CEVA®

CEVA-XM4™

RTL V1.1.3.F Real-Time Trace Architecture Specification

Rev. 1.1.3.F

June 2016

DSP Division

Documentation Control

History Table

Version	Date	Description	Remarks
V1.1.0.F	12 June 2015	First CEVA-XM4 Version	
V1.1.1.F	18 January 2016	Updated version	
V1.1.2.F	14 March 2016	Updated version	
V1.1.3.F	8 June 2016	Updated version	

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.

CEVA®, CEVA-XC™, CEVA-XC5™, CEVA-XC8™, CEVA-XC321™, CEVA-XC323™, CEVA-XC324™, 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	Rm 709, Tower A SCC Financial Centre No. 88 First Haide Avenue Nanshan District Shenzhen 518064 China **Tel**: +86-755-8435 6038 **Fax**: +86-755-8435 6077	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	1-6-5 Shibuya SK Aoyama Bldg. 3F Shibuya-ku, Tokyo 150-0002 Japan **Tel**: +81-3-5774-8250	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

List of Figures

List of Tables

1. Introduction

1.1 Scope

This document describes the CEVA-XM4™ DSP core family Real-Time Trace (RTT) Wrapper architecture. It also describes how the RTT Wrapper fits into the CEVA-XM4 Real-Time Trace solution.

1.2 Related Documents

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

1. *ETM Interface, ARM DDI 0363A, Chapter 12*
2. *CoreSight Components and Systems, ARM DGI 0012A, Chapter 2*
3. *Embedded Trace Macrocell ETMv1.0 to ETMv3.4, Architecture Specification, ARM IHI 0014N*
4. *On-Chip Emulation (OCEM)*
5. *CEVA-XM4 On-Chip Emulation (OCEM) Reference Guide*

1.3 Overview

The purpose of the CEVA-XM4 Real-Time Trace is to facilitate program instruction zero impact on the CEVA-XM4 core's performance. The program instruction trace indicates all of the instructions executed by the CEVA-XM4 core, including whether each instruction passed its predicate code.

The CEVA-XM4 RTT solution combines an ARM ETM-R4 macrocell, the CEVA-XM4 core, and a CEVA-XM4 Trace Wrapper interface block. The Trace Wrapper maps the CEVA-XM4 core data to the ETM-R4 interface. The ETM-R4 macrocell architecture is ETM V3.3 compliant. Program flow reconstruction is performed by the debugger software resident on an external analyzer.

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

Important: *Because the RTT does not trace Vector Processing Unit (VPU) load or store accesses, Data Trace cannot be performed on VPU instructions.*

2. System Integration

2.1 Integration with CoreSight DAP

The CoreSight Debug Access Port (DAP) enables all trace components to be configured via the debug APB, independently of the system APB.

Figure 2-1 shows how the DAP connects to the ETM-R4 and the TPIU via the debug APB. The DAP enables the CEVA-XM4 system APB to access the debug APB via an APB multiplexer. The DAP allows for up to eight JTAG devices to be connected on eight individual JTAG AP buses. In this example, one DAP JTAG AP bus is used to drive the CEVA-XM4 OCEM.

