EVE Non–BAM Applets User Guide

Last Updated: July, 2014

Texas Instruments Inc.

Contents

Chapter 1

Introduction

Non-BAM EVELIB library is a collection of frame-level processing APIs leveraging the optimized signal and image processing and vision kernels from IMGSIGLIB, KERNELSLIB and VLIB on EVE (VCOP). These APIs on ARP32 abstract out the details of EDMA programming and VCOP and DMA pipelining aspects for block-based processing that is required to process an entire frame on EVE. The user need to invoke just a single function in order to process a single input frame.

1.1 Purpose and Scope

The purpose of this document is to detail the various non-BAM EVELIB functions. This document provides details of the non-BAM EVELIB function interfaces, their usage, recomended memory map to be employed and the expected performance of the APIs.

1.2 Abbreviations

The following abbreviations are used in this document.

Abbreviation	Description
EVE	Embedded Vision/Vector Engine
VCOP	Vector Co-Processor
VLIB	Vision Library
IMGSIGLIB	Image & Signal Processing Library
ARP32	32-bit Application Specific RISC Processor
DMEM	Data Memory in EVE Subsystem
WBUF	Work Buffer in EVE Subsystem
IBUF	Image Buffer in EVE Subsystem
EDMA	Enhanced Direct Memory Access
BAM	Block-based Acceleration Manager

1.3 Library and Source Paths

The Non-BAM EVELIB functions are packaged into the library *apps_nonbam/lib/libeveapps.eve.lib*. The interface for these functions are present at *apps_nonbam/inc*. The source code for these functions are present at *apps_nonbam/src*. The test code for each of these non-BAM EVELIB functions are present at *apps_nonbam/test*

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

File Documentation

3.1 evelib_edma_frame_padding.h File Reference

Border Padding Applet using EDMA.

```
#include <stdint.h>
```

Functions

- void EVELIB_padVertEDMA (uint8_t *padSourcePtr, uint8_t *padDestPtr, uint32_t padWidth, uint32_t padHeight, int32_t srcDataStride, int32_t dstDataStride, int32_t padPixelSize)

  Used for padding the top and bottom borders of the frame.
- void EVELIB_padHorzEDMA (uint8_t *padSourcePtr, uint8_t *padDestPtr, uint32_t padWidth, uint32_t padHeight, int32_t srcDataStride, int32_t dstDataStride, int32_t padPixelSize)

  Used for padding the left and right borders of the frame.

3.1.1 Detailed Description

Border Padding Applet using EDMA. This header defines all types, constants, and functions shared by all implementations of the EDMA Padding interface.

3.1.2 Function Documentation

3.1.2.1 void EVELIB_padHorzEDMA (uint8_t * *padSourcePtr*, uint8_t * *padDestPtr*, uint32_t *padWidth*, uint32_t *padHeight*, int32_t *srcDataStride*, int32_t *dstDataStride*, int32_t *padPixelSize*)

Used for padding the left and right borders of the frame.

Parameters

in	*padSourcePtr*	Starting address of the padding data
in	*padDestPtr*	Starting address of the location to be padded.
in	*padWidth*	Padding width (in bytes).
in	*padHeight*	Padding Height.
in	*srcDataStride*	Stride of the source buffer.
in	*dstDataStride*	Stride of the destination buffer.
in	*padPixelSize*	Size of the data to be padded (in lines). A total of padPixelSize*padHeight lines will be copied.

```
EVELIB_padHorzEDMA(uint8_t   *padSourcePtr,
                   uint8_t   *padDestPtr,
                   uint32_t   padWidth,
                   uint32_t   padHeight,
                   int32_t    srcDataStride,
                   int32_t    dstDataStride,
                   int32_t    padPixelSize)
```

Returns

Void

Remarks

This functions expects dmaStateStruct and edmaCc are initialized before the call.

