ACF/APEX Kernel SDK

Generated by Doxygen 1.8.8

Fri Dec 14 2018 18:20:12

Contents

Chapter 1

Main Page

The herein provided kernel SDK is meant as a starting point for the implementation of computing kernels for the S32V234 APEX device. It has the purpose of being an inspiration source for the development of own kernels, since it touches on the different programming aspects of the APEX device.

Organization of the SDK:

The SDK has several component libraries:

- **Arithmetic kernels** : Provide basic operators for element-wise addition, subtraction, multiplication, division and arithmetic shifting
- **Comparison kernels** : Provide basic element-wise comparison operators like less than, less-than-or-equal, binary AND operator and binary descriptor matches
- **Conversion kernels** : Conversion kernels from 16 to 8 bit and from RGB format to grayscale
- **Display kernels** : Gives examples of marking an image at certain points as overlay or in a certain color channel.
- **Feature detection** : Provides two corner detection algorithms FAST9 and Harris corner detection
- **Filtering** : Offers kernels for general purpose filtering, and also the most used filters like Gaussian filtering, gradient computation, non-maximum suppression and saturation
- **Geometry** : Provides geometric transformations, like rotations and bilinear interpolation and also a replacement for indirect inputs, called offset selection
- **Morphology** : Example of a morphological dilation operator.
- **Object detection** : Two object detection algorithms : Haar cascade and LBP (local binary pattern) cascade
- **Optical Flow** : Kernels needed for the Optical flow algorithm
- **Optimization** : Implementation of the Integral Image (SAT) kernel and a SAT-based box filter.
- **Resizing** : Provides downsampling and upsampling kernels (gives examples of size changes inside a filter)
- **Statistics** : Provides kernels for statistics computations, such as a Histogram kernel, a vector-to-scalar reduction kernel and an accumulation kernel.

The library's directory has following subcomponents:

- <category>_kernels/src : the directory containing the kernel source and header files of each library
- **BUILD.mk** : this file will be included by Makefiles which are used for target compilations, containing a list of all sources from the <category>_kernels/src subdirectory which shall be included into the library and all the necessary dependencies of these sources.

- <category>_kernels/build-deskwin32/mvc : Contains Visual Studio projects which can be included into higher level application Visual Studio projects .
- <category>_kernels/build-<platform>-<compiler>-<OS>-d/ : Makefile builds for ACF/APEX Will contain build results for the corresponding platform.

Inside the <category>_kernels/src directory the files obey a certain naming convention which is as follows:

- **apu_<KernelCategory>_impl.{c,h}** : contain the kernel code which shall be executed on the APEX
- **apu_<KernelCategory>.{c,h}** : contain the ACF-kernel wrappers which provide the interface for the ACF-Graphs

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Module Documentation

5.1 UserAPI

5.1.1 Detailed Description

This is the group of enum, structure and functions needs to be exposed to APEX MATH library user

5.2 Apexcv_pro

5.3 Fast

5.3.1 Detailed Description

Histogram.

Functions

- KERNEL_INFO **fast_offset** (" fast_offset ", 3, __port(__index(0), __identifier("OUTPUT_OFFSETS"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(1), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("CIRCUMFERENCE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO **fast** (" fast ", 4, __port(__index(0), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("THRESHOLD"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OFFSET_TABLE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(16, 1)))
- KERNEL_INFO **fast_serialized** (" fast_serialized ", 6, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("THRESHOLD"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OFFSET_TABLE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(3), __identifier("OUT_PACKED"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(8192, 1)), __port(__index(4), __identifier("COUNTER"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("OUT_MAX_SIZE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO **fast_nms** (" fast_nms ", 4, __port(__index(0), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("THRESHOLD"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OFFSET_TABLE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(16, 1)))
- KERNEL_INFO nms3x3 (" nms3x3 ", 2, __port(__index(0), __identifier("input"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("output"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

5.3.2 Function Documentation

5.3.2.1 KERNEL_INFO nms3x3 (" nms3x3 " , 2 , __port(__index(0), __identifier("input"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("output"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Non-maximum suppression 3x3 kernel metadata.

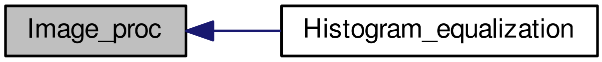
Parameters

2	Number of ports
Port NMS16_KN_IN	Define for name of input image (unsigned 8bit)
Port NMS16_K↩ N_OUT	Define for name of output image (unsigned 8bit)

5.4 Image_proc

5.4.1 Detailed Description

Collaboration diagram for Image_proc:



Modules

- Histogram_equalization

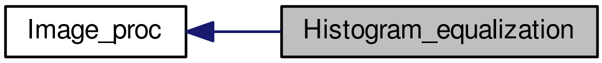
  Histogram.

5.5 Histogram_equalization

5.5.1 Detailed Description

Histogram.

Collaboration diagram for Histogram_equalization:



Functions

- KERNEL_INFO apu_histogram_equalization (" apu_histogram_equalization ", 4, __port(__index(0), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LUT_IN"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(256, 1)), __port(__index(3), __identifier("VECTORIZED_LUT_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(256, 1)))
- KERNEL_INFO apu_generate_lut (" apu_generate_lut ", 3, __port(__index(0), __identifier("LUT_OUT"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(256, 1)), __port(__index(1), __identifier("IMAGE_HISTOGRAM"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)), __port(__index(2), __identifier("NUM_PIXELS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

5.5.2 Function Documentation

5.5.2.1 KERNEL_INFO apu_generate_lut (" apu_generate_lut " , 3 , __port(__index(0), __identifier("LUT_OUT"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(256, 1)) , __port(__index(1), __identifier("IMAGE_HISTOGRAM"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)) , __port(__index(2), __identifier("NUM_PIXELS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Histogram_equalization kernel metadata.

Parameters

apu_equalized_lut	Kernel name
3	Number of ports
Port IMAGE_HISTOGRAM	input image histogram (unsigned 32bit)

Port LUT	histogram equalization look up table, CU count X 256 (unsigned 8bit)
Port NUM_PIXELS	number of pixels in input image

5.5.2.2 KERNEL_INFO apu_histogram_equalization (" apu_histogram_equalization " , 4 , __port(__index(0), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LUT_IN"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(256, 1)) , __port(__index(3), __identifier("VECTORIZED_LUT_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(256, 1)))

Histogram_equalization kernel metadata.

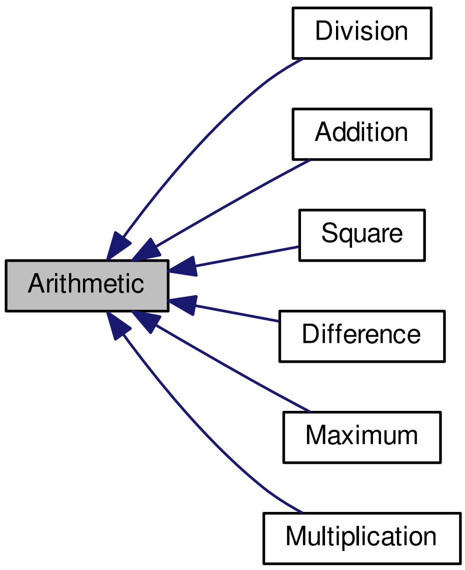
Parameters

apu_histogram↵_equalization	Kernel name
2	Number of ports
Port INPUT_0	input image (unsigned 8bit)
Port OUTPUT_0	histogram equalization vector output, CU count X 256 (unsigned 8bit)
Port LUT	histogram equalization look up table, CU count X 256 (unsigned 8bit)

5.6 Arithmetic

5.6.1 Detailed Description

Collaboration diagram for Arithmetic:



Modules

- Addition

  Element-wise addition.
- Difference

  Element-wise difference.
- Division

  Element-wise division.
- Multiplication

  Element-wise multiplication.
- Square

  Element-wise square.
- Maximum

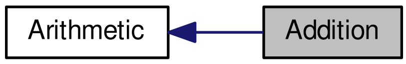
  Element-wise maximum.

5.7 Addition

5.7.1 Detailed Description

Element-wise addition.

Collaboration diagram for Addition:



Functions

- KERNEL_INFO apu_add (" apu_add ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_add_in16s_out32s (" apu_add_in16s_out32s ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_add_in32s_out32s (" apu_add_in32s_out32s ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_add_in32Q3_28_out32Q3_28 (" apu_add_in32Q3_28_out32Q3_28 ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUTPUT_Frac"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_add_in64s_out64s (" apu_add_in64s_out64s ", 6, __port(__index(0), __identifier("INPUT_0_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_0_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_1_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("INPUT_1_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("OUTPUT_0_HIGH"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("OUTPUT_0_LOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

- void add (vec16u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit addition => unsigned 16bit.
- void add_in16s_out32s (vec32s ∗dst, vec16s ∗srcImage0, vec16s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 16bit addition => signed 32bit.
- void add_in32s_out32s (vec32s ∗dst, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit addition => signed 32bit.
- void add_in32u_out32u (vec32u ∗dst, vec32u ∗srcImage0, vec32u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 32bit addition => unsigned 32bit.
- void add_in64s_out64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcImage0_high, vec32u ∗srcImage0_low, vec32s ∗srcImage1_high, vec32u ∗srcImage1_low, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 64bit addition => signed 64bit.
- void add_in64u_out64u (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗srcImage0_high, vec32u ∗srcImage0_low, vec32u ∗srcImage1_high, vec32u ∗srcImage1_low, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 64bit addition => unsigned 64bit.
- void add_in32Q3_28_out32Q3_28 (vec32s ∗dstInt, vec32s ∗dstFrac, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise 32bit Q3.28 addition => 64bit in [Q3.28] (integer part) and {Q3.28} (fractional part) format.
- KERNEL_INFO apu_gauss_3x1 (" apu_gauss_3x1 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void gauss_3x1 (vec08u ∗dst, vec08u ∗srcImage0, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit addition => unsigned 16bit.
- KERNEL_INFO apu_disparity (" apu_disparity ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 64, 1, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void disparity (vec08u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int cw, int ch, int inStrideW0, int inStrideW1, int outStrideW)

  Elementwise unsigned 8bit addition => unsigned 16bit.

5.7.2 Function Documentation

5.7.2.1 void add (vec16u ∗ *dst*, vec08u ∗ *srcImage0*, vec08u ∗ *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise unsigned 8bit addition => unsigned 16bit.

Addition btw two unsigned 8bit matrices with 16bit result: dst[i] = srcImage0[i] + srcImage1[i]

Parameters

dst	- [Output] 16bit destination block pointer
srcImage0	- [Input] 8bit source block pointer of img 0
srcImage1	- [Input] 8bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height

inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.7.2.2 void add_in16s_out32s (vec32s ∗ *dst,* vec16s ∗ *srcImage0,* vec16s ∗ *srcImage1,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise signed 16bit addition => signed 32bit.

Addition btw two signed 16bit matrices with 32bit result: dst[i] = srcImage0[i] + srcImage1[i]

Parameters

dst	- [Output] 32bit destination block pointer
srcImage0	- [Input] 16bit source block pointer of img 0
srcImage1	- [Input] 16bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.7.2.3 void add_in32Q3_28_out32Q3_28 (vec32s ∗ *dstInt,* vec32s ∗ *dstFrac,* vec32s ∗ *srcImage0,* vec32s ∗ *srcImage1,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise 32bit Q3.28 addition => 64bit in [Q3.28] (integer part) and {Q3.28} (fractional part) format.

Addition btw two 32bit fixed point numbers in Q3.28 format: dstInt[i] = int(srcImage0[i] + srcImage1[i]); dstFrac[i] = frac(srcImage0[i] + srcImage1[i]); (the number below zero, which is represented here as 0.n ∗ 2^28

Parameters

dstInt	- [Output] 32bit the integer part of the addition
dstFrac	- [Output] 32bit the fractional part of the addition in Q3.28 fixed point format
srcImage0	- [Input] 32bit source block pointer of vector 0 in Q3.28 fixed point format
srcImage1	- [Input] 32bit source block pointer of vector 1 in Q3.28 fixed point format
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.7.2.4 void add_in32s_out32s (vec32s ∗ *dst,* vec32s ∗ *srcImage0,* vec32s ∗ *srcImage1,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise signed 32bit addition => signed 32bit.

Addition btw two signed 32bit matrices: dst[i] = srcImage0[i] + srcImage1[i]

Warning

No out of range above 32bits is checked

Parameters

dst	- [Output] signed 32bit destination block pointer

srcImage0	- [Input] signed 32bit source block pointer of img 0
srcImage1	- [Input] signed 32bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.7.2.5 void add_in32u_out32u (vec32u ∗ *dst,* vec32u ∗ *srcImage0,* vec32u ∗ *srcImage1,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise unsigned 32bit addition => unsigned 32bit.

Addition btw two unsigned 32bit matrices: dst[i] = srcImage0[i] + srcImage1[i]

Warning

No out of range above 32bits is checked

Parameters

dst	- [Output] unsigned 32bit destination block pointer
srcImage0	- [Input] unsigned 32bit source block pointer of img 0
srcImage1	- [Input] unsigned 32bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.7.2.6 void add_in64s_out64s (vec32s ∗ *dst_high,* vec32u ∗ *dst_low,* vec32s ∗ *srcImage0_high,* vec32u ∗ *srcImage0_low,* vec32s ∗ *srcImage1_high,* vec32u ∗ *srcImage1_low,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise signed 64bit addition => signed 64bit.

Addition btw two signed 64bit matrices: dst[i] = srcImage0[i] + srcImage1[i]

Warning

No out of range above 64bits is checked

Parameters

dst_high	- [Output] High word 32bit destination block pointer
dst_low	- [Output] Low word 32bit destination block pointer
srcImage0_high	- [Input] High word 32bit source block pointer of img 0
srcImage0_low	- [Input] Low word 32bit source block pointer of img 0
srcImage1_high	- [Input] High word 32bit source block pointer of img 1
srcImage1_low	- [Input] Low word 32bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.7.2.7 void add_in64u_out64u (vec32u ∗ *dst_high,* vec32u ∗ *dst_low,* vec32u ∗ *srcImage0_high,* vec32u ∗ *srcImage0_low,* vec32u ∗ *srcImage1_high,* vec32u ∗ *srcImage1_low,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise unsigned 64bit addition => unsigned 64bit.

Addition btw two unsigned 64bit matrices: dst[i] = srcImage0[i] + srcImage1[i]

Warning

No out of range above 64bits is checked

Parameters

dst_high	- [Output] High word 32bit destination block pointer
dst_low	- [Output] Low word 32bit destination block pointer
srcImage0_high	- [Input] High word 32bit source block pointer of img 0
srcImage0_low	- [Input] Low word 32bit source block pointer of img 0
srcImage1_high	- [Input] High word 32bit source block pointer of img 1
srcImage1_low	- [Input] Low word 32bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.7.2.8 KERNEL_INFO apu_add (" apu_add " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

Addition kernel metadata. Adds pixelwise two unsigned 8bit images. Outputs 16bit unsigned addition result.

Parameters

ADD_KN	Define for Kernel name
3	Number of ports
Port ADD_KN_INA	Define for name of first input image (unsigned 8bit)
Port ADD_KN_INB	Define for name of second input image (unsigned 8bit)
Port ADD_KN_OUT	Define for name of addition result of the two images (unsigned 16bit).

5.7.2.9 KERNEL_INFO apu_add_in16s_out32s (" apu_add_in16s_out32s " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Addition kernel metadata. Adds pixelwise two signed 16bit images. Outputs signed 32bit addition result

Parameters

ADD_In16s_↩ Out32s_KN	Define for Kernel name
3	Number of ports
Port ADD_KN_INA	Define for name of first input image (signed 16bit)

Port ADD_KN_INB	Define for name of second input image (signed 16bit)
Port ADD_KN_OUT	Define for name of addition result (signed 32bit).

5.7.2.10 KERNEL_INFO apu_add_in32Q3_28_out32Q3_28 (" apu_add_in32Q3_28_out32Q3_28 " , 4 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("OUTPUT_Frac"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Addition kernel metadata. Adds pixelwise two images in fixed point format of the type Q3.28. Outputs signed 32bit the integer addition result (in normal integer format) and the fractional addition result in Q3.28 format

Warning

Does not check out of range values above/below +/-2^(31-1)

Parameters

ADD_32S_Q3←_28_KN	Define for Kernel name
4	Number of ports
Port ADD_KN_INA	Define for name of first input image (signed 32bit) in fixed point Q3.28 format
Port ADD_KN_INB	Define for name of second input image (signed 32bit) in fixed point Q3.28 format
Port ADD_KN_OUT	Define for name of 32bit signed integer part of addition result of the two images.
Port ADD_KN_←OUT_FRAC	Define for name of 32bit fractional part of addition result of the two images in fixed point Q3.28 format.

5.7.2.11 KERNEL_INFO apu_add_in32s_out32s (" apu_add_in32s_out32s " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Addition kernel metadata. Adds pixelwise two signed 32bit images. Outputs signed 32bit addition result

Warning

Does not check out of range values above/below +/-2^(31-1)

Parameters

ADD_In32s_←Out32s_KN	Define for Kernel name
3	Number of ports
Port ADD_KN_INA	Define for name of first input image (signed 32bit)
Port ADD_KN_INB	Define for name of second input image (signed 32bit)
Port ADD_KN_OUT	Define for name of addition result (signed 32bit).

5.7.2.12 KERNEL_INFO apu_add_in64s_out64s (" apu_add_in64s_out64s " , 6 , __port(__index(0), __identifier("INPUT_0_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_0_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("INPUT_1_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("INPUT_1_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("OUTPUT_0_HIGH"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(5), __identifier("OUTPUT_0_LOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Addition kernel metadata. Adds pixelwise two signed 64bit images. Outputs signed 64bit addition result

Warning

Does not check out of range values above 2^(63-1)

Parameters

ADD_In64s_Out64s_KN	Define for Kernel name
6	Number of ports
Port ADD_KN_INA_HIGH	Define for name of signed 32bit high word of first input image
Port ADD_KN_INA_LOW	Define for name of unsigned 32bit low word of firstinput image
Port ADD_KN_INB_HIGH	Define for name of signed 32bit high word of second input image
Port ADD_KN_INB_LOW	Define for name of unsigned 32bit low word of second input image
Port ADD_KN_OUT_HIGH	Define for name of signed 32bit high word of addition result
Port ADD_KN_OUT_LOW	Define for name of unsigned 32bit low word of addition result

5.7.2.13 KERNEL_INFO apu_disparity (" apu_disparity " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 64, 1, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Addition kernel metadata. Adds pixelwise two unsigned 8bit images. Outputs 16bit unsigned addition result.

Parameters

ADD_KN	Define for Kernel name
3	Number of ports
Port ADD_KN_INA	Define for name of first input image (unsigned 8bit)
Port ADD_KN_INB	Define for name of second input image (unsigned 8bit)
Port ADD_KN_OUT	Define for name of addition result of the two images (unsigned 16bit).

5.7.2.14 KERNEL_INFO apu_gauss_3x1 (" apu_gauss_3x1 " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Addition kernel metadata. Adds pixelwise two unsigned 8bit images. Outputs 16bit unsigned addition result.

Parameters

ADD_KN	Define for Kernel name
3	Number of ports
Port ADD_KN_INA	Define for name of first input image (unsigned 8bit)
Port ADD_KN_INB	Define for name of second input image (unsigned 8bit)
Port ADD_KN_OUT	Define for name of addition result of the two images (unsigned 16bit).

5.7.2.15 void disparity (vec08u ∗ *dst,* vec08u ∗ *srcImage0,* vec08u ∗ *srcImage1,* int *bw,* int *bh,* int *cw,* int *ch,* int *inStrideW0,* int *inStrideW1,* int *outStrideW*)

Elementwise unsigned 8bit addition => unsigned 16bit.

Addition btw two unsigned 8bit matrices with 16bit result: dst[i] = srcImage0[i] + srcImage1[i]

Parameters

dst	- [Output] 16bit destination block pointer
srcImage0	- [Input] 8bit source block pointer of img 0
srcImage1	- [Input] 8bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.7.2.16 void gauss_3x1 (vec08u ∗ *dst,* vec08u ∗ *srcImage0,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise unsigned 8bit addition => unsigned 16bit.

Addition btw two unsigned 8bit matrices with 16bit result: dst[i] = srcImage0[i] + srcImage1[i]

Parameters

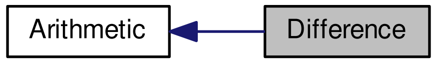
dst	- [Output] 16bit destination block pointer
srcImage0	- [Input] 8bit source block pointer of img 0
srcImage1	- [Input] 8bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.8 Difference

5.8.1 Detailed Description

Element-wise difference.

Collaboration diagram for Difference:



Functions

- KERNEL_INFO apu_diff_in08u_out16s (" apu_diff_in08u_out16s ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_diff_in16s_out16s (" apu_diff_in16s_out16s ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_diff_in16s_out32s (" apu_diff_in16s_out32s ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_diff_in32s_out32s (" apu_diff_in32s_out32s ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_diff_in64s_out64s (" apu_diff_in64s_out64s ", 6, __port(__index(0), __identifier("INPUT_A_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_A_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_B_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("INPUT_B_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("OUTPUT_HIGH"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("OUTPUT_LOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- void difference_filter_in08u_out16s (vec16s ∗dst, vec08u ∗srcA, vec08u ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

Pixel-wise difference between two unsigned 8bit images => signed 16bit.

- void difference_filter_in16s_out16s (vec16s ∗dst, vec16s ∗srcA, vec16s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Pixel-wise difference between two signed 16bit images => signed 16bit.

- void difference_filter_in16s_out32s (vec32s ∗dst, vec16s ∗srcA, vec16s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Pixel-wise difference between two signed 16bit images => signed 32bit.

- void difference_filter_in32s_out32s (vec32s ∗dst, vec32s ∗srcA, vec32s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Pixel-wise difference between two signed 16bit images => signed 32bit.

- void difference_filter_in32u_out32s (vec32s ∗dst, vec32u ∗srcA, vec32u ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Pixel-wise difference between two unsigned 32bit images => signed 32bit.

- void difference_filter_in32s_out64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA, vec32s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Pixel-wise difference between two signed 32bit images => signed 64bit.

- void difference_filter_in64s_out64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA_high, vec32u ∗srcA_low, vec32s ∗srcB_high, vec32u ∗srcB_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Pixel-wise difference between two signed 64bit images => signed 64bit.

- void difference_filter_in64u_out64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32u ∗srcA_high, vec32u ∗srcA_low, vec32u ∗srcB_high, vec32u ∗srcB_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Pixel-wise difference between two unsigned 64bit images => signed 64bit.

5.8.2 Function Documentation

5.8.2.1 KERNEL_INFO apu_diff_in08u_out16s (" apu_diff_in08u_out16s " , 3 , __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Difference kernel metadata. Computes pixelwise the difference between two unsigned 8bit images. Outputs 16bit signed difference result.

Parameters

DIFF_In08u_Out16s_KN	Macro Definition for Kernel name
3	Number of ports
Port DIFF_KN_INA	Define for name of first input image (unsigned 8bit)
Port DIFF_KN_INB	Define for name of second input image (unsigned 8bit)
Port DIFF_KN_OUT	Define for name of difference result of the two images (signed 16bit).

5.8.2.2 KERNEL_INFO apu_diff_in16s_out16s (" apu_diff_in16s_out16s " , 3 , __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Difference kernel metadata. Computes pixelwise the difference between two signed 16bit images. Outputs 16bit signed difference result.

Warning

No sanity checks for out of bounds values above/below +/-2^(15-1) are performed.

Parameters

DIFF_In16s_←Out16s_KN	Macro Definition for Kernel name
3	Number of ports
Port DIFF_KN_INA	Define for name of first input image (signed 16bit)
Port DIFF_KN_INB	Define for name of second input image (signed 16bit)
Port DIFF_KN_OUT	Define for name of result of the two images (signed 16bit).

5.8.2.3 KERNEL_INFO apu_diff_in16s_out32s (" apu_diff_in16s_out32s " , 3 , __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Difference kernel metadata. Computes pixelwise the difference between two signed 16bit images. Outputs 32bit signed difference result.

Parameters

DIFF_In16s_←Out32s_KN	Macro Definition for Kernel name
3	Number of ports
Port DIFF_KN_INA	Define for name of first input image (signed 16bit)
Port DIFF_KN_INB	Define for name of second input image (signed 16bit)
Port DIFF_KN_OUT	Define for name of result of the two images (signed 32bit).

5.8.2.4 KERNEL_INFO apu_diff_in32s_out32s (" apu_diff_in32s_out32s " , 3 , __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Difference kernel metadata. Computes pixelwise the difference between two signed 32bit images. Outputs 32bit signed difference result.

Warning

No sanity checks for out of bounds values above/below +/-2^(31-1) are performed.

Parameters

DIFF_In32s_↵ Out32s_KN	Macro Definition for Kernel name
3	Number of ports
Port DIFF_KN_INA	Define for name of first input image (signed 32bit)
Port DIFF_KN_INB	Define for name of second input image (signed 32bit)
Port DIFF_KN_OUT	Define for name of result of the two images (signed 32bit).

5.8.2.5 KERNEL_INFO apu_diff_in64s_out64s (" apu_diff_in64s_out64s " , 6 , __port(__index(0), __identifier("INPUT_A_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_A_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("INPUT_B_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("INPUT_B_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("OUTPUT_HIGH"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(5), __identifier("OUTPUT_LOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Difference kernel metadata. Computes pixelwise the difference between two signed 64bit images. Outputs 64bit signed difference result.

Warning

No sanity checks for out of bounds values above/below +/- 2^(63-1) are performed.

Parameters

DIFF_In64s_↵ Out64s_KN	Macro Definition for Kernel name
6	Number of ports
Port DIFF_KN↵ _INA_HIGH	Define for name of first signed 32bit high word of input image
Port DIFF_KN↵ _INA_LOW	Define for name of first unsigned 32bit low word of input image
Port DIFF_KN↵ _INB_HIGH	Define for name of first signed 32bit high word of input image
Port DIFF_KN↵ _INB_LOW	Define for name of first unsigned 32bit low word of input image
Port DIFF_KN↵ _OUT_HIGH	Define for name of signed 32bit high word of difference result
Port DIFF_KN↵ _OUT_LOW	Define for name of unsigned 32bit low word of difference result

5.8.2.6 void difference_filter_in08u_out16s (vec16s * *dst,* vec08u * *srcA,* vec08u * *srcB,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Pixel-wise difference between two unsigned 8bit images => signed 16bit.

