CEVA-XM4™

RTL V1.1.1.F
Integration Reference Guide

Rev. 1.1.1.F

January 2016

DSP Division

CONFIDENTIAL • UNAUTHORIZED REPRODUCTION PROHIBITED

Documentation Control

History Table

Version	Date	Description	Remarks
V1.0.0.A	26/2/2015	Initial release	
V1.0.0.F	12/4/2015	Updated IO table and Reset sequence chapter	
V1.1.0.F	6/8/2015	Updated for major release	
V1.1.1.F	19/1/2016	Update version and IO table	

Disclaimer and Proprietary Information Notice

The information contained in this document is subject to change without notice and does not represent a commitment on any part of CEVA®, Inc. CEVA®, Inc. and its subsidiaries make no warranty of any kind with regard to this material, including, but not limited to implied warranties of merchantability and fitness for a particular purpose whether arising out of law, custom, conduct or otherwise.

While the information contained herein is assumed to be accurate, CEVA®, Inc. assumes no responsibility for any errors or omissions contained herein, and assumes no liability for special, direct, indirect or consequential damage, losses, costs, charges, claims, demands, fees or expenses, of any nature or kind, which are incurred in connection with the furnishing, performance or use of this material.

This document contains proprietary information, which is protected by U.S. and international copyright laws. All rights reserved. No part of this document may be reproduced, photocopied, or translated into another language without the prior written consent of CEVA®, Inc.

CEVA®, CEVA-XC™, CEVA-XC5™, CEVA-XC8™, CEVA-XC321™, CEVA-XC323™, CEVA-Xtend™, CEVA-XC4000™, CEVA-XC4100™, CEVA-XC4200™, CEVA-XC4210™, CEVA-XC4400™, CEVA-XC4410™, CEVA-XC4500™, CEVA-XC4600™, CEVA-TeakLite™, CEVA-TeakLite-II™, CEVA-TeakLite-III™, CEVA-TL3210™, CEVA-TL3211™, CEVA-TeakLite-4™, CEVA-TL410™, CEVA-TL411™, CEVA-TL420™, CEVA-TL421™, CEVA-Quark™, CEVA-Teak™, CEVA-X™, CEVA-X1620™, CEVA-X1622™, CEVA-X1641™, CEVA-X1643™, Xpert-TeakLite-II™, Xpert-Teak™, CEVA-XS1100A™, CEVA-XS1200™, CEVA-XS1200A™, CEVA-TLS100™, Mobile-Media™, CEVA-MM1000™, CEVA-MM2000™, CEVA-SP™, CEVA-VP™, CEVA-MM3000™, CEVA-MM3100™, CEVA-MM3101™, CEVA-XM™, CEVA-XM4™, CEVA-X2™, CEVA-Audio™, CEVA-HD-Audio™, CEVA-VoP™, CEVA-Bluetooth™, CEVA-SATA™, CEVA-SAS™, CEVA-Toolbox™, SmartNcode™ are trademarks of CEVA, Inc.

All other product names are trademarks or registered trademarks of their respective owners.

Support

CEVA® makes great efforts to provide a user-friendly software and hardware development environment. Along with this, CEVA provides comprehensive documentation, enabling users to learn and develop applications on their own. Due to the complexities involved in the development of DSP applications that might be beyond the scope of the documentation, an online Technical Support Service has been established. This service includes useful tips and provides fast and efficient help, assisting users to quickly resolve development problems.

How to Get Technical Support:

- **FAQs**: Visit our website http://www.ceva-dsp.com or your company’s protected page on the CEVA website for the latest answers to frequently asked questions.
- **Application Notes**: Visit our website http://www.ceva-dsp.com or your company’s protected page on the CEVA website for the latest application notes.
- **Email**: Use the CEVA central support email address ceva-support@ceva-dsp.com. Your email will be forwarded automatically to the relevant support engineers and tools developers who will provide you with the most professional support to help you resolve any problem.
- **License Keys**: Refer any license key requests or problems to sdtkeys@ceva-dsp.com. For SDT license keys installation information, see the *SDT Installation and Licensing Scheme Guide*.

Email: ceva-support@ceva-dsp.com

Visit us at: www.ceva-dsp.com

List of Sales and Support Centers

Israel	USA	Ireland	Sweden
2 Maskit Street P.O. Box 2068 Herzelia 46120 Israel **Tel**: +972 9 961 3700 **Fax**: +972 9 961 3800	1174 Castro Street Suite 210 Mountain View, CA 94040 USA **Tel**: +1-650-417-7923 **Fax**: +1-650-417-7924	Segrave House 19/20 Earlsfort Terrace 3rd Floor Dublin 2 Ireland **Tel**: +353 1 237 3900 **Fax**: +353 1 237 3923	Klarabergsviadukten 70 Box 70396 107 24 Stockholm Sweden **Tel**: +46(0)8 506 362 24 **Fax**: +46(0)8 506 362 20
China (Shanghai)	**China (Beijing)**	**China (Shenzhen)**	**Hong Kong**
Unit 1203, Building E Chamtime Plaza Office Lane 2889, Jinke Road Pudong New District Shanghai, 201203 China **Tel**: +86-21-20577000 **Fax**: +86-21-20577111	Rm 503, Tower C Raycom InfoTech Park No.2, Kexueyuan South Road Haidian District Beijing 100190 China **Tel**: +86-10 5982 2285 **Fax**: +86-10 5982 2284	2702-09 Block C Tiley Central Plaza II Wenxin 4th Road Nanshan District Shenzhen 518054 China **Tel**: +86-755-86595012	Level 43, AIA Tower 183 Electric Road North Point Hong Kong **Tel**: +852-39751264
South Korea	**Taiwan**	**Japan**	**France**
#478, Hyundai Arion 147, Gumgok-Dong Bundang-Gu Sungnam-Si Kyunggi-Do, 463-853 South Korea **Tel**: +82-31-704-4471 **Fax**:+82-31-704-4479	Room 621 No.1, Industry E, 2nd Rd Hsinchu, Science Park Hsinchu 300 Taiwan R.O.C **Tel**: +886 3 5798750 **Fax**: +886 3 5798750	3014 Shinoharacho Kasho Bldg. 4/F Kohoku-ku Yokohama Kanagawa 222-0026 Japan **Tel**: +81 045-430-3901 **Fax**: +81 045-430-3904	RivieraWaves S.A.S 400, avenue Roumanille Les Bureaux Green Side 5, Bât 6 06410 Biot - Sophia Antipolis France **Tel**: +33 4 83 76 06 00 **Fax**: +33 4 83 76 06 01

Table of Contents

1. INTRODUCTION 1

1.1 Scope 1

1.2 Audience 1

1.3 Related Documents 1

1.4 CEVA-XM4 IP Description 2

2. I/O DESCRIPTION 5

3. INTERNAL MEMORY 35

3.1 Internal Program Memory 35
3.1.1 INTERNAL PROGRAM MEMORY BLOCK0 35
3.1.2 INTERNAL PROGRAM CACHE MEMORY 36
3.1.3 INTERNAL PROGRAM MEMORY PARITY BLOCKS 39

3.2 Internal Data Memory 39

3.3 Memory Interface Support 41

3.4 Internal Program Memory Connectivity 42

3.5 Internal Data Memory Connectivity 48

3.6 PSU Memory Interface 48

3.7 BIST Support 49

4. INTERRUPT HANDSHAKE 51

4.1 VINT 52

5. MULTI-CORE SUPPORT 53

5.1 AXI Monitor Mechanism 53

5.2 Multicore Command Interface 56
5.2.1 MCCI SINGLE DWORD 56
5.2.2 MCCI SLVERR 57

5.3 Multi-Core Status Register Space 58

5.4 Internal TCM Snooping 60
5.4.1 SNOOP READ TRANSACTION 60
5.4.2 SNOOP WRITE TRANSACTION 61

6. DMA MANAGER INTERFACE 63

6.1 QMAN Interrupts 63

6.2 Queue Descriptor Enable Signal 63

6.3 Queue Semaphore 64

7. TRAP INDICATION 67

8. VERIFICATION INDICATION 69

9. OPERATION MODES 71

10. BOOT SEQUENCE 75

10.1 Boot Using the Boot Input 75
10.2 Boot Using an External Master (via EDAP) 76
10.3 Boot Using PDMA (Configured by the EDAP) 77
10.4 Boot Using the Program Cache Memory (via SWOP) 78

11. RESET SEQUENCE 81
11.1 Core Reset 81
11.2 System Reset 83
11.3 OCEM Reset 84
11.4 Global Reset 84

12. POWER SCALING UNIT 87
12.1 Light Sleep Mode 88
12.2 Standby Mode 89
12.3 Retention Mode 89
12.4 Shutdown Mode 90
12.5 AXI Low-Power Interface 91

13. GENERAL PURPOSE INPUT/OUTPUT 93

14. UNDEFINED OPCODES 95

15. CEVA-XTEND HARDWARE 97

16. DESIGNWARE 99

17. EXTERNAL ACCESS 101
17.1 Automatic Power Saving Mode 102
17.2 External Program Port 103
17.2.1 EPP READ FROM A NON-CACHEABLE REGION 104
17.2.2 EPP READ FROM NON-CACHEABLE REGION (ECC ON AXI) 106
17.2.3 EPP READ WITH SLOWER AXI 107
17.2.4 EPP READ FROM CACHEABLE REGION 108
17.2.5 PDMA DOWNLOAD (EPP READ) 110
17.2.6 EPP OCEM WRITE 113
17.3 External Master Ports (EDP/AXIM0/ AXIM1) 114
17.3.1 EDP DSP READ 115
17.3.2 EDP DSP READ (ECC ON AXI) 117
17.3.3 EDP DSP WRITE BUFFER WRITE ACCESS 118
17.3.4 EDP DSP WRITE WITH SLOWER AXI AND READY 120
17.3.5 EDP WIDE NON-ALIGNED READ 121
17.3.6 EDP WRITE WITH SPARSE WRITE STROBES 122
17.3.7 DDMA QUEUE NON-AUTOMATIC MODE 124
17.3.8 READ-AFTER-WRITE CONSIDERATIONS 125
17.3.9 DATA DMA DEBUG MATCH 125
17.4 AXI Slave Port (EDAP/AXIsX) 127
17.4.1 EDAP READ SINGLE FOLLOWED BY READ BURST 128
17.4.2 EDAP WRITE SINGLE AND BURST 130
17.4.3 EDAP DOWNLOAD TO TCM AFTER RESET WITH EXTERNAL WAIT 131

17.5 I/O Port 132
17.5.1 I/O PORT READ 132
17.5.2 I/O PORT WRITE 134

18. AXI CAPABILITIES 137

18.1 EPP AXI Capabilities 137
18.1.1 OUTSTANDING TRANSACTIONS 138
18.1.2 INTERLEAVING/OUT-OF-ORDER SUPPORT 139
18.1.3 SUPPORTED AXI SIGNAL VALUES 139

18.2 EDP AXI Capabilities 140
18.2.1 4KB BOUNDARY 140
18.2.2 AXI IDS 141
18.2.3 OUTSTANDING TRANSACTIONS 141
18.2.4 INTERLEAVING/OUT-OF-ORDER SUPPORT 143
18.2.5 SUPPORTED AXI SIGNAL VALUES 143

18.3 AXI Master Port Capabilities 144
18.3.1 4KB BOUNDARY 144
18.3.2 AXI IDS 145
18.3.3 INTERLEAVING/OUT-OF-ORDER SUPPORT 145
18.3.4 SUPPORTED AXI SIGNAL VALUES 146

18.4 AXI Slave Port Capabilities 147
18.4.1 OUTSTANDING TRANSACTIONS 147
18.4.2 INTERLEAVING/OUT-OF-ORDER SUPPORT 148
18.4.3 SUPPORTED AXI SIGNAL VALUES 148

19. EXTERNAL WAIT 151

20. DFT SUPPORT 153

21. OCEM INTERFACE 155

21.1 External Breakpoint Request 155

21.2 JTAG Interface 156
21.2.1 CLOCK 156

21.3 Clock Synchronization 157

21.4 Boundary Scan Data Out 157

21.5 OCEM TDO 158

21.6 TDO Output Enable 159

21.7 OCEM APB Slave Port 159
21.7.1 OCEM APB SLAVE PORT READ TRANSFER 160
21.7.2 OCEM APB SLAVE PORT WRITE TRANSFER 160

21.8 External Program Memory Indication 161

21.9 General Purpose Outputs 162

21.10 OCEM Core Reset 163

22. REAL-TIME TRACE INTEGRATION 164

22.1 Integration with ETM-R4 164

22.2 RTT Reset 170

22.3 Integrating CEVA-XM4 Trace Triggers 171

23. MULTI-CYCLE PATHS 175

24. INPUT CLOCK SYNCHRONIZATION 177

25. INPUT SINGLE SAMPLING 179

26. CEVA-XM4 INSTANTIATION 181

27. VALIDATION FOR SIGNOFF 183

28. GLOSSARY 185

List of Figures

Figure 1-1: CEVA-XM4 Hierarchy 2
Figure 3-1: Internal Program Memory block0 Default Partition 35
Figure 3-2: Internal Program Cache Set Memory Partition (One-Block Configuration) 37
Figure 3-3: Internal Program Cache TAG Memory Partition 38
Figure 3-4: Internal Data Memory Partition – Four-Block Configuration (without Memory ECC) 40
Figure 3-5: Internal Program block0 Memory (Default Connectivity) 42
Figure 3-6: Internal Program TAG Memory (Default Connectivity) 43
Figure 3-7: Internal Program Set Memory (Default Connectivity) 44
Figure 3-8: Internal Program Memory block0 Parity (ECC Mode) 45
Figure 3-9: Internal Program Memory TAG Parity (ECC Mode) 46
Figure 3-10: Internal Program Memory Set Parity (ECC Mode) 47
Figure 3-11: Internal Data Memory Connectivity 48
Figure 4-1: Interrupt Handshake Timing Diagram 51
Figure 4-2: VINT Timing Diagram 52
Figure 5-1: Exclusive Read Succeeded (EXOK Response) Timing Diagram 54
Figure 5-2: Exclusive Read Failed (OK Response) Timing Diagram 54
Figure 5-3: Exclusive Write Succeeded (EXOK Response) Timing Diagram 55
Figure 5-4: Exclusive Write Failed (OK Response) Timing Diagram 55
Figure 5-5: EDAP MCCI Single dword Write Timing Diagram 56
Figure 5-6: EDAP Read with Unequal Clocks Timing Diagram 59
Figure 5-7: Snoop Detection during EDAP Read Transaction Timing Diagram 60
Figure 5-8: Snoop Detection during EDAP Write Transaction Timing Diagram 61
Figure 6-1: Queue 0 Semaphore Implementation Example 65
Figure 6-2: Queue Semaphore Handshake Timing Diagram 65
Figure 7-1: Trap Indication Routine Timing Diagram 67
Figure 8-1: Error Test Example 69
Figure 8-2: Ending Test 69
Figure 9-1: External Permission Violation Timing Diagram 71
Figure 9-2: Operation Mode Indication Timing Diagram 72
Figure 10-1: Boot Sequence Timing Diagram 75
Figure 10-2: Boot Using the PDMA Timing Diagram 77

Figure 10-3: Boot Using SWOP Mechanism Timing Diagram 78
Figure 11-1: Core Reset Timing Diagram 81
Figure 11-2: System Reset Timing Diagram 83
Figure 11-3: OCEM Reset Timing Diagram 84
Figure 11-4: Global Reset Timing Diagram 84
Figure 12-1: Entering Light Sleep Mode Timing Diagram 88
Figure 12-2: Exiting Light Sleep Mode Timing Diagram 88
Figure 12-3: Entering Standby Mode Timing Diagram 89
Figure 12-4: Entering/Exiting Retention Mode Timing Diagram 89
Figure 12-5: Entering/Exiting Shutdown Mode Timing Diagram 90
Figure 12-6: AXI Low-Power Interface Timing Diagram 91
Figure 13-1: GPIO Timing Diagram 93
Figure 14-1: uop Interrupt, Status, and Packet Address Timing Diagram 95
Figure 17-1: Example APS Usage 102
Figure 17-2: APB3 APS Usage Timing Diagram 103
Figure 17-3: EPP Read from Non-Cacheable Region Timing Diagram 104
Figure 17-4: EPP Read from Non-Cacheable Region with (ECC on AXI) Timing Diagram 106
Figure 17-5: EPP Read with Slower AXI Timing Diagram 107
Figure 17-6: EPP Read from Cacheable Region Timing Diagram 108
Figure 17-7: PDMA Download (EPP Read) Timing Diagram 110
Figure 17-8: EPP OCEM Write Timing Diagram 113
Figure 17-9: EDP DSP Read Timing Diagram 115
Figure 17-10: EDP DSP Read Transaction (ECC on AXI) Timing Diagram 117
Figure 17-11: EDP DSP Write Buffer Write Access Timing Diagram 118
Figure 17-12: EDP DSP Write with Slower AXI and Ready Timing Diagram 120
Figure 17-13: EDP Wide Non-Aligned Read Timing Diagram 121
Figure 17-14: EDP Write with Sparse Write Strobes Timing Diagram 122
Figure 17-15: DDMA next_ddma Timing Diagram 124
Figure 17-16: DMA Debug Match Indication 126
Figure 17-17: DMA Debug Match Indication Timing Diagram 127
Figure 17-18: EDAP Read Single Followed by Read Burst Timing Diagram 128
Figure 17-19: EDAP Write Single and Burst Timing Diagram 130
Figure 17-20: EDAP Download to TCM after Reset with External Wait Timing Diagram 131
Figure 17-21: I/O Port Read Timing Diagram 132
Figure 17-22: I/O Port Write Timing Diagram 134
Figure 21-1: External Breakpoint Request Timing Diagram 155
Figure 21-2: TDO and RTCK Generation Timing Diagram 156
Figure 21-3: bs_reg_tdo Timing Diagram 158
Figure 21-4: cevaxm4_ocm_tdo_r Timing Diagram 158
Figure 21-5: cevaxm4_ocm_tdo_oen_r Timing Diagram 159
Figure 21-6: OCEM APB Slave Read CORE_CONFIG Register Timing Diagram 160
Figure 21-7: OCEM APB Slave Write to Scan Chains Timing Diagram 160
Figure 21-8: External Program Memory Indication Timing Diagram 161
Figure 21-9: cevaxm4_ocm_gp_out_r Timing Diagram 162

Figure 21-10: OCEM Core Reset Timing Diagram 163
Figure 22-1: RTT Integration (Internal ETM-R4) 165
Figure 22-2: RTT Integration (External ETM-R4) 168
Figure 22-3: Connecting CTI to CEVA-XM4 Trace Triggers (Example) 172
Figure 26-1: cevaxm4_sim_top.v Location 181

List of Tables

Table 2-1: CEVA-XM4 Interface 6
Table 3-1: Internal Program TCM block0 Hardware Configurations 36
Table 3-2: Internal Program Cache Set Memory Hardware Configurations 36
Table 3-3: Internal Program Cache TAG Memory Hardware Configurations 38
Table 3-4: Internal Program Parity Memory Sizes 39
Table 3-5: Internal Data TCM Hardware Configurations 39
Table 3-6: Internal Program TCM block0 Memory Controls in Default Mode 41
Table 3-7: Internal Program TAG Memory Controls 41
Table 3-8: Internal Program Set Memory Controls 41
Table 3-9: Internal Data TCM Memory Controls 41
Table 3-10: Internal Memory PSU Controls 49
Table 4-1: Interrupt Handshake Signal Description 51
Table 4-2: VINT Signal Description 52
Table 5-1: MCCI Single dword Signal Description 57
Table 5-2: CORE_VERSION and CORE_ID Address Mapping 58
Table 5-3: EDAP Read with Unequal Clocks Signal Description 59
Table 5-4: Snoop Read Transaction Signal Description 61
Table 5-5: Snoop Write Transaction Signal Description 62
Table 6-1: Queue Semaphore Handshake Signal Description 66
Table 9-1: External Permission Violation Signal Description 72
Table 9-2: Operation Mode Indication Signal Description 73
Table 10-1: Boot Sequence Signal Description 76
Table 11-1: Core Reset Signal Description 82
Table 11-2: System Reset Signal Description 83
Table 11-3: OCEM Reset Signal Description 84
Table 11-4: Global Reset Signal Description 85
Table 12-1: Power Mode Value Registers 87
Table 15-1: CEVA-Xtend Interface (SPU) 97
Table 15-2: CEVA-Xtend Interface (VPU) 98
Table 17-1: EPP Read from Non-Cacheable Region Signal Description 105
Table 17-2: EPP Read from Cacheable Region Signal Description 109
Table 17-3: PDMA Download (EPP Read) Signal Description 112
Table 17-4: EPP OCEM Write Signal Description 114
Table 17-5: EDP AXI Fixed Signals 114
Table 17-6: EDP DSP Read Signal Description 116
Table 17-7: EDP DSP Write Buffer Write Access Signal Description 119

Table 17-8: DDMA next_ddma Signal Description 124
Table 17-9: EDAP Read Single Followed by Read Burst Signal Description 129
Table 17-10: EDAP Write Single and Burst Signal Description 131
Table 17-11: External Device Access Port Read with Unequal Clocks 132
Table 17-12: I/O Port Read Signal Description 133
Table 17-13: I/O Port Write Signal Description 135
Table 18-1: EPP AXI IDs 137
Table 18-2: EPP AXI Master Outstanding Transactions 138
Table 18-3: EPP Write Interleaving Depth 139
Table 18-4: EPP Out-of-Order Support 139
Table 18-5: Supported EPP AXI Signal Values 139
Table 18-6: EDP AXI IDs 141
Table 18-7: EDP AXI Master Outstanding Transactions 141
Table 18-8: EDP Write Interleaving Depth 143
Table 18-9: EDP Out-of-Order Support 143
Table 18-10: Supported EDP AXI Signal Values 143
Table 18-11: AXI Master AXI IDs 145
Table 18-12: AXI Master Ports Write Interleaving Depth 145
Table 18-13: AXI Master Ports Out-of-Order Support 145
Table 18-14: Supported AXI Master Ports AXI Signal Values 146
Table 18-15: AXI Slave Port Outstanding Transactions 147
Table 18-16: AXI Slave Port Write Interleaving Depth Support 148
Table 18-17: AXI Slave Port Out-of-Order Support 148
Table 18-18: Supported EDAP AXI Slave Signal Values 148
Table 18-19: Supported AXIsX Slave Signal Values 149
Table 19-1: cevaxm4_psu_core_wait_r Memory Effects 152
Table 20-1: DFT Interface 153
Table 21-1: External Breakpoint Request Signal Description 155
Table 22-1: Internal ETM within CEVA-XM4 RTT Mapping 166
Table 22-2: External ETM within CEVA-XM4 RTT Mapping 168
Table 22-3: External ETM Connections 169
Table 22-4: cevaxm4 OCEM Triggers 171
Table 24-1: Input Clock Synchronization 177
Table 25-1: Input Single Sampling 179
Table 28-1: Acronyms 185

1. Introduction

1.1 Scope

This document describes common topics that make the integration of the CEVA-XM4™ DSP Core SIP (soft IP) easier.

Important: *The ETM/RTT module referred to in this document is an add-on feature that is licensed separately.*

1.2 Audience

This document is intended for ASIC engineers who are integrating the CEVA-XM4 into their design.

1.3 Related Documents

The following documents are related to the information in this document:

1. *CEVA-XM4 Architecture Specification*
2. *CEVA- XM4 Simulation Reference Guide*
3. *CEVA- XM4 Backend Flow Reference Guide*
4. *CEVA- XM4 Database Reference Guide*
5. *CEVA- XM4 Release Notes*
6. *CEVA- XM4 Real-Time Trace Architecture Specification*
7. *CEVA-XM4 Power Modes Reference Guide*

1.4 CEVA-XM4 IP Description

Figure 1-1 describes the CEVA-XM4 hierarchy.

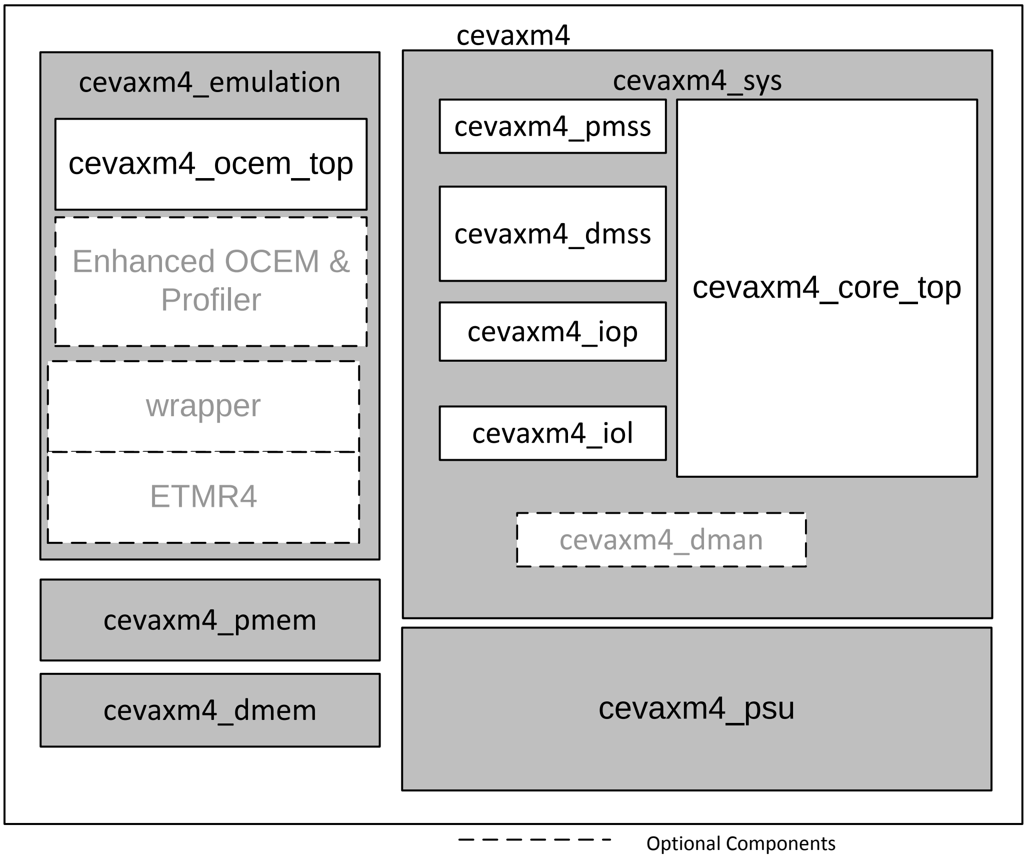

cevaxm4
cevaxm4_emulation
cevaxm4_ocem_top
Enhanced OCEM & Profiler
wrapper
ETMR4
cevaxm4_sys
cevaxm4_pmss
cevaxm4_dmss
cevaxm4_iop
cevaxm4_iol
cevaxm4_core_top
cevaxm4_dman
cevaxm4_pmem
cevaxm4_dmem
cevaxm4_psu
Optional Components

Figure 1-1: CEVA-XM4 Hierarchy

The CEVAXM4_SYS, CEVAXM4_EMULATION, and CEVAXM4_PSU (Power Scaling Unit) blocks and their contents are SIP-configured at the CEVA-XM4 upon installation. They do not require any adaptation by the user.

The memories of the CEVA-XM4 are contained in separate hierarchies: CEVAXM4_DMEM for data memories and CEVAXM4_PMEM for program memories. These memories should be replaced by the integrator’s memories according to the vendor’s technology. The interface to the memories should be adapted accordingly.

The internal program memory consists of:

- **block0**: TCM; can be configured as 32 KB, 64 KB, 128 KB, or 256 KB
- **Internal Program Cache**: Total cache size of 32 KB, 64 KB, or 128 KB arranged in four ways, plus the TAG memory

When ECC is included in the configuration, additional memory cuts are introduced into the PMEM. These memories extend the block0 TCM, Set memories, and TAG memories, and are used for the ECC redundancy (parity) bits.

The internal data memory supports the following configuration options:

- Four-block configuration
- Total data memory size configuration (128 KB, 256 KB, or 512 KB)

Note: *The Wrapper is SIP but the ETM-R4 is licensed separately from ARM. In addition, the ETM-R4 can be internal or external to the CEVA-XM4 top module (as shown in* Figure 1-1*).*

2. I/O Description

Table 2-1 describes the interfaces of the CEVA-XM4, and the signals are arranged according to their functionality. The table has the following columns:

- **SAR:** This column indicates the State after Reset of the outputs.
- **Default:** This column indicates the value that MUST be connected when the input is not used.
- **Configuration**: The column indicates the I/O signals that might or might not be present depending on the device configuration:
  - **CEVAXM4-ECC:** This I/O signal is present only when the CEVAXM4-ECC configuration is used.
  - **VEH:** This I/O signal is present only when the VPU Xtend configuration is used.
  - **SEH**: This I/O signal is present only when the SPU Xtend configuration is used.
  - **DMAN**: This I/O signal is present only when the CEVA-XM4 DMA Manager is included in the configuration.
  - **AXIM**: This I/O signal is present only if the AXI masters are included in the configuration.
  - **AXIsX**: This I/O signal is present only if the relevant AXI slaves are included in the configuration.
  - **BusECC**: This I/O signal is present only if the AMBA Bus ECC is included in the configuration.

Table 2-1: CEVA-XM4 Interface

Name	Size	I/O	Description	SAR	Default	Configuration
External Program Port (EPP) Interface (Master AXI)						
arready_epp	1	I	EPP Read Address Channel (RAC) Ready		1’b0	
cevaxm4_epp_arid_r	4	O	EPP RAC ID	4’b0		
cevaxm4_epp_araddr_r	32	O	EPP RAC Address	32’h0		
cevaxm4_epp_arlen_r	8	O	EPP RAC Length	8’h0		
cevaxm4_epp_arsize_r	3	O	EPP RAC Size	3’b100		
cevaxm4_epp_arburst_r	2	O	EPP RAC Burst Type	2’b01		
cevaxm4_epp_arlock_r	1	O	EPP RAC Lock	1’b0		
cevaxm4_epp_arcache_r	4	O	EPP RAC Cache	4’h0		
cevaxm4_epp_arprot_r	3	O	EPP RAC Protection Bits	3’b100		
cevaxm4_epp_arqos_r	4	O	EPP RAC QoS Type	4’b0		
cevaxm4_epp_rready_r	1	O	EPP Read Data Channel (RDC) Ready	1’b1		
cevaxm4_epp_arvalid_r	1	O	EPP RAC Address Valid	1’b0		
rid_epp	4	I	EPP RDC ID		4’b0	
rdata_epp	128	I	EPP RDC Read Data		128’h0	
rresp_epp	2	I	EPP RDC Response		2’b0	
rlast_epp	1	I	EPP RDC Last Data		1’b0	
rvalid_epp	1	I	EPP RDC Data Valid		1’b0	
awready_epp	1	I	EPP Write Address Channel (WAC) Ready		1’b0	
cevaxm4_epp_awid_r	4	O	EPP WAC ID	4’b0		
cevaxm4_epp_awaddr_r	32	O	EPP WAC Address	32’h0		

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_epp_awlen_r	8	O	EPP WAC Length	8’h0		
cevaxm4_epp_awsize_r	3	O	EPP WAC Size	3’b0		
cevaxm4_epp_awburst_r	2	O	EPP WAC Burst Type	2’b01		
cevaxm4_epp_awlock_r	1	O	EPP WAC Lock	1’b0		
cevaxm4_epp_awcache_r	4	O	EPP WAC Cache	4’h0		
cevaxm4_epp_awprot_r	3	O	EPP WAC Protection Bits	3’b100		
cevaxm4_epp_awqos_r	4	O	EPP WAC QoS Type	4’h0		
cevaxm4_epp_awvalid_r	1	O	EPP WAC Address Valid	1’b0		
wready_epp	1	I	EPP Write Data Channel (WDC) Ready		1’b0	
cevaxm4_epp_wid_r	4	O	EPP WDC ID	4’h0		
cevaxm4_epp_wdata_r	128	O	EPP WDC Write Data	128’h0		
cevaxm4_epp_wstrb_r	16	O	EPP WDC Strobes	16’h0		
cevaxm4_epp_wlast_r	1	O	EPP WDC Last Data	1’b0		
cevaxm4_epp_wvalid_r	1	O	EPP WDC Data Valid	1’b0		
bid_epp	4	I	EPP Write Response Channel (WRC) ID		4’b0	
bresp_epp	2	I	EPP WRC Response		2’b0	
bvalid_epp	1	I	EPP WRC Valid		1’b0	
cevaxm4_epp_bready_r	1	O	EPP WRC Ready	1’b1		
cevaxm4_epp_aps_r	1	O	EPP Automatic Power Save Indication	1’b0		
div_en_epp	1	I	Slow clock enable for the EPP AXI interface		1’b1	
cevaxm4_epp_awvalid_pty	1	O	Odd Parity bit for cevaxm4_epp_awvalid_r	1’b1		BusECC
cevaxm4_epp_awaddr_pty	4	O	Parity bits for cevaxm4_epp_awaddr_r	4’h0		BusECC

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_epp_aw_pty	4	O	Parity bits for AW channel control	4’h0		BusECC
cevaxm4_epp_w_pty	3	O	Parity bits for W channel control	3’h0		BusECC
cevaxm4_epp_wvalid_pty	1	O	Odd parity bit for cevaxm4_epp_wvalid_r	1’b1		BusECC
cevaxm4_epp_wdata_ecc	28	O	ECC bits for cevaxm4_epp_wdata_r	28’h0		BusECC
cevaxm4_epp_bready_pty	1	O	Odd parity bit for cevaxm4_epp_bready_r	1’b0		BusECC
cevaxm4_epp_arvalid_pty	1	O	Odd parity bit for cevaxm4_epp_arvalid_r	1’b1		BusECC
cevaxm4_epp_araddr_pty	4	O	Parity bits for cevaxm4_epp_araddr_r	4’h0		BusECC
cevaxm4_epp_ar_pty	4	O	Parity bits for AR channel control	4’b0101		BusECC
cevaxm4_epp_rready_pty	1	O	Odd parity bit for cevaxm4_epp_rready_r	1’b0		BusECC
epp_awready_pty	1	I	Odd parity bit for awready_epp		1’b1	BusECC
epp_wready_pty	1	I	Odd parity bit for wready_epp		1’b1	BusECC
epp_bvalid_pty	1	I	Odd parity bit for bvalid_epp		1’b1	BusECC
epp_b_pty	1	I	Parity bit for bid_epp and bresp_epp		1’b0	BusECC
epp_arready_pty	1	I	Odd parity bit for arready_epp		1’b1	BusECC
epp_rvalid_pty	1	I	Odd parity bit for rvalid_epp		1’b1	BusECC
epp_rdata_ecc_pty	28	I	ECC bits for rdata_epp		28’h0	BusECC
epp_r_pty	2	I	Parity bits for R channel control signals		2’h0	BusECC
cevaxm4_epp_rdata_error_ind_d1	1	O	Indicates a correctable error on rdata_epp	1’b0		BusECC
cevaxm4_epp_fatal	1	O	Indicates a fatal error on the AXI bus	1’b0		BusECC
cevaxm4_epp_fatal_src	6	O	Indicates the source of the fatal error	6’h0		BusECC
err_ack_epp	1	I	Acknowledgement from system to clear EPP ECC error indications		1’b0	BusECC

Name	Size	I/O	Description	SAR	Default	Configuration
External Data Port Interface (Master AXI)						
arready_edp	1	I	EDP RAC Ready		1’b0	
cevaxm4_edp_arid_r	4	O	EDP RAC ID	4’b0		
cevaxm4_edp_araddr_r	32	O	EDP RAC Address	32’h0		
cevaxm4_edp_arlen_r	8	O	EDP RAC Length	8’h0		
cevaxm4_edp_arsize_r	3	O	EDP RAC Size	3’b0		
cevaxm4_edp_arburst_r	2	O	EDP RAC Burst Type	2’b0		
cevaxm4_edp_arlock_r	1	O	EDP RAC Lock	1’b0		
cevaxm4_edp_arcache_r	4	O	EDP RAC Cache	4’h0		
cevaxm4_edp_arqos_r	4	O	EDP RAC QoS Type	4’h0		
cevaxm4_edp_arprot_r	3	O	EDP RAC Protection	3’b0		
cevaxm4_edp_arvalid_r	1	O	EDP RAC Address Valid	1’b0		
rid_edp	4	I	EDP RDC ID		4’b0	
rdata_edp	128/256	I	EDP RDC Read Data		128’h0	
rresp_edp	4	I	EDP RDC Response		4’b0	
rlast_edp	1	I	EDP RDC Last Data		1’b0	
rvalid_edp	1	I	EDP RDC Data Valid		1’b0	
cevaxm4_edp_rready_r	1	O	EDP RDC Ready	1’b0		
awready_edp	1	I	EDP WAC Ready		1’b0	
cevaxm4_edp_awid_r	4	O	EDP WAC ID	4’b0		
cevaxm4_edp_awaddr_r	32	O	EDP WAC Address	32’h0		
cevaxm4_edp_awlen_r	8	O	EDP WAC Length	8’h0		

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_edp_awsize_r	3	O	EDP WAC Size	3’b0		
cevaxm4_edp_awburst_r	2	O	EDP WAC Burst Type	2’b0		
cevaxm4_edp_awlock_r	1	O	EDP WAC Lock	1’b0		
cevaxm4_edp_awcache_r	4	O	EDP WAC Cache	4’h0		
cevaxm4_edp_awqos_r	4	O	EDP WAC QoS Type	4’h0		
cevaxm4_edp_awprot_r	3	O	EDP WAC Protection Bits	3’b0		
cevaxm4_edp_awvalid_r	1	O	EDP WAC Address Valid	1’b0		
wready_edp	1	I	EDP WDC Ready		1’b0	
cevaxm4_edp_wid_r	4	O	EDP WDC ID	4’b0		
cevaxm4_edp_wdata_r	128/256	O	EDP WDC Write Data	128’h0		
cevaxm4_edp_wstrb_r	16/32	O	EDP WDC Strobes	16’h0		
cevaxm4_edp_wlast_r	1	O	EDP WDC Last Data	1’b0		
cevaxm4_edp_wvalid_r	1	O	EDP WDC Data Valid	1’b0		
bid_edp	4	I	EDP WRC ID		4’b0	
bresp_edp	2	I	EDP WRC Response		2’b0	
bvalid_edp	1	I	EDP WRC Valid		1’b0	
cevaxm4_edp_bready_r	1	O	EDP WRC Ready	1’b1		
cevaxm4_edp_aps_r	1	O	EDP Automatic Power Save Indication	1’b0		
cevaxm4_edp_awvalid_pty_r	1	O	Odd Parity bit for cevaxm4_edp_awvalid_r	1’b1		BusECC
cevaxm4_edp_awaddr_pty_r	4	O	Parity bits for cevaxm4_edp_awaddr_r	4’hf		BusECC
cevaxm4_edp_aw_pty_r	5	O	Parity bits for AW channel control	5’h1f		BusECC
cevaxm4_edp_w_pty_r	3/5	O	Parity bits for W channel control	3/5{1’b1}		BusECC

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_edp_wvalid_pty_r	1	O	Odd parity bit for cevaxm4_edp_wvalid_r	1’b1		BusECC
cevaxm4_edp_wdata_ecc_r	28/56	O	ECC bits for cevaxm4_edp_wdata_r	28/56’h0		BusECC
cevaxm4_edp_bready_pty	1	O	Odd parity bit for cevaxm4_edp_bready_r	1’b0		BusECC
cevaxm4_edp_arvalid_pty_r	1	O	Odd parity bit for cevaxm4_edp_arvalid_r	1’b1		BusECC
cevaxm4_edp_araddr_pty_r	4	O	Parity bits for cevaxm4_edp_araddr_r	4’hf		BusECC
cevaxm4_edp_ar_pty_r	5	O	Parity bits for AR channel control	5’h1f		BusECC
cevaxm4_edp_rready_pty_r	1	O	Odd parity bit for cevaxm4_edp_rready_r	1’b1		BusECC
awready_pty_edp	1	I	Odd parity bit for awready_edp		1’b1	BusECC
wready_pty_edp	1	I	Odd parity bit for wready_edp		1’b1	BusECC
bvalid_pty_edp	1	I	Odd parity bit for bvalid_edp		1’b1	BusECC
b_pty_edp	1	I	Parity bit for bid_edp and bresp_edp		1’b0	BusECC
arready_pty_edp	1	I	Odd parity bit for arready_edp		1’b1	BusECC
rvalid_pty_edp	1	I	Odd parity bit for rvalid_edp		1’b1	BusECC
rdata_ecc_edp	28/56	I	ECC bits for rdata_edp		28/56’h0	BusECC
r_pty_edp	2	I	Parity bits for R channel control signals		2’h0	BusECC
cevaxm4_edp_fatal_err_r	1	O	Indicates a fatal error on the AXI bus	1’b0		BusECC
cevaxm4_edp_fatal_src_r	5	O	Indicates the source of the fatal error	5’h0		BusECC
cevaxm4_edp_corr_err_r	1	O	Indicates a correctable error on rdata_edp	1’b0		BusECC
err_ack_edp	1	I	Acknowledgement from system to clear EDP ECC error indications		1’b0	BusECC

Name	Size	I/O	Description	SAR	Default	Configuration
AXI Master Port Interface (AXIM0, AXIM1) X = 0,1 if used						
arready_aximX	1	I	AXI Master RAC Ready		1’b0	AXIM
cevaxm4_aximX_arid_r	4	O	AXI Master RAC ID	4’b0		AXIM
cevaxm4_aximX_araddr_r	32	O	AXI Master RAC Address	32’h0		AXIM
cevaxm4_aximX_arlen_r	8	O	AXI Master RAC Length	8’h0		AXIM
cevaxm4_aximX_arsize_r	3	O	AXI Master RAC Size	3’b0		AXIM
cevaxm4_aximX_arburst_r	2	O	AXI Master RAC Burst Type	2’b0		AXIM
cevaxm4_aximX_arlock_r	1	O	AXI Master RAC Lock	1’b0		AXIM
cevaxm4_aximX_arcache_r	4	O	AXI Master RAC Cache	4’h0		AXIM
cevaxm4_aximX_arqos_r	4	O	AXI Master RAC QoS Type	4’h0		AXIM
cevaxm4_aximX_arprot_r	3	O	AXI Master RAC Protection	3’b0		AXIM
cevaxm4_aximX_arvalid_r	1	O	AXI Master RAC Address Valid	1’b0		AXIM
rid_aximX	4	I	AXI Master RDC ID		4’b0	AXIM
rdata_aximX	128	I	AXI Master RDC Read Data		128’h0	AXIM
rresp_aximX	4	I	AXI Master RDC Response		4’b0	AXIM
rlast_aximX	1	I	AXI Master RDC Last Data		1’b0	AXIM
rvalid_aximX	1	I	AXI Master RDC Data Valid		1’b0	AXIM
cevaxm4_aximX_rready_r	1	O	AXI Master RDC Ready	1’b0		AXIM
awready_aximX	1	I	AXI Master WAC Ready		1’b0	AXIM
cevaxm4_aximX_awid_r	4	O	AXI Master WAC ID	4’b0		AXIM
cevaxm4_aximX_awaddr_r	32	O	AXI Master WAC Address	32’h0		AXIM
cevaxm4_aximX_awlen_r	8	O	AXI Master WAC Length	8’h0		AXIM

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_aximX_awsize_r	3	O	AXI Master WAC Size	3’b0		AXIM
cevaxm4_aximX_awburst_r	2	O	AXI Master WAC Burst Type	2’b0		AXIM
cevaxm4_aximX_awlock_r	1	O	AXI Master WAC Lock	1’b0		AXIM
cevaxm4_aximX_awcache_r	4	O	AXI Master WAC Cache	4’h0		AXIM
cevaxm4_aximX_awqos_r	4	O	AXI Master WAC QoS Type	4’h0		AXIM
cevaxm4_aximX_awprot_r	3	O	AXI Master WAC Protection Bits	3’b0		AXIM
cevaxm4_aximX_awvalid_r	1	O	AXI Master WAC Address Valid	1’b0		AXIM
wready_aximX	1	I	AXI Master WDC Ready		1’b0	AXIM
cevaxm4_aximX_wid_r	4	O	AXI Master WDC ID	4’b0		AXIM
cevaxm4_aximX_wdata_r	128	O	AXI Master WDC Write Data	128’h0		AXIM
cevaxm4_aximX_wstrb_r	16	O	AXI Master WDC Strobes	16’h0		AXIM
cevaxm4_aximX_wlast_r	1	O	AXI Master WDC Last Data	1’b0		AXIM
cevaxm4_aximX_wvalid_r	1	O	AXI Master WDC Data Valid	1’b0		AXIM
bid_aximX	4	I	AXI Master WRC ID		4’b0	AXIM
bresp_aximX	2	I	AXI Master WRC Response		2’b0	AXIM
bvalid_aximX	1	I	AXI Master WRC Valid		1’b0	AXIM
cevaxm4_aximX_bready_r	1	O	AXI Master WRC Ready	1’b1		AXIM
cevaxm4_aximX_aps_r	1	O	AXI Master Automatic Power Save Indication	1’b0		AXIM
div_en_aximX	1	I	Slow clock enable for AXIMX interfaces		1’b1	AXIM
cevaxm4_aximX_awvalid_pty_r	1	O	Odd Parity bit for cevaxm4_aximX_awvalid_r	1’b1		AXIM and BusECC
cevaxm4_aximX_awaddr_pty_r	4	O	Parity bits for cevaxm4_aximX_awaddr_r	4’hf		AXIM and BusECC

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_aximX_aw_pty_r	5	O	Parity bits for AW channel control	5’h1f		AXIM and BusECC
cevaxm4_aximX_w_pty_r	3/5	O	Parity bits for W channel control	3/5{1’b1}		AXIM and BusECC
cevaxm4_aximX_wvalid_pty_r	1	O	Odd parity bit for cevaxm4_aximX_wvalid_r	1’b1		AXIM and BusECC
cevaxm4_aximX_wdata_ecc_r	28/56	O	ECC bits for cevaxm4_aximX_wdata_r	28/56’h0		AXIM and BusECC
cevaxm4_aximX_bready_pty	1	O	Odd parity bit for cevaxm4_aximX_bready_r	1’b0		AXIM and BusECC
cevaxm4_aximX_arvalid_pty_r	1	O	Odd parity bit for cevaxm4_aximX_arvalid_r	1’b1		AXIM and BusECC
cevaxm4_aximX_araddr_pty_r	4	O	Parity bits for cevaxm4_aximX_araddr_r	4’hf		AXIM and BusECC
cevaxm4_aximX_ar_pty_r	5	O	Parity bits for AR channel control	5’h1f		AXIM and BusECC
cevaxm4_aximX_rready_pty_r	1	O	Odd parity bit for cevaxm4_aximX_rready_r	1’b1		AXIM and BusECC
awready_pty_aximX	1	I	Odd parity bit for awready_aximX		1’b1	AXIM and BusECC
wready_pty_aximX	1	I	Odd parity bit for wready_aximX		1’b1	AXIM and BusECC
bvalid_pty_aximX	1	I	Odd parity bit for bvalid_aximX		1’b1	AXIM and BusECC
b_pty_aximX	1	I	Parity bit for bid_aximX and bresp_aximX		1’b0	AXIM and BusECC
arready_pty_aximX	1	I	Odd parity bit for arready_aximX		1’b1	AXIM and BusECC
rvalid_pty_aximX	1	I	Odd parity bit for rvalid_aximX		1’b1	AXIM and BusECC
rdata_ecc_aximX	28/56	I	ECC bits for rdata_aximX		28/56’h0	AXIM and BusECC
r_pty_aximX	2	I	Parity bits for R channel control signals		2’h0	AXIM and BusECC
cevaxm4_aximX_fatal_err_r	1	O	Indicates a fatal error on the AXI bus	1’b0		AXIM and BusECC
cevaxm4_aximX_fatal_src_r	5	O	Indicates the source of the fatal error	5’h0		AXIM and BusECC
cevaxm4_aximX_corr_err_r	1	O	Indicates a correctable error on rdata_aximX	1’b0		AXIM and BusECC
err_ack_aximX	1	I	Acknowledgement from system to clear AXIMX ECC error indications		1’b0	AXIM and BusECC

Name	Size	I/O	Description	SAR	Default	Configuration
External Device Access Port Interface (Slave AXI)						
cevaxm4_edap_arready_r	1	O	EDAP RAC Ready	1’b0		
arid_edap	16	I	EDAP RAC ID		16’h0	
araddr_edap	23	I	EDAP RAC Address		23’h0	
arlen_edap	8	I	EDAP RAC Length		8’h0	
arsize_edap	3	I	EDAP RAC Size		3’b0	
arburst_edap	2	I	EDAP RAC Burst Type		2’b0	
arvalid_edap	1	I	EDAP RAC Address Valid		1’b0	
cevaxm4_edap_rid_r	16	O	EDAP RDC ID	16’h0		
cevaxm4_edap_rdata_r	128	O	EDAP RDC Read Data	128’h0		
cevaxm4_edap_rresp_r	2	O	EDAP RDC Response	2’b0		
cevaxm4_edap_rlast_r	1	O	EDAP RDC Last Data	1’b0		
cevaxm4_edap_rvalid_r	1	O	EDAP RDC Data Valid	1’b0		
rready_edap	1	I	EDAP RDC Ready		1’b0	
cevaxm4_edap_awready_r	1	O	EDAP WAC Ready	1’b0		
awid_edap	16	I	EDAP WAC ID		16’h0	
awaddr_edap	23	I	EDAP WAC Address		23’h0	
awlen_edap	8	I	EDAP WAC Length		8’h0	
awsize_edap	3	I	EDAP WAC Size		3’b0	
awburst_edap	2	I	EDAP WAC Burst Type		2’b0	
awvalid_edap	1	I	EDAP WAC Address Valid		1’b0	
cevaxm4_edap_wready_r	1	O	EDAP WDC Ready	1’b0		