3.1.2.2 void EVELIB_padVertEDMA (uint8_t * *padSourcePtr,* uint8_t * *padDestPtr,* uint32_t *padWidth,* uint32_t *padHeight,* int32_t *srcDataStride,* int32_t *dstDataStride,* int32_t *padPixelSize*)

Used for padding the top and bottom borders of the frame.

Parameters

in	*padSourcePtr*	Starting address of the padding data
in	*padDestPtr*	Starting address of the location to be padded.
in	*padWidth*	Padding width (in bytes).
in	*padHeight*	Padding Height.
in	*srcDataStride*	Stride of the source buffer.
in	*dstDataStride*	Stride of the destination buffer.
in	*padPixelSize*	Size of the data to be padded (in lines). A total of padPixelSize∗padHeight lines will be copied.

```
EVELIB_padVertEDMA(uint8_t   *padSourcePtr,
                   uint8_t   *padDestPtr,
                   uint32_t   padWidth,
                   uint32_t   padHeight,
                   int32_t    srcDataStride,
                   int32_t    dstDataStride,
                   int32_t    padPixelSize)
```

Returns

Void

Remarks

This functions expects dmaStateStruct and edmaCc are initialized before the call.

3.2 evelib_fir_filter_2d.h File Reference

2D FIR filtering Applet

Functions

- void EVELIB_firFilter2D (unsigned char ∗src, unsigned int srcImageWidth, unsigned int srcImageHeight, int srcBufferPitch, int srcBufferHeight, unsigned int srcBytesPP, unsigned char ∗dst, unsigned int dstImageWidth, unsigned int dstImageHeight, int dstBufferPitch, int dstBufferHeight, unsigned int dstBytesPP, char coeff[], unsigned int coeffH, unsigned int coeffW, unsigned int dnsmplVert, unsigned int dnsmplHorz, int rndShift)

  Apply 2D FIR filter on a given grayscale image.

3.2.1 Detailed Description

2D FIR filtering Applet

Date

March 2013

3.2.2 Function Documentation

3.2.2.1 void EVELIB_firFilter2D (unsigned char * *src*, unsigned int *srcImageWidth*, unsigned int *srcImageHeight*, int *srcBufferPitch*, int *srcBufferHeight*, unsigned int *srcBytesPP*, unsigned char * *dst*, unsigned int *dstImageWidth*, unsigned int *dstImageHeight*, int *dstBufferPitch*, int *dstBufferHeight*, unsigned int *dstBytesPP*, char *coeff[]*, unsigned int *coeffH*, unsigned int *coeffW*, unsigned int *dnsmplVert*, unsigned int *dnsmplHorz*, int *rndShift*)

Apply 2D FIR filter on a given grayscale image.

Parameters

in	*src*	source image pointer
in	*srcImageWidth*	source image width in pixels
in	*srcImageHeight*	source image height in pixels
in	*srcBufferPitch*	source image picth in bytes
in	*srcBufferHeight*	source image picth in bytes
in	*srcBytesPP*	source bytes per pixel (Must be 1 for grayscale. Only 1 is supported.)
out	*dst*	destimation image pointer
in	*dstImageWidth*	destination image width in pixels
in	*dstImageHeight*	destination image height in pixels
in	*dstBufferPitch*	destination image picth in bytes
in	*dstBufferHeight*	destination image picth in bytes
in	*dstBytesPP*	dest bytes per pixel (Must be 1 for grayscale. Only 1 is supported.)
in	*coeff*	filter coefficitents array pointer
in	*coeffH*	vertical filter taps (=M in an MxN filter)
in	*coeffW*	horizontal filter taps (=N in an MxN filter)
in	*dnsmplVert*	vertical downsample factor
in	*dnsmplHorz*	horizontal downsample factor (only 1 is supported currently)
in	*rndShift*	Shift factor for the given set of coefficients.

Remarks

Grayscale filtering function - 2D FIR filter.
The performance of this algorithm is best when either the srcImageWidth∗srcBytesPP or srcImagePitch is a multiple of 64 or 32.

