m_i[MW-1:0]	In	<p>Master Read ID Tag</p> <p>Function: This signal is the ID tag of the read data group of signals. The rid value is generated by the Slave and must match the arid value of the read transaction to which it is responding.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rdata_m_i[N-1:0]	In	<p>Master Read Data</p> <p>Function: The read data bus can be 32-bit, 64-bit, or 128-bit wide and takes the value of parameter "System Data width" (DATAWIDTH).</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rresp_m_i	In	<p>Master Read Response</p> <p>Function: This signal indicates the status of the read transfer. The valid responses are OKAY(2'b00), EXOKAY(2'b01), SLVERR(2'b10), and DECERR(2'b11).</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rlast_m_i	In	<p>Master Read Last</p> <p>Function: This signal indicates the last transfer in a read burst.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>

Table 5-13 AXI Master Interface Signals (Continued)

Signal	I/O	Description
rvalid_m_i	In	<p>Master Read Valid</p> <p>Function: This signal indicates that the required read data is available and the read transfer can complete:</p> <ul style="list-style-type: none"> • 1: read data available • 0: read data not available <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rready_m_o	Out	<p>Master Read Ready</p> <p>This signal indicates that the master can accept the read data and response information:</p> <ul style="list-style-type: none"> • 1: master ready • 0: master not ready <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>

5.2.13 AXI Low Power Interface Signals

The AXI Low Power Interface interface signals are described in [Table 5-14](#).



The timing and behavior of the GMAC-AXI low-power interface signals is according to the Figure 12-2 of the [AMBA 3 Specification](#).

Table 5-14 AXI Low Power Interface Signals

Signal	I/O	Description
csysreq_i	In	<p>Clock Controller System low-power request.</p> <p>When asserted low, this signal indicates that the system clock controller has requested the GMAC-AXI to enter a low-power state. When asserted high, this signal indicates that the GMAC-AXI has to come out of low-power state.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
csysack_o	Out	<p>Low-Power Request Acknowledgement</p> <p>This signal is the acknowledgement from the GMAC-AXI to the system for low-power request.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>

Table 5-14 AXI Low Power Interface Signals

Signal	I/O	Description
cactive_o	Out	<p>Clock Active</p> <p>This signal indicates that the GMAC-AXI requires its clock signal:</p> <ul style="list-style-type: none"> • 1: GMAC-AXI clock required • 0: GMAC-AXI clock not required <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to:</p> <ul style="list-style-type: none"> - clk_rx_i when self-initiating request to come out of low-power mode - aclk_i in all other cases.

5.2.14 AXI Slave Interface Signals

The AXI Slave interface signals are described in [Table 5-15](#).

Table 5-15 AXI Slave Interface Signals

Signal	I/O	Description
awid_s_i [GS_ID-1:0]	In	<p>Slave Write Address ID</p> <p>Function: This signal is the identification tag for the write address group of signals.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
awaddr_s_i[31:0]	In	<p>Slave Write Address</p> <p>Function: The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
awlen_s_i[3:0]	In	<p>Slave Burst Length</p> <p>Function: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
awsizer_s_i[2:0]	In	<p>Slave Burst Size</p> <p>Function: This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>

Table 5-15 AXI Slave Interface Signals (Continued)

Signal	I/O	Description
awburst_s_i[1:0]	In	<p>Slave Burst Type</p> <p>Function: The burst type, along with the size information, specifies how the address for each transfer within the burst is calculated.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
awlock_s_i[1:0]	In	<p>Slave Lock Type</p> <p>Function: This signal provides additional information about the atomic characteristics of the transfer. GMAC AXI Slave does not support Exclusive access.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
awcache_s_i[3:0]	In	<p>Slave Cache Type</p> <p>Function: This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction. The core does not use this input as it supports only normal data transactions.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
awprot_s_i[2:0]	In	<p>Slave Protection Type</p> <p>Function: This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. The core does not use this input as it supports only normal data transactions.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
awvalid_s_i	In	<p>Slave Write Address Valid</p> <p>Function: This signal indicates that valid write address and control information are available:</p> <ul style="list-style-type: none"> • 1: address and control information available • 0: address and control information not available <p>The address and control information remain stable until the address acknowledge signal, awready_s_o, goes high.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>

Table 5-15 AXI Slave Interface Signals (Continued)

Signal	I/O	Description
awready_s_o	Out	<p>Slave Write Address Ready</p> <p>Function: This signal indicates that the slave is ready to accept an address and associated control signals:</p> <ul style="list-style-type: none"> • 1: slave ready • 0: slave not ready <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
wid_s_i [GS_ID-1:0]	In	<p>Slave Write ID Tag</p> <p>Function: This signal is the ID tag of the write data transfer. The wid value must match the awid value of the write transaction. This input is not used as the AXI slave does not accept more than one write request at a time.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
wdata_s_i[N-1:0]	In	<p>Slave Write Data</p> <p>Function: The write data bus can be 32-bit, 64-bit, or 128-bit wide. It takes the value of the "System Bus Width" parameter (DATAWIDTH).</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
wstrb_s_i[N/8-1:0]	In	<p>Slave Write Strobes</p> <p>Function: This signal indicates the byte lanes that are updated in the memory. One write strobe is equal to the eight bits of the write data bus.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
wlast_s_i	In	<p>Slave Write Last</p> <p>Function: This signal indicates the last transfer in a write burst. This input is not used as the AXI master always provides the input about the required number of the write data transfers to the slave.</p> <p>Active State: High</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>
wvalid_s_i	In	<p>Slave Write Valid</p> <p>Function: This signal indicates that valid write data and strobes are available:</p> <ul style="list-style-type: none"> • 1: write data and strobes available • 0: write data and strobes not available <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>

Table 5-15 AXI Slave Interface Signals (Continued)

Signal	I/O	Description
wready_s_o	Out	<p>Slave Write Ready</p> <p>Function: This signal indicates that the slave can accept the write data:</p> <ul style="list-style-type: none"> • 1: slave ready • 0: slave not ready <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
bid_s_o [GS_ID-1:0]	Out	<p>Slave Response ID</p> <p>Function: The identification tag of the write response. The bid value is equal to the awid value of the write transaction to which the slave is responding.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
bresp_s_o[1:0]	Out	<p>Slave Write Response</p> <p>Function: This signal indicates the status of the write transaction. The valid responses are OKAY(2'b00), EXOKAY(2'b01), SLVERR(2'b10), and DECERR(2'b11).</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
bvalid_s_o	Out	<p>Slave Write Response valid</p> <p>Function: This signal indicates that a valid write response is available:</p> <ul style="list-style-type: none"> • 1: write response available • 0: write response not available <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
bready_s_i	In	<p>Slave Response Ready</p> <p>Function: This signal indicates that the master can accept the response information.</p> <ul style="list-style-type: none"> • 1: slave ready • 0: slave not ready <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
arid_s_i [GS_ID-1:0]	In	<p>Slave Read Address ID</p> <p>Function: This signal is the identification tag for the read address group of signals.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>

Table 5-15 AXI Slave Interface Signals (Continued)

Signal	I/O	Description
araddr_s_i[31:0]	In	<p>Slave Read Address</p> <p>Function: The read address bus gives the initial address of a read burst transaction. It provides the start address of the burst and the control signals that are issued with the address. The control signals specify how the address is calculated for the remaining transfers in the burst.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
arlen_s_i[3:0]	In	<p>Slave Burst Length</p> <p>Function: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
arsize_s_i[2:0]	In	<p>Slave Burst Size</p> <p>Function: This signal indicates the size of each transfer in the burst.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
arburst_s_i[1:0]	In	<p>Slave Burst Type</p> <p>Function: The burst type, along with the size information, specifies how the address for each transfer within the burst is calculated.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
arlock_s_i[1:0]	In	<p>Lock Type</p> <p>Function: This signal provides additional information about the atomic characteristics of the transfer.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
arcache_s_i[3:0]	In	<p>Master Cache Type</p> <p>Function: This signal provides additional information about the cacheable characteristics of the transfer. The core does not use this input as it supports only normal data transactions.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: aclk_i</p>

Table 5-15 AXI Slave Interface Signals (Continued)

Signal	I/O	Description
arprot_s_i[2:0]	In	<p>Slave Protection Type</p> <p>Function: This signal provides protection unit information for the transaction. The GMAC AXI Slave does not support Exclusive access. The core does not use this input as it supports only normal data transactions.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: aclk_i</p>
arvalid_s_i	In	<p>Slave Read Address Valid</p> <p>Function: When high, this signal indicates that the read address and control information is valid. This signal remains stable until the address acknowledge signal, arready, is high.</p> <ul style="list-style-type: none"> • 1: address and control information valid • 0: address and control information not valid <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
arready_s_i	In	<p>Slave Read Address Ready</p> <p>Function: This signal indicates that the slave is ready to accept an address and associated control signals:</p> <ul style="list-style-type: none"> • 1: slave ready • 0: slave not ready <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rid_s_o [GS_ID-1:0]	Out	<p>Slave Read ID Tag</p> <p>Function: This signal is the ID tag of the read data group of signals. The rid value is generated by the slave. The rid value is equal to the arid value of the read transaction to which it is responding.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rdata_s_o[N-1:0]	Out	<p>Slave Read Data</p> <p>Function: The read data bus can be 32-bit, 64-bit, or 128-bit wide. It takes the value of 'System Data Bus width' parameter (DATAWIDTH).</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>

Table 5-15 AXI Slave Interface Signals (Continued)

Signal	I/O	Description
rresp_s_o[1:0]	Out	<p>Slave Read Response</p> <p>Function: This signal indicates the status of the read transfer. The valid responses are OKAY(2'b00), EXOKAY(2'b01), and SLVERR(2'b10).</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rlast_s_o	Out	<p>Slave Read Last</p> <p>Function: This signal indicates the last transfer in a read burst.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rvalid_s_o	Out	<p>Slave Read Valid</p> <p>Function: This signal indicates that the required read data is available and the read transfer can complete:</p> <ul style="list-style-type: none"> • 1: read data available • 0: read data not available <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>
rready_s_i	In	<p>Slave Read Ready</p> <p>This signal indicates that the slave can accept the read data and response information:</p> <ul style="list-style-type: none"> • 1: AXI Master ready • 0: Master not ready <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: aclk_i</p>

5.2.15 External DPRAM Interface Signals

The External DPRAM interface signals are described in [Table 5-16](#). When you select the AV feature, these signals correspond to channel 0 in Tx and Rx paths.



Note The External DPRAM interface signals on the application side are synchronous to hclk_i in the GMAC-AHB configuration, aclk_i in the GMAC-AXI configuration, and clk_app_i in the GMAC-MTL and GMAC-DMA configurations.

Table 5-16 External DPRAM Interface Signals

Signal	I/O	Description																																
twc_wr_addr_o[X:0]	Out	<p>Transmit FIFO Write Address</p> <p>Function: This signal gives the address to the Tx FIFO write port. It is valid when twc_wr_csn_o is asserted. The width of the bus depends on the Tx FIFO depth and width. For example, as shown in the following table, for a Tx FIFO depth of 2,048 bytes and a data bus width of 64 bits, X is equal to 7.</p> <table border="1"> <thead> <tr> <th>Tx FIFO Depth</th> <th>32-Bits Data Bus Width</th> <th>64-Bits Data Bus Width</th> <th>128-Bits Data Bus Width</th> </tr> </thead> <tbody> <tr><td>256</td><td>5</td><td>4</td><td>3</td></tr> <tr><td>512</td><td>6</td><td>5</td><td>4</td></tr> <tr><td>1024</td><td>7</td><td>6</td><td>5</td></tr> <tr><td>2048</td><td>8</td><td>7</td><td>6</td></tr> <tr><td>4096</td><td>9</td><td>8</td><td>7</td></tr> <tr><td>8192</td><td>10</td><td>9</td><td>8</td></tr> <tr><td>16384</td><td>11</td><td>10</td><td>9</td></tr> </tbody> </table> <p>The address width is displayed in the coreConsultant after you configure the Tx FIFO depth. The MTL TX FIFO Address-bus Width box of the Buffer Management dialog box displays the address width.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>	Tx FIFO Depth	32-Bits Data Bus Width	64-Bits Data Bus Width	128-Bits Data Bus Width	256	5	4	3	512	6	5	4	1024	7	6	5	2048	8	7	6	4096	9	8	7	8192	10	9	8	16384	11	10	9
Tx FIFO Depth	32-Bits Data Bus Width	64-Bits Data Bus Width	128-Bits Data Bus Width																															
256	5	4	3																															
512	6	5	4																															
1024	7	6	5																															
2048	8	7	6																															
4096	9	8	7																															
8192	10	9	8																															
16384	11	10	9																															
twc_wr_data_o[Y:0]	Out	<p>Transmit FIFO Write Data</p> <p>Function: This bus carries the data to be written in the Tx FIFO. It is valid when twc_wr_csn_o and tfc_wr_en_o are asserted. The width of this data bus (Y = 34, 67, or 132) depends on the data bus width (32, 64, or 128 respectively) selected during configuration. The additional bits (3, 4, or 5 in 32-bit, 64-bit, or 128-bit data) are used for storing the tag or byte enable bits for the valid data lanes.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>																																
twc_wr_en_o	Out	<p>Transmit FIFO Write Enable</p> <p>Function: When high, indicates that twc_wr_data_o should be written into memory.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>																																
twc_wr_csn_o	Out	<p>Transmit FIFO Write Port Chip Select</p> <p>Function: Qualifies the twc_wr_addr_o, twc_wr_data_o, and twc_wr_en_o signals.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>																																

Table 5-16 External DPRAM Interface Signals (Continued)

Signal	I/O	Description
trc_rd_addr_o[X:0]	Out	<p>Transmit FIFO Read Address</p> <p>Function: This gives the address to the Tx FIFO read port. It is valid when trc_rd_csn_o is asserted. The width of address bus depends on the Tx FIFO Depth and the data bus width.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>
trc_rd_data_i[Y:0]	In	<p>Transmit FIFO Read Data</p> <p>Function: This bus carries the data read from the Tx FIFO. It is valid when trc_rd_csn_o and trc_rd_en_o are asserted. The width of the data bus (Y = 34, 67, or 132) depends on the data bus width (32, 64, or 128, respectively) selected during configuration.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
trc_rd_en_o	Out	<p>Transmit FIFO Read Enable</p> <p>Function: When high, indicates that the trc_rd_data_i should contain valid data.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>
trc_rd_csn_o	Out	<p>Transmit FIFO Read Port Chip Select</p> <p>Function: Qualifies the trc_rd_addr_o signal.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
rwc_wr_addr_o[Z:0]	Out	<p>Receive FIFO Write Address</p> <p>Function: This gives the address to the Rx FIFO write port. It is valid when rwc_wr_csn_o is asserted. The width of address bus depends on the Rx FIFO depth and the data bus width. For example, for an Rx FIFO depth of 2,048 bytes and a data bus width of 32-bits, Z is equal to 8. For an Rx FIFO depth of 8192 bytes and a data bus width of 64-bits, Z is equal to 9.</p> <p>The address width is displayed in the coreConsultant after you configure the Rx FIFO depth. The MTL RX FIFO Address-bus Width box of the Buffer Management dialog box displays the address width.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>

Table 5-16 External DPRAM Interface Signals (Continued)

Signal	I/O	Description
rwc_wr_data_o[Y:0]	Out	<p>Receive FIFO Write Data</p> <p>Function: This bus carries the data to be written in the Rx FIFO. It is valid when rwc_wr_csn_o and rwc_wr_en_o are asserted. The width of the FIFO write data bus is 35 (Y=34), 68 (Y=67), or 133(Y=132) depending on the data bus width (32, 64, or 128, respectively) selected during the coreConsultant configuration. The additional bits (3, 4, or 5 in 32-bit, 64-bit, or 128-bit data) are used for storing the tag or byte enable bits for the valid data lanes.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
rwc_wr_en_o	Out	<p>Receive FIFO Write Enable</p> <p>Function: When high, indicates that the rwc_wr_data_o should be written into memory</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_i</p>
rwc_wr_csn_o	Out	<p>Receive FIFO Write Port Chip Select</p> <p>Function: Qualifies the rwc_wr_addr_o, rwc_wr_data_o, and rwc_wr_en_o signals.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_i</p>
rrc_rd_addr_o[Z:0]	Out	<p>Receive FIFO Read Address</p> <p>Function: This gives the address to the Rx FIFO read port. It is valid when rrc_rd_csn_o is asserted. The width of address bus depends on the Rx FIFO Depth and the data bus width.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/clk_app_i</p>
rrc_rd_data_i[Y:0]	In	<p>Receive FIFO Read Data</p> <p>Function: This bus carries the data read from the Rx FIFO. It is valid when rrc_rd_csn_o and rrc_rd_en_o are asserted. The width of the data bus (Y = 34, 67, or 132) depends on the data bus width (32, 64, or 128, respectively) selected during the coreConsultant configuration.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>
rrc_rd_en_o	Out	<p>Receive Read Enable</p> <p>Function: When high, indicates that rrc_rd_data_i should contain valid data.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>

Table 5-16 External DPRAM Interface Signals (Continued)

Signal	I/O	Description
rrc_rd_csn_o	Out	<p>Receive Read Qualification</p> <p>Function: Qualifies the rrc_rd_addr_o signal.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>

5.2.16 External DPRAM Interface Signals for Channel 1/Channel 2

[Table 5-17](#) describes the External DPRAM interface signals for channel 1. These signals are available when you enable the AV Feature. The twc_* and trc_* signals are available when you select an additional Tx channel. The rwc_* and rrc_* signals are available only when you select an additional Rx channel.

For additional channels, the TxFIFO and RxFIFO depth is similar to the depth configured for channel 0. Therefore, in [Table 5-16](#), the corresponding address bus width for additional channels is similar to the address bus width of channel 0.



The external DPRAM interface signals for channel 2 are similar to the signals for channel 1. For channel 2, ch1 in the signal name gets replaced by ch2. For example, for channel 1, the signal name is trc_ch1_rd_csn_o and for channel 2, the signal name is trc_ch2_rd_csn_o.

Table 5-17 External DPRAM Interface Signals for Channel 1

Signal	I/O	Description
tvc_ch1_wr_addr_o[X:0]	Out	<p>Transmit FIFO Write Address</p> <p>Function: This signal gives the address to the Tx FIFO write port. It is valid when tvc_ch1_wr_csn_o is asserted. The width of the bus depends on the Tx FIFO depth and width. For example, for a Tx FIFO depth of 2,048 bytes and a data bus width of 64 bits, X is equal to 6.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>
tvc_ch1_wr_data_o[Y:0]	Out	<p>Transmit FIFO Write Data</p> <p>Function: This bus carries the data to be written in the Tx FIFO. It is valid when tvc_ch1_wr_csn_o and tvc_ch1_wr_en_o are asserted. The width of the data bus (Y = 34, 67, or 132) depends on the data bus width (32, 64, or 128 respectively) selected during the coreConsultant configuration.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>

Table 5-17 External DPRAM Interface Signals for Channel 1

Signal	I/O	Description
twc_ch1_wr_en_o	Out	<p>Transmit FIFO Write Enable</p> <p>Function: When high, indicates that the twc_ch1_wr_data_o should be written into memory.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>
twc_ch1_wr_csn_o	Out	<p>Transmit FIFO Write Port Chip Select</p> <p>Function: Qualifies the twc_ch1_wr_addr_o, twc_ch1_wr_data_o, and twc_ch1_wr_en_o signals.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>
trc_ch1_rd_addr_o[X:0]	Out	<p>Transmit FIFO Read Address</p> <p>Function: This signal gives the address to the Tx FIFO read port. It is valid when trc_ch1_rd_csn_o is asserted. The width of address bus depends on the Tx FIFO Depth and the data bus width.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>
trc_ch1_rd_data_i[Y:0]	In	<p>Transmit FIFO Read Data</p> <p>Function: This bus carries the data read from the Tx FIFO. It is valid when trc_ch1_rd_csn_o and trc_ch1_rd_en_o are asserted. The width of the data bus (Y = 34, 67, or 132) depends on the data bus width (32, 64, or 128, respectively) selected during the coreConsultant configuration.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
trc_ch1_rd_en_o	Out	<p>Transmit FIFO Read Enable</p> <p>Function: When high, indicates that the trc_ch1_rd_data_i should contain valid data.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>
trc_ch1_rd_csn_o	Out	<p>Transmit FIFO Read Port Chip Select</p> <p>Function: Qualifies the trc_ch1_rd_addr_o signal.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>

Table 5-17 External DPRAM Interface Signals for Channel 1

Signal	I/O	Description
rwc_ch1_wr_addr_o[Z:0]	Out	<p>Receive FIFO Write Address</p> <p>Function: This gives the address to the Rx FIFO write port. It is valid when rwc_ch1_wr_csn_o is asserted. The width of address bus depends on the Rx FIFO depth and the data bus width. For example, for an Rx FIFO depth of 2,048 bytes and a data bus width of 32-bits, Z is equal to 8. For an Rx FIFO depth of 8192 bytes and a data bus width of 64-bits, Z is equal to 9.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
rwc_ch1_wr_data_o[Y:0]	Out	<p>Receive FIFO Write Data</p> <p>Function: This bus carries the data to be written in the Rx FIFO. It is valid when rwc_ch1_wr_csn_o and rwc_ch1_wr_en_o are asserted. The width of the data bus (Y = 34, 67, or 132) depends on the data bus width (32, 64, or 128, respectively) selected during the coreConsultant configuration.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
rwc_ch1_wr_en_o	Out	<p>Receive FIFO Write Enable</p> <p>Function: When high, indicates that the rwc_ch1_wr_data_o should be written into the memory.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_i</p>
rwc_ch1_wr_csn_o	Out	<p>Receive FIFO Write Port Chip Select</p> <p>Function: Qualifies the rwc_ch1_wr_addr_o, rwc_ch1_wr_data_o, and rwc_ch1_wr_en_o signals.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_i</p>
rrc_ch1_rd_addr_o[Z:0]	Out	<p>Receive FIFO Read Address</p> <p>Function: This gives the address to the Rx FIFO read port. It is valid when rrc_ch1_rd_csn_o is asserted. The width of the address bus depends on the Rx FIFO depth and the data bus width.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>

Table 5-17 External DPRAM Interface Signals for Channel 1

Signal	I/O	Description
rrc_ch1_rd_data_i[Y:0]	In	<p>Receive FIFO Read Data</p> <p>Function: This bus carries the data read from the Rx FIFO. It is valid when rrc_ch1_rd_csn_o and rrc_ch1_rd_en_o are asserted. The width of the data bus (Y = 34, 67, or 132) depends on the data bus width (32, 64, or 128, respectively) selected during the coreConsultant configuration.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>
rrc_ch1_rd_en_o	Out	<p>Receive Read Enable</p> <p>Function: When high, indicates that rrc_ch1_rd_data_i should contain valid data.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>
rrc_ch1_rd_csn_o	Out	<p>Receive Read Qualification</p> <p>Function: Qualifies the rrc_ch1_rd_addr_o signal.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i</p>

5.2.17 External DPRAM (Frame Length FIFO) Interface Signals

The External DPRAM Frame Length FIFO interface signals are described in [Table 5-18](#). These signals can be configured to be active only in the GMAC-MTL configuration (refer to [Table 7-7](#).)

Table 5-18 External DPRAM Interface Signals

Signal	I/O	Description
rwc_len_wr_addr_o[V:0]	Out	<p>Receive Frame Length FIFO Write Address</p> <p>Function: Gives the address to the Rx Frame Length FIFO write port. It is valid when rwc_len_wr_csn_o is asserted. The width of the address bus depends on the maximum number of frames stored in the Rx FIFO. For example, for an Rx FIFO depth of 2,048 bytes and a minimum receive frame size of 16 bytes, the maximum number of frames that can be stored in the Rx FIFO is 128. Therefore, V is equal to 6 for this configuration.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>

Table 5-18 External DPRAM Interface Signals (Continued)

Signal	I/O	Description
rwc_len_wr_en_o	Out	<p>Receive Frame Length FIFO Write Enable</p> <p>Function: When high, indicates that rwc_wr_data_o[Y + 16:16] should be written into DPRAM, where Y = Width of the Frame Length FIFO. Note that the rwc_wr_data_o is the same data bus written to the MTL Receive FIFO.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_i</p>
rwc_err_frame_o	Out	<p>Frame Error Status</p> <p>Function: When high, indicates that the frame whose status is being written is an error frame and the frame should be dropped by the Rx FIFO Read Controller. This bit should be concatenated as MSB to the frame length data (rwc_wr_data_o[Y+16:16]), written to the Rx Frame Length FIFO.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
rwc_len_wr_csn_o	Out	<p>Receive Frame Length FIFO Write Port Chip Select</p> <p>Function: Indicates that the rwc_len_wr_addr_o, rwc_wr_data_o, and rwc_len_wr_en_o signals have valid values.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_i</p>
rrc_len_rd_addr_o[V:0]	Out	<p>Receive Frame Length FIFO Read Address</p> <p>Function: This gives the address to the Rx Frame Length FIFO read port. It is valid when rrc_len_rd_csn_o is asserted. The width of this port is same as rwc_len_wr_addr_o.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
rrc_len_rd_data_i [W + 1:0]	In	<p>Receive FIFO Read Data</p> <p>Function: This bus carries the data read from the Frame Length FIFO. It is valid when rrc_len_rd_csn_o and rrc_len_rd_en_o are asserted. You can select the width of the data bus during coreConsultant configuration. The width of the data bus can be from 12 bits to 15 bits (W = 10 to 13). The MS bit gives the frame-error status while the other bits indicate the length of the frame stored in the top of Rx FIFO.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>

Table 5-18 External DPRAM Interface Signals (Continued)

Signal	I/O	Description
rrc_len_rd_en_o	Out	<p>Receive Read Enable</p> <p>Function: When high, indicates that rrc_rd_data_i contains valid data.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
rrc_len_rd_csn_o	Out	<p>Receive Read Qualification</p> <p>Function: Qualifies the rrc_len_rd_addr_o signal.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
ari_frame_len_o[W:0]	Out	<p>Receive Frame Length</p> <p>Function: This bus carries the frame length of the received frame at the top of the MTL Rx FIFO. Frame length is valid when ari_frame_len_val_o is asserted. You can select the width (W) of ari_frame_len_o during coreConsultant configuration.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ari_frame_len_val_o	Out	<p>Receive Frame Length Qualifier</p> <p>Function: Qualifies the ari_frame_len_o signal.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
ari_err_frame_o	Out	<p>Frame Error Status</p> <p>Function: When high, this signal indicates that the frame being transferred on the ARI is an error frame and can be dropped by the application. This signal has a valid value when ari_frame_len_val_o is asserted.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>

5.2.18 RGMII/RTBI Interface Signals (Optional)

The optional RGMII/RTBI interface signals are described in [Table 5-19](#).

Table 5-19 RGMII/RTBI Interface Signals (Optional)

Signal	I/O	Description
clk_tx_180_i	In	<p>Transmit Clock</p> <p>Function: The RGMII transmit interface uses the rising edge of clk_tx_180_i to represent the falling edge of clk_tx_i. This clock input is not used in the RTBI interface</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: clk_tx_i</p>
clk_rx_180_i	In	<p>Receive Clock</p> <p>Function: The RGMII receive interface uses the positive edge of clk_rx_180_i to represent the falling edge of clk_rx_i. The RTBI receive interface uses the positive edge of clk_rx_180_i to represent the falling edge of clk_rx_125_i.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: clk_rx_i</p>
rst_clk_tx_180_n	In	<p>Transmit Clock Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_180_i</p>
rst_clk_rx_180_n	In	<p>Receive Clock Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_180_i</p>

5.2.19 RMII Interface Signals (Optional)

The optional RMII interface signals are described in [Table 5-20](#).

Table 5-20 RMII Interface Signals (Optional)

Signal	I/O	Description
clk_rmii_i	In	<p>RMII Clock</p> <p>Function: This is the 50-MHz clock used by the RMII interface. clk_rx_i should be 25/2.5 MHz, derived from this clock when the RMII interface is selected.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>

Table 5-20 RMII Interface Signals (Optional) (Continued)

Signal	I/O	Description
rst_clk_rmii_n	In	<p>RMII Clock Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_rmii_i</p>
mac_speed_i	In	<p>MAC Speed</p> <p>Function: When high, indicates 100-Mbps operation. When low, indicates 10-Mbps operation. If you choose the output signal (mac_speed_o) during configuration, this input pin is removed.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: N/A</p>

5.2.20 RevMII Interface Signal (Optional)

The RevMII interface signals are described in [Table 5-21](#). When the CSR slave port is synchronous to the respective system clock, all signals, synchronous to clk_csr_i clock in [Table 5-21](#), become synchronous to the aclk_i/hclk_i/clk_app_i clocks.

Table 5-21 RevMII Interface Signal (Optional)

Signal	I/O	Description
clk_revmii_tx_i	In	<p>Transmit Clock</p> <p>Function: This clock signal is used to register the input data at the RevMII-Remote MAC Interface (phy_rxd_i, phy_rxrv_i, and phy_rxe_i) before sending it to the data mux.</p> <p>For more information about this signal, see “Clocks With RevMII” on page 434.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
rst_clk_revmii_tx_n	In	<p>RevMII Transmit Clock Asynchronous Reset</p> <p>Function: This is an active-low reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, or GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_revmii_tx_i</p>

Table 5-21 RevMII Interface Signal (Optional)

Signal	I/O	Description
clk_revmii_rx_i	In	<p>Receive Clock</p> <p>Function: This clock signal is used to register output data at the RevMII-Remote MAC interface (phy_txd_o, phy_txen_o, and phy_txer_o) before sending it to the remote MAC.</p> <p>For more information about this signal, see “Clocks With RevMII” on page 434.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
rst_clk_revmii_rx_n	In	<p>RevMII Receive Clock Asynchronous Reset</p> <p>Function: This is an active-low reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, or GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_revmii_rx_i</p>
core_phy_addr_i[4:0]	In	<p>PHY Address</p> <p>Function: This bus gives the PHY address of the Station Management registers that are connected to the MAC. The PHY address has a static value.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>
revmii_phy_addr_i[4:0]	In	<p>PHY Address</p> <p>Function: This bus gives the PHY address of the Station Management registers connected to the Remote MAC. The PHY address has a static value.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: revmii_mdc_i</p>
gmii_lopbk_o	Out	<p>Loopback mode for MAC</p> <p>Function: This signal is used to select the clk_rx_i source in the loopback mode.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>
gmii_phy_intr_o	Out	<p>Interrupt to the host connected to the MAC</p> <p>Function: This signal gives the interrupt to the local host. This signal is generated when the Link Status Change interrupt is generated.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>

Table 5-21 RevMII Interface Signal (Optional)

Signal	I/O	Description
revmii_phy_intr_o	Out	<p>Interrupt to Remote MAC</p> <p>Function: This signal gives the interrupt to the remote MAC. This signal is generated when the Link Status Change interrupt is generated.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_csr_i</p>
revmii_crs_o	Out	<p>Carrier Sense</p> <p>Function: The RevMII asserts this signal to indicate to the remote MAC that either the transmit or receive medium is not idle. The RevMII de-asserts this signal when both transmit and receive mediums are idle.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: Asynchronous</p>
revmii_col_o	Out	<p>Collision Detect</p> <p>Function: The RevMII asserts this signal when a collision is detected on the medium. This signal remains asserted till the collision condition persists.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: Asynchronous</p>
revmii_mdc_i	In	<p>MDC clock input from remote MAC</p> <p>Function: The remote MAC provides this clock signal. This clock signal is used to synchronously transfer data in and out of the RevMII using revmii_mdio pin. This clock operates at a maximum frequency of 2.5 MHz (as specified in IEEE 802.3u standard).</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
revmii_mdc_RST_n	In	<p>MDC clock asynchronous reset from remote MAC</p> <p>Function: This is an active-low reset signal. This signal is available only in GMAC-CORE and GMAC-MTL configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: revmii_mdc_i</p>

Table 5-21 RevMII Interface Signal (Optional)

Signal	I/O	Description
revmii_clkmux_sel_o[2:0]	Out	<p>Remote MAC Control register bits for clock muxing</p> <p>Function: These bits are used for clock selection as described in “Clocks With RevMII” on page 434. The bit 2 is asserted during loopback operation. The bits[1:0] select the clocks according to the operation speed. Speeds are indicated as follows:</p> <ul style="list-style-type: none"> • 2'b0x: 1000 Mbps (GMII) • 2'b10: 10 Mbps (MII) • 2'b11: 100 Mbps (MII) <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: revmii_mdc_i</p>

5.2.21 SMII Interface Signal (Optional)

The optional SMII sync signal is described in [Table 5-22](#).

Table 5-22 SMII Interface Signal (Optional)

Signal	I/O	Description
smii_txsync_i	In	<p>SMII Transmit Synchronization - input signal in source synchronous (SSSMII) mode</p> <p>Function: This signal is available when the SSSMII_TXSYNC_IN parameter is enabled during the coreConsultant configuration.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_125_i</p>

5.2.22 PMT Interrupt Signal (Optional)

The optional PMT Interrupt signal is described in [Table 5-23](#).

Table 5-23 PMT Interrupt Signal (Optional)

Signal	I/O	Description
pmt_intr_o	Out	<p>PMT Interrupt</p> <p>Function: When high, indicates an interrupt event in the optional PMT module of the core. This interrupt pin is always present when the optional PMT is included and can be used to wake the system from Sleep mode. This interrupt has no masking control.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_i</p>

5.2.23 SGMII/TBI/RTBI Interface Signals (Optional)

The optional SGMII/TBI/RTBI interface signals are described in [Table 5-24](#).

Table 5-24 SGMII/TBI/RTBI Interface Signals (Optional)

Signal	I/O	Description
clk_tx_125_i	In	<p>125-MHz Transmit Clock</p> <p>Function: This is the 125 MHz clock input for the SGMII/TBI/RTBI Transmit interface. The clk_tx_i clock (125/25/2.5 MHz) should be synchronous with this clock.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
clk_rx_125_i	In	<p>125-MHz Receive Clock</p> <p>Function: This is the 125-MHz clock input for the SGMII/TBI/RTBI Receive interface. The clk_rx_i clock (125/25/2.5 MHz) must be synchronous with this clock</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
clk_tx_125_180_i	In	<p>125-MHz Transmit Clock (180)</p> <p>Function: The SGMII/TBI/RTBI transmit interface uses the rising edge of clk_tx_125_180_i to represent the falling edge of clk_tx_125_i.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: clk_tx_125_i</p>
rst_clk_tx_125_n	In	<p>125-MHz Transmit Clock Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, or GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_125_i</p>
rst_clk_rx_125_n	In	<p>125-MHz Receive Clock Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, or GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_rx_125_i</p>
rst_clk_tx_125_180_n	In	<p>125-MHz Transmit Clock (180) Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, or GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_125_180_i</p>

Table 5-24 SGMII/TBI/RTBI Interface Signals (Optional) (Continued)

Signal	I/O	Description
pcs_aquired_sync_o	Out	<p>PCS Acquired Synchronization</p> <p>Function: When high, indicates that the PCS interface has synchronized to the running disparity of the receive code-groups.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_125_i /clk_rx_i</p>
pcs_ewrap_o	Out	<p>PCS Enable Wrap</p> <p>Function: This signal enables the PHY loop back. When asserted high, the PHY should loop back the transmit serialized data to the receive path. This bit mirrors the ELE bit of the PCS Control register.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>
pcs_lck_ref_o	Out	<p>PCS Lock Reference</p> <p>Function: This signal mirrors the LR bit of the PCS Control register. It is low after reset.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>
pcs_en_cdet_o	Out	<p>PCS Enable Detect</p> <p>Function: This signal mirrors the ECD bit of the PCS Control register. It is low after reset.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>

5.2.24 SGMII Interface Signals (Optional)

The optional SGMII interface signals are described in [Table 5-25](#).

Table 5-25 SGMII Interface Signals (Optional)

Signal	I/O	Description
sgmii_link_speed_o[1:0]	Out	<p>SGMII Link Speed</p> <p>Function: Indicates link speed and should be used to change the frequency of clk_tx_i and clk_rx_i:</p> <ul style="list-style-type: none"> • 00: 10 Mbps • 01: 100 Mbps • 10: 1000 Mbps • 11: Reserved: <p>Note: If auto-negotiation is disabled in SGMII, this signal retains the value and does not change. The default value after reset is 10.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>

5.2.25 TBI Interface Signals

The additional TBI interface signals are described in [Table 5-26](#).

Table 5-26 TBI Interface Signals

Signal	I/O	Description
clk_pmarx0_i	In	<p>Recovered Receive Clock</p> <p>Function: 62.5-MHz recovered receive clock used by the GMAC to register the received data odd code groups.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: N/A</p>
clk_pmarx1_i	In	<p>Recovered Receive Clock (180)</p> <p>Function: The 62.5-MHz recovered receive clock. This clock is 180 degrees out of phase with clk_pmarx0_i and is used by the GMAC to register the even code groups of the received data</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: clk_pmarx0_i</p>
rst_clk_pmarx1_n	In	<p>Recovered Receive Clock (180) Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: clk_pmarx0_i</p>

Table 5-26 TBI Interface Signals (Continued)

Signal	I/O	Description
tbi_sigdet_i	In	TBI Signal Detect Function: When high, informs the GMAC that a signal has been detected at the PMD device. As TBI is generally used with Fibre media, it indicates the detection of light. Active State: High Registered: No Synchronous to: clk_rx_125_i

5.2.26 SMA Interface Signals (Optional)

The optional SMA interface signals are described in [Table 5-27](#). The SMA signals are also valid when the MAC is operating with RevMII PHY interface. In this mode, the remote MAC uses the MDIO interface to program its corresponding RevMII Registers. In RevMII mode, the data signals are synchronous to revmii_mdc_i.



Note When you enable the SMA interface and the RevMII interface, then the phy_intf_sel_i input signal, sampled at reset, selects the active interface. This means that when phy_intf_sel_i is 3'b111, the MDIO is the slave interface in RevMII (synchronous to revmii_mdc_i). In all other cases, the MDIO is the master interface synchronous to clk_csr_i.

Table 5-27 SMA Interface Signals (Optional)

Signal	I/O	Description
gmii_mdo_o	Out	Management Data Out Function: The GMAC uses this signal to transfer control and data information to the PHY. In RevMII mode, this signal gives the data output during read operations initiated by the remote MAC. Active State: N/A Registered: Yes Synchronous to: clk_csr_i/revmii_mdc_i
gmii_mdi_i	In	Management Data In Function: The PHY generates this signal to transfer register data during a read operation. This signal is driven synchronously with the gmii_mdc_o clock. In the RevMII mode, this signal transfers the control information and data from the remote MAC. Active State: N/A Registered: Yes Synchronous to: clk_csr_i/revmii_mdc_i
gmii_mdo_o_e	Out	Management Data Output Enable Function: This enable signal drives the gmii_mdo_o signal from an external three-state I/O buffer. This signal is asserted whenever valid data is driven on the gmii_mdo_o signal. Active State: High Registered: Yes Synchronous to: clk_csr_i/revmii_mdc_i

Table 5-27 SMA Interface Signals (Optional) (Continued)

Signal	I/O	Description
gmii_mdc_o	Out	<p>Management Data Clock</p> <p>Function: In non-RevMII mode, the GMAC provides timing reference for the gmii_mdi_i and gmii_mdo_o signals on GMII/MII through this aperiodic clock. The maximum frequency of this clock is 2.5 MHz. This clock is generated from the application clock through a clock divider.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>

5.2.27 APB Interface Signals (Optional)

The optional APB interface signals (instead of the AHB/AXI Slave Interface in the GMAC-AHB/GMAC-AXI configuration or the MAC Control Interface in other configurations) are described in [Table 5-28](#).



Note The signals in this interface are synchronous to hclk_i (for GMAC-AHB configuration), aclk_i (for GMAC-AXI configuration), and clk_app_i (for GMAC-DMA and GMAC-MTL configurations) by default. All signals are synchronous to clk_csr_i clock when the optional clk_csr_i clock input is selected. In the GMAC-CORE configuration, all signals are synchronous to clk_csr_i.

Table 5-28 APB Interface Signals (Optional)

Signal	I/O	Description
psel_i	In	<p>APB Slave Select</p> <p>Function: The assertion of this signal is the start of transaction and indicates the validity of the control signals and pwdata.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i/clk_csr_i</p>
paddr_i[15:0]	In	<p>APB Address</p> <p>Function: This signal carries the register address of the CSR module. The address is valid only when signal psel_i is asserted.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i/clk_csr_i</p>
penable_i	In	<p>APB Enable</p> <p>Function: When high, this signal completes the read or write transaction cycle.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i/clk_csr_i</p>

Table 5-28 APB Interface Signals (Optional) (Continued)

Signal	I/O	Description
pwrite_i	In	<p>APB Read/Write</p> <p>Function: When high, indicates a write operation; when low, indicates a read operation</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i/clk_csr_i</p>
pwdata_i[31:0]	In	<p>Write Data</p> <p>Function: This signal carries the write data from the Application for the CSR module Control registers.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i/clk_csr_i</p>
prdata_o[31:0]	Out	<p>Read Data</p> <p>Function: This signal outputs the read data from the CSR module to the APB bus. Valid data is output 1 clock cycle after psel_i is asserted. If psel_i continues to be asserted for burst transfers, then valid data is output every 2 clock cycles during this period.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: hclk_i/aclk_i/clk_app_i/clk_csr_i</p>

5.2.28 Sideband Signals (Optional)

These signals are optional and applicable to specific configurations as mentioned in its description. By default configuration, these signals are not enabled.

Table 5-29 Sideband Signals (Optional)

Signal	I/O	Description
sbd_flowctrl_i	In	<p>Sideband Flow Control</p> <p>Function: When set high, instructs the GMAC to transmit PAUSE Control frames in Full-duplex mode. In Half-duplex mode, the GMAC enables the back-pressure function until this signal is made low again. This signal is an optional port applicable to only GMAC-AHB, GMAC-AXI, GMAC-DMA and GMAC-MTL configuration.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: Asynchronous</p>

Table 5-29 Sideband Signals (Optional) (Continued)

Signal	I/O	Description
sbd_dis_transmit_i	In	<p>Sideband MAC Transmitter Disable Control</p> <p>Function: When set high, this signal instructs the GMAC transmitter to stop transmitting frames after the completion of any current frame. The GMAC transmitter restarts transmission only if this signal is reset to low and the MAC Configuration register Transmitter Enable bit [3] is set high.</p> <p>This signal is an optional port applicable to GMAC-CORE and GMAC-MTL configurations only.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: Asynchronous</p>
sbd_dis_receiver_i	In	<p>Sideband MAC Receiver Disable Control</p> <p>Function: When set high, this signal instructs the GMAC receiver to stop receiving frames after the completion of any current frame. The GMAC receiver restarts reception only if this signal is reset to low and the MAC Configuration Register Receiver Enable bit [2] is set high.</p> <p>This signal is an optional port applicable to GMAC-CORE and GMAC-MTL configurations only.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: Asynchronous</p>
sbd_data_endianess_i	In	<p>Sideband Data Endianness Control</p> <p>Function: When set high, this signal configures the DMA to transfer data in big-endian format. When low (by default), the data is transferred in little-endian format. This signal is sampled during active reset (including soft-reset) only and ignored after reset is de-asserted. In GMAC-AXI configuration, sbd_data_endianess_i is also used for descriptor endianness.</p> <p>This signal is an optional port applicable only to the GMAC-DMA, GMAC-AXI, and GMAC-AHB configurations and is enabled when Endianness is configured for “BOTH_OPTIONS” (see Table 7-2 for parameter details).</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i/hclk_i/aclk_i/clk_csr_i (depending on which is the slave interface clock)</p>
sbd_desc_endianess_i	In	<p>Sideband Descriptor Endianness Control</p> <p>Function: When set high, this signal configures the DMA to transfer descriptors in reverse endianness of the data format. When low (by default), the descriptors are transferred in the same endian format as the data. This signal is sampled during active reset (including soft-reset) only and ignored after reset is de-asserted.</p> <p>This signal is an optional port applicable only to the GMAC-DMA and GMAC-AHB configurations and is enabled when the Descriptor Endianness is configured to BOTH_OPTIONS (Table 7-2).</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i/hclk_i</p>

Table 5-29 Sideband Signals (Optional) (Continued)

Signal	I/O	Description
ati_txfifoflush_i	In	<p>Tx FIFO Flush</p> <p>Function: When there is a pulse on this input, the Tx FIFO is flushed completely. The ati_rdy_o signal is de-asserted immediately on sampling a high on this input. When the Tx FIFO Flush operation completes, ati_rdy_o is asserted. The core starts storing a frame only on receiving an SOF after a successful Flush operation. The ati_txfifoflush_i signal should be de-asserted after 1 clock cycle.</p> <p>This signal is an optional port applicable to the GMAC-MTL configuration.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
ati_chksum_ctrl_i[1:0]	In	<p>Sideband Checksum Offload Control</p> <p>Function: These input signals control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6, as follows:</p> <ul style="list-style-type: none"> • 2'b00: Do nothing. Checksum engine is bypassed • 2'b01: Insert IPv4 Header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram. • 2'b10: Insert TCP, UDP, or ICMP checksum with pseudo-header checksum available in Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram is IPv4. • 2'b11: Insert TCP, UDP, or ICMP checksum. This checksum, including pseudo-header bytes, is fully calculated in the Checksum Offload engine. An IPv4 header checksum is also inserted if the encapsulated datagram is IPv4. <p>This signal is sampled only when ati_val_i and ati_sof_i are high. This signal is valid only in the GMAC-MTL configuration.</p> <p>Active State: N/A</p> <p>Registered: No</p> <p>Synchronous to: clk_app_i</p>
mac_speed_o[1:0]	Out	<p>Sideband MAC Speed Control</p> <p>Function: This signal indicates the 10-Mbps, 100-Mbps, or 1000-Mbps operating speed and is driven by the GMAC Configuration register's PS (Port Select) and FES (Speed) bits (Table 6-29).</p> <p>Speeds are indicated as follows:</p> <ul style="list-style-type: none"> • 2'b0x: 1000 Mbps (GMII) • 2'b10: 10 Mbps (MII) • 2'b11: 100 Mbps (MII) <p>When the MAC is operating with an RevMII interface, this signal is available by default. When the MAC is operating with an RMII, SMII, SGMII, or RGMII PHY interface, you can select this signal during coreConsultant configuration to control the clock divider that is required to generate the GMAC transmit clock (clk_tx_i).</p> <p>Active state: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_csr_i</p>

5.2.29 IEEE 1588 Timestamp Signals (Optional)

Table 5-30 describes the optional signals that are added to the core when you enable the IEEE 1588 Timestamp feature during configuration. Some of these signals depend on the GMAC-UNIV configuration also, as described in the table.



Note You can leave the output signals `ptp_timestamp_o[63:0]` and `ptp_pps_o` unconnected when no other component in the system requires timestamping.

Table 5-30 IEEE 1588 Timestamp Signals (Optional)

Signal	I/O	Description
<code>clk_ptp_ref_i</code>	In	<p>Reference Clock for the Timestamp Update Logic</p> <p>Function: This is the reference clock, provided by an external oscillator or timing source, that is used for the timestamp logic. See “Frequency Range of Reference Timing Clock” on page 120 to decide the frequency of <code>clk_ptp_ref_i</code>.</p> <p>Active State: N/A</p> <p>Registered: N/A</p> <p>Synchronous to: NA</p>
<code>rst_clk_ptp_ref_i</code>	In	<p>Timestamp Logic Reset</p> <p>Function: Reset signal. This input is not present in the GMAC-AHB, GMAC-AXI, or GMAC-DMA configurations.</p> <p>Active State: Low</p> <p>Registered: No</p> <p>Synchronous to: <code>clk_ptp_ref_i</code></p>
<code>ptp_timestamp_i[63:0]</code>	In	<p>64-Bit Timestamp Input</p> <p>Function: This bus provides the system time used to time-stamp the specific transmit packets or received frames. This is available only when you select the External Timestamp Input Enable option during configuration.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: <code>clk_ptp_ref_i</code></p>
<code>ptp_timestamp_o[63:0]</code>	Out	<p>64-Bit Reference Timestamp Output</p> <p>Function: This bus provides the system timestamp output when the timestamp is generated internally. This output pin is available only when you select the External Timestamp Input option during configuration.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: <code>clk_ptp_ref_i</code></p>

Table 5-30 IEEE 1588 Timestamp Signals (Optional) (Continued)

Signal	I/O	Description
ptp_pps_o	Out	<p>Pulse Per Second</p> <p>Function: This signal is asserted every time the Seconds counter is incremented. This output pin is available only when the External Time Stamp Input option is not selected during configuration. When Advanced Time-Stamping is enabled, the frequency of this signal is controlled by the PPS Control register.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_ptp_ref_i</p>
ptp_aux_ts_trig_i	In	<p>Auxiliary Timestamp Trigger</p> <p>Function: This signal is asserted to take an auxiliary snapshot of the time and store it in the auxiliary timestamp FIFO. A rising edge on this port is used to trigger the auxiliary snapshot.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: Asynchronous</p>
mti_ena_timestamp_i	In	<p>Transmit Enable Timestamping</p> <p>Function: When set high, this signal directs the MAC to capture the timestamp for the transmit frame. This signal is available only in the GMAC-CORE configuration. The application should assert it only for the PTP frames that require time stamping. This signal is sampled only when mti_sof_i and mti_val_i are high.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: clk_tx_i</p>
mti_timestamp_o[63:0]	Out	<p>Transmit Frame Timestamping</p> <p>Function: This bus, available only in GMAC-CORE configuration, provides the timestamp of the frame transmitted by the MAC. This value on this bus is valid only when mti_txstatus_val_o signal is high.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_tx_i</p>
ati_ena_timestamp_i	In	<p>Transmit Enable Timestamping</p> <p>Function: When set high, instructs the MAC to capture the timestamp for the transmit frame. This signal is available only in the GMAC-MTL configuration. The application should assert it only for the PTP frames that require time stamping. This signal is sampled only when ati_sof_i and ati_val_i are high.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>

Table 5-30 IEEE 1588 Timestamp Signals (Optional) (Continued)

Signal	I/O	Description
ati_timestamp_o[63:0]	Out	<p>Transmit Frame TimeStamp</p> <p>Function: This bus provides the timestamp of the transmitted frame for which the status is available on the ati_txstatus_o bus. The value on this bus is qualified by ati_txstatus_val_o signal.</p> <p>This signal is available only in the GMAC-MTL configuration.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>
mri_timestamp_o[63:0]	Out	<p>Receive Timestamp Value</p> <p>Function: This bus provides the timestamp of the received frame. This signal is available only in the GMAC-CORE configuration, and is valid only when mri_eof_o asserted.</p> <p>Active State: N/A</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>
ari_timestamp_val_o	Out	<p>Receive Timestamp Valid</p> <p>Function: When asserted, this signal indicates that a valid timestamp value is available on the ARI data bus (ari_data_o). This signal is available only in GMAC-MTL configuration.</p> <p>In 32-bit configuration, this signal is asserted for the two clock cycles required to transfer two 32-bit words. The lower 32 bits of the timestamp (the Nanoseconds field) is given first on ari_data_o, followed by the upper word (the Seconds field).</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_app_i</p>

5.2.30 Energy Efficient Ethernet Signal

The Energy Efficient Ethernet (EEE) signals are described in [Table 5-31](#).

Table 5-31 Energy Efficient Ethernet Signal

Signal	I/O	Description
lpi_intr_o	Out	<p>LPI interrupt</p> <p>Function: This signal is asserted when the MAC receiver exits the LPI state.</p> <p>Active State: High</p> <p>Registered: Yes</p> <p>Synchronous to: clk_rx_i</p>

5.2.31 Test Mode

The Test Mode signals are described in [Table 5-32](#).

Table 5-32 Test Mode Signal

Signal	I/O	Description
test_mode	In	<p>Test Mode</p> <p>Function: This input (available only in GMAC-AHB, GMAC_AXI, and GMAC-DMA configurations) enables bypassing of the internally generated resets and directly connecting these resets to the external reset input (hreset_n, areset_n, or pwr_on_rst_n respectively). This input should normally be tied to zero and set only during scan testing.</p> <p>Active State: High</p> <p>Registered: No</p> <p>Synchronous to: N/A</p>

6 Registers

This chapter describes the GMAC-UNIV registers. It contains the following sections:

- ❖ “[Register Maps](#)” on page 281
- ❖ “[Register Descriptions](#)” on page 294

6.1 Register Maps

The register maps in this section provide high-level summaries of each register or group of registers. The register names in the register maps (tables) are cross-referenced to the detailed register descriptions in “[Register Descriptions](#)” on page 294. To view the detailed description of a register, double-click the register name.

6.1.1 DMA Register Map

[Table 6-1](#) provides the address map of the DMA registers. When you enable the AV feature, the information in [Table 6-1](#) becomes specific to channel 0.

Table 6-1 DMA Register Map

Register No.	Offset Address	Register Name and Description
0	0x1000	Register 0 (Bus Mode Register) Controls the Host Interface Mode.
1	0x1004	Register 1 (Transmit Poll Demand Register) Used by the host to instruct the DMA to poll the Transmit Descriptor list.
2	0x1008	Register 2 (Receive Poll Demand Register) Used by the Host to instruct the DMA to poll the Receive Descriptor list.
3	0x100C	Register 3 (Receive Descriptor List Address Register) Points the DMA to the start of the Receive Descriptor list.
4	0x1010	Register 4 (Transmit Descriptor List Address Register) Points the DMA to the start of the Transmit Descriptor list.

Table 6-1 DMA Register Map (Continued)

Register No.	Offset Address	Register Name and Description
5	0x1014	Register 5 (Status Register) The Software driver (application) reads this register during interrupt service routine or polling to determine the status of the DMA.
6	0x1018	Register 6 (Operation Mode Register) Establishes the Receive and Transmit operating modes and command. Note: Register 6 (Operation Mode register) is valid and present in the GMAC-MTL configuration.
7	0x101C	Register 7 (Interrupt Enable Register) Enables the interrupts reported by the Status Register.
8	0x1020	Register 8 (Missed Frame and Buffer Overflow Counter Register) Contains the counters for discarded frames because no host Receive Descriptor was available, and discarded frames because of Receive FIFO Overflow.
9	0x1024	Register 9 (Receive Interrupt Watchdog Timer Register) Watchdog time-out for Receive Interrupt (RI) from DMA.
10	0x1028	Register 10 (AXI Bus Mode Register) Controls AXI Master behavior (mainly controls burst splitting and number of outstanding requests).
11	0x102C	Register 11 (AXI Status Register) Gives the idle status of the AXI master's read/write channels.
12–17	0x1030–0x1044	Reserved
18	0x1048	Register 18 (Current Host Transmit Descriptor Register) Points to the start of current Transmit Descriptor read by the DMA.
19	0x104C	Register 19 (Current Host Receive Descriptor Register) Points to the start of current Receive Descriptor read by the DMA.
20	0x1050	Register 20 (Current Host Transmit Buffer Address Register) Points to the current Transmit Buffer address read by the DMA.
21	0x1054	Register 21 (Current Host Receive Buffer Address Register) Points to the current Receive Buffer address read by the DMA.
22	0x1058	Register 22 (HW Feature Register) Indicates the presence of the optional features of the core.

Table 6-2 provides the address map of the DMA registers for channel 1. The address map in Table 6-2 is applicable only when you enable the AV Feature and select one or more additional Transmit channels.

In GMAC-MTL configuration, only registers from address 0x1160 to 0x1174 are present. In other configurations, all registers described in Table 6-2 are present.

Table 6-2 Channel 1 DMA Register Map

Register No.	Offset Address	Register Name and Description
64	0x1100	Register 64 (Channel 1 Bus Mode Register) Controls the Host Interface mode for channel 1. For information about this register, see Register 0 (Bus Mode Register) .
65	0x1104	Register 65 (Channel 1 Transmit Poll Demand Register) Used by the host to instruct the DMA to poll the Transmit Descriptor list. For information about this register, see Register 1 (Transmit Poll Demand Register) .
66	0x1108	Register 66 (Channel 1 Receive Poll Demand Register) Used by the Host to instruct the DMA to poll the Receive Descriptor list. For information about this register, see Register 2 (Receive Poll Demand Register) .
67	0x110C	Register 67 (Channel 1 Receive Descriptor List Address Register) Points the DMA to the start of the Receive Descriptor list. For information about this register, see Register 3 (Receive Descriptor List Address Register) .
68	0x1110	Register 68 (Channel 1 Transmit Descriptor List Address Register) Points the DMA to the start of the Transmit Descriptor list. For information about this register, see Register 4 (Transmit Descriptor List Address Register) .
69	0x1114	Register 69 (Channel 1 Status Register) The Software driver (application) reads this register during interrupt service routine or polling to determine the status of the DMA. Bits 29:26 are reserved for the Channel 1 Status Register. For information about this register, see Register 5 (Status Register) .
70	0x1118	Register 70 (Channel 1 Operation Mode Register) Establishes the Receive and Transmit operating modes and command. For information about this register, see Register 6 (Operation Mode Register) .
71	0x111C	Register 71 (Channel 1 Interrupt Enable Register) Enables the interrupts reported by the Status Register. For information about this register, see Register 7 (Interrupt Enable Register) .
72	0x1120	Register 72 (Channel 1 Missed Frame and Buffer Overflow Counter Register) Contains the counters for discarded frames because no host Receive Descriptor was available, and discarded frames because of Receive FIFO Overflow. For information about this register, see Register 8 (Missed Frame and Buffer Overflow Counter Register) .
73	0x1124	Register 73 (Channel 1 Receive Interrupt Watchdog Timer Register) Watchdog time-out for Receive Interrupt (RI) from DMA. For information about this register, see Register 9 (Receive Interrupt Watchdog Timer Register) .
74-75	0x1128-0x112C	Reserved for this configuration.

Table 6-2 Channel 1 DMA Register Map

Register No.	Offset Address	Register Name and Description
76	0x1130	Register 76 (Channel 1 Slot Function Control and Status Register) Contains the control bits for slot function and its status for channel 1 transmit path.
77-81	0x1134-0x1144	Reserved
82	0x1148	Register 82 (Channel 1 Current Host Transmit Descriptor Register) Points to the start of current Transmit Descriptor read by the DMA. For information about this register, see Register 18 (Current Host Transmit Descriptor Register) .
83	0x114C	Register 83 (Channel 1 Current Host Receive Descriptor Register) Points to the start of current Receive Descriptor read by the DMA. For information about this register, see Register 19 (Current Host Receive Descriptor Register) .
84	0x1150	Register 84 (Channel 1 Current Host Transmit Buffer Address Register) Points to the current Transmit Buffer address read by the DMA. For information about this register, see Register 20 (Current Host Transmit Buffer Address Register) .
85	0x1154	Register 85 (Channel 1 Current Host Receive Buffer Address Register) Points to the current Receive Buffer address read by the DMA. For information about this register, see Register 21 (Current Host Receive Buffer Address Register) .
86-87	0x1158-0x115C	Reserved
88	0x1160	Register 88 (Channel 1 CBS Control Register) Controls the channel 1 credit shaping operation on the transmit path.
89	0x1164	Register 89 (Channel 1 CBS Status Register) Provides the average traffic transmitted in channel 1.
90	0x1168	Register 90 (Channel 1 idleSlopeCredit Register) Contains the idleSlope credit value required for the credit-based shaper algorithm for channel 1.
91	0x116C	Register 91 (Channel 1 sendSlopeCredit Register) Contains the sendSlope credit value required for the credit-based shaper algorithm for channel 1.
92	0x1170	Register 92 (Channel 1 hiCredit Register) Contains the hiCredit value required for the credit-based shaper algorithm for channel 1.
93	0x1174	Register 93 (Channel 1 loCredit Register) Contains the loCredit value required for the credit-based shaper algorithm for channel 1.

[Table 6-3](#) provides the address map of the DMA registers for channel 2. The address map in [Table 6-3](#) is applicable only when you enable the AV feature and select two additional Transmit channels.

In GMAC-MTL configuration, only registers from address 0x1260 to 0x1274 are present. In other configurations, all registers mentioned in [Table 6-2](#) are present.

Table 6-3 Channel 2 DMA Register Map

Register No.	Offset Address	Register Name and Description
128	0x1200	Register 128 (Channel 2 Bus Mode Register) Controls the Host Interface mode for channel 2. For information about this register, see Register 0 (Bus Mode Register) .
129	0x1204	Register 129 (Channel 2 Transmit Poll Demand Register) Used by the host to instruct the DMA to poll the Transmit Descriptor list. For information about this register, see Register 1 (Transmit Poll Demand Register) .
130	0x1208	Register 130 (Channel 2 Receive Poll Demand Register) Used by the Host to instruct the DMA to poll the Receive Descriptor list. For information about this register, see Register 2 (Receive Poll Demand Register) .
131	0x120C	Register 131 (Channel 2 Receive Descriptor List Address Register) Points the DMA to the start of the Receive Descriptor list. For information about this register, see Register 3 (Receive Descriptor List Address Register) .
132	0x1210	Register 132 (Channel 2 Transmit Descriptor List Address Register) Points the DMA to the start of the Transmit Descriptor List. For information about this register, see Register 4 (Transmit Descriptor List Address Register) .
133	0x1214	Register 133 (Channel 2 Status Register) The Software driver (application) reads this register during interrupt service routine or polling to determine the status of the DMA. Bits 29:26 are reserved for the Channel 2 Status Register. For information about this register, see Register 5 (Status Register) .
134	0x1218	Register 134 (Channel 2 Operation Mode Register) Establishes the Receive and Transmit operating modes and command. For information about this register, see Register 6 (Operation Mode Register) .
135	0x121C	Register 135 (Channel 2 Interrupt Enable Register) Enables the interrupts reported by the Status Register. For information about this register, see Register 7 (Interrupt Enable Register) .
136	0x1220	Register 136 (Channel 2 Missed Frame and Buffer Overflow Counter Register) For information about this register, see Register 8 (Missed Frame and Buffer Overflow Counter Register) .

Table 6-3 Channel 2 DMA Register Map

Register No.	Offset Address	Register Name and Description
137	0x1224	Register 137 (Channel 2 Receive Interrupt Watchdog Timer Register) Watchdog time-out for Receive Interrupt (RI) from DMA. For information about this register, see Register 9 (Receive Interrupt Watchdog Timer Register) .
138-139	0x1228-0x122C	Reserved for this configuration.
140	0x1230	Register 140 (Channel 2 Slot Function Control and Status Register) Contains the control bits for slot function and its status for channel 2 transmit path. For information about this register, see Register 76 (Channel 1 Slot Function Control and Status Register) .
141-145	0x1234-0x1244	Reserved
146	0x1248	Register 146 (Channel 2 Current Host Transmit Descriptor Register) Points to the start of current Transmit Descriptor read by the DMA. For information about this register, see Register 18 (Current Host Transmit Descriptor Register) .
147	0x124C	Register 147 (Channel 2 Current Host Receive Descriptor Register) Points to the start of current Receive Descriptor read by the DMA. For information about this register, see Register 19 (Current Host Receive Descriptor Register) .
148	0x1250	Register 148 (Channel 2 Current Host Transmit Buffer Address Register) Points to the current Transmit Buffer address read by the DMA. For information about this register, see Register 20 (Current Host Transmit Buffer Address Register) .
149	0x1254	Register 149 (Channel 2 Current Host Receive Buffer Address Register) Points to the current Receive Buffer address read by the DMA. For information about this register, see Register 21 (Current Host Receive Buffer Address Register) .
150-151	0x1258-0x125C	Reserved
152	0x1260	Register 152 (Channel 2 CBS Control Register) Controls the channel 2 credit shaping operation on the transmit path. For information about this register, see Register 88 (Channel 1 CBS Control Register) .
153	0x1264	Register 153 (Channel 2 CBS Status Register) Provides the average traffic transmitted in channel 2. For information about this register, see Register 89 (Channel 1 CBS Status Register) .
154	0x1268	Register 154 (Channel 2 idleSlopeCredit Register) Contains the idleSlope credit value required for the credit-based shaper algorithm for channel 2. For information about this register, see Register 90 (Channel 1 idleSlopeCredit Register) .

Table 6-3 Channel 2 DMA Register Map

Register No.	Offset Address	Register Name and Description
155	0x126C	Register 155 (Channel 2 sendSlopeCredit Register) Contains the sendSlope credit value required for the credit-based shaper algorithm for channel 2. For information about this register, see Register 91 (Channel 1 sendSlopeCredit Register) .
156	0x1270	Register 156 (Channel 2 hiCredit Register) Contains the hiCredit value required for the credit-based shaper algorithm for channel 2. For information about this register, see Register 92 (Channel 1 hiCredit Register) .
157	0x1274	Register 157 (Channel 2 loCredit Register) Contains the loCredit value required for the credit-based shaper algorithm for channel 2. For information about this register, see Register 93 (Channel 1 loCredit Register) .

6.1.2 GMAC Register Map

[Table 6-4](#) provides the address map of the GMAC core registers. Most of the registers are optional and present in the source code only if selected during coreConsultant configuration. If a register is not configured, then that address is reserved.

Table 6-4 GMAC Register Map

Register No.	Offset Address	Register Name and Description
0	0x0000	Register 0 (MAC Configuration Register) This is the operation mode register for the MAC.
1	0x0004	Register 1 (MAC Frame Filter) Contains the frame filtering controls.
2	0x0008	Register 2 (Hash Table High Register) Contains the higher 32 bits of the Multicast Hash table. This register is present only when you select the Hash filter function in coreConsultant. (See Table 7-9 .)
3	0x000C	Register 3 (Hash Table Low Register) Contains the lower 32 bits of the Multicast Hash table. This register is present only when you select the Hash filter function in coreConsultant. (See Table 7-9 .)
4	0x0010	Register 4 (GMII Address Register) Controls the management cycles to an external PHY. This register is present only when you select the Station Management (MDIO) feature in coreConsultant. (See Table 7-24 .)

Table 6-4 GMAC Register Map (Continued)

Register No.	Offset Address	Register Name and Description
5	0x0014	Register 5 (GMII Data Register) Contains the data to be written to or read from the PHY register. This register is present only when you select the Station Management (MDIO) feature in coreConsultant. (See Table 7-24 .)
6	0x0018	Register 6 (Flow Control Register) Controls the generation of control frames.
7	0x001C	Register 7 (VLAN Tag Register) Identifies IEEE 802.1Q VLAN type frames.
8	0x0020	Register 8 (Version Register) Identifies the version of the Core.
9	0x0024	Register 9 (Debug Register) Gives the status of various internal blocks for debugging.
10	0x0028	Remote Wake-Up Frame Filter Register (on page 161) This is the address through which the application writes or reads the remote wake-up frame filter registers (wkupfmfilter_reg). The wkupfmfilter_reg register is a pointer to eight wkupfmfilter_reg registers. The wkupfmfilter_reg register is loaded by sequentially loading the eight register values. Eight sequential writes to this address (0x0028) writes all wkupfmfilter_reg registers. Similarly, eight sequential reads from this address (0x0028) read all wkupfmfilter_reg registers. This register contains the higher 16 bits of the seventh MAC address. This register is present only when you select the PMT module Remote Wake-up feature in coreConsultant. (See Table 7-18 .)
11	0x002C	PMT Control and Status Register (on page 160) This register is present only when you select the PMT module in coreConsultant. (See Table 7-18 .)
12	0x0030	Register 12 (LPI Control and Status Register) Controls the Low Power Idle (LPI) operations and provides the LPI status of the core. This register is present only when you select the Energy Efficient Ethernet feature in coreConsultant.
13	0x0034	Register 13 (LPI Timers Control Register) Controls the timeout values in LPI states. This register is present only when you select the Energy Efficient Ethernet feature in coreConsultant.
14	0x0038	Register 14 (Interrupt Status Register) Contains the interrupt status.
15	0x003C	Register 15 (Interrupt Mask Register) Contains the masks for generating the interrupts.
16	0x0040	Register 16 (MAC Address0 High Register) Contains the higher 16 bits of the first MAC address.

Table 6-4 GMAC Register Map (Continued)

Register No.	Offset Address	Register Name and Description
17	0x0044	Register 17 (MAC Address0 Low Register) Contains the lower 32 bits of the first MAC address.
18	0x0048	Register 18 (MAC Address1 High Register) Contains the higher 16 bits of the second MAC address. This register is present only when Enable MAC Address1 is selected in coreConsultant. (See Table 7-8).
19	0x004C	Register 19 (MAC Address1 Low Register) Contains the lower 32 bits of the second MAC address. This register is present only when Enable MAC Address1 is selected in coreConsultant. (See Table 7-8).
20	0x0050	MAC Address2 High Register Contains the lower 32 bits of the third MAC address. This register is present only when Enable MAC Address2 is selected in coreConsultant. (See Table 7-8).
21	0x0054	MAC Address2 Low Register Contains the lower 32 bits of the third MAC address. This register is present only when Enable MAC Address2 is selected in coreConsultant. (See Table 7-8).
22	0x0058	MAC Address3 High Register Contains the higher 16 bits of the fourth MAC address. This register is present only when Enable MAC Address3 is selected in coreConsultant. (See Table 7-8).
23	0x005C	MAC Address3 Low Register Contains the lower 32 bits of the fourth MAC address. This register is present only when Enable MAC Address3 is selected in coreConsultant. (See Table 7-8).
24	0x0060	MAC Address4 High Register Contains the higher 16 bits of the fifth MAC address. This register is present only when Enable MAC Address4 is selected in coreConsultant. (See Table 7-8).
25	0x0064	MAC Address4 Low Register Contains the lower 32 bits of the fifth MAC address. This register is present only when Enable MAC Address4 is selected in coreConsultant. (See Table 7-8).
26	0x0068	MAC Address5 High Register Contains the higher 16 bits of the sixth MAC address. This register is present only when Enable MAC Address5 is selected in coreConsultant. (See Table 7-8).
27	0x006C	MAC Address5 Low Register Contains the lower 32 bits of the sixth MAC address. This register is present only when Enable MAC Address5 is selected in coreConsultant. (See Table 7-8).
28	0x0070	MAC Address6 High Register Contains the higher 16 bits of the seventh MAC address. This register is present only when Enable MAC Address6 is selected in coreConsultant. (See Table 7-8).

Table 6-4 GMAC Register Map (Continued)

Register No.	Offset Address	Register Name and Description
29	0x0074	MAC Address6 Low Register Contains the lower 32 bits of the seventh MAC address. This register is present only when Enable MAC Address6 is selected in coreConsultant. (See Table 7-8).
30	0x0078	MAC Address7 High Register Contains the higher 16 bits of the eighth MAC address. This register is present only when Enable MAC Address7 is selected in coreConsultant. (See Table 7-8).
31	0x007C	MAC Address7 Low Register Contains the lower 32 bits of the eighth MAC address. This register is present only when Enable MAC Address7 is selected in coreConsultant. (See Table 7-8).
32	0x0080	MAC Address8 High Register Contains the higher 16 bits of the ninth MAC address. This register is present only when Enable MAC Address8 is selected in coreConsultant. (See Table 7-8).
33	0x0084	MAC Address8 Low Register Contains the lower 32 bits of the ninth MAC address. This register is present only when Enable MAC Address8 is selected in coreConsultant. (See Table 7-8).
34	0x0088	MAC Address9 High Register Contains the higher 16 bits of the tenth MAC address. This register is present only when Enable MAC Address9 is selected in coreConsultant. (See Table 7-8).
35	0x008C	MAC Address9 Low Register Contains the lower 32 bits of the tenth MAC address. This register is present only when Enable MAC Address9 is selected in coreConsultant. (See Table 7-8).
36	0x0090	MAC Address10 High Register Contains the higher 16 bits of the eleventh MAC address. This register is present only when Enable MAC Address10 is selected in coreConsultant. (See Table 7-8).
37	0x0094	MAC Address10 Low Register Contains the lower 32 bits of the eleventh MAC address. This register is present only when Enable MAC Address10 is selected in coreConsultant. (See Table 7-8).
38	0x0098	MAC Address11 High Register This register contains the higher 16 bits of the twelfth MAC address. This register is present only when Enable MAC Address11 is selected in coreConsultant. (See Table 7-8).
39	0x009C	MAC Address11 Low Register Contains the lower 32 bits of the twelfth MAC address. This register is present only when Enable MAC Address11 is selected in coreConsultant. (See Table 7-8).
40	0x00A0	MAC Address12 High Register Contains the higher 16 bits of the thirteenth MAC address. This register is present only when Enable MAC Address12 is selected in coreConsultant. (See Table 7-8).

Table 6-4 GMAC Register Map (Continued)

Register No.	Offset Address	Register Name and Description
41	0x00A4	MAC Address12 Low Register Contains the lower 32 bits of the thirteenth MAC address. This register is present only when Enable MAC Address12 is selected in coreConsultant. (See Table 7-8).
42	0x00A8	MAC Address13 High Register Contains the higher 16 bits of the fourteenth MAC address. This register is present only when Enable MAC Address13 is selected in coreConsultant. (See Table 7-8).
43	0x00AC	MAC Address13 Low Register Contains the lower 32 bits of the fourteenth MAC address. This register is present only when Enable MAC Address13 is selected in coreConsultant. (See Table 7-8).
44	0x00B0	MAC Address14 High Register Contains the higher 16 bits of the fifteenth MAC address. This register is present only when Enable MAC Address14 is selected in coreConsultant. (See Table 7-8).
45	0x00B4	MAC Address14 Low Register Contains the lower 32 bits of the fifteenth MAC address. This register is present only when Enable MAC Address14 is selected in coreConsultant. (See Table 7-8).
46	0x00B8	MAC Address15 High Register Contains the higher 16 bits of the sixteenth MAC address. This register is present only when Enable MAC Address15 is selected in coreConsultant. (See Table 7-8).
47	0x00BC	MAC Address15 Low Register Contains the lower 32 bits of the sixteenth MAC address. This register is present only when Enable MAC Address15 is selected in coreConsultant. (See Table 7-8).
48	0x00C0	Register 48 (AN Control Register) Enables and/or restarts auto-negotiation. This register also enables the Physical Coding Sublayer (PCS) loopback. This register is present only when you select the TBI, RTBI, or SGMII interface in coreConsultant.
49	0x00C4	Register 49 (AN Status Register) Indicates the link and auto-negotiation status. This register is present only when you select the TBI, RTBI, or SGMII interface in coreConsultant.
50	0x00C8	Register 50 (Auto-Negotiation Advertisement Register) This register is configured before auto-negotiation begins. It contains the advertised ability of the GMAC. This register is present only when you select the TBI, RTBI, or SGMII interface in coreConsultant.
51	0x00CC	Register 51 (Auto-Negotiation Link Partner Ability Register) Contains the advertised ability of the link partner. Its value is valid after successful completion of auto-negotiation or when a new base page has been received (indicated in the Auto-Negotiation Expansion Register). This register is present only when you select the TBI, RTBI, or SGMII interface in coreConsultant.

Table 6-4 GMAC Register Map (Continued)

Register No.	Offset Address	Register Name and Description
52	0x00D0	Register 52 (Auto-Negotiation Expansion Register) Indicates whether a new base page has been received from the link partner. This register is present only when you select the TBI, RTBI, or SGMII interface in coreConsultant.
53	0x00D4	Register 53 (TBI Extended Status Register) Indicates all modes of operation of the GMAC. This register is present only when you select the TBI, RTBI, or SGMII interface in coreConsultant.
54	0x00D8	Register 54 (SGMII/RGMII/SMII Status Register) Indicates the status signals received from the PHY through the SGMII, RGMII, or SMII interface. This register is present only when you select the SGMII, RGMII, or SMII interface in coreConsultant.
55–63	0x00DC–0x00FC	Reserved
64–191	0x0100–0x02FC	MMC Register Map
192–447	0x0300–0x06FC	Reserved
448	0x0700	Register 448 (Timestamp Control Register) Controls the timestamp generation and update logic. This register is present only when IEEE1588 time stamping is enabled during coreConsultant configuration.
449	0x0704	Register 449 (Sub-Second Increment Register) Contains the 8-bit value by which the Sub-Second register is incremented. This register is present only when IEEE1588 time stamping is enabled without an external timestamp input.
450	0x0708	Register 450 (System Time - Seconds Register) Contains the lower 32 bits of the seconds field of the system time. This register is present only when IEEE1588 time stamping is enabled without an external timestamp input.
451	0x070C	Register 451 (System Time - Nanoseconds Register) Contains 32 bits of the nano-seconds field of the system time. This register is only present when IEEE1588 time stamping is enabled without an external timestamp input.
452	0x0710	Register 452 (System Time - Seconds Update Register) Contains the lower 32 bits of the seconds field to be written to, added to, or subtracted from the System Time value. This register is only present when IEEE1588 time stamping is enabled without an external timestamp input.
453	0x0714	Register 453 (System Time - Nanoseconds Update Register) Contains 32 bits of the nano-seconds field to be written to, added to, or subtracted from the System Time value. This register is only present when IEEE1588 time stamping is enabled without an external timestamp input.

Table 6-4 GMAC Register Map (Continued)

Register No.	Offset Address	Register Name and Description
454	0x0718	Register 454 (Timestamp Addend Register) This register is used by the software to readjust the clock frequency linearly to match the master clock frequency. This register is only present when IEEE1588 time stamping is enabled without an external timestamp input.
455	0x071C	Register 455 (Target Time Seconds Register) Contains the higher 32 bits of time to be compared with the system time for interrupt event generation. This register is only present when IEEE1588 time stamping is enabled without an external timestamp input.
456	0x0720	Register 456 (Target Time Nanoseconds Register) Contains the lower 32 bits of time to be compared with the system time for interrupt event generation. This register is only present when IEEE1588 time stamping is enabled without an external timestamp input.
457	0x0724	Register 457 (System Time - Higher Word Seconds Register) Contains the most significant 16-bits of the timestamp seconds value. This register is optional and can be selected using the parameter mentioned in “ IEEE 1588 Timestamp Block ” on page 367 .
458	0x0728	Register 458 (Timestamp Status Register) Contains the PTP status. This register is available only when the advanced IEEE 1588 timestamp feature is selected.
459	0x072C	Register 459 (PPS Control Register) This register is used to control the interval of the PPS signal output, such that a duration of less than 1 second can be achieved between pulses.
460	0x0730	Register 460 (Auxiliary Timestamp - Nanoseconds Register) Contains the lower 32 bits (nano-seconds field) of the auxiliary timestamp register.
461	0x0734	Register 461 (Auxiliary Timestamp - Seconds Register) Contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.
462	0x0738	Register 462 (AV MAC Control Register) Controls the AV traffic and queue management in the MAC Receiver. This register is present only when you select the AV feature in coreConsultant.
463–511	0x073C–0x7FC	Reserved
512	0x0800	MAC Address 16 High Register Contains the higher 16 bits of the seventeenth MAC address. This register is present only when Enable MAC Address16 is selected in coreConsultant. (See Table 7-8).
513	0x0804	MAC Address 16 Low Register Contains the lower 32 bits of the seventeenth MAC address. This register is present only when Enable MAC Address16 is selected in coreConsultant. (See Table 7-8).

Table 6-4 GMAC Register Map (Continued)

Register No.	Offset Address	Register Name and Description
514	0x0808	MAC Address 17 High Register Contains the higher 16 bits of the eighteenth MAC address. This register is present only when Enable MAC Address17 is selected in coreConsultant. (See Table 7-8).
...		
543	0x087C	MAC Address 31 Low Register Contains the lower 32 bits of the thirty-second MAC address. This register is present only when Enable MAC Address 31 is selected in coreConsultant. (See Table 7-8).
544-1023	0x0880-0x0FFC	Reserved

6.2 Register Descriptions

The Access column of each register description that follows specifies how the application and the core can access the register fields of the CSRs.

The Access column uses the following conventions:

Read Only (RO)	Register field can only be read by the application. Writes to read-only fields have no effect.
Write Only (WO)	Register field can only be written by the application.
Read and Write (R_W)	Register field can be read and written by the application. The application can set this field by writing 1'b1 and can clear it by writing 1'b0.
Read, Self Set, and Write Clear (R_SS_WC)	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and can be cleared to 1'b0 by the application with a register write of 1'b1 (Write Clear). A register write of 1'b0 has no effect on this field. The conditions under which the core sets this field are explained in detail in the field's description. (Example, interrupt bits)
Read, Write Set, and Self Clear (R_WS_SC)	Register field can be read by the application (Read), can be set to 1'b1 by the application with a register write of 1'b1 (Write Set), and is cleared to 1'b0 by the core (Self Clear). The application cannot clear this type of field, and a register write of 1'b0 to this bit has no effect on this field. The conditions under which the core clears this field are explained in detail in the field's description. (Example, soft-reset signals)
Read, Self Set, and Self Clear or Write Clear (R_SS_SC_WC)	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and can be cleared to 1'b0 either by the core itself (Self Clear) or by the application with a register write of 1'b0 (Write Clear). A register write of 1'b1 to this bit has no effect on this field. The conditions under which the core sets or clears this field are explained in detail in the field's description.
Read Only and Write Trigger (RO_WT)	Register field can be read by the application, and when a write operation is performed with any data value, an event is triggered, as explained in the field's description. (Example: Tx Poll Demand register)



Read, Self Set, and Read Clear (R_SS_RC)	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and is automatically cleared to 1'b0 on a register read. A register write has no effect on this field. The conditions under which the core sets this field are explained in detail in the field's description. (Example: Overflow counter.)
Read, Write, and Self Update (R_W_SU)	Register field can be read and written by the application. The register field updates itself based on the event. For example, system time in PTP configuration.
Read, Self Set, Self Clear, and Latch Low (R_SS_SC_LLO)	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), can be cleared to 1'b0 by the core (Self Clear), and on reset, it is latched to low value (Latch Low). For example, link up and down status (LS: Link Status) in AN Status Register.

Note

- All CSRs are implemented as 32-bit registers and can be accessed in 1 read/write transaction with a 32-bit MCI interface (or 32-bit AHB slave port). If you initiate a non-32-bit register access, the byte enable signals (mci_be_i[3:0]) qualify the corresponding register bytes. Only write operations are qualified with byte enables, while read operations are always 32-bit, unless the register is affected by a read operation. Byte enable signals qualify reads to such registers as, for example, the MMC counter registers.
- Register bits[7:0] correspond to address[1:0] = 00 in little-endian mode, while register bits[31:24] correspond to address[1:0] = 00 in big-endian mode.
- The transfer of particular register contents (e.g., MAC DA address register) to the Transmit or Receive clock domains are initiated when a particular byte is written. This is indicated in the register descriptions, where applicable.
- When any Register content is being transferred to a different clock domain after a write operation, there should not be any further writes to the same location until the first write is updated. Otherwise, the second write operation does not get updated to the destination clock domain. Thus the delay between two writes to the same register location should be at least 4 cycles of the destination clock (PHY receive clock, PHY transmit clock, or PTP clock).

6.2.1 DMA Register Description

This section defines the bits for each DMA register. The write data inputs to the DMA registers are qualified with the corresponding mci_be_i signal inputs (MCI interface). Thus, non-32 bit accesses are allowed as long as the address is Word-aligned. For the APB interface, only 32-bit accesses are possible, while for AHB slave interfaces, byte, half-word, or word accesses are possible.

6.2.1.1 Register 0 (Bus Mode Register)

The Bus Mode register establishes the bus operating modes for the DMA.