Name	Size	I/O	Description	SAR	Default	Configuration
wdata_edap	128	I	EDAP WDC Write Data		128’h0	
wstrb_edap	16	I	EDAP WDC Strobes		16’h0	
wlast_edap	1	I	EDAP WDC Last Data		1’b0	
wvalid_edap	1	I	EDAP WDC Data Valid		1’b0	
cevaxm4_edap_bid_r	16	O	EDAP WRC ID	16’h0		
cevaxm4_edap_bresp_r	2	O	EDAP WRC Response	2’b0		
cevaxm4_edap_bvalid_r	1	O	EDAP WRC Valid	1’b0		
bready_edap	1	I	EDAP WRC Ready		1’b0	
cevaxm4_edap_wr_awready_pty_r	1	O	Odd parity bit for cevaxm4_edap_awready_r	1’b1		Bus ECC
cevaxm4_edap_wr_wready_pty_r	1	O	Odd parity bit for cevaxm4_edap_wready_r	1’b1		Bus ECC
cevaxm4_edap_wr_bvalid_pty_r	1	O	Odd parity bit for cevaxm4_edap_bvalid_r	1’b1		Bus ECC
cevaxm4_edap_wr_b_pty_cr	3	O	Parity bits for cevaxm4_edap_bid_r and cevaxm4_edap_bresp_r	3’h0		Bus ECC
cevaxm4_edap_rd_arready_pty_r	1	O	Odd parity bit for cevaxm4_edap_arready_r	1’b1		Bus ECC
cevaxm4_edap_rd_rvalid_pty_r	1	O	Odd parity bit for cevaxm4_edap_rvalid_r	1’b1		Bus ECC
cevaxm4_edap_rd_data_ecc_r	28/56	O	ECC bits for cevaxm4_edap_rdata_r	28/56’h0		Bus ECC
cevaxm4_edap_rd_r_pty_cr	3	O	Parity bits for R channel control signals	3’h0		Bus ECC
awvalid_pty_edap	1	I	Odd Parity bit for awvalid_edap		1’b1	Bus ECC
awaddr_pty_edap	3	I	Parity bits for awaddr_edap		3’h0	Bus ECC
aw_pty_edap	4	I	Parity bits for AW channel control		4’h0	Bus ECC
w_pty_edap	3/5	I	Parity bits for W channel control		3/5’h0	Bus ECC
wvalid_pty_edap	1	I	Odd parity bit for wvalid_edap		1’b1	Bus ECC

Name	Size	I/O	Description	SAR	Default	Configuration
wdata_ecc_edap	28/56	I	ECC bits for wdata_edap		28/56’h0	Bus ECC
bready_pty_edap	1	I	Odd parity bit for bready_edap		1’b1	Bus ECC
arvalid_pty_edap	1	I	Odd parity bit for arvalid_edap		1’b1	Bus ECC
araddr_pty_edap	3	I	Parity bits for araddr_edap		3’h0	Bus ECC
ar_pty_edap	4	I	Parity bits for AR channel control		4’h0	Bus ECC
rready_pty_edap	1	I	Odd parity bit for rready_edap		1’b1	Bus ECC
cevaxm4_edap_fatal_err_r	1	O	Indicates a fatal error on the AXI bus	1’b0		Bus ECC
cevaxm4_edap_fatal_src_r	5	O	Indicates the source of the fatal error	5’h0		Bus ECC
cevaxm4_edap_corr_err_r	1	O	Indicates a correctable error on araddr_edap	1’b0		Bus ECC
err_ack_edap	1	I	Acknowledgement from system to clear EDAP ECC error indications		1’b0	Bus ECC
AXI Slave Port Interface AXIsX (slv0, slv1 and slv2) X = 0,1,2 if used						
cevaxm4_slvX_arready_r	1	O	AXIsX RAC Ready	1’b0		AXIsX
arid_slvX	16	I	AXIsX RAC ID		16’h0	AXIsX
araddr_slvX	23	I	AXIsX RAC Address		23’h0	AXIsX
arlen_slvX	8	I	AXIsX RAC Length		8’h0	AXIsX
arsize_slvX	3	I	AXIsX RAC Size		3’b0	AXIsX
arburst_slvX	2	I	AXIsX RAC Burst Type		2’b0	AXIsX
arvalid_slvX	1	I	AXIsX RAC Address Valid		1’b0	AXIsX
cevaxm4_slvX_rid_r	16	O	AXIsX RDC ID	16’h0		AXIsX
cevaxm4_slvX_rdata_r	128, 256	O	AXIsX RDC Read Data	128’h0, 256’h0		AXIsX
cevaxm4_slvX_rresp_r	2	O	AXIsX RDC Response	2’b0		AXIsX

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_slvX_rlast_r	1	O	AXIsX RDC Last Data	1’b0		AXIsX
cevaxm4_slvX_rvalid_r	1	O	AXIsX RDC Data Valid	1’b0		AXIsX
rready_slvX	1	I	AXIsX RDC Ready		1’b0	AXIsX
cevaxm4_slvX_awready_r	1	O	AXIsX WAC Ready	1’b0		AXIsX
awid_slvX	16	I	AXIsX WAC ID		16’h0	AXIsX
awaddr_slvX	23	I	AXIsX WAC Address		23’h0	AXIsX
awlen_slvX	8	I	AXIsX WAC Length		8’h0	AXIsX
awsize_slvX	3	I	AXIsX WAC Size		3’b0	AXIsX
awburst_slvX	2	I	AXIsX WAC Burst Type		2’b0	AXIsX
awvalid_slvX	1	I	AXIsX WAC Address Valid		1’b0	AXIsX
cevaxm4_slvX_wready_r	1	O	AXIsX WDC Ready	1’b0		AXIsX
wdata_slvX	128, 256	I	AXIsX WDC Write Data		128’h0, 256’h0	AXIsX
wstrb_slvX	16, 32	I	AXIsX WDC Strobes		16’h0, 32’h0	AXIsX
wlast_slvX	1	I	AXIsX WDC Last Data		1’b0	AXIsX
wvalid_slvX	1	I	AXIsX WDC Data Valid		1’b0	AXIsX
cevaxm4_slvX_bid_r	16	O	AXIsX WRC ID	16’h0		AXIsX
cevaxm4_slvX_bresp_r	2	O	AXIsX WRC Response	2’b0		AXIsX
cevaxm4_slvX_bvalid_r	1	O	AXIsX WRC Valid	1’b0		AXIsX
bready_slvX	1	I	AXIsX WRC Ready		1’b0	AXIsX
cevaxm4_slvX_wr_awready_pty_r	1	O	Odd parity bit for cevaxm4_axisX_awready_r	1’b1		AXIsX and Bus ECC
cevaxm4_slvX_wr_wready_pty_r	1	O	Odd parity bit for cevaxm4_axisX_wready_r	1’b1		AXIsX and Bus ECC

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_slvX_wr_bvalid_pty_r	1	O	Odd parity bit for cevaxm4_axisX_bvalid_r	1’b1		AXIsX and Bus ECC
cevaxm4_slvX_wr_b_pty_cr	3	O	Parity bits for cevaxm4_axisX_bid_r and cevaxm4_axisX_bresp_r	3’h0		AXIsX and Bus ECC
cevaxm4_slvX_rd_arready_pty_r	1	O	Odd parity bit for cevaxm4_axisX_arready_r	1’b1		AXIsX and Bus ECC
cevaxm4_slvX_rd_rvalid_pty_r	1	O	Odd parity bit for cevaxm4_axisX_rvalid_r	1’b1		AXIsX and Bus ECC
cevaxm4_slvX_rd_data_ecc_r	28/56	O	ECC bits for cevaxm4_axisX_rdata_r	28/56’h0		AXIM and BusECC
cevaxm4_slvX_rd_r_pty_cr	3	O	Parity bits for R channel control signals	3’h0		AXIM and BusECC
awvalid_pty_slvX	1	I	Odd Parity bit for awvalid_axisX		1’b1	AXIsX and Bus ECC
awaddr_pty_slvX	3	I	Parity bits for awaddr_axisX		3’h0	AXIsX and Bus ECC
aw_pty_slvX	4	I	Parity bits for AW channel control		4’h0	AXIsX and Bus ECC
w_pty_slvX	3/5	I	Parity bits for W channel control		3/5’h0	AXIsX and Bus ECC
wvalid_pty_slvX	1	I	Odd parity bit for wvalid_axisX		1’b1	AXIM and BusECC
wdata_ecc_slvX	28/56	I	ECC bits for wdata_axisX		28/56’h0	AXIM and BusECC
bready_pty_slvX	1	I	Odd parity bit for bready_axisX		1’b1	AXIsX and Bus ECC
arvalid_pty_slvX	1	I	Odd parity bit for arvalid_axisX		1’b1	AXIsX and Bus ECC
araddr_pty_slvX	3	I	Parity bits for araddr_axisX		3’h0	AXIsX and Bus ECC
ar_pty_slvX	4	I	Parity bits for AR channel control		4’h0	AXIsX and Bus ECC
rready_pty_slvX	1	I	Odd parity bit for rready_axisX		1’b0	AXIM and BusECC
cevaxm4_slvX_fatal_err_r	1	O	Indicates a fatal error on the AXI bus	1’b0		AXIM and BusECC
cevaxm4_slvX_fatal_src_r	5	O	Indicates the source of the fatal error	5’h0		AXIM and BusECC
cevaxm4_slvX_corr_err_r	1	O	Indicates a correctable error on araddr_axisX	1’b0		AXIM and BusECC

Name	Size	I/O	Description	SAR	Default	Configuration
err_ack_slvX	1	I	Acknowledgement from system to clear AXISX ECC error indications		1'b0	AXIM and BusECC
div_en_slvX	1	I	Slow clock enable for AXIsX interfaces		1'b1	AXIsX
Multicore Messaging Interface						
cevaxm4_mcci_mes_int	1	O	Multicore Messaging Interface interrupt	1'b0		
cevaxm4_mcci_rd_ind_r	32	O	Multicore Messaging Interface read indication	32'h0		
cevaxm4_snoop_sn_int	1	O	AXI Slave (EDAP, AXIs0, AXIs1 and AXIs2) snoop interrupt	1'b0		
DMA Manager Interface						
qn_desc_en	16	I	QMAN enabled descriptors increment signal		16'h0	DMAN
qman_semaphore_grant	16	I	QMAN semaphore grant		16'h0 (see note below)	DMAN
qman_semaphore_req	16	O	QMAN semaphore request	16'h0		DMAN
qman_irq	1	O	QMAN violation indications interrupt	1'b0		DMAN
I/O Port Interface (APB3 Master)						
prdata_iop	32	I	I/O Port data-in bus		32'h0	
pready_iop	1	I	I/O Port ready signal		1'b0	
pslverr_iop	1	I	I/O Port transfer failure		1'b0	
cevaxm4_iop_pwrite_r	1	O	I/O Port read/write strobe	1'b0		
cevaxm4_iop_psel_r	1	O	I/O Port select signal	1'b0		
cevaxm4_iop_paddr_r	32	O	I/O Port address bus	32'h0		
cevaxm4_iop_pwdata_r	32	O	I/O Port data-out bus	32'h0		
cevaxm4_iop_penable_r	1	O	I/O Port enable signal	1'b0		

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_iop_aps_r	1	O	I/O Port Auto power save indication	1’b0		
cevaxm4_iop_paddr_pty_r	4	O	Parity bits for cevaxm4_iop_paddr_r	4’h0		Bus ECC
cevaxm4_iop_pwdata_ecc_r	7	O	ECC bits for cevaxm4_iop_pwdata_r	7’h0		Bus ECC
cevaxm4_iop_op_pty_r	1	O	Parity bit for cevaxm4_iop_pwrite_r, cevaxm4_iop_psel_r, cevaxm4_iop_penable_r	1’b0		Bus ECC
prdata_ecc_iop	7	I	ECC bits for prdata_iop		7’h0	Bus ECC
ip_pty_iop	1	I	Parity bit for pready_iop , pslverr_iop		1’b1	Bus ECC
cevaxm4_iop_fatal_err_r	1	O	Indicates a fatal error on the APB bus	1’b0		Bus ECC
cevaxm4_iop_fatal_src_r	2	O	Indicates the source of the fatal error	2’h0		Bus ECC
cevaxm4_iop_corr_err_r	1	O	Indicates a correctable error on prdata_iop	1’b0		Bus ECC
err_ack_iop	1	I	Acknowledgement from system to clear APB ECC error indications		1’b0	Bus ECC
div_en_iop	1	I	Slow clock enable for I/O Port APB3 interface		1’b1	
CEVA-Xtend Interface (SPU)						
xtend_scalar_dst_data	32	I	Destination result from scalar external hardware at stage V1		32’h0	SEH
cevaxm4_xh_src0_m_r	32	O	Scalar Xtend unit src0 data at stage E1	32’h0		SEH
cevaxm4_xh_src1_m_r	32	O	Scalar Xtend unit src1 data at stage E1	32’h0		SEH
cevaxm4_xh_pr_en_e1	1	O	Scalar Xtend unit predicate valid indication	1’b0		SEH
cevaxm4_xh_pir_a1_r	23	O	Scalar Xtend unit instruction register at A2	23’h0		SEH
cevaxm4_xh_pir_valid_a1_r	1	O	Scalar Xtend unit instruction valid at stage A2	1’b0		SEH

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_xh_pext_a1_cr	26	O	Scalar Xtend unit output from Core extension word at stage a2	26’h0		SEH
cevaxm4_xh_pext10_valid_a1_r	1	O	Scalar Xtend unit output from Core Dispatcher 10 extension valid indication	1’b0		SEH
cevaxm4_xh_pext26_valid_a1_r	1	O	Scalar Xtend unit output from Core Dispatcher 26 extension valid indication	1’b0		SEH
CEVA-Xtend Interface (VPU)						
cxm_vpu_xtend_dest_v4_r	256	I	Destination result from VPU external hardware at stage V5		256’h0	VEH
cxm_vpu_xtend_en_v4_r	32	I	Destination enable result from VPU external hardware at stage V5		32’h0	VEH
cxm_vpu_xtend_src0_e2	256	O	VPU Xtend unit src0 data at stage E2	256’h0		VEH
cxm_vpu_xtend_src1_e2	256	O	VPU Xtend unit src1 data at stage E2	256’h0		VEH
cxm_vpu_xtend_vpr_e2	32	O	VPU Xtend unit predicate vector indication	32’h0		VEH
cxm_vpu_xtend_pir_e1_r	23	O	VPU Xtend unit instruction register at E2	23’h0		VEH
cxm_vpu_xtend_pext_e1_r	26	O	VPU Xtend unit output from Core extension word at stage E2	26’h0		VEH
cxm_vpu_xtend_pir_valid_e1_r	1	O	VPU Xtend unit instruction valid at stage E2	1’b0		VEH
cxm_vpu_xtend_pext10_valid_e1_r	1	O	VPU Xtend unit output from Core Dispatcher 10 extension valid indication	1’b0		VEH
cxm_vpu_xtend_pext26_valid_e1_r	1	O	VPU Xtend unit output from Core Dispatcher 26 extension valid indication	1’b0		VEH
Interrupts Interface						
int0	1	I	Maskable interrupt 0		1'b0	
int1	1	I	Maskable interrupt 1		1'b0	

Name	Size	I/O	Description	SAR	Default	Configuration
int2	1	I	Maskable interrupt 2		1'b0	
nmi	1	I	Non-maskable interrupt		1'b0	
vint	1	I	Request signal for vector interrupt		1'b0	
vector	32	I	Address of the vector interrupt		32'h0	
cevaxm4_code_ecc_int_r	1	O	One ECC error interrupt output pin	1’b0		CEVAXM4-ECC
cevaxm4_code_ecc_2err_int_r	1	0	Two ECC error interrupt output pin	1’b0		CEVAXM4-ECC
cevaxm4_code_tag_ecc_int_r	1	O	ECC stretched error interrupt for tag array	1’b0		CEVAXM4-ECC
cevaxm4_uop_int_r	1	O	Undefined Opcode stretched interrupt	1’b0		
cevaxm4_seq_int0_ack_n_r	1	O	int0 stretched acknowledge active low	1'b1		
cevaxm4_seq_int1_ack_n_r	1	O	int1 stretched acknowledge active low	1'b1		
cevaxm4_seq_int2_ack_n_r	1	O	int2 stretched acknowledge active low	1'b1		
cevaxm4_seq_int3_ack_n_r	1	O	Int3 stretched acknowledge active low	1'b1		
cevaxm4_seq_int4_ack_n_r	1	O	Int4 stretched acknowledge active low	1'b1		
cevaxm4_seq_bp_ack_n_r	1	O	*trape*/breakpoint stretched acknowledge active low	1'b1		
cevaxm4_seq_nmi_ack_n_r	1	O	*nmi* stretched acknowledge active low	1'b1		
cevaxm4_seq_vint_ack_n_r	1	O	*vint* stretched acknowledge active low	1'b1		
cevaxm4_ecccor_int_r	1	O	Single correctable error on L1DM access – Interrupt output pin	1’b0		CEVAXM4-ECC
cevaxm4_eccerr_int_r	1	O	Double error detected on L1DM access – Interrupt output pin	1’b0		CEVAXM4-ECC
cevaxm4_epp_wdog_viol_r	1	O	Indicates EPP Watchdog timeout	1’b0		
cevaxm4_edp_wdog_viol_r	1	O	Indicates EDP Watchdog timeout	1’b0		

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_aximX_wdog_viol_r	1	O	Indicates AXIMX Watchdog timeout	1’b0		AXImX
cevaxm4_iop_wdog_viol_r	1	O	Indicates IOP Watchdog timeout	1’b0		
Semaphore Interface						
cevaxm4_seq_trp_srv_r	1	O	Trap service routine indication, set when the Core is in a trap	1’b0		
APB Debug Slave Port						
ocem_apbs_paddr	32	I	OCEM APB Slave paddr signal (only bits [11:2] are connected)		32’b0	
ocem_apbs_penable	1	I	OCEM APB Slave penable signal		1’b0	
ocem_apbs_psel	1	I	OCEM APB Slave psel signal		1’b0	
ocem_apbs_pwdata	32	I	OCEM APB Slave pwdata signal		32’b0	
ocem_apbs_pwrite	1	I	OCEM APB Slave pwrite signal		1’b0	
cevaxm4_ocem_apbs_prdata	32	O	OCEM APB Slave read data	32’b0		
cevaxm4_ocem_apbs_pready	1	O	OCEM APB Slave Ready Signal	1’b0		
cevaxm4_ocem_apbs_pslverr	1	O	OCEM APB Slave Error Signal	1’b0		
Real-Time Trace Interface (Internal ETM-R4 Only)						
ext_trigger	3	I	External trace triggers		3’b0	RTT
cevaxm4_w_inrange_r	3	O	Comparator In-range indications	3’h0		RTT
cevaxm4_ocm_pa_mtch_d1	1	O	OCEM Program address breakpoint #1 match at D1 stage	1’b0		RTT
cevaxm4_ocm_bpint_trig	5	O	OCEM breakpoint triggers	5’h00		RTT
etm_ni_dbg_en	1	I	Non-invasive Debug Enable		1’b0	RTT
etm_in_dbg_en	1	I	Invasive Debug Enable		1’b0	RTT

Name	Size	I/O	Description	SAR	Default	Configuration
etm_atb_clk	1	I	ATB Interface Clock		1’b0	RTT
etm_atb_clk_en	1	I	ATB Clock Enable		1’b0	RTT
etm_atb_ready	1	I	ATB Interface Data can be accepted		1’b0	RTT
etm_atb_fvalid	1	I	ATB Data Valid		1’b0	RTT
etm_trigger_ack	1	I	ATB Trigger Acknowledge		1’b0	RTT
etm_trigger_bypass	1	I	ATB Trigger Synchronization Bypass		1’b0	RTT
etm_rst_bypass	1	I	Reset Synchronization Bypass		1’b0	RTT
etm_max_ext_in	3	I	Number of External Inputs supported by the IC (max 4)		3’b0	RTT
etm_max_ext_out	2	I	Number of External Outputs supported by the IC (max 2)		2’b0	RTT
etm_evnt_bus	47	I	Status of Performance Monitoring Events or extended External Inputs		47’h0	RTT
etm_ext_in	4	I	External Inputs		4’h0	RTT
cevaxm4_etm_pwr_up	1	O	Asserted when the ETM-R4 is in use	1’b0		RTT
cevaxm4_etm_wfi_ready_n	1	O	Indicates that the ETM-R4 FIFO is empty	1’b0		RTT
cevaxm4_etm_atb_data	32	O	ATB Interface Data	32’h 66666667		RTT
cevaxm4_etm_atb_valid	1	O	ATB Interface Data Valid	1’b0		RTT
cevaxm4_etm_atb_bytes	2	O	ATB Data Size	2’b0		RTT
cevaxm4_etm_atb_fready	1	O	ATB Interface FIFO Flush Complete	1’b1		RTT
cevaxm4_etm_trigger	1	O	Trigger Request Status	1’b0		RTT
cevaxm4_etm_atb_id	7	O	ATB Interface Trace Source ID	7’h0		RTT
cevaxm4_etm_en	1	O	Trace Output Enabled	1’b0		RTT

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_etm_asic	8	O	Contents of ETM-R4 ASICCTL register	8’h0		RTT
cevaxm4_etm_fifo_peek	7	O	Indicates when certain events occur before being written to ETM-R4 FIFO	7’h0		RTT
cevaxm4_etm_ext_out	2	O	External Outputs	2’b0		RTT
apb_clk_dbg	1	I	APB Slave Clock		1’b0	RTT
apb_clk_en_dbg	1	I	APB Slave Clock Enable		1’b0	RTT
apb_reset_dbg_n	1	I	APB Slave Interface Reset		1’b0	RTT
apb_enable_dbg	1	I	APB Slave Interface is enabled for transfer		1’b0	RTT
apb_sel_dbg	1	I	APB Slave Select		1’b0	RTT
apb_addr_dbg	32	I	APB Slave Address Bus ● Bits [11:2] are connected to ETM. ● Bit [31] = 1 indicates access from hardware. ● Bit [31] = 0 indicates access from software.		32’h0	RTT
apb_write_dbg	1	I	APB Slave Write Access (negated during a Read)		1’b0	RTT
apb_wdata_dbg	32	I	APB Slave write data		32’h0	RTT
cevaxm4_etm_apb_rdata_dbg	32	O	APB Slave Read Data	32’h0		RTT
cevaxm4_etm_apb_ready_dbg	1	O	APB Slave Ready signal to extend APB transfers	1’b1		RTT
sysporeset_n	1	I	Power on reset for trace modules only		Mandatory	

Name	Size	I/O	Description	SAR	Default	Configuration
Real-Time Trace Interface (External ETM-R4 Only)						
ext_trigger	3	I	External trace triggers		3’b0	RTT
etm_wfi_ready_n	1	I	Indicates that the ETM-R4 FIFO is empty		1’b0	RTT
cevaxm4_w_etm_wfi_pending	1	O	Core is going into Core Idle	1’b0		RTT
cevaxm4_w_inrange_r	3	O	Comparator In-range indications	3’h0		RTT
cevaxm4_ocm_pa_mtch_d1	1	O	OCEM Program address breakpoint #1 match at D1 stage	1’b0		RTT
cevaxm4_ocm_bpint_trig	5	O	OCEM breakpoint triggers	5’h00		RTT
cevaxm4_w_debug_e5	1	O	Core is in debug mode	1’b0		RTT
cevaxm4_w_etm1da_r	32	O	ETM1 data address vector	32’h0		RTT
cevaxm4_w_etm1dctl_r	12	O	ETM1 data interface control vector	12’h0		RTT
cevaxm4_w_etm1dd_r	64	O	ETM1 data vector	64’h0		RTT
cevaxm4_w_etm1ia_r	31	O	ETM1 instruction address vector	31’h0		RTT
cevaxm4_w_etm1iactl_r	14	O	ETM1 instruction interface control	14’h0		RTT
cevaxm4_w_etm_cid_r	32	O	Core predicate data	32’h0		RTT
sysporeset_n	1	I	Power on reset for trace modules only		Mandatory	
OCEM Interface						
ext_bp1_req	1	I	External breakpoint request #1		1'b0	
ext_bp2_req	1	I	External breakpoint request #2		1'b0	
ext_pmem_wr	1	I	External program memory write indication		1’b0	
bs_reg_tdo	1	I	Boundary scan register Test Data Out signal		1'b0	
tck	1	I	JTAG clock		1'b0	

Name	Size	I/O	Description	SAR	Default	Configuration
jt_ap	1	I	OCEM scan-chains access type JTAG/APB		1’b0	
tms	1	I	JTAG state machine control signal		1'b0	
tdi	1	I	JTAG protocol test data-in		1'b0	
cevaxm4_ocm_tdo_r	1	O	JTAG protocol test data-out	1'b0		
cevaxm4_ocm_tdo_oen_r	1	O	JTAG protocol test data output enable signal	1'b1		
cevaxm4_ocm_jtag_state_r	4	O	JTAG state	4'hf		
cevaxm4_ocm_gp_out_r	4	O	OCEM general purpose outputs	4'b0		
cevaxm4_ocm_ext1_ack_r	1	O	Acknowledge signal for breakpoint #1	1'b0		
cevaxm4_ocm_ext2_ack_r	1	O	Acknowledge signal for breakpoint #2	1'b0		
cevaxm4_ocm_core_rst_r	1	O	The OCEM reset signal to the core, active high	1’b0		
cevaxm4_ocm_debug_r	1	O	OCEM debug mode indication	1’b0		
PSU and AXI Low-Power Interfaces (For more details about the PSU interface, see the *PSU* section in the *CEVA-XM4 Architecture Specification Volume III (MSS)*)						
csysreq	1	I	AXI Low Power Request to switch Light Sleep to Standby and vice versa		1’b1	
core_rcvr	1	I	Recover from STAND BY/LIGHT SLEEP modes		1’b0	
stop_sd	1	I	Stop core from shutdown. Restores the Core’s power after Core is under power OFF state.		1’b0	
cevaxm4_psu_cactive_r	1	O	AXI Low Power Active indication	1’b1		
cevaxm4_psu_csysack_r	1	O	AXI Low Power Acknowledge	1’b1		

Name	Size	I/O	Description	SAR	Default	Configuration
cevaxm4_psu_rtck_r	1	O	Return Test Clock (tck synchronized to ceva_free_clk)	1’b0		
cevaxm4_psu_dsp_idle_r	1	O	DSP Idle Indication (both clock and MSS clock can be turned off)	1'b0		
cevaxm4_psu_core_idle_r	1	O	Core Idle Indication (Core clocks from PSU to core are shut down)	1'b0		
cevaxm4_psu_pshtdwn_r	8	O	DSP Power Shutdown Request per unit domains {1 core+ MSS/1 Emulation/4 blocks of D-TCM, 1 for P-TCM and 1 for P-$}	8'hff		
cevaxm4_psu_sys_pshtdwn_r	6	O	Memories retention mode- indication for Deep Sleep {4 blocks of D-TCM, 1 for P-TCM and 1 for P-$}	6'3ff		
DFT Interface (For more details, see Section 20.)						
testmodep	1	I	Controls all clocks and reset signals during test (ensure that only the ATPG clock and reset are used)		1’b0	
tst_gatedclock	1	I	Provides external control on the clock gaters for various test modes		1’b0	
tst_mem_gatedclock	1	I	Provides external control on the memory clock gaters for various test modes		1’b0	
bist_prog_in	U	I	PMEM BIST input bus		U	BIST
bist_data_in	U	I	DMEM BIST input bus		U	BIST
cevaxm4_bist_prog_out	U	O	PMEM BIST output bus	U		BIST
cevaxm4_bist_data_out	U	O	DMEM BIST output bus	U		BIST
cevaxm4_bist_dcdata_out	U	O	DC DMEM BIST output bus	U		BIST

Name	Size	I/O	Description	SAR	Default	Configuration
Verification Indications						
cevaxm4_cverbit_r	1	O	Verification error bit, set if a functional test fails.	1’b0		
cevaxm4_seq_eotbit_r	1	O	End of Test bit, set when a functional test is finished	1’b0		
Data DMA Indications						
next_ddma	1	I	External control over DDMA Q (indication to progress to the next Q entry)		1’b0	
ext_ddma_dbg_match_ack	1	I	External Acknowledge for DDMA debug match		1’b0	
cevaxm4_ddma_dbg_match_r	1	O	DDMA debug match indication	1’b0		
cevaxm4_gvi_r	1	O	General Violation Indication	1’b0		
Operation Mode Support (For more details, see the *Operation Modes* section in the *CEVA-XM4 Architecture Specification Volume I* and the *modq Register* section in the *CEVA-XM4 Architecture Specification Volume II*.)						
ext_pv	1	I	External Permission Violation indication, set if violation occurs in the users system		1’b0	
ext_vom	2	I	External Permission Violation operation mode		2’b0	
cevaxm4_seq_om_r	2	O	Operation Mode	2’b0		
cevaxm4_seq_pi_out_r	1	O	Permission Interrupt Output	1’b0		
General						
ceva_free_clk	1	I	Root clock		Mandatory	
ceva_core_rst_n			Asynchronous reset for the Core, active low		Mandatory	

Name	Size	I/O	Description	SAR	Default	Configuration
ceva_sys_rst_n			Asynchronous reset for the Core and MSS, active low		Mandatory	
ceva_global_rst_n	1	I	Asynchronous reset, active low		Mandatory	
ceva_ocem_rst_n	1	I	Asynchronous reset for the OCEM, active low		Mandatory	
Reset Sequence (For more details, see Section 11.)						
div_en	1	I	Slow clock enable for AXI interfaces – EDP, EDAP		1’b1	
ceva_sys_wdog_clk	1	I	System watchdog clock synchronous to root clock		Mandatory	
ceva_epp_wdog_clk	1	I	EPP watchdog clock synchronous to root clock		Mandatory	
ceva_edp_wdog_clk	1	I	EDP watchdog clock synchronous to root clock		Mandatory	
ceva_iop_wdog_clk	1	I	IOP watchdog clock synchronous to root clock		Mandatory	
ceva_amX_wdog_clk	1	I	AXIMX watchdog clock synchronous to root clock		Mandatory	AXImX
Boot Sequence (For more details, see Section 10.)						
boot	1	I	Boot request signal. This signal should only be set during reset. After boot, the core jumps to the address of the vector interrupt.		1’b0	

Name	Size	I/O	Description	SAR	Default	Configuration
External Wait (For more details, see Section 19.)						
external_wait	1	I	External wait request		1'b0	
Program Cache Invalidate						
mcache_invalidate_strap	1	I	Memory Cache Invalidate Strap		1'b0/1'b1	
General Purpose Input/Output (For more details, see Section 13.)						
gp_in	32	I	General Purpose Inputs		32'h0	
cevaxm4_gpout	32	O	General Purpose Outputs	32'h0		
Multicore Status Register Space (For more details, see Section 5.3.)						
core_id	32	I	Core ID identification		32'h0	
cevaxm4_psu_core_wait_r	1	O	Wait indication from the core	1'b0		
EDP AXI Capabilities (For more details, see Section 18.2.)						
cevaxm4_psu_mapv_r	1	O	Access protection violation indication	1'b0		
bus_parity	1	I	Select parity configuration for MSS AMBA Bus ECC. Can only be changed when MSS is in reset mode. ● 0x1: Odd Parity ● 0x0: Even Parity		1'b1	Bus ECC

Name	Size	I/O	Description	SAR	Default	Configuration
acu_lock	1	I	DACU and IACU lock indication. When asserted, the DSP and external masters cannot change the configuration of the DACU and IACU.		1’b0	
acu_slv_acc	1	I	When asserted, the external masters only can change the DACU/IACU configuration. When not asserted, the DSP only can change the DACU/IACU configuration (if the DSP is in supervisor mode).		1’b0	

Note: *When there is only one core with no dedicated semaphore hardware, the default value of qman_semaphore_grant should be 16’h1.*

.

3. Internal Memory

The internal memories are embedded in separate levels of hierarchy, as follows:

- **cevaxm4_dmem.v** is the data memories module.
- **cevaxm4_pmem.v** is the program memories module.

The internal memory's simulative modules are used in the simulation environment, as described in the *CEVA-XM4 Simulation Reference Guide*.

The internal memory's synthesizable modules are used in the backend flow, as described in the *CEVA-XM4 Backend Flow Reference Guide*.

The IP integrator should integrate their own memories into the design instead of the existing memories, according to their vendor's technology, and then modify the interfaces to the memories as appropriate.

3.1 Internal Program Memory

3.1.1 Internal Program Memory block0

The internal program TCM memory contains a single block (**block0**). The block is divided into eight banks of 32-bit width each. The depth of the banks is dependent on the block0 size.

Figure 3-1 shows the internal block0 program memory partition in default mode.

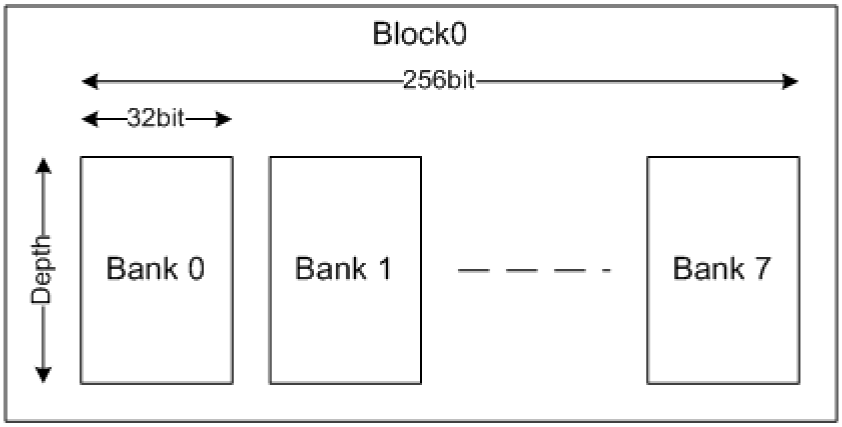


Figure 3-1: Internal Program Memory block0 Default Partition

Note: *The division into eight banks of 32 bits is a recommendation. The memory structure can be changed if another structure is preferred.*

The size of block0 is configurable, as listed in Table 3-1.

Table 3-1: Internal Program TCM block0 Hardware Configurations

Program TCM Bank Depth	Size Option				
	0 KB	32 KB	64 KB	128 KB	256 KB
One block	None	1K	2K	4K	8K

3.1.2 Internal Program Cache Memory

The internal program cache memory consists of Set memory and TAG memory.

3.1.2.1 Internal Program Cache Set Memory

The program cache Set memory consists of four identical blocks for **Way0**, **Way1**, **Way2**, and **Way3**. Each **WayX** block is divided into eight banks of 32 bits. Table 3-2 describes the configurations.

Table 3-2: Internal Program Cache Set Memory Hardware Configurations

Program Cache Set	Size Option		
	32 KB	64 KB	128 KB
Bank depth	0.25K	0.5K	1k

Figure 3-2 shows the internal program cache Set memory partition in the four-way configuration.

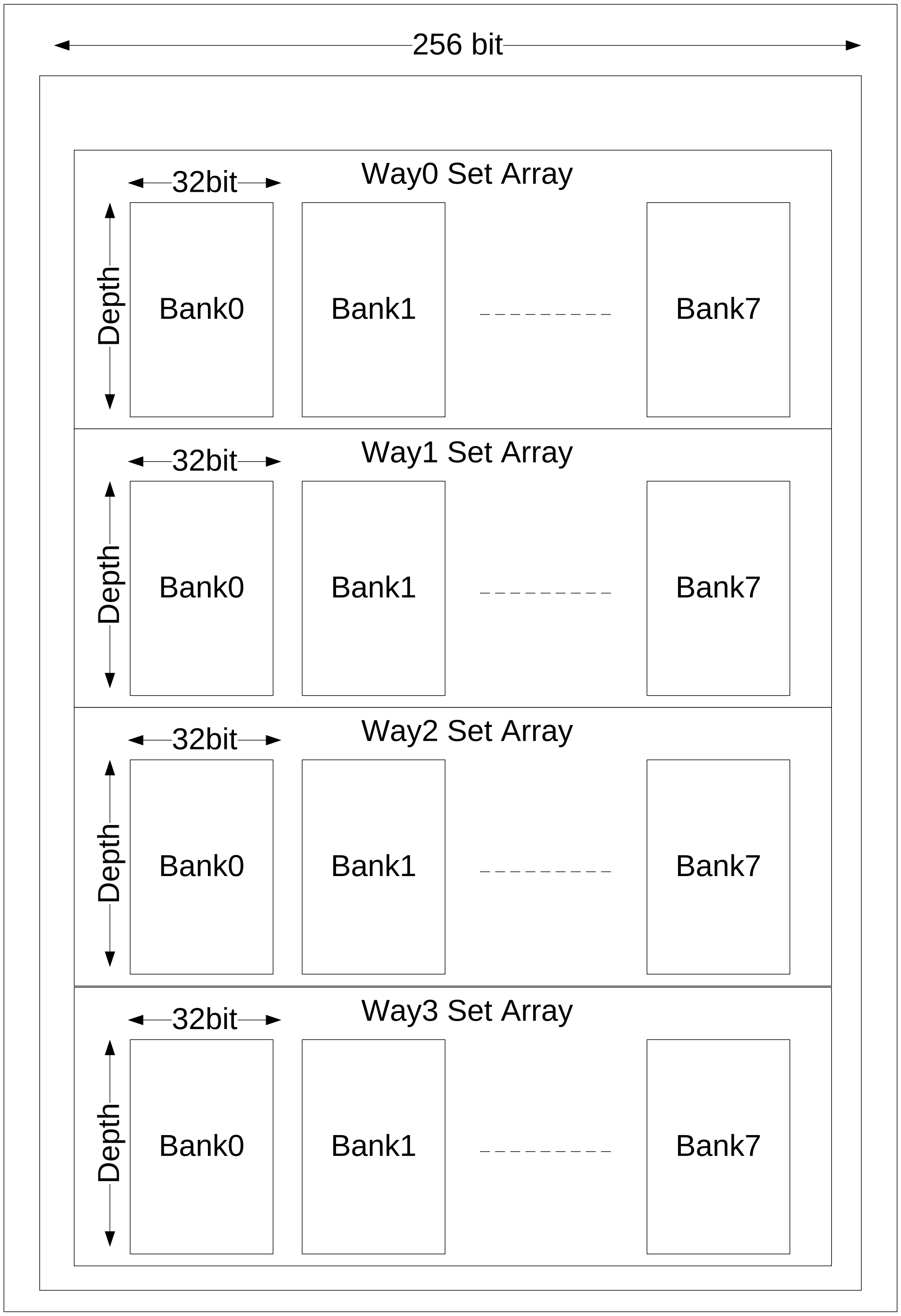

256 bit
Way0 Set Array
32bit
Depth
Bank0
Bank1
Bank7
Way1 Set Array
32bit
Depth
Bank0
Bank1
Bank7
Way2 Set Array
32bit
Depth
Bank0
Bank1
Bank7
Way3 Set Array
32bit
Depth
Bank0
Bank1
Bank7

Figure 3-2: Internal Program Cache Set Memory Partition (One-Block Configuration)

3.1.2.2 Internal Program Cache TAG Memory

The program cache TAG memory consists of four identical memory banks for **Way0**, **Way1**, **Way2**, and **Way3**. Table 3-3 describes the configurations.

Note: *The division into eight banks of 32 bits is a recommendation. The memory structure can be changed if another structure is preferred.*

Table 3-3: Internal Program Cache TAG Memory Hardware Configurations

Program Cache Tag	**Size Option**		
	32 KB	**64 KB**	**128 KB**
Bank width	21 bits	20 bits	19 bits
Bank depth for 64-byte cache block size	128	256	512

Figure 3-3 shows the internal program cache TAG memory partition.

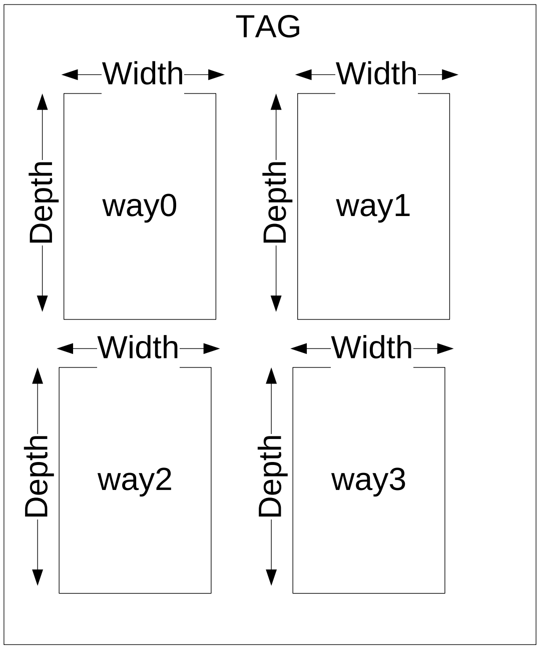


Figure 3-3: Internal Program Cache TAG Memory Partition

3.1.3 Internal Program Memory Parity Blocks

When the CEVA-XM4 memory ECC configuration is included, the internal program memory is extended with parity memories, as follows:

- For the block0 program (PTCM) and program cache Set memories, eight parity bits are added for each 32 bits of code (a total of 56 parity bits are added for each fetch line).
- For the TAG memory, eight parity bits are added for each TAG entry in each Way.

Table 3-4 lists the parity memory sizes for each program memory type.

Table 3-4: Internal Program Parity Memory Sizes

Memory Type	Parity Memory Sizes
block0 parity	56Xdepth
Program cache way X set parity (X = 0,1,2,3)	56Xdepth
Program cache way X tag parity (X = 0,1,2,3)	8Xdepth

Note: *The division into single banks of 56 bits is a recommendation. The memory structure can be changed if another structure is preferred.*

3.2 Internal Data Memory

The internal data TCM memory is configured as four blocks. Each block is divided into 16 banks of 32 bits each (39 bits each when the memory ECC is configured). Table 3-5 describes the configurations.

Table 3-5: Internal Data TCM Hardware Configurations

Data TCM	Size Option		
	128 KB	256 KB	512 KB
Bank depth for 4 blocks	0.5K	1K	2K

Note: *The sizes do not include the addition of the ECC redundancy.*

Figure 3-4 shows the internal data memory partition.

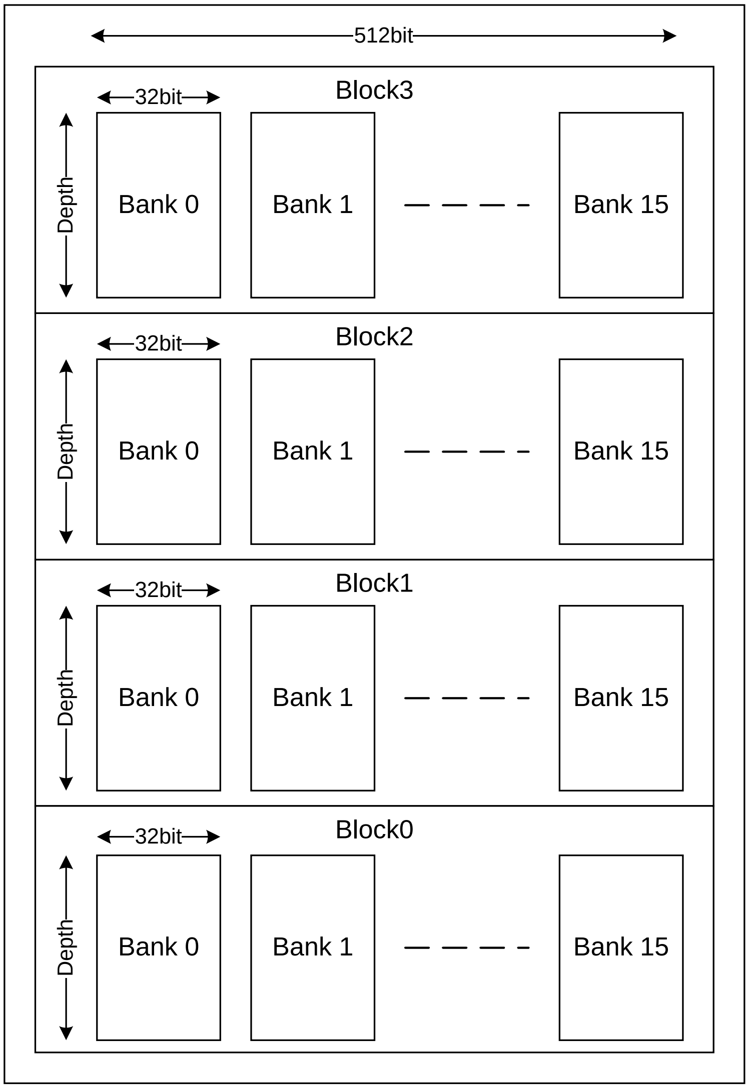

512bit
Block3
32bit
Depth
Bank 0
Bank 1
Bank 15
Block2
32bit
Depth
Bank 0
Bank 1
Bank 15
Block1
32bit
Depth
Bank 0
Bank 1
Bank 15
Block0
32bit
Depth
Bank 0
Bank 1
Bank 15

Figure 3-4: Internal Data Memory Partition – Four-Block Configuration (without Memory ECC)

3.3 Memory Interface Support

The CEVA-XM4 contains various controls for the memories that make it easier to integrate the different memory types.

Table 3-6, Table 3-7, and Table 3-8 describe the internal program memory controls.

Table 3-6: Internal Program TCM block0 Memory Controls in Default Mode

Control	Description	Name in CEVA-XM4
block0 and block0 Parity wr	Write strobe active high	*mp_iarb_b0_wr*
block0 and block0 Parity cs	Chip select active high	*mp_iarb_b0_cs*

Table 3-7: Internal Program TAG Memory Controls

Control	Description	Name in CEVA-XM4
Tag memory and Tag parity wr	Write strobe active high	*mp_iarb_tag_wr*
Tag memory and Tag parity cs	Chip select active high	*mp_iarb_tag_cs*

Table 3-8: Internal Program Set Memory Controls

Control	Description	Name in CEVA-XM4
Set memory and Set Parity wr	Write strobe active high	*mp_iarb_set_wr*
Set memory and Set Parity cs	Chip select active high	*mp_iarb_set_cs*

Table 3-9 describes the internal data memory controls.

Table 3-9: Internal Data TCM Memory Controls

Control	Description	Name in CEVA-XM4
rd	Read strobe active high	*dmss_blk_bnk_rd*
wr	Write strobe active high	*dmss_blk_bnk_wr*
cs	Chip select active high	*dmss_blk_bnk_cs*

3.4 Internal Program Memory Connectivity

Figure 3-5 shows how the internal program memory block0 (PTCM) should be connected for the default configuration. All relevant bus widths are automatically adjusted in the RTL depending on the chosen block0 memory size.

