CEVA-XM4™ TCE Integration Guide

Rev 1.1.3.F

June 2016

Documentation Control

History Table

Version	Date	Description	Remarks
1.1.1.6	December 2015	Beta release	No change
1.1.1.A	January 2016	Official release	
1.1.1.F	February 2016	Official release	
1.1.2.F	March 2016	Official release	
1.1.3.F	June 2016	Official release	

Disclaimer and Proprietary Information Notice

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

1 Introduction

Modern DSP systems use Multiple DSP Cores and hardware accelerator to handle the high computational requirements of advanced communication protocols. CEVA-XM4™ DSP implements special features enabling the user to add Tightly Coupled Extensions (TCE) serving as an hardware or software accelerators to the system without overloading the DSP with the data traffic management to and from the TCE.

1.1 Scope

This document describes basic methods for designing TCEs, connecting them to a CEVA-XM4 DSP and controlling the data traffic between the DSP and the TCEs.

Many other methods and flows can be implemented using the CEVA-XM4 DDMA Manager or using special hardware to comply with special requirements. These methods are not part of the scope of this document.

1.2 Applicable Documents

The following documents relate to the CEVA-XM4 Specification:

- CEVA-XM4 Architecture Specification

1.3 Conventions

The following terms are frequently used throughout this document, and are explained here for ease of reference:

- *CPM* – CEVA-XM4 core programming model
- *DDMA – CEVA-XM4 data DMA*
- *Channel* – a channel is defined as an information route from a source to a destination. Data flowing through a channel reaches the destination at the same order it was read from the source.
- *Queue* – the term queue is used for describing a DMA task Queue used for holding DDMA task descriptors to be executed sequentially by the DDMA. A queue is usually used for holding DDMA tasks related to a single channel (although a user is free to mix several channel tasks into a single queue).
- *DMSS* – CEVA-XM4 Data Memory Subsystem
- *QMAN* – CEVA-XM4 DDMA Queue Manager
- *IDM* – Internal Data Memory (CEVA-XM4 L1 data memory)
- *TCE* – Tightly Coupled Extension. Hardware of software extension to the CEVA-XM4.

1.4 Memory Subsystem Overview

The CEVA-XM4 supports up to four gigabytes of unified memory space, and has up to eight separate physical interfaces to the data memory. This enables the DSP core to simultaneously access the SoC memories in parallel to multiple Tightly Coupled Extensions (TCE).

The MSS contains standard interfaces for connecting the core to external devices and/or peripherals. These interfaces include up to three AXI master ports, up to four AXI slave ports and an APB3 port. All ports are fully compliant with Advanced Micro-controller Bus Architecture (AMBA).

The CEVA-XM4 DMSS uses Data DMA (DDMA) to transfer data between the local memory and an external memory, without interfering with core execution. In addition, the DMSS implements a special DMA Queue manager (QMAN) mechanism that enables the user to activate the DDMA without core intervention and without having to use real-time software.

1.5 DDMA Queue Manager

The CEVA-XM4 DDMA queue manager supports the following main features:

- Up to eight independent queues
- Handles DDMA tasks using queues
- Dispatch tasks to the CEVA-XM4 DDMA
- Uses prioritized scheduling to guarantee QoS
- Offloads DDMA handling from the DSP core

For further information on DDMA Manager refer to CEVA-XM4 MSS Architecture Specification.

2 TCE Interface Definition

Defining TCE interfaces require selection of the following factors:

- Data traffic
- Buffer
- Port
- Flow control

2.1 TCE Data Traffic

The data traffic between the CEVA-XM4 and the TCE is composed of TCE Ingress data and TCE Egress data as illustrated in Figure 2-1.

The Ingress data contains all data required by the TCE Core for performing the data processing. The Ingress data can be obtained from a single source in the system or from several sources. A data source may be another TCE, a task running within a CEVA-XM4 core, or any other system component producing data packets.

The Egress data contains the output data from the TCE Core to memory external to the TCE. The Egress data can be directed to a single destination or it can be fragmented and directed to several destinations.

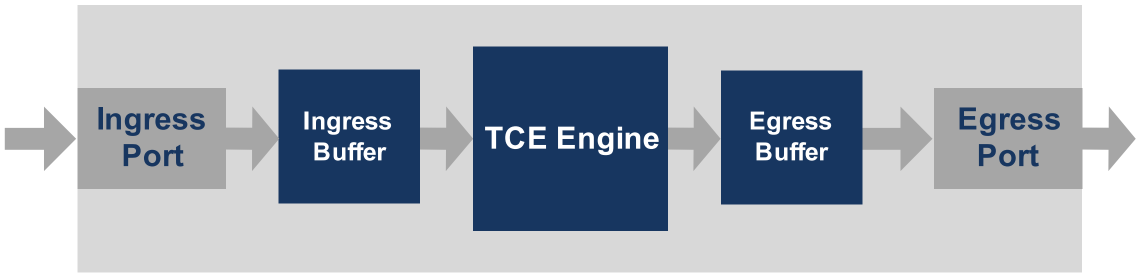

Figure 2-1. TCE Data Traffic

2.2 Buffer Types

A TCE requires an Ingress buffer to allow bursty ingress data traffic, while providing elasticity to the system demands. A TCE also requires an Egress buffer to absorb contentions on the egress bus due to traffic contentions and system latency considerations. An optimal design of a TCE ensures that the TCE Core is constantly operating. The ingress buffer should have the next input data to prevent TCE starvation, and the egress buffer should have available space to prevent TCE back-pressure.

This document presents one type of buffer, a double buffer (that can be scaled to a triple, quadruple, or larger buffer structure)

2.2.1 Double Buffer

A double buffer is composed of two buffers which are accessed in a Ping-Pong fashion – where new data is written into one buffer while the other buffer is being read.

To ensure data is always ready at the ingress buffer, the time consumed by moving a new packet to the ingress buffer must be less than the time used by the TCE to process the previous packet. If the TCE finishes processing the previous packet before the next packet is written to the ingress buffer then the TCE will be in starvation.

To ensure the egress buffer always has space available to be written with new data, the time consumed by moving a data packet from the egress buffer to memory external to the TCE must be less than the time used by the TCE to generate the next packet. If the TCE finishes processing the next packet before the egress buffer has moved the previous data packet then the TCE is stalled.

In cases where the size of the packets varies rapidly then the constraints above may result in low utilization of the TCE. For example, if the TCE is processing a small packet that is followed by a large packet, then the TCE may complete the processing of the small packet before the next ingress buffer is ready. This will result in TCE starvation. A similar example can occur with the egress buffer if the TCE has processed two consecutive large packets followed by a small packet, then the TCE may complete the processing of the small packet before a single egress buffer is emptied. This will result in the TCE being stalled.

2.3 Port Types

Each TCE may contain an ingress port and an egress port that connects to the DSP processor. Each port can be either a master or a slave.

For the ingress traffic the TCE can use either a slave port for receiving write transactions from an external master or a master port for sending read transactions to an external slave.

For the egress traffic the TCE can use either a slave port for receiving read transactions from an external master or a master port for sending write transactions to an external slave.

Write transactions usually can achieve better bus utilization with lower latencies than read transactions. Therefore, it is generally preferred to use an ingress slave port and an egress master port.

However, in some specific cases a user may benefit from using read transactions, rather than write transactions. The following are examples of two such cases:

- A TCE that is shared by several DSP cores, where the ingress packets can come from any number of source cores. In this case the use of the TCE requires arbitration, where a complete packet is received from core A before the next packet is received from core B. Using an ingress master read port enables the TCE to perform the arbitration internally and to select the source of the next ingress packet.
- A TCE that is shared by several DSP cores, where the egress packets can be used by any number of consuming cores. Using a slave read port in this case allows several DSP cores to read from the egress buffer. The cores compete on reading the next egress packet using a semaphore and the granted core can read the packet from the TCE.

2.4 Flow Control

Flow control is required when data is moved from a source to a destination in order to prevent under-run at the source or overflow at the destination.

This section will describe the following flow control methods:

- Packet-ready indication
- Read-done end indication

2.4.1 Packet-Ready Indication

The packet-ready indication is used when a master at the destination device is reading data from the source device. The source device informs the destination device about packet availability using a Packet-Ready indication as shown in Figure 2-2.

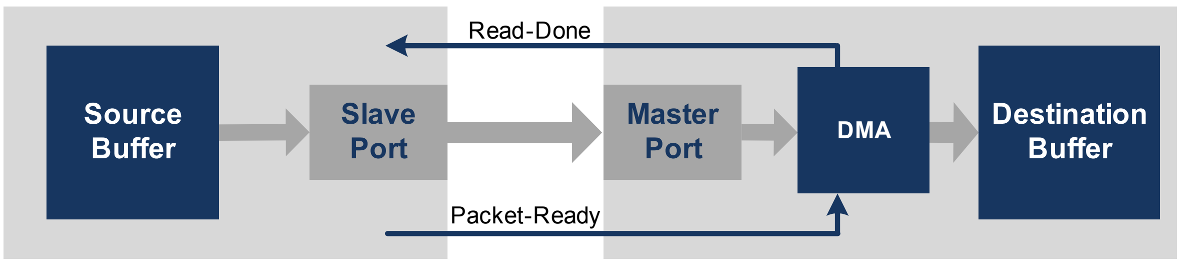

Figure 2-2. Packet-Ready Flow Control

Once the master at the destination device completes reading the packet it can inform the source using a read-done indication. This enables the source to release the memory consumed by the packet and reuse it for new data. Optionally the source device can detect that the packet was completely read when the last byte of the packet is read. Using the packet-ready indication the packet transfer cannot start before the entire packet is available at the source.

This flow control is mostly used when the destination device is reading from a double buffer at the source device.

2.4.2 Buffer-Ready Indication

The buffer-ready indication is used when a master at the source device is writing data to the destination device. The destination device informs the source device that memory was allocated for the next packet by using a buffer-ready indication as shown in Figure 2-3.

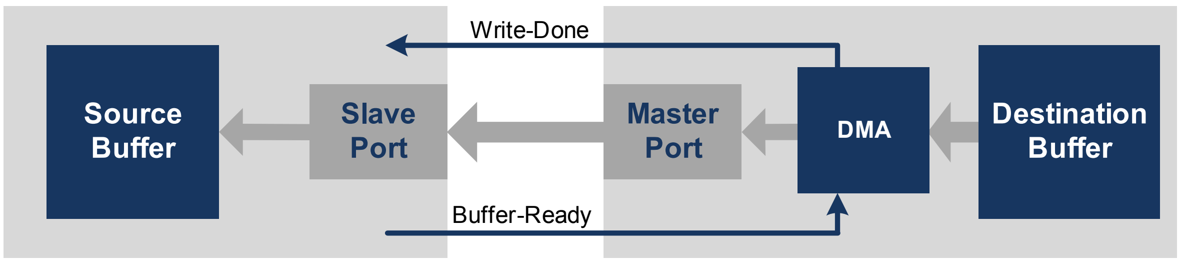

Figure 2-3. Buffer-Ready Flow Control

Once the master at the source device completes writing the packet it can inform the destination device using a write-done indication. This informs the destination device that the entire packet was transferred and it can start consuming the packet. Using the buffer-ready indication the packet transfer cannot start before the destination has released sufficient memory for the next packet.

This flow control is mostly used when the source device is writing to a double buffer at the destination device.

3 QMAN Flow Control

The queue manager implemented at the CEVA-XM4 is used for downloading the CEVA-XM4 input data into its IDM and for uploading the output data from the CEVA-XM4 IDM. By that the queue manager offloads the DDMA handling from the DSP core.

Each queue is usually used for moving data packets from a single source to a single destination. The task descriptors can be pushed to the queue by the DSP or by external device even before the data is available at the source. In addition, each packet-transfer task can be preceded or followed by a message task that can send 32-bit messages to the CEVA-XM4or to devices external to it.

When the packet size used is fixed the overhead of pushing new tasks to the queue can be reduced by reusing the task descriptors in the queue. In this case the user writes all task descriptors to the queue and sets the queue write pointer to an address larger than the last queue address. The queue read pointer will never be equal to the queue write pointer (due to pointer wrap around at the last queue address) and therefore the QMAN will continue reading the same tasks from the queue in a loop.

The following CEVA-XM4 and QMAN flow control options are described in this section:

- DSP Input Double buffer at IDM
- DSP Output Double buffer at IDM

3.1 DSP Input Double Buffer at IDM

illustrates a system with DDMA (controlled by QMAN0) writing data to double buffer at IDM and the DSP core reading data from the same double buffer. This case corresponds to the Packet-Ready indication flow control described at section 2.4.2 with the double buffer at the IDM as the destination buffer.

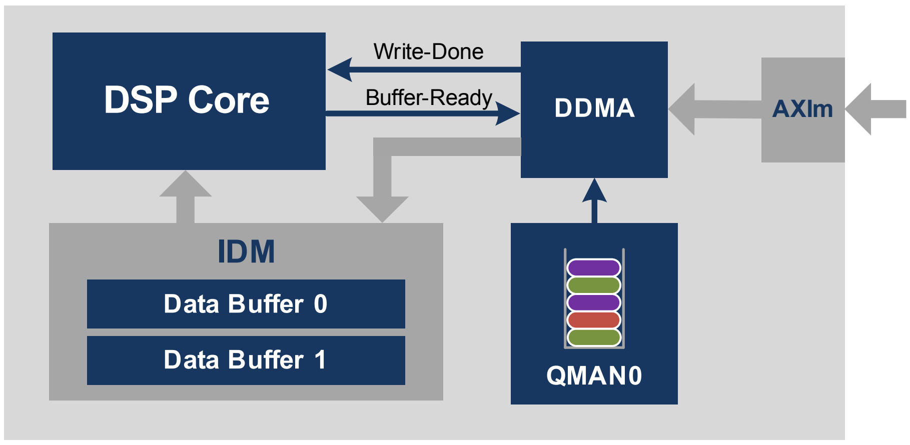

Figure 3-1. DSP Input Double Buffer

3.1.1 Buffer-Ready Indication

The DSP core can send a buffer-ready indication (after reading a packet from the double buffer) to the QMAN by simply incrementing the QX_EN_CNT (writing the value of 0x1_0001 to QX_DESC_EN_INCx_REG at the CEVA-XM4 MSS CPM). This will enable the next task at the QMAN and will thereby allow the DDMA to move the next packet to the evacuated buffer.

In some systems the QMAN use the QX_EN_CNT for receiving buffer-ready indication from a TCE (as described in section 4.2.1) and therefore cannot be used also for the buffer-ready indication internally. In these cases the DSP can use the queue write pointer for enabling a task (a task is not considered written until the queue write pointer is incremented). The descriptor push configuration registers (defined at the CEVA-XM4 MSS Architecture Specification) should not be used since it automatically increment the queue write-pointer at the time of writing the task to the queue. Instead, the tasks should be written directly to the queue without incrementing the queue write pointer. The DSP then sends the buffer-ready indications to the QMAN by incrementing the queue write-pointer. This enables enable the QMAN to write the next packet to the released buffer.

The QMAN is designed to disable a queue once tasks are enabled and no task descriptor is available. This prevents the QMAN from polling on the write pointer. Such a case can happen if a TCE increments the QX_EN_CNT before the DSP releases a packet from the double buffer and increments the write pointer. Therefore after incrementing the write pointer the DSP is also required to re enable the queue by re writing the QX_EN_DEPTH register.

3.1.2 Write-Done Indication

The QMAN can send a write-done indication to the DSP core using a message DDMA task. A message DDMA task can write a predefined value to an address at the CEVA-XM4 IDM or at an address external to the CEVA-XM4. Each packet transfer task should be followed by a message task for sending the write done indication.

For systems requiring the CEVA-XM4 DSP to be interrupted when receiving a write done indication the DDMA message task can write to an external interrupt controller for generating the interrupt.

For systems requiring the CEVA-XM4 DSP to read the number of write done indications sent by the QMAN to the DSP core (polling) the message task writes the packet number to a predefined address at the IDM. A two bit wrap-around numbering is sufficient for a system with double buffer (three bits for systems with 3 to 4 buffers, four bits for systems with 5 to 8 buffers, etc.). The DSP can check the number of write done indications by comparing (using unsigned subtraction) the last read value with the current value. The difference between the current packet number and the previous packet number indicates the number of write done indications sent to the DSP core since the last read of the packet number.

3.2 DSP Output Double Buffer at IDM

Figure 3-2 illustrates a CEVA-XM4 system with the DSP core writing data to a double buffer at IDM and the DDMA (controlled by QMAN0) reads data from the same double buffer. This case corresponds to the Packet-Ready indication flow control described at section 2.4.1 with the double buffer at the IDM as the source buffer.

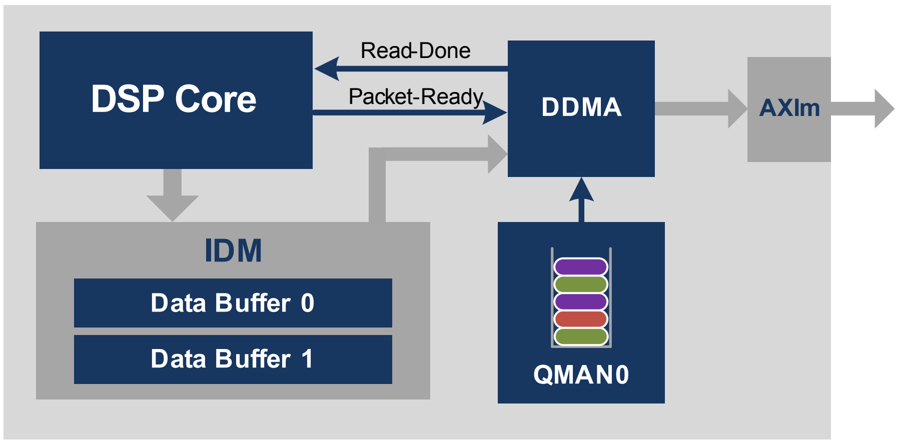

Figure 3-2. DSP output Double Buffer

3.2.1 Packet-Ready Indication

The DSP core can send a packet-ready indication (after writing a packet to the double buffer) to the QMAN by simply incrementing the QX_EN_CNT (writing the value of 0x1_0001 to QX_DESC_EN_INC_REG at the CEVA-XM4 MSS CPM). This will enable the next task at the QMAN and will thereby allow the DDMA to read the next packet from the double buffer.

In some systems the QMAN use the QX_EN_CNT for receiving packet-ready indication from a TCE (as described in section 4.1.1) and therefore cannot be used also for the packet-ready indication internally. In such cases the DSP can use the queue write pointer similarly to the way described in section 3.1.1.

3.2.2 Read-Done Indication

The QMAN can send a read-done indication to the DSP core using a message DDMA task. The method of sending this message is similar to the method of sending a write-done indication described in section 3.1.2.

4 Connectivity Flows

The connectivity flow between the CEVA-XM4 and a TCE depends on the type of traffic (ingress or egress), and the type of the port (master read/write port or slave read/write port).

The following flows are described in this section.

- TCE Ingress Master Reading from a Double Buffer
- TCE Ingress Slave using Double Buffer at TCE
- TCE Egress Master writing to a Double Buffer
- TCE Egress Slave Using Double Buffer at TCE

4.1 TCE Ingress Flows

4.1.1 TCE Ingress Master

Figure 4-1 illustrates an example of a basic flow for transferring data packets from CEVA-XM4 to a TCE with an ingress master at the TCE reading from double buffer at the CEVA-XM4. The packet data flow use the Packet-Ready indication flow control described at section 2.4.1.

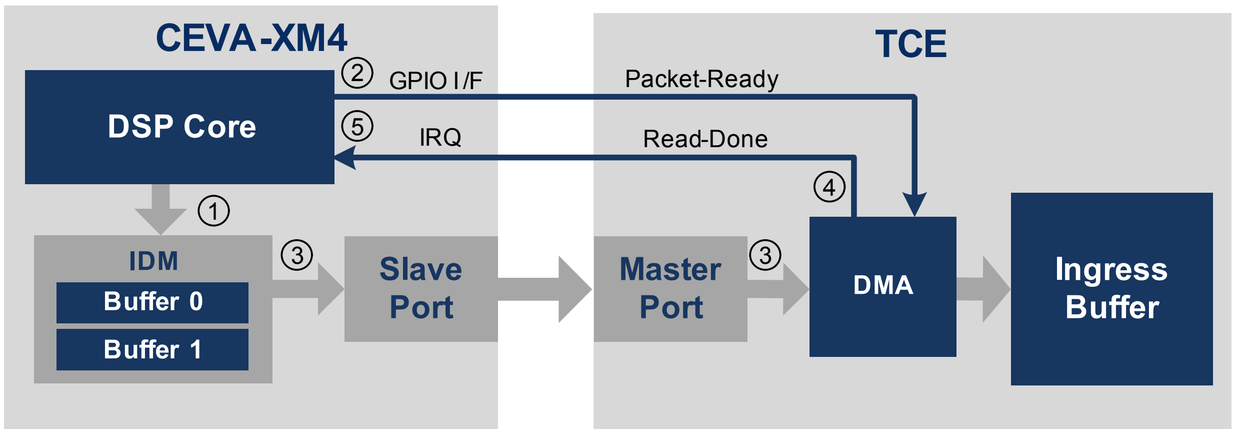

Figure 4-1. TCE Ingress Master

The following steps form the packet data flow, where the numbering in the figure corresponds to the steps below:

1. TCE ingress packet is written by the CEVA-XM4 DSP core to buffer0 or buffer1.
2. DSP core outputs a packet-ready indication to the TCE (for example using GPIO signals).
3. TCE reads the packet from buffer0 or buffer1.
4. TCE outputs packet read done indication once it completes reading the packet.
5. DSP core receives the packet read-done indication as an interrupt, enabling writes to the next buffer.

Note that the read-done indication via interrupt is just an example and different methods are also valid like, for example, polling of the core on external register that implement a bi-directional counter that increment by the TCE and decrement by the CEVA-XM4.

4.1.2 TCE Ingress Slave

Figure 4-2 illustrates an example of a system with CEVA-XM4 DDMA writing packets to a TCE with a double ingress buffer. The packet data flow use the packet write-done flow control described at section 3.1.1. The write-done indication can be generated automatically by the TCE using packet size written at the packet header and comparing it with the number of bytes written to the ingress buffer.

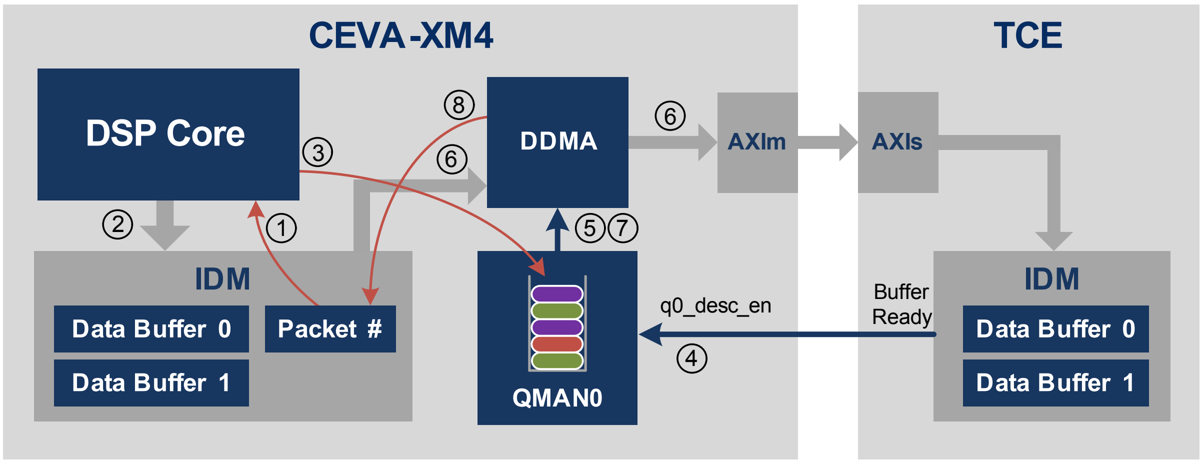

Figure 4-2. TCE Ingress Slave

The following steps form the packet data flow, where the numbering in the figure corresponds to the steps below:

1. DSP core checks for a free buffer (based on DSP Output Double Buffer at IDM flow, refer to section 3.1.23.2.2).
2. DSP core writes packet data to the next buffer at the double buffer.
3. DSP core increments queue0 write pointer by two (data move task and the following read done message task to update packet number).
4. TCE releases one of its ingress buffers and enables next two tasks at QMAN0.
5. QMAN0 loads packet transfer task to DDMA.
6. DDMA moves the packet to the TCE.
7. QMAN0 loads read done message to DDMA.
8. DDMA writes packet number.

The system should be initialized with the first two tasks enabled (the TCE can receive two new packets).

4.2 TCE Egress Flows

4.2.1 TCE Egress Master

Figure 4-3 illustrates an example of a basic flow for transferring data packets from TCE to CEVA-XM4 using a direct write egress master.

The packet data flow use the buffer-ready flow control described at section 3.1.1.

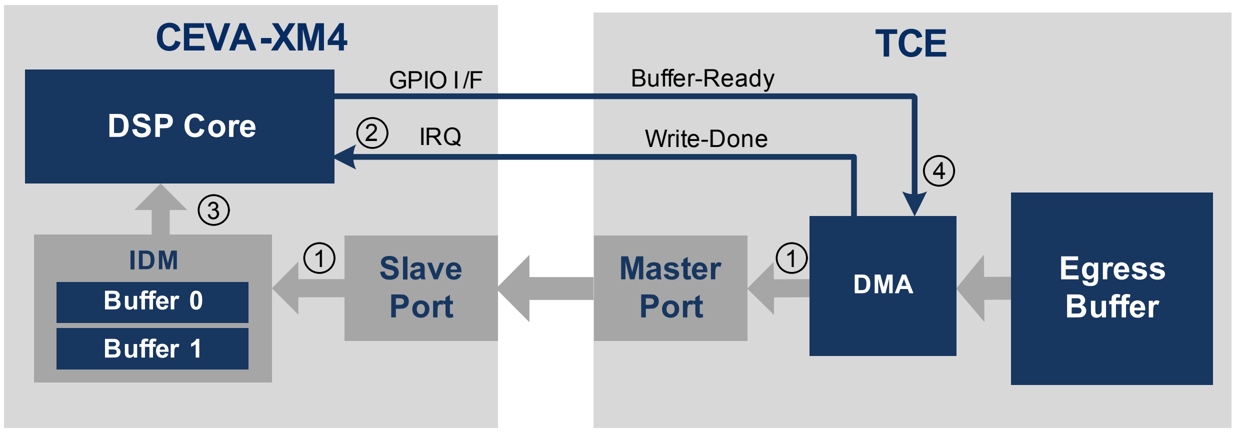

Figure 4-3. TCE Egress Master

The following steps form the packet data flow, where the numbering in the figure corresponds to the steps below:

1. TCE writes data to bufferX.
2. TCE signals write-done to DSP core using interrupt.
3. DSP core reads new packet from bufferX.
4. DSP core signals buffer-ready to TCE (enabling it to write to the next buffer).

4.2.2 TCE Egress Slave

Figure 4-4 illustrates an example of a system with CEVA-XM4 reading packet data from a double buffer at a TCE.

The packet data flow use the Packet-Ready flow control described at section 2.4.1.

The read done indication is omitted from the flow. The read done indication can be generated automatically by the TCE by counting the number of read bytes.

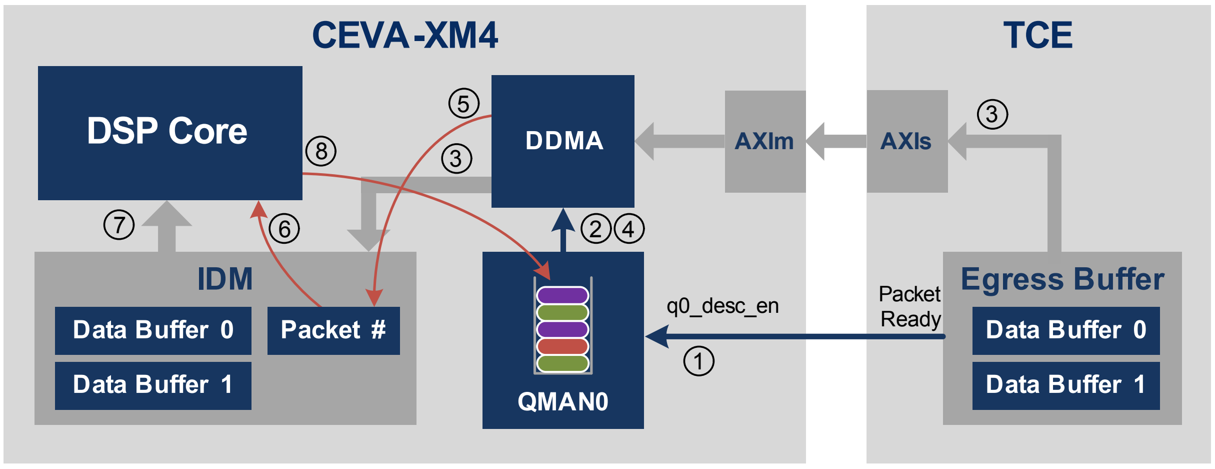

Figure 4-4. TCE Egress Slave

The following steps form the packet data flow, where the numbering in the figure corresponds to the steps below:

1. TCE writes output packet data to one of its egress buffers and enables next two tasks at QMAN0.
2. QMAN0 loads packet transfer task to DDMA.
3. DDMA moves the packet from TCE to the double buffer at CEVA-XM4 IDM.
4. QMAN loads write done message to DDMA.
5. DDMA write packet number.
6. DSP core checks for new input packets (based on DSP Input Double Buffer at IDM, refer to section 3.1.23.1).
7. DSP core reads new input packet from the IDM double buffer.
8. DSP core increments queue0 write pointer by 2.

This flow should be initialized with four tasks written at queue0 (write pointer is initialized to 4).