Pixel-wise difference between two unsigned 8bit images with signed 16bit result. dst[i] = srcImageA[i] - srcImageB[i];

Parameters

dst	- [Output] 16bit Destination block pointer
srcA	- [Input] 8bit Source block pointer of img A
srcB	- [Input] 8bit Source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Output block width (in elements not bytes) including padding

5.8.2.7 void difference_filter_in16s_out16s (vec16s * *dst*, vec16s * *srcA*, vec16s * *srcB*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Pixel-wise difference between two signed 16bit images => signed 16bit.

Pixel-wise difference between two signed 16bit images with 16bit result.

Warning

No out of range is checked

Parameters

dst	- [Output] signed 16bit Destination block pointer
srcA	- [Input] signed 16bit Source block pointer of img A
srcB	- [Input] signed 16bit Source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Output block width (in elements not bytes) including padding

5.8.2.8 void difference_filter_in16s_out32s (vec32s * *dst*, vec16s * *srcA*, vec16s * *srcB*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Pixel-wise difference between two signed 16bit images => signed 32bit.

Pixel-wise difference between two signed 16bit images with signed 32bit result

Parameters

dst	- [Output] signed 32bit Destination block pointer
srcA	- [Input] signed 16bit Source block pointer of img A
srcB	- [Input] signed 16bit Source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Output block width (in elements not bytes) including padding

5.8.2.9 void difference_filter_in32s_out32s (vec32s * *dst*, vec32s * *srcA*, vec32s * *srcB*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Pixel-wise difference between two signed 16bit images => signed 32bit.

Pixel-wise difference between two signed 32bit images with signed 32bit result

Warning

No out of range is checked!

Parameters

dst	- [Output] signed 32bit Destination block pointer
srcA	- [Input] signed 32bit source block pointer of img A
srcB	- [Input] signed 32bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Output block width (in elements not bytes) including padding

5.8.2.10 void difference_filter_in32s_out64s (vec32s ∗ *dst_high,* vec32u ∗ *dst_low,* vec32s ∗ *srcA,* vec32s ∗ *srcB,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Pixel-wise difference between two signed 32bit images => signed 64bit.

Pixel-wise difference between two signed 32bit images with signed 64 bit result

Parameters

dst_high	- [Output] signed 32bit high word of destination block pointer
dst_low	- [Output] unsigned 32bit low word of destination block pointer
srcA	- [Input] signed 32bit source block pointer of img A
srcB	- [Input] signed 32bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Output block width (in elements not bytes) including padding

5.8.2.11 void difference_filter_in32u_out32s (vec32s ∗ *dst,* vec32u ∗ *srcA,* vec32u ∗ *srcB,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Pixel-wise difference between two unsigned 32bit images => signed 32bit.

Pixel-wise difference between two unsigned 32bit images with signed 32bit result.

Warning

No out of range is checked. If the unsigned inputs are higher than 0x7fff ffff, wrong results will arise

Parameters

dst	- [Output] 32bit high word of destination block pointer
srcA	- [Input] unsigned 32bit source block pointer of img A
srcB	- [Input] unsigned 32bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Output block width (in elements not bytes) including padding

5.8.2.12 void difference_filter_in64s_out64s (vec32s ∗ *dst_high,* vec32u ∗ *dst_low,* vec32s ∗ *srcA_high,* vec32u ∗ *srcA_low,* vec32s ∗ *srcB_high,* vec32u ∗ *srcB_low,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Pixel-wise difference between two signed 64bit images => signed 64bit.

Pixel-wise difference between two signed 64bit images, given as low and high word blocks with signed 64bit result

Parameters

dst_high	- [Output] signed 32bit high word of destination block pointer
dst_low	- [Output] unsigned 32bit low word of destination block pointer
srcA_high	- [Input] signed 32bit high word source block pointer of img A
srcA_low	- [Input] unsigned 32bit low word source block pointer of img A
srcB_high	- [Input] signed 32bit high word source block pointer of img B
srcB_low	- [Input] unsigned 32bit low word source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Output block width (in elements not bytes) including padding

5.8.2.13 void difference_filter_in64u_out64s (vec32s ∗ *dst_high*, vec32u ∗ *dst_low*, vec32u ∗ *srcA_high*, vec32u ∗ *srcA_low*, vec32u ∗ *srcB_high*, vec32u ∗ *srcB_low*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Pixel-wise difference between two unsigned 64bit images => signed 64bit.

Pixel-wise difference between two unsigned 64bit images, given as low and high word blocks with signed 64bit result

Parameters

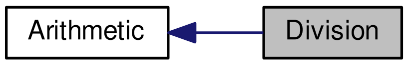
dst_high	- [Output] signed 32bit high word of destination block pointer
dst_low	- [Output] unsigned 32bit low word of destination block pointer
srcA_high	- [Input] unsigned 32bit high word source block pointer of img A
srcA_low	- [Input] unsigned 32bit low word source block pointer of img A
srcB_high	- [Input] unsigned 32bit high word source block pointer of img B
srcB_low	- [Input] unsigned 32bit low word source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Output block width (in elements not bytes) including padding

5.9 Division

5.9.1 Detailed Description

Element-wise division.

Collaboration diagram for Division:



Functions

- KERNEL_INFO apu_dot_division (" apu_dot_division ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_log2 (" apu_dot_log2 ", 2, __port(__index(0), __identifier("Log2_input"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("Log2_fact"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_inv_NewtonRaphson (" apu_dot_inv_NewtonRaphson ", 4, __port(__index(0), __identifier("Inv_Inverse_divisor"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("Inv_divisor"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Inv_log2fact"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("Inv_shiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_division_N64s_D32s_Q64s (" apu_dot_division_N64s_D32s_Q64s ", 6, __port(__index(0), __identifier("Div_N64s_NOM_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("Div_N64s_NOM_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Div_N64s_DIVISOR"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("Div_N64s_OUT_HIGH"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("Div_N64s_OUT_LOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("Div_N64s_OUT_REM"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- void dot_division_filter (vec32s *res, vec32s *numerator, vec32s *denominator, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise division of signed 32bit integers.
- void computeLog2 (vec08u *log2Fact, vec32s *input, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  res[i] = log2(abs(input[i])) + 1, where input is a signed 32bit integer
- void computeLog2u (vec08u *log2Fact, vec32u *input, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

res[i] = log2(input[i]) + 1, where input is a unsigned 32bit integer

- void compute64bitLog2 (vec08u ∗log2Fact, vec32s ∗input_high, vec32u ∗input_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  res[i] = log2(abs(input[i])) + 1, where input is a signed 64bit integer
- void compute64bitLog2u (vec08u ∗log2Fact, vec32u ∗input_high, vec32u ∗input_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  res[i] = log2(input[i]) + 1, where input is a unsigned 64bit integer
- void computeInv_NewtonRaphson (vec32s ∗invDiv, vec32s ∗div, vec08u ∗log2Fact, const int08s shiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise inverse of a signed integer vector using the NewtonRaphson algorithm.
- void dot_division_filter_N64s_D32s_Q64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32u ∗dst_rem, vec32s ∗nom_high, vec32u ∗nom_low, vec32s ∗divisor, int16s bw, int16s bh, int16s inStrideWidth, int16s outStride↩ Width)

  Elementwise division of a 64bit nominator by a 32bit divisor.

5.9.2 Function Documentation

5.9.2.1 KERNEL_INFO apu_dot_division (" apu_dot_division " , 3 , __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Integer division kernel metadata. Divides pixelwise two signed 32bit images. Outputs signed 32bit integer division result.

Warning

Division by zero returns zero and not an out of range number.

Parameters

DIV_In32s_↩ Out_32s_KN	Define for Kernel name
3	Number of ports
Port DIV_KN_INA	Define for name of first input image (signed 32bit).
Port DIV_KN_INB	Define for name of second input image (signed 32bit).
Port DIV_KN_OUT	Define for name of division result (signed 32bit).

5.9.2.2 KERNEL_INFO apu_dot_division_N64s_D32s_Q64s (" apu_dot_division_N64s_D32s_Q64s " , 6 , __port(__index(0), __identifier("Div_N64s_NOM_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("Div_N64s_NOM_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("Div_N64s_DIVISOR"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("Div_N64s_OUT_HIGH"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("Div_N64s_OUT_LOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(5), __identifier("Div_N64s_OUT_REM"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Integer division kernel metadata. Divides pixelwise a 64bit integer image with a 32bit integer. Outputs signed 64bit integer division result.

Warning

Division by zero returns zero and not an out of range number.

Parameters

DIV_N64s_↩ D32s_Q64s_K	Define for Kernel name
6	Number of ports
Port DIV_N64s↩ _D32s_Q64s_↩ KN_NOM_HIGH	Define for name of signed 32bit high word of input image (i.e. the nominator)
Port DIV_N64s↩ _D32s_Q64s_↩ KN_NOM_LOW	Define for name of unsigned 32bit low word of input image (i.e. the nominator)
Port DIV_N64s↩ _D32s_Q64s_↩ KN_DIVISOR	Define for name of of divisor (i.e. the denominator) (signed 32bit)
Port DIV_N64s↩ _D32s_Q64s_↩ KN_OUT_HIGH	Define for name of signed 32bit high word of division result
Port DIV_N64s↩ _D32s_Q64s_↩ KN_OUT_LOW	Define for name of unsigned 32bit low word of division result
Port DIV_N64s↩ _D32s_Q64s_↩ KN_OUT_REM	Define for name of integer remainder of the division (unsigned 32bit)

5.9.2.3 KERNEL_INFO apu_dot_inv_NewtonRaphson (" apu_dot_inv_NewtonRaphson " , 4 , __port(__index(0), __identifier("Inv_Inverse_divisor"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("Inv_divisor"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("Inv_log2fact"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("Inv_shiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)))

Inverse computation kernel metadata. Computes inverse of each image pixel with the Newton-Raphson method Outputs signed 32bit inverse of the input in fixed point Q<ShiftFact>.(31-<ShiftFact>) format

Parameters

COMPUTE_IN↩ V_KN	Define for Kernel name
4	Number of ports
Port INV_KN_I↩ NV_DIV	Define for name of the inverse of div in Q<ShiftFact>.(31-<ShiftFact>)
Port INV_KN_DIV	Define for name of the image containing 32bit numbers to be inverted
Port INV_KN_↩ LOG2Fact	Define for name of the returned image of unsigned 8bit log_2 values of each member of div
Port INV_KN_↩ SHIFTFact	Define for name of the shift factor for the fixed point Q<ShiftFact>.(31-<ShiftFact>) format

5.9.2.4 KERNEL_INFO apu_dot_log2 (" apu_dot_log2 " , 2 , __port(__index(0), __identifier("Log2_input"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("Log2_fact"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Computes log2 of the absolute value of each image pixel. Outputs log2(abs(Input)) + 1.

Parameters

LOG2_KN	Define for Kernel name
2	Number of ports
Port LOG2_KN_IN	Define for name of input image (signed 32bit).
Port LOG2_KN↩_LOG2Fact	Define for name of logarithm result (signed 32bit).

5.9.2.5 void compute64bitLog2 (vec08u ∗ *log2Fact,* vec32s ∗ *input_high,* vec32u ∗ *input_low,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

res[i] = log2(abs(input[i])) + 1, where input is a signed 64bit integer

Computes the log_2 + 1 of all the elements of the signed 64bit input. This is a helper function

Parameters

log2Fact	- [Output] the value representing 1+log_2(input)
input_high	- [Input] the high word of the 64bit input vector
input_low	- [Input] the low word of the 64bit input vector
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width(in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width(in elements not bytes) including padding

5.9.2.6 void compute64bitLog2u (vec08u ∗ *log2Fact,* vec32u ∗ *input_high,* vec32u ∗ *input_low,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

res[i] = log2(input[i]) + 1, where input is a unsigned 64bit integer

Computes the log_2 + 1 of all the elements of the unsigned 64bit input. This is a helper function

Parameters

log2Fact	- [Output] the value representing 1+log_2(input)
input_high	- [Input] the high word of the 64bit input vector
input_low	- [Input] the low word of the 64bit input vector
bw	- [Input] Block width
bh	- [I nput] Block height
inStrideWidth	- [Input] Source block width(in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width(in elements not bytes) including padding

5.9.2.7 void computeInv_NewtonRaphson (vec32s ∗ *invDiv,* vec32s ∗ *div,* vec08u ∗ *log2Fact,* const int08s *shiftFact,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise inverse of a signed integer vector using the NewtonRaphson algorithm.

Compute the inverse of a 32bit integer in Q<ShiftFact>.(31-<ShiftFact>) format with the NewtonRaphson algorithm

Parameters

invDiv	- [Output] the inverse of div in Q<ShiftFact>.(31-<ShiftFact>)

div	- [Input] the vector containing the numbers to be inverted (signed 32bit)
log2Fact	- [Output] Returns the vector of log_2 values of each member of div
shiftFact	- [Input] the shift factor for the fixed point Q<ShiftFact>.(31-<ShiftFact>) format
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.9.2.8 void computeLog2 (vec08u ∗ *log2Fact,* vec32s ∗ *input,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

res[i] = log2(abs(input[i])) + 1, where input is a signed 32bit integer

Pixelwise Computation the log_2 + 1 of all the elements of the input. This is a helper function for computeInv_NewtonRaphson

Parameters

log2Fact	- [Output] the values representing 1+log_2(abs(input))
input	- [Input] the 32bit signed input vector
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width(in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width(in elements not bytes) including padding

5.9.2.9 void computeLog2u (vec08u ∗ *log2Fact,* vec32u ∗ *input,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

res[i] = log2(input[i]) + 1, where input is a unsigned 32bit integer

Computes the log_2 + 1 of all the elements of the input. This is a helper function for computeInv_NewtonRaphson

Parameters

log2Fact	- [Output] the values representing 1+log_2(input)
input	- [Input] the 32bit unsigned input vector
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width(in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width(in elements not bytes) including padding

5.9.2.10 void dot_division_filter (vec32s ∗ *res,* vec32s ∗ *numerator,* vec32s ∗ *denominator,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise division of signed 32bit integers.

Divide 32 bit numbers with the APEX internal division. quot[i] = numerator[i]/denominator[i], foreach i

Parameters

quot	- [Output] Destination block pointer
numerator	- [Input] Source block pointer of img A
denominator	- [Input] Source block pointer of img B

bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.9.2.11 void dot_division_filter_N64s_D32s_Q64s (vec32s ∗ *dst_high*, vec32u ∗ *dst_low*, vec32u ∗ *dst_rem*, vec32s ∗ *nom_high*, vec32u ∗ *nom_low*, vec32s ∗ *divisor*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Elementwise division of a 64bit nominator by a 32bit divisor.

Compute the 64bit division of a 64bit integer with a 32bit integer

Parameters

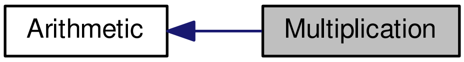
dst_high	- [Output] High word of destination block pointer
dst_low	- [Output] Low word of destination block pointer
dst_rem	- [Output] Remainder of the division (i.e. decimal part as integer)
nom_high	- [Input] Source block pointer to high word of nominator
nom_low	- [Input] Source block pointer to low word of nominator
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10 Multiplication

5.10.1 Detailed Description

Element-wise multiplication.

Collaboration diagram for Multiplication:



Functions

- KERNEL_INFO apu_dot_mult_in16s_out32s (" apu_dot_mult_in16s_out32s ", 3, __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_in32s_out32s (" apu_dot_mult_in32s_out32s ", 3, __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_in32s_out64s (" apu_dot_mult_in32s_out64s ", 4, __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("DotMultKn_OUT_High"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("DotMultKn_OUT_Low"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_in32s_in16s_out32s (" apu_dot_mult_in32s_in16s_out32s ", 3, __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_scalar_in08u_out16s (" apu_dot_mult_scalar_in08u_out16s ", 3, __port(__index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultScalKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_scalar_in32s_out32s (" apu_dot_mult_scalar_in32s_out32s ", 3, __port(__index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultScalKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), _

_ek_size(1, 1)), __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_left_shift_in16u_out16s (" apu_dot_left_shift_in16u_out16s ", 3, __port(__index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultScalKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_right_shift_in64s_out64s (" apu_dot_right_shift_in64s_out64s ", 5, __port(__index(0), __identifier("RightShift_64bit_InHigh"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("RightShift_64bit_InLOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("RightShift_64bit_OutHigh"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("RightShift_64bit_OutLOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("RghtShft_64bit_ShiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_right_shift_in64s_out32s (" apu_dot_right_shift_in64s_out32s ", 4, __port(__index(0), __identifier("RightShift_64bit_InHigh"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("RightShift_64bit_InLOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("RightShift_32bit_Out"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("RghtShft_64bit_ShiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_lsh1_in32s_out32s (" apu_dot_lsh1_in32s_out32s ", 2, __port(__index(0), __identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultBy2Kn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_lsh1_in32s_out64s (" apu_dot_lsh1_in32s_out64s ", 3, __port(__index(0), __identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultBy2Kn_Output_high"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("MultBy2Kn_Output_low"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_lsh1_in32s_Q3_28_out64s (" apu_dot_lsh1_in32s_Q3_28_out64s ", 3, __port(__index(0), __identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultBy2_Q3_28_Kn_Output_int"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("MultBy2_Q3_28_Kn_Output_frac"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- vbool hasSign (vec32s &a, vec32s &b)

  *sign(a) * sign(b) == -1*
- void change64bitSign (vec32s &highWord, vec32u &lowWord)

  In place sign change of a 64bit integer, i.e. a = -a.
- void dot_mult_in16s_out32s_filter (vec32s *dst, vec16s *srcA, vec16s *srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 16bit multiplication => signed 32bit.
- void dot_mult_in32s_out32s_filter (vec32s *dst, vec32s *srcA, vec32s *srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit multiplication => signed 32bit.
- void dot_mult_in32s_in16s_out32s_filter (vec32s *dst, vec32s *srcA, vec16s *srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit and 16bit multiplication => signed 32bit.

- void dot_mult_in32s_out64s_filter (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA, vec32s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit multiplication => signed 64bit.
- void dot_mult_in32u_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗srcA, vec32u ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise unsigned 32bit multiplication => unsigned 64bit.
- void dot_mult_in64s_out64s_filter (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA_high, vec32u ∗srcA_low, vec32s ∗srcB_high, vec32u ∗srcB_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 64bit multiplication => signed 64bit.
- void dot_mult_in64u_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗srcA_high, vec32u ∗srcA_low, vec32u ∗srcB_high, vec32u ∗srcB_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise unsigned 64bit multiplication => unsigned 64bit.
- void dot_mult_scalar_in08u_out16s_filter (vec16s ∗dst, vec08u ∗srcA, int32s scalar, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise unsigned 8bit multiplication with a fixed scalar => signed 16bit.
- void dot_mult_scalar_in32s_out32s_filter (vec32s ∗dst, vec32s ∗srcA, int32s scalar, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit multiplication with a fixed scalar => signed 32bit.
- void lsh_in16u_out16s_filter (vec16s ∗upShifted, vec16u ∗src, vec16s ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 16bit left shift with a signed 16bit shift vector => signed 16bit.
- void lsh_in32u_out32u_filter (vec32u ∗upShifted, vec32u ∗src, vec08u ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 32bit left shift with an unsigned 8bit shift vector => unsigned 32bit.
- void lsh_in32s_out32s_filter (vec32s ∗upShifted, vec32s ∗src, vec08u ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 32bit left shift with an unsigned 8bit shift vector => signed 32bit.
- void lsh_in32s_out64s_filter (vec32s ∗upShifted_high, vec32u ∗upShifted_low, vec32s ∗src, vec08u ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 32bit left shift with an unsigned 8bit shift vector => signed 64bit.
- void lsh_in32u_out64u_filter (vec32u ∗upShifted_high, vec32u ∗upShifted_low, vec32u ∗src, vec08u ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 32bit left shift with an unsigned 8bit shift vector => unsigned 64bit.
- void lsh_in32s_Q3_28_out64s_filter (vec32s ∗upShifted_int, vec32s ∗upShifted_frac, vec32s ∗src, vec08u ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 32bit left shift of a 32bit matrix in Q3_28 format with an unsigned 8bit shift vector => signed 64bit.
- void rsh_in32u_out32u_filter (vec32u ∗downShift, vec32u ∗src, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 32bit right shift with an unsigned 8bit shift vector => unsigned 32bit.
- void rsh_in32s_out32s_filter (vec32s ∗downShift, vec32s ∗src, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 32bit right shift with a signed 8bit shift vector => signed 32bit.
- void rsh_in64s_out64s_filter (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗in_high, vec32u ∗in_low, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 64bit.
- void rsh_in64u_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗in_high, vec32u ∗in_low, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 64bit.
- void rsh_in64s_out32s_filter (vec32s ∗dst, vec32s ∗in_high, vec32u ∗in_low, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 32bit.

5.10.2 Function Documentation

5.10.2.1 KERNEL_INFO apu_dot_left_shift_in16u_out16s (" apu_dot_left_shift_in16u_out16s " , 3 , __port(__index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MultScalKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Left shift kernel metadata. Shifts to the left each pixel of a unsigned 16bit image by a scalar shift value. Outputs signed 16bit shift result

Warning

No checks are performed for carry-over resulting from the left shift.

Parameters

LEFT_SHIFT_In16u_Out16s_KN	Define for Kernel name
3	Number of ports
Port MULT_SCALAR_KN_IN	Define for name of input image (unsigned 16bit)
Port MULT_SCALAR_KN_OUT	Define for name of shift result (signed 16bit)
Port MULT_SCALAR_KN_IN_SCALAR	Define for name of scalar used to left shift the image pixels (unsigned 32bit).

5.10.2.2 KERNEL_INFO apu_dot_lsh1_in32s_out32s (" apu_dot_lsh1_in32s_out32s " , 2 , __port(__index(0), __identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MultBy2Kn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Left shift kernel metadata by one. Shifts to the left each pixel of a signed 32bit image by one.Outputs signed 32bit shift result

Warning

No checks are performed for carry-over resulting from the left shift.

Parameters

MULT_BY_2_In32s_Out32s_KN	Define for Kernel name
2	Number of ports
Port MULT_BY_2_In32s_Out32s_KN_IN	Define for name of the input image (signed 32bit)

Port MULT_BY_2_↩ In32s_Out32s↩ _KN_OUT	Define for name of the shift result (signed 32bit)

5.10.2.3 KERNEL_INFO apu_dot_lsh1_in32s_out64s (" apu_dot_lsh1_in32s_out64s " , 3 , __port(__index(0), __identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MultBy2Kn_Output_high"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("MultBy2Kn_Output_low"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Left shift kernel metadata by one. Shifts to the left each pixel of a signed 32bit image by one. Outputs signed 64bit shift result

Parameters

MULT_BY_2_↩ In32s_Out64s↩ _KN	Define for Kernel name
3	Number of ports
Port MULT_BY↩ _2_In32s_↩ Out32s_KN_IN	Define for name of the input image (signed 32bit)
Port MULT_BY_2_↩ In32s_Out32s↩ _KN_OUT_H	Define name of signed 32bit high word of shift result
Port MULT_BY_2_↩ In32s_Out32s↩ _KN_OUT_LOW	Define name of unsigned 32bit low word of shift result

5.10.2.4 KERNEL_INFO apu_dot_lsh1_in32s_Q3_28_out64s (" apu_dot_lsh1_in32s_Q3_28_out64s " , 3 , __port(__index(0), __identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MultBy2_Q3_28_Kn_Output_int"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("MultBy2_Q3_28_Kn_Output_frac"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Left shift kernel metadata by one for fixed point numbers in Q3.28 format. Shifts to the left by one each pixel of a signed 32bit image in fixed point Q3.28 format. Outputs signed 64bit shift result in fixed point Q3.28 format

Parameters

MULT_BY_2_↩ Q3_28_In32s_↩ Out64s_KN	Define for Kernel name
3	Number of ports
Port MULT_BY↩ _2_In32s_↩ Out32s_KN_IN	Define for name of the input image (signed 32bit)

Port MULT_BY←_2_Q3_28_KN←_OUT_INT	Define name of signed 32bit integer part of shift result (in normal integer format)
Port MULT_BY←_2_Q3_28_KN←_OUT_FRAC	Define name of unsigned 32bit fractional part of shift result (in fixed point Q3.28 format).

5.10.2.5 KERNEL_INFO apu_dot_mult_in16s_out32s (" apu_dot_mult_in16s_out32s " , 3 , __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Multiplication kernel metadata. Multiplies pixelwise two signed 16bit images. Outputs signed 32bit multiplication result

Parameters

MULT_In16s_←Out32s_KN	Define for Kernel name
3	Number of ports
Port MULT_KN_INA	Define for name of first input image name (signed 16bit)
Port MULT_KN_INB	Define for name of second input image name (signed 16bit)
Port MULT_KN_OUT	Define name of multiplication result (signed 32bit).

5.10.2.6 KERNEL_INFO apu_dot_mult_in32s_in16s_out32s (" apu_dot_mult_in32s_in16s_out32s " , 3 , __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Multiplication kernel metadata. Multiplies pixelwise two signed 32bit images. Outputs signed 64bit multiplication result

Parameters

MULT_In32s_←Out64s_KN	Define for Kernel name
4	Number of ports
Port MULT_KN_INA	Define for name of first input image (signed 32bit)
Port MULT_KN_INB	Define for name of second input image (signed 32bit)
Port MULT_K←N_OUT_HIGH	Define for name of signed 32bit high word of multiplication result
Port MULT_K←N_OUT_LOW	Define for name of unsigned 32bit low word of multiplication result

5.10.2.7 KERNEL_INFO apu_dot_mult_in32s_out32s (" apu_dot_mult_in32s_out32s " , 3 , __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Multiplication kernel metadata. Multiplies pixelwise two signed 32bit images. Outputs signed 32bit multiplication result

Warning

Does not check out of range values above/below +/-2^(31-1)

Parameters

MULT_In32s_↩ Out32s_KN	Define for Kernel name
3	Number of ports
Port MULT_KN_INA	Define for name of first input image (signed 32bit)
Port MULT_KN_INB	Define for name of second input image (signed 32bit)
Port MULT_KN_OUT	Define for name of multiplication result (signed 32bit).