Table 6-5 Register 0 (Bus Mode Register)

Field	Description	Reset	Access
31:30	Reserved	0	RO

Table 6-5 Register 0 (Bus Mode Register) (Continued)

Field	Description	Reset	Access
29:28	PRWG: Channel Priority Weights Sets the priority weights for channel 0 during the round-robin arbitration between the DMA channels for the system bus. Value Priority Weight 00 1 01 2 10 3 11 4 These bits are present in all GMAC configurations except GMAC-AXI when you select the AV feature. Otherwise, these bits are reserved and are read-only (RO). For more information about the priority weights of DMA channels, see “ DMA Arbiter Functions ” on page 134 .	0	R_W
27	TXPR: Transmit Priority When set, this bit indicates that the transmit DMA has higher priority than the receive DMA during arbitration for the system-side bus. In GMAC-AXI configuration, this bit is reserved and is read-only (RO). For more information about the priority scheme between the transmit and receive paths, see Table 4-12 in “ DMA Arbiter Functions ” on page 134 .	0	R_W
26	MB: Mixed Burst When this bit is set high and FB bit is low, the AHB master interface starts all bursts of length more than 16 with INCR (undefined burst) whereas it reverts to fixed burst transfers (INCRx and SINGLE) for burst-length of 16 and below. This bit is valid only in GMAC-AHB configuration and reserved in all others.	0	R_W
25	AAL: Address-Aligned Beats When this bit is set high and the FB bit equals 1, the AHB/AXI interface generates all bursts aligned to the start address LS bits. If the FB bit is equal to 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address. This bit is valid only in the GMAC-AHB and GMAC-AXI configuration and is reserved (RO with default value 0) in all other configurations.	0	R_W
24	8xPBL Mode When set high, this bit multiplies the PBL value programmed (bits [22:17] and bits [13:8]) eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value. Note: This bit function is not backward compatible. Before release 3.50a, this bit was 4xPBL.	0	R_W
23	USP: Use Separate PBL When set high, this bit configures the RxDMA to use the value configured in bits [22:17] as PBL while the PBL value in bits [13:8] is applicable to TxDMA operations only. When reset to low, the PBL value in bits [13:8] is applicable for both DMA engines.	0	R_W

Table 6-5 Register 0 (Bus Mode Register) (Continued)

Field	Description	Reset	Access										
22:17	RPBL: RxDMA PBL These bits indicate the maximum number of beats to be transferred in one RxDMA transaction. This is the maximum value that is used in a single block Read/Write. The RxDMA always attempts to burst as specified in RPBL every time it starts a Burst transfer on the host bus. RPBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. These bits are valid and applicable only when USP is set high.	01H	R_W										
16	FB: Fixed Burst This bit controls whether the AHB/AXI Master interface performs fixed burst transfers or not. When set, the AHB uses only SINGLE, INCR4, INCR8, or INCR16 during start of normal burst transfers. When reset, the AHB/AXI uses SINGLE and INCR burst transfer operations. In GMAC-AXI configuration, see bit 0 (UNDEF: AXI Undefined Burst Length) of the AXI Bus Mode register for detailed description.	0	R_W										
15:14	PR: Priority Ratio These bits controls the priority ratio in the weighted round-robin arbitration between the RxDMA and TxDMA. These bits are valid only when bit 1 (DA: DMA Arbitration Scheme) is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether bit 27 (TXPR: Transmit Priority) is reset or set. <table> <thead> <tr> <th>Value</th> <th>Priority Ratio</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>1:1</td> </tr> <tr> <td>01</td> <td>2:1</td> </tr> <tr> <td>10</td> <td>3:1</td> </tr> <tr> <td>11</td> <td>4:1</td> </tr> </tbody> </table> In GMAC-AXI configuration, these bits are reserved and are read-only (RO). For more information about the priority scheme between the transmit and receive paths, see Table 4-12 in “DMA Arbiter Functions” on page 134 .	Value	Priority Ratio	00	1:1	01	2:1	10	3:1	11	4:1	0	R_W
Value	Priority Ratio												
00	1:1												
01	2:1												
10	3:1												
11	4:1												

Table 6-5 Register 0 (Bus Mode Register) (Continued)

Field	Description	Reset	Access																																																													
13:8	<p>PBL: Programmable Burst Length</p> <p>These bits indicate the maximum number of beats to be transferred in one DMA transaction. This is the maximum value that is used in a single block Read/Write. The DMA always attempts to burst as specified in PBL each time it starts a Burst transfer on the host bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. When USP is set high, this PBL value is applicable for TxDMA transactions only.</p> <p>If the number of beats to be transferred is more than 32, then perform the following steps:</p> <ol style="list-style-type: none"> 1. Set the 8xPBL mode. 2. Set the PBL. <p>For example, if the maximum number of beats to be transferred is 64, then first set 8xPBL to 1 and then set PBL to 8.</p> <p>The PBL values have the following limitation: The maximum number of possible beats (PBL) is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO, except when specified.</p> <p>For different data bus widths and FIFO sizes, the valid PBL range (including x8 mode) is provided in the following table. If the PBL is common for both transmit and receive DMA, the minimum Rx FIFO and Tx FIFO depths must be considered.</p> <table border="1"> <thead> <tr> <th>Data Bus Width</th> <th>FIFO Depth</th> <th>Valid PBL Range in Full Duplex Mode</th> <th>Valid PBL Range (for TxFIFO only) in Half Duplex Mode</th> </tr> </thead> <tbody> <tr> <td rowspan="10">32</td> <td>128 bytes</td> <td>16 or less</td> <td>8 or less (10/100 Mbps only)</td> </tr> <tr> <td>256 bytes</td> <td>32 or less</td> <td>32 or less (10/100 Mbps only)</td> </tr> <tr> <td>512 bytes</td> <td>64 or less</td> <td>64 or less (10/100 Mbps only)</td> </tr> <tr> <td>1 KB</td> <td>128 or less</td> <td>128 or less in 10/100 Mbps, 64 or less in 1000-Mbps</td> </tr> <tr> <td>2 KB and above</td> <td>All</td> <td>All</td> </tr> <tr> <td rowspan="6">64</td> <td>128 bytes</td> <td>8 or less</td> <td>4 or less (10/100 Mbps only)</td> </tr> <tr> <td>256 bytes</td> <td>16 or less</td> <td>16 or less (10/100 Mbps only)</td> </tr> <tr> <td>512 bytes</td> <td>32 or less</td> <td>32 or less (10/100 Mbps only)</td> </tr> <tr> <td>1 KB</td> <td>64 or less</td> <td>64 in 10/100-Mbps operation, 32 or less in 1000-Mbps</td> </tr> <tr> <td>2 KB</td> <td>128 or less</td> <td>128 or less</td> </tr> <tr> <td>4 KB and above</td> <td>All</td> <td>All</td> </tr> <tr> <td rowspan="8">128</td> <td>128 bytes</td> <td>4 or less</td> <td>2 or less (10/100 Mbps only)</td> </tr> <tr> <td>256 bytes</td> <td>8 or less</td> <td>8 or less (10/100 Mbps only)</td> </tr> <tr> <td>512 bytes</td> <td>16 or less</td> <td>16 or less (10/100 Mbps only)</td> </tr> <tr> <td>1KB</td> <td>32 or less</td> <td>32 in 10/100-Mbps operation, 16 or less in 1000-Mbps</td> </tr> <tr> <td>2KB</td> <td>64 or less</td> <td>64 or less</td> </tr> <tr> <td>4KB</td> <td>128 or less</td> <td>128 or less</td> </tr> <tr> <td>8KB and above</td> <td>All</td> <td>All</td> </tr> </tbody> </table>	Data Bus Width	FIFO Depth	Valid PBL Range in Full Duplex Mode	Valid PBL Range (for TxFIFO only) in Half Duplex Mode	32	128 bytes	16 or less	8 or less (10/100 Mbps only)	256 bytes	32 or less	32 or less (10/100 Mbps only)	512 bytes	64 or less	64 or less (10/100 Mbps only)	1 KB	128 or less	128 or less in 10/100 Mbps, 64 or less in 1000-Mbps	2 KB and above	All	All	64	128 bytes	8 or less	4 or less (10/100 Mbps only)	256 bytes	16 or less	16 or less (10/100 Mbps only)	512 bytes	32 or less	32 or less (10/100 Mbps only)	1 KB	64 or less	64 in 10/100-Mbps operation, 32 or less in 1000-Mbps	2 KB	128 or less	128 or less	4 KB and above	All	All	128	128 bytes	4 or less	2 or less (10/100 Mbps only)	256 bytes	8 or less	8 or less (10/100 Mbps only)	512 bytes	16 or less	16 or less (10/100 Mbps only)	1KB	32 or less	32 in 10/100-Mbps operation, 16 or less in 1000-Mbps	2KB	64 or less	64 or less	4KB	128 or less	128 or less	8KB and above	All	All	01H	R_W
Data Bus Width	FIFO Depth	Valid PBL Range in Full Duplex Mode	Valid PBL Range (for TxFIFO only) in Half Duplex Mode																																																													
32	128 bytes	16 or less	8 or less (10/100 Mbps only)																																																													
	256 bytes	32 or less	32 or less (10/100 Mbps only)																																																													
	512 bytes	64 or less	64 or less (10/100 Mbps only)																																																													
	1 KB	128 or less	128 or less in 10/100 Mbps, 64 or less in 1000-Mbps																																																													
	2 KB and above	All	All																																																													
	64	128 bytes	8 or less	4 or less (10/100 Mbps only)																																																												
		256 bytes	16 or less	16 or less (10/100 Mbps only)																																																												
		512 bytes	32 or less	32 or less (10/100 Mbps only)																																																												
		1 KB	64 or less	64 in 10/100-Mbps operation, 32 or less in 1000-Mbps																																																												
		2 KB	128 or less	128 or less																																																												
4 KB and above		All	All																																																													
128	128 bytes	4 or less	2 or less (10/100 Mbps only)																																																													
	256 bytes	8 or less	8 or less (10/100 Mbps only)																																																													
	512 bytes	16 or less	16 or less (10/100 Mbps only)																																																													
	1KB	32 or less	32 in 10/100-Mbps operation, 16 or less in 1000-Mbps																																																													
	2KB	64 or less	64 or less																																																													
	4KB	128 or less	128 or less																																																													
	8KB and above	All	All																																																													

Table 6-5 Register 0 (Bus Mode Register) (Continued)

Field	Description	Reset	Access
7	<p>ATDS: Alternate Descriptor Size</p> <p>When set, the size of the alternate descriptor (described in “Alternate or Enhanced Descriptors” on page 414) increases to 32 bytes (8 DWORDS). This is required when the Advanced Timestamp feature or Full IPC Offload Engine is enabled in the receiver. This bit is present only when you select the Alternate Descriptor feature and any one of the following features during configuration:</p> <ul style="list-style-type: none"> Advanced Timestamp feature IPC Full Checksum Offload (type 2) feature <p>Otherwise, this bit is reserved and is read-only.</p> <p>When reset, the descriptor size reverts back to 4 DWORDs (16 bytes).</p> <p>This bit preserves the backward compatibility for the descriptor size. In versions prior to 3.50a, the descriptor size is 16 bytes for both normal and enhanced descriptor. In 3.50a, descriptor size is increased to 32 bytes because of the Advanced Timestamp and IPC Full Checksum Offload features.</p> <p>The enhanced descriptor is not required if the Advanced Timestamp and IPC Full Checksum Offload features are not enabled. In such case, you can use the 16 bytes descriptor to save 4 bytes of memory.</p>	0	R_W
6:2	<p>DSL: Descriptor Skip Length</p> <p>This bit specifies the number of Word/Dword/Lword (depending on 32/64/128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When DSL value is equal to zero, then the descriptor table is taken as contiguous by the DMA, in Ring mode.</p>	00H	R_W
1	<p>DA: DMA Arbitration Scheme</p> <p>This bit specifies the arbitration scheme between the transmit and receive paths of channel 0.</p> <ul style="list-style-type: none"> 0: Weighted Round-robin with Rx:Tx or Tx:Rx. The priority between the paths is according to the priority specified in bits 15:14 (PR: Priority Ratio) and priority weights specified in bit 27 (TXPR: Transmit Priority). 1: Fixed priority. Tx has priority over Rx when bit 27 (TXPR: Transmit Priority) is set. Otherwise, Rx has priority over Tx. <p>In the GMAC-AXI configuration, these bits are reserved and are read-only (RO).</p> <p>For more information about the priority scheme between the transmit and receive paths, see Table 4-12 in “DMA Arbiter Functions” on page 134.</p>	0	R_W
0	<p>SWR: Software Reset</p> <p>When this bit is set, the MAC DMA Controller resets all GMAC Subsystem internal registers and logic. It is cleared automatically after the reset operation has completed in all of the core clock domains. Read a 0 value in this bit before re-programming any register of the core.</p> <p>Note: The reset operation is completed only when all the resets in all the active clock domains are de-asserted. Therefore, it is essential that all the PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion.</p>	1	R_WS_SC

6.2.1.2 Register 1 (Transmit Poll Demand Register)

The Transmit Poll Demand register enables the Transmit DMA to check whether or not the current descriptor is owned by DMA. The Transmit Poll Demand command is given to wake up the TxDMA if it is in Suspend mode. The TxDMA can go into Suspend mode because of an Underflow error in a transmitted frame or because of the unavailability of descriptors owned by Transmit DMA. You can give this command anytime and the TxDMA resets this command once it starts re-fetching the current descriptor from host memory.

Table 6-6 Register 1 (Transmit Poll Demand Register)

Field	Description	Reset	Access
31:0	TPD: Transmit Poll Demand When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 18. If that descriptor is not available (owned by Host), transmission returns to the Suspend state and DMA Register 5[2] is asserted. If the descriptor is available, transmission resumes.	0000_0000H	RO_WT

6.2.1.3 Register 2 (Receive Poll Demand Register)

The Receive Poll Demand register enables the receive DMA to check for new descriptors. This command is given to wake up the RxDMA from SUSPEND state. The RxDMA can go into SUSPEND state only because of the unavailability of descriptors owned by it.

Table 6-7 Register 2 (Receive Poll Demand Register)

Field	Description	Reset	Access
31:0	RPD: Receive Poll Demand When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 19. If that descriptor is not available (owned by Host), reception returns to the Suspended state and Register 5[7] is not asserted. If the descriptor is available, the Receive DMA returns to the active state.	0000_0000H	RO_WT

6.2.1.4 Register 3 (Receive Descriptor List Address Register)

The Receive Descriptor List Address register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word/Dword/Lword-aligned (for 32/64/128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to Register 3 is permitted only when reception is stopped. When stopped, Register 3 must be written to before the receive Start command is given.

You can write to Register 3 only when Rx DMA has stopped, that is, bit 1 (SR) is set to zero in [Register 6 \(Operation Mode Register\)](#). When stopped, Register 3 can be written with a new descriptor list address. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

If the Register 3 is not changed when the SR bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

Table 6-8 Register 3 (Receive Descriptor List Address Register)

Field	Description	Reset	Access
31:0	Start of Receive List This field contains the base address of the First Descriptor in the Receive Descriptor list. The LSB bits [1/2/3:0] for 32/64/128-bit bus width are ignored and are taken as all-zero by the DMA internally. Therefore, these LSB bits are Read-Only (RO).	0000_0000H	R_W

6.2.1.5 Register 4 (Transmit Descriptor List Address Register)

The Transmit Descriptor List Address register points to the start of the Transmit Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word/DWORD/LWORD-aligned (for 32/64/128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to Register 4 only when Tx DMA has stopped, that is, bit 13 (ST) is set to zero in [Register 6 \(Operation Mode Register\)](#). When stopped, Register 4 can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly programmed descriptor base address.

If the Register 4 is not changed when the ST bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

Table 6-9 Register 4 (Transmit Descriptor List Address Register)

Field	Description	Reset	Access
31:0	Start of Transmit List This field contains the base address of the First Descriptor in the Transmit Descriptor list. The LSB bits [1/2/3:0] for 32/64/128-bit bus width are ignored and are taken as all-zero by the DMA internally. Therefore, these LSB bits are Read-Only (RO).	0000_0000H	R_W

6.2.1.6 Register 5 (Status Register)

The Status register contains all the status bits that the DMA reports to the host. The Software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. Register 5 bits are not cleared when read. Writing 1'b1 to (unreserved) bits in Register 5[16:0] clears these bits and writing 1'b0 has no effect. Each field (bits[16:0]) can be masked by masking the appropriate bit in Register 7.

Table 6-10 Register 5 (Status Register)

Field	Description	Reset	Access
31	Reserved	0	RO

Table 6-10 Register 5 (Status Register) (Continued)

Field	Description	Reset	Access
30	<p>GLPII: GMAC LPI Interrupt (for channel 0)</p> <p>This bit indicates an interrupt event in the LPI logic of GMAC core. To reset this bit to 1'b0, the software must read the corresponding registers in the GMAC core to get the exact cause of interrupt and clear its source.</p> <p>Note: GLPII status is given only in Channel 0 DMA register and is applicable only when Energy Efficient Ethernet feature is enabled. Otherwise, this bit is reserved.</p> <p>-or-</p> <p>GTMSI: GMAC TMS Interrupt (for channel 1 and channel 2)</p> <p>This bit indicates an interrupt event in the traffic manager and scheduler logic of GMAC core. To reset this bit, the software must read the corresponding registers (Channel Status Register) to get the exact cause of the interrupt and clear its source.</p> <p>Note: GTMSI status is given only in Channel 1 and Channel 2 DMA register when AV feature is enabled and corresponding additional transmit channels are present. Otherwise, this bit is reserved.</p>	0	RO
29	<p>TTI: Time-Stamp Trigger Interrupt</p> <p>This bit indicates an interrupt event in the GMAC core's Timestamp Generator block. The software must read the corresponding registers in the GMAC core to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. When this bit is high, the interrupt signal from the GMAC subsystem (sbd_intr_o) is high.</p>	0	RO
28	<p>GPI: GMAC PMT Interrupt</p> <p>This bit indicates an interrupt event in the GMAC core's PMT module. The software must read the corresponding registers in the GMAC core to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. The interrupt signal from the GMAC subsystem (sbd_intr_o) is high when this bit is high.</p> <p>Note: This interrupt is separate from the pmt_intr_o interrupt.</p>	0	RO
27	<p>GMI: GMAC MMC Interrupt</p> <p>This bit reflects an interrupt event in the MMC module of the GMAC core. The software must read the corresponding registers in the GMAC core to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the GMAC subsystem (sbd_intr_o) is high when this bit is high.</p>	0	RO
26	<p>GLI: GMAC Line interface Interrupt</p> <p>This bit reflects an interrupt event in the GMAC Core's PCS, SMII, or RGMII interface block. The software must read the corresponding registers in the GMAC core to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the GMAC subsystem (sbd_intr_o) is high when this bit is high.</p>	0	RO

Table 6-10 Register 5 (Status Register) (Continued)

Field	Description	Reset	Access
25:23	EB: Error Bits These bits indicate the type of error that caused a Bus Error (e.g., error response on the AHB/AXI interface). These bits are valid only when bit 13 (FBI: Fatal Bus Error Interrupt) is set. This field does not generate an interrupt. Bit 23 1'b1 Error during data transfer by TxDMA 1'b0 Error during data transfer by RxDMA Bit 24 1'b1 Error during read transfer 1'b0 Error during write transfer Bit 25 1'b1 Error during descriptor access 1'b0 Error during data buffer access	000	RO
22:20	TS: Transmit Process State These bits indicate the Transmit DMA FSM state. This field does not generate an interrupt. <ul style="list-style-type: none">• 3'b000: Stopped; Reset or Stop Transmit Command issued.• 3'b001: Running; Fetching Transmit Transfer Descriptor.• 3'b010: Running; Waiting for status.• 3'b011: Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO).• 3'b100: TIME_STAMP write state.• 3'b101: Reserved for future use.• 3'b110: Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow.• 3'b111: Running; Closing Transmit Descriptor.	000	RO
19:17	RS: Receive Process State These bits indicate the Receive DMA FSM state. This field does not generate an interrupt. <ul style="list-style-type: none">• 3'b000: Stopped; Reset or Stop Receive Command issued.• 3'b001: Running; Fetching Receive Transfer Descriptor.• 3'b010: Reserved for future use.• 3'b011: Running; Waiting for receive packet.• 3'b100: Suspended; Receive Descriptor Unavailable.• 3'b101: Running; Closing Receive Descriptor.• 3'b110: TIME_STAMP write state.• 3'b111: Running; Transferring the receive packet data from receive buffer to host memory.	000	RO

Table 6-10 Register 5 (Status Register) (Continued)

Field	Description	Reset	Access
16	NIS: Normal Interrupt Summary Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in DMA Register 7: <ul style="list-style-type: none">• Register 5[0]: Transmit Interrupt• Register 5[2]: Transmit Buffer Unavailable• Register 5[6]: Receive Interrupt• Register 5[14]: Early Receive Interrupt Only unmasked bits affect the Normal Interrupt Summary bit. This is a sticky bit and must be cleared (by writing a 1 to this bit) each time a corresponding bit that causes NIS to be set is cleared.	0	R_SS_WC
15	AIS: Abnormal Interrupt Summary Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in DMA Register 7: <ul style="list-style-type: none">• Register 5[1]: Transmit Process Stopped• Register 5[3]: Transmit Jabber Timeout• Register 5[4]: Receive FIFO Overflow• Register 5[5]: Transmit Underflow• Register 5[7]: Receive Buffer Unavailable• Register 5[8]: Receive Process Stopped• Register 5[9]: Receive Watchdog Timeout• Register 5[10]: Early Transmit Interrupt• Register 5[13]: Fatal Bus Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared.	0	R_SS_WC
14	ERI: Early Receive Interrupt This bit indicates that the DMA had filled the first data buffer of the packet. Receive Interrupt Register 5[6] automatically clears this bit.	0	R_SS_WC
13	FBI: Fatal Bus Error Interrupt This bit indicates that a bus error occurred, as detailed in [25:23]. When this bit is set, the corresponding DMA engine disables all its bus accesses.	0	R_SS_WC
12:11	Reserved	00	RO
10	ETI: Early Transmit Interrupt This bit indicates that the frame to be transmitted was fully transferred to the MTL Transmit FIFO.	0	R_SS_WC
9	RWT: Receive Watchdog Timeout This bit is asserted when a frame with a length greater than 2,048 bytes is received (10,240 when Jumbo Frame mode is enabled).	0	R_SS_WC
8	RPS: Receive Process Stopped This bit is asserted when the Receive Process enters the Stopped state.	0	R_SS_WC

Table 6-10 Register 5 (Status Register) (Continued)

Field	Description	Reset	Access
7	RU: Receive Buffer Unavailable This bit indicates that the Next Descriptor in the Receive List is owned by the host and cannot be acquired by the DMA. Receive Process is suspended. To resume processing Receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, Receive Process resumes when the next recognized incoming frame is received. Register 5[7] is set only when the previous Receive Descriptor was owned by the DMA.	0	R_SS_WC
6	RI: Receive Interrupt This bit indicates the completion of frame reception. Specific frame status information has been posted in the descriptor. Reception remains in the Running state.	0	R_SS_WC
5	UNF: Transmit Underflow This bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.	0	R_SS_WC
4	OVF: Receive Overflow This bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to application, the overflow status is set in RDES0[11].	0	R_SS_WC
3	TJT: Transmit Jabber Timeout This bit indicates that the Transmit Jabber Timer expired, meaning that the transmitter had been excessively active. The transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.	0	R_SS_WC
2	TU: Transmit Buffer Unavailable This bit indicates that the Next Descriptor in the Transmit List is owned by the host and cannot be acquired by the DMA. Transmission is suspended. Bits[22:20] explain the Transmit Process state transitions. To resume processing transmit descriptors, the host should change the ownership of the bit of the descriptor and then issue a Transmit Poll Demand command.	0	R_SS_WC
1	TPS: Transmit Process Stopped This bit is set when the transmission is stopped.	0	R_SS_WC
0	TI: Transmit Interrupt This bit indicates that frame transmission is finished and TDES1[31] is set in the First Descriptor.	0	R_SS_WC

6.2.1.7 Register 6 (Operation Mode Register)

The Operation Mode register establishes the Transmit and Receive operating modes and commands. Register 6 should be the last CSR to be written as part of DMA initialization. This register is also present in the GMAC-MTL configuration with bits 24, 13, 2, and 1 unused and reserved.

Table 6-11 Register 6 (Operation Mode Register)

Field	Description	Reset	Access
31:27	Reserved	000H	RO
26	DT: Disable Dropping of TCP/IP Checksum Error Frames When this bit is set, the core does not drop frames that only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors in the encapsulated payload only. When this bit is reset, all error frames are dropped if the FEF bit is reset. If the Full Checksum Offload engine (Type 2) is disabled, this bit is reserved (RO with value 1'b0).	0	R_W
25	RSF: Receive Store and Forward When this bit is set, the MTL only reads a frame from the Rx FIFO after the complete frame has been written to it, ignoring RTC bits. When this bit is reset, the Rx FIFO operates in Cut-Through mode, subject to the threshold specified by the RTC bits.	0	R_W
24	DFF: Disable Flushing of Received Frames When this bit is set, the RxDMA does not flush any frames because of the unavailability of receive descriptors/buffers as it does normally when this bit is reset. (See " "Receive Process Suspended" on page 66 .) This bit is reserved (and RO) in GMAC-MTL configuration.	0	R_W
23	RFA[2]: MSB of Threshold for Activating Flow Control If the GMAC-UNIV is configured for an Rx FIFO depth of 8 KB or more, this bit (when set) provides additional threshold levels for activating the Flow Control in both Half-Duplex and Full-Duplex modes. This bit (as Most Significant Bit) along with the RFA (bits [10:9]) give the following thresholds for activating flow control. <ul style="list-style-type: none"> • 100 Full minus 5 KB • 101 Full minus 6 KB • 110 Full minus 7 KB • 111 Reserved This bit is reserved (and RO) if the Rx FIFO is 4 KB or less deep.	0	R_W
22	RFD[2]: MSB of Threshold for Deactivating Flow Control If the GMAC-UNIV is configured for an Rx FIFO depth of 8 KB or more, this bit (when set) provides additional threshold levels for deactivating the Flow Control in both Half-Duplex and Full-Duplex modes. This bit (as Most Significant Bit) along with the RFD (bits [12:11]) give the following thresholds for deactivating flow control. <ul style="list-style-type: none"> • 100 Full minus 5 KB • 101 Full minus 6 KB • 110 Full minus 7 KB • 111 Reserved This bit is reserved (and RO) if the Rx FIFO is 4 KB or less deep.	0	R_W
21	TSF: Transmit Store and Forward When this bit is set, transmission starts when a full frame resides in the MTL Transmit FIFO. When this bit is set, the TTC values specified in Register 6[16:14] are ignored. This bit should be changed only when transmission is stopped.	0	R_W

Table 6-11 Register 6 (Operation Mode Register) (Continued)

Field	Description	Reset	Access
20	FTF: Flush Transmit FIFO When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the Tx FIFO is lost/flushed. This bit is cleared internally when the flushing operation is completed fully. The Operation Mode register should not be written to until this bit is cleared. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission. Note: The flush operation completes only after emptying the TxFIFO of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host. In order to complete this flush operation, the PHY transmit clock (clk_tx_i) is required to be active.	0	R_WS_SC
19:17	Reserved	000H	RO
16:14	TTC: Transmit Threshold Control These three bits control the threshold level of the MTL Transmit FIFO. Transmission starts when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when the TSF bit (Bit 21) is reset. <ul style="list-style-type: none">• 000 64• 001 128• 010 192• 011 256• 100 40• 101 32• 110 24• 111 16	000	R_W
13	ST: Start/Stop Transmission Command When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Register 4, or from the position retained when transmission was stopped previously. If the current descriptor is not owned by the DMA, transmission enters the Suspended state and Transmit Buffer Unavailable (Register 5[2]) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting DMA Register 4, then the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and becomes the current position when transmission is restarted. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state. Note: For information about how to pause the transmission, see “Stopping and Starting Transmission” on page 550.	0	R_W

Table 6-11 Register 6 (Operation Mode Register) (Continued)

Field	Description	Reset	Access
12:11	RFD: Threshold for deactivating flow control (in both HD and FD) These bits control the threshold (Fill-level of Rx FIFO) at which the flow-control is de-asserted after activation. <ul style="list-style-type: none">• 00 Full minus 1 KB• 01 Full minus 2 KB• 10 Full minus 3 KB• 11 Full minus 4 KB Note that the de-assertion is effective only after flow control is asserted. If the Rx FIFO is 8 KB or more, an additional bit (RFD[2]) is used for more threshold levels as described in bit [22]. These bits are reserved and read-only when RxFIFO depth is less than 4 KB.	00	R_W
10:9	RFA: Threshold for activating flow control (in both HD and FD) These bits control the threshold (Fill level of Rx FIFO) at which flow control is activated. <ul style="list-style-type: none">• 00 Full minus 1 KB• 01 Full minus 2 KB• 10 Full minus 3 KB• 11 Full minus 4 KB Note that the above only applies to Rx FIFOs of 4 KB or more when the EFC bit is set high. If the Rx FIFO is 8 KB or more, an additional bit (RFA[2]) is used for more threshold levels as described in bit [23]. These bits are reserved and read-only when RxFIFO depth is less than 4 KB.	00	R_W
8	EFC: Enable HW flow control When this bit is set, the flow control signal operation based on fill-level of Rx FIFO is enabled. When reset, the flow control operation is disabled. This bit is not used (reserved and always reset) when the Rx FIFO is less than 4 KB.	0	R_W
7	FEF: Forward Error Frames When this bit is reset, the Rx FIFO drops frames with error status (CRC error, collision error, GMII_ER, giant frame, watchdog timeout, overflow). However, if the frame's start byte (write) pointer is already transferred to the read controller side (in Threshold mode), then the frames are not dropped. In GMAC-MTL configuration in which the Frame Length FIFO is also enabled during coreConsultant configuration, the Rx FIFO drops the error frames if that frame's start byte is not transferred (output) on the ARI bus. When FEF bit is set, all frames except runt error frames are forwarded to the DMA. But when RxFIFO overflows when a partial frame is written, then such frames are dropped even when FEF is set.	0	R_W
6	FUF: Forward Undersized Good Frames When set, the Rx FIFO forwards Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC). When reset, the Rx FIFO drops all frames of less than 64 bytes, unless it is already transferred because of lower value of Receive Threshold (e.g., RTC = 01).	0	R_W
5	Reserved	0	RO

Table 6-11 Register 6 (Operation Mode Register) (Continued)

Field	Description	Reset	Access
4:3	RTC: Receive Threshold Control These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are transferred automatically. Note that value of 11 is not applicable if the configured Receive FIFO size is 128 bytes. These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1. <ul style="list-style-type: none">• 00 64• 01 32• 10 96• 11 128	00	R_W
2	OSF: Operate on Second Frame When this bit is set, this bit instructs the DMA to process a second frame of Transmit data even before status for first frame is obtained.	0	R_W
1	SR: Start/Stop Receive When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes incoming frames. Descriptor acquisition is attempted from the current position in the list, which is the address set by DMA Register 3 or the position retained when the Receive process was previously stopped. If no descriptor is owned by the DMA, reception is suspended and Receive Buffer Unavailable (Register 5[7]) is set. The Start Receive command is effective only when reception has stopped. If the command was issued before setting DMA Register 3, DMA behavior is unpredictable. When this bit is cleared, RxDMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state. Note: For information about how to pause the transmission, see “Stopping and Starting Transmission” on page 550.	0	R_W
0	Reserved	0	RO

6.2.1.8 Register 7 (Interrupt Enable Register)

The Interrupt Enable register enables the interrupts reported by Register 5. Setting a bit to 1'b1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

Table 6-12 Register 7 (Interrupt Enable Register)

Field	Description	Reset	Access
31:17	Reserved	00H	RO

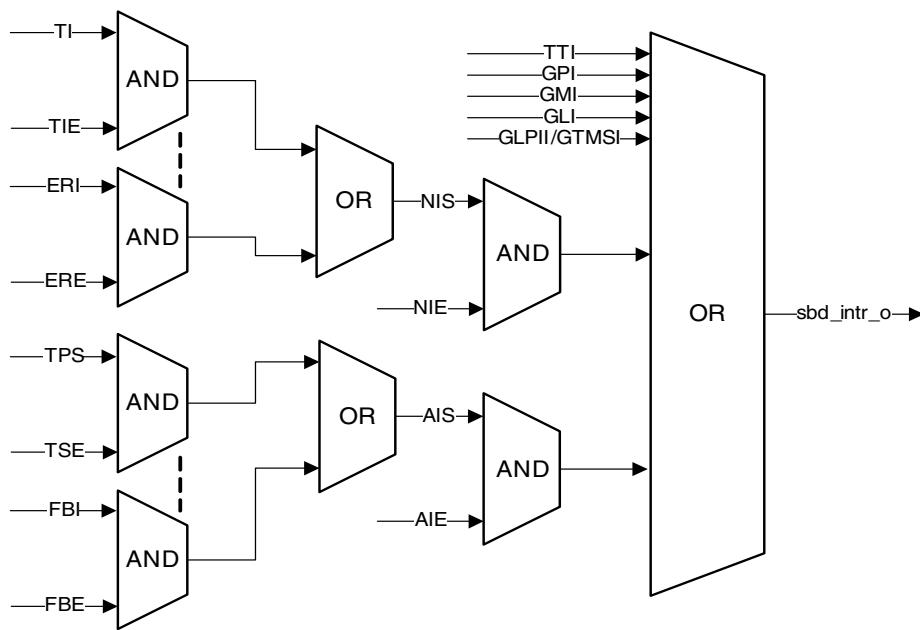
Table 6-12 Register 7 (Interrupt Enable Register) (Continued)

Field	Description	Reset	Access
16	NIE: Normal Interrupt Summary Enable When this bit is set, a normal interrupt is enabled. When this bit is reset, a normal interrupt is disabled. This bit enables the following bits: <ul style="list-style-type: none">• Register 5[0] Transmit Interrupt• Register 5[2] Transmit Buffer Unavailable• Register 5[6] Receive Interrupt• Register 5[14] Early Receive Interrupt	0	R_W
15	AIE: Abnormal Interrupt Summary Enable When this bit is set, an Abnormal Interrupt is enabled. When this bit is reset, an Abnormal Interrupt is disabled. This bit enables the following bits <ul style="list-style-type: none">• Register 5[1] Transmit Process Stopped• Register 5[3] Transmit Jabber Timeout• Register 5[4] Receive Overflow• Register 5[5] Transmit Underflow• Register 5[7] Receive Buffer Unavailable• Register 5[8] Receive Process Stopped• Register 5[9] Receive Watchdog Timeout• Register 5[10] Early Transmit Interrupt• Register 5[13] Fatal Bus Error	0	R_W
14	ERE: Early Receive Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (Register 7[16]), Early Receive Interrupt is enabled. When this bit is reset, Early Receive Interrupt is disabled.	0	R_W
13	FBE: Fatal Bus Error Enable When this bit is set with Abnormal Interrupt Summary Enable (Register 7[15]), the Fatal Bus Error Interrupt is enabled. When this bit is reset, Fatal Bus Error Enable Interrupt is disabled.	0	R_W
12:11	Reserved	00	RO
10	ETE: Early Transmit Interrupt Enable When this bit is set with an Abnormal Interrupt Summary Enable (Register 7[15]), Early Transmit Interrupt is enabled. When this bit is reset, Early Transmit Interrupt is disabled.	0	R_W
9	RWE: Receive Watchdog Timeout Enable When this bit is set with Abnormal Interrupt Summary Enable (Register 7[15]), the Receive Watchdog Timeout Interrupt is enabled. When this bit is reset, Receive Watchdog Timeout Interrupt is disabled.	0	R_W
8	RSE: Receive Stopped Enable When this bit is set with Abnormal Interrupt Summary Enable (Register 7[15]), Receive Stopped Interrupt is enabled. When this bit is reset, Receive Stopped Interrupt is disabled.	0	R_W

Table 6-12 Register 7 (Interrupt Enable Register) (Continued)

Field	Description	Reset	Access
7	RUE: Receive Buffer Unavailable Enable When this bit is set with Abnormal Interrupt Summary Enable (Register 7[15]), Receive Buffer Unavailable Interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.	0	R_W
6	RIE: Receive Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (Register 7[16]), Receive Interrupt is enabled. When this bit is reset, Receive Interrupt is disabled.	0	R_W
5	UNE: Underflow Interrupt Enable When this bit is set with Abnormal Interrupt Summary Enable (Register 7[15]), Transmit Underflow Interrupt is enabled. When this bit is reset, Underflow Interrupt is disabled.	0	R_W
4	OVE: Overflow Interrupt Enable When this bit is set with Abnormal Interrupt Summary Enable (Register 7[15]), Receive Overflow Interrupt is enabled. When this bit is reset, Overflow Interrupt is disabled	0	R_W
3	TJE: Transmit Jabber Timeout Enable When this bit is set with Abnormal Interrupt Summary Enable (Register 7[15]), Transmit Jabber Timeout Interrupt is enabled. When this bit is reset, Transmit Jabber Timeout Interrupt is disabled.	0	R_W
2	TUE: Transmit Buffer Unavailable Enable When this bit is set with Normal Interrupt Summary Enable (Register 7[16]), Transmit Buffer Unavailable Interrupt is enabled. When this bit is reset, Transmit Buffer Unavailable Interrupt is disabled.	0	R_W
1	TSE: Transmit Stopped Enable When this bit is set with Abnormal Interrupt Summary Enable (Register 7[15]), Transmission Stopped Interrupt is enabled. When this bit is reset, Transmission Stopped Interrupt is disabled.	0	R_W
0	TIE: Transmit Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (Register 7[16]), Transmit Interrupt is enabled. When this bit is reset, Transmit Interrupt is disabled.	0	R_W

The sbd_intr_o interrupt is generated as shown in [Figure 6-1](#). It is asserted only when the TTI, GPI, GMI, or GLI bits of the DMA Status register is asserted, or when the NIS/AIS Status bit is asserted and the corresponding Interrupt Enable bits (NIE/AIE) are enabled.

Figure 6-1 Sbd_intr_o Generation

Note: Signals NIS and AIS are registered.



When multiple DMA channels are present in the Audio Video (AV) mode, the sbd_intr_o interrupt generated from each channel is combined by using the OR function and is given as a single bit output. Therefore, in AV mode, the software must read the DMA Interrupt Status register of all channels to get the source of interrupt.

6.2.1.9 Register 8 (Missed Frame and Buffer Overflow Counter Register)

The DMA maintains two counters to track the number of missed frames during reception. This register reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames because of the host buffer being unavailable. Bits[27:17] indicate missed frames because of buffer overflow conditions (MTL and GMAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

Table 6-13 Register 8 (Missed Frame and Buffer Overflow Counter Register)

Field	Description	Reset	Access
31:29	Reserved	000	RO
28	Overflow bit for FIFO Overflow Counter	0	R_SS_RC
27:17	Indicates the number of frames missed by the application. This counter is incremented each time the MTL asserts the sideband signal mtl_rxoverflow_o. The counter is cleared when this register is read with mci_be_i[2] at 1'b1.	000H	R_SS_RC
16	Overflow bit for Missed Frame Counter	0	R_SS_RC

Table 6-13 Register 8 (Missed Frame and Buffer Overflow Counter Register) (Continued)

Field	Description	Reset	Access
15:0	Indicates the number of frames missed by the controller because of the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is read with mci_be_i[0] at 1'b1.	0000H	R_SS_RC

6.2.1.10 Register 9 (Receive Interrupt Watchdog Timer Register)

This register, when written with non-zero value, enables the watchdog timer for Receive Interrupt (RI) (DMA Register 5[6]).

Table 6-14 Register 9 (Receive Interrupt Watchdog Timer Register)

Field	Description	Reset	Access
31:8	Reserved	000000H	RO
7:0	RIWT: RI Watchdog Timer count Indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the RxDMA completes the transfer of a frame for which the RI status bit is not set because of the setting in the corresponding descriptor RDES1[31]. When the watch-dog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when RI bit is set high because of automatic setting of RI as per RDES1[31] of any received frame.	00H	R_W

6.2.1.11 Register 10 (AXI Bus Mode Register)

The AXI Bus Mode Register controls the AXI master behavior. It is mainly used to control the burst splitting and the number of outstanding requests. This register is present and valid only in GMAC-AXI configuration. In addition, this register is valid only in the channel 0 DMA when multiple channels are present in the AV mode.

Table 6-15 Register 10 (AXI Bus Mode Register)

Field	Description	Reset	Access
31	EN_LPI: Enable LPI (Low Power Interface) When set to 1, enables the LPI (Low Power Interface) mode supported by the GMAC-AXI and accepts the LPI request from the AXI System Clock controller. When set to 0, disables the LPI mode and always denies the LPI request from the AXI System Clock controller.	0	R_W
30	UNLCK_ON_MGK_RWK: Unlock on Magic Packet or Remote Wake Up When set to 1, enables the GMAC-AXI to request coming out of Low Power mode only when Magic Packet or Remote Wake Up Packet is received. When set to 0, enables the GMAC-AXI requests to come out of LPI mode when any frame is received.	0	R_W
29:24	Reserved	'h00	RO

Table 6-15 Register 10 (AXI Bus Mode Register) (Continued)

Field	Description	Reset	Access
23:20	WR_OSR_LMT: AXI Maximum Write Outstanding Request Limit This value limits the maximum outstanding request on the AXI write interface. Maximum outstanding requests = WR_OSR_LMT+1 Note: <ul style="list-style-type: none">• Bit 22 is reserved if AXI_GM_MAX_WR_REQUESTS = 4.• Bit 23 bit is reserved if AXI_GM_MAX_WR_REQUESTS != 16.	'h1	R_W
19:16	RD_OSR_LMT: AXI Maximum Read Outstanding Request Limit This value limits the maximum outstanding request on the AXI read interface. Maximum outstanding requests = RD_OSR_LMT+1 Note: <ul style="list-style-type: none">• The bit 18 is reserved if AXI_GM_MAX_RD_REQUESTS = 4.• The bit 19 is reserved if AXI_GM_MAX_RD_REQUESTS != 16.	'h1	R_W
15:13	Reserved	0	RO
12	AXI_AAL: Address-Aligned Beats This bit is read-only bit and reflects the AAL bit Register0 (Bus Mode Register[25]). When this bit is set to 1, GMAC-AXI performs address-aligned burst transfers on both read and write channels.	0	RO
11:8	Reserved		
7	BLEN256: AXI Burst Length 256 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 256 on the AXI master interface. This bit is present only when the configuration parameter AXI_BL is set to 256. Otherwise, this bit is reserved and is read-only (RO).	0	R_W
6	BLEN128: AXI Burst Length 128 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 128 on the AXI master interface. This bit is present only when the configuration parameter AXI_BL is set to 128 or more. Otherwise, this bit is reserved and is read-only (RO).	0	R_W
5	BLEN64: AXI Burst Length 64 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 64 on the AXI master interface. This bit is present only when the configuration parameter AXI_BL is set to 64 or more. Otherwise, this bit is reserved and is read-only (RO).	0	R_W
4	BLEN32: AXI Burst Length 32 When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 32 on the AXI master interface. This bit is present only when the configuration parameter AXI_BL is set to 32 or more. Otherwise, this bit is reserved and is read-only (RO).	0	R_W

Table 6-15 Register 10 (AXI Bus Mode Register) (Continued)

Field	Description	Reset	Access
3	BLEN16: AXI Burst Length 16 When this bit is set to 1, or when UNDEF is set to 1, the GMAC-AXI is allowed to select a burst length of 16 on the AXI master interface.	0	R_W
2	BLEN8: AXI Burst Length 8 When this bit is set to 1, or when UNDEF is set to 1, the GMAC-AXI is allowed to select a burst length of 8 on the AXI master interface.	0	R_W
1	BLEN4: AXI Burst Length 4 When this bit is set to 1, or when UNDEF is set to 1, the GMAC-AXI is allowed to select a burst length of 4 on the AXI master interface.	0	R_W
0	UNDEF: AXI Undefined Burst Length This bit is read-only bit and indicates the complement (invert) value of FB bit in Register0 (Bus Mode Register[16]). <ul style="list-style-type: none">• When this bit is set to 1, the GMAC-AXI is allowed to perform any burst length equal to or below the maximum allowed burst length as programmed in bits[7:1]• When this bit is set to 0, the GMAC-AXI is allowed to perform only fixed burst lengths as indicated by BLEN256/128/64/32/16/8/4, or a burst length of 1.	1	RO

6.2.1.12 Register 11 (AXI Status Register)

The active status of the AXI interface's read and write channel are given in this register. This register is present and valid only in GMAC-AXI configuration and useful for debug purposes. In addition, this register is valid only in the channel 0 DMA when multiple channels are present in the AV mode.

Table 6-16 Register 11 (AXI Status Register)

Field	Description	Reset	Access
31:2	Reserved	0000_0000H	RO
1	When high, it indicates that AXI Master's read channel is active and transferring data.	0	RO
0	When high, it indicates that AXI Master's write channel is active and transferring data.	0	RO

6.2.1.13 Register 18 (Current Host Transmit Descriptor Register)

The Current Host Transmit Descriptor register points to the start address of the current Transmit Descriptor read by the DMA.

Table 6-17 Register 18 (Current Host Transmit Descriptor Register)

Field	Description	Reset	Access
31:0	Host Transmit Descriptor Address Pointer Cleared on Reset. Pointer updated by DMA during operation.	0000_0000H	RO

6.2.1.14 Register 19 (Current Host Receive Descriptor Register)

The Current Host Receive Descriptor register points to the start address of the current Receive Descriptor read by the DMA.

Table 6-18 Register 19 (Current Host Receive Descriptor Register)

Field	Description	Reset	Access
31:0	Host Receive Descriptor Address Pointer Cleared on Reset. Pointer updated by DMA during operation.	0000_0000H	RO

6.2.1.15 Register 20 (Current Host Transmit Buffer Address Register)

The Current Host Transmit Buffer Address register points to the current Transmit Buffer Address being read by the DMA.

Table 6-19 Register 20 (Current Host Transmit Buffer Address Register)

Field	Description	Reset	Access
31:0	Host Transmit Buffer Address Pointer Cleared on Reset. Pointer updated by DMA during operation.	0000_0000H	RO

6.2.1.16 Register 21 (Current Host Receive Buffer Address Register)

The Current Host Receive Buffer Address register points to the current Receive Buffer address being read by the DMA.

Table 6-20 Register 21 (Current Host Receive Buffer Address Register)

Field	Description	Reset	Access
31:0	Host Receive Buffer Address Pointer Cleared on Reset. Pointer updated by DMA during operation.	0000_0000H	RO

6.2.1.17 Register 22 (HW Feature Register)

This register indicates the presence of the optional features/functions of the core. It is useful for SW driver to dynamically enable or disable the programs related to the optional blocks.

Table 6-21 Register 22 (HW Feature Register)

Field	Description	Access
Note: All bits are set or reset as per the selection of features during RTL configuration.		
31:25	Reserved	RO
24	Alternate (Enhanced Descriptor)	RO
23:22	Number of additional Tx Channels	RO
21:20	Number of additional Rx Channels	RO
19	RxFIFO > 2048 Bytes	RO
18	IP Checksum Offload (Type 2) in Rx	RO

Table 6-21 Register 22 (HW Feature Register) (Continued)

Field	Description	Access
17	IP Checksum Offload (Type 1) in Rx	RO
16	Checksum Offload in Tx	RO
15	AV Feature	RO
14	Energy Efficient Ethernet Feature	RO
13	IEEE 1588-2008 Advanced Time-Stamp	RO
12	IEEE 1588-2002 Time-Stamp	RO
11	RMON module	RO
10	PMT Magic Packet	RO
9	PMT Remote Wakeup	RO
8	SMA (MDIO) Interface	RO
7	Reserved	RO
6	PCS registers (TBI/SGMII/RTBI PHY interface)	RO
5	Multiple MAC Address Registers	RO
4	HASH Filter	RO
3	Reserved	RO
2	Half-Duplex support	RO
1	1000 Mbps support	RO
0	10/100 Mbps support	RO

6.2.1.18 Register 76 (Channel 1 Slot Function Control and Status Register)

This register controls the slot comparison feature that the channel 1 transmit DMA uses to fetch the buffer data from system memory.

Table 6-22 Register 76 (Channel 1 Slot Function Control and Status Register)

Field	Description	Reset	Access
31:20	Reserved	0	RO
19:16	RSN: Reference Slot Number These bits give the current value of the reference slot number in DMA used for comparison checking.	0	RO
15:2	Reserved	0	RO

Table 6-22 Register 76 (Channel 1 Slot Function Control and Status Register)

Field	Description	Reset	Access
1	ASC: Advance Slot Check When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM), programmed in the transmit descriptor, is equal to the reference slot number given in bits [19:16] or is ahead of the reference slot number by up to two slots. This bit is applicable only when bit 0 (ESC) is set.	0	R_W
0	ESC: Enable Slot Comparison When set, this bit enables the checking of the slot numbers, programmed in the transmit descriptor, with the current reference given in bits [19:16]. The DMA fetches the data from the corresponding buffer only when the slot number is equal to the reference slot number or is ahead of the reference slot number by 1 slot. When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed.	0	R_W

6.2.1.19 Register 88 (Channel 1 CBS Control Register)

This register controls the credit-based shaper algorithm in the Traffic Manager for scheduling the frames for transmission. This register is present only when you select the Transmit channel 1 in the AV mode.

Table 6-23 Register 88 (Channel 1 CBS Control Register)

Field	Description	Reset	Access
31:18	Reserved	0	RO
17	ABPSSIE: Average Bits Per Slot Interrupt Enable When this bit is set, the GMAC core asserts an interrupt (sbd_intr_o or mci_intr_o) when the average bits per slot status is updated (bit 17: ABSU in Register 89) for Channel 1. When this bit is cleared, interrupt is not asserted for such an event.	0	R_W
16:7	Reserved	0	RO
6:4	SLC: Slot Count The software can program the number of slots (of duration 125 micro-sec) over which the average transmitted bits per slot (provided in the CBS Status register) needs to be computed for channel 1 when the credit-based shaper algorithm is enabled. The encoding is as follows: 3'b000 1 Slot 3'b001 2 Slots 3'b010 4 Slots 3'b011 8 Slots 3'b100 16 Slots 3'b101-3'b111 Reserved	0	R_W
3:2	Reserved	0	RO

Table 6-23 Register 88 (Channel 1 CBS Control Register)

Field	Description	Reset	Access
1	CC: Credit Control When reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no frame to transmit in channel 1. When there is no frame waiting in channel 1 and other channel is transmitting, no credit is accumulated. When set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no frame to transmit in channel 1. The credit accumulates even when there is no frame waiting in channel 1 and other channel is transmitting.	0	R_W
0	CBSD: Credit-Based Shaper Disable When set, the core disables the credit-based shaper algorithm for channel 1 traffic and makes the traffic management algorithm to strict priority for channel 1 over channel 0. When reset, the credit-based shaper algorithm schedules the traffic in channel 1 for transmission.	0	R_W

6.2.1.20 Register 89 (Channel 1 CBS Status Register)

This register provides the average traffic transmitted in channel 1. This register is present only when you select the Transmit channel 1 in the AV mode.

Table 6-24 Register 89 (Channel 1 CBS Status Register)

Field	Description	Reset	Access
31:18	Reserved	0	RO
17	ABSU: ABS Updated When set, it indicates that the core has updated the ABS value. This bit is cleared when the ABS value is read by the application.	0	R_SS_SC
16:0	ABS: Average Bits per Slot Contains the average transmitted bits per slot. These bits are computed over programmed number of slots (SLC bits in the CBS Control Register) for channel 1 traffic. The maximum value is 0x30D4 for 100 Mbps and 0x1E848 for 1000 Mbps.	0	RO

6.2.1.21 Register 90 (Channel 1 idleSlopeCredit Register)

This register provides the bandwidth allocated for the AV traffic on channel 1. This register is present only when you select the Transmit channel 1 in the AV mode.

Table 6-25 Register 90 (Channel 1 idleSlopeCredit Register)

Field	Description	Reset	Access
31:14	Reserved	0	RO

Table 6-25 Register 90 (Channel 1 idleSlopeCredit Register)

Field	Description	Reset	Access
13:0	ISC: idleSlopeCredit Contains the idleSlopeCredit value required for the credit-based shaper algorithm for channel 1. This is the rate of change of credit in bits per cycle (40ns and 8ns for 100 Mbps and 1000 Mbps respectively) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1024. The maximum value is portTransmitRate, that is, 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode.	0	R_W

6.2.1.22 Register 91 (Channel 1 sendSlopeCredit Register)

This register provides the bandwidth that is available for the AV traffic on other channels. This register is present only when you select the Transmit channel 1 in the AV mode.

Table 6-26 Register 91 (Channel 1 sendSlopeCredit Register)

Field	Description	Reset	Access
31:14	Reserved	0	RO
13:0	SSC: sendSlopeCredit Contains the sendSlopeCredit value required for credit-based shaper algorithm for channel 1. This is the rate of change of credit in bits per cycle (40ns and 8ns for 100 Mbps and 1000 Mbps respectively) when the credit is decreasing. The software should program this field with computed credit in bits per cycle scaled by 1024. The maximum value is portTransmitRate, that is, 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode. This field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when channel 1 is selected for transmission.	0	R_W

6.2.1.23 Register 92 (Channel 1 hiCredit Register)

This register provides the maximum value that can be accumulated for channel 1 in the credit parameter of the credit-based shaper algorithm. This register is present only when you select the Transmit channel 1 in the AV mode.

Table 6-27 Register 92 (Channel 1 hiCredit Register)

Field	Description	Reset	Access
31:29	Reserved	0	RO
28:0	HC: hiCredit Contains the hiCredit value required for the credit-based shaper algorithm for channel 1. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1024. The maximum value is maxInterferenceSize, that is, best-effort maximum frame size which is 16384 bytes or 131072 bits. The value to be specified is $131072 \times 1024 = 134217728$ or 0x0800_0000.	0	R_W

6.2.1.24 Register 93 (Channel 1 IoCredit Register)

This register provides the minimum value that can be accumulated for channel 1 in the credit parameter of the credit-based shaper algorithm. This register is present only when you select Transmit channel 1 in the AV mode.

Table 6-28 Register 93 (Channel 1 IoCredit Register)

Field	Description	Reset	Access
31:29	Reserved	0x7	RO
28:0	LC: IoCredit Contains the IoCredit value required for the credit-based shaper algorithm for channel 1. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1024. The maximum value is maxInterferenceSize, that is, best-effort maximum frame size which is 16384 bytes or 131072 bits. The value to be specified is $131072 \times 1024 = 134217728$ or 0x0800_0000. The programmed value is 2's complement (negative number), that is, 0xF800_0000.	0xFFFF_FFFF	R_W

6.2.2 GMAC Register Description

6.2.2.1 Register 0 (MAC Configuration Register)

The MAC Configuration register establishes receive and transmit operating modes.

Table 6-29 Register 0 (MAC Configuration Register)

Field	Description	Reset	Access
31:27	Reserved	00H	RO
26	SFTERR: SMII Force Transmit Error When set, indicates to the PHY to force transmit error in the SMII frame being transmitted. This bit is reserved if SMII PHY port is not selected during core configuration.	0	R_W
25	CST: CRC stripping for Type frames When set, the last 4 bytes (FCS) of all frames of Ether type (type field greater than 0x0600) are stripped and dropped before forwarding the frame to the application. This function is not valid when the MAC receiver has IP Checksum Engine (Type 1) enabled.	0	R_W
24	TC: Transmit Configuration in RGMII/SGMII/SMII When set, this bit enables the transmission of duplex mode, link speed, and link up/down information to the PHY in the RGMII/SMII/SGMII ports. When this bit is reset, no such information is driven to the PHY. This bit is reserved (and RO) if neither RGMII nor SMII nor SGMII PHY port is selected during core configuration. The details of this feature are explained in the following sections: <ul style="list-style-type: none">• “Reduced Gigabit Media Independent Interface” on page 189• “Serial Media Independent Interface” on page 185• “Serial Gigabit Media Independent Interface” on page 195.	0	R_W

Table 6-29 Register 0 (MAC Configuration Register) (Continued)

Field	Description	Reset	Access
23	WD: Watchdog Disable When this bit is set, the GMAC disables the watchdog timer on the receiver, and can receive frames of up to 16,384 bytes. When this bit is reset, the GMAC allows no more than 2,048 bytes (10,240 if JE is set high) of the frame being received and cuts off any bytes received after that.	0	R_W
22	JD: Jabber Disable When this bit is set, the GMAC disables the jabber timer on the transmitter, and can transfer frames of up to 16,384 bytes. When this bit is reset, the GMAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.	0	R_W
21	BE: Frame Burst Enable When this bit is set, the GMAC allows frame bursting during transmission in GMII Half-Duplex mode. This bit is reserved (and RO) in 10/100 Mbps only or Full-Duplex-only configurations.	0	R_W
20	JE: Jumbo Frame Enable When this bit is set, GMAC allows Jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.	0	R_W
19:17	IFG: Inter-Frame Gap These bits control the minimum IFG between frames during transmission. 000 96 bit times 001 88 bit times 010 80 bit times ... 111 40 bit times In half-duplex mode, the minimum IFG can be configured for 64 bit times (IFG = 100) only. Lower values are not considered. In 1000-Mbps mode, the minimum IFG supported is 64 bit times (and above) in the GMAC-CORE configuration and 80 bit times (and above) in other system configurations.	000	R_W
16	DCRS: Disable Carrier Sense During Transmission When set high, this bit makes the MAC transmitter ignore the (G)MII CRS signal during frame transmission in Half-Duplex mode. This request results in no errors generated because of Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors because of Carrier Sense and can even abort the transmissions. This bit is reserved (and RO) when the core is selected for Full-Duplex-only operation during core configuration.	0	R_W

Table 6-29 Register 0 (MAC Configuration Register) (Continued)

Field	Description	Reset	Access
15	<p>PS: Port Select Selects between GMII and MII:</p> <p>0 GMII (1000 Mbps) 1 MII (10/100 Mbps)</p> <p>This bit is read-only with the appropriate value in the 10/100 Mbps-only (always 1) or 1000 Mbps-only (always 0) configurations, and R_W in the default 10/100/1000 Mbps configuration.</p>	0	R_W
14	<p>FES: Speed Indicates the speed in Fast Ethernet (MII) mode:</p> <p>0 10 Mbps 1 100 Mbps</p> <p>This bit is reserved (RO) by default and is enabled only when RMII/SMII/RGMII/SGMII/RevMII is enabled during configuration. This bit generates link speed encoding when TC (Bit 24) is set in RGMII/SMII/SGMII mode.</p> <p>This signal is always driven for RevMII. If the core is configured with an RMII, SMII, SGMII, or RGMII PHY interface, this signal can optionally be driven as an output signal (mac_speed_o[0]). In addition, this bit is reserved when RMII is enabled during configuration without enabling the "Output Port for speed selection" option.</p>	0	R_W
13	<p>DO: Disable Receive Own When this bit is set, the GMAC disables the reception of frames when the gmii_txen_o is asserted in Half-Duplex mode.</p> <p>When this bit is reset, the GMAC receives all packets that are given by the PHY while transmitting.</p> <p>This bit is not applicable if the GMAC is operating in Full-Duplex mode.</p> <p>This bit is reserved (RO with default value) if the GMAC is configured for Full-Duplex-only operation during configuration.</p>	0	R_W
12	<p>LM: Loopback Mode When this bit is set, the GMAC operates in loopback mode at GMII/MII. The (G)MII Receive clock input (clk_rx_i) is required for the loopback to work properly, as the Transmit clock is not looped-back internally.</p>	0	R_W
11	<p>DM: Duplex Mode When this bit is set, the GMAC operates in a Full-Duplex mode where it can transmit and receive simultaneously. This bit is RO with default value of 1'b1 in Full-Duplex-only configuration.</p>	0	R_W

Table 6-29 Register 0 (MAC Configuration Register) (Continued)

Field	Description	Reset	Access				
10	<p>IPC: Checksum Offload</p> <p>When this bit is set, the GMAC calculates the 16-bit one's complement of the one's complement sum of all received Ethernet frame payloads. It also checks whether the IPv4 Header checksum (assumed to be bytes 25–26 or 29–30 (VLAN-tagged) of the received Ethernet frame) is correct for the received frame and gives the status in the receive status word. The GMAC core also appends the 16-bit checksum calculated for the IP header datagram payload (bytes after the IPv4 header) and appends it to the Ethernet frame transferred to the application (when Type 2 COE is deselected).</p> <p>When this bit is reset, this function is disabled.</p> <p>When Type 2 COE is selected, this bit, when set, enables IPv4 checksum checking for received frame payloads' TCP/UDP/ICMP headers. When this bit is reset, the COE function in the receiver is disabled and the corresponding PCE and IP HCE status bits (see Table 3-8 on page 109) are always cleared.</p> <p>If the IP Checksum Offload feature is not enabled during coreConsultant configuration, this bit is reserved (RO with default value).</p>	0	R_W				
9	<p>DR: Disable Retry</p> <p>When this bit is set, the GMAC attempts only 1 transmission. When a collision occurs on the GMII/MII, the GMAC ignores the current frame transmission and report a Frame Abort with excessive collision error in the transmit frame status.</p> <p>When this bit is reset, the GMAC attempts retries based on the settings of BL (bit [6:5]). This bit is applicable only to Half-Duplex mode and is reserved (RO with default value) in Full-Duplex-only configuration.</p>	0	R_W				
8	<p>LUD: Link Up/Down</p> <p>Indicates whether the link is up or down during the transmission of configuration in RGMII/SGMII/SMII interface:</p> <table> <tr> <td>0</td> <td>Link Down</td> </tr> <tr> <td>1</td> <td>Link Up</td> </tr> </table> <p>This bit is reserved (RO with default value) and is enabled when RGMII/SGMII/SMII is enabled during configuration.</p>	0	Link Down	1	Link Up	0	R_W
0	Link Down						
1	Link Up						
7	<p>ACS: Automatic Pad/CRC Stripping</p> <p>When this bit is set, the GMAC strips the Pad/FCS field on incoming frames only if the length's field value is less than or equal to 1,500 bytes. All received frames with length field greater than or equal to 1,501 bytes are passed to the application without stripping the Pad/FCS field.</p> <p>When this bit is reset, the GMAC passes all incoming frames to the Host unmodified.</p>	0	R_W				

Table 6-29 Register 0 (MAC Configuration Register) (Continued)

Field	Description	Reset	Access
6:5	BL: Back-Off Limit The Back-Off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) the GMAC waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only to Half-Duplex mode and is reserved (RO) in Full-Duplex-only configuration. 00 $k = \min(n, 10)$ 01 $k = \min(n, 8)$ 10 $k = \min(n, 4)$ 11 $k = \min(n, 1)$ where n = retransmission attempt. The random integer r takes the value in the range $0 \leq r < 2^k$	00	R_W
4	DC: Deferral Check When this bit is set, the deferral check function is enabled in the GMAC. The GMAC issues a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status when the transmit state machine is deferred for more than 24,288 bit times in 10/100-Mbps mode. If the Core is configured for 1000 Mbps operation, or if the Jumbo frame mode is enabled in 10/100-Mbps mode, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active CRS (carrier sense) signal on the GMII/MII. Defer time is not cumulative. If the transmitter defers for 10,000 bit times, then transmits, collides, backs off, and then has to defer again after completion of back-off, the deferral timer resets to 0 and restarts. When this bit is reset, the deferral check function is disabled and the GMAC defers until the CRS signal goes inactive. This bit is applicable only in Half-Duplex mode and is reserved (RO) in Full-Duplex-only configuration.	0	R_W
3	TE: Transmitter Enable When this bit is set, the transmit state machine of the GMAC is enabled for transmission on the GMII/MII. When this bit is reset, the GMAC transmit state machine is disabled after the completion of the transmission of the current frame, and does not transmit any further frames.	0	R_W
2	RE: Receiver Enable When this bit is set, the receiver state machine of the GMAC is enabled for receiving frames from the GMII/MII. When this bit is reset, the GMAC receive state machine is disabled after the completion of the reception of the current frame, and does not receive any further frames from the GMII/MII.	0	R_W
1:0	Reserved	00	RO

6.2.2.2 Register 1 (MAC Frame Filter)

The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

Table 6-30 Register 1 (MAC Frame Filter)

Field	Description	Reset	Access
31	RA: Receive All When this bit is set, the GMAC Receiver module passes to the Application all frames received irrespective of whether they pass the address filter. The result of the SA/DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is reset, the Receiver module passes to the Application only those frames that pass the SA/DA address filter.	0	R_W
30:11	Reserved	0000_000H	RO
10	HPF: Hash or Perfect Filter When set, this bit configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by HMC or HUC bits. When low and if the HUC/HMC bit is set, the frame is passed only if it matches the Hash filter. This bit is reserved (and RO) if the Hash filter is not selected during core configuration.	0	R_W
9	SAF: Source Address Filter Enable The GMAC core compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison matches, then the SAMatch bit of RxStatus Word is set high. When this bit is set high and the SA filter fails, the GMAC drops the frame. When this bit is reset, then the GMAC Core forwards the received frame to the application and with the updated SA Match bit of the RxStatus depending on the SA address comparison.	0	R_W
8	SAIF: SA Inverse Filtering When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers are marked as failing the SA Address filter. When this bit is reset, frames whose SA does not match the SA registers are marked as failing the SA Address filter.	0	R_W
7:6	PCF: Pass Control Frames These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). Note that the processing of PAUSE control frames depends only on RFE of Flow Control Register[2]. depends only on RFE of Flow Control Register[2]. 00 GMAC filters all control frames from reaching the application. 01 GMAC forwards all control frames except PAUSE control frames to application even if they fail the Address filter. 10 GMAC forwards all control frames to application even if they fail the Address Filter. 11 GMAC forwards control frames that pass the Address Filter.	0	R_W
5	DBF: Disable Broadcast Frames When this bit is set, the AFM module filters all incoming broadcast frames. In addition, it overrides all other filter settings. When this bit is reset, the AFM module passes all received broadcast frames.	0	R_W

Table 6-30 Register 1 (MAC Frame Filter) (Continued)

Field	Description	Reset	Access
4	PM: Pass All Multicast When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed. When reset, filtering of multicast frame depends on HMC bit.	0	R_W
3	DAIF: DA Inverse Filtering When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames. When reset, normal filtering of frames is performed.	0	R_W
2	HMC: Hash Multicast When set, MAC performs destination address filtering of received multicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the DA field with the values programmed in DA registers. If Hash Filter is not selected during coreConsultant configuration, this bit is reserved (and RO).	0	R_W
1	HUC: Hash Unicast When set, MAC performs destination address filtering of unicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the DA field with the values programmed in DA registers. If Hash Filter is not selected during coreConsultant configuration, this bit is reserved (and RO).	0	R_W
0	PR: Promiscuous Mode When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address. The SA/DA Filter Fails status bits of the Receive Status Word are always cleared when PR is set.	0	R_W

6.2.2.3 Register 2 (Hash Table High Register)

The 64-bit Hash table is used for group address filtering. For hash filtering, the contents of the destination address in the incoming frame is passed through the CRC logic, and the upper 6 bits of the CRC register are used to index the contents of the Hash table. The most significant bit determines the register to be used (Hash Table High/Hash Table Low), and the other 5 bits determine which bit within the register. A hash value of 5b'00000 selects Bit 0 of the selected register, and a value of 5b'11111 selects Bit 31 of the selected register.

For example, if the DA of the incoming frame is received as 0x1F52419CB6AF (0x1F is the first byte received on GMII interface), then the internally calculated 6-bit Hash value is 0x2C and the HTL register bit[12] is checked for filtering. If the DA of the incoming frame is received as 0xA00A98000045, then the calculated 6-bit Hash value is 0x07 and the HTL register bit[7] is checked for filtering.



Note To help you program the hash table, a sample C routine that generates a DA's 6-bit hash has been included in your workspace's /sample_codes/ directory.

If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. If the PM (Pass All Multicast) bit is set in Register 1, then all multicast frames are accepted regardless of the multicast hash values.

If the Hash Table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table High/Low registers are written to. Please note that consecutive writes to these register should be performed only after at least 4 clock cycles in the destination clock domain when double-synchronization is enabled.

The Hash Table High register contains the higher 32 bits of the Hash table.

Table 6-31 Register 2 (Hash Table High Register)

Field	Description	Reset	Access
31:0	HTH: Hash Table High This field contains the upper 32 bits of Hash table.	0000_0000H	R_W

6.2.2.4 Register 3 (Hash Table Low Register)

The Hash Table Low register contains the lower 32 bits of the Hash table. Both Register 2 and Register 3 and corresponding HMC and HUC bits in Filter Register are reserved if the Hash Filter Function is disabled during coreConsultant configuration.

Table 6-32 Register 3 (Hash Table Low Register)

Field	Description	Reset	Access
31:0	HTL: Hash Table Low This field contains the lower 32 bits of Hash table.	0000_0000H	R_W

6.2.2.5 Register 4 (GMII Address Register)

The GMII Address register controls the management cycles to the external PHY through the management interface.

Table 6-33 Register 4 (GMII Address Register)

Field	Description	Reset	Access
31:16	Reserved	0000H	RO
15:11	PA: Physical Layer Address This field tells which of the 32 possible PHY devices are being accessed. For RevMII, this field gives the PHY Address of the RevMII module.	00H	R_W
10:6	GR: GMII Register These bits select the desired GMII register in the selected PHY device. For RevMII, these bits select the desired CSR register in RevMII Registers set.	00H	R_W

Table 6-33 Register 4 (GMII Address Register) (Continued)

Field	Description	Reset	Access
5:2	CR: CSR Clock Range The CSR Clock Range selection determines the frequency of the MDC clock as per the clk_csr_i frequency used in your design. The suggested range of clk_csr_i frequency applicable for each value below (when Bit[5] = 0) ensures that the MDC clock is approximately between the frequency range 1.0 MHz - 2.5 MHz.	0000	R_W
	Selection clk_csr_i MDC Clock		
0000	60-100 MHz	clk_csr_i/42	
0001	100-150 MHz	clk_csr_i/62	
0010	20-35 MHz	clk_csr_i/16	
0011	35-60 MHz	clk_csr_i/26	
0100	150-250 MHz	clk_csr_i/102	
0101	250-300 MHz	clk_csr_i/124	
0110, 0111	Reserved		
	When bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE 802.3 specified frequency limit of 2.5 MHz and program a clock divider of lower value. For example, when clk_csr_i is of frequency 100 Mhz and you program these bits as "1010", then the resultant MDC clock is of 12.5 Mhz which is outside the limit of IEEE 802.3 specified range. Please program the values given below only if the interfacing chips supports faster MDC clocks.		
	Selection MDC Clock		
1000	clk_csr_i/4		
1001	clk_csr_i/6		
1010	clk_csr_i/8		
1011	clk_csr_i/10		
1100	clk_csr_i/12		
1101	clk_csr_i/14		
1110	clk_csr_i/16		
1111	clk_csr_i/18		
	These bits are not used for accesssing RevMII. These bits are read-only if the RevMII interface is selected as single PHY interface in coreConsultant.		
1	GW: GMII Write When set, this bit tells the PHY/RevMII that this is a Write operation using the GMII Data register. If this bit is not set, this is a Read operation, placing the data in the GMII Data register.	0	R_W

Table 6-33 Register 4 (GMII Address Register) (Continued)

Field	Description	Reset	Access
0	<p>GB: GMII Busy</p> <p>This bit should read a logic 0 before writing to Register 4 and Register 5. During a PHY/RevMII register access, the Software sets this bit to 1'b1 to indicate that a Read or Write access is in progress.</p> <p>The Register 5 is invalid until this bit is cleared by the GMAC. Therefore, Register 5 (GMII Data) should be kept valid until the GMAC clears this bit during a PHY Write operation. Similarly for read operation, Register 5 contents are not valid until this bit is cleared.</p> <p>The subsequent read/write operation should happen only after the previous operation is complete.</p>	0	R_WS_SC

6.2.2.6 Register 5 (GMII Data Register)

The GMII Data register stores Write data to be written to the PHY register located at the address specified in Register 4. Register 5 also stores Read data from the PHY register located at the address specified by Register 4.

Table 6-34 Register 5 (GMII Data Register)

Field	Description	Reset	Access
31:16	Reserved	0000H	RO
15:0	<p>GD: GMII Data</p> <p>This contains the 16-bit data value read from the PHY/RevMII after a Management Read operation or the 16-bit data value to be written to the PHY/RevMII before a Management Write operation.</p>	0000H	R_W

6.2.2.7 Register 6 (Flow Control Register)

The Flow Control register controls the generation and reception of the Control (Pause Command) frames by the GMAC's Flow control module. A Write to a register with the Busy bit set to '1' triggers the Flow Control block to generate a Pause Control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Busy bit remains set until the control frame is transferred onto the cable. The Host must make sure that the Busy bit is cleared before writing to the register.

Table 6-35 Register 6 (Flow Control Register)

Field	Description	Reset	Access
31:16	<p>PT: Pause Time</p> <p>This field holds the value to be used in the Pause Time field in the transmit control frame. If the Pause Time bits is configured to be double-synchronized to the (G)MII clock domain, then consecutive writes to this register should be performed only after at least 4 clock cycles in the destination clock domain.</p>	0000H	R_W
15:8	Reserved	000H	RO

Table 6-35 Register 6 (Flow Control Register) (Continued)

Field	Description	Reset	Access
7	DZPQ: Disable Zero-Quanta Pause When set, this bit disables the automatic generation of Zero-Quanta Pause Control frames on the de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i/mti_flowctrl_i). When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.	0	R_W
6	Reserved	0	RO
5:4	PLT: Pause Low Threshold This field configures the threshold of the PAUSE timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of PAUSE Frame. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot-times), and PLT = 01, then a second PAUSE frame is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256 – 28) slot-times after the first PAUSE frame is transmitted. Selection Threshold 00 Pause time minus 4 slot times 01 Pause time minus 28 slot times 10 Pause time minus 144 slot times 11 Pause time minus 256 slot times	00	R_W
	The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII/MII interface.		
3	UP: Unicast Pause Frame Detect When this bit is set, the GMAC detects the Pause frames with the station's unicast address specified in MAC Address0 High Register and MAC Address0 Low Register, in addition to the detecting Pause frames with the unique multicast address. When this bit is reset, the GMAC detects only a Pause frame with the unique multicast address specified in the 802.3x standard.	0	R_W
2	RFE: Receive Flow Control Enable When this bit is set, the GMAC decodes the received Pause frame and disable its transmitter for a specified (Pause Time) time. When this bit is reset, the decode function of the Pause frame is disabled.	0	R_W
1	TFE: Transmit Flow Control Enable In Full-Duplex mode, when this bit is set, the GMAC enables the flow control operation to transmit Pause frames. When this bit is reset, the flow control operation in the GMAC is disabled, and the GMAC does not transmit any Pause frames. In Half-Duplex mode, when this bit is set, the GMAC enables the back-pressure operation. When this bit is reset, the back-pressure feature is disabled.	0	R_W

Table 6-35 Register 6 (Flow Control Register) (Continued)

Field	Description	Reset	Access
0	<p>FCB/BPA: Flow Control Busy/Backpressure Activate</p> <p>This bit initiates a Pause Control frame in Full-Duplex mode and activates the backpressure function in Half-Duplex mode if TFE bit is set.</p> <p>In Full-Duplex mode, this bit should be read as 1'b0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 1'b1. During a transfer of the Control Frame, this bit continues to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the GMAC resets this bit to 1'b0. The Flow Control register should not be written to until this bit is cleared.</p> <p>In Half-Duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the GMAC Core. During backpressure, when the GMAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically OR'ed with the mti_flowctrl_i input signal for the backpressure function. When the GMAC is configured to Full-Duplex mode, the BPA is automatically disabled.</p>	0	R_WS_SC(FCB) R_W (BPA)

6.2.2.8 Register 7 (VLAN Tag Register)

The VLAN Tag register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16'h8100, and the following 2 bytes are compared with the VLAN tag; if a match occurs, it sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1518 bytes to 1522 bytes.

If the VLAN Tag register is configured to be double-synchronized to the (G)MII clock domain, then consecutive writes to these register should be performed only after at least 4 clock cycles in the destination clock domain.

Table 6-36 Register 7 (VLAN Tag Register)

Field	Description	Reset	Access
31:17	Reserved	0000H	RO
16	<p>ETV: Enable 12-Bit VLAN Tag Comparison</p> <p>When this bit is set, a 12-bit VLAN identifier, rather than the complete 16-bit VLAN tag, is used for comparison and filtering. Bits[11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged frame.</p> <p>When this bit is reset, all 16 bits of the received VLAN frame's fifteenth and sixteenth bytes are used for comparison.</p>	0	R/W
15:0	<p>VL: VLAN Tag Identifier for Receive Frames</p> <p>This contains the 802.1Q VLAN tag to identify VLAN frames, and is compared to the fifteenth and sixteenth bytes of the frames being received for VLAN frames. Bits[15:13] are the User Priority, Bit[12] is the Canonical Format Indicator (CFI) and bits[11:0] are the VLAN tag's VLAN Identifier (VID) field. When the ETV bit is set, only the VID (Bits[11:0]) is used for comparison.</p> <p>If VL (VL[11:0] if ETV is set) is all zeros, the GMAC does not check the fifteenth and sixteenth bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 to be VLAN frames.</p>	0000H	R_W

6.2.2.9 Register 8 (Version Register)

The Version register's contents identify the version of the core. This register contains two bytes, one that Synopsys uses to identify the core release number, and the other that you set during coreConsultant configuration.

Table 6-37 Register 8 (Version Register)

Field	Description	Reset	Access
15:8	User-defined version (configured with coreConsultant)	xxH	RO
7:0	Synopsys-defined version (3.6)	36H	RO

6.2.2.10 Register 9 (Debug Register)

This debug register gives the status of all the main modules of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC core is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.

Table 6-38 Register 9 (Debug Register)

Field	Description	Reset	Access
31:26	Reserved	0	RO
25	When high, it indicates that the MTL TxStatus FIFO is full and hence the MTL cannot accept any more frames for transmission. This bit is reserved in the GMAC-AHB and GMAC-DMA configurations.	0	RO
24	When high, it indicates that the MTL TxFIFO is not empty and has some data left for transmission.	0	RO
23	Reserved	0	RO
22	When high, it indicates that the MTL TxFIFO Write Controller is active and transferring data to the TxFIFO.	0	RO
21:20	This indicates the state of the TxFIFO read Controller:	0	RO
00	IDLE state		
01	READ state (transferring data to MAC transmitter)		
10	Waiting for TxStatus from MAC transmitter		
11	Writing the received TxStatus or flushing the TxFIFO		
19	When high, it indicates that the MAC transmitter is in PAUSE condition (in full-duplex only) and hence does not schedule any frame for transmission	0	RO
18:17	This indicates the state of the MAC Transmit Frame Controller module:	0	RO
00	IDLE state		
01	Waiting for Status of previous frame or IFG/backoff period to be over		
10	Generating and transmitting a PAUSE control frame (in full duplex mode)		
11	Transferring input frame for transmission		
16	When high, it indicates that the MAC GMII/MII transmit protocol engine is actively transmitting data and not in IDLE state.	0	RO

Table 6-38 Register 9 (Debug Register) (Continued)

Field	Description	Reset	Access
15:10	Reserved	0	RO
9:8	This gives the status of the RxFIFO Fill-level: 00 RxFIFO Empty 01 RxFIFO fill-level below flow-control de-activate threshold 10 RxFIFO fill-level above flow-control activate threshold 11 RxFIFO Full	0	RO
7	Reserved	0	RO
6:5	It gives the state of the RxFIFO read Controller: 00 IDLE state 01 Reading frame data 10 Reading frame status (or time-stamp) 11 Flushing the frame data and Status	0	RO
4	When high, it indicates that the MTL RxFIFO Write Controller is active and transferring a received frame to the FIFO.	0	RO
3	Reserved	0	RO
2:1	When high, it indicates the active state of the small FIFO Read and Write controllers respectively of the MAC receive Frame Controller module (section 3.7.4.4)	0	RO
0	When high, it indicates that the MAC GMII/MII receive protocol engine is actively receiving data and not in IDLE state.	0	RO

6.2.2.11 Register 12 (LPI Control and Status Register)

This register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read. This register is present only when you select the Energy Efficient Ethernet feature in coreConsultant.

Table 6-39 Register 12 (LPI Control and Status Register)

Field	Description	Reset	Access
31:19	Reserved	0	RO
18	PLSEN: PHY Link Status Enable This bit enables the link status received on the RGMII, SGMII, or SMII receive paths (see Register 54 (SGMII/RGMII/SMII Status Register)) to be used for activating the LPI LS TIMER. When set, the MAC uses the link-status bits of Register 54 and the PLS bit for the LPI LS Timer trigger. When cleared, the MAC ignores the link-status bits of Register 54. This bit is RO and reserved if you have not selected RGMII, SGMII, or SMII PHY interface.	0	R_W

Table 6-39 Register 12 (LPI Control and Status Register)

Field	Description	Reset	Access
17	PLS: PHY Link Status This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (okay) at least for the time indicated by the LPI LS TIMER. When set, the link is considered to be okay (up) and when reset, the link is considered to be down.	0	R_W
16	LPIEN: LPI Enable When set, this bit instructs the MAC Transmitter to enter the LPI state. When reset, this bit instructs the MAC to exit the LPI state and resume normal transmission.	0	R_W
15:10	Reserved	0	RO
9	RLPIST: Receive LPI State When set, this bit indicates that the MAC is receiving LPI pattern on the GMII/MII interface.	0	RO
8	TLPIST: Transmit LPI State When set, this bit indicates that the MAC is transmitting LPI pattern on the GMII/MII interface.	0	RO
7:4	Reserved	0	RO
3	RLPIEX: Receive LPI Exit When set, this bit indicates that the MAC Receiver has stopped receiving the LPI pattern on GMII/MII, has exited the LPI state, and has resumed the normal reception. This bit is cleared by a read into this register.	0	R_SS_RC
2	RLPIEN: Receive LPI Entry When set, this bit indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register.	0	R_SS_RC
1	TLPIEX: Transmit LPI Exit When set, this bit indicates that the MAC transmitter has exited the LPI state after the user has cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register.	0	R_SS_RC
0	TLPIEN: Transmit LPI Entry When set, this bit indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register.	0	R_SS_RC

6.2.2.12 Register 13 (LPI Timers Control Register)

This register controls the timeout values in LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission. This register is present only when you select the Energy Efficient Ethernet feature in coreConsultant.

Table 6-40 Register 13 (LPI Timers Control Register)

Field	Description	Reset	Access
31:26	Reserved	0	RO
25:16	LIT: LPI LS TIMER These bits specify the minimum time (in milliseconds) for which the link-status from the PHY should be up (okay) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.	0x3E8	R_W
15:0	TWT: LPI TW TIMER These bits specify the minimum time (in microseconds) for which the MAC waits after it has stopped transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.	0	R_W

6.2.2.13 Register 14 (Interrupt Status Register)

The Interrupt Status register contents identify the events in the GMAC-CORE that can generate interrupt. Note that all the interrupt events are generated only when the corresponding optional feature is selected during coreConsultant configuration and enabled during operation. Hence, these bits are reserved when the corresponding features are not present in the core.