See Also

EVELIB_algoDMAAutoIncrInit(), EVELIB_algoDMAAutoIncrProcess(), EVELIB_algoDMAAutoIncrDeInit()

3.3 evelib_fir_filter_2d_scatter_gather.h File Reference

2D FIR filtering with scatter gather EDMA

Functions

- void EVELIB_firFilter2D_scatterGather (unsigned char ∗src, unsigned int srcImageWidth, unsigned int srcImageHeight, int srcImagePitch, unsigned char ∗dst, unsigned int dstImageWidth, unsigned int dstImage-

Height, int dstImagePitch, char coeff[], unsigned int coeffH, unsigned int coeffW, unsigned int dnsmplVert, unsigned int dnsmplHorz, int rndShift)

Apply 2D FIR filter on a given grayscale image.

3.3.1 Detailed Description

2D FIR filtering with scatter gather EDMA

3.3.2 Function Documentation

3.3.2.1 void EVELIB_firFilter2D_scatterGather (unsigned char * *src*, unsigned int *srcImageWidth*, unsigned int *srcImageHeight*, int *srcImagePitch*, unsigned char * *dst*, unsigned int *dstImageWidth*, unsigned int *dstImageHeight*, int *dstImagePitch*, char *coeff[]*, unsigned int *coeffH*, unsigned int *coeffW*, unsigned int *dnsmplVert*, unsigned int *dnsmplHorz*, int *rndShift*)

Apply 2D FIR filter on a given grayscale image.

Parameters

in	*src*	source image pointer
in	*srcImageWidth*	source image width in pixels
in	*srcImageHeight*	source image height in pixels
in	*srcImagePitch*	source image picth in bytes
out	*dst*	destination image pointer
in	*dstImageWidth*	destination image width in pixels
in	*dstImageHeight*	destination image height in pixels
in	*dstImagePitch*	destination image picth in bytes
in	*coeff*	filter coefficitents array pointer
in	*coeffH*	horizontal filter taps (=M in an MxN filter)
in	*coeffW*	horizontal filter taps (=N in an MxN filter)
in	*dnsmplVert*	vertical downsample factor
in	*dnsmplHorz*	horizontal downsample factor (only 1 is supported currently)
in	*rndShift*	Shift factor for the given set of coefficients.

Remarks

Grayscale filtering function - 2D FIR filter. This uses scatter gather EDMA internally. This is meant as a test for scatter gather EDMA APIs.
The performance of this algorithm is best when either the srcImageWidth*srcBytesPP or srcImagePitch is a multiple of 64 or 32.

See Also

EVELIB_algoDMAScatterGatherInit(), EVELIB_algoDMAScatterGatherProcess(), EVELIB_algoDMAScatterGatherDeInit()

3.4 evelib_grayscale_morphology.h File Reference

Grayscale Morphological Filtering Applet.

Enumerations

- enum GrayscaleMorphologyOperation { GRAYSCALE_MORPH_DILATE = 0, GRAYSCALE_MORPH_ERODE, GRAYSCALE_MORPH_OPEN, GR-

AYSCALE_MORPH_CLOSE,
GRAYSCALE_MORPH_TOPHAT, GRAYSCALE_MORPH_BOTHAT, GRAYSCALE_MORPH_GRADIENT
}

Grayscale Morphological Operation type.

Functions

- void EVELIB_grayscaleMorphology (unsigned char *src, unsigned int srcImageWidth, unsigned int srcImageHeight, int srcImagePitch, unsigned char *dst, unsigned int dstImageWidth, unsigned int dstImageHeight, int dstImagePitch, unsigned char struct_elem[], unsigned char struct_elem_refl[], unsigned int se_height, unsigned int se_width, GrayscaleMorphologyOperation operation)

  Frame-level Grayscale Morphological Filtering API.

3.4.1 Detailed Description

Grayscale Morphological Filtering Applet.