5.10.2.8 KERNEL_INFO apu_dot_mult_in32s_out64s (" apu_dot_mult_in32s_out64s " , 4 , __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("DotMultKn_OUT_High"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("DotMultKn_OUT_Low"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Multiplication kernel metadata. Multiplies pixelwise two signed 32bit images. Outputs signed 64bit multiplication result

Parameters

MULT_In32s_↩ Out64s_KN	Define for Kernel name
4	Number of ports
Port MULT_KN_INA	Define for name of first input image (signed 32bit)
Port MULT_KN_INB	Define for name of second input image (signed 32bit)
Port MULT_K↩ N_OUT_HIGH	Define for name of signed 32bit high word of multiplication result
Port MULT_K↩ N_OUT_LOW	Define for name of unsigned 32bit low word of multiplication result

5.10.2.9 KERNEL_INFO apu_dot_mult_scalar_in08u_out16s (" apu_dot_mult_scalar_in08u_out16s " , 3 , __port(__index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MultScalKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Scalar multiplication kernel metadata. Multiplies pixelwise a unsigned 8bit image with a scalar value. Outputs signed 16bit multiplication result

Warning

The type of the scalar value is int32s for convenience. It should have values in the maximal range of shorts. No checks are performed for out of range values below/above +/-2^(15-1).

Parameters

MULT_SCALAR_In08u_Out16s_KN	Define for Kernel name
3	Number of ports
Port MULT_SCALAR_KN_IN	Define for name of input image (signed 32bit)
Port MULT_SCALAR_KN_OUT	Define for name of multiplication result (signed 32bit)
Port MULT_SCALAR_KN_IN_SCALAR	Define for name of scalar used to multiply the image (signed 32bit).

5.10.2.10 KERNEL_INFO apu_dot_mult_scalar_in32s_out32s (" apu_dot_mult_scalar_in32s_out32s " , 3 , __port(__index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MultScalKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Scalar multiplication kernel metadata. Multiplies pixelwise a signed 32bit image with a scalar value. Outputs signed 32bit multiplication result

Warning

No checks are performed for out of range values below/above +/-2^(31-1).

Parameters

MULT_SCALAR_In32s_Out32s_KN	Define for Kernel name
3	Number of ports
Port MULT_SCALAR_KN_IN	Define for name of the input image (signed 32bit)

Port MULT_SCALA↵R_KN_OUT	Define for name of multiplication result (signed 32bit)
Port MULT_S↵CALAR_KN_I↵N_SCALAR	Define for name of scalar used to multiply the image (signed 32bit).

5.10.2.11 KERNEL_INFO apu_dot_right_shift_in64s_out32s (" apu_dot_right_shift_in64s_out32s " , 4 , __port(__index(0), __identifier("RightShift_64bit_InHigh"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("RightShift_64bit_InLOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("RightShift_32bit_Out"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("RghtShft_64bit_ShiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Right shift kernel metadata. Shifts to the right each pixel of a signed 64bit image by a scalar shift value. Outputs signed 32bit shift result

Parameters

RIGHT_SHIFT↵_In64s_↵Out32s_KN	Define for Kernel name
5	Number of ports
Port RIGHT_SHIFT↵_In64s_HIGH	Define for name of the signed 32bit high word of the input image
Port RIGHT_SHIFT↵_In64s_LOW	Define for name of the unsigned 32bit low word of the input image
Port RIGHT_SHIFT↵_Out64s_HIGH	Define for name of the signed 32bit high word of the shift result
Port RIGHT_SHIFT↵_Out64s_LOW	Define for name of the unsigned 32bit low word of the shift result
Port RIGHT_S↵HIFT_Fact	Define for name of the unsigned 32bit scalar used to left shift the image pixels.

5.10.2.12 KERNEL_INFO apu_dot_right_shift_in64s_out64s (" apu_dot_right_shift_in64s_out64s " , 5 , __port(__index(0), __identifier("RightShift_64bit_InHigh"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("RightShift_64bit_InLOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("RightShift_64bit_OutHigh"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("RightShift_64bit_OutLOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("RghtShft_64bit_ShiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

Right shift kernel metadata. Shifts to the right each pixel of a signed 64bit image by a scalar shift value. Outputs signed 64bit shift result

Parameters

RIGHT_SHIFT↩_In64s_↩Out64s_KN	Define for Kernel name
5	Number of ports
Port RIGHT_SHIFT↩_In64s_HIGH	Define for name of the signed 32bit high word of the input image
Port RIGHT_SHIFT↩_In64s_LOW	Define for name of the unsigned 32bit low word of the input image
Port RIGHT_SHIFT↩_Out64s_HIGH	Define for name of the signed 32bit high word of the shift result
Port RIGHT_SHIFT↩_Out64s_LOW	Define for name of the unsigned 32bit low word of the shift result
Port RIGHT_S↩HIFT_Fact	Define name of the unsigned 32bit scalar used to left shift the image pixels.

5.10.2.13 void change64bitSign (vec32s & *highWord,* vec32u & *lowWord*)

In place sign change of a 64bit integer, i.e. a = -a.

Change the sign of 64bit values stored in two vectors, a high and a low-word vector.

Parameters

highWord	- [Input/output] high word input/output. Result is stored directly into same vector
lowhWord	- [Input/output] low word input/output. Result is stored directly into same vector

5.10.2.14 void dot_mult_in16s_out32s_filter (vec32s ∗ *dst,* vec16s ∗ *srcA,* vec16s ∗ *srcB,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 16bit multiplication => signed 32bit.

Dot multiplication between two 16bit matrices (i.e multiply elementwise the matrix coeffs), with signed 32bit result. dst[i] = srcImageA[i] ∗ srcImageB[i]

Parameters

dst	- [Output] signed 32bit destination block pointer
srcA	- [Input] signed 16bit source block pointer of img A
srcB	- [Input] signed 16bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.15 void dot_mult_in32s_in16s_out32s_filter (vec32s ∗ *dst,* vec32s ∗ *srcA,* vec16s ∗ *srcB,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 32bit and 16bit multiplication => signed 32bit.

Dot multiplication between a signed 32bit and a signed 16bit matrix (i.e multiply elementwise the matrix coeffs), with signed 32bit result

Warning

No out of range is taken into consideration!

Parameters

dst	- [Output] signed 32bit destination block pointer
srcA	- [Input] signed 32bit source block pointer of img A
srcB	- [Input] signed 16bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.16 void dot_mult_in32s_out32s_filter (vec32s ∗ *dst,* vec32s ∗ *srcA,* vec32s ∗ *srcB,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 32bit multiplication => signed 32bit.

Dot multiplication between two 32bit matrices (i.e multiply elementwise the matrix coeffs), with signed 32bit result

Warning

No out of range is taken into consideration!

Parameters

dst	- [Output] signed 32bit destination block pointer
srcA	- [Input] signed 32bit source block pointer of img A
srcB	- [Input] signed 32bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.17 void dot_mult_in32s_out64s_filter (vec32s ∗ *dst_high,* vec32u ∗ *dst_low,* vec32s ∗ *srcA,* vec32s ∗ *srcB,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 32bit multiplication => signed 64bit.

Dot 64bit multiplication between two signed 32bit matrices (i.e multiply elementwise the matrix coeffs). Results have 64bits and are organized into two 32bit matrices (i.e the lower and the higher word matrices of the multiplication result)

Parameters

dst_high	- [Output] signed 32bit high word of destination block pointer
dst_low	- [Output] signed 32bit low word of destination block pointer
srcA	- [Input] signed 32bit source block pointer of img A
srcB	- [Input] signed 32bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding

outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.18 void dot_mult_in32u_out64u_filter (vec32u ∗ *dst_high,* vec32u ∗ *dst_low,* vec32u ∗ *srcA,* vec32u ∗ *srcB,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise unsigned 32bit multiplication => unsigned 64bit.

Dot 64bit multiplication between two unsigned 32bit matrices (i.e multiply elementwise the matrix coeffs). Results have 64bits and are organized into two 32bit matrices (i.e the lower and the higher word matrices of the multiplication result)

Parameters

dst_high	- [Output] unsigned 32bit high word of destination block pointer
dst_low	- [Output] signed 32bit low word of destination block pointer
srcA	- [Input] unsigned 32bit source block pointer of img A
srcB	- [Input] unsigned 32bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.19 void dot_mult_in64s_out64s_filter (vec32s ∗ *dst_high,* vec32u ∗ *dst_low,* vec32s ∗ *srcA_high,* vec32u ∗ *srcA_low,* vec32s ∗ *srcB_high,* vec32u ∗ *srcB_low,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 64bit multiplication => signed 64bit.

Dot 64bit multiplication between two signed 64bit matrices (i.e multiply elementwise the matrix coeffs). Results have 64bits and are organized into two 32bit matrices (i.e the lower and the higher word matrices of the multiplication result)

Warning

out of ranges over the 64bit limit are not taken into account!

Parameters

dst_high	- [Output] signed 32bit high word of destination block pointer
dst_low	- [Output] unsigned 32bit low word of destination block pointer
srcA_high	- [Input] signed 32bit high word of 64bit source block pointer of img A
srcA_low	- [Input] unsigned 32bit low word of 64bit source block pointer of img A
srcB_high	- [Input] signed 32bit high word of 64bit source block pointer of img B
srcB_low	- [Input] unsigned 32bit low word of 64bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.20 void dot_mult_in64u_out64u_filter (vec32u ∗ *dst_high,* vec32u ∗ *dst_low,* vec32u ∗ *srcA_high,* vec32u ∗ *srcA_low,* vec32u ∗ *srcB_high,* vec32u ∗ *srcB_low,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise unsigned 64bit multiplication => unsigned 64bit.

Dot 64bit multiplication between two unsigned 64bit matrices (i.e multiply elementwise the matrix coeffs). Results have 64bits and are organized into two 32bit matrices (i.e the lower and the higher word matrices of the multiplication result)

Warning

out of ranges over the 64bit limit are not taken into account!

Parameters

dst_high	- [Output] unsigned 32bit high word of 64bit destination block pointer
dst_low	- [Output] unsigned 32bit low word of 64bit destination block pointer
srcA_high	- [Input] unsigned 32bit high word of 64bit source block pointer of img A
srcA_low	- [Input] unsigned 32bit low word of 64bit source block pointer of img A
srcB_high	- [Input] unsigned 32bit high word of 64bit source block pointer of img B
srcB_low	- [Input] unsigned 32bit low word of 64bit source block pointer of img B
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.21 void dot_mult_scalar_in08u_out16s_filter (vec16s * *dst,* vec08u * *srcA,* int32s *scalar,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise unsigned 8bit multiplication with a fixed scalar => signed 16bit.

Dot multiplication of all elements of a 8bit matrix with a scalar value, with signed 16bit output. dst[i] = srcImage0[i] * scalar;

Warning

The scalar value should have ideally a 8bit value, or maximally a 16bit value.
No out of range is checked!

Parameters

dst	- [Output] signed 16bit destination block pointer
srcA	- [Input] unsigned 8bit source block pointer of img A
scalar	- [Input] signed 32bit the scalar value to be multiplied with the matrix
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.22 void dot_mult_scalar_in32s_out32s_filter (vec32s * *dst,* vec32s * *srcA,* int32s *scalar,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 32bit multiplication with a fixed scalar => signed 32bit.

Dot multiplication of all elements of a 32bit matrix with a 32bit scalar value. Output is of signed 32bits

Warning

No out of range is checked above the 32bit value range!

Parameters

dst	- [Output] 16bit destination block pointer
srcA	- [Input] 8bit source block pointer of img A
scalar	- [Input] 32bit the scalar value to be multiplied with the matrix
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.23 vbool hasSign (vec32s & *a*, vec32s & *b*)

sign(a) $*$ sign(b) == -1

Tests if a times/div b would give a negative result

Parameters

a	- [Input] first operand
b	- [Input] second operand

Returns

a boolean vector where the elements reflect the test operation result

5.10.2.24 void lsh_in16u_out16s_filter (vec16s $*$ *upShifted*, vec16u $*$ *src*, vec16s $*$ *leftShiftFact*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Chunk-wise unsigned 16bit left shift with a signed 16bit shift vector => signed 16bit.

Dot left-shift unsigned 16bit operator by a signed 16bit input vector (each element of one chunk_i is shifted by the shift_vect(i) factor) with signed 16bit output. dst_chunk[i] = srcImage0_chunk[i] << leftShiftFact[i]

Warning

No out of range is checked above the 16bit value range!

Parameters

upShifted	- [Output] signed 16bit destination block pointer for the upShifteded result
src	- [Input] unsigned 16bit source block pointer
leftShiftFact	- [Input] signed 16bit vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.25 void lsh_in32s_out32s_filter (vec32s $*$ *upShifted*, vec32s $*$ *src*, vec08u $*$ *leftShiftFact*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Chunk-wise signed 32bit left shift with an unsigned 8bit shift vector => signed 32bit.

Dot left-shift signed 32bit operator (each element of one chunk_i is shifted by the shift_vect(i) factor) with signed 32bit output

Warning

No out of range is checked above the 32bit value range!

Parameters

upShifted	- [Output] signed 32bit destination block pointer for the upShifteded result
src	- [Input] signed 32bit source block pointer
leftShiftFact	- [Input] unsigned 8bit vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.26 void lsh_in32s_out64s_filter (vec32s ∗ *upShifted_high,* vec32u ∗ *upShifted_low,* vec32s ∗ *src,* vec08u ∗ *leftShiftFact,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Chunk-wise signed 32bit left shift with an unsigned 8bit shift vector => signed 64bit.

Dot left-shift signed 32bit operator (each element of one chunk_i is shifted by the shift_vect(i) factor) with signed 64bit output

Parameters

upShifted_high	- [Output] signed 32bit high word of destination block pointer for the upShifted result
upShifted_low	- [Output] unsigned 32bit low word of destination block pointer for the upShifted result
src	- [Input] signed 32bit source block pointer
leftShiftFact	- [Input] unsigned 8bit vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.27 void lsh_in32s_Q3_28_out64s_filter (vec32s ∗ *upShifted_int,* vec32s ∗ *upShifted_frac,* vec32s ∗ *src,* vec08u ∗ *leftShiftFact,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Chunk-wise unsigned 32bit left shift of a 32bit matrix in Q3_28 format with an unsigned 8bit shift vector => signed 64bit.

Dot left-shift operator for a Q3_28 fixed point input parameter format with 64bit output The output "high" word contains [x] - integer part of the left shifted fixed point input number and the output "low" word contains {x} - fractional part of the left shifted fixed point input number

Parameters

upShifted_int	- [Output] [x] - signed 32bit integer part of the left shifted fixed point input number
upShifted_frac	- [Output] {x} - signed 32bit fractional part of the left shifted fixed point input number
src	- [Input] signed 32bit source block pointer containing fixed point numbers in Q3_28 format
leftShiftFact	- [Input] unsigned 8bit vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.28 void lsh_in32u_out32u_filter (vec32u ∗ *upShifted,* vec32u ∗ *src,* vec08u ∗ *leftShiftFact,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Chunk-wise unsigned 32bit left shift with an unsigned 8bit shift vector => unsigned 32bit.

Dot left-shift unsigned 32bit operator (each element of one chunk_i is shifted by the shift_vect(i) factor) with unsigned 32bit output

Warning

No out of range is checked above the 32bit value range!

Parameters

upShifted	- [Output] unsigned 32bit destination block pointer for the upShifteded result
src	- [Input] unsigned 32bit source block pointer
leftShiftFact	- [Input] unsigned 8bit vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.29 void lsh_in32u_out64u_filter (vec32u ∗ *upShifted_high,* vec32u ∗ *upShifted_low,* vec32u ∗ *src,* vec08u ∗ *leftShiftFact,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Chunk-wise unsigned 32bit left shift with an unsigned 8bit shift vector => unsigned 64bit.

Dot left-shift unsigned 32bit operator (each element of one chunk_i is shifted by the shift_vect(i) factor) with unsigned 64bit output

Parameters

upShifted	- [Output] unsigned 32bit destination block pointer for the upShifted result
src	- [Input] unsigned 32bit source block pointer
leftShiftFact	- [Input] the vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.30 void rsh_in32s_out32s_filter (vec32s ∗ *downShift,* vec32s ∗ *src,* vec08u ∗ *rightShiftFact,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Chunk-wise signed 32bit right shift with a signed 8bit shift vector => signed 32bit.

Dot right-shift signed 32bit operators (each element of one chunk_i is shifted by the shift_vect(i) factor) with signed 32bit output

Parameters

downShift	- [Output] 32bit destination block pointer for the downshifted result
src	- [Input] 32bit source block pointer
rightShiftFact	- [Input] the vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.31 void rsh_in32u_out32u_filter (vec32u ∗ *downShift,* vec32u ∗ *src,* vec08u ∗ *rightShiftFact,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Chunk-wise unsigned 32bit right shift with an unsigned 8bit shift vector => unsigned 32bit.

Dot right-shift unsigned 32bit operators (each element of one chunk_i is shifted by the shift_vect(i) factor) with unsigned 32bit output dst_chunk[i] = srcImage0_chunk[i] >> rightShiftFact[i]

Parameters

downShift	- [Output] 32bit destination block pointer for the downshifted result
src	- [Input] 32bit source block pointer
rightShiftFact	-[Input] the vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.32 void rsh_in64s_out32s_filter (vec32s ∗ *dst*, vec32s ∗ *in_high*, vec32u ∗ *in_low*, vec08u ∗ *rightShiftFact*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 32bit.

Dot right-shift operator for 64bit signed input (each element of one chunk_i is shifted by the shift_vect(i) factor)

Warning

No check is performed to see that the right shifted values became less than 2^32

Parameters

dst	- [Output] signed 32bit destination block pointer for the downshifted result
in_high	- [Input] signed 32bit high word of source block pointer
in_low	- [Input] unsigned 32bit low word of source block pointer
rightShiftFact	- [Input] unsigned 8bit shift values, one for each block
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.33 void rsh_in64s_out64s_filter (vec32s ∗ *dst_high*, vec32u ∗ *dst_low*, vec32s ∗ *in_high*, vec32u ∗ *in_low*, vec08u ∗ *rightShiftFact*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 64bit.

Dot right-shift operator for 64bit signed input (each element of one chunk_i is shifted by the shift_vect(i) factor)

Parameters

dst_high	- [Output] 32bit signed high word of destination block pointer for the downshifted result
dst_low	- [Output] 32bit unsigned low word of destination block pointer for the downshifted result
in_high	- [Input] 32bit signed high word of source block pointer
in_low	- [Input] 32bit unsigned low word of source block pointer
rightShiftFact	- [Input] the vector of shift values, one for each chunk
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.10.2.34 void rsh_in64u_out64u_filter (vec32u ∗ *dst_high*, vec32u ∗ *dst_low*, vec32u ∗ *in_high*, vec32u ∗ *in_low*, vec08u ∗ *rightShiftFact*, int16s *bw*, int16s *bh*, int16s *inStrideWidth*, int16s *outStrideWidth*)

Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 64bit.

Dot right-shift operator for 64bit unsigned input (each element of one chunk_i is shifted by the shift_vect(i) factor)

Parameters

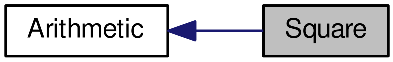
dst_high	- [Output] unsigned 32bit high word of destination block pointer for the downshifted result
dst_low	- [Output] unsigned 32bit low word of destination block pointer for the downshifted result
in_high	- [Input] unsigned 32bit high word of source block pointer
in_low	- [Input] unsigned 32bit low word of source block pointer
rightShiftFact	- [Input] unsigned 8bit shift values, one for each block
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block width (in elements not bytes) including padding
outStrideWidth	- [Input] Destination block width (in elements not bytes) including padding

5.11 Square

5.11.1 Detailed Description

Element-wise square.

Collaboration diagram for Square:



Functions

- KERNEL_INFO apu_dot_sqr_in16s_out32u (" apu_dot_sqr_in16s_out32u ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_sqr_in32s_out32u (" apu_dot_sqr_in32s_out32u ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_sqr_in32s_out64u (" apu_dot_sqr_in32s_out64u ", 3, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_High"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_Low"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- vec32u vsqrt_32 (vec32u a)
- void dot_sqr_in16s_out32u_filter (vec32u *dst, vec16s *srcA, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 16bit square => unsigned 32bit.
- void dot_sqr_in32s_out32u_filter (vec32u *dst, vec32s *srcA, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit square => unsigned 32bit.
- void dot_sqr_in32s_out64u_filter (vec32u *dst_high, vec32u *dst_low, vec32s *srcA, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit square => unsigned 64bit.
- void dot_sqr_in32u_out64u_filter (vec32u *out_high, vec32u *out_low, vec32u *srcA, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise unsigned 32bit square => unsigned 64bit.
- void dot_sqr_in64s_out64u_filter (vec32u *dst_high, vec32u *dst_low, vec32s *srcA_high, vec32u *srcA_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 64bit square => unsigned 64bit.
- void dot_sqr_in64u_out64u_filter (vec32u *dst_high, vec32u *dst_low, vec32u *srcA_high, vec32u *srcA_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise unsigned 64bit square => unsigned 64bit.

5.11.2 Function Documentation

5.11.2.1 KERNEL_INFO apu_dot_sqr_in16s_out32u (" apu_dot_sqr_in16s_out32u " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Square kernel metadata. Computespixelwise the square of a input signed 16bit images. Outputs unsigned 32bit square result

Parameters

SQR_In16s_↩ Out32u_KN	Define for Kernel name
2	Number of ports
Port SQR_KN_IN	Define for name of input image (signed 16bit)
Port SQR_KN_OUT	Define for name of square of input result (unsigned 32bit)

5.11.2.2 KERNEL_INFO apu_dot_sqr_in32s_out32u (" apu_dot_sqr_in32s_out32u " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Square kernel metadata. Computespixelwise the square of a input signed 32bit images. Outputs unsigned 32bit square result

Warning

Does not check out of range values above/below +/-2^(31-1)

Parameters

SQR_In32s_↩ Out32u_KN	Define for Kernel name
2	Number of ports
Port SQR_KN_IN	Define for name of input image (signed 32bit)
Port SQR_KN_OUT	Define for name of square of input result (unsigned 32bit)

5.11.2.3 KERNEL_INFO apu_dot_sqr_in32s_out64u (" apu_dot_sqr_in32s_out64u " , 3 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_High"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_Low"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Square kernel metadata. Computespixelwise the square of a input signed 32bit images. Outputs unsigned 64bit square result

Parameters

SQR_In32s_↩ Out32u_KN	Define for Kernel name
2	Number of ports
Port SQR_KN_IN	Define for name of input image (signed 32bit)
Port SQR_KN↩ _OUT_HIGH	Define for name of high word of square of input result (unsigned 32bit)
Port SQR_KN↩ _OUT_LOW	Define for name of low word of square of input result (unsigned 32bit)

5.11.2.4 void dot_sqr_in16s_out32u_filter (vec32u ∗ *dst,* vec16s ∗ *srcA,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 16bit square => unsigned 32bit.

Computes the 32bit elementwise-square of the 16bit input matrix. dst[i] = srcImageA[i].^2;

Warning

Out of range values are not taken into account!

Parameters

dst	- [Output] 32bit Destination block pointer
srcA	- [Input] 32bit Source block pointer of img A
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block stride width (i.e. nr. of elements including padding)
outStrideWidth	- [Input] Destination block stride width (i.e. nr. of elements including padding)

5.11.2.5 void dot_sqr_in32s_out32u_filter (vec32u ∗ *dst,* vec32s ∗ *srcA,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 32bit square => unsigned 32bit.

Computes the 32bit elementwise-square of the signed 16bit input matrix.

Warning

Out of range values are not taken into account!

Parameters

dst	- [Output] 32bit Destination block pointer
srcA	- [Input] 32bit Source block pointer of img A
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block stride width (i.e. nr. of elements including padding)
outStrideWidth	- [Input] Destination block stride width (i.e. nr. of elements including padding)

5.11.2.6 void dot_sqr_in32s_out64u_filter (vec32u ∗ *dst_high,* vec32u ∗ *dst_low,* vec32s ∗ *srcA,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 32bit square => unsigned 64bit.

Computes the 64 bit elementwise-square of the signed 32bit input matrix.

Parameters

dst_high	- [Output] unsigned 32bit high word destination block pointer
dst_low	- [Output] unsigned 32bit low word destination block pointer
srcA	- [Input] signed 32bit source block pointer of img A
bw	- [Input] Block width

bh	- [Input] Block height
inStrideWidth	- [Input] Source block stride width (i.e. nr. of elements including padding)
outStrideWidth	- [Input] Destination block stride width (i.e. nr. of elements including padding)

5.11.2.7 void dot_sqr_in32u_out64u_filter (vec32u ∗ *out_high,* vec32u ∗ *out_low,* vec32u ∗ *srcA,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise unsigned 32bit square => unsigned 64bit.

Computes the 64 bit elementwise-square of the unsigned 32bit input matrix

Parameters

dst_high	- [Output] unsigned 32bit high-word destination block pointer
dst_low	- [Output] unsigned 32bit low-word destination block pointer
srcA	- [Input] unsigned 32bit source block pointer of img A
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block stride width (i.e. nr. of elements including padding)
outStrideWidth	- [Input] Destination block stride width (i.e. nr. of elements including padding)

5.11.2.8 void dot_sqr_in64s_out64u_filter (vec32u ∗ *dst_high,* vec32u ∗ *dst_low,* vec32s ∗ *srcA_high,* vec32u ∗ *srcA_low,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise signed 64bit square => unsigned 64bit.

Computes the 64 bit elementwise-square of the signed 64bit input matrix

Warning

Out of range values are not taken into account!

Parameters

dst_high	- [Output] unsigned 32bit high-word destination block pointer
dst_low	- [Output] unsigned 32bit low-word destination block pointer
srcA_high	- [Input] Block pointer to sighed 32bit high word of source
srcA_low	- [Input] Block pointer to unsigned 32bit low word of source
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block stride width (i.e. nr. of elements including padding)
outStrideWidth	- [Input] Destination block stride width (i.e. nr. of elements including padding)

5.11.2.9 void dot_sqr_in64u_out64u_filter (vec32u ∗ *dst_high,* vec32u ∗ *dst_low,* vec32u ∗ *srcA_high,* vec32u ∗ *srcA_low,* int16s *bw,* int16s *bh,* int16s *inStrideWidth,* int16s *outStrideWidth*)

Elementwise unsigned 64bit square => unsigned 64bit.