Table 6-41 Register 14 (Interrupt Status Register)

Field	Description	Reset	Access
31:11	Reserved	0	RO
10	LPIIS: LPI Interrupt Status When the Energy Efficient Ethernet feature is enabled, this bit is set for any LPI state entry or exit in MAC Transmitter or Receiver. This bit is cleared on reading the byte 0 of Register 12 (LPI Control and Status Register) . In all other modes, this bit is reserved.	0	RO
9	Timestamp Interrupt Status When Advanced Timestamp feature is enabled, this bit is set when any of the following conditions is true: <ul style="list-style-type: none">• The system time value equals or exceeds the value specified in the Target Time High and Low registers• There is an overflow in the seconds register• The Auxiliary snapshot trigger is asserted This bit is cleared on reading the byte 0 of the “Timestamp Status register (see “Register 458 (Timestamp Status Register)” on page 351). Otherwise, when default Time stamping is enabled, this bit when set indicates that the system time value equals or exceeds the value specified in the Target Time registers. In this mode, this bit is cleared after the completion of the read of this Interrupt Status Register[9]. In all other modes, this bit is reserved.	0	RO/ R_SS_RC
8	Reserved	000H	RO

Table 6-41 Register 14 (Interrupt Status Register) (Continued)

Field	Description	Reset	Access
7	MMC Receive Checksum Offload Interrupt Status This bit is set high whenever an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register . This bit is cleared when all the bits in this interrupt register are cleared. This bit is valid only when you select the optional MMC module and Checksum Offload Engine (Type 2) during configuration.	0	RO
6	MMC Transmit Interrupt Status This bit is set high whenever an interrupt is generated in the MMC Transmit Interrupt Register . This bit is cleared when all the bits in this interrupt register are cleared. This bit is valid only when you select the optional MMC module during configuration.	0	RO
5	MMC Receive Interrupt Status This bit is set high whenever an interrupt is generated in the MMC Receive Interrupt Register . This bit is cleared when all the bits in this interrupt register are cleared. This bit is valid only when you select the optional MMC module during configuration.	0	RO
4	MMC Interrupt Status This bit is set high whenever any of bits 7:5 is set high and cleared only when all of these bits are low. This bit is valid only when you select the optional MMC module during configuration.	0	RO
3	PMT Interrupt Status This bit is set whenever a Magic packet or Wake-on-LAN frame is received in Power-Down mode (See bits 5 and 6 in the “PMT Control and Status Register” on page 160). This bit is cleared when both bits[6:5] are cleared because of a read operation to the PMT Control and Status register. This bit is valid only when you select the optional PMT module during configuration.	0	RO
2	PCS Auto-Negotiation Complete This bit is set when the Auto-negotiation is completed in the TBI/RTBI/SGMII PHY interface (Bit 5 in Register 49 (AN Status Register)). This bit is cleared when the user makes a read operation to the AN Status register. This bit is valid only when you select the optional TBI/RTBI/SGMII PHY interface during configuration and operation.	0	RO
1	PCS Link Status Changed This bit is set because of any change in Link Status in the TBI/RTBI/SGMII PHY interface (Bit 2 in Register 49 (AN Status Register)). This bit is cleared when the user makes a read operation the AN Status register. This bit is valid only when you select the optional TBI/RTBI/SGMII PHY interface during configuration and operation.	0	RO
0	RGMII/SMII Interrupt Status This bit is set because of any change in value of the Link Status of RGMII/SMII interface (Bit 3 in Register 54 (SGMII/RGMII/SMII Status Register)). This bit is cleared when the user makes a read operation the SGMII/RGMII/SMII Status register. This bit is valid only when you select the optional RGMII/SMII PHY interface during configuration and operation.	0	RO

6.2.2.14 Register 15 (Interrupt Mask Register)

The Interrupt Mask Register bits enables the user to mask the interrupt signal because of the corresponding event in the Interrupt Status Register. The interrupt signal is sbd_intr_o in GMAC-AHB, GMAC-AXI, and GMAC-DMA configuration while the interrupt signal is mci_intr_o in the GMAC-MTL and GMAC-CORE configuration.

Table 6-42 Register 15 (Interrupt Mask Register)

Field	Description	Reset	Access
31:11	Reserved	0	RO
10	LPIIM: LPI Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of LPI Interrupt Status bit in Register 14 (Interrupt Status Register) . This bit is valid only when you select the Energy Efficient Ethernet feature during configuration. In all other modes, this bit is reserved.	0	R_W
9	Timestamp Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of Timestamp Interrupt Status bit in Register 14 (Interrupt Status Register) . This bit is valid only when IEEE1588 time stamping is enabled. In all other modes, this bit is reserved.	0	R_W
8:4	Reserved	000H	RO
3	PMT Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of PMT Interrupt Status bit in Register 14 (Interrupt Status Register) .	0	R_W
2	PCS AN Completion Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of PCS Auto-negotiation complete bit in Register 14 (Interrupt Status Register) caused because of the completion of Auto-negotiation event.	0	R_W
1	PCS Link Status Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of the PCS Link-status changed bit in Register 14 (Interrupt Status Register) caused because of change in link-status event.	0	R_W
0	RGMII/SMII Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of RGMII/SMII Interrupt Status bit in Register 14 (Interrupt Status Register) .	0	R_W

6.2.2.15 Register 16 (MAC Address0 High Register)

The MAC Address0 High register holds the upper 16 bits of the 6-byte first MAC address of the station. Note that the first DA byte that is received on the (G)MII interface corresponds to the LS Byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 is the first byte) on the (G)MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211.

If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address Low Register (Register 17) are written to. Please note that consecutive writes to

this Address Low Register should be performed only after at least 4 clock cycles in the destination clock domain for proper synchronization updates.

Table 6-43 Register 16 (MAC Address0 High Register)

Field	Description	Reset	Access
31	MO: Always 1.	1	RO
30:16	Reserved	0000H	RO
15:0	A[47:32]: MAC Address0 [47:32] This field contains the upper 16 bits (47:32) of the 6-byte first MAC address. This is used by the MAC for filtering for received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.	FFFFH	R_W

6.2.2.16 Register 17 (MAC Address0 Low Register)

The MAC Address0 Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

Table 6-44 Register 17 (MAC Address0 Low Register)

Field	Description	Reset	Access
31:0	A[31:0]: MAC Address0 [31:0] This field contains the lower 32 bits of the 6-byte first MAC address. This is used by the MAC for filtering for received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.	FFFF_FFFFH	R_W

6.2.2.17 Register 18 (MAC Address1 High Register)

The MAC Address1 High register holds the upper 16 bits of the 6-byte second MAC address of the station.

If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address Low Register (Register 19) are written to. Consecutive writes to this Address Low Register must be performed only after at least 4 clock cycles in the destination clock domain for proper synchronization updates.

Table 6-45 Register 18 (MAC Address1 High Register)

Field	Description	Reset	Access
31	AE: Address Enable When this bit is set, the Address filter module uses the second MAC address for perfect filtering. When reset, the address filter module ignores the address for filtering.	0	R_W
30	SA: Source Address When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address1[47:0] is used to compare with the DA fields of the received frame.	0	R_W

Table 6-45 Register 18 (MAC Address1 High Register)

Field	Description	Reset	Access
29:24	MBC: Mask Byte Control These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the GMAC core does not compare the corresponding byte of received DA/SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: Bit 29 Register 18[15:8] Bit 28 Register 18[7:0] Bit 27 Register 19[31:24] ... Bit 24 Register 19[7:0]	000000	R_W
23:16	Reserved	00H	RO
15:0	A[47:32]: MAC Address1 [47:32] This field contains the upper 16 bits (47:32) of the 6-byte second MAC address.	FFFFH	R_W

6.2.2.18 Register 19 (MAC Address1 Low Register)

The MAC Address1 Low register holds the lower 32 bits of the 6-byte second MAC address of the station.

Table 6-46 Register 19 (MAC Address1 Low Register)

Field	Description	Reset	Access
31:0	A[31:0]: MAC Address1 [31:0] This field contains the lower 32 bits of the 6-byte second MAC address. The content of this field is undefined until loaded by the Application after the initialization process.	FFFF_FFFFH	R_W



- The descriptions for registers 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, and 46 (MAC Address2 High Register through MAC Address15 High Register) are the same as for the Register 18 MAC Address1 High Register.
- The descriptions for registers 21, 23, 25, 27, 29, 31, 33, 35, 37, 38, 41, 43, 45, and 47 (MAC Address2 Low Register through MAC Address15 Low Register) are the same as for the Register 19 MAC Address1 Low Register.
- The descriptions for registers 512, 514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536, 538, 540, and 542 (MAC Address16 High Register through MAC Address31 High Register) are the same as for the Register 18 MAC Address1 High Register.
- The descriptions for registers 513, 515, 517, 519, 521, 523, 525, 527, 529, 531, 533, 535, 537, 539, 541, and 543 (MAC Address16 Low Register through MAC Address31 Low Register) is the same as for the Register 19 MAC Address1 Low Register.

6.2.2.19 Register 48 (AN Control Register)

The AN Control register enables and/or restarts auto-negotiation. It also enables PCS loopback. This register is optional and is present only when the core is configured for TBI/SGMII/RTBI PHY interfaces.

Table 6-47 Register 48 (AN Control Register)

Field	Description	Reset	Access
31:19	Reserved	0000H	RO
18	SGMII RAL Control When set, this bit forces the SGMII RAL block to operate in the speed configured in the MAC Configuration register's Speed and Port Select bits. This is useful when the SGMII interface is used in a direct MAC-MAC connection (without a PHY) and any MAC must reconfigure the speed. When reset, the SGMII RAL block operates according to the link speed status received on the SGMII (from the PHY). This bit is reserved (and RO) if the SGMII PHY interface is not selected during coreConsultant configuration.	0	R_W
17	LR: Lock to Reference When set, enables the PHY to lock its PLL to the 125 MHz reference clock. This bit controls the pcs_lck_ref_o signal on the TBI/RTBI/SGMII.	0	R_W
16	ECD: Enable Comma Detect When set, enables the PHY for comma detect and word resynchronization. This bit controls the pcs_en_cdet_o signal on the TBI/RTBI/SGMII.	0	R_W
15	Reserved	0	RO
14	ELE: External Loopback Enable When set, this bit causes the PHY to loopback the transmit data into the receive path. The pcs_ewrap_o signal is asserted high when this bit is set.	0	R_W
13	Reserved	0	RO
12	ANE: Auto-Negotiation Enable When set, this bit enables the GMAC to perform auto-negotiation with the link partner. Clearing this bit disables auto-negotiation.	0	R_W
11:10	Reserved	00	RO
9	RAN: Restart Auto-Negotiation When set, this bit causes auto-negotiation to restart if the ANE is set. This bit is self-clearing after auto-negotiation starts. This bit should be cleared for normal operation.	0	R_WS_SC
8:0	Reserved	00H	RO

6.2.2.20 Register 49 (AN Status Register)

The AN Status register indicates the link and the auto-negotiation status. This register is optional and is present only when the core is configured for TBI/SGMII/RTBI PHY interfaces.

Table 6-48 Register 49 (AN Status Register)

Field	Description	Reset	Access
31:9	Reserved	00_0000H	RO

Table 6-48 Register 49 (AN Status Register) (Continued)

Field	Description	Reset	Access
8	ES: Extended Status This bit is tied to high, because the GMAC supports extended status information in Register 53.	1	RO
7:6	Reserved	00	RO
5	ANC: Auto-Negotiation Complete When set, this bit indicates that the auto-negotiation process is completed. This bit is cleared when auto-negotiation is reinitiated.	0	RO
4	Reserved	0	RO
3	ANA: Auto-Negotiation Ability This bit is always high, because the GMAC supports auto-negotiation.	1	RO
2	LS: Link Status When set, this bit indicates that the link is up. When cleared, this bit indicates that the link is down.	0	R_SS_SC_LLO
1:0	Reserved	00	RO



LLO indicates that when the bit is reset to 1'b0, its value is registered until the application reads the register. This bit is updated only after the read.

6.2.2.21 Register 50 (Auto-Negotiation Advertisement Register)

The Auto-Negotiation Advertisement register indicates the link and the auto-negotiation status. This register is optional and is present only when the core is configured for TBI/RTBI PHY interfaces.

Table 6-49 Register 50 (Auto-Negotiation Advertisement Register)

Field	Description	Reset	Access
31:16	Reserved	0000H	RO
15	NP: Next Page Support This bit is tied to low, because the GMAC does not support the next page.	0	RO
14	Reserved	0	RO
13:12	RFE: Remote Fault Encoding These 2 bits provide a remote fault encoding, indicating to a link partner that a fault or error condition has occurred. The encoding of these 2 bits is defined in IEEE 802.3z, Section 37.2.1.5.	00	R_W
11:9	Reserved	000	RO

Table 6-49 Register 50 (Auto-Negotiation Advertisement Register) (Continued)

Field	Description	Reset	Access
8:7	PSE: Pause Encoding These 2 bits provide an encoding for the PAUSE bits, indicating that the GMAC is capable of configuring the PAUSE function as defined in IEEE 802.3x. The encoding of these 2 bits is defined in IEEE 802.3z, Section 37.2.1.4.	11	R_W
6	HD: Half-Duplex This bit, when set high, indicates that the GMAC supports Half-Duplex. This bit is tied to low (and RO) when the GMAC is configured for Full-Duplex-only operation.	1	R_W
5	FD: Full-Duplex This bit, when set high, indicates that the GMAC supports Full-Duplex.	1	R_W
4:0	Reserved	00000	RO

6.2.2.22 Register 51 (Auto-Negotiation Link Partner Ability Register)

The Auto-Negotiation Link Partner Ability register contains the advertised ability of the link partner. This register is optional and is present only when the core is configured for TBI/RTBI PHY interfaces.

Table 6-50 Register 51 (Auto-Negotiation Link Partner Ability Register)

Field	Description	Reset	Access
31:16	Reserved	0000H	RO
15	NP: Next Page Support When set, this bit indicates that more next page information is available. When cleared, this bit indicates that next page exchange is not desired.	0	RO
14	ACK: Acknowledge When set, this bit is used by the auto-negotiation function to indicate that the link partner has successfully received the GMAC's base page. When cleared, it indicates that a successful receipt of the base page has not been achieved.	0	RO
13:12	RFE: Remote Fault Encoding These 2 bits provide a remote fault encoding, indicating a fault or error condition of the link partner. Their encoding is defined in IEEE 802.3z, Section 37.2.1.5.	00	RO
11:9	Reserved	000	RO
8:7	PSE: Pause Encoding These 2 bits provide an encoding for the PAUSE bits, indicating that the link partner's capability of configuring the PAUSE function as defined in IEEE 802.3x. The encoding of these 2 bits is defined in IEEE 802.3z, Section 37.2.1.4.	00	RO
6	HD: Half-Duplex When set, this bit indicates that the link partner has the ability to operate in Half-Duplex mode. When cleared, the link partner does not have the ability to operate in Half-Duplex mode.	0	RO

Table 6-50 Register 51 (Auto-Negotiation Link Partner Ability Register) (Continued)

Field	Description	Reset	Access
5	FD: Full-Duplex When set, this bit indicates that the link partner has the ability to operate in Full-Duplex mode. When cleared, the link partner does not have the ability to operate in Full-Duplex mode.	0	RO
4:0	Reserved	00000	RO

6.2.2.23 Register 52 (Auto-Negotiation Expansion Register)

The Auto-Negotiation Expansion register indicates whether a new base page from the link partner has been received. This register is optional and is present only when the core is configured for TBI/RTBI PHY interfaces.

Table 6-51 Register 52 (Auto-Negotiation Expansion Register)

Field	Description	Reset	Access
31:3	Reserved	0000_0000H	RO
2	NPA: Next Page Ability This bit is tied to low, because the GMAC does not support next page function.	0	RO
1	NPR: New Page Received When set, this bit indicates that a new page has been received by the GMAC. This bit is cleared when read.	0	RO
0	Reserved	0	RO

6.2.2.24 Register 53 (TBI Extended Status Register)

The TBI Extended Status register indicates all modes of operation of the GMAC. This register is optional and is present only when the core is configured for TBI/RTBI PHY interfaces.

Table 6-52 Register 53 (TBI Extended Status Register)

Field	Description	Reset	Access
31:16	Reserved	0000H	RO
15	GFD: 1000BASE-X Full-Duplex Capable This bit indicates that the GMAC is able to perform Full-Duplex, 1000BASE-X operations.	1	RO
14	GHD: 1000BASE-X Half-Duplex Capable This bit indicates that the GMAC is able to perform Half-Duplex, 1000BASE-X operations. This bit is tied to low when the GMAC is configured for Full-Duplex-only operation during coreConsultant configuration.	1	RO
13:0	Reserved	0000H	RO

6.2.2.25 Register 54 (SGMII/RGMII/SMII Status Register)

The SGMII/RGMII/SMII Status register indicates the status signals received by the SGMII/RGMII/SMII (whichever is selected at reset) from the PHY. This register is optional and is present only when the core is configured for RGMII/SGMII PHY interfaces.

Table 6-53 Register 54 (SGMII/RGMII/SMII Status Register)

Field	Description	Reset	Access
31:6	Reserved	000_0000H	RO
5	False Carrier Detected Indicates whether SMII PHY detected false carrier (1'b1). Reserved when the core is configured for SGMII/RGMII PHY interfaces.	0	RO
4	Jabber Timeout Indicates whether there was jabber timeout error (1'b1) in received frame. Reserved when the core is configured for SGMII/RGMII PHY interfaces.	0	RO
3	Link Status Indicates whether the link is up (1'b1) or down (1'b0).	0	RO
2:1	Link Speed Indicates the current speed of the link: 00 2.5 MHz 01 25 MHz 10 125 MHz Bit 2 is reserved when the core is configured for SMII PHY interface.	<ul style="list-style-type: none"> • 10 for SGMII • 00 for RGMII/SMII 	RO
0	Link Mode Indicates the current mode of operation of the link: 1'b0 Half-Duplex mode 1'b1 Full-Duplex mode	0	RO

6.2.3 IEEE 1588 Timestamp Registers

This section describes the updated registers required to support the IEEE 1588 functions. These registers, described at a high level in [Table 6-4](#), are described in detail in [Tables 6-54](#) through [6-69](#).

6.2.3.1 Register 448 (Timestamp Control Register)

This register controls the operation of the System Time generator and the processing of PTP packets for time-stamping in the Receiver.



- Bits [5:1] are reserved when External Timestamp Input is selected during configuration. Bits[19:8] are reserved and read-only when Advanced Timestamp feature is NOT enabled.
- In the current release, the functions of bits 17 and 16 (SNAPTPSEL) have changed. These functions are not backward compatible with the functions described in release 3.50a.

Table 6-54 Register 448 (Timestamp Control Register)

Field	Description	Reset	Access
31:20	Reserved	000H	RO
19	ATSF: Auxiliary Snapshot FIFO Clear When set, it resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, auxiliary snapshots gets stored in the FIFO. This bit is reserved when "Add IEEE 1588 Auxiliary Snapshot" is not selected.	0	R_WS_SC
18	TSENMACADDR: Enable MAC address for PTP frame filtering When set, uses the DA MAC address (that matches any MAC Address register) to filter the PTP frames when PTP is sent directly over Ethernet.	0	R_W
17:16	SNAPTYPSEL: Select PTP packets for taking snapshots These bits along with bit 15 and 14 decide the set of PTP packet types for which snapshot needs to be taken. The encoding is given in Table 6-55 .	0	R_W
15	TSMSTRENA: Enable Snapshot for Messages Relevant to Master When set, the snapshot is taken for messages relevant to master node only else snapshot is taken for messages relevant to slave node.	0	R_W
14	TSEVNTEA: Enable Timestamp Snapshot for Event Messages When set, the timestamp snapshot is taken for event messages only (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see Table 6-55 .	0	R_W
13	TSIPV4ENA: Enable Processing of PTP frames sent over IPv4-UDP When set, the GMAC receiver processes the PTP packets encapsulated in UDP over IPv4 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv4 packets. This bit is set by default.	1	R_W
12	TSIPV6ENA: Enable Processing of PTP frames sent over IPv6-UDP When set, the GMAC receiver processes PTP packets encapsulated in UDP over IPv6 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv6 packets.	0	R_W
11	TSIPENA: Enable Processing of PTP over Ethernet frames When set, the GMAC receiver processes the PTP packets encapsulated directly in the Ethernet frames. When this bit is clear, the MAC ignores PTP over Ethernet packets.	0	R_W
10	TSVER2ENA: Enable PTP packet processing for version 2 format When set, the PTP packets are processed using the 1588 version 2 format else processed using the version 1 format (IEEE 1588 Version 1 and Version 2 format are described in " PTP Processing and Control " on page 124).	0	R_W

Table 6-54 Register 448 (Timestamp Control Register) (Continued)

Field	Description	Reset	Access
9	TSCTRLSSR: Timestamp Digital or Binary rollover control When set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (i.e., 1 nanosecond accuracy) and increments the timestamp (High) seconds. When reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment has to be programmed correctly depending on the PTP reference clock frequency and this bit value.	0	R_W
8	TSENALL: Enable Timestamp for All Frames When set, the timestamp snapshot is enabled for all frames received by the core.	0	R_W
7:6	Reserved		
5	TSADDREG: Addend Reg Update When set, the contents of the Timestamp Addend register is updated in the PTP block for fine correction. This is cleared when the update is completed. This register bit should be zero before setting it. This is a reserved bit when only coarse correction option is selected.	0	R_WS_SC
4	TSTRIG: Timestamp Interrupt Trigger Enable When set, the timestamp interrupt is generated when the System Time becomes greater than the value written in Target Time register. This bit is reset after the generation of the Timestamp Trigger Interrupt.	0	R_WS_SC
3	TSUPDT: Timestamp Update When set, the system time is updated (added/subtracted) with the value specified in Register 452 (System Time - Seconds Update Register) and Register 453 (System Time - Nanoseconds Update Register) . This register bit should be read zero before updating it. This bit is reset once the update is completed in hardware. The “Timestamp Higher Word” register (if enabled using coreConsultant configuration) is not updated.	0	R_WS_SC
2	TSINIT: Timestamp Initialize When set, the system time is initialized (over-written) with the value specified in the Register 452 (System Time - Seconds Update Register) and Register 453 (System Time - Nanoseconds Update Register) . This register bit should be read zero before updating it. This bit is reset once the initialize is complete. The “Timestamp Higher Word” register (if enabled using coreConsultant configuration) can only be initialized.	0	R_WS_SC
1	TSCFUPDT: Timestamp Fine or Coarse Update When set, indicates that the system times update to be done using fine update method. When reset it indicates the system timestamp update to be done using Coarse method. This bit is reserved if the fine correction option is not enabled.	0	R_W
0	TSENA: Timestamp Enable When set, the time stamping is enabled for transmit and receive frames. When disabled timestamp is not added for transmit and receive frames and the Timestamp Generator is also suspended. You need to initialize the TimeStamp (system time) after enabling this mode. On the receive side, the GMAC core processes the 1588 frames only if this bit is set.	0	R_W

[Table 6-55](#) indicates the PTP messages, for which a snapshot is taken depending on the bits 17:14 (SNAPTYPSEL), in [Register 448 \(Timestamp Control Register\)](#).

Table 6-55 Time-Snapshot Dependency on Register Bits

SNAPTYPSEL (bits 17:16)	TSMSTRENA (bit 15)	TSEVNTENA (bit 14)	PTP Messages
00	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	X	X	SYNC, Delay_Req
11	X	X	Pdelay_Req, Pdelay_Resp

6.2.3.2 Register 449 (Sub-Second Increment Register)

This register is present only when the IEEE 1588 timestamp feature is selected without an external timestamp input. In Coarse Update mode (TSCFUPDT bit in Register 448), the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

Table 6-56 Register 449 (Sub-Second Increment Register)

Field	Description	Reset	Access
31:8	Reserved	0000000H	RO
7:0	SSINC: Sub-second increment value The value programmed in this register is accumulated every clock cycle (of clk_ptp_i) with the contents of the sub-second register. For example, when PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time- Nanoseconds register has an accuracy of 1 ns (TSCTRLSSR bit is set). When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465ns. In this case, you should program a value of 43 (0x2B) that is derived by 20ns/0.465.	00H	R_W

6.2.3.3 Register 450 (System Time - Seconds Register)

The System Time - Seconds register, along with System Time - Nanoseconds register, indicates the current value of the system time maintained by the core. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to clk_csr_i).

These registers (450 and 451) are present only when the IEEE 1588 Timestamp feature is selected without external time-stamp input.

Table 6-57 Register 450 (System Time - Seconds Register)

Field	Description	Reset	Access
31:0	TSS: Timestamp Second The value in this field indicates the current value in seconds of the System Time maintained by the core.	0000000H	RO

6.2.3.4 Register 451 (System Time - Nanoseconds Register)**Table 6-58 Register 451 (System Time - Nanoseconds Register)**

Field	Description	Reset	Access
31	Reserved	0	RO
30:0	TSSS: Timestamp Sub Seconds The value in this field has the sub second representation of time, with an accuracy of 0.46 nano-second. (When TSCTRLSSR is set, each bit represents 1 ns and the maximum value is 0x3B9A_C9FF, after which it rolls-over to zero).	0000000H	RO

6.2.3.5 Register 452 (System Time - Seconds Update Register)

The System Time - Seconds Update register, along with the System Time - Nanoseconds Update register, initialize or update the system time maintained by the core. You must write both of these registers before setting the TSINIT or TSUPDT bits in the Timestamp Control register. This register is present only when the IEEE 1588 Timestamp feature is selected without external time-stamp input.

Table 6-59 Register 452 (System Time - Seconds Update Register)

Field	Description	Reset	Access
31:0	TSS: Timestamp Second The value in this field indicates the time, in seconds, to be initialized or added to the system time.	0000000H	R_W

6.2.3.6 Register 453 (System Time - Nanoseconds Update Register)

This register is present only when IEEE 1588 timestamp feature is selected without external timestamp input.

Table 6-60 Register 453 (System Time - Nanoseconds Update Register)

Field	Description	Reset	Access
31	ADDSUB: Add or subtract time When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.	0	R_W
30:0	TSSS: Timestamp Sub Seconds The value in this field has the sub second representation of time, with an accuracy of 0.46 nano-second. (When TSCTRLSSR is set in the timestamp control register, each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF.)	0000000H	R_W

6.2.3.7 Register 454 (Timestamp Addend Register)

This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input. This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in Register 448). This register content is added to a 32-bit accumulator in every clock cycle (of clk_ptp_ref_i) and the system time is updated whenever the accumulator overflows.

Table 6-61 Register 454 (Timestamp Addend Register)

Field	Description	Reset	Access
31:0	TSAR: Timestamp Addend Register This register indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.	00000000H	R_W

6.2.3.8 Register 455 (Target Time Seconds Register)

The Target Time Seconds register, along with Target Time Nanoseconds register, are used to schedule an interrupt event (TSTARTGT bit in register 458 when Advanced Timestamping is enabled, or otherwise, TS interrupt bit in Register14[9]) when the system time exceeds the value programmed in these registers.

This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

Table 6-62 Register 455 (Target Time Seconds Register)

Field	Description	Reset	Access
31:0	TSTR: Target Time Seconds Register This register stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC, if enabled, generates an interrupt.	00000000H	R_W

6.2.3.9 Register 456 (Target Time Nanoseconds Register)

This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

Table 6-63 Register 456 (Target Time Nanoseconds Register)

Field	Description	Reset	Access
31	Reserved	0	RO
30:0	TSTR: Target Timstamp Low Register This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the Target Timestamp registers (both), the MAC generates an interrupt if enabled. (This value should not exceed 0x3B9A_C9FF when TSCTRLSSR is set in the Timestamp control register.)	00000000H	R_W

6.2.3.10 Register 457 (System Time - Higher Word Seconds Register)

This register is present only when the IEEE 1588 Advanced Timestamp feature is selected without an external timestamp input.

Table 6-64 Register 457 (System Time - Higher Word Seconds Register)

Field	Description	Reset	Access
31:16	Reserved	0H	RO
15:0	TSHWR: Timestamp Higher Word Register Contains the most significant 16-bits of the timestamp seconds value. This register is optional and can be selected using the coreConsultant parameter mentioned in chapter 6. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the System Time - Seconds register.	0000H	R_W_SU

6.2.3.11 Register 458 (Timestamp Status Register)

This register is present only when Advanced IEEE 1588 Timestamp feature is selected. All bits except Bits[27:25] gets cleared after this register is read by the host.

Table 6-65 Register 458 (Timestamp Status Register)

Field	Description	Reset	Access
31:28	Reserved	0	RO
27:25	ATSNS: Auxiliary Timestamp Number of Snapshots These bits indicate the number of Snapshots available in the FIFO. A value of 4 (100) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 000) when the Auxiliary snapshot FIFO clear bit is set. This bit is valid only if the Auxiliary Snapshot coreConsultant parameter is enabled.	000	RO
24	ATSSTM: Auxiliary Timestamp Snapshot Trigger Missed This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot was not stored in the FIFO. This bit is valid only if the Auxiliary Snapshot coreConsultant parameter is enabled.	0	R_SS_RC
23:3	Reserved	0	RO
2	Auxiliary Timstamp Trigger Snapshot This bit is set high when the auxiliary snapshot is written to the FIFO. This bit is valid only if the Auxiliary Snapshot coreConsultant parameter is enabled	0	R_SS_RC
1	TSTARTG: Timestamp Target Time Reached When set, indicates the value of system time is greater or equal to the value specified in the Target Time High and Low registers	0	R_SS_RC
0	TSSOVF: Timestamp Seconds Overflow When set, indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF.	0	R_SS_RC

6.2.3.12 Register 459 (PPS Control Register)

This register is present only when Advanced Timestamp feature is selected and External Time-stamping is not enabled.

Table 6-66 Register 459 (PPS Control Register)

Field	Description	Reset	Access
31:4	Reserved	0H	RO
3:0	PPSCTRL: Controls the frequency of the PPS output These bits control the behavior of the PPS output (ptp_pps_o) signal. The default value of PPSCTRL is 0000 and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:	0H	RW

PPSCTRL	Binary Rollover	Digital Rollover
0001	2 Hz	1 Hz
0010	4 Hz	2 Hz
0011	8 Hz	4 Hz
0100	16 Hz	8 Hz
...		
1111	32.768 KHz	16.384 KHz

Note:

In the binary rollover mode, the PPS output (ptp_pps_o) has a duty cycle of 50 percent with these frequencies.

In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

- When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms
- When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of:
 - One clock of 50 percent duty cycle and 537 ms period
 - Second clock of 463 ms period (268 ms low and 195 ms high)
- When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of:
 - Three clocks of 50 percent duty cycle and 268 ms period
 - Fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of the bits in the digital rollover mode in [Register 451 \(System Time - Nanoseconds Register\)](#).

6.2.3.13 Register 460 (Auxiliary Timestamp - Nanoseconds Register)

This register along with Register 461 gives the 64-bit timestamp stored as auxiliary snapshot. The two registers together form the read port of a 4-deep 64-bit wide FIFO. Multiple snapshots can be stored in this FIFO and the fill-level of this FIFO is indicated by ATSNS bits in the Timestamp Status register. The top of the FIFO is removed only when the last byte of Register 461 is read. In little-endian mode, this means when Bits[31:24] is read while in big-endian mode it corresponds to the reading of Bits[7:0] of Register 461.

Both these registers are present only when "Auxiliary Snapshot Enable" is selected during coreConsultant configuration.

Table 6-67 Register 460 (Auxiliary Timestamp - Nanoseconds Register)

Field	Description	Reset	Access
31:0	Contains the lower 32 bits (nano-seconds field) of the auxiliary timestamp	0H	RO

6.2.3.14 Register 461 (Auxiliary Timestamp - Seconds Register)**Table 6-68 Register 461 (Auxiliary Timestamp - Seconds Register)**

Field	Description	Reset	Access
31:0	Contains the lower 32 bits of the Seconds field of the auxiliary timestamp	0H	RO

6.2.3.15 Register 462 (AV MAC Control Register)

This register controls the AV traffic by identifying the AV traffic and queuing it to appropriate channel. This register is present only when you select the AV feature in coreConsultant.

Table 6-69 Register 462 (AV MAC Control Register)

Field	Description	Reset	Access
31:26	Reserved	0000H	RO
25:24	PTPCH: Channel on which PTP packets must be queued These bits specify the channel in which the untagged PTP packets (sent over the Ethernet payload and not over IPv4 or IPv6) are queued. 00 Channel 0 01 Channel 1 10 Channel 2 11 Reserved These bits are reserved if the receive paths of channel 1 or channel 2 are not enabled.	0	R_W
23	Reserved		
22:21	AVCH: Channel on which the AV control packets must be queued. These bits specify the channel in which the received untagged AV control packets are queued. 00 Channel 0 01 Channel 1 10 Channel 2 11 Reserved These bits are reserved if channel 1 or channel 2 receive paths are not enabled.	0	R_W
20	AVCD : AV Channel Disable When set, the MAC forwards all packets to the default channel 0 and the values programmed in AVP, AVCH, and PTPCH are ignored. This bit is reserved and read-only if channel 1 or channel 2 receive paths are not selected during configuration.	0	R_W
19	Reserved		

Table 6-69 Register 462 (AV MAC Control Register)

Field	Description	Reset	Access
18:16	<p>AVP: AV Priority for queuing The value programmed in these bits control the receive channel (0, 1, or 2) to which an AV packet with a given priority must be queued.</p> <ul style="list-style-type: none"> If only channel 1 receive path is enabled, then the AV packets with priority value greater than or equal to the programmed value are queued on channel 1 and all other packets are queued on channel 0. If channel 2 receive path is also enabled, then the AV packets with priority value greater than or equal to the programmed value are queued on channel 2. The AV packets with value less than the programmed value on channel 1 and all other packets are queued on channel 0. <p>These bits are applicable only if at least one additional receive channel is selected in the AV mode.</p>	0H	R_W
15:0	<p>AVT: AV etherType value These bits contain the value that is compared with the EtherType field of the incoming (tagged or untagged) ethernet frame to detect an AV packet.</p>	0	R_W

Parameters

This chapter provides a detailed description of the GMAC-UNIV configuration options and parameters. It contains the following sections:

- ❖ “Features” on page 355
- ❖ “Optional Modules” on page 366
- ❖ “Low Power Support” on page 395

7.1 Features

In coreConsultant, you can use the Specify Configuration activity to specify the configuration options. The configuration parameter options are organized into the following categories:

- ◆ “System Interface” on page 356
- ◆ “General Features” on page 360
 - ❖ “Buffer Management” on page 360
 - ❖ “MAC Address” on page 362
 - ❖ “Miscellaneous” on page 362
- ◆ “PHY Line Interfaces (RGMII, RMII, SMII, SGMII, TBI, RTBI, and RevMII)” on page 363

7.1.1 System Interface

Table 7-1 Top-Level Configuration

Option	Definition
Top-Level Configuration	<p>Description: Enables the core with FIFO, DMA, AHB, or AXI interface.</p> <p>Enable one of the following options from the drop-down menu:</p> <ul style="list-style-type: none"> • GMAC-AHB • GMAC-CORE • GMAC-MTL • GMAC-DMA • GMAC-AXI <p>Value Range: N/A</p> <p>Default Value: GMAC-AHB</p> <p>Dependencies: None</p> <p>HDL Parameter Name:</p> <ul style="list-style-type: none"> • GMAC_AHB_SUBSYS • GMAC_CORE • GMAC_MTL_SUBSYS • GMAC_DMA_SUBSYS • GMAC_AXI_SUBSYS

Table 7-2 System Data Path

Option	Definition
Datawidth	<p>Description: Configures the width of the data (32, 64, or 128 only)</p> <p>Value Range: 32, 64, or 128</p> <p>Default Value: 32</p> <p>Dependencies: None</p> <p>HDL Parameter Name: DATAWIDTH</p>
Endianness	<p>Description: Configures the GMAC-UNIV to operate in either little-endian or big-endian mode, or to have an input pin (sbd_data_endianess_i) to configure both modes.</p> <p>In coreConsultant, select one of the following options from the 'Select Little/Big Endian Mode' list:</p> <ul style="list-style-type: none"> • LITTLE_ENDIAN • BIG_ENDIAN • BOTH_OPTIONS <p>The input pin sbd_data_endianess_i is added when you select BOTH_OPTIONS.</p> <p>Value Range: N/A</p> <p>Default Value: LITTLE_ENDIAN</p> <p>Dependencies: None</p> <p>HDL Parameter Name: ENDIANNESS</p>

Table 7-2 System Data Path (Continued)

Option	Definition
Descriptor Endianness	<p>Description: Configures the format (endianness) of the DMA Descriptors with respect to the data bus endianness. The DMA Descriptor format is given in “Normal Descriptor Formats” on page 397 with different options.</p> <p>In coreConsultant, select one of the following options from the ‘Select Descriptor Endianess w.r.t Data Endianess’ list:</p> <ul style="list-style-type: none"> • SAME_ENDIANESS • REVERSE_ENDIANESS • BOTH_OPTIONS <p>Selecting SAME_ENDIANESS configures the DMA descriptor as little-endian or big-endian, as selected by the Endianness parameter above. Selecting the Reverse configures the DMA Descriptors to the reverse of the endianness of the data bus. BOTH_OPTIONS results in an input pin (sbd_desc_endianess_i) for the core in the GMAC_AHB and GMAC-DMA configurations. In GMAC-AXI configuration, sbd_data_endianess_i is also used for descriptor endianess.</p> <p>Value Range: N/A</p> <p>Default Value: BOTH_OPTIONS when Data Endianness = BOTH_OPTIONS, else SAME_ENDIANESS</p> <p>Dependencies: This is enabled only in GMAC-AHB and GMAC-DMA configurations when Data Endianness is not set to BOTH_OPTIONS.</p> <p>HDL Parameter Name: DESC_ENDIANESS</p>
Alternate Descriptor Format	<p>Description: Selects the alternate descriptor format for the DMA. This enables the DMA to transfer data to and from buffers of size up to 8 KB. This descriptor format is given in “Alternate or Enhanced Descriptors” on page 414</p> <p>Value Range: N/A</p> <p>Default Value: Disabled by default. Enabled when Advanced Time Stamp or AV feature is selected.</p> <p>Dependencies: This is enabled only in GMAC-AXI, GMAC-AHB, or GMAC-DMA configurations.</p> <p>HDL Parameter Name: DESC_ENHANCED_FORMAT</p>



The Alternate or Enhanced Descriptor format is not supported with the normal Timestamp mode. You need to either enable the Advanced Timestamp feature or de-select the Enhanced Descriptor mode.

Table 7-3 CSR Port

Option	Definition
CSR Port	<p>Description: Enables the CSR port to have the MCI, APB, AHB, or AXI Slave interface.</p> <p>Value Range: N/A</p> <p>Default Value: AHB interface in GMAC-AHB configuration, AXI interface in GMAC-AXI configuration, else MCI interface</p> <p>Dependencies:</p> <ul style="list-style-type: none"> • MCI interface cannot be selected in GMAC-AHB or GMAC-AXI configuration. • AHB interface can be selected only in GMAC-AHB or GMAC-AXI configuration. • AXI interface cannot be selected in non-GMAC-AXI configuration. <p>HDL Parameter Name: CSR_PORT</p>
Clock	<p>Description: Enables a separate clock port instead of the default application clock for the CSR interface.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled by default but enabled when CSR_PORT has AHB Slave interface in GMAC-AXI configuration</p> <p>Dependencies: Parameter selection is enabled only if the top-level configuration is anything other than GMAC_CORE or GMAC-AXI with AXI Slave interface.</p> <p>HDL Parameter Name: CSR_SLV_CLK</p>
CSR Port Datawidth	<p>Description: Configures the width of the data (32, 64, or 128 only)</p> <p>Value Range: 32, 64, or 128</p> <p>Default Value: 32</p> <p>Dependencies: Parameter selection is enabled only when the CSR port is configured as an AHB slave interface. When CSR_PORT has AXI Slave interface, the CSR_DATAWIDTH takes the value of DATAWIDTH parameter.</p> <p>HDL Parameter Name: CSR_DATAWIDTH</p>

Table 7-4 Sideband Input Signals

Option	Definition
Rx Flow Control	<p>Description: Adds the sideband sbd_flowctrl_i as input port for triggering flow control (back pressure or PAUSE frames).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled when the top-level configuration is not GMAC-CORE</p> <p>HDL Parameter Name: SBD_FC_EN</p>
MAC Transmitter /Receiver Disable	<p>Description: Adds the sideband signals sbd_dis_transmit_i and sbd_dis_receive_i input ports for Disable control of MAC transmitter and receiver.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only in GMAC-CORE and GMAC-MTL configurations</p> <p>HDL Parameter Name: ADD_TXRX_DIS_IO</p>

Table 7-4 Sideband Input Signals (Continued)

Option	Definition
Transmit FIFO Flush	<p>Description: Adds the sideband signal ati_txfifoflush_i input port for MTL Tx FIFO Flush control.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only in GMAC-MTL configurations</p> <p>HDL Parameter Name: ADD_TXFIFO_FLUSH_IO</p>

Table 7-5 AXI Interface

Option	Definition
AXI Maximum Burst Length	<p>Description: Selects the maximum burst length allowed on the AXI bus.</p> <p>Value Range: 16, 32, 64, 128, or 256</p> <p>Default Value: 16</p> <p>Dependencies: Parameter selection is only enabled for GMAC-AXI configuration.</p> <p>HDL Parameter Name: AXI_BL</p>
AXI Maximum Write Requests	<p>Description: Maximum number of outstanding Write Requests on AXI interface.</p> <p>Value Range: 4, 8, 16</p> <p>Default Value: 4</p> <p>Dependencies: Parameter selection is only enabled for GMAC-AXI configuration</p> <p>HDL Parameter Name: AXI_GM_MAX_WR_REQUESTS</p>
AXI Maximum Read Requests	<p>Description: Maximum number of outstanding Read Requests on AXI interface.</p> <p>Value Range: 4, 8, 16</p> <p>Default Value: 4</p> <p>Dependencies: Parameter selection is only enabled for GMAC-AXI configuration</p> <p>HDL Parameter Name: AXI_GM_MAX_RD_REQUESTS</p>
AXI Master Bus ID Width	<p>Description: Selects the AXI Master bus ID width.</p> <p>Value Range: 1-12</p> <p>Default Value: 4</p> <p>Dependencies: Parameter selection is only enabled for GMAC-AXI configuration.</p> <p>HDL Parameter Name: AXI_GM_AXI_ID_WIDTH</p>
AXI Slave Bus ID Width	<p>Description: Selects the AXI Slave bus ID width.</p> <p>Value Range: 2-16</p> <p>Default Value: 8</p> <p>Dependencies: Parameter selection is only enabled for GMAC-AXI configuration.</p> <p>HDL Parameter Name: GS_ID</p>

7.1.2 General Features

Table 7-6 General Features

Option	Definition
Operating Mode	<p>Description: Configures the core to work in 10/100/1000-Mbps mode. Select 10/100/1000 Mbps for enabling both Fast Ethernet as well as Gigabit operations, 10/100 Mbps for Fast Ethernet-only operations, and 1000 Mbps for Gigabit-only operations.</p> <p>Value Range: N/A</p> <p>Default Value: 10/100/1000 Mbps with Gigabit License, 10/100 with Fast Ethernet license</p> <p>Dependencies: This is not configurable with Fast Ethernet License</p> <p>HDL Parameter Name: OP_MODE</p>
User Defined Version	<p>Description: Configurable 8-bit value that is hard-coded into bits[15:8] of the GMAC Version Register (for example, 8'h10 for version 1.0).</p> <p>Value Range: 0x10 to 0xFF</p> <p>Default Value: 0x10</p> <p>Dependencies: None</p> <p>HDL Parameter Name: USER_VER</p>
Half-Duplex/Full-Duplex	<p>Description: Configures the core to work only in full-duplex mode.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: FDUPLEX_ONLY</p>
Reset Mode	<p>Description: Enables the asynchronous reset for all flip-flops. When deselected, all register flip-flops have synchronous resets.</p> <p>Value Range: N/A</p> <p>Default Value: Enabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: ASYNC_RST</p>

7.1.2.1 Buffer Management

Table 7-7 Buffer Management

Option	Definition
Rx FIFO Size	<p>Description: Selects the Rx FIFO size.</p> <p>Value Range: 128 bytes to 32 KB in powers of 2</p> <p>Default Value: 2 KB</p> <p>Dependencies: Parameter selection is enabled for all GMAC configurations except GMAC-CORE.</p> <p>HDL Parameter Name: RXFIFO_SIZE</p>

Table 7-7 Buffer Management (Continued)

Option	Definition
Tx FIFO Size	<p>Description: Selects the Tx FIFO size.</p> <p>Value Range: 256 bytes to 16 KB in powers of 2</p> <p>Default Value: 2 KB</p> <p>Dependencies: The parameter does not apply if the top-level configuration is GMAC-CORE. 256-byte and 512-byte values can only be selected in 10/100-only mode, or if the core is configured for only full-duplex mode.</p> <p>HDL Parameter Name: TXFIFO_SIZE</p>
Maximum Number of Frames in the Tx FIFO	<p>Description: Selects the maximum number of frames that can be pushed into the Tx FIFO at a time without any transmit status being read out on the ATI interface.</p> <p>Value Range: 2, 4, or 8</p> <p>Default Value: 2</p> <p>Dependencies: Parameter selection is only enabled in GMAC-MTL configuration.</p> <p>HDL Parameter Name: MAX_FRAME_IN_TXFIFO</p>
DPRAM TYPE	<p>Description: Selects the use of synchronous or asynchronous DPRAM type.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only if top-level configuration is anything other than GMAC-CORE.</p> <p>HDL Parameter Name: ASYNC_RAM</p>
Min Size of Rx Frames	<p>Description: Configures the minimum size of frame that is being stored in the MTL Rx FIFO. This configures the width of the counter used for storing the number of frames present in Rx FIFO.</p> <p>Value Range: 16, 32, 64</p> <p>Default Value: 16</p> <p>Dependencies: Parameter selection is enabled only when the top-level configuration is GMAC-MTL and the RX_FRAME_LEN_IF is enabled.</p> <p>HDL Parameter Name: MIN_FRAME_SIZE</p>
Rx Frame Length FIFO	<p>Description: Enables the Frame Length FIFO-related logic in the MTL Receive path.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when the top-level configuration is GMAC-MTL.</p> <p>Parameter Name: RX_FRAME_LEN_IF</p>
Rx Frame Length FIFO Width	<p>Description: Configures the width of MTL Rx Frame Length FIFO.</p> <p>Value Range: 12 to 15</p> <p>Default Value: 15</p> <p>Dependencies: Parameter selection is enabled only when the top-level configuration is GMAC-MTL and the RX_FRAME_LEN_IF is enabled.</p> <p>HDL Parameter Name: FRAME_LEN_FIFO_WIDTH</p>

7.1.2.2 MAC Address

Table 7-8 MAC Address Registers

Option	Definition
Enable all MAC Address Registers	<p>Description: Enables all MAC Address registers (1 to 31) for the Address Filter block.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: ALL_MAC_ADDR_EN</p>
Enable MAC Address Register 1	<p>Description: Enable MAC Address register 1 for the Address Filter block.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only if All MAC Addresses is disabled.</p> <p>HDL Parameter Name: MAC_ADDR1</p> <p>Note: MAC Address registers 2 through 31 can be enabled in the same manner.</p>
Enable MAC Address Register N	<p>Description: Enable MAC Address registers 2–30 for the Address Filter block.</p>
Enable MAC Address Register 31	<p>Description: Enables MAC Address register 31 for the Address Filter block.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only if All MAC Addresses is disabled.</p> <p>HDL Parameter Name: MAC_ADDR31</p>

7.1.2.3 Miscellaneous

Table 7-9 Destination Address Filter

Option	Definition
Disable Address Filter Hash Table	<p>Description: Removes the Hash Table registers and related address filter logic.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: HASH_DIS</p>

Table 7-10 Synchronization

Option	Definition
Sync CSR MAC Address to Tx/Rx clock domain	<p>Description: Enable the synchronization of MAC address registers to the transmit/receive clock domain. Otherwise, the CSR MAC Address Register value written with the application clock is directly used in the transmit/receive clock domains.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: SYNC_MAC_ADDR</p>
Sync Pause Time to Tx clock domain	<p>Description: Enable the synchronization of the Pause Timer register to the transmit clock domain.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: SYNC_PAUSETIME</p>
Sync VLAN Tag to Rx clock domain	<p>Description: Enables the synchronization of the VLAN Tag register to the receive clock domain.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: SYNC_VLANTAG</p>
Sync Hash table to Rx clock domain	<p>Description: Enable the synchronization of the Hash Table register to the receive clock domain.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: SYNC_HASHTABLE</p>

7.1.3 PHY Line Interfaces (RGMII, RMII, SMII, SGMII, TBI, RTBI, and RevMII)

The GMII/MII interface is the default selection, and is always available. The interfaces you select from [Table 7-11](#) are added to the core as depicted in [Figure 4-40](#).

Table 7-11 Line Interface

Option	Definition
Reduced GMI Interface (RGMII)	<p>Description: Enables the Reduced GMI Interface (RGMII).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Not enabled in 10/100 Mbps-only mode</p> <p>HDL Parameter Name: RGMII_EN</p>

Table 7-11 Line Interface (Continued)

Option	Definition
Reduced MI Interface (RMII)	<p>Description: Enables the Reduced MI Interface (RMII).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: This interface is available only for the 10/100-Mbps mode and is not available in 1000 Mbps-only mode</p> <p>HDL Parameter Name: RMII_EN</p>
Serial MI Interface (SMII)	<p>Description: Enables the Serial MI Interface (SMII).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: This interface is only applicable for the 10/100-Mbps and 10/100/1000-Mbps modes, and is not available in 1000 Mbps-only mode.</p> <p>HDL Parameter Name: SMII_EN</p>
Serial GMI Interface (SGMII)	<p>Description: Enables the Serial GMI Interface (SGMII).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Not enabled in 10/100 Mbps-only mode</p> <p>HDL Parameter Name: SGMII_EN</p>
Ten Bit Interface (TBI)	<p>Description: Enables the Ten Bit Interface (TBI).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Not enabled in 10/100 Mbps-only mode</p> <p>HDL Parameter Name: TBI_EN</p>
Reduced Ten Bit Interface (RTBI)	<p>Description: Enables the Reduced Ten Bit Interface (RTBI).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Not enabled in 10/100 Mbps-only mode</p> <p>HDL Parameter Name: RTBI_EN</p>
Reverse MII Interface (RevMII)	<p>Description: Enables the Reverse MII Interface (RevMII).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: REVMII_EN</p>

Table 7-12 Miscellaneous Features for RGMII/RTBI Line Interface

Option	Definition
Dual edge flip-flop	<p>Description: Selects the dual edge flip-flop from Xilinx Virtex 2V FPGA Library for the RGMII or RTBI only (for synthesis).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when you select the single PHY interface and RGMII or RTBI interface.</p> <p>HDL Parameter Name: XLNX_2V</p>

Table 7-13 Miscellaneous Features for RGMII/RevMII/SGMII/RMII/SMII Line Interface

Option	Definition
Add output port to indicate speed (10, 100, or 1000 Mbps)	<p>Description: Additional output port for speed selection (10, 100, or 1000 Mbps) for RGMII, SGMII, SMII, RevMII, or RMII.</p> <p>Value Range: N/A</p> <p>Default Value: Enabled except when the RMII interface or RevMII interface is selected.</p> <p>Dependencies: When RevMII is selected, this parameter is available by default. Parameter selection is enabled only when RGMII, SGMII, SMII, or RMII interface is selected.</p> <p>HDL Parameter Name: SPEED_SELECT</p>
Source Synchronous SMII (SSSMII)	<p>Description: Enables the Source Synchronous Serial MI Interface (SSSMII).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: This option is applicable only when SMII_EN is enabled.</p> <p>HDL Parameter Name: SSSMII_EN</p>
TXSYNC as Input in SSSMII	<p>Description: Converts the TXSYNC as input signal in SSSMII.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: This option is applicable only when SSSMII_EN is enabled.</p> <p>HDL Parameter Name: SSSMII_TXSYNC_IN</p>

Table 7-14 Option for Single PHY Interface

Option	Definition
Enable single PHY interface	<p>Description: A single PHY interface is selected without any MUX logic at the input or output. When this option is enabled, the phy_intf_sel_i input port is removed.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when one of the PHY interfaces or one of the GMII/MII interfaces is selected.</p> <p>HDL Parameter Name: SINGLE_PHY_INTF</p>

7.2 Optional Modules

This section describes the parameters for the following optional modules of GMAC-UNIV:

- ◆ [Energy Efficient Ethernet \(EEE\)](#)
- ◆ [AV Feature Support](#)
- ◆ [IEEE 1588 Timestamp Block](#)
- ◆ [Power Management Block](#)
- ◆ [IP Checksum Block](#)
- ◆ [GMAC Management \(RMON\) Counters](#)
 - ✧ [GMAC Management \(RMON\) Receive Counters](#)
 - ✧ [GMAC Management \(RMON\) Transmit Counters](#)
 - ✧ [GMAC Management \(RMON\) Receive IPC Counters](#)
- ◆ [Station Management Block](#)

7.2.1 Energy Efficient Ethernet (EEE)

Table 7-15 Energy Efficient Ethernet (EEE) Parameters

Option	Definition
Energy Efficient Ethernet (EEE) Enable	<p>Description: Adds logic for supporting the Energy Efficient Ethernet (EEE) functionality.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Not available for selection when TBI, RTBI, SMII, RMII, or SGMII single PHY interface is selected.</p> <p>HDL Parameter Name: EEE_EN</p>

7.2.2 AV Feature Support

Table 7-16 AV Feature Support Parameters

Option	Definition
Select AV Feature	<p>Description: Adds logic for supporting the AV functionality and provides the option to select additional channels for queueing the time-sensitive traffic.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: This feature is not supported in the GMAC-MTL configuration.</p> <p>HDL Parameter Name: AV_ENABLE</p> <p>Note: When ‘Select AV Feature’ is enabled, the Advanced Time Stamping, Enable Full Receive Checksum Offload, and Alternate (Enhanced) Descriptor features are enabled by default.</p>

Table 7-16 AV Feature Support Parameters

Option	Definition
Select AV Transmit Channel(s)	<p>Description: Selects the number of additional Transmit channels.</p> <p>Value Range: 0, 1, 2</p> <p>Default Value: 1: When AV support is enabled. 0: When AV support is not enabled</p> <p>Dependencies: Enabled only when the AV Support is enabled.</p> <p>HDL Parameter Name: AV_TX_CHANNEL</p>
Select AV Receive Channel(s)	<p>Description: Selects the number of additional Receive channels.</p> <p>Value Range: 0, 1, 2</p> <p>Default Value: 0</p> <p>Dependencies: Enabled only when the AV Support is enabled.</p> <p>HDL Parameter Name: AV_RX_CHANNEL</p>

7.2.3 IEEE 1588 Timestamp Block

Table 7-17 IEEE 1588 Timestamp Block Parameters

Option	Definition
IEEE1588 Time Stamping	<p>Description: Adds logic for supporting IEEE1588 time stamping protocol. Time stamp is captured for all the receive frames. Timestamp is also captured for transmit frames marked as PTP frames.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled except when AV feature is selected</p> <p>Dependencies: Not enabled for selection when the AV feature is selected</p> <p>HDL Parameter Name: TIME_STAMPING</p>
IEEE1588 External Timestamp Input Enable	<p>Description: This removes the System Time generation logic in the core and uses a 64-bit reference time input to take the timestamps that are provided with the status.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Enabled when IEEE1588 Time Stamping is enabled.</p> <p>HDL Parameter Name: EXT_TIME_STAMPING</p>
IEEE 1588 Advanced Timestamp Support	<p>Description: This enables the advanced timestamp features like snooping PTP packets along with support for IEEE 1588-2008 specifications.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled except when the AV feature is selected</p> <p>Dependencies: Enabled for selection when IEEE1588 Time Stamping is enabled and AV feature is not selected.</p> <p>HDL Parameter Name: ADV_TIME_STAMPING</p> <p>Note: When Advanced Timestamp Support is selected, Enable Full Receive Checksum Offload and Alternate (Enhanced) Descriptor are enabled, by default.</p>

Table 7-17 IEEE 1588 Timestamp Block Parameters (Continued)

Option	Definition
IEEE1588 Higher Word Register Enable	<p>Description: This adds a register to store the most significant 16 bits [47:32] of the seconds value if IEEE 1588 version 2 format of timestamp is used.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Enabled when IEEE1588 Advanced Time Stamping is enabled.</p> <p>HDL Parameter Name: ADV_TIME_HIGH_WORD</p>
IEEE 1588 Auxiliary Snapshot Enable	<p>Description: This adds the capability of storing the timestamp snapshots taken with an external trigger, into a 4-deep FIFO.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Enabled when IEEE1588 Advanced Time Stamping is enabled.</p> <p>HDL Parameter Name: ADV_TIME_AUX_SNAP</p>

7.2.4 Power Management Block

Table 7-18 Power Management Block

Option	Definition
Enable Power Management block (PMT)	<p>Description: Enables the Power Management block in the core.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: PMT_EN</p>
Enable detection of remote wake-up frame	<p>Description: Enables the detection of a remote wake-up frame.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only if PMT is enabled.</p> <p>HDL Parameter Name: PMT_RWK_EN</p>
Enable detection of magic packet frame	<p>Description: Enables the detection of a magic packet frame.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only if PMT is enabled.</p> <p>HDL Parameter Name: PMT_MGK_EN</p>

7.2.5 IP Checksum Block

Table 7-19 MDC Features Configuration Options

Option	Definition
Enable IP Checksum for received frames	<p>Description: Enables the appending of the IP checksum (Type 1) for only received frames.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: IPCHKSUM_EN</p>
Enable Full Checksum Offload	<p>Description: Enables the enhanced Checksum Offload engine (Type 2) in the receive path. In this mode, the Checksum Offload engine checks and detects the IPv4 header checksum errors and TCP, UDP, or ICMP checksum errors encapsulated in IPv4 or IPv6 datagrams payload of the received frames.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Enabled only when IP Checksum (IPCHKSUM_EN) is selected.</p> <p>HDL Parameter Name: IPC_FULL_OFFLOAD</p>
Enable Transmit COE	<p>Description: Enabled the Checksum Offload Engine in the transmit path. In this mode, the core can calculate and insert the checksums of TCP, UDP, or ICMP segments encapsulated in IPv4 or IPv6 datagrams. The core can insert IPv4 header checksums as well in the transmitted frames.</p> <p>Value range: N/A</p> <p>Default value: Disabled</p> <p>Dependencies: Applicable only for non GMAC-CORE configurations.</p> <p>HDL parameter Name: TX_COE</p>

7.2.6 GMAC Management (RMON) Counters

Table 7-20 RMON Counters

Option	Definition
Enable GMAC Management (RMON) Counters	<p>Description: Enables the GMAC Management (RMON) counters.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: MMC_EN</p>
Enable GMAC Management (RMON) IPC Counters	<p>Description: Enables the GMAC Management (RMON) counters for the Checksum Offload module.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Enabled only when the RMON counters and Full Checksum Offload engine (Type 2) are selected.</p> <p>HDL Parameter Name: MMC_IPC_EN</p>

7.2.6.1 GMAC Management (RMON) Receive Counters

Table 7-21 RMON Receive Counters

Option	Definition
Enable all Rx frame counters	<p>Description: Enables all of the Rx frame counters.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only if GMAC Management (RMON) Counters are enabled.</p> <p>HDL Parameter Name: MMC_EN_RX_ALL</p>
Rx frame counter for good and bad frames	<p>Description: Enables Rx frame counter for received good or bad frames.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXFRMGB_CNT_EN</p>
Enable 16-bit Rx frame counter for good and bad frames	<p>Description: Changes the Rx frame counter width, for received good or bad frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXFRMGB_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXFRMGB_CNT_16BIT_EN</p>
Rx counter for octets in good frames	<p>Description: Enables the Rx counter for octets in received good frames.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXOCTG_CNT_EN</p>
Enable 16-bit Rx counter for octets in good frames	<p>Description: Changes the width of the Rx counter, for octets in received good frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXOCTG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXOCTG_CNT_16BIT_EN</p>
Rx counter for octets in good and bad frames	<p>Description: Enables the Rx counter for octets in good and bad frames received.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXOCTGB_CNT_EN</p>

Table 7-21 RMON Receive Counters (Continued)

Option	Definition
Enable 16-bit Rx counter for octets in good and bad frames	<p>Description: Changes the width of the Rx counter, for octets in received good and bad frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXOCTGB_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXOCTGB_CNT_16BIT_EN</p>
Rx frame counter for good broadcast frames	<p>Description: Enables the Rx frame counter for good broadcast frames received.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXBCASTG_CNT_EN</p>
Enable 16-bit Rx counter for good broadcast frames	<p>Description: Changes the width of the Rx counter, for octets in received good broadcast frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXBCASTG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXBCASTG_CNT_16BIT_EN</p>
Rx frame counter for good multicast frames	<p>Description: Enables the Rx frame counter for good multicast frames received.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXMCASTG_CNT_EN</p>
Enable 16-bit Rx counter for good multicast frames	<p>Description: Changes the width of the Rx counter, for octets in received good multicast frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXMCASTG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXMCASTG_CNT_16BIT_EN</p>
Rx frame counter for frames with CRC errors	<p>Description: Enables the Rx frame counter for frames received with CRC errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXCRCERR_CNT_EN</p>

Table 7-21 RMON Receive Counters (Continued)

Option	Definition
Enable 16-bit Rx counter for frames with CRC errors	<p>Description: Changes the width of the Rx counter, for frames received with CRC errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXCRCERR_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXCRCERR_CNT_16BIT_EN</p>
Rx frame counter for frames with Alignment errors	<p>Description: Enables the Rx frame counter for frames received with Alignment errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled. It is not enabled for 1000 Mbps-only configuration</p> <p>HDL Parameter Name: MMC_RXALGNERR_CNT_EN</p>
Enable 16-bit Rx counter for frames with Alignment errors	<p>Description: Changes the width of the Rx counter, for frames received with Alignment errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXALGNERR_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXALGNERR_CNT_16BIT_EN</p>
Rx frame counter for frames with Runt errors	<p>Description: Enables the Rx frame counter for frames received with runt errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXRUNTERR_CNT_EN</p>
Enable 16-bit Rx counter for frames with Runt errors	<p>Description: Changes the width of the Rx counter, for frames received with Runt errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXRUNTERR_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXRUNTERR_CNT_16BIT_EN</p>
Rx frame counter for frames with Jabber error	<p>Description: Enables the Rx frame counter for frames received with Jabber error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXJABERR_CNT_EN</p>

Table 7-21 RMON Receive Counters (Continued)

Option	Definition
Enable 16-bit Rx counter for frames with Jabber errors	<p>Description: Changes the width of the Rx counter, for frames received with Jabber errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXJABERR_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXJABERR_CNT_16BIT_EN</p>
Rx frame counter for undersized good frames	<p>Description: Enables the Rx frame counter for undersized good frames received.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXUNDERSZG_CNT_EN</p>
Enable 16-bit Rx counter for undersized good frames	<p>Description: Changes the width of the Rx counter, for received undersized good frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXUNDERSZG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXUNDERSZG_CNT_16BIT_EN</p>
Rx frame counter for giant frames	<p>Description: Enables the Rx frame counter for giant frames</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXOVERSZG_CNT_EN</p>
Enable 16-bit Rx counter for giant frames	<p>Description: Changes the width of the Rx counter for giant frames to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXOVERSZG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXOVERSZG_CNT_16BIT_EN</p>
Rx Octet counters for various frame sizes	<p>Description: Enables the octet counters for received frames. Separate octet counters for (good and bad) received frames sized equal to 64, 65–127, 128–255, 256–511, 512–1,023 and 1024_to_max bytes are provided.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXADDR_RNG_CNT_EN</p>

Table 7-21 RMON Receive Counters (Continued)

Option	Definition
Enable 16-bit Rx Octet counter for various frame sizes	<p>Description: Changes the width of the Rx counter for received frames to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXADDR_RNG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXADDR_RNG_CNT_16BIT_EN</p>
Rx frame counter for good unicast frames	<p>Description: Enables the Rx frame counter for good unicast frames received.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXUCASTG_CNT_EN</p>
Enable 16-bit Rx frame counter for good unicast frames	<p>Description: Changes the width of the Rx counter, for received good unicast frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXUCASTG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXUCASTG_CNT_16BIT_EN</p>
Rx frame counter for frames with Length error	<p>Description: Enables the Rx frame counter for frames received with length errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXLENERR_CNT_EN</p>
Enable 16-bit Rx frame counter for frames with Length error	<p>Description: Changes the width of the Rx counter, for frames received with length errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXLENERR_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXLENERR_CNT_16BIT_EN</p>
Rx frame counter for out-of-range length frames	<p>Description: Enables the Rx frame counter for out-of-range length frames (with length field values greater than 1500 and less than 1536) received.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXOUTOFRNG_TYP_CNT_EN</p>

Table 7-21 RMON Receive Counters (Continued)

Option	Definition
Enable 16-bit Rx frame counter for out-of-range length frames	<p>Description: Changes the width of the Rx counter, for received out-of-range length frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXOUTOFRNG_TYP_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXOUTOFRNG_TYP_CNT_16BIT_EN</p>
Rx frame counter for pause frames	<p>Description: Enables the Rx frame counter for pause frames received.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXPAUSE_CNT_EN</p>
Enable 16-bit Rx frame counter for pause frames	<p>Description: Changes the width of the Rx counter, for received pause frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXPAUSE_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXPAUSE_CNT_16BIT_EN</p>
Rx frame counter for number of FIFO overflow conditions	<p>Description: Enables the Rx frame counter for the number of FIFO overflow conditions.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled and when the top-level configuration is anything other than GMAC-CORE.</p> <p>HDL Parameter Name: MMC_RXFIFOVFL_CNT_EN</p>
Enable 16-bit Rx frame counter for number of FIFO overflow conditions	<p>Description: Changes the width of the Rx counter, for the number of FIFO overflow conditions, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXFIFOVFL_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXFIFOVFL_CNT_16BIT_EN</p>
Rx frame counter for VLAN good and bad frames	<p>Description: Enables the Rx frame counter for VLAN good and bad frames received.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXVLANGB_CNT_EN</p>

Table 7-21 RMON Receive Counters (Continued)

Option	Definition
Enable 16-bit Rx frame counter for VLAN good and bad frames	<p>Description: Changes the width of the Rx counter, for VLAN good and bad frames received, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXVLANGB_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXVLANGB_CNT_16BIT_EN</p>
Rx frame counter for frames with watchdog error	<p>Description: Enables the Rx frame counter for frames received with watchdog errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and Enable all Rx frame counters is disabled.</p> <p>HDL Parameter Name: MMC_RXWDGERR_CNT_EN</p>
Enable 16-bit Rx frame counter for frames with watchdog error	<p>Description: Changes the width of the Rx counter, for frames received with watchdog errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RXWDGERR_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RXWDGERR_CNT_16BIT_EN</p>

7.2.6.2 GMAC Management (RMON) Transmit Counters

Table 7-22 RMON Transmit Counters

Option	Definition
Enable all Tx frame counters	<p>Description: Enables all Tx frame counters.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only if GMAC Management (RMON) counters are enabled.</p> <p>HDL Parameter Name: MMC_EN_TX_ALL</p>
Tx octet counters for various frame sizes	<p>Description: Enables the octet counters for transmitted frames. Separate octet counters for (good and bad) frames sized equal to 64, 65-127, 128-255, 256-511, 512-1,023 and 1024_to_max bytes are provided.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXADDR_RNG_CNT_EN</p>

Table 7-22 RMON Transmit Counters (Continued)

Option	Definition
Enable 16-bit Tx octet counter for various frame sizes	<p>Description: Changes the width of the Tx octet counter for transmitted frames to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXADDR_RNG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXADDR_RNG_CNT_16BIT_EN</p>
Tx octet counter for good and bad frames	<p>Description: Enables the Tx octet counter for good and bad frames transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXOCTGB_CNT_EN</p>
Enable 16-bit Tx octet counter for good and bad frames	<p>Description: Changes the width of the Tx octet counter, for transmitted good and bad frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXOCTGB_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXOCTGB_CNT_16BIT_EN</p>
Tx frame counter for good and bad frames	<p>Description: Enables the Tx frame counter for good and bad frames transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXFRMGB_CNT_EN</p>
Enable 16-bit Tx frame counter for good and bad frames	<p>Description: Changes the width of the Tx frame counter, for transmitted good and bad frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXFRMGB_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXFRMGB_CNT_16BIT_EN</p>
Tx frame counter for good broadcast frames	<p>Description: Enables the Tx frame counter for good broadcast frames.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXBCASTG_CNT_EN</p>

Table 7-22 RMON Transmit Counters (Continued)

Option	Definition
Enable 16-bit Tx frame counter for good broadcast frames	<p>Description: Changes the width of the Tx frame counter, for transmitted good broadcast frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXBCASTG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXBCASTG_CNT_16BIT_EN</p>
Tx frame counter for good multicast frames	<p>Description: Enables the Tx frame counter for good multicast frames transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXMCASTG_CNT_EN</p>
Enable 16-bit Tx frame counter for good multicast frames	<p>Description: Changes the width of the Tx frame counter, for transmitted good multicast frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXMCASTG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXMCASTG_CNT_16BIT_EN</p>
Tx frame counter for good and bad unicast frames	<p>Description: Enables the Tx frame counter for good and bad unicast frames.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXUCASTGB_CNT_EN</p>
Enable 16-bit Tx frame counter for good and bad unicast frames	<p>Description: Changes the width of the Tx frame counter, for transmitted good and bad unicast frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXUCASTGB_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXUCASTGB_CNT_16BIT_EN</p>
Tx frame counter for good and bad multicast frames	<p>Description: Enables the Tx frame counter for good and bad multicast frames transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXMCASTGB_CNT_EN</p>

Table 7-22 RMON Transmit Counters (Continued)

Option	Definition
Enable 16-bit Tx frame counter for good and bad multicast frames	<p>Description: Changes the width of the Tx frame counter, for transmitted good and bad multicast frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXMCICASTGB_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXMCICASTGB_CNT_16BIT_EN</p>
Tx frame counter for good and bad broadcast frames	<p>Description: Enables the Tx frame counter for good and bad broadcast frames transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXBCASTGB_CNT_EN</p>
Enable 16-bit Tx frame counter for good and bad broadcast frames	<p>Description: Changes the width of the Tx frame counter, for transmitted good and bad broadcast frames, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXBCASTGB_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXBCASTGB_CNT_16BIT_EN</p>
Tx frame counter for frames with underflow errors	<p>Description: Enables the Tx frame counter for frames transmitted with underflow errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXUNRFLW_CNT_EN</p>
Enable 16-bit Tx frame counter for frames with underflow errors	<p>Description: Changes the width of the Tx frame counter, for frames transmitted with underflow errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXUNRFLW_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXUNRFLW_CNT_16BIT_EN</p>
Tx frame counter for good frames with single collision	<p>Description: Enables the Tx frame counter for good frames transmitted with single collisions.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXSNGLCOLG_CNT_EN</p>

Table 7-22 RMON Transmit Counters (Continued)

Option	Definition
Enable 16-bit Tx frame counter for good frames with single collision	<p>Description: Changes the width of the Tx frame counter, for frames transmitted with single collisions, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXSNGLCOLG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXSNGLCOLG_CNT_16BIT_EN</p>
Tx frame counter for good frames with multiple collision	<p>Description: Enables the Tx frame counter for good frames transmitted with multiple collisions.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXMULTCOLG_CNT_EN</p>
Enable 16-bit Tx frame counter for good frames with multiple collision	<p>Description: Changes the width of the Tx frame counter, for frames transmitted with multiple collisions, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXMULTCOLG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXMULTCOLG_CNT_16BIT_EN</p>
Tx frame counter for frames deferred before transmit	<p>Description: Enables the Tx frame counter for frames that were deferred before being transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXDEFRD_CNT_EN</p>
Enable 16-bit Tx frame counter for frames deferred before transmit	<p>Description: Changes the width of the Tx frame counter, for frames that were deferred before being transmitted, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXDEFRD_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXDEFRD_CNT_16BIT_EN</p>
Tx frame counter for frames with late collision	<p>Description: Enables the Tx frame counter for frames aborted because of late collisions.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXLATECOL_CNT_EN</p>

Table 7-22 RMON Transmit Counters (Continued)

Option	Definition
Enable 16-bit Tx frame counter for frames with late collision	<p>Description: Changes the width of the Tx frame counter, for frames aborted because of late collisions, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXLATECOL_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXLATECOL_CNT_16BIT_EN</p>
Tx frame counter for frames aborted after excessive collision	<p>Description: Enables the Tx frame counter for frames aborted after excessive collisions.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXEXSCOL_CNT_EN</p>
Enable 16-bit Tx frame counter for frames aborted after excessive collision	<p>Description: Changes the width of the Tx frame counter, for frames aborted after excessive collisions, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXEXSCOL_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXEXSCOL_CNT_16BIT_EN</p>
Tx frame counter for frames with carrier error	<p>Description: Enables the Tx frame counter for frames aborted with carrier errors (loss of carrier or no carrier).</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled, half-duplex mode is enabled, and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXCARR_CNT_EN</p>
Enable 16-bit Tx frame counter for frames with carrier error	<p>Description: Changes the width of the Tx frame counter, for frames aborted with carrier errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXCARR_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXCARR_CNT_16BIT_EN</p>
Tx octet counter for good frames	<p>Description: Enables the Tx octet counter for good frames.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXOCTG_CNT_EN</p>

Table 7-22 RMON Transmit Counters (Continued)

Option	Definition
Enable 16-bit Tx octet counter for good frames	<p>Description: Changes the width of the Tx octet counter for good frames to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXOCTG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXOCTG_CNT_16BIT_EN</p>
Tx frame counter for good frames	<p>Description: Enables the Tx frame counter for good frames transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXFRMG_CNT_EN</p>
Enable 16-bit Tx frame counter for good frames	<p>Description: Changes the width of the Tx frame counter for good frames to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXFRMG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXFRMG_CNT_16BIT_EN</p>
Tx frame counter for frames aborted because of excessive deferral	<p>Description: Enables the Tx frame counter for frames that were aborted because of excessive deferral.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXEXSDEF_CNT_EN</p>
Enable 16-bit Tx frame counter for frames aborted because of excessive deferral	<p>Description: Changes the width of the Tx frame counter, for frames that were aborted because of excessive deferral, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXFRMG_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXEXSDEF_CNT_16BIT_EN</p>
Tx frame counter for pause frames	<p>Description: Enables the Tx frame counter for pause frames transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXPAUSE_CNT_EN</p>

Table 7-22 RMON Transmit Counters (Continued)

Option	Definition
Enable 16-bit Tx frame counter for pause frames	<p>Description: Changes the width of the Tx frame counter for pause frames to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXPAUSE_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXPAUSE_CNT_16BIT_EN</p>
Tx frame counter for VLAN frames	<p>Description: Enables the Tx frame counter for VLAN frames transmitted.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters are enabled and “Enable all Tx frame counters” is disabled.</p> <p>HDL Parameter Name: MMC_TXVLAN_CNT_EN</p>
Enable 16-bit Tx frame counter for VLAN frames	<p>Description: Changes the width of the Tx frame counter for VLAN frames to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_TXVLAN_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_TXVLAN_CNT_16BIT_EN</p>

7.2.6.3 GMAC Management (RMON) Receive IPC Counters

Table 7-23 GMAC Management (RMON) Receive IPC Counters

Option	Definition
Enable all Rx IPC Counters	<p>Description: Enables all Rx IPC counters.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only if GMAC Management (RMON) counters and RMON IPC counters are enabled and Full Checksum Offload Engine (Type 2) is selected</p> <p>HDL Parameter Name: MMC_IPC_EN_RX_ALL</p>
Enable counter for Rx IPv4 Good frames	<p>Description: Enables Rx frame counter for good IPv4 datagrams received without any header or checksum errors and having TCP, UDP, or ICMP payload.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC counters, and Full Checksum Offload Engine (Type 2) are enabled and Enable all Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_GD_FRMS_CNT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable 16-bit counter for Rx IPv4 Good frame	<p>Description: Changes the width of the Rx frame counter, for good IPv4 datagrams received without any header or checksum errors and having TCP, UDP, or ICMP payload, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_GD_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_GD_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx IPv4 datagrams with header error	<p>Description: Enables Rx frame counter for all IPv4 datagrams received with header errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, and Full Checksum Offload Engine (Type 2) are enabled and Enable all Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_HDRERR_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx IPv4 datagrams with header error	<p>Description: Changes the width of the Rx frame counter, for all IPv4 datagrams received with header errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_HDRERR_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_HDRERR_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx IPv4 datagrams not having TCP/UDP/ ICMP	<p>Description: Enables Rx frame counter for all IPv4 datagrams received with payload other than TCP, UDP, or ICMP.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_NOPAY_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx IPv4 datagrams not having TCP/UDP/ ICMP	<p>Description: Changes the width of the Rx frame counter, for all IPv4 datagrams received with payload other than TCP, UDP, or ICMP, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_NOPAY_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_NOPAY_FRMS_CNT_16BIT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable counter for Rx IPv4 datagrams having fragmentation	<p>Description: Enables Rx frame counter for all IPv4 datagrams received with fragments.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, and Full Checksum Offload Engine (Type 2) are enabled and Enable all Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_FRAG_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx IPv4 datagrams having fragmentation	<p>Description: Changes the width of the Rx frame counter, for all IPv4 datagrams received with fragments, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_FRAG_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_FRAG_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx UDP over IPv4 datagrams having UDP checksum disabled	<p>Description: Enables Rx frame counter for all IPv4 datagrams received that have checksum-disabled UDP segments.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, and Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_UDSBL_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx UDP over IPv4 datagrams having UDP checksum disabled	<p>Description: Changes the width of the Rx frame counter, for all IPv4 datagrams received that have checksum-disabled UDP segments, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_UDSBL_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_UDSBL_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx IPv6 Good frame	<p>Description: Enables Rx frame counter for good IPv6 datagrams that encapsulate TCP, UDP, or ICMP data and are received without header or checksum errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, and Full Checksum Offload Engine (Type 2) are enabled and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_GD_FRMS_CNT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable 16-bit counter for Rx IPv6 Good frame	<p>Description: Changes the width of the Rx frame counter, for good IPv6 datagrams that encapsulate TCP, UDP, or ICMP data and are received without header or checksum errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV6_GD_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_GD_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx IPv6 datagrams with header error	<p>Description: Enables an Rx frame counter for all IPv6 datagrams received with header errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, and Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_HDRERR_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx IPv6 datagrams with header error	<p>Description: Changes the width of the Rx frame counter, for all IPv6 datagrams received with header errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV6_HDRERR_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_HDRERR_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx IPv6 datagrams not having TCP/UDP/ ICMP	<p>Description: Enables an Rx frame counter for all IPv6 datagrams received with payload other than TCP, UDP, or ICMP.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_NOPAY_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx IPv6 datagrams not having TCP/UDP/ ICMP	<p>Description: Changes the width of the Rx frame counter, for all IPv6 datagrams received with payload other than TCP, UDP, or ICMP, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV6_NOPAY_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_NOPAY_FRMS_CNT_16BIT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable counter for Rx UDP segments with good checksum	<p>Description: Enables an Rx frame counter for all UDP segments received with no checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_UDP_GD_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx UDP segments with good checksum	<p>Description: Changes the width of the Rx frame counter, for UDP segments received with no checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_UDP_GD_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_UDP_GD_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx UDP segments with checksum error	<p>Description: Enables an Rx frame counter for UDP segments received with a checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_UDP_ERR_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx UDP segments with checksum error	<p>Description: Changes the width of the Rx frame counter, for UDP segments received with checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_UDP_ERR_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_UDP_ERR_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx TCP segments with good checksum	<p>Description: Enables an Rx frame counter for all TCP segments received with no checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_TCP_GD_FRMS_CNT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable 16-bit counter for Rx TCP segments with good checksum	<p>Description: Changes the width of the Rx frame counter, for TCP segments received with no checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_TCP_GD_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_TCP_GD_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx TCP segments with checksum error	<p>Description: Enables an Rx frame counter for all TCP segments received with a checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_TCP_ERR_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx TCP segments with checksum error	<p>Description: Changes the width of the Rx frame counter, for TCP segments received with a checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_TCP_ERR_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_TCP_ERR_FRMS_CNT_16BIT_EN</p>
Enable counter for Rx ICMP segments with good checksum	<p>Description: Enables an Rx frame counter for all ICMP segments received with no checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_ICMP_GD_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx ICMP segments with good checksum	<p>Description: Changes the width of the Rx frame counter, for ICMP segments received with no checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_ICMP_GD_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_ICMP_GD_FRMS_CNT_16BIT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable counter for Rx ICMP segments with checksum error	<p>Description: Enables an Rx frame counter for all ICMP segments received with a checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_ICMP_ERR_FRMS_CNT_EN</p>
Enable 16-bit counter for Rx ICMP segments with checksum error	<p>Description: Changes the width of the Rx frame counter, for ICMP segments received with a checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_ICMP_ERR_FRMS_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_ICMP_ERR_FRMS_CNT_16BIT_EN</p>
Enable byte counter for Rx IPv4 good frame	<p>Description: Enables an Rx octet counter for good IPv4 datagrams that encapsulate TCP, UDP or ICMP data that are received with no errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_GD_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx IPv4 good frame	<p>Description: Changes the width of the Rx octet counter, for good IPv4 datagrams that encapsulate TCP, UDP or ICMP data and are received with no errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_GD_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_GD_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx IPv4 datagrams with header error	<p>Description: Enables an Rx octet counter for all IPv4 datagrams received with a header error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_HDRERR_OCTET_CNT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable 16-bit byte counter for Rx IPv4 datagrams with header error	<p>Description: Changes the width of the Rx octet counter, for IPv4 datagrams received with a header error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_HDRERR_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_HDRERR_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx IPv4 datagrams not having TCP/UDP/ ICMP	<p>Description: Enables an Rx octet counter for all IPv4 datagrams received with a payload other than TCP, UDP, or ICMP.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_NOPAY_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx IPv4 datagrams not having TCP/UDP/ ICMP	<p>Description: Changes the width of the Rx octet counter, for IPv4 datagrams received with a payload other than TCP, UDP, or ICMP, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_NOPAY_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_NOPAY_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx IPv4 datagrams having fragmentation	<p>Description: Enables an Rx octet counter for all IPv4 datagrams received with fragments.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_FRAG_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx IPv4 datagrams having fragmentation	<p>Description: Changes the width of the Rx octet counter, for IPv4 datagrams received with fragments, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_FRAG_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_FRAG_OCTET_CNT_16BIT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable byte counter for Rx UDP over IPv4 datagrams having UDP checksum disabled	<p>Description: Enables an Rx octet counter for all IPv4 datagrams with UDP segments received with checksum disabled.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_UDSBL_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx UDP over IPv4 datagrams having UDP checksum disabled	<p>Description: Changes the width of the Rx octet counter, for IPv4 datagrams with UDP segments received with checksum disabled, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV4_UDSBL_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_UDSBL_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx IPv6 Good frame	<p>Description: Enables an Rx octet counter for good IPv6 datagrams that encapsulate TCP, UDP, or ICMP data and are received without any errors.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_GD_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx IPv6 Good frame	<p>Description: Changes the width of the Rx octet counter, for good IPv6 datagrams that encapsulate TCP, UDP, or ICMP data and are received without any errors, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV6_GD_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_GD_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx IPv6 datagrams with header error	<p>Description: Enables an Rx octet counter for IPv6 datagrams received with a header error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_HDRERR_OCTET_CNT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable 16-bit byte counter for IPv6 datagrams with header error	<p>Description: Changes the width of the Rx octet counter, for IPv6 datagrams received with a header error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV6_HDRERR_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV4_HDRERR_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx IPv6 datagrams not having TCP/UDP/ ICMP	<p>Description: Enables an Rx octet counter for IPv6 datagrams received with a payload other than TCP, UDP, or ICMP.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_NOPAY_OCTET_CNT_EN</p>
Enable 16-bit byte counter for IPv6 datagrams not having TCP/UDP/ ICMP	<p>Description: Changes the width of the Rx octet counter, for IPv6 datagrams received with a payload other than TCP, UDP, or ICMP, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_IPV6_NOPAY_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_IPV6_NOPAY_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx UDP segments with good checksum	<p>Description: Enables an Rx octet counter for all UDP segments received with no checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_UDP_GD_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx UDP segments with good checksum	<p>Description: Changes the width of the Rx octet counter, for all UDP segments received with no checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_UDP_GD_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_UDP_GD_OCTET_CNT_16BIT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable byte counter for Rx UDP segments with checksum error	<p>Description: Enables an Rx octet counter for all UDP segments received with a checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_UDP_ERR_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx UDP segments with checksum error	<p>Description: Changes the width of the Rx octet counter, for all UDP segments received with a checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_UDP_ERR_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_UPD_ERR_GD_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx TCP segments with good checksum	<p>Description: Enables an Rx octet counter for TCP segments received with no checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_TCP_GD_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx TCP segments with good checksum	<p>Description: Changes the width of the Rx octet counter, for TCP segments received with no checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_TCP_GD_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_TCP_GD_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx TCP segments with checksum error	<p>Description: Enables an Rx octet counter for TCP segments received with checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_TCP_ERR_OCTET_CNT_EN</p>

Table 7-23 GMAC Management (RMON) Receive IPC Counters (Continued)

Option	Definition
Enable 16-bit byte counter for Rx TCP segments with checksum error	<p>Description: Changes the width of the Rx octet counter, for TCP segments received with a checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_TCP_ERR_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_TCP_ERR_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx ICMP segments with good checksum	<p>Description: Enables an Rx octet counter for ICMP segments received with no checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_ICMP_GD_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx ICMP segments with good checksum	<p>Description: Changes the width of the Rx octet counter, for ICMP segments received with no checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_ICMP_GD_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_ICMP_GD_OCTET_CNT_16BIT_EN</p>
Enable byte counter for Rx ICMP segments with checksum error	<p>Description: Enables an Rx octet counter for all ICMP segments received with checksum error.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled if GMAC Management (RMON) Counters, RMON IPC Counters, Full Checksum Offload Engine (Type 2) are enabled, and Enable All Rx IPC Counters is disabled.</p> <p>HDL Parameter Name: MMC_RX_ICMP_ERR_OCTET_CNT_EN</p>
Enable 16-bit byte counter for Rx ICMP segments with checksum error	<p>Description: Changes the width of the Rx octet counter, for ICMP segments received with checksum error, to 16-bit instead of the default 32-bit.</p> <p>Value Range: N/A</p> <p>Default Value: Disabled</p> <p>Dependencies: Parameter selection is enabled only when MMC_RX_ICMP_ERR_OCTET_CNT_EN is enabled.</p> <p>HDL Parameter Name: MMC_RX_ICMP_ERR_OCTET_CNT_16BIT_EN</p>

7.2.7 Station Management Block

Table 7-24 Station Management Block

Option	Definition
Enable Station Management Block (MDIO Interface)	<p>Description: Enables the Station Management block (MDIO Master Interface).</p> <p>Value Range: N/A</p> <p>Default Value: Enabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: SMA_EN</p>

7.3 Low Power Support

Table 7-25 Low Power Support

Option	Definition
Enable UPF Based Low Power Support	<p>Description: Changes the architecture of GMAC-UNIV and adds Unified Power Format (UPF) constraints for the low-power mode support.</p> <p>Value Range: N/A</p> <p>Default Value: Enabled</p> <p>Dependencies: None</p> <p>HDL Parameter Name: LP_MODE_EN</p> <p>Note: When you enable the UPF Based Low Power Support, the Power Management Block with magic packet support is enabled by default.</p>

8

Descriptors

This chapter describes the GMAC UNIV descriptors. It contains the following sections:

- ❖ “Normal Descriptor Formats” on page 397
- ❖ “Alternate or Enhanced Descriptors” on page 414

8.1 Normal Descriptor Formats

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors, as explained in “DMA Controller” on page 55. The default descriptor formats (common for both Receive and Transmit Descriptors) are shown in Figures 8-1 to 8-10, and field descriptions are provided in Sections 8.1.1–8.1.2.