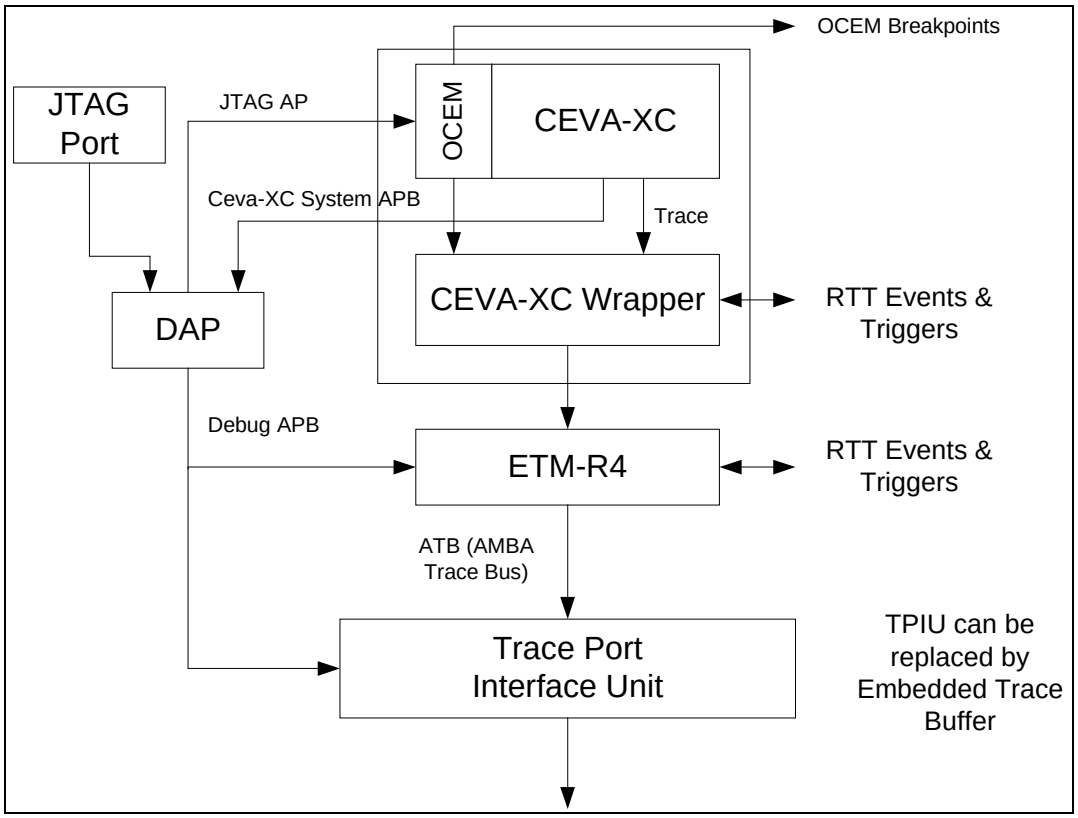

Figure 2-1: Trace Integration with DAP

2.2 CoreSight Compliant Multiple Core System

The use of the ARM ETM-R4 macrocell enables the CEVA-XM4 RTT to be incorporated in a CoreSight-compliant architecture for single core and multiple core debug, as shown in Figure 2-2, in which a single ETM-R4 is used. In this example, the CEVA-XM4 Wrapper is configured through either the CEVA-XM4 OCEM JTAG sequences or the CEVA-XM4 *in/out* instructions. Configuration of the ETMs and other CoreSight modules is performed via the CoreSight DAP and the debug APB.