Computes the 64 bit elementwise-square of the unsigned 64bit input matrix

Warning

Out of range values are not taken into account!

Parameters

dst_high	- [Output] unsigned 32bit high-word destination block pointer
dst_low	- [Output] unsigned 32bit low-word destination block pointer
srcA_high	- [Input] Block pointer to unsigned 32bit high word of source
srcA_low	- [Input] Block pointer to unsigned 32bit low word of source
bw	- [Input] Block width
bh	- [Input] Block height
inStrideWidth	- [Input] Source block stride width (i.e. nr. of elements including padding)
outStrideWidth	- [Input] Destination block stride width (i.e. nr. of elements including padding)

5.11.2.10 vec32u vsqrt_32 (vec32u *a*)

Computes the 32bit integer square root of the input parameter

va Source vector

Returns

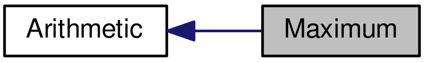
sqrt(va)

5.12 Maximum

5.12.1 Detailed Description

Element-wise maximum.

Collaboration diagram for Maximum:



Functions

- KERNEL_INFO apu_max (" apu_max ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void max (vec08u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

    Element-wise maximum.

5.12.2 Function Documentation

5.12.2.1 KERNEL_INFO apu_max (" apu_max " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Element-wise maximum kernel metadata.

Parameters

apu_max	Define for Kernel name
3	Number of ports
Port MAX_KN_INA	Define for name of first input image (unsigned 8bit)
Port MAX_KN_INB	Define for name of second input image (unsigned 8bit)
Port MAX_KN_OUT	Define for name of output image (unsigned 8bit)

5.12.2.2 void max (vec08u ∗ *dst*, vec08u ∗ *srcImage0*, vec08u ∗ *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Element-wise maximum.

Element-wise maximum. out[i] = max(in0[i], in1[i])

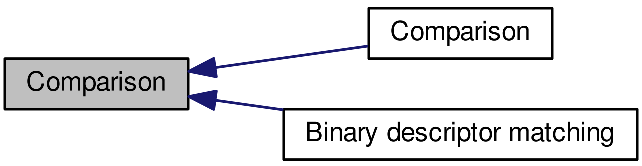
Parameters

dst	- [Output] Pointer to the destination buffer
srcImage0	- [Input] Pointer to the first source buffer
srcImage1	- [Input] Pointer to the second source buffer
bw	- [Input] Width of one data block
bh	- [Input] Height of one data block
inStrideW	- [Input] Line stride of the source data
outStrideW	- [Input] Line stride of the destination data

5.13 Comparison

5.13.1 Detailed Description

Collaboration diagram for Comparison:



Modules

- Comparison

    Element-wise comparison.
- Binary descriptor matching

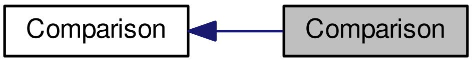
    Binary descriptor matching.

5.14 Comparison

5.14.1 Detailed Description

Element-wise comparison.

Collaboration diagram for Comparison:



Functions

- KERNEL_INFO apu_lower (" apu_lower ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_scalar (" apu_lower_scalar ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_Scalar"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_in16s (" apu_lower_in16s ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_in32s (" apu_lower_in32s ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_abs_lower_in32s (" apu_abs_lower_in32s ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_abs_lower_in32s_scalar16u (" apu_abs_lower_in32s_scalar16u ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO apu_lowerEqual_in32s (" apu_lowerEqual_in32s ", 3, __port(__index(0), __identifier("LO↩WER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __↩e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_AT↩TR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(_↩_index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_mask8b (" apu_mask8b ", 3, __port(__index(0), __identifier("MASK_IN_IMG"), __↩attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __↩ek_size(1, 1)), __port(__index(1), __identifier("MASK_IN_MASK"), __attributes(ACF_ATTR_VEC_IN), _↩_spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), ↩__identifier("MASK_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_↩type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_and (" apu_and ", 3, __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF↩_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), ↩__port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("AND_OUT"), _↩_attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __↩ek_size(1, 1)))
- KERNEL_INFO apu_and_in16u_out16u (" apu_and_in16u_out16u ", 3, __port(__index(0), __identifier("A↩ND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0↩_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VE↩C_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0↩_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_and_3Pt_in16u_out16u (" apu_and_3Pt_in16u_out16u ", 4, __port(__index(0), __↩identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR↩_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(↩__index(2), __identifier("AND_IN_2"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __↩e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("AND_OUT"), __↩attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __↩ek_size(1, 1)))
- KERNEL_INFO apu_and_in08u_out16u (" apu_and_in08u_out16u ", 3, __port(__index(0), __identifier("A↩ND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0↩_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VE↩C_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0↩_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_and_in08u_in16u_out16u (" apu_and_in08u_in16u_out16u ", 3, __port(__index(0), __↩identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_↩VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_↩data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_in64s (" apu_lower_in64s ", 5, __port(__index(0), __identifier("LOWER_KN_I↩N_0_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0↩_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_0_LOW"), __attributes(A↩CF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_IN_1_HIGH"), __attributes(ACF_ATTR_VEC_IN), __↩spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __↩identifier("LOWER_KN_IN_1_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_↩data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_in64u (" apu_lower_in64u ", 5, __port(__index(0), __identifier("LOWER_KN↩_IN_0_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __↩e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_0_LOW"), __attributes(A↩CF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1,

1)), __port(__index(2), __identifier("LOWER_KN_IN_1_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("LOWER_KN_IN_1_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

- void lower (vbool ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit "<" operation => bool.
- void lower_scalar (vec08u ∗dst, vec08u ∗srcImage, unsigned char scalar, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit "<" operation => bool.
- void lower_in16s (vbool ∗dst, vec16s ∗srcImage0, vec16s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 16bit "<" operation => bool.
- void lower_in32s (vbool ∗dst, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit "<" operation => bool.
- void absLower_in32s (vbool ∗dst, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit "<" operation between the absolute values of the operands => bool.
- void absLower_in32s_scalar16u (vbool ∗dst, vec32s ∗srcImage0, int16u compVal, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit "<" operation between an image and a fixed unsigned 16bit scalar => bool.
- void lowerEqual_in32s (vbool ∗dst, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit "<=" operation => bool.
- void lower_in64u (vbool ∗dst, vec32u ∗srcImage0_high, vec32u ∗srcImage0_low, vec32u ∗srcImage1_high, vec32u ∗srcImage1_low, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 64bit "<" operation => bool.
- void lower_in64s (vbool ∗dst, vec32s ∗srcImage0_high, vec32u ∗srcImage0_low, vec32s ∗srcImage1_high, vec32u ∗srcImage1_low, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 64bit "<" operation => bool.
- void mask_kn (vec08u ∗dst, vec08u ∗srcImage, vec08u ∗srcMask, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit mask operation => unsigned 8bit.
- void and_kn (vec08u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit "&&" operation => unsigned 8bit.
- void and_in16u_out16u (vec16u ∗dst, vec16u ∗srcImage0, vec16u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 16bit "&&" operation => unsigned 16bit.
- void and_in08u_out16u (vec16u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit "&&" operation => unsigned 16bit.
- void and_in08u_in16u_out16u (vec16u ∗dst, vec08u ∗srcImage0, vec16u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit "&&" unsigned 16bit operation => unsigned 16bit.
- void and_3Pt_in16u_out16u (vec16u ∗dst, vec16u ∗srcImage0, vec16u ∗srcImage1, vec16u ∗srcImage2, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 3-point 16bit "&&" operation => unsigned 16bit.

5.14.2 Function Documentation

5.14.2.1 void absLower_in32s (vbool * *dst*, vec32s * *srcImage0*, vec32s * *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise signed 32bit "<" operation between the absolute values of the operands => bool.

Comparison btw the absolute values of two 32bit matrices: dst[i] = (abs(srcImage0[i]) < abs(srcImage1[i]))

Parameters

dst	- [Output] boolean destination block pointer
srcImage0	- [Input] signed 32bit source block pointer of img 0
srcImage1	- [Input] signed 32bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.2 void absLower_in32s_scalar16u (vbool * *dst*, vec32s * *srcImage0*, int16u *compVal*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise signed 32bit "<" operation between an image and a fixed unsigned 16bit scalar => bool.

Comparison btw the absolute values of a signed 32bit matrix and a scalar unsigned 16bit value: dst[i] = (abs(srcImage0[i]) < val)

Parameters

dst	- [Output] boolean destination block pointer
srcImage0	- [Input] signed 32bit source block pointer of img 0
compVal	- [Input] unsigned 16bit value to compare to
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.3 void and_3Pt_in16u_out16u (vec16u * *dst*, vec16u * *srcImage0*, vec16u * *srcImage1*, vec16u * *srcImage2*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise unsigned 3-point 16bit "&&" operation => unsigned 16bit.

Elementwise AND btw three unsigned 16bit numbers: dst[i] = (srcImage0[i]!= 0 && srcImage1[i]!= 0 && srcImage2[i]!= 0). Result is converted to unsigned 16bit

Parameters

dst	- [Output] 16bit Destination block pointer
srcImage0	- [Input] 16bit source block pointer of first image
srcImage1	- [Input] 16bit source block pointer of second image
srcImage2	- [Input] 16bit source block pointer of third image
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding

outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.4 void and_in08u_in16u_out16u (vec16u ∗ *dst*, vec08u ∗ *srcImage0*, vec16u ∗ *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise unsigned 8bit "&&" unsigned 16bit operation => unsigned 16bit.

Elementwise AND btw one unsigned 8bit and one unsigned 16bit blocks: dst[i] = (srcImage0[i] && srcImage1[i]). Result is converted to unsigned 16bit

Parameters

dst	- [Output] unsigned 16bit destination block pointer
srcImage0	- [Input] unsigned 8bit source block pointer of first image
srcImage1	- [Input] unsigned 16bit source block pointer of second image
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.5 void and_in08u_out16u (vec16u ∗ *dst*, vec08u ∗ *srcImage0*, vec08u ∗ *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise unsigned 8bit "&&" operation => unsigned 16bit.

Elementwise AND btw two unsigned 16bit numbers: dst[i] = (srcImage0[i] && srcImage1[i]). Result is converted to unsigned 16bit

Parameters

dst	- [Output] unsigned 16bit destination block pointer
srcImage0	- [Input] signed 8bit source block pointer of first image
srcImage1	- [Input] signed 8bit source block pointer of second image
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.6 void and_in16u_out16u (vec16u ∗ *dst*, vec16u ∗ *srcImage0*, vec16u ∗ *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise unsigned 16bit "&&" operation => unsigned 16bit.

Elementwise AND btw two unsigned 16bit numbers: dst[i] = (srcImage0[i] && srcImage1[i])

Parameters

dst	- [Output] unsigned 16bit destination block pointer
srcImage0	- [Input] unsigned 16bit source block pointer of first image
srcImage1	- [Input] unsigned 16bit source block pointer of second image
bw	- [Input] Block width
bh	- [Input] Block height

instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.7 void and_kn (vec08u ∗ *dst,* vec08u ∗ *srcImage0,* vec08u ∗ *srcImage1,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise unsigned 8bit "&&" operation => unsigned 8bit.

Elementwise AND btw two unsigned 8bit numbers: dst[i] = (srcImage0[i] && srcImage1[i])

Parameters

dst	- [Output] unsigned 8bit destination block pointer
srcImage0	- [Input] unsigned 8bit source block pointer of first image
srcImage1	- [Input] unsigned 8bit source block pointer of second image
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.8 KERNEL_INFO apu_abs_lower_in32s (" apu_abs_lower_in32s " , 3 , __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Absolute Lower kernel metadata. Compares pixelwise the absolute values of two signed 32bit images. Outputs unsigned 8bit comparison result. Is true if abs(INPUTA) <= abs(INPUTB)

Parameters

ABS_LOWER←↩ _In32s_KN	Define for Kernel name
3	Number of ports
Port LOWER_←↩ KN_INA	Define for name of first input image (signed 32bit)
Port LOWER_←↩ KN_INB	Define for name of second input image (signed 32bit)
Port LOWER_←↩ KN_OUT	Define for name ofcomparison result of the two images (unsigned 8bit).

5.14.2.9 KERNEL_INFO apu_abs_lower_in32s_scalar16u (" apu_abs_lower_in32s_scalar16u " , 3 , __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Absolute Lower kernel metadata. Compares pixelwise the absolute values of signed 32bit image with a single scalar unsigned 16bit value. Outputs unsigned 8bit comparison result. Is true if abs(INPUTA) <= scalar

Parameters

ABS_LOWER↩ _In32s_↩ scalar16u_KN	Define for Kernel name
3	Number of ports
Port LOWER_↩ KN_INA	Define for name of first input image (signed 32bit)
Port LOWER_↩ KN_INB	Define for name of scalar value (unsigned 16bit)
Port LOWER_↩ KN_OUT	Define for name of comparison result of the two images (unsigned 8bit).

5.14.2.10 KERNEL_INFO apu_and (" apu_and " , 3 , __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

AND operator kernel metadata. Pixelwise "AND" operator between two unsigned 8bit images. Outputs unsigned 8bit comparison result. Is true if (INPUTA != 0) && (INPUTB != 0)

Parameters

AND_K	Define for Kernel name
3	Number of ports
Port AND_KN_INA	Define for name of first input image (unsigned 16bit)
Port AND_KN_INB	Define for name of second input image (unsigned 16bit)
Port AND_KN_OUT	Define for name of comparison result of the two images (unsigned 16bit).

5.14.2.11 KERNEL_INFO apu_and_3Pt_in16u_out16u (" apu_and_3Pt_in16u_out16u " , 4 , __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("AND_IN_2"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

AND operator kernel metadata. Pixelwise "AND" operator between three unsigned 16bit images. Outputs unsigned 16bit comparison result. Is true if (INPUTA != 0) && (INPUTB != 0) && (INPUTC != 0)

Parameters

AND_3Pt_↩ In16u_Out16u↩ _K	Define for Kernel name
3	Number of ports
Port AND_KN_INA	Define for name of first input image (unsigned 16bit)
Port AND_KN_INB	Define for name of second input image (unsigned 16bit)

Port AND_KN_INC	Define for name of third input image (unsigned 16bit)
Port AND_KN_OUT	Define for name of comparison result of the two images (unsigned 16bit).

5.14.2.12 KERNEL_INFO apu_and_in08u_in16u_out16u (" apu_and_in08u_in16u_out16u " , 3 , __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

AND operator kernel metadata. Pixelwise "AND" operator between two unsigned 16bit images. Outputs unsigned 16bit and operator result. Is true if (INPUTA != 0) && (INPUTB != 0)

Parameters

AND_In08u_In16u_Out16u_KN	Define for Kernel name
3	Number of ports
Port AND_KN_INA	Define for name of first input image (unsigned 8bit)
Port AND_KN_INB	Define for name of second input image (unsigned 8bit)
Port AND_KN_OUT	Define for name of comparison result of the two images (unsigned 16bit).

5.14.2.13 KERNEL_INFO apu_and_in08u_out16u (" apu_and_in08u_out16u " , 3 , __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

AND operator kernel metadata. Pixelwise "AND" operator between two unsigned 8bit images. Outputs unsigned 16bit and operator result. Is true if (INPUTA != 0) && (INPUTB != 0)

Parameters

AND_In08u_Out16u_KN	Define for Kernel name
3	Number of ports
Port AND_KN_INA	Define for name of first input image (unsigned 8bit)
Port AND_KN_INB	Define for name of second input image (unsigned 8bit)
Port AND_KN_OUT	Define for name of comparison result of the two images (unsigned 16bit).

5.14.2.14 KERNEL_INFO apu_and_in16u_out16u (" apu_and_in16u_out16u " , 3 , __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

AND operator kernel metadata. Pixelwise "AND" operator between two unsigned 16bit images. Outputs unsigned 16bit comparison result. Is true if (INPUTA != 0) && (INPUTB != 0)

Parameters

AND_In16u_↩ Out16u_K	Define for Kernel name
3	Number of ports
Port AND_KN_INA	Define for name of first input image (unsigned 16bit)
Port AND_KN_INB	Define for name of second input image (unsigned 16bit)
Port AND_KN_OUT	Define for name of comparison result of the two images (unsigned 16bit).

5.14.2.15 KERNEL_INFO apu_lower (" apu_lower " , 3 , __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Lower kernel metadata. Compares pixelwise two unsigned 8bit images. Outputs unsigned 8bit comparison result. Is true if INPUTA $<$ INPUTB

Parameters

LOWER_KN	Define for Kernel name
3	Number of ports
Port LOWER_↩ KN_INA	Define for name of first input image (unsigned 8bit)
Port LOWER_↩ KN_INB	Define for name of second input image (unsigned 8bit)
Port LOWER_↩ KN_OUT	Define for name of comparison result of the two images (unsigned 8bit).

5.14.2.16 KERNEL_INFO apu_lower_in16s (" apu_lower_in16s " , 3 , __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Lower kernel metadata. Compares pixelwise two signed 16bit images. Outputs unsigned 8bit comparison result. Is true if INPUTA $<$ INPUTB

Parameters

LOWER_In16s↩ _KN	Define for Kernel name
3	Number of ports
Port LOWER_↩ KN_INA	Define for name of first input image (signed 16bit)
Port LOWER_↩ KN_INB	Define for name of second input image (signed 16bit)
Port LOWER_↩ KN_OUT	Define for name of comparison result of the two images (unsigned 8bit).

5.14.2.17 KERNEL_INFO apu_lower_in32s (" apu_lower_in32s " , 3 , __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Lower kernel metadata. Compares pixelwise two signed 32bit images. Outputs unsigned 8bit comparison result. Is true if INPUTA $<$ INPUTB

Parameters

LOWER_In32s←↩ _KN	Define for Kernel name
3	Number of ports
Port LOWER_←↩ KN_INA	Define for name of first input image (signed 32bit)
Port LOWER_←↩ KN_INB	Define for name of second input image (signed 32bit)
Port LOWER_←↩ KN_OUT	Define for name of comparison result of the two images. (unsigned 8bit)

5.14.2.18 KERNEL_INFO apu_lower_in64s (" apu_lower_in64s " , 5 , __port(__index(0), __identifier("LOWER_KN_IN_0_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_IN_0_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_IN_1_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("LOWER_KN_IN_1_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Lower kernel metadata. Compares pixelwise the values of two signed 64bit images. Outputs unsigned 8bit comparison result. Is true if INPUTA <= INPUTB

Parameters

LOWER_In64s←↩ _KN	Define for Kernel name
3	Number of ports
Port LOWER_←↩ KN_INA_HIGH	Define for name of signed 32bit high word of first signed 64bit input image
Port LOWER_←↩ KN_INA_LOW	Define for name of unsigned 32bit low word of first signed 64bit input image
Port LOWER_←↩ KN_INB_HIGH	Define for name of signed 32bit high word of second signed 64bit input image
Port LOWER_←↩ KN_INB_LOW	Define for name of unsigned 32bit low word of second signed 64bit input image
Port LOWER_←↩ KN_OUT	Define for name of unsigned 8bit comparison result of the two images.

5.14.2.19 KERNEL_INFO apu_lower_in64u (" apu_lower_in64u " , 5 , __port(__index(0), __identifier("LOWER_KN_IN_0_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_IN_0_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_IN_1_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("LOWER_KN_IN_1_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Lower kernel metadata. Compares pixelwise the values of two unsigned 64bit images. Outputs unsigned 8bit comparison result. Is true if INPUTA <= INPUTB

Parameters

LOWER_↩ In64u_KN	Define for Kernel name
3	Number of ports
Port LOWER_↩ KN_INA_HIGH	Define for name of unsigned 32bit high word of first unsigned signed 64bit input image
Port LOWER_↩ KN_INA_LOW	Define for name of unsigned 32bit low word of first unsigned signed 64bit input image
Port LOWER_↩ KN_INB_HIGH	Define for name of unsigned 32bit high word of second unsigned 64bit input image
Port LOWER_↩ KN_INB_LOW	Define for name of unsigned 32bit low word of second unsigned 64bit input image
Port LOWER_↩ KN_OUT	Define for name of unsigned 8bit comparison result of the two images.

5.14.2.20 KERNEL_INFO apu_lower_scalar (" apu_lower_scalar " , 3 , __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_Scalar"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Lower kernel metadata. Compares pixelwise an unsigned 8bit image with a scalar value. Outputs unsigned 8bit comparison result. Is true if INPUTA $<$ INPUTB

Parameters

LOWER_SCA↩ LAR_KN	Define for Kernel name
3	Number of ports
Port LOWER_↩ KN_INA	Define for name of first input image (unsigned 8bit)
Port LOWER_↩ KN_SCALAR	Define for name of scalar value(unsigned 8bit)
Port LOWER_↩ KN_OUT	Define for name of comparison result of the two images (unsigned 8bit).

5.14.2.21 KERNEL_INFO apu_lowerEqual_in32s (" apu_lowerEqual_in32s " , 3 , __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Lower or equal kernel metadata. Compares pixelwise the values of two signed 32bit images. Outputs unsigned 8bit comparison result. Is true if INPUTA $<=$ INPUTB

Parameters

LOWER_EQU↩ AL_In32s_KN	Define for Kernel name
3	Number of ports
Port LOWER_↩ KN_INA	Define for name of first input image (signed 32bit)

Port LOWER_↩ KN_INB	Define for name of second input image (signed 32bit)
Port LOWER_↩ KN_OUT	Define for name of comparison result of the two images (unsigned 8bit).

5.14.2.22 KERNEL_INFO apu_mask8b (" apu_mask8b " , 3 , __port(__index(0), __identifier("MASK_IN_IMG"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MASK_IN_MASK"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("MASK_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

MASK operator kernel metadata. Returns the values of the pixels where mask is not zero. Outputs unsigned 8bit mask result. Is IN_IMG if (MASK != 0) otherwise 0

Parameters

MASK_K	Define for Kernel name
3	Number of ports
Port MASK_K↩ N_IN_IMG	Define for name of first input image (unsigned 16bit)
Port MASK_K↩ N_IN_MASK	Define for name of second input image (unsigned 16bit)
Port MASK_K↩ N_OUT	Define for name of comparison result of the two images (unsigned 16bit).

5.14.2.23 void lower (vbool ∗ *dst*, vec08u ∗ *srcImage0*, vec08u ∗ *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise unsigned 8bit "<" operation => bool.

Comparison btw two 8bit matrices: dst[i] = (srcImage0[i] < srcImage1[i])

Parameters

dst	- [Output] boolean destination block pointer
srcImage0	- [Input] unsigned 8bit source block pointer of img 0
srcImage1	- [Input] unsigned 8bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.24 void lower_in16s (vbool ∗ *dst*, vec16s ∗ *srcImage0*, vec16s ∗ *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise signed 16bit "<" operation => bool.

Comparison btw two 16bit matrices: dst[i] = (srcImage0[i] < srcImage1[i])

Parameters

dst	- [Output] boolean destination block pointer
srcImage0	- [Input] signed 16bit source block pointer of img 0
srcImage1	- [Input] signed 16bit source block pointer of img 1

bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.25 void lower_in32s (vbool ∗ *dst,* vec32s ∗ *srcImage0,* vec32s ∗ *srcImage1,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise signed 32bit "<" operation => bool.

Comparison btw two 32bit matrices: dst[i] = (srcImage0[i] < srcImage1[i])

Parameters

dst	- [Output] boolean destination block pointer
srcImage0	- [Input] signed 32bit source block pointer of img 0
srcImage1	- [Input] signed 32bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.26 void lower_in64s (vbool ∗ *dst,* vec32s ∗ *srcImage0_high,* vec32u ∗ *srcImage0_low,* vec32s ∗ *srcImage1_high,* vec32u ∗ *srcImage1_low,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise signed 64bit "<" operation => bool.

Comparison btw two signed 64bit matrices: dst[i] = (srcImage0[i] < srcImage1[i])

Parameters

dst	- [Output] Boolean Destination block pointer
srcImage0_high	- [Input] signed 32bit high word source block pointer of img 0
srcImage0_low	- [Input] unsigned 32bit low word source block pointer of img 0
srcImage1_high	- [Input] signed 32bit high word source block pointer of img 1
srcImage1_low	- [Input] unsigned 32bit low word source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.27 void lower_in64u (vbool ∗ *dst,* vec32u ∗ *srcImage0_high,* vec32u ∗ *srcImage0_low,* vec32u ∗ *srcImage1_high,* vec32u ∗ *srcImage1_low,* int *bw,* int *bh,* int *inStrideW,* int *outStrideW*)

Elementwise unsigned 64bit "<" operation => bool.

Comparison btw two unsigned 64bit matrices: dst[i] = (srcImage0[i] < srcImage1[i])

Parameters

dst	- [Output] Boolean destination block pointer
srcImage0_high	- [Input] unsigned 32bit high word of source block pointer of img 0
srcImage0_low	- [Input] unsigned 32bit low word of source block pointer of img 0

srcImage1_high	- [Input] unsigned 32bit high word of source block pointer of img 1
srcImage1_low	- [Input] unsigned 32bit low word of source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.28 void lower_scalar (vec08u ∗ *dst*, vec08u ∗ *srcImage*, unsigned char *scalar*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise unsigned 8bit "<" operation => bool.

Comparison of an 8bit image with a scalar value: dst[i] = (srcImage0[i] < scalar)

Parameters

dst	- [Output] boolean destination block pointer
srcImage	- [Input] unsigned 8bit source block pointer of img 0
scalar	- [Input] unsigned 8bit source block pointer of img 1
bw	- [Input] Block width
bh	- [Input] Block height
inStrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.29 void lowerEqual_in32s (vbool ∗ *dst*, vec32s ∗ *srcImage0*, vec32s ∗ *srcImage1*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise signed 32bit "<=" operation => bool.

Comparison btw two 32bit numbers: dst[i] = (srcImage0[i] <= srcImage1[i])

Parameters

dst	- [Output] boolean destination block pointer
srcImage0	- [Input] signed 32bit source block pointer of first image
srcImage1	- [Input] signed 32bit source block pointer of second image
bw	- [Input] Block width
bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.14.2.30 void mask_kn (vec08u ∗ *dst*, vec08u ∗ *srcImage*, vec08u ∗ *srcMask*, int *bw*, int *bh*, int *inStrideW*, int *outStrideW*)

Elementwise unsigned 8bit mask operation => unsigned 8bit.