- All descriptions in Sections 8.1.1 and 8.1.2 correspond to the default descriptor format. The alternate enhanced descriptor format is described in “Alternate or Enhanced Descriptors” on page 414.
- Changes to the default descriptor format when IEEE1588 time stamping is enabled are described in “Descriptor Format With IEEE 1588 Timestamps Enabled” on page 410.

Each descriptor contains two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory management schemes.

The descriptor addresses must be aligned to the bus width used (Word/DWord/LWord for 32/64/128-bit buses). The descriptor can be configured for Same Endianness as the data bus or Reverse Endianness to the data bus (refer to “System Interface” on page 356). The data bus can be configured for either little-endian or big-endian format.

Figures 8-1 and 8-2 show the descriptor format in a 32-bit data bus.

Figure 8-1 Rx/Tx Descriptors in Same-Endian Mode for 32-Bit, Little-Endian Format; Rx/Tx Descriptors in Reverse-Endian Mode for 32-Bit, Big-Endian Format Data Bus

	31	23	15	7	0
DES0	O W N		Status [30:0]		
DES1		Control Bits [9:0]	Byte Count Buffer2 [10:0]	Byte Count Buffer1[10:0]	
DES2			Buffer1 Address[31:0]		
DES3			Buffer2 Address [31:0] / Next Descriptor Address [31:0]		

Figure 8-2 Rx/Tx Descriptors in Same-Endian Mode for 32-Bit, Big-Endian Format; Rx/Tx Descriptors in Reverse-Endian Mode for 32-Bit, Little-Endian Format Data Bus

	31	23	15	7	0
DES0	Status [7:0]	Status [15:8]	Status [23:16]	O W N	Status [30:24]
DES1	BCB1 [7:0]	BCB1 [10:8]	BCB2 [4:0]	CB [1:0]	BCB2 [10:5] Control Bits [9:2]
DES2	Buffer1 Addr [7:0]	Buffer1 Addr [15:8]	Buffer1 Addr [23:16]		Buffer1 Addr [31:24]
DES3	Buffer2 Addr / Next Desc Addr [7:0]	Buffer2 Addr / Next Desc Addr [15:8]	Buffer2 Addr / Next Desc Addr [23:16]		Buffer2 Addr / Next Desc Addr [31:24]

Figures 8-3 through 8-6 show the same descriptors with a 64-bit data bus.

Figure 8-3 Rx/Tx Descriptors in Same-Endian Mode for 64-Bit, Little-Endian Format Data Bus

	63	55	47	39	31	23	15	7	0
DES1-DES0	Control Bits [9:0]	Byte Count Buffer2 [10:0]	Byte Count Buffer1[10:0]	O W N		Status [30:0]			
DES3-DES2		Buffer2 Address [31:0] / Next Descriptor Address [31:0]			Buffer1 Address[31:0]				

Figure 8-4 Rx/Tx Descriptors in Reverse-Endian Mode for 64-Bit, Little-Endian Format Data Bus

	63	55	47	39	31	23	15	7	0	
DES1-DES0	BCB1 [7:0]	BCB2 [4:0]	BCB1 [10:8]	CB [1:0]	BCB2 [10:5]	Control Bits [9:2]	Status [7:0]	Status [15:8]	Status [23:16]	O W N Status [30:24]
DES3-DES2	Buffer2 Addr / Next Desc Addr [7:0]	Buffer2 Addr / Next Desc Addr [15:8]	Buffer2 Addr / Next Desc Addr [23:16]	Buffer2 Addr / Next Desc Addr [31:24]	Buffer1 Addr [7:0]	Buffer1 Addr [15:8]	Buffer1 Addr [23:16]	Buffer1 Addr [31:24]		

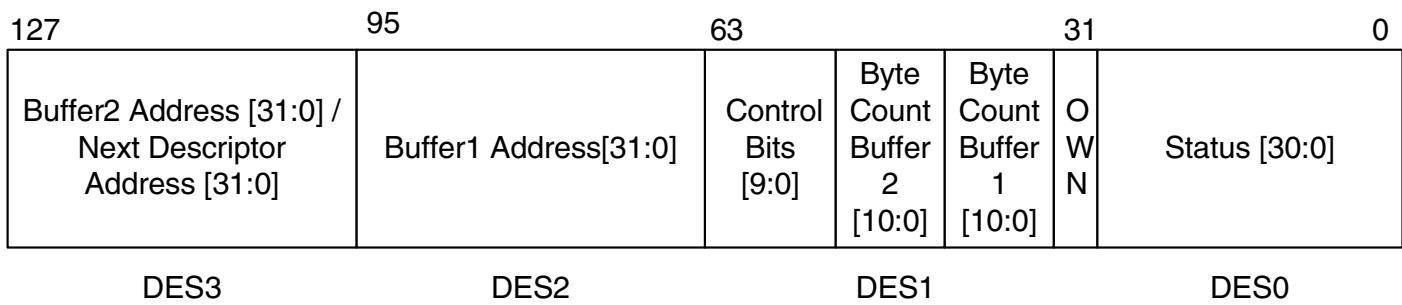
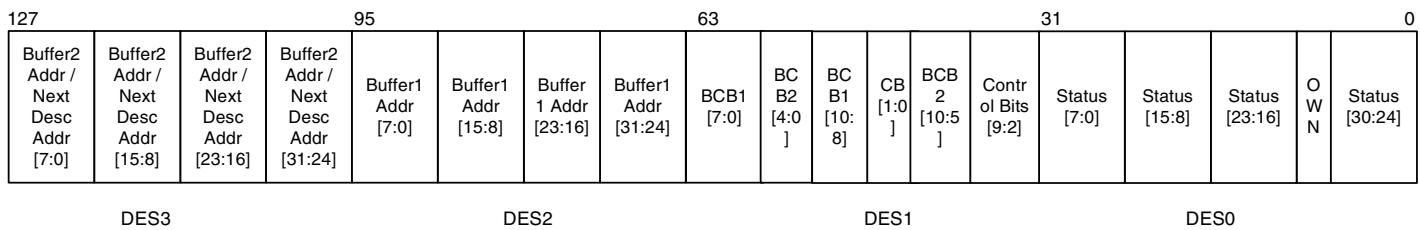
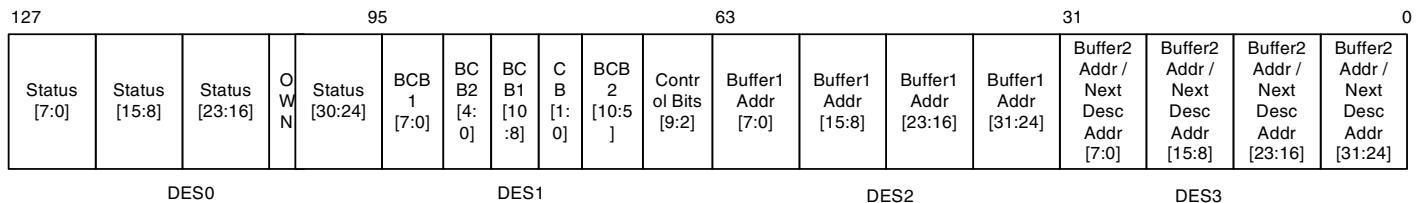
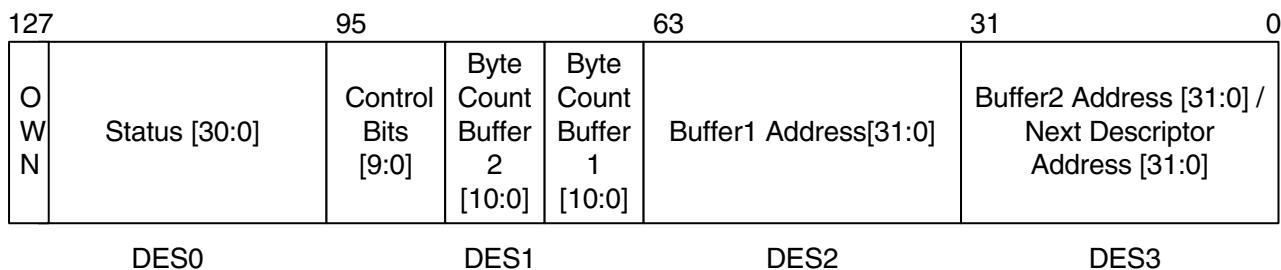
Figure 8-5 Rx/Tx Descriptors in Same-Endian Mode for 64-Bit, Big-Endian Format Data Bus

	63	55	47	39	31	23	15	7	0
DES0-DES1	Status [7:0]	Status [15:8]	Status [23:16]	O W N	Status [30:24]	BCB1 [7:0]	BCB2 [4:0]	BCB1 [10:8]	CB [1:0] BCB2 [10:5] Control Bits [9:2]
DES2-DES3	Buffer1 Addr [7:0]	Buffer1 Addr [15:8]	Buffer1 Addr [23:16]	Buffer1 Addr [31:24]	Buffer2 Addr / Next Desc Addr [7:0]	Buffer2 Addr / Next Desc Addr [15:8]	Buffer2 Addr / Next Desc Addr [23:16]	Buffer2 Addr / Next Desc Addr [31:24]	

Figure 8-6 Rx/Tx Descriptors in Reverse-Endian Mode for 64-Bit, Big-Endian Format Data Bus

	63	55	47	39	31	23	15	7	0
DES0-DES1	O W N	Status [30:0]			Control Bits [9:0]		Byte Count Buffer2 [10:0]		Byte Count Buffer1[10:0]
DES2-DES3	Buffer1 Address[31:0]					Buffer2 Address [31:0] / Next Descriptor Address [31:0]			

Figures 8-7 through 8-10 show the same descriptors with a 128-bit data bus.

Figure 8-7 Rx/Tx Descriptors in Same-Endian Mode for 128-Bit, Little-Endian Format Data Bus**Figure 8-8 Rx/Tx Descriptors in Reverse-Endian Mode for 128-Bit, Little-Endian Format Data Bus****Figure 8-9 Rx/Tx Descriptors in Same-Endian Mode for 128-Bit, Big-Endian Format Data Bus****Figure 8-10 Rx/Tx Descriptors in Reverse-Endian Mode for 128-Bit, Big-Endian Format Data Bus**

In big-endian data bus format with the Same Endianness descriptor format, the descriptor bytes are swapped internally. If the DUT is configured for a 32-bit data bus (Figure 8-2), then byte lanes 3 and 0 are swapped while byte lanes 1 and 2 are swapped. That is, bits [31:24] are available on data bus [7:0], and vice-versa. Hence, you must ensure that the descriptors are created accordingly in system memory. This is true for little-endian data bus with Reverse Endianness for descriptors.

Similarly, byte swapping is illustrated in Figures 8-4 and 8-5 for a 64-bit data bus and in Figures 8-8 and 8-9 for 128-bit data bus.

In 64-bit big-endian data bus systems, when the descriptors are configured for Reverse Endianness ([Figure 8-6](#)), the order of the descriptor subset is reversed from the default ([Figure 8-3](#)). For example, bits [63:32] are DES0 while bits [31:0] are DES1.

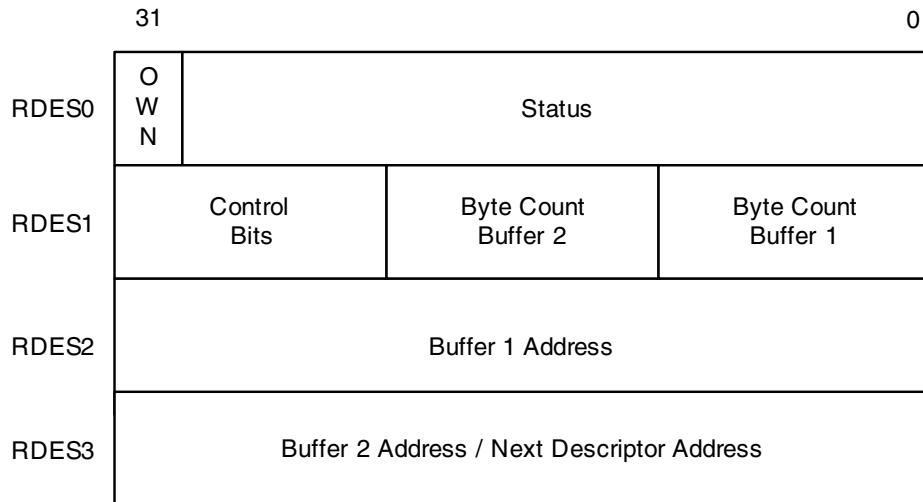
[Figure 8-10](#) illustrates similar phenomenon for 128-bit big-endian data bus with Reverse Endian descriptors.

8.1.1 Receive Descriptor

The GMAC Subsystem requires at least two descriptors when receiving a frame. The Receive state machine of the DMA (in the GMAC Subsystem) always attempts to acquire an extra descriptor in anticipation of an incoming frame. (The size of the incoming frame is unknown). Before the RxDMA closes a descriptor, it attempts to acquire the next descriptor even if no frames are received.

In a single descriptor (receive) system, the subsystem generates a descriptor error if the receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. Thus, the Host is forced to increase either its descriptor pool or the buffer size. Otherwise, the subsystem starts dropping all incoming frames.

Figure 8-11 Receive Descriptor Format in Little-Endian Mode With a 32-bit Data Bus



8.1.1.1 Receive Descriptor 0 (RDES0)

RDES0 contains the received frame status, the frame length, and the descriptor ownership information. The descriptions in the following tables ([Tables 8-1](#) through [8-9](#)) are for the default mode of a little-endian 32-bit data bus with Same Endian descriptors, or big-endian data bus with Reverse-Endian descriptors. If the DUT is configured for big-endian mode, 32-bit data bus with Same Endian descriptors or little-endian data bus with Reverse Endian descriptors, then byte lanes 3 and 0 are swapped while byte lanes 1 and 2 are swapped, that is, bits [31:24] are available on data bus [7:0], and vice-versa.

Table 8-1 Receive Descriptor 0

Bit	Description
31	<p>OWN: Own Bit</p> <p>When set, this bit indicates that the descriptor is owned by the DMA of the GMAC Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.</p>

Table 8-1 Receive Descriptor 0 (Continued)

Bit	Description
30	AFM: Destination Address Filter Fail When set, this bit indicates a frame that failed in the DA Filter in the GMAC Core.
29:16	FL: Frame Length These bits indicate the byte length of the received frame that was transferred to host memory (including CRC). This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bits are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame. This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame.
15	ES: Error Summary Indicates the logical OR of the following bits: <ul style="list-style-type: none"> • RDES0[0]: Payload Checksum Error • RDES0[1]: CRC Error • RDES0[3]: Receive Error • RDES0[4]: Watchdog Timeout • RDES0[6]: Late Collision • RDES0[7]: IPC Checksum (Type 2) / Giant Frame • RDES0[11]: Overflow Error • RDES0[14]: Descriptor Error This field is valid only when the Last Descriptor (RDES0[8]) is set.
14	DE: Descriptor Error When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.
13	SAF: Source Address Filter Fail When set, this bit indicates that the SA field of frame failed the SA Filter in the GMAC Core.
12	LE: Length Error When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset. Length error status is not valid when CRC error is present.
11	OE: Overflow Error When set, this bit indicates that the received frame was damaged due to buffer overflow in MTL.
10	VLAN: VLAN Tag When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the GMAC Core.
9	FS: First Descriptor When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.

Table 8-1 Receive Descriptor 0 (Continued)

Bit	Description
8	LS: Last Descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame
7	IPC Checksum Error/Giant Frame When IP Checksum Engine (Type 1) is enabled, this bit, when set, indicates that the 16-bit IPv4 Header checksum calculated by the core did not match the received checksum bytes. The Error Summary bit[15] is NOT set when this bit is set in this mode. If this bit is set when Full Checksum Offload Engine (Type 2) is enabled, it indicates an error in the IPv4 or IPv6 header. This error can be due to inconsistent Ethernet Type field and IP header Version field values, a header checksum mismatch in IPv4, or an Ethernet frame lacking the expected number of IP header bytes. Refer to Table 8-2 for more details. If you do not select IP Checksum Module during core configuration, this bit, when set, indicates that the received frame was a Giant Frame. Giant frames are larger-than-1,518-byte (or 1,522-byte for VLAN) normal frames and larger-than-9,018-byte (9,022-byte for VLAN) frame when Jumbo Frame processing is enabled.
6	LC: Late Collision When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.
5	FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes. Refer to Table 8-2 for more details.
4	RWT: Receive Watchdog Timeout When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.
3	RE: Receive Error When set, this bit indicates that the gmii_rxer_i signal is asserted while gmii_rxdv_i is asserted during frame reception. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error ($rxd \neq 0f$) during extension.
2	DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.
1	CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.
0	Rx MAC Address/Payload Checksum Error When set, this bit indicates that the Rx MAC Address registers value (1 to 15) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field. If Full Checksum Offload Engine is enabled, this bit, when set, indicates the TCP, UDP, or ICMP checksum the core calculated does not match the received encapsulated TCP, UDP, or ICMP segment's Checksum field. This bit is also set when the received number of payload bytes does not match the value indicated in the Length field of the encapsulated IPv4 or IPv6 datagram in the received Ethernet frame. Refer to Table 8-2 for more details.

When the Full Checksum Offload Engine (Type 2) is enabled, the permutations of bits 5, 7, and 0 reflect the conditions discussed in [Table 8-2](#).

Table 8-2 Receive Descriptor 0 When COE (Type 2) Is Enabled

Bit 5: Frame Type	Bit 7: IPC Checksum Error	Bit 0: Payload Checksum Error	Frame Status
0	0	0	IEEE 802.3 Type frame (Length field value is less than 0x0600.)
1	0	0	IPv4/IPv6 Type frame, no checksum error detected
1	0	1	IPv4/IPv6 Type frame with a payload checksum error (as described for PCE) detected
1	1	0	IPv4/IPv6 Type frame with an IP header checksum error (as described for IPC CE) detected
1	1	1	IPv4/IPv6 Type frame with both IP header and payload checksum errors detected
0	0	1	IPv4/IPv6 Type frame with no IP header checksum error and the payload check bypassed, due to an unsupported payload
0	1	1	A Type frame that is neither IPv4 or IPv6 (the Checksum Offload engine bypasses checksum completely.)
0	1	0	Reserved



The first five conditions are backward-compatible to versions 3.30a and previous. The last two conditions (001, 011), which had been reserved, are not backward-compatible, because the Frame Type (FT) bit is reset during bypasses, even for valid Type frames.

8.1.1.2 Receive Descriptor 1 (RDES1)

RDES1 contains the buffer sizes and other bits that control the descriptor chain/ring.



See “[Buffer Size Calculations](#)” on page 58 for further detail on calculating buffer sizes.

Table 8-3 Receive Descriptor 1

Bit	Description
31	Disable Interrupt on Completion When set, this bit prevents the setting of the RI (CSR5[6]) bit of the Status Register for the received frame that ends in the buffer pointed to by this descriptor. This, in turn, disables the assertion of the interrupt to Host due to RI for that frame.
30:26	Reserved

Table 8-3 Receive Descriptor 1 (Continued)

Bit	Description
25	RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring.
24	RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When RDES1[24] is set, RBS2 (RDES1[21-11]) is a “don’t care” value. RDES1[25] takes precedence over RDES1[24].
23:22	Reserved
21:11	RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size in bytes. The buffer size must be a multiple of 4/8/16 depending upon the bus widths (32/64/128), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. This field is not valid if RDES1[24] is set.
10:0	RBS1: Receive Buffer 1 Size Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4/8/16 depending upon the bus widths (32/64/128), even if the value of RDES2 (buffer1 address pointer) is not aligned. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 24).

8.1.1.3 Receive Descriptor 2 (RDES2)

RDES2 contains the address pointer to the first data buffer in the descriptor.



See “[Host Data Buffer Alignment](#)” on page [57](#) for further detail on buffer address alignment.

Table 8-4 Receive Descriptor 2 (Default Operation)

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[3/2/1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3/2/1:0] (corresponding to bus width of 128/64/32) if the address pointer is to a buffer where the middle or last part of the frame is stored.

8.1.1.4 Receive Descriptor 3 (RDES3)

RDES3 contains the address pointer either to the second data buffer in the descriptor or to the next descriptor.

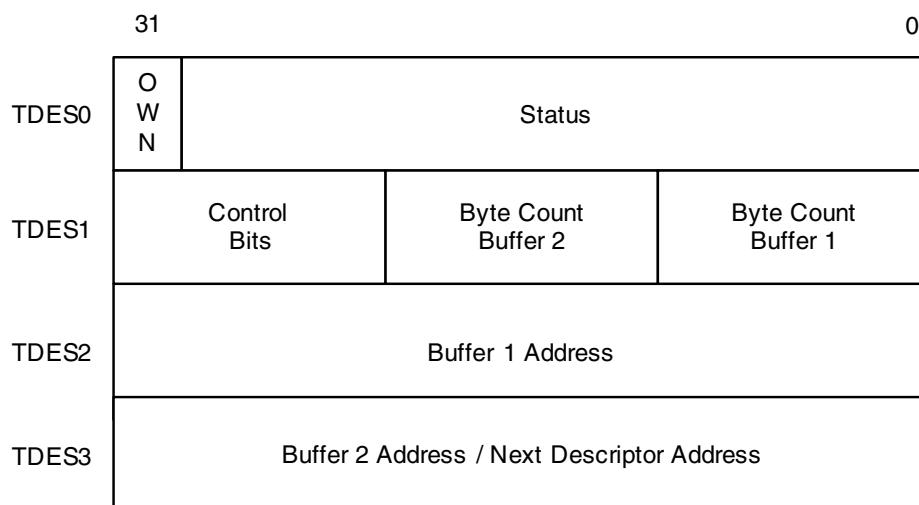
Table 8-5 Receive Descriptor 3

Bit	Description
31:0	<p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present.</p> <p>If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[3, 2, or 1:0] = 0, corresponding to a bus width of 128, 64, or 32. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[3, 2, or 1:0] (corresponding to a bus width of 128, 64, or 32) if the address pointer is to a buffer where the middle or last part of the frame is stored.</p>

8.1.2 Transmit Descriptor

The descriptor addresses must be aligned to the bus width used (32/64/128). [Figure 8-12](#) shows the transmit descriptor format in little-endian mode with a 32-bit data bus. Refer to [Figures 8-3](#) through [8-10](#) to see the expected descriptor format in other configurations.

Each descriptor is provided with two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory-management schemes.

Figure 8-12 Transmit Descriptor Format in Little-Endian Mode With a 32-bit Data Bus

8.1.2.1 Transmit Descriptor 0 (TDES0)

TDES0 contains the transmitted frame status and the descriptor ownership information.

Table 8-6 Transmit Descriptor 0

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty. The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.
30:18	Reserved
17	TTSS: Tx Timestamp Status This status bit indicates that a timestamp has been captured for the corresponding transmit frame. When this bit is set, TDES2 and TDES3 have timestamp values that were captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES1[30]) in a descriptor is set. This bit is valid only when IEEE1588 time stamping feature is enabled; otherwise, it is reserved.
16	IHE: IP Header Error When set, this bit indicates that the Checksum Offload engine detected an IP header error and consequently did not modify the transmitted frame for any checksum insertion. This bit is valid only when Full Checksum Offload is enabled; otherwise, it is reserved.
15	ES: Error Summary Indicates the logical OR of the following bits: <ul style="list-style-type: none"> • TDES0[14]: Jabber Timeout • TDES0[13]: Frame Flush • TDES0[11]: Loss of Carrier • TDES0[10]: No Carrier • TDES0[9]: Late Collision • TDES0[8]: Excessive Collision • TDES0[2]: Excessive Deferral • TDES0[1]: Underflow Error
14	JT: Jabber Timeout When set, this bit indicates the GMAC transmitter has experienced a jabber time-out. This bit is only set when the GMAC configuration register's JD bit is not set.
13	FF: Frame Flushed When set, this bit indicates that the DMA/MTL flushed the frame due to a SW flush command given by the CPU.

Table 8-6 Transmit Descriptor 0 (Continued)

Bit	Description
12	<p>PCE: Payload Checksum Error</p> <p>This bit, when set, indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either due to insufficient bytes, as indicated by the IP Header's Payload Length field, or the MTL starting to forward the frame to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet frame being transmitted: to avoid deadlock, the MTL starts forwarding the frame when the FIFO is full, even in Store-and-Forward mode.</p> <p>When the Full Checksum Offload engine is not enabled during configuration, this bit is reserved.</p>
11	<p>LC: Loss of Carrier</p> <p>When set, this bit indicates that Loss of Carrier occurred during frame transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during frame transmission). This is valid only for the frames transmitted without collision and when the GMAC operates in Half-Duplex Mode.</p>
10	<p>NC: No Carrier</p> <p>When set, this bit indicates that the carrier sense signal from the PHY was not asserted during transmission.</p>
9	<p>LC: Late Collision</p> <p>When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte times including Preamble in MII Mode and 512 byte times including Preamble and Carrier Extension in GMII Mode). Not valid if Underflow Error is set.</p>
8	<p>EC: Excessive Collision</p> <p>When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the GMAC Configuration Register is set, this bit is set after the first collision and the transmission of the frame is aborted.</p>
7	<p>VF: VLAN Frame</p> <p>When set, this bit indicates that the transmitted frame was a VLAN-type frame.</p>
6:3	<p>CC: Collision Count</p> <p>This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.</p>
2	<p>ED: Excessive Deferral</p> <p>When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000-Mbps mode, or in Jumbo Frame enabled mode) if the Deferral Check (DC) bit is set high in the GMAC Control Register.</p>
1	<p>UF: Underflow Error</p> <p>When set, this bit indicates that the GMAC aborted the frame because data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty Transmit Buffer while transmitting the frame. The transmission process enters the suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]).</p>
0	<p>DB: Deferred Bit</p> <p>When set, this bit indicates that the GMAC defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.</p>

8.1.2.2 Transmit Descriptor 1 (TDES1)

TDES1 contains the buffer sizes and other bits which control the descriptor chain/ring and the frame being transferred.



See “[Buffer Size Calculations](#)” on page [58](#) for further detail on calculating buffer sizes.

Table 8-7 **Transmit Descriptor 1**

Bit	Description
31	IC: Interrupt on Completion When set, this bit sets Transmit Interrupt (Register 5[0]) after the present frame has been transmitted.
30	LS: Last Segment When set, this bit indicates that the buffer contains the last segment of the frame. When this bit is set, the TBS1: Transmit Buffer 1 Size or TBS2: Transmit Buffer 2 Size field should have a non-zero value.
29	FS: First Segment When set, this bit indicates that the buffer contains the first segment of a frame.
28:27	CIC: Checksum Insertion Control These bits control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6 as described below. <ul style="list-style-type: none"> • 2'b00: Do nothing. Checksum Engine is bypassed • 2'b01: Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram. • 2'b10: Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, UDP, or ICMP segment only and the TCP, UDP, or ICMP pseudo-header checksum is assumed to be present in the corresponding input frame’s Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram conforms to IPv4. • 2'b11: Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. In other words, the TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame’s corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4. The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly.
26	DC: Disable CRC When set, the GMAC does not append the Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES1[29]).
25	TER: Transmit End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The returns to the base address of the list, creating a descriptor ring.
24	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES1[24] is set, TBS2 (TDES1[21–11]) are “don’t care” values. TDES1[25] takes precedence over TDES1[24].

Table 8-7 Transmit Descriptor 1 (Continued)

Bit	Description
23	DP: Disable Padding When set, the GMAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes and the CRC field is added despite the state of the DC (TDES1[26]) bit. This is valid only when the first segment (TDES1[29]) is set.
22	TTSE: Transmit Timestamp Enable When set, this bit enables IEEE1588 hardware time stamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES1[29]) is set.
21:11	TBS2: Transmit Buffer 2 Size These bits indicate the Second Data Buffer in bytes. This field is not valid if TDES1[24] is set.
10:0	TBS1: Transmit Buffer 1 Size These bits indicate the First Data Buffer byte size. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of TCH (Bit 24).

8.1.2.3 Transmit Descriptor 2 (TDES2)

TDES2 contains the address pointer to the first buffer of the descriptor.

Table 8-8 Transmit Descriptor 2

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. See “ Host Data Buffer Alignment ” on page 57 for further detail on buffer address alignment.

8.1.2.4 Transmit Descriptor 3 (TDES3)

TDES3 contains the address pointer either to the second buffer of the descriptor or the next descriptor.

Table 8-9 Transmit Descriptor 3

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)

8.1.3 Descriptor Format With IEEE 1588 Timestamps Enabled

The default descriptor format (as described in “[Receive Descriptor](#)” on page 401 and “[Transmit Descriptor](#)” on page 406), and field descriptions remain unchanged when created by software (Own bit is set in DES0). However, if the software has enabled IEEE 1588 functionality, the DES2 and DES3 descriptor fields (see [Figure 8-13](#)) take on a different meaning when the DMA closes the descriptor (own bit in DES0 is cleared).

The DMA updates the DES2 and DES3 with the timestamp value before clearing the Own bit in DES0.

When the core is operating in 32-bit mode ([Figures 8-1](#) and [8-2](#)), DES2 is updated with the lower 32 timestamp bits (the Sub-Second field, called TSL in subsequent sections) and DES3 is updated with the upper 32 timestamp bits (the Seconds field, called TSH in subsequent sections).

Figure 8-13 Receive Descriptor Fields When DMA Clears the Own Bit

31	1
DES0	
DES1	
DES2	Timestamp Low[31:0]
DES3	Timestamp High[31:0]

The above description holds true for all descriptor formats depicted in [Figures 8-3](#), [8-5](#), [8-7](#), and [8-9](#). However, when the core operates in 64- or 128-bit and with a reverse-endian descriptor ([Figures 8-4](#), [8-6](#), [8-8](#), and [8-10](#)), DES2 is updated with TSH and DES3 is updated with TSL. This ensures that the 64-bit timestamp can be processed as-is without any swapping between the two 32-bit words.

The following sections describe the details specific to receive and transmit descriptors in this mode.

8.1.3.1 Receive Descriptor

8.1.3.1.1 Receive Timestamp

The following tables describe the fields that have different meaning for RDES2 and RDES3 when the receive descriptor is closed and time stamping is enabled. The fields in RDES2 and RDES3 are swapped when the core operates in 64-bit or 128-bit and with the descriptor in Reverse-Endian mode.



Note When software disables the Timestamp feature (the TSENA bit in Register 448 is low), the DMA does not update the descriptor's RDES2/RDES3 fields before closing the RDES0.

Table 8-10 Receive Descriptor Fields (RDES2)

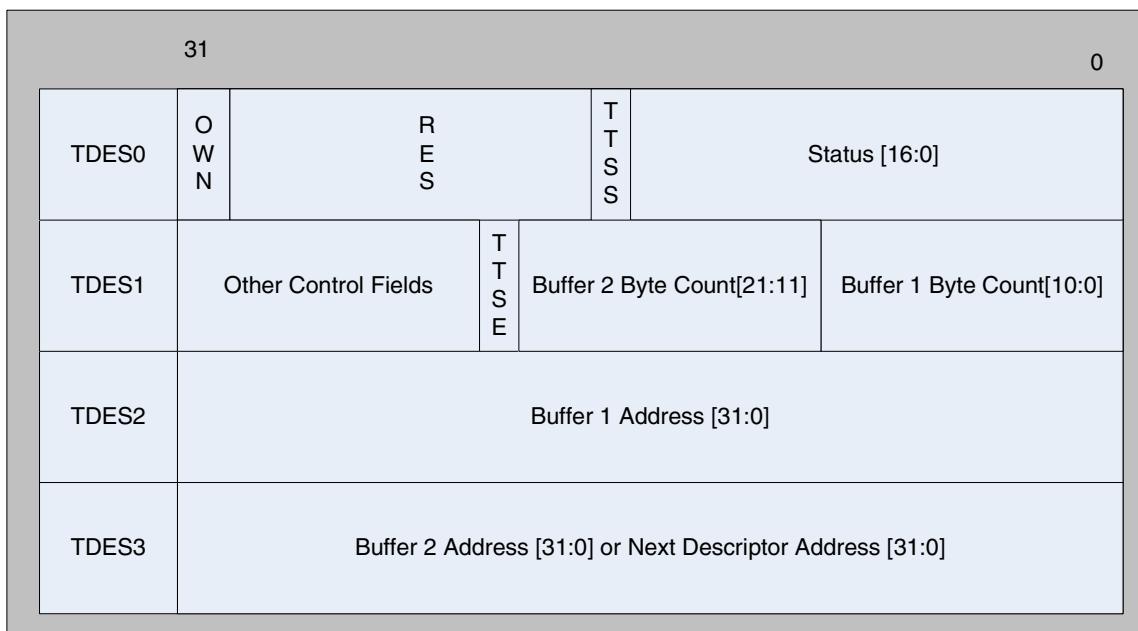
Bit	Description
31:0	<p>RTSL: Receive Frame Timestamp Low</p> <p>The DMA updates this field with the least significant 32 bits of the timestamp captured for the corresponding receive frame. The DMA updates this field only for the last descriptor of the receive frame indicated by Last Descriptor status bit (RDES0[8]). When this field and the RTSH field in RDES3 show an all-ones value, the timestamp must be treated as corrupt.</p>

Table 8-11 Receive Descriptor Fields (RDES3)

Bit	Description
31:0	<p>RTSH: Receive Frame Timestamp High The DMA updates this field with the most significant 32 bits of the timestamp captured for the corresponding receive frame. The DMA updates this field only for the last descriptor of the receive frame indicated by Last Descriptor status bit (RDES0[8]). When this field and RDES2's RTSL field show all-ones values, the timestamp must be treated as corrupt.</p>

8.1.3.2 Transmit Descriptor

In addition to the changes described in “Descriptor Format With IEEE 1588 Timestamps Enabled” on page 410, the Transmit descriptor has additional control and status bits (TTSE and TTSS, respectively) for time stamping, as shown in Figure 8-14. Software sets the TTSE bit (when the Own bit is set), instructing the core to generate a timestamp for the corresponding Ethernet frame being transmitted. The DMA sets the TTSS bit if the timestamp has been updated in the TDES2 and TDES3 fields when the descriptor is closed (Own bit is cleared).

Figure 8-14 Transmit Descriptor Fields– Normal Format

8.1.3.2.1 Transmit Timestamp Control and Status Fields

The position of these fields is different for normal transmit descriptor and enhanced format transmit descriptor. The value of this field in both the cases shall be preserved by the DMA at the time of closing the descriptor.

Updates to Tables 8-5 and 8-6 for the default (normal) descriptor format are described below.

Table 8-12 Transmit Timestamp Status – Normal Descriptor Format Case (TDES0)

Bit	Description
17	<p>TTSS: Transmit Timestamp Status</p> <p>This field is a status bit indicating that a timestamp was captured for the corresponding transmit frame. When this bit is set, both TDES2 and TDES3 have a timestamp value that was captured for the transmit frame.</p> <p>This field is valid only when the Last Segment control bit (TDES1[30] in the descriptor) is set.</p>

Table 8-13 Transmit Timestamp Control – Normal Descriptor Format Case (TDES1)

Bit	Description
22	<p>TTSE: Transmit Timestamp Enable</p> <p>When set, this field enables IEEE1588 hardware time stamping for the transmit frame described by the descriptor.</p> <p>This field is valid only when the First Segment control bit (TDES1[29] in the descriptor) is set.</p>

8.1.3.2.2 Transmit Timestamp Field

The transmit descriptor format and field descriptions remain unchanged when they are created by software (when the Own bit is set). However, when the DMA closes the last descriptor (marked, in the alternative descriptor format, by the LS bit in TDES1 or TDES0) and IEEE 1588 functionality is enabled (the Own bit is cleared), the TDES2 and TDES3 descriptor fields are updated with the timestamp, if taken, for that frame.

The fields in TDES2 and TDES3 are swapped when the core operates in 64- or 128-bit and with the descriptor in Reverse-Endian mode

Tables 8-14 and 8-15 describe the fields that have different meaning when the descriptor is closed.

Table 8-14 Transmit Descriptor Fields (TDES2)

Bit	Description
31:0	<p>TTSL: Transmit Frame Timestamp Low</p> <p>This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment control bit (LS) in the descriptor is set.</p>

Table 8-15 Transmit Descriptor Fields (TDES3)

Bit	Description
31:0	<p>TTSH: Transmit Frame Timestamp High</p> <p>This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment control bit (LS) in the descriptor is set.</p>

8.2 Alternate or Enhanced Descriptors

The alternate (or enhanced) descriptor structure can have 8 DWORDS (32-bytes) instead of the 4 DWORDS as in the case of normal descriptor format. The alternate (or enhanced) descriptor structure is the only supported format when the IEEE 1588-2008 Advanced Timestamp feature or the AV feature is enabled. The features of the alternate descriptor structure are:

- ❖ The normal descriptor structure allows data buffers of up to 2,048 bytes. The alternative descriptor structure has been implemented to support buffers of up to 8 KB (useful for Jumbo frames).
- ❖ There is a re-assignment of control and status bits in TDES0, TDES1, RDES0 (Advanced timestamp or IPC full offload configuration), and RDES1.
- ❖ The transmit descriptor stores the timestamp in TDES6 and TDES7 when you select the Advanced Timestamp.
- ❖ This receive descriptor structure is also used for storing the extended status (RDES4) and timestamp (RDES6 and RDES7) when advanced timestamp feature or IPC full offload is selected.
- ❖ When alternate descriptor mode is selected, and Timestamp feature is enabled, the software needs to allocate 32-bytes (8 DWORDS) of memory for every descriptor. When Timestamping or Receive IPC FullOffload engine are not enabled, the extended descriptors are not required and the SW can use alternate descriptors with the default size of 16 bytes. The core also needs to be configured for this change using the bit 7 ([ATDS: Alternate Descriptor Size](#)) of [Register 0 \(Bus Mode Register\)](#).
- ❖ When alternate descriptor is chosen without Timestamp or Full IPC Offload feature, the descriptor size is always 4 DWORDS (DES0-DES3).

The description or bit-mapping alternate descriptor structure (in little-endian mode) is given below.



- The effect of big-endian mode (byte-swap) explained in [“Normal Descriptor Formats”](#) on page 397 apply to this descriptor structure as well.
- When alternate descriptor with only Full IPC Checksum Offload (Type 2) is selected, it is not backward compatible to the previous release 3.4x with respect to status bits[7,5,0] in RDES0. In this mode, you should enable the extended descriptor mode (8 DWORDS) to get the IPC checksum engine status in RDES4.

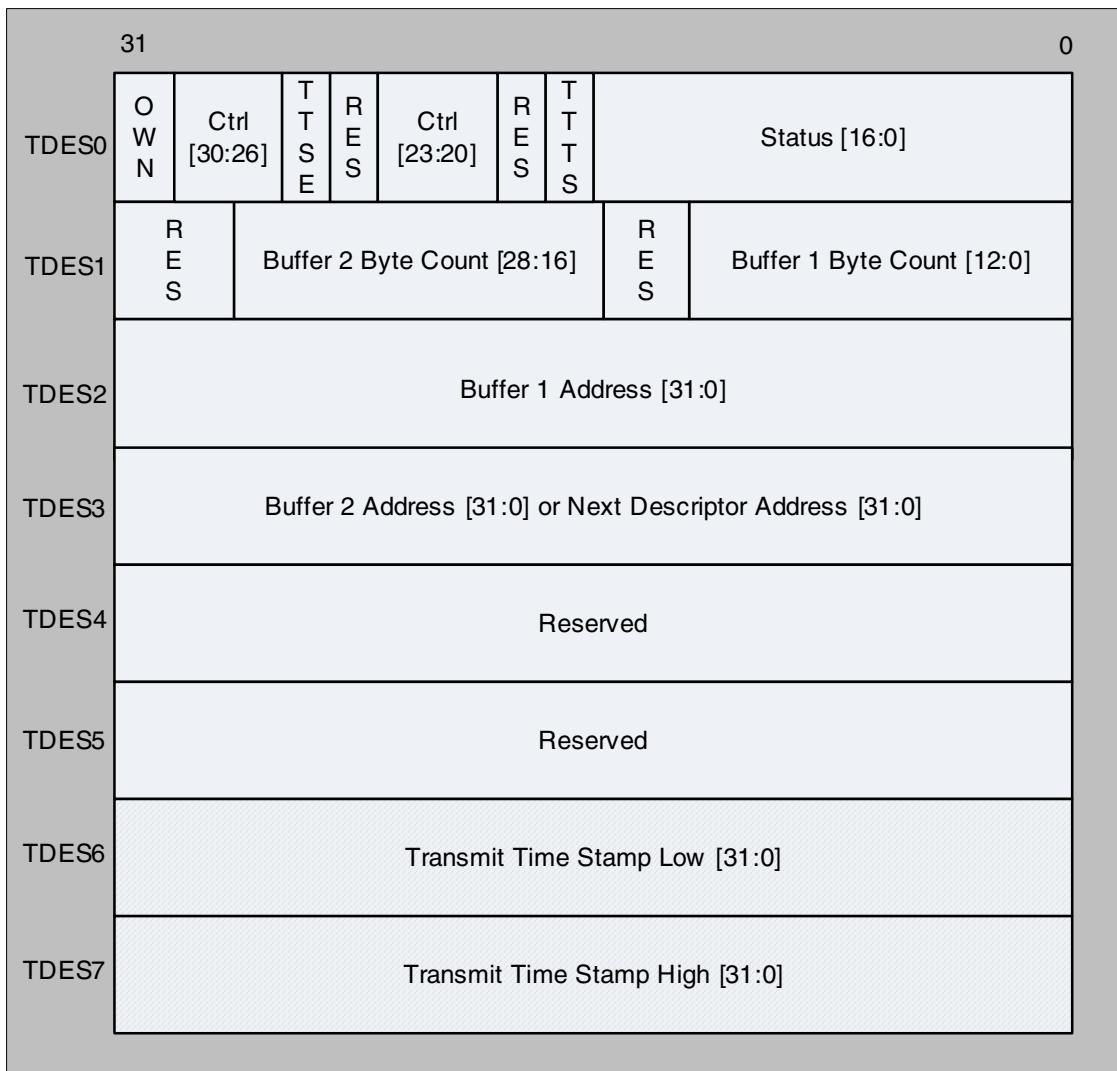
8.2.1 Transmit Descriptor

The transmit descriptor structure is shown in [Figure 8-15](#). The application software must program the control bits TDES0[31:20] during descriptor initialization. When the DMA updates the descriptor, it writes back all the control bits except the OWN bit (which it clears) and updates the status bits[19:0]. The contents of the transmitter descriptor word 0 (TDES0) through word 3 (TDES3) are given in [Tables 8-16](#) through [8-19](#), respectively.

With the advance timestamp support, the snapshot of the timestamp to be taken can be enabled for a given frame by setting the “TTSE: Transmit Timestamp Enable” (bit-25 of TDES0). When the descriptor is closed (i.e. when the OWN bit is cleared), the time-stamp is written into TDES6 and TDES7. This is indicated by the status bit “TTSS: Transmit Timestamp Status” (bit-17 of TDES0). This is shown in [Figure 8-15](#). The contents of TDES6 and TDES7 are mentioned in [Table 8-20](#) and [Table 8-21](#).

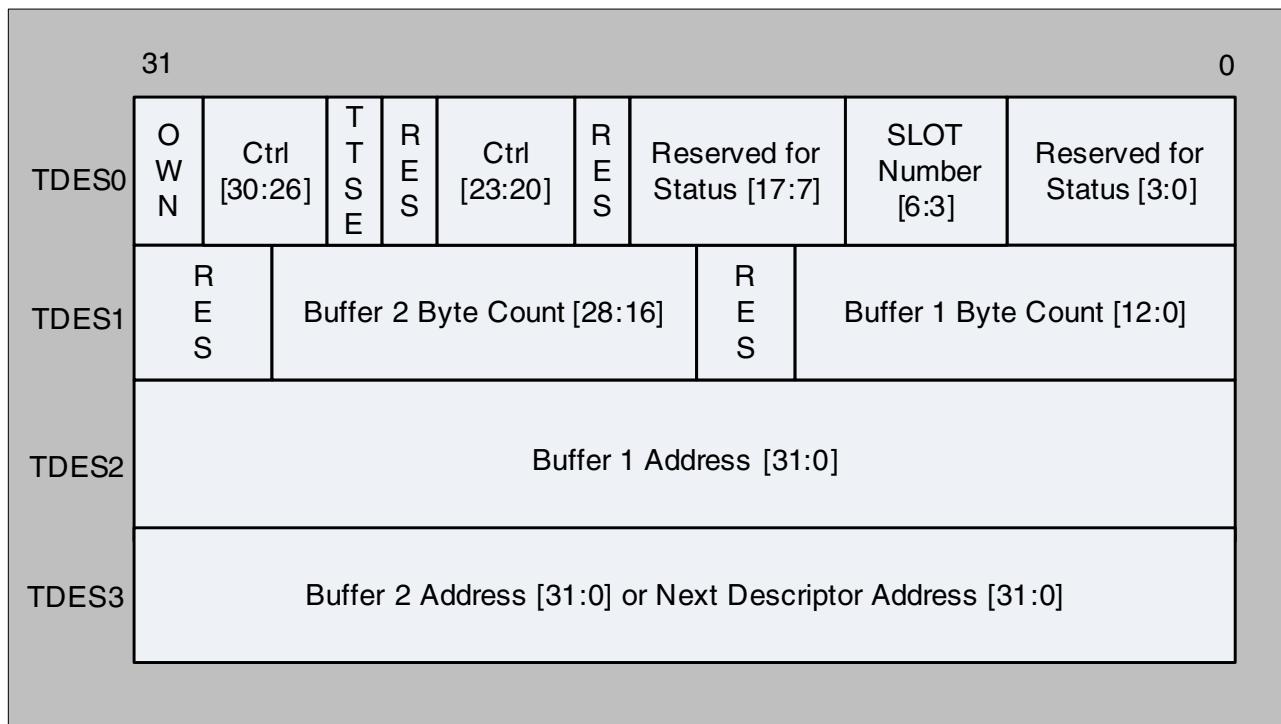


When either Advanced Timestamp or IPC Offload (Type 2) features are enabled, the SW should set the DMA Bus Mode register[7], so that the DMA operates with extended descriptor size. When this control bit is reset, the TDES4-TDES7 descriptor space are not valid.

Figure 8-15 Transmitter Descriptor Fields - Alternate (Enhanced) Format

The description for TDES6 and TDES7 in [Figures 8-15](#) is applicable to all descriptor formats depicted in [Figures 8-3, 8-5, 8-7, and 8-9](#). However, when the core operates in 64-bit or 128-bit and with a reverse-endian descriptor ([Figures 8-4, 8-6, 8-8, and 8-10](#)), TDES6 is updated with TTSH and TDES7 is updated with TTSL. This ensures that the 64-bit timestamp can be processed as it is without any kind of swapping between the two 32-bit words.

The DMA always reads or fetches four DWORDS of the descriptor from system memory to obtain the buffer and control information as shown in [Figure 8-16](#). When AV feature support is enabled, TDES0 has additional control bits[6:3] for channel 1 and channel 2. For channel 0, the bits 6:3 are ignored. The bits 6:3 are described in [Table 8-16](#).

Figure 8-16 Transmit Descriptor Fetch (Read) for Alternate (Enhanced) Format**Table 8-16 Transmit Descriptor Word 0 (TDES0)**

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are read completely. The ownership bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.
30	IC: Interrupt on Completion When set, this bit sets the Transmit Interrupt (Register 5[0]) after the present frame has been transmitted.
29	LS: Last Segment When set, this bit indicates that the buffer contains the last segment of the frame. When this bit is set, the TBS1: Transmit Buffer 1 Size or TBS2: Transmit Buffer 2 Size field in TDES1 should have a non-zero value.
28	FS: First Segment When set, this bit indicates that the buffer contains the first segment of a frame.
27	DC: Disable CRC When this bit is set, the GMAC does not append a cyclic redundancy check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set.

Table 8-16 Transmit Descriptor Word 0 (TDES0) (Continued)

Bit	Description
26	DP: Disable Pad When set, the GMAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit. This is valid only when the first segment (TDES0[28]) is set.
25	TTSE: Transmit Timestamp Enable When set, this bit enables IEEE1588 hardware time stamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES0[28]) is set.
24	Reserved
23:22	CIC: Checksum Insertion Control These bits control the checksum calculation and insertion. Bit encodings are as shown below. <ul style="list-style-type: none"> • 2'b00: Checksum Insertion Disabled. • 2'b01: Only IP header checksum calculation and insertion are enabled. • 2'b10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. • 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is reserved when the IPC_FULL_OFFLOAD configuration parameter is not selected.
21	TER: Transmit End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.
20	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a “don’t care” value. TDES0[21] takes precedence over TDES0[20].
19:18	Reserved
17	TTSS: Transmit Timestamp Status This field is used as a status bit to indicate that a timestamp was captured for the described transmit frame. When this bit is set, TDES2 and TDES3 have a timestamp value captured for the transmit frame. This field is only valid when the descriptor’s Last Segment control bit (TDES0[29]) is set.
16	IHE: IP Header Error When set, this bit indicates that the GMAC transmitter detected an error in the IP datagram header. The transmitter checks the header length in the IPv4 packet against the number of header bytes received from the application and indicates an error status if there is a mismatch. For IPv6 frames, a header error is reported if the main header length is not 40 bytes. Furthermore, the Ethernet Length/Type field value for an IPv4 or IPv6 frame must match the IP header version received with the packet. For IPv4 frames, an error status is also indicated if the Header Length field has a value less than 0x5.

Table 8-16 Transmit Descriptor Word 0 (TDES0) (Continued)

Bit	Description
15	<p>ES: Error Summary Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • TDES0[14]: Jabber Timeout • TDES0[13]: Frame Flush • TDES0[11]: Loss of Carrier • TDES0[10]: No Carrier • TDES0[9]: Late Collision • TDES0[8]: Excessive Collision • TDES0[2]: Excessive Deferral • TDES0[1]: Underflow Error • TDES0[16]: IP Header Error • TDES0[12]: IP Payload Error
14	<p>JT: Jabber Timeout When set, this bit indicates the GMAC transmitter has experienced a jabber time-out. This bit is only set when the GMAC configuration register's JD bit is not set.</p>
13	<p>FF: Frame Flushed When set, this bit indicates that the DMA/MTL flushed the frame due to a software Flush command given by the CPU.</p>
12	<p>IPE: IP Payload Error When set, this bit indicates that GMAC transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload. The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch.</p>
11	<p>LC: Loss of Carrier When set, this bit indicates that a loss of carrier occurred during frame transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during frame transmission). This is valid only for the frames transmitted without collision when the GMAC operates in Half-Duplex mode.</p>
10	<p>NC: No Carrier When set, this bit indicates that the Carrier Sense signal from the PHY was not asserted during transmission.</p>
9	<p>LC: Late Collision When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte-times, including preamble, in MII mode and 512 byte-times, including preamble and carrier extension, in GMII mode). This bit is not valid if the Underflow Error bit is set.</p>
8	<p>EC: Excessive Collision When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the GMAC Configuration register is set, this bit is set after the first collision, and the transmission of the frame is aborted.</p>
7	<p>VF: VLAN Frame When set, this bit indicates that the transmitted frame was a VLAN-type frame.</p>

Table 8-16 Transmit Descriptor Word 0 (TDES0) (Continued)

Bit	Description
6:3	CC: Collision Count (Status field) These status bits indicate the number of collisions that occurred before the frame was transmitted. This count is not valid when the Excessive Collisions bit (TDES0[8]) is set. The core updates this status field only in the half-duplex mode. -or- SLOTNUM: Slot Number Control Bits in AV Mode These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES2 or TDES3. When the transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the core (Register 11xx). It fetches the data from the buffers only if there is a match in values. These bits are valid only for the AV channels (not channel 0).
2	ED: Excessive Deferral When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1,000-Mbps mode or if Jumbo Frame is enabled) if the Deferral Check (DC) bit in the GMAC Control register is set high.
1	UF: Underflow Error When set, this bit indicates that the GMAC aborted the frame because data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty transmit buffer while transmitting the frame. The transmission process enters the Suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]).
0	DB: Deferred Bit When set, this bit indicates that the GMAC defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.

Table 8-17 Transmit Descriptor Word 1 (TDES1)

Bit	Description
31:29	Reserved
28:16	TBS2: Transmit Buffer 2 Size These bits indicate the second data buffer size in bytes. This field is not valid if TDES0[20] is set. See “ Buffer Size Calculations ” on page 58 for further detail on calculating buffer sizes.
15:13	Reserved
12:0	TBS1: Transmit Buffer 1 Size These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]).

Table 8-18 Transmit Descriptor 2 (TDES2)

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. See “ Host Data Buffer Alignment ” on page 57 for further detail on buffer address alignment.

Table 8-19 Transmit Descriptor 3 (TDES3)

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)

Table 8-20 Transmit Descriptor 6 (TDES6)

Bit	Description
31:0	TTSL: Transmit Frame Timestamp Low This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

Table 8-21 Transmit Descriptor 7 (TDES7)

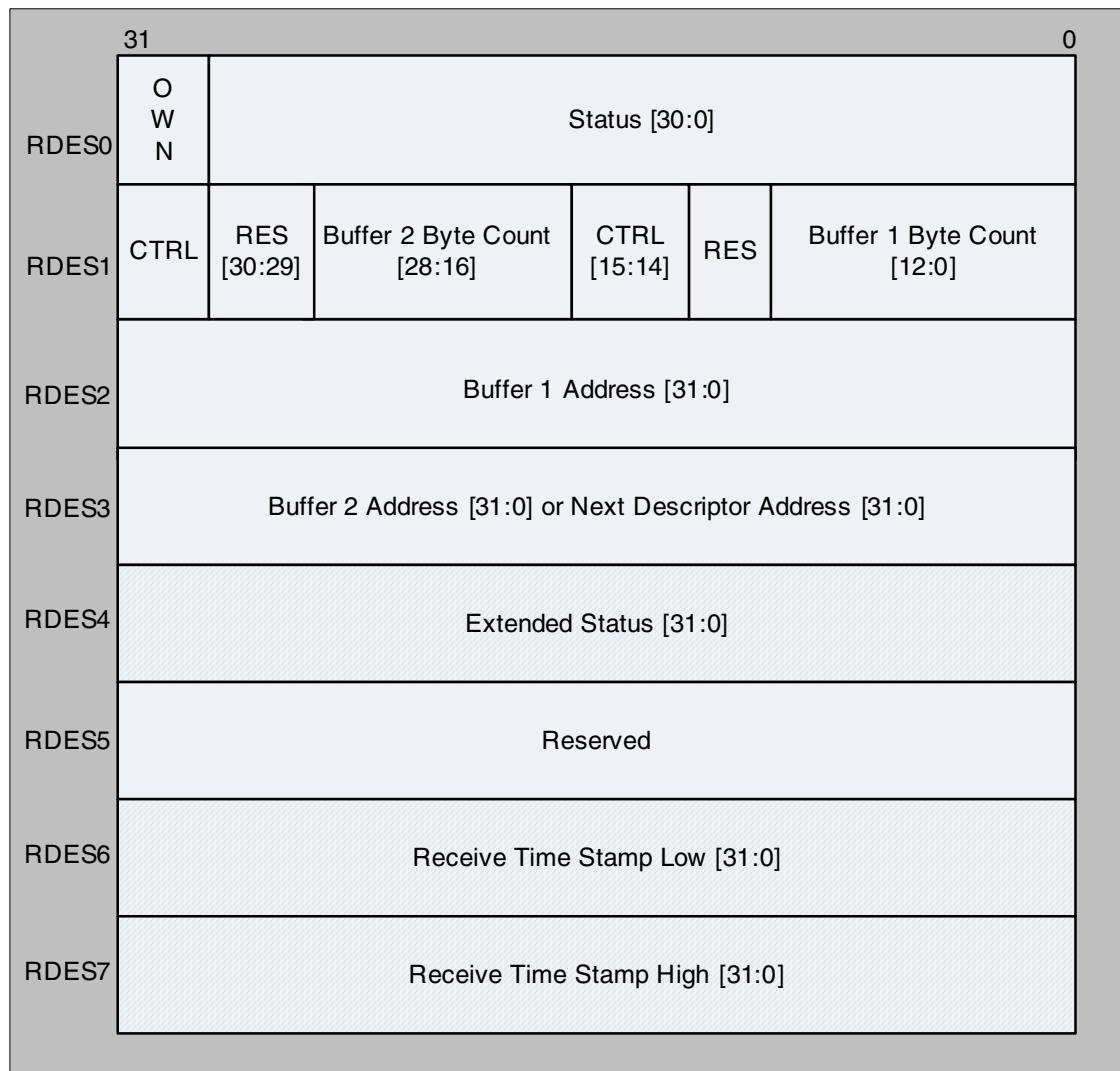
Bit	Description
31:0	TTSH: Transmit Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

8.2.2 Receive Descriptor

The structure of the received descriptor is shown in [Figure 8-17](#). This can have 32 bytes of descriptor data (8 DWORDs) when Advanced Timestamp or IPC Full Offload feature is selected.



Note When either of these features is enabled, the SW should set the DMA Bus Mode register[7] so that the DMA operates with extended descriptor size. When this control bit is reset, RDES0[7] and RDES0[0] is always cleared and the RDES4-RDES7 descriptor space are not valid.

Figure 8-17 Receive Descriptor Fields - Alternate (Enhanced) Format

The contents of RDES0 are identified in [Table 8-22](#). The contents of RDES1 through RDES3 are identified in [Tables 8-23](#) through [8-25](#), respectively.



Note Some of the bit functions of RDES0 are not backward compatible to Release 3.41a and previous versions. These bits are Bit[7], Bit[0], and Bit[5]. The function of Bit[5] is backward compatible to Release 3.30a and previous versions.

Table 8-22 Receive Descriptor Fields (RDES0)

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA of the GMAC Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.

Table 8-22 Receive Descriptor Fields (RDES0)

Bit	Description
30	AFM: Destination Address Filter Fail When set, this bit indicates a frame that failed in the DA Filter in the GMAC Core.
29:16	FL: Frame Length These bits indicate the byte length of the received frame that was transferred to host memory (including CRC). This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bits are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame. This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame.
15	ES: Error Summary Indicates the logical OR of the following bits: <ul style="list-style-type: none">• RDES0[1]: CRC Error• RDES0[3]: Receive Error• RDES0[4]: Watchdog Timeout• RDES0[6]: Late Collision• RDES0[7]: Giant Frame• RDES4[4:3]: IP Header/Payload Error• RDES0[11]: Overflow Error• RDES0[14]: Descriptor Error This field is valid only when the Last Descriptor (RDES0[8]) is set.
14	DE: Descriptor Error When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.
13	SAF: Source Address Filter Fail When set, this bit indicates that the SA field of frame failed the SA Filter in the GMAC Core.
12	LE: Length Error When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset.
11	OE: Overflow Error When set, this bit indicates that the received frame was damaged due to buffer overflow in MTL.
10	VLAN: VLAN Tag When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the GMAC Core.
9	FS: First Descriptor When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.

Table 8-22 Receive Descriptor Fields (RDES0)

Bit	Description
8	<p>LS: Last Descriptor</p> <p>When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame</p>
7	<p>Timestamp Available/IP Checksum Error (Type1) /Giant Frame</p> <p>When Advanced Timestamp feature is present, when set, this bit indicates that a snapshot of the Timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set.</p> <p>When IP Checksum Engine (Type 1) is selected, this bit, when set, indicates that the 16-bit IPv4 Header checksum calculated by the core did not match the received checksum bytes.</p> <p>Otherwise, this bit, when set, indicates the Giant Frame Status. Giant frames are larger-than-1,518-byte (or 1,522-byte for VLAN) normal frames and larger-than-9,018-byte (9,022-byte for VLAN) frame when Jumbo Frame processing is enabled.</p>
6	<p>LC: Late Collision</p> <p>When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.</p>
5	<p>FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes.</p>
4	<p>RWT: Receive Watchdog Timeout</p> <p>When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.</p>
3	<p>RE: Receive Error</p> <p>When set, this bit indicates that the gmii_rxer_i signal is asserted while gmii_rxrdv_i is asserted during frame reception. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error ($rxd \neq 0f$) during extension.</p>
2	<p>DE: Dribble Bit Error</p> <p>When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.</p>
1	<p>CE: CRC Error</p> <p>When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.</p>
0	<p>Extended Status Available/Rx MAC Address</p> <p>When either Advanced Timestamp or IP Checksum Offload (Type 2) is present, this bit, when set, indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set.</p> <p>When Advance Timestamp Feature or IPC Full Offload is not selected, this bit indicates Rx MAC Address status. When set, this bit indicates that the Rx MAC Address registers value (1 to 31) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field.</p>

Table 8-23 Receive Descriptor Fields 1 (RDES1)

Bit	Description
31	DIC: Disable Interrupt on Completion When set, this bit prevents setting the Status Register's RI bit (CSR5[6]) for the received frame ending in the buffer indicated by this descriptor. This, in turn, disables the assertion of the interrupt to Host due to RI for that frame.
30:29	Reserved
28:16	RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size, in bytes. The buffer size must be a multiple of 4, 8, or 16, depending on the bus widths (32, 64, or 128, respectively), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. If the buffer size is not an appropriate multiple of 4, 8, or 16, the resulting behavior is undefined. This field is not valid if RDES1[14] is set. See " Buffer Size Calculations " on page 58 for further details on calculating buffer sizes.
15	RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.
14	RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, RBS2 (RDES1[28:16]) is a "don't care" value. RDES1[15] takes precedence over RDES1[14].
13	Reserved
12:0	RBS1: Receive Buffer 1 Size Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4, 8, or 16, depending upon the bus widths (32, 64, or 128), even if the value of RDES2 (buffer1 address pointer) is not aligned. When the buffer size is not a multiple of 4, 8, or 16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 14). See " Buffer Size Calculations " on page 58 for further details on calculating buffer sizes.

Table 8-24 Receive Descriptor Fields 2 (RDES2)

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[3/2/1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3/2/1:0] (corresponding to bus width of 128/64/32) if the address pointer is to a buffer where the middle or last part of the frame is stored. See " Host Data Buffer Alignment " on page 57 for further details on buffer address alignment.

Table 8-25 Receive Descriptor Fields 3 (RDES3)

Bit	Description
31:0	<p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present.</p> <p>If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[3, 2, or 1:0] = 0, corresponding to a bus width of 128, 64, or 32. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3 [3, 2, or 1:0] (corresponding to a bus width of 128, 64, or 32) if the address pointer is to a buffer where the middle or last part of the frame is stored.</p>

The extended status written is as shown in [Table 8-26](#). The extended status is written only when there is status related to IPC or timestamp available. The availability of extended status is indicated by bit-0 of RDES0. This status is available only when Advance Timestamp or IPC Full Offload feature is selected.

Table 8-26 Receive Descriptor Fields 4 (RDES4)

Bit	Description
31:21	Reserved
20:18	<p>VLAN Tag Priority Value</p> <p>These bits give the VLAN tag's user value in the received packet. These bits are valid only when the RDES4 bits 16 and 17 are set.</p>
17	<p>AV Tagged Packet Received</p> <p>When set, this bit indicates that an AV tagged packet is received. Otherwise, this bit indicates that an untagged AV packet is received. This bit is valid when bit 16 (AV Packet Received) is set.</p>
16	<p>AV Packet Received</p> <p>When set, this bit indicates that an AV packet is received.</p>
15	Reserved
14	<p>Timestamp Dropped</p> <p>When set, this bit indicates that the timestamp was captured for this frame but got dropped in the MTL RxFIFO because of overflow. This bit is available only when you select the Advanced Timestamp feature. Otherwise, this bit is reserved.</p>
13	<p>PTP Version</p> <p>When set, this bit indicates that the received PTP message is having the IEEE 1588 version 2 format. When reset, it has the version 1 format. This is valid only if the message type is non-zero. This bit is available only if Advance Timestamp feature is selected else it is reserved.</p>
12	<p>PTP Frame Type</p> <p>When set, this bit indicates that the PTP message is sent directly over Ethernet. When this bit is not set and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information on IPv4 or IPv6 can be obtained from bits 6 and 7. This bit is available only if Advanced Timestamp feature is selected.</p>

Table 8-26 Receive Descriptor Fields 4 (RDES4)

Bit	Description
11:8	<p>Message Type These bits are encoded to give the type of the message received.</p> <ul style="list-style-type: none"> • 0000: No PTP message received • 0001: SYNC (all clock types) • 0010: Follow_Up (all clock types) • 0011: Delay_Req (all clock types) • 0100: Delay_Resp (all clock types) • 0101: Pdelay_Req (in peer-to-peer transparent clock) • 0110: Pdelay_Resp (in peer-to-peer transparent clock) • 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 1000: Announce • 1001: Management • 1010: Signaling • 1011-1110: Reserved • 1111 : PTP packet with Reserved message type <p>These bits are valid only when you select the Advance Timestamp feature.</p> <p>Note: Values 1000, 1001, and 1010 are not backward compatible with release 3.50a.</p>
7	<p>IPv6 Packet Received When set, this bit indicates that the received packet is an IPv6 packet.</p>
6	<p>IPv4 Packet Received When set, this bit indicates that the received packet is an IPv4 packet.</p>
5	<p>IP Checksum Bypassed When set, this bit indicates that the checksum offload engine is bypassed.</p>
4	<p>IP Payload Error When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the core calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field.</p>
3	<p>IP Header Error When set, this bit indicates either that the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or that the IP datagram version is not consistent with the Ethernet Type value.</p>
2:0	<p>IP Payload Type These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE). The COE also sets these bits to 2'b00 if it does not process the IP datagram's payload due to an IP header error or fragmented IP.</p> <ul style="list-style-type: none"> • 3'b000: Unknown or did not process IP payload • 3'b001: UDP • 3'b010: TCP • 3'b011: ICMP • 3'b1xx: Reserved

RDES6 and RDES7 contain the snapshot of the time-stamp. The availability of the snapshot of the time-stamp in RDES6 and RDES7 is indicated by bit-7 in the RDES0 descriptor. The contents of RDES6 and RDES7 are identified in [Table 8-27](#) and [Table 8-28](#), respectively.

Table 8-27 Receive Descriptor Fields 6 (RDES6)

Bit	Description
31:0	RTSL: Receive Frame Timestamp Low This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).

Table 8-28 Receive Descriptor Fields 7 (RDES7)

Bit	Description
31:0	RTSH: Receive Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).

The above description holds true for all descriptor formats depicted in [Figures 8-3, 8-5, 8-7, and 8-9](#). However, when the core operates in 64- or 128-bit and with a reverse-endian descriptor ([Figures 8-4, 8-6, 8-8, and 8-10](#)), RDES6 is updated with RTSH and RDES7 is updated with RTSL. This ensures that the 64-bit timestamp can be processed as-is without any swapping between the two 32-bit words.

Implementation Guidelines

This chapter provides information about the clocking schemes for various GMAC-UNIV interfaces. It contains the following sections:

- ❖ “Clocks With GMII/MII” on page 429
- ❖ “Clocks With TBI” on page 431
- ❖ “Clocks With SGMII” on page 431
- ❖ “Clocks With RMII” on page 433
- ❖ “Clocks With RevMII” on page 434
- ❖ “Clocks With SMII” on page 436
- ❖ “Clocks With RGMII” on page 438
- ❖ “Clocks With RTBI” on page 439
- ❖ “Host (System Interface) Clock” on page 440
- ❖ “Resets” on page 441
- ❖ “SMA to PHY Interface” on page 441
- ❖ “Timing Guidelines for Dual Data Rate RGMII/RTBI Outputs” on page 442
- ❖ “Synthesis Guidelines” on page 448

9.1 Clocks With GMII/MII

The clocking scheme for the GMAC-AHB is shown in [Figure 9-1](#). The GMAC-AHB uses three clock inputs for normal operation with the GMII/MII interface:

- ❖ One clock (`clk_tx_i`) for transmission functions
- ❖ One clock (`clk_rx_i`) for reception functions
- ❖ One application clock (`hclk_i`)

All clocks except the Application clock are generated from an external PHY. The Application clock is used for the control interface of the GMAC-AHB. Transmit clock `clk_tx_i` is used for the transmission function of the GMAC-AHB. Similarly, receive clock `clk_rx_i` is used for the receive function of the GMAC-AHB. The transfer of data from/to the application clock domain to the Transmit and Receive clock domains is performed in the MTL module.

Clock `clk_tx_i` switches between the MII transmit clock input from an external PHY (MII mode) and the 125-MHz clock input from an oscillator (GMII mode), based on the `mac_portselect_o` signal from the GMAC-AHB. The external PHY outputs a 25-MHz or 2.5-MHz transmit clock on `MII_CLK` when it operates at 100 Mbps or 10 Mbps, respectively.

The frequency of clock `clk_rx_i` is same as receive clock `GMII/MII_RX_CLK` generated by the external PHY. The PHY outputs a 125-MHz, 25-MHz, or 2.5-MHz receive clock on `GMII/MII_RX_CLK` when it operates at 1000 Mbps, 100 Mbps, or 10 Mbps, respectively.



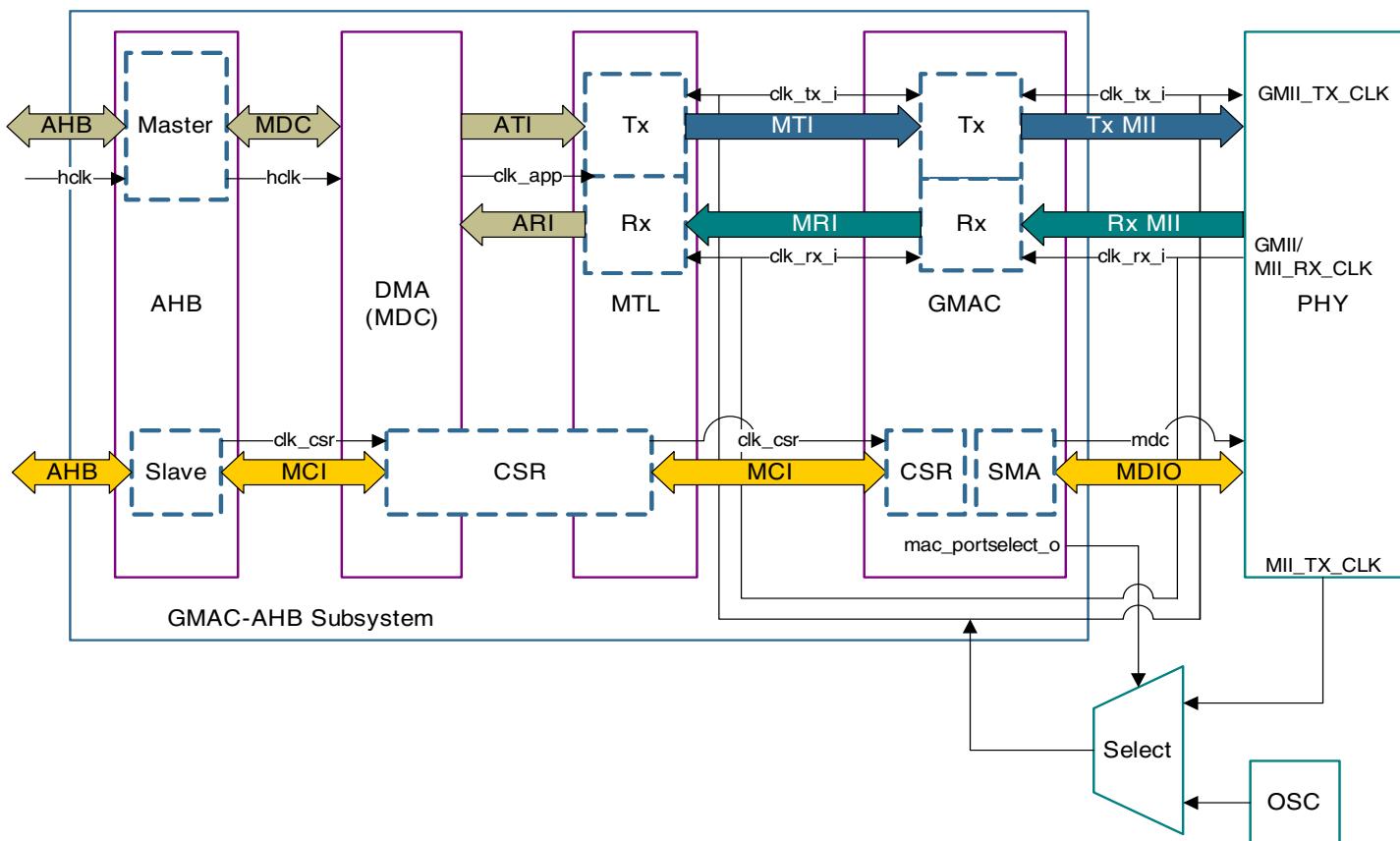
Caution

In all PHY interfaces explained in [Sections 9.1 to 9.8](#) (except MII, SGMII, SMII, and RMII), your chip containing the MAC core must drive the Transmit clock towards the PHY. All the transmit data and control signals output from the core must be timed with respect to this output clock, as specified in the respective PHY interface specifications (See the IEEE 802.3, SGMII, RMII, and RGMII/RTBI specifications).

For MII and SGMII, the transmit data and control signals output from the core must be timed with respect to either the transmit clock input from the PHY chip (MII) or the SerDes (SGMII).

For RMII/SMII, you can either connect the Oscillator clock directly to both the MAC and PHY or drive an RMII/SMII clock towards the PHY, according to the recommendations in the RMII/SMII specification.

Figure 9-1 GMAC-AHB Clocking Scheme



9.2 Clocks With TBI

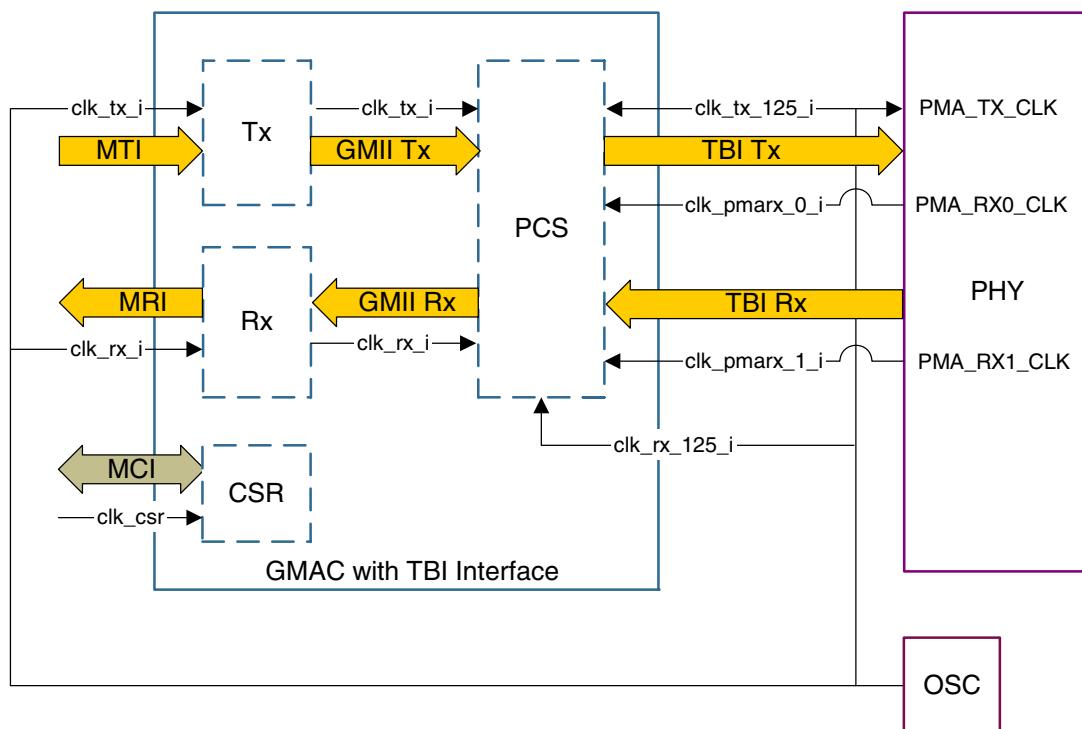
The clocking scheme used for the GMAC-CORE in TBI mode is shown in [Figure 9-2](#). The clocking strategy is the same for other GMAC-UNIV configurations such as GMAC-AHB, GMAC-MTL, etc. The GMAC-CORE uses four clock inputs for normal operation:

- ❖ One clock (`clk_tx_i`) for transmission functions
- ❖ Two clocks (`clk_pmarx0_i` and `clk_pmarx1_i`) for reception functions
- ❖ One application clock (`clk_csr_i`)

Note that you should connect the same `clk_tx_i` to the `clk_tx_125_i`, `clk_rx_i`, and `clk_rx_125_i` ports of the GMAC core in TBI mode. The `clk_tx_125_180_i` port should be connected to the invert of `clk_tx_i`.

The GMAC-CORE outputs the transmit code group to the external PHY synchronous to the `clk_tx_i` (125-MHz clock). The GMAC-CORE internally transfers the receive code group from the `clk_pmarx_x_i` clock domain to the `clk_rx_i` (same as `clk_tx_i`) clock domain.

Figure 9-2 GMAC-CORE Clocking Scheme (TBI Mode)



9.3 Clocks With SGMII

The clocking scheme used for the GMAC-CORE in SGMII mode is shown in [Figure 9-3](#). The clocking strategy is the same for other GMAC-UNIV configurations such as GMAC-AHB, GMAC-MTL, etc. The GMAC-CORE uses five clock inputs for normal operation:

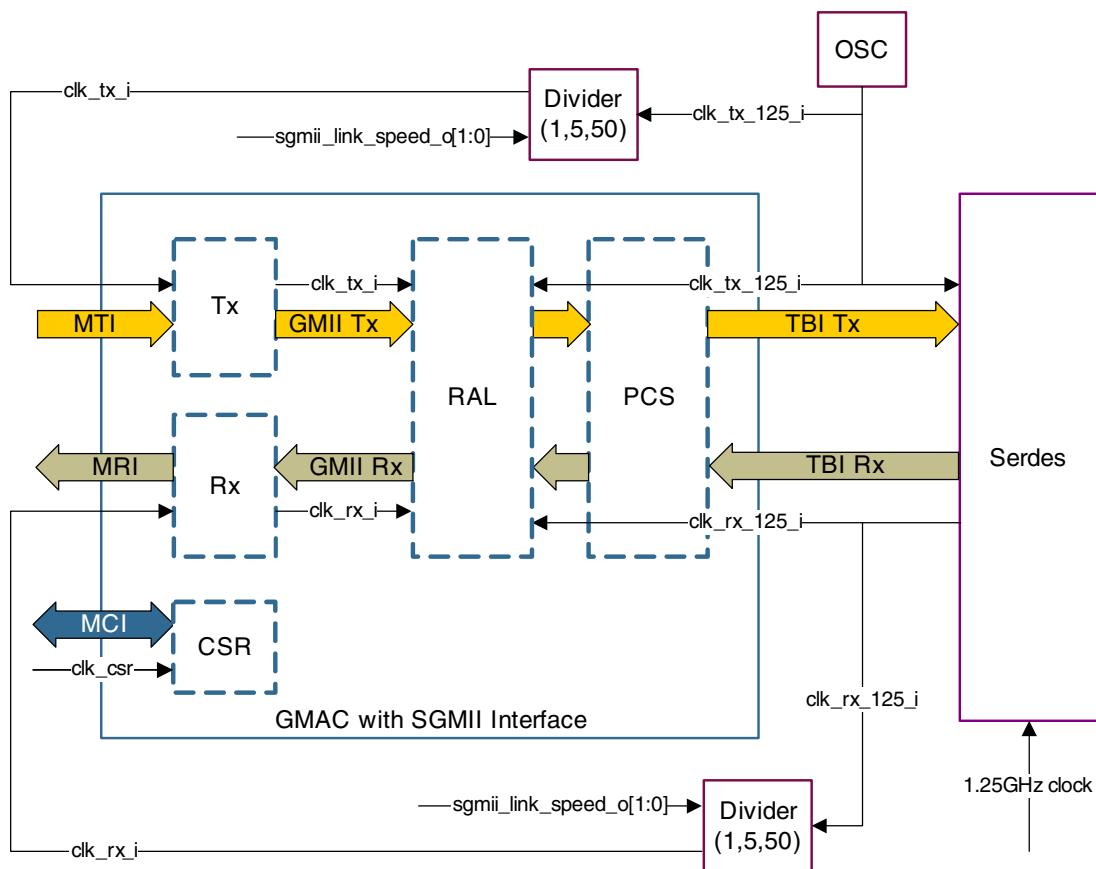
- ❖ One clock (`clk_tx_125_i`) for transmission functions with the RAL, PCS, and SerDes (Note that the `clk_tx_125_180_i` port should be connected to the invert of `clk_tx_125_i`.)
- ❖ One clock (`clk_rx_125_i`) for reception functions with the RAL, PCS, and SerDes

- ❖ One clock (clk_tx_i) for DWC Ether MAC 10/100/1000 Universal transmission functions
- ❖ One clock (clk_rx_i) for DWC Ether MAC 10/100/1000 Universal reception functions
- ❖ One application clock (clk_csr_i)

The GMAC-CORE outputs the transmit code group to the RAL synchronous to clk_tx_i. Similarly, the GMAC-CORE inputs the receive code group from the RAL, synchronous to clk_rx_i. The PCS layer runs on a 125-MHz clock while the GMII interface may run on a 2.5-MHz, 25-MHz, or 125-MHz clock. Therefore the clk_tx_i clock is derived from clock clk_tx_125_i, with a frequency either equal to the frequency of clk_tx_125_i, one fifth the frequency, or one fiftieth the frequency of clk_tx_125_i, as determined by signal sgmii_link_speed_o. (Refer to “[Serial Gigabit Media Independent Interface](#)” on page 195 for a functional description of the SGMII interface.) Similarly, clk_rx_i is derived from clk_rx_125_i by a clock divider circuit that changes the frequency with the link speed. You can also choose the output signal mac_speed_o (instead of sgmii_link_speed_o) as the control signal for the Transmit clock divider according to your system requirements.

Because clk_tx_i and clk_rx_i are derived synchronously (divided clock) from the corresponding clk_tx_125_i and clk_rx_125_i, it is easier to meet the static timing of signals crossing the clock domains if the derived clock’s rising edge is synchronous to the master clocks’ falling edge. Hence, the timing budget for such signals is half the clock width of the master clock.

If the timing delays are not met, the clock phases between the master and derived clock can be shifted, with the signals in the clk_tx_i domain sampled in the clk_tx_125_i domain in the transmit path and the signals in clk_rx_125_i are sampled in the clk_rx_i domain. Thus, clk_tx_i must lead clk_tx_125_i, while clk_rx_i must lag clk_rx_125_i.

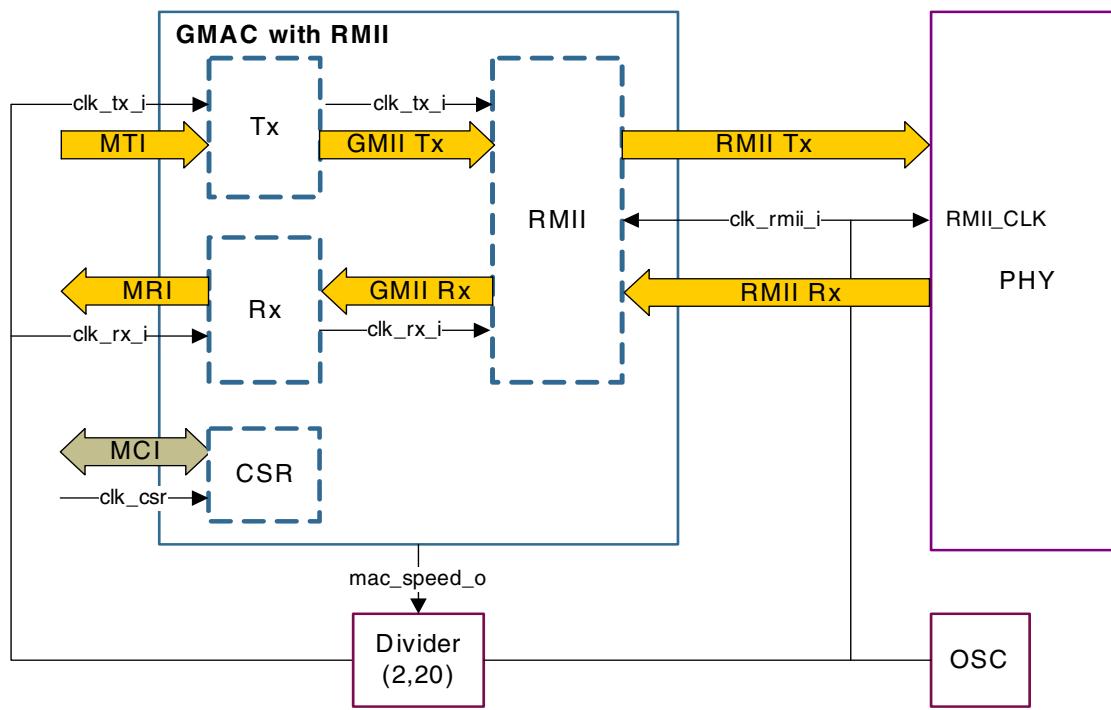
Figure 9-3 GMAC Clocking Scheme (SGMII Mode)

9.4 Clocks With RMII

The clocking scheme used for the GMAC-CORE in RMII mode is shown in [Figure 9-4](#). The clocking strategy is the same for other GMAC-UNIV configurations, such as GMAC-AHB, GMAC-MTL, and so forth. The GMAC-CORE uses three clock inputs for normal operation:

- ❖ One clock (clk_rx_i) for transmission and reception functions
- ❖ One clock (clk_rmii_i) for RMII
- ❖ One application clock (clk_csr_i)

You should connect the same clk_rx_i to the clk_tx_i ports of the GMAC core in RMII mode. The clk_rx_i should be synchronously derived from the clk_rmii_i, by dividing it by 20 for 10-Mbps operation or by 2 for 100-Mbps operation. You can use either the mac_speed_i or mac_speed_o[0] signals as the control signal to the clock divider.

Figure 9-4 Clocking Scheme (RMII Mode)

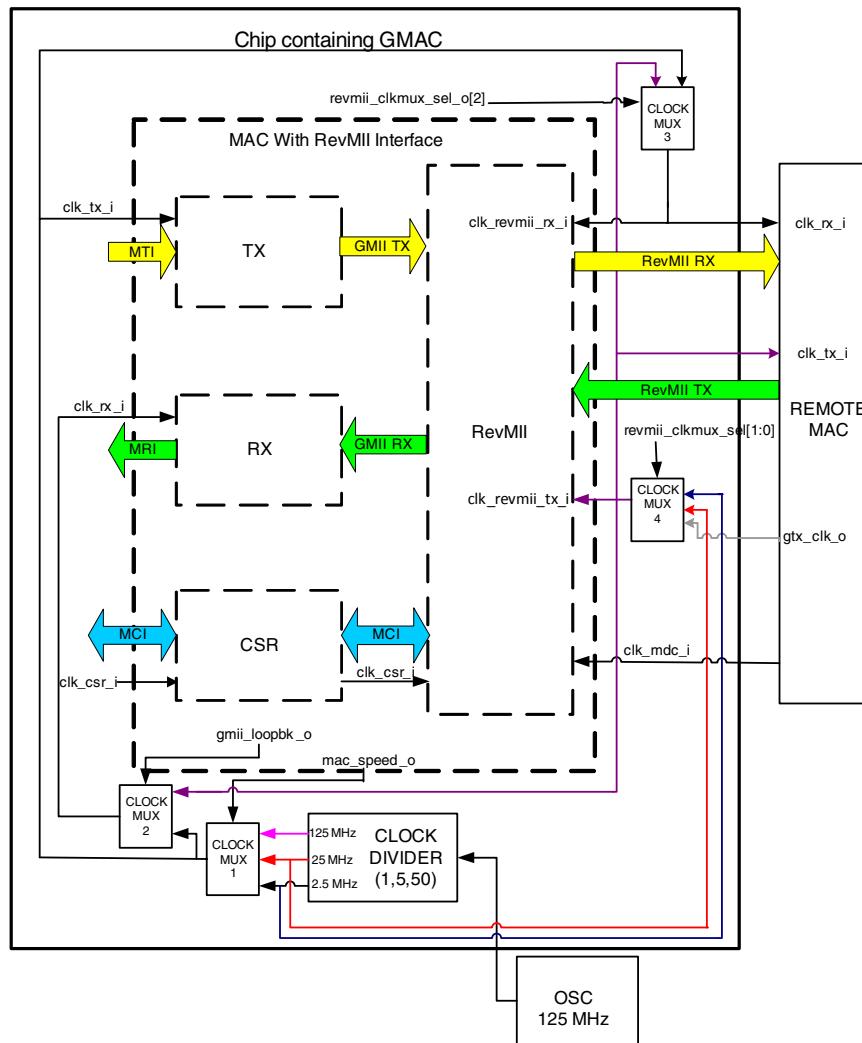
Because `clk_tx_i` (which is the same as `clk_rx_i`) is synchronously derived (divided) from the corresponding `clk_rmii_i`, it is easier to meet the static timing of signals crossing the clock domains if the derived clock's rising edge is synchronous to the RMII clock's falling edge. You therefore get half the RMII clock's clock width as the timing budget for such signals.

If the timing delays are not met, the clock phases between the master and derived clock can be shifted (during clock tree synthesis). Because the signals in the `clk_tx_i` domain are sampled in the `clk_rmii_i` domain in the transmit path, while the signals in `clk_rmii_i` are sampled in the `clk_rx_i` domain, `clk_tx_i` must lead `clk_rmii_i` and `clk_rx_i` must lag `clk_rmii_i`.

9.5 Clocks With RevMII

Figure 9-5 shows the clocking scheme used for all GMAC-UNIV configurations in the RevMII mode. The GMAC-UNIV configurations use the following six inputs for normal operation:

- ❖ One clock (`clk_tx_i`) for transmission functions.
- ❖ One clock (`clk_rx_i`) for reception functions.
- ❖ One application clock (`clk_csr_i`).
- ❖ One clock (`clk_revmii_tx_i`) for RevMII remote MAC transmission.
- ❖ One clock (`clk_revmii_rx_i`) for RevMII remote MAC reception.
- ❖ One clock (`revmii_mdc_i`) for serial MDIO operation.

Figure 9-5 Clocking Scheme (RevMII Mode)

The output of an oscillator that runs at 125 MHz is divided to generate the following three clocks:

- ❖ 125 MHz for 1000 Mbps (not required when MAC is configured for 10/100 Mbps only)
- ❖ 25 MHz for 100 Mbps
- ❖ 2.5 MHz for 10 Mbps

A clock MUX (clock MUX 1 and clock MUX 4 in Figure 9-5) selects a proper clock out of these three divided clocks. The clock MUX selects a proper clock by using the Speed Selection bits present in the Control Registers of the MAC and remote MAC. The Speed Selection bits represent the speed supported by a MAC. If the supported speed is same for both MACs, then this speed becomes the speed of operation. Otherwise, the Link Status bit goes low, indicating that the link is down.

The clock MUX 2 and clock MUX3 controls the clock connected to the clk_rx_i and clk_revmii_rx_i inputs. During normal operation, clk_rx_i gets the same clock as clk_revmii_tx_i. Similarly, clk_revmii_rx_i gets the same clock as clk_tx_i.

When the MAC enables the loopback mode (bit 14) in [Register 0 \(RevMII PHY Control Register\)](#), then the clk_tx_i gets connected to clk_rx_i. Similarly, when the remote MAC enables the loopback mode, the clk_revmii_tx_i gets connected to clk_revmii_rx_i.

In [Figure 9-5](#), clk_tx_i/clk_rx_i for remote MAC are same as clk_revmii_tx_i/clk_revmii_rx_i. [Table 9-1](#) provides more information about the clocking scheme in 10, 100, and 1000 Mbps mode.

Table 9-1 RevMII Clocking Scheme

Mode	Description
10 Mbps or 100 Mbps Mode	<ul style="list-style-type: none"> The clk_tx_i/clk_rx_i clocks of the MAC and remote MAC are same.
1000 Mbps Normal Mode	<ul style="list-style-type: none"> The clk_rx_i of the MAC is same as the clk_revmii_tx_i clock of the remote MAC (selected by clock MUX 2 in Figure 9-5). The clk_revmii_rx_i clock of the remote MAC is same as the clk_tx_i clock of the MAC (selected by clock MUX 3 in Figure 9-5).
1000 Mbps Loopback Mode	<ul style="list-style-type: none"> The clk_rx_i of the MAC is same as the clk_tx_i of the MAC (selected by clock MUX 2 in Figure 9-5). The clk_revmii_rx_i clock of the remote MAC is same as the clk_revmii_tx_i clock of the remote MAC (selected by clock MUX 3 in Figure 9-5).

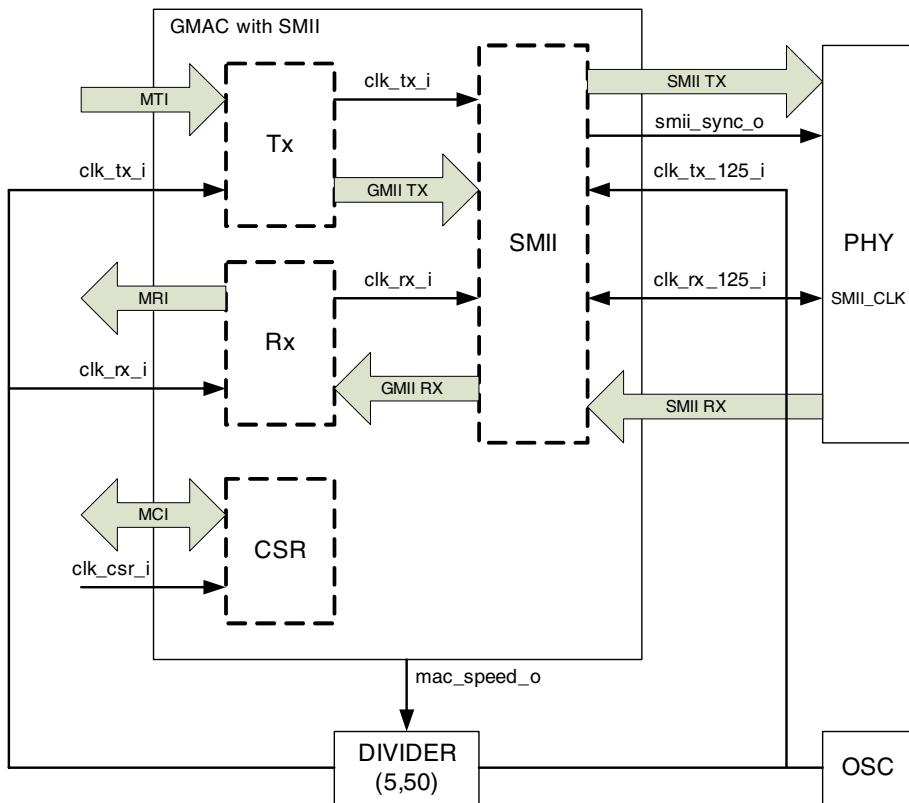
9.6 Clocks With SMII

9.6.1 Non Source Synchronous Mode

The clocking scheme used for the GMAC-CORE in SMII Non Source Synchronous mode is shown in [Figure 9-6](#). The clocking strategy is the same for other GMAC-UNIV configurations, such as GMAC-AHB, GMAC-MTL, and so forth. The GMAC-CORE uses three clock inputs for normal operation:

- ❖ One clock (clk_tx_i) for transmission and reception functions
- ❖ One clock (clk_tx_125_i) for SMII
- ❖ One application clock (clk_csr_i)

You should connect the same clk_tx_i to the clk_rx_i ports of the GMAC core in SMII mode. The clk_tx_i should be synchronously derived from the clk_tx_125_i, by dividing it by 50 for 10 Mbps operation or by 5 for 100 Mbps operation. You can use either the mac_speed_i or mac_speed_o[0] signals as the control signal to the clock divider. The global sync signal smii_sync_o is provided by the SMII block.

Figure 9-6 Non Source Synchronous Mode Clocking Scheme

Because clk_{tx_i} (which is the same as clk_{rx_i}) is synchronously derived (divided) from the corresponding $\text{clk}_{\text{tx}_\text{125}_i}$, it is easier to meet the static timing of signals crossing the clock domains if the derived clock's rising edge is synchronous to the SMII clock's falling edge. You therefore get half the SMII clock's clock width as the timing budget for such signals.

If the timing delays are not met, the clock phases between the master and derived clock can be shifted (during clock tree synthesis). Because the signals in the clk_{tx_i} domain are sampled in the $\text{clk}_{\text{tx}_\text{125}_i}$ domain in the transmit path, while the signals in $\text{clk}_{\text{rx}_\text{125}_i}$ are sampled in the clk_{rx_i} domain, clk_{tx_i} must lead $\text{clk}_{\text{tx}_\text{125}_i}$ and clk_{rx_i} must lag $\text{clk}_{\text{rx}_\text{125}_i}$.

9.6.2 Source Synchronous Mode

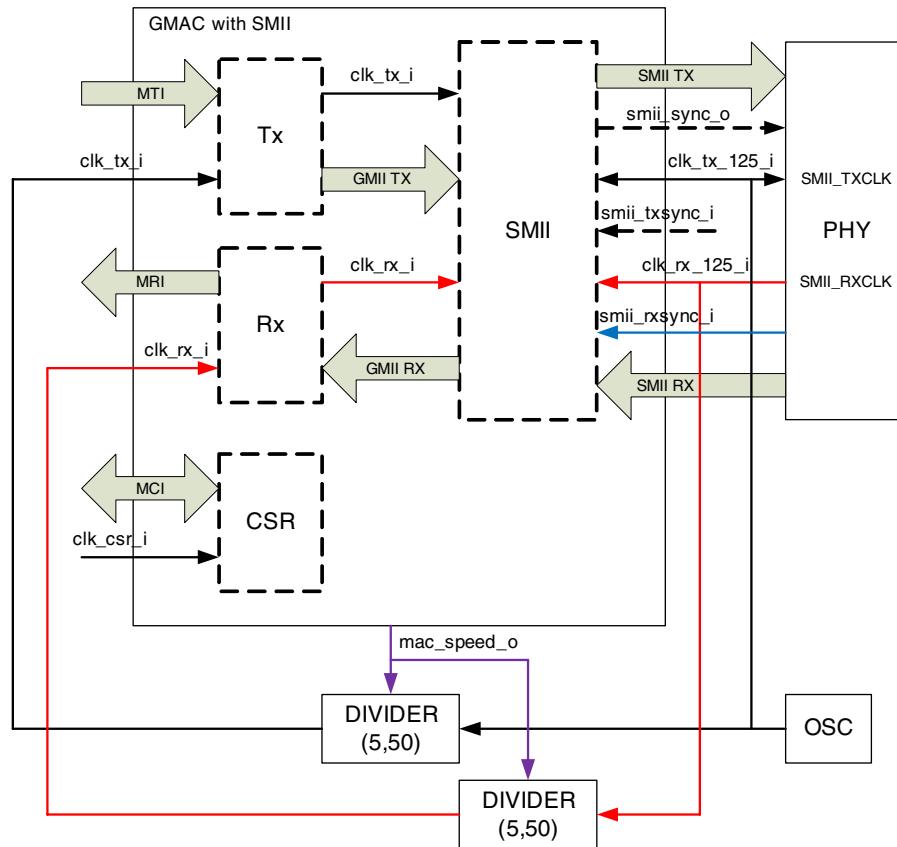
The clocking scheme used for the GMAC-CORE in SMII Source Synchronous mode is shown in [Figure 9-7](#). The clocking strategy is the same for other GMAC-UNIV configurations, such as GMAC-AHB, GMAC-MTL, and so forth. The GMAC-CORE uses five clock inputs for normal operation:

- ❖ One clock (clk_{tx_i}) for transmission function
- ❖ One clock (clk_{rx_i}) for reception function
- ❖ One clock ($\text{clk}_{\text{rx}_\text{125}_i}$) for SMII receive path
- ❖ One clock ($\text{clk}_{\text{tx}_\text{125}_i}$) for SMII transmit path
- ❖ One application clock ($\text{clk}_{\text{csr}_i}$)

The clk_{rx_i} should be synchronously derived from the $\text{clk}_{\text{rx}_\text{125}_i}$, by dividing it by 50 for 10 Mbps operation or by 5 for 100 Mbps operation. Similarly, the clk_{tx_i} should be synchronously derived from the $\text{clk}_{\text{tx}_\text{125}_i}$, by dividing it by 50 for 10 Mbps operation or by 5 for 100 Mbps operation. You can use either

the mac_speed_i or mac_speed_o[0] signals as the control signal to the clock dividers. The TXSYNC signal is provided by the SMII block if SSSMII_TXSYNC_IN configuration parameter is not selected. However, if SSSMII_TXSYNC_IN configuration parameter is selected, the SMII transmit block transmits synchronized to the smii_txsync_i input. The smii_rxsync_i needs to be driven by the transmitting source.

Figure 9-7 Source Synchronous Mode Clocking Scheme



Because clk_tx_i (and clk_rx_i) is synchronously derived (divided) from the corresponding clk_tx_125_i (or clk_rx_125_i), it is easier to meet the static timing of signals crossing the clock domains if the derived clock's rising edge is synchronous to the SMII clock's falling edge. You therefore get half the SMII clock's clock width as the timing budget for such signals.

If the timing delays are not met, the clock phases between the master and derived clock can be shifted (during clock tree synthesis). Because the signals in the clk_tx_i domain are sampled in the clk_tx_125_i domain in the transmit path, while the signals in clk_rx_125_i are sampled in the clk_rx_i domain, clk_tx_i must lead clk_tx_125_i and clk_rx_i must lag clk_rx_125_i.

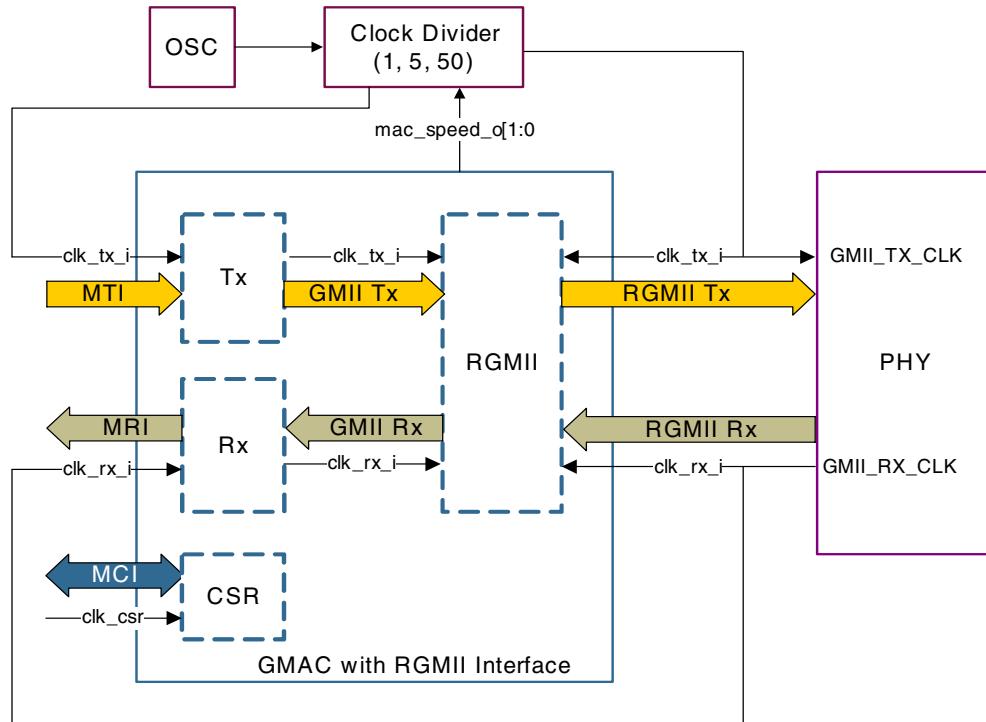
9.7 Clocks With RGMII

The clocking scheme used for the GMAC-CORE in RGMII mode is shown in Figure 9-8. The clocking strategy is the same for other GMAC-UNIV configurations such as GMAC-AHB, GMAC-MTL, etc. The GMAC-CORE uses three clock inputs for normal operation:

- ❖ One clock (clk_tx_i) for transmission function. The clk_tx_180_i port should be connected to the invert of clk_tx_i in RGMII mode.

- ❖ One clock (`clk_rx_i`) for reception function. The `clk_rx_180_i` port should be connected to the invert of `clk_rx_i`.
- ❖ One application clock (`clk_csr_i`)

Figure 9-8 Clocking Scheme (RGMII Mode)



The RGMII specifications state that the transmit clock (`clk_tx_i`) frequency must be 2.5, 25, or 125 MHz, respectively, for 10-, 100-, or 1000-Mbps operation. Hence the 125-MHz oscillator must be divided dynamically based on the operating speed. The signal output from the GMAC-UNIV (`mac_speed_o[1:0]`) controls the clock divider as follows:

<code>mac_speed_o[1,0]</code>	Clock Divider
2'b0x	Divide by 1
2'b11	Divide by 5
2'b10	Divide by 50

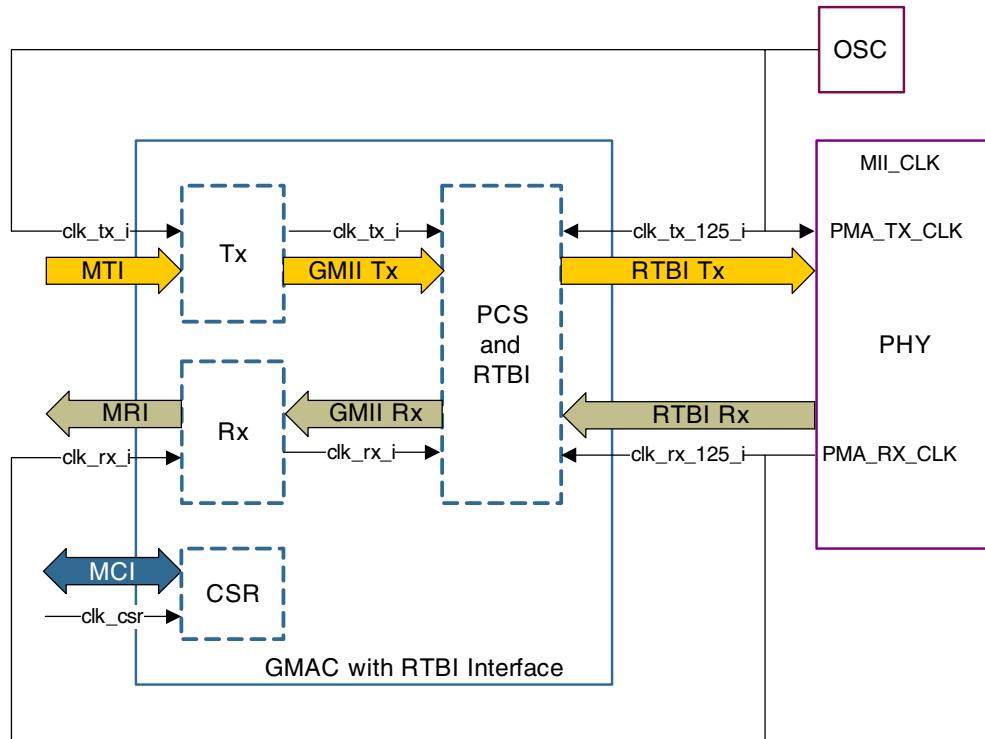
9.8 Clocks With RTBI

The clocking scheme used for the GMAC-CORE in RTBI mode is shown in Figure 9-9. The clocking strategy is the same for other GMAC-UNIV configurations such as GMAC-AHB, GMAC-MTL, and so on. The GMAC-CORE uses three clock inputs for normal operation:

- ❖ One clock (`clk_tx_i`) for transmission function. The `clk_tx_125_180_i` port should be connected to the inverse of `clk_tx_i` in RTBI mode, while the `clk_tx_125_i` port should be the same as `clk_tx_i`.
- ❖ One clock (`clk_rx_i`) for reception function. The `clk_rx_180_i` port should be connected to the inverse of `clk_rx_i`, while the `clk_rx_125_i` port should be the same as `clk_rx_i`.

- ❖ One application clock (clk_csr_i)

Figure 9-9 Clocking Scheme (RTBI Mode)



9.9 Host (System Interface) Clock

The GMAC-UNIV has different clock names for different configuration as listed below for the transfer of data to and from the GMAC to the host. Even though the names are different, the required properties and frequency ranges for all the clocks are same. In addition to this, you have an option to have a separate clock (clk_csr_i) for accessing the CSR through the slave port.

GMAC-DMA, GMAC-MTL : clk_app_i

GMAC-AHB: hclk_i

GMAC-AXI: aclk_i

The minimum and maximum clock frequency are specified between the range 20MHz-300 MHz for the host clock. This is not due to any functional limitation but due to the fact that we verify it & synthesize the design for this range. You can go for a higher frequency as your technology permits.

Another factor to keep in mind is that the host clock (or clk_csr_i when present) is used to generate the MDC clock for the SMA interface. Due to the requirement of IEEE 802.3 that the MDC clock should have a maximum frequency of 2.5 MHz, your clock frequency may get limited by the clock-divider options provided. But if your system (or PHY chip) permits a higher frequency for MDC clock than specified by 802.3, then you can ignore this constraint for the maximum frequency possible.

When the RMON (MMC) counters are present and the core is operating in gigabit speed, then the CSR clock should have a minimum frequency of 25 MHz for proper collection of statistics of all consecutive small-sized (or runt) frames.



Note You can dynamically change the frequency of the host clock (provided it is glitch-free) when the GMAC is active (without any software intervention) to save power consumption of your chip. This has been tested in the regression run in our simulation environment between the following frequency range: 25Mhz -300 MHz.

9.10 Resets

When the GMAC-UNIV is configured for DMA (with or without AHB/ AXI interface), internal reset generation logic is also automatically instantiated. This reset logic takes in the hreset_n for GMAC-AHB (pwr_on_rst_n for GMAC-DMA or areset_n in GMAC-AXI) input signal, and generates resets for all other clock domains internally. It also manages reset assertion when the host triggers the SWR bit of DMA Register 0. Internal reset generation (for other clock domains) can be bypassed by setting the test_mode signal input high. In this mode, the input reset (hreset_n, areset_n, or pwr_on_rst_n) is directly used in all clock domains. This mode is normally used during scan tests only.

For GMAC-CORE and GMAC-MTL configurations, no reset-generation logic is present in the core. you must therefore input separate resets corresponding to each clock input. These resets must always be de-asserted synchronously with their corresponding clock edges. For proper core initialization, all reset inputs must be asserted together for at least 3 cycles of the minimum-frequency clock. Otherwise, the clock domain transfer logic is not properly initialized, with possible corruption of the first frame after such a reset.

9.11 SMA to PHY Interface

As defined in IEEE 802.3, Clause 22, the MAC station management interface to the external PHY consists of two signals:

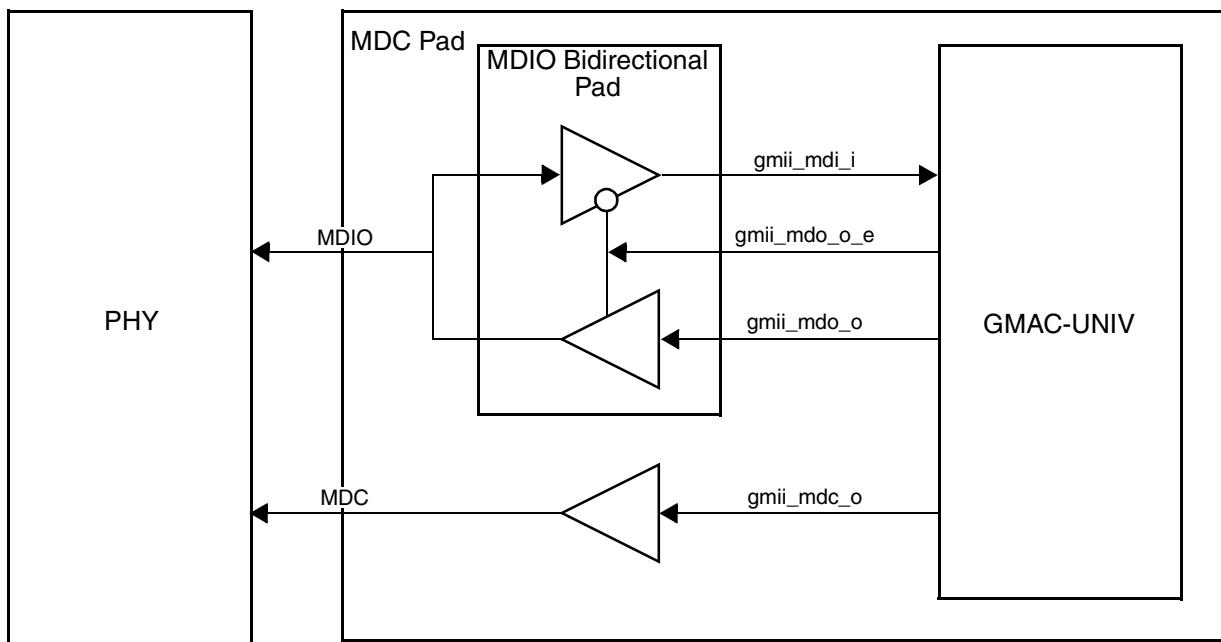
- ❖ MDIO: Bidirectional data
- ❖ MDC: Clock input to the PHYs from the MAC

These signals are detailed in IEEE 802.3, Clause 22.

The GMAC-UNIV core supports this interface with the following four signals (to be connected to the ASIC's MDIO and MDC pads):

- ❖ gmac_mdi_i: input MDIO data signal (from the PHY to the MAC)
- ❖ gmac_mdo_o: output MDIO data signal (from the MAC to the PHY)
- ❖ gmac_mdo_e: a control signal to the bidirectional MDIO pad, which signal controls the pad's input/output direction. When asserted, this signal indicates that the MDIO is in Output mode.
- ❖ gmac_mdc_o: output MDC clock signal (from the MAC to PHY).

A connectivity example is shown in [Figure 9-10](#)

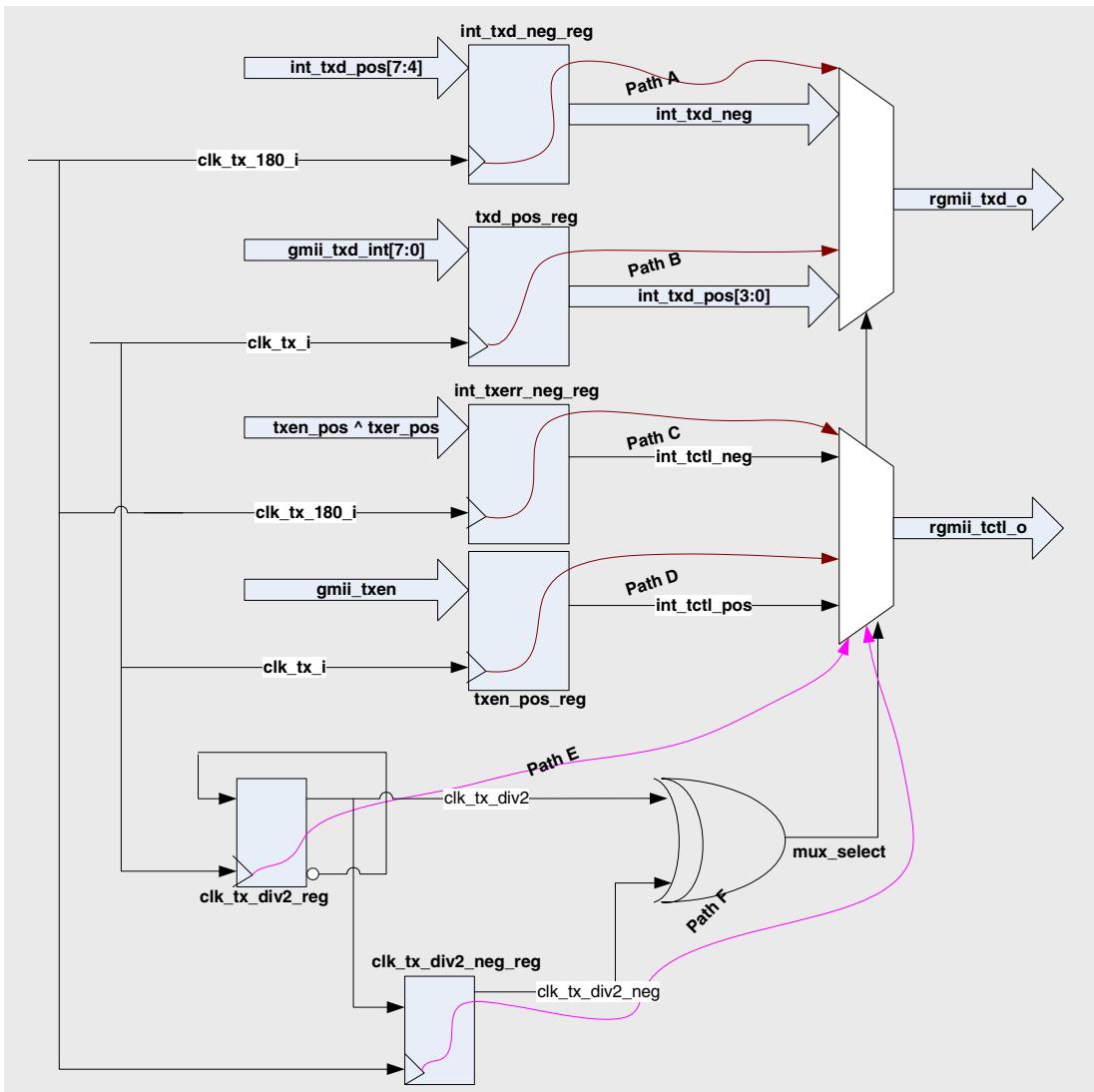
Figure 9-10 SMA to PHY Connectivity

9.12 Timing Guidelines for Dual Data Rate RGMII/RTBI Outputs



Note Though this section refers only to the RGMII PHY, it also applies when the core is RTBI-PHY configured.

When the RGMII PHY interface is configured, the data and control signals are valid for both clock edges. The transmit output data `rgmii_txd_o` and control `rgmii_tctl_o` can transition at both clock edges of `clk_tx_i`. This logic is implemented in GMAC using two clocks (`clk_tx_i` and `clk_tx_180_i`, which is 180° out of phase with `clk_tx_i`) as shown in [Figure 9-11](#).

Figure 9-11 Dual Data Rate Output Paths

This logic can be found in the DWC_gmac_rgmii_gmrt.v file in the <config>/src/rgmii/ directory. Paths A-D in [Figure 9-11](#) are considered the data path delays (clock-to-output delay for the registers, plus the net delay from the register output to the multiplexer's input). Similarly, paths E and F are considered the control path delay for the MUX Select signal. To ensure that no glitches are generated in the output data, the data signals at the input of the MUX (paths A-D) must be stable before the MUX select signal (paths E-F) is asserted. The control path E-F timing delays must be greater than the data path delays. The following paragraphs give a guideline on how to ensure such delays during implementation.

The data path timing report in [Example 9-1](#) shows the total delay for the data path to be 0.57 ns. This timing report can be obtained by giving the following command in DC shell after completion of synthesis.

```
report_timing -from DWC_gmac_inst/DWC_gmac_rgmii_inst/DWC_gmac_rgmii_gmrt_inst/
txd_pos_reg*/*CK
```



- Check your technology library for the exact clock port name.
- You can obtain a similar timing report for int_txd_neg_reg.

Example 9-1 Data Path Timing Report

Startpoint: DWC_gmac_inst/DWC_gmac_rgmii_inst/DWC_gmac_rgmii_gmrt_inst/txd_pos_reg[3]
(rising edge-triggered flip-flop clocked by clk_tx_i)

Endpoint: phy_txd_o[3] (output port clocked by clk_tx_180_i)

Path Group: clk_tx_i_outputs

Path Type: max

Des/Clust/PortWire Load ModelLibrary

DWC gmac toptsmc090 wl10slow

Point	Incr	Path
clock clk_tx_i (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
DWC_gmac_inst/DWC_gmac rgmii_inst/DWC_gmac rgmii_gmrt_inst/txd_pos_reg[3]/CK (DFFRQX2)	0.00	0.00 r
DWC_gmac_inst/DWC_gmac rgmii_inst/DWC_gmac rgmii_gmrt_inst/txd_pos_reg[3]/Q (DFFRQX2)	0.36	0.36 f
DWC_gmac_inst/DWC_gmac rgmii_inst/DWC_gmac rgmii_gmrt_inst/U34/Y (MX2X1)	0.21	0.57 f
DWC_gmac_inst/DWC_gmac rgmii_inst/DWC_gmac rgmii_gmrt_inst/rgmii_txd_o[3] (DWC_gmac rgmii)	0.00	0.57 f
DWC_gmac_inst/DWC_gmac rgmii_inst/rgmii_txd_o[3] (DWC_gmac rgmii)	0.00	0.57 f
DWC_gmac_inst/DWC_gmac phy_tmxmux_inst/rgmii_txd_i[3] (DWC_gmac phy_tmxmux)	0.00	0.57 f
DWC_gmac_inst/DWC_gmac phy_tmxmux_inst/U9/Y (MX2X1)	0.20	0.76 f
DWC_gmac_inst/DWC_gmac phy_tmxmux_inst/phy_txd_o[3] (DWC_gmac phy_tmxmux)	0.00	0.76 f
DWC_gmac_inst/phy_txd_o[3] (DWC_gmac)	0.00	0.76 f
phy_txd_o[3] (out)	0.00	0.76 f
data arrival time	0.76	
clock clk_tx_180_i (rise edge)	4.00	4.00
clock network delay (ideal)	0.00	4.00
output external delay	-1.60	2.40
data required time	2.40	

data required time	2.40	
data arrival time	-0.76	

slack (MET)	1.64	

The Multiplexer Select signal shows a 1.08-ns path delay. The timing report below can be obtained by giving the following command in DC shell.

```
report_timing -from DWC_gmac_inst/DWC_gmac_rgmii_inst/DWC_gmac_rgmii_gmrt_inst/  
clk tx_div2 req*/CK
```



You can obtain a similar report for clk_tx_div2_neg_req.

Example 9-2 Multiplexer Select Signal Timing Report

```

Startpoint: DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/clk_tx_div2_reg
(rising edge-triggered flip-flop clocked by clk_tx_i)
Endpoint: phy_txd_o[0] (output port clocked by clk_tx_180_i)
Path Group: clk_tx_i_outputs
Path Type: max

Des/Clust/PortWire Load ModelLibrary
-----
DWC_gmac_toptsmc090_wl10 slow

Point                                     Incr      Path
-----
clock clk_tx_i (rise edge)                0.00      0.00
clock network delay (ideal)              0.00      0.00
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/clk_tx_div2_reg/CK (DFFRQX2)
                                         0.00      0.00 r
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/clk_tx_div2_reg/Q (DFFRQX2)
                                         0.36      0.36 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/U26/Y (CLKXOR2X2)
                                         0.47      0.83 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/U36/Y (MX2X1)
                                         0.26      1.08 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/rgmii_txd_o[0]
(DWC_gmac_rgmi_i_gmrt)                  0.00      1.08 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/rgmii_txd_o[0] (DWC_gmac_rgmi_i)
                                         0.00      1.08 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/rgmii_txd_i[0] (DWC_gmac_phy_txmux)
                                         0.00      1.08 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/U12/Y (MX2X1)          0.20      1.28 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/phy_txd_o[0] (DWC_gmac_phy_txmux)
                                         0.00      1.28 f
DWC_gmac_inst/phy_txd_o[0] (DWC_gmac)           0.00      1.28 f
phy_txd_o[0] (out)                         0.00      1.28 f
data arrival time                         1.28

clock clk_tx_180_i (rise edge)            4.00      4.00
clock network delay (ideal)              0.00      4.00
output external delay                   -1.60     2.40
data required time                      2.40

data required time                      2.40
data arrival time                        -1.28

slack (MET)                            1.12

```

For proper circuit operation, the multiplexer select path delay (see paths E and F in Figure 9-11) must be greater than the data path delay (Paths A-D). In the Multiplexer Select Signal Timing Report, this constraint is met because the net delay is obtained using a statistical wire load model. In reality, the net delay for the data path can be greater than the delay for the Multiplexer Select signal because of routing delays. Based on the preliminary routing delays, you must set a proper minimum delay (value greater than the data arrival time of the data paths, that is, paths A-D) to the Multiplexer Select signal path using the sample constraint mentioned below.

```
set_min_delay <delay> -from {DWC_gmac_inst/DWC_gmac_rgmii_inst/DWC_gmac_rgmii_gmrt_inst/clk_tx_div2_reg/CK DWC_gmac_inst/DWC_gmac_rgmii_inst/DWC_gmac_rgmii_gmrt_inst/clk_tx_div2_neg_reg/CK} -to [find port phy_txd_o]
```



- Rerun the synthesis with Synopsys Design Compiler, with compile options.
- Because the valid endpoint for a constraint can only be the input of a sequential cell or a port, you must ensure that the minimum delay value set takes care of the delay from output of the MUX to the port.

A synthesis pragma has been added to ensure that a multiplexer is inferred for the rgmii_txd_o and rgmii_tctl_o outputs. This code is as follows:

```
always @(*)
begin
    case (mux_select) // synopsys infer_mux
        1'b1: begin
            rgmii_txd_o = int_txd_pos;
            rgmii_tctl_o = int_tctl_pos;
        end
        1'b0: begin
            rgmii_txd_o = int_txd_neg;
            rgmii_tctl_o = int_tctl_neg;
        end
    endcase
end
```

It is necessary to set minimum delays. For example, if the net delay for the data path after routing is 0.63 ns, the total path delay is 1.2 ns and the minimum delay must be set to 1.5 (taking care of 0.2-ns delay for the output MUX), as shown below.

```
set_min_delay 1.5 -from {DWC_gmac_inst/DWC_gmac_rgmii_inst/DWC_gmac_rgmii_gmrt_inst/clk_tx_div2_reg/CK DWC_gmac_inst/DWC_gmac_rgmii_inst/DWC_gmac_rgmii_gmrt_inst/clk_tx_div2_neg_reg/CK} -to [find port phy_txd_o]
```

The set_min_delay constraint ensures that buffers or gates are added to delay the path. Make sure that this delay is not greater than (2.4 – clock net delay) ns.

The following timing report shows the additional delay in the Multiplexer Select path achieved using a combination of INVX2 and OAI2BB2XL gates.

Example 9-3 Multiplexer Select Signal Timing Report With set_min_delay of 1.5 Applied

```
Startpoint: DWC_gmac_inst/DWC_gmac_rgmii_inst/DWC_gmac_rgmii_gmrt_inst/clk_tx_div2_reg
            (rising edge-triggered flip-flop clocked by clk_tx_i)
Endpoint: phy_txd_o[0]
            (output port clocked by clk_tx_180_i)
Path Group: clk_tx_i_outputs
Path Type: max
```

```
Des/Clust/PortWire Load ModelLibrary
```

```
-----
```

```
DWC_gmac_top_tsmc090_wl10slow
```

Point	Incr	Path
clock clk_tx_i (rise edge)	0.00	0.00

clock network delay (ideal)	0.00	0.00
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_gmrt_inst/clk_tx_div2_reg/CK (DFFRQX2)	0.00	0.00 r
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_gmrt_inst/clk_tx_div2_reg/Q (DFFRQX2)	0.33	0.33 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_gmrt_inst/U21/Y (INVX2)	0.14	0.47 r
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_gmrt_inst/U3/Y (CLKXOR2X4)	0.41	0.87 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_gmrt_inst/U4/Y (OAI2BB2XL)	0.35	1.22 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_gmrt_inst/rgmii_txd_o[0] (DWC_gmac_rgmi_gmrt)	0.00	1.22 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/rgmii_txd_o[0] (DWC_gmac_rgmi)	0.00	1.22 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/rgmii_txd_i[0] (DWC_gmac_phy_txmux)	0.00	1.22 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/U5/Y (AO22X1)	0.35	1.57 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/phy_txd_o[0] (DWC_gmac_phy_txmux)	0.00	1.57 f
DWC_gmac_inst/phy_txd_o[0] (DWC_gmac)	0.00	1.57 f
phy_txd_o[0] (out)	0.00	1.57 f
data arrival time	1.57	
clock clk_tx_180_i (rise edge)	4.00	4.00
clock network delay (ideal)	0.00	4.00
output external delay	-1.60	2.40
data required time	2.40	

data required time	2.40	
data arrival time	-1.57	

slack (MET)	0.83	

Another example timing report for the Multiplexer Select path, where the set_min_delay value is 1.9, is shown below. An additional delay, using an INVX2 and OAI22X1 gate to meet the minimum delay constraint, is added.

Example 9-4 Multiplexer Select Path Timing Report With set_min_delay Value of 1.9 Applied

```

Startpoint: DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_gmrt_inst/clk_tx_div2_reg
            (rising edge-triggered flip-flop clocked by clk_tx_i)
Endpoint: phy_txd_o[0]
            (output port clocked by clk_tx_180_i)
Path Group: clk_tx_i_outputs
Path Type: max

```

Des/Clust/PortWire Load ModelLibrary

DWC_gmac_toptsmc090_wl10slow

Point	Incr	Path
clock clk_tx_i (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00

```

DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/clk_tx_div2_reg/CK (DFFRQX2)
                                         0.00      0.00 r
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/clk_tx_div2_reg/Q (DFFRQX2)
                                         0.33      0.33 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/U23/Y (INVX2)
                                         0.14      0.46 r
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/U12/Y (CLKXOR2X2)
                                         0.47      0.93 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/U6/Y (INVX2)
                                         0.38      1.32 r
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/U14/Y (OAI22X1)
                                         0.25      1.57 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/DWC_gmac_rgmi_i_gmrt_inst/rgmii_txd_o[0]
(DWC_gmac_rgmi_i_gmrt)
                                         0.00      1.57 f
DWC_gmac_inst/DWC_gmac_rgmi_i_inst/rgmii_txd_o[0] (DWC_gmac_rgmi_i)
                                         0.00      1.57 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/rgmii_txd_i[0] (DWC_gmac_phy_txmux)
                                         0.00      1.57 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/U5/Y (AO22X1)
                                         0.34      1.91 f
DWC_gmac_inst/DWC_gmac_phy_txmux_inst/phy_txd_o[0] (DWC_gmac_phy_txmux)
                                         0.00      1.91 f
DWC_gmac_inst/phy_txd_o[0] (DWC_gmac)
                                         0.00      1.91 f
phy_txd_o[0] (out)
data arrival time
                                         1.91

clock clk_tx_180_i (rise edge)
                                         4.00      4.00
clock network delay (ideal)
                                         0.00      4.00
output external delay
                                         -1.60     2.40
data required time
                                         2.40

-----
data required time
                                         2.40
data arrival time
                                         -1.91

-----
slack (MET)
                                         0.49

```

9.13 Synthesis Guidelines

The GMAC-UNIV supports the complete synthesis flow on your configured RTL core using Synopsys Design Compiler. All the default and required synthesis constraints are automatically generated and applied as per your configured core. After the synthesis is completed, this is generated as `DWC_gmac_top.sdc` in the `<workspace>/export/` directory. For users who do not have a license for Design Compiler, an example synthesis script and constraints file is given in `<workspace>/resources/synth/` directory

The list below gives the major types of constraints applied to the core during synthesis.

1. By default, all input and output signals have 60% of corresponding clock period as SetInput and SetOutput delays. Some of the input/output signals which have higher delays, are constrained to 40% clock-period. You can change the constraints as per your requirement.
2. Multi-cycle paths are defined for the RMON (MMC) counter updates just for saving area. Functionally, they operate properly even if the multi-cycle path constraints are removed.

3. False paths are only set for static input signals such as phy_select_i, etc.
4. Signals crossing clock domains are NOT set as False-Paths. Instead, they are constrained to have a maximum delay of 1 clock-period of the destination clock. This constraint is recommended to ensure proper functioning of the CDC synchronizers (especially the data path synchronizer) as intended.
5. Clock period of the various PHY-side clock inputs are defined as per Ethernet PHY interface requirements. For the host/system clock input, the default clock period is set as 6 ns.

10

Unified Power Format Support

This chapter describes how GMAC-UNIV supports the IEEE 1801 standard. It contains the following sections:

- ❖ “Introduction” on page [451](#)
- ❖ “Block Diagram” on page [453](#)
- ❖ “UPF Implementation” on page [453](#)
- ❖ “UPF Flow and Methodology” on page [457](#)
- ❖ “Area and Power” on page [461](#)
- ❖ “UPF Flow Files” on page [462](#)

10.1 Introduction

Power gating is an effective technique for reducing the leakage power. In power gating, the circuit blocks that are not in use are temporarily turned off to reduce the leakage power of the chip. This temporary shutdown time is known as low-power mode or inactive mode. The circuit blocks switch to the active mode when required for operation again. These two modes are switched at the appropriate time and in the suitable manner to maximize power performance.

The hardware description languages (HDL) do not provide a mechanism for describing the power connections at RTL level. Therefore, verification and backend implementation of a power gated circuit at the RTL level is challenging. The IEEE 1801 standard, also known as the Unified Power Format (UPF), provides a consistent format that you can use to specify power-aware design information that cannot be specified in the HDL code. In addition, you can use the UPF when you do not want to directly specify power-aware design information within the HDL logic.

The UPF defines a language format and simulation semantics for defining a power domain and a set power supplies to the power domain. A power domain is a group of elements in the design that share a common set of power supply needs. The UPF language specifies the following:

- ❖ How to create a supply network to supply power to each design element.

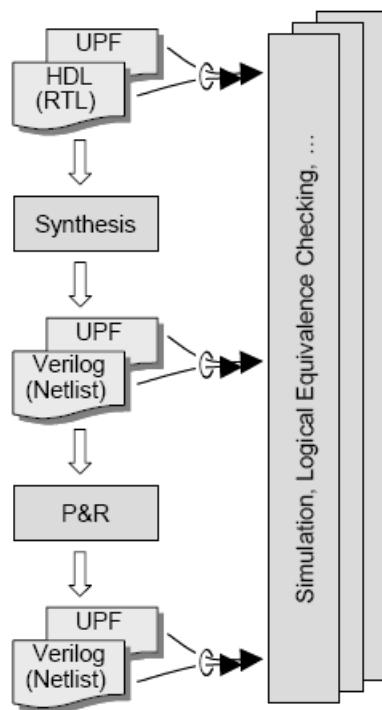
The supply network consists of supply ports, switches, and supply nets. A supply net is a conductor that carries a supply voltage or ground throughout a given power domain. By controlling the operating voltages of each supply net and whether the supply nets (and their connected design elements) are turned on or off, the supply network provides power at the level that is required by the functional areas of the chip to complete the computational task in a timely manner.

- ❖ How individual supply nets behave with other supply nets.

- ❖ How the logic functionality is extended to support dynamic power switching to design elements.

Figure 10-1 shows the UPF implementation.

Figure 10-1 UPF Implementation



10.1.1 Advantages of UPF

UPF has the following advantages:

- ❖ UPF is independent of the RTL code and can be used to add power-related functionality to RTL without modifying it.
- ❖ UPF enables easy mapping from UPF to the logic design. Anything created in a UPF specification is created within a specific scope of the logic design. This facilitates writing the UPF code and debugging and analyzing the low-power design specification.
- ❖ UPF defines a support package in SystemVerilog and VHDL to facilitate verification. This package defines routines for querying the current state of a supply net or port and setting the state of a supply port. This allows the testbench to mimic the off-chip power supplies that are connected to the supply pads of a chip.

The ability to define a UPF supply net to an HDL logic value conversion is provided when supply net is connected to a logic port in the design. This facilitates integrating the supply network defined in UPF to power-aware models in the design without requiring re-writing of model.

- ❖ UPF provides a consistent way to specify power implementation intent throughout the design process including synthesis, physical implementation, and verification. This helps in better checking, interoperability, and productivity with mixed EDA flows.

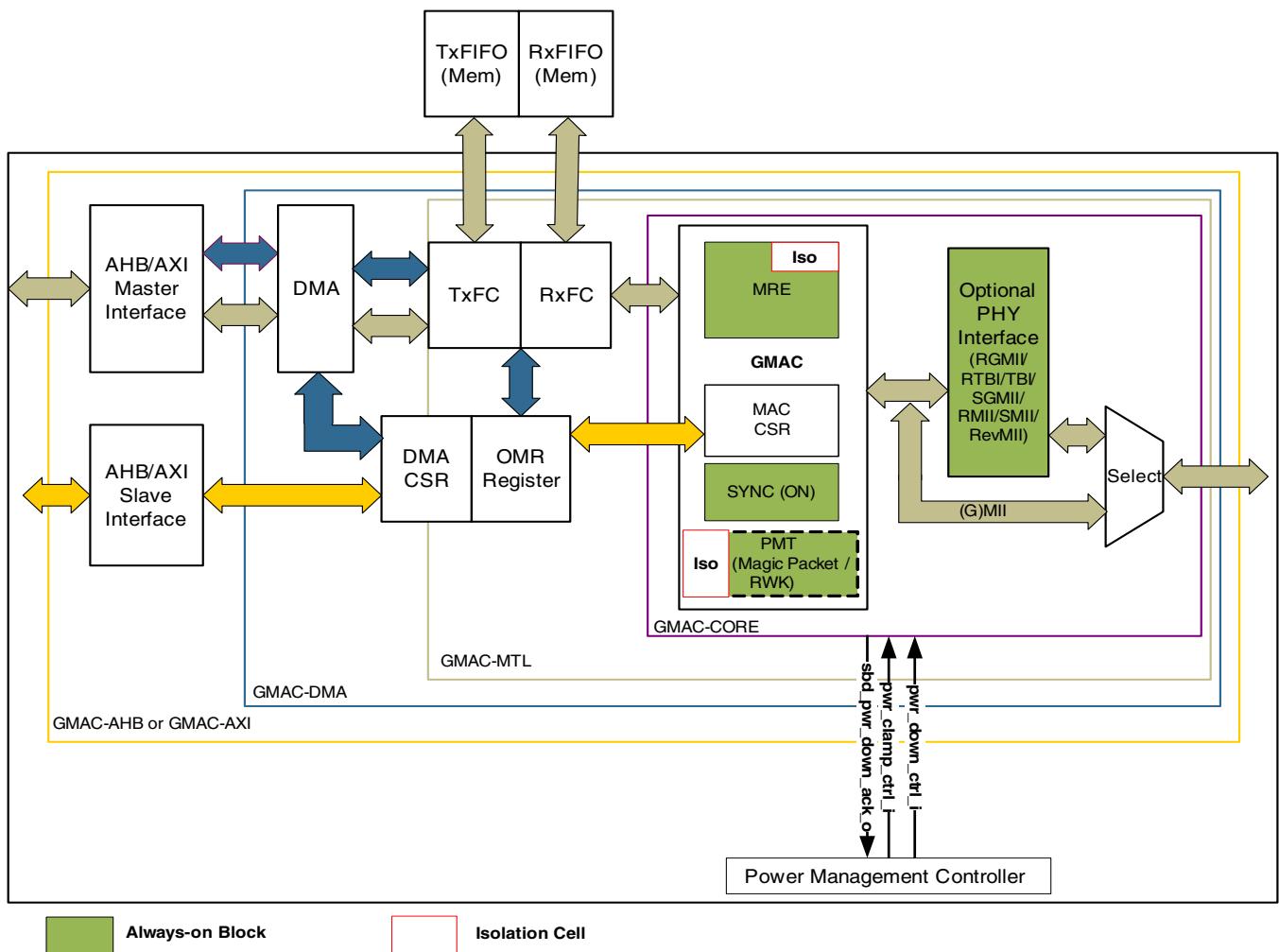
The UPF file is the input to several tools such as simulation, synthesis, formal verification, and place-and-route tools. Synthesis tools can read the RTL or UPF design input files and produce a netlist.

10.2 Block Diagram

The **Power Management Block** (PMT block) of GMAC-UNIV provides the power management functionality. The Power Management block enables a system to go into the low-power or sleep mode and then wake up on receiving a specific Ethernet packet. In the sleep mode, only MAC receiver requires power. Therefore, you can power off the rest of the logic of the system. The UPF support described in this chapter helps in isolating the MAC receiver logic into a separate voltage island in the implementation flow.

Figure 10-2 shows the block diagram of GMAC-UNIV with UPF support.

Figure 10-2 UPF Flow Block Diagram



10.3 UPF Implementation

The following sections describe how the isolation scheme, address filtering, and clocking scheme is implemented in GMAC-UNIV. It also explains the power-down and power-sequence implementation.

10.3.1 Isolation Scheme

An isolation cell is a logic cell that can isolate a power-down domain from a domain that is not powered down. The Isolation cells isolate power gated block from the normal ON block.

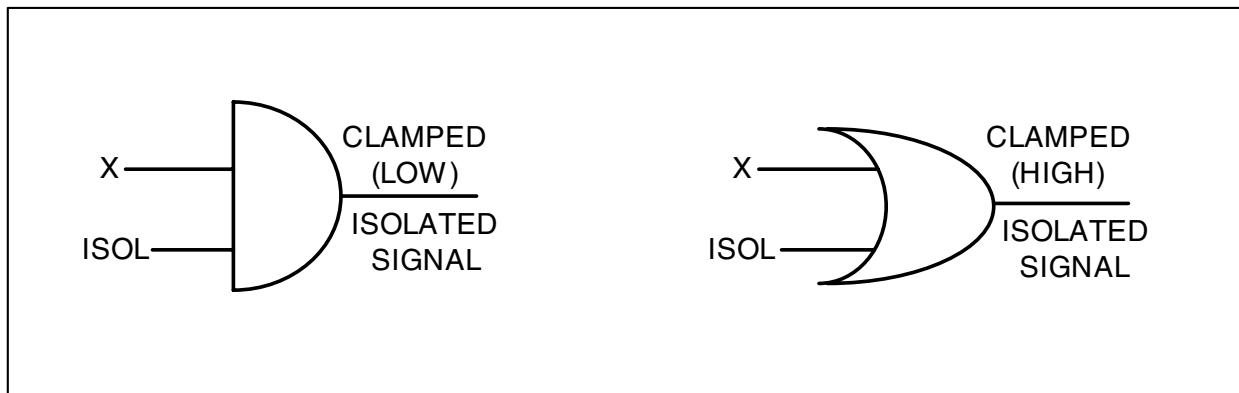
You can use the isolation cells to control the output of the powered down blocks. An isolation cell can clamp the output to a specific and legal value. The isolation cells are of the following three types:

- ❖ Isolation cells that clamp the signal to 0.
- ❖ Isolation cells that clamp the signal to 1.
- ❖ Isolation cells that latch the signal to most recent value.

When the power-gated block is powered down, the powered block receives the clamped signal values. If these signals are active high and also clamped high, the destination may interpret these signals as commands and act incorrectly. To avoid this issue, you should clamp the signal output to its inactive state.

When using the active high logic, you can clamp the value to 0 by using an AND gate function. Similarly, for the active low logic, you can clamp the value to 1 by using an OR gate function as shown in [Figure 10-3](#).

Figure 10-3 Isolation Scheme



During power down, the always-on block is kept powered up so that it can respond to the reception of a magic packets. The rest of the GMAC core is power gated. The pwr_clamp_ctrl signal controls the isolation cells and the VDD powers the isolation cells. If the outputs are isolated at the receiving blocks, then you may require more than one isolation cell for each output. Therefore, isolating the outputs of the power-gated block is more area efficient approach. Isolating outputs of the power-gated block also makes the analysis easier. If these signals are isolated in the always-on block, then you need to check each fan-out of the output signal to ensure that there is an isolation cell on it. In [Figure 10-2](#), the isolation cells are shown in the always-on block for easy reference. However, the isolation cells are actually implemented at the output of the power-gated block.

10.3.2 Address Filtering

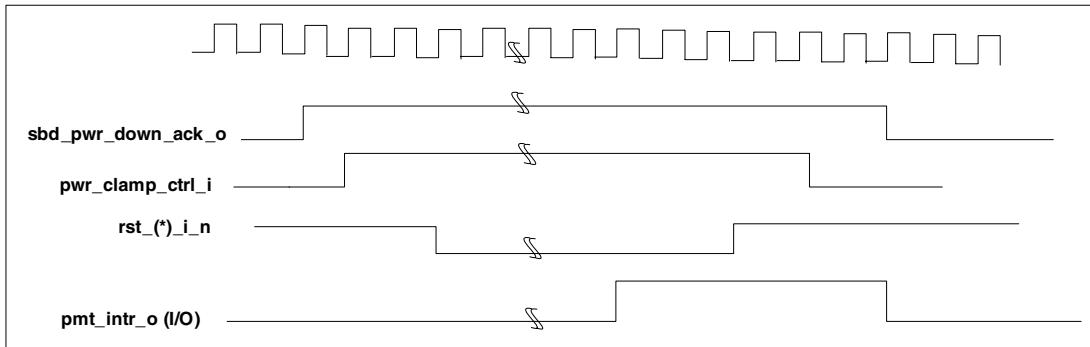
The PMT block processes all packets that are passed by the [Address Filtering Module](#). You can control the address filtering by programming the MAC Frame Filter register (offset 0x0004). When the GMAC-UNIV is configured for the UPF support, then only one MAC Address (MAC Address 0) is used for filtering as rest of the Address Filter block is in the power-off domain.

In UPF configuration, the PMT block processes any packet whose destination address (DA) matches the value in MAC Address 0. However, for magic packet and remote wake-up packet detection, the PMT block accepts all multicast packets (including broadcast).

10.3.3 Power-Down and Power-Up Sequence

Figure 10-4 shows the power-down and power-up sequence.

Figure 10-4 Power-Down and Power-Up Sequence



10.3.3.1 Power-Down Sequence

The following list describes the power-down sequence:

1. The software should perform the following tasks:
 - a. Disable the Transmit DMA (if applicable) by clearing bit 13 (ST) of [Register 6 \(Operation Mode Register\)](#).
 - b. Wait for any previous frame transmissions to complete. You can check this by reading the bit 24 and bit 16 of [Register 9 \(Debug Register\)](#).
 - c. Disable the MAC transmitter and MAC receiver by clearing the bit 3 ([TE: Transmitter Enable](#)) and bit 2 ([RE: Receiver Enable](#)) in [Register 0 \(MAC Configuration Register\)](#).
 - d. Wait till the Receive DMA empties all frames from the Rx FIFO (in GMAC-MTL and GMAC-DMA configurations).
You can check this by reading bit [9:8] of the Register 9 (Debug Register). If both bits are set to zero, it indicates that the Rx FIFO is empty.
 - e. Configure the magic packet (bit 2) and/or remote wake-up (bit 1) detection in the [PMT Control and Status Register](#) (offset 0x002C).
 - f. Enable the MAC Receiver by setting bit 2 (RE: Receiver Enable) and then set the bit 0 ([Power Down](#)) in the PMT Control and Status register to initiate the power-down sequence in MAC.
2. The MAC asserts the `sbd_pwr_down_ack_o` signal after all FIFOs are empty and the power down is set in the PMT block.
This indicates that the always-on block is actively programmed and the power management controller can start switching off the power.
3. The Power Management Controller should perform the following tasks:
 - a. Assert the power clamp control (`pwr_clamp_ctrl_i`) to clamp the voltages of the isolation cells.
 - b. Assert the `pwr_down_ctrl_i` signal to shutdown the power to the blocks in the power-OFF hierarchy.

10.3.3.2 Power-Up Sequence

The MAC wakes up on receiving the magic packet or remote wake-up frame. The following list describes the power-up sequence:

1. The MAC asserts pmt_intr_o.
2. The Power Management Controller should perform the following tasks:
 - a. De-assert the pwr_down_ctrl_i signal to enable power to the blocks in the power-OFF hierarchy.
 - b. Assert resets to ensure that all registers in the blocks that were shutdown are reset. To properly initialized all asynchronous logic, the reset should be active for at least 4-clocks duration of the slowest clock in GMAC-UNIV.
 - c. De-assert the resets.
 - d. De-assert the pwr_clamp_ctrl_i signal.



Note In GMAC-DMA, GMAC-AHB, and GMAC-AXI configurations, the application reset is synchronized to all clock domains. The application must wait for the de-assertion of resets before it de-asserts pwr_clamp_ctrl_i.

3. The software should perform the following tasks:
 - a. De-assert the pmt_intr_o by reading the PMT Control and Status register.
 - b. Perform a write operation (with reset values) to the [PMT Control and Status Register](#) and the [Remote Wake-Up Frame Detection](#) register (if present) so that the corresponding values in the always-on block gets synchronized. Otherwise, the values of these registers are different.
 - c. Perform write operations to the [Register 0 \(MAC Configuration Register\)](#) and MAC Address0 registers to synchronize the values in the CSR module and the respective bits in the always-on block. Otherwise, the MAC receiver will be ON even though the Receive Enable bit is set to 0.

After completing these steps, the software must initialize all registers, enable the transmitter, and program the DMA (in DMA configurations) to resume the normal operation.

10.3.4 Clocks

During the low-power mode, only receive clock clk_rx_i is active for default GMII or MII PHY interface. The other clocks may also be active depending upon the corresponding optional PHY interface. The clk_rx_i clocks the power-OFF and power-ON domains and drives the registers of the always-on and power-OFF modules in the receive path. The modules that are ON during the power-down mode are nested one-level below in hierarchy than the top-level module.

The clk_rx_i clock does not require separate ports for the power-OFF and power-ON domains. However, you need ensure that the clk_rx_i clock can propagate to the always-on and power-OFF modules. In the physical layout, if the clk_rx_i clock in the GMAC core comes from an OFF domain to always-on domain, then the clock network should be buffered by the always ON clock buffers. You can do this by using the following methods:

- ❖ When the always ON clock buffers are not available in the library, you can use the normal buffers. However, you need to place these buffers so that their power signal is always ON.
- ❖ When the always ON clock buffers are available in the library, you can place it as normal cells. The always ON clock buffers have the following dual-power signals:
 - ◆ One connected to always-on power supply.

- ◆ One connected to the power rail of the OFF domain.

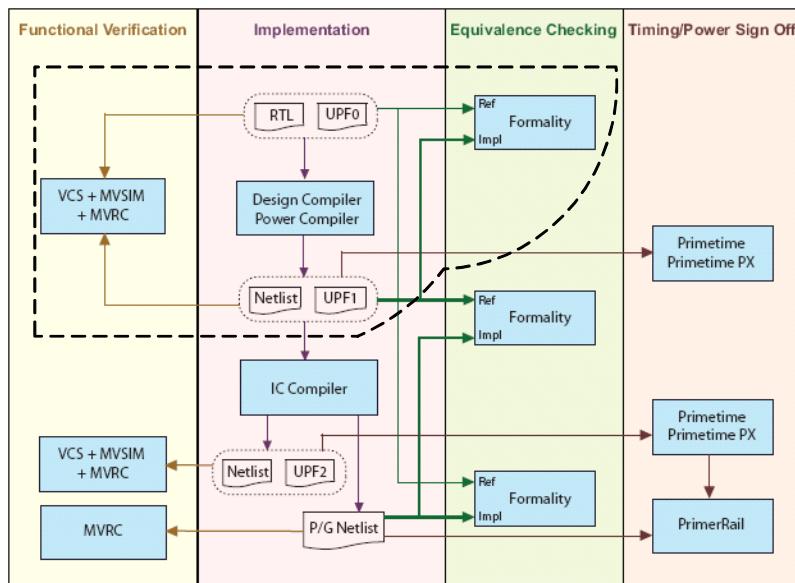
You need to create few power straps of always-on power supply in the OFF domain to connect to the dual-power clock buffers present in the OFF domain.

Alternatively, you can place top-level clock port in the always-on domain so that the always-on buffers are not required.

10.4 UPF Flow and Methodology

[Figure 10-5](#) shows the Synopsys low-power synthesis, implementation, and verification flow. The dotted line in [Figure 10-5](#) indicates the flow that is supported directly with the coreConsultant. The coreConsultant automatically generates the scripts for this flow.

Figure 10-5 UPF Flow

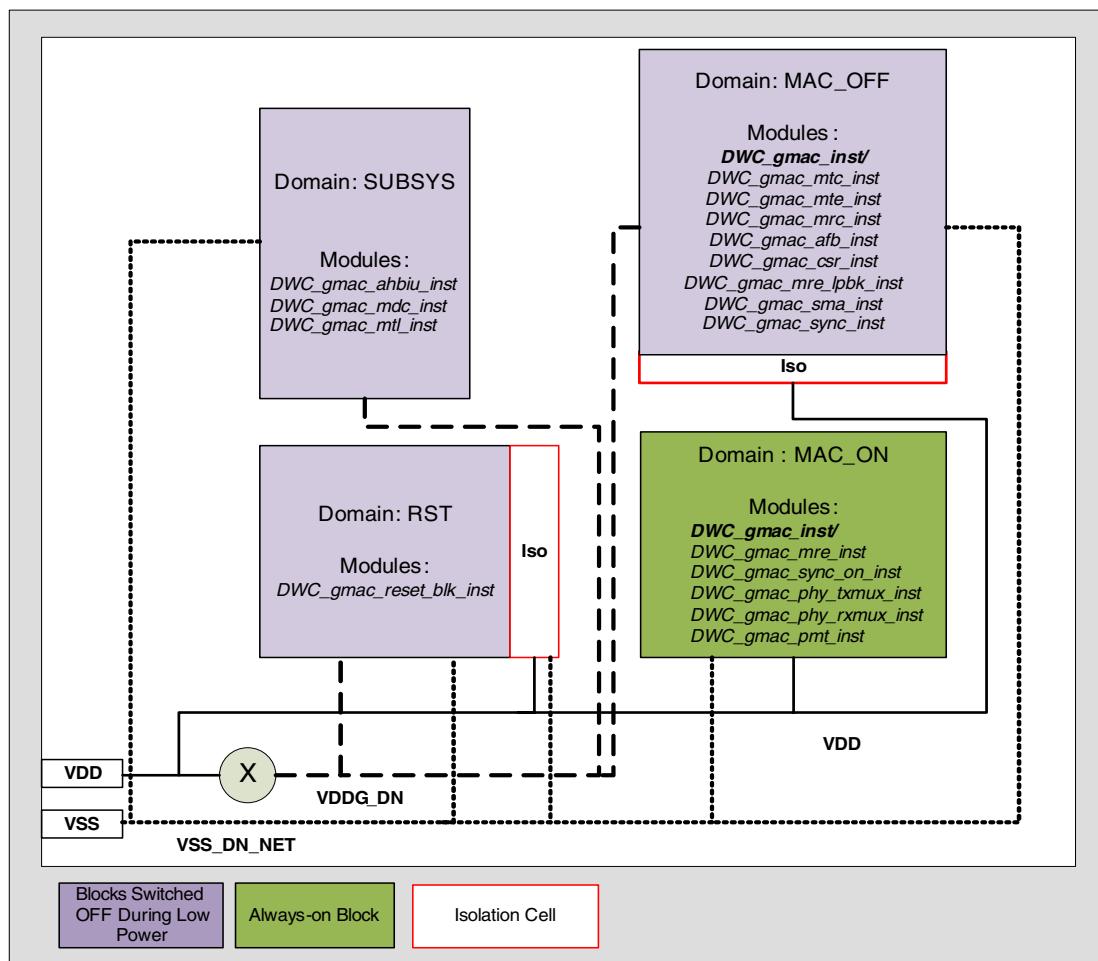


The UPF flow files are created in <config>/resources/upf_scripts directory. These files are created based on the GMAC configuration. For example, [Table 10-1](#) describes the UPF files for the GMAC-AHB configuration with PMT enabled and default GMII or MII PHY interface.

Table 10-1 UPF Flow Files

File	Description
README	Instructions for the UPF flow and setup.
DWC_gmac_syn.upf	UPF file for the synthesis flow.
DWC_gmac_sim.upf	UPF file for the simulation flow.
set_voltage.tcl	Tcl file to set the voltage values.

[Figure 10-6](#) shows the power domains for the GMAC-AHB configuration.

Figure 10-6 GMAC-AHB Block Power Domains

10.4.1 Simulation

You can use the VCS simulator and MVSIM multivoltage simulation tool for the functional verification of the design with multivoltage features. [Figure 10-7](#) shows the simulation flow using the MVSIM tool.

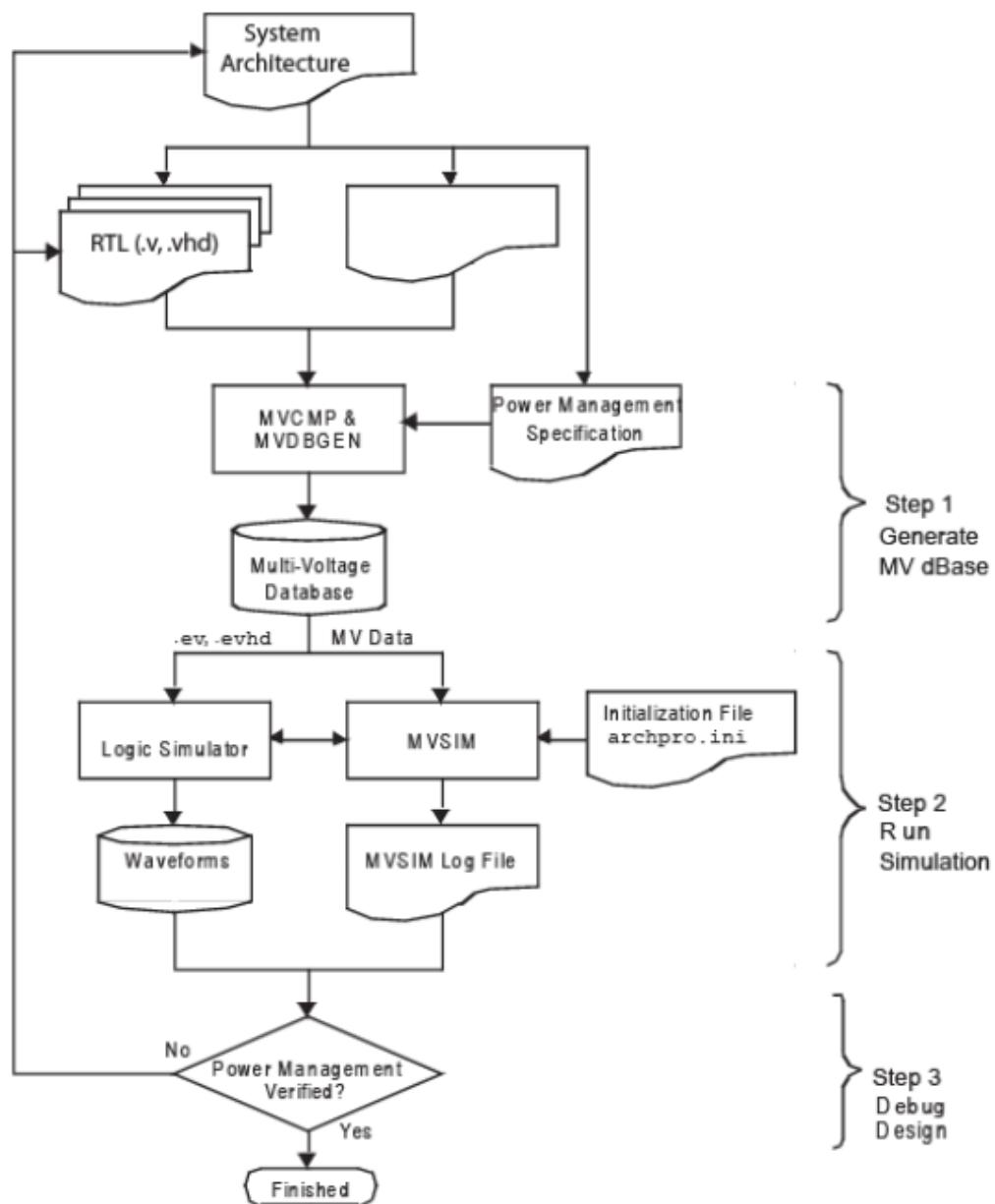
Figure 10-7 Multivoltage Simulation Flow

Table 10-2 describes the files that are required for the MVSIM tool.

Table 10-2 MVSIM Support Files

File Name	Location	Description
archpro.ini	<config>/sim/	You can use the archpro.ini file specify the MVSIM settings. You must include the library that your design uses in the archpro.ini file as shown in the following example: <pre># library path and file search_path = /global/cust_apps_sgip001/libs/tsmc_065/digital/Front_ End /timing_power_noise/NLDM/tcbn65lp_140b/ link_library = tcbn65lpwc.db tcbn65lpwcl1d081d08.db</pre>
mvscript	<config>/sim/<sim_dir>/scripts/	You can use the mvscript to automatically run the compilation, db generation, and simulations. The mvscript can use the generated Makefile and also automatically select the simulator (VCS, NCSIM, or MTI). You can execute the mvscript in the standalone mode by using the following command: <pre>./gmac_subsys_vtb/scripts/mvscript <testcase_name> [options]</pre> The MV Tools create the following reports in the sim/mvreports directory: <ul style="list-style-type: none"> • mvcmp.log - Log for the compilation step • mvdbgen.log - Log for the db generation step • mvsim.log - Log for MV simulation step For more information about these files, see " UPF Flow Files " on page 462 .



You can use the following testcases for the low-power UPF flow with MV tools:

- rgf_power_down_mp_lpm: For low-power testing using magic packet.
- rgf_power_down_rwk_lpm: For low-power testing using remote wake-up.

10.4.2 Synthesis

The coreConsultant supports the synthesis flow using UPF for the following strategies:

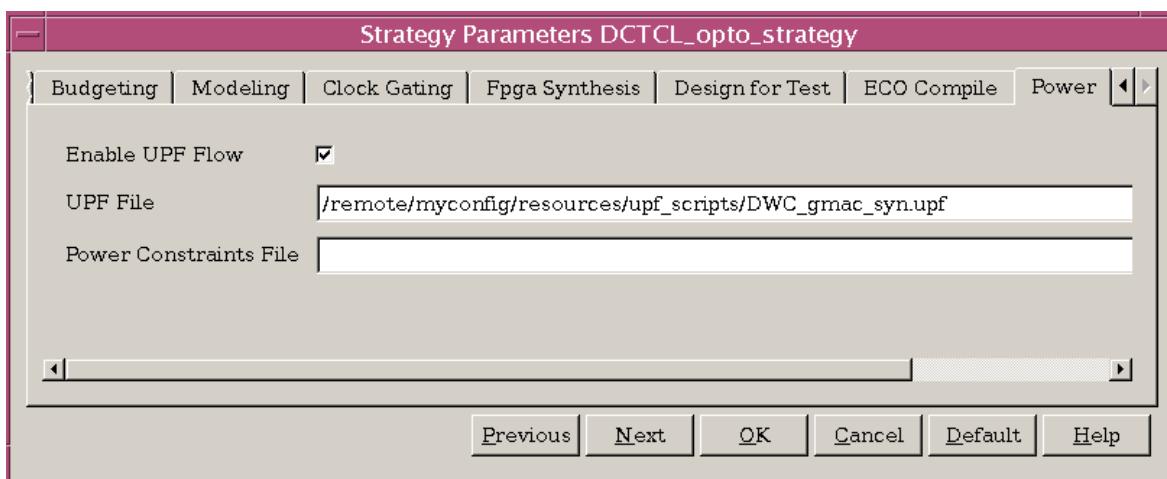
- ❖ DCTCL_opto_strategy
- ❖ DCTCL_one_pass_compile_ultra

To enable the synthesis flow using UPF, perform the following steps in coreConsultant:

1. In the Specify Configuration activity, select **Low Power Support**, and then select **Enable UPF Based Low Power Support**.
2. In the Synthesize activity, click **Options**.
3. In the Power tab, select **Enable UPF Flow**.

The following UPF file is loaded by default as shown in [Figure 10-8](#):

<config>/resources/upf_scripts/DWC_gmac_syn.upf

Figure 10-8 Strategy Parameters Dialog Box

4. (Optional) In the Power Constraints File field, set the voltage values by specifying a power constraint (set_voltage.tcl) file.

A sample power constraint (set_voltage.tcl) file is available in <config>/resources/upf_scripts directory. By default, in the set_voltage.tcl file, the voltage value is set to 1.08V for using the 1.08 voltage library as shown in the following example:

```
set_voltage 1.08 -object_list {VDD VDDG_DN}
set_voltage 0.0 -object_list {VSS_DN_NET}
```

You can modify the voltage value depending on the library that you are using.

[Table 10-3](#) describes the files that are generated for synthesis flow using UPF.

Table 10-3 Synthesis Flow Files

File	Location	Description
DWC_gmac_top.upf	<config>/syn/final/db	The output of the synthesis. You can use this file for validating the UPF file given as an input (DWC_gmac_syn.upf).
DWC_gmac_top_power_domains.rpt	<config>/syn/final/report	Contains a list of the power domains.
DWC_gmac_top_isolation_cell.rpt	<config>/syn/final/report	Contains a list of signals having isolation cells.
DWC_gmac_top_power_switch.rpt	<config>/syn/final/report	Contains a list of power switches in the design.