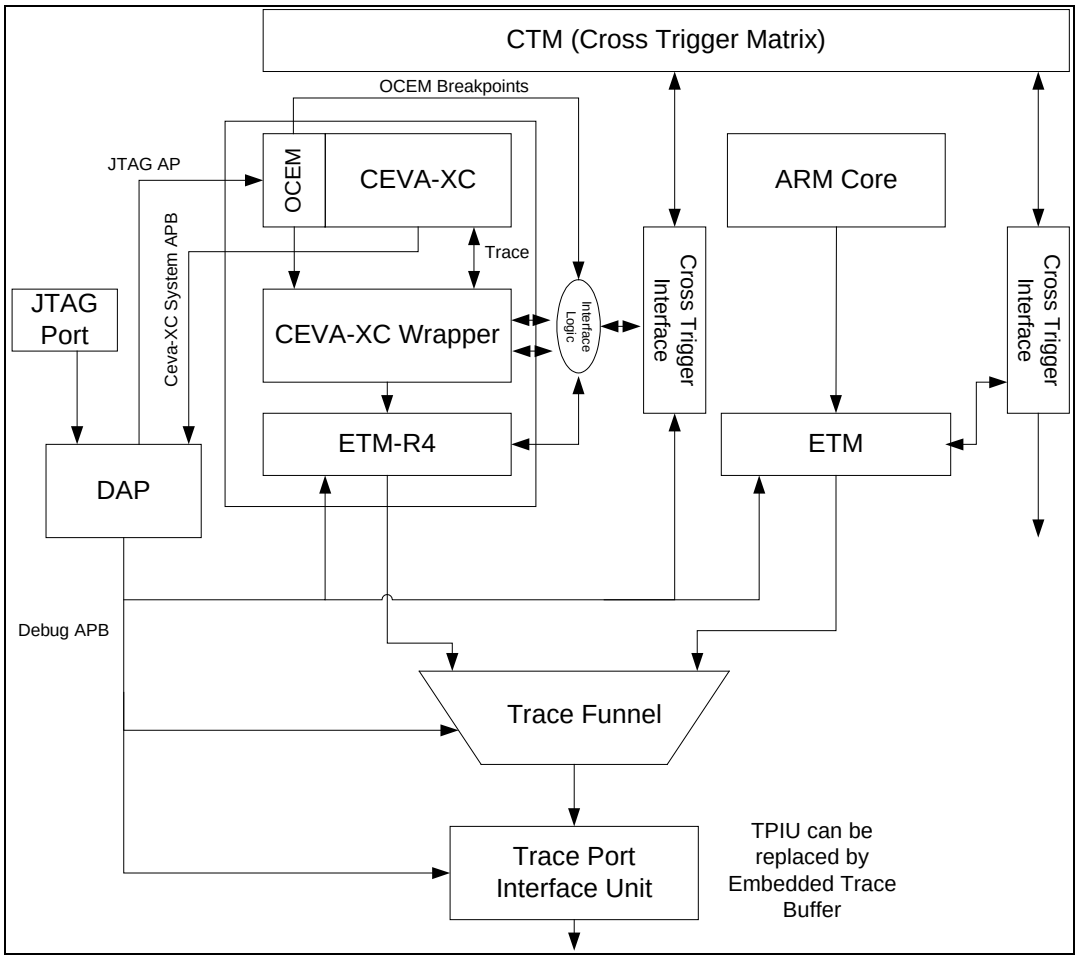

Figure 2-2: Multi-Core Real-Time Trace

2.3 OCEM Breakpoint Triggers

Several OCEM breakpoints are made available as primary outputs that can be used as additional trace triggers, as shown in Figure 2-3. The available OCEM breakpoints are described in Table 2-1.

Table 2-1: OCEM Breakpoint 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

2.4 PSU Interface

To prevent the loss of any trace data, the ETM-R4 must be informed if the core being traced is going into Standby mode so it can drain its trace FIFO. The ETM-R4 has pending/ready handshake signals for this purpose. When the ETM-R4 is external to the CEVA-XM4, these signals are interconnected as described in Table 2-2 and Figure 2-3. The Wrapper generates the *cevaxm4_w_etm_wfi_pending* request to the ETM-R4, which responds with *etm_wfi_ready_n* when it has drained its FIFO.

Table 2-2: External ETM-R4 Interconnection

ETM-R4 Ports	Mapped to CEVA-XM4 Ports	CEVA-XC I/O Direction	Description
nETMWFIREADY	etm_wfi_ready_n	Input	Indicates that the ETM-R4 FIFO is empty, and that the CEVA-XM4 can be put into Standby mode
ETMWFIPENDING	cevaxm4_w_etm_wfi_pending	Output	Indicates that the CEVA-XM4 is about to go into Standby mode, and that the ETM-R4 must drain its FIFO

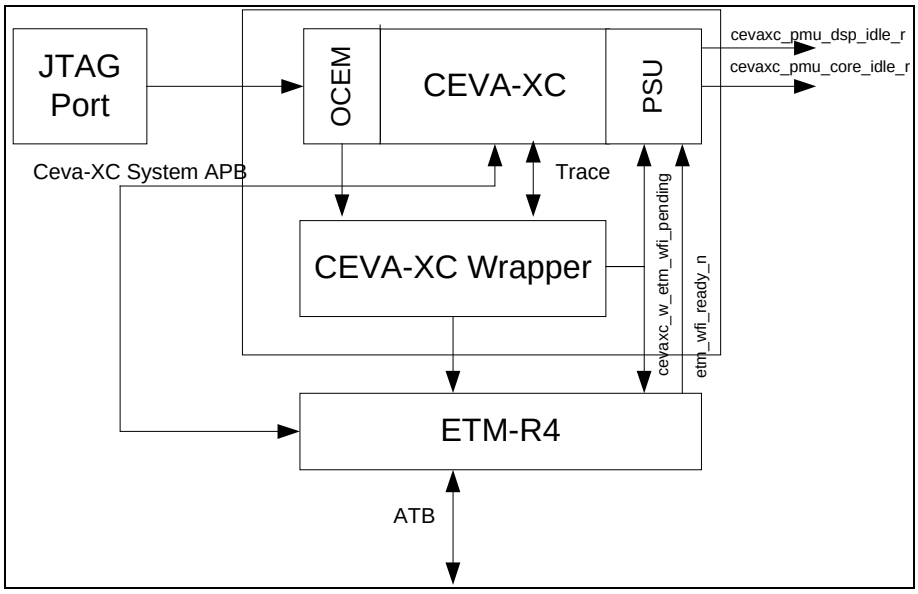

Figure 2-3: PSU Interface with External ETM-R4

Figure 2-4 shows an example of the timing of this handshake.

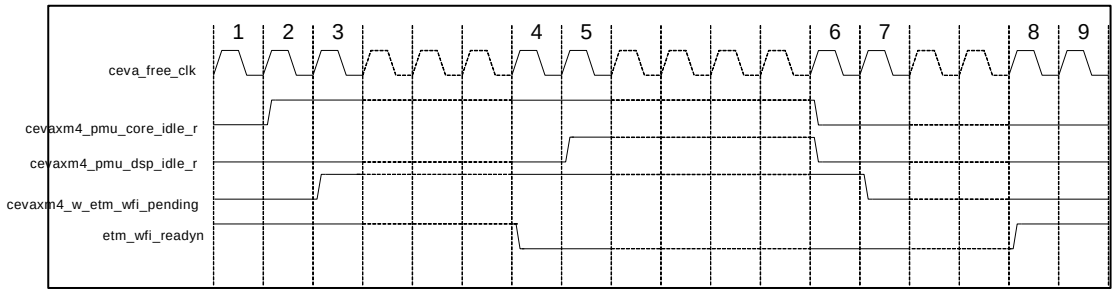

Figure 2-4: PSU Interface Handshake Example with External ETM-R4

- **Cycle 2**: The PSU puts the CEVA-XM4 into IDLE state and sets *cevaxm4_pmu_core_idle_r*.
- **Cycle 3**: The Wrapper sets *cevaxm4_w_etm_wfi_pending* to the PSU and ETM-R4 to indicate that the core under trace is going into Standby, Deep Sleep, or Shutdown modes.
- **Cycle 4**: A number of cycles later, ETM-R4 responds by clearing *etm_wfi_ready_n* to indicate that its FIFO is empty.
- **Cycle 5**: The PSU puts the CEVA-XM4 into Standby, Deep Sleep, or Shutdown mode, and then sets *cevaxm4_pmu_dsp_idle_r*.
- **Cycle 6**: Sometime later, the CEVA-XM4 recovers and the PSU clears *cevaxm4_pmu_core_idle_r* and *cevaxm4_pmu_dsp_idle_r*.
- **Cycle 7**: The Wrapper clears the *cevaxm4_w_etm_wfi_pending* to the PSU and ETM-R4.
- **Cycle 8**: When the ETM-R4 receives new trace data into its FIFO, it sets the *etm_wfi_ready_n*.

If the ETM-R4 is internal to the CEVA-XM4, there is no need for external connections, as shown in Figure 2-5. The *cevaxm4_etm_wfi_ready_n* indication is available externally for monitoring.

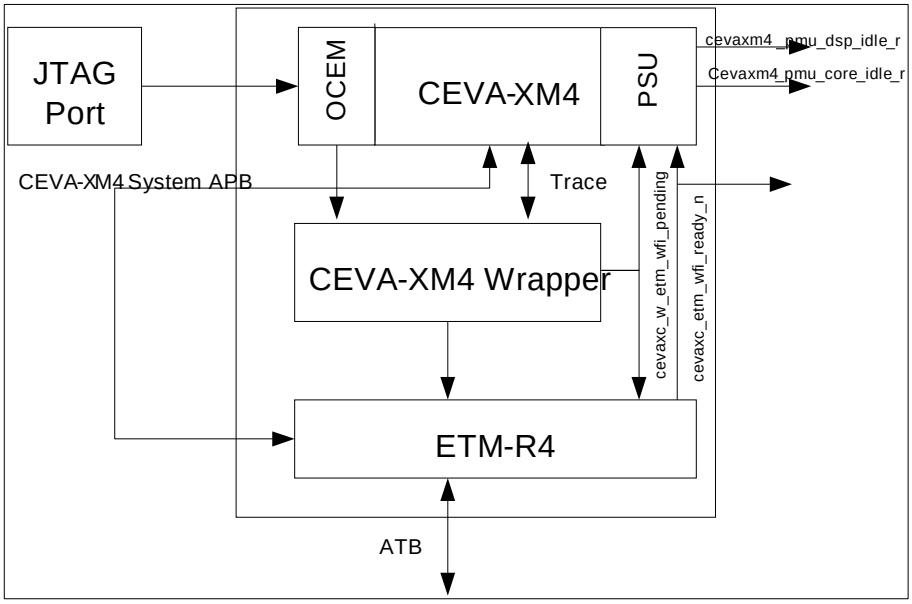

Figure 2-5: PSU Interface with Internal ETM-R4

2.5 High-Level Feature List

The CEVA-XM4 RTT solution provides the following features:

- **Program Instruction Flow Trace**:
  - Cycle- and non-cycle-accurate program address traces:
    - Cycle-accurate traces indicate the number of cycles required to execute each instruction under most trace conditions.
    - Non-cycle-accurate traces indicate the execution of an instruction without indicating the number of cycles required.
  - High compression achievable by tracing change-of-flow only.
  - Predicate traces facilitate condition code reconstruction by the debugger:
    - The CEVA-XM4 core supports instruction predication in which an instruction can be executed (or not) depending on the value of a specified predicate flag.
    - The predicate flags can be changed by dedicated instructions.
    - Knowing the predicate flags and the code image enables the debugger to determine if a predicated instruction was executed or not.
    - Predicate flags are sent to the ETM in normal data and ContextID packets.

- **Additional Features:**
  - Cycle-accurate traces are supported at the maximum frequency of the core.
  - Both the Trace Wrapper and ETM can be configured either through non-intrusive JTAG or through CEVA-XM4 instructions. The ETM is programmed through the CoreSight debug APB bus. The Wrapper has a dedicated programming interface to the CEVA-XM4 OCEM module.
  - Traces can be either accessed via a real-time high-speed trace port, or stored on-chip in an Embedded Trace Buffer for subsequent access at a lower clock speeds via JTAG or AHB. The trace port width and clock frequency is selected by the user.

    For more details, see Ref. [2].
  - The ETM trace logic provides user-programmable trace triggers and programmable trace filtering via the trace event resources.

    For more details, see Ref. [3].

3. Wrapper

3.1 Wrapper Block Diagram

Figure 3-1 shows a block representation of the Wrapper.

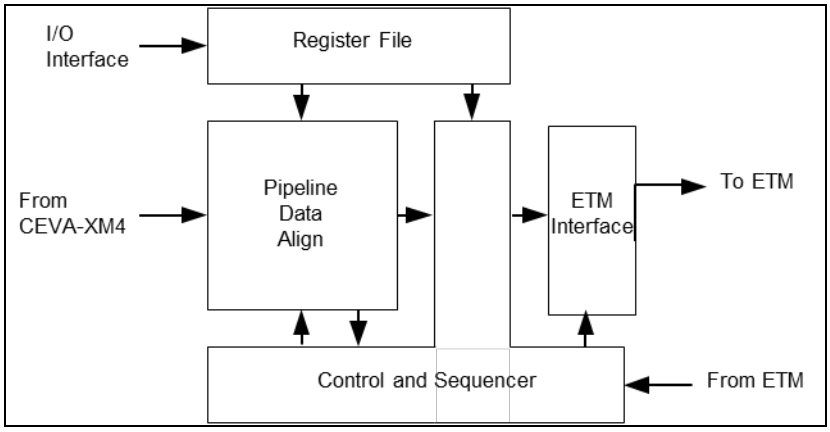

Figure 3-1: Wrapper Block Diagram

Table 3-1 describes each of the Wrapper's modules.

Table 3-1: Wrapper Modules

Module	Description
Register File	Contains the architectural registers that configure the Wrapper. The registers are loaded from a dedicated interface on the OCEM.
Pipeline Data Align	• Registers the data from the CEVA-XM4 core pipeline and aligns it to a single vector • Accepts program and control information from different stages of the CEVA-XM4 pipeline • Clocked at the same frequency as the CEVA-XM4 core • Can be stalled by a wait signal (*c_stall_r*) from the CEVA-XM4 core For more details, see Section 3.4.
ETM Interface	• Registers all information before it is sent to the ETM • Clocked at the same frequency as the CEVA-XM4 core but is never stalled (this facilitates cycle-accurate traces by the ETM) For more details, see Section 3.5.

Module	Description
Control and Sequencer	• Receives control signals from the OCEM • Monitors changes in the predicate bits • Sequences the transactions to the ETM • The output vector is mapped to the ETM-R4 macrocell signals. For more details, see Section 3.6.

The Wrapper does not operate when the core is in debug mode; instead, it is flushed and all data in the wrapper is lost.

3.2 Input Pin List

Table 3-2 describes the input pin list.

Table 3-2: Wrapper Input Pin List

Name	Size	Source	Description	Comment
clk_core	1	System	CEVA-XM4 Core clock Free running	
reset_n	1	System	Active low reset	Asynchronous
testmodep	1	System	Test mode indication	
tst_gatedclock	1	System	Used to bypass the clock gaters of the design during test mode	
c_stall_r	1	CEVA-XM4	The core is stalled.	The wrapper clock is gated when c_stall_r is asserted.
c_pc_a1_r	31	CEVA-XM4	Program counter from a1 pipe stage	
c_pc_valid_d2_r	1	CEVA-XM4	Indicates that an instruction at an address pointed to by *c_pc_d2_r* is executing in the core	If *c_pc_valid_d2_r* = **0**, then all related program and data information is invalid.
c_core_idle_r	1	PSU	Core Idle indication	
c_plp_r	1	CEVA-XM4	Core is currently executing a loop.	