Date

22 March 2013

This applet supports grayscale morphological filtering using a generic flat structuring element. The structuring element need to be specified as a mask of ones and zeros in an array of length se_width x se_height in 'struct_elem'. The user is also expected to provide the reflected structuring element array required for dilation in 'struct_elem_refl'. The input grayscale images are assumed to be 8-bit. The output is also an 8-bit grayscale image.

The supported grayscale morphological operations include: dilation, erosion, opening, closing, top hat, bottom hat and morphological gradient. Internally the applet processes the input image in blocks of 32x32. Hence the srcImageWidth and srcImageHeight needs to be multiples of 32 inorder to process the entire image.

3.4.2 Enumeration Type Documentation

3.4.2.1 enum GrayscaleMorphologyOperation

Grayscale Morphological Operation type.

This enum type lists the set of grayscale morphological operations that can be performed. The set of supported operations include grayscale dilation, erosion, opening, closing, top hat, bottom hat and morphological gradient.

Enumerator

GRAYSCALE_MORPH_DILATE Grayscale Dilation

GRAYSCALE_MORPH_ERODE Grayscale Erosion

GRAYSCALE_MORPH_OPEN Grayscale Opening

GRAYSCALE_MORPH_CLOSE Grayscale Closing

GRAYSCALE_MORPH_TOPHAT Grayscale Top Hat Operation

GRAYSCALE_MORPH_BOTHAT Grayscale Bottom Hat Operation

GRAYSCALE_MORPH_GRADIENT Grayscale Morphological Gradient

3.4.3 Function Documentation

3.4.3.1 void EVELIB_grayscaleMorphology (unsigned char * *src*, unsigned int *srcImageWidth*, unsigned int *srcImageHeight*, int *srcImagePitch*, unsigned char * *dst*, unsigned int *dstImageWidth*, unsigned int *dstImageHeight*, int *dstImagePitch*, unsigned char *struct_elem[]*, unsigned char *struct_elem_refl[]*, unsigned int *se_height*, unsigned int *se_width*, GrayscaleMorphologyOperation *operation*)

Frame-level Grayscale Morphological Filtering API.

EVELIB_grayscaleMorphology performs the specified grayscale morphological operation on the input 8-bit grayscale image provided in src buffer with the structuring element provided in struct_elem and writes the output into dst buffer.

```
void EVELIB_grayscaleMorphology(unsigned char  *src,
                                unsigned int    srcImageWidth,
                                unsigned int    srcImageHeight,
                                int             srcImagePitch,
                                unsigned char  *dst,
                                unsigned int    dstImageWidth,
                                unsigned int    dstImageHeight,
                                int             dstImagePitch,
                                unsigned char   struct_elem[],
                                unsigned char   struct_elem_refl[],
                                unsigned int    se_height,
                                unsigned int    se_width,
                                GrayscaleMorphologyOperation  operation)
```

Parameters

in	*src*	Pointer to the source buffer in SDRAM.
in	*srcImageWidth*	Input image width. Need to be a multiple of 32.
in	*srcImageHeight*	Input image height. Need to be a multiple of 32.
in	*srcImagePitch*	Stride of the input buffer.
out	*dst*	Pointer to the destination buffer in SDRAM.
in	*dstImageWidth*	Output image width.
in	*dstImageHeight*	Output image height.
in	*dstImagePitch*	Stride of the output buffer.
in	*struct_elem*	Pointer to structuring element array.
in	*struct_elem_refl*	Pointer to reflected structuring element array.
in	*se_height*	Structuring element height.
in	*se_width*	Structuring element width.
in	*operation*	Grayscale morphological operation to be performed.

3.5 evelib_harris_corner_detection.h File Reference

Harris Corner Detection Applet.

Functions

- void EVELIB_harrisCornerDetection (unsigned char *src, unsigned int srcImageWidth, unsigned int srcImageHeight, int srcBufferPitch, int srcBufferHeight, unsigned int srcBytesPP, unsigned char *dst, unsigned int dstImageWidth, unsigned int dstImageHeight, int dstBufferPitch, int dstBufferHeight, unsigned int srcBlkWidth, unsigned int srcBlkHeight, unsigned int dstBytesPP, short harrisScoreScalingFactor, short nmsThresh)

  Compute corners for a given grayscale image using Shi & Tomasi corner detector which is based on Harris Corner Detection algorithm.

3.5.1 Detailed Description

Harris Corner Detection Applet.

Date

April 2013

3.5.2 Function Documentation

3.5.2.1 void EVELIB_harrisCornerDetection (unsigned char * *src*, unsigned int *srcImageWidth*, unsigned int *srcImageHeight*, int *srcBufferPitch*, int *srcBufferHeight*, unsigned int *srcBytesPP*, unsigned char * *dst*, unsigned int *dstImageWidth*, unsigned int *dstImageHeight*, int *dstBufferPitch*, int *dstBufferHeight*, unsigned int *srcBlkWidth*, unsigned int *srcBlkHeight*, unsigned int *dstBytesPP*, short *harrisScoreScalingFactor*, short *nmsThresh*)

Compute corners for a given grayscale image using Shi & Tomasi corner detector which is based on Harris Corner Detection algorithm.

Parameters

in	*src*	source image pointer
in	*srcImageWidth*	source image width in pixels
in	*srcImageHeight*	source image height in pixels
in	*srcBufferPitch*	source buffer pitch in bytes
in	*srcBufferHeight*	source buffer height in bytes
in	*srcBytesPP*	source bytes per pixel (Must be 1 for grayscale. Only 1 is supported.)
out	*dst*	destimation image pointer
in	*dstImageWidth*	destination image width in pixels
in	*dstImageHeight*	destination image height in pixels
in	*dstBufferPitch*	destination buffer pitch in bytes
in	*dstBufferHeight*	destination buffer height in bytes
in	*srcBlkWidth*	block width of the source
in	*srcBlkHeight*	block height of the source
in	*dstBytesPP*	dest bytes per pixel (Must be 1 for grayscale. Only 1 is supported.)
in	*harrisScore-ScalingFactor*	scaling factor used by harris score kernel
in	*nmsThresh*	threshold parameter for non maximum suppression kernel

Remarks

Harris corner detector function.
Currently, the algorithm works fine when srcImageWidth∗srcBytesPP or srcImagePitch is a multiple of 16.

See Also

EVELIB_algoDMAAutoIncrInit(), EVELIB_algoDMAAutoIncrProcess(), EVELIB_algoDMAAutoIncrDeInit()

3.6 evelib_memcopy_dma_2d.h File Reference

EDMA 2D Memcopy Applet.

```
#include <stdint.h>
```

Functions

- void EVELIB_memcopyDMA2D (uint8_t ∗sourcePtr, uint8_t ∗destPtr, uint32_t widthBytes, uint32_t height, int32_t srcStride, int32_t dstStride)

  Used for Performing 2D copy using DMA.

3.6.1 Detailed Description

EDMA 2D Memcopy Applet. This header defines all types, constants, and functions shared by all implementations of the EDMA 2D Copy interface.

3.6.2 Function Documentation

3.6.2.1 void EVELIB_memcopyDMA2D (uint8_t * *sourcePtr,* uint8_t * *destPtr,* uint32_t *widthBytes,* uint32_t *height,* int32_t *srcStride,* int32_t *dstStride*)

Used for Performing 2D copy using DMA.

Parameters

in	*sourcePtr*	Starting address of the padding data
in	*destPtr*	Starting address of the location to be padded.
in	*widthBytes*	Width of the buffer (in bytes).
in	*height*	Height of the buffer.
in	*srcStride*	Stride of the source buffer.
in	*dstStride*	Stride of the destination buffer.

```
void EVELIB_memcopyDMA2D(uint8_t     *sourcePtr,
                         uint8_t     *destPtr,
                         uint32_t     widthBytes,
                         uint32_t     height,
                         int32_t      srcStride,
                         int32_t      dstStride)
```

Returns

Void

Index