10.5 Area and Power

The low-power mode reduces the leakage power as well as the dynamic power during power down. [Table 10-4](#) gives the area estimate of the Power ON blocks.

Table 10-4 Power ON Blocks Area

Module	Estimated Area
PMT (Magic Packet + RWK)	5-6 K

Table 10-4 Power ON Blocks Area

Module	Estimated Area
PMT (Magic Packet)	~1K
MRE (RPE+CRC)	2-3.5K
LP Control Logic	~0.6 K
Total Area	6-10K

10.6 UPF Flow Files

The following sections provide samples of the UPF files that are created for simulation and synthesis using UPF.

- ❖ Synthesis Files
 - ◆ [DWC_gmac_syn.upf](#)
 - ◆ [DWC_gmac_top_power_domains.rpt](#)
- ❖ Simulation Files
 - ◆ [mvcmp.log](#)
 - ◆ [mvsim.log](#)

10.6.1 DWC_gmac_syn.upf

The DWC_gmac_syn.upf file is the synthesis flow file. The following is a sample DWC_gmac_syn.upf file:

```
# Setting the UPF version
set upf_version 1.0

# Setting the top-level power down modules
create_power_domain TOP

create_power_domain SUBSYS -elements {DWC_gmac_ahbiu_inst DWC_gmac_mdc_inst
DWC_gmac_mtl_inst}

create_power_domain RST -elements {DWC_gmac_reset_blk_inst}

# GMAC modules which are in power-down domain
create_power_domain MAC_OFF -elements {DWC_gmac_inst/DWC_gmac_mtc_inst \
DWC_gmac_inst/DWC_gmac_mte_inst \
DWC_gmac_inst/DWC_gmac_mrc_inst \
DWC_gmac_inst/DWC_gmac_afb_inst \
DWC_gmac_inst/DWC_gmac_csr_inst \
DWC_gmac_inst/DWC_gmac_mre_1pbk_inst \
DWC_gmac_inst/DWC_gmac_sma_inst \
DWC_gmac_inst/DWC_gmac_sync_inst
}

# GMAC modules which are in power-on domain
create_power_domain MAC_ON -elements {DWC_gmac_inst/DWC_gmac_mre_inst \
DWC_gmac_inst/DWC_gmac_sync_on_inst \}
```

```

DWC_gmac_inst/DWC_gmac_pmt_inst \
DWC_gmac_inst/DWC_gmac_phy_txmux_inst \
DWC_gmac_inst/DWC_gmac_phy_rxmux_inst
}

# Create the top-level supply ports
create_supply_port VDD
create_supply_port VSS

# Create the top-level supply nets
create_supply_net VDD -domain TOP
create_supply_net VDDG_DN -domain TOP

# Create supply ports/nets for individual domains
create_supply_net VSS_DN_NET -domain TOP
create_supply_net VSS_DN_NET -domain SUBSYS -reuse
create_supply_net VDDG_DN -domain SUBSYS -reuse

create_supply_net VDD -domain RST -reuse
create_supply_net VSS_DN_NET -domain RST -reuse
create_supply_net VDDG_DN -domain RST -reuse

#Create supply ports/nets for GMAC power-down modules
create_supply_net VSS_DN_NET -domain MAC_OFF -reuse
create_supply_net VDD -domain MAC_OFF -reuse
create_supply_net VDDG_DN -domain MAC_OFF -reuse

#Create supply ports/nets for GMAC always ON modules
create_supply_net VSS_DN_NET -domain MAC_ON -reuse
create_supply_net VDD -domain MAC_ON -reuse

# Connect the supply nets to the ports
connect_supply_net VDD -ports VDD
connect_supply_net VSS_DN_NET -ports {VSS}

# Set the domain supply nets
set_domain_supply_net TOP -primary_power_net VDD -primary_ground_net VSS_DN_NET
set_domain_supply_net SUBSYS -primary_power_net VDDG_DN -primary_ground_net VSS_DN_NET
set_domain_supply_net RST -primary_power_net VDDG_DN -primary_ground_net VSS_DN_NET
set_domain_supply_net MAC_OFF -primary_power_net VDDG_DN -primary_ground_net VSS_DN_NET
set_domain_supply_net MAC_ON -primary_power_net VDD -primary_ground_net VSS_DN_NET

# Set the isolation logic for outputs of different domains
set_isolation_reset_sync -domain RST -isolation_power_net VDD -isolation_ground_net
VSS_DN_NET -clamp_value 1 -applies_to outputs
set_isolation_control reset_sync -domain RST -isolation_signal {pwr_clamp_ctrl_i} -
-isolation_sense high -location parent
map_isolation_cell reset_sync -domain RST -lib_cells {ISOHID1, ISOHID2, ISOHID4,
ISOHID8}

set_isolation_reset_sync_inp -domain RST -isolation_power_net VDD -isolation_ground_net
VSS_DN_NET -clamp_value 1 -elements {DWC_gmac_reset_blk_inst/dcr_soft_reset_i}

```

```

set_isolation_control reset_sync_inp -domain RST -isolation_signal {pwr_clamp_ctrl_i} -
-isolation_sense high -location parent
map_isolation_cell reset_sync_inp -domain RST -lib_cells {ISOHID1, ISOHID2, ISOHID4,
ISOHID8}

set_isolation mac_off_blk -domain MAC_OFF -isolation_power_net VDD -
-isolation_ground_net VSS_DN_NET -clamp_value 0 -applies_to outputs
set_isolation_control mac_off_blk -domain MAC_OFF -isolation_signal {pwr_clamp_ctrl_i}
-isolation_sense high -location parent
map_isolation_cell mac_off_blk -domain MAC_OFF -lib_cells {ISOLOD1, ISOLOD2, ISOLOD4,
ISOLOD8}

# Create power switch for different domains
create_power_switch power_switch -domain TOP -input_supply_port {vin VDD} -
output_supply_port {vout VDDG_DN} -control_port {ctrl pwr_down_ctrl_i} -on_state {ON
vin {!ctrl}} -off_state {OFF {ctrl}}

# Create port states
add_port_state VSS -state {ON 0.0}
add_port_state VDD -state {ON 1.08}
add_port_state power_switch/vout -state {ON 1.08} -state {off_state off}

# Create power state table
create_pst state_table -supplies {VDD VDDG_DN VSS}
add_pst_state NORMAL -pst state_table -state {ON ON ON}
add_pst_state POWER_DOWN -pst state_table -state {ON off_state ON}

```

10.6.2 DWC_gmac_top_power_domains.rpt

The DWC_gmac_top_power_domains.rpt file is a report file that gives a list of the power domains. The following is a sample DWC_gmac_top_power_domains.rpt file:

```
*****
Report : power_domain
Design : DWC_gmac_top
Version: C-2009.06-SP1
Date   : Tue Sep  8 16:22:35 2009
*****  

-----  

Power Domain      : MAC_ON
Current Scope     : DWC_gmac_top
Elements          : DWC_gmac_inst/DWC_gmac_mre_inst,
                   DWC_gmac_inst/DWC_gmac_sync_on_inst,
                   DWC_gmac_inst/DWC_gmac_phy_txmux_inst,
                   DWC_gmac_inst/DWC_gmac_phy_rxmux_inst
Available Supply Nets : Only primary and ground nets available on this domain.  

Connections           -- Power --           -- Ground --
Primary: (Max Op. Voltage) VDD (1.08)           VSS_DN_NET (0.00)
-----
```



```

Power Domain      : MAC_OFF
Current Scope    : DWC_gmac_top
Elements         : DWC_gmac_inst/DWC_gmac_mtc_inst,
                  DWC_gmac_inst/DWC_gmac_mte_inst,
                  DWC_gmac_inst/DWC_gmac_mrc_inst,
                  DWC_gmac_inst/DWC_gmac_afb_inst,
                  DWC_gmac_inst/DWC_gmac_csr_inst,
                  DWC_gmac_inst/DWC_gmac_mre_lpbk_inst,
                  DWC_gmac_inst/DWC_gmac_sma_inst,
                  DWC_gmac_inst/DWC_gmac_sync_inst
Available Supply Nets : Only primary and ground nets available on this domain.

Connections      -- Power --          -- Ground --
Primary: (Max Op. Voltage) VDDG_DN (1.08)   VSS_DN_NET (0.00)
Isolation: mac_off_blk     VDD (1.08)        VSS_DN_NET (0.00)
-----
```

10.6.3 DWC_gmac_top_isolation_cell.rpt

The DWC_gmac_top_isolation_cell.rpt file is a report file that gives a list of the signals having isolation cells. The following is a sample DWC_gmac_top_isolation_cell.rpt file:

```
*****
Report : isolation_cell
Design : DWC_gmac_top
Version: C-2009.06-SP1
Date   : Tue Sep  8 16:22:35 2009
*****
```

Power Domain : MAC_OFF

```
=====
| Port | Port Dir.| ISO Cell Name | ISO Lib Cell | ISO Strategy |
=====
| mti_rdy_o | Out | DWC_gmac_inst/mti_rdy_o_UPF_ISO | ISOLOD8 | mac_off_blk |
| mti_txstatus_o[31] | Out | DWC_gmac_inst/mti_txstatus_o[31]_UPF_ISO | ISOLOD8 |
mac_off_blk
```

10.6.4 DWC_gmac_top_power_switch.rpt

The DWC_gmac_top_power_switch.rpt file is a report file that gives a list of the power switches in the design. The following is a sample DWC_gmac_top_power_switch.rpt file:

```
*****
Report : power_switch
Design : DWC_gmac_top
Version: C-2009.06-SP1
Date   : Tue Sep  8 16:22:35 2009
*****
```

```
-----
Power Switch      : power_switch
Current Scope    : DWC_gmac_top
Switch Power Domain : TOP
```

```

Input port net          : VDD (1.08)
Output port net        : VDDG_DN (1.08)
-----
```

10.6.5 mvcmp.log

The mvcmp.log file is the log file for compilation. This log file contains information about when the system was shutdown or powered up, that is, the power state changing from ACTIVE to SHUTDOWN or vice versa. It also provides information about how the isolation control signals changed their state. The following is a sample mvcmp.log file:

```

ArchPro MVCMP
Version C-2009.06 for Linux 2.6.9-67.ELhugemem -- 30 May, 2009
Copyright (c) 2004-2009 by Synopsys, Inc.
```

ALL RIGHTS RESERVED

This software and the associated documentation are confidential and proprietary to Synopsys, Inc. Your use or disclosure of this software is subject to the terms and conditions of a written license agreement between you, or your company, and Synopsys, Inc.

```

mvcmp -upf ../resources/upf_scripts/DWC_gmac_sim.upf
=====
MVCMP completed with 0 warning(s), 0 error(s).
Total time = 0.30sec CPU time = 0.3sec 12% CPU utilized
Peak Memory = 20.43 MB
```

10.6.6 mvsim.log

The mvsim.log file is the log file for simulation. The following is a sample mvsim.log file:

```

ArchPro MVSIM
Version C-2009.06 for Linux 2.6.9-67.ELhugemem -- 30 May, 2009
Copyright (c) 2004-2009 by Synopsys, Inc.
```

ALL RIGHTS RESERVED

This software and the associated documentation are confidential and proprietary to Synopsys, Inc. Your use or disclosure of this software is subject to the terms and conditions of a written license agreement between you, or your company, and Synopsys, Inc.

```

[MVSIM] INFO 5003: MVSIM initialised in PROTECTED mode.
[MVSIM] INFO 5001: Design database found. Loading database from apdb/mvdb/design.db.
[MVSIM] INFO 5001: Class Library database found. Loading database from
apdb/cdb/mvspec.db.
[MVSIM] INFO 5001: Cross-over database found. Loading database from apdb/mvdb/xover.db.
[MVSIM] INFO 5632: SupplyNet VDD started with Voltage ON State/Voltage value 1.080000.
[MVSIM] INFO 5632: SupplyNet VDDG_DN started with Voltage ON State/Voltage value
1.080000.
[MVSIM] INFO 5632: SupplyNet VSS_DN_NET started with Voltage ON State/Voltage value
0.000000.
[MVSIM] INFO 5631: PowerSwitch power_switch started with ON state. Control port
power_switch/ctrl started with Voltage ON State/Voltage value 0.000000, Input supply
port power_switch/vin started with Voltage ON State/Voltage value 1.080000 and with ON
```

state, Output supply port power_switch/vout started with Voltage ON State/Voltage value 1.080000 and with ON state.

```
[MVSIM] INFO 5028: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i corresponding to Isolation Policy 'mac_off_blk' started with value 0.
[MVSIM] INFO 5034: Isolation removed for Isolation Policy 'mac_off_blk' at start of simulation.
[MVSIM] INFO 5028: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i corresponding to Isolation Policy 'reset_sync' started with value 0.
[MVSIM] INFO 5034: Isolation removed for Isolation Policy 'reset_sync' at start of simulation.
[MVSIM] INFO 5028: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i corresponding to Isolation Policy 'reset_sync_inp' started with value 0.
[MVSIM] INFO 5034: Isolation removed for Isolation Policy 'reset_sync_inp' at start of simulation.
[MVSIM] INFO 5027: Simulation started with Power Domain MAC_OFF in mode ACTIVE
[MVSIM] INFO 5027: Simulation started with Power Domain MAC_ON in mode ACTIVE
[MVSIM] INFO 5027: Simulation started with Power Domain RST in mode ACTIVE
[MVSIM] INFO 5027: Simulation started with Power Domain SUBSYS in mode ACTIVE
[MVSIM] INFO 5027: Simulation started with Power Domain TOP in mode ACTIVE
[MVSIM] INFO 5005: In Power State Table state_table Simulation started with design in NORMAL state
[MVSIM] INFO 5102: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i corresponding to Isolation Policy 'reset_sync_inp' changed to value = 1 at time = 4994000 ps.
[MVSIM] INFO 5105: Isolation applied on Isolation Policy 'reset_sync_inp' at time = 4994000 ps.
[MVSIM] INFO 5102: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i corresponding to Isolation Policy 'reset_sync' changed to value = 1 at time = 4994000 ps.
[MVSIM] INFO 5105: Isolation applied on Isolation Policy 'reset_sync' at time = 4994000 ps.
[MVSIM] INFO 5102: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i corresponding to Isolation Policy 'mac_off_blk' changed to value = 1 at time = 4994000 ps.
[MVSIM] INFO 5105: Isolation applied on Isolation Policy 'mac_off_blk' at time = 4994000 ps.
[MVSIM] WARNING 5203: Reset wiggled in SHUTDOWN_MODE. Reset test_bench.dut_inst.DWC_gmac_inst.DWC_gmac_mre_1pbk_inst.incr_wr_ptr_synzr.rst_d_n wiggled for Power Domain MAC_OFF at time 5069200 ps.
.....
[MVSIM] INFO 5718: PowerSwitch power_switch changed from OFF state to ON state. Control port power_switch/ctrl changed to Voltage value 0.000000, Output supply port power_switch/vout changed to Voltage ON State/Voltage value 1.080000 and changed from off_state state to ON state at time 9266000 ps.
[MVSIM] INFO 5719: SupplyNet VDDG_DN changed to Voltage ON State/Voltage value 1.080000 at time 9266000 ps.
[MVSIM] INFO 5112: Design state changed. In Power State Table state_table, Design transitioned from POWER_DOWN state to NORMAL state at time 9266000 ps.
[MVSIM] INFO 5104: Power Domain state changed. Power Domain MAC_OFF state changed from SHUTDOWN to ACTIVE at time = 9266000 ps.
[MVSIM] INFO 5104: Power Domain state changed. Power Domain RST state changed from SHUTDOWN to ACTIVE at time = 9266000 ps.
[MVSIM] INFO 5104: Power Domain state changed. Power Domain SUBSYS state changed from SHUTDOWN to ACTIVE at time = 9266000 ps.
```

```
[MVSIM] INFO 5102: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i  
corresponding to Isolation Policy 'reset_sync_inp' changed to value = 0 at time =  
10784400 ps.  
[MVSIM] INFO 5106: Isolation removed for Isolation Policy 'reset_sync_inp' at time =  
10784400 ps.  
[MVSIM] INFO 5102: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i  
corresponding to Isolation Policy 'reset_sync' changed to value = 0 at time = 10784400  
ps.  
[MVSIM] INFO 5106: Isolation removed for Isolation Policy 'reset_sync' at time =  
10784400 ps.  
[MVSIM] INFO 5102: Isolation Enable signal test_bench.dut_inst.pwr_clamp_ctrl_i  
corresponding to Isolation Policy 'mac_off_blk' changed to value = 0 at time = 10784400  
ps.  
[MVSIM] INFO 5106: Isolation removed for Isolation Policy 'mac_off_blk' at time =  
10784400 ps.  
[MVSIM] INFO 5100: Simulation completed in 88319600 ps.  
=====  
MVSIM completed with 469 warning(s), 0 error(s).  
Total time = 1min 28.46sec CPU time = 39.58sec 44% CPU utilized  
Peak Memory = 262.80 MB
```

A

Workspace Directory and Files

This appendix provides information about the directories and files that get created when you configure the core tool (coreConsultant). This appendix contains the following sections:

- ❖ “[Workspace File Directory Structure](#)” on page [469](#)
- ❖ “[Workspace Directory and File Descriptions](#)” on page [475](#)

A.1 Workspace File Directory Structure

A.1.1 Directory Structures of Files (For Multiple Configurations)

After installation, you create a workspace where you generate, simulate, and synthesize your own configuration of the DWC Ether MAC 10/100/1000 Universal. [Figure A-1](#) shows the directory that is visible once coreConsultant is invoked and the configuration is created. Multiple configurations can be created, as shown in [Figure A-1](#).

Figure A-1 Directory Structure After Configuration

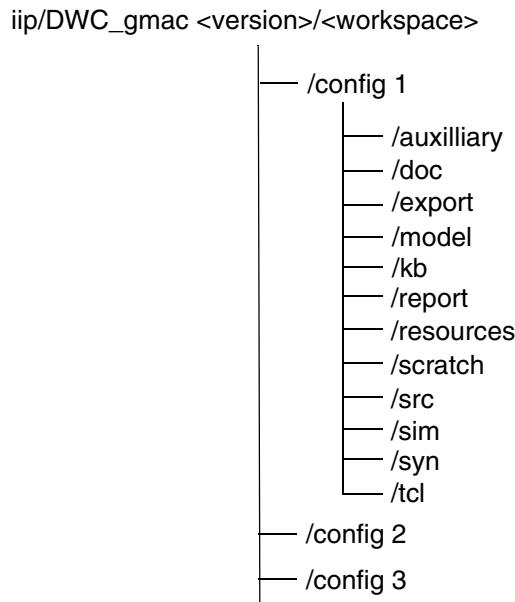
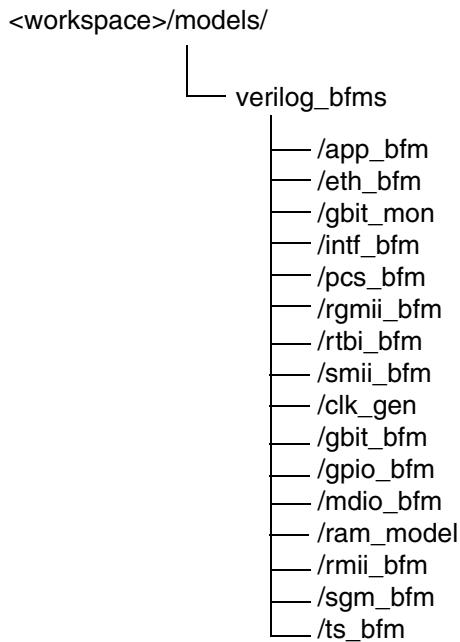


Figure A-2 shows the directories created in models. The models contain various bus functional models required for the test environment.

Figure A-2 Directory Structure for Models



The source directories (src/) are based on the selected configuration. For example, if you chose the GMAC-CORE option, directories mdc/, ahb/, axi/, and mtl/ do not appear. Directories ahb/, axi/, and mdc/ are not available for the GMAC_MTL option. Directories ahb/ and axi/ are not available for the GMAC-DMA option. Directories ahb/ and axi/ are available for GMAC-AHB and GMAC-AXI configurations respectively. Each of the PHY interface directories (sgmii/, rgmii/, rmii/, smii/, pcs/, revmii/ and rtbi/) is available only when enabled during configuration. For detailed descriptions of all configuration options and parameters, see “Parameters” on page 355.

Figure A-3 shows all source directories.

Figure A-3 Directory Structure for Source Directories

```
<workspace>/src/
    └── DWC_gmac_top_cc_constants.v
    └── DWC_gmac_top_params.v
    └── DWC_gmac_top-undef.v
    └── DWC_gmac_top.lst
    └── /mac
    └── /mdc
    └── /ahb
    └── /axi
    └── /common
    └── /mtl
    └── /pcs
    └── /sgmii
    └── /rgmii
    └── /rtbi
    └── /rmii
    └── /smii
    └── /revmii
    └── /top
```

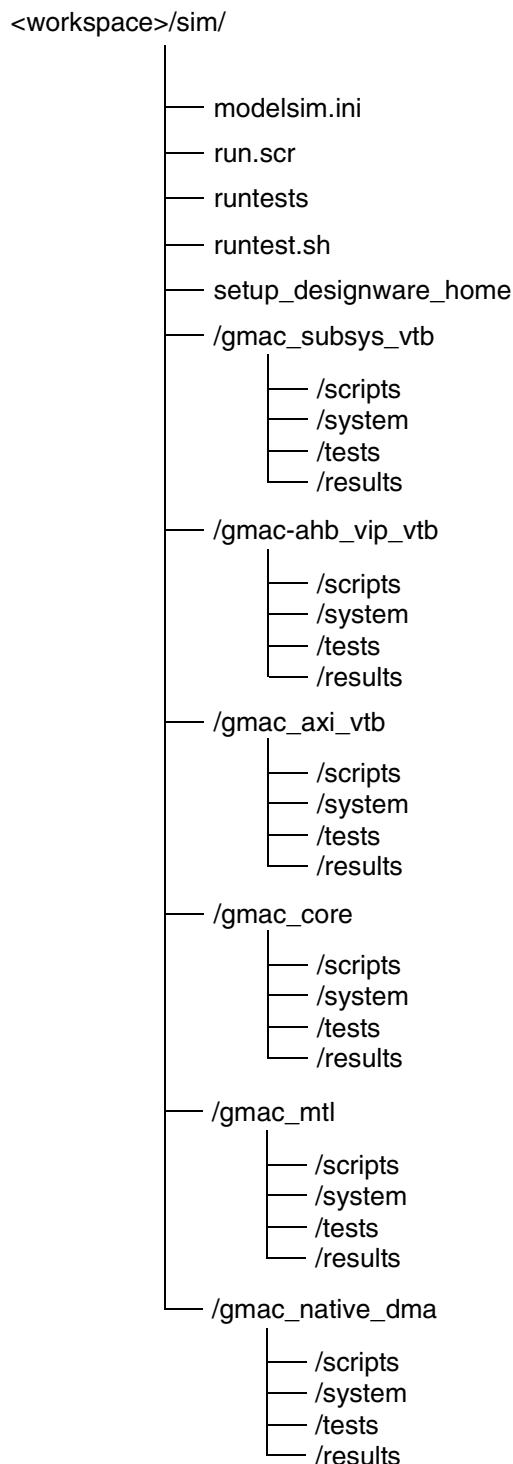
The `DWC_gmac_top.lst` file contains a list of files in compilation order. The files in this list are added based on the configuration selected. The top is the top-level directory that contains the top-level file `DWC_gmac_top.v`. The top-level design name is `DWC_gmac_top`.

`DWC_gmac_top_cc_constants.v` is a coreConsultant-generated file that contains parameter values and derived parameter values based on the parameter selection during configuration. For detailed descriptions of all configuration options and parameters, see “[Parameters](#)” on page 355.

The `DWC_gmac_top_params.v` contains the constant defines used in the RTL files.

A.1.2 Directory Structures for the Simulation Environment

All simulation (`sim/`) directory files are created during the `Simulate` activity of `coreConsultant`. The `sim/` directories are also created conditionally, based on the options selected. For example, the `gmac_subsys_vtb/` and `gmac-ahb_vip_vtb/` directories are created only when you chose AHB subsystem. The `run.scr` is the top-level run script. All of the simulation in command line mode should be run from the `sim/` directory. The `sim/` directory structure is shown in [Figure A-4](#).

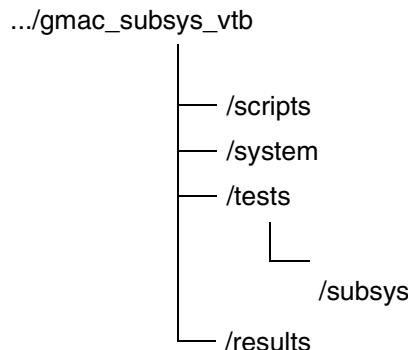
Figure A-4 Directory Structure for Simulation

The following list explains the directory structure shown in [Figure A-4](#):

- ❖ The runtest is a Perl file generated by coreConsultant to run various tests selected. This file also performs various tasks such as setting the environment variables for third-party simulators and providing pass/fail status for tests that are run.

- ❖ The runtest.sh file contains environment variable definitions such as VERA_HOME, VCS_HOME and the runtest command with different options based on the configuration.
 - ❖ The setup_designware_home file contains the paths of the DESIGNWARE_HOME and VERA_HOME environment variables.
 - ❖ The modelsim.ini file is the initialization file required for Modelsim.
 - ❖ One directory (gmac_axi_vtb/, gmac_subsys_vtb/ and gmac_ahb_vip_vtb/, gmac_native_dma/, gmac_core/, or gmac_mtl/) is created, as determined by the selected configuration (GMAC-AXI, GMAC-AHB, GMAC-DMA, GMAC-CORE, or GMAC-MTL).
- The gmac_subsys_vtb/ directory structure is shown in [Figure A-5](#). The gmac_ahb_vip_vtb/ and gmac_native_dma/ directory structures and file contents are similar to that of gmac_subsys_vtb/.
- ❖ The scripts directory contains the following files (only top-level, which can be modified are explained):
 - ◆ run_mti is a run script for ModelSim called by runall_subsys or runall_userlist.
 - ◆ run_vcs is a run script for VCS called by runall_subsys or runall_userlist.
 - ◆ run_ncv is a run script for NC Verilog called by runall_subsys or runall_userlist
 - ◆ runall_subsys is a generated script that has all the tests defined based on the selected configuration.
 - ◆ runall_userlist is a script containing a subset of the runall_subsys for a quicker check.
 - ❖ The system/ directory contains the testbench files.
 - ❖ The tests/ directory contains a subsys/ directory that contains test case files.
 - ❖ The results/ directory has a testlog/ directory where the results of the individual test cases are stored. The compiled result summary is stored as runtest.log in the sim/ directory.

Figure A-5 gmac_subsys_vtb Directory Structure

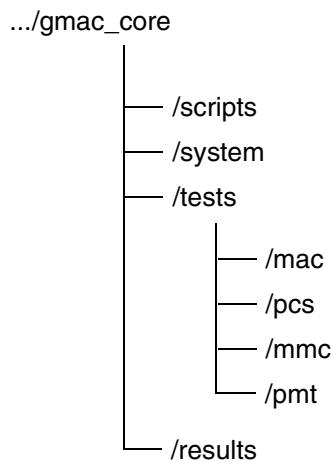


The gmac_core/ directory structure is shown in [Figure A-6](#). The system and the results directories are similar to what is explained above. The tests directory contains mac/, pcs/, mmc/, and pmt/ subdirectories, which are test case directories for MAC tests, PCS tests, RMON counter tests, and Power Management block tests, respectively.

The scripts directory contains the following important script files:

- ❖ run_mti, which is a run script for ModelSim called by runall_xxx or runall_userlist.
- ❖ run_vcs, which is a run script for VCS called by runall_xxx or runall_userlist.
- ❖ run_ncv, which is a run script for NC Verilog called by runall_core or runall_userlist.

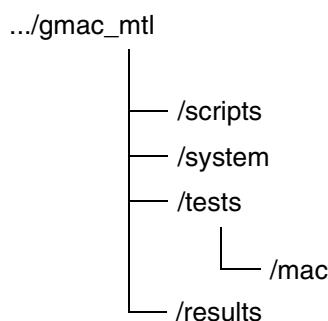
- ❖ runall_core, which is a script used to run all GMAC-CORE test cases.
- ❖ runall_userlist, which is a script that contains a subset of the runall_core tests for quick checks. The appropriate tests are run based on the selected interface.

Figure A-6 gmac_core Directory Structure

The gmac_mtl/ directory structure is shown in [Figure A-7](#). The system and the results directories are similar to what is explained for gmac_subsys_vtb. The tests contains the mac directory that contains all of the test cases.

The scripts directory contains the following files.

- ❖ run_mti is a run script for ModelSim called by runall_gmac_mtl or runall_userlist.
- ❖ run_vcs is a run script for VCS called by runall_gmac_mtl or runall_userlist.
- ❖ run_ncv is a run script for NC Verilog called by runall_gmac_mtl or runall_userlist.
- ❖ runall_gmac_mtl is a script that runs all of the MTL tests.
- ❖ runall_userlist contains a subset of the runall_gmac_mtl test for quick simulation.

Figure A-7 gmac_mtl Directory Structure

Note that all of the tests can be run through the coreConsultant GUI, and that the results are displayed on the GUI (a sample output is shown below). The same output is logged as a text file (runtest.log).

```
=====
Results of the Script Run on Date: 06:09:05 Time: 20:21:34
=====
Note: Running all tests using VCS
```

```

Start of GMAC MTL Half Duplex GMII Tests
DATE: 06/09/05 TIME : 20:21:47 PASSED : TEST_gth_lcra_APPCLK_TYP_____
DATE: 06/09/05 TIME : 20:21:57 PASSED : TEST_gth_fifo_full_APPCLK_TYP_____
DATE: 06/09/05 TIME : 20:22:08 PASSED : TEST_grh_ra_APPCLK_TYP_____
DATE: 06/09/05 TIME : 20:22:18 PASSED : TEST_grh_runt_drop_APPCLK_TYP_____
Start of GMAC MTL Full Duplex GMII Tests
DATE: 06/09/05 TIME : 20:22:58 PASSED : TEST_gtf_def_APPCLK_TYP_____
Start of GMAC MTL Half Duplex MII Tests
DATE: 06/09/05 TIME : 20:23:10 PASSED : TEST_mth_lcra_APPCLK_TYP_____
DATE: 06/09/05 TIME : 20:23:29 PASSED : TEST_mth_lcra_APPCLK_TYP_BASE10_____
DATE: 06/09/05 TIME : 20:23:39 PASSED : TEST_mrh_ra_APPCLK_TYP_____
DATE: 06/09/05 TIME : 20:23:51 PASSED : TEST_mrh_ra_APPCLK_TYP_BASE10_____
DATE: 06/09/05 TIME : 20:24:01 PASSED : TEST_mrh_runt_drop_APPCLK_TYP_____
DATE: 06/09/05 TIME : 20:24:12 PASSED : TEST_mrh_runt_drop_APPCLK_TYP_BASE10_____
Start of GMAC MTL Full Duplex MII Tests
DATE: 06/09/05 TIME : 20:25:31 PASSED : TEST_mtf_def_APPCLK_TYP_____
DATE: 06/09/05 TIME : 20:29:54 PASSED : TEST_mtf_def_APPCLK_TYP_BASE10_____

```

A.2 Workspace Directory and File Descriptions

[Table A-1](#) describes the contents of the various directories and files.

Table A-1 **Workspace Directory and File Descriptions**

Directory	Contents
/doc/README_POST_INSTALL/	Contains this core's readme file
/ip/DWC_gmac/<version>	Installed DWC_gmac files
<workspace>	<ul style="list-style-type: none"> • Top-level configured core directory • Created in coreConsultant during the Specify Configuration phase • Named by the user (for example, config1) • Contains user specified configuration information and supporting files
<workspace>/config#/	Contains the files for specified configurations. Created in coreConsultant during the Specify configuration phase (for example, config1, config2, and so forth).
/config#/doc/	<p>Contains the following documentation for the GMAC-UNIV</p> <ul style="list-style-type: none"> • Databook • Release Notes <p>Also contains links to the following documents:</p> <ul style="list-style-type: none"> • Quick Start • Installation Guide
/config#/auxillary/	coreConsultant directory common to all DesignWare products containing build and tool information.

Table A-1 Workspace Directory and File Descriptions (Continued)

Directory	Contents
/config#/export/	coreConsultant generated files <ul style="list-style-type: none"> • src – Configured source code • DWC_gmac_top.lst – List of source files in proper analysis order • DWC_gmac_top_inst.v – Verilog testbench template • DWC_gmac_top.db – DB format for the top design • DWC_gmac_top.v – Netlist of top design • DWC_gmac_top.sdc – SDC file for the net list • DWC_gmac_top.xml – IP-XACT component representation of the IP (XML format)
/config#/model/	Reserved for future use
/config#/kb/	.kb files used by coreConsultant to store data
/config#/models/verilog_bfms	Contains the bus functional models and monitors for the following <ul style="list-style-type: none"> • Application (app_bfm) • Clock generator (clk_gen) • Ethernet Interface (eth_bfm) • GBIT (gbit_bfm) • GBIT Monitor (gbit_mon) • GPIO (gpio_bfm) • PCS (pcs_bfm) • PHY Interfaces (intf_bfm) • RAM (ram_model) • RGMII (rgmii_bfm) • RTBI (rtbi_bfm) • RMII (rmii_bfm) • RevMII (mdio_bfm) • SGMII (sgm_bfm) • SMII (smii_bfm) • Timestamp BFM (ts_bfm)
/config#/report/	Simulation and synthesis report HTML files
/config#/resources/	Contains the following sub-directories <ul style="list-style-type: none"> • atpg: ATPG and Synopsys TetraMAX scripts • leda_reports: LEDA RTL check reports for 1 (example) configuration • pvc: Example source code and application note for implementing a PVCI/VCI interface for the GMAC-DMA core. • synth: Example Synopsys DC synthesis scripts for the default configuration.
/config#/scratch/	Temporary files generated and used by CoreConsultant
/config#/src/	coreConsultant configured source RTL files
/config#/sim/	Scripts and files generated for simulations and parsing the results

Table A-1 Workspace Directory and File Descriptions (Continued)

Directory	Contents
/config#/syn/	Synthesis strategy scripts and Design compiler output files. coreConsultant controls all operations with these directories and files.
/config#/tcl/	Synthesis input files for coreTools

B

GMAC-AHB Verification Environment Using AMBA and Ethernet VIPs

This appendix describes the example Verilog Testbench (VTB) used to test the GMAC-AHB using AMBA Verification IP (VIP) and Ethernet Verification IP. The VTB provides a starting point for understanding how to use AMBA Verification IP (VIP), Ethernet VIP, and configured DWC Ethernet core (configured for an AHB system interface) together in the verification environment. This appendix also provides guidelines for modifying the test environment to use in SoC-level verification.

This appendix contains the following sections:

- ❖ “[VTB Overview](#)” on page [479](#)
- ❖ “[Verification Flow](#)” on page [481](#)
- ❖ “[Directory Structure](#)” on page [489](#)
- ❖ “[GPIO](#)” on page [490](#)
- ❖ “[SoC-Level Verification Guidelines](#)” on page [492](#)
- ❖ “[Running Simulations](#)” on page [493](#)
- ❖ “[Simulation PASS/FAIL](#)” on page [493](#)

B.1 VTB Overview

The example VTB demonstrates the following:

- ❖ How to connect the DesignWare AMBA VIP, Ethernet VIP, and DWC Ethernet (RTL) synthesizable components in a VTB.
- ❖ How to use the AMBA VIP and Ethernet VIP with the DWC Ethernet (RTL) synthesizable component to perform basic operating functions.

The system uses AMBA VIP on the application side and Ethernet VIP on the Ethernet side. The VIPs have built-in tasks to perform various operations that verify the DUT. For more details about Verification IP, refer to the following documents:

- ❖ *DesignWare AHB Verification IP Databook*, Synopsys Inc.
<https://www.synopsys.com/dw/doc.php/vip/amba/latest/doc/ahbvipdb.pdf>
- ❖ *DesignWare Ethernet Verification IP User Manual*, Synopsys Inc.
https://www.synopsys.com/dw/doc.php/vip/ethernet/latest/doc/ethernet_vmt_user.pdf

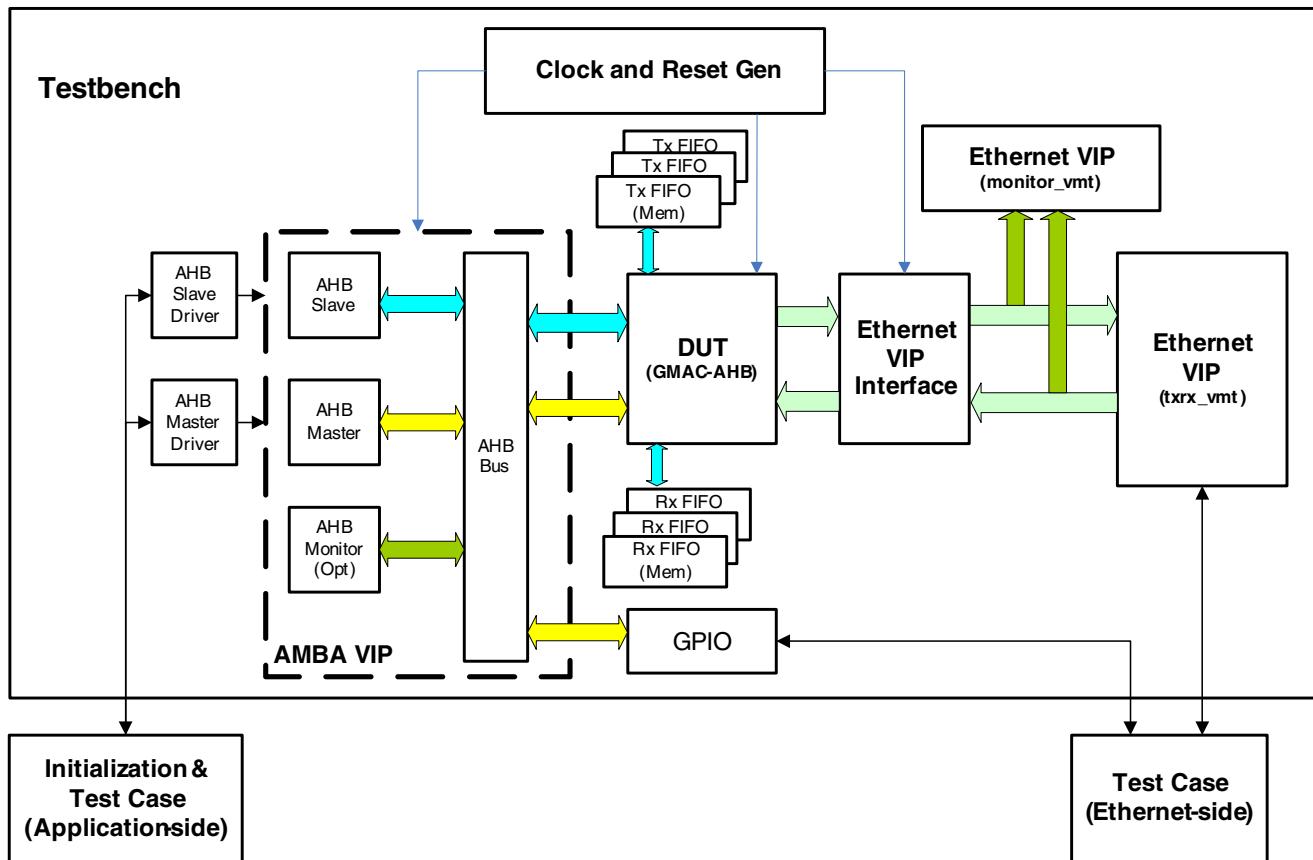
- ❖ VMT User's Manual, Synopsys Inc.

<https://www.synopsys.com/dw/doc.php/vip/vmt/latest/doc/vmtum.pdf>

These documents are also included in the DWC Ethernet image.

The basic structure of VTB is shown in [Figure B-1](#). The VTB supports use of the MII, RMII, GMII, RGMII, TBI, or SGMII to connect the DUT and Ethernet VIP. VTB is made compatible to SoC-level verification by constraining the application to communicate with Ethernet VIP only through general-purpose I/O (GPIO).

Figure B-1 GMAC-AHB_VIP Verification Environment



The testbench consists of the following blocks:

- ❖ **DUT:** The device under test (DWC Ethernet core). The VTB instantiates the core configured to AHB configuration.
- ❖ **AHB VIP:** The DW AMBA Bus VIP (`ahb_bus_vmt`), which is an AHB bus fabric (arbiter and decoder) that can connect up to 15 additional masters and slaves.
- ❖ **AHB Master VIP:** The DW AMBA VIP (`ahb_master_vmt`), an AHB master that can perform reads and writes to the slaves currently in the system
- ❖ **AHB Slave VIP:** The DW AMBA VIP (`ahb_slave_vmt`), an AHB slave that interfaces to a Host RAM where data resides. The test case commits writes to memory by means of back-door commands, not through actual AHB transactions.
- ❖ **Master and Slave Drivers:** A collection of bus-independent tasks called from the Test - Configuration and Control block

- ❖ GPIO: All communication between the application-end and Ethernet-end test cases is routed only through the GPIO block, making the VTB reusable for SoC-level verification. For further details on the GPIO block, see “[Verification Flow](#)” on page [481](#).
- ❖ Ethernet VIP Interface: This block connects DUT ports to corresponding Ethernet VIP ports. Depending on the specified interface (MII, RMII, GMII, RevMII, RGMII, TBI, or SGMII), the Ethernet VIP permits only specific ports to be connected, while leaving other VIP ports unconnected as required.
- ❖ Ethernet VIP Monitor: This block passively monitors Ethernet transactions, generates reports about transactions, and logs either all or selected behavior.
- ❖ Ethernet VIP Transceiver model: A transceiver model with a command-based interface that transmits and receives correct and erroneous traffic on the supported interfaces (MII, RMII, GMII, RevMII, RGMII, TBI, or SGMII).
- ❖ Clock and Reset Gen: This block generates the clock and reset signals the VTB requires for each interface.
- ❖ Test case (application-side): The application-side test case initializes the VTB and controls the application side of the DUT. This also consists of a set of tasks that can be used to control the AMBA VIP. For SoC-level verification, this set of test cases can be rewritten in a high-level language, such as C.
- ❖ Test case (Ethernet-side): This set of test cases controls the Ethernet VIP using predefined tasks. This block communicates with the other VTB blocks only through GPIO.

B.2 Verification Flow

The basic verification flow is explained in this section. The transmission-side and reception-side verification flows are split, the first half pointing to the application side, the second to the Ethernet VIP side.

All communications and handshakes occur between the application and line sides through the GPIO port on the AMBA bus in the testbench. The GPIO port bits are detailed in “[GPIO](#)” on page [490](#)

B.2.1 Transmission Verification Flow

B.2.1.1 Application-Side Test Case Flow

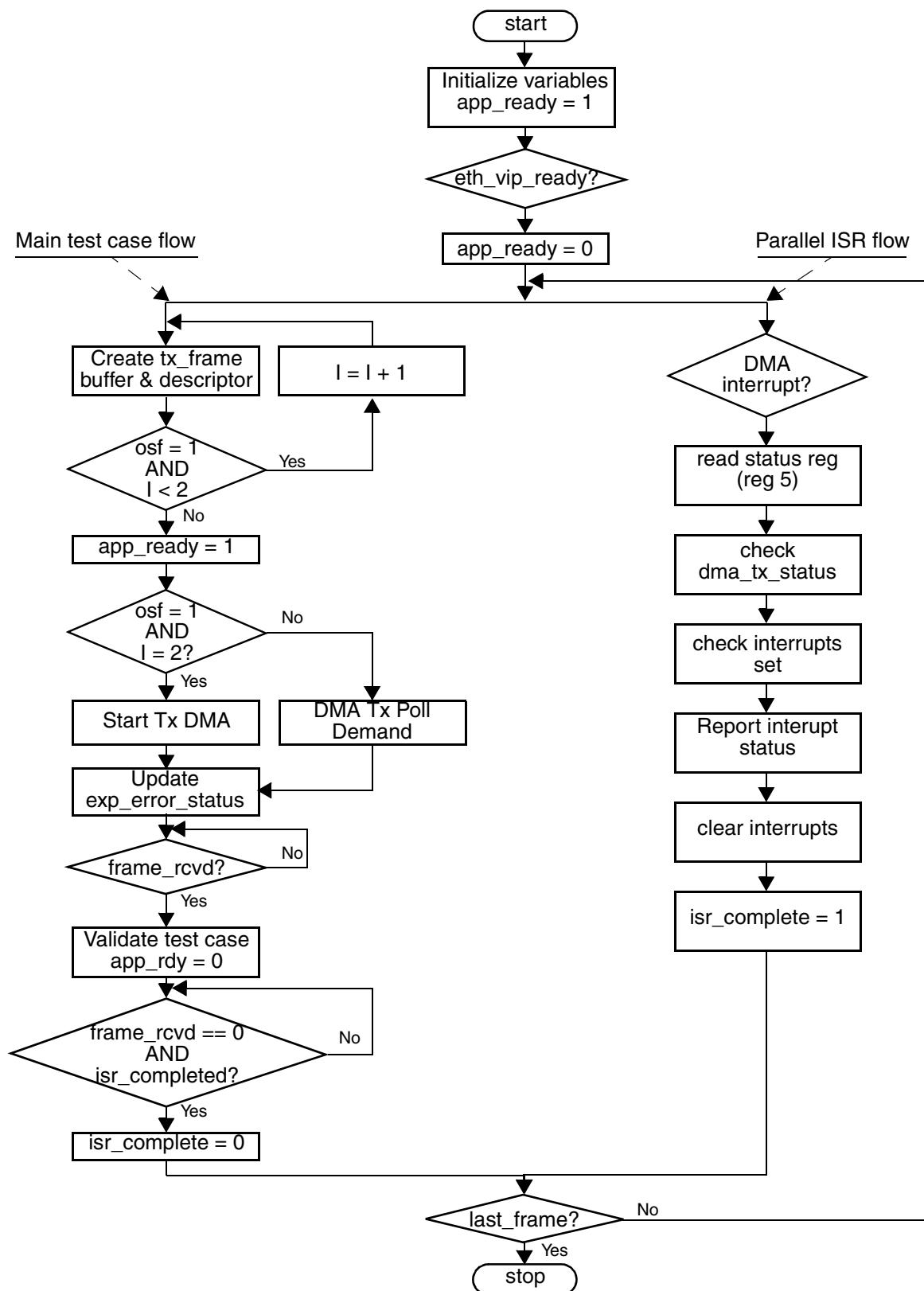
The basic data flow at the host end for verification of the GMAC subsystem (transmission) is as depicted in [Figure B-2](#). The flow is as follows:

1. The application initiates the data transfer by programming the MAC, then initializing all the application-side variables used by the test case.
2. The application-side test case and the Ethernet-side test case perform a handshake to synchronize. The application-side test case sets the app_ready bit in the GPIO, then reads the GPIO to check the eth_vip_ready bit's setting. The Ethernet-side test case sets the eth_vip_ready bit after it completes its initialization. The application then clears the app_ready bit.
3. The application-side test case creates a transmit descriptor and sets the app_ready bit in the GPIO to indicate that a frame is pending for transmission.
4. DUT starts the frame transmission and initialize the expected transmission status in the internal exp_error_status variable. You need to program the DUT by setting Start TxDMA or Tx Poll Demand.
5. The host waits for the Ethernet VIP to set the frame_rcvd bit in the GPIO.

6. When the Ethernet VIP sets the frame_rcvd bit, the application-side test case reads rcvd_error_status from the GPIO and compares it to exp_error_status.
7. The test case passes if exp_error_status and rcvd_error_status match, and if the Payload Check Status bit (exp_data_rcvd) in the GPIO is set.
8. The host clears the app_ready bit and waits for the Ethernet VIP to clear the frame_rcvd bit.
9. The host waits for the setting of the isr_complete bit, then clears it before transmitting the next frame. This step synchronizes the main loop of the test case with the Interrupt Service Routine loop of the test case (shown in [Figure B-2](#) on page 483) for every frame transmitted.
10. If more frames are to be transmitted, the process returns to [Step 2](#), continuing through [Step 9](#).
11. If the OSF bit is set, the host creates two descriptors before transmitting the first frame.

In [Figure B-2](#) on page 483, a second thread or process (on the right side) is executed in parallel with the above steps. This is the emulation of the Interrupt Service Routine (ISR) in verilog. You must port the code given in the second thread to the ISR. The isr_complete variable is used to handshake and synchronize the application-side testcase routines inside and outside of the ISR in this flow. This loop is triggered whenever the DUT asserts an interrupt. The ISR reads the status registers, checking whether the proper and expected interrupt bits are set, then clears the interrupt and waits for the next interrupt.

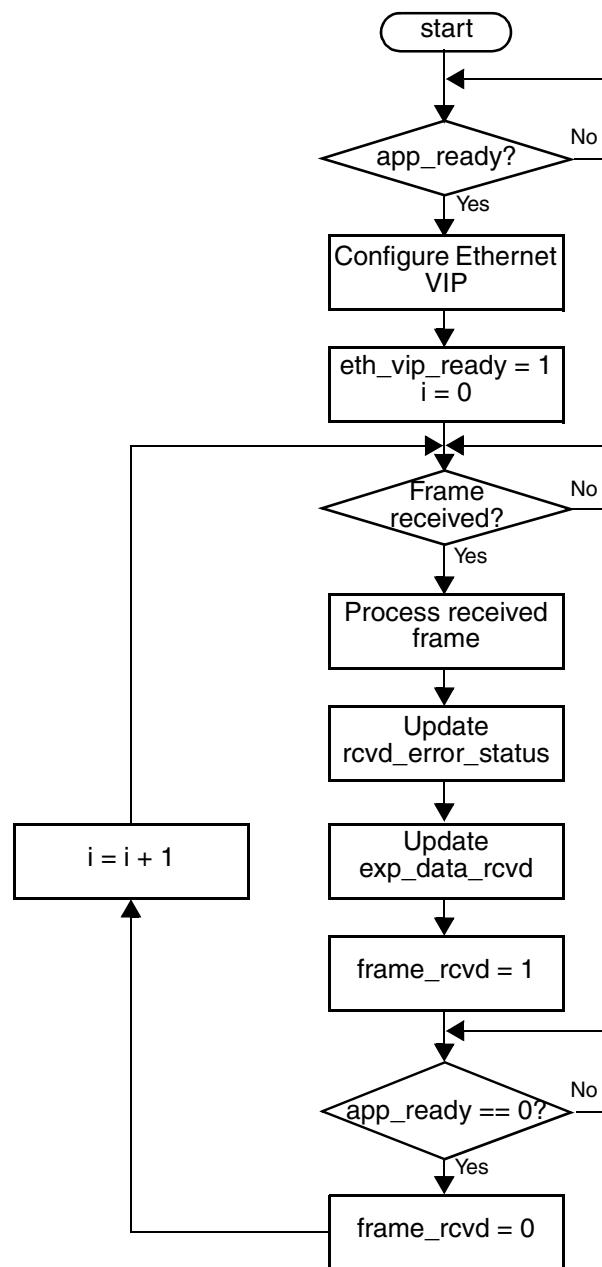
Similarly, the frame_rcvd variable is used as handshake to synchronize the routines in the application-side test case and the Ethernet-side test case (as explained in “[Ethernet-Side Test Case Flow](#)” on page 484).

Figure B-2 Application-Side Test Case Flow (Tx)

B.2.1.2 Ethernet-Side Test Case Flow

The basic test case flow at the Ethernet VIP (transmission) is as shown in [Figure B-3](#). The flow is as follows:

1. The Ethenet_VIP test case checks for the app_ready bit to be set
2. When the app_ready bit is set, the test case configures the Ethernet VIP and sets the eth_vip_ready bit.
3. The Ethernet VIP waits for the frame. When it receives the frame, the test case processes it, checking for any errors the Ethernet VIP may have reported.
4. The corresponding error status is set in rcvd_error_status and exp_data_rcvd bit is set if there is no mismatch in the expected and received payload.
5. The frame_rcvd bit is set, until the host clears the app_ready bit in the GPIO.
6. The Ethernet VIP returns to [Step 3](#)

Figure B-3 Ethernet-Side Test Case Flow (Tx)

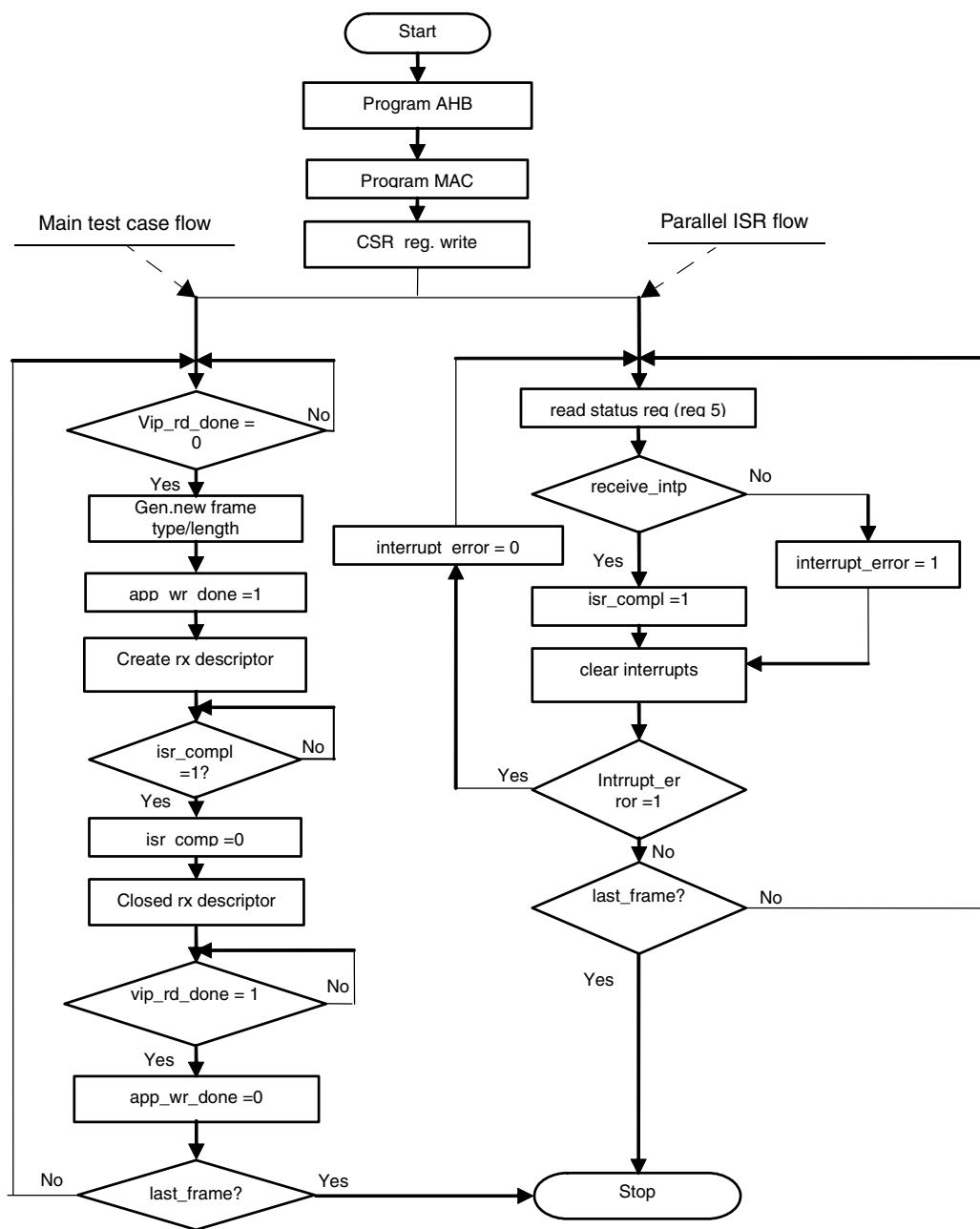
B.2.2 Receive Verification Flow

B.2.2.1 Host-Side Test Case Flow (Rx)

The basic data flow at the host end (receiving) for the verification of GMAC subsystem is depicted in [Figure B-4](#). The flow is as follows:

1. The application configures the AHB VIP and MAC and initializes all CSRs.
2. The application waits until the `vip_rd_done` bit in the `GPIO_IN` register is cleared.
3. The application determines the frame's type and length and writes the frame type and length to the appropriate `GPIO_OUT` fields.

4. After writing all information, the application sets the app_wr_done bit.
5. The application creates an Rx descriptor of the same frame length it wrote to GPIO_OUT.
6. The application waits for a Receive interrupt by checking whether the isr_complete variable is set. When the interrupt occurs, the application clears the isr_complete variable and moves to [Step 7](#). This step synchronizes the main loop with the Interrupt Service Routine (ISR) loop for each frame. The DUT's interrupt assertion triggers the ISR loop, which checks whether the expected interrupt is asserted. If the expected interrupt is not asserted, the ISR sets the interrupt_error variable, clears all interrupts, then waits for the next one.
7. After closing the descriptor task, the application reads the GPIO_IN register until it finds the vip_rd_done bit set, then clears the app_wr_done bit in GPIO_OUT.
8. The application repeats [steps 2](#) through [7](#) for the next frame, until the application-side test case sets the last_frame bit in the GPIO.

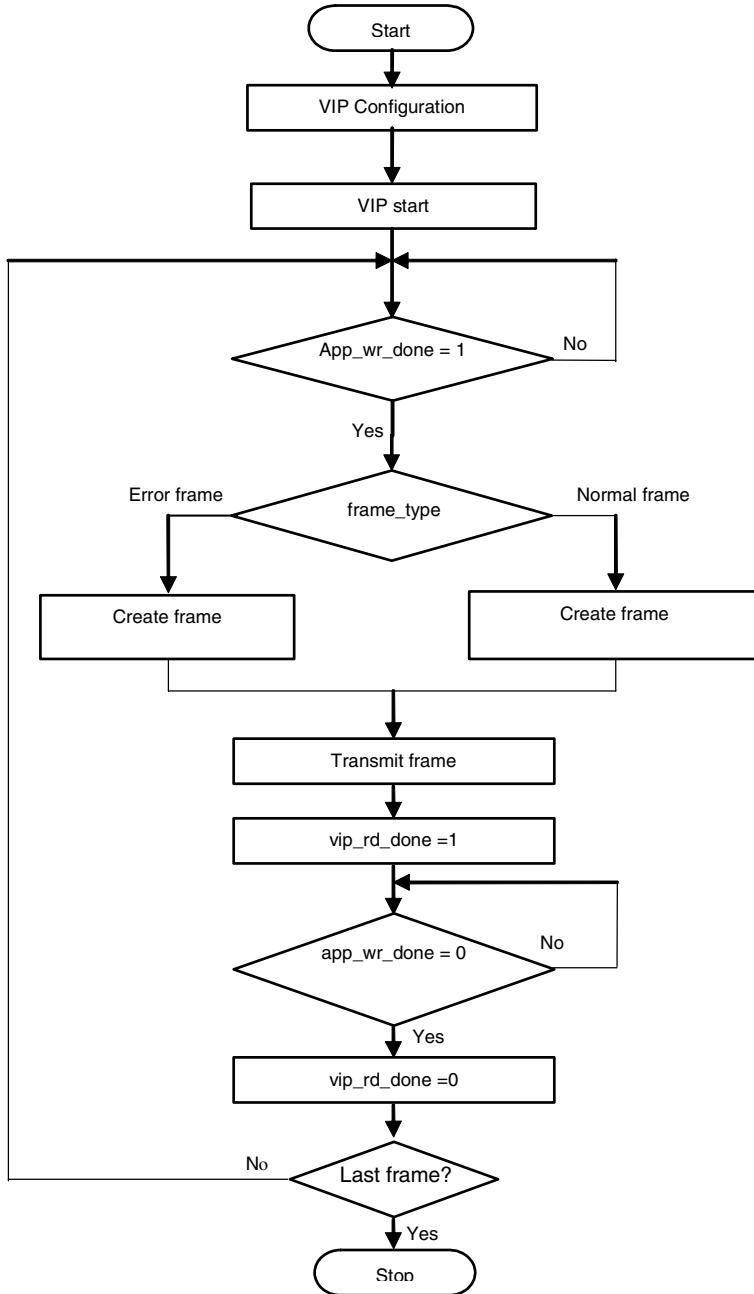
Figure B-4 Application-Side Test Case Flow (Rx)**B.2.2.2 Ethernet-Side Test Case Flow (Rx)**

The basic test case flow at the Ethernet VIP (receiving) side is depicted in [Figure B-5](#). The flow is as follows:

1. The VIP is configured, and then started.
2. The VIP waits for an application write task on `GPIO_OUT`.
3. When `app_wr_done` is set, the VIP reads the current frame information from `GPIO_OUT`, creates a frame based on that information, and transmits it to DUT.
4. After transmitting a new frame, the VIP sets `vip_rd_done` on `GPIO_IN`, waits for `app_wr_done` to be cleared, then clears the `vip_rd_done` signal.

5. Steps 2 through 4 are repeated for the next frame, until the application-side test case sets the Last Frame bit.

Figure B-5 Ethernet-Side Test Case Flow (Rx)



B.3 Directory Structure

Figure B-6 gmac-ahb_vip_vtb Directory Structure

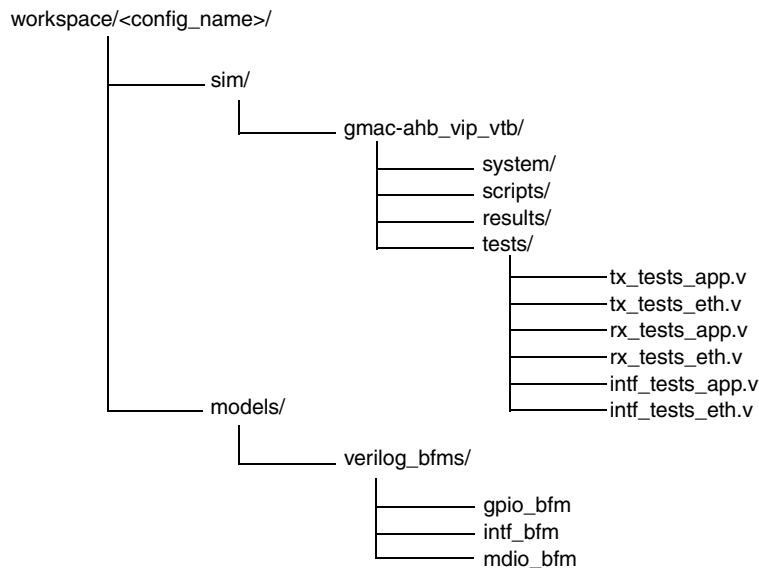


Figure B-6 shows the organization of testbench files, BFM s, and test vectors for the gmac-ahb_vip. The <config_name> directory contains the configured RTL and the testbench. Other directories are explained in Table B-1.

Table B-1 gmac-ahb_vip_vtb Directory Structure

File Name	Description
sim	Simulation directory containing the testbench, test vectors, and scripts. All simulations are run from this directory
system	Directory containing the testbench, the clock generation module, and the Ethernet VIP tasks, host tasks, host commands, and so forth.
scripts	Directory containing scripts for running test vectors.
results	This directory's test_log subdirectory is a repository for simulation logs.
tests	This directory contains all the transmit and receive test vectors. Test case files with an “_app” suffix contain the application-side code. Test case files with an “_eth” suffix control the Ethernet VIP.
gpio_bfm	This directory contains a GPIO model that the application and Ethernet sides use to communicate.
intf_bfm	This consists of an interface file used for proper port mapping between the DUT and the Ethernet VIP, depending on the selected interface.
mdio_bfm	This directory contains the MDIO bfm and MDIO read-write tasks files for the RevMII model.

B.4 GPIO

The GPIO is used for communication between the host and the Ethernet VIP. The GPIO has two types of register sets for input ports (GPIO_IN) and output ports (GPIO_OUT). The register set is further classified by whether the test case controls frame transmission or reception.

The GPIO base address in the testbench is 0x1000_C000. The GPIO BFM implements a set of sixteen 32-bit registers to control the I/O ports, with each register bit corresponding to a port bit. The Bit 5 address indicates whether the register is used by the transmit test case (Bit 5 = 0) or the receive test case (Bit 5 = 1). Similarly, Bit 4 of the address decodes whether the register controls the output ports (Bit 4 = 0) or the input ports (Bit 4 = 1).

The address mapping for GPIO is as follows:

Table B-2 GPIO Address Map

Register	Address	Description
1	1000_C000	Transmission-side GPIO_OUT control register
2–4	1000_C004–1000_C00C	Reserved for transmission-side GPIO_OUT
5	1000_C010	Transmission-side GPIO_IN control register
6	1000_C014	Transmission-side GPIO_IN error_status [63:32]
7	1000_C018	Transmission-side GPIO_IN error_status [31:0]
8	1000_C01C	Reserved for Transmission-side GPIO_IN
9	1000_C020	Receive-side GPIO_OUT control register
10–12	1000_C024–1000_C02C	Reserved for Receive-side GPIO_OUT
13	1000_C030	Receive-side GPIO_IN control register
14–16	1000_C034	Reserved for Receive-side GPIO_IN

B.4.1 GPIO_OUT (Tx)

The GPIO_OUT structure for transmission is shown in [Figure B-7](#).

Figure B-7 GPIO_OUT Structure (Tx)



B.4.2 GPIO_IN(Tx)

The GPIO_IN structure for transmission is shown in [Figure B-8](#), GPIO_IN uses three 32-bit registers: one for the control variables, the other two for the error status received from the Ethernet VIP.

GPIO_IN fields are as follows:

- ❖ eth_vip_ready: This bit is set when the Ethernet VIP is ready to receive the frame.
- ❖ frame_rcvd: This bit is set when the Ethernet VIP has received the frame.

- ❖ Payload check status: The VIP sets this bit when it receives data that matches the expected data frame.
- ❖ rcvd_error_status[63:0]: The 64-bit error status generated by the Ethernet VIP for the received frame is stored in two 32-bit registers, as shown [Figure B-8](#).

Figure B-8 **GPIO_IN Structure (Tx)**

eth_vip_ready [31]	frame_rcvd [30]	Payload check status [29]	Reserved [28:0]
rcvd_error_status [63:32]			
rcvd_error_status [31:0]			

B.4.3 GPIO_OUT (Rx)

The GPIO_OUT structure is shown in [Figure B-9](#).

The fields in GPIO_OUT (Rx) are:

- ❖ app_wr_done: After writing all information, the host side sets this bit, which VIP requires to create a new frame for transmission.
- ❖ last_frame: The host sets this bit when the last frame is being processed.
- ❖ frame_type: These bits [29:26] indicate the type of frame that the VIP must create for processing. The types of different error the VIP can insert in the frame are based on these bits. Details of these bits are presented in [Table B-3](#)
- ❖ frame_length [13:0]: These bits indicate the length of the frame.

Table B-3 **Frame Type Description**

GPIO[29:26]	Frame Type
0000	Normal frame
0001	CRC error frame
0010	Destination Address Filter error frame
0011	Checksum error

Figure B-9 gpio_out Structure (Rx)

app_wr_done[31]	last_frame[30]	frame_type[29:26]	Reserved[25:14]	frame_length[13:0]
-----------------	----------------	-------------------	-----------------	--------------------

B.4.4 GPIO_IN (Rx)

As shown in [Figure B-10](#), the GPIO_IN structure uses 32-bit registers of which it currently uses only the following port:

- ❖ vip_rd_done: The Ethernet VIP sets this bit when it has read out GPIO_OUT.

Figure B-10 gpio_in Structure (Rx)

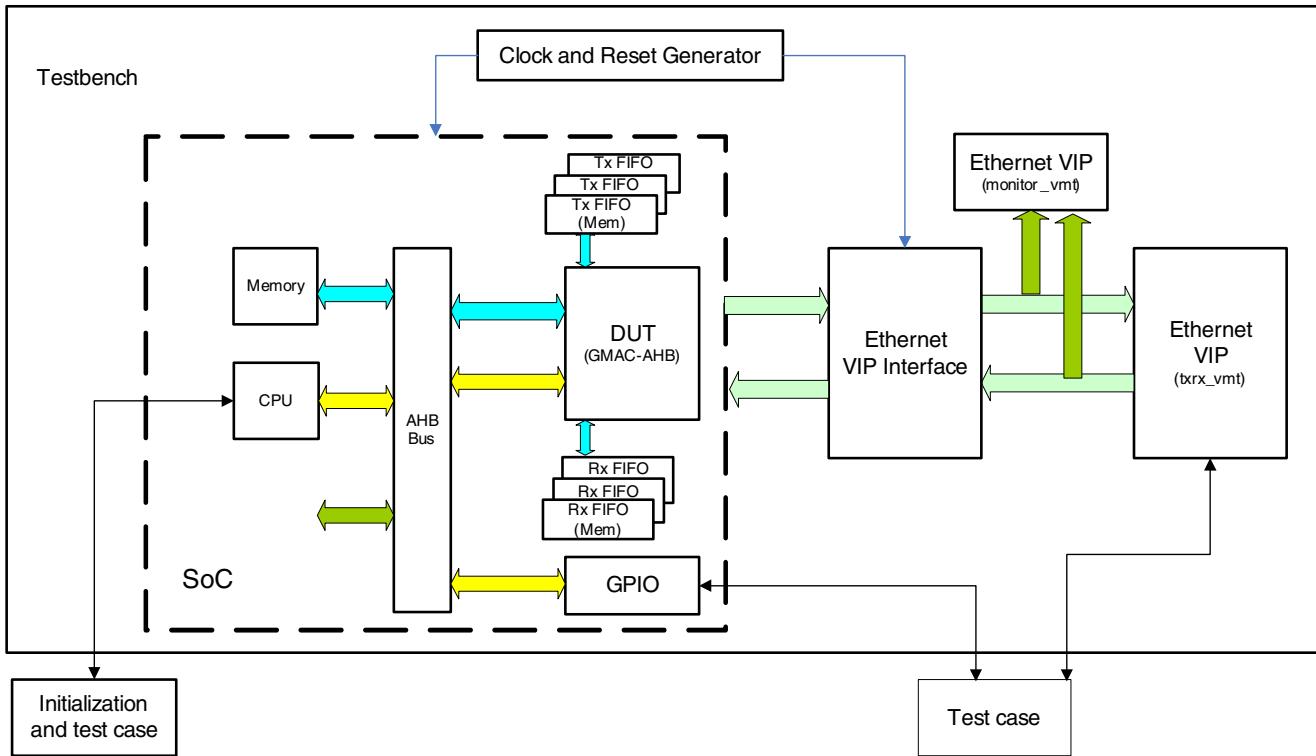
vip_rd_done[31]	Reserved[30:0]
-----------------	----------------

B.5 SoC-Level Verification Guidelines

This section provides guidelines for SoC-level verification. The test environment is depicted in [Figure B-11](#). In this environment, the SoC, which contains an ARM core (CPU) is the DUT. The application-side vectors are now created in software and loaded into the memory for the CPU to execute.

The verilog testcases given in this testbench can be easily ported to create test cases that can be used to verify SoC integration. The verification strategy must adhere to the following guidelines.

- ❖ In SoC-level verification, you must ensure that all communication between the application-side test case and the Ethernet-side test case passes through the GPIO only.
- ❖ You must rewrite the application-side test case in a high-level language, such as C, and compile it to create machine code. Load this code into the system memory (instruction code memory) of the SoC, then start the simulation. The CPU reads the instructions from memory and executes the test as indicated in the flow charts provided in “[Verification Flow](#)” on page 481.
- ❖ The Ethernet VIP is controlled only with the existing Ethernet-side test case in verilog.
- ❖ You can use the reserved fields in the GPIO to pass control data between the application and the Ethernet VIP.

Figure B-11 Setup for SoC-Level Verification

B.6 Running Simulations

To run the individual test case simulations from the command line, enter the following commands from the configured workspace: <config_name>/sim/ directory:

- ❖ For VCS, enter:

```
./gmac-ahb_vip_vtb/scripts/run_vcs <TEST_CASE_NAME> <OTHER_OPTIONS>
```



To run the receive-side test case, you must pass the RX_TEST_EN variable from the command line. The details of the <TEST_CASE_NAME> and <OTHER_OPTIONS> can be obtained from the ./gmac-ahb_vip_vtb/scripts/runall_subsys file.

You can use the logged output of the run (the verilog.log file) to check for errors.

- ❖ For MTI, enter:

```
./gmac-ahb_vip_vtb/scripts/run_mti <TEST_CASE_NAME> <OTHER_OPTIONS>
```

- ❖ For NCV, enter:

```
./gmac-ahb_vip_vtb/scripts/run_ncv <TEST_CASE_NAME> <OTHER_OPTIONS>
```

B.7 Simulation PASS/FAIL

The simulation run status is logged to the <config_name>/sim/runttest.log file. You can access the details of the run from the <config_name>/sim/gmac-ahb_vip_vtb/results/testlog/<TEST_NAME>.log file at the end of the simulation.

C

GMAC-AHB Verification Environment

This appendix describes the example Verilog Testbench (VTB) used to test GMAC-AHB and explains how to use it.

This appendix contains the following sections:

- ❖ “[VTB Overview](#)” on page [495](#)
- ❖ “[Initialization](#)” on page [496](#)
- ❖ “[Using VTB](#)” on page [497](#)
- ❖ “[Tool and Setup Requirements](#)” on page [499](#)
- ❖ “[Running Simulations](#)” on page [499](#)
- ❖ “[Simulation PASS/FAIL](#)” on page [500](#)

C.1 VTB Overview

The VTB is packaged in the DWC Ethernet image. The VTB provides a starting point for understanding how to use DesignWare AMBA Verification IP (VIP), Verilog BFM models, and the configured DWC Ethernet core (configured for AHB system interface) together in the verification environment.

The example VTB demonstrates the following:

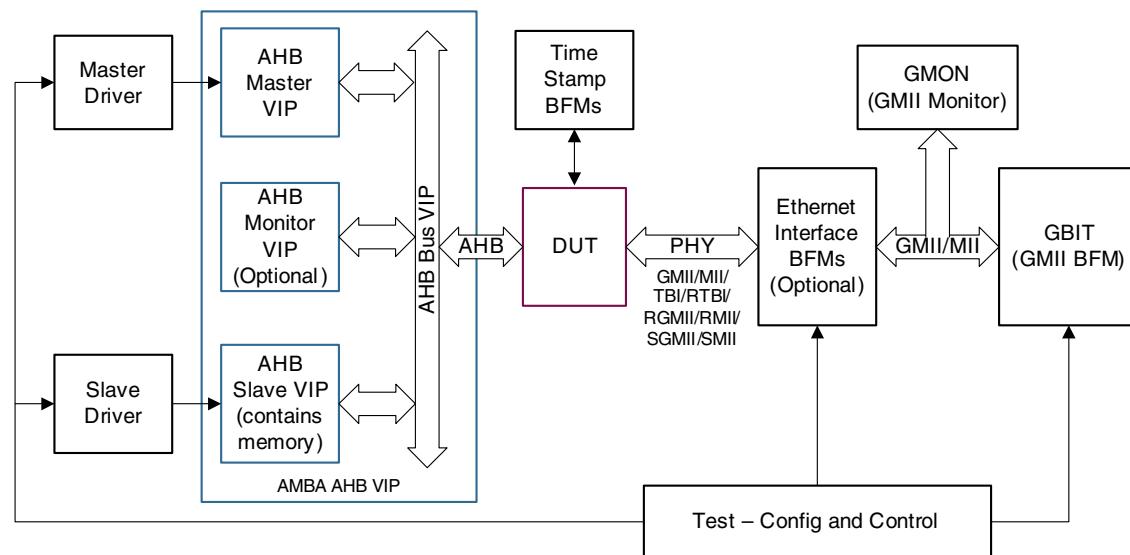
- ❖ How to connect the DesignWare AMBA VIP, Verilog BFMs, and DWC Ethernet (RTL) synthesizable component in a VTB
- ❖ How to initialize and program (using Verilog VIP commands) the DWC Ethernet to perform basic operating functions.

On the Ethernet side, the DWC Ethernet DUT provides a GMII/MII/RMII/RGMII/SGMII/TBI/RTBI/SMII interface. On the application side, it interfaces to an AHB. The VIP is used in the VTB as regular Verilog instances. These VIP have their own built-in tasks that can be used just as any regular Verilog task in the VTB.

For more details about VIP, refer to the following documents.

- ❖ *DesignWare AHB Verification IP Databook*, Synopsys Inc.
<https://www.synopsys.com/dw/doc.php/vip/amba/latest/doc/ahbvipdb.pdf>
- ❖ *VMT User's Manual*, Synopsys Inc.
<https://www.synopsys.com/dw/doc.php/vip/vmt/latest/doc/vmtum.pdf>

These documents are also included in DWC Ethernet image.

Figure C-1 GMAC-AHB Verification Environment

The testbench consists of the following blocks:

- ❖ DUT: The device under test – The DWC Ethernet core. The VTB instantiates the configured core.
- ❖ AHB Bus VIP: The DW AMBA VIP (ahb_bus_vmt) is used, which is an AHB bus fabric (arbiter and decoder) that can connect up to 15 additional masters and slaves.
- ❖ AHB Master VIP: The DW AMBA VIP (ahb_master_vmt) is used, which is an AHB Bus Master that can perform reads and writes to the slaves presently in the system.
- ❖ Master/Slave Driver: Is a collection of bus independent tasks called from the Test - Config and control block.
- ❖ AHB Slave VIP: The DW AMBA VIP (ahb_slave_vmt) is used, which is an AHB Bus Slave that interfaces to a Host RAM where data resides. The writes to memory occur by means of back-door commands and not through actual AHB bus transactions.
- ❖ Ethernet Interface BFM: Translates the interface specific protocol (SGMII/TBI etc.) to the GMII/MII and feeds the GBIT BFM, and vice-versa.
- ❖ GMII Monitor: Monitors the protocol on the GMII/MII bus.
- ❖ GBIT BFM: Transmits/receives the Ethernet frames as per the control of TEST - Config and Control block.
- ❖ Time Stamp BFM: Contains the time stamping BFM files. Captures time stamp values at GMII for verification, and generates time stamps if the External Time Stamp option is selected.
- ❖ TEST - Config and Control: This block is the main test block that configures the VIP and performs the actual tests. It also includes clock/reset blocks. The main testbench control is coded in a simple verilog format. The tests in the VTB support all three speed modes (1000, 100 and 10 Mbps), in Full- and Half-Duplex modes.

C.2 Initialization

The following sequential steps describe how the DWC Ethernet is initialized and how the VIP is set up in the VTB.

Restart, Start, and Initialization:

1. Perform a hardware power-on reset to the DWC Ethernet (DUT).
2. Configure the VIP (ahb_master, ahb_slave, ahb_bus, and ahb_monitor) and start them. This is done by calling the ahb_vip_init task.
3. Configure the DUT. The default configuration is programmed by the task program_mac_default.
4. Run the specific scenario of the test case. (For example, see the tf_basic test case in <config_name>/sim/gmac_subsys_vtb/tests/subsys/tx_tests.v file.)

C.3 Using VTB

This section briefly describes the files used in the VTB.

Directory Structure and Files:

All files related to this VTB environment are under the configured workspace:

- ❖ <config_name>/sim/gmac_subsys_vtb/system
- ❖ <config_name>/sim/gmac_subsys_vtb/scripts.

The test case files are under the configured workspace

- ❖ <config_name>/sim/gmac_subsys_vtb/tests/subsys.

[Table C-1](#) describes the VTB source files.

Table C-1 GMAC-AHB Verification Environment Directory Structure and Files

File Name	Description
sim/gmac_subsys_vtb/system/test_bench.v	Main testbench Verilog module that has all instantiations, testbench control, tests, and tasks
sim/gmac_subsys_vtb/system/ahb_vip_init.v	AHB initialization task
sim/gmac_subsys_vtb/system/clk_div.v	Clock divider module
sim/gmac_subsys_vtb/system/clk_pll.v	Clock generator module
sim/gmac_subsys_vtb/system/eth_commands.v	This file includes all test files and the mac init tasks
sim/gmac_subsys_vtb/system/eth_host_tasks.v	All tasks required to control the application and Ethernet side are defined here
sim/gmac_subsys_vtb/system/master_driver.v	This file has the memory read and write tasks to the DUT
sim/gmac_subsys_vtb/system/mac_init_tasks.v	This file configures the MAC core in the DUT and the GBIT BFM
sim/gmac_subsys_vtb/scripts/mon_cmds.v	Controls certain parameters in the GBIT BFM
sim/gmac_subsys_vtb/scripts/timescale.v	Defines the time scale for the simulation
sim/gmac_subsys_vtb/scripts/dump_incl.v	In post processing mode, the user can put commands in this file
sim/gmac_subsys_vtb/tests/subsys/framex.dat	Data in this file is used by the Rx Test cases (rgf_ipc) to check the frames with Header IP checksum

Table C-1 GMAC-AHB Verification Environment Directory Structure and Files (Continued)

File Name	Description
sim/gmac_subsys_vtb/tests/subsys/gtf_ipc_xx_xx.dat	Data in this file is used by the Tx Test cases (gtf_ipc) to send the IPv4/IPv6 frames
sim/gmac_subsys_vtb/tests/subsys/rgf_ipc_xx_xx.dat	Data in this file is used by the Rx Test cases (rgf_ipc) to receive the IPv4/IPv6 frames
sim/gmac_subsys_vtb/tests/subsys/grh_mp2.dat	Data in this file is used by the Rx Test case (rgf_power_down_mp) to receive Magic Packets.
sim/gmac_subsys_vtb/tests/subsys/intf_tests.v	Has the test for different PHY interfaces
sim/gmac_subsys_vtb/tests/subsys/rx_tests.v	Has all the Rx tests
sim/gmac_subsys_vtb/tests/subsys/tx_tests.v	Has all the Tx tests
sim/gmac_subsys_vtb/tests/subsys/txrx_tests.v	Has a combined Tx and Rx test case
sim/gmac_subsys_vtb/tests/subsys/user_tests.v	You can code your own test cases in this file
sim/gmac_subsys_vtb/tests/subsys/README.rx_tests	README for the RX tests
sim/gmac_subsys_vtb/tests/subsys/README.tx_tests	README for the TX tests

All BFMs are under the configured workspace: <config_name>/models/verilog_bfms/ directory. Note that the Ethernet Interface BFM instantiates all PHY interface BFMs. [Table C-2](#) defines the BFM directories.

Table C-2 BFM Directories

Directory Name	Description
models/app_bfm	The application BFM for the GMAC-CORE and GMAC_MTL tests is present in this directory
models/clk_gen	Contains the clock generator files
models/eth_bfm	Contains the Ethernet Interface BFM files
models/gbit_bfm	Contains the GBIT BFM files
models/gbit_mon	Contains the GBIT Monitor files
models/gpio_bfm	Contains the GPIO BFM files
models/intf_bfm	Contains the PHY interface files
models/mdio_bfm	Contains the RevMII BFM file
models/pcs_bfm	Contains all PCS BFM-related files
models/ram_model	Contains the DPRAM model, which is instantiated in the testbench
models/rgmii_bfm	Contains the RGMII BFM file
models/rmii_bfm	Contains the RMII BFM file

Table C-2 BFM Directories (Continued)

Directory Name	Description
models/rtbi_bfm	Contains the RTBI BFM file
models/sgm_bfm	Contains the SGMII-related files
models/smii_bfm	Contains the SMII BFM file
models/ts_bfm	Contains the time stamping BFM files

Table C-2 describes the simulation output files in the <config_name>/sim/ directory.

Table C-3 <config_name>/sim Directory Simulation Output Files

File	Description
gmac_subsys_vtb/results/testlog/*.log	The complete log file of the run corresponding to each test case. This is common for all simulators
runttest.log	The PASS/FAIL status of each test case run. This is common for all simulators.
simv, simv.daidir, csrc	VCS related files
veriuser*, work	Intermediate files generated by MTI simulator
veriuser*, hdl.var, cds.lib, INCA_LIBS	Intermediate files generated by NC Verilog simulator

C.4 Tool and Setup Requirements

The prerequisites for running VTB are:

- ❖ All necessary tools are installed as per the requirements.
All tools required to run the GMAC-UNIV simulations are required to run the VTB. Please refer to the DesignWare Cores Ethernet MAC Universal Release Notes for tools/license requirements.
- ❖ Environment variables are set properly.
Set the following environment variables:
 - ◆ \$SYNOPSYS – this must be set to a valid DC installation root. The minimum/latest version of the DC version that is supported for VTB is the same as that of the GMAC-UNIV.
 - ◆ \$DESIGNWARE_HOME – This must point to the required DesignWare home installation directory.
- ❖ AHB VIP is installed in the DesignWare home. (This step is done as a part of simulation).
- ❖ Simulations are running properly.

C.5 Running Simulations

Currently, VTB supports simulations with VCS, MTI, and NC-Verilog simulators through the GUI by default.

To run the individual test-case simulations from command line prompt, just enter the command as follows from the configured workspace: <config_name>/sim/ directory:

- ❖ For VCS, enter the command:

```
./gmac_subsys_vtb/scripts/run_vcs <TEST_CASE_NAME> <OTHER_OPTIONS>
```

The details of the <TEST_CASE_NAME> and <OTHER_OPTIONS> can be obtained from the ./gmac_subsys_vtb/scripts/runall_subsys file. The output of the run is logged in a file called verilog.log file, and can be used to check errors.

- ❖ For MTI, enter the command:

```
./gmac_subsys_vtb/scripts/run_mti <TEST_CASE_NAME> <OTHER_OPTIONS>
```

The details of the <TEST_CASE_NAME> and <OTHER_OPTIONS> can be obtained from the ./gmac_subsys_vtb/scripts/runall_subsys file. The output of the run is logged in a file called verilog.log file, and can be used to check errors.

- ❖ For NC Verilog, enter the command:

```
./gmac_subsys_vtb/scripts/run_ncv <TEST_CASE_NAME> <OTHER_OPTIONS>
```

The details of the <TEST_CASE_NAME> and <OTHER_OPTIONS> can be obtained from the ./gmac_subsys_vtb/scripts/runall_subsys file. The output of the run is logged in a file called verilog.log file, and can be used to check errors.

C.6 Simulation PASS/FAIL

The simulation run status is logged into the workspace: <config_name>/sim/runttest.log file. The details of the run can be accessed from the

<config_name>/sim/gmac_subsys_vtb/results/testlog/<TEST_NAME>.log file at the end of simulation.

D

GMAC-DMA Verification Environment

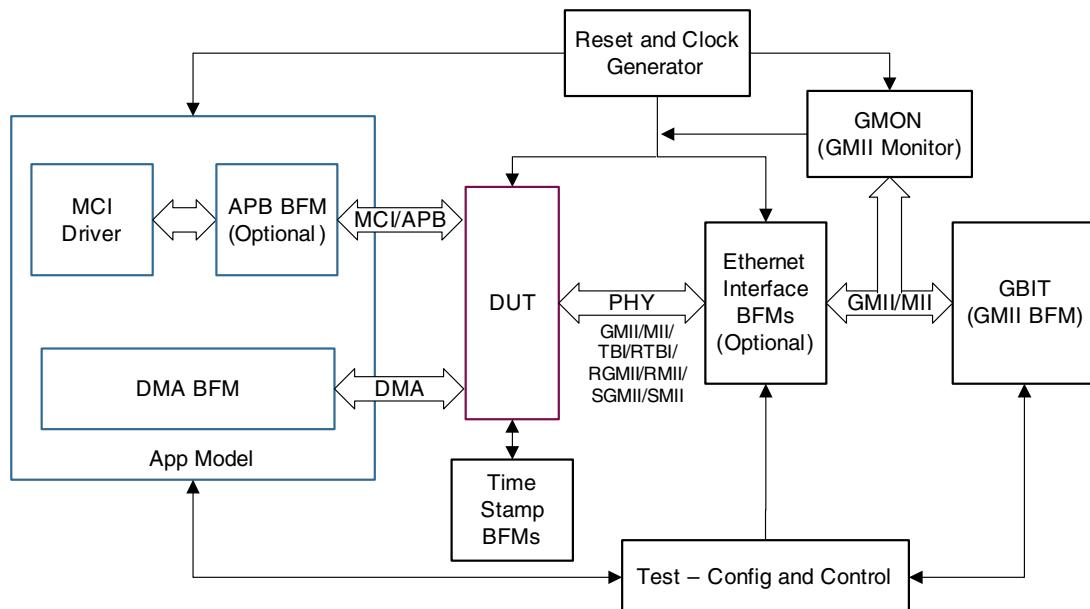
This appendix describes the example verification environment that is packaged in the DWC Ethernet image for the GMAC-DMA configuration, and explains how to use it. This appendix contains the following sections:

- ❖ “Testbench Overview” on page 501
- ❖ “Test Flow” on page 502
- ❖ “Directory Structure” on page 503
- ❖ “Running Simulations” on page 505
- ❖ “Simulation PASS/FAIL” on page 506

D.1 Testbench Overview

Figure D-1 shows the Verification Environment used for GMAC Subsystem with Native DMA Interface.

Figure D-1 GMAC-DMA Verification Environment

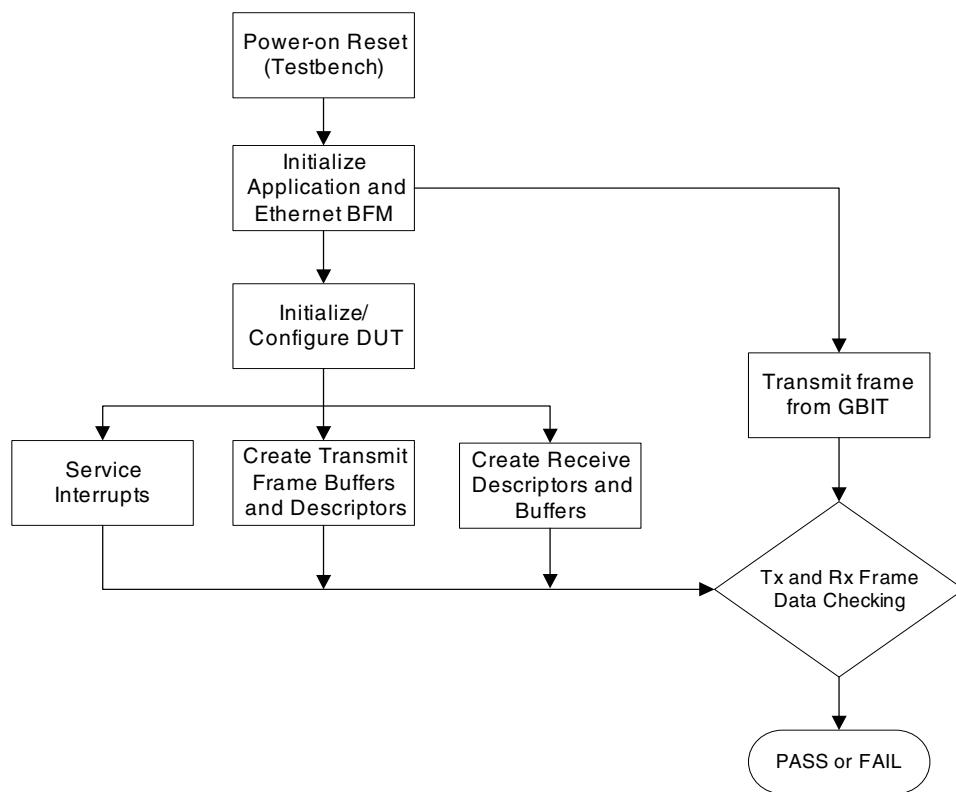


The testbench contains the following blocks.

- ❖ Clock Reset Generator: This verilog block generates the clocks and resets for the DUT including the clocks for the PHY interface (GMII, MII, RMII, RGMII, SGMII, SMII, and RTBI) and for all the testbench BFM models.
- ❖ Application BFM: This model contains the
 - ◆ DMA BFM which interacts with the DMA of the DUT. This module access the Host memory resided in the testbench for each and every DMA transaction. It has the capability for Generating the Variable PUSH/POP lengths and also error conditions for the MDC interface
 - ◆ MCI Driver: This module implements the MCI interface protocol to access the CSR Register set of the DUT.
 - ◆ Optional APB BFM: This module implements the APB Protocol required for accessing the internal register set of DUT when optional APB port is selected.
- ❖ Ethernet BFM: Translates the interface specific protocol (SGMII/TBI etc.) to the GMII/MII and feeds the GBIT BFM, and vice-versa.
- ❖ GMII Monitor: Monitors the protocol on the GMII/MII bus.
- ❖ GBIT BFM: Transmits/receives the Ethernet frames with (G)MII protocol as per the control of TEST - Config and Control block.
- ❖ Time Stamp BFM: Contains the time stamping BFM files. Captures time stamp values at GMII for verification. Also generates time stamps if the External Time Stamp option is selected.
- ❖ TEST - Config and Control: This block is the main test block that configures the BFM, DUT and performs the actual tests. The main testbench control is coded in a simple verilog format. The tests in this environment support all 3 speed modes (1000, 100 and 10 Mbps), in Full- and Half-duplex modes.

D.2 Test Flow

Figure D-2 shows the typical test flow of the GMAC subsystem for Receive and Transmit operations. In both the cases the POR sequence and Initial DUT configuration remains the same.

Figure D-2 GMAC-DMA Verification Environment Test Flow

D.2.1 Frame Receive Flow

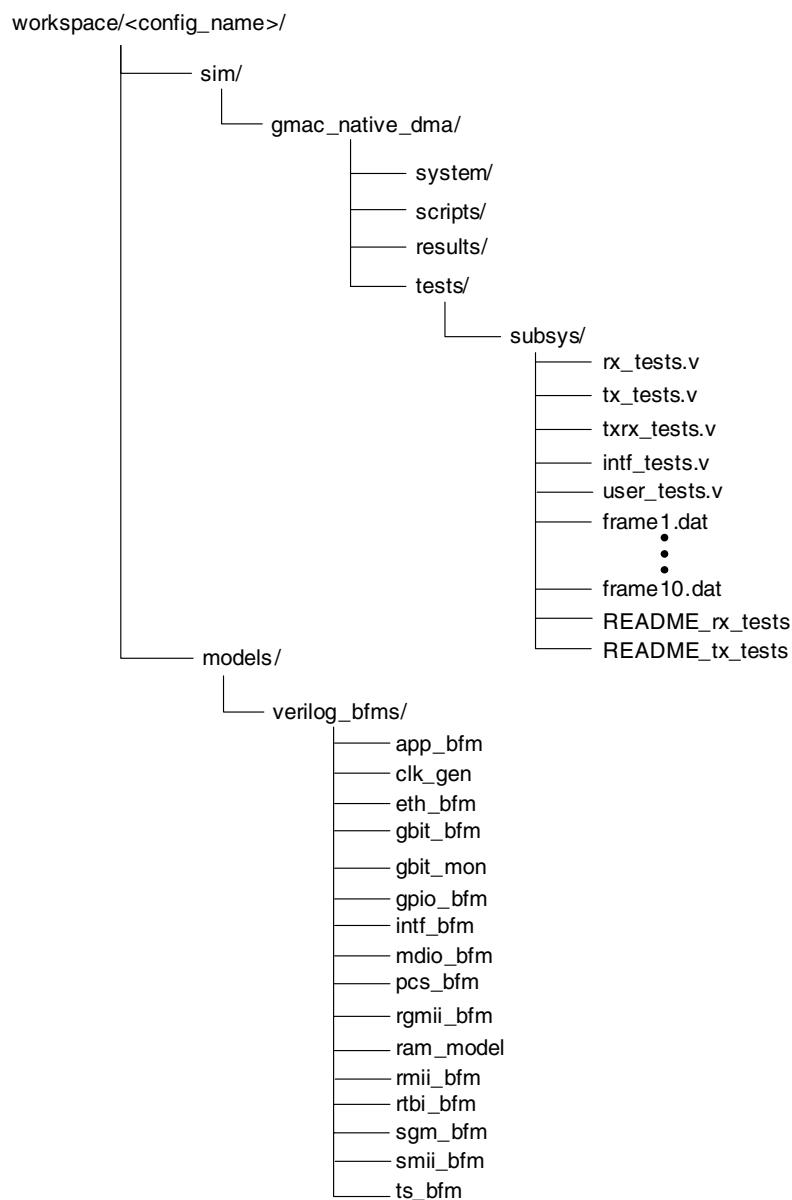
The host task creates the receive descriptor on the host memory and then configures the DUT to start the DMA transaction. It then initiates the Ethernet model to send the frame with a predefined data pattern. The DUT starts transferring the data to the host once it started receiving the frame. Upon completion of the reception host compares the received data with the predefined data pattern.

D.2.2 Frame Transmit Flow

The host fills the host buffer with the predefined data pattern and creates the transmit descriptor on the host memory. It then initiates the DUT to transmit the frame. The DUT starts transferring the data from the host memory through DMA protocol. It waits till the complete frame reaches the Ethernet BFM. Upon reception it does the data integrity check.

D.3 Directory Structure

[Figure D-3](#) illustrates how the testbench files, BFM models, and the test vectors for the GMAC-DMA are organized.

Figure D-3 GMAC-DMA Verification Environment Directory Structure

The <config_name> directory contains the configured RTL and the testbench. The other directories are described in [Table D-1](#).

Table D-1 GMAC-DMA Verification Environment Directory Files

File Name	Description
sim	Simulation directory containing the testbench, test vectors, and scripts. All simulations are run from this directory.
system	Directory containing the testbench and the clock generation module, and also the DMA BFM and MCI driver modules, host tasks, host commands etc.

Table D-1 GMAC-DMA Verification Environment Directory Files (Continued)

File Name	Description
tests	Directory containing transmit and receive test vectors
results	This directory has a sub-directory called test log which is a repository for simulation logs
scripts	Directory containing the scripts for running the test vectors
models	Directory consisting of all the verilog BFM's used in the test environment. <ul style="list-style-type: none"> • app_bfm: directory consisting of application side models • clk_gen: directory consisting of clock generator files • gbit_bfm: directory consisting of Ethernet models • gbit_mon: directory hosting the GBIT monitor model • gpio_bfm: directory consisting of gpio models • eth_bfm: directory consisting of Ethernet interface BFM • intf_bfm: directory consisting of PHY Interfaces BFM • pcs_bfm: directory consisting of PCS models • rgmii_bfm: directory consisting of RGMII interface model • sgm_bfm: directory consisting of SGMII interface model • rtbi_bfm: directory consisting of RTBI interface model • ram_model: directory hosting the RAM model for Tx FIFO and Rx FIFO • ts_bfm: directory consisting of the time stamping BFM files • smii_bfm: directory consisting of SMII interface model

D.4 Running Simulations

Currently, the verification environment supports simulations with VCS, MTI, and NC-Verilog simulators through the GUI by default.

To run the individual test-case simulations from command line prompt, enter the command as follows from the configured workspace:

<config_name>/sim directory:

To run the simulation in VCS use run_vcs script with the following arguments

```
./gmac_native_dma/scripts/run_vcs <test_name> <appclock_freq> < Tx/Rxfifo_size> <base>
< duplex_mode>
```

where

<test_name> = Name of the testcase (You can get the list of testcase names from the
`./gmac_native_dma/scripts/runall_subsys` script file.)

<appclock_freq> = APPCLK_MIN, APPCLK_MAX or APPCLK_TYP

<Tx/Rxfifo_size> = Size of the Tx FIFO. Permissible values are:
`TXFIFO_SIZE_512, TXFIFO_SIZE_1K, TXFIFO_SIZE_2K,`
`TXFIFO_SIZE_4K, TXFIFO_SIZE_8k`.

`RXFIFO_SIZE_512, RXFIFO_SIZE_1K, RXFIFO_SIZE_2K,`
`RXFIFO_SIZE_4K, RXFIFO_SIZE_8k, RXFIFO_SIZE_128,`
`RXFIFO_SIZE_256`

<base> = Operational speed mode. Values allowed are:
BASE10
BASE100
BASE1000

<duplex mode> = The duplex mode of operation. Allowed values are
FULL
HALF

Similarly, simulations can be run on MTI and NC Verilog as follows:

- ❖ For MTI, enter the command:

```
./gmac_native_dma/scripts/run_mti <test_name> <appclock_freq> <Tx/Rxfifo_size> <base>  
<duplex_mode>
```

- ❖ For NC Verilog, enter the command:

```
./gmac_native_dma/scripts/run_ncv <test_name> <appclock_freq> <Tx/Rxfifo_size> <base>  
<duplex_mode>
```

D.5 Simulation PASS/FAIL

The simulation run status available in the following file:

<config_name>/sim/runttest.log file (if simulation is run from GUI).

The details of the run can be accessed from the

<config_name>/sim/gmac_native_dma/results/testlog/<TEST_NAME>.log file at the end of the simulation.

E

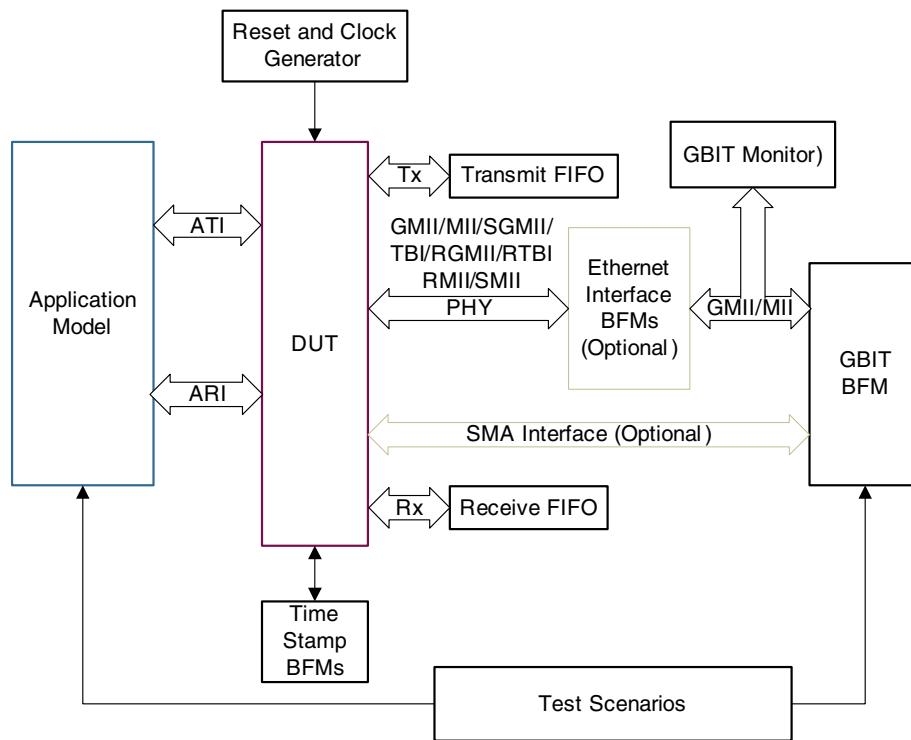
GMAC-MTL Verification Environment

This appendix describes the example verification environment that is packaged in the DesignWare Cores Ethernet MAC Universal image for the GMAC-MTL configuration, and explains how to use it. This appendix contains the following sections:

- ❖ “Testbench Overview” on page [507](#)
- ❖ “Test Flow” on page [509](#)
- ❖ “Directory Structure” on page [509](#)
- ❖ “Running Simulations” on page [511](#)
- ❖ “Simulation PASS/FAIL” on page [512](#)

E.1 Testbench Overview

Figure E-1 shows the verification environment used for GMAC-MTL Subsystem with Native FIFO Interface.

Figure E-1 GMAC-MTL Verification Environment

The testbench contains the following blocks.

- ❖ **Clock and Reset:** Verilog model generating the clocks and resets for the DUT. The clocks for the PHY interface (GMII, MII, RGMII, SGMII, and RTBI) of the DUT and the BFM are also generated by this block.
- ❖ **Application BFM:** The application BFM provides set of tasks for transmitting frames on the transmit interface (ATI) of the DUT. This Model also provides tasks for reading and checking the Transmit Frame status from DUT and tasks for configuring the DUT.
- Frames received from the DUT on the receive interface (ARI) of the DUT are retrieved by the application model and checked for data integrity with the data transmitted from the GBIT model. As the application interface of GMAC-MTL is similar to the GMAC-CORE most of the application model tasks are reused in this subsystem verification.
- ❖ **Ethernet Interface BFM:** The PHY Interface BFM, which connects with the DUT through one or many of these PHY interfaces (GMII/MII/SGMII/RGMII/TBI/RTBI/RMII/SMII). Irrespective of the interface on the DUT, this model transforms these into GMII/MII signals which are understood by the GBIT model.
- ❖ **GBT BFM:** A gigabit model has a gigabit transmitter model and a gigabit receiver model. The transmitter model enables you to send all kinds of packets. The transmitter model transfers the frames through the Ethernet bus, per the transmission protocol. The receiver model performs data integrity checks on frames received from the DUT. The gigabit model also provides a PHY model that can be connected to the DUT Station Management interface.
- ❖ **GBT Monitor:** The Monitor Model is connected to the GMII/MII of the Ethernet Interface BFM. This model monitors the DUT's GMII/MII for Ethernet bus protocol conformance and reports misbehavior.

❖ **Memory Model:** A simple verilog model that implements the MTL transmit and receive FIFOs. This memory model supports both asynchronous and synchronous read options.

❖ **Time Stamp BFM:** Contains the time-stamping BFM files. Captures time stamp values at the GMII for verification. Also generates time stamps if the External Time Stamp option is selected.

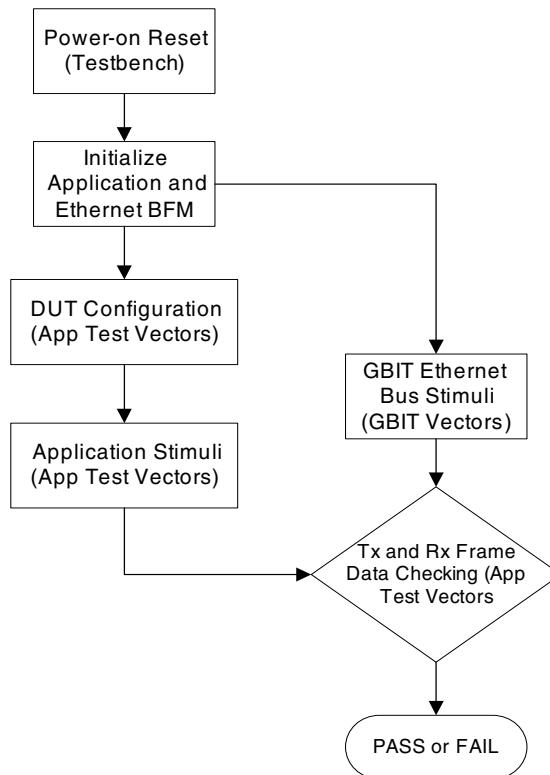
Different test scenarios involve frame transmission from the application and GBIT models. The test cases direct the DUT's configuration through the MAC Control interface.

E.2 Test Flow

The synchronization of the application and GBIT Ethernet bus stimuli is illustrated through the flow diagram shown in [Figure E-2](#). The figure identifies the different steps involved in the test flow and the testbench components involved in these steps.

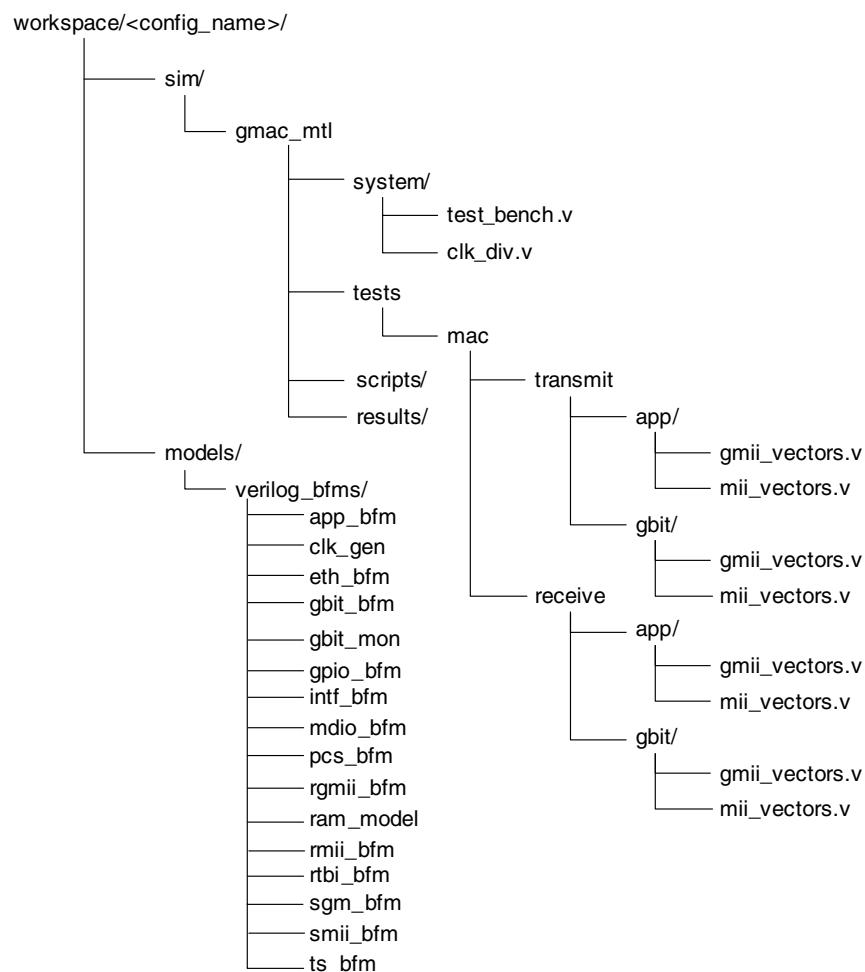
The application test vectors and GBIT vectors are organized in different verilog files and run in tandem. The synchronization between the application and GBIT stimuli is accomplished through shared variables.

Figure E-2 GMAC-MTL Verification Environment Test Flow



E.3 Directory Structure

[Figure E-3](#) illustrates how the testbench files and the test vectors for the GMAC-MTL verification are organized.

Figure E-3 GMAC-MTL Verification Environment Directory Structure

The <config_name> directory contains the configured RTL and the testbench. The other directories are described in [Table E-1](#).

Table E-1 GMAC-DMA Verification Environment Directory Files

File Name	Description
sim	Simulation directory containing the testbench and the test vectors and the scripts All the simulations are run from this directory.
system	Directory containing the testbench and the clock generation module
tests	Directory containing transmit and receive test vectors
transmit	Directory containing the application and GBIT test vectors for all the transmit test cases
receive	Directory containing the application and GBIT test vectors for all of the receive test cases
app	Directory having the application test vectors for GMII and MII modes
gbit	Directory having the gbit side test vectors for GMII and MII modes

Table E-1 GMAC-DMA Verification Environment Directory Files (Continued)

File Name	Description
results	This directory has a sub-directory called testlog, which is a repository for simulation logs.
scripts	Directory containing the scripts for running the test vectors
models	Directory consisting of all the verilog BFM's used in the test environment: <ul style="list-style-type: none"> • app_bfm: directory consisting of application side models • clk_gen: directory consisting of clock generator files • gbit_bfm: directory consisting of Ethernet models • gbit_mon: directory hosting the GBIT monitor model • gpio_bfm: directory consisting of gpio models • eth_bfm: directory consisting of Ethernet interface BFM • intf_bfm: directory consisting of PHY Interfaces BFM • pcs_bfm: directory consisting of PCS models • rgmii_bfm: directory consisting of RGMII interface model • sgm_bfm: directory consisting of SGMII interface model • rtbi_bfm: directory consisting of RTBI interface model • ram_model: directory hosting the RAM model for Tx FIFO and Rx FIFO • ts_bfm: directory consisting of the time stamping BFM files • smii_bfm: directory consisting of SMII interface model

E.4 Running Simulations

Currently, the verification environment supports simulations with VCS, MTI, and NC-Verilog simulators through the GUI by default.

To run the individual test-case simulations from command line prompt, enter the command as follows from the configured workspace:

<config_name>/sim directory:

To run the simulation in VCS use run_vcs script with the following arguments:

```
./gmac_mtl/scripts/run_vcs <test_name> <appclock_freq> <apbclk_freq>
|<csrclk_freq> <base> where
```

<test_name> =	Name of the testcase (You can get the list of testcase names from the ./gmac_mtl/scripts/runall_gmac_mtl script file.)
<appclock_freq> =	can be set to any one of the following values: APPCLK_MIN, APPCLK_MAX or APPCLK_TYP
<apbclk_freq> =	can be set to any one of the following values: APBCLK_MIN, APBCLK_MAX or APBCLK_TYP
<csrclk_freq> =	can be set to any one of the following values: CSRCLK_MIN, CSRCLK_MAX, CSR_CLK_TYP

apbclk_freq and csrclk_freq are mutually exclusive options. Different values of apbclk_freq are used with configuration having APB_SLAVE and a different (from the application clock) CSR clock. csrclk_freq is used with configurations having a different CSR clock and the normal MCI interface.

<base> = option to configure 10/100-Mbps mode (BASE10 BASE100 respectively) only for MII mode testcases.

Similarly, simulations can be run on MTI and NC-Verilog simulators as follows:

- ❖ For MTI, enter the command:

```
./gmac_mtl/scripts/run_mti <test_name> <appclock_freq>
```

- ❖ For NC-Verilog, enter the command:

```
./gmac_mtl/scripts/run_ncv <test_name> <appclock_freq>
```

E.5 Simulation PASS/FAIL

The simulation run status is logged into the workspace:

<config_name>/sim/runttest.log file (if run from GUI).

The details of the run can be accessed from the

<config_name>/sim/gmac_mtl/results/testlog/<TEST_NAME>.log file at the end of simulation.

F

GMAC-Core Verification Environment

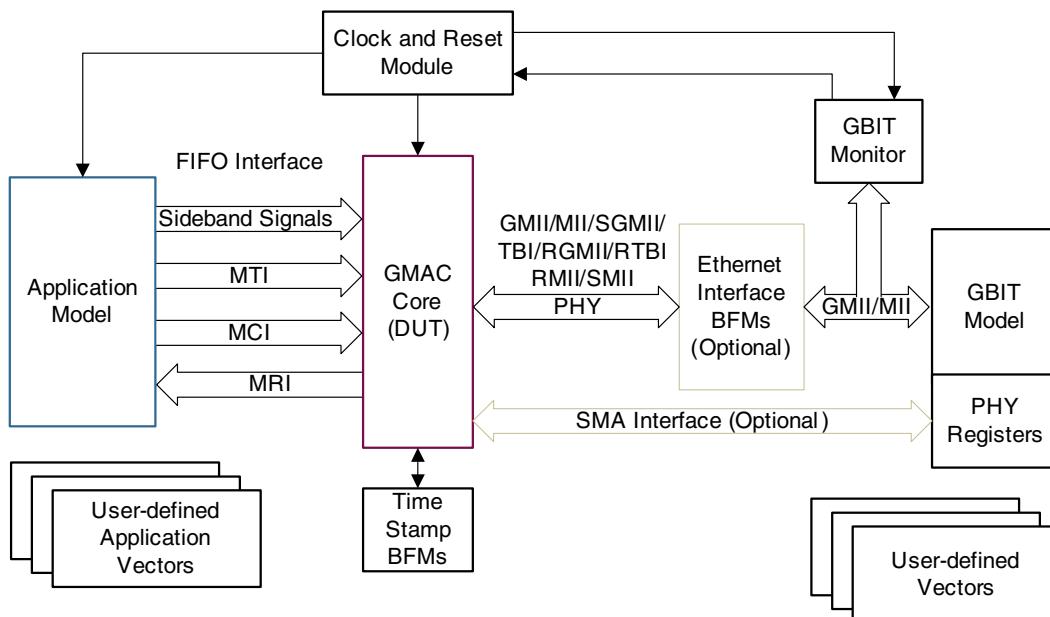
This appendix describes the example verification environment that is packaged in the DesignWare Cores Ethernet MAC Universal image for the GMAC-CORE configuration, and explains how to use it. This appendix contains the following sections:

- ❖ “Testbench Overview” on page 513
- ❖ “Test Flow” on page 514
- ❖ “Directory Structure” on page 515
- ❖ “Running Simulations” on page 517
- ❖ “Simulation PASS/FAIL” on page 518

F.1 Testbench Overview

Figure F-1 shows the Verification Environment used for GMAC-CORE with native FIFO Interface

Figure F-1 GMAC-Core Verification Environment With Native FIFO Interface



The testbench contains the following blocks:

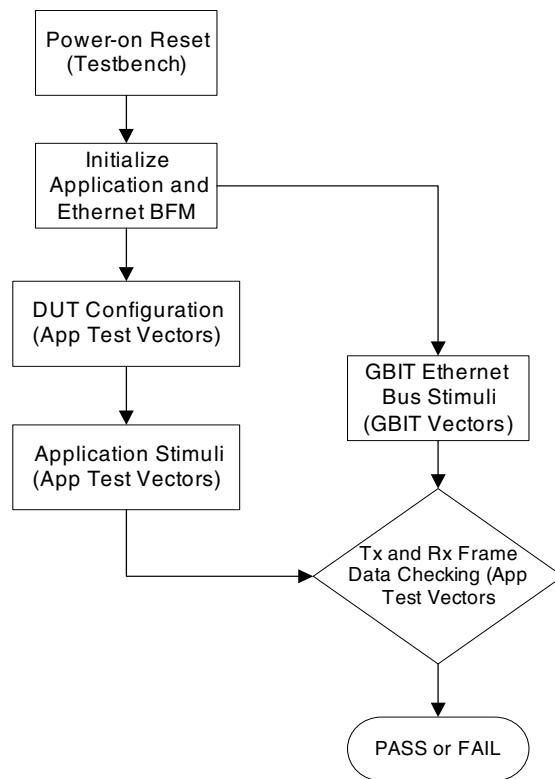
- ❖ **Clock and Reset:** Verilog Model generating the clocks and resets for the DUT. The clocks for the PHY interface (GMII, MII, RMII, RGMII, SGMII, SMII, and RTBI) of the DUT and the BFM are also generated by this block.
 - ❖ **Application BFM:** The application BFM provides set of tasks for transmitting frames on the transmit interface (MTI) of the DUT. This model also provides tasks for reading and checking the transmit frame status from DUT and tasks for configuring the DUT.
- Frames received from the DUT on the receive interface (MRI) of the DUT are retrieved by the application model and checked for data integrity with the data transmitted from the GBIT model.
- ❖ **Ethernet Interface BFM:** The PHY Interface BFM which connects with the DUT through one or many of these PHY interfaces (GMII, MII, SGMII, RGMII, TBI, RMII, SMII, and RTBI). Irrespective of the interface on the DUT, this model transforms these into GMII/MII signals that the GBIT model can process.
 - ❖ **GBT BFM:** A gigabit model has a gigabit transmitter model and a gigabit receiver model. The transmitter model enables you to send all kinds of packets. The transmitter model transfers the frames through the Ethernet bus, per the transmission protocol. The receiver model performs data integrity checks on the frames received from the DUT. The gigabit model also provides a PHY model that can be connected to the DUT Station Management interface.
 - ❖ **GBT Monitor:** The monitor model is connected to the Ethernet interface BFM's GMII/MII. This model monitors the DUT's GMII/MII for Ethernet Bus protocol conformance and reports misbehavior.
 - ❖ **Time Stamp BFM:** Contains time stamping BFM files. Captures time stamp values at the GMII for verification. Generates time stamps if the External Time Stamp option is selected.

Different test scenarios involves frames transmission from the application and GBIT models. The test cases direct configuration of the DUT through the MAC Control interface.

F.2 Test Flow

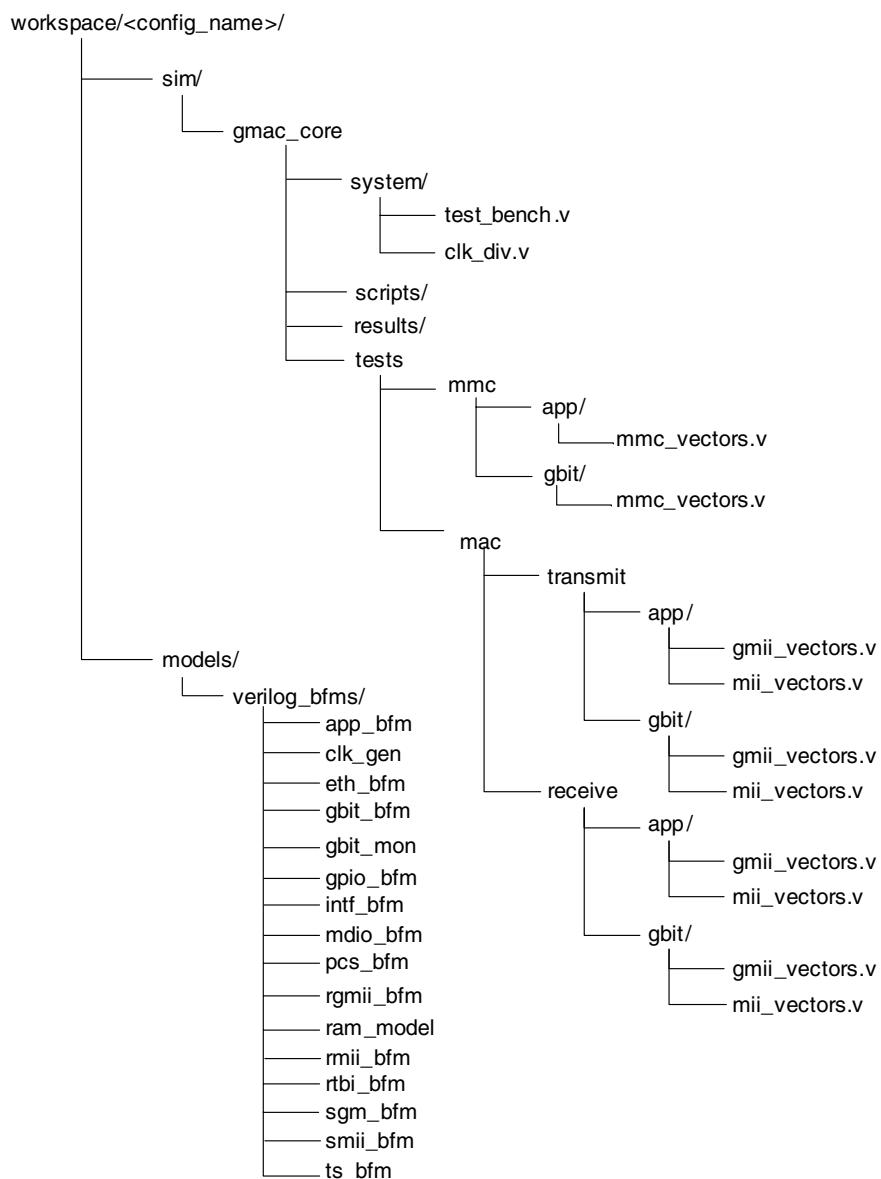
The synchronization of the application and GBIT Ethernet bus stimuli is illustrated through the flow diagram shown in [Figure F-2](#). The figure identifies the different steps involved in the test flow and the testbench components involved in these steps.

The application test vectors and GBIT vectors are organized in different verilog files and run in tandem. The synchronization between the application and GBIT stimuli is accomplished through shared variables.

**Figure F-2 GMAC-Core Verification Environment Test Flow**

F.3 Directory Structure

Figure F-3 illustrates how the testbench files and the test vectors for the GMAC-CORE verification are organized.

Figure F-3 GMAC-Core Verification Environment Directory Structure

The <config_name> directory contains the configured RTL and the testbench. The other directories are described in [Table F-1](#).

Table F-1 GMAC-Core Verification Environment Directory Files

File Name	Description
sim	Simulation directory containing the testbench and the test vectors and the scripts All the simulations are run from this directory. This is the root directory of the simulation environment.
system	Directory containing the testbench and the clock generation module

Table F-1 GMAC-Core Verification Environment Directory Files

File Name	Description
tests	Directory containing transmit and receive test vectors: <ul style="list-style-type: none"> • transmit: Directory containing the application and GBIT test vectors for all of the transmit test cases. This directory in turn contains two sub directories: <ul style="list-style-type: none"> - app: Directory having the application test vectors for GMII and MII modes - gbit: Directory having the GBIT side test vectors for GMII and MII modes • receive: Directory containing the application and GBIT test vectors for all the receive test cases. This directory in turn contains two sub directories: <ul style="list-style-type: none"> - app: Directory having the application test vectors for GMII and MII modes - gbit: Directory having the GBIT side test vectors for GMII and MII modes
results	This directory has a sub-directory called testlog which is a repository for simulation logs
scripts	Directory containing the scripts for running the test vectors
models	Directory consisting of all the verilog BFM's used in the test environment: <ul style="list-style-type: none"> • app_bfm: directory consisting of application-side models • eth_bfm: directory hosting Ethernet interface BFM • clk_gen: directory consisting of clock generator files • gbit_bfm: directory consisting of Ethernet models • gbit_mon: directory hosting the GBIT monitor model • gpio_bfm: directory consisting of GPIO models • intf_bfm: directory consisting of PHY interfaces models • mdio_bfm: directory consisting of RevMII interface model • pcs_bfm: directory consisting of PCS models • ram_model: directory consisting of RAM models • rgmii_bfm: directory consisting of RGMII interface model • rtbi_bfm: directory consisting of RTBI interface model • rmii_bfm: directory consisting of RMII interface model • sgm_bfm: directory consisting of SGMII interface model • smii_bfm: directory consisting of SMII interface model • ts_bfm: directory consisting of the time stamping BFM files

F.4 Running Simulations

The simulation environment is generated after the simulate activity is completed in the coreConsultant. You can run the simulations through the GUI or generate scripts only and run the scripts using the command prompt.

Sample scripts are available in the scripts directory to run simulations using VCS, MTI, and NC-Verilog.

To run an individual test in VCS use run_vcs script from the sim/ directory with the following arguments

```
./gmac_core/scripts/run_vcs <test_name> <base> <interface> where
<test_name>= Name of the testcase
<base>= BASE10, BASE100 (used in MII mode, to differentiate between 10-Mbps and
          100-Mbps tests.
<interface>= If you have opted for any interface other than GMII, the following must be
          passed as an argument:
```

RGMII_SEL: RGMII interface
 SGMII_SEL: SGMII interface
 TBI_SEL: TBI interface
 RTBI_SEL: RTBI interface
 RMII_SEL: RMII interface
 SMII_SEL: SMII interface

Similarly, simulations can be run on MTI and NC-Verilog as follows:

- ❖ For MTI, enter the command:

```
./gmac_core/scripts/run_mti test_name base interface
```

- ❖ For NC-Verilog, enter the command:

```
./gmac_core/scripts/run_ncv test_name base interface
```

F.5 Simulation PASS/FAIL

The simulation run status is logged into the workspace:

```
<config_name>/sim/runttest.log file (if run from the GUI)
```

A sample log file of the runall_userlist is shown below. The details of the individual test case run can be accessed from the results/testlog/<TEST_NAME>.log file at the end of simulation.

```
+-----+
| Summary of all Results |
+-----+
+-----+
| Verification Activity Log |
+-----+
```

This logfile is created by the runtest.sh script found in
<workspace>/sim

The simulator and testbench preparation steps should follow immediately.

```
+-----+
| Testbench Preparation |
+-----+
```

(this section of runtest.log supplied by runtest script)

```
runttest: To recreate the run from this point for debug, do the following
runttest: % cd /remote/dept-SG-Reg/ethernet_regress/revankar/simulation/config4/sim
runttest: % runtest\
          --test test_core\
          --SimChoice VCS\
          --DesignView RTL\
          --NetlistDir <coreKit>/src
```

```
runttest: coreKit in file:/simulation
runttest: runtest in file:/simulation/config
runttest: test is in file:/simulation/config/sim
```

```
runttest: Creating simulation command file:  
runttest:   file:/simulation/config/sim/test.sim_command  
runttest: Creating simulation start script test.startsim containing:  
runttest:   % gmac_core/scripts/runall_m1000 VCS > /dev/null  
runttest: Running test.startsim at Thu Sep  8 16:38:44 IST 2005
```

```
+-----+  
| Individual Testcase Logs |  
+-----+
```

(The logs are stored in gmac_core/results/testlog directory)

```
+-----+  
| Compile Execution |  
+-----+
```

(this section of runtest.log supplied by test.startsim script)

```
=====  
Results of the Script Run on Date: 09:08:05 Time: 16:38:44  
=====
```

Note: Running GMII tests using VCS

Start of Transmit Tests

Start of Transmit Half Duplex GMII Tests

```
DATE: 09/08/05 TIME : 16:39:28 PASSED : TEST_gth_cop_____  
DATE: 09/08/05 TIME : 16:40:12 PASSED : TEST_gth_cod_____  
DATE: 09/08/05 TIME : 16:40:38 PASSED : TEST_gth_ncrs_____  
DATE: 09/08/05 TIME : 16:53:50 PASSED : TEST_gth_rcnt_____
```

Start of Transmit Full Duplex GMII Tests

```
DATE: 09/08/05 TIME : 17:25:34 PASSED : TEST_gtf_def_____  
DATE: 09/08/05 TIME : 17:28:00 PASSED : TEST_gtf_dc_____  
DATE: 09/08/05 TIME : 17:30:57 PASSED : TEST_gtf_dp_____  
DATE: 09/08/05 TIME : 17:33:29 PASSED : TEST_gtf_dpdc_____
```

Start of Receive Tests

Start of Receive Half Duplex GMII Tests

```
DATE: 09/08/05 TIME : 17:45:20 PASSED : TEST_grh_ra_____  
DATE: 09/08/05 TIME : 17:45:27 PASSED : TEST_grh_rper_____  
DATE: 09/08/05 TIME : 17:45:34 PASSED : TEST_grh_de_____  
DATE: 09/08/05 TIME : 17:45:41 PASSED : TEST_grh_dro_____  
DATE: 09/08/05 TIME : 17:47:23 PASSED : TEST_grh_mc_____
```

Start of Receive Full Duplex GMII Tests

```
DATE: 09/08/05 TIME : 18:59:23 PASSED : TEST_grf_pass_____  
DATE: 09/08/05 TIME : 18:59:40 PASSED : TEST_grf_lc_____  
DATE: 09/08/05 TIME : 18:59:48 PASSED : TEST_grf_ty_____  
DATE: 09/08/05 TIME : 19:00:06 PASSED : TEST_grf_pctrl_____
```


G

GMAC-AXI Verification Environment

This appendix describes the example Verilog Testbench used to perform functional verification of the GMAC-AXI subsystem. This VTB provides a starting point for understanding how to use the DesignWare AMBA Verification IP (VIP), Ethernet VIP, and the configured DWC Ethernet core (configured for AXI system interface) together in the verification environment.

This appendix contains the following sections:

- ❖ [VTB Overview](#)
- ❖ [Verification Flow](#)
- ❖ [Directory Structure](#)
- ❖ [GPIO](#)
- ❖ [SoC-Level Verification Guidelines](#)
- ❖ [Running Simulations](#)
- ❖ [Simulation PASS/FAIL](#)

G.1 VTB Overview

The example VTB demonstrates the following:

- ❖ How to connect the AMBA VIP, Ethernet VIP, and DWC Ethernet (RTL) synthesizable components in a VTB.
- ❖ How to initialize and program (using Verilog VIP commands) the DWC Ethernet to perform basic operating functions.

The system uses AMBA VIP on the application side and Ethernet VIP on the Ethernet side (see [Figure G-1](#)). The VIPs have their own built-in tasks that are used to perform various operations to verify the DUT. This VTB is made compatible to SoC level verification by constraining the application-side test to communicate with Ethernet VIP only through GPIO.

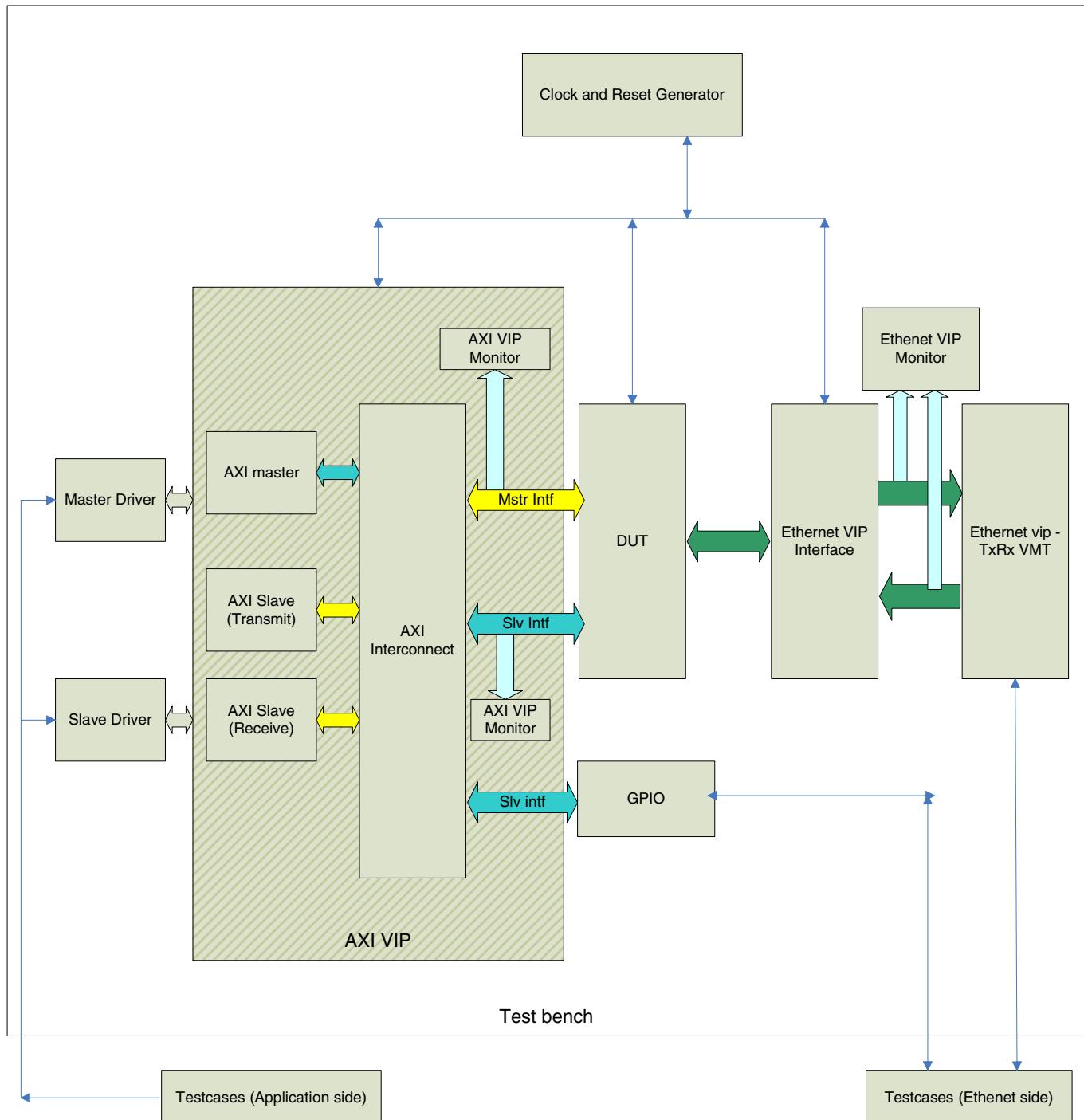
For more details about verification IP, refer to the following documents:

- ❖ *DesignWare AXI Verification IP Databook*, Synopsys Inc.
https://www.synopsys.com/dw/doc.php/vip/amba/latest/doc/axi_um.pdf
- ❖ *DesignWare Ethernet Verification IP User Manual*, Synopsys Inc.
https://www.synopsys.com/dw/doc.php/vip/ethernet/latest/doc/ethernet_vmt_user.pdf
- ❖ *VMT User.s Manual*, Synopsys Inc.

<https://www.synopsys.com/dw/doc.php/vip/vmt/latest/doc/vmtum.pdf>

These documents are also included in the DWC Ethernet image.

Figure G-1 GMAC AXI Verification Environment



The VTB consists of the following blocks:

- ❖ DUT: The device under test – DW_GMAC_AXI_SUBSYS core, The VTB instantiates the core configured to AXI configuration.
- ❖ AXI Interconnect: The DW AMBA VIP “axi_interconnect_vmt”, an Interconnect to connect the AXI masters and slaves.
- ❖ AXI Master: The DW AMBA VIP “axi_master_vmt”, an AXI Master Model that can perform reads and writes to the slaves present in the system. This is used to write and read the DUT CSR for initialization, interrupt service, etc. This is also used to write from and read into the GPIO which is used to control the Ethernet VIP operation.
- ❖ AXI Slave: The DW AMBA VIP “axi_slave_vmt”, an AXI Slave model that interfaces to Memory. The test case transfers data to/from this memory by means of back-door commands, not through actual AXI transactions. But, the DUT accesses are through proper interface transactions.
- ❖ AXI Monitor VIP: The DW AMBA VIP “axi_monitor_vmt”, an AXI monitor to perform AXI protocol checking.
- ❖ Master Driver: This master driver is a collection of bus independent tasks called from the test control block. These tasks in turn call the AXI Master VIP commands.
- ❖ Slave Driver: It consists of bus independent tasks to perform read and writes to the memory in axi_slave_vmt through “backdoor” commands.
- ❖ Ethernet TxRX VIP: The DW ETHERNET VIP “ethernet_txrx_vmt”, A transceiver model with a command-based interface that transmits and receives correct and erroneous traffic on the supported interfaces (MII, RMII, GMII, RGMII, TBI, or SGMII).
- ❖ Ethernet monitor VIP: The DW ETHERNET VIP “Ethernet_monitor_vmt”, This block passively monitors Ethernet transactions, generates reports about transactions and logs. Either all or selected behavior is used to track the various transactions with Ethernet VIP.
- ❖ GPIO: All communication between the application-end and Ethernet-end test cases is routed only through the GPIO block, making the VTB reusable for SoC-level verification.
- ❖ Interface BFM: This block connects DUT ports to corresponding Ethernet VIP ports. Depending on the specified interface (MII, RMII, GMII, RGMII, TBI, SMII, RevMII, or SGMII), the Ethernet VIP permits only specific ports to be connected, while leaving other VIP ports left unconnected as required.
- ❖ Test case (application-side): The application-side or host-side test case initializes the VTB and controls the application side of the DUT. This also consists of a set of tasks that can be used to control the AMBA VIP. For SoC-level verification, this set of test cases can be rewritten in a high-level language, such as C.
- ❖ Test case (Ethernet-side): This set of test cases controls the Ethernet VIP using predefined tasks. This block communicates with the other VTB blocks only through GPIO.
- ❖ Clock and Reset Generation: This block is used to generate the Clock and Reset required for the entire system.

G.2 Verification Flow

The basic verification flow is explained in this section. The transmission- and reception-side verification flows are split, the first half pointing to the application side, the second to the Ethernet VIP side. All communications and handshakes occur between the application and line sides through the GPIO port only.

G.2.1 Transmission Verification Flow

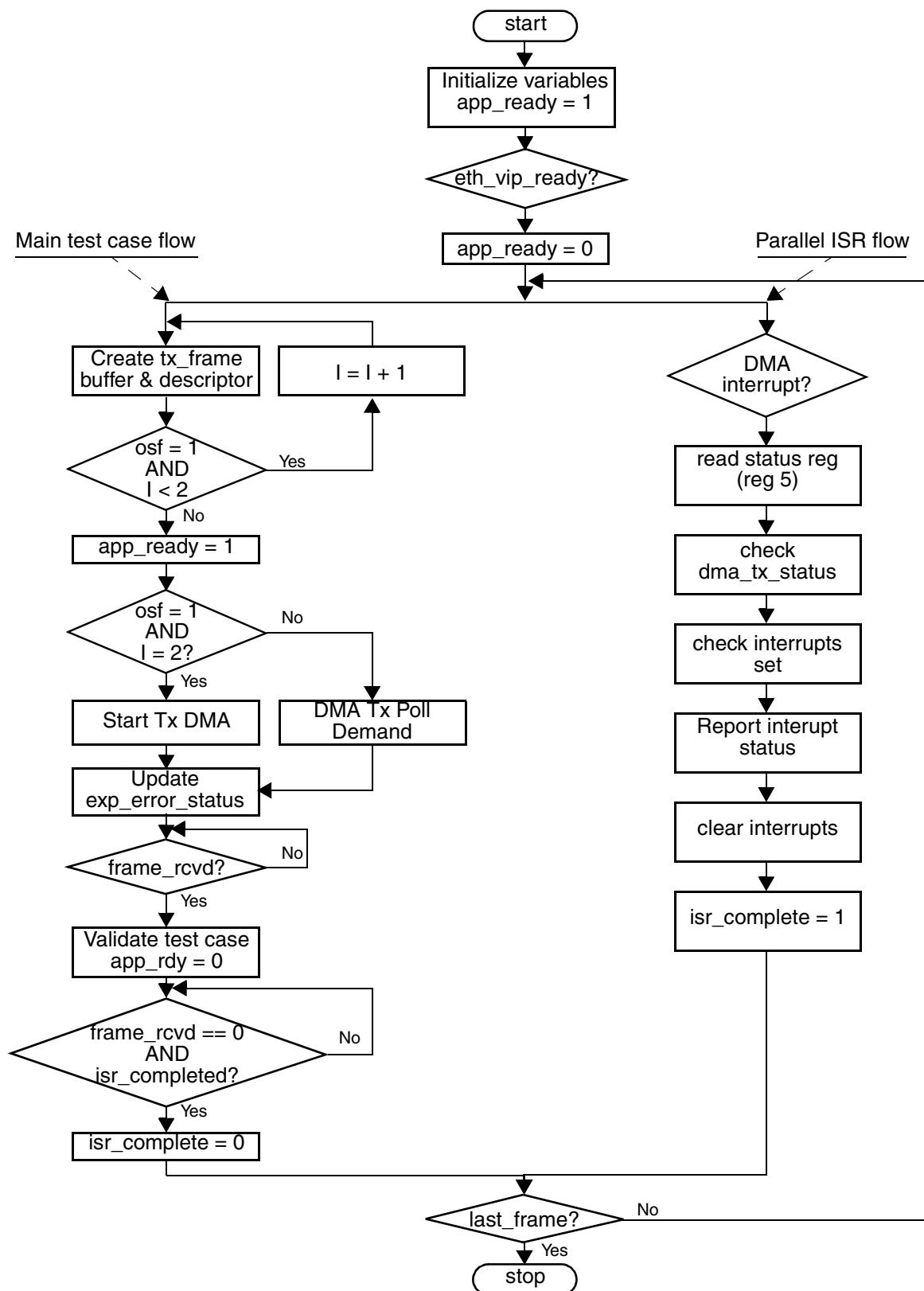
G.2.1.1 Application-Side Test Case Flow

The basic data flow at the host end for verification of the GMAC subsystem (transmission) is as depicted in [Figure G-2](#). The flow is as follows:

1. The application initiates the data transfer by programming the MAC, then initializing all the application-side variables used by the test case.
2. The application-side test case and the Ethernet-side test case perform a handshake to synchronize. The application-side test case sets the app_ready bit in the GPIO, then reads the GPIO to check the eth_vip_ready bit setting. The Ethernet-side test case sets the eth_vip_ready bit after it completes its initialization. The application then clears the app_ready bit.
3. The application-side test case creates a transmit descriptor and sets the app_ready bit in the GPIO to indicate that a frame is pending for transmission.
4. Program the DUT (set Start TxDMA or Tx Poll Demand) to start the frame transmission and initialize the expected transmission status in the internal exp_error_status variable.
5. The host waits for the Ethernet VIP to set the frame_rcvd bit in the GPIO.
6. When the Ethernet VIP sets the frame_rcvd bit, the application-side test case reads rcvd_error_status from the GPIO and compares it to exp_error_status.
7. The test case passes if exp_error_status and rcvd_error_status match, and if the Payload Check Status bit (exp_data_rcvd) in the GPIO is set.
8. The host clears the app_ready bit and waits for the Ethernet VIP to clear the frame_rcvd bit.
9. The host waits for the setting of the isr_complete bit, then clears it before transmitting the next frame. This step synchronizes the main loop of the test case with the Interrupt Service Routine loop of the test case (shown in [Figure G-2](#) on page 525) for every frame transmitted.
10. If more frames are to be transmitted, the process returns to [Step 2](#), continuing through [Step 9](#).
11. If the OSF bit is set, the host creates two descriptors before transmitting the first frame.

In [Figure G-2](#) on page 525, a second thread or process (on the right side) is executed in parallel with the above steps. This is the emulation of the Interrupt Service Routine (ISR) in verilog. You must port the code given in the second thread to the ISR. The isr_complete variable is used to handshake and synchronize the application-side testcase routines inside and outside of the ISR in this flow. This loop is triggered whenever the DUT asserts an interrupt. The ISR reads the status registers, checking whether the proper and expected interrupt bits are set, then clears the interrupt and waits for the next interrupt.

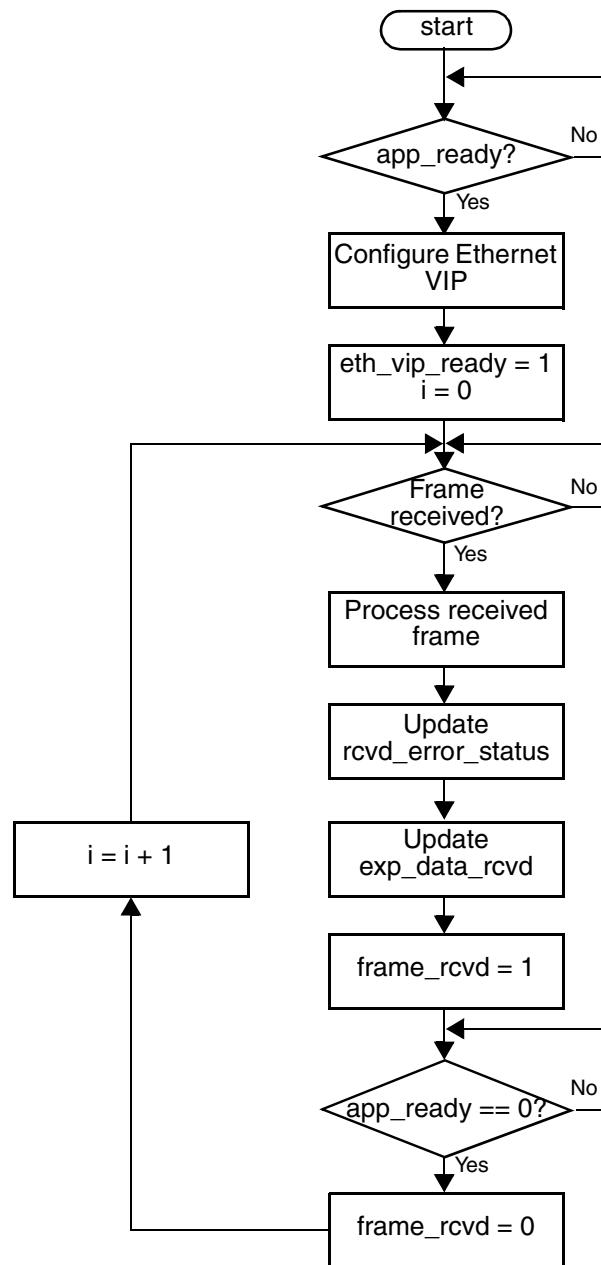
Similarly, the frame_rcvd variable is used as handshake to synchronize the routines in the application-side test case and the Ethernet-side test case (as explained in "[Ethernet-Side Test Case Flow](#)" on page 526).

Figure G-2 Application-Side Test Case Flow (Tx)

G.2.1.2 Ethernet-Side Test Case Flow

The basic test case flow at the Ethernet VIP (transmission) is as shown in [Figure G-3](#). The flow is as follows:

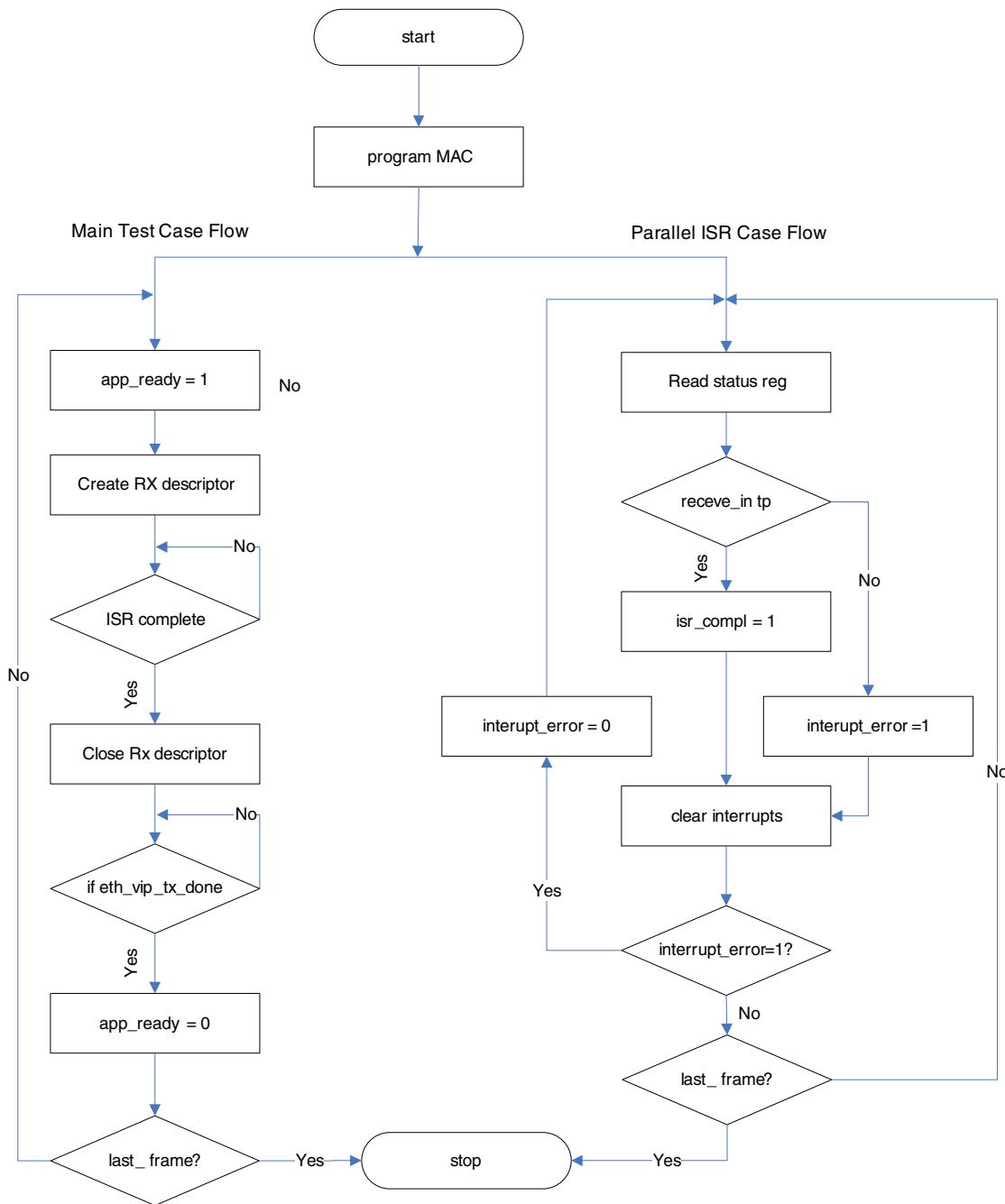
1. The Ethenet_VIP test case checks for the app_ready bit to be set
2. When the app_ready bit is set, the test case configures the Ethernet VIP and sets the eth_vip_ready bit.
3. The Ethernet VIP waits for the frame. When it receives the frame, the test case processes it, checking for any errors the Ethernet VIP may have reported.
4. The corresponding error status is set in rcvd_error_status and exp_data_rcvd bit is set if there is no mismatch in the expected and received payload.
5. The frame_rcvd bit is set, until the host clears the app_ready bit in the GPIO.
6. The Ethernet VIP returns to [Step 3](#)

Figure G-3 Ethernet-Side Test Case Flow (Tx)

G.2.2 Receive Verification Flow

G.2.2.1 Application-Side Test Case Flow (Rx)

The basic test case flow at the host end (Rx) for the verification of GMAC subsystem is shown in [Figure G-4](#). A brief description of the flow follows.

Figure G-4 Host Side Test Case Flow (Rx)

1. The host initializes the DUT by programming the MAC and initializing all the variables used by the test case.
2. The host creates a receive descriptor for the expected frame and then programs the RxDMA for proper operation, including enabling the appropriate interrupts.

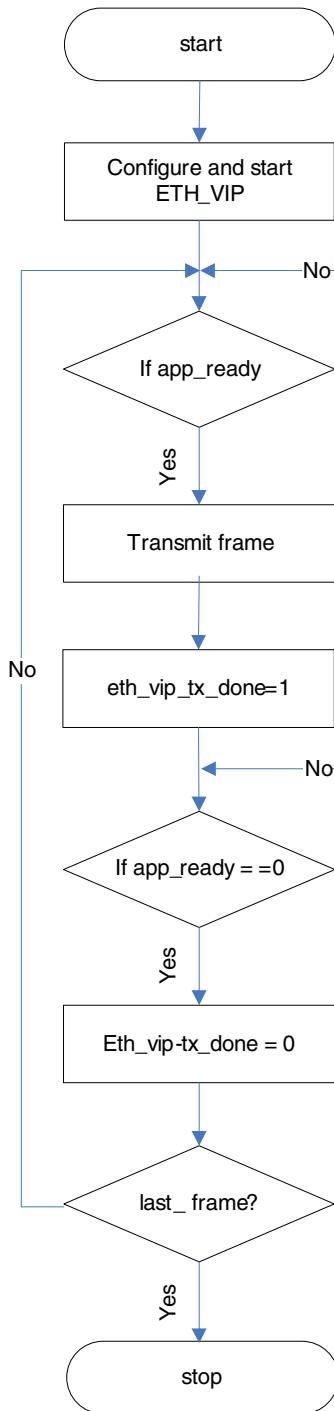
3. The application-side test case and the Ethernet-side test case perform a handshake to synchronize its activities. The host sets the app_ready bit in the GPIO to indicate to the Ethernet-side that it is ready to receive a frame.
4. It then waits for the frame by checking whether the isr_complete flag is set. This flag is set in the parallel ISR after it completes the processing of the interrupt asserted due to a received frame.
5. The host then checks the Received status and frame data for errors.
6. The host resets the app_ready bit in GPIO after it finds the vip_tx_done bit as high in the GPIO_IN register. This is the handshake process to synchronizing both the sides of the testcase.
7. Steps 2 to 6 are repeated until all the expected frames are received.

In [Figure G-4](#), a second thread or process (on the right side) is executed in parallel with the above steps. This is the emulation of the Interrupt Service Routine (ISR) in verilog. You must port the code given in the second thread to the ISR. The isr_complete variable is used to handshake and synchronize the application-side testcase routines inside and outside of the ISR in this flow. This loop is triggered whenever the DUT asserts an interrupt. The ISR reads the status registers, checking whether the proper and expected interrupt bits are set, then clears the interrupt and waits for the next interrupt.

G.2.2.2 Ethernet-Side Test Case Flow (Rx)

The basic test case flow at the Ethernet VIP (Rx) is as shown in [Figure G-5](#). The flow is as follows.

1. The Ethernet VIP is configured to correct mode of operation and then started.
2. The test case waits for the assertion of app_ready bit in GPIO.
3. Once the app_ready bit is set, the Ethernet VIP transmits a frame of size equal to looping index K.
4. The testcase then synchronizes with the host-side testcase by first setting the vip_tx_frame bit and then waiting for the de-assertion of app_ready_bit.
5. It then clears the vip_tx_frame bit input to the GPIO.
6. Repeat steps 2 to 6 until all the expected frames are transmitted.

Figure G-5 Ethernet Side Test Case Flow (Rx)

G.2.3 Transmit-Receive Verification Flow

G.2.3.1 Application-Side Test Case Flow (TxRx)

The basic data flow at the host end for the verification of GMAC subsystem when both transmission and reception are occurring simultaneously is as show in [Figure G-6](#) and [Figure G-7](#). A brief description of the flow follows.

1. The Application initiates the data transfer, by programming the MAC and initializing all the variables used by the test case.
2. Then the app_ready bit in the GPIO is set, till the eth_vip_ready bit in the GPIO_IN is set, then the app_ready bit is cleared.
3. Then transmit descriptor and receive descriptor are created simultaneously and the app_ready bit in the GPIO is set.
4. Then the frame is transmitted and the expected error status is stored in internal variable exp_error_status, at the same time the frame transmitted by the Ethernet VIP is received and processed.
5. Then host waits for isr_complete bit.
6. The host then waits for the frame_rcvd bit in the GPIO to be set by the eth_vip.
7. When the Ethernet VIP sets the frame_rcvd bit, the rcvd_error_status is read and compared with the exp_error_status.
8. The test-case is said to pass if frame has been received without any errors and for transmit frame exp_error_status and the rcvd_error_status matches and the exp_data_rcvd bit in the GPIO is set.
9. The host then clears the app_ready bit and waits for the Ethernet VIP to clear the frame_rcvd bit.
10. Then host clears the isr_complete bit before transmitting the next frame.
11. If there are more frames to be transmitted, steps 2 to 9 are repeated.

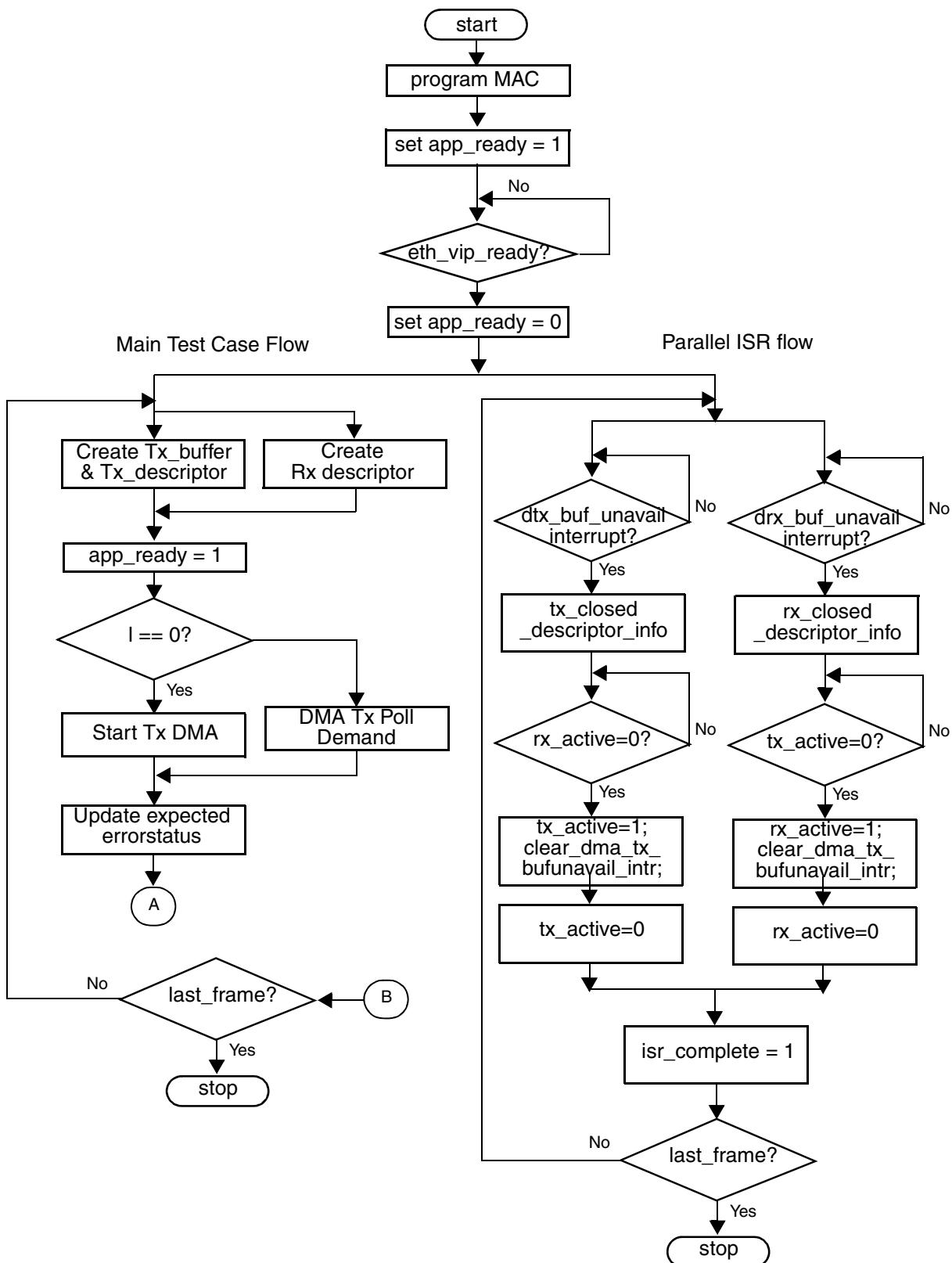
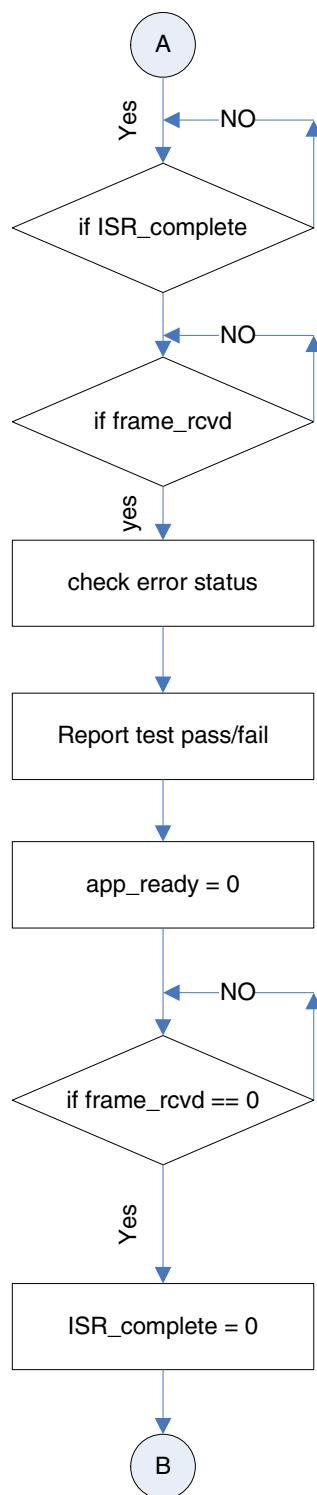
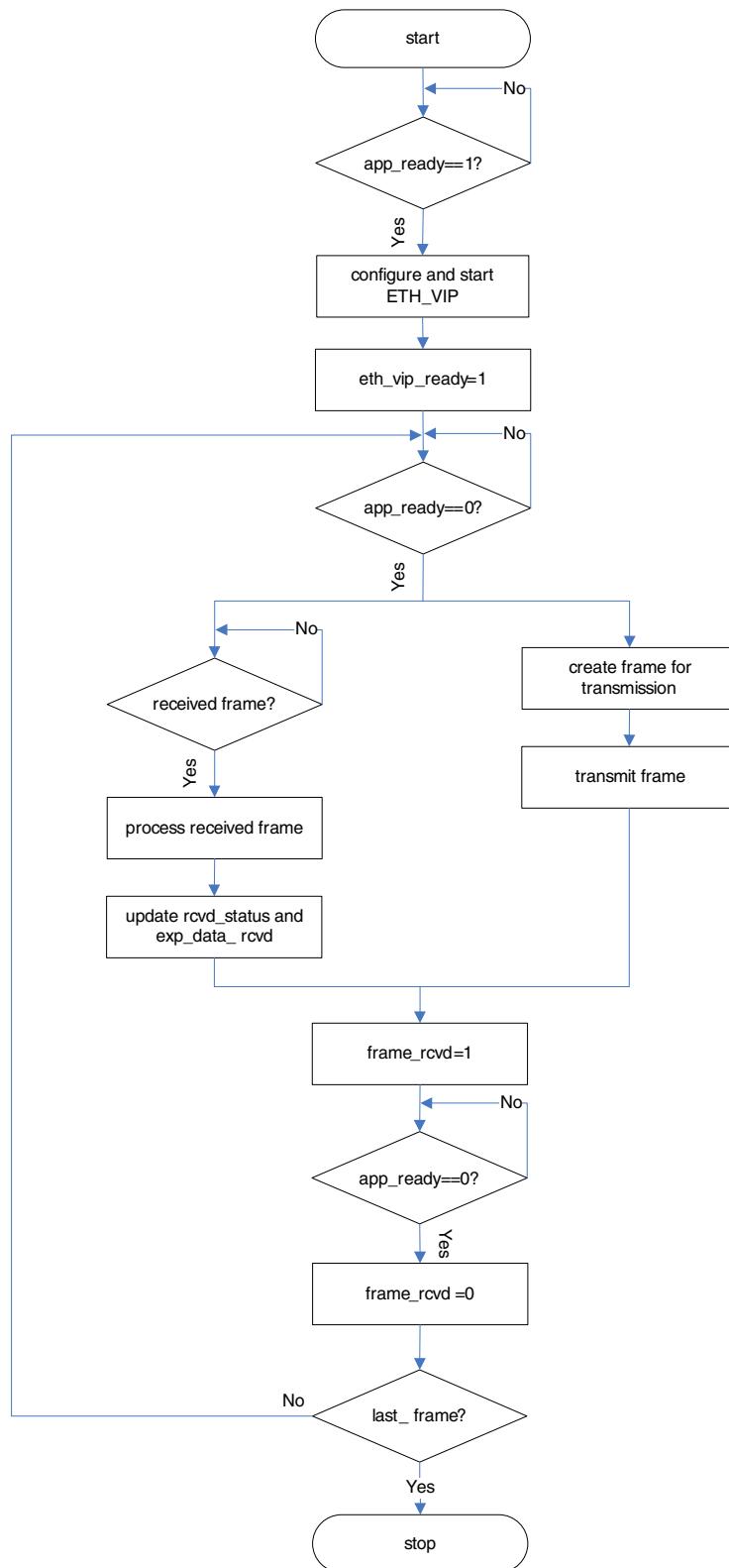
Figure G-6 Application Side Test Case Flow (TxRx) Part 1

Figure G-7 Application Side Test Case Flow (TxRx) Part 2

G.2.3.2 Ethernet-side Test Case Flow (TxRx)

The basic test case flow at the Ethernet VIP (Tx and Rx) is as shown in [Figure G-8](#). An overview of the flow follows.

1. Ethenet_VIP test case keep checking for the app_ready bit to be set.
2. When app_ready bit is set it configures the Ethernet VIP, and sets the eth_vip_ready bit.
3. Then Ethernet vip transmits a frame and also waits for reception of the frame transmitted by the DUT.
4. Ethernet VIP then processes the received frame.
5. The corresponding error status is set in rcvd_error_status and rcvd_exp_data bit is set if there is no mismatch in the expected and received payload.
6. Then the frame_rcvd bit is set.
7. Then wait for the host to reset the app_ready bit.
8. Then the frame_rcvd bit is reset.
9. If there are more frames to be transmitted and received, steps 3 to 9 are repeated.

Figure G-8 Ethernet VIP Side Test Case Flow (TxRx)

G.3 Directory Structure

Figure G-9 shows the organization of testbench files, BFMAs, and test vectors for the gmac_axi_subsys_vtb. The <config_name> directory contains the configured RTL and the testbench. Other directories are explained Table G-1.