Elementwise MASK and image : dst[i] = (srcImage[i] if (srcMask[i]!= 0) otherwise 0

Parameters

dst	- [Output] unsigned 8bit destination block pointer
srcImage	- [Input] unsigned 8bit source block pointer of image
srcMask	- [Input] unsigned 8bit source block pointer of mask
bw	- [Input] Block width

bh	- [Input] Block height
instrideW	- [Input] Source block width (in elements not bytes) including padding
outStrideW	- [Input] Destination block width (in elements not bytes) including padding

5.15 Binary descriptor matching

5.15.1 Detailed Description

Binary descriptor matching.

Collaboration diagram for Binary descriptor matching:

Functions

- KERNEL_INFO apu_match_descriptors (" apu_match_descriptors ", 5, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_CONFIG"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(512, 1)), __port(__index(4), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(512, 1)))
- void Match (const vec08u ∗apcDescriptors0, unsigned int aDescriptor0Count, const vec08u ∗apcDescriptors1, unsigned int aDescriptor1Count, int16s ∗apMatches0, int16s ∗apMatches1, int08u aThreshold, int08u aRangeCheck)

  Matches binary descriptors.

5.15.2 Function Documentation

5.15.2.1 KERNEL_INFO apu_match_descriptors (" apu_match_descriptors " , 5 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("INPUT_CONFIG"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(3), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(512, 1)) , __port(__index(4), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(512, 1)))

Descriptor matching kernel metadata.

Parameters

MATCH_DESCRIPTORS_KN	Define for Kernel name

5	Number of ports
Port MATCH_↩ DESCR_KN_IN0	Define for name of first input descriptors array (unsigned 8bit)
Port MATCH_↩ DESCR_KN_IN1	Define for name of second input descriptors array (unsigned 8bit)
Port MATCH_DES↩ CR_KN_CFG	Define for name of configuration data: number of descriptors in IN0 (16-bit) number of descriptors in IN1 (16-bit) matching threshold (max Hamming distance) (8-bit) range check (min Hamming distance between the closest and the second closest descriptors found) (8-bit)
Port MATCH_DES↩ CR_KN_OUT0	Define for name of first elements of match pairs array (unsigned 16bit)
Port MATCH_DES↩ CR_KN_OUT1	Define for name of second elements of match pairs array (unsigned 16bit)

5.15.2.2 void Match (const vec08u ∗ *apcDescriptors0,* unsigned int *aDescriptor0Count,* const vec08u ∗ *apcDescriptors1,* unsigned int *aDescriptor1Count,* int16s ∗ *apMatches0,* int16s ∗ *apMatches1,* int08u *aThreshold,* int08u *aRangeCheck*)

Matches binary descriptors.

Matches binary descriptors from group A to binary descriptors from group B. Matches with hamming distance greater than provided threshold are rejected. Ambiguous matches (i.e. the next best match's hamming distance is lesser than the provided check range) are rejected.

distance(desc0, desc1) = popcount(desc0 xor desc1)

After a descriptor desc0 (from group A) has been matched with descriptor desc1 (group B), both desc0 and desc1 are removed from further processing.
dist(desc0, desc1) has to be <= threshold, otherwise the descriptors won't be matched.
If dist(desc0, desc1A) is the smallest distance for desc0 and dist(desc0, desc1B) is the second smallest distance for desc0, the difference of these distances has to be > range check, otherwise desc0 won't be matched to anything

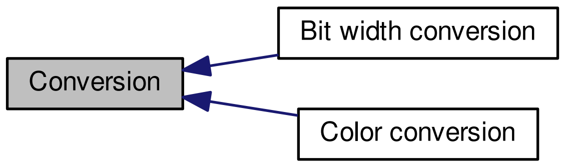
Parameters

apcDescriptors0	- [Input] First input descriptor array
aDescriptor0↩ Count	- [Input] First input descriptor array size
apcDescriptors1	- [Input] Second input descriptor array
aDescriptor1↩ Count	- [Input] Second input descriptor array size
apMatches0	- [Output] First elements of match pairs
apMatches1	- [Output] Second elements of match pairs
aThreshold	- [Input] Matching threshold (max Hamming distance)
aRangeCheck	- [Input] Range check (min Hamming distance between the closest and the second closest descriptors found)

5.16 Conversion

5.16.1 Detailed Description

Collaboration diagram for Conversion:



Modules

- Bit width conversion

    Bit width conversion.
- Color conversion

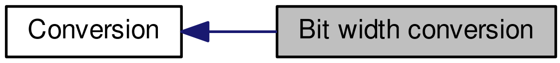
    Color conversion.

5.17 Bit width conversion

5.17.1 Detailed Description

Bit width conversion.

image 16low_to_8 implementation for APEXCollaboration diagram for Bit width conversion:



Functions

- KERNEL_INFO apu_16low_to_8 (" apu_16low_to_8 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("APU_16LOWTO8_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void f16low_to_8 (vec08u ∗dst, vec16u ∗src, int bw, int bh)

  Extracts the lower bytes.

5.17.2 Function Documentation

5.17.2.1 KERNEL_INFO apu_16low_to_8 (" apu_16low_to_8 " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("APU_16LOWTO8_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Low part of 16-bit image extraction kernel metadata.

Parameters

_16_LOW_TO_8_KN	Define for Kernel name
2	Number of ports
Port_16_LOW_TO_8_KN_IN	Define for name of input image (unsigned 16bit)
Port_16_LOW_TO_8_KN_OUT	Define for name of output image (unsigned 8bit)

5.17.2.2 void f16low_to_8 (vec08u ∗ *dst*, vec16u ∗ *src*, int *bw*, int *bh*)

Extracts the lower bytes.

Extracts lower parts of the 16-bit image pixels into 8-bit image.

Parameters

dst	- [Output] Pointer to the destination buffer
src	- [Input] Pointer to the source buffer
bw	- [Input] Width of one data block
bh	- [Input] Height of one data block

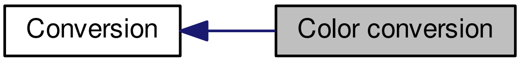
5.18 Color conversion

5.18.1 Detailed Description

Color conversion.

RGB to HSV transformation implementation for APEX.

RGB to grayscale transformation implementation for APEX.Collaboration diagram for Color conversion:



Functions

- KERNEL_INFO apu_rgb_to_grayscale (" apu_rgb_to_grayscale ", 2, __port(__index(0), __identifier("INP↩ UT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), ↩ __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void rgb_to_grayscale (vec08u ∗apDest, const vec08u ∗apcSrc, int aBlockWidth, int aBlockHeight, int a↩ OutputSpan, int aInputSpan)

  Transforms RGB to grayscale.
- KERNEL_INFO apu_rgb_to_hsv_sat (" apu_rgb_to_hsv_sat ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek↩ _size(1, 1)), __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_↩ dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_rgb_to_hsv_hue_sat (" apu_rgb_to_hsv_hue_sat ", 3, __port(__index(0), __↩ identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATT↩ R_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __↩ port(__index(2), __identifier("OUT_HUE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_rgb_to_hsv_hue_sat_grey (" apu_rgb_to_hsv_hue_sat_grey ", 4, __port(__index(0), _↩ _identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_↩ VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(↩ __index(2), __identifier("OUT_HUE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __↩ e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUT_GREY"), __↩ attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __↩ ek_size(1, 1)))
- KERNEL_INFO apu_rgb_to_hsv_svr (" apu_rgb_to_hsv_svr ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek↩ _size(1, 1)), __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_↩ dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("O↩ UT_VAL"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUT_RED"), __attributes(ACF_ATTR_VEC_O↩ UT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void rgb_to_hsv_sat (vec08u ∗apSat, const vec08u ∗apcSrc, int aBlockWidth, int aBlockHeight, int aOutput↩ Span, int aInputSpan)

  Transforms RGB to HSV => S.

- void rgb_to_hsv_hue_sat (vec16u ∗apHue, vec08u ∗apSat, const vec08u ∗apcSrc, int aBlockWidth, int a↩BlockHeight, int aOutputSpan, int aInputSpan)

    Transforms RGB to HSV => (Hue,Sat)

- void rgb_to_hsv_hue_sat_grey (vec16u ∗apHue, vec08u ∗apSat, vec08u ∗grey, const vec08u ∗apcSrc, int aBlockWidth, int aBlockHeight, int aOutputSpan, int aInputSpan)

    Transforms RGB to HSV,Grey => (Hue,Sat, Grey)

- void rgb_to_hsv_svr (vec08u ∗apSat, vec08u ∗apVal, vec08u ∗apRed, const vec08u ∗apcSrc, int aBlock↩Width, int aBlockHeight, int aOutputSpan, int aInputSpan)

    Transforms RGB to HSV => (S,V,Red)

5.18.2 Function Documentation

5.18.2.1 KERNEL_INFO apu_rgb_to_grayscale (" apu_rgb_to_grayscale " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

RGB to grayscale conversion kernel metadata.

Parameters

RGB_TO_GR↩AY_KN_IN	Define for Kernel name
2	Number of ports
Port RGB_TO↩_GRAY_KN_IN	Define for name of input RGB image (unsigned 3x8bit)
Port RGB_TO_GR↩AY_KN_OUT	Define for name of output grayscale image (unsigned 8bit)

5.18.2.2 KERNEL_INFO apu_rgb_to_hsv_hue_sat (" apu_rgb_to_hsv_hue_sat " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUT_HUE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

RGB to Hue and Saturation (of HSV) conversion kernel metadata.

Parameters

RGB_HSV_KN	Define for Kernel name
3	Number of ports
Port RGB_HS↩V_KN_IN	Define for name of input RGB image (unsigned 3x8bit)
Port RGB_HS↩V_KN_SAT	Define for name of output SATURATION image (unsigned 8bit)
Port RGB_HS↩V_KN_HUE	Define for name of output HUE image (unsigned 16bit)

5.18.2.3 KERNEL_INFO apu_rgb_to_hsv_hue_sat_grey (" apu_rgb_to_hsv_hue_sat_grey " , 4 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUT_HUE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("OUT_GREY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

RGB to Hue, Saturation (of HSV) and to Grey conversion kernel metadata.

Parameters

RGB_HSV_H↩ S_GREY_KN	Define for Kernel name
4	Number of ports
Port RGB_HS↩ V_KN_IN	Define for name of input RGB image (unsigned 3x8bit)
Port RGB_HS↩ V_KN_SAT	Define for name of output SATURATION image (unsigned 8bit)
Port RGB_HS↩ V_KN_HUE	Define for name of output HUE image (unsigned 16bit)
Port RGB_HS↩ V_KN_GREY	Define for name of output Grey image (unsigned 8bit)

5.18.2.4 KERNEL_INFO apu_rgb_to_hsv_sat (" apu_rgb_to_hsv_sat " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

RGB to HSV conversion kernel metadata.

Parameters

RGB_HSV_KN	Define for Kernel name
2	Number of ports
Port RGB_HS↩ V_KN_IN	Define for name of input RGB image (unsigned 3x8bit)
Port RGB_HS↩ V_KN_SAT	Define for name of output SATURATION image (unsigned 8bit)

5.18.2.5 KERNEL_INFO apu_rgb_to_hsv_svr (" apu_rgb_to_hsv_svr " , 4 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUT_VAL"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("OUT_RED"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

RGB to Saturation, Value, Red (of HSV) conversion kernel metadata.

Parameters

RGB_HSV_SV↩ R_KN	Define for Kernel name
4	Number of ports
Port RGB_HS↩ V_KN_IN	Define for name of input RGB image (unsigned 3x8bit)
Port RGB_HS↩ V_KN_SAT	Define for name of output SATURATION image (unsigned 8bit)
Port RGB_HS↩ V_KN_VAL	Define for name of output VALUE image (unsigned 8bit)
Port RGB_HS↩ V_KN_RED	Define for name of output Red channel of the input image (unsigned 8bit)

5.18.2.6 void rgb_to_grayscale (vec08u ∗ *apDest,* const vec08u ∗ *apcSrc,* int *aBlockWidth,* int *aBlockHeight,* int *aOutputSpan,* int *aInputSpan*)

Transforms RGB to grayscale.

Transforms RGB images to grayscale images.

Parameters

apDest	- [Output] Pointer to the destination buffer
apcSrc	- [Input] Pointer to the source buffer
aBlockWidth	- [Input] Width of one data block
aBlockHeight	- [Input] Height of one data block
aOutputSpan	- [Input] Span of the destination data
aInputSpan	- [Input] Span of the source data

5.18.2.7 void rgb_to_hsv_hue_sat (vec16u ∗ *apHue,* vec08u ∗ *apSat,* const vec08u ∗ *apcSrc,* int *aBlockWidth,* int *aBlockHeight,* int *aOutputSpan,* int *aInputSpan*)

Transforms RGB to HSV => (Hue,Sat)

Transforms RGB images to HSV images. Returns the saturation(=(max(R,G,B)-min(R,G,B))/max(R,G,B)), the Hue + 90 deg(such that red lies at 90 degrees)

Parameters

apHue	- [Output] Pointer to the returned hue buffer (values are shifted by 90 degrees)
apSat	- [Output] Pointer to the returned saturation buffer
apcSrc	- [Input] Pointer to the source buffer (in rgb format, where rgb pixels come one after each other)
aBlockWidth	- [Input] Width of one data block
aBlockHeight	- [Input] Height of one data block
aOutputSpan	- [Input] Span of the destination data
aInputSpan	- [Input] Span of the source data

5.18.2.8 void rgb_to_hsv_hue_sat_grey (vec16u ∗ *apHue,* vec08u ∗ *apSat,* vec08u ∗ *grey,* const vec08u ∗ *apcSrc,* int *aBlockWidth,* int *aBlockHeight,* int *aOutputSpan,* int *aInputSpan*)

Transforms RGB to HSV,Grey => (Hue,Sat, Grey)

Transforms RGB images to an HSV and a grey image. Returns the saturation(=(max(R,G,B)-min(R,G,B))/max(R,G,B)), the Hue + 90 deg(such that red lies at 90 degrees)

Parameters

apHue	- [Output] Pointer to the returned hue buffer (values are shifted by 90 degrees)
apSat	- [Output] Pointer to the returned saturation buffer
apGrey	- [Output] Pointer to the returned grey buffer
apcSrc	- [Input] Pointer to the source buffer (in rgb format, where rgb pixels come one after each other)
aBlockWidth	- [Input] Width of one data block
aBlockHeight	- [Input] Height of one data block
aOutputSpan	- [Input] Span of the destination data
aInputSpan	- [Input] Span of the source data

5.18.2.9 void rgb_to_hsv_sat (vec08u ∗ *apSat*, const vec08u ∗ *apcSrc*, int *aBlockWidth*, int *aBlockHeight*, int *aOutputSpan*, int *aInputSpan*)

Transforms RGB to HSV => S.

Transforms RGB images to HSV images. Returns the saturation (=(max(R,G,B)-min(R,G,B))/max(R,G,B))

Parameters

apSat	- [Output] Pointer to the returned saturation buffer
apcSrc	- [Input] Pointer to the source buffer (in rgb format, where rgb pixels come one after each other)
aBlockWidth	- [Input] Width of one data block
aBlockHeight	- [Input] Height of one data block
aOutputSpan	- [Input] Span of the destination data
aInputSpan	- [Input] Span of the source data

5.18.2.10 void rgb_to_hsv_svr (vec08u ∗ *apSat*, vec08u ∗ *apVal*, vec08u ∗ *apRed*, const vec08u ∗ *apcSrc*, int *aBlockWidth*, int *aBlockHeight*, int *aOutputSpan*, int *aInputSpan*)

Transforms RGB to HSV => (S,V,Red)

Transforms RGB images to HSV images. Returns the saturation (=(max(R,G,B)-min(R,G,B))/max(R,G,B)), the Value(==max(R,G,B)) and the R = (Red -(Green+Blue)/2)

Parameters

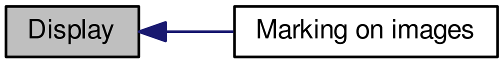
apSat	- [Output] Pointer to the returned saturation buffer
apVal	- [Output] Pointer to the returned value buffer
apRed	- [Output] Pointer to the returned red image channel buffer
apcSrc	- [Input] Pointer to the source buffer (in rgb format, where rgb pixels come one after each other)
aBlockWidth	- [Input] Width of one data block
aBlockHeight	- [Input] Height of one data block
aOutputSpan	- [Input] Span of the destination data
aInputSpan	- [Input] Span of the source data

5.19 Display

5.19.1 Detailed Description

Image marking.

Collaboration diagram for Display:



Modules

- Marking on images

  Image marking.

Functions

- KERNEL_INFO apu_mark_color_channel (" apu_mark_color_channel ", 4, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_MARKER"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_CHANNEL_INDEX"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)))

5.19.2 Function Documentation

5.19.2.1 KERNEL_INFO apu_mark_color_channel (" apu_mark_color_channel " , 4 , __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_MARKER"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("INPUT_CHANNEL_INDEX"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)))

Color channel marking kernel metadata.

Parameters

MARK_COL_KN_IN	Define for Kernel name
4	Number of ports

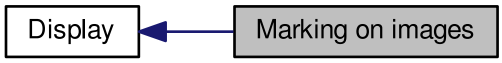
Port MARK_C↩ OL_KN_IN	Define for name of input image (unsigned 8bit)
Port MARK_COL_↩ KN_MARKER	Define for name of input marker image (unsigned 8bit)
Port MARK_COL_↩ KN_CHN_IDX	Define for name of color channel index (static unsigned 8bit)
Port MARK_C↩ OL_KN_OUT	Define for name of output image (unsigned 8bit)

5.20 Marking on images

5.20.1 Detailed Description

Image marking.

Collaboration diagram for Marking on images:



Functions

- KERNEL_INFO apu_mark (" apu_mark ", 3, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_MARKER"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void mark (vec08u ∗dst, vec08u ∗srcImage, vec08u ∗srcMarker, int bw, int bh, int sstride, int mstride)

    Marks the image.
- void mark_color_channel (vec08u ∗dst, vec08u ∗srcImage, vec08u ∗srcMarker, int bw, int bh, int08u channel, int inStride, int inMarkerStride, int outStride)

    Marks a color channel of the image.

5.20.2 Function Documentation

5.20.2.1 KERNEL_INFO apu_mark (" apu_mark " , 3 , __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_MARKER"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Image marking kernel metadata.

Parameters

MARK_KN	Define for Kernel name
3	Number of ports
Port MARK_KN_IN	Define for name of input image (unsigned 8bit)
Port MARK_KN_MARKER	Define for name of input marker image (unsigned 8bit)
Port MARK_KN_OUT	Define for name of output image (unsigned 8bit)

5.20.2.2 void mark (vec08u * *dst*, vec08u * *srcImage*, vec08u * *srcMarker*, int *bw*, int *bh*, int *sstride*, int *mstride*)

Marks the image.

Marks the image. Output pixels are copied from the source image in positions where the marker pixels are zero and from the marker image otherwise.

Parameters

dst	- [Output] Pointer to the destination buffer
srcImage	- [Input] Pointer to the source buffer
srcMarker	- [Input] Pointer to the marker buffer
bw	- [Input] Width of one data block
bh	- [Input] Height of one data block
sstride	- [Input] Input stride
mstride	- [Input] Marker stride

5.20.2.3 void mark_color_channel (vec08u * *dst*, vec08u * *srcImage*, vec08u * *srcMarker*, int *bw*, int *bh*, int08u *channel*, int *inStride*, int *inMarkerStride*, int *outStride*)

Marks a color channel of the image.

Marks the image. Output pixels are copied from the source image in positions where the marker pixels are zero and from the marker image otherwise.

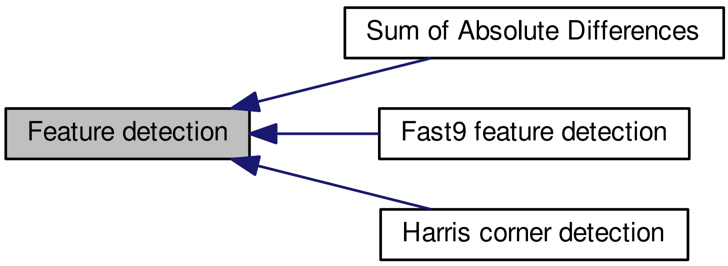
Parameters

dst	- [Output] Pointer to the destination buffer
srcImage	- [Input] Pointer to the source buffer
srcMarker	- [Input] Pointer to the marker buffer
bw	- [Input] Width of one data block
bh	- [Input] Height of one data block
channel	- [Input] Index of the color channel to mark (0, 1 or 2)
inStride	- [Input] Input stride
inMarkerStride	- [Input] Marker stride
outStride	- [Input] Output stride

5.21 Feature detection

5.21.1 Detailed Description

Collaboration diagram for Feature detection:



Modules

- Fast9 feature detection

    FAST9 feature point detection.
- Harris corner detection

    Harris feature point detection.
- Sum of Absolute Differences

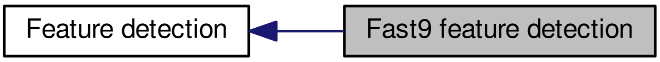
    SAD (Sum of Absolute Differences ∼= image similarity)

5.22 Fast9 feature detection

5.22.1 Detailed Description

FAST9 feature point detection.

Collaboration diagram for Fast9 feature detection:



Functions

- KERNEL_INFO apu_fast9 (" apu_fast9 ", 3, __port(__index(0), __identifier("IN_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("IN_Thr"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void apu_fast9_unsuppressed_score (const vec08u ∗apcSrc, vec08u ∗apDst, int aSourceStride, int aDestinationStride, int aBlockWidth, int aBlockHeight, int08u aThreshold)

  FAST9 corner detection.

5.22.2 Function Documentation

5.22.2.1 KERNEL_INFO apu_fast9 (" apu_fast9 " , 3 , __port(__index(0), __identifier("IN_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("IN_Thr"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

FAST9 feature point detection kernel metadata.

Parameters

FAST9_KN	Define for Kernel name
3	Number of ports
Port FAST9_KN_IN0	Define for name of input image (unsigned 8bit)
Port 'FAST9_KN_OUT	Define for name of output image (unsigned 8bit)
Port FAST9_KN_IN1	Define for name of threshold used for classifying ring pixels (unsigned 8bit) (brighter/darker/similar)

5.22.2.2 void apu_fast9_unsuppressed_score (const vec08u ∗ *apcSrc*, vec08u ∗ *apDst*, int *aSourceStride*, int *aDestinationStride*, int *aBlockWidth*, int *aBlockHeight*, int08u *aThreshold*)

FAST9 corner detection.

Finds the corners in the input data using the FAST9 algorithm. Outputs corner scores or 0 if not a corner.

For each input pixel a 16-pixel circle centered at the processed pixel is considered. The circle pixels are classified as darker, brighter or similar to the central pixel depending on the provided threshold. The central pixel is considered as a corner if and only if there is a contiguous segment of 9 pixels which are all classified as brighter or darker in the circle.

See http://www.edwardrosten.com/work/fast.html

Parameters

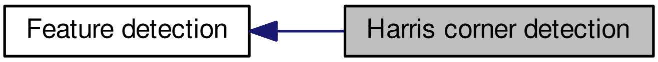
apcSrc	- [Input] pointer to the source buffer
apcDst	- [Output] pointer to the destination buffer
aSourceStride	- [Input] line stride of the source data
aDestinationStride	- [Input] line stride of the destination data
aBlockWidth	- [Input] width of one data block
aBlockHeight	- [Input] height of one data block
aThreshold	- [Input] threshold used for pixel classification

5.23 Harris corner detection

5.23.1 Detailed Description

Harris feature point detection.

Collaboration diagram for Harris corner detection:



Functions

- KERNEL_INFO apu_harris (" apu_harris ", 7, __port(__index(0), __identifier("INPUT_GX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep((4 >> 1),(4 >> 1),(4 >> 1),(4 >> 1)), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_GY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep((4 >> 1),(4 >> 1),(4 >> 1),(4 >> 1)), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_K_RBS_WINDOW"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("OUTPUT_RESPONSE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("OUTPUT_X2TEMP_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(16+2 ∗(4 >> 1), 16+2 ∗(4 >> 1))), __port(__index(5), __identifier("OUTPUT_Y2TEMP_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(16+2 ∗(4 >> 1), 16+2 ∗(4 >> 1))), __port(__index(6), __identifier("OUTPUT_XYTEMP_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(16+2 ∗(4 >> 1), 16+2 ∗(4 >> 1))))
- void apuHarrisResponse (vec16u ∗apResponse, vec16s ∗apGradX, vec16s ∗apGradY, vec16s ∗apXSqrTmp, vec16s ∗apYSqrTmp, vec16s ∗apXYTemp, int aBlockWidth, int aBlockHeight, int aStride, int k, int responseBitShift, int aWindowSize, int thresh, bool isFirstSlice)

  Harris corner detection.
- KERNEL_INFO apu_harris (" apu_harris ", 5, __port(__index(0), __identifier("INPUT_GX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_GY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_K_RBS_WINDOW"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("OUTPUT_RESPONSE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("OUTPUT_TEMP_BUFFER"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- void apuHarrisResponse (vec16u ∗apResponse, vec16s ∗apGradX, vec16s ∗apGradY, vec16s ∗apTemp, int aBlockWidth, int aBlockHeight, int aStride, int k, int responseBitShift, int aWindowSize, int thresh, bool isFirstSlice)

  Harris corner detection.

5.23.2 Function Documentation

5.23.2.1 KERNEL_INFO apu_harris (" apu_harris " , 5 , __port(__index(0), __identifier("INPUT_GX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_GY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("INPUT_K_RBS_WINDOW"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(3), __identifier("OUTPUT_RESPONSE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("OUTPUT_TEMP_BUFFER"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE, HARRIS_HALF_WINDOW_SIZE), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Harris corner detection kernel metadata.