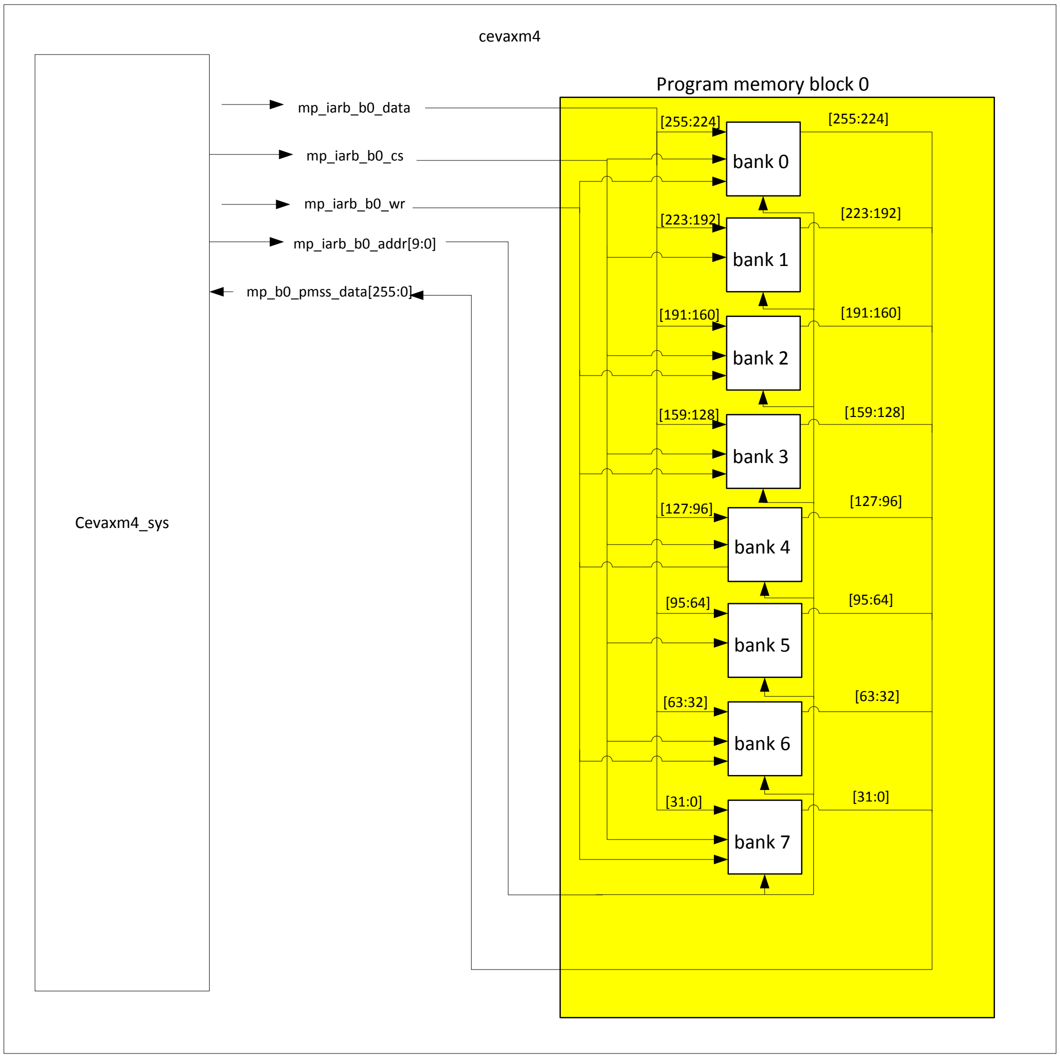

cevaxm4
Cevaxm4_sys
Program memory block 0
mp_iarb_b0_data
mp_iarb_b0_cs
mp_iarb_b0_wr
mp_iarb_b0_addr[9:0]
mp_b0_pmss_data[255:0]
bank 0
bank 1
bank 2
bank 3
bank 4
bank 5
bank 6
bank 7
[255:224]
[223:192]
[191:160]
[159:128]
[127:96]
[95:64]
[63:32]
[31:0]

Figure 3-5: Internal Program block0 Memory (Default Connectivity)

Figure 3-6 shows how the internal program TAG memory should be connected for the default configuration. All relevant bus widths are automatically adjusted in the RTL depending on the chosen program cache memory size.

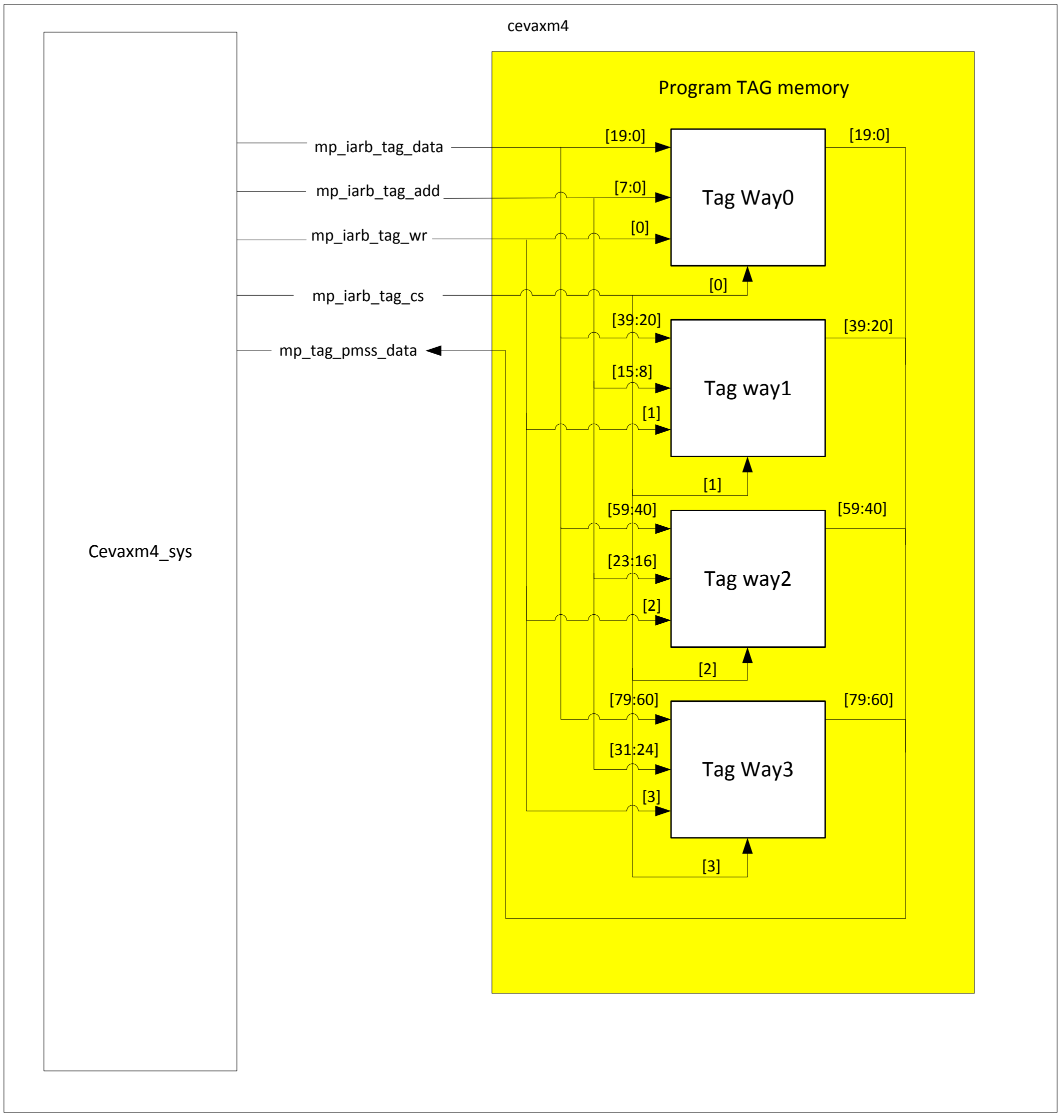

cevaxm4
Cevaxm4_sys
Program TAG memory
mp_iarb_tag_data
mp_iarb_tag_add
mp_iarb_tag_wr
mp_iarb_tag_cs
mp_tag_pmss_data
Tag Way0
Tag way1
Tag way2
Tag Way3
[19:0]
[7:0]
[0]
[39:20]
[15:8]
[1]
[59:40]
[23:16]
[2]
[79:60]
[31:24]
[3]

Figure 3-6: Internal Program TAG Memory (Default Connectivity)

Figure 3-7 shows how the internal program Set memory should be connected for the default configuration. All relevant bus widths are automatically adjusted in the RTL depending on the chosen program cache memory size.

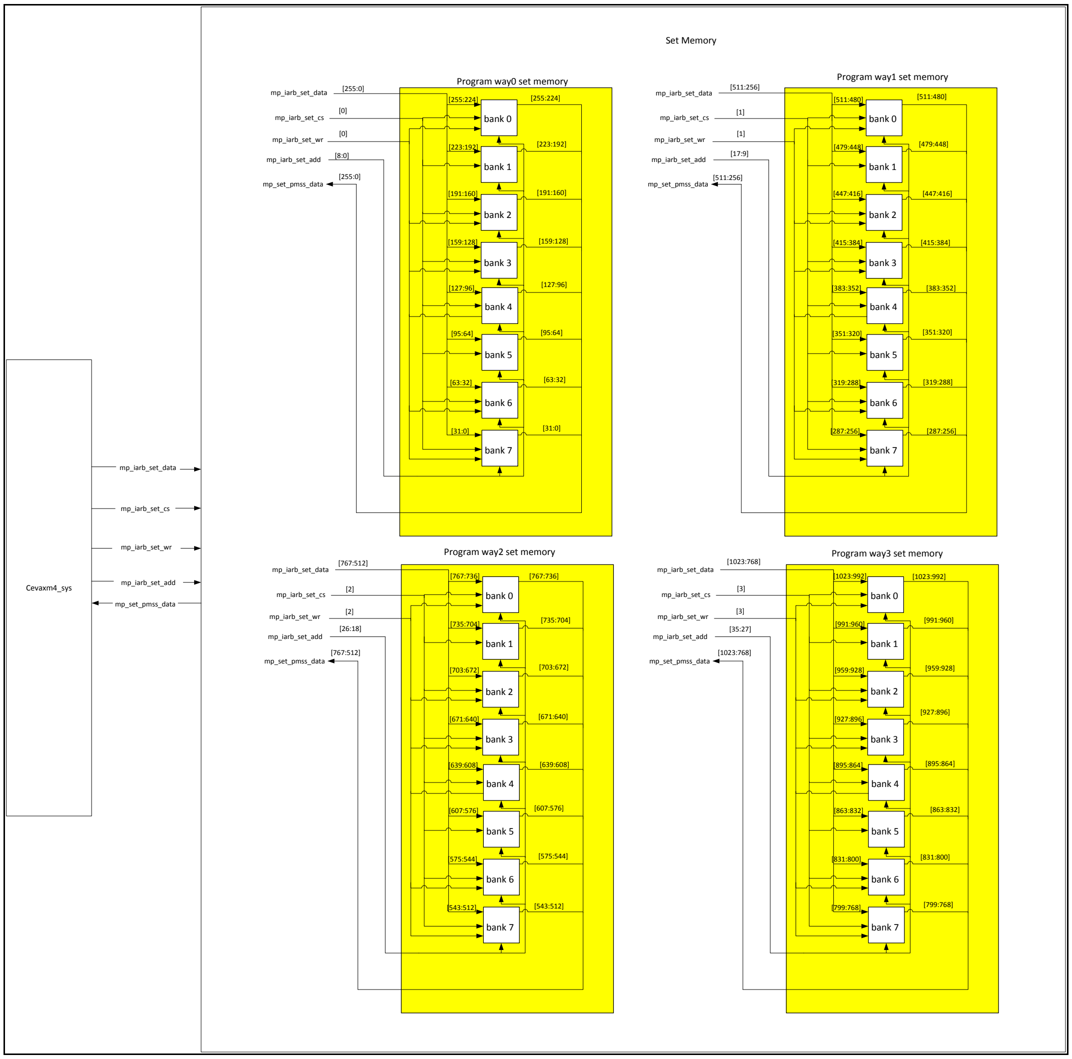


Figure 3-7: Internal Program Set Memory (Default Connectivity)

Figure 3-8 shows how the internal program parity memory of block0 should be connected for the ECC configuration. The block0 parity address bus width is automatically adjusted in the RTL depending on the chosen block0 memory size.

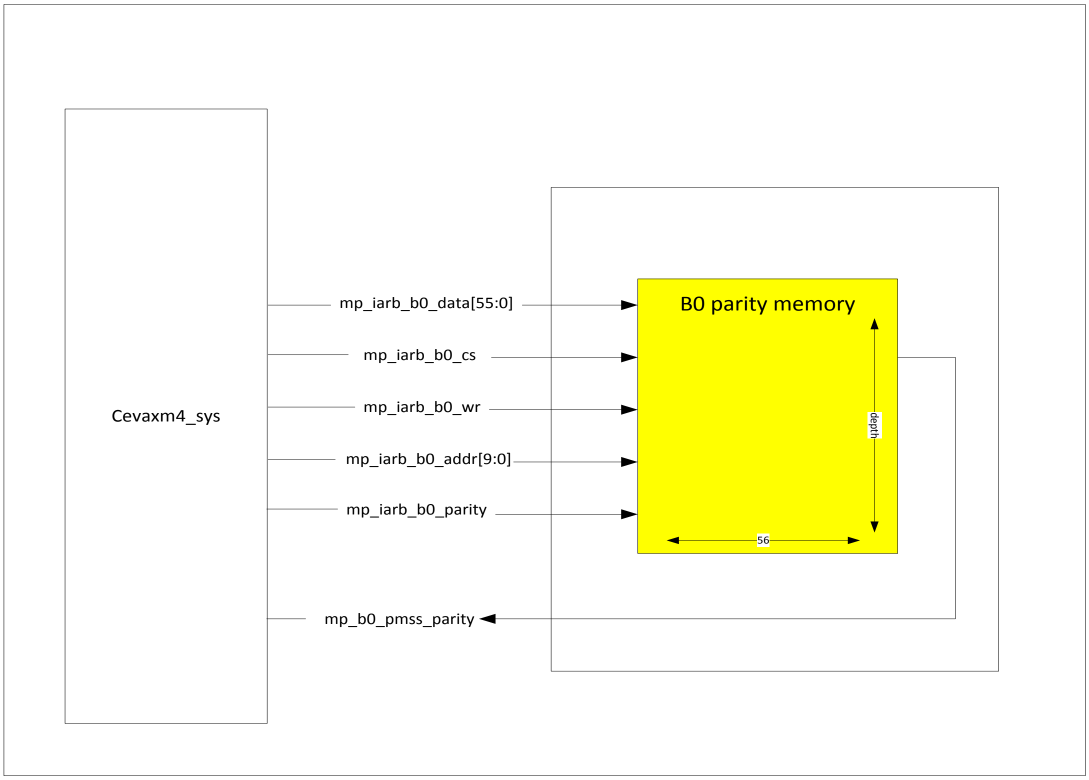

Cevaxm4_sys
mp_iarb_b0_data[55:0]
mp_iarb_b0_cs
mp_iarb_b0_wr
mp_iarb_b0_addr[9:0]
mp_iarb_b0_parity
mp_b0_pmss_parity
B0 parity memory
depth
56

Figure 3-8: Internal Program Memory block0 Parity (ECC Mode)

Figure 3-9 shows how the internal program parity memory of the TAG memory should be connected for the ECC configuration. The address bus width is automatically adjusted in the RTL depending on the chosen program memory size memory size.

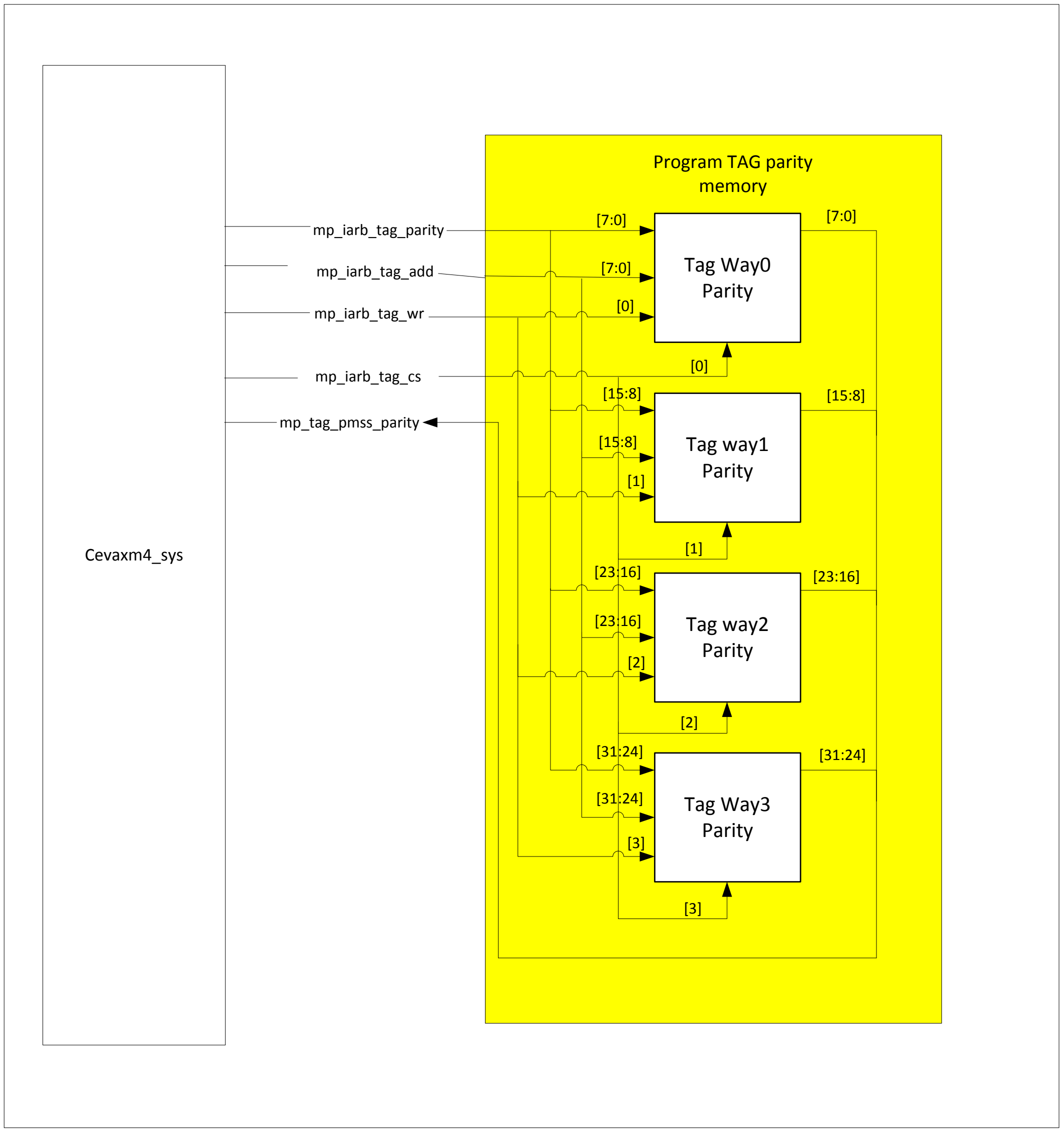

Cevaxm4_sys
mp_iarb_tag_parity
mp_iarb_tag_add
mp_iarb_tag_wr
mp_iarb_tag_cs
mp_tag_pmss_parity
Program TAG parity memory
Tag Way0 Parity
Tag way1 Parity
Tag way2 Parity
Tag Way3 Parity
[7:0]
[7:0]
[0]
[0]
[7:0]
[15:8]
[15:8]
[1]
[1]
[15:8]
[23:16]
[23:16]
[2]
[2]
[23:16]
[31:24]
[31:24]
[3]
[3]
[31:24]

Figure 3-9: Internal Program Memory TAG Parity (ECC Mode)

Figure 3-10 shows how the internal program parity memory of the Set memory should be connected for the ECC configuration. The address bus width is automatically adjusted in the RTL depending on the chosen program memory size memory size.

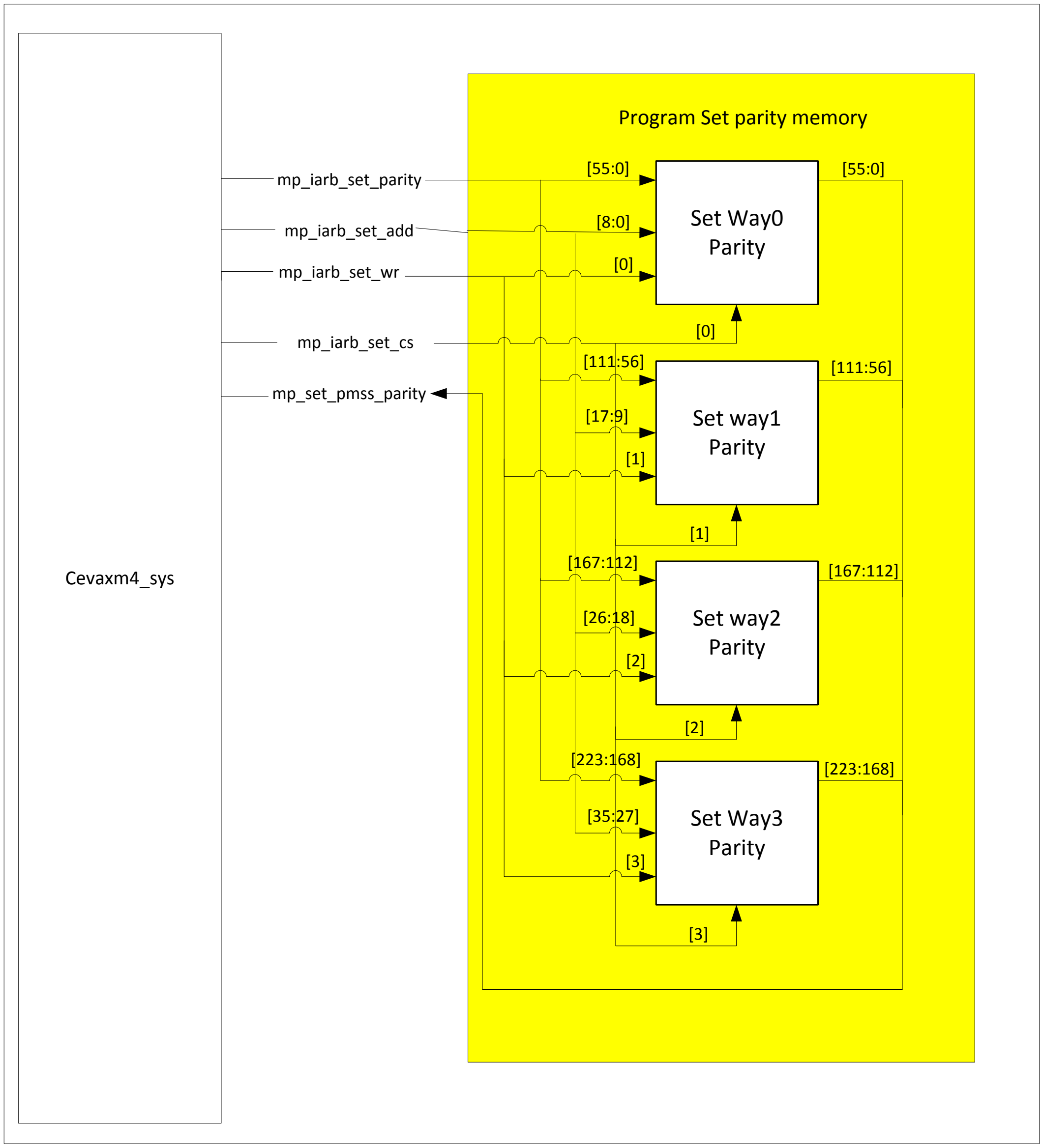
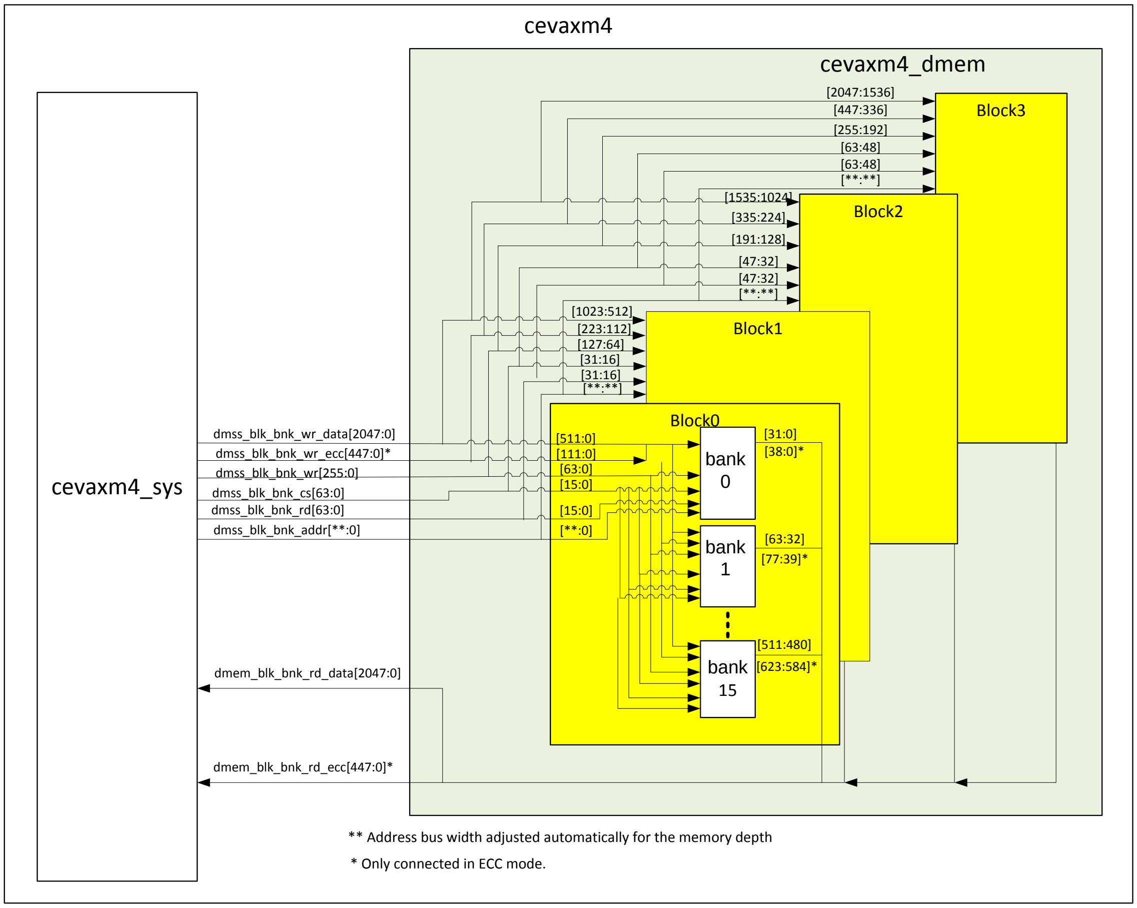

Cevaxm4_sys
Program Set parity memory
mp_iarb_set_parity
mp_iarb_set_add
mp_iarb_set_wr
mp_iarb_set_cs
mp_set_pmss_parity
[55:0]
[8:0]
[0]
Set Way0 Parity
[55:0]
[0]
[111:56]
[17:9]
[1]
Set way1 Parity
[111:56]
[1]
[167:112]
[26:18]
[2]
Set way2 Parity
[167:112]
[2]
[223:168]
[35:27]
[3]
Set Way3 Parity
[223:168]
[3]

Figure 3-10: Internal Program Memory Set Parity (ECC Mode)

3.5 Internal Data Memory Connectivity

Figure 3-11 shows how the internal data memory should be connected. All relevant bus widths are automatically adjusted in the RTL depending on the chosen memory size.


cevaxm4
cevaxm4_dmem
cevaxm4_sys
[2047:1536]
[447:336]
[255:192]
[63:48]
[63:48]
[**:**]
Block3
[1535:1024]
[335:224]
[191:128]
[47:32]
[47:32]
[**:**]
Block2
[1023:512]
[223:112]
[127:64]
[31:16]
[31:16]
[**:**]
Block1
Block0
dmss_blk_bnk_wr_data[2047:0]
dmss_blk_bnk_wr_ecc[447:0]*
dmss_blk_bnk_wr[255:0]
dmss_blk_bnk_cs[63:0]
dmss_blk_bnk_rd[63:0]
dmss_blk_bnk_addr[**:0]
[511:0]
[111:0]
[63:0]
[15:0]
[15:0]
[**:0]
bank 0
[31:0]
[38:0]*
bank 1
[63:32]
[77:39]*
bank 15
[511:480]
[623:584]*
dmem_blk_bnk_rd_data[2047:0]
dmem_blk_bnk_rd_ecc[447:0]*

** Address bus width adjusted automatically for the memory depth

* Only connected in ECC mode.

Figure 3-11: Internal Data Memory Connectivity

3.6 PSU Memory Interface

Both the data and program memories support PSU interfaces to enable control of the memory blocks for the Deep Sleep and Shutdown modes (as described in Section 12).

The Deep Sleep and Shutdown modes (and the PSU memory interface) are available only when the memory power gating option is chosen during installation.

Table 3-10 describes which outputs from the PSU control which memory blocks.

Table 3-10: Internal Memory PSU Controls

P Input Control	Bit	Memory Block
psu_block_deep_sleep_switch_r	[0]	PMEM block0 memories (including parity when configured)
	[1]	PMEM Cache memories (including parities when configured)
	[2]	DMEM block0 memories
	[3]	DMEM block1 memories
	[4]	DMEM block2 memories
	[5]	DMEM block3 memories
psu_domain_shut_down_switch_r	[2]	PMEM block0 memories (including parity when configured)
	[3]	PMEM Cache memories (including parities when configured)
	[4]	DMEM block0 memories
	[5]	DMEM block1 memories
	[6]	DMEM block2 memories
	[7]	DMEM block3 memories

3.7 BIST Support

If the user wishes to integrate a BIST, the BIST wrapper must be added within the **cevaxm4_dmem.v** and **cevaxm4_pmem.v** memory files. These files are in the **<install_dir>/ /design/top/** directory.

The BIST must be integrated with the top-level BIST pins. For convenience purposes:

- *cevaxm4_bist_prog_out* and *bist_prog_in* are connected through the hierarchy to the **cevaxm4_pmem.v** block.
- *cevaxm4_bist_data_out* and *bist_data_in* are connected through the hierarchy to the **cevaxm4_dmem.v** block.

4. Interrupt Handshake

Interrupt indications are synchronized inside the CEVA-XM4. The interrupt should be held high until the acknowledge indication is cleared. The acknowledge indication stays low for XCI_COR+1 cycles.

In the example shown in Figure 4-1, *int0* is set. The interrupt is held high until the *int0* acknowledge indication, *iack0n_r*, is cleared. The value of the **XCI_COR** register is **4**, which means that the stretched acknowledge is low for five cycles.

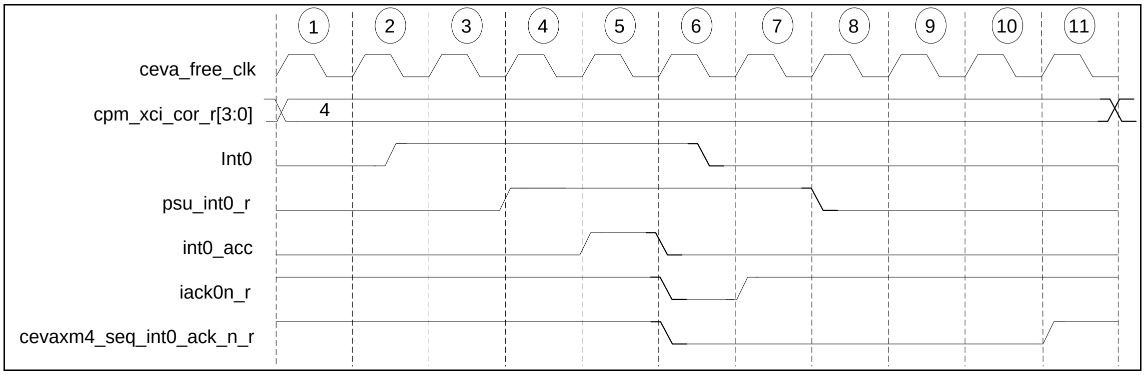


Figure 4-1: Interrupt Handshake Timing Diagram

Table 4-1: Interrupt Handshake Signal Description

Signal Name	Description
ceva_free_clk	CEVA free clock
cpm_xci_cor_r[3:0]	XM4 Interface Configuration Register
int0	Interrupt 0
pmu_int0_r	Sampled int0 inside the PSU
int0_acc	Sampled *int0* inside the core
iack0n_r	Interrupt 0 acknowledge inside the core
cevaxm4_seq_int0_ack_n_r	Interrupt 0 acknowledge output, stretched for XCI_COR+1 cycles

This example is applicable for all interrupts, including external breakpoints (*int0-2*, *nmi*, *vint*, and *ext_bp1/2_req*).

The interrupt is level sensitive. After accepting an interrupt, the core is in a non-interruptible state for five cycles, after which it can accept interrupts again.

The interrupt acknowledge is an active low signal and is de-asserted for XCI_COR+1 cycles (for slower clock domains) when the interrupt is accepted (as shown in Figure 4-1). The **XCI_COR** register should not be changed until the interrupt acknowledge is back high.

4.1 VINT

VINT is a vectored interrupt, and the destination address of the interrupt is taken from a core input bus. This means that multiple interrupts can be serviced by the same interrupt pin.

The VINT and VECTOR inputs should not be changed until the vector interrupt is acknowledged.

Note: *CORE_RCVR behavior is the same as VINT when it is accepted.*

In the example shown in Figure 4-2, the transaction sequence is as follows:

1. **Cycle 1**: *vint* rises.
2. **Cycle 5**: *vint* is accepted by the core *cevaxm4_seq_vint_ack_n_r*.

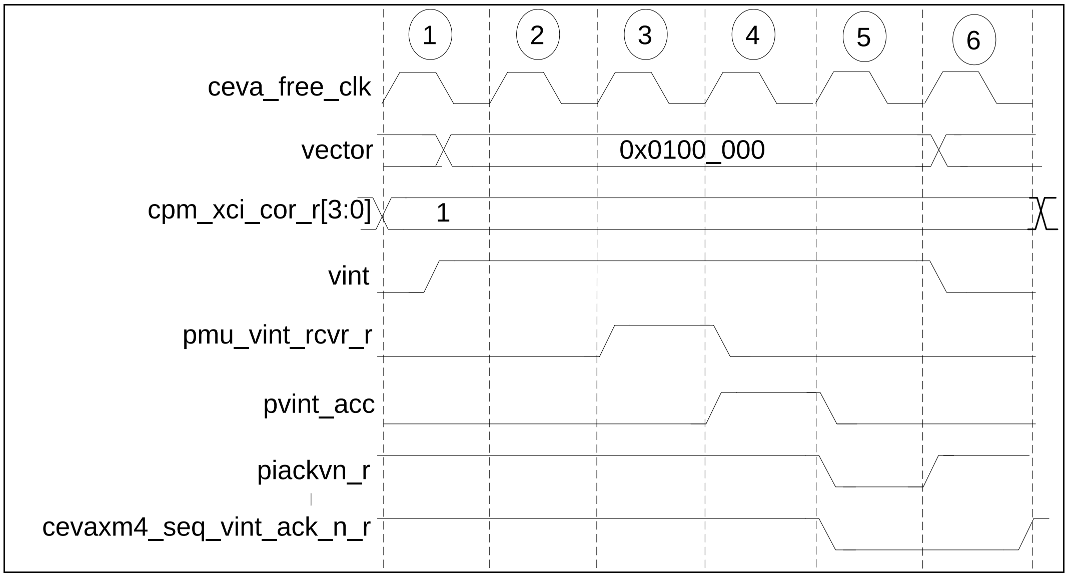


Figure 4-2: VINT Timing Diagram

Table 4-2: VINT Signal Description

Signal Name	Description
ceva_free_clk	CEVA free clock
cpm_xci_cor_r[3:0]	XM4 Interface Configuration Register
vint	Vector interrupt input
vector	Vector address
pmu_vint_rcvr_r	Sample Vector interrupt input by PSU
pvint_acc	Vector interrupt input accepted by the core
cevaxm4_seq_vint_ack_n_r	Sequencer VINT acknowledge output, stretched for XCI_COR+1 cycles

5. Multi-Core Support

The following mechanisms are supported for synchronizing multicore systems:

- External shared memory monitoring via the AXI monitor mechanism, as described in Section 5.1
- Messaging interface for passing commands, as described Section 5.2
- Multicore status register space, as described in Section 5.3
- Internal memory access snooping, as described in Section 5.4

5.1 AXI Monitor Mechanism

The external memory synchronization is achieved using exclusive accesses on the AXI bus. For a full description of the requirements and restrictions associated with AXI Exclusive Accesses, see the *AMBA® AXI Protocol*.

The basic process for this is as follows:

1. The core performs an exclusive read from an address location.
2. If the AXI slave supports exclusive access, then the AXI slave is returned with EXOKAY, as shown in Figure 5-1.

   If the AXI slave returns an OKAY response, then exclusive access is not supported. In this case, the GVI and ER_EXOK bits are set in the **DBG_GEN** register, as shown in Figure 5-2.
3. At some later time, the core attempts to complete the exclusive operation by performing an exclusive write to the same address location.

Note: *AXI requires that the address and size of the exclusive read and write must be exactly the same. The EDP chooses the same ID (LS0 ID) for both.*

4. If no other master has written to that location between the read and write accesses, then the AXI slave returns with EXOKAY and the memory location is updated. The MS bit in the **MODQ** register is cleared, as shown in Figure 5-3.

   If another master has written to that memory location between the read and write accesses, then the address location is not updated and the AXI Slave returns an OKAY response. The MS bit in the **MODQ** is set, as shown in Figure 5-4.

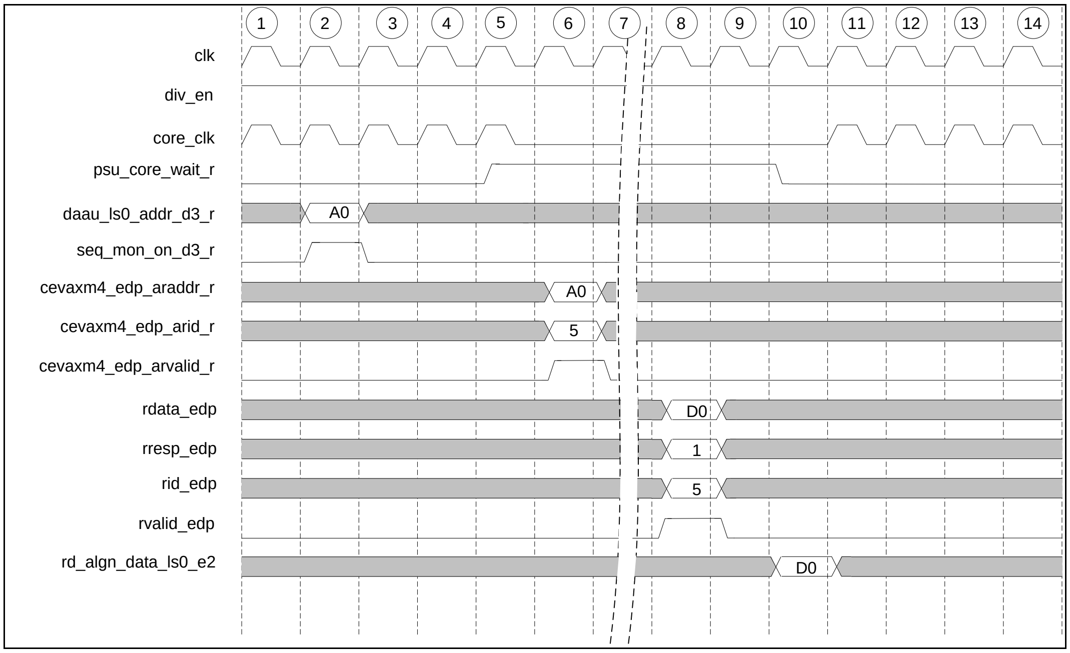

1
2
3
4
5
6
7
8
9
10
11
12
13
14
clk
div_en
core_clk
psu_core_wait_r
daau_ls0_addr_d3_r
A0
seq_mon_on_d3_r
cevaxm4_edp_araddr_r
A0
cevaxm4_edp_arid_r
5
cevaxm4_edp_arvalid_r
rdata_edp
D0
rresp_edp
1
rid_edp
5
rvalid_edp
rd_algn_data_ls0_e2
D0

Figure 5-1: Exclusive Read Succeeded (EXOK Response) Timing Diagram

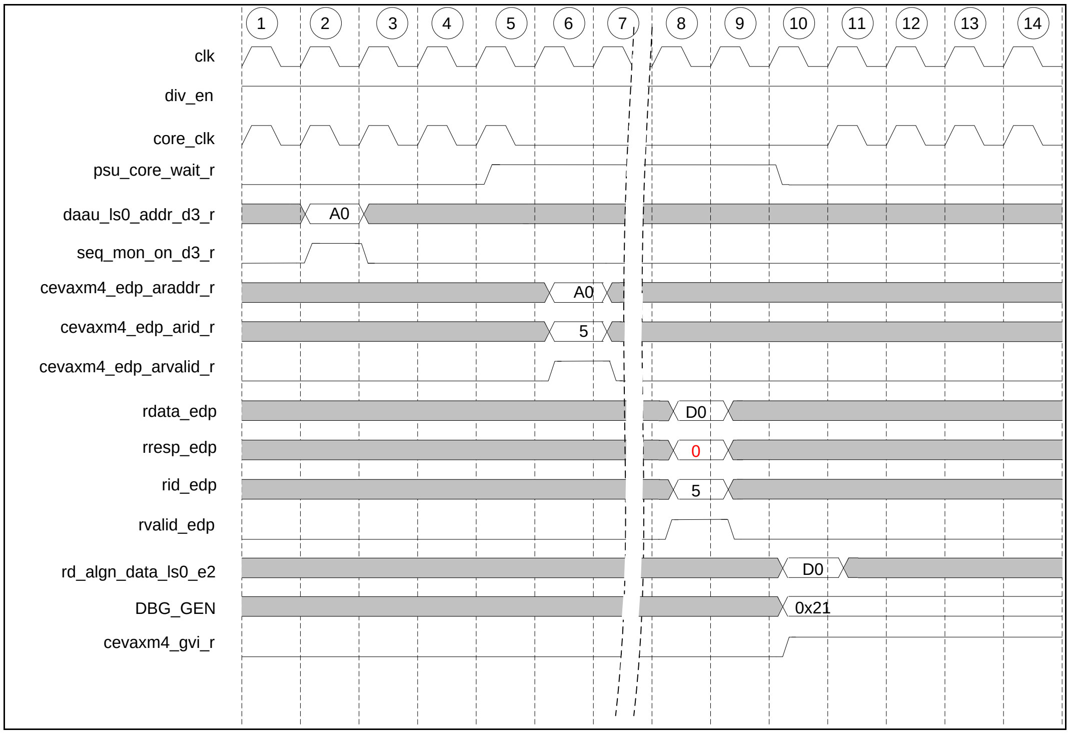

1
2
3
4
5
6
7
8
9
10
11
12
13
14
clk
div_en
core_clk
psu_core_wait_r
daau_ls0_addr_d3_r
A0
seq_mon_on_d3_r
cevaxm4_edp_araddr_r
A0
cevaxm4_edp_arid_r
5
cevaxm4_edp_arvalid_r
rdata_edp
D0
rresp_edp
0
rid_edp
5
rvalid_edp
rd_algn_data_ls0_e2
D0
DBG_GEN
0x21
cevaxm4_gvi_r

Figure 5-2: Exclusive Read Failed (OK Response) Timing Diagram

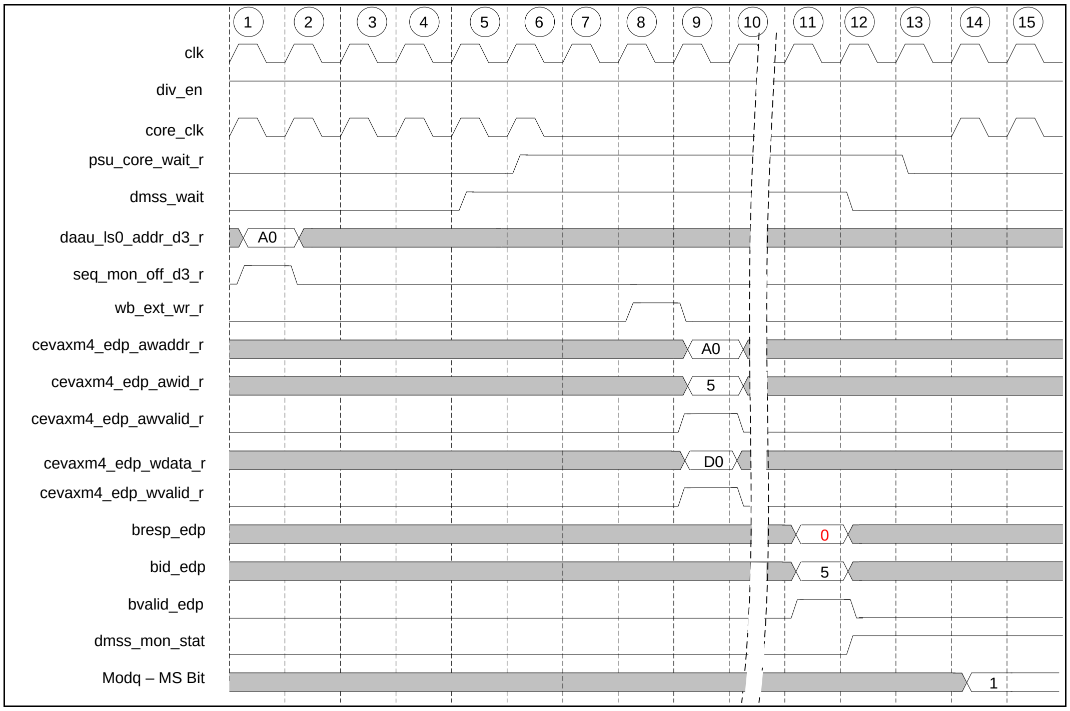

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
clk
div_en
core_clk
psu_core_wait_r
dmss_wait
daau_ls0_addr_d3_r
A0
seq_mon_off_d3_r
wb_ext_wr_r
cevaxm4_edp_awaddr_r
A0
cevaxm4_edp_awid_r
5
cevaxm4_edp_awvalid_r
cevaxm4_edp_wdata_r
D0
cevaxm4_edp_wvalid_r
bresp_edp
0
bid_edp
5
bvalid_edp
dmss_mon_stat
Modq – MS Bit
1

Figure 5-3: Exclusive Write Succeeded (EXOK Response) Timing Diagram

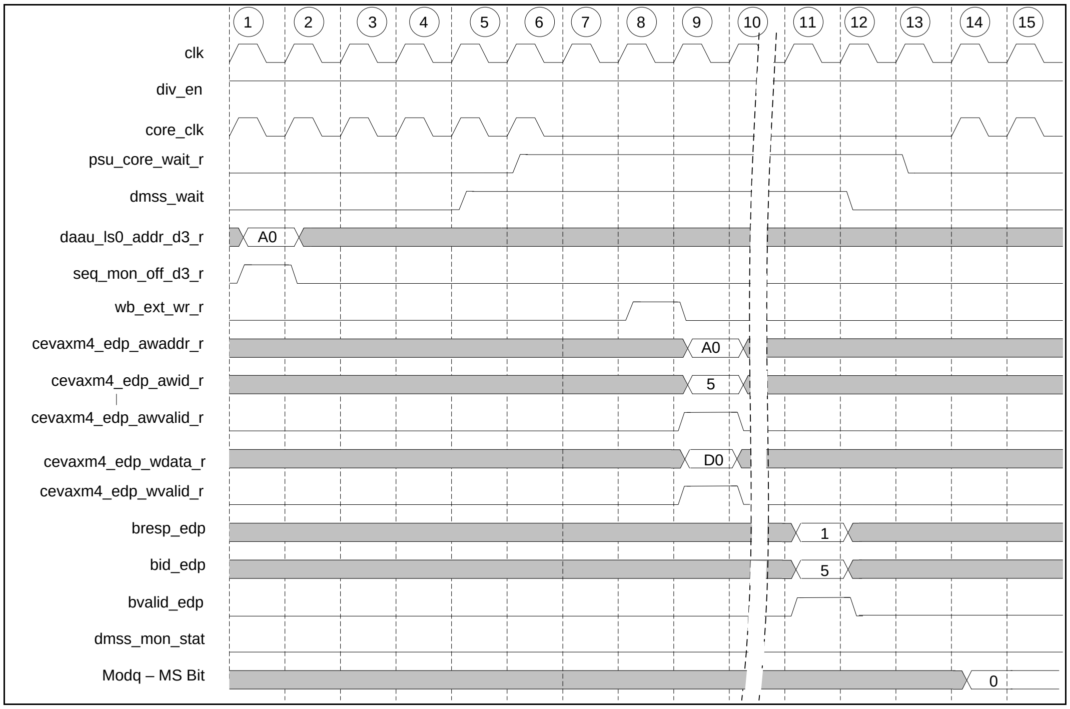

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
clk
div_en
core_clk
psu_core_wait_r
dmss_wait
daau_ls0_addr_d3_r
A0
seq_mon_off_d3_r
wb_ext_wr_r
cevaxm4_edp_awaddr_r
A0
cevaxm4_edp_awid_r
5
cevaxm4_edp_awvalid_r
cevaxm4_edp_wdata_r
D0
cevaxm4_edp_wvalid_r
bresp_edp
1
bid_edp
5
bvalid_edp
dmss_mon_stat
Modq – MS Bit
0

Figure 5-4: Exclusive Write Failed (OK Response) Timing Diagram

5.2 Multicore Command Interface

The Multicore Command Interface (MCCI) is a messaging channel for a multicore environment. The interface consists of 32 command registers, write status and interrupt enable registers, and core status registers.

A message is passed between cores using the command registers. These command registers can be written by an external core via the AXI slave port (EDAP or AXIsX) and read by the host core. The *cevaxm4_mcci_mes_int* output can be asserted when a command register is written by an external core. Read indications for each command register indicate to the external core when a command has been read.

5.2.1 MCCI Single dword

Figure 5-5 shows how a single dword should be passed using the command interface. For safe behavior, the user should first clear the status bit and then read the command register. In this example, the AXI clock is slower than the core clock.