Figure G-9 gmac_axi_subsys_vtb Directory Structure

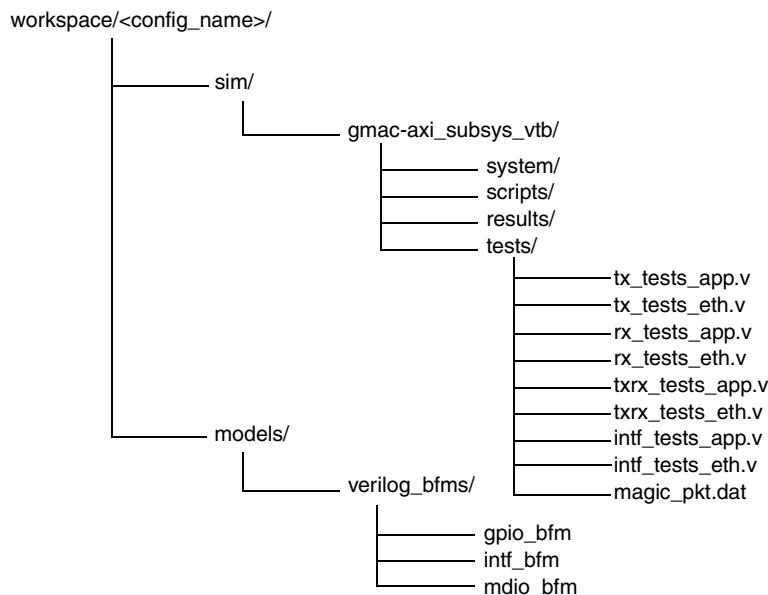


Table G-1 gmac_axi_subsys_vtb Directory Structure

File Name	Description
sim	Simulation directory containing the testbench, test vectors, and scripts. All simulations are run from this directory
system	Directory containing the testbench, the clock generation module, and the Ethernet VIP tasks, host tasks, host commands, etc.
scripts	Directory containing scripts for running test vectors.
results	This directory's test_log subdirectory is a repository for simulation logs.
tests	This directory contains all the transmit and receive test vectors. Test case files with an “_app” suffix contain the application-side code. Test case files with an “_eth” suffix control the Ethernet VIP.
gpio_bfm	This directory contains a GPIO model that the application and Ethernet sides use to communicate.
intf_bfm	This consists of an interface file used for proper port mapping of the ports, depending on the selected interface.
mdio_bfm	This directory contains the MDIO bfm and MDIO read-write tasks files for the RevMII model.

G.4 GPIO

The GPIO is used for communication between the host and the Ethernet VIP. The GPIO has two types of register sets for input ports (GPIO_IN) and output ports (GPIO_OUT). The register set is further classified by whether the test case controls frame transmission or reception.

The GPIO base address in the testbench is 0x0000_C000. The GPIO BFM implements a set of sixteen 32-bit registers to control the I/O ports, with each register bit corresponding to a port bit. The Bit 5 address indicates whether the register is used by the transmit test case (Bit 5 = 0) or the receive test case (Bit 5 = 1). Similarly, Bit 4 of the address decodes whether the register controls the output ports (Bit 4 = 0) or the input ports (Bit 4 = 1).

The address mapping for GPIO is as follows:

Table G-2 GPIO Address Map

Register	Address	Description
1	0000_C000	Transmission-side GPIO_OUT control register
2–4	0000_C004–0000_C00C	Reserved for transmission-side GPIO_OUT
5	0000_C010	Transmission-side GPIO_IN control register
6	0000_C014	Transmission-side GPIO_IN error_status [63:32]
7	0000_C018	Transmission-side GPIO_IN error_status [31:0]
8	0000_C01C	Reserved for Transmission-side GPIO_IN
9	0000_C020	Receive-side GPIO_OUT control register
10–12	0000_C024–0000_C02C	Reserved for Receive-side GPIO_OUT
13	0000_C030	Receive-side GPIO_IN control register
14–16	0000_C034	Reserved for Receive-side GPIO_IN

G.4.1 GPIO_OUT (Tx)

The GPIO_OUT structure for transmission is shown in [Figure G-10](#).

Figure G-10 GPIO_OUT Structure (Tx)



G.4.2 GPIO_IN (Tx)

The GPIO_IN structure for transmission is shown in [Figure G-11](#), GPIO_IN uses three 32-bit registers: one for the control variables, the other two for the error status received from the Ethernet VIP. GPIO_IN fields are as follows:

- ❖ eth_vip_ready: This bit is set when the Ethernet VIP is ready to receive the frame.
- ❖ frame_recieved: This bit is set when the Ethernet VIP has received the frame.

- ❖ Payload check status: The VIP sets this bit when it receives data that matches the expected data frame.
- ❖ rcvd_error_status[63:0]: The 64-bit error status generated by the Ethernet VIP for the received frame is stored in two 32-bit registers, as shown in [Figure G-11](#).

Figure G-11 **GPIO_IN Structure (Tx)**

eth_vip_ready [31]	frame_rcvd [30]	Payload check status [29]	Reserved [28:0]
rcvd_error_status [63:32]			
rcvd_error_status [31:0]			

G.4.3 **GPIO_OUT (Rx)**

The GPIO_OUT structure for transmission is shown in [Figure G-12](#).

Figure G-12 **GPIO_OUT Structure (Rx)**

app_ready[31]	Reserved[30]
---------------	--------------

G.4.4 **GPIO_IN (Rx)**

As shown in [Figure G-13](#), the GPIO_IN structure uses 32-bit registers of which it currently uses only the following port:

vip_rd_done: The Ethernet VIP sets this bit when it has read out GPIO_OUT.

Figure G-13 **GPIO_IN Structure (Rx)**

vip_rd_done[31]	Reserved[30:0]
-----------------	----------------

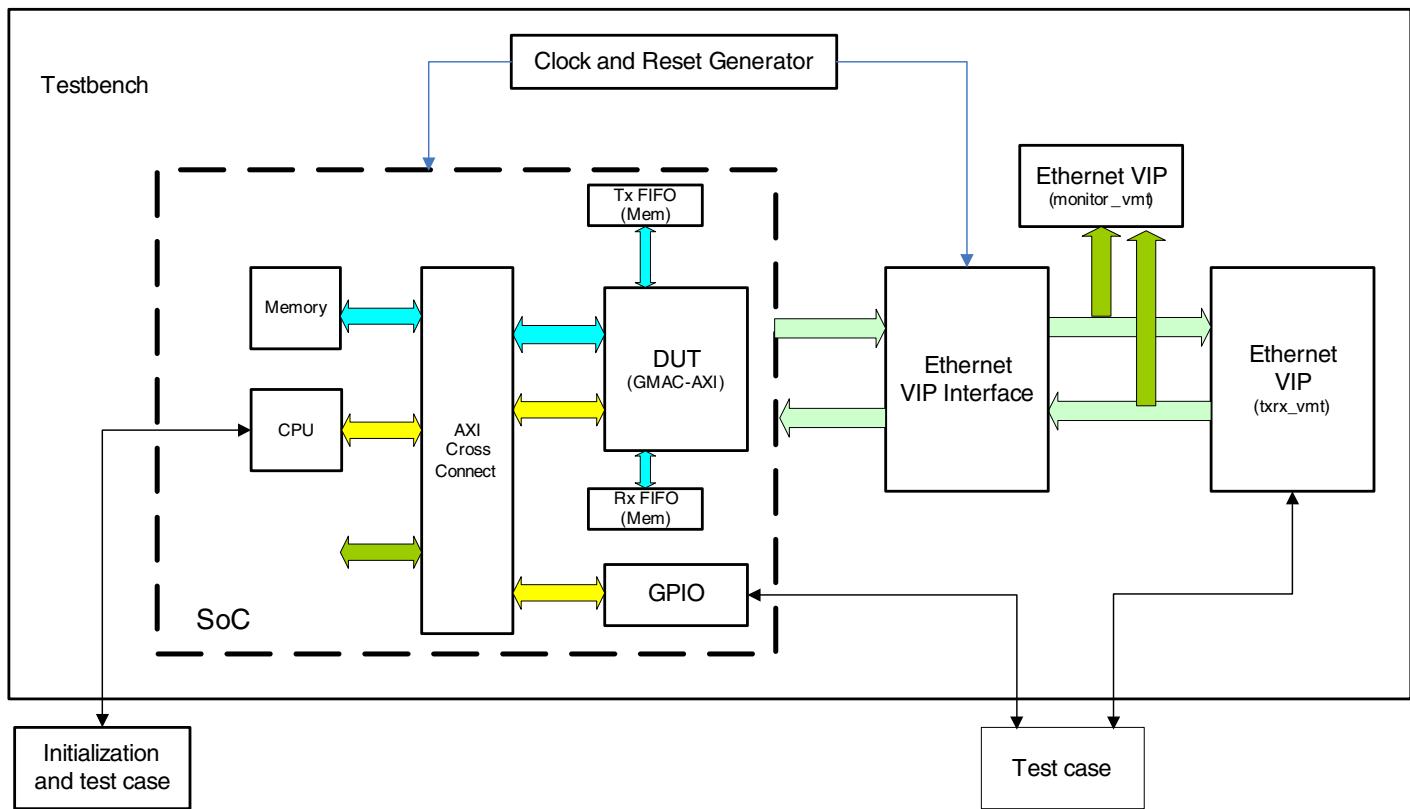
G.5 **SoC-Level Verification Guidelines**

This section provides guidelines for SoC-level verification. The test environment is depicted in [Figure G-14](#). In this environment, the SoC, which contains an ARM core (CPU) is the DUT. The application-side vectors are now created in software and loaded into the memory for the CPU to execute.

The verilog testcases given in this testbench can be easily ported to create test cases that can be used to verify SoC integration. The verification strategy must adhere to the following guidelines.

- ❖ In SoC-level verification, you must ensure that all communication between the application-side test case and the Ethernet-side test case passes through the GPIO only.
- ❖ You must rewrite the application-side test case in a high-level language, such as C, and compile it to create machine code. Load this code into the system memory (instruction code memory) of the SoC, then start the simulation. The CPU reads the instructions from memory and executes the test as indicated in the flow charts provided in “[Verification Flow](#)” on page [523](#).
- ❖ The Ethernet VIP is controlled only with the existing Ethernet-side test case in verilog.
- ❖ You can use the reserved fields in the GPIO to pass control data between the application and the Ethernet VIP.

Figure G-14 Setup for SoC-Level Verification



G.6 Running Simulations

To run the individual test case simulations from the command line, enter the following commands from the <config_name>/sim/ directory:

- ❖ For VCS enter:

```
./gmac_axi_subsys_vtb/scripts/run_vcs <TEST_CASE_NAME> <OTHER_OPTIONS>
```
- ❖ For NCV enter:

```
./gmac_axi_subsys_vtb/scripts/run_ncv <TEST_CASE_NAME> <OTHER_OPTIONS>
```
- ❖ For MTI enter:

```
./gmac_axi_subsys_vtb/scripts/run_mti <TEST_CASE_NAME> <OTHER_OPTIONS>
```

G.7 Simulation PASS/FAIL

The simulation run status is logged into the workspace: <config_name>/sim/runttest.log file. The details of the run can be accessed from the results/testlog/<TEST_NAME>.log file at the end of simulation.

H

Area and Power

H.1 Area

H.1.1 Gate Count Summary

This section summarizes the area in terms of gate count for the various modules/configurations listed in [Table H-1](#). Gate count was calculated with the following parameters:

DC Version:	2009.06-SP1
Technology synthesized for:	45n, 65n, 90n, 130n
Application clock (10/100 MAC only):	100 MHz
Application clock (Gigabit MAC):	167 MHz to 250 MHz
MII/GMII clock:	25/125 MHz
Gate size:	NAND2x1
Data bus width:	32 bits

Table H-1 Gate Count

Sr No.	Blocks	Gate Count
1	MAC core (10/100 only)	13–15K
2	GMAC Core (1000 only)	14–17K
3	GMAC Core (10/100/1000)	14–17K
4	Disable Half Duplex	~(2K)
5	Disable Hash Filter	~(0.8K)

Note: By disabling the Half Duplex or Hash Filter functions in coreConsultant, you can reduce the gate count.

Table H-1 Gate Count (Continued)

Sr No.	Blocks	Gate Count
6	MMC (RMON with all 49 MAC counters) module MMC (RMON with all 77 counters) module MMC with all 79 counters as 16 bit	22-24K 42-45K (20K)
7	PMT (Power Mgmt) module	7-8K
8	Optional module MAC Address Reg (31)	18-19K
9	Double-synchronization Address Reg (all 32)	18-19K
10	IP Checksum module (Type 1) Receive Checksum Offload (Type 2)	~ 1K ~3K
11	SMA module	~ 0.9K
12	Double-synchronization Hash/VLAN/PT registers	~ 1K
13	RGMII	~ 0.35K
14	RevMII	~1.3K
15	RMII	~ 0.5K
16	SMII	~0.7K
17	TBI + RTBI + SGMII	6-8K
18	MTL (default FIFO layer)	6.5-8K
19	TxCOE (Checksum Offload Engine)	~6K
20	MDC (DMA) module	~12K
21	IEEE-1588 TimeStamping - Default with external time reference (GMAC-AHB)	~4.5K
22	IEEE-1588 TimeStamping - Advanced with external time reference (GMAC-AHB)	~6K
23	IEEE-1588 TimeStamping - with internal time reference (add to above)	~6K
24	IEEE-1588 TimeStamping - Auxiliary Snapshot feature	~3K
25	IEEE-1588 TimeStamping - Advanced - Higher 16-bit for seconds time	~200
26	AV Feature - One additional Transmit channel (with DMA)	~16K
27	AV Feature - Two additional Transmit channels	~30K
28	AV Feature - One additional Receive channel	~9K
29	AV Feature - Two additional Receive channels	~17K
30	EEE Feature	1K
31	Separate CSR clock in DMA	~4.5K
32	AHB (interface) module	3-4K

Table H-1 Gate Count (Continued)

Sr No.	Blocks	Gate Count
33	AXI Master interface	7-8K
34	AXI Slave interface	~3K
35	MAC (10/100) Subsystem with AHB (Min config)	~ 35K
36	Gigabit MAC Subsystem with AHB (Min config)	~ 37K
37	Gigabit MAC Subsystem with AXI (Max config = Enable all)	~225K

The parentheses () in the Gate Count column indicate reduced gate count.

[Table H-1](#) gives the gate count numbers in 32-bit data bus configuration. [Table H-2](#) gives the increase in approximate area because of a 64-bit or 128-bit data bus configuration (in affected blocks only, with respect to the numbers given in [Table H-1](#)).

Table H-2 Area Increases in 64-Bit and 128-Bit Data Bus Configurations

Data Bus	Module	Gate Count
64-bit	AHB (interface)	1K
	AXI Master interface	3K
	AXI Slave interface	1K
	MDC (DMA) module	3K
	MTL (default FIFO layer)	2K
	MTL (with Tx checksum offload)	6K
128-bit	GMAC core	0.8K
	AHB (interface)	3K
	AXI Master interface	9K
	AXI Slave interface	3K
	MDC (DMA) module	10K
	MTL (default FIFO layer)	6K
	MTL (with Tx checksum offload)	16K
	GMAC core	2.5K

With all features enabled, the Gigabit MAC Subsystem with AXI has approximately 250K gates in the 64-bit data bus configuration and 300K gates in the 128-bit data bus configuration.

H.1.2 Area Differences Due to Scan Ready and Clock Gating

[Table H-4](#) shows the area differences because of scan ready and clock gating for two configurations.

Table H-3 Sample Configuration for Estimating Area

Main Parameter	Configuration 1	Configuration 2
Core Features	<ul style="list-style-type: none"> • 10/100 Mbps only • Async Reset • System interface = AHB • 32-bit, little-endian • MTL Rx/Tx FIFO = 2K • PHY IF = MII only • 1 MAC address • Optional modules = SMA 	<ul style="list-style-type: none"> • 10/100/1000 Mbps • Async Reset • System interface = AHB • 32-bit, little-endian • MTL Rx/Tx FIFO = 2K • PHY IF = GMII only • 1 MAC address • Optional modules = SMA
Technology	TSMC 90G	TSMC 90G
Gate Size	NAND2X1 cells (area 2.42)	NAND2X1 cells (area 2.42)
Application Clock Frequency	100 MHz	200 MHz
PHY Clock Frequency	25 MHz	125 MHz

Table H-4 Area Differences Because of Scan Ready and Clock Gating

Clock Gating	Percentage Gating	Scan Ready	Test Coverage	Gate Count
Configuration 1				
No	NA	No	N/A	39,850
No	NA	Yes	99.35%	46,504
Yes	78.7	No	N/A	34,687
Yes	78.7	Yes	99.51%	40,840
Configuration 2				
No	NA	No	N/A	41,158
No	NA	Yes	99.36%	48,030
Yes	78.9	No	N/A	35,783
Yes	78.9	Yes	99.48%	42,159

Tetramax, version 2008.09, is used for test coverage estimation.

H.2 Power

Table H-6 shows the power dissipation estimates with and without clock gating. The power estimates are for simulations with continuous transmission and reception of about 100 frames each, at the Application clock frequency given in Table H-6.

Table H-5 Sample Configuration for Estimating Power Dissipation

Main Parameter	Configuration 1	Configuration 2
Core Features	<ul style="list-style-type: none"> • 10/100 Mbps only • Async Reset • System interface = AHB • 32-bit, little-endian • MTL Rx/Tx FIFO = 2K • PHY Interface = MII only • 1 MAC address • Optional modules = SMA 	<ul style="list-style-type: none"> • 10/100/1000 Mbps • Async Reset • System interface = AHB • 32-bit, little-endian • MTL Rx/Tx FIFO = 2K • PHY Interface = GMII only • 1 MAC address • Optional modules = SMA
Technology	TSMC 90G	TSMC 90G
Global Operating Voltage	0.9 volts	0.9 volts
Gate Size	NAND2X1 cells (area 2.42)	NAND2X1 cells (area 2.42)
Application Clock Frequency	100 MHz	200 MHz
PHY Clock Frequency	25 MHz	125 MHz

Table H-6 Power Dissipation with and without Clock Gating

Clock Gating	Gate Count	Comb / Seq Counts	Dyn Power (mW)	Static Power (mW)
Configuration 1				
No	39,848	8077 / 3115	1.56	0.344
Yes	34,687	8395 / 3303	0.564	0.313
Configuration 2				
No	41,158	8499 / 3204	2.953	0.354
Yes	35,783	8829 / 3396	1.214	0.320

Prime Power, version 2008.09, is used for power estimation.

Programming Guide

This appendix provides the instructions for initializing the DMA/MAC registers in the proper sequence. It contains the following sections:

- ❖ “[Initializing DMA](#)” on page [547](#)
- ❖ “[Initializing MAC](#)” on page [548](#)
- ❖ “[Performing Normal Receive and Transmit Operation](#)” on page [549](#)
- ❖ “[Stopping and Starting Transmission](#)” on page [550](#)
- ❖ “[Programming Guidelines for GMII Link Transitions](#)” on page [550](#)
- ❖ “[Programming Guidelines for IEEE 1588 Time Stamping](#)” on page [551](#)
- ❖ “[Programming Guidelines for AV Feature](#)” on page [552](#)
- ❖ “[Programming Guidelines for Energy Efficient Ethernet](#)” on page [554](#)

I.1 Initializing DMA

This initialization sequence is used for GMAC configurations without Audio Video (AV) feature. Perform the following steps to initialize the DMA:

1. Provide a software reset. This resets all of the GMAC internal registers and logic. ([DMA Register 0 \(Bus Mode Register\)](#) – bit 0).
2. Wait for the completion of the reset process (poll bit 0 of the [DMA Register 0 \(Bus Mode Register\)](#), which is only cleared after the reset operation is completed).
3. Program the following fields to initialize the Bus Mode Register by setting values in [DMA Register 0 \(Bus Mode Register\)](#):
 - a. Mixed Burst and AAL (only if GMAC-AHB/GMAC-AXI configuration is selected)
 - b. Fixed burst or undefined burst
 - c. Burst length values and burst mode values.
 - d. Descriptor Length (only valid if Ring Mode is used)
 - e. Tx and Rx DMA Arbitration scheme
4. Program the AXI Interface options in [Register 10 \(AXI Bus Mode Register\)](#). If fixed burst-length is enabled, then select the maximum burst-length possible on the AXI bus (bits[7:1]).

5. Create a proper descriptor chain for transmit and receive. In addition, ensure that the receive descriptors are owned by DMA (bit 31 of descriptor should be set). When OSF mode is used, at least two descriptors are required.
For more information about descriptors, see “[Descriptors](#)” on page [397](#).
6. Make sure that your software creates three or more different transmit or receive descriptors in the chain before reusing any of the descriptors.
7. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor ([Register 3 \(Receive Descriptor List Address Register\)](#) and [Register 4 \(Transmit Descriptor List Address Register\)](#) respectively).
8. Program the following fields to initialize the mode of operation in [Register 6 \(Operation Mode Register\)](#)
 - a. Receive and Transmit Store And Forward
 - b. Receive and Transmit Threshold Control (RTC and TTC)
 - c. Hardware Flow Control enable
 - d. Flow Control Activation and De-activation thresholds for MTL Receive and Transmit FIFO (RFA and RFD)
 - e. Error Frame and undersized good frame forwarding enable
 - f. OSF Mode
9. Clear the interrupt requests, by writing to those bits of the status register (interrupt bits only) that are set. For example, by writing 1 into bit 16, the normal interrupt summary clears this bit ([DMA Register 5 \(Status Register\)](#)).
10. Enable the interrupts by programming the [Register 7 \(Interrupt Enable Register\)](#).
11. Start the Receive and Transmit DMA by setting SR (bit 1) and ST (bit 13) of the control register ([DMA Register 6 \(Operation Mode Register\)](#)).



For information about the DMA initialization steps for configurations with AV feature, see “[Initializing the DMA](#)” on page [552](#).

I.2 Initializing MAC

The following MAC Initialization operations can be performed after DMA initialization. If the MAC initialization is done before the DMA is set-up, then enable the MAC receiver (last step below) only after the DMA is active. Otherwise, received frames fills the RxFIFO and overflow. Steps (1) and (2) are to be followed if the TBI, SGMII, or RTBI PHY interface is enabled, otherwise follow steps (3) and (4).

1. Program the GMAC [Register 48 \(AN Control Register\)](#) to enable Auto-negotiation ANE (bit-12). Setting ELE (bit-14) of this register enables the PHY to loop back the transmit data and RAN (bit-9) can be set to restart Auto negotiation (only for TBI, SGMII, and RTBI PHY interfaces).
2. Check the GMAC [Register 49 \(AN Status Register\)](#) for completion of the Auto-negotiation process. ANC (bit-5) should be set. The link status (bit-2), when set, indicates that the link is up (only for TBI, SGMII, and RTBI PHY interfaces).
3. Program the GMAC [Register 4 \(GMII Address Register\)](#) for controlling the management cycles for external PHY. For example, Physical Layer Address PA (bits 15-11). In addition, set bit 0 (GMII Busy) for writing into PHY and reading from PHY.

4. Read the 16-bit data of [Register 5 \(GMII Data Register\)](#) from the PHY for link up, speed of operation, and mode of operation, by specifying the appropriate address value in bits 15-11 of [Register 4 \(GMII Address Register\)](#).
5. Provide the MAC address registers ([Register 16 \(MAC Address0 High Register\)](#) and [Register 17 \(MAC Address0 Low Register\)](#)). If more than one MAC address is enabled in your configuration (during configuration in coreConsultant), program the MAC addresses appropriately.
6. If Hash filtering is enabled in your configuration, program [Register 2 \(Hash Table High Register\)](#) and [Register 3 \(Hash Table Low Register\)](#).
7. Program the following fields to set the appropriate filters for the incoming frames in [Register 1 \(MAC Frame Filter\)](#):
 - a. Receive All
 - b. Promiscuous mode
 - c. Hash or Perfect Filter
 - d. Unicast, multicast, broadcast, and control frames filter settings
8. Program the following fields for proper flow control in [Register 6 \(Flow Control Register\)](#):
 - a. Pause time and other pause frame control bits
 - b. Receive and Transmit Flow control bits
 - c. Flow Control Busy/Backpressure Activate
9. Program the Interrupt Mask register bits, as required, and if applicable, for your configuration.
10. Program the appropriate fields in [Register 0 \(MAC Configuration Register\)](#). For example, Inter-frame gap while transmission and jabber disable. Based on the Auto-negotiation you can set the Duplex mode (bit 11) or port select (bit 15).
11. Set the bits Transmit enable (TE bit-3) and Receive Enable (RE bit-2) in [Register 0 \(MAC Configuration Register\)](#).

I.3 Performing Normal Receive and Transmit Operation

For normal operation, perform the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the Host (either transmit or receive).
2. Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and issuing a poll demand by writing 0 into the Tx/Rx poll demand register ([Register 1 \(Transmit Poll Demand Register\)](#) and [Register 2 \(Receive Poll Demand Register\)](#)).
4. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process ([Register 18 \(Current Host Transmit Descriptor Register\)](#) and [Register 19 \(Current Host Receive Descriptor Register\)](#)).
5. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process ([Register 20 \(Current Host Transmit Buffer Address Register\)](#) and [Register 21 \(Current Host Receive Buffer Address Register\)](#)).

I.4 Stopping and Starting Transmission

Perform the following steps to pause the transmission for some time:

1. Disable the Transmit DMA (if applicable), by clearing bit 13 ([ST: Start/Stop Transmission Command](#)) of [Register 6 \(Operation Mode Register\)](#).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of [Register 9 \(Debug Register\)](#).
3. Disable the MAC transmitter and MAC receiver by clearing the bit 3 ([TE: Transmitter Enable](#)) and bit 2 ([RE: Receiver Enable](#)) in [Register 0 \(MAC Configuration Register\)](#).
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading Register 9 (Debug Register)).
5. Make sure that both Tx FIFO and Rx FIFO are empty.
6. To re-start the operation, first start the DMAs, and then enable the MAC Transmitter and Receiver.

I.5 Programming Guidelines for GMII Link Transitions

I.5.1 Transmit and Receive Clocks are Running when the Link is Down

Perform the following steps when the link is down but the Transmit and Receive clocks are running:

1. Disable the Transmit DMA (if applicable), by clearing bit 13 (ST) of [Register 6 \(Operation Mode Register\)](#).
2. Disable the MAC receiver by clearing the bit 2 (RE) of [Register 0 \(MAC Configuration Register\)](#).
3. Wait for any previous frame transmissions to complete from the Tx FIFO. You can do this by reading the appropriate bits of [Register 9 \(Debug Register\)](#).
-or-
Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing bit 3 (TE) in [Register 0 \(MAC Configuration Register\)](#).
5. After the link is up, read the PHY registers to know the latest configuration and accordingly program the MAC registers.
6. Restart the operation by starting the Tx DMA, and then enabling the MAC Transmitter and Receiver. You do not need to disable the Rx DMA. As the Receiver is disabled, the FIFO does not get any data in the Rx FIFO.

I.5.2 Transmit and Receive Clocks are Stopped when the Link is Down

Perform the following steps when the link is down and the Transmit and Receive clocks are stopped:

1. Wait till the link is up and the Transmit and Receive clocks are active.
When the Transmit and Receive clocks are stopped, then disabling the transmit or receive operations does not have any effect. Therefore, the software must wait till the link is up again.
2. Disable the Transmit DMA (if applicable), by clearing bit 13 (ST) of the [Register 6 \(Operation Mode Register\)](#).
3. Disable the MAC receiver by clearing the bit 2 (RE) of [Register 0 \(MAC Configuration Register\)](#).

4. Wait for any previous frame transmissions to complete from the Tx FIFO. You can do this by reading the appropriate bits of [Register 9 \(Debug Register\)](#).
- or-
- Flush the Tx FIFO for faster empty operation.
5. Disable the MAC transmitter by clearing bit 3 (TE) in [Register 0 \(MAC Configuration Register\)](#).
 6. After the link is up, read the PHY registers to know the latest configuration and accordingly program the MAC registers.
 7. Restart the operation by starting the Tx DMA, and then enabling the MAC Transmitter and Receiver

I.6 Programming Guidelines for IEEE 1588 Time Stamping

I.6.1 Initialization Guideline for System Time Generation

You can enable the timestamp feature by setting bit 0 of the Timestamp control register. However, it is essential that the timestamp counter should be initialized after this bit is set. Perform the following steps during GMAC core initialization:

1. Mask the Timestamp Trigger interrupt by setting the bit 9 of [Register 15 \(Interrupt Mask Register\)](#).
 2. Program the bit 0 in [Register 448 \(Timestamp Control Register\)](#) to enable time stamping.
 3. Program the [Register 449 \(Sub-Second Increment Register\)](#) based on the PTP clock frequency.
 4. If you are using the Fine Correction approach, program the [Register 454 \(Timestamp Addend Register\)](#) and set the bit 5 of Register 448 (Timestamp Control Register).
 5. Poll the Timestamp Control register until the bit 5 is cleared.
 6. Program the Timestamp Control register bit 1 to select the Fine Update method (if required).
 7. Program [Register 452 \(System Time - Seconds Update Register\)](#) and [Register 453 \(System Time - Nanoseconds Update Register\)](#) with the appropriate time value.
 8. Set the bit 2 in Register 448 (Timestamp Control Register).
- The Timestamp counter starts operation as soon as it is initialized with the value written in the Timestamp Update registers.
9. Enable the MAC receiver and transmitter for proper time stamping.



Note If timestamp operation is disabled by clearing bit 0 of Register 448 (Timestamp Control Register), you need to repeat all these steps to restart the timestamp operation.

I.6.2 System Time Correction

To synchronize or update the system time in one process (coarse correction method), perform the following steps:

1. Set the offset (positive or negative) in the Timestamp Update registers ([Register 452 \(System Time - Seconds Update Register\)](#) and [Register 453 \(System Time - Nanoseconds Update Register\)](#)).
2. Set bit 3 (TSUPDT) of the [Register 448 \(Timestamp Control Register\)](#).
3. The value in the Timestamp Update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), perform the following steps:

1. With the help of the algorithm explained in “[System Time Register Module](#)” on page [117](#), calculate the rate by which you want to make the system time increments slower or faster.
2. Update the [Register 454 \(Timestamp Addend Register\)](#) with the new value and set the bit 5 of the [Register 448 \(Timestamp Control Register\)](#).
3. Wait for the time for which you want the new value of the Addend register to be active. You can do this by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in [Register 455 \(Target Time Seconds Register\)](#) and [Register 456 \(Target Time Nanoseconds Register\)](#).
5. Unmask the Timestamp interrupt by clearing bit 9 of [Register 15 \(Interrupt Mask Register\)](#).
6. Set bit 4 in Register 448 (Timestamp Control Register).
7. When this trigger causes an interrupt, read [Register 14 \(Interrupt Status Register\)](#).
8. Reprogram Register 454 (Timestamp Addend Register) with the old value and set bit 5 again.

I.7 Programming Guidelines for AV Feature

I.7.1 Initializing the DMA

This initialization sequence is used only for GMAC-AHB configuration with AV feature. Perform the following steps to initialize the DMA:

1. Provide a software reset to reset all GMAC internal registers and logic (bit 0 in [Register 0 \(Bus Mode Register\)](#)).
2. Wait for the completion of the reset process. Poll bit 0 of the Register 0 (Bus Mode Register), which is cleared only after the reset operation is completed.
3. Program the following fields to initialize the Bus Mode Register by setting the values in Register 0 (Bus Mode Register)
 - a. Mixed Burst and AAL (only if GMAC-AHB/GMAC-AXI configuration is selected)
 - b. Fixed burst or undefined burst
 - c. Burst length values and burst mode values
 - d. Descriptor Length (only valid if Ring Mode is used)
 - e. Tx and Rx DMA Arbitration scheme and two level priority weight for the channel
4. Create a proper descriptor chain for transmit and receive. In addition, ensure that the DMA owns the receive descriptors by setting the bit 31 of the descriptor. When OSF mode is used, at least two descriptors are required.

For more information about descriptors, see “[Descriptors](#)” on page [397](#).

5. Make sure that your software creates three or more different transmit or receive descriptors in the chain before reusing any of the descriptors.
6. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor ([Register 3 \(Receive Descriptor List Address Register\)](#) and [Register 4 \(Transmit Descriptor List Address Register\)](#), respectively).

7. Program the following fields to initialize the mode of operation by setting the values in DMA Register 6 (Operation Mode Register):
 - a. Receive and Transmit Store And Forward
 - b. Receive and Transmit Threshold Control (RTC and TTC)
 - c. Error Frame and undersized good frame forwarding enable
 - d. OSF Mode
8. Clear the interrupt requests, by writing to those bits of the status register (interrupt bits only) that are set. For example, writing 1 into bit 16 (normal interrupt summary) clears this bit ([Register 5 \(Status Register\)](#)).
9. Enable the interrupts by programming the [Register 7 \(Interrupt Enable Register\)](#).
10. Repeat steps 3 through 9 for all additional channels of AV feature.
11. Program the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of channel 1 and channel 2.
12. Start the Receive and Transmit DMA by setting SR (bit 1) and ST (bit 13) of the control registers (DMA Register 6, Register 70 and Register 134 operation mode register) for all the channels.

I.7.2 Enabling Slot Number Checking

You can use the slot number check feature to specify the intervals at which the channel 1 or channel 2 DMA fetches the frames from the AHB/AXI system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the host memory. The feature is available only when you enable time stamping and program the [Register 449 \(Sub-Second Increment Register\)](#). Perform the following steps to enable the slot number checking:



Note These steps should be performed after [Step 11](#) and before [Step 12](#) of “[Initializing the DMA](#)” on page [552](#).

1. Enable time stamping by following the steps described in “[Initialization Guideline for System Time Generation](#)” on page [551](#).
2. Make sure that the SLOTNUM field (bits 6:3) of [Transmit Descriptor Word 0 \(TDES0\)](#) contains a valid slot number. You can read the current reference slot number from the Slot Function Control and Status register.
3. Set the bit 0 ([ESC: Enable Slot Comparison](#)) of the Slot Function Control and Status register of a channel to enable the slot number checking.

I.7.3 Enabling Average Bits Per Slot Reporting

The CBS Status register of the additional AV channels (channel 1 and channel 2) provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Perform the following steps to enable average bits per slot reporting:

1. Enable time stamping by following the steps described in “[Initialization Guideline for System Time Generation](#)” on page [551](#).
2. Program the bits 6:4 ([SLC: Slot Count](#)) of the CBS Control register of a channel with number of slots over which the average transmitted bits per slot need to be computed.

3. Enable the bit 17 ([ABPSSIE: Average Bits Per Slot Interrupt Enable](#)) of the CBS Control register of a channel to generate the average bits per slot interrupt.



- Note**
- The frequency of this interrupt depends on the value programmed in the [Step 2](#). For example, when you program value 0 in the SLC field, the interrupt is generated at every 125 microsecond.
 - When not required, you can disable this interrupt to stop the interrupt flooding.

4. Read the bits 16:0 ([ABS: Average Bits per Slot](#)) from the CBS Status register of a channel on each interrupt.



Note The software can read the ABS bits in polling mode even if the ABPSSIE bit is not enabled. When high, bit 17 ([ABSU: ABS Updated](#)) of the CBS Status register indicates that a new value is updated in the ABS field.

I.8 Programming Guidelines for Energy Efficient Ethernet

You can configure the Energy Efficient Ethernet (EEE) feature in coreConsultant. Perform the following steps during GMAC core initialization:

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop RX clock in LPI mode.)
3. Program the bits [5:16] ([LIT: LPI LS TIMER](#)) and bits [15:0] ([TWT: LPI TW TIMER](#)) in [Register 13 \(LPI Timers Control Register\)](#).
4. Read the link status of the PHY chip by using the MDIO interface and update the bit 17 (PLS) of [Register 12 \(LPI Control and Status Register\)](#) accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Set the bit 16 ([LPIEN: LPI Enable](#)) of [Register 12 \(LPI Control and Status Register\)](#) to make the MAC enter the LPI state.

The MAC enters the LPI mode after completing the transmission in progress and sets the bit 0 ([TLPIEN: Transmit LPI Entry](#)).



- Note**
- If you want to make the MAC enter the LPI state only after it completes the transmission of all queued frames in the TxFIFO, you should stop the DMA before setting the LPIEN bit. For information about how to stop the DMA, see [steps 1 and 2](#) in [“Stopping and Starting Transmission”](#) on page [550](#).
 - If you want to switch off the CSR clock, GMII transmit clock, or power to the rest of the system during the LPI state, you should wait for the TLPIEN interrupt of [Register 12 \(LPI Control and Status Register\)](#) to be generated. Restore the clocks before performing the [Step 6](#) when you want to come out of the LPI state.

6. Reset the bit 16 ([LPIEN: LPI Enable](#)) of [Register 12 \(LPI Control and Status Register\)](#) to bring the MAC out of the LPI state.

The MAC waits for the time programmed in the bits [15:0] ([TWT: LPI TW TIMER](#)) before setting the TLPIEX interrupt status bit and resuming the transmission.

J

Synchronizer Methods

This appendix describes the synchronizer methods (blocks of synchronizer functionality) that are used in GMAC-UNIV to cross clock boundaries.

This appendix contains the following sections:

- ❖ “[Synchronizer 1: Simple Double Register Synchronizer \(DWC_gmac_bcm21.v\)](#)” on page [555](#)
- ❖ “[Synchronizer 2: Pulse Synchronizer \(DWC_gmac_bcm22.v\)](#)” on page [559](#)
- ❖ “[Synchronizer 3: Pulse Synchronizer with Acknowledge \(DWC_gmac_bcm23.v\)](#)” on page [559](#)
- ❖ “[Synchronizer 4: Gray Coded Synchronizer \(DWC_gmac_bcm24.v\)](#)” on page [561](#)

The GMAC-UNIV uses a few more components like Asynchronous register FIFO (DWC_gmac_async_fifo) and bus synchronizers (DWC_gmac_bus_synczr). These components are built with some of the above synchronizers for the CDC signals.



Note The DesignWare Building Blocks (DWBB) contain several synchronizer components with functionality similar to methods documented in this appendix. For more information about DWBB synchronizer components, refer to this page:

www.synopsys.com/products/designware/docs/doc/dwf/datasheets/interface_cdc_overview.pdf

J.1 Synchronizer 1: Simple Double Register Synchronizer (DWC_gmac_bcm21.v)

This is a single clock data bus synchronizer for synchronizing data that crosses asynchronous clock boundaries. Your core may have parameters that allow you to configure for the number of synchronizing stages (2 or 3), the style of first-stage capturing flip-flop needed (negative or positive edge-triggered), and insertion of 'hold latches' to facilitate scan testing.

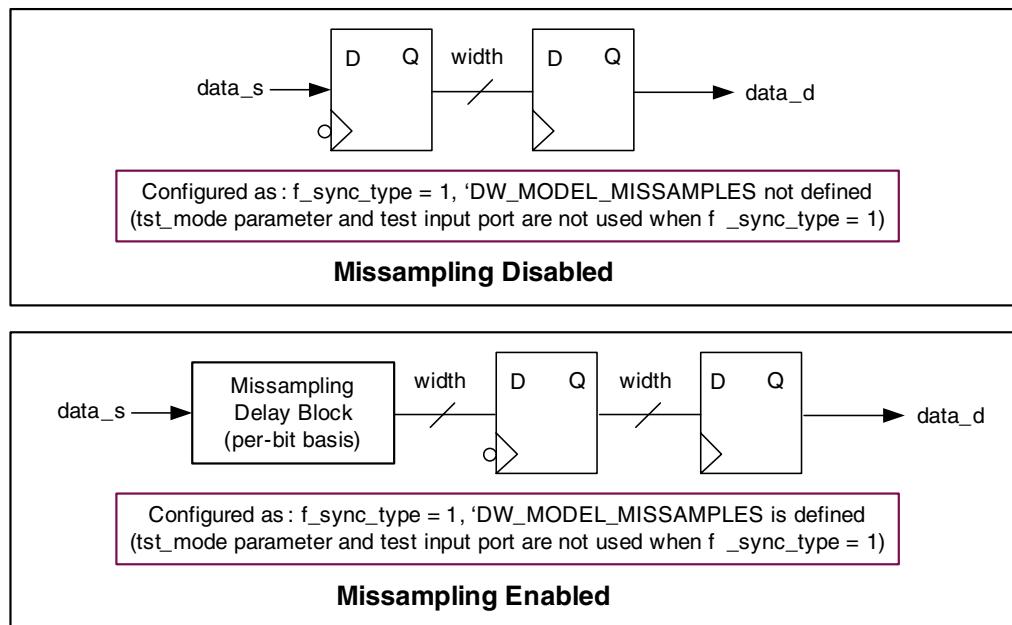
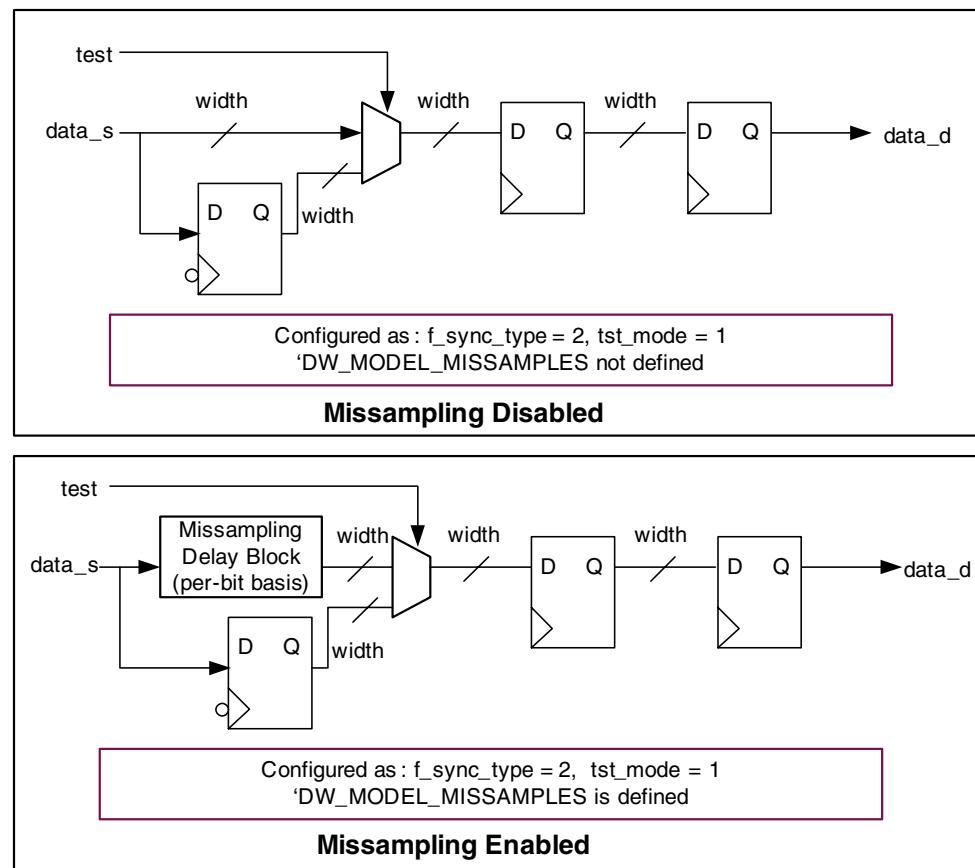
Figure J-1 Synchronizer 1: Synchronization Type 1**Figure J-2 Synchronizer 1: Synchronization Type 2**

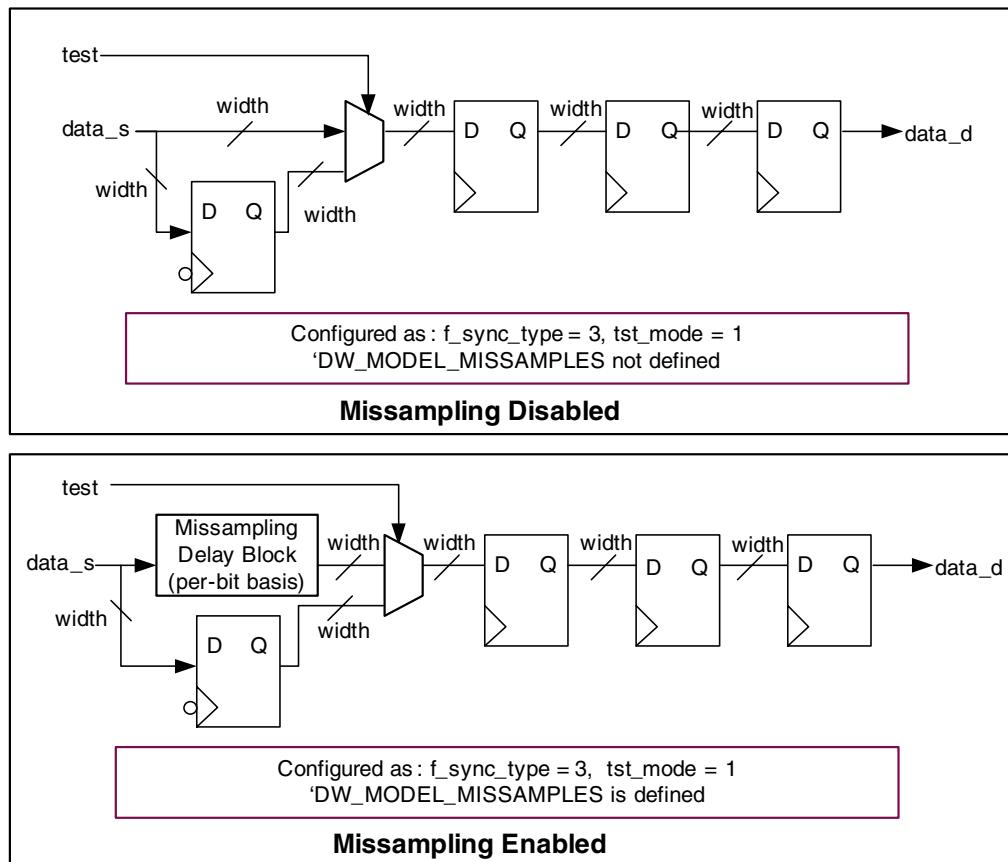
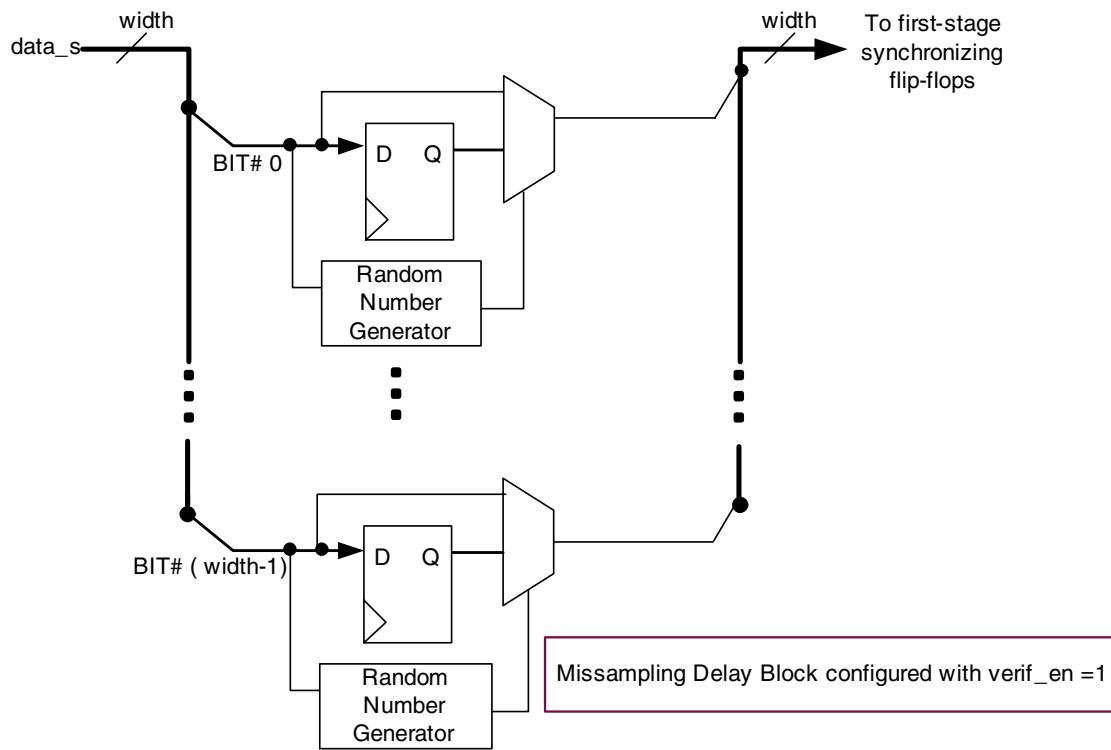
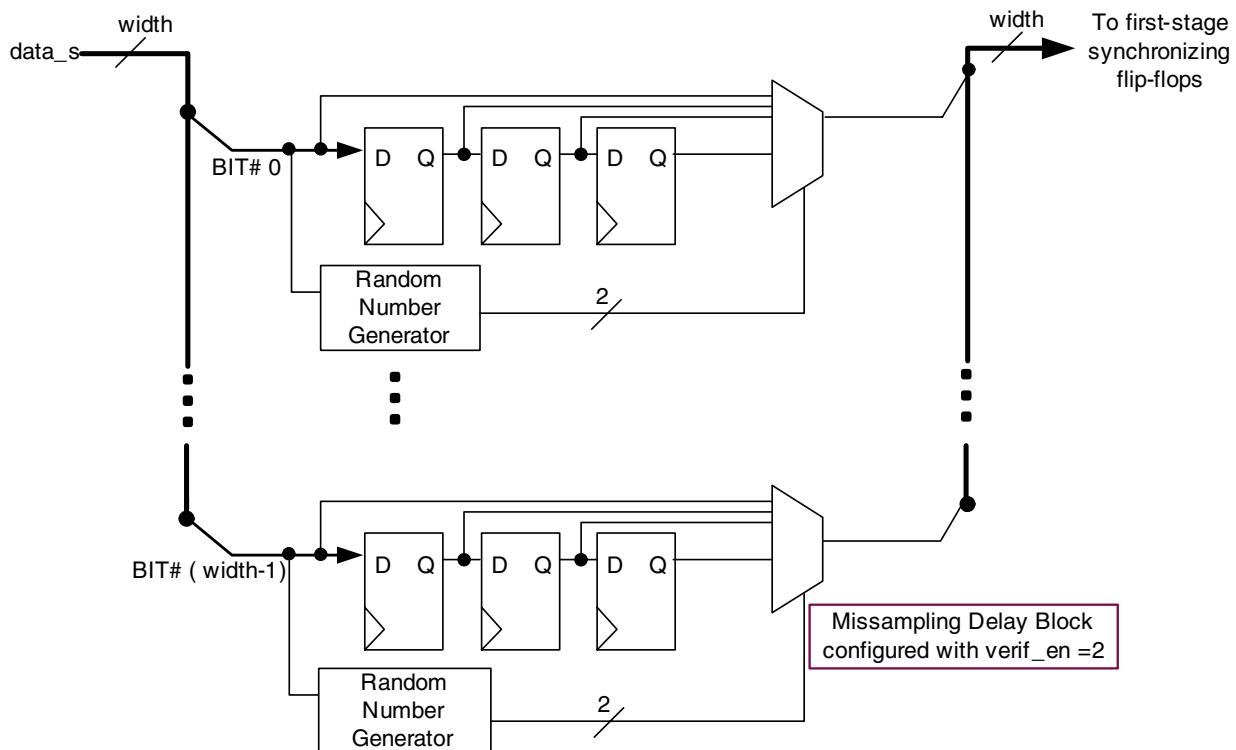
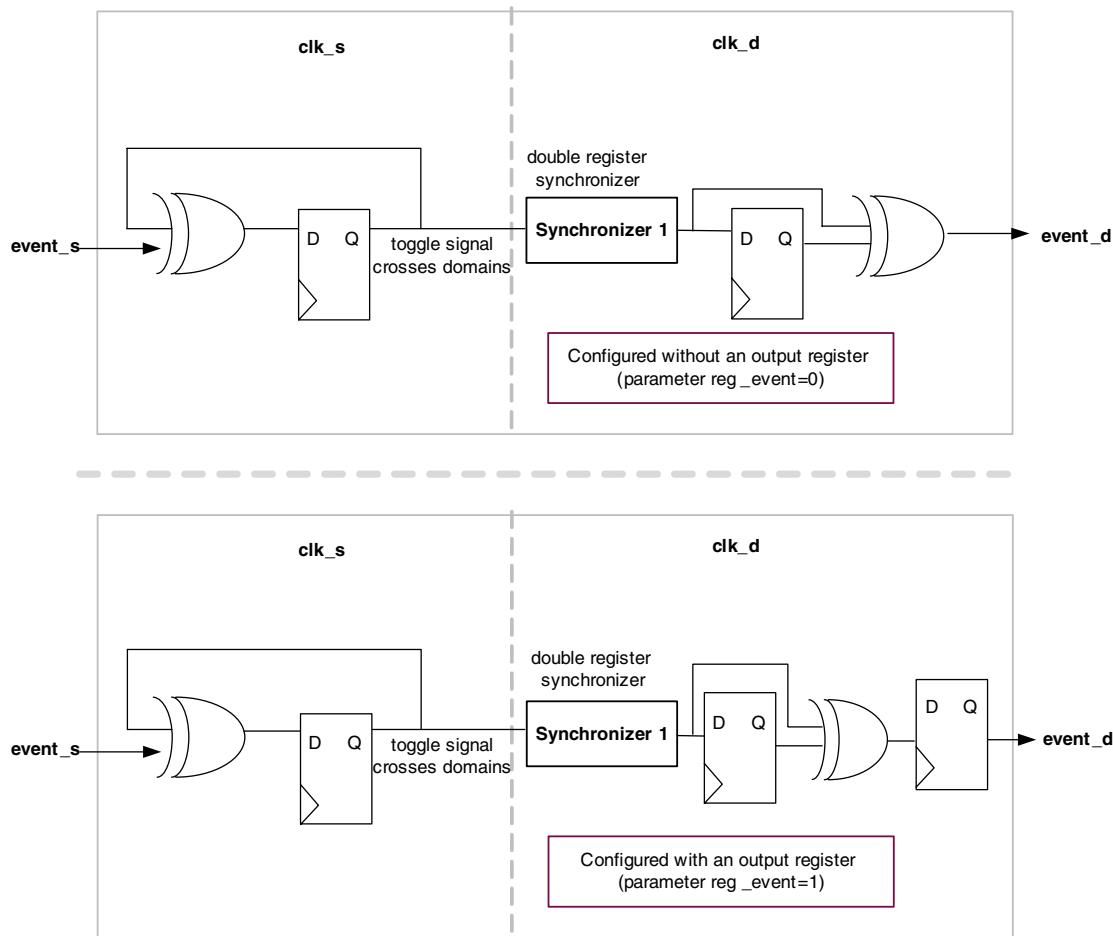
Figure J-3 Synchronizer 1: Synchronization Type 3**Figure J-4 Synchronizer 1: Missampling Model Type 0**

Figure J-5 Synchronizer 1: Missampling Model Type 1**Figure J-6 Synchronizer 1: Missampling Model Type 2**

J.2 Synchronizer 2: Pulse Synchronizer (DWC_gmac_bcm22.v)

This is a dual clock pulse synchronizer for transmitting single clock cycle pulses between two different clock domains. This method uses clock domain crossing techniques to safely transfer pulses between logic operating on different clocks.

Figure J-7 Synchronizer 2 Block Diagram



J.3 Synchronizer 3: Pulse Synchronizer with Acknowledge (DWC_gmac_bcm23.v)

This synchronizer passes an event (represented by a single clock cycle pulse) from the source domain to the destination domain and passes an acknowledge. This synchronizer is used within other synchronizers to implement handshake mechanisms. It is constructed with feedback and an interlock.

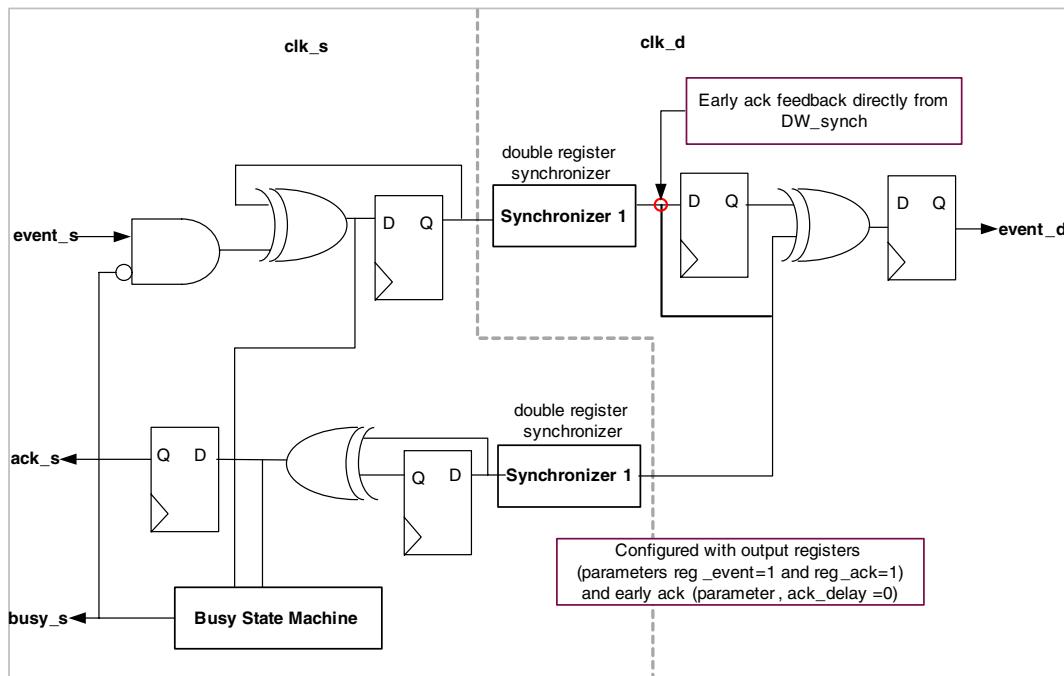
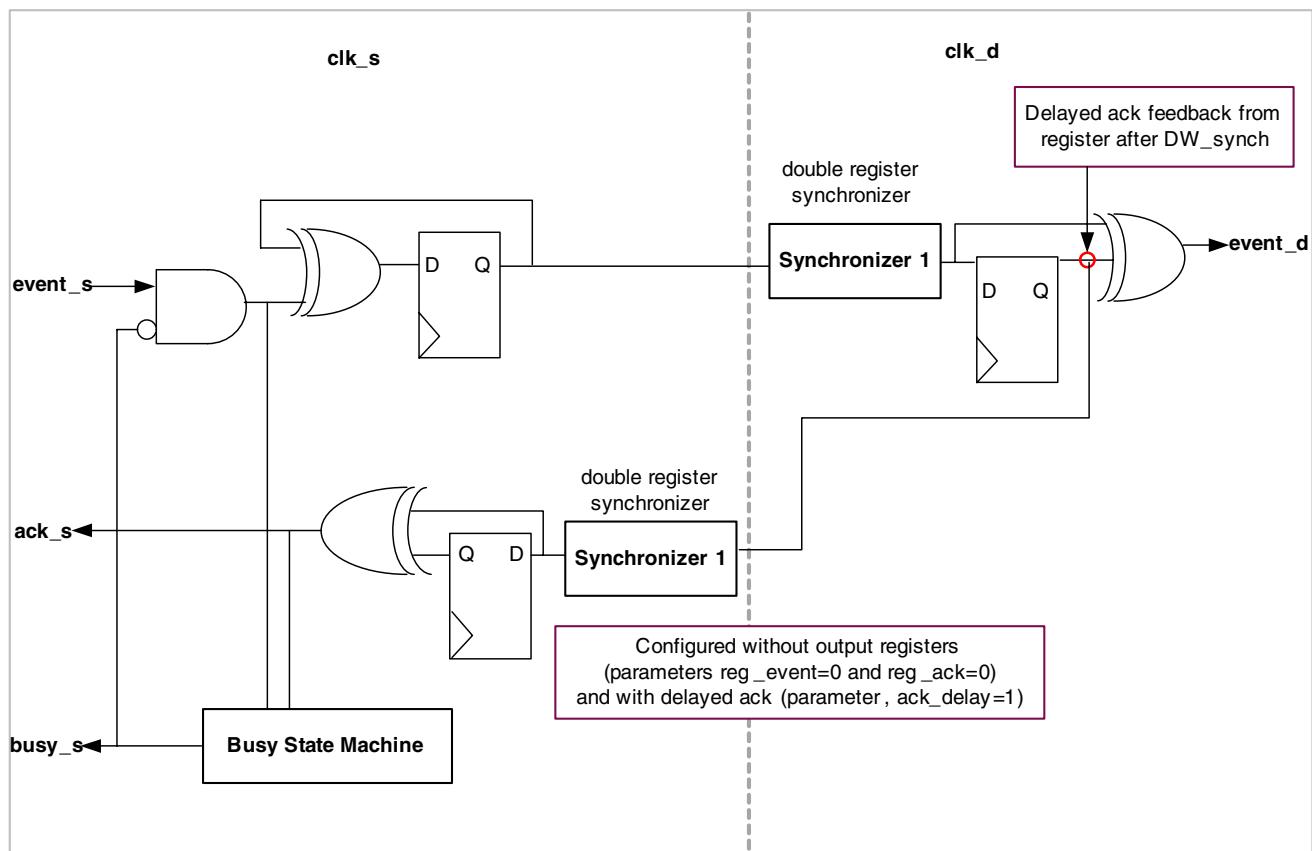
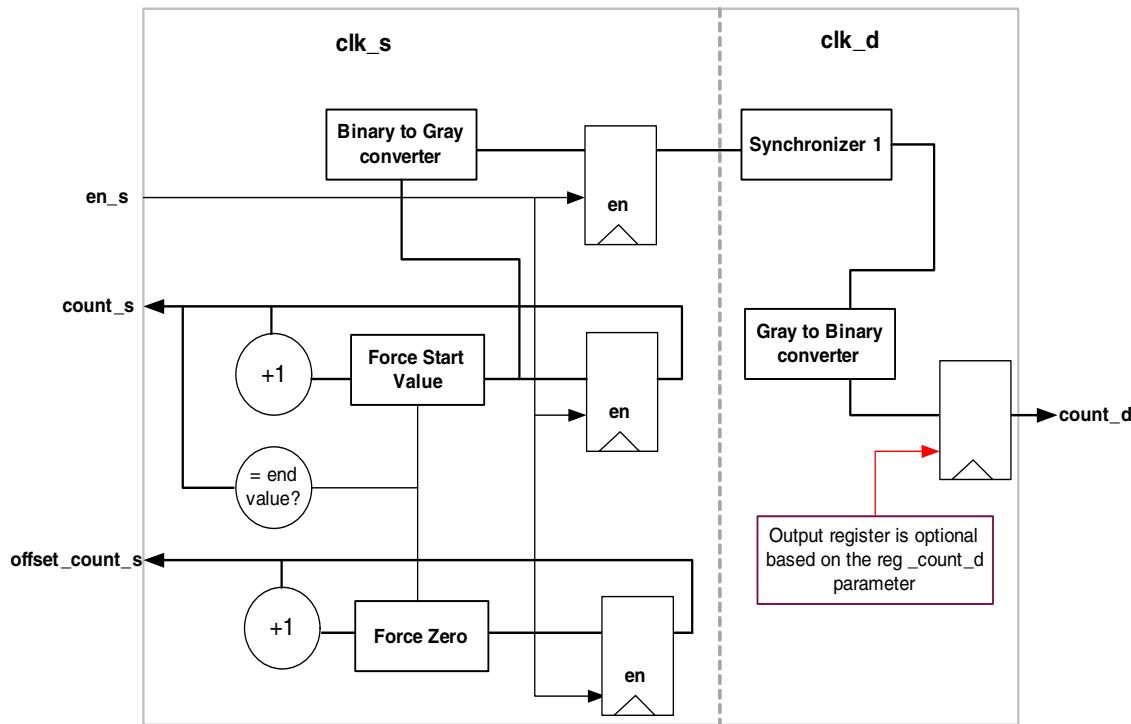
Figure J-8 Synchronizer 3 (with Output Registers)

Figure J-9 Synchronizer 3 (without Output Registers)

J.4 Synchronizer 4: Gray Coded Synchronizer (DWC_gmac_bcm24.v)

This synchronizer includes a binary counter in the source domain whose value is recoded as a binary-reflected-Gray code. The recoded count value is synchronized to the destination domain and then decoded from binary-reflected-Gray back to binary. The binary count value and an offset version (which is only useful for non-integer power-of-two sequences) are both available in the clk_s domain as `count_s` and `offset_count_s`. This synchronizer is used within other synchronizers to pass incrementing pointers across domains.

Figure J-10 Synchronizer 4 Block Diagram

K

Endian Support

This appendix provides information about the endian modes that GMAC-UNIV supports for 32-bit, 64-bit, and 128-bit data bus. It contains the following sections:

- ❖ “[Introduction](#)” on page [563](#)
- ❖ “[32-bit Data Bus Transfers](#)” on page [564](#)
- ❖ “[64-bit Data Bus Transfers](#)” on page [566](#)
- ❖ “[128-bit Data Bus Transfers](#)” on page [570](#)

K.1 Introduction

The GMAC-UNIV supports the following endian modes:

- ❖ [Little-Endian Mode](#)
- ❖ [Big-Endian Mode](#)
- ❖ [Byte-Invariant Big-Endian Mode](#)

K.1.1 Little-Endian Mode

In little-endian (LE) mode, the lowest storage address accesses the least significant byte (LSB) lane of the data bus and the highest address accesses the most significant byte (MSB) lane. This mode is supported in all GMAC configurations.

K.1.2 Big-Endian Mode

In conventional big-endian (CBE) mode, the lowest storage address accesses the most significant byte lane of the data bus and the highest address accesses the least significant byte. This mode is supported in all GMAC configurations except GMAC-AXI which supports byte-Invariant big-endian mode. However, when you select the AHB slave interface in the GMAC-AXI configuration, then the CBE mode is applicable to the AHB slave interface.

K.1.3 Byte-Invariant Big-Endian Mode

In a byte-invariant big-endian (BIBE) or address invariant mode, the address of each byte of memory remains unchanged when switching between the little-endian and big-endian mode. When a data item larger than a byte is loaded from or stored to memory, the bytes making up that data item are arranged in the correct order depending on the endianess of the memory access. This mode is supported only in the GMAC-AXI configuration on the AXI bus interface.



Note Although the GMAC-AXI master port supports the byte-invariant big-endian mode, it works in the similar way as the conventional big-endian mode. This is because the GMAC-AXI master always uses transfers size equal to AXI bus datawidth. Such transfers are identical in both conventional and byte-invariant big-endian modes.

K.2 32-bit Data Bus Transfers

This section provides information about the little-endian, conventional big-endian, and byte-invariant big-endian mode accesses, on 32-bit data bus, for Byte, HalfWord, and Word transfers. It contains the following sections:

- ❖ [Master Interface](#)
 - ◆ [Little-Endian Data Transfer](#)
 - ◆ [Conventional Big-Endian/ Byte-Invariant Big-Endian Data Transfer](#)
- ❖ [Slave Interface](#)
 - ◆ [Little-Endian Data Transfer](#)
 - ◆ [Conventional Big-Endian Data Transfer](#)
 - ◆ [Byte-Invariant Big-Endian Data Transfer](#)

K.2.1 Master Interface

K.2.1.1 Little-Endian Data Transfer

[Table K-1](#) provides information about the data transfers in little-endian mode on the GMAC master interface.

Table K-1 32-Bit Data Bus: Little-Endian Support for Master Interface

Transfer Size	Address Offset A[1:0]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
Always Word Transfer	0	MSB	MSB-1	LSB+1	LSB

K.2.1.2 Conventional Big-Endian/ Byte-Invariant Big-Endian Data Transfer

[Table K-2](#) provides information about the data transfers in conventional big-endian and byte-invariant big-endian mode on the GMAC master interface.

Table K-2 32-Bit Data Bus: Big-Endian/ Byte-Invariant Support for Master Interface

Transfer Size	Address Offset A[1:0]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
Always Word Transfer	0	LSB	LSB+1	MSB-1	MSB

K.2.2 Slave Interface

The GMAC Control and Status Registers are defined as 32-bit data structure. However, this data structure can break because of the byte-lane swapping during endian conversion. To retain the 32-bit data structure in all modes, the register byte-address is defined as shown in [Table K-3](#).

For example, in little-endian mode, the bits[7:0] correspond to address A0 and in CBE or BIBE mode, the bits[31:24] correspond to address A0.

Table K-3 32-Bit Data Bus: Reference CSR Register for Slave Interface

CSR Bits	31:24	23:16	15:18	7:0
Data Notation of CSR Register	D3	D2	D1	D0
Byte Address in LE	A3	A2	A1	A0
Byte Address in CBE/BIBE	A0	A1	A2	A3

K.2.2.1 Little-Endian Data Transfer

[Table K-4](#) provides information about the data transfers in little-endian mode on the GMAC slave interface. This is the default mode in which the lane data corresponds to the same bits in the Control and Status Registers.

Table K-4 32-Bit Data Bus: Little-Endian Support for Slave Interface

Transfer Size	Address Offset A[1:0]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
Word	0	D3	D2	D1	D0
HalfWord	0			D1	D0
HalfWord	2	D3	D2		
Byte	0				D0
Byte	1			D1	
Byte	2		D2		
Byte	3	D3			

K.2.2.2 Conventional Big-Endian Data Transfer

[Table K-5](#) provides information about the data transfers in conventional big-endian mode on the GMAC slave interface. In this mode, the lane data corresponds to the same bits of the Control and Status Registers (CSR) and therefore, there is no lane swapping.

Table K-5 32-Bit Data Bus: Conventional Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[1:0]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
Word	0	D3	D2	D1	D0
HalfWord	0	D3	D2		

Table K-5 32-Bit Data Bus: Conventional Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[1:0]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
HalfWord	2			D1	D0
Byte	0	D3			
Byte	1		D2		
Byte	2			D1	
Byte	3				D0

K.2.2.3 Byte-Invariant Big-Endian Data Transfer

Table K-6 provides information about the data transfers in byte-invariant big-endian mode on the GMAC slave interface. In this mode, a Word access (32-bit) is identical to the little-endian and conventional big-endian mode but transfers of smaller size (16-bit and 8-bit) are different. For example, a byte-access to A0 results in bits[7:0] of the AHB/AXI bus written to bits[31:24] (denoted as D3 in Table K-6) of the corresponding Control and Status register.

Table K-6 32-Bit Data Bus: Conventional Byte-Invariant Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[1:0]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
Word	0	D3	D2	D1	D0
HalfWord	0			D3	D2
HalfWord	2	D1	D0		
Byte	0				D3
Byte	1			D2	
Byte	2		D1		
Byte	3	D0			

K.3 64-bit Data Bus Transfers

This section provides information about the little-endian, conventional big-endian, and byte-invariant big-endian mode accesses, on 64-bit data bus, for Byte, HalfWord, Word, and DWord transfers. It contains the following sections:

- ❖ Master Interface
 - ◆ Little-Endian Data Transfer
 - ◆ Conventional Big-Endian/ Byte-Invariant Big-Endian Data Transfer
- ❖ Slave Interface
 - ◆ Little-Endian Data Transfer
 - ◆ Conventional Big-Endian Data Transfer
 - ◆ Byte-Invariant Big-Endian Data Transfer

K.3.1 Master Interface

The AXI master interface initiates only DWord transfers. The other master ports (AHB or native) also initiate DWord transfers except for Descriptor Status writes for which only Word transfers are initiated. Therefore, this section does not describe the transfer types such as HalfWord, Byte, and Word at address offset 4.

K.3.1.1 Little-Endian Data Transfer

[Table K-7](#) provides information about the data transfers in little-endian mode on the GMAC master interface.

Table K-7 64-Bit Data Bus: Little-Endian Support for Master Interface

Transfer Size	Address Offset A[2:0]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
DWord	0	MSB	MSB-1	MSB-2	MSB-3	LSB+3	LSB+2	LSB+1	LSB
Word	0					LSB+3	LSB+2	LSB+1	LSB

K.3.1.2 Conventional Big-Endian/ Byte-Invariant Big-Endian Data Transfer

[Table K-8](#) provides information about the data transfer in conventional big-endian and byte-invariant big-endian mode on the GMAC master interface.

Table K-8 64-Bit Data Bus: Big-Endian/ Byte-Invariant Big-Endian Support for Master Interface

Transfer Size	Address Offset A[2:0]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
DWord	0	LSB	LSB-1	LSB+2	LSB+3	MSB-3	MSB-2	MSB-1	MSB
Word	0	LSB	LSB-1	LSB+2	LSB+3				

K.3.2 Slave Interface

The GMAC registers and internal bus are 32-bit wide. Therefore, the DWord (64-bit) CSR access internally get converted into two 32-bit CSR accesses as shown in [Table K-9](#). The DWord access through Slave interface is supported only with the AXI Slave port. The DWord accesses are not supported on the AHB Slave port.

Table K-9 64-Bit Data Bus: Reference CSR Register for Slave Interface

CSR Bits	31:24	23:16	15:8	7:0
First CSR Register	D3	D2	D1	D0
Second CSR Register	D7	D6	D5	D4
Address in LE	A3	A2	A1	A0
	A7	A6	A5	A4
Address in CBE/BIBE	A0	A1	A2	A3
	A4	A5	A6	A7

K.3.2.1 Little-Endian Data Transfer

Table K-10 provides information about data transfers in little-endian mode on the GMAC slave interface. In Table K-10, the DWord access is applicable only for the AXI Slave port.

Table K-10 64-Bit Data Bus: Little-Endian Support for Slave Interface

Transfer Size	Address Offset A[2:0]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
DWord	0	D7	D6	D5	D4	D3	D2	D1	D0
Word	0					D3	D2	D1	D0
Word	4	D7	D6	D5	D4				
HalfWord	0							D1	D0
HalfWord	2					D3	D2		
HalfWord	4			D5	D4				
HalfWord	6	D7	D6						
Byte	0								D0
Byte	1							D1	
Byte	2						D2		
Byte	3					D3			
Byte	4				D4				
Byte	5			D5					
Byte	6		D6						
Byte	7	D7							

K.3.2.2 Conventional Big-Endian Data Transfer

Table K-11 provides information about data transfers in conventional big-endian mode on the GMAC slave interface. The DWord access is not supported on the AHB slave port and therefore, it is not described in Table K-11.

Table K-11 64-Bit Data Bus: Conventional Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[2:0]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
Word	0	D3	D2	D1	D0				
Word	4					D7	D6	D5	D4
HalfWord	0	D3	D2						
HalfWord	2			D1	D0				
HalfWord	4					D7	D6		

Table K-11 64-Bit Data Bus: Conventional Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[2:0]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
HalfWord	6							D5	D4
Byte	0	D3							
Byte	1		D2						
Byte	2			D1					
Byte	3				D0				
Byte	4					D7			
Byte	5						D6		
Byte	6							D5	
Byte	7								D4

K.3.2.3 Byte-Invariant Big-Endian Data Transfer

Table K-12 provides information about the data transfers in byte-invariant big-endian mode on the AXI slave interface. For DWord access at offset 0, the bits[63:32] of the AXI bus get transferred to bits[31:0] of the Register 0 and bits[31:0] get transferred to bits[31:0] of Register 1. For Word access at offset 0, bits[31:0] of the AXI bus get transferred to Register 0.

For HalfWord access to Address A0, bits[15:0] of the AXI bus get transferred to bits[31:16] of the Control and Status Register.

Table K-12 64-Bit Data Bus: Big-Endian/Byte-Invariant Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[2:0]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
DWord	0	D3	D2	D1	D0	D7	D6	D5	D4
Word	0					D3	D2	D1	D0
Word	4	D7	D6	D5	D4				
HalfWord	0							D3	D2
HalfWord	2					D1	D0		
HalfWord	4			D7	D6				
HalfWord	6	D5	D4						
Byte	0								D3
Byte	1							D2	
Byte	2						D1		
Byte	3					D0			

Table K-12 64-Bit Data Bus: Big-Endian/Byte-Invariant Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[2:0]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
Byte	4				D7				
Byte	5			D6					
Byte	6		D5						
Byte	7	D4							

K.4 128-bit Data Bus Transfers

This section provides information about the little-endian, conventional big-endian, and byte-invariant big-endian mode accesses, on 128-bit data bus, for byte, HalfWord, Word, DWord, and QWord transfers. It contains the following sections:

- ❖ Master Interface
 - ◆ Little-Endian Data Transfer
 - ◆ Conventional Big-Endian/Byte-Invariant Big-Endian Data Transfer
- ❖ Slave Interface
 - ◆ Little-Endian Data Transfer
 - ◆ Conventional Big-Endian Data Transfer
 - ◆ Byte-Invariant Big-Endian Data Transfer

K.4.1 Master Interface

The AXI master interface initiates only QWord transfers. The other master ports (AHB or native) also initiate QWord transfers except for Descriptor Status writes (32-bit at address 0) and Descriptor Timestamp writes (64-bit). For these writes, only Word and DWord transfers are initiated respectively. Therefore, this section does not describe the transfer types such as HalfWord and Byte.

K.4.1.1 Little-Endian Data Transfer

Table K-13 provides information about the data transfers in little-endian mode on the GMAC master interface.

Table K-13 128-Bit Data Bus: Little-Endian Support for Master Interface

Transfer Size	Address Offset A[3:0]	Data [127:120]	Data [119:112]	Data [111:104]	Data [103:96]	Data [95:88]	Data [87:80]	Data [79:72]	Data [71:64]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]	
QWord	0	MSB	MSB -1	MSB -2					MSB -7	LSB +7						LSB +1	LSB	
DWord	8	MSB	MSB -1	MSB -2	MSB -3	MSB -4	MSB -5	MSB -6	MSB -7									
Word	0														LSB +3	LSB +2	LSB +1	LSB

K.4.1.2 Conventional Big-Endian/Byte-Invariant Big-Endian Data Transfer

Table K-14 provides information about the data transfers in conventional big-endian/byte-invariant big-endian mode on the GMAC master interface. In byte-invariant big-endian mode for AXI, only QWord transfers are applicable.

Table K-14 128-Bit Data Bus: Conventional Big-Endian/Byte-Invariant Big-Endian Support for Master Interface

Transfer Size	Address Offset A[3:0]	Data [127:120]	Data [119:112]	Data [111:104]	Data [103:96]	Data [95:88]	Data [87:80]	Data [79:72]	Data [71:64]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
QWord	0	LSB	LSB +1	LSB +2					LSB +7	MSB -7						MSB -1	MSB
DWord	8									MSB -7						MSB -1	MSB
Word	0	LSB	LSB +1	LSB +2	LSB +3												

K.4.2 Slave Interface

The Control and Status Register are 32-bit wide. Therefore, the DWord/QWord CSR access internally gets converted into two/four 32-bit CSR accesses as shown in Table K-15. The DWord/QWord access through Slave interface is supported only with the GMAC-AXI Slave interface.

Table K-15 128-Bit Data Bus: Reference CSR Register for Slave Interface

CSR Bits	31:24	23:16	15:8	7:0
First CSR Register	D3	D2	D1	D0
Second CSR Register	D7	D6	D5	D4
Third CSR Register	D11	D10	D9	D8
Fourth CSR Register	D15	D14	D13	D12
Address in LE	A3	A2	A1	A0
	A7	A6	A5	A4
	AB	AA	A9	A8
	AF	AE	AD	AC
	A0	A1	A2	A3
Address in CBE/BIBE	A4	A5	A6	A7
	A8	A9	AA	AB
	AC	AD	AE	AF

K.4.2.1 Little-Endian Data Transfer

Table K-16 provides information about the data transfers in little-endian mode on the GMAC slave interface. In Table K-16, QWord and DWord accesses are applicable only for AXI Slave port.

Table K-16 128-Bit Data Bus: Little-Endian Support for Slave Interface

Transfer Size	Address Offset A[3:0]	Data [127:120]	Data [119:112]	Data [111:104]	Data [103:96]	Data [95:88]	Data [87:80]	Data [79:72]	Data [71:64]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
QWord	0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
DWord	0									D7	D6	D5	D4	D3	D2	D1	D0
DWord	8	D15	D14	D13	D12	D11	D10	D9	D8								
Word	0													D3	D2	D1	D0
Word	4									D7	D6	D5	D4				
Word	8					D11	D10	D9	D8								
Word	0xC	D15	D14	D13	D12												
HalfWord	0															D1	D0
HalfWord	2													D3	D2		
HalfWord	4											D5	D4				
HalfWord	6								D7	D6							
HalfWord	8							D9	D8								
HalfWord	0xA					D11	D10										
HalfWord	0xC			D13	D12												
HalfWord	0xE	D15	D14														
Byte	0																D0
Byte	1																D1
Byte	x																
Byte	7									D7							
Byte	8								D8								
Byte	x											
Byte	0xE		D14														
Byte	0xF	D15															

In Table K-16, the ellipses (...) represent Dx where x is the byte offset.

K.4.2.2 Conventional Big-Endian Data Transfer

[Table K-17](#) provides information about the data transfers in conventional big-endian mode on the GMAC slave interface. The QWord and DWord access is not supported on the AHB slave port and therefore, these are not described in [Table K-17](#).

Table K-17 128-Bit Data Bus: Conventional Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[3:0]	Data [127:120]	Data [119:112]	Data [111:104]	Data [103:96]	Data [95:88]	Data [87:80]	Data [79:72]	Data [71:64]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
Word	0	D3	D2	D1	D0												
Word	4					D7	D6	D5	D4								
Word	8									D11	D10	D9	D8				
Word	0xC													D15	D14	D13	D12
HalfWord	0	D3	D2														
HalfWord	2			D1	D0												
HalfWord	4					D7	D6										
HalfWord	6							D5	D4								
HalfWord	8									D11	D10						
HalfWord	0xA											D9	D8				
HalfWord	0xC													D9	D8		
HalfWord	0xE															D13	D12
Byte	0	D3															
Byte	1		D2														
Byte	x											
Byte	7								D4								
Byte	8									D11							
Byte	x										
Byte	0xE															D13	
Byte	0xF																D12

In [Table K-17](#), the ellipses (...) represent Dx where x is the byte offset.

K.4.2.3 Byte-Invariant Big-Endian Data Transfer

[Table K-18](#) provides information about the data transfers in byte-invariant big-endian mode on the GMAC slave interface. For a QWord access, the bits[127:96] get transferred to bits[31:0] of Register 0 (at address offset 0) in the first internal cycle, and bits[31:0] of the AXI bus get transferred to bits[31:0] of Register 3 (at address offset 0xC) in the fourth internal cycle.

Table K-18 128-Bit Data Bus: Byte-Invariant Big-Endian Support for Slave Interface

Transfer Size	Address Offset A[3:0]	Data [127:120]	Data [119:112]	Data [111:104]	Data [103:96]	Data [95:88]	Data [87:80]	Data [79:72]	Data [71:64]	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
QWord	0	D3	D2	D1	D0	D7	D6	D5	D4	D11	D10	D9	D8	D15	D14	D13	D12
DWord	0									D3	D2	D1	D0	D7	D6	D5	D4
DWord	8	D11	D10	D9	D8	D15	D14	D13	D12								
Word	0													D3	D2	D1	D0
Word	4									D7	D6	D5	D4				
Word	8					D11	D10	D9	D8								
Word	0xC	D15	D14	D13	D12												
HalfWord	0															D3	D2
HalfWord	2													D1	D0		
HalfWord	4												D7	D6			
HalfWord	6									D5	D4						
HalfWord	8							D11	D10								
HalfWord	0xA					D9	D8										
HalfWord	0xC			D15	D14												
HalfWord	0xE	D13	D12														
Byte	0																D3
Byte	1																D2
Byte	x											
Byte	7									D4							
Byte	8								D11								
Byte	x											
Byte	0xE		D13														
Byte	0xF	D12															

In Table K-18, the ellipses (...) represent Dx where x is the byte offset.