Name	Size	Source	Description	Comment
c_branch_a1_r	2	CEVA-XM4	Program change of flow indication from the A2 pipe stage. Instruction at *c_pc_d2_r* is the branch instruction.	branch type (= {c_branch_d3_r c_branch_e1_r}) **00, 00** = Sequential **00, 01** = Indirect branch **00, 10** = Direct branch **00, 11** = Bkrep foldback[Note] **01, 00** = Indirect branch **01, 01** = Sequential **10, 00** = Direct branch **10, 10** = Sequential **11, 00** = Bkrep foldback [Note] **Others** = Sequential
c_branch_e1_r	2	CEVA-XM4	Program change of flow indication from the E2 pipe stage. Instruction at *c_pc_d2_r* is the branch instruction.	
c_ds_a1_r	1	CEVA-XM4	Current instruction is a delay slot.	Used to delay the branch indication to the ETM when delay slots follow a branch instruction
c_int_code_a1_r	4	CEVA-XM4	Indicates the interrupt type at the A2 pipe stage	**0000** = No exception **0001** = Reset **0010** = Boot **0011** = trap **0100** = trape **0101** = BI **0110** = nmi **0111** = int0 **1000** = int1 **1001** = int2 **1010** = int3 **1011** = int4 **1100** = vint **1101** = PABP **1110** = crcall **Others** = Reserved
c_pbkrep_a1	1	CEVA-XM4	Loop start indication	
c_prg_e3_r	15	CEVA-XM4	General predicate bits	
c_debug_r	1	CEVA-XM4	Core is in debug mode from the A1 pipe stage	Debug is aligned to the E5 pipe stage and resets the Wrapper internal pipeline.

Name	Size	Source	Description	Comment
o_addr	4	OCEM	4-bit address for accessing the wrapper internal registers	
o_data	32	OCEM	Data to be written to the wrapper internal registers	
o_valid	1	OCEM	Data and address valid indication	Indicates *o_addr* and *o_data* are valid in the current cycle.

Note: *Branch indication 11 is reserved for bkrep foldback (that is, the last instruction of a loop on all but the last iteration).*

3.3 Output Pin List

Table 3-3 describes the output pin list.

Table 3-3: Wrapper Output Pin List

Name	Size	Destination	Description	Comment
w_etm1iactl_r	14	ETM	Instruction interface control	See Table 3-4
w_etm1ia_r	31	ETM	Instruction address	Derived from *c_pc_d2_r*
w_etm1da_r	32	ETM	Data address	Derived from *c_ls0_addr_d3_r* and *c_ls1_addr_d3_r*
w_etm1dd_r	64	ETM	Read/write data from LS unit	Derived from *c_ls0_drd_e1_r*, *c_ls0_dwr_e3_r*, *c_ls1_drd_e1_r*, and *c_ls1_dwr_e3_r*
w_etm1dctl_r	12	ETM	Data interface control	See Table 3-5
w_etm_cid_r	32	ETM	CEVA-XM4 predicates in bits [10:0]	The status of the predicate flags to be sampled by ETM; *w_etm_cid_r*[31:11]=21'b0
w_debug_e5	1	ETM	Core in debug mode	Indicates that the core is in debug mode
w_dataout	32	OCEM	Contents of the register at address *o_addr*	Read data is sampled by the OCEM.
w_etm1_wfi_pending	1	ETM + PSU	Indicates to the ETM that the core is in Idle mode and going to Standby mode, so the ETM must drain its FIFO	When the ETM FIFO is empty, it de-asserts *etm_wfi_ready_n* to the PSU.

Table 3-4: w_etm1iactl_r Signal Description

Name	Bits	Description	Notes	Qualified by	Derived from
iexcancel	13	Used with iextype	{iexcancel, iextype} **x0000** = No exception **11000** = Reset **01100** = trape, Bi, PABP **01110** = trap, nmi, int0, int1, int2, int3, int4, vint, crcall **01111** = Boot	iexception & ivalid[0]	*c_int_code_a1_r* See Table 3-2
iextype[3:0] Note1	12:9	Interrupt type indication			
iexception	8	Current instruction is at an interrupt vector.	*c_pc_d2_r* is the target address.	ivalid[0]	*c_int_code_a1_r*
ibrtype[1:0] Note3	7:6	Indicates the branch status	**00** = No branch **01** = Indirect branch **10** = Direct branch **11** = Reserved	ivalid[1:0] != 00	*c_branch_a1_r* and *c_branch_e1_r*
iccfail[1:0]	5:4	Not used	00	ivalid[1:0] != 00	
i32bit Note2	3	16-bit or 32-bit instruction	**0** = 16-bit instruction **1** = 32-bit instruction	ivalid[1:0] != 00	
itbit	2	ARM/Thumb state Trace CEVA-XM4 as Thumb	**0** = ARM (not used) **1** = Thumb	ivalid[1:0] != 00	
ivalid[1:0]	1:0	Indicates the current instruction status	00 = No instruction 01 = Single instruction 10 = Reserved 11 = Dual instruction		*c_pc_valid_d2_r*

Notes:

1. *The CEVA-XM4 interrupt type can be inferred from the target address (except in the case of Boot and vint (programmable vector)).*
2. *i32bit and itbit affect the packet output from the ETM-R4 during a branch and exception. For CEVA-XM4, itbit=1 and i32bit=1 will enable a 31-bit c_pc_d2_r trace.*

   For more details, see Ref. [1].
3. *The branch is indicated with the address of the branch instruction.*

Table 3-5: Description of w_etm1dctl_r signal

Name	Bits	Description	Notes	Qualified by
dabort	11	Negated (not used)	0	dvalid
dtype[2:0]	10:8	Data type transferred	**000** = Normal data **111** = Context ID **Others** = Reserved	dvalid
dfail	7	Negated	0	dvalid
dnseq	6	Indicates if the data address is sequential or non-sequential from the last address	1	dvalid
dbigend	5	Negated. Data is little-endian.	00	dvalid
dsize[2:0]	4:2	Data transfer size, in bytes	dsize+1 bytes	dvalid
dnrw	1	Data transfer direction	**0** = Data read **1** = Data write	dvalid
dvalid	0	Data valid indication	**1** = Data valid	

3.4 Pipeline Data Align

The Pipeline Data Align module is comprised of a register file that aligns the CEVA-XM4 core data to a single vector. Indirect branch indications from the core are output to the ETM Interface if the next instruction from the core is not a delay slot. If the next instruction is a delay slot, then the indirect branch indication is delayed until the last delay slot associated with the indirect branch.

The *c_debug_r* indication from the core is aligned and causes all of the stages in the Pipeline Data Align module to be reset.

All Wrapper pipe stages are stalled when the CEVA-XM4 core is stalled. However, if *dmss_core_no_wait*/*dmss_vu_no_wait* is asserted while *c_stall_r* is asserted, then the Wrapper must sample and hold the write data presented on *c_ls0[1]_dwr_e3_r*. The Wrapper then aligns this write data with the write access parameters when *c_stall_r* is negated.

3.5 ETM Interface

The vector transferred to the ETM on clk_core is:

V=I{w_etm1iactl_r, w_etm1ia_r}+D{ w_etm1da_r, w_etm1dd_r, w_etm1dctl_r}

Vectors are transferred by the Wrapper to the ETM as Thumb instructions to enable 31-bit program address traces. The bit assignments for the instruction vectors I and D are shown in Table 3-6 and Table 3-7. Data is packed into *w_etmdd_r* in little-endian format.

Table 3-6: ETM Instruction Vectors

Vector(I)	i32bit	itbit	iccfail	ibrtype	ivalid	Notes
IAN	x	x	x	x	00	Instruction not valid, no instruction executing, or core stalled
IS	0	1	00	00, 01, or 10	01	Single-issue instruction

Table 3-7: ETM Data Vectors

Vector(D)	dtype	dsize	dvalid	etmda	ivalid	Notes
D	000	000 to 111	1		!=10	Normal data, up to 64 bits
64D	000	111	1	00,01 or 10	!=10	
CID	111	011	1	x	!=10	ContextID Data (Predicate flags) See Table 3-8
PD	000	001	1	pred_addr	!=10	Predicate flags, 2 bytes See Table 3-8 Test mode: if cid_cnt[3:0]=0, then PD is replaced by CID.

3.6 Control and Sequencer

The output from the Pipeline Data Align module (the instruction atom) is mapped by the Control and Sequencer module onto a single ETM vector that is transferred to the ETM in one clock cycle.

The predicate flags from the core are registered, and a change in one or more of the flags is detected. The predicate flags can be sent to the ETM either as normal data with a data address specified in the **PRED_ADDR** register, or as a ContextID. If the *cid_en* bit is asserted and a ContextID has not been transferred to the ETM in the last *cid_cnt*[3:0]+1, then the predicate flags are transferred to the ETM as a ContextID. In all other cases the predicate flags are transferred to the ETM as normal data.

The predicate bits are packed as indicated in Table 3-8.

Table 3-8: Transferring Predicate Bits to ETM#1

Vector	w_etm1dd_r	
Bits	63:15	14:0
Contents	0	Prg[14:0]

The debugger software can use the contents of the ContextID packet and the data value traced at the address specified by the **PRED_ADDR** register to evaluate the predicate flag of each subsequently traced instruction.

Each CEVA-XM4 instruction packet is deconstructed into a set of basic atoms, as listed in Table 3-9.

Table 3-9: Logical Expressions for Instruction Atoms

Atom	Logical
IN	c_pc_valid_d2_r=0
I	c_pc_valid_d2_r=1
P	Predicate {prg, prs, pra}[T] != {prg, prs, pra}[T-1], clock cycle T

The ContextID must be sent with a valid instruction indication.

3.7 ETM-R4 Packets

The information in Table 3-10 and Table 3-11 enables the design of debugger software, and is the relevant subset of the tables in the ETMv3 Signal Protocol (as described in Ref. [3]). The contents of the packet headers might differ from the native ETMv3.3 interpretation.

Table 3-10: ETM-R4 Packet Types

Packet Header	Value	Payload (Max Bytes)	Type	Notes
Branch Address	Cxxxxxx1	5	Instruction	**C** = Another byte follows **xxxxxx** = Instruction address[6:1]
A-sync	00000000	0	Sync	Alignment synchronization
Cycle Count	00000100	5	Instruction	Payload contains cycle count {0, 1, 2^{32}}
I-sync	00001000	14	Sync	Instruction flow synchronization
Trigger	00001100	0	Trace Port	Cycle-accurate trace trigger
I-sync with Cycle Count	01110000	19	Sync	Instruction flow synchronization
Normal Data	00A0SS10	9	Data	**A** = Address expected (5 bytes) **SS** = Data size (4 bytes)
Data Suppressed	01100010	0	Data	Output when ETM FIFO is full in place of data packets
Ignore	01100110	0	Trace Port	Filler byte, no information
Value Not Traced	011A1010	5	Data	**A** = Address expected (5 bytes)
ContextID	01101110	4	Instruction	Predicate information, 2 bytes required See Table 3-8
P-Header	1xxxxxx0	0	Instruction	

Table 3-11: ETM-R4 P-Header Types

P-Header	Value	Notes
Non-Cycle Accurate		
Format 1	1NEEEE00	**EEEE** = Number of instructions executed **N** = 0
Cycle-Accurate		
Format 1	1N0EEE00	**EEE** = Number of single-cycle instructions executed sequentially **N** = 0
Format 3	1E1WWW00	**WWW+1** = Number of cycles with no instructions executed, followed by: **E** = One single-cycle instruction executed
Others	Not expected to be used, or reserved	

3.8 Debugger Notes

- The debugger must assume sequential flow or direct branches depending on the instruction executed, unless an ETM packet indicates a change of flow.

  The debugger can determine the direct branch target addresses from the code image. In normal usage, and except for normal synchronization packets from the ETM (I-sync), the instruction address is sent from the ETM only when an indirect branch or interrupt (exception) occurs. The instruction address is sent from the ETM in a Thumb branch packet.
- The user can select whether to trace CEVA-XM4 predicate flags. The user can also select to trace the predicate flags either in normal data packets or in a combination of ContextID and normal data packets. Accurate trace reconstruction is achieved by determining if a traced instruction was executed or not according to the applicable predicate flag.
- When tracing predicate flags in normal data packets, the user must select a logical data address to dedicate to the CEVA-XM4 predicate flag trace. The address selected should be a data address.

  The user must configure the ETM ViewData function to trace data values at the specified address. The predicate flags are then output from the ETM as a normal data packet while TraceEnable is active.

- At the start of a trace run, the ETM outputs an I-sync packet that contains the first instruction address of the trace and, if enabled, the most recent ContextID. If the predicate flags have changed since the most recent ContextID was sent by the wrapper to the ETM, then one or more predicate flags extracted from the ContextID will be incorrect. This uncertainty continues until either a normal data packet containing predicate flags or a new ContextID is received.

  If the program flow during the window of uncertainty includes many direct branches, then code reconstruction might be difficult. In such cases, alignment is achieved when the next I-sync or direct branch occurs.
- The ETM generates I-sync packets periodically, upon a trace being enabled and when a trace has restarted following ETM FIFO overflow.
- Each traced CEVA-XM4 instruction packet maps to a single instruction executed field in the ETM P-headers. For each traced instruction packet and wrapper configuration, the debugger must correctly unpack the P-headers and maintain alignment with the CEVA-XM4 program counter.
- The loop foldback indication can be programmed by the user to be either an indirect branch or a combination of direct and indirect branches. The latter results in higher trace compression.
- When a combination of direct and indirect branching is used, the indirect branch is restricted to the first foldback of an inner loop and the first foldback following a change of flow during loop execution. The foldback of outer loops can be made indirect branches by setting *cmp_cfg*[6]. All foldbacks can be made indirect branches by setting *cmp_cfg*[7].

4. Glossary

Table 4-1 defines the acronyms used in this document.

Table 4-1: Acronyms

Term	Definition
CID	Context ID
DAP	Debug Access Port
DSP	Digital Signal Processor
ETM	Embedded Trace Macrocell
FIFO	First In, First Out
JTAG	Joint Test Action Group
OCEM	On-Chip Emulation Module
PSU	Power Scaling Unit
RTT	Real-Time Trace
SDT	Software Development Tools
VPU	Vector Processing Unit