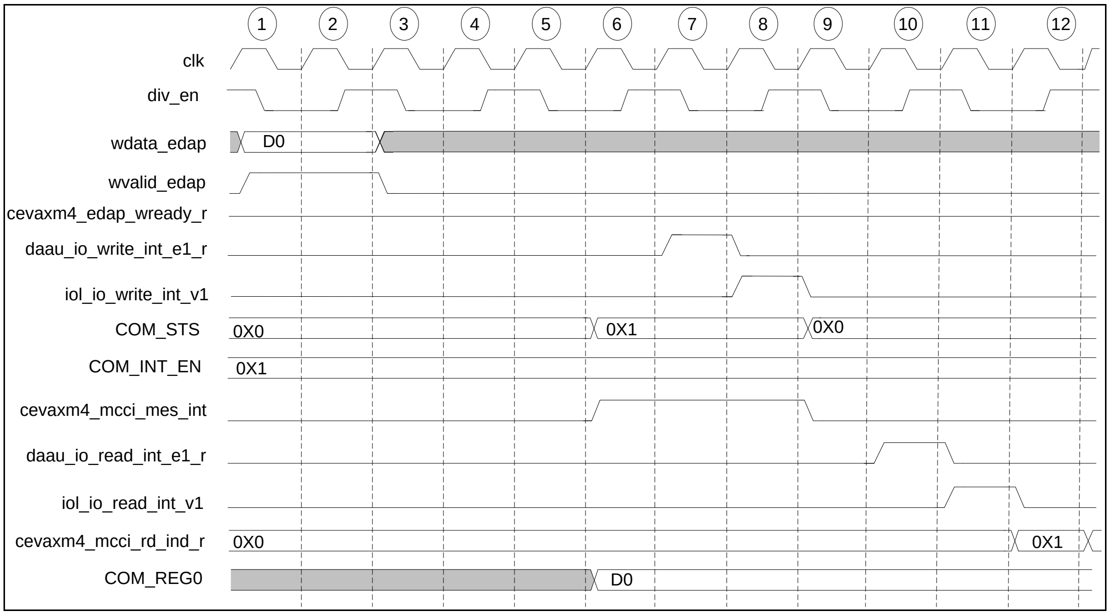

1
2
3
4
5
6
7
8
9
10
11
12
clk
div_en
wdata_edap
D0
wvalid_edap
cevaxm4_edap_wready_r
daau_io_write_int_e1_r
iol_io_write_int_v1
COM_STS
0X0
0X1
0X0
COM_INT_EN
0X1
cevaxm4_mcci_mes_int
daau_io_read_int_e1_r
iol_io_read_int_v1
cevaxm4_mcci_rd_ind_r
0X0
0X1
COM_REG0
D0

Figure 5-5: EDAP MCCI Single dword Write Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1-2**: AXI dword access on the data channel.
2. **Cycle 6**: **COM_REG0** updated with new data D0.
3. **Cycles 6-8**: **COM_STS**[0] = 1’b1 indicates a new command in **COM_REG0**.
4. **Cycles 6-8**: MCCI message interrupt active when both **COM_STS**[0] and **COM_INT_EN**[0] are set.

5. **Cycle 7**: IO writes 32'h1 to COM_STS clears **COM_STS**[0] and message interrupt de-asserts.
6. **Cycle 10**: IO reads **COM_REG0**.
7. **Cycle 12**: *cevaxm4_mcci_rd_ind_r*[0] is set for one clock cycle, as defined in **XCI_COR** (CPM address 0xC60).

Table 5-1: MCCI Single dword Signal Description

Name	Description
clk	Free clock
div_en	DSP clock
wdata_edap	AXI Host WDC
wvalid_edap	AXI Host WDC valid
cevaxm4_edap_wready_r	AXI slave (EDAP) data channel ready
daau_io_write_int_e1_r	Core IO *e1_r* write indication
iol_io_write_int_v1	Internal v1 write
COM_STS	MCCI COM REG status register
COM_INT_EN	MCCI interrupt enable register
cevaxm4_mcci_mss_int	MCCI interrupt
daau_io_read_int_e1_r	Core IO *e1_r* read indication
iol_io_read_int_v1	Internal *v1* read
cevaxm4_mcci_rd_ind_r	Core COM_REG read indication outputs
COM_REG0	Command register 0

5.2.2 MCCI SLVERR

An error response is returned on the EDAP or AXIsX AXI bus by the MCCI for the following access types:

- A transaction is either not aligned or has an unsupported width.
- A burst transaction that is aligned but with an address that extends over the MCCI address range.

Note: *Command registers are updated for this class of error response.*

5.3 Multi-Core Status Register Space

The CEVA-XM4 has a Multi-Core status register space that holds the core identification status in the following registers:

- **CORE_VERSION**: Holds the value of the DSP core type and the DSP RTL version (read-only). It can be read by the following:
    - EDAP/AXIsX slaves at address 0x400174
    - IN instruction or external master read as part of the OCEM programming model at address 0x174
    - OCEM scan chain number 0x72
- CORE_ID: Holds the value of the DSP core ID that is directly connected to the core_id[31:0] input and is sampled once as the reset is de-activated (read-only). It can be read by the following:
    - EDAP/AXIsX slaves at address 0x400178
    - IN instruction or external master as part of the OCEM programming model at address 0x178
    - OCEM scan chain number 0x78

Table 5-2 describes the CORE_VERSION and CORE_ID address mapping.

Table 5-2: CORE_VERSION and CORE_ID Address Mapping

	EDAP/AXIsX Slave Address	OCEM Chain Number	I/O Address
CORE_VERSION	x40_0174	x72	x174
CORE_ID	x40_0178	x78	x178

When reading of these registers is done via the EDAP/AXIsX slave, read access width can be only 32 bits wide and aligned accordingly. A SLVERR response is returned for accesses that are non-aligned or have an unsupported width.

Figure 5-6 shows the EDAP read of the status register space with unequal clocks.

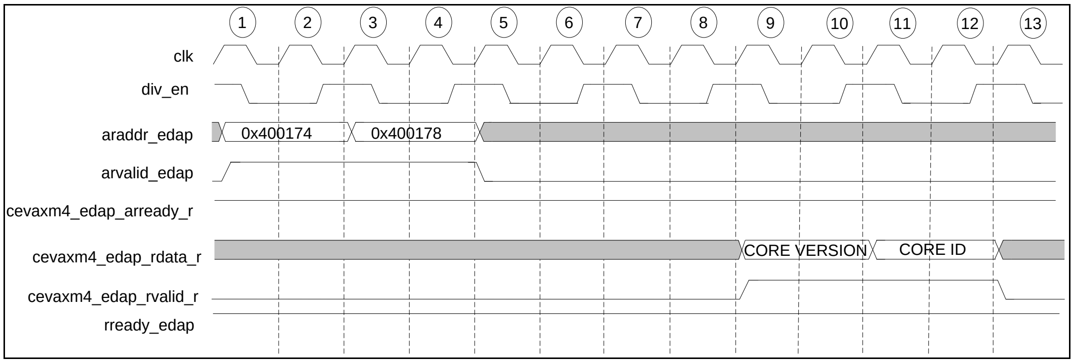

1
2
3
4
5
6
7
8
9
10
11
12
13
clk
div_en
araddr_edap
0x400174
0x400178
arvalid_edap
cevaxm4_edap_arready_r
cevaxm4_edap_rdata_r
CORE VERSION
CORE ID
cevaxm4_edap_rvalid_r
rready_edap

Figure 5-6: EDAP Read with Unequal Clocks Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: External master issues read of core version register.
2. **Cycle 3**: External master issues read of core ID register.
3. **Cycle 9**: Core version is presented to the AXI Master on the RDC.
4. **Cycle 11**: Core ID is presented to the AXI Master on the RDC.

Table 5-3: EDAP Read with Unequal Clocks Signal Description

Name	Description
clk	Free clock
div_en	AXI slave port clock enable
araddr_edap	AXI Master read address bus
arvalid_edap	AXI Master read address valid
cevaxm4_edap_arready_r	AXI slave read address ready
cevaxm4_edap_rdata_r	AXI slave read data bus
cevaxm4_edap_rvalid_r	AXI slave read data valid
rready_edap	AXI Master ready

5.4 Internal TCM Snooping

The snooping mechanism can be set up to alert the core when an external host accesses a region of internal data TCM. The mechanism consists of a base address register, a top address register, and a control and status register. When the control register is configured, an interrupt can be raised on any AXI slave (EDAP/AXIsX) read- or write-granted RRA arbitration at an address that is within the address held in the base and end address registers.

5.4.1 Snoop Read Transaction

Figure 5-7 shows the behavior of the snoop mechanism during an EDAP burst read. In this example, the address range a0+2 to a0+3 falls within the snoop address range.

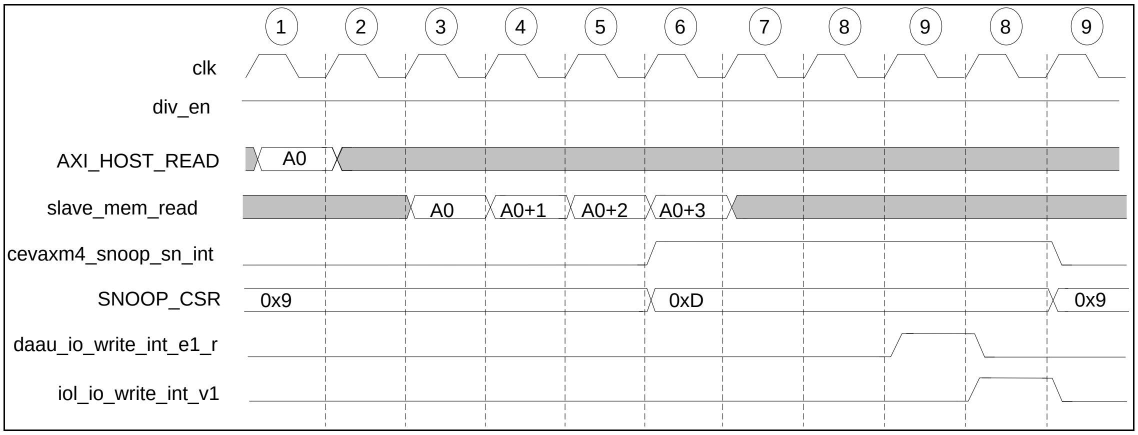


Figure 5-7: Snoop Detection during EDAP Read Transaction Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: **SNOOP_CSR** (CPM address 0x0098) is set up for EDAP Snoop on read and interrupt enable.
2. **Cycle 1**: Host AXI issues TCM read burst of four beats.
3. **Cycles 3-6**: EDAP read access requests RRA.
4. **Cycles 5-6**: Address A0+2 and A0+3 within snoop base and end address.
5. **Cycle 6**: Snoop interrupt is asserted, and **SNOOP_CSR** [2] (CPM address 0x0098) and **SNOOP_EDAP_STS** are set.
6. **Cycle 9**: I/O writes zero to **SNOOP_CSR** [2] (CPM address 0x0098), clears **SNOOP_EDAP_STS**, and snoop interrupt de-asserts.

Table 5-4: Snoop Read Transaction Signal Description

Name	Description
clk	Free clock
div_en	DSP clock
AXI_HOST_READ	AXI host master read channel
slave_mem_read	EDAP read RRA request
cevaxm4_edap_snoop_sn_int	Snoop interrupt
SNOOP_CSR	Snoop status and control register
daau_io_write_int_e1_r	Core I/O *e1_r* write indication
iol_io_write_int_v1	Internal *v1* write indication

5.4.2 Snoop Write Transaction

Figure 5-8 shows the behavior of the snoop mechanism during an EDAP burst write. In this example, addresses for data D0+2 and D0+3 fall within the snoop address range.

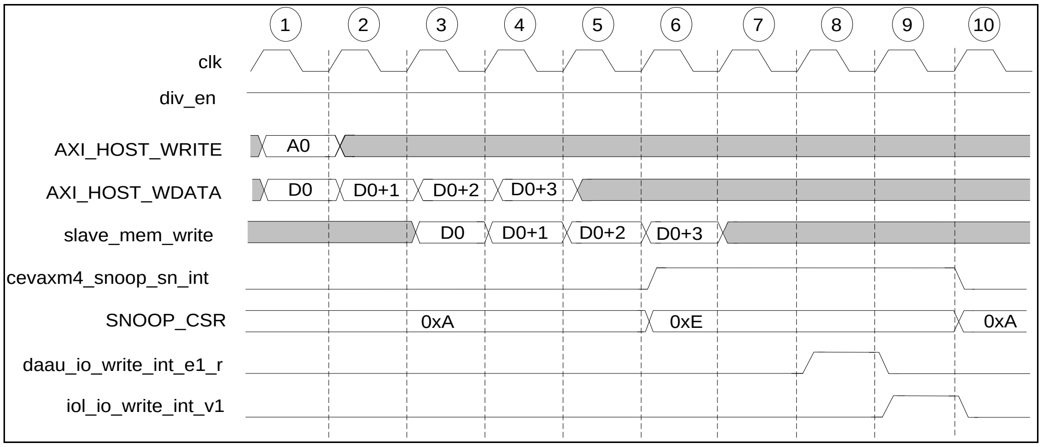

1
2
3
4
5
6
7
8
9
10
clk
div_en
AXI_HOST_WRITE
A0
AXI_HOST_WDATA
D0
D0+1
D0+2
D0+3
slave_mem_write
D0
D0+1
D0+2
D0+3
cevaxm4_snoop_sn_int
SNOOP_CSR
0xA
0xE
0xA
daau_io_write_int_e1_r
iol_io_write_int_v1

Figure 5-8: Snoop Detection during EDAP Write Transaction Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: **SNOOP_CSR** (CPM address 0x0098) is set up for EDAP snoop on write and interrupt enable.
2. **Cycle 1**: Host AXI issues TCM write burst of four accesses.
3. **Cycles 1-4**: Host AXI transfers write data.
4. **Cycles 3-6**: EDAP write access requests RRA.
5. **Cycles 5-6**: Data D2 and D3 within snoop base and end address.

6. **Cycle 6**: Snoop interrupt is asserted, and **SNOOP_EDAP_CSR**[2] (CPM address 0x0098) and **SNOOP_STS** are set.
7. **Cycle 8**: IO writes zero to **SNOOP_CSR** [2] (CPM address 0x0098), clears **SNOOP_EDAP_STS**, and snoop interrupt de-asserts.

Table 5-5: Snoop Write Transaction Signal Description

Name	Description
clk	Free clock
div_en	DSP clock
AXI_HOST_WRITE	AXI host master write address channel
AXI_HOST_WDATA	AXI host master WDC
slave_mem_write	EDAP write access requests RRA
cevaxm4_snoop_sn_int	Snoop interrupt
SNOOP_CSR	Snoop status and control register
daau_io_write_int_e1_r	Core IO *e1_r* write indication
Iol_io_write_int_v1	Internal *v1* write indication

6. DMA Manager Interface

6.1 QMAN Interrupts

The Queue Manager (QMAN) has a dedicated level interrupt signal, *qman_irq*, which is asserted when one of the following occurs:

- If a queue is empty but still has enabled tasks.

  In this case, unless masked, the queue resets the **QX_EN** register and asserts the *qman_irq* signal to the CEVA-XM4 core or the system master. When new tasks are written to the queue, the **QX_EN** register can be set again.
- If two or more tasks are accessing the queue arbitration with same absolute priority and non-consecutive frame number.

  In this case, unless masked, the QMAN asserts the *qman_irq* signal.

6.2 Queue Descriptor Enable Signal

The QMAN enables the user to assign DDMA tasks without checking for the availability of the data at the source and without checking that the destination is capable of receiving the data. The QMAN implements an enabled tasks counter for maintaining the number of DDMA tasks in the queue that are allowed to be executed by the DDMA. The enabled tasks counter can be incremented by an external device using a dedicated input signal *qn_desc_en[X]*. This enables an external device to directly inform the QMAN whether the next tasks (one or more, depending on the application) can be executed without involving the CEVA-XM4 core.

Two enabled tasks counters are implemented at each queue. When both are enabled, one enabled tasks counter delays head tasks until data is available at the source, and the other delays head tasks until the destination memory is free. The CEVA-XM4 QMAN waits until both counters are larger than zero before it sends the head task to the DDMA. To indicate that the next DDMA task in a queue can be executed, the source and the destination counters can be incremented using the **QX_DSC_EN_INC0** or **QX_DSC_EN_INC1** registers, where up to 63 DDMA tasks can be enabled using a single CPM access.

In addition, enabled tasks counter 0 can be incremented using a dedicated input signal *qn_desc_en[X]* (where *X* is replaced with the relevant queue number). Enabled tasks counter 0 is incremented with the value configured in the **QX_EN_INC_VAL0** field in the **QX_DSC_EN_INC0** register for every cycle where *qn_desc_en[X]* is set. This signal can be used to enable DMA tasks written in a queue according to a task end indication from an external TCE or another CEVA-XM4 core.

For more details, see the *Queue Manager* section in the *CEVA-XM4 Architecture Specification.*

6.3 Queue Semaphore

To prevent two or more QMANs from different cores from reading the same task of a shared queue, a hardware semaphore should be implemented by the user. This ensures that only one QMAN is granted access to the queue at a given time.

The QMAN requests ownership of the semaphore before reading the queue read pointer. This is realized by setting the semaphore request (that is, *qman_semaphore_req[X]* for queue *X*) signal.

Unless the requesting QMAN is granted (by assertion of *qman_semaphore_grant[X]* for queue *X*), it will not read the queue read pointer and will keep requesting ownership of the semaphore in the following cycle.

If a queue is not shared, then the user can keep *qman_semaphore_grant[X]* asserted continuously to ensure that the queue read pointer can increment when the queue semaphore request is made.

When the QMAN is granted, the user is responsible for ensuring that no other QMAN is granted until the write access to the read pointer is complete, meaning that the granted QMAN has successfully updated the read pointer.

Figure 6-1 shows a simple implementation of a semaphore with three requesters using a priority encoder. The read pointer sniffer signal (*q0_read_ptr_sniffer*) ensures that the other QMAN is not granted until the previous write access to the read pointer is complete.

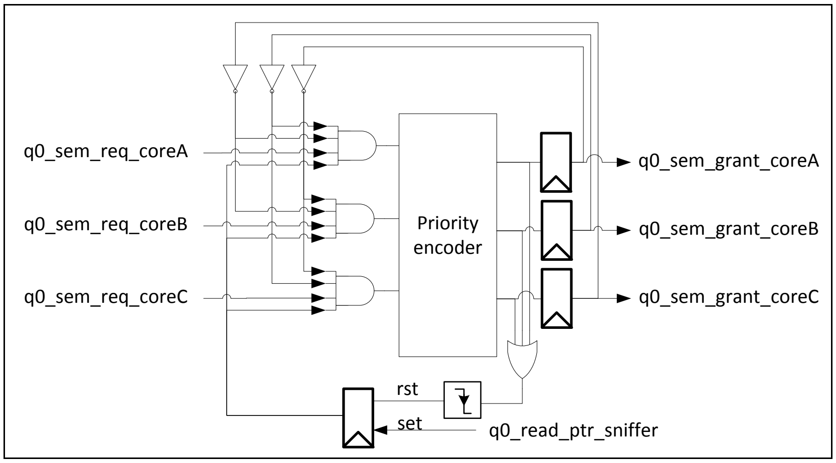

q0_sem_req_coreA
q0_sem_req_coreB
q0_sem_req_coreC
Priority encoder
q0_sem_grant_coreA
q0_sem_grant_coreB
q0_sem_grant_coreC
rst
set
q0_read_ptr_sniffer

Figure 6-1: Queue 0 Semaphore Implementation Example

Figure 6-2 shows the signals that participate in the semaphore handshake.

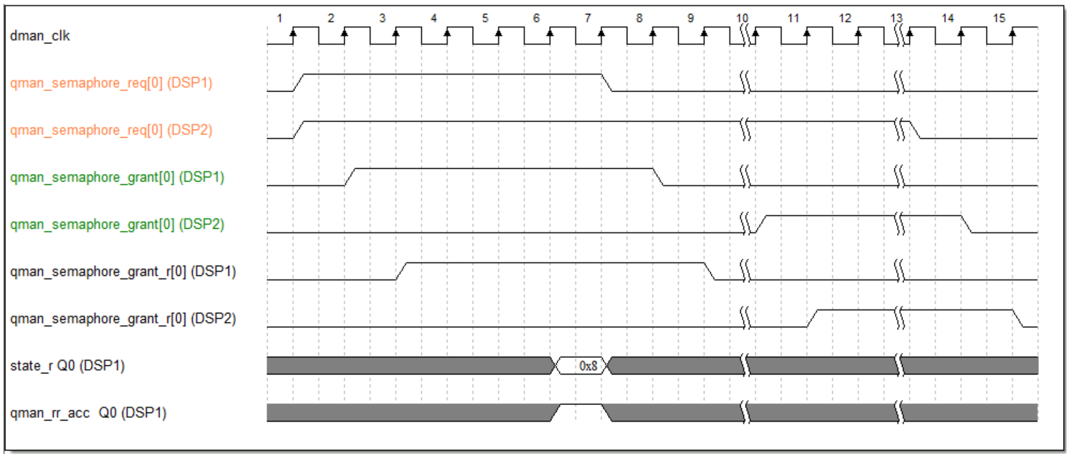


Figure 6-2: Queue Semaphore Handshake Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: Both QMAN1 (DSP1) and QMAN2 (DSP2) request a semaphore, but only QMAN1 receives the grant.
2. **Cycle 2**: Grant is sampled by QMAN1, and QMAN1 begins reading pointer and task.
3. **Cycle 6**: QMAN1 is granted by the QMAN round-robin arbiter to update the read pointer.

4. **Cycle 7**: As a result, QMAN1 de-asserts semaphore request.
5. **Cycle 10**: After making sure the read pointer was updated, the external arbiter grants QMAN2.

Table 6-1: Queue Semaphore Handshake Signal Description

Name	Description
dman_clk	DMAN clock
qman_semaphore_req[0]	Queue 0 semaphore request
qman_semaphore_grant[0]	Semaphore grant to queue 0
state_r	Queue task fetch FSM state
qman_rr_acc	QMAN round-robin arbiter access grant signal

7. Trap Indication

The CEVA-XM4 has a dedicated *cevaxm4_seq_trp_srv_r* indication for *trap* service routines. This indication is set when a *trap* instruction is located in A2 and reset by a *reti* instruction at E2.

Figure 7-1 shows the trap indication operation.

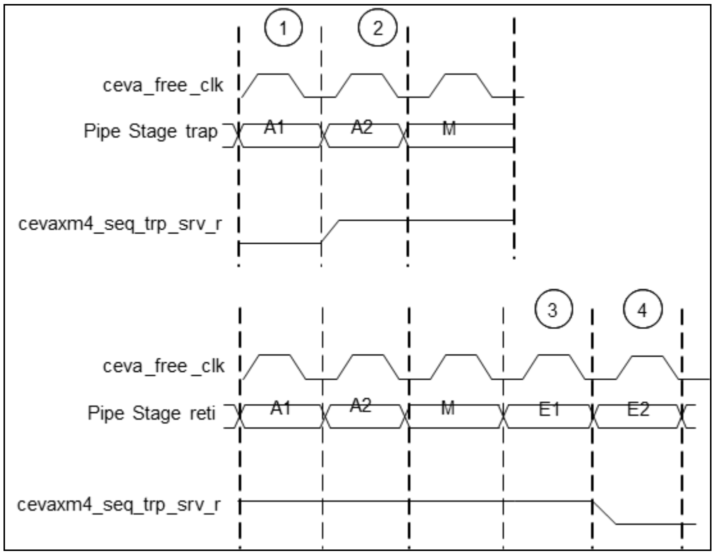

①
②
ceva_free_clk
Pipe Stage trap
A1
A2
M
cevaxm4_seq_trp_srv_r
③
④
ceva_free_clk
Pipe Stage reti
A1
A2
M
E1
E2
cevaxm4_seq_trp_srv_r

Figure 7-1: Trap Indication Routine Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The sequencer detects a *trap* instruction in A1.
2. **Cycle 2**: *cevaxm4_seq_trp_srv_r* is asserted in A2.
3. **Cycle 3**: The sequencer detects a *reti* instruction in E2.
4. **Cycle 4**: *cevaxm4_seq_trp_srv_r* is de-asserted in E2.

8. Verification Indication

The CEVA-XM4 includes dedicated verification indications. Two output signals indicate the end of test status (either passed or failed). To evoke these two signals, the *verifeqs*, *verifeq*, or *verifend* instructions should be used.

- If the *verifeqs* or *verifeq* comparison fails, then *cevaxm4_cverbit_r* is asserted.
- If the test reaches verifend, then *cevaxm4_seq_eotbit_r* is asserted.

For example:

```
M0.verifeq a0, #0x1
```

The *verifeq* instruction compares accumulator a0 to zero, even though the accumulator a0 value is 0x012345678. When the *verifeq* instruction reaches the E4 stage, *cevaxm4_cverbit_r* is asserted (cycle 1), as shown in Figure 8-1.

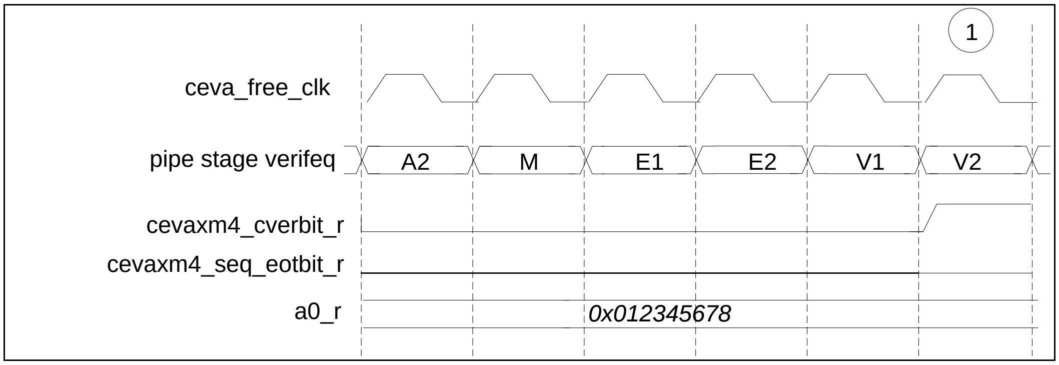

1
ceva_free_clk
pipe stage verifeq
A2
M
E1
E2
V1
V2
cevaxm4_cverbit_r
cevaxm4_seq_eotbit_r
a0_r
0x012345678

Figure 8-1: Error Test Example

For example:

```
SQ.verifend
```

When the *verifend* instruction reaches the E4 stage, *cevaxm4_seq_eotbit_r* is asserted (cycle 1), as shown in Figure 8-2.

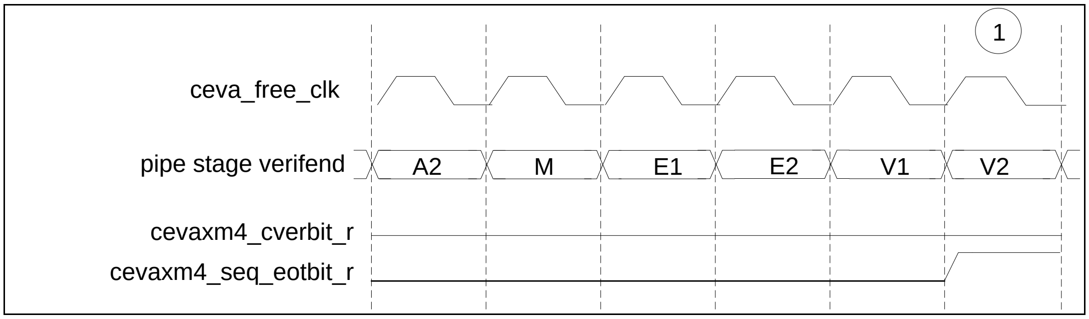

1
ceva_free_clk
pipe stage verifend
A2
M
E1
E2
V1
V2
cevaxm4_cverbit_r
cevaxm4_seq_eotbit_r

Figure 8-2: Ending Test

9. Operation Modes

The CEVA-XM4 includes the following operation modes:

- Supervisor
- USER0
- USER1

In USER0 and USER1 operation modes, permission violation can occur. When an external permission violation indication (*ext_pv*) occurs, it is sampled into the **MODQ** *PV* field, as follows:

- **0** = No violation
- **1** = Violation

This bit is set by the hardware while generating a permission violation. This bit is sticky, which means that it must be explicitly cleared by the software before further permission violations can be generated.

This bit is cleared at core reset, and can be modified by writing to this register using *pop* or *mov* instructions.

External permission violation operation mode (*ext_vom*) is sampled into the **MODQ** Violation Operation Mode (*VOM*) field.

These bits are written by the hardware, and contain the value of the OM bits at the time of the first permission violation. These bits are sticky, which means that they are **not** written to by the hardware if the PV bit is already set.

These bits are cleared at core reset, and can be modified by writing to this register using *pop* or *mov* instructions.

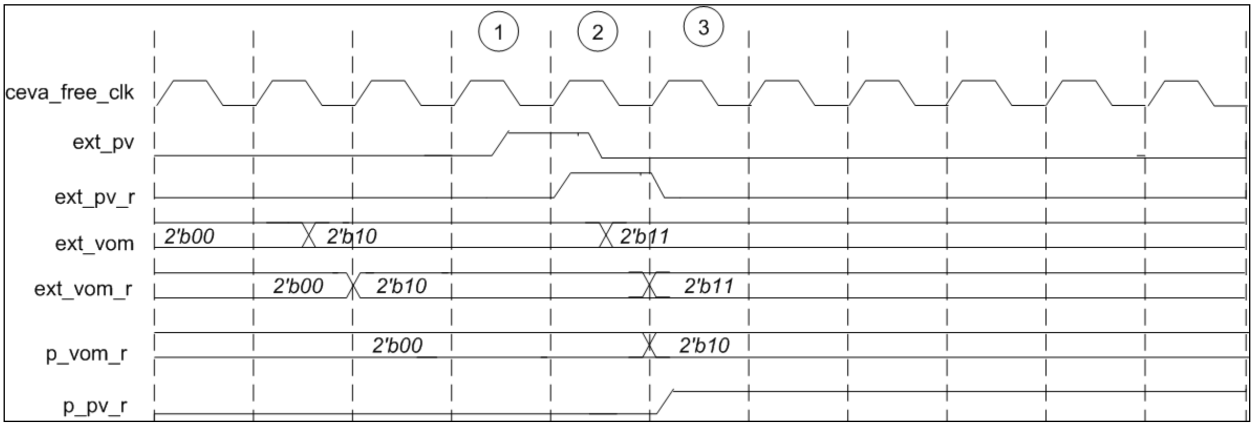


Figure 9-1: External Permission Violation Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: *ext_pv* rises.
2. **Cycle 2**: Sampled by Sequencer *ext_pv_r*.
3. **Cycle 3**: The **MODQ** *PV* and *VOM* fields are changed with *p_pv_r*, *p_vom_r*.

Table 9-1: External Permission Violation Signal Description

Name	Description
ceva_free_clk	CEVA free clock
ext_pv	External PV input
ext_pv_r	Sampled External PV by Sequencer
ext_vom	External VOM input
ext_pv_r	Sampled External VOM by Sequencer
p_vom_r	*VOM* field in **MODQ**
p_pv_r	*PV* field in **MODQ**

The *cevaxm4_seq_om_r*[1:0] output reflects the *OM* field of **MODQ** register.

The *cevaxm4_seq_pi_out_r* permission interrupt output is a sample of permission violation (*ppv_set*). It is sampled only if the **MODQ** *PI* field is set.

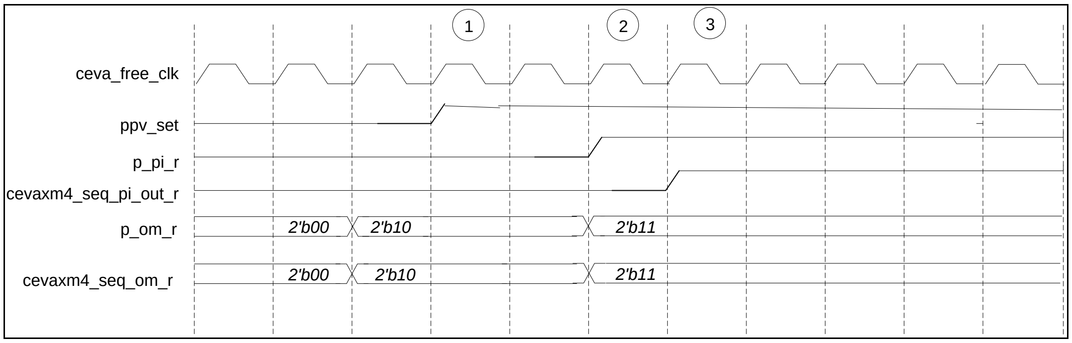

1
2
3
ceva_free_clk
ppv_set
p_pi_r
cevaxm4_seq_pi_out_r
p_om_r
2'b00
2'b10
2'b11
cevaxm4_seq_om_r
2'b00
2'b10
2'b11

Figure 9-2: Operation Mode Indication Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: Permission violation occurs, and *ppv_set* is set.
2. **Cycle 2**: The *p_pi_r PI* field is set and enables the Permission Interrupt output pin.

   cevaxm4_seq_om_r is the reflection of the *OM* operation mode field in **MODQ**.
3. **Cycle 3**: *cevaxm4_seq_pi_out_r* rises.

Table 9-2: Operation Mode Indication Signal Description

Name	Description
ceva_free_clk	CEVA free clock
ppv_set	Permission violation bit set
p_pi_r	Enables or disables the Permission Interrupt output pin
cevaxm4_seq_pi_out_r	Permission Interrupt Output
p_om_r	*OM* field in **MODQ**
cevaxm4_seq_om_r	Outside reflection of *OM* field in **MODQ**

10. Boot Sequence

The following sections describe the CEVA-XM4 boot sequence. The following are the possible booting options:

- Use the boot input signal for booting from an external location (see Section 10.1)
- Boot from PC 0x0 while the program is preloaded before releasing the core's reset signal
- Boot from PC 0x0 while the external master activates the internal program DMA and data DMA to download the code before the core reset is released (see Section 10.3)
- Boot from the program cache memory (the cache must be preloaded with the program before de-asserting the core's reset; see Section 10.4)

10.1 Boot Using the Boot Input

To enable the boot sequence, a boot signal should be set together with the boot vector when the external reset is released (as shown in Figure 10-1). The boot indication and vector should be held for eight cycles (from global reset rise edge) for the PCU to jump to the vector address.

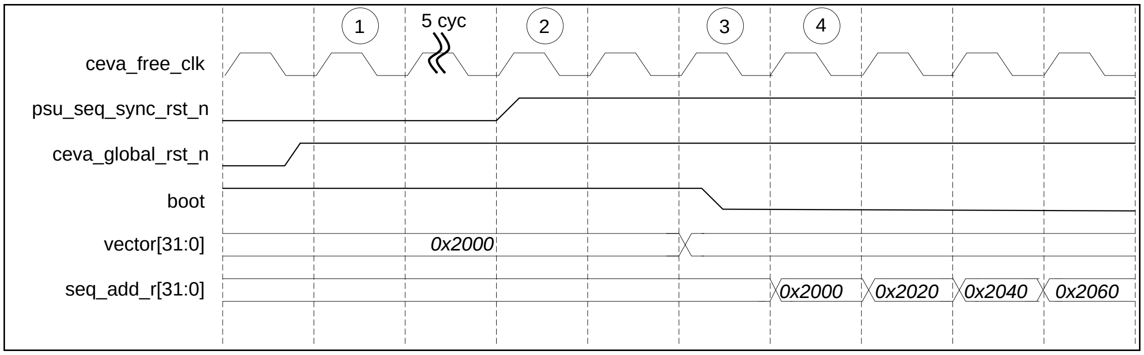

1
5 cyc
2
3
4
ceva_free_clk
psu_seq_sync_rst_n
ceva_global_rst_n
boot
vector[31:0]
0x2000
seq_add_r[31:0]
0x2000
0x2020
0x2040
0x2060

Figure 10-1: Boot Sequence Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: External core reset is released.
2. **Cycle 2**: After seven cycles, the PCU reset is released.
3. **Cycle 3**: After two cycles from PCU reset release, boot and vector can be released.
4. **Cycle 4**: Boot vector address is fetched from the program memory.

Table 10-1: Boot Sequence Signal Description

Name	Description
ceva_free_clk	Free clock
pmu_seq_sync_rst_n	PCU reset
ceva_global_rst_n	External core reset
boot	Boot indication
vector[31:0]	Boot vector address
seq_add_r[31:0]	Program fetch address

10.2 Boot Using an External Master (via EDAP)

Booting from PC 0x0 is done when the boot input is low when the reset is released. To ensure that the DSP is running on a valid code, the user should preload the program and data TCM using an external master through the EDAP. For more details, see Section 11.1.

10.3 Boot Using PDMA (Configured by the EDAP)

Similar to the boot option described in Section 10.2, the user can configure the PMSS PDMA registers (**PDEA**, **PDIA**, and **PDTC**) to download code to the PTCM. When the data resides in the internal memory, the user can release the core reset and start booting from PC 0x0.

The *core_rst* can be de-asserted when the PDST (bit 29 in the **PDTC** register) is reset, which means that the PDMA has finished the transfer.

Figure 10-2 shows this option.

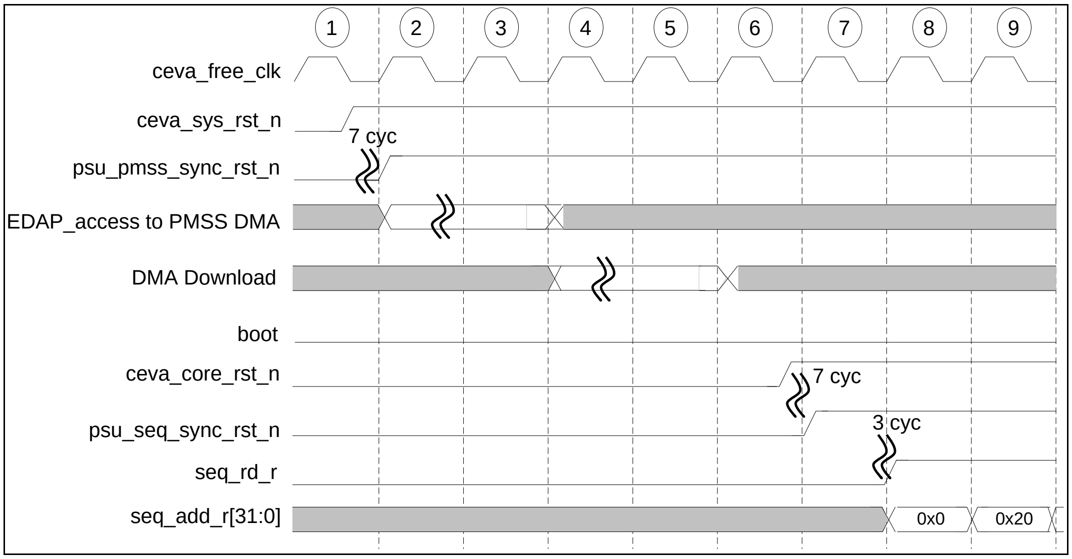

1
2
3
4
5
6
7
8
9
ceva_free_clk
ceva_sys_rst_n
7 cyc
psu_pmss_sync_rst_n
EDAP_access to PMSS DMA
DMA Download
boot
ceva_core_rst_n
7 cyc
psu_seq_sync_rst_n
3 cyc
seq_rd_r
seq_add_r[31:0]
0x0
0x20

Figure 10-2: Boot Using the PDMA Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: Release *global_rst* and *sys_rst*.
2. **Cycle 2-3** Update the PMSS DMA registers (**PDEA**, **PDIA**, and **PDTC**).
3. **Cycle 6**: When the code resides in the TCM, the user can de-assert the *core_rst* input.
4. Ten cycles later, the DSP starts fetching memory from PC 0x0.

10.4 Boot Using the Program Cache Memory (via SWOP)

Another option for booting is to use the program cache memory. In this option, the user preloads the cache memory with the PMSS SWOP mechanism and boots from an external location (boot input is high when the reset is released), while the DSP fetches the program from the cache and not from the external memory.

To configure the PMSS SWOP registers, the user should use the EDAP. In this option, *sys_rst* should also be de-asserted and the *core_rst* should be held low until the data resides in the cache memory.

Figure 10-1 shows this option.

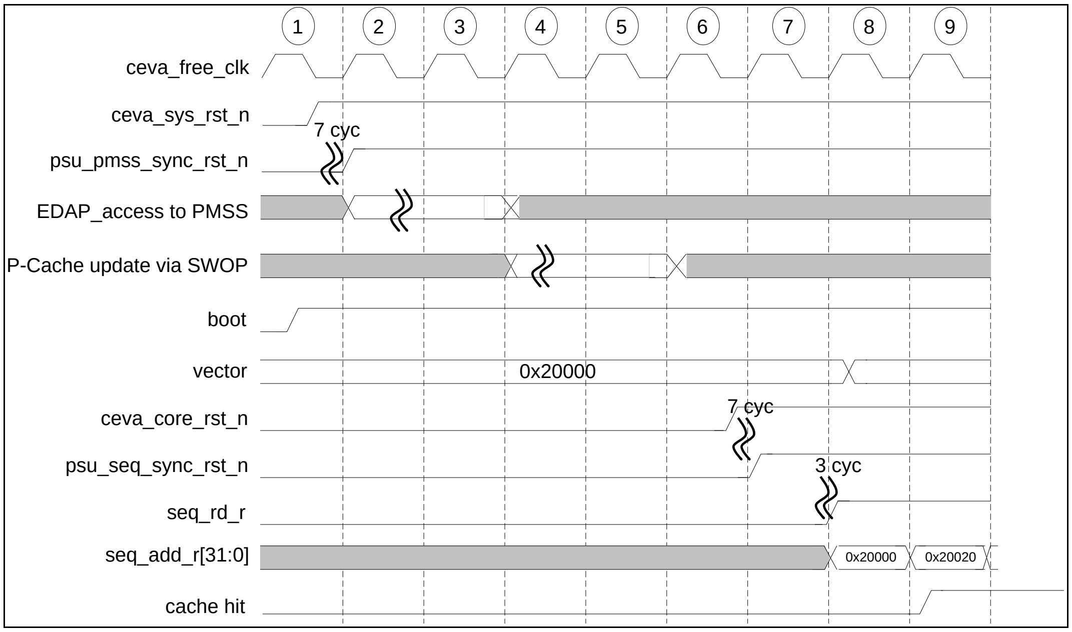

1
2
3
4
5
6
7
8
9
ceva_free_clk
ceva_sys_rst_n
7 cyc
psu_pmss_sync_rst_n
EDAP_access to PMSS
P-Cache update via SWOP
boot
vector
0x20000
ceva_core_rst_n
7 cyc
psu_seq_sync_rst_n
3 cyc
seq_rd_r
seq_add_r[31:0]
0x20000
0x20020
cache hit

Figure 10-3: Boot Using SWOP Mechanism Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: Release *global_rst* and *sys_rst*; boot and vector should be already valid.
2. Update the PMSS registers (invalidate the cache, enable it, and then program the SWOP mechanism to download code from the external memory to the cache).

   Polling the status bits of the PMSS CPM (for example, P_L1ICO) should be done via the EDAP.

3. **Cycle 6**: When the code resides in the PCache, the user can de-assert the *core_rst* input.
4. Ten cycles later, the DSP starts fetching memory from the address pointed to by the Vector input.

   The PMSS detects that the address resides in the cache and asserts the hit indication for each access, allowing zero wait-states of program fetch.

11. Reset Sequence

The CEVA-XM4 has the following resets:

- **Core Reset**, as described in Section 11.1
- **System Reset**, as described in Section 11.2
- **OCEM Reset**, as described in Section 11.3
- **Global Reset**, as described in Section 11.4

These resets must be asserted together.

Note: *De-assertion can be done separately under certain conditions (for example, leaving the core_rst only asserted while other resets are released, as described in Section 10).*

11.1 Core Reset

The core reset only resets the core (without the MSS, PSU, or OCEM).

This reset can be used during boot when another master uploads the code and data to the DSP while the DSP is in reset. When the upload is completed, the reset can be released and the DSP will start reading.

Figure 11-1 shows the reset procedure when the code and data are uploaded by an external master to the internal memories while the core is kept in reset.

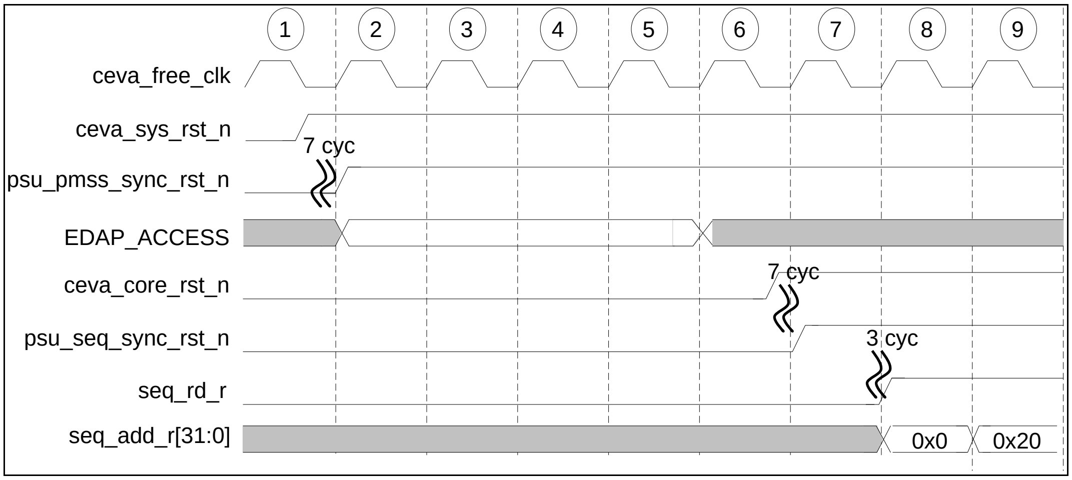

1
2
3
4
5
6
7
8
9
ceva_free_clk
ceva_sys_rst_n
7 cyc
psu_pmss_sync_rst_n
EDAP_ACCESS
ceva_core_rst_n
7 cyc
psu_seq_sync_rst_n
3 cyc
seq_rd_r
seq_add_r[31:0]
0x0
0x20

Figure 11-1: Core Reset Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: System (core and MSS) reset is asserted.
2. **Cycle 2**: EDAP transaction can begin seven cycles after SYS Reset is released.
3. **Cycle 6**: Core reset is released.
4. **Cycle 7**: PSU releases the reset to the DSP.
5. **Cycle 8**: Fetch request from program memory addresses.

Note: *ceva_global_rst_n should also be asserted before or with the ceva_sys_rst_n signal.*

Table 11-1: Core Reset Signal Description

Name	Description
ceva_free_clk	Free clock
ceva_sys_rst_n	Core and MSS reset
psu_pmss_sync_rst_n	PSU reset to the MSS
ceva_core_rst_n	Core Reset
psu_seq_sync_rst_n	PSU reset to the Sequencer
seq_rd_r	Program read indication
seq_add_r[31:0]	Program fetch address

11.2 System Reset

The system reset (*ceva_sys_rst_n*) resets the system of the DSP (core and MSS) while the On-Chip Emulation Module (OCEM) and PSU are still active.

System reset enables the user to reset the DSP while leaving the OCEM unaffected, allowing a daisy-chain debug with multicore to continue.

Figure 11-2 shows the behavior of the core and MSS internal resets after the system reset is deactivated.

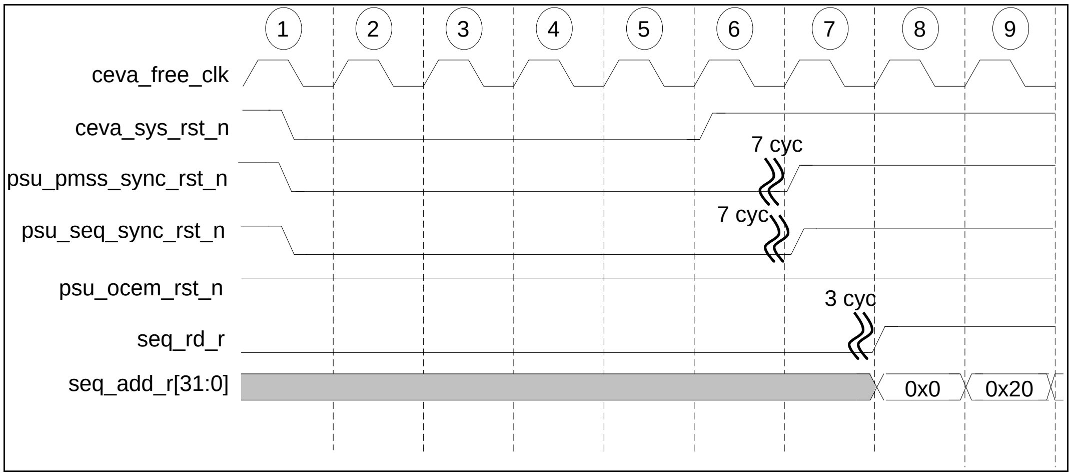

1
2
3
4
5
6
7
8
9
ceva_free_clk
ceva_sys_rst_n
psu_pmss_sync_rst_n
psu_seq_sync_rst_n
psu_ocem_rst_n
seq_rd_r
seq_add_r[31:0]
7 cyc
7 cyc
3 cyc
0x0
0x20

Figure 11-2: System Reset Timing Diagram

Table 11-2: System Reset Signal Description

Name	Description
ceva_free_clk	Free clock
ceva_sys_rst_n	Core and MSS reset
psu_pmss_sync_rst_n	PSU reset to the MSS
psu_seq_sync_rst_n	PSU reset to the Sequencer
psu_ocem_rst_n	PSU reset to the OCEM
seq_rd_r	Program read indication
seq_add_r[31:0]	Program fetch address

11.3 OCEM Reset

The OCEM reset only resets the OCEM. It does not affect any other resets inside the DSP (that is, there is no core, MSS, or PSU reset).

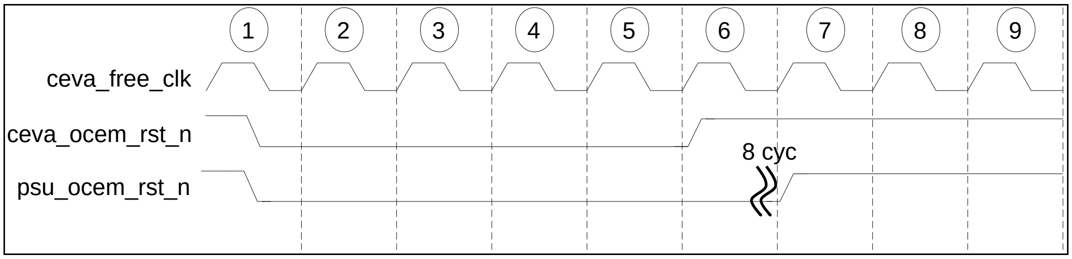

1
2
3
4
5
6
7
8
9
ceva_free_clk
ceva_ocem_rst_n
psu_ocem_rst_n
8 cyc

Figure 11-3: OCEM Reset Timing Diagram

Table 11-3: OCEM Reset Signal Description

Name	Description
ceva_free_clk	Free clock
ceva_ocem_rst_n	OCEM reset
psu_ocem_rst_n	PSU reset to OCEM

11.4 Global Reset

The global reset (*ceva_global_rst_n*) resets the entire DSP (Core, MSS, PSU, and OCEM).

Figure 11-4 shows the global reset sequence.

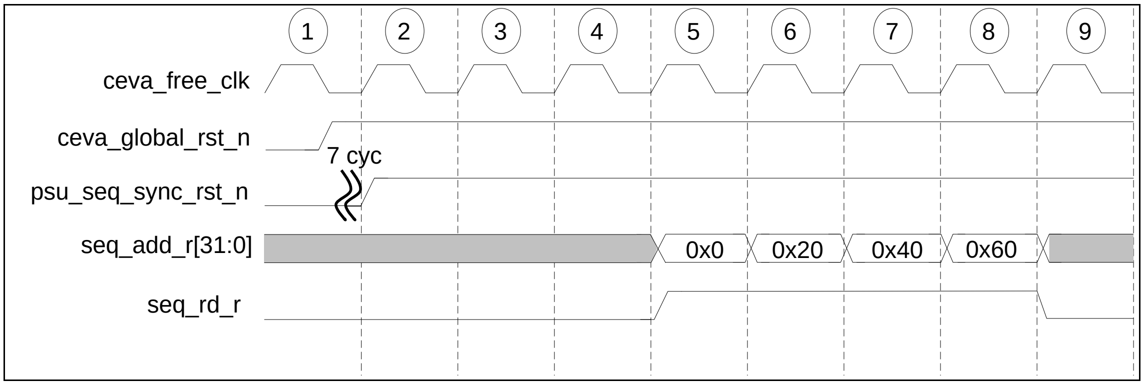

1
2
3
4
5
6
7
8
9
ceva_free_clk
ceva_global_rst_n
7 cyc
psu_seq_sync_rst_n
seq_add_r[31:0]
0x0
0x20
0x40
0x60
seq_rd_r

Figure 11-4: Global Reset Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: Global reset is asserted.
2. **Cycle 5**: Fetch request from program memory addresses.

Table 11-4: Global Reset Signal Description

Name	Description
ceva_free_clk	Free clock
ceva_global_rst_n	Global reset
psu_seq_sync_rst_n	PSU reset to Sequencer
seq_rd_r	Program read indication
seq_add_r[31:0]	Program fetch address

12. Power Scaling Unit

The following sections describe the Power Scaling Unit (PSU). The PSU has the following modes:

- **Free Run**: Related to unit clock control. All clocks to all units are toggling.
- **Dynamic Power Save (DPS)**: Related to unit clock control. Each unit asserts active and gets a clock according to its active status.
- **Light Sleep**: Related to unit clock control. No core units get clocks.
- **Standby**: Related to unit clock control. No core units or system units (for example, DMSS or PMSS) get clocks.
- **Deep Sleep**: Related to unit power control. Core and system power is down, memories are in retention mode.
- **Shutdown**: Related to unit power control. All power to the core, system, and memories is down.

The Deep Sleep and Shutdown modes are available only when the memory power gating option is chosen during installation.

The modes are changed by writing to the **PSVM** register in the PSU, via the *psu* instruction.

For details about the restrictions and limitations for switching between these modes, see the *CEVA-XM4 Architecture Specification*.

Table 12-1 describes the CEVA-XM4 output signal values in each PSU mode.

Table 12-1: Power Mode Value Registers

PSU Signal / Mode	cevaxm4_pmu_ core_idle_r	cevaxm4_pmu_ dsp_idle_r	cevaxm4_pmu_ pshtdwn_r	cevaxm4_pmu_ sys_pshtdwn_r
Free Running/DPS	0	0	8’b1111_1111	6’b11_1111
Light Sleep	1	0	8’b1111_1111	6’b11_1111
Standby	1	1	8’b1111_1111	6’b11_1111
Deep Sleep	1	1	8’b1111_1100	6’b00_0000
Shutdown	1	1	8’b0000_0000	6’b11_1111

12.1 Light Sleep Mode

To enter Light Sleep mode, the core raises *cevaxm4_psu_core_idle_r* six cycles after the *psu* instruction, as shown in Figure 12-1.

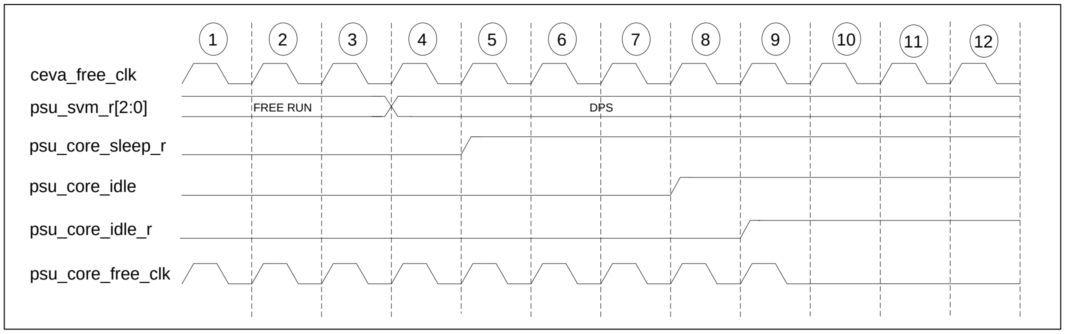

1
2
3
4
5
6
7
8
9
10
11
12
ceva_free_clk
psu_svm_r[2:0]
FREE RUN
DPS
psu_core_sleep_r
psu_core_idle
psu_core_idle_r
psu_core_free_clk

Figure 12-1: Entering Light Sleep Mode Timing Diagram

To exit Light Sleep, one of the following needs to be asserted, as shown in Figure 12-2:

- *int0/1/2*
- *ext_bp_req1/2*
- *vint* (all interrupts are accepted if their respective interrupt mask bit is asserted)
- *NMI*
- *core_rcvr*

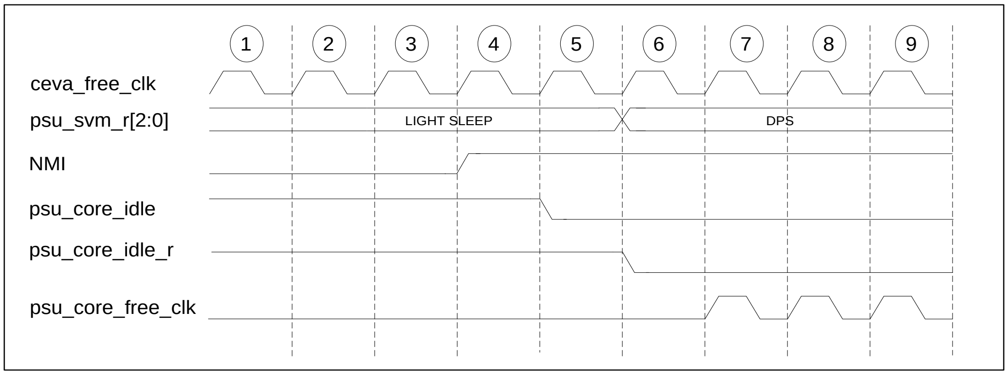

1
2
3
4
5
6
7
8
9
ceva_free_clk
psu_svm_r[2:0]
LIGHT SLEEP
DPS
NMI
psu_core_idle
psu_core_idle_r
psu_core_free_clk

Figure 12-2: Exiting Light Sleep Mode Timing Diagram

12.2 Standby Mode

To enter Standby, six cycles are needed to clear the core pipe, and all MSS units have to be in idle. Only then can *cevaxm4_psu_dsp_idle_r* rise, as shown in Figure 12-3.

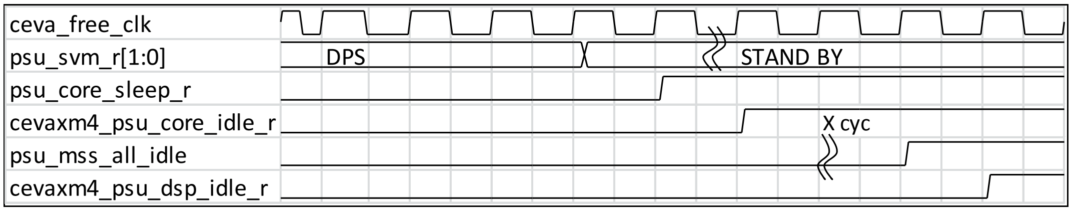

ceva_free_clk
psu_svm_r[1:0]
DPS
STAND BY
psu_core_sleep_r
cevaxm4_psu_core_idle_r
X cyc
psu_mss_all_idle
cevaxm4_psu_dsp_idle_r

Figure 12-3: Entering Standby Mode Timing Diagram

12.3 Retention Mode

To put the PMSS TCM, PMSS cache, and data TCM blocks 1 and 3 into Retention mode, the user should write zeros to the corresponding bits in the *RET* field of the **PSVM** register using a *psu* instruction. As a result, the relevant *cevaxm4_psu_sys_pshtdwn_r* bits de-assert and provide the user an indication about the blocks' Retention status.

To resume normal operation of the memories, the user should write ones to the corresponding bits in the *RET_LO* fields of the **PSVM_LOW** register using an *out* instruction.

Figure 12-4 shows entering and exiting Retention mode.

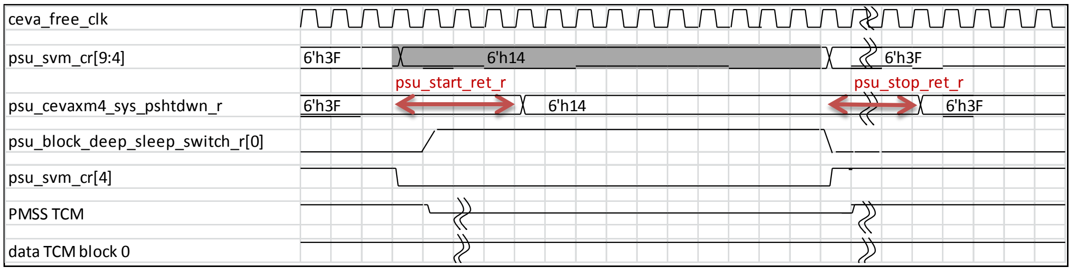

ceva_free_clk
psu_svm_cr[9:4]
6'h3F
6'h14
6'h3F
psu_start_ret_r
psu_stop_ret_r
psu_cevaxm4_sys_pshtdwn_r
6'h3F
6'h14
6'h3F
psu_block_deep_sleep_switch_r[0]
psu_svm_cr[4]
PMSS TCM
data TCM block 0

Figure 12-4: Entering/Exiting Retention Mode Timing Diagram

12.4 Shutdown Mode

To enter Shutdown mode, the user should write zeros to the corresponding bits in the *SHTDWN_HI* and *SHTDWN_LO* fields of the **PSVM_HI** and **PSVM_LOW** registers using *out* instructions.

The PSU checks that the MSS, core, and RTT (if RTT is installed) are idle, and then the relevant *cevaxm4_pmu_pshtdwn_r* bits de-assert and provide the user an indication of the blocks' Shutdown status.

Shutting down the core and MSS using the corresponding bit in the *SHTDWN_LO* field of the **PSVM_LO** register also shuts down the OCEM and RTT.

Figure 12-5 shows entering and exiting Shutdown mode.

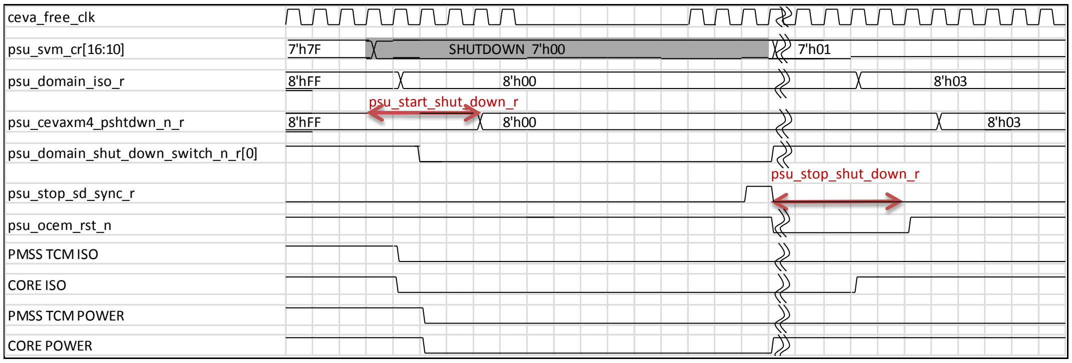

ceva_free_clk
psu_svm_cr[16:10]
7'h7F
SHUTDOWN_7'h00
7'h01
psu_domain_iso_r
8'hFF
8'h00
8'h03
psu_start_shut_down_r
psu_cevaxm4_pshtdwn_n_r
8'hFF
8'h00
8'h03
psu_domain_shut_down_switch_n_r[0]
psu_stop_shut_down_r
psu_stop_sd_sync_r
psu_ocem_rst_n
PMSS TCM ISO
CORE ISO
PMSS TCM POWER
CORE POWER

Figure 12-5: Entering/Exiting Shutdown Mode Timing Diagram

Note: *Memory Retention/OFF modes are dependent on the memory modules provided by the vendor. These abilities should be integrated as part of the vendor's memory solution.*

12.5 AXI Low-Power Interface

It is possible to move from the Light Sleep to the Standby modes and vice versa (without waking up the core) by de-asserting *csysreq*.

After all pending transactions in the MSS are completed, the PSU de-asserts *cevaxm4_psu_cactive_r*. One cycle later, *cevaxm4_psu_csysack_r* is de-asserted, as shown in Figure 12-6.

To move from Standby mode to Light Sleep, *csysreq* must be asserted. On the next cycle, the active indication is asserted. After one more cycle, *csysack* rises.

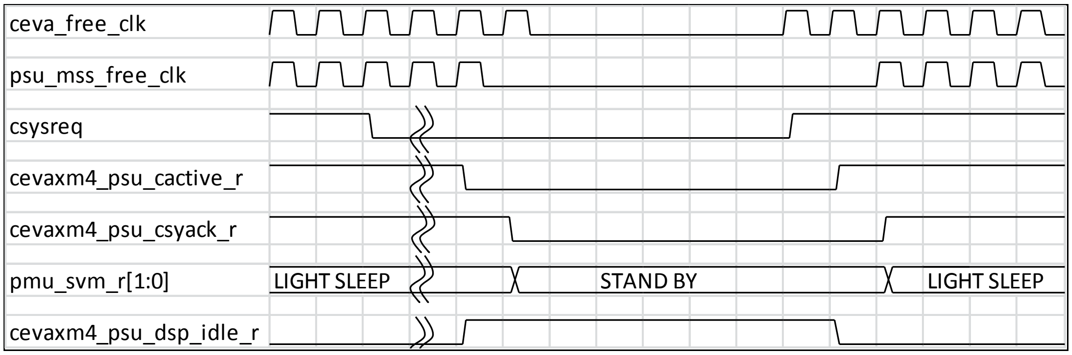

ceva_free_clk
psu_mss_free_clk
csysreq
cevaxm4_psu_cactive_r
cevaxm4_psu_csyack_r
pmu_svm_r[1:0]
LIGHT SLEEP
STAND BY
LIGHT SLEEP
cevaxm4_psu_dsp_idle_r

Figure 12-6: AXI Low-Power Interface Timing Diagram

13. General Purpose Input/Output

There are two 32-bit general purpose registers in the CEVA-XM4 that can be accessed by *in*/*out* instructions in the same way as all other CPM registers.

- The General Purpose Input (**GPIN**) register at address 0x000034 is connected directly to the *bus gp_in*[31:0] input. These pins are connected directly to the CPM registers' read MUX, and are assumed to be timed to the core clock. Any necessary retiming must be carried out externally.
- The General Purpose Output (**GPOUT**) register at address 0x000038 is connected directly to the *cevaxm4_gpout*[31:0] output bus.

Figure 13-1 shows the timing of a write access to the **GPOUT** register and a read access from the **GPIN** register.

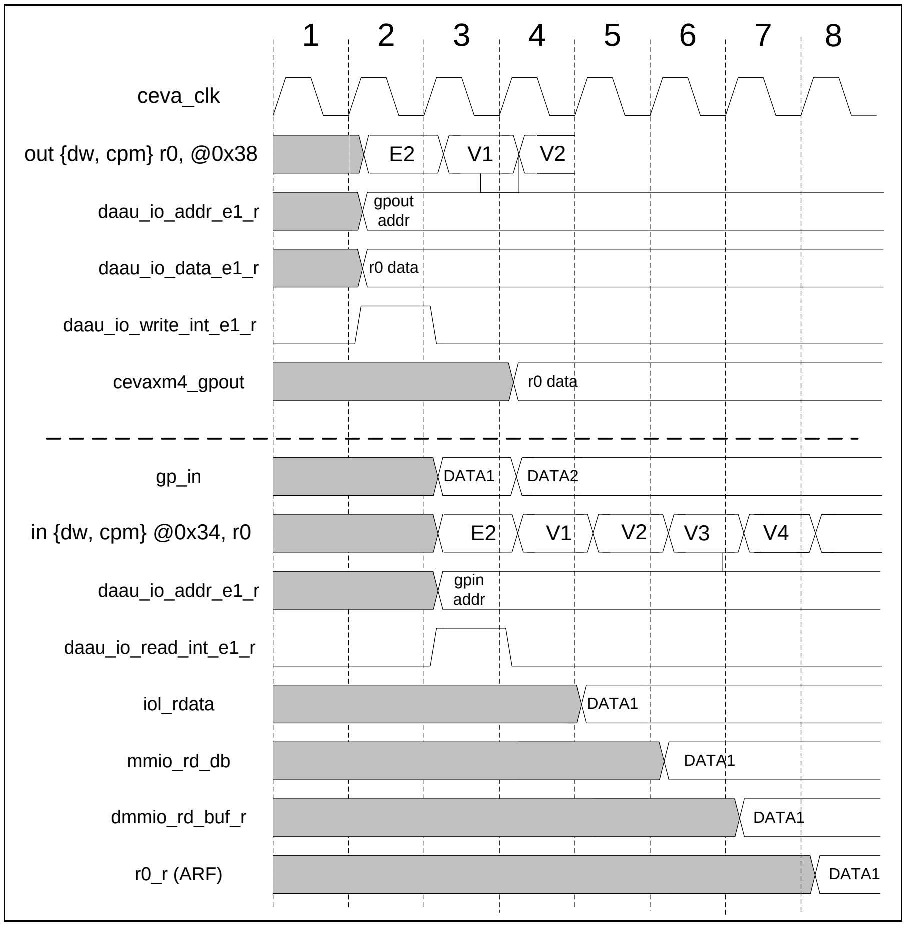

1
2
3
4
5
6
7
8
ceva_clk
out {dw, cpm} r0, @0x38
E2
V1
V2
daau_io_addr_e1_r
gpout addr
daau_io_data_e1_r
r0 data
daau_io_write_int_e1_r
cevaxm4_gpout
r0 data
gp_in
DATA1
DATA2
in {dw, cpm} @0x34, r0
E2
V1
V2
V3
V4
daau_io_addr_e1_r
gpin addr
daau_io_read_int_e1_r
iol_rdata
DATA1
mmio_rd_db
DATA1
dmmio_rd_buf_r
DATA1
r0_r (ARF)
DATA1

Figure 13-1: GPIO Timing Diagram

The transaction sequence for write access to **GPOUT** is as follows:

1. **Cycle 2**: *out* instruction to **GPOUT** at pipe stage E2; I/O address, data and control output from core.
2. **Cycle 3**: *out* instruction buffered in DMSS.
3. **Cycle 4**: **GPOUT** and *cevaxm4_gpout* output bus updated with data value.

The transaction sequence for read access from **GPIN** is as follows:

1. **Cycle 3**: *in* instruction from **GPIN** at pipe stage E2; I/O address and control output from core.
2. **Cycle 4**: *in* instruction buffered in DMSS.
3. **Cycle 5**: *gp_in* DATA1 captured into DMSS data register at pipe stage V2.
4. **Cycle 6**: DATA1 captured into I/O read data register at pipe stage V3.
5. **Cycle 7**: DATA1 transferred to scalar unit read buffer at pipe stage V4.
6. **Cycle 8**: Destination ARF register updated with DATA1.

14. Undefined Opcodes

The CEVA-XM4 implements a dedicated mechanism for handling undefined opcodes. When the core detects an undefined opcode, it translates it as a *nop* instruction, stores the packet address, and then issues an interrupt signal.

In addition, when an undefined opcode occurs, the corresponding unit bit in the **UOP_STS** register is set.

In the example shown in Figure 14-1, the following occurs:

1. An undefined opcode is detected in the VB unit.
2. *uop_int* is raised after a cycle and is stretched for XCI_COR+1 cycles.
3. Bit 10 in the **UOP_STS** register (the *vb uop* bit), and the relevant address is written to **UOP_PAR**.

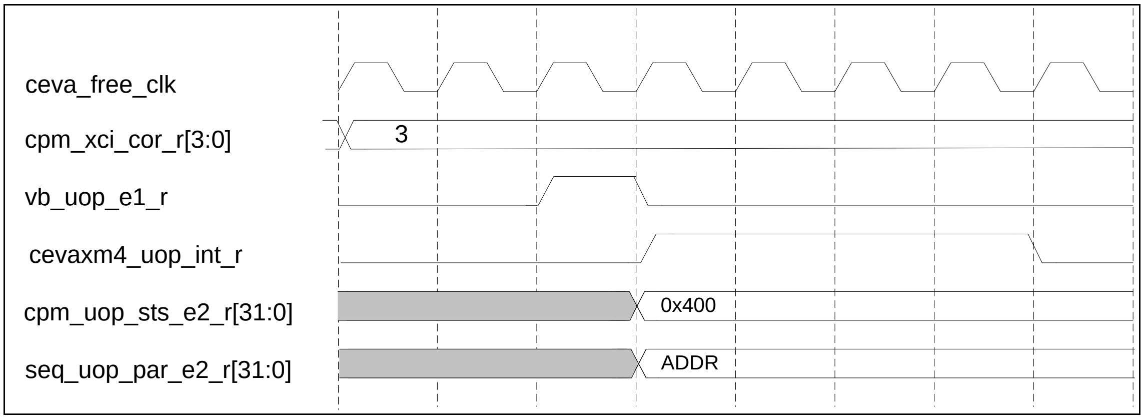

ceva_free_clk
cpm_xci_cor_r[3:0]
3
vb_uop_e1_r
cevaxm4_uop_int_r
cpm_uop_sts_e2_r[31:0]
0x400
seq_uop_par_e2_r[31:0]
ADDR

Figure 14-1: uop Interrupt, Status, and Packet Address Timing Diagram

15. CEVA-Xtend Hardware

The CEVA-XM4 DSP core family provides end users with the ability to customize the system according to a specific application(s). CEVA-Xtend™ units are additional user-defined instructions intended to expand the original core's instruction set for specific applications.

CEVA-Xtend units can be integrated along with a Scalar Processing Unit (SPU) and Vector Processing Unit (VPU). The end user can create new instructions that activate the CEVA-Xtend hardware units. The syntax and the encoding of these instructions are customized and defined according to application needs and architecture guidelines.

For more details about CEVA-Xtend, see the *CEVA-XM4 Architecture Specification.*

Table 15-1 and Table 15-2 list the CEVA-XM4 Xtend interfaces and their matching names in the architecture specification.

Table 15-1: CEVA-Xtend Interface (SPU)

Interface Name	Name in Architecture Specification
xtend_scalar_dst_data	CXS DEST
cevaxm4_xh_src0_m_r	CXS SRC0
cevaxm4_xh_src1_m_r	CXS SRC1
cevaxm4_xh_pr_en_e1	CXS PRX
cevaxm4_xh_pir_a1_r	CXS OP
cevaxm4_xh_pir_valid_a1_r	CXS OP VALID
cevaxm4_xh_pext_a1_cr	CXS EXT
cevaxm4_xh_pext10_valid_a1_r	CXS 10EXT VALID
cevaxm4_xh_pext26_valid_a1_r	CXS 26EXT VALID

Table 15-2: CEVA-Xtend Interface (VPU)

Interface Name	Name in Architecture Specification
cxm_vpu_xtend_dest_v4_r	CXV DEST
cxm_vpu_xtend_en_v4_r	CXV WR EN
cxm_vpu_xtend_src0_e2	CXV SRC0
cxm_vpu_xtend_src1_e2	CXV SRC1
cxm_vpu_xtend_vpr_e2	CXV VPRX
cxm_vpu_xtend_pir_e1_r	CXV OP
cxm_vpu_xtend_pext_e1_r	CXV EXT
cxm_vpu_xtend_pir_valid_e1_r	CXV OP VALID
cxm_vpu_xtend_pext10_valid_e1_r	CXV 10EXT VALID
cxm_vpu_xtend_pext26_valid_e1_r	CXV 26EXT VALID

16. DesignWare

The RTL uses several DesignWare (DW) modules. The locations of all DW modules in the RTL can be found in the **cevaxm4.scan.resources** file located in the backend reference directory in **cevaXM4_V1.1.1.F_Reference_ files.tgz**.

17. External Access

The CEVA-XM4 contains internal data and program memories. The internal memories are accessible without latency. The program and data internal memory sizes are configurable, as described in Section 3.

To access a larger memory space, the CEVA-XM4 includes dedicated AXI Master interfaces that specify the following separated standard ports for external access:

- External Program Port (EPP) Master
- External Data Port (EDP)/AXImX Master

The External Device Access Port (EDAP) and AXIsX AXI slave interfaces allow an external Master to access the internal data memories or multicore messaging registers.

An I/O port is also supported to allow the control of external peripheral devices. This I/O port is an APB3 Master interface, and is dedicated for the interface between the CEVA-XM4 and external devices or peripherals.

Accesses to external space require certain latency. The following sections detail the latency for each access type:

- EPP access
- EDP/AXImX access
- EDAP/AXIsX transfers
- I/O port access

17.1 Automatic Power Saving Mode

The EDP, EPP, AXImX, and I/O interfaces generate an Automatic Power Saving (APS) signal that can be used by the external environment to save power by only clocking the AXI or APB subsystems when required to do so by the CEVA-XM4. An example of how to use the signals to reduce power when not in use is shown in Figure 17-1.

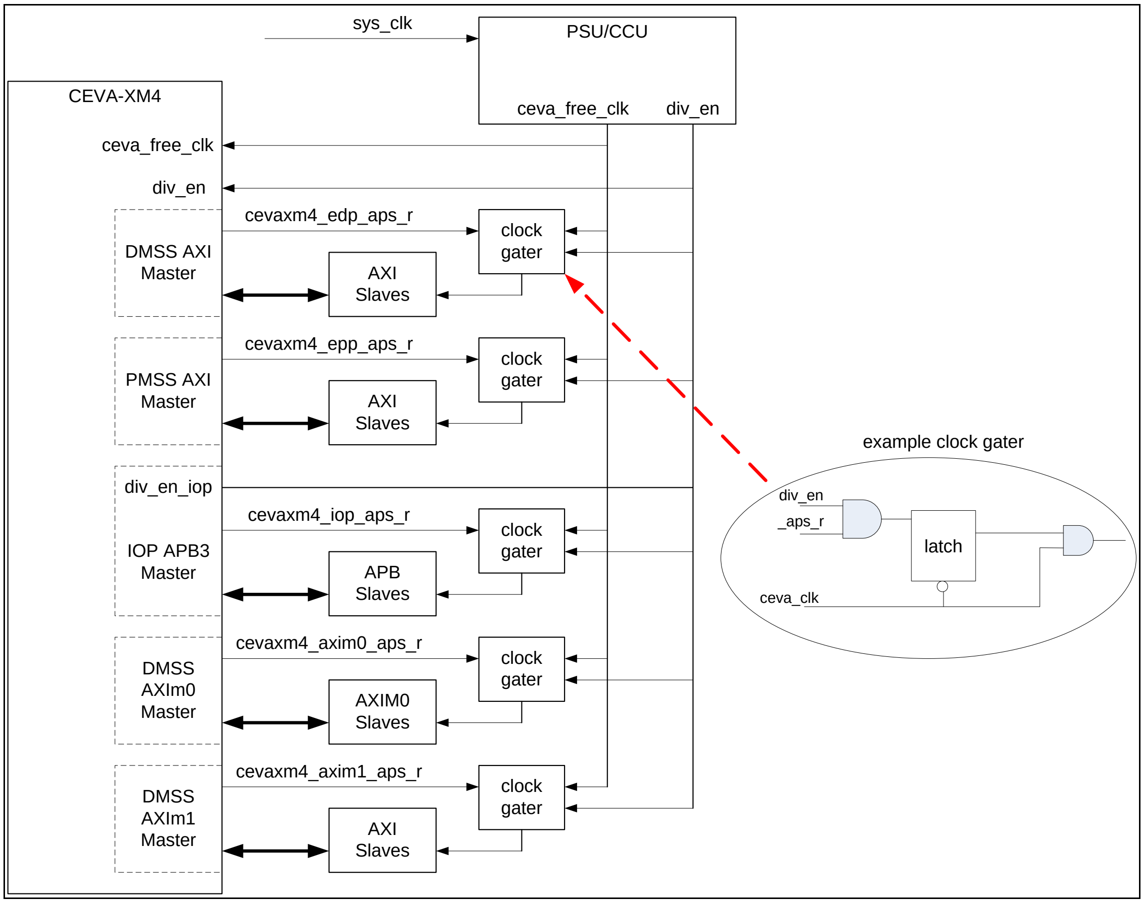

sys_clk
PSU/CCU
ceva_free_clk
div_en
CEVA-XM4
ceva_free_clk
div_en
cevaxm4_edp_aps_r
DMSS AXI Master
clock gater
AXI Slaves
cevaxm4_epp_aps_r
PMSS AXI Master
clock gater
AXI Slaves
div_en_iop
cevaxm4_iop_aps_r
IOP APB3 Master
clock gater
APB Slaves
cevaxm4_axim0_aps_r
DMSS AXIm0 Master
clock gater
AXIM0 Slaves
cevaxm4_axim1_aps_r
DMSS AXIm1 Master
clock gater
AXI Slaves
example clock gater
div_en
_aps_r
latch
ceva_clk

Figure 17-1: Example APS Usage

In this example, the APS signals can be used to gate off the AXI/APB clock to save power when no transactions are taking place on the buses. The gating can occur inside either the PSU or Clock Control Unit (CCU). The *div_en/div_en_axim0/1/div_en_epp/div_en_iop* signals **must** be constantly driven to the CEVA-XM4; otherwise, no APS signals are output. The *div_en/div_en_axim0/1/div_en_epp/div_en_iop* signals **must** be high one core clock cycle before the rising edge of the AXI/APB clock, and low at all other times. As shown, the PSU/CCU produces a common *div_en* for the interfaces.

Figure 17-2 shows an example of how the APS signal is used to enable the APB3 slave clock.

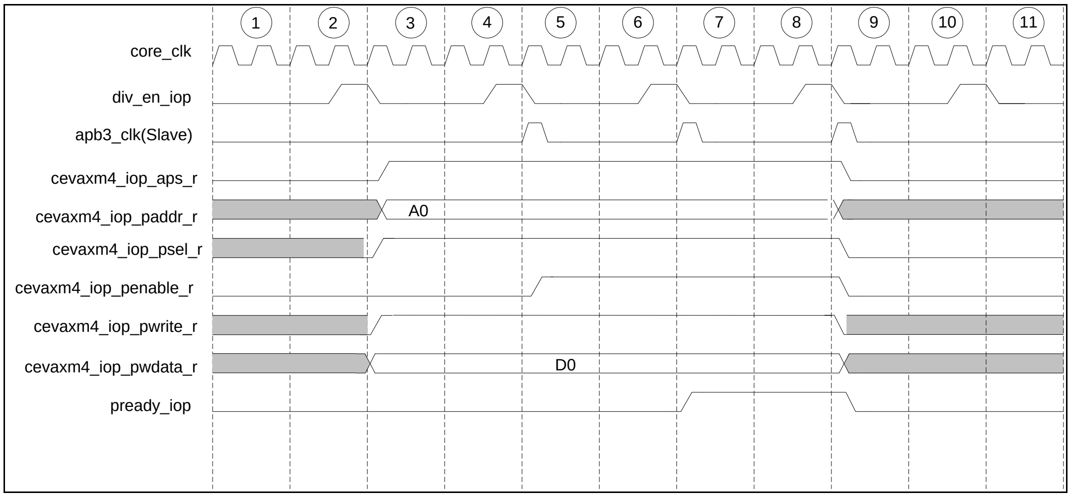

1
2
3
4
5
6
7
8
9
10
11
core_clk
div_en_iop
apb3_clk(Slave)
cevaxm4_iop_aps_r
cevaxm4_iop_paddr_r
A0
cevaxm4_iop_psel_r
cevaxm4_iop_penable_r
cevaxm4_iop_pwrite_r
cevaxm4_iop_pwdata_r
D0
pready_iop

Figure 17-2: APB3 APS Usage Timing Diagram

In this example, an I/O port write transaction enables the slave clock for the duration of the transaction. *pready* can be used by the slave to extend the transaction if required. The CEVA-XM4 only changes its APB3 outputs when *div_en_iop* is high.

17.2 External Program Port

The PMSS accesses the external memory via a dedicated AMBA AXI interface port (EPP). This port is hardware configured to 128-bit width.

The external memory is accessed in the following cases:

- **Read Request**: The core requests an instruction from a fetch address that is outside of the TCM address range, and the address is mapped to a non-cacheable region by the IACU.
- **Cache Miss**: The core requests an instruction from a fetch address that is outside of the TCM address range, the address is mapped to a cacheable region by the IACU, and the requested fetch line is not valid in the cache (cache miss).
- **Read Access**: The PDMA download reads from the external memory and writes to the internal memory.

  The software operation unit will initiate read access from the EPP when it is configured to do a pre-fetch operation and the requested cache blocks are not valid in the cache.
- **Read and Write Access**: The user accesses via the OCEM.

For more details about the AMBA AXI interface, see the *AMBA AXI Protocol Specification*.

17.2.1 EPP Read from a Non-Cacheable Region

Figure 17-3 shows an EPP read from a non-cacheable region (that is, a Miss followed by a No Read). It also shows the internal flow of a read requested by the core (*prd_r*) until the fetch line is provided to it (*mp_pmc_epp_rdata*[255:0]).

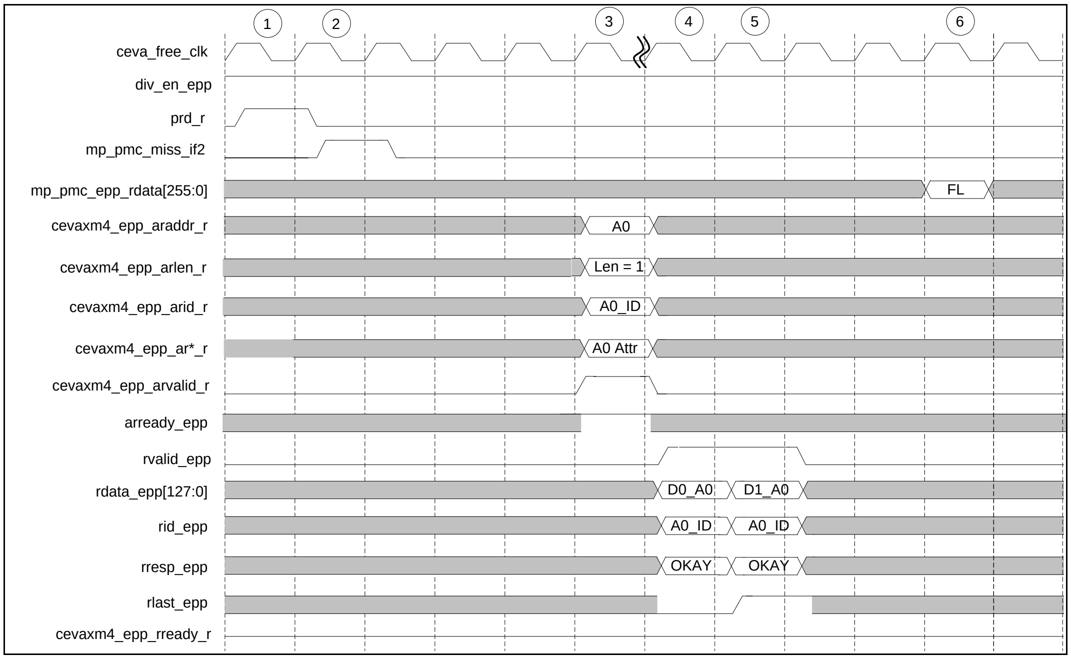

1
2
3
4
5
6
ceva_free_clk
div_en_epp
prd_r
mp_pmc_miss_if2
mp_pmc_epp_rdata[255:0]
FL
cevaxm4_epp_araddr_r
A0
cevaxm4_epp_arlen_r
Len = 1
cevaxm4_epp_arid_r
A0_ID
cevaxm4_epp_ar*_r
A0 Attr
cevaxm4_epp_arvalid_r
arready_epp
rvalid_epp
rdata_epp[127:0]
D0_A0
D1_A0
rid_epp
A0_ID
A0_ID
rresp_epp
OKAY
OKAY
rlast_epp
cevaxm4_epp_rready_r

Figure 17-3: EPP Read from Non-Cacheable Region Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The core requests data from the MSS (program memory).
2. **Cycle 2**: The program memory controller indicates a miss.
3. **Cycle 3**: The core issues a read request for a single fetch line from the external AXI slave.
4. **Cycle 4**: The AXI slave drives the first phase of the read response data.
5. **Cycle 5**: The AXI slave drives the second and last phases of read response data.
6. **Cycle 6**: The fetch line data is sent to the core.

Table 17-1: EPP Read from Non-Cacheable Region Signal Description

Name	Description
ceva_free_clk	Free clock
div_en_epp	EPP AXI clock enable
prd_r	DSP sequencer read indication
mp_pmc_miss_if2	PMSS miss indication
mp_pmc_epp_rdata	Data bus from PMSS to core
cevaxm4_epp_araddr_r	AXI RAC address
cevaxm4_epp_arid_r	AXI RAC ID
cevaxm4_epp_arlen_r	AXI RAC length
cevaxm4_epp_arvalid_r	AXI RAC valid
arready_epp	AXI RAC ready
rvalid_epp	AXI RDC valid
rdata_epp	AXI RDC data
rid_epp	AXI RDC ID
rresp_epp	AXI RDC response indication
rlast_epp	AXI RDC last
cevaxm4_epp_rready_r	AXI RDC ready

17.2.2 EPP Read from Non-Cacheable Region (ECC on AXI)

When the AMBA bus ECC is enabled, there is one extra cycle of latency when performing an external read using the EPP, as shown in Figure 17-4.

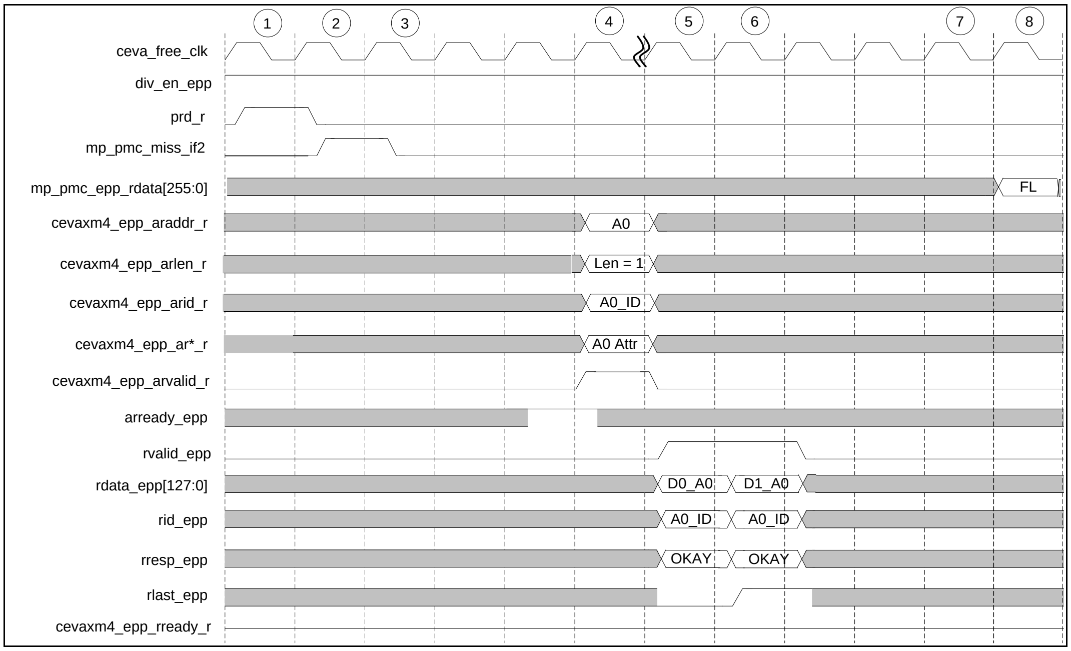

1
2
3
4
5
6
7
8
ceva_free_clk
div_en_epp
prd_r
mp_pmc_miss_if2
mp_pmc_epp_rdata[255:0]
FL
cevaxm4_epp_araddr_r
A0
cevaxm4_epp_arlen_r
Len = 1
cevaxm4_epp_arid_r
A0_ID
cevaxm4_epp_ar*_r
A0 Attr
cevaxm4_epp_arvalid_r
arready_epp
rvalid_epp
rdata_epp[127:0]
D0_A0
D1_A0
rid_epp
A0_ID
A0_ID
rresp_epp
OKAY
OKAY
rlast_epp
cevaxm4_epp_rready_r

Figure 17-4: EPP Read from Non-Cacheable Region with (ECC on AXI) Timing Diagram

17.2.3 EPP Read with Slower AXI

Figure 17-5 shows a core read from the EPP when the AXI clock is half of the frequency of the core clock. All AXI signals are launched and captured on the clock's rising edge when *div_en_epp* is high.

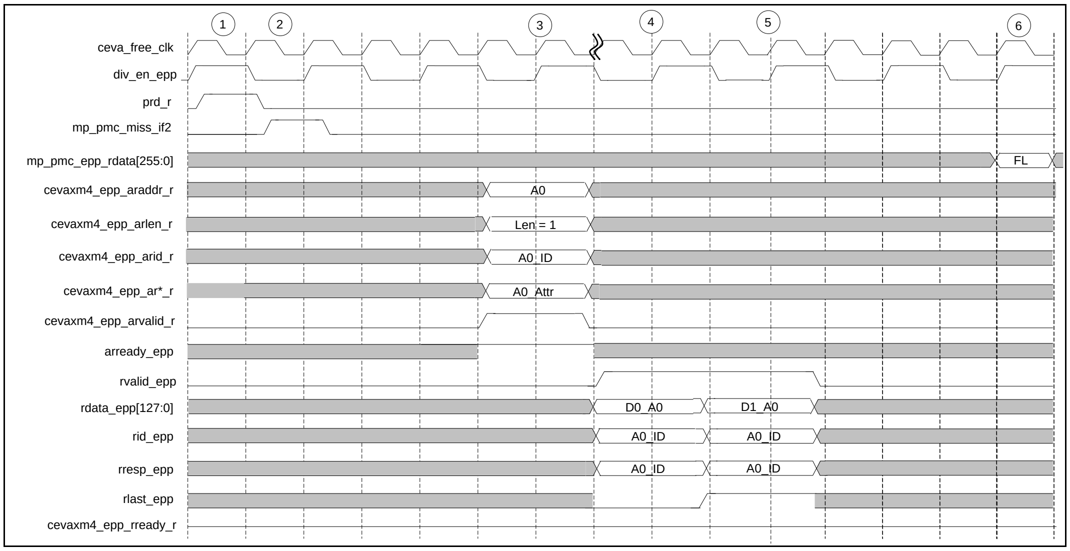

①
②
③
④
⑤
⑥
ceva_free_clk
div_en_epp
prd_r
mp_pmc_miss_if2
mp_pmc_epp_rdata[255:0]
FL
cevaxm4_epp_araddr_r
A0
cevaxm4_epp_arlen_r
Len = 1
cevaxm4_epp_arid_r
A0_ID
cevaxm4_epp_ar*_r
A0_Attr
cevaxm4_epp_arvalid_r
arready_epp
rvalid_epp
rdata_epp[127:0]
D0_A0
D1_A0
rid_epp
A0_ID
A0_ID
rresp_epp
A0_ID
A0_ID
rlast_epp
cevaxm4_epp_rready_r

Figure 17-5: EPP Read with Slower AXI Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The core requests data from the MSS (program memory).
2. **Cycle 2**: The program memory controller indicates a miss.
3. **Cycle 3**: The core issues a read request for a single fetch line from the external AXI slave.
4. **Cycle 4**: The AXI slave drives the first phase of the read response data.
5. **Cycle 5**: The AXI slave drives the second and last phases of read response data.
6. **Cycle 6**: The fetch line data is sent to the core.

17.2.4 EPP Read from Cacheable Region

Figure 17-6 shows an EPP read from a cacheable region (that is, a Miss). It also shows the internal flow of a read requested by the core (*prd_r*) until the fetch line is provided to the core (*mp_pmc_epp_rdata*[255:0]) and the cache block is written to the cache.

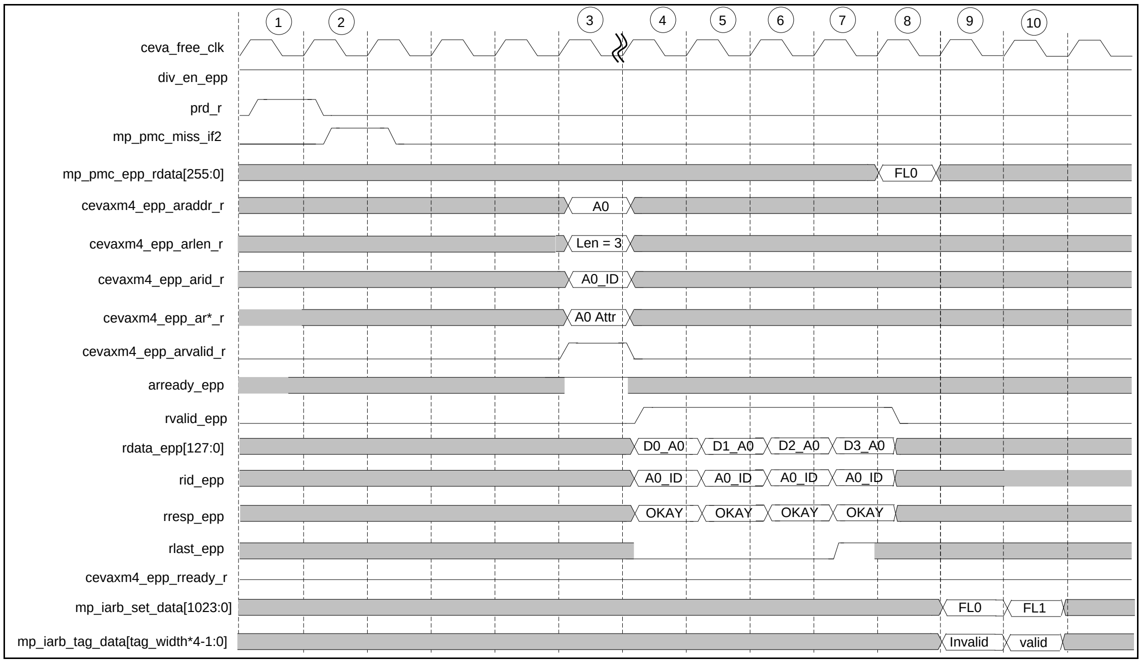

1
2
3
4
5
6
7
8
9
10
ceva_free_clk
div_en_epp
prd_r
mp_pmc_miss_if2
mp_pmc_epp_rdata[255:0]
FL0
cevaxm4_epp_araddr_r
A0
cevaxm4_epp_arlen_r
Len = 3
cevaxm4_epp_arid_r
A0_ID
cevaxm4_epp_ar*_r
A0 Attr
cevaxm4_epp_arvalid_r
arready_epp
rvalid_epp
rdata_epp[127:0]
D0_A0
D1_A0
D2_A0
D3_A0
rid_epp
A0_ID
A0_ID
A0_ID
A0_ID
rresp_epp
OKAY
OKAY
OKAY
OKAY
rlast_epp
cevaxm4_epp_rready_r
mp_iarb_set_data[1023:0]
FL0
FL1
mp_iarb_tag_data[tag_width*4-1:0]
Invalid
valid

Figure 17-6: EPP Read from Cacheable Region Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The core issues a read request for a single fetch line from the address mapped to the cacheable region.
2. **Cycle 2**: The program memory controller indicates a miss (that is, the requested fetch line is not valid in the cache).
3. **Cycle 3**: The core issues a read request for two fetch lines from the external AXI slave (fetch cache block).
4. **Cycle 4**: The AXI slave drives the first phase of read response data.
5. **Cycle 5**: The AXI slave drives the second phase of read response data (the first full fetch line is received from the EPP).
6. **Cycle 6**: The AXI slave drives the third read response data.
7. **Cycle 7**: The AXI slave drives the last read response data (the second full fetch line is received from the EPP).
8. **Cycle 8**: The first fetch line data is sent to the core.