Parameters

HARRIS_KN	Define for Kernel name
4	Number of ports
Port HARRIS_↩ KN_IN_GX	Define for name of gradient X component input image (signed 16bit)
Port HARRIS_↩ KN_IN_GY	Define for name of gradient Y component input image (signed 16bit)
Port HARRIS_KN_I↩ N_K_RBS_WIN	Define for name of detector sensitivity, response bit shift and window size (unsigned 16bit)
Port HARRIS_↩ KN_OUT_RESP	Define for name of output image (unsigned 16bit)
Port HARRIS_↩ KN_OUT_TEM↩ P_BUF	Define for name of temporary buffer (signed 16bit)

5.23.2.2 KERNEL_INFO apu_harris (" apu_harris " , 7 , __port(__index(0), __identifier("INPUT_GX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep((4 >> 1),(4 >> 1),(4 >> 1),(4 >> 1)), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_GY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep((4 >> 1),(4 >> 1),(4 >> 1),(4 >> 1)), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("INPUT_K_RBS_WINDOW"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(3), __identifier("OUTPUT_RESPONSE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("OUTPUT_X2TEMP_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(16+2 *(4 >> 1), 16+2 *(4 >> 1))) , __port(__index(5), __identifier("OUTPUT_Y2TEMP_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(16+2 *(4 >> 1), 16+2 *(4 >> 1))) , __port(__index(6), __identifier("OUTPUT_XYTEMP_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(16+2 *(4 >> 1), 16+2 *(4 >> 1))))

Harris corner detection kernel metadata.

Parameters

HARRIS_KN	Define for Kernel name

7	Number of ports
Port HARRIS_KN_IN_GX	Define for name of gradient X component input image (signed 16bit)
Port HARRIS_KN_IN_GY	Define for name of gradient Y component input image (signed 16bit)
Port HARRIS_KN_IN_K_RBS_WIN	Define for name of detector sensitivity, response bit shift and window size (unsigned 16bit)
Port HARRIS_KN_OUT_RESP	Define for name of output image (unsigned 16bit)
Port HARRIS_KN_OUT_X2TEMP_BUF	Define for name of temporary buffer holding gx∗gx values (signed 16bit)
Port HARRIS_KN_OUT_Y2TEMP_BUF	Define for name of temporary buffer holding gy∗gy values (signed 16bit)^M
Port HARRIS_KN_OUT_XYTEMP_BUF	Define for name of temporary buffer holding gx∗gy values (signed 16bit)^M

5.23.2.3 void apuHarrisResponse (vec16u ∗ *apResponse,* vec16s ∗ *apGradX,* vec16s ∗ *apGradY,* vec16s ∗ *apTemp,* int *aBlockWidth,* int *aBlockHeight,* int *aStride,* int *k,* int *responseBitShift,* int *aWindowSize,* int *thresh,* bool *isFirstSlice*)

Harris corner detection.

Finds the corners in the input data using the Harris algorithm. Outputs a Harris response value for each pixel.

out[i] = (det(A[i]) - k ∗ trace(A[i])^2) >> responseBitShift
where A[i] is a structure tensor for pixel i

See http://docs.opencv.org/doc/tutorials/features2d/trackingmotion/harris_detector/harris_detector.html

Parameters

apResponse	- output Harris response image
apGradX	- image gradient x component
apGradY	- image gradient y component
aBlockWidth	- CU block width
aBlockHeight	- CU block height
aStride	- horizontal gradient image stride
k	- Harris detector sensitivity
responseBitShift-	bit shift that will be applied to the output response
aWindowSize	- Harris detector window size
thresh	- threshold, below which corner values are set to zero
isFirstSlice	- if we are at first slice, initialization is performed

5.23.2.4 void apuHarrisResponse (vec16u ∗ *apResponse,* vec16s ∗ *apGradX,* vec16s ∗ *apGradY,* vec16s ∗ *apXSqrTmp,* vec16s ∗ *apYSqrTmp,* vec16s ∗ *apXYTemp,* int *aBlockWidth,* int *aBlockHeight,* int *aStride,* int *k,* int *responseBitShift,* int *aWindowSize,* int *thresh,* bool *isFirstSlice*)

Harris corner detection.

Finds the corners in the input data using the Harris algorithm. Outputs a Harris response value for each pixel.

out[i] = (det(A[i]) - k ∗ trace(A[i])^2) >> responseBitShift
where A[i] is a structure tensor for pixel i

See http://docs.opencv.org/doc/tutorials/features2d/trackingmotion/harris_detector/harris_detector.html

Parameters

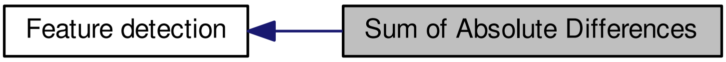
apResponse	- output Harris response image
apGradX	- image gradient x component
apGradY	- image gradient y component
apXSqrTmp	- temporary buffer to store gradX^2 values on overlapping zones
apYSqrTmp	- temporary buffer to store gradY^2 values on overlapping zones
apXYTmp	- temporary buffer to store gradX*gradY values on overlapping zones
aBlockWidth	- CU block width
aBlockHeight	- CU block height
aStride	- horizontal gradient image stride
k	- Harris detector sensitivity
responseBitShift-	bit shift that will be applied to the output response
aWindowSize	- Harris detector window size
thresh	- threshold, below which corner values are set to zero
isFirstSlice	- if we are at first slice, initialization is performed

5.24 Sum of Absolute Differences

5.24.1 Detailed Description

SAD (Sum of Absolute Differences ∼= image similarity)

Collaboration diagram for Sum of Absolute Differences:



Functions

- KERNEL_INFO apu_sad (" apu_sad ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 4)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(8, 8)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(4, 1), __ek_size(1, 1)))
- void apu_sad_impl (vec08u ∗lpvIn0, int16_t lStrideIn0, int16_t lChunkWidthIn0, int16_t lChunkHeightIn0, vec08u ∗lpvIn1, int16_t lStrideIn1, int16_t lChunkWidthIn1, int16_t lChunkHeightIn1, vec32u ∗lpvOut0, int16_t lStrideOut0, int16_t lChunkWidthOut0, int16_t lChunkHeightOut0)

    Sum of absolute differences. Store the minimum of all differences and the location of the minimum in image0.

5.24.2 Function Documentation

5.24.2.1 KERNEL_INFO apu_sad (" apu_sad " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 4)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(8, 8)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(4, 1), __ek_size(1, 1)))

SAD Sum of Absolute Differences kernel metadata.

Parameters

SAD_KN	Definition for kernel name
3	Number of ports
Port SAD_KN_IN0	Define for name of first input image (unsigned 8bit)
Port SAD_KN_IN1	Define for name of second input image (unsigned 8bit)
Port SAD_KN_OUT	Define for name of output image containing the SAD values for each pixel (unsigned 8bit)

5.24.2.2 void apu_sad_impl (vec08u ∗ *lpvIn0*, int16_t *lStrideIn0*, int16_t *lChunkWidthIn0*, int16_t *lChunkHeightIn0*, vec08u ∗ *lpvIn1*, int16_t *lStrideIn1*, int16_t *lChunkWidthIn1*, int16_t *lChunkHeightIn1*, vec32u ∗ *lpvOut0*, int16_t *lStrideOut0*, int16_t *lChunkWidthOut0*, int16_t *lChunkHeightOut0*)

Sum of absolute differences. Store the minimum of all differences and the location of the minimum in image0.

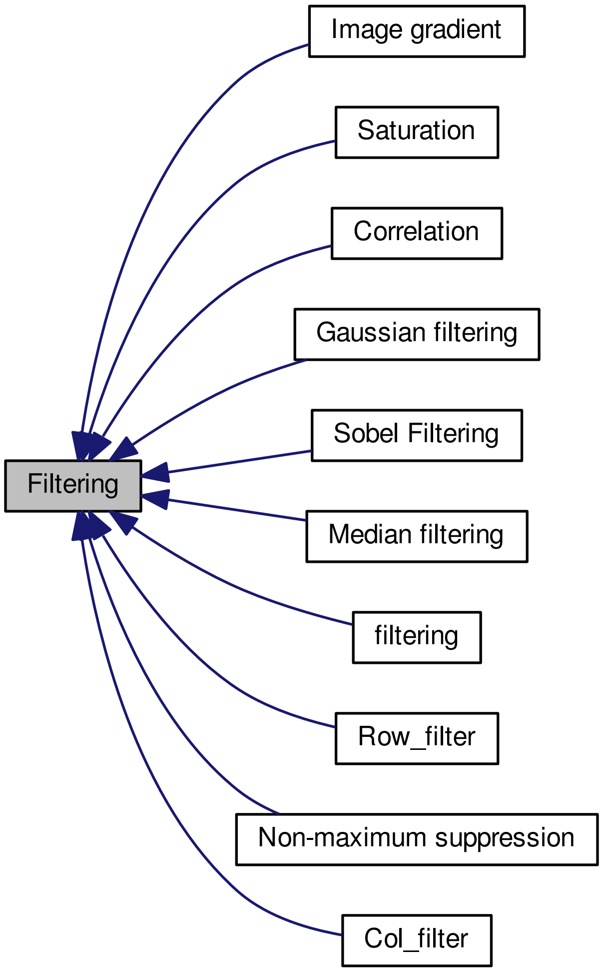
Parameters

lpvIn0	- [Input] pointer to the first source buffer
lStrideIn0	- [Input] line stride of the first source buffer
lChunkWidthIn0	- [Input] width of one data block of the first source buffer
lChunkHeightIn0	- [Input] height of one data block of the first source buffer
lpvIn1	- [Input] pointer to the second source buffer
lStrideIn1	- [Input] line stride of the second source buffer
lChunkWidthIn1	- [Input] width of one data block of the second source buffer
lChunkHeightIn1	- [Input] height of one data block of the second source buffer \
lpvOut0	- [Output] pointer to the destination buffer
lStrideOut0	- [Input] line stride of the destination data
lChunkWidth↩ Out0	- [Input] width of one data block of the destination buffer
lChunkHeight↩ Out0	- [Input] height of one data block of the destination buffer

5.25 Filtering

5.25.1 Detailed Description

Collaboration diagram for Filtering:



Modules

- Col_filter

  Filter an image row with a column_filter (i.e. Matrix ∗ Col_Vector multiplication)
- Correlation

  Correlation, respectively convolution with a reversed filter.
- filtering

  General Filtering.
- Median filtering

  Median filtering.
- Sobel Filtering

Sobel filtering.

- Gaussian filtering

  Gaussian filtering.

- Image gradient

  Image gradient.

- Non-maximum suppression

  Non-maximum suppression.

- Row_filter

  Filter an image column with a row_filter(i.e. Row_Vector $*$ Matrix multiplication)

- Saturation

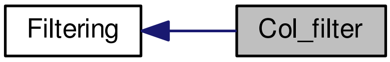
  Saturation.

5.26 Col_filter

5.26.1 Detailed Description

Filter an image row with a column_filter (i.e. Matrix ∗ Col_Vector multiplication)

Collaboration diagram for Col_filter:



Functions

- KERNEL_INFO col_filter (" col_filter ", 3, __port(__index(0), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(COL_PADDING, COL_PADDING, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("COEFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(FILTER_COLS, 1)), __port(__index(2), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void col_filter (vec08u ∗dst, int dstStride, const vec08u ∗src, int srcStride, int rows, int cols, const unsigned char ∗filter, int filterSize, int filterQ)

  Apply a column filter to an image.

Variables

- const int FILTER_COLS = 3
- const int COL_PADDING = FILTER_COLS >> 1

5.26.2 Function Documentation

5.26.2.1 KERNEL_INFO col_filter (" col_filter " , 3 , __port(__index(0), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(COL_PADDING, COL_PADDING, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("COEFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(FILTER_COLS, 1)) , __port(__index(2), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Column filter kernel metadata.

Parameters

COL_FILTER_KN	Define for Kernel name
3	Number of ports
Port COL_FILTER_KN_SRC	Define for name of input image (unsigned 8bit)

Port COL_FILTER_↩ KN_COEFFS	Define for name of filter coefficients(unsigned 8bit)
Port COL_FIL↩ TER_KN_DST	Define for name of output image (unsigned 8bit)

5.26.2.2 void col_filter (vec08u ∗ *dst,* int *dstStride,* const vec08u ∗ *src,* int *srcStride,* int *rows,* int *cols,* const unsigned char ∗ *filter,* int *filterSize,* int *filterQ*)

Apply a column filter to an image.

First we decrement the number of fractional bits in anticipation of rounding the final result.

Next, we offset the source pointer so that it points to the beginning of the padded region of the source image. The offset amount is the number of padded columns (*see* The Column Filter).

For each pixel, we accumulate the weighted sum in a 16-bit register to avoid overflow. When casting the weighted sum to an integer we round the value. This means we right bit shift the sum by 1 less the fractional bits, add 1, then right shift one more.

Parameters

dst	Pointer to the destination image.
dstStride	Stride of the destination image.
src	Pointer to the source image. The source image is assumed to be padded according to the filter size. However, `src` points the top left corner of the *unpadded* image region.
srcStride	Stride of the padded source image.
rows	Rows of the source image, not including padding.
cols	Columns of the source image, not including padding.
filter	Pointer to column filter coefficients. The coefficients are unsigned 8-bit fixed point numbers.
filterSize	Size of the column filter. The size must be odd so that the filter is centered about the pixel.
filterQ	Number of fractional bits for the filter coefficients.

5.26.3 Variable Documentation

5.26.3.1 const int COL_PADDING = FILTER_COLS >> 1

The number of padded columns that must be added to both sides of the source image.

5.26.3.2 const int FILTER_COLS = 3

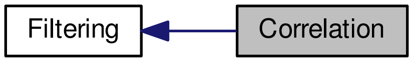
The number of columns (i.e. size) of the column filter.

5.27 Correlation

5.27.1 Detailed Description

Correlation, respectively convolution with a reversed filter.

Collaboration diagram for Correlation:



Macros

- #define **inputFiltUpScale** 3
- #define **scharrFiltUpscale** 1

Typedefs

- typedef void(∗ **corrKernelPtr**)(vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

Functions

- KERNEL_INFO apu_gradient_x (" apu_gradient_x ", 2, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_gradient_y (" apu_gradient_y ", 2, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_scharr_x (" apu_scharr_x ", 2, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_scharr_y (" apu_scharr_y ", 2, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation (" apu_correlation ", 5, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 3)), __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO apu_correlation_1x3 (" apu_correlation_1x3 ", 5, __port(__index(0), __identifier("INPU↩ T_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_V↩ EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩ index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 3)), __port(__index(3), __identifier("INPUT↩ _FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data↩ _type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __↩ attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_↩ size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation_3x1 (" apu_correlation_3x1 ", 5, __port(__index(0), __identifier("INPU↩ T_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 0, 0), __e0_data_type(d08u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_V↩ EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩ index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 1)), __port(__index(3), __identifier("INPUT↩ _FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data↩ _type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __↩ attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_↩ size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation_3x3 (" apu_correlation_3x3 ", 5, __port(__index(0), __identifier("INPU↩ T_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_V↩ EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩ index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 3)), __port(__index(3), __identifier("INPUT↩ _FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data↩ _type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __↩ attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_↩ size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation_5x5 (" apu_correlation_5x5 ", 5, __port(__index(0), __identifier("INPU↩ T_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_V↩ EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩ index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(5, 5)), __port(__index(3), __identifier("INPUT↩ _FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data↩ _type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __↩ attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_↩ size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation_7x7 (" apu_correlation_7x7 ", 5, __port(__index(0), __identifier("INPU↩ T_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_V↩ EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩ index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(7, 7)), __port(__index(3), __identifier("INPUT↩ _FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data↩ _type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __↩ attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_↩ size(1, 1), __ek_size(1, 1)))
- void initFilters ()

  Initializes the array of filter function pointers.
- void performCorrelation (int16u filterFlags, vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s scaleFact, const int16s ∗filter↩ Coefs)

  Computes time optimized the correlation with a general filter.
- void correlation_filter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

- void correlation__antisymXfilter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

  Computes the correlation with an anti-symmetric X filter.
- void correlation__symXfilter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

  Computes the correlation with a symmetric X filter.
- void correlation__symYfilter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

  Computes the correlation with a symmetric Y filter.
- void correlation__antisymYfilter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

  Computes the correlation with an anti-symmetric Y filter.
- void correlation__symXYfilter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

  Computes the correlation with a symmetric in X and Y filter.
- void correlation__symXantisymYfilter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filter↩Coefs)

  Computes the correlation with a symmetric in X and anti-symmetric in Y filter.
- void correlation__symXantisymYfilter_16s (vec16s ∗dst, vec16s ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s scaleFact, const int16s ∗filter↩Coefs)

  Computes the 16bit correlation with a symmetric in X and anti-symmetric in Y filter.
- void correlation__antisymXsymYfilter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filter↩Coefs)

  Computes the correlation with an anti-symmetric in X and symmetric in Y filter.
- void correlation__antisymXsymYfilter_16s (vec16s ∗dst, vec16s ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filter↩Coefs)

  Computes the 16bit correlation with an anti-symmetric in X and symmetric in Y filter.
- void correlation__antisymXYfilter (vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

  Computes the correlation with an anti-symmetric in X and anti-symmetric in Y filter.

5.27.2 Function Documentation

5.27.2.1 KERNEL_INFO apu_correlation (" apu_correlation " , 5 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 3)) , __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC↩_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

General correlation kernel metadata. General correlation of input unsigned 8bit image with a 1D or 2D filter. Outputs correlation result (signed 16bit).

Warning

The filter width and height (max 11x11) have to be defined in the macro definitions FW and FH for the currently used kernel!

Parameters

CORR_KN	Define for Kernel name
5	Number of ports
Port CORR_↩ Kernel_Input	Define for name of input image (unsigned 8bit). The spatial dependencies of this port - HFW and HFH - are half the filter width/height. They are defined in the header of this file as HFW FW>>1 and HFH FH >> 1
Port CORR_↩ Kernel_Output	Define for name of correlation result (signed 16bit)
Port CORR_↩ Kernel_Filter	Define for name of filter coefficients the input has to be convolved with (signed 16bit)
Port CORR_Kernel↩ _FilterScale	Define for name of the scalar value of the normalization factor used for the filter (signed 16bit)
Port CORR_Kernel↩ _FiltSymmFl	Define for name of the scalar value of the symmetry flag defined for this filter as it is specified in "symmetry_flags.h" file (unsigned 16bit)

5.27.2.2 KERNEL_INFO apu_correlation_1x3 (" apu_correlation_1x3 " , 5 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 3)) , __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC↩ _FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

General correlation kernel metadata. General correlation of input unsigned 8bit image with a filter of size 1x3 (convenience kernel for CORR_K). Outputs correlation result (signed 16bit).

Warning

The filter width and height have to be defined in the macro definitions FW and FH for the currently used kernel!

Parameters

CORR_KN_↩ 1x3_KN	Define for Kernel name
5	Number of ports
Port CORR_↩ Kernel_Input	Define for name of input image (unsigned 8bit). The spatial dependencies of this port - HFW and HFH - are half the filter width/height. They are defined in the header of this file as HFW FW>>1 and HFH FH >> 1
Port CORR_↩ Kernel_Output	Define for name of correlation result (signed 16bit)
Port CORR_↩ Kernel_Filter	Define for name of filter coefficients the input has to be convolved with (signed 16bit)
Port CORR_Kernel↩ _FilterScale	Define for name of the scalar value of the normalization factor used for the filter (signed 16bit)

Port CORR_Kernel←_FiltSymmFl	Define for name of the scalar value of the symmetry flag defined for this filter as it is specified in "symmetry_flags.h" file (unsigned 16bit)

5.27.2.3 KERNEL_INFO apu_correlation_3x1 (" apu_correlation_3x1 " , 5 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 1)) , __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC←_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

General correlation kernel metadata. General correlation of input unsigned 8bit image with a filter of size 3x1 (convenience kernel for CORR_K). Outputs correlation result (signed 16bit).

Warning

The filter width and height have to be defined in the macro definitions FW and FH for the currently used kernel!

Parameters

CORR_KN_←3x1_KN	Define for Kernel name
5	Number of ports
Port CORR_←Kernel_Input	Define for name of input image (unsigned 8bit). The spatial dependencies of this port - HFW and HFH - are half the filter width/height. They are defined in the header of this file as HFW FW$>>$1 and HFH FH $>>$ 1
Port CORR_←Kernel_Output	Define for name of correlation result (signed 16bit)
Port CORR_←Kernel_Filter	Define for name of filter coefficients the input has to be convolved with (signed 16bit)
Port CORR_Kernel←_FilterScale	Define for name of the scalar value of the normalization factor used for the filter (signed 16bit)
Port CORR_Kernel←_FiltSymmFl	Define for name of the scalar value of the symmetry flag defined for this filter as it is specified in "symmetry_flags.h" file (unsigned 16bit)

5.27.2.4 KERNEL_INFO apu_correlation_3x3 (" apu_correlation_3x3 " , 5 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 3)) , __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC←_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

General correlation kernel metadata. General correlation of input unsigned 8bit image with a filter of size 3x3 (convenience kernel for CORR_K). Outputs correlation result (signed 16bit).

Warning

The filter width and height have to be defined in the macro definitions FW and FH for the currently used kernel!

Parameters

CORR_KN_↩ 3x3_KN	Define for Kernel name
5	Number of ports
Port CORR_↩ Kernel_Input	Define for name of input image (unsigned 8bit). The spatial dependencies of this port - HFW and HFH - are half the filter width/height. They are defined in the header of this file as HFW FW>>1 and HFH FH >> 1
Port CORR_↩ Kernel_Output	Define for name of correlation result (signed 16bit)
Port CORR_↩ Kernel_Filter	Define for name of filter coefficients the input has to be convolved with (signed 16bit)
Port CORR_Kernel↩ _FilterScale	Define for name of the scalar value of the normalization factor used for the filter (signed 16bit)
Port CORR_Kernel↩ _FiltSymmFl	Define for name of the scalar value of the symmetry flag defined for this filter as it is specified in "symmetry_flags.h" file (unsigned 16bit)

5.27.2.5 KERNEL_INFO apu_correlation_5x5 (" apu_correlation_5x5 " , 5 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(5, 5)) , __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC↩ _FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

General correlation kernel metadata. General correlation of input unsigned 8bit image with a filter of size 5x5 (convenience kernel for CORR_K). Outputs correlation result (signed 16bit).

Warning

The filter width and height have to be defined in the macro definitions FW and FH for the currently used kernel!

Parameters

CORR_KN_↩ 5x5_KN	Define for Kernel name
5	Number of ports
Port CORR_↩ Kernel_Input	Define for name of input image (unsigned 8bit). The spatial dependencies of this port - HFW and HFH - are half the filter width/height. They are defined in the header of this file as HFW FW>>1 and HFH FH >> 1
Port CORR_↩ Kernel_Output	Define for name of correlation result (signed 16bit)
Port CORR_↩ Kernel_Filter	Define for name of filter coefficients the input has to be convolved with (signed 16bit)
Port CORR_Kernel↩ _FilterScale	Define for name of the scalar value of the normalization factor used for the filter (signed 16bit)

Port CORR_Kernel←↩ _FiltSymmFl	Define for name of the scalar value of the symmetry flag defined for this filter as it is specified in "symmetry_flags.h" file (unsigned 16bit)

5.27.2.6 KERNEL_INFO apu_correlation_7x7 (" apu_correlation_7x7 " , 5 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(7, 7)) , __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC←↩ _FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

General correlation kernel metadata. General correlation of input unsigned 8bit image with a filter of size 7x7 (convenience kernel for CORR_K). Outputs correlation result (signed 16bit).

Warning

The filter width and height have to be defined in the macro definitions FW and FH for the currently used kernel!

Parameters

CORR_KN_←↩ 7x7_KN	Define for Kernel name
5	Number of ports
Port CORR_←↩ Kernel_Input	Define for name of input image (unsigned 8bit). The spatial dependencies of this port - HFW and HFH - are half the filter width/height. They are defined in the header of this file as HFW FW>>1 and HFH FH >> 1
Port CORR_←↩ Kernel_Output	Define for name of correlation result (signed 16bit)
Port CORR_←↩ Kernel_Filter	Define for name of filter coefficients the input has to be convolved with (signed 16bit)
Port CORR_Kernel←↩ _FilterScale	Define for name of the scalar value of the normalization factor used for the filter (signed 16bit)
Port CORR_Kernel←↩ _FiltSymmFl	Define for name of the scalar value of the symmetry flag defined for this filter as it is specified in "symmetry_flags.h" file (unsigned 16bit)

5.27.2.7 KERNEL_INFO apu_gradient_x (" apu_gradient_x " , 2 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Gradient in X direction kernel metadata. Convolution of input unsigned 8bit image with a [-1 0 1] row-filter. Outputs correlation result (signed 16bit) (i.e. the gradients in X direction of the input image)

Parameters

CORR_GRAD←↩ _X_KN	Define for Kernel name

2	Number of ports
Port CORR_↩ Kernel_Input	Define for name of input image (unsigned 8bit)
Port CORR_↩ Kernel_Output	Define for name of correlation result (signed 16bit)

5.27.2.8 KERNEL_INFO apu_gradient_y (" apu_gradient_y " , 2 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Gradient in Y direction kernel metadata. Convolution of input unsigned 8bit image with a [-1 0 1]^T column-filter. Outputs correlation result (signed 16bit) (i.e. the gradients in Y direction of the input image)

Parameters

CORR_GRAD↩ _Y_KN	Define for Kernel name
2	Number of ports
Port CORR_↩ Kernel_Input	Define for name of input image (unsigned 8bit)
Port CORR_↩ Kernel_Output	Define for name of correlation result (signed 16bit)

5.27.2.9 KERNEL_INFO apu_scharr_x (" apu_scharr_x " , 2 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Gradient in X direction kernel metadata for Scharr_X filter. Convolution of input unsigned 8bit image with a scharr_x filter. Outputs correlation result (signed 16bit) (i.e. the gradients in X direction of the input image)

Parameters

CORR_SCHA↩ RR_X_KN	Define for Kernel name
2	Number of ports
Port CORR_↩ Kernel_Input	Define for name of input image (unsigned 8bit)
Port CORR_↩ Kernel_Output	Define for name of correlation result (signed 16bit)

5.27.2.10 KERNEL_INFO apu_scharr_y (" apu_scharr_y " , 2 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Gradient in Y direction kernel metadata for Scharr_Y filter. Convolution of input unsigned 8bit image with a scharr_y filter. Outputs correlation result (signed 16bit) (i.e. the gradients in Y direction of the input image)

Parameters

CORR_SCHA←RR_Y_KN	Define for Kernel name
2	Number of ports
Port CORR_←Kernel_Input	Define for name of input image (unsigned 8bit)
Port CORR_←Kernel_Output	Define for name of correlation result (signed 16bit)

5.27.2.11 void correlation__antisymXfilter (vec16s ∗ *dst*, vec08u ∗ *src*, int16s *sstr*, int16s *bw*, int16s *bh*, int16s *destBw*, int16s *xSkip*, int16s *ySkip*, int16u *filtWidth*, int16u *filtHeight*, int16s *filtScaling*, const int16s ∗ *filterCoefs*)

Computes the correlation with an anti-symmetric X filter.

Calculate the correlation with a filter, which is anti-symmetric in X direction.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
destBw	- [Input] Block width (including stride) of the output
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.12 void correlation__antisymXsymYfilter (vec16s ∗ *dst*, vec08u ∗ *src*, int16s *sstr*, int16s *bw*, int16s *bh*, int16s *destBw*, int16s *xSkip*, int16s *ySkip*, int16u *filtWidth*, int16u *filtHeight*, int16s *filtScaling*, const int16s ∗ *filterCoefs*)

Computes the correlation with an anti-symmetric in X and symmetric in Y filter.

Calculate the correlation with a filter, which is anti-symmetric in X and symmetric in Y directions.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.13 void correlation__antisymXsymYfilter_16s (vec16s ∗ *dst*, vec16s ∗ *src*, int16s *sstr*, int16s *bw*, int16s *bh*, int16s *destBw*, int16s *xSkip*, int16s *ySkip*, int16u *filtWidth*, int16u *filtHeight*, int16s *filtScaling*, const int16s ∗ *filterCoefs*)

Computes the 16bit correlation with an anti-symmetric in X and symmetric in Y filter.

Calculate the correlation of a signed 16bit image with a filter, which is anti-symmetric in X and symmetric in Y directions.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.14 void correlation__antisymXYfilter (vec16s ∗ *dst,* vec08u ∗ *src,* int16s *sstr,* int16s *bw,* int16s *bh,* int16s *destBw,* int16s *xSkip,* int16s *ySkip,* int16u *filtWidth,* int16u *filtHeight,* int16s *filtScaling,* const int16s ∗ *filterCoefs*)

Computes the correlation with an anti-symmetric in X and anti-symmetric in Y filter.

Calculate the output of correlation with a filter, which is anti-symmetric in X and anti-symmetric in Y directions.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.15 void correlation__antisymYfilter (vec16s ∗ *dst,* vec08u ∗ *src,* int16s *sstr,* int16s *bw,* int16s *bh,* int16s *destBw,* int16s *xSkip,* int16s *ySkip,* int16u *filtWidth,* int16u *filtHeight,* int16s *filtScaling,* const int16s ∗ *filterCoefs*)

Computes the correlation with an anti-symmetric Y filter.

Calculate the correlation with a filter, which is anti-symmetric in Y direction.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
destBw	- [Input] Block width (including stride) of the output

xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.16 void correlation__symXantisymYfilter (vec16s ∗ *dst,* vec08u ∗ *src,* int16s *sstr,* int16s *bw,* int16s *bh,* int16s *destBw,* int16s *xSkip,* int16s *ySkip,* int16u *filtWidth,* int16u *filtHeight,* int16s *filtScaling,* const int16s ∗ *filterCoefs*)

Computes the correlation with a symmetric in X and anti-symmetric in Y filter.

Calculate the correlation with a filter, which is symmetric in X and anti-symmetric in Y directions.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.17 void correlation__symXantisymYfilter_16s (vec16s ∗ *dst,* vec16s ∗ *src,* int16s *sstr,* int16s *bw,* int16s *bh,* int16s *destBw,* int16s *xSkip,* int16s *ySkip,* int16u *filtWidth,* int16u *filtHeight,* int16s *scaleFact,* const int16s ∗ *filterCoefs*)

Computes the 16bit correlation with a symmetric in X and anti-symmetric in Y filter.

Calculate the correlation of a signed 16bit image with a filter, which is symmetric in X and anti-symmetric in Y directions.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.18 void correlation__symXfilter (vec16s ∗ *dst,* vec08u ∗ *src,* int16s *sstr,* int16s *bw,* int16s *bh,* int16s *destBw,* int16s *xSkip,* int16s *ySkip,* int16u *filtWidth,* int16u *filtHeight,* int16s *filtScaling,* const int16s ∗ *filterCoefs*)

Computes the correlation with a symmetric X filter.

Calculate the correlation with a filter, which is symmetric in X direction.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
destBw	- [Input] Block width (including stride) of the output
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.19 void correlation__symXYfilter (vec16s * *dst*, vec08u * *src*, int16s *sstr*, int16s *bw*, int16s *bh*, int16s *destBw*, int16s *xSkip*, int16s *ySkip*, int16u *filtWidth*, int16u *filtHeight*, int16s *filtScaling*, const int16s * *filterCoefs*)

Computes the correlation with a symmetric in X and Y filter.

Calculate the correlation with a filter, which is symmetric in X and in Y directions.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
destBw	- [Input] Block width (including stride) of the output
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.20 void correlation__symYfilter (vec16s * *dst*, vec08u * *src*, int16s *sstr*, int16s *bw*, int16s *bh*, int16s *destBw*, int16s *xSkip*, int16s *ySkip*, int16u *filtWidth*, int16u *filtHeight*, int16s *filtScaling*, const int16s * *filterCoefs*)

Computes the correlation with a symmetric Y filter.

Calculate the correlation with a filter, which is symmetric in Y direction.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
destBw	- [Input] Block width (including stride) of the output
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)

filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.21 void correlation_filter (vec16s ∗ *dst,* vec08u ∗ *src,* int16s *sstr,* int16s *bw,* int16s *bh,* int16s *destBw,* int16s *xSkip,* int16s *ySkip,* int16u *filtWidth,* int16u *filtHeight,* int16s *filtScaling,* const int16s ∗ *filterCoefs*)

Calculate the output of correlation with a general filter.

Parameters

dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
destBw	- [Input] Block width (including stride) of the output
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.27.2.22 void initFilters ( )

Initializes the array of filter function pointers.

Initializes the array of filter function pointers, to correspond to the right filters. Should be called once, before filtering.

5.27.2.23 void performCorrelation (int16u *filterFlags,* vec16s ∗ *dst,* vec08u ∗ *src,* int16s *sstr,* int16s *bw,* int16s *bh,* int16s *destBw,* int16s *xSkip,* int16s *ySkip,* int16u *filtWidth,* int16u *filtHeight,* int16s *scaleFact,* const int16s ∗ *filterCoefs*)

Computes time optimized the correlation with a general filter.

Based on the symmetry flag, this function chooses automatically the correct filter kernel and calculates the output of correlation with a general filter.

Parameters

filterFlags	- [Input] flags indicating the symmetry of the filter coefficients. For possible values see the definitions in this file
dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height
destBw	- [Input] Block width (including stride) of the output
xSkip,ySkip	- [Input] number of pixels to be skipped in x/y direction before convolving again, i.e. if destination has to be directly downsampled (=1 for no downsampling)

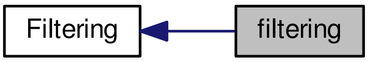
filtWidth	- [Input] the x-size of the filter
filtHeight	- [Input] the y-size of the filter
filtScaling	- [Input] the factor to divide the output of the filter with
filterCoefs	- [Input] the coefficients of the filter

5.28 filtering

5.28.1 Detailed Description

General Filtering.

Collaboration diagram for filtering:



Functions

- KERNEL_INFO apu_filter_a (" apu_filter_a ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3 >> 1, 3 >> 1, 3 >> 1, 3 >> 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_COEF"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(3 *3, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void apu_filter_a_impl (vec08u *src, int16_t sstr, uint8_t *coef, vec08u *dst, int16_t dstr, int16_t bw, int16_t bh)

  General filtering function with a 1D/2D-filter.

5.28.2 Function Documentation

5.28.2.1 KERNEL_INFO apu_filter_a (" apu_filter_a " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3 >> 1, 3 >> 1, 3 >> 1, 3 >> 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_COEF"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(3 *3, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

General filtering function with a 2D-filter

Parameters

FILTER_A_KN	Define for Kernel name
3	Number of ports
Port FILTER_A_KN_IN	Define for name of input image (unsigned 8bit)
Port FILTER_A_KN_IN_COEF	Define for name of filter coefficients (totally FILTER_W * FILTER_H coefficients are allowed right now) (unsigned 8bit)
Port FILTER_A_KN_OUT	Define for name of filtering result (unsigned 8bit)

5.28.2.2 void apu_filter_a_impl (vec08u * *src*, int16_t *sstr*, uint8_t * *coef*, vec08u * *dst*, int16_t *dstr*, int16_t *bw*, int16_t *bh*)

General filtering function with a 1D/2D-filter.

Calculate the output of convolution with a general filter.

Parameters

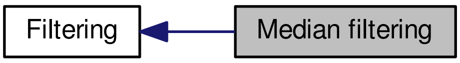
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
coef	- [Input] The filter coefficients
dst	- [Output] Destination block pointer
dstr	- [Input] Destination block stride
bw	- [Input] Block width
bh	- [Input] Block height

5.29 Median filtering

5.29.1 Detailed Description

Median filtering.

Collaboration diagram for Median filtering:



Functions

- KERNEL_INFO median_3x3_8bpp (" median_3x3_8bpp ", 2, __port(__index(0), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

    Filtering with a Median 3x3-filter.
- void apu_flt_median_3x3 (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh)

    Filter with a 3x3 median filter.

5.29.2 Function Documentation

5.29.2.1 void apu_flt_median_3x3 (vec08u ∗ *dst*, int *dstr*, const vec08u ∗ *src*, int *sstr*, int *bw*, int *bh*)

Filter with a 3x3 median filter.

Parameters

dst	- [Output] Destination block pointer
dstr	- [Input] Destination block stride
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height

5.29.2.2 KERNEL_INFO median_3x3_8bpp (" median_3x3_8bpp " , 2 , __port(__index(0), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Filtering with a Median 3x3-filter.

Filtering with a Median 3x3-filter

Parameters

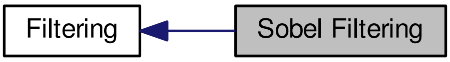
MEDIAN_3X3↩_8BPP_KN	Define for Kernel name
2	Number of ports
Port MEDIAN_↩3X3_8BPP_IN	Define for name of input image (unsigned 8bit)
Port MEDIANL_3↩X3_8BPP_OUT	Define for name of filtering result (unsigned 8bit)

5.30 Sobel Filtering

5.30.1 Detailed Description

Sobel filtering.

Collaboration diagram for Sobel Filtering:



Functions

- KERNEL_INFO sobel_3x3_8bpp (" sobel_3x3_8bpp ", 2, __port(__index(0), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

  Calculate sum of absolute values of first order derivatives x and y using sobel 3x3.
- void apu_flt_sobel_3x3_x (vec08u *dst, int dstr, const vec08u *src, int sstr, int bw, int bh)

  Calculate first order derivative x using sobel 3x3.
- void apu_flt_sobel_3x3_y (vec08u *dst, int dstr, const vec08u *src, int sstr, int bw, int bh)

  Calculate first order derivative y using sobel 3x3.
- void apu_flt_sobel_3x3 (vec08u *dst, int dstr, const vec08u *src, int sstr, int bw, int bh)

  Calculate sum of absolute values of first order derivatives x and y using sobel 3x3.

5.30.2 Function Documentation

5.30.2.1 void apu_flt_sobel_3x3 (vec08u * *dst*, int *dstr*, const vec08u * *src*, int *sstr*, int *bw*, int *bh*)

Calculate sum of absolute values of first order derivatives x and y using sobel 3x3.

Parameters

dst	- [Output] Destination block pointer
dstr	- [Input] Destination block stride
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height

5.30.2.2 void apu_flt_sobel_3x3_x (vec08u * *dst*, int *dstr*, const vec08u * *src*, int *sstr*, int *bw*, int *bh*)

Calculate first order derivative x using sobel 3x3.

Parameters

dst	- [Output] Destination block pointer
dstr	- [Input] Destination block stride
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height

5.30.2.3 void apu_flt_sobel_3x3_y (vec08u ∗ *dst*, int *dstr*, const vec08u ∗ *src*, int *sstr*, int *bw*, int *bh*)

Calculate first order derivative y using sobel 3x3.

Parameters

dst	- [Output] Destination block pointer
dstr	- [Input] Destination block stride
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height

5.30.2.4 KERNEL_INFO sobel_3x3_8bpp (" sobel_3x3_8bpp " , 2 , __port(__index(0), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Calculate sum of absolute values of first order derivatives x and y using sobel 3x3.

Parameters

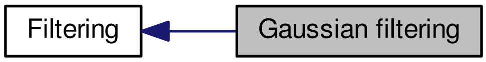
SOBEL_3X3_8BPP_KN	Define for Kernel name
2	Number of ports
Port SOBEL_3X3_8BPP_IN	Define for name of input image (unsigned 8bit)
Port SOBEL_3X3_8BPP_OUT	Define for name of filtering result (unsigned 8bit)

5.31 Gaussian filtering

5.31.1 Detailed Description

Gaussian filtering.

Collaboration diagram for Gaussian filtering:



Functions

- KERNEL_INFO apu_gauss_3x3 (" apu_gauss_3x3 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void apu_gauss_3x3 (vec08u ∗apOut, const vec08u ∗apcIn, int aOutStride, int aInStride, int aTileWidth, int aTileHeight)

    3x3 gaussian filter.
- KERNEL_INFO apu_gauss_5x5 (" apu_gauss_5x5 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void Gauss_5x5__filter (vec08u ∗dst, vec08u ∗src, int sstr, int bw, int bh)

    5x5 gaussian filter.

5.31.2 Function Documentation

5.31.2.1 KERNEL_INFO apu_gauss_3x3 (" apu_gauss_3x3 " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

3x3 Gaussian blur kernel metadata.

Parameters

GAUSS_3x3_KN	Define for Kernel name
2	Number of ports
Port GAUSS_3x3_KN_IN	Define for name of input image (unsigned 8bit)
Port GAUSS_3x3_KN_OUT	Define for name of blurred output image (unsigned 8bit)

5.31.2.2 void apu_gauss_3x3 (vec08u ∗ *apOut,* const vec08u ∗ *apcIn,* int *aOutStride,* int *aInStride,* int *aTileWidth,* int *aTileHeight*)

3x3 gaussian filter.

Calculate the output of 3x3 gaussian filter.

Parameters

apOut	- [Output] pointer to the destination buffer
apcIn	- [Input] pointer to the source buffer
aOutStride	- [Input] line stride of the destination data
aInStride	- [Input] line stride of the source data
aTileWidth	- [Input] width of one data block
aTileHeight	- [Input] height of one data block

Returns

number of corners found

5.31.2.3 KERNEL_INFO apu_gauss_5x5 (" apu_gauss_5x5 " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

5x5 Gaussian blur kernel metadata.

Parameters

GAUSS_5x5_↩ KN	Define for Kernel name
2	Number of ports
Port GAUSS_↩ 5x5_KN_IN	Define for name of input image (unsigned 8bit)
Port GAUSS_↩ 5x5_KN_OUT	Define for name of blurred output image name (unsigned 8bit)

5.31.2.4 void Gauss_5x5__filter (vec08u ∗ *dst,* vec08u ∗ *src,* int *sstr,* int *bw,* int *bh*)

5x5 gaussian filter.

Calculate the output of 5x5 gaussian filter.

Parameters

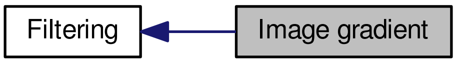
dst	- [Output] Destination block pointer
src	- [Input] Source block pointer
sstr	- [Input] Source block stride
bw	- [Input] Block width
bh	- [Input] Block height

5.32 Image gradient

5.32.1 Detailed Description

Image gradient.

Collaboration diagram for Image gradient:



Functions

- KERNEL_INFO apu_gradient (" apu_gradient ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_GX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_GY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO **apu_gradient_out08s** (" apu_gradient_out08s ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_GX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_GY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_gradient_abs (" apu_gradient_abs ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_GX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_GY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUT_ABSSUM"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_gr_abs (" apu_gr_abs ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_ABSSUM"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void apuGradient (vec16s ∗apcSobelX, vec16s ∗apcSobelY, const vec08u ∗apInput, int aBlockWidth, int aBlockHeight, int aStride)

  Image gradient. ==> GradX, GradY.
- void apuGradient_out08s (vec08s ∗apcSobelX, vec08s ∗apcSobelY, const vec08u ∗apInput, int aBlockWidth, int aBlockHeight, int aStride)

  Image gradient. ==> GradX, GradY.
- void apuGradientAbs (vec08s ∗apcSobelX, vec08s ∗apcSobelY, vec08u ∗apcAbsSum, const vec08u ∗apInput, int aBlockWidth, int aBlockHeight, int aStride)

  Image gradient ==> GradX, GradY, |GradX|+|GradY|.
- void apuGradAbs (vec08u ∗apcAbsSum, const vec08u ∗apInput, int aBlockWidth, int aBlockHeight, int aStride)

  Image gradient absulute sum ==> |GradX|+|GradY|.

5.32.2 Function Documentation

5.32.2.1 KERNEL_INFO apu_gr_abs (" apu_gr_abs " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUT_ABSSUM"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Image gradient kernel metadata.

Parameters

GRADIENT_ABS_KN	Define for Kernel name
2	Number of ports
Port GRADIENT_KN_IN	Define for name of input image (unsigned 8bit)
Port GRADIENT_KN_ABSGRAD	Define for name of sum of absolute values of grad X/Y components output image (unsigned 16bit)

5.32.2.2 KERNEL_INFO apu_gradient (" apu_gradient " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUT_GX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUT_GY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Image gradient kernel metadata.

Parameters

GRADIENT_KN	Define for Kernel name
3	Number of ports
Port GRADIENT_KN_IN	Define for name of input image (unsigned 8bit)
Port GRADIENT_KN_OUTX	Define for name of Gradient X component output image (signed 16bit)
Port GRADIENT_KN_OUTY	Define for name of Gradient Y component output image (signed 16bit)

5.32.2.3 KERNEL_INFO apu_gradient_abs (" apu_gradient_abs " , 4 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUT_GX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUT_GY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("OUT_ABSSUM"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Image gradient kernel metadata.

Parameters

GRADIENT_ABS_KN	Define for Kernel name

4	Number of ports
Port GRADIEN↵ T_KN_IN	Define for name of input image (unsigned 8bit)
Port GRADIEN↵ T_KN_OUTX	Define for name of Gradient X component output image (signed 16bit)
Port GRADIEN↵ T_KN_OUTY	Define for name of Gradient Y component output image (signed 16bit)
Port GRADIENT_K↵ N_ABSGRAD	Define for name of sum of absolute values of grad X/Y components output image (unsigned 16bit)

5.32.2.4 void apuGradAbs (vec08u ∗ *apcAbsSum,* const vec08u ∗ *apInput,* int *aBlockWidth,* int *aBlockHeight,* int *aStride*)

Image gradient absulute sum ==> |GradX|+|GradY|.

Computes image gradient (horizontal and vertical Sobel filter). Returns sum of their absolute values

Parameters

apcAbsSum	- [Output] Sum of absolute values of GradX/Y destination block pointer (unsigned 16bit)
apInput	- [Input] Source block pointer (unsigned 8bit)
aBlockWidth	- [Input] Block width
aBlockHeight	- [Input] Block height
aStride	- [Input] Source block stride

5.32.2.5 void apuGradient (vec16s ∗ *apcSobelX,* vec16s ∗ *apcSobelY,* const vec08u ∗ *apInput,* int *aBlockWidth,* int *aBlockHeight,* int *aStride*)

Image gradient. ==> GradX, GradY.

Computes image gradient (horizontal and vertical Sobel filter). Returns Gradient X/Y

Parameters

apcSobelX	- [Output] Horizontal Sobel Filter destination block pointer (signed 16bit)
apcSobelY	- [Output] Vertical Sobel Filter destination block pointer (signed 16bit)
apInput	- [Input] Source block pointer (unsigned 8bit)
aBlockWidth	- [Input] Block width
aBlockHeight	- [Input] Block height
aStride	- [Input] Source block stride

5.32.2.6 void apuGradient_out08s (vec08s ∗ *apcSobelX,* vec08s ∗ *apcSobelY,* const vec08u ∗ *apInput,* int *aBlockWidth,* int *aBlockHeight,* int *aStride*)

Image gradient. ==> GradX, GradY.

Computes image gradient (horizontal and vertical Sobel filter). Returns Gradient X/Y

Parameters

apcSobelX	- [Output] Horizontal Sobel Filter destination block pointer (signed 08bit)
apcSobelY	- [Output] Vertical Sobel Filter destination block pointer (signed 08bit)
apInput	- [Input] Source block pointer (unsigned 8bit)

aBlockWidth	- [Input] Block width
aBlockHeight	- [Input] Block height
aStride	- [Input] Source block stride

5.32.2.7 void apuGradientAbs (vec08s ∗ *apcSobelX,* vec08s ∗ *apcSobelY,* vec08u ∗ *apcAbsSum,* const vec08u ∗ *apInput,* int *aBlockWidth,* int *aBlockHeight,* int *aStride*)

Image gradient ==> GradX, GradY, |GradX|+|GradY|.

Computes image gradient (horizontal and vertical Sobel filter). Returns Gradient X/Y and sum of their absolute values

Parameters

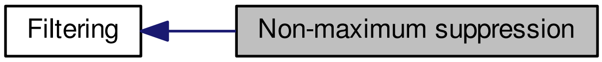
apcSobelX	- [Output] Horizontal Sobel Filter destination block pointer (signed 8bit)
apcSobelY	- [Output] Vertical Sobel Filter destination block pointer (signed 8bit)
apcSobelY	- [Output] Sum of absolute values of GradX/Y destination block pointer (unsigned 16bit)
apInput	- [Input] Source block pointer (unsigned 8bit)
aBlockWidth	- [Input] Block width
aBlockHeight	- [Input] Block height
aStride	- [Input] Source block stride

5.33 Non-maximum suppression

5.33.1 Detailed Description

Non-maximum suppression.

Collaboration diagram for Non-maximum suppression:



Functions

- KERNEL_INFO apu_nms (" apu_nms ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_nms16 (" apu_nms16 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- void apu_nms_impl (const vec08u ∗apcIn, vec08u ∗apOut, int aInStride, int aOutStride, int aTileWidth, int aTileHeight)

  Non-maximum suppression.
- void apu_nms16 (const vec16u ∗apcIn, vec16u ∗apOut, int aInStride, int aOutStride, int aTileWidth, int aTileHeight)

  Non-maximum suppression, 16-bit.

5.33.2 Function Documentation

5.33.2.1 KERNEL_INFO apu_nms (" apu_nms " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Non-maximum suppression kernel metadata.

Parameters

NMS_KN	Define for Kernel name
2	Number of ports
Port NMS16_KN_IN	Define for name of input image (unsigned 8bit)
Port NMS16_KN_OUT	Define for name of output image (unsigned 8bit)

5.33.2.2 void apu_nms16 (const vec16u ∗ *apcIn*, vec16u ∗ *apOut*, int *aInStride*, int *aOutStride*, int *aTileWidth*, int *aTileHeight*)

Non-maximum suppression, 16-bit.

Sets values which are not maximal in their 3x3 neighborhood (8 pixels) to 0. 16-bit version.

Parameters

apcIn	- [Input] Pointer to the source buffer
apOut	- [Output] Pointer to the destination buffer
aInStride	- [Input] Line stride of the source data
aOutStride	- [Input] Line stride of the destination data
aTileWidth	- [Input] Width of one data block
aTileHeight	- [Input] Height of one data block

5.33.2.3 KERNEL_INFO apu_nms16 (" apu_nms16 " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

16-bit non-maximum suppression kernel metadata.

Parameters

NMS16_KN	Define for Kernel name
2	Number of ports
Port NMS16_KN_IN	Define for name of input image (unsigned 16bit)
Port NMS16_K↩N_OUT	Define for name of output image (unsigned 16bit)

5.33.2.4 void apu_nms_impl (const vec08u ∗ *apcIn*, vec08u ∗ *apOut*, int *aInStride*, int *aOutStride*, int *aTileWidth*, int *aTileHeight*)

Non-maximum suppression.

Sets values which are not maximal in their 3x3 neighborhood (8 pixels) to 0. 8-bit version

Parameters

apcIn	- [Input] Pointer to the source buffer
apOut	- [Output] Pointer to the destination buffer
aInStride	- [Input] Line stride of the source data
aOutStride	- [Input] Line stride of the destination data
aTileWidth	- [Input] Width of one data block
aTileHeight	- [Input] Height of one data block

5.34 Row_filter

5.34.1 Detailed Description

Filter an image column with a row_filter(i.e. Row_Vector * Matrix multiplication)

Collaboration diagram for Row_filter:

Functions

- KERNEL_INFO row_filter (" row_filter ", 3, __port(__index(0), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, ROW_PADDING, ROW_PADDING), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("COEFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(FILTER_ROWS, 1)), __port(__index(2), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void row_filter_impl (vec08u *dst, int dstStride, const vec08u *src, int srcStride, int rows, int cols, const unsigned char *filter, int filterSize, int filterQ)

Variables

- const int **FILTER_ROWS** = 5
- const int **ROW_PADDING** = FILTER_ROWS >> 1

5.34.2 Function Documentation

5.34.2.1 KERNEL_INFO row_filter (" row_filter " , 3 , __port(__index(0), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, ROW_PADDING, ROW_PADDING), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("COEFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(FILTER_ROWS, 1)) , __port(__index(2), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

ROW filter kernel metadata.

Parameters

ROW_FILTER_KN	Define for Kernel name
3	Number of ports
Port ROW_FILTER_KN_SRC	Define for name of input image (unsigned 8bit)

Port ROW_FILTER← _KN_COEFFS	Define for name of filter coefficients (unsigned 8bit)
Port ROW_FIL← TER_KN_DST	Define for name of output image (unsigned 8bit)

5.34.2.2 void row_filter_impl (vec08u ∗ *dst*, int *dstStride*, const vec08u ∗ *src*, int *srcStride*, int *rows*, int *cols*, const unsigned char ∗ *filter*, int *filterSize*, int *filterQ*)

Parameters

dst	Pointer to the destination image.
dstStride	Stride of the destination image.
src	Pointer to the source image.
srcStride	Stride of the source image.
rows	Rows of the source image, not including padding.
cols	Columns of the source image, not including padding.
filter	Pointer to the row filter coefficients. The coefficients are unsigned 8-bit fixed point numbers.
filterSize	Size of the column filter.
filterQ	Number of fractional bits for the filter coefficients.

5.35 Saturation

5.35.1 Detailed Description

Saturation.

Collaboration diagram for Saturation:

Functions

- KERNEL_INFO apu_saturate_nonzero (" apu_saturate_nonzero ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_binarize (" apu_binarize ", 2, __port(__index(0), __identifier("BIN_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("BIN_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_mask (" apu_mask ", 5, __port(__index(0), __identifier("MASK_INFL"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MASK_INX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("MASK_INY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("MASK_OUTX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("MASK_OUTY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- void saturate_nonzero (vec08u ∗dst, vec08u ∗src, int dstr, int sstr, int bw, int bh)

  Non-zero pixel saturation.
- void binarize (vec08u ∗dst, vec32u ∗src, int, int, int bw, int bh)

  Non-zero pixel binarization.
- void mask (vec16s ∗dstX, vec16s ∗dstY, vec32u ∗srcFlags, vec16s ∗inX, vec16s ∗inY, int, int, int bw, int bh)

  Masking to zero with the srcFlags.

5.35.2 Function Documentation

5.35.2.1 KERNEL_INFO apu_binarize (" apu_binarize " , 2 , __port(__index(0), __identifier("BIN_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("BIN_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Non-zero saturation kernel metadata.

Parameters

SAT_NONZER↩ O_KN	Define for Kernel name
2	Number of ports
Port SAT_NO↩ NZERO_IN	Define for name of input image (unsigned 8bit)
Port SAT_NO↩ NZERO_OUT	Define for name of output image (unsigned 8bit)

5.35.2.2 KERNEL_INFO apu_mask (" apu_mask " , 5 , __port(__index(0), __identifier("MASK_INFL"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MASK_INX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("MASK_INY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("MASK_OUTX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("MASK_OUTY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Masking kernel metadata. Mask input to zero if flags are zero

Parameters

MASK_KN	Define for Kernel name
5	Number of ports
Port MASK_K_↩ INFLAGS	Define for name of masking flags input image (unsigned 32bit)
Port MASK_INX	Define for name of input X image to be masked(signed 16bit)
Port MASK_INY	Define for name of input Y image to be masked(signed 16bit)
Port MASK_OUTX	Define for name of output X image (signed 16bit)
Port MASK_OUTY	Define for name of output Y image (signed 16bit)

5.35.2.3 KERNEL_INFO apu_saturate_nonzero (" apu_saturate_nonzero " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Non-zero saturation kernel metadata.

Parameters

SAT_NONZER↩ O_KN	Define for Kernel name
2	Number of ports
Port SAT_NO↩ NZERO_IN	Define for name of input image (unsigned 8bit)
Port SAT_NO↩ NZERO_OUT	Define for name of output image (unsigned 8bit)

5.35.2.4 void binarize (vec08u ∗ *dst,* vec32u ∗ *src,* int , int , int *bw,* int *bh*)

Non-zero pixel binarization.

Changes non-zero pixel values to maximal values.

Parameters

dst	- [Output] Pointer to the destination buffer (unsigned 8bit)
src	- [Input] Pointer to the source buffer (unsigned 32bit)
dstr	- [Input] Line stride of the destination data
sstr	- [Input] Line stride of the source data
bw	- [Input] Width of one data block
bh	- [Input] Height of one data block

5.35.2.5 void mask (vec16s ∗ *dstX,* vec16s ∗ *dstY,* vec32u ∗ *srcFlags,* vec16s ∗ *inX,* vec16s ∗ *inY,* int *,* int *,* int *bw,* int *bh*)

Masking to zero with the srcFlags.

Masks the output pixel values to zero, if the srcFlags are zero. Useful to mask thresholded GradientX/Y

Parameters

dstX	- [Output] Pointer to the X destination buffer (signed 16bit)
dstY	- [Output] Pointer to the Y destination buffer (signed 16bit)
srcFlags-	[Input] Pointer to the mask source buffer (unsigned 32bit)
inX	- [Output] Pointer to the X source buffer to be masked (signed 16bit)
inY	- [Output] Pointer to the Y source buffer to be masked(signed 16bit)
dstr	- [Input] Line stride of the destination data
sstr	- [Input] Line stride of the source data
bw	- [Input] Width of one data block
bh	- [Input] Height of one data block

5.35.2.6 void saturate_nonzero (vec08u ∗ *dst,* vec08u ∗ *src,* int *dstr,* int *sstr,* int *bw,* int *bh*)

Non-zero pixel saturation.

Changes non-zero pixel values to maximal values.

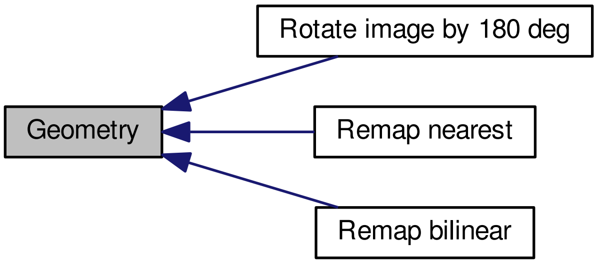
Parameters

dst	- [Output] Pointer to the destination buffer (unsigned 8bit)
src	- [Input] Pointer to the source buffer (unsigned 8bit)
dstr	- [Input] Line stride of the destination data
sstr	- [Input] Line stride of the source data
bw	- [Input] Width of one data block
bh	- [Input] Height of one data block

5.36 Geometry

5.36.1 Detailed Description

Collaboration diagram for Geometry:



Modules

- Remap bilinear

    Element-wise bilinear interpolation.
- Remap nearest

    Element-wise nearest neighbor interpolation.
- Rotate image by 180 deg

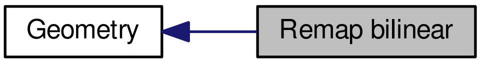
    Image rotation.

5.37 Remap bilinear

5.37.1 Detailed Description

Element-wise bilinear interpolation.

Collaboration diagram for Remap bilinear:



Functions

- KERNEL_INFO remap_bilinear_rgb (" remap_bilinear_rgb ", 4, __port(__index(0), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(8, 9)), __port(__index(2), __identifier("OFFSET"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("DELTA"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(2, 1), __ek_size(1, 1)))
- KERNEL_INFO remap_bilinear_grayscale (" remap_bilinear_grayscale ", 4, __port(__index(0), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(36, 12)), __port(__index(2), __identifier("OFFSET"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("DELTA"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(2, 1), __ek_size(1, 1)))
- void remap_bilinear_rgb_impl (vec32u ∗dst, vec32u ∗src, vec16u ∗offset, vec08u ∗delta, int sstride, int dstride, int bw, int bh)

  Elementwise bilinear interpolation.
- void remap_bilinear_grayscale_impl (vec08u ∗dst, vec08u ∗src, vec16u ∗offset, vec08u ∗delta, int sstride, int dstride, int bw, int bh)

  Elementwise bilinear interpolation.

5.37.2 Function Documentation

5.37.2.1 KERNEL_INFO remap_bilinear_grayscale (" remap_bilinear_grayscale " , 4 , __port(__index(0), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(36, 12)) , __port(__index(2), __identifier("OFFSET"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("DELTA"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(2, 1), __ek_size(1, 1)))

GrayScale interpolation kernel metadata. Interpolates between neighboring pixels of a grayscale image. Outputs 8bit unsigned interpolation result.

Parameters

REMAP_BILIN↩_GRAY_KN	Define for Kernel name
4	Number of ports
Port REMAP_↩BILIN_KN_DST	Define for name of interpolation result (unsigned 8bit).
Port REMAP_↩BILIN_KN_SRC	Define for name of first input image (unsigned 8bit)
Port REMAP_BILIN↩_KN_OFFS	Define for name of input offsets (unsigned 16bit)
Port REMAP_BILIN↩_KN_DELTA	Define for name of second input deltas (unsigned 8bit)

5.37.2.2 void remap_bilinear_grayscale_impl (vec08u ∗ *dst,* vec08u ∗ *src,* vec16u ∗ *offset,* vec08u ∗ *delta,* int *sstride,* int *dstride,* int *bw,* int *bh*)

Elementwise bilinear interpolation.

remap_bilinear btw the grayscale pixels of an image

Parameters

dst	- [Output] unsigned 8bit destination block pointer
srcImage0-	[Input] unsigned 8bit source block pointer of img 0
offset	- [Input] unsigned 16bit offsets inside source block pointer of img 0
delta	- [Input] unsigned 8bit deltas inside source block pointer of img 0
sstride	- [Input] Source block width (in elements not bytes) including padding
dstride	- [Input] Destination block width (in elements not bytes) including padding
bw	- [Input] Block width
bh	- [Input] Block height

5.37.2.3 KERNEL_INFO remap_bilinear_rgb (" remap_bilinear_rgb " , 4 , __port(__index(0), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(8, 9)) , __port(__index(2), __identifier("OFFSET"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("DELTA"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(2, 1), __ek_size(1, 1)))

RGB interpolation kernel metadata. Interpolates between neighboring pixels of a RGB image. Outputs 8bit unsigned interpolation result.

Parameters

REMAP_BILIN↩_GRAY_KN	Define for Kernel name
4	Number of ports
Port REMAP_↩BILIN_KN_DST	Define for name of interpolation result (unsigned 32bit).

Port REMAP_↩ BILIN_KN_SRC	Define for name of first input image (unsigned 32bit)
Port REMAP_BILIN↩ _KN_OFFS	Define for name of input offsets (unsigned 16bit)
Port REMAP_BILIN↩ _KN_DELTA	Define for name of second input deltas (unsigned 8bit)

5.37.2.4 void remap_bilinear_rgb_impl (vec32u ∗ *dst,* vec32u ∗ *src,* vec16u ∗ *offset,* vec08u ∗ *delta,* int *sstride,* int *dstride,* int *bw,* int *bh*)

Elementwise bilinear interpolation.

remap_bilinear btw the rgb pixels of an image

Parameters

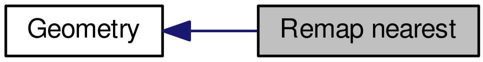
dst	- [Output] unsigned 32bit destination block pointer
srcImage0-	[Input] unsigned 32bit source block pointer of img 0
offset	- [Input] unsigned 16bit offsets inside source block pointer of img 0
delta	- [Input] unsigned 8bit deltas inside source block pointer of img 0
sstride	- [Input] Source block width (in elements not bytes) including padding
dstride	- [Input] Destination block width (in elements not bytes) including padding
bw	- [Input] Block width
bh	- [Input] Block height

5.38 Remap nearest

5.38.1 Detailed Description

Element-wise nearest neighbor interpolation.

Collaboration diagram for Remap nearest:



Functions

- KERNEL_INFO remap_nearest_grayscale (" remap_nearest_grayscale ", 3, __port(__index(0), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OFFSET"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- void remap_nearest_grayscale_impl (vec08u ∗dst, vec08u ∗src, vec16u ∗offset, int sstride, int dstride, int bw, int bh)

  Elementwise nearest neighbor interpolation.

5.38.2 Function Documentation

5.38.2.1 KERNEL_INFO remap_nearest_grayscale (" remap_nearest_grayscale " , 3 , __port(__index(0), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OFFSET"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

GrayScale interpolation kernel metadata. Interpolates between neighboring pixels of a grayscale image. Outputs 8bit unsigned interpolation result.

Parameters

REMAP_NEAREST_GRAY_KN	Define for Kernel name
4	Number of ports
Port REMAP_NEAREST_KN_DST	Define for name of interpolation result (unsigned 8bit).

Port REMAP_NEAREST_KN_SRC	Define for name of first input image (unsigned 8bit)
Port REMAP_NEAREST_KN_OFFS	Define for name of input offsets (unsigned 16bit)

5.38.2.2 void remap_nearest_grayscale_impl (vec08u ∗ *dst*, vec08u ∗ *src*, vec16u ∗ *offset*, int *sstride*, int *dstride*, int *bw*, int *bh*)

Elementwise nearest neighbor interpolation.

remap_nearest btw the grayscale pixels of an image

Parameters

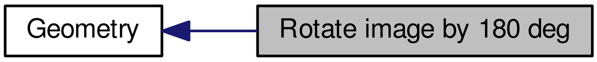
dst	- [Output] unsigned 8bit destination block pointer
srcImage0-	[Input] unsigned 8bit source block pointer of img 0
offset	- [Input] unsigned 16bit offsets inside source block pointer of img 0
sstride	- [Input] Source block width (in elements not bytes) including padding
dstride	- [Input] Destination block width (in elements not bytes) including padding
bw	- [Input] Block width
bh	- [Input] Block height

5.39 Rotate image by 180 deg

5.39.1 Detailed Description

Image rotation.

Collaboration diagram for Rotate image by 180 deg:



Macros

- #define **ROTATE_K** apu_rotate_180
- #define **ROTATE_KN** XSTR(ROTATE_K)
- #define **ROTATE_KN_IN** "INPUT"
- #define **ROTATE_KN_OUT** "OUTPUT"

Functions

- KERNEL_INFO apu_rotate_180 (" apu_rotate_180 ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- **extKernelInfoDecl** (ROTATE_180_K)
- void **apu_rotate_180** (kernel_io_desc lIn, kernel_io_desc lOut)
- **extKernelInfoDecl** (apu_rotate_180)
- void rotate_180 (vec08u ∗dst, vec08u ∗src, int bw, int bh, int sstr)

  180-degree rotation.

5.39.2 Function Documentation

5.39.2.1 KERNEL_INFO apu_rotate_180 (" apu_rotate_180 " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

180-degree rotation kernel metadata.

Parameters

ROTATE_KN	Define for Kernel name
2	Number of ports
Port ROTATE_KN_IN	Define for name of input image (unsigned 8bit)

Port ROTATE_↩ KN_OUT	Define for name of output image (unsigned 8bit)

5.39.2.2 void rotate_180 (vec08u ∗ *dst*, vec08u ∗ *src*, int *bw*, int *bh*, int *sstr*)

180-degree rotation.

Rotates the image 180 degrees.

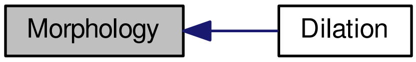
Parameters

dst	- [Output] pointer to the destination buffer
src	- [Input] pointer to the source buffer
bw	- [Input] width of one data block
bh	- [Input] height of one data block
sstr	- [Input] source stride

5.40 Morphology

5.40.1 Detailed Description

Collaboration diagram for Morphology:



Modules

- Dilation

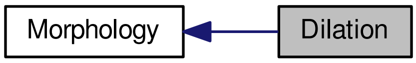
  Image dilation.

5.41 Dilation

5.41.1 Detailed Description

Image dilation.

Collaboration diagram for Dilation:



Functions

- KERNEL_INFO apu_dilate_diamond (" apu_dilate_diamond ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void dilate_diamond (vec08u ∗dst, vec08u ∗src, int dstr, int sstr, int bw, int bh)

    5x5 diamond dilation.

5.41.2 Function Documentation

5.41.2.1 KERNEL_INFO apu_dilate_diamond (" apu_dilate_diamond " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Diamond dilation kernel metadata.

Parameters

DILATE_DIAMOND_KN	Define for Kernel name
2	Number of ports
Port DILATE_DIAMOND_KN_IN	Define for name of input image (unsigned 8bit)
Port DILATE_DIAMOND_KN_OUT	Define for name of output image (unsigned 8bit)

5.41.2.2 void dilate_diamond (vec08u ∗ *dst*, vec08u ∗ *src*, int *dstr*, int *sstr*, int *bw*, int *bh*)

5x5 diamond dilation.

Dilates the image using 5x5 diamond structure element.

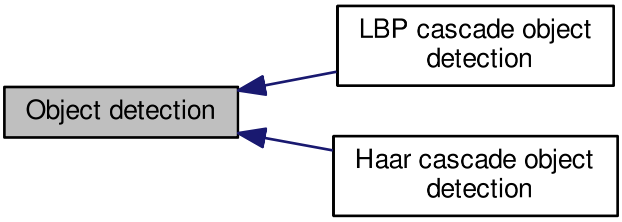
Parameters

dst	- [Output] pointer to the destination buffer
src	- [Input] pointer to the source buffer
dstr	- [Input] line stride of the destination data
sstr	- [Input] line stride of the source data
bw	- [Input] width of one data block
bh	- [Input] height of one data block

5.42 Object detection

5.42.1 Detailed Description

Collaboration diagram for Object detection:



Modules

- Haar cascade object detection

    Haar cascade object detection.
- LBP cascade object detection

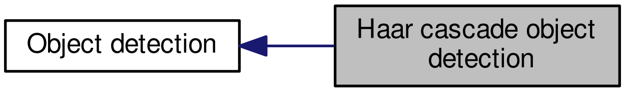
    LBP cascade object detection.

5.43 Haar cascade object detection

5.43.1 Detailed Description

Haar cascade object detection.

Collaboration diagram for Haar cascade object detection:



Classes

- struct APEX_HaarCascadeFeature
- struct APEX_HaarCascadeStage

Typedefs

- typedef vec16u **FEATURE_FIXED_POINT_TYPE**
- typedef vec16u **STAGE_FIXED_POINT_TYPE**

Functions

- KERNEL_INFO apu_haar_cascade (" apu_haar_cascade ", 11, __port(__index(0), __identifier("INPUT_INTEGRAL_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_INTEGRAL_IMAGE_SQUARED"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LINE_INDEX"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("WINDOW_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(6+1, 32)), __port(__index(4), __identifier("WINDOW_BUFFER_SQUARED"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(6+1, 32)), __port(__index(5), __identifier("INPUT_CASCADE_SIZES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(6), __identifier("INPUT_CASCADE_FEATURES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(19800, 1)), __port(__index(7), __identifier("INPUT_CASCADE_STAGES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(70, 1)), __port(__index(8), __identifier("INPUT_PIXEL_SHIFTS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(32, 1)), __port(__index(9), __identifier("INPUT_PIXEL_OFFSETS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(32, 1)), __port(__index(10), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void haar_cascade (vec08u ∗apOut, vec32u ∗apInII, vec32u ∗apInII2, int aOutStride, int aInStride, int aTileWidth, int aTileHeight, int16u aLineIndex, vec32u ∗apWindowBuffer, vec32u ∗apWindowBuffer2, const APEX_HaarCascadeFeature ∗apcFeatures, int aStageCount, const APEX_HaarCascadeStage ∗apcStages, const int08u ∗apcXshifts, const int08u ∗apcXoffsets)

Haar object detection.

Variables

- const int **FEATURE_FRACTIONAL_BITS** = 13
- const int **FEATURE_FIXED_POINT_MULTIPLIER** = (1 << FEATURE_FRACTIONAL_BITS)
- const int **STAGE_FRACTIONAL_BITS** = 10
- const int **STAGE_FIXED_POINT_MULTIPLIER** = (1 << STAGE_FRACTIONAL_BITS)
- const int32u **featureFixedCoefficientSqrt** = 91
- const int32s **invWindowAreaScalar** = 25

5.43.2 Function Documentation

5.43.2.1 KERNEL_INFO apu_haar_cascade (" apu_haar_cascade " , 11 , __port(__index(0), __identifier("INPU↩T_INTEGRAL_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_INTEGRAL_IMAGE_SQUARED"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("LINE_INDEX"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("WINDOW_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(6+1, 32)) , __port(__index(4), __identifier("WINDOW_BUFFER_SQUARED"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(6+1, 32)) , __port(__index(5), __identifier("INPUT_CASCADE_SIZES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 1)) , __port(__index(6), __identifier("INPUT_CASCADE_FEATURES"), __attributes(ACF_ATTR_↩SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(19800, 1)) , __port(__index(7), __identifier("INPUT_CASCADE_STAGES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(70, 1)) , __port(__index(8), __identifier("INPUT_PIXEL_SHIFTS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(32, 1)) , __port(__index(9), __identifier("INPUT_PIXEL_OFFSETS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(32, 1)) , __port(__index(10), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Haar cascade object detection kernel metadata.

Parameters

apu_haar_↩cascade	Kernel name
11	Number of ports
Port HAAR_C↩ASC_IN_INTE↩GR_IMG	Define for name of input integral image (unsigned 32bit)
Port HAAR_C↩ASC_IN_INTE↩GR_IMG_SQR	Define for name of input integral image of squared values (unsigned 32bit)
Port HAAR_C↩ASC_LINE_IDX	Define for name of line index buffer (unsigned 16bit)
Port HAAR_CASC_↩WND_BUF	Define for name of integral image window buffer (unsigned 32bit)

Port HAAR_CASC_WND_BUF_SQR	Define for name of integral image of squared values window buffer (unsigned 32bit)
Port HAAR_CASC_IN_CASC_SZ	Define for name of number of features and stages (2 16-bit values) (unsigned 16bit)
Port HAAR_CASC_IN_CASC_FEAT	Define for name of array of Haar-like features (unsigned 8bit)
Port HAAR_CASC_IN_CASC_STAGES	Define for name of array of cascade stages (unsigned 8bit)
Port HAAR_CASC_IN_PIX_SHFT	Define for name of LUT containing tile shifts needed for horizontal pixel offsets (unsigned 8bit)
Port HAAR_CASC_IN_PIX_OFFSETS	Define for name of LUT containing in-tile offsets needed for horizontal pixel offsets (unsigned 8bit)
Port HAAR_CASC_OUT	Define for name of output image (unsigned 8bit)

5.43.2.2 void haar_cascade (vec08u ∗ *apOut*, vec32u ∗ *apInII*, vec32u ∗ *apInII2*, int *aOutStride*, int *aInStride*, int *aTileWidth*, int *aTileHeight*, int16u *aLineIndex*, vec32u ∗ *apWindowBuffer*, vec32u ∗ *apWindowBuffer2*, const APEX_HaarCascadeFeature ∗ *apcFeatures*, int *aStageCount*, const APEX_HaarCascadeStage ∗ *apcStages*, const int08u ∗ *apcXshifts*, const int08u ∗ *apcXoffsets*)

Haar object detection.

Detects objects using Haar-like feature cascades. The algorithm searches for 20x20-pixel objects using a Haar-like classifier provided by the user. For each input pixel, it outputs 255 if the pixel is a lower left corner of an object and 0 otherwise. The classifier consists of a number of stages, each stage contains multiple Haar-like features. The window has to pass every stage in order to be classified as containing an object.

See http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html

The format of the classifier stage is:

```
struct APEX_HaarCascadeStage
{
  int08u featureCount;
  int16u thresholdFixed;
};
```

Where featureCount is a number of Haar-like features included in the stage.
ThresholdFixed is a fixed point threshold (10 fractional bits) which has to be exceeded by the sum of the feature test results in order for the stage to pass.

A Haar-like feature is a rectangle subdivided into weighted sub-rectangles. The pixels inside the feature region are weighted according to the weight of the sub-rectangle they belong to and then they are summed together. The resulting sum is then compared with the feature's threshold and the resulting boolean tells whether the feature passed or not.

```
struct APEX_HaarCascadeStage
{
  int16u threshold;
  int16u leftVal, rightVal;
  int16u x5y5w5_1;
  int08u h5type3;
};
```

Where threshold is a fixed-point number (13 fractional bits) which has to be exceeded by the weighted sum of the feature rectangles in order for the feature to pass.
leftVal, rightVal are fixed-point values (10 fractional bits), leftVal is added to the stage sum if the feature doesn't pass, righVal is added to the stage sum if the feature passes.
x5y5w5_1 contains 3 values in its bits, starting from the most significant bit: upper left corner feature position x component in the first 5 bits, the y component in the next bits and the width of the feature in the next 5 bits. The least significant bit is not used.
h5type3 contains 2 values in its bits, starting from the most significant bit: feature height in the first bit, feature type in the remaining 3 bits.

There are 8 possible types of feature:
type 0 ... horizontal edge, upper half positive
type 1 ... horizontal edge, lower half positive
type 2 ... vertical edge, left half positive
type 3 ... vertical edge, right half positive
type 4 ... line, horizontal
type 5 ... line, vertical
type 6 ... 4 rectangles, upper left and lower right rectangles positive ()
type 7 ... 4 rectangles, lower left and upper right rectangles positive (/)
The kernel expects an integral image and an integral image of squared values. The implementation retains a rolling window for the integral images, therefore sufficiently sized buffers have to be supplied.

Parameters

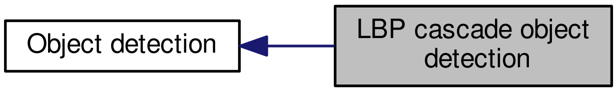
apOut	- [Output] pointer to the destination buffer
apInII	- [Input] pointer to the source integral image buffer
apInII2	- [Input] pointer to the source integral image of squared values buffer
aOutStride	- [Input] line stride of the destination data
aInStride	- [Input] line stride of the source data
aTileWidth	- [Input] width of one data tile
aTileHeight	- [Input] height of one data tile
aLineIndex	- [Input] index of currently processed line
apWindowBuffer	- [Buffer] pointer to the input integral image window buffer
apWindow↔ Buffer2	- [Buffer] pointer to the input integral image of squared values window buffer
apcFeatures	- [Input] pointer to the Haar-like feature array
aStageCount	- [Input] number of Haar-like feature cascade stages
apcStages	- [Input] pointer to the Haar-like feature cascade stages
apcXshifts	- [Input] pointer to LUT containing tile shifts needed for horizontal pixel offsets
apcXoffsets	- [Input] pointer to LUT containing in-tile offsets needed for horizontal pixel offsets

5.44 LBP cascade object detection

5.44.1 Detailed Description

LBP cascade object detection.

Collaboration diagram for LBP cascade object detection:



Classes

- struct APEX_lbpFeature
- struct APEX_lbpStage

Typedefs

- typedef vec32s **STAGE_FIXED_POINT_TYPE**
- typedef int32s **STAGE_FIXED_POINT_TYPE_SCALAR**

Functions

- KERNEL_INFO apu_lbp_cascade (" apu_lbp_cascade ", 9, __port(__index(0), __identifier("INPUT_INTEGRAL_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LINE_INDEX"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("WINDOW_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(20+1, 32)), __port(__index(3), __identifier("INPUT_CASCADE_SIZES_AND_SKIP"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(3, 1)), __port(__index(4), __identifier("INPUT_CASCADE_FEATURES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(10000, 1)), __port(__index(5), __identifier("INPUT_CASCADE_STAGES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(200, 1)), __port(__index(6), __identifier("INPUT_PIXEL_SