CEVA-XM4™

DDMA
User Guide

Rev. 1.0

June 2016

DSP Division

Documentation Control

History Table

Version	Date	Description	Remarks
1.0	8 June 2016	Initial version	

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

List of Examples

List of Tables

1. Introduction

1.1 Scope

This document describes how to use direct memory access (DDMA) to do algorithm optimization for the CEVA-XM4™ core.

The CEVA-XM4 core is designed to deal with highly demanding image-processing applications. It includes a strong vector-processing unit that delivers outstanding performance. The CEVA-XM4 uses a very efficient Data DMA (DDMA) to handle data transfer from internal to external, external to internal, and internal to internal.

1.2 Audience

This document is intended for designers who want to optimize algorithms by using the PingPang buffer with the DDMA.

1.3 Related Documents

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

1. *CEVA-XM4_Arch_Spec_Vol-I*
2. *CEVA-XM4_Arch_Spec_Vol-III_MSS*
3. *CEVA-XM4_DMA_Driver_doc.html*

1.4 Terminology

Table 1-1 defines the terms that are used throughout this document.

Table 1-1: Terminology

Term	Definition
DDMA Queue	The DDMA can accept up to three pending requests using a dedicated queue. This enables the user to program the DDMA in advance.
DDMA Descriptor	Each DDMA transaction is configured using a DDMA task descriptor. Task descriptors are configured for the DDMA either using the CPM or by the QMAN.
DDMA Sync Point	The sync point is used to ensure that the DMA queue task is done.

1.5 DMA Functions

The following are the most commonly used DDMA functions:

- **dma_create_2d_desc()**: Creates a new DDMA descriptor
- **dma_dma_enqueue_desc()**: Enqueues a new DDMA descriptor
- **dma_enqueue_sync_point()**: Enqueues a new DDMA sync point, and then returns it to the user
- **dma_wait_sync_point()**: Waits for a specific DDMA sync point
- **dma_update_2d_desc_size()**: Updates a pre-existing DDMA descriptor with a different size for loading (instead of creating a new, different descriptor)

2. Basic Application

2.1 Initialization

Use the following functions to initialize the DDMA manager and queue:

- **dma_init_manager();**
- **dma_allocate_queue(& g_queue_base, (unsigned int *)g_queue , 4, sizeof(g_queue) / sizeof (dma_desc_t));**

2.2 Workflow

Figure 2-1 shows the workflow of a basic application with the DDMA.

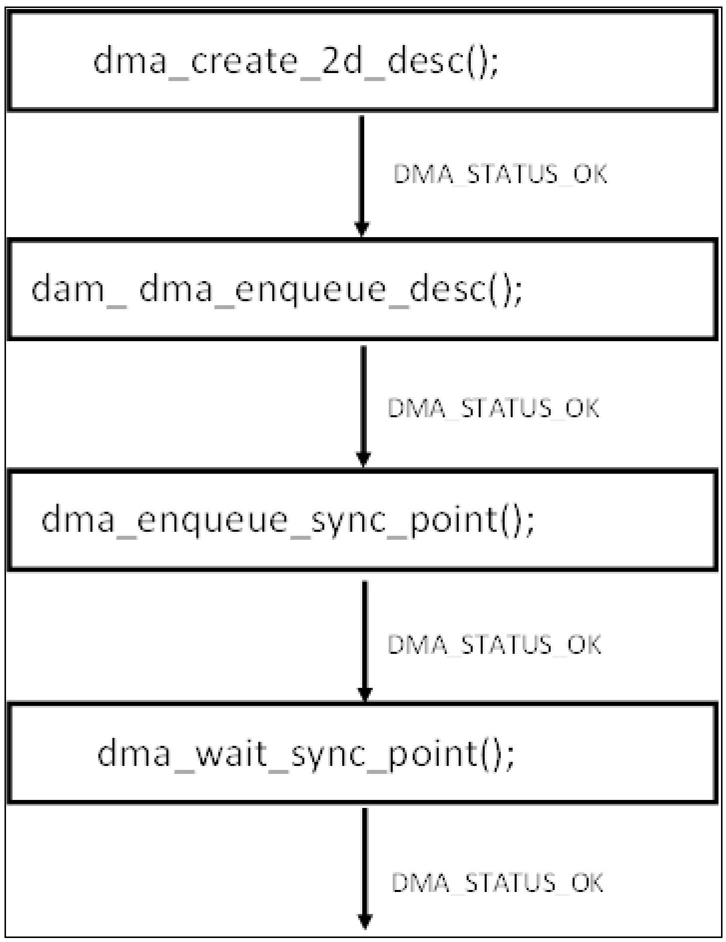

Figure 2-1: Basic Application Workflow

2.3 Code Example

Example 2-1 demonstrates a basic application code of a DDMA workflow.

Example 2-1: Basic Application Code

```
void transfer_2d()
{
    unsigned int transfer_width = 49;
    unsigned int transfer_height = 20;
    unsigned int src_stride = 49;
    unsigned int dst_stride = 64;
    unsigned int src_offset = 1039;
    unsigned int dst_offset = 2043;

    unsigned int sync_point_val;

    printf("Starting transfer_2d Test:\n");
    //Create one dimension task descriptor
    if (DMA_STATUS_OK != dma_create_2d_desc(&desc_2d,
transfer_width, transfer_height, src_stride, dst_stride,
DMA_DIR_EXTERNAL_INTERNAL, DMA_TYPE_LINEAR))
    {
        printf("Error creating 2d descriptor - Test failed!\n");
    }
    else
    {
        //enqueue task
        if (DMA_STATUS_OK != dma_enqueue_desc(&myqueue_base,
&desc_2d, ext_buff + src_offset, int_buff+dst_offset))
        {
            printf("Error enqueuing 2d descriptor - Test
failed!\n");
        }
        else
        {
```

```
            //Set sync point
            if (DMA_STATUS_OK !=
dma_enqueue_sync_point(&myqueue_base, &sync_point_val))
            {
                printf("Error enqueuing sync point - Test
failed!\n");
            }
            else
            {
                //Poll on sync point
                dma_wait_sync_point(&myqueue_base,
sync_point_val);
                printf("Test Passed\n");
            }
        }
    }
}
```

2.4 Image Process

Figure 2-2 shows the DDMA algorithm optimization.

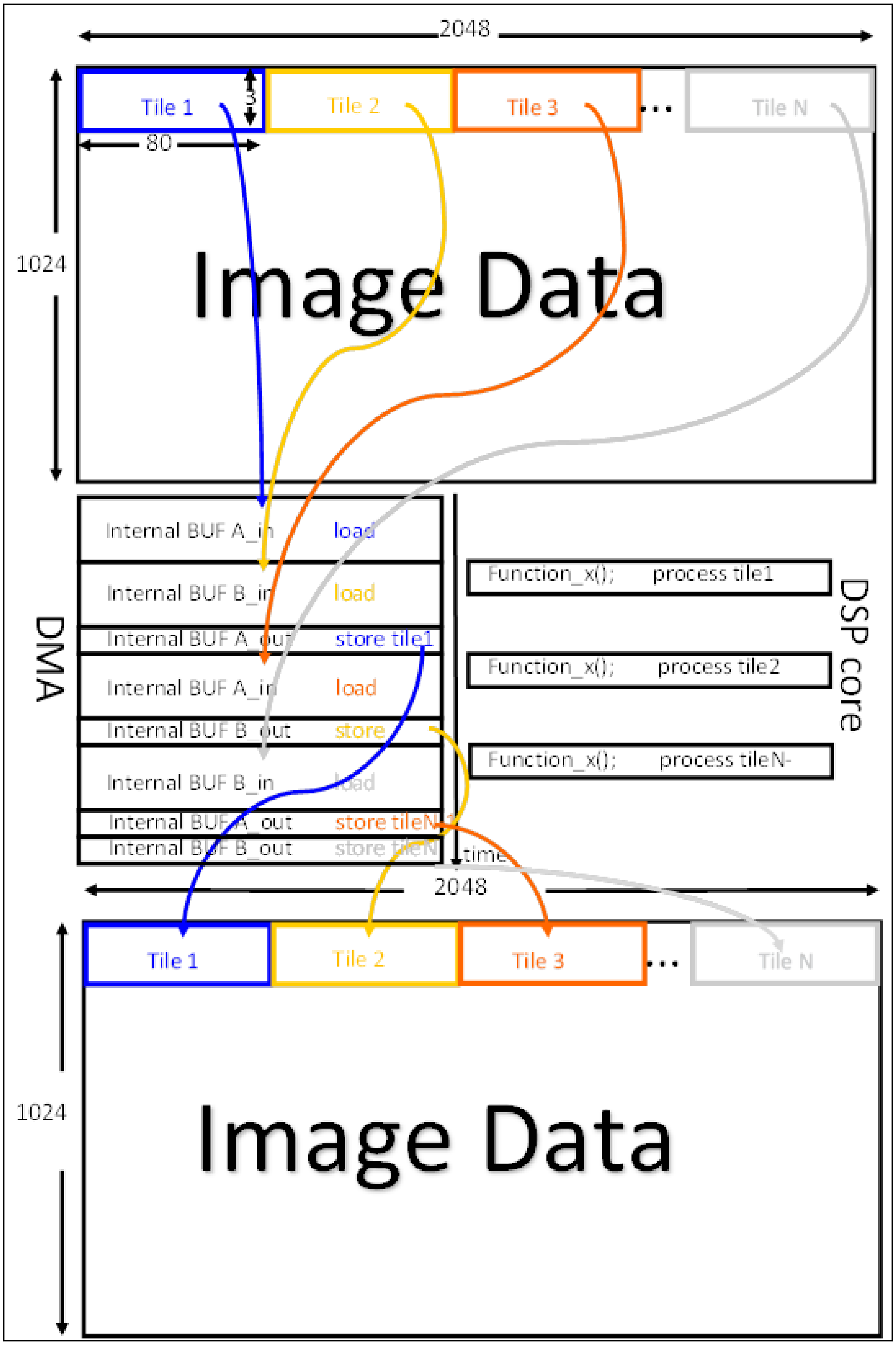

Figure 2-2: DDMA Algorithm Optimization

Example 2-2 demonstrates the code for a PingPang buffer with the DDMA.

Example 2-2: Image Process Code

```
void Img_process_with_DMA_demo()
{
    unsigned char ext_buf[EXT_BUF_SIZE];
    unsigned char buf_A_in[BUF_SIZE];
    unsigned char buf_A_out[BUF_SIZE];
    unsigned char buf_B_in[BUF_SIZE];
    unsigned char buf_B_out[BUF_SIZE];
    unsigned char*p_buf_src[2] = { buf_A_in, buf_B_in };
    unsigned char*p_buf_dst[2] = { buf_A_out, buf_B_out };



    //Create task descriptor
    dma_create_desc(&desc_2d, transfer_width, transfer_height,
src_stride, dst_stride, DMA_DIR_EXTERNAL_INTERNAL,
DMA_TYPE_LINEAR))


        for (i = 0; i < row_loop; i += row_step)
        {
            for (j = 0; j < line_loop; j += line_step)
            {
                //DMA process
                dma_enqueue_desc(&myqueue_base, &desc_2d,
p_buf_src[], p_buf_dst[phase ^ 1]);
                dma_enqueue_sync_point(&myqueue_base,
&sync_point_val[phase^ 1]););
                //DSP core process
                dma_wait_sync_point(&myqueue_base,
sync_point_val[phase]);
                process(buffer[[phase])]);
                phase ^= 1;
            }
        }
}
```