9. **Cycle 9**: The first fetch line data is written to the cache (during this write, the cache block is invalidated because it is now in the middle of writing).
10. **Cycle 10**: The second fetch line is written to the cache (after this write, the new cache block will be valid in the cache).

Table 17-2: EPP Read from Cacheable Region Signal Description

Name	Description
ceva_free_clk	Free clock
div_en_epp	EPP AXI clock enable
prd_r	DSP sequencer read indication
mp_pmc_miss_if2	PMSS miss indication
mp_pmc_epp_rdata	Data bus from PMSS to core
cevaxm4_epp_araddr_r	AXI RAC address
cevaxm4_epp_arid_r	AXI RAC ID
cevaxm4_epp_arlen_r	AXI RAC length
cevaxm4_epp_arvalid_r	AXI RAC valid
arready_epp	AXI RAC ready
rvalid_epp	AXI RDC valid
rdata_epp	AXI RDC data
rid_epp	AXI RDC ID
rresp_epp	AXI RDC response indication
rlast_epp	AXI RDC last
cevaxm4_epp_rready_r	AXI RDC ready
mp_iarb_set_data	Write data bus to the set array
mp_iarb_tag_data	Write data bus to the tag array

17.2.5 PDMA Download (EPP Read)

Figure 17-7 shows the download of five fetch lines using the DMA. The DMA configuration is burst length of three, with two outstanding reads. It is assumed that the DMA is not interrupted and there are zero wait-states on the external memory.

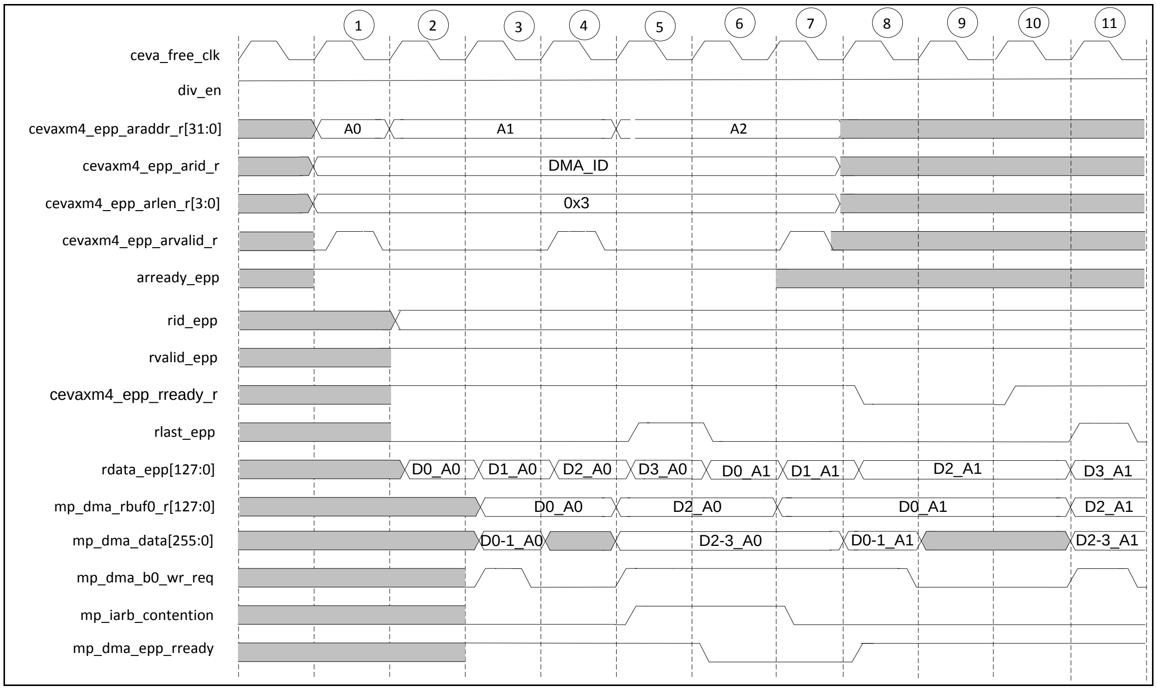

1
2
3
4
5
6
7
8
9
10
11
ceva_free_clk
div_en
cevaxm4_epp_araddr_r[31:0]
A0
A1
A2
cevaxm4_epp_arid_r
DMA_ID
cevaxm4_epp_arlen_r[3:0]
0x3
cevaxm4_epp_arvalid_r
arready_epp
rid_epp
rvalid_epp
cevaxm4_epp_rready_r
rlast_epp
rdata_epp[127:0]
D0_A0
D1_A0
D2_A0
D3_A0
D0_A1
D1_A1
D2_A1
D3_A1
mp_dma_rbuf0_r[127:0]
D0_A0
D2_A0
D0_A1
D2_A1
mp_dma_data[255:0]
D0-1_A0
D2-3_A0
D0-1_A1
D2-3_A1
mp_dma_b0_wr_req
mp_iarb_contention
mp_dma_epp_rready

Figure 17-7: PDMA Download (EPP Read) Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The EPP issues a read request for the first fetch line from the external AXI slave with a burst length of 0x3.
2. **Cycle 2**: The AXI slave drives the first phase of the first read request.
3. **Cycle 3**:
    - 3.a The AXI slave drives the second phase of first read request.
    - 3.b The DMA accesses the TCM with the first fetch line.
    - 3.c The internal arbiter grants the first request.
4. **Cycle 4**: The AXI slave drives the third phase of the first read request, and the EPP issues the second read request.

5. **Cycle 5**:
    - 5.a The AXI slave drives the fourth and last phases of the second read request.
    - 5.b The DMA accesses the TCM with the second fetch line.
    - 5.c The DMA request loses arbitration because of contention issues with the RAB.
6. **Cycle 6**:
    - 6.a The AXI slave drives the first phase of the second read request.
    - 6.b The DMA keeps accessing the TCM with the second fetch line.
    - 6.c The DMA request loses arbitration because of contention issues with the RAB.
    - 6.d The DMA buffer is full and the DMA ready indication goes low.
7. **Cycle 7**:
    - 7.a The EPP issues a read request for the third fetch line from the external memory.
    - 7.b The AXI slave drives the second phase of second read request.
    - 7.c The DMA keeps accessing the TCM with the second fetch line.
    - 7.d The internal arbiter grants the second request.
    - 7.e The buffer is still full and the DMA ready indication is still low.
8. **Cycle 8**:
    - 8.a The AXI slave drives the third phase of the second read request.
    - 8.b The EPP lowers the ready signal because of a full buffer.
    - 8.c The DMA accesses the TCM with the third fetch line.
    - 8.d The internal arbiter grants the third request.
    - 8.e The DMA ready indication is asserted.
9. **Cycle 9**: The AXI slave keeps driving the third phase of the second read request.
10. **Cycle 10**: The AXI slave keeps driving the third phase of the second read request, and the EPP asserts the ready signal.
11. **Cycle 11**:
    - 11.a The AXI slave drives the fourth phase of the second read request.
    - 11.b The DMA accesses the TCM with the fourth fetch line.
    - 11.c The internal arbiter grants the fourth request.

Table 17-3: PDMA Download (EPP Read) Signal Description

Name	Description
ceva_free_clk	Free clock
div_en	AXI clock enable
cevaxm4_epp_araddr_r	AXI RAC address
cevaxm4_epp_arid_r	AXI RAC ID
cevaxm4_epp_alen_r	AXI RAC length
cevaxm4_epp_arvalid_r	AXI RAC valid
arready_epp	AXI RAC ready
rid_epp	AXI RDC ID
rvalid_epp	AXI RDC valid
cevaxm4_epp_rready_r	AXI RDC ready
rlast_epp	AXI RDC last
rdata_epp	AXI RDC data
mp_dma_rbuf0_r	PDMA 128-bit data buffer #0
mp_dma_data	PDMA fetch line to the internal arbiter
mp_dma_b0_wr_req	PDMA write request to the internal arbiter block0
mp_iarb_dma_wr_grant	PDMA write grant from the internal arbiter
mp_dma_epp_rready	PDMA ready indication to the EPP

17.2.6 EPP OCEM Write

Figure 17-8 shows the OCEM write to the external program memory.

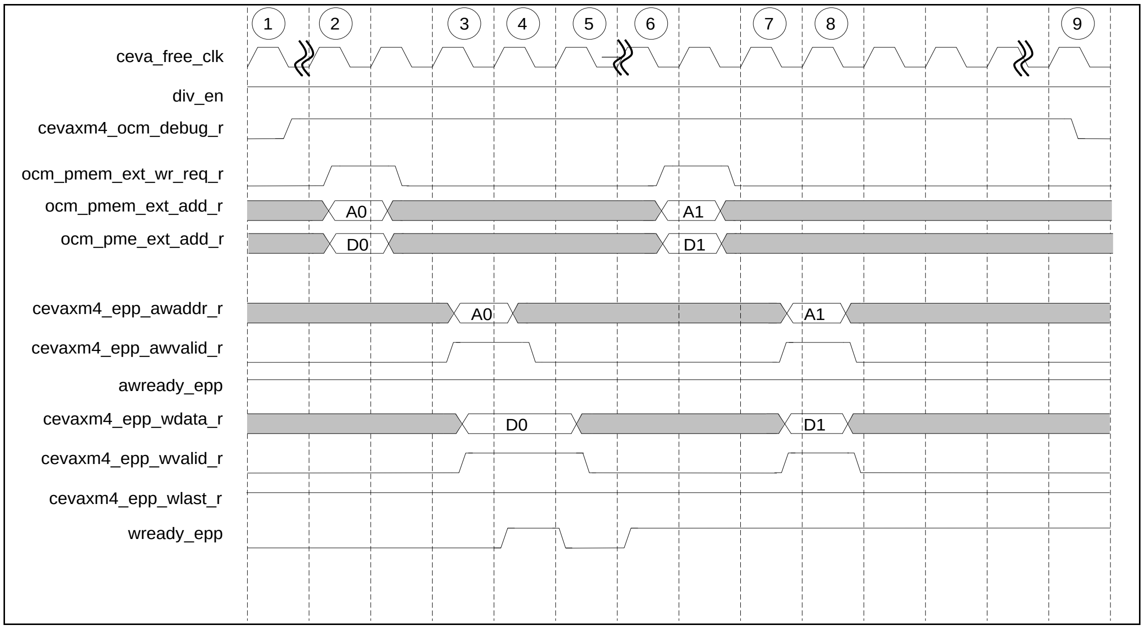

1
2
3
4
5
6
7
8
9
ceva_free_clk
div_en
cevaxm4_ocm_debug_r
ocm_pmem_ext_wr_req_r
ocm_pmem_ext_add_r
A0
A1
ocm_pme_ext_add_r
D0
D1
cevaxm4_epp_awaddr_r
A0
A1
cevaxm4_epp_awvalid_r
awready_epp
cevaxm4_epp_wdata_r
D0
D1
cevaxm4_epp_wvalid_r
cevaxm4_epp_wlast_r
wready_epp

Figure 17-8: EPP OCEM Write Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The core is moved to Debug mode.
2. **Cycle 2**: An OCEM external write request is issued towards the EPP.
3. **Cycle 3**: Address A0 and Data D0 are issued on the EPP AXI bus.
4. **Cycle 4**: Address A0 is accepted by the AXI slave.
5. **Cycle 5**: Data D0 is accepted by the AXI slave.
6. **Cycle 6**: An OCEM external write request is issued towards the EPP.
7. **Cycle 7**: Address A1 and Data D1 are issued on the EPP AXI bus.
8. **Cycle 8**: Address A1 and Data D1 are accepted by the AXI slave.
9. **Cycle 9**: The core exits Debug mode.

Table 17-4: EPP OCEM Write Signal Description

Name	Description
ceva_free_clk	Free clock
div_en	AXI clock enable
cevaxm4_ocm_debug_r	Core in debug mode indication
ocm_pmem_ext_wr_req_r	External write request from OCEM to EPP
ocm_pmem_ext_add_r	External write request address from OCEM to EPP
ocm_pmem_ext_data_r	External write request data from OCEM to EPP
cevaxm4_epp_awaddr_r	AXI write address channel address
cevaxm4_epp_awvalid_r	AXI write address channel valid
awready_epp	AXI write address channel ready
cevaxm4_epp_wdata_r	AXI WDC data
cevaxm4_epp_wvalid_r	AXI WDC valid
cevaxm4_epp_wlast_r	AXI WDC last
wready_epp	AXI WDC ready

17.3 External Master Ports (EDP/AXIM0/AXIM1)

There are up to three external AXI Master ports in the CEVA-XM4. The EDP, AXIM0, and AXIM1 all have the same behavior for the timing of read and write transactions through them. While the figures in the following sections refer to the EDP, they are also applicable to the AXIM0 and AXIM1 as well.

The external masters have an integral AXI Master port that can perform simultaneous read and write transactions. LS0, LS1, DMAN, and the DMA request read transactions, and the WB, DMAN, and DMA request write transactions. Each individual source has a unique AXI ID that allows read data to be returned out of order.

Table 17-5 shows which external master port AXI signals have fixed values and the value of those outputs.

Table 17-5: EDP AXI Fixed Signals

AXI Signal Name	Value	Description
BREADY	1’b1	The EDP is always ready to accept write response for writes it has issued.

For more details about the AMBA AXI interface, see the *AMBA AXI and ACE Protocol Issue D Specification*.

The following sections describe the access latency of each type of master.

17.3.1 EDP DSP Read

Figure 17-9 shows a DSP read after a read transaction from the core to the EDP. Each single DSP read access from the EDP causes a latency of four cycles; however, the EDP can process three transactions per LS interface (for a total of six). They are non-blocking, which means that wait will be de-asserted to the core when the first requested data is received by the EDP from the AXI bus. Core wait will subsequently be asserted again if the next requested data is not yet received from the AXI bus.

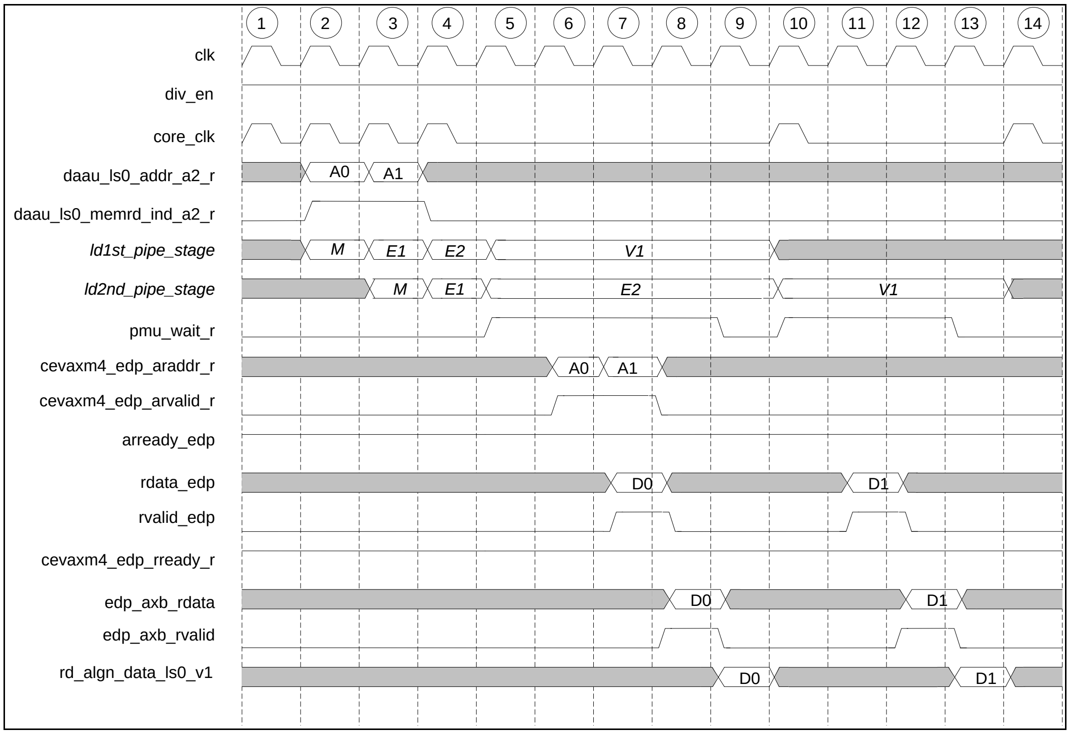

1
2
3
4
5
6
7
8
9
10
11
12
13
14
clk
div_en
core_clk
daau_ls0_addr_a2_r
A0
A1
daau_ls0_memrd_ind_a2_r
ld1st_pipe_stage
M
E1
E2
V1
ld2nd_pipe_stage
M
E1
E2
V1
pmu_wait_r
cevaxm4_edp_araddr_r
A0
A1
cevaxm4_edp_arvalid_r
arready_edp
rdata_edp
D0
D1
rvalid_edp
cevaxm4_edp_rready_r
edp_axb_rdata
D0
D1
edp_axb_rvalid
rd_algn_data_ls0_v1
D0
D1

Figure 17-9: EDP DSP Read Timing Diagram

The transaction sequence is as follows:

1. **Cycle 2**: The core issues a read request to address A0.
2. **Cycle 3**: The core issues a second read request to address A1.
3. **Cycle 5**: A wait is asserted to the core.
4. **Cycle 6**: The EDP puts the first address request on the AXI RAC.
5. **Cycle 7**: The EDP puts the second address on AXI RAC, and the AXI slave returns read data for the first request on the RDC.
6. **Cycle 8**: The EDP sends read data to the read buffer.
7. **Cycle 9**: A wait is de-asserted to the core, and read data is captured by core.
8. **Cycle 10**: A wait is re-asserted for the second read request.

9. **Cycle 11**: The AXI slave returns read data for the second request on the RDC.
10. **Cycle 12**: The EDP sends read data to the read buffer.
11. **Cycle 13**: A wait is de-asserted to the core, and read data is captured by the core.

Table 17-6: EDP DSP Read Signal Description

Name	Description
clk	Free clock
core_clk	DSP clock
div_en	AXI clock enable
daau_ls0_addr_a2_r	DSP LS0 read address
daau_ls0_memrd_ind_a2_r	DSP LS0 read indication
ld1st_pipe_stage	First load instruction pipe stage ***Note**: This is not a signal in the design; it is only a reference point for descriptive purposes.*
ld2nd_pipe_stage	Second load instruction pipe stage ***Note**: This is not a signal in the design; it is only a reference point for descriptive purposes.*
pmu_wait_r	Wait signal to core
cevaxm4_edp_araddr_r	AXI RAC address
cevaxm4_edp_arvalid_r	AXI RAC valid
arready_edp	AXI RAC ready
rdata_edp	AXI RDC data
rvalid_edp	AXI RDC valid
cevaxm4_edp_rready_r	AXI RDC ready
edp_axb_rdata	EDP read data to read buffer
edp_axb_rvalid	EDP read data valid
rd_algn_data_ls0_v1	Read data to core

17.3.2 EDP DSP Read (ECC on AXI)

When the AMBA bus ECC is enabled, there is one extra cycle of latency when performing an external read using the EDP/AXIM0/AXIM1, as shown in Figure 17-10.

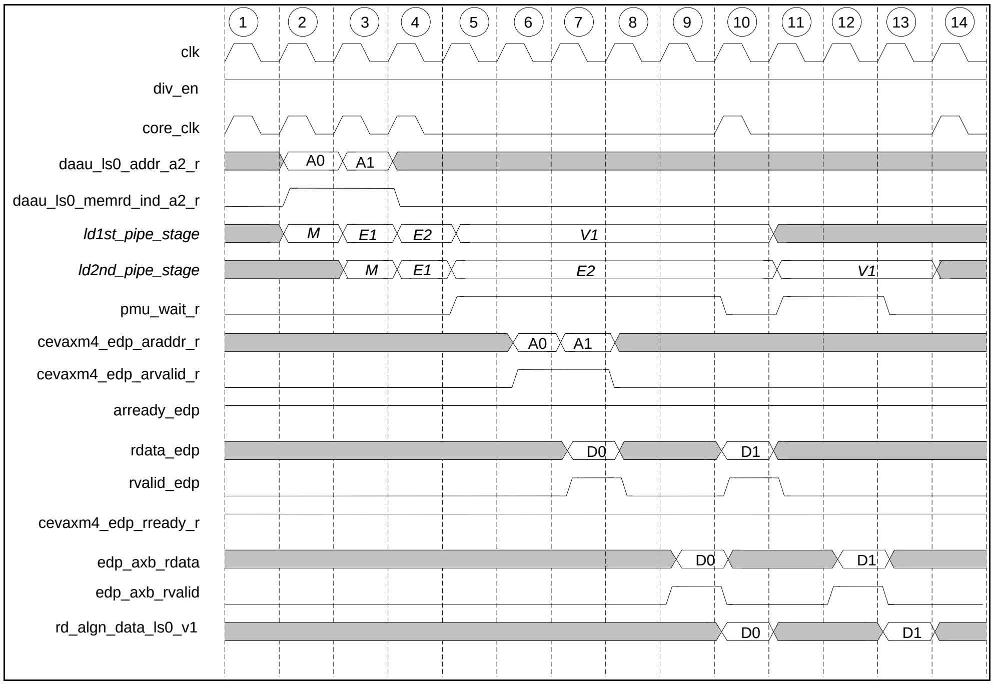

1
2
3
4
5
6
7
8
9
10
11
12
13
14
clk
div_en
core_clk
daau_ls0_addr_a2_r
A0
A1
daau_ls0_memrd_ind_a2_r
ld1st_pipe_stage
M
E1
E2
V1
ld2nd_pipe_stage
M
E1
E2
V1
pmu_wait_r
cevaxm4_edp_araddr_r
A0
A1
cevaxm4_edp_arvalid_r
arready_edp
rdata_edp
D0
D1
rvalid_edp
cevaxm4_edp_rready_r
edp_axb_rdata
D0
D1
edp_axb_rvalid
rd_algn_data_ls0_v1
D0
D1

Figure 17-10: EDP DSP Read Transaction (ECC on AXI) Timing Diagram

17.3.3 EDP DSP Write Buffer Write Access

Figure 17-11 shows two DSP write transactions through the EOS to the EDP. The DSP write access to the EDP has no latency other than the write buffer and EOS latency.

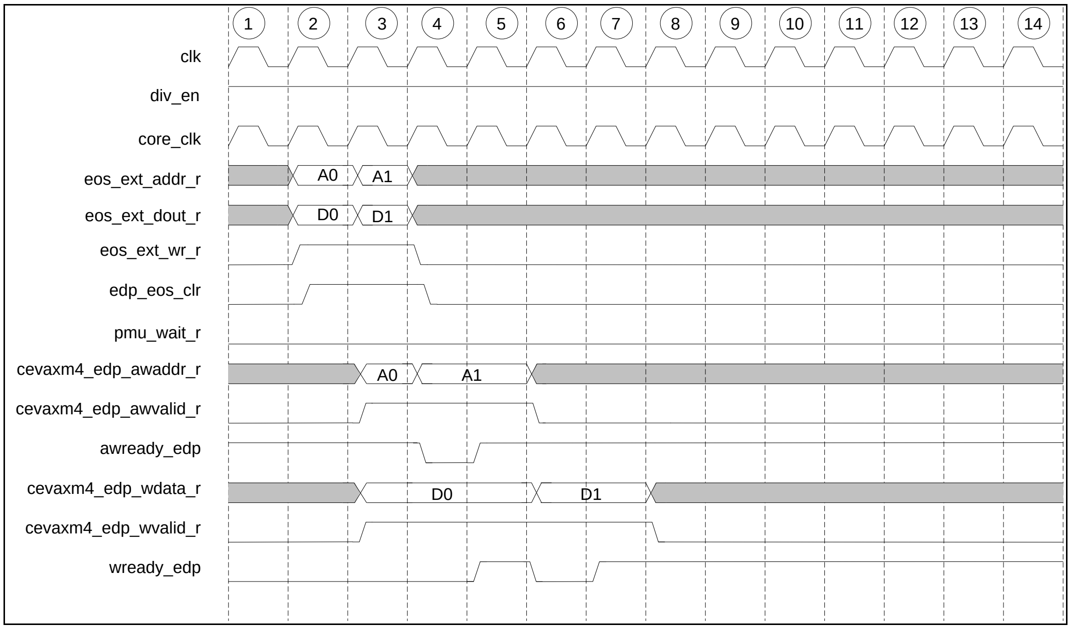

1
2
3
4
5
6
7
8
9
10
11
12
13
14
clk
div_en
core_clk
eos_ext_addr_r
A0
A1
eos_ext_dout_r
D0
D1
eos_ext_wr_r
edp_eos_clr
pmu_wait_r
cevaxm4_edp_awaddr_r
A0
A1
cevaxm4_edp_awvalid_r
awready_edp
cevaxm4_edp_wdata_r
D0
D1
cevaxm4_edp_wvalid_r
wready_edp

Figure 17-11: EDP DSP Write Buffer Write Access Timing Diagram

The transaction sequence is as follows:

1. **Cycle 2**: The EOS issues the first write request, and *clr* is asserted to the EOS.
2. **Cycle 3**:
   - 2.a The EDP issues the first write transaction on the WAC and WDC.
   - 2.b The WAC ready is high, indicating that the AXI slave accepts the write address.
   - 2.c The WDC ready is low, indicating that the AXI slave cannot accept the write data yet.
   - 2.d The EOS issues the second write request, and *clr* is asserted to the EOS.

3. **Cycle 4**:
    - 3.a The EDP issues the second write transaction on the WAC.
    - 3.b The WAC ready is low, indicating that the AXI slave cannot accept the write address yet.
4. **Cycle 5**:
    - 4.a The WAC ready is high, indicating that the AXI slave has accepted the second write address.
    - 4.b The WDC ready is high, indicating that the AXI slave has accepted the first write data.
5. **Cycle 6**:
    - 5.a The EDP issues the second write data on the WDC.
    - 5.b The WDC ready is low, indicating that the AXI slave cannot accept the write data yet.
6. **Cycle 7**: The WDC ready is high, and write data is accepted.

Table 17-7: EDP DSP Write Buffer Write Access Signal Description

Name	Description
clk	Free clock
core_clk	DSP clock
div_en	AXI clock enable
eos_ext_addr_r	External write address from EOS
eos_ext_wr_r	External write indication
edp_eos_clr	EDP clearing EOS indication
pmu_wait_r	Wait signal to core
cevaxm4_edp_awaddr_r	AXI WAC address
cevaxm4_edp_awvalid_r	AXI WAC valid
awready_edp	AXI WAC Ready
cevaxm4_edp_wdata_r	AXI WDC data
cevaxm4_edp_wvalid_r	AXI WDC valid
wready_edp	AXI WDC ready

17.3.4 EDP DSP Write with Slower AXI and Ready

Figure 17-12 shows two DSP write transaction through the EOS to the EDP. The DSP write access to the EDP has no latency other than the write buffer and EOS latency.

The AXI clock is half of the frequency of the core clock. The *div_en* signal is asserted one core clock cycle before the rising edge of the AXI clock. All of the AXI signals are launched and captured on the clock's rising edge when *div_en* is high.

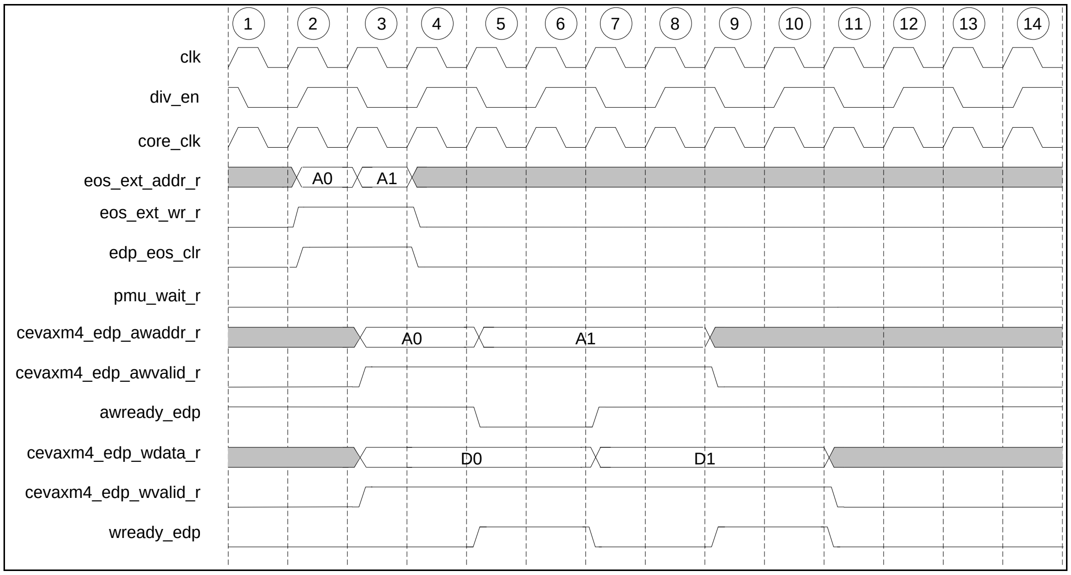

1
2
3
4
5
6
7
8
9
10
11
12
13
14
clk
div_en
core_clk
eos_ext_addr_r
A0
A1
eos_ext_wr_r
edp_eos_clr
pmu_wait_r
cevaxm4_edp_awaddr_r
A0
A1
cevaxm4_edp_awvalid_r
awready_edp
cevaxm4_edp_wdata_r
D0
D1
cevaxm4_edp_wvalid_r
wready_edp

Figure 17-12: EDP DSP Write with Slower AXI and Ready Timing Diagram

17.3.5 EDP Wide Non-Aligned Read

The CEVA-XM4 supports unaligned reads and writes. To enable support for unaligned writes and reads to/from the external data memory, the EDP splits any unaligned access into two or more separate word transactions.

Figure 17-13 shows a read of width 512 bits (from a *VPOP* instruction), which is reading from an unaligned address. An unaligned address in this case is any address in which the six LSBs are not zero. The AXI port size is 128 bits.

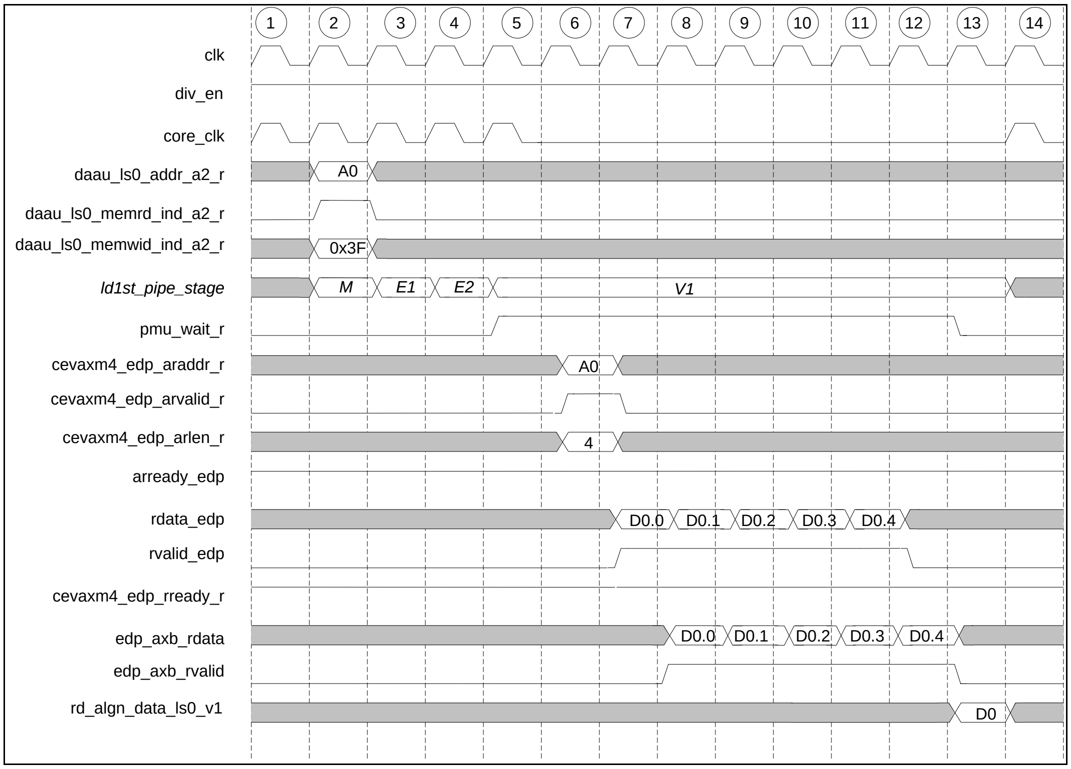

1
2
3
4
5
6
7
8
9
10
11
12
13
14
clk
div_en
core_clk
daau_ls0_addr_a2_r
A0
daau_ls0_memrd_ind_a2_r
daau_ls0_memwid_ind_a2_r
0x3F
ld1st_pipe_stage
M
E1
E2
V1
pmu_wait_r
cevaxm4_edp_araddr_r
A0
cevaxm4_edp_arvalid_r
cevaxm4_edp_arlen_r
4
arready_edp
rdata_edp
D0.0
D0.1
D0.2
D0.3
D0.4
rvalid_edp
cevaxm4_edp_rready_r
edp_axb_rdata
D0.0
D0.1
D0.2
D0.3
D0.4
edp_axb_rvalid
rd_algn_data_ls0_v1
D0

Figure 17-13: EDP Wide Non-Aligned Read Timing Diagram

The transaction sequence is as follows:

1. **Cycle 2**: The core issues an LS0 read request to the EDP.
2. **Cycle 5**:
   - 2.a The EDP arbitrates the request and determines that a burst of five transactions is required because the port size is configured to 128 bits and the access is unaligned.
   - 2.b A wait is asserted to the core, and the burst request is placed on the RAC.
3. **Cycle 7**: The AXI slave returns the first beat of the burst read data.
4. **Cycle 8**: The AXI slave returns the second beat of the burst read data, and the EDP sends the first data part to the LS0 read buffer.

5. **Cycle 9**: The AXI slave returns the third beat of the burst read data, and the EDP sends the second data part to the LS0 read buffer.
6. **Cycle 10**: The AXI slave returns the fourth beat of the burst read data, and the EDP sends the third data part to the LS0 read buffer.
7. **Cycle 11**: The AXI slave returns the final beat of the burst read data, and the EDP sends the fourth data part to the LS0 read buffer.
8. **Cycle 12**: The EDP sends the final data part to the LS0 read buffer.
9. **Cycle 13**: A wait is de-asserted to the core, and the core captures read data.

17.3.6 EDP Write with Sparse Write Strobes

The CEVA-XM4 supports the ability to store individual bytes from a vector register to the external memory using sparse write strobes. These sparse write strobes are caused by using the vector predicate registers.

Figure 17-14 shows a vector write where most of the vector predicate bits are zero and the store address is unaligned. The EDP converts this into a burst of three writes and uses the write strobes set to zero for the gap.

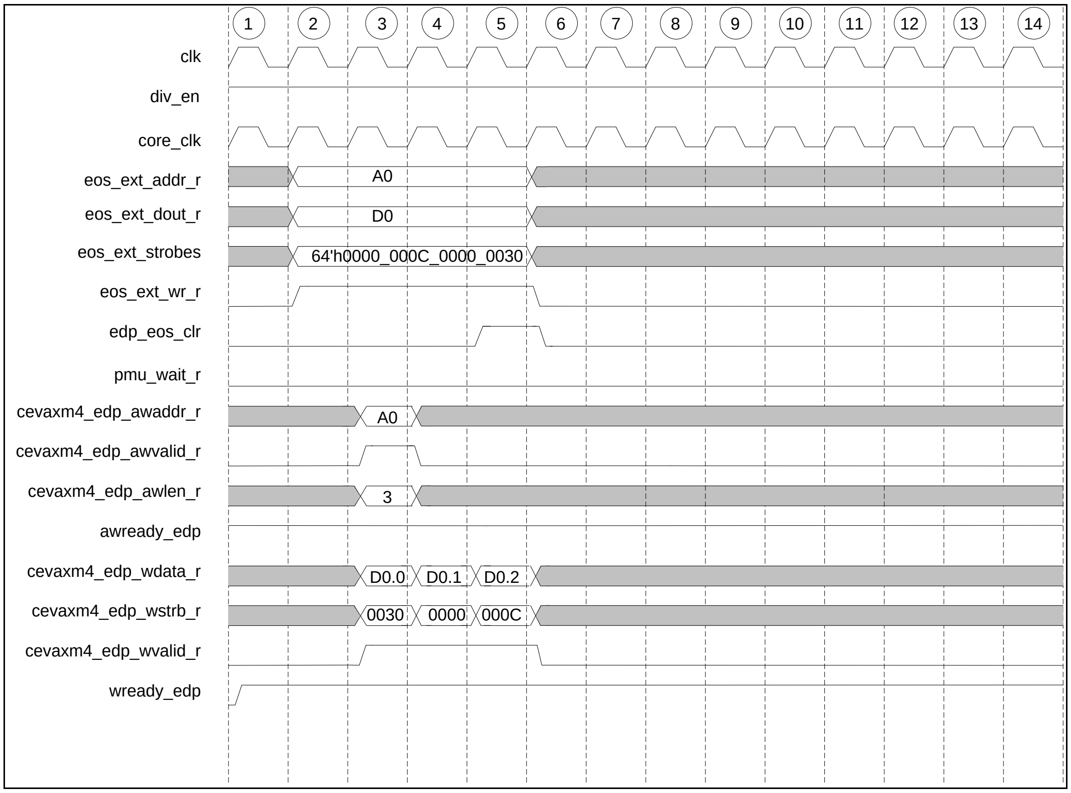

1
2
3
4
5
6
7
8
9
10
11
12
13
14
clk
div_en
core_clk
eos_ext_addr_r
A0
eos_ext_dout_r
D0
eos_ext_strobes
64'h0000_000C_0000_0030
eos_ext_wr_r
edp_eos_clr
pmu_wait_r
cevaxm4_edp_awaddr_r
A0
cevaxm4_edp_awvalid_r
cevaxm4_edp_awlen_r
3
awready_edp
cevaxm4_edp_wdata_r
D0.0
D0.1
D0.2
cevaxm4_edp_wstrb_r
0030
0000
000C
cevaxm4_edp_wvalid_r
wready_edp

Figure 17-14: EDP Write with Sparse Write Strobes Timing Diagram

The transaction sequence is as follows:

1. **Cycle 2**:
    - 1.a The EOS issues the 256-bit write with gaps caused by the vector predicates to the EDP.
    - 1.b The EDP does not assert the clear because it is a wide write (that is, greater than the AXI port width).
2. **Cycle 3**:
    - 2.a The EDP puts the write address, burst length, and so on on the WAC.

      The WAC ready is high, so it is accepted with no delay.
    - 2.b The EDP puts the first write data component on the WDC.

      The WDC ready is high, so the data is accepted with no delay.
3. **Cycle 4**: The EDP puts the second write data component on the WDC. The strobes for this component are held low (invalid data), so this data is ignored by the AXI slave.
4. **Cycle 5**: The EDP puts the third write data component on the WDC. Because this write data is valid, the strobes are set.

17.3.7 DDMA Queue Non-Automatic Mode

The DDMA queue can be configured with up to three pending transfers at a time. When the QAUTO bit in the **MSS_DDQS** register is set, the DDMA only starts a pending transfer upon detection of the rising edge of the *next_ddma* input.

In the example shown in Figure 17-15, the *next_ddma* input is treated as asynchronous and is synchronized inside the DDMA. The DDMA clock is turned off because it is in DPS mode and is awaiting the *next_ddma* input to start the next transaction in the DDMA queue. The *next_ddma* signal must be at least one core clock cycle wide. Only the rising edge advances the DDMA to the next queue entry.

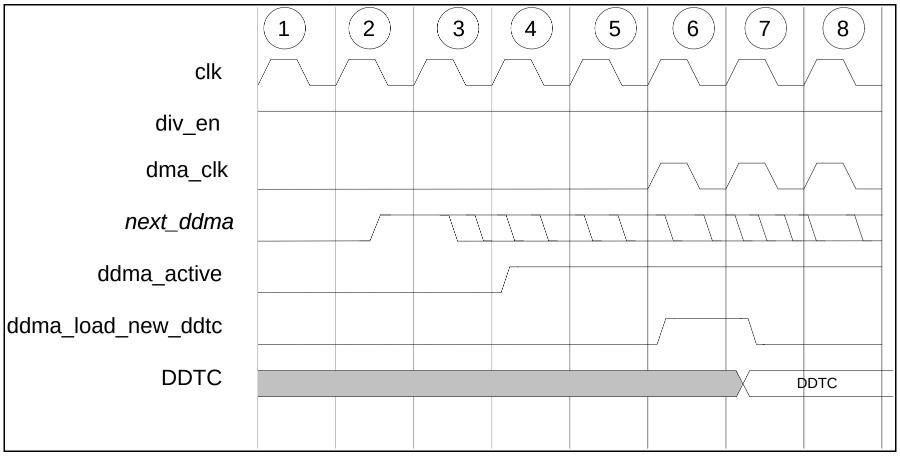


Figure 17-15: DDMA next_ddma Timing Diagram

Table 17-8: DDMA next_ddma Signal Description

Name	Description
clk	Free clock
div_en	AXI clock enable
dma_clk	DMA clock
next_ddma	Asynchronous input from external that advances DDMA queue
ddma_active	DMA active indication to PSU
ddma_load_new_ddtc	Loads next transfer from queue to DDTC
DDTC	Data DMA Transfer Count register

17.3.8 Read-after-Write Considerations

Because of the pipeline architecture nature of the CEVA-XM4, a write followed closely by a read requires special consideration.

For external read-after-write (RAW) scenarios, the read is delayed until the write has been completed, and then the read is allowed to progress. During the delay, the core is held in wait. In Protected RAW (PRAW) mode (default), the CEVA-XM4 waits until a write response has been received for the matching write transaction before releasing the matching read transaction to the AXI bus. This ensures that the read data is always the last data written to that location.

This protection is only between LS reads and writes. There is no protection for DMA reads following LS writes and vice versa. It is the user’s responsibility to ensure that the DMA reads and writes to the internal and external memories do not access memory locations that are currently being accessed by LS reads and writes.

In addition, when the DDMA queue is used and the DDMA is configured to be in automatic mode, it is possible that when a DDMA download is started (automatically) following a DDMA upload, delays inside the AXI Matrix could mean that external memory locations common to both upload and download could be read before being written. It is the user’s responsibility to ensure that RAW transactions issued on the AXI to the same address are completed in order.

17.3.9 Data DMA Debug Match

The data DMA can be configured so that a debug match indication is output when a configured address matches the address of the current DMA transaction. A match can be configured on either an internal TCM address or an external AXI addresses. It can also be configured for either DMA download or DMA upload. For more details about configuring the DMA debug match indication, see the *CEVA-XM4 Architecture Specification Volume III (MSS)*.

The DMA debug match indication is an output at the top level of the CEVA-XM4 where it is called *cevaxm4_ddma_dbg_match_r*. There is an acknowledge input called *ext_ddma_dbg_match_ack* that can be used to acknowledge and clear down the debug match indication. Because this input is assumed to be asynchronous, it is synchronized to the core clock inside the CEVA-XM4.

The DMA debug match indication also goes to the OCEM and is connected to the SA_EXT4 Stand-Alone External Breakpoint 4 input. This means that the DMA debug match can trigger a breakpoint in the OCEM, if the OCEM is configured to accept that external breakpoint. If configured, the OCEM acknowledges the DMA debug match indication automatically. For more details about the configuration registers in the OCEM, see the *CEVA-XM4_On_Chip_Emulation_Ref* in the release package.

The DMA address match is triggered when the actual read or write has been completed. For example, if triggering on the internal address during a DMA download, the DMA can perform the memory write over several cycles because of contention issues with higher priority sources on one or more banks. In this case, the DMA debug match indication is not asserted until the DMA has completed the entire write to all banks.

Figure 17-16 shows the external debug match indication, the external acknowledge, and the OCEM acknowledge.

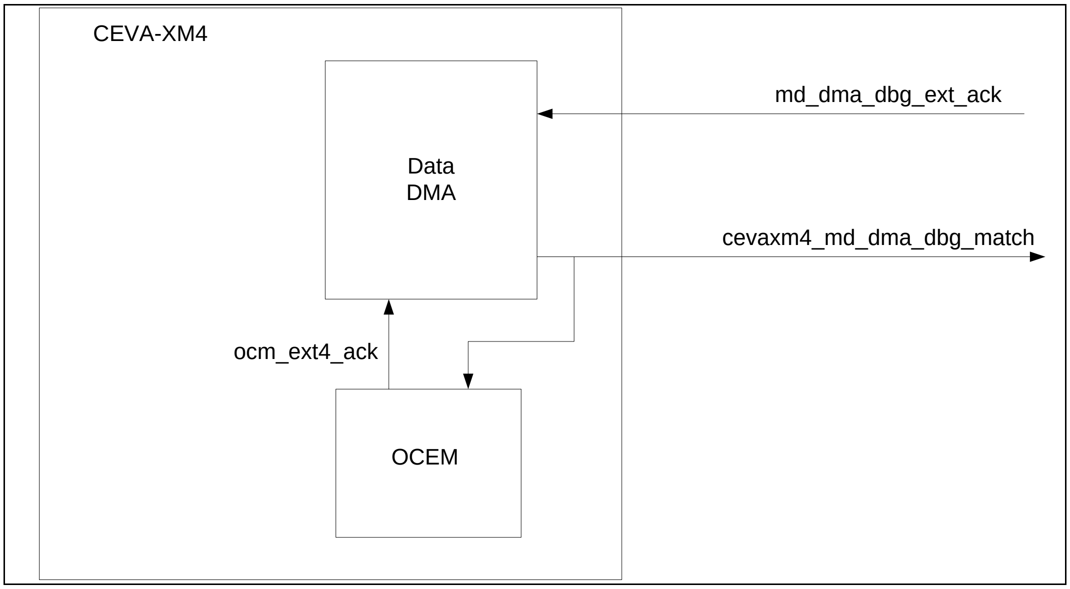


Figure 17-16: DMA Debug Match Indication

Figure 17-17 shows an example of a debug match indication and the subsequent acknowledgements.

Note: *The external acknowledge is synchronized first* ***before*** *being used to clear the match indication.*

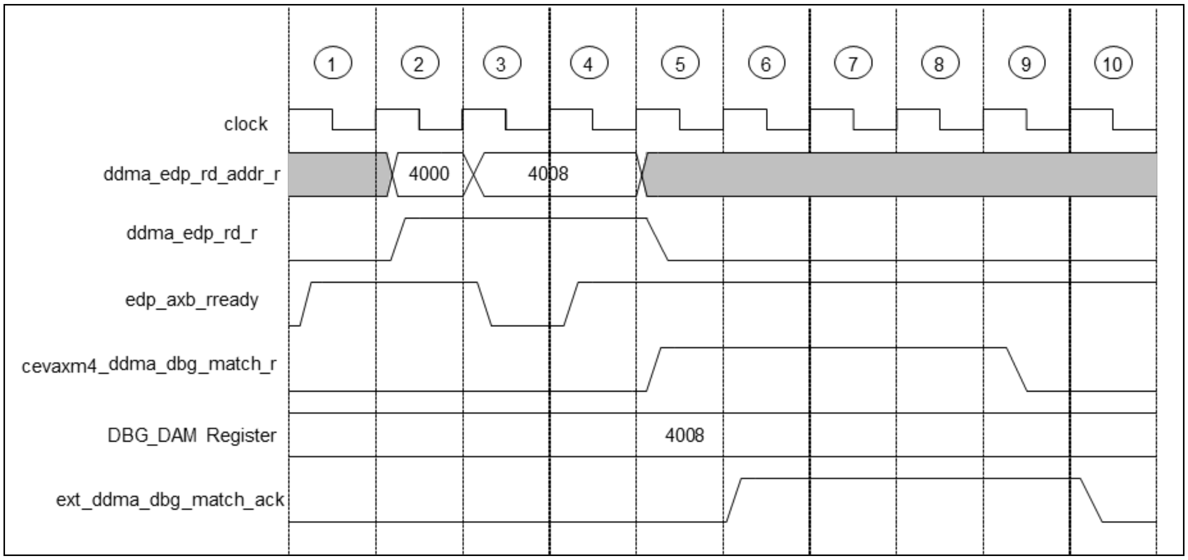

1
2
3
4
5
6
7
8
9
10
clock
ddma_edp_rd_addr_r
4000
4008
ddma_edp_rd_r
edp_axb_rready
cevaxm4_ddma_dbg_match_r
DBG_DAM Register
4008
ext_ddma_dbg_match_ack

Figure 17-17: DMA Debug Match Indication Timing Diagram

17.4 AXI Slave Port (EDAP/AXIsX)

Up to four AXI slave ports are implemented in the MSS, EDAP, and AXI slave ports (AXIs0, AXIs1, and AXIs2).

The slave port allows external devices to access the internal memory for read or write transactions. All slave ports have the same behavior for the timing of read and write transactions. A read transaction of any burst starts with a seven-core-cycle wait-state (assuming that the core clock is equal to the system clock) and continues with no penalty (assuming that the internal arbiter is not occupied with other transactions). When the AMBA bus ECC is enabled, there is one extra cycle of latency when performing a read.

A write transaction is performed with no penalty (assuming that the internal arbiter is not accessed with other transactions).

17.4.1 EDAP Read Single Followed by Read Burst

Figure 17-18 shows an EDAP read single followed by a read burst of six beats.

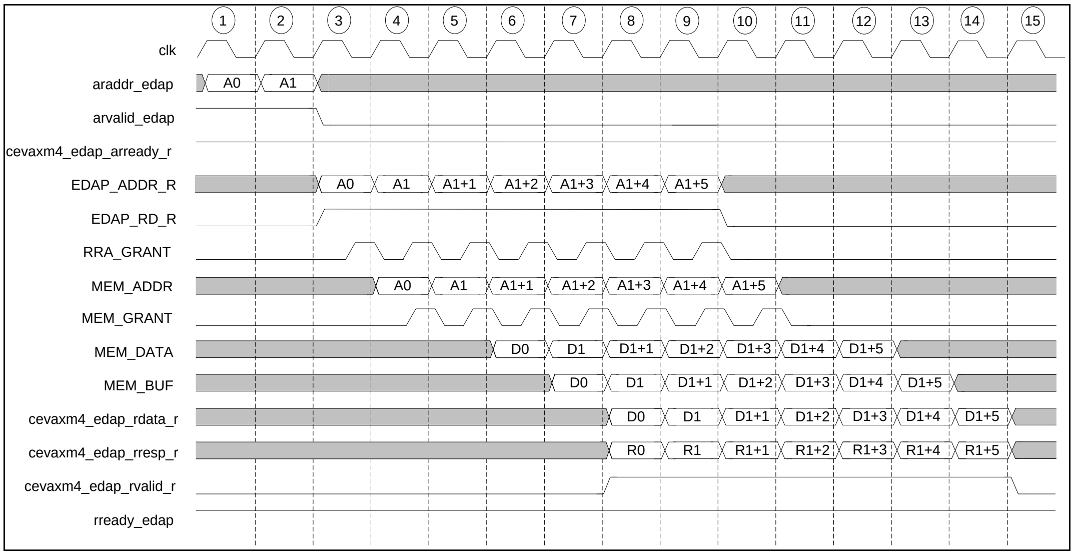

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
clk
araddr_edap
arvalid_edap
cevaxm4_edap_arready_r
EDAP_ADDR_R
EDAP_RD_R
RRA_GRANT
MEM_ADDR
MEM_GRANT
MEM_DATA
MEM_BUF
cevaxm4_edap_rdata_r
cevaxm4_edap_rresp_r
cevaxm4_edap_rvalid_r
rready_edap
A0
A1
A1+1
A1+2
A1+3
A1+4
A1+5
D0
D1
D1+1
D1+2
D1+3
D1+4
D1+5
R0
R1
R1+1
R1+2
R1+3
R1+4
R1+5

Figure 17-18: EDAP Read Single Followed by Read Burst Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The external Master issues a single read transaction.
2. **Cycle 2**: The external Master issues a burst read transaction.
3. **Cycle 3**: The EDAP requests the round-robin arbiter.
4. **Cycle 4**: The EDAP requests data from the memory junction box.
5. **Cycle 6**: Read data is returned by the TCM.
6. **Cycle 8**: Read data and response for the single transaction on the RDC.
7. **Cycles 9–14**: Read data and response for the burst transaction on the RDC.

Table 17-9: EDAP Read Single Followed by Read Burst Signal Description

Name	Description
clk	Free clock
araddr_edap	AXI Master read address bus
arvalid_edap	AXI Master read address valid
cevaxm4_edap_arready_r	AXI slave read address ready
EDAP_ADDR_R	EDAP address to round-robin arbiter
EDAP_RD_R	EDAP read request to round-robin arbiter
RRA_GRANT	Round-robin arbiter grant
MEM_ADDR	TCM address bus
MEM_GRANT	TCM access grant
MEM_DATA	TCM read data bus
MEM_BUF	Internal memory buffer
cevaxm4_edap_rdata_r	AXI slave read data bus
cevaxm4_edap_rresp_r	AXI slave read response bus
cevaxm4_edap_rvalid_r	AXI slave read data valid
rready_edap	AXI Master read ready

17.4.2 EDAP Write Single and Burst

Figure 17-19 shows an EDAP write single followed by a write burst of six beats.

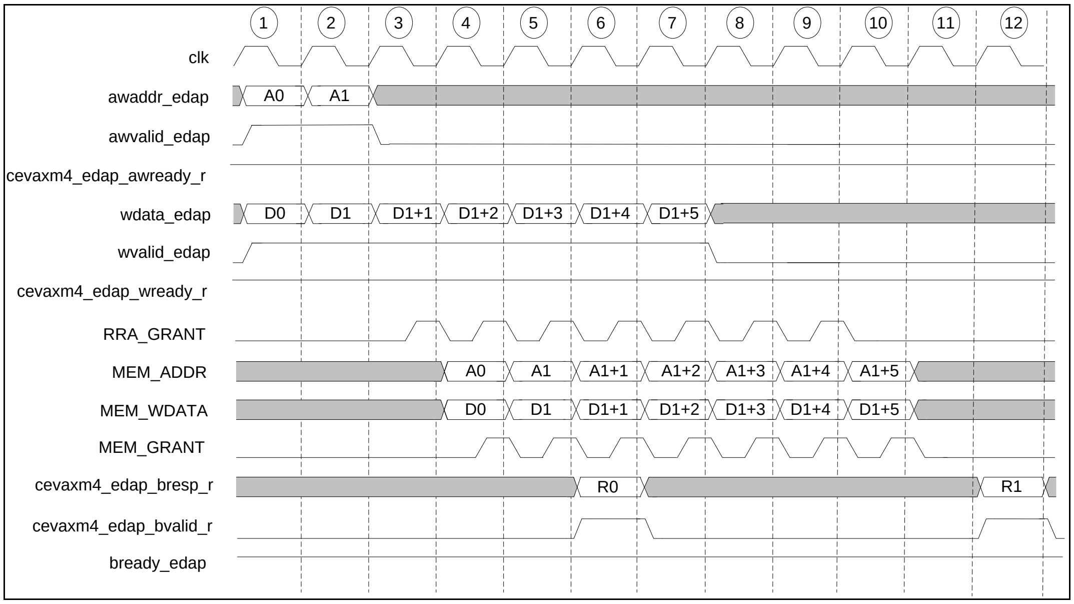

1
2
3
4
5
6
7
8
9
10
11
12
clk
awaddr_edap
A0
A1
awvalid_edap
cevaxm4_edap_awready_r
wdata_edap
D0
D1
D1+1
D1+2
D1+3
D1+4
D1+5
wvalid_edap
cevaxm4_edap_wready_r
RRA_GRANT
MEM_ADDR
A0
A1
A1+1
A1+2
A1+3
A1+4
A1+5
MEM_WDATA
D0
D1
D1+1
D1+2
D1+3
D1+4
D1+5
MEM_GRANT
cevaxm4_edap_bresp_r
R0
R1
cevaxm4_edap_bvalid_r
bready_edap

Figure 17-19: EDAP Write Single and Burst Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The external Master issues a single write transaction.
2. **Cycle 2**: The external Master issues a burst write transaction.
3. **Cycle 3**: The EDAP requests the round-robin arbiter.
4. **Cycle 4**: Internal memory accesses start.
5. **Cycle 6**: Write response for single write on the WRC.
6. **Cycles 5-10**: Memory accesses continue with no contention issues.
7. **Cycle 12**: Burst response on the WRC.

Table 17-10: EDAP Write Single and Burst Signal Description

Name	Description
clk	Free clock
awaddr_edap	AXI Master write address bus
awvalid_edap	AXI Master write address valid
cevaxm4_edap_awready_r	AXI slave write address ready
wdata_edap	AXI Master write data bus
wvalid_edap	AXI Master write data valid
cevaxm4_edap_wready_r	AXI Master write data ready
RRA_GRANT	Round-robin arbiter grant
MEM_ADDR	TCM address bus
MEM_WDATA	TCM write data bus
MEM_GRANT	TCM access grant
cevaxm4_edap_bresp_r	AXI slave write response
cevaxm4_edap_bvalid_r	AXI slave write response valid
bready_edap	AXI Master response ready

17.4.3 EDAP Download to TCM after Reset with External Wait

The EDAP can be used to download data into the internal program and data memories after the core reset has been de-asserted but while the core is held in a wait-state because of the assertion of *external_wait*, as shown in Figure 17-20.

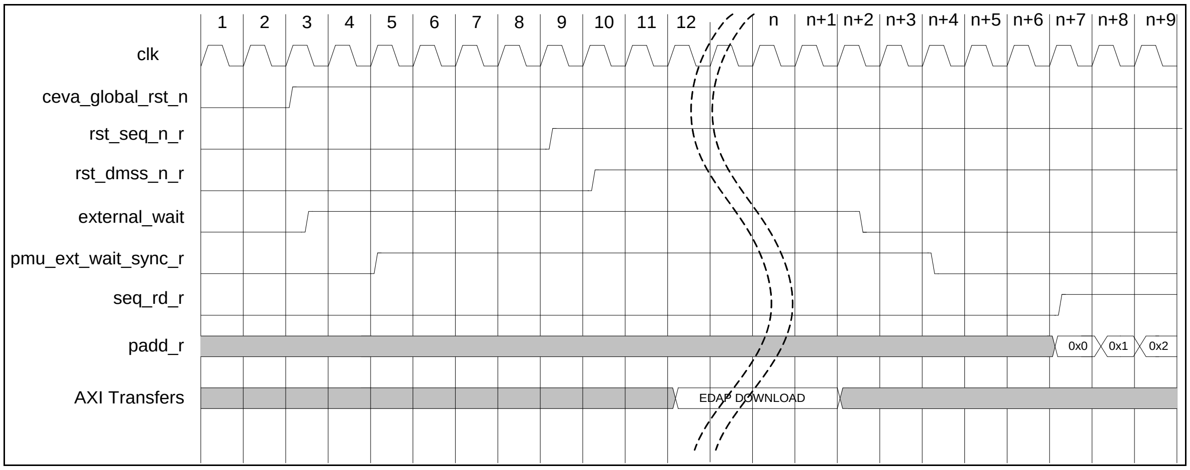

1
2
3
4
5
6
7
8
9
10
11
12
n
n+1
n+2
n+3
n+4
n+5
n+6
n+7
n+8
n+9
clk
ceva_global_rst_n
rst_seq_n_r
rst_dmss_n_r
external_wait
pmu_ext_wait_sync_r
seq_rd_r
padd_r
0x0
0x1
0x2
AXI Transfers
EDAP DOWNLOAD

Figure 17-20: EDAP Download to TCM after Reset with External Wait Timing Diagram

Table 17-11: External Device Access Port Read with Unequal Clocks

Name	Description
clk	Free clock
ceva_global_rst_n	Asynchronous reset to CEVA-XM4
rst_seq_n_r	Synchronized reset to Sequencer
rst_dmss_n_r	Synchronized reset to DMSS
external_wait	Asynchronous external wait input
pmu_ext_wait_sync_r	Synchronized external wait
seq_rd_r	Sequencer read strobe
padd_r	Program address
AXI Transfers	External AXI Master transfers to CEVA-XM4 EDAP AXI slave

17.5 I/O Port

The CEVA-XM4 uses the I/O port to access peripheral devices via the APB3 protocol. The APB3 port is a non-pipeline protocol, and can issue a new transaction only after the previous one has ended.

17.5.1 I/O Port Read

Figure 17-21 shows two I/O read transactions. Each access has two latency cycles.

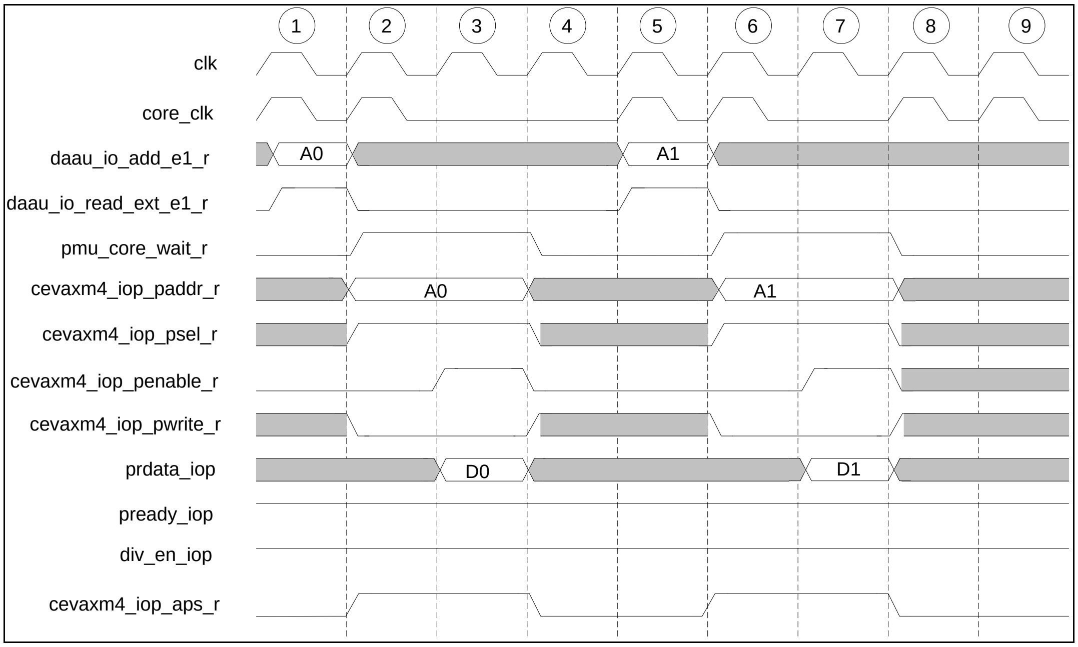

1
2
3
4
5
6
7
8
9
clk
core_clk
daau_io_add_e1_r
A0
A1
daau_io_read_ext_e1_r
pmu_core_wait_r
cevaxm4_iop_paddr_r
A0
A1
cevaxm4_iop_psel_r
cevaxm4_iop_penable_r
cevaxm4_iop_pwrite_r
prdata_iop
D0
D1
pready_iop
div_en_iop
cevaxm4_iop_aps_r

Figure 17-21: I/O Port Read Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The core issues the first I/O read request.
2. **Cycle 2**:
    - 2.a The I/O port asserts a wait.
    - 2.b The I/O port starts an APB3 read transaction.
    - 2.c The APB3 setup phase (first latency cycle) starts.
3. **Cycle 3**:
    - 3.a The APB3 access phase starts.
    - 3.b The external device returns data.
    - 3.c The APB3 read is complete (second latency cycle).
4. **Cycle 4**: The first read data is transferred to the core from the I/O port in V2, and a wait is de-asserted to the core.
5. **Cycle 5**: The core issues a second I/O read request.
6. **Cycle 6**: The I/O port asserts a wait for second read transaction, which proceeds with the same timing as the first.

Note: *The cevaxm4_iop_aps_r signal is asserted for the full duration of each APB transaction.*

Table 17-12: I/O Port Read Signal Description

Name	Description
clk	Free clock
core_clk	DSP clock
daau_io_add_e1_r[31:0]	DSP I/O read address
daau_io_read_ext_e1_r	DSP I/O read indication
pmu_core_wair_r	Wait signal
cevaxm4_iop_paddr_r[31:0]	I/O port address
cevaxm4_iop_psel_r	I/O port select
cevaxm4_iop_penable_r	I/O port enable
cevaxm4_iop_pwrite_r	I/O port read/write indication
prdata_iop[31:0]	I/O port read data bus (datain)
pready_iop	I/O port ready indication
div_en_iop	I/O port clock enable
cevaxm4_iop_aps_r	I/O port automatic power save mode signal

17.5.2 I/O Port Write

Figure 17-22 shows two I/O write transactions. Each access has two latency cycles.

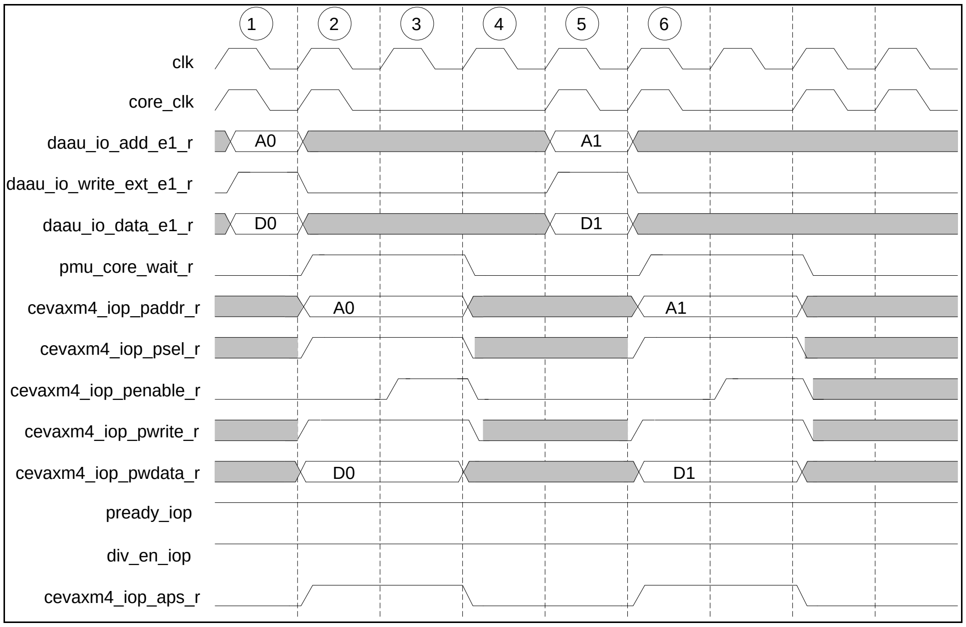

1
2
3
4
5
6
clk
core_clk
daau_io_add_e1_r
A0
A1
daau_io_write_ext_e1_r
daau_io_data_e1_r
D0
D1
pmu_core_wait_r
cevaxm4_iop_paddr_r
A0
A1
cevaxm4_iop_psel_r
cevaxm4_iop_penable_r
cevaxm4_iop_pwrite_r
cevaxm4_iop_pwdata_r
D0
D1
pready_iop
div_en_iop
cevaxm4_iop_aps_r

Figure 17-22: I/O Port Write Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The core issues the first I/O write request.
2. **Cycle 2**:
    - 2.a The I/O port asserts a wait.
    - 2.b The I/O port starts an APB3 write transaction.
    - 2.c The APB3 setup phase (first latency cycle) starts.
3. **Cycle 3**:
    - 3.a The APB3 access phase starts.
    - 3.b The APB3 write is complete (second latency cycle).
4. **Cycle 4**: A wait is de-asserted to the core.
5. **Cycle 5**: The core issues a second I/O write request.
6. **Cycle 6**: The I/O port asserts a wait for second write transaction, which proceeds with the same timing as the first.

Note: *The cevaxm4_iop_aps_r signal is asserted for the full duration of each APB transaction.*

Table 17-13: I/O Port Write Signal Description

Name	Description
clk	Free clock
core_clk	DSP clock
daau_io_add_e1_r[31:0]	DSP I/O write address
daau_io_write_ext_e1_r	DSP I/O write indication
daau_io_data_e1_r	DSP I/O write data
pmu_core_wair_r	Wait signal
cevaxm4_iop_paddr_r[31:0]	I/O port address
cevaxm4_iop_psel_r	I/O port select
cevaxm4_iop_penable_r	I/O port enable
cevaxm4_iop_pwrite_r	I/O port read/write indication
prdata_iop[31:0]	I/O port write data bus (dataout)
pready_iop	I/O port ready indication
div_en_iop	I/O port clock enable
cevaxm4_iop_aps_r	I/O port automatic power save mode signal

18. AXI Capabilities

The CEVA-XM4 has dedicated AXI interfaces that consist of two AXI Masters and up to four AXI slaves:

- **AXI Masters**:
  - EPP
  - EDP/AXIm0/AXIm1
- **AXI Slaves**: EDAP, AXIs0, AXIs1, and AXIs2

The AXI capabilities and behavior of each interface is independent of the other interfaces and is elaborated in the following sections.

18.1 EPP AXI Capabilities

Table 18-1 describes the AXI ID signals used by the EPP.

Table 18-1: EPP AXI IDs

Transaction Type	ID Description
Read	● Core Fetch ID is configurable via the CPM (default is 0x0). ● Program DMA ID is configurable via the CPM (default is 0x2). ● OCEM ID is configurable via the CPM (default is 0x7).
Write	OCEM ID is configurable via the CPM (default is 0x7).

18.1.1 Outstanding Transactions

Table 18-2 describes the maximum number of outstanding transactions initiated by the EPP.

Table 18-2: EPP AXI Master Outstanding Transactions

Transaction Type	Maximum Number	Description
Read	4	All IDs combined together. The maximum number of outstanding read transactions is four, of these: ● Up to four addresses can be sent by the EPP originating from core fetch or software pre-fetch ● Up to four addresses can be sent by the EPP originating from the PDMA ● Up to one address can be sent by the EPP originating from the OCEM The total number of outstanding software pre-fetch reads can be configured (the maximum number is four). The total number of outstanding DMA can be configured (the maximum number is four). The total number of outstanding reads from any source can be configured (the maximum number is four).
Write	1	Up to one address can be sent by the EPP originating from the OCEM
Total read and write	4	The maximum number of four transactions (read and write combined) can be sent by the EPP.

18.1.2 Interleaving / Out-of-Order Support

Table 18-3 describes the EPP write interleaving depth.

Table 18-3: EPP Write Interleaving Depth

AXI Attribute	Maximum Number	Description
Write interleaving depth	1	The EPP interleaves the write data to a depth of 1. Because the EPP sends only one outstanding write transaction, there is no relevance for interleaving.

Table 18-4 describes the EPP out-of-order support.

Table 18-4: EPP Out-of-Order Support

AXI Attribute	Support	Description
Out-of-order support in read transactions	Yes	The EPP can issue read transactions with two different IDs, one for PDMA and one for Core fetches or OCEM. The data can be returned in any order.

18.1.3 Supported AXI Signal Values

Table 18-5 describes the supported EPP AXI signal values.

Table 18-5: Supported EPP AXI Signal Values

AXI Signal Name	Possible Values	Description
cevaxm4_epp_arsize_r cevaxm4_epp_awsize_r	0b100	Transaction size is always 16 bytes.
cevaxm4_epp_arlock_r cevaxm4_epp_awlock_r	0b00	Transactions always use normal access.
cevaxm4_epp_arprot_r cevaxm4_epp_awprot_r	0b100	Transactions always use secure instruction access.
cevaxm4_epp_arcache_r cevaxm4_epp_awcache_r	0x0	Memory attribute signaling is always non-cacheable and non-bufferable.
cevaxm4_epp_arid_r	0x0-0xF	● Core Fetch ID is configurable via the CPM (default is 0x0). ● Program DMA ID is configurable via the CPM (default is 0x2). ● OCEM ID is configurable via the CPM (default is 0x7).

AXI Signal Name	Possible Values	Description
cevaxm4_epp_arlen_r	0x1, 0x3, 0x7, 0xF	● Core fetches use a burst length of two during non-cacheable access, and a burst length of four during cacheable access. ● Software pre-fetch uses a burst length of four. ● PDMA burst length is configurable via the CPM (default is 0x1) and can be 0x1, 0x3, 0x7, or 0xF. ● The OCEM always uses a burst length of two.
cevaxm4_epp_arburst_r	2’b01, 2’b10	Core cacheable reads and software pre-fetch use WRAP bursts. All other bursts are INCR.
cevaxm4_epp_awburst_r		EPP write bursts are always INCR.
cevaxm4_epp_awid_r cevaxm4_epp_wid_r	0x0-0xF	OCEM ID is configurable via the CPM (default is 0x7). The OCEM is the only initiator of EPP write transactions.
cevaxm4_epp_awlen_r	0x0	Write burst length is always a single transfer.
cevaxm4_epp_wstrb_r	0xffff	Write strobes are always 1.
cevaxm4_epp_bready_r	0b1	Write responses are always accepted by the EPP.
cevaxm4_epp_rready_r	0, 1	*rready* can be de-asserted when internal data buffers are full.

18.2 EDP AXI Capabilities

18.2.1 4KB Boundary

The DMSS does not allow bursts to cross the 4KB boundary. Instead, it splits the bursts into single transactions. The maximum number of singles that can result from splitting a transaction is five for reads and three for writes. This happens when performing wide reads (for example, *VPOP* (512 bits)) or wide writes (for example, *VPUSH* (256 bits)) from/to an unaligned address.

18.2.2 AXI IDs

The IDs are programmable. Table 18-6 describes the **default** AXI ID values used by the EDP.

Table 18-6: EDP AXI IDs

Transaction Type	ID Description
Read	0x8 = Used by LS0 0x9 = Used by LS1 0xA = Used by Data DMA 0xB = Used by DMA Manager
Write	0x1 = Used by WB 0xF = Used by Data DMA 0x3 = Used by DMA Manager
Read and write	0x7 = Used by OCEM

18.2.3 Outstanding Transactions

Table 18-7 describes the maximum number of outstanding transactions initiated by the EDP. **Outstanding** means that the transaction has been issued on to the AXI bus but no response has yet been received from the AXI slave.

Table 18-7: EDP AXI Master Outstanding Transactions

Source	Transaction Type	Maximum Number	Description
DDMA	Read	16	Up to 16 addresses can be sent by the EDP from the DDMA.
LS0	Read	15	Up to three addresses can be sent by the EDP from LS0, which if they cross the 4KB boundary, can each be split into up to five separate reads. This can potentially result in 15 read transactions.
LS1	Read	9	Up to three addresses can be sent by the EDP from LS1, which if they cross the 4KB boundary, can each be split into up to three separate reads. This can potentially result in nine read transactions.
DMAN	Read	2	Up to two addresses can be sent by the EDP from the DMA Manager.

Source	Transaction Type	Maximum Number	Description
DDMA	Write	32	Up to 32 addresses can be sent by the EDP from the DDMA.
DMAN	Write	1023	Up to 1023 addresses can be sent by the EDP from the DMA Manager.
LS0 and LS1	Write (PRAW OFF)	1023	In non-PRAW mode, up to 1023 addresses can be sent by the EDP from the write buffer.
LS0 and LS1	Write (PRAW ON)	12	In PRAW mode, up to four addresses can be sent by the EDP from the write buffer If all four cross the 4K boundary, then 12 individual writes will be generated.

Notes: *The maximum number of outstanding write transactions is 1023 regardless of their initiator. If after 1023 outstanding write transactions, another write transaction is started, a General Violation Indication (GVI) will be asserted.*

PRAW mode is configured by the ***MSS_DPRAW*** *register. For more details, see the CEVA-XM4 Architecture Specification Volume III (MSS).*

When Bus ECC configuration is selected, the maximum number of outstanding reads per RID is limited to two.

18.2.4 Interleaving / Out-of-Order Support

Table 18-8 describes the EDP write interleaving depth.

Table 18-8: EDP Write Interleaving Depth

AXI Attribute	Maximum Number	Description
Write interleaving depth	1	The EDP interleaves the write data to a depth of 1. This means that the EDP will not interleave data with two different IDs.

Table 18-9 describes the EDP out-of-order support.

Table 18-9: EDP Out-of-Order Support

AXI Attribute	Support	Description
Out-of-Order support in read transactions	Yes	The EDP can issue read transactions with four different IDs: LS0, LS1, Data DMA, and DMA Manager. The data can be returned in any order.

18.2.5 Supported AXI Signal Values

Table 18-10 describes the supported EDP AXI signal values.

Table 18-10: Supported EDP AXI Signal Values

AXI Signal Name	Possible Values	Description
cevaxm4_edp_arlen_r cevaxm4_edp_awlen_r	0-255	• LS0 and LS1 use burst lengths of 1-5 transfers. • Data DMA uses a burst length of 1-256. • DMA Manager uses a burst length of 1 only.
cevaxm4_edp_arsize_r cevaxm4_edp_awsize_r	0-4	• LS0 and LS1 use transaction sizes of 1-16 bytes. • Data DMA uses a transaction size of 1-16 bytes. • DMA Manager uses a transaction size of 16 bytes only.
cevaxm4_edp_arid_r	0x0-0xF	As configured in the **PORT_RID** register.
cevaxm4_edp_awid_r cevaxm4_edp_wid_r	0x0-0xF	As configured in the **PORT_WID** register.
cevaxm4_edp_arburst_r	0b00, 0b01	DMA uses INCR and FIXED bursts. All other bursts are INCR.

AXI Signal Name	Possible Values	Description
cevaxm4_edp_awburst_r	0b00, 0b01	DMA uses INCR and FIXED bursts. All other bursts are INCR.
cevaxm4_edp_arprot_r cevaxm4_edp_awprot_r	0x0	Transactions always use normal, secure, data access.
cevaxm4_edp_arlock_r cevaxm4_edp_awlock_r	0b0 0b1	Data DMA, DMA Manager, and LS0, LS1 non-exclusive access use 0 - normal access. LS0, LS1 use 1 for exclusive accesses.
cevaxm4_edp_arcache_r cevaxm4_edp_awcache_r	0x0-0xF	As per the DACU configuration.
cevaxm4_edp_wstrb_r	0x0-0xFFFF	● For full data width transfers (*awsize* = 3'b100) with a 128-bit AXI, the *cevaxm4_edp_wstrb_r* value is 16'hFFFF. ● For narrow transfers where the size is less than the size of the AXI data bus, the *wstrb* value is according to the address of the bytes being transferred, according to the AXI standard. The EDP can drive sparse write strobes.
cevaxm4_edp_bready_r	0b1	Write responses are always accepted by the EDP.
cevaxm4_edp_rready_r	0, 1	The EDP never de-asserts *RREADY* for LS reads. DDMA can de-assert *RREADY* for downloads.

18.3 AXI Master Port Capabilities

18.3.1 4KB Boundary

The DMSS does not allow bursts to cross the 4KB boundary. Instead, it splits the bursts into single transactions. The maximum number of singles that can result from splitting a transaction is five for reads and three for writes. This happens when performing wide reads (for example, *VPOP* (512 bits)) or wide writes (for example, *VPUSH* (256 bits)) from/to an unaligned address.

18.3.2 AXI IDs

The IDs are programmable. Table 18-11 describes the **default** AXI ID values used by AXI Master ports.

Table 18-11: AXI Master AXI IDs

Transaction Type	ID Description
Read	0x8 = Used by LS0 0x9 = Used by LS1 0xA = Used by Data DMA 0xB = Used by DMA Manager
Write	0x1 = Used by WB 0xF = Used by Data DMA 0x3 = Used by DMA Manager

18.3.3 Interleaving / Out-of-Order Support

Table 18-12 describes the AXI Master ports write interleaving depth.

Table 18-12: AXI Master Ports Write Interleaving Depth

AXI Attribute	Maximum Number	Description
Write interleaving depth	1	The AXI Master ports interleave the write data to a depth of 1. This means that the AXI Master ports will not interleave data with two different IDs.

Table 18-13 describes the AXI Master Ports out-of-order support.

Table 18-13: AXI Master Ports Out-of-Order Support

AXI Attribute	Support	Description
Out-of-order support in read transactions	Yes	The AXI Master ports can issue read transactions with four different IDs: LS0, LS1, Data DMA, and DMA Manager. The data can be returned in any order.

18.3.4 Supported AXI Signal Values

Table 18-14 describes the supported AXI Master ports AXI signal values.

Note: *The **X** in the signal name denotes either AXIm0 or AXIm1.*

Table 18-14: Supported AXI Master Ports AXI Signal Values

AXI Signal Name	Possible Values	Description
cevaxm4_aximX_arlen_r cevaxm4_aximX_awlen_r	0-256	● LS0 and LS1 use burst lengths of 1-5 transfers. ● Data DMA uses a burst length of 1-256. ● DMA Manager uses a burst length of 1 only.
cevaxm4_aximX_arsize_r cevaxm4_aximX_awsize_r	0-4 (128-bit wide port) or 0-5 (256-bit wide port)	● AXI Master port width =128 bits ○ LS0 and LS1 use transaction sizes of 1-16 bytes. ○ Data DMA uses a transaction size of 1-16 bytes. ○ DMA Manager uses a transaction size of 16 bytes only. ● AXI Master port width =256 bits ○ LS0 and LS1 use transaction sizes of 1-32 bytes. ○ Data DMA uses a transaction size of 1-32 bytes. ○ DMA Manager uses a transaction size of 32 bytes only.
cevaxm4_aximX_arid_r	0x0-0xF	As configured in the **PORT_RID** register.
cevaxm4_aximX_awid_r cevaxm4_aximX_wid_r	0x0-0xF	As configured in the **PORT_WID** register.
cevaxm4_aximX_arburst_r	0b00, 0b01	DMA uses INCR and FIXED bursts. All other bursts are INCR.
cevaxm4_aximX_awburst_r	0b00, 0b01	DMA uses INCR and FIXED bursts. All other bursts are INCR
cevaxm4_aximX_arprot_r cevaxm4_aximX_awprot_r	0x0	Transactions always use normal, secure, data access.
cevaxm4_aximX_arlock_r cevaxm4_aximX_awlock_r	0b0	Data DMA, DMA Manager, and LS0, LS1 access use 0 - normal access.

AXI Signal Name	Possible Values	Description
cevaxm4_aximX_arcache_r cevaxm4_aximX_awcache_r	0x0-0xF	As per the DACU configuration.
cevaxm4_aximX_wstrb_r	0x0-0xFFFF (128- bit wide port) or 0x0-0XFFFFFFFF (256-bit wide port)	The AXI Master drives the *WSTRB* according to the address and size of the transfer. The AXI Master can drive sparse write strobes.
cevaxm4_aximX_bready_r	0b1	Write responses are always accepted by the AXI Master ports.
cevaxm4_aximX_rready_r	0, 1	The AXI Master ports never de-assert *RREADY* for LS reads. DDMA can de-assert *RREADY* for downloads.

18.4 AXI Slave Port Capabilities

18.4.1 Outstanding Transactions

Table 18-15 describes the maximum number of outstanding transactions accepted by the AXI slave port.

Table 18-15: AXI Slave Port Outstanding Transactions

Transaction Type	Maximum Number	Description
Read	9	A maximum of nine read addresses can be accepted by the AXI slave.
Write	7	A maximum of seven write addresses can be accepted by the AXI slave.

18.4.2 Interleaving / Out-of-Order Support

Table 18-16 describes the AXI slave port write interleaving depth support.

Table 18-16: AXI Slave Port Write Interleaving Depth Support

AXI Attribute	Maximum Number	Description
Write interleaving depth	1	The AXI slave port supports write interleaving depth of 1. This means that the slave does not support interleaved data with two different IDs.

Table 18-17 describes the AXI slave port out-of-order support.

Table 18-17: AXI Slave Port Out-of-Order Support

AXI Attribute	Support	Description
Out-of-order support in read transactions	No	The AXI slave port does not send read data out of order. This means that the slave will not send read data from the next transaction before it finishes sending the read data for the current transaction.

18.4.3 Supported AXI Signal Values

Table 18-18 describes the supported EDAP AXI slave signal values.

Table 18-18: Supported EDAP AXI Slave Signal Values

Signal Name	Possible Values	Description
arlen_edap awlen_edap	0-255	● Supports FIXED and WRAP transaction bursts up to 16 transfers. ● Supports INCR transaction bursts up to 256 transfers.
arsize_edap awsize_edap	0-4	Supports transaction sizes of 1-16 bytes.
arid_edap awid_edap cevaxm4_edap_rid_r cevaxm4_edap_bid_r	0x0-0xFFFF	Supports any ID, 16 bits wide
arburst_edap awburst_edap	0b00, 0b01, 0b10	Supports FIXED, INCR, and WRAP modes.
cevaxm4_edap_arready_r	0, 1	*ARREADY* is driven low to prevent an overflow of the two read address buffers.

Signal Name	Possible Values	Description
cevaxm4_edap_awready_r	0, 1	*AWREADY* is driven low to prevent an overflow of the two write address buffers.
cevaxm4_edap_wready_r	0, 1	The interface can accept a maximum of two wdata if the internal access is denied through contention.
cevaxm4_edap_rresp_r cevaxm4_edap_bresp_r	0b00, 0b10	EDAP responds with either OKAY or SLVERR.

Table 18-19 describes the supported AXIsX (AXIs0, AXIs1, and AXIs2) slave signal values.

Table 18-19: Supported AXIsX Slave Signal Values

Signal Name	Possible Values	Description
arlen_slvX awlen_slvX	0-255	● Supports FIXED and WRAP transaction bursts up to 16 transfers. ● Supports INCR transaction bursts up to 256 transfers.
arsize_slvX awsize_slvX	0-5	● When using a data width of 128 bits, supports transaction sizes of 1-16 bytes. ● When using a data width of 256 bits, supports transaction sizes of 1-32 bytes.
arid_slvX awid_slvX cevaxm4_slvX_rid_r cevaxm4_slvX_bid_r	0x0-0xFFFF	Supports any ID, 16 bits wide
arburst_slvX awburst_slvX	0b00, 0b01, 0b10	Supports FIXED, INCR, and WRAP modes.
cevaxm4_slvX_arready_r	0, 1	*ARREADY* is driven low to prevent an overflow of the two read address buffers.
cevaxm4_slvX_awready_r	0, 1	*AWREADY* is driven low to prevent an overflow of the two write address buffers.
cevaxm4_slvX_wready_r	0, 1	The interface can accept a maximum of two *wdata* if the internal access is denied through contention.
cevaxm4_slvX_rresp_r cevaxm4_slvX_bresp_r	0b00, 0b10	EDAP responds with either OKAY or SLVERR.

19. External Wait

The *external_wait* input can be used to drive the CEVA-XM4 core into wait. It is important to understand which elements of the CEVA-XM4 are stopped during the wait.

1. The *external_wait* input is first synchronized to the CEVA-XM4 clock domain.
2. It is then combined with all of the internal sources of wait and registered to produce *cevaxm4_psu_core_wait_r*.
3. When *cevaxm4_psu_core_wait_r* is asserted, the following happens:
   - The core is stopped and its interfaces hold their values (that is, LS0, LS1, and I/O).
   - In the write buffer:
     - Address delay line stages ADL0_0, ADL0_1, ADL0_2, ADL0_3, ADL0_4, ADL0_5, ADL1_2, ADL1_3, ADL1_4, and ADL1_5 are stopped.
     - ADL0_6 and ADL1_6 are not stopped and can still move to the L1DMQ or L1DCQ.
     - The L1DMQ, L1DCQ and external output stage are not stopped.
   - The LS0/LS1 read address buffers (RABs) buffer the address and control information for outstanding read accesses.
   - The L1DM Data TCM are not stopped (this includes the JBOXs).
   - The LS0/LS1 read buffers buffer any returning data for outstanding read accesses.
   - The EDP and AXI Masters are not stopped.
   - The EPP Master is not stopped.
   - The DDMA and QMAN are not stopped.
   - The EDAP and AXI slaves are not stopped.

Table 19-1 describes how *cevaxm4_psu_core_wait_r* affects the different memory accesses.

Table 19-1: cevaxm4_psu_core_wait_r Memory Effects

Access	Effect
DDMA	Any DDMA continues to operate regardless of *cevaxm4_psu_core_wait_r*. Because the core I/O port is stopped during *cevaxm4_psu_core_wait_r*, no new DDMA transfers can be programmed by the core while *cevaxm4_psu_core_wait_r* is asserted.
QMAN	The QMAN continues to operate regardless of *cevaxm4_psu_core_wait_r*. Because the core I/O port is stopped during *cevaxm4_psu_core_wait_r*, no new QMAN operations can be programmed by the core while *cevaxm4_psu_core_wait_r* is asserted.
PDMA	Any PDMA download that is in progress continues regardless of *cevaxm4_psu_core_wait_r*. Because the core I/O port is stopped during *cevaxm4_psu_core_wait_r*, no new PDMA transfers can be programmed by the core while *cevaxm4_psu_core_wait_r* is asserted.
EDAP Slave	The EDAP slaves continue to operate regardless of *cevaxm4_psu_core_wait_r*.

20. DFT Support

Table 20-1 describes the CEVA-XM4 DFT interface.

Table 20-1: DFT Interface

DFT Interface	Direction	Bus Size [Bits]	Description
testmodep	Input	1	Test mode indication
scanen	Input	1	Scan enable signal
sci0,1...X	Inputs	1	Scan in signals
sco0,1..X	Outputs	1	Scan out signals
tst_gatedclock	Input	1	External gated clock test enable indication
tst_mem_gatedclock	Input	1	External gated clock test enable indication (memories)

Note: *The scanen, sciX, and scoX signals are generated by the backend flow and the number (**X**) reflects the number of scan chains.*

The *testmodep* indication is used to bypass the clocks and resets of the design during test mode, ensuring that the ATPG clock and reset are used. During test modes, the *testmodep* input should be held high (equal to 1’b1).

The scan enable signal is added by the backend flow and used to control the scan chains. For scan tests, the *scanen* input is used to allow scan chain shift and capture. During the scan session and during scan shift, *scanen* should be high (equal to 1’b1); during scan capture, it should be low (equal to 1’b0). The *scanen* input is not used in the RTL, but is connected as part of the backend flow (when scan insertion is performed).

The *tst_gatedclock* input is used to bypass the clock gaters of the design (excluding the memories) during test mode. During scan, BIST, or any other test modes, *tst_gatedclock* should be high (equal to 1’b1) to enable the clock gater bypassing.

The *tst_mem_gatedclock* input is used to bypass the clock gaters of the memories in the design during test mode. During scan, BIST, or any other test modes, *tst_mem_gatedclock* should be high (equal to 1’b1) to enable the clock gater bypassing.

Note: *In the NFF configuration, the testmodep indication is used to increase the scanability over the memory logic. In this configuration, additional MUXs are introduced in the PMEM block.*

21. OCEM Interface

21.1 External Breakpoint Request

The CEVA-XM4 has two external breakpoint inputs named *ext_bp1_req* and *ext_bp2_req*.

If a breakpoint is accepted, the core asserts either *cevaxm4_ocm_ext1_ack* or *cevaxm4_ocm_ext2_ack*, depending on the breakpoint request asserted.

Figure 21-1 shows the sequence of external breakpoint generation and acknowledge signals

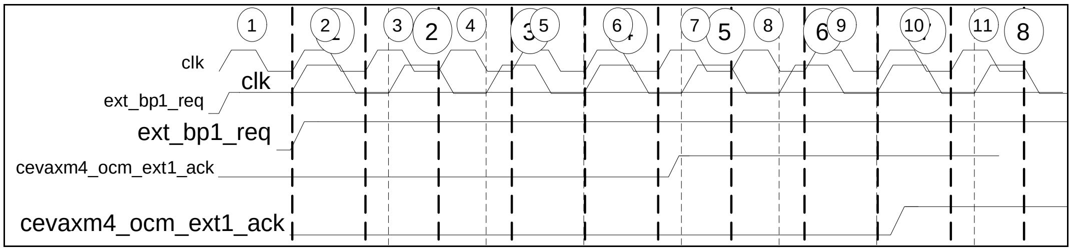


Figure 21-1: External Breakpoint Request Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The external breakpoint is asserted.
2. **Cycle 2**: The external breakpoint is sampled in the CEVA-XM4.
3. **Cycle 7**: An acknowledge signal is asserted by the OCEM.

Table 21-1: External Breakpoint Request Signal Description

Name	Description
clk	Free clock
ext_bp1_req	External breakpoint 1 request
ext_bp2_req	External breakpoint 2 request
cevaxm4_ocm_ext1_ack_r	Acknowledge signal for external breakpoint 1
cevaxm4_ocm_ext2_ack_r	Acknowledge signal for external breakpoint 2

21.2 JTAG Interface

21.2.1 Clock

All FFs within the CEVA-XM4 are sampled by the fast clock (*cevaxm4_free_clk*). The JTAG clock (*tck*) is at least eight times slower than the fast clock (*cevaxm4_free_clk*).

All JTAG inputs are synchronized within the CEVA-XM4 to *ceva_free_clk*. The user should mark these paths as false paths in backend EDA tools. The JTAG output from the CEVA-XM4 (*cevaxm4_ocm_tdo_r*) is generated on the falling edge of the *tck* synchronized to *cevaxm4_free_clk*, which means that it should be defined as multi-cycle of two or more (if the clock is more than eight times slower than the internal clock).

cevaxm4_psu_rtck_r is the return test clock (*tck*). It is delayed by two cycles of *cevaxm4_free_clk* from the synchronized rising edge of the *tck*.

Notes: *tck clock is asynchronous to cevaxm4_free_clk.*

cevaxm4_psu_rtck_r is synchronized to cevaxm4_free_clk.

Figure 21-2 shows *cevaxm4_ocm_tdo_r* and *cevaxm4_psu_rtck_r* generation according to *tck* and *cevaxm4_free_clk*:

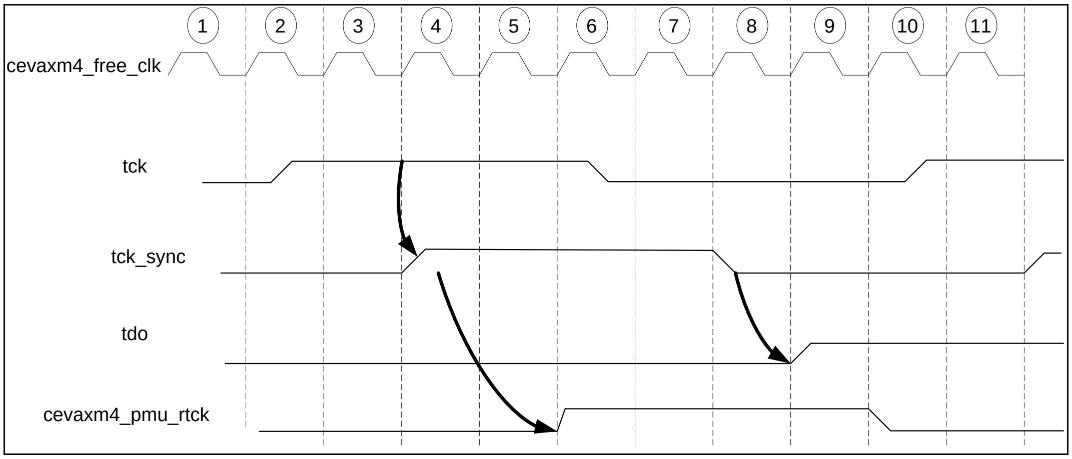


Figure 21-2: TDO and RTCK Generation Timing Diagram

The transaction sequence is as follows:

1. **Cycle 2**: Rising edge of *tck*
2. **Cycle 4**: Rising edge of *tck* (synchronized)
3. **Cycle 6**: Rising edge of *cevaxm4_psu_rtck_r*
4. **Cycle 8**: Falling edge of *tck* (synchronized)
5. **Cycle 9**: Rising edge of *tdo*

Because *cevaxm4_ocm_tdo_r* is stable at the rising edge of *cevaxm4_psu_rtck_r* (as shown in Figure 21-2(, it can be sampled at that time.

21.3 Clock Synchronization

The following are the test clock synchronization signals:

- *tck*: Test clock
- *tms*: Test mode select
- *tdi*: Test data input, sampled into the shift register upon *tck_set* according to the JTAG state machine

21.4 Boundary Scan Data Out

bs_reg_tdo is the boundary scan register data out signal.

If the JTAG instruction is one of the following:

- EXTEST (8’h00)
- Sample preload (8’hA1)
- INTEST (8’hA2)
- RUNBIST (8’hA3)

The value of the *bs_reg_tdo* input is bypassed to *cevaxm4_ocm_tdo_r*.

The *bs_reg_tdo* input is bypassed to *cevaxm4_ocm_tdo_r* after three cycles of *ceva_free_clk* following a *tck* negedge.

Figure 21-3 shows the bypass of *cevaxm4_ocm_tdo_r* according to the *bs_reg_tdo* input when the JTAG instruction is EXTEST.

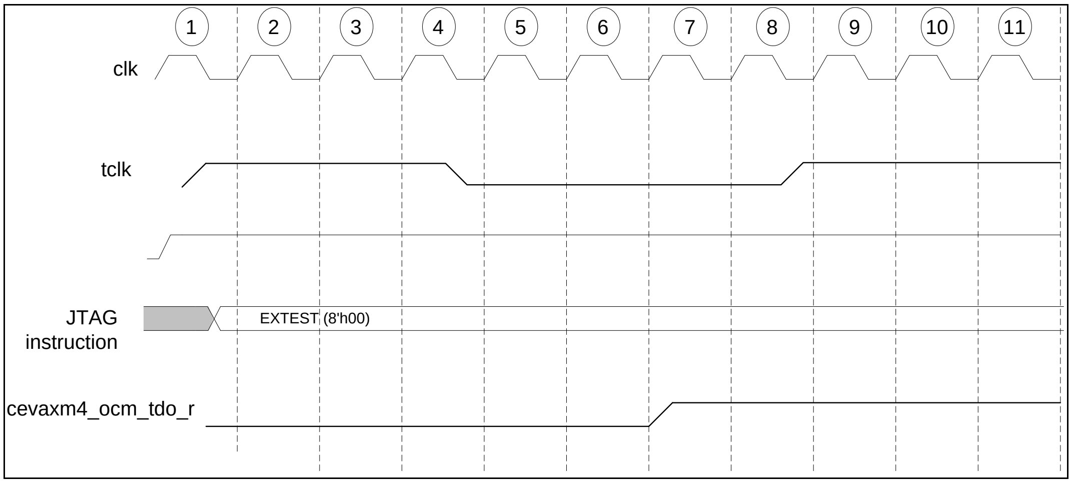

1
2
3
4
5
6
7
8
9
10
11
clk
tclk
JTAG instruction
EXTEST (8'h00)
cevaxm4_ocm_tdo_r

Figure 21-3: bs_reg_tdo Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: *bs_reg_tdo* is asserted.
2. **Cycle 4**: *tck* negedge.
3. **Cycle 7**: *cevaxm4_ocm_tdo_r* is bypassed according to *bs_reg_tdo*, three cycles of *ceva_free_clk* following a *tck* negedge.

Note: *The JTAG instruction is EXTEST in all cycles.*

21.5 OCEM TDO

cevaxm4_ocm_tdo_r is the test data out upon *tck_rst* according to the JTAG state machine.

Figure 21-4 shows a rising of *cevaxm4_ocm_tdo_r* after the rising edge of *cevaxm4_psu_rtck_r*.

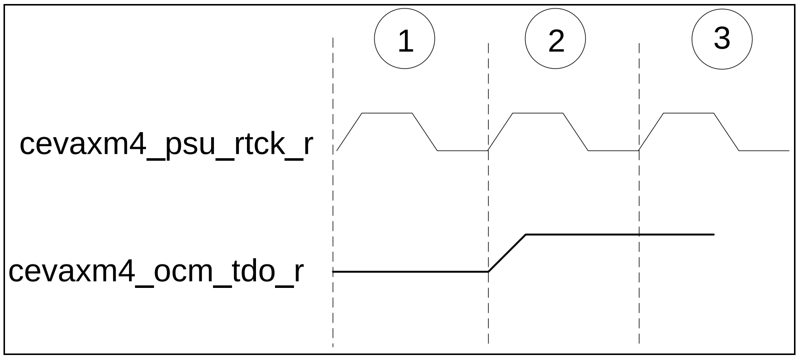

1
2
3
cevaxm4_psu_rtck_r
cevaxm4_ocm_tdo_r

Figure 21-4: cevaxm4_ocm_tdo_r Timing Diagram

21.6 TDO Output Enable

cevaxm4_ocm_tdo_oen_r is the test data out pad output enable, and is a supplement to the JTAG interface signals. It is active low, which denotes that the *cevaxm4_ocm_tdo_r* is valid on the rising edge of *cevaxm4_psu_rtck_r*. This signal is also asserted during scan tests, when the *testmodep* signal is asserted.

Figure 21-5 shows the following two scenarios:

- *cevaxm4_ocm_tdo_r* is not enabled (in cycle 1, *cevaxm4_ocm_tdo_oen_r* is high).
- *cevaxm4_ocm_tdo_r* is enabled (in cycle 2, *cevaxm4_ocm_tdo_oen_r* is low).

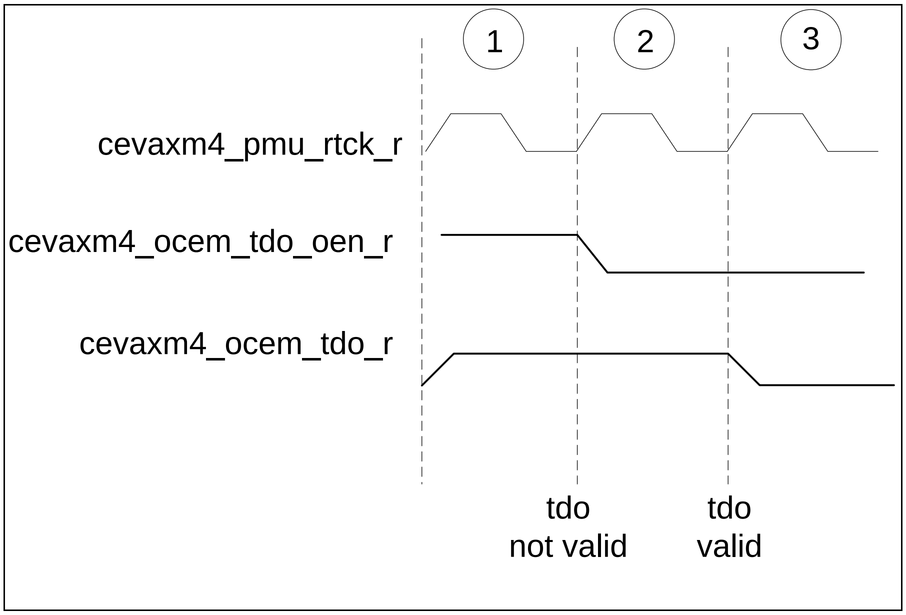


Figure 21-5: cevaxm4_ocm_tdo_oen_r Timing Diagram

21.7 OCEM APB Slave Port

The CEVA-XM4 OCEM APB slave port can receive APB transfers to read/write OCEM I/O and RTT registers, as well as use OCEM scan chains.

APB accesses are only accepted if the *jt_ap* input signal is set.

Transfers should be initiated sequentially. A subsequent transfer will be received only if the previous transfer's *pready* was set, as defined in the APB protocol.

PSLVERR will rise according to the conditions described in the *CEVA-XM4 OCEM Reference Guide*.

21.7.1 OCEM APB Slave Port Read Transfer

Figure 21-6 shows a read of the **CORE_CONFIG** register (CPM address 0x17C). Only bits 11:2 of the address are referred.

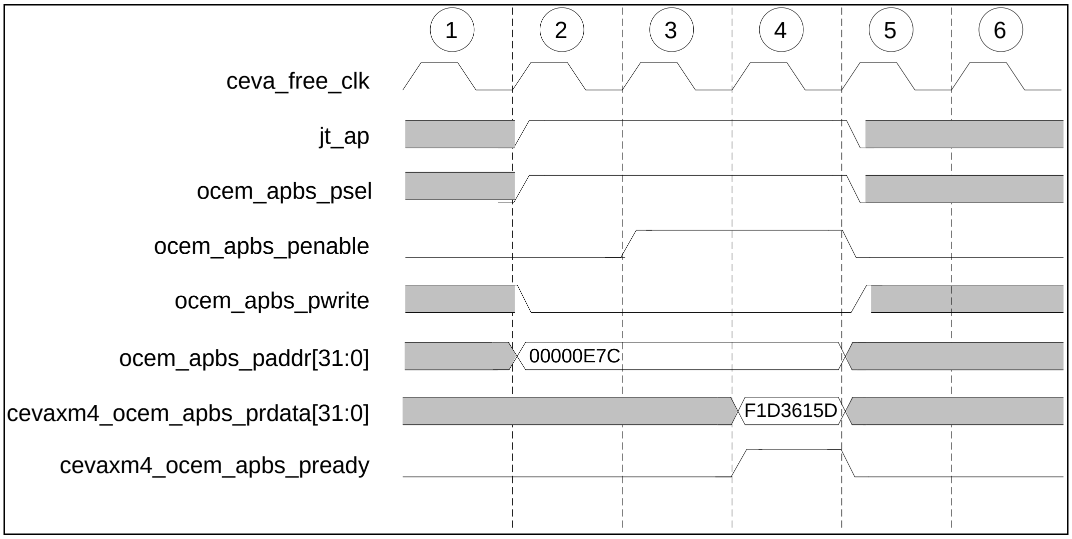

1
2
3
4
5
6
ceva_free_clk
jt_ap
ocem_apbs_psel
ocem_apbs_penable
ocem_apbs_pwrite
ocem_apbs_paddr[31:0]
00000E7C
cevaxm4_ocem_apbs_prdata[31:0]
F1D3615D
cevaxm4_ocem_apbs_pready

Figure 21-6: OCEM APB Slave Read CORE_CONFIG Register Timing Diagram

21.7.2 OCEM APB Slave Port Write Transfer

Figure 21-7 shows the following two write transfers:

- The first configures the scan chains code configuration (**SCCO**) register (APB address 0xD00) to write the *s_pmem_int_ctrl* scan chain (code 0x84).
- The second writes the required operation to the scan chains data configuration (**SCDA**) register (APB address 0xD04).

Following these two writes, the OCEM prepares to write to the program set (internal address 0x008018).

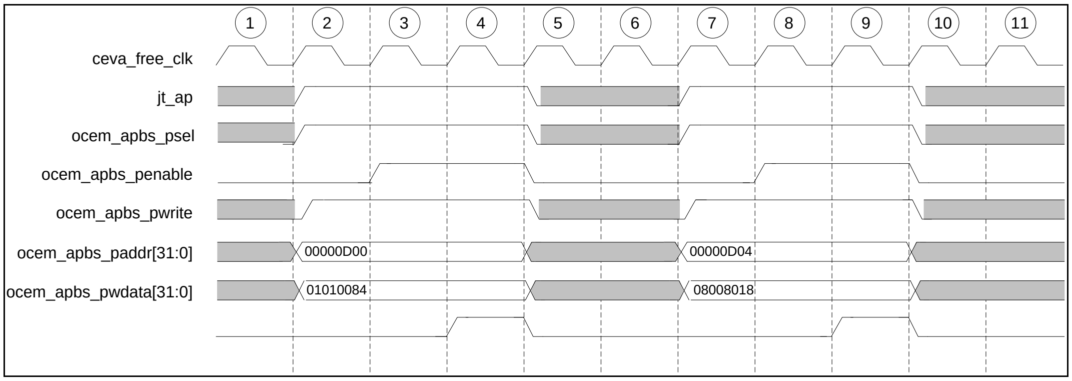

1
2
3
4
5
6
7
8
9
10
11
ceva_free_clk
jt_ap
ocem_apbs_psel
ocem_apbs_penable
ocem_apbs_pwrite
ocem_apbs_paddr[31:0]
00000D00
00000D04
ocem_apbs_pwdata[31:0]
01010084
08008018

Figure 21-7: OCEM APB Slave Write to Scan Chains Timing Diagram

21.8 External Program Memory Indication

ext_pmem_wr is a one-bit indication that the external program was written to. This bit is cleared when the debugger reads the status register.

It suggests to the user that the cache might be “dirty”, that is, that the external memory content is different from the cache content.

The *ext_pmem_wr* input is sampled in bit 4 of the **OCM_STATUS** register (CPM address 0x160).

Figure 21-8 shows the relationship between the *ext_pmem_wr* input port and the **OCM_STATUS** register (bit 4).

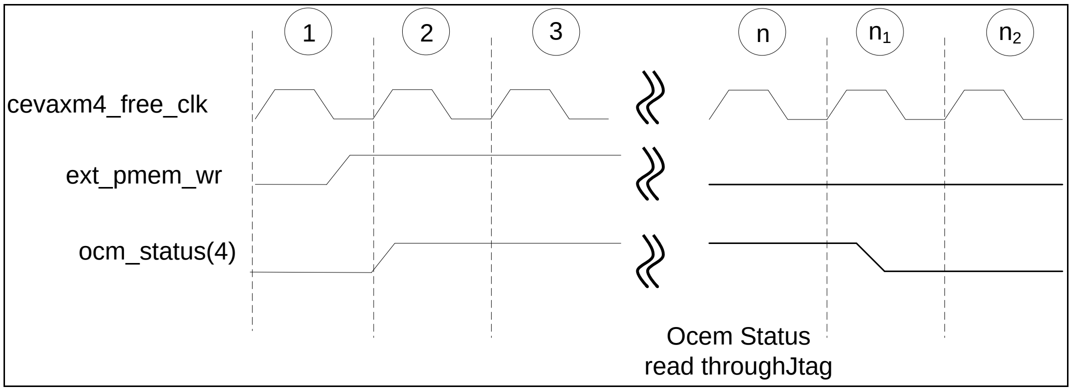


Figure 21-8: External Program Memory Indication Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: When *ext_pmem_wr* is asserted, the **OCM_STATUS** register (bit 4) is also asserted.
2. **Cycle n**: When the bit is read through a JTAG instruction, the bit is de-asserted.

21.9 General Purpose Outputs

cevaxm4_ocm_gp_out_r[3:0] is a four-bit OCEM general purpose output. It reflects the value of bits 15:12 in the **OCM_CONTROL** register (CPM address 0x150).

Figure 21-9 shows how the cevax*m4_ocm_gp_out_r* output port follows bits 15:12 in the **OCM_CONTROL** register.

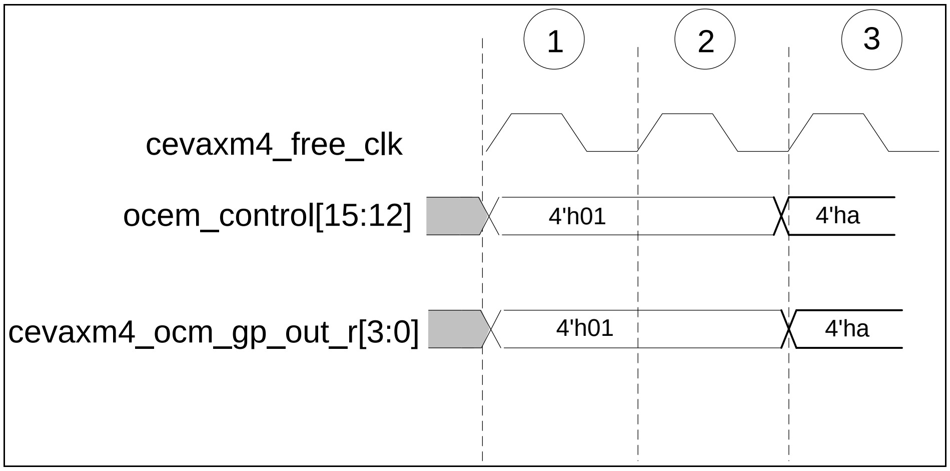

1
2
3
cevaxm4_free_clk
ocem_control[15:12]
4'h01
4'ha
cevaxm4_ocm_gp_out_r[3:0]
4'h01
4'ha

Figure 21-9: cevaxm4_ocm_gp_out_r Timing Diagram

The transaction sequence is as follows:

1. **Cycle 1**: The value of both signals is 4’h01.
2. **Cycle 3**: The value of both signals is 4’ha.

21.10 OCEM Core Reset

The OCEM reset signal to the core, active high. It reflects the value of bit 2 in the **OCM_CONTROL** register (CPM address 0x150). When reset is asserted, it resets all of the blocks in the core.

Figure 21-10 shows how a register inside the core (in this case, **DAAU**) is affected by *cevaxm4_ocm_core_rst_r*.

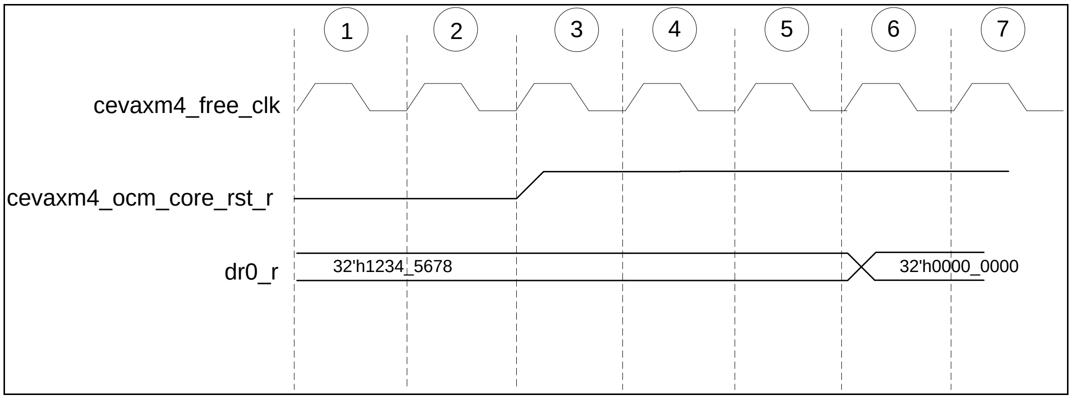

1
2
3
4
5
6
7
cevaxm4_free_clk
cevaxm4_ocm_core_rst_r
dr0_r
32'h1234_5678
32'h0000_0000

Figure 21-10: OCEM Core Reset Timing Diagram

The transaction sequence is as follows:

1. **Cycle 3**: *cevaxm4_ocm_core_rst_r* rises.
2. Three cycles after that, the **DR0_R** register gets the reset value of 32’h0000_0000.

22. Real-Time Trace Integration

The Real-Time Trace modules (RTT, ETM-R4, and Trace Wrapper) are optional and depend on the RTT option being selected during the installation phase.

22.1 Integration with ETM-R4

The figures in this section show examples of connections between the CEVA-XM4 RTT Wrapper and the Trace modules required to extract the compressed trace data.

Notes: *Because the RTT does not trace Vector Computation Unit (VCU) load or store accesses, data trace cannot be performed on VCU instructions.*

In the Light Sleep/ Standby PSU modes, RTT transactions towards the core do not enable the core clocks in any way. Only an OCEM transaction towards the memories and core wakes the core clocks.

When the PSU is putting the core into the Idle state, the Wrapper asserts the cevaxm4_w_etm_wfi_pending signal to the ETM-R4 to cause it to drain its FIFO. The PSU delays moving to the Standby, Shutdown, or Deep Sleep modes until the ETM-R4 indicates that its FIFO is empty by de-asserting the cevaxm4_etm_wfi_ready_n signal.

In Figure 22-1, the ETM-R4 module is **internal** to the CEVA-XM4, and the external TPIU module interfaces between the ETM-R4 and an external analyzer data port. Optionally, the TPIU can be replaced by an on-chip CoreSight-compliant trace buffer. The ETM-R4 and TPIU modules can be configured via CEVA-XM4 *in/out* instructions over the APB3 interface.

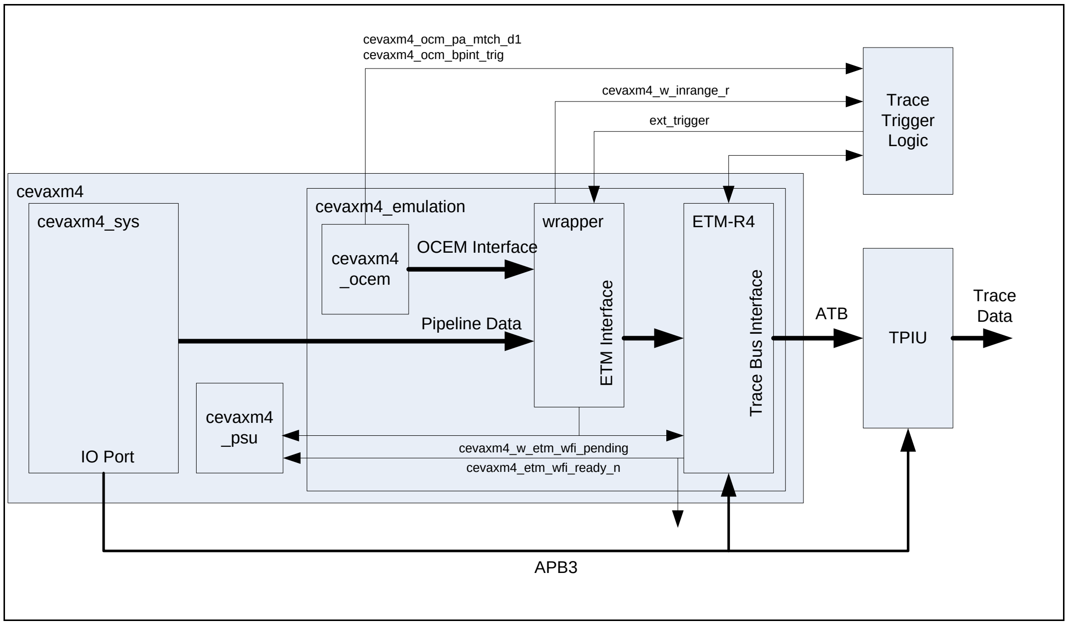

cevaxm4_ocm_pa_mtch_d1
cevaxm4_ocm_bpint_trig
cevaxm4_w_inrange_r
ext_trigger
Trace Trigger Logic
cevaxm4
cevaxm4_sys
cevaxm4_emulation
cevaxm4 _ocem
OCEM Interface
wrapper
ETM-R4
Pipeline Data
ETM Interface
Trace Bus Interface
ATB
TPIU
Trace Data
cevaxm4 _psu
IO Port
cevaxm4_w_etm_wfi_pending
cevaxm4_etm_wfi_ready_n
APB3

Figure 22-1: RTT Integration (Internal ETM-R4)

The optional Trace Trigger Logic (TLL) module is the responsibility of the user. This module combines and synchronizes triggers and events to generate triggers for the trace wrapper (*ext_trigger_r*) and ETM-R4 to control the start/stop of the RTT.

While Figure 22-1 shows that the *cevaxm4_ocm_bpint_trig*, *cevaxm4_ocm_pa_mtch_d1* (described in Table 22-4), and *cevaxm4_w_inrange_r* (described in the *CEVA-XM4 Real-Time Trace Architecture Specification*) signals are connected to the TLL module, the user can connect and synchronize other events and triggers as required.

Table 22-1 lists the mapping of the CEVA-XM4 signals to the ETM-R4 interface when the ETM-R4 is internal to the CEVA-XM4.

Table 22-1: Internal ETM within CEVA-XM4 RTT Mapping

ETM-R4 Ports	Mapped to CEVA-XM4 Signals	CEVA-XM4 I/O or Internal
ETMPWRUP	*cevaxm4_etm_pwr_up*	Output
ETMDBGRQ	-	Not connected
PRDATADBG	*etm_apb_rdata_dbg*	Internal (muxed to output)
PREADYDBG	*etm_apb_ready_dbg*	Internal (muxed to output)
nETMWFIREADY	*cevaxm4_etm_wfi_ready_n*	Internal (PSU) + Output
ATDATA	*cevaxm4_etm_atb_data*	Output
ATVALID	*cevaxm4_etm_atb_valid*	Output
ATBYTES	*cevaxm4_etm_atb_bytes*	Output
AFREADY	*cevaxm4_etm_atb_fready*	Output
TRIGGER	*cevaxm4_etm_trigger*	Output
ATID	*cevaxm4_etm_atb_id*	Output
ETMEN	*cevaxm4_etm_en*	Output
ASICCTL	*cevaxm4_etm_asic*	Output
CORESELECT	-	Not connected
FIFOPEEK	*cevaxm4_etm_fifo_peek*	Output
EXTOUT	*cevaxm4_etm_ext_out*	Output
nSYSPORESET	*sysporeset_n*	Input
CLK	*ceva_free_clk*	Input
PCLKDBG	*apb_clk_dbg*	Input
PCLKENDBG	*apb_clk_en_dbg*	Input
PRESETDBGn	*apb_reset_dbg_n*	Input
ATCLK	*etm_atb_clk*	Input
ATCLKEN	*etm_atb_clk_en*	Input
NIDEN	*etm_ni_dbg_en*	Input
DBGEN	*etm_in_dbg_en*	Input
ETMIA	*cevaxm4_w_etm1ia_r*	Internal (Wrapper)
ETMICTL	*cevaxm4_w_etm1iactl_r*	Internal (Wrapper)
ETMDA	*cevaxm4_w_etm1da_r*	Internal (Wrapper)
ETMDCTL	*cevaxm4_w_etm1dctl_r*	Internal (Wrapper)
ETMDD	*cevaxm4_w_etm1dd_r*	Internal (Wrapper)

ETM-R4 Ports	Mapped to CEVA-XM4 Signals	CEVA-XM4 I/O or Internal
ETMCID	*cevaxm4_w_etm_cid_r*	Internal (Wrapper)
DBGACK	*cevaxm4_w_debug_e5*	Internal (Wrapper)
PENABLEDBG	*apb_enable_dbg*	Input
PSELDBG	*apb_sel_dbg*	Input
PADDRDBG	*apb_addr_dbg[11:2]*	Input
PADDRDBG31	*apb_addr_dbg[31]*	Input
PWRITEDBG	*apb_write_dbg*	Input
PWDATADBG	*apb_wdata_dbg*	Input
ETMWFIPENDING	*w_etm_wfi_pending*	Internal(Wrapper)
ATREADY	*etm_atb_ready*	Input
AFVALID	*etm_atb_fvalid*	Input
TRIGGERACK	*etm_trigger_ack*	Input
TRIGSBYPASS	*etm_trigger_bypass*	Input
SE	*tst_gatedclock*	Input
RSTBYPASS	*etm_rst_bypass*	Input
MAXEXTIN	*etm_max_ext_in*	Input
MAXEXTOUT	*etm_max_ext_out*	Input
EVNTBUS	*etm_evnt_bus*	Input
MAXCORES	3’b000 (1 core only)	
EXTIN	*etm_ext_in*	Input

In Figure 22-2, the ETM-R4 module is **external** to the CEVA-XM4. This enables the ETM_R4 module to be shared with other cores.

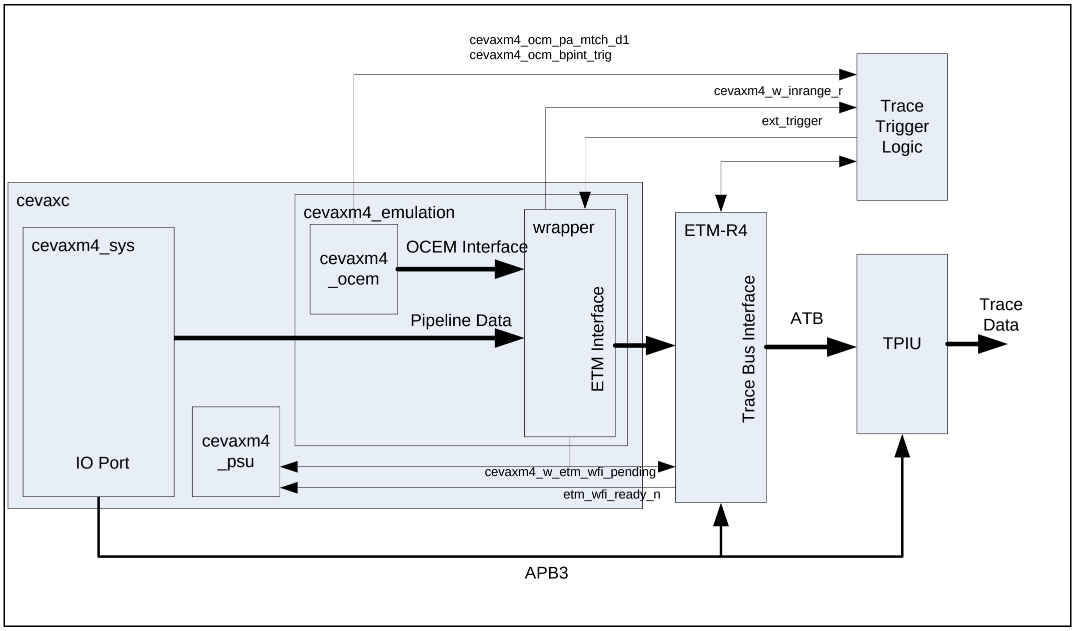

cevaxm4_ocm_pa_mtch_d1
cevaxm4_ocm_bpint_trig
cevaxm4_w_inrange_r
ext_trigger
Trace Trigger Logic
cevaxc
cevaxm4_emulation
wrapper
cevaxm4_sys
cevaxm4 _ocem
OCEM Interface
Pipeline Data
ETM Interface
ETM-R4
Trace Bus Interface
ATB
TPIU
Trace Data
IO Port
cevaxm4 _psu
cevaxm4_w_etm_wfi_pending
etm_wfi_ready_n
APB3

Figure 22-2: RTT Integration (External ETM-R4)

Table 22-2 lists the mapping of the CEVA-XM4 signals to the ETM-R4 interface when a single ETM-R4 is external to the CEVA-XM4.

Table 22-2: External ETM within CEVA-XM4 RTT Mapping

ETM-R4 Ports	Mapped to CEVA-XM4 Signals	CEVA-XM4 I/O Direction
nETMWFIREADY	*etm_wfi_ready_n*	Input
ETMIA	*cevaxm4_w_etm1ia_r*	Output
ETMICTL	*cevaxm4_w_etm1iactl_r*	Output
ETMDA	*cevaxm4_w_etm1da_r*	Output
ETMDCTL	*cevaxm4_w_etm1dctl_r*	Output
ETMDD	*cevaxm4_w_etm1dd_r*	Output
ETMCID	*cevaxm4_w_etm_cid_r*	Output
DBGACK	*cevaxm4_w_dbug_e5*	Output
ETMWFIPENDING	*cevaxm4_w_etm_wfi_pending*	Output

During the installation of the CEVA-XM4, the following directives are set automatically in the **cevaXM4_gendef.v** and **param_file.v** files:

- **`define EN_ETM**
- **`define TRACE**
- **`define EXT_ETM_R4 (if external ETM)**
- **parameter CEVAXM4_TRACE = 2 (0=no RTT, 1=internal RTT, 2=external RTT)**

Table 22-3 lists how the external ETM-R4 macrocell can be connected to the CEVA-XM4 and the TPIULITE macrocell.

Table 22-3: External ETM Connections

ETM-R4 Ports	Connect To/From	ETM-R4 I/O Direction
ETMPWRUP	User output	Output
ETMDBGRQ	User output	Output
PRDATADBG	APB3 Subsystem	Output
PREADYDBG	APB3 Subsystem	Output
nETMWFIREADY	etm_wfi_ready_n	Output
ATDATA	To TPIU	Output
ATVALID	To TPIU	Output
ATBYTES	To TPIU	Output
AFREADY	To TPIU	Output
TRIGGER	To TPIU	Output
ATID	User output	Output
ETMEN	User output	Output
ASICCTL	User output	Output
CORESELECT	User output	Output
FIFOPEEK	User output	Output
EXTOUT	User output	Output
nSYSPORESET	System reset	Input
CLK	CEVA free clk	Input
PCLKDBG	APB3 clock	Input
PCLKENDBG	APB3 clock enable	Input
PRESETDBGn	APB3 Subsystem reset	Input
ATCLK	Trace Bus clock	Input
ATCLKEN	Trace Bus clock enable	Input
NIDEN	User input	Input

ETM-R4 Ports	Connect To/From	ETM-R4 I/O Direction
DBGEN	User input	Input
ETMIA	*cevaxm4_w_etm1ia_r*	Input
ETMICTL	*cevaxm4_w_etm1iactl_r*	Input
ETMDA	*cevaxm4_w_etm1da_r*	Input
ETMDCTL	*cevaxm4_w_etm1dctl_r*	Input
ETMDD	*cevaxm4_w_etm1dd_r*	Input
ETMCID	*cevaxm4_w_cid_r*	Input
DBGACK	*cevaxm4_w_debug_e5*	Input
PENABLEDBG	APB3 Subsystem	Input
PSELDBG	APB3 Subsystem	Input
PADDRDBG	APB3 Subsystem	Input
PADDRDBG31	APB3 Subsystem	Input
PWRITEDBG	APB3 Subsystem	Input
PWDATADBG	APB3 Subsystem	Input
ETMWFIPENDING	*cevaxm4_w_etm_wfi_pending*	Input
ATREADY	From TPIU	Input
AFVALID	From TPIU	Input
TRIGGERACK	From TPIU	Input
TRIGSBYPASS	User input	Input
SE	Scan enable	Input
RSTBYPASS	User input	Input
MAXEXTIN	User input	Input
MAXEXTOUT	User input	Input
EVNTBUS	User input	Input
MAXCORES	User input	Input
EXTIN	User input	Input

22.2 RTT Reset

The Trace Wrapper is reset by negating the *sysporeset_n* input. This input should also be connected to the ETM-R4 and other trace components.

Because the *sysporest_n* input is independent of the CEVA-XM4 core reset, the core can be reset while still keeping the trace components active. This allows trace throughout the core boot sequence.

22.3 Integrating CEVA-XM4 Trace Triggers

The CEVA-XM4 outputs trigger events from both the Trace Wrapper (*cevaxm4_w_inrange_r*) and the OCEM. The triggers can be used to control traces in the Trace Wrapper and the ETM. They can also be used to control traces in other trace modules connected to the same trace architecture. All of the CEVA-XM4 trace trigger outputs are synchronous to *cevaxm4_clk*.

Table 22-4 describes the OCEM trigger outputs. All triggers that are output from the OCEM are asserted when the OCEM event is true (for example, if a data value match breakpoint event occurs for one cycle, it causes the corresponding *cevaxm4_ocm_bpint_trig*[2] OCEM trigger to assert for one cycle). These triggers can be connected to either the ETM external inputs or the ETM EVNTBUS, which provides two additional extended external inputs. The ETM external inputs are clocked by the ETM clock, which is the same as *cevaxm4_clk*.

Table 22-4: cevaxm4 OCEM Triggers

Trigger	Description
cevaxm4_ocm_pa_mtch_d1	Program address breakpoint #1 match, D1 stage
cevaxm4_ocm_bpint_trig[4]	Standalone data address (read) breakpoint #1 request
cevaxm4_ocm_bpint_trig[3]	Standalone data address (write) breakpoint #1 request
cevaxm4_ocm_bpint_trig[2]	Standalone data value match breakpoint request
cevaxm4_ocm_bpint_trig[1]	Standalone combined data address and value (read) breakpoint #1 request
cevaxm4_ocm_bpint_trig[0]	Standalone combined data address and value (write) breakpoint #1 request

The Trace Wrapper outputs three *cevaxm4_w_inrange_r* range indications that are level sensitive and synchronized to *cevaxm4_clk*. Edge-detect logic can be added to the in-range outputs to convert them into event triggers, which are then subsequently routed directly to the ETM external inputs or *EVNTBUS*.

The Trace Wrapper can also be configured to start/stop traces based on level-sensitive external trigger inputs (*ext_trigger*). External logic can be added to sample and hold external event triggers to convert them into level-sensitive inputs. The Trace Wrapper synchronizes the external trigger inputs.

The OCEM triggers can be used to control trace outputs in the Trace Wrapper. In this case, external logic to sample and hold the OCEM triggers is required.

Figure 22-3 shows an example of how the CEVA-XM4 trace triggers can be connected to a CoreSight Cross-Trigger Interface (CTI) and shared with other RTT components.

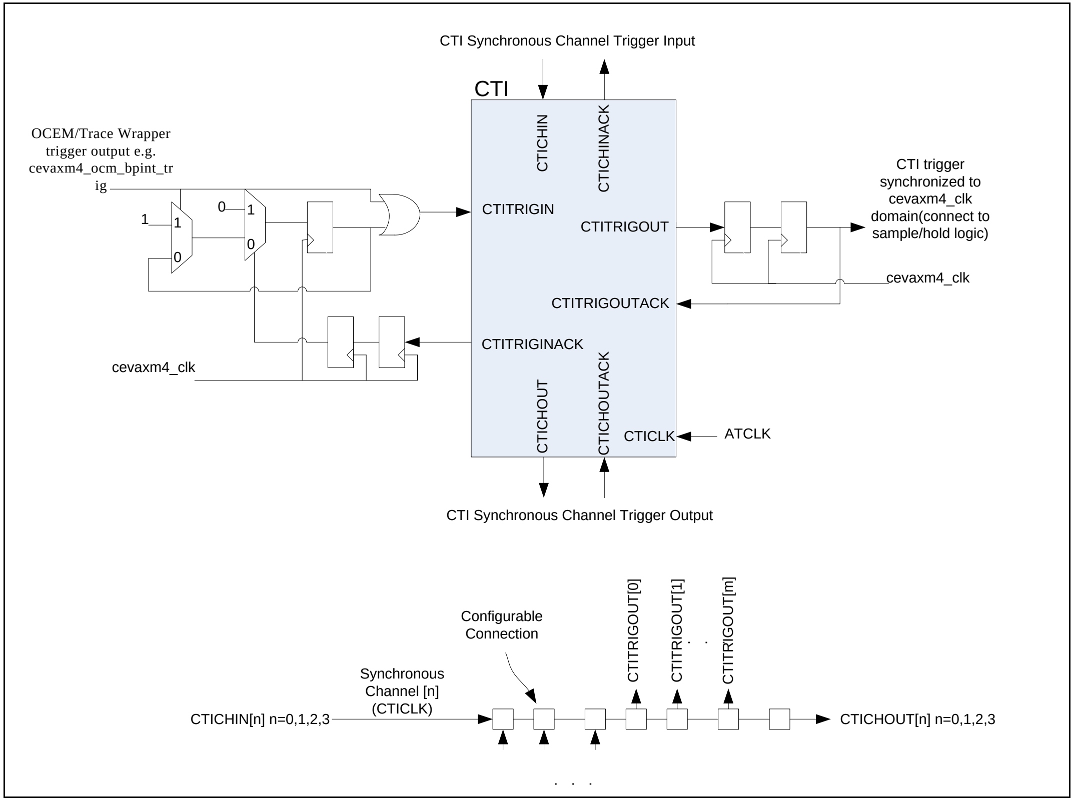

CTI Synchronous Channel Trigger Input
CTI
CTICHIN
CTICHINACK
OCEM/Trace Wrapper trigger output e.g. cevaxm4_ocm_bpint_trig
1
1
0
0
1
0
CTITRIGIN
CTITRIGOUT
CTI trigger synchronized to cevaxm4_clk domain(connect to sample/hold logic)
cevaxm4_clk
CTITRIGOUTACK
CTITRIGINACK
cevaxm4_clk
CTICHOUT
CTICHOUTACK
CTICLK
ATCLK
CTI Synchronous Channel Trigger Output
Configurable Connection
CTITRIGOUT[0]
CTITRIGOUT[1]
CTITRIGOUT[m]
Synchronous Channel [n] (CTICLK)
CTICHIN[n] n=0,1,2,3
CTICHOUT[n] n=0,1,2,3

Figure 22-3: Connecting CTI to CEVA-XM4 Trace Triggers (Example)

Triggers from other trace components can also be connected to CEVA-XM4 trace modules. The CTI provides a number of trigger channels synchronous to the trace bus clock (*ATCLK*). The synchronous channel inputs can optionally be connected to the synchronous channel outputs or any of the CTI trigger outputs. Additionally, any CTI trigger input can optionally be connected to a synchronous channel output or the CTI trigger outputs. The synchronous channel outputs can be connected to other CTI modules or a cross-trigger matrix (CTM).

CEVA-XM4 triggers are sampled and held before transfer to the CTI *CTITRIGIN*. The corresponding *CTITRIGINACK* acknowledgment clears the sample and holds logic to allow other triggers to be captured. The *CTITRIGOUT* output triggers are synchronized to the *cevaxm4_clk* before connecting to the CEVA-XM4 trace modules. The *CTITRIGOUTACK* can be generated from the synchronized triggers.

It is not mandatory to use the CTI; instead, CEVA-XM4 triggers can be connected directly to the ETM-R4. It is possible to connect signals synchronous to *cevaxm4_clk* directly to the *EXTIN* external inputs and the *EVNTBUS* inputs of the ETM-R4 macrocell, and then use these signals to control traces in the ETM-R4. Two *EVNTBUS* inputs can be configured as external extended inputs at any one time.

23. Multi-Cycle Paths

In accordance with the Memory Subsystem (MSS) specification, when *div_en* of a specific interface is set every *n* fast clocks, it creates a clock division and enables the CEVA-XM4 to connect to an external synchronous clock that runs at a slower frequency.

Setting *div_en* every other clock in conjunction with the slower external clock enables setting multi-cycle path constraints between the fast clock and the slow clock. The fast clock is the CEVA-XM4 clock, and the slow clock is the external clock that runs at an integer multiply of the fast clock cycle.

The user should define the maximal latency (that is, the number of fast clocks, *N* in the example here) of the multi-cycle path.

The multi-cycle path from the CEVA-XM4 core looks like this:

```
set_multicycle_path N -start -setup -from < CEVA-XM4-
clock> -to <slow-clock>

set_multicycle_path (N-1) -start -hold -from < CEVA-XM4-
clock> -to <slow-clock>
```

The multi-cycle path to the CEVA-XM4 core looks like this:

```
set_multicycle_path N -end -setup -from <slow-clock> -to
< CEVA-XM4-clock>

set_multicycle_path (N-1) -end -hold -from <slow-clock>
-to < CEVA-XM4-clock>
```

24. Input Clock Synchronization

Inputs to the CEVA-XM4 that are issued by a different clock domain (that is, not the same as the CEVA-XM4 clock) are synchronized inside the CEVA-XM4. Table 24-1 describes these signals.

Table 24-1: Input Clock Synchronization

Input Name	Description
ceva_global_rst_n	Global asynchronous reset (active low)
ceva_ocem_rst_n	OCEM external reset (active low)
external_wait	External wait request
tck	Test clock
tms	Test mode select
tdi	Test data in
stop_sd	● Stops the core from shutting down ● Restores the core’s power after it is in the power OFF state
bus_parity	Selects parity configuration for MSS AMBA Bus ECC
acu_lock	DACU and IACU lock indication
acu_slv_acc	Indicates if external slaves or DSP can change the DACU/IACU configuration

Note: *The boot input signal is not synchronized by the CEVA-XM4. This input must be externally synchronized and be stable at least two clock cycles before the global asynchronous reset is released.*

25. Input Single Sampling

Table 25-1 describes the inputs to the CEVA-XM4 that are single-sampled by the CEVA-XM4. These inputs are assumed to be already synchronized to the CEVA-XM4 clock domain.

Table 25-1: Input Single Sampling

Input Name	Description
int0	Maskable interrupt 0
int1	Maskable interrupt 1
int2	Maskable interrupt 2
nmi	Non-maskable interrupt 0
vint	Vectored interrupt
core_rcvr	Core recovery
ext_bp1_req	External breakpoint request #1
ext_bp2_req	External breakpoint request #2
qn_desc_en	QMAN-enabled descriptors increment signal

26. CEVA-XM4 Instantiation

The **cevaxm4_sim_top.v** file (located in the **simulation/asm/verilog/top** directory, as shown in Figure 26-1) contains an example of a full CEVA-XM4instantiation.

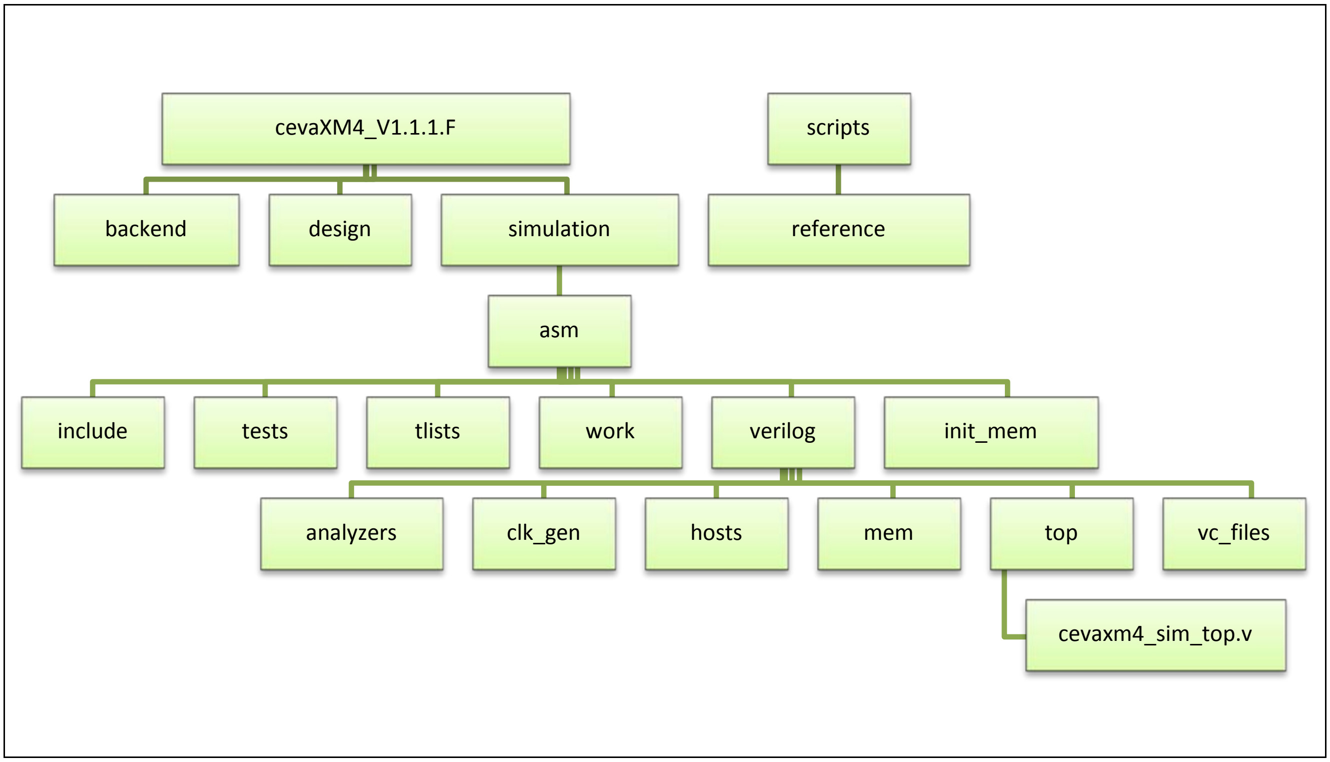

cevaXM4_V1.1.1.F
scripts
backend
design
simulation
reference
asm
include
tests
tlists
work
verilog
init_mem
analyzers
clk_gen
hosts
mem
top
vc_files
cevaxm4_sim_top.v

Figure 26-1: cevaxm4_sim_top.v Location

The start of the CEVA-XM4 instantiation is indicated by the following:

```
//
********************************************************
**********

// CEVAXM4 INSTANTIATION START

//
********************************************************
**********
```

The end of the CEVA-XM4 instantiation is indicated by the following:

```
//
********************************************************
**********

// CEVAXM4 INSTANTIATION END

//
********************************************************
**********
```

CEVA-XM4 unused I/O signals must be set to default values, as described in Table 2-1.

27. Validation for Signoff

To guarantee correct functionality during the process of implementing the CEVA-XM4, the IP integrator must ensure that the following sign-off stages have passed:

- The entire test suite passes in RTL-Level simulation
- Equivalence checking of RTL against the pre/post-layout netlist
- Clean STA with no violation in all corners
- The entire test suite passes Gate-Level simulation with timing (Dynamic Timing Analysis - SDF)
- Clean LVS DRC

For further information regarding the simulation and verification environment of the CEVA-XM4, see the *CEVA-XM4 Simulation Reference Guide.*

28. Glossary

Table 28-1 defines the acronyms used in this document.

Table 28-1: Acronyms

Term	Definition
APS	Automatic Power Save
BIST	Built In Self-Test
CCU	Clock Control Unit
CTI	CoreSight Cross-Trigger Interface
CTM	Cross-Trigger Matrix
DACU	Data Access Control Unit
DDMA	Data DMA
DFT	Design For Test
DMAN	DMA Manager
DMEM	Data Memory
DMSS	Data Memory Subsystem
DPS	Dynamic Power Save mode
DSP	Digital Signal Processor
DW	DesignWare
ECC	Error Checking and Correction
EDAP	External Data Access Port
EDP	External Data Port
EOS	External Output Stage
EPP	External Program Port
FIFO	First In, First Out
GVI	General Violation Indication
IACU	Instruction Access Control Unit
IOP	I/O Port
MCCI	Multicore Command Interface
MSS	Memory Subsystem
OCEM	On-Chip Emulation
PCU	Program Control Unit
PMEM	Program Memory
PMSS	Program Memory Subsystem
PRAW	Protected RAW mode

Term	Definition
PSU	Power Scaling Unit
PTCM	Program TCM
QMAN	Queue Manager
RAB	Read Address Buffer
RAC	Read Address Channel
RAW	Read after Write
RDC	Read Data Channel
RID	Read ID
RRA	Round-Robin Arbitration
RTT	Real-Time Trace
SAR	State after Reset
SDF	Simulation Delay File
SDT	Software Development Tools
SIP	Soft IP
SPU	Scalar Processing Unit
STA	Static Timing Analysis
SWOP	Software Operation
TLL	Trace Trigger Logic
VCU	Vector Computation Unit
VINT	Vector Interrupt
VPU	Vector Processing Unit
WAC	Write Address Channel
WDC	Write Data Channel
WRC	Write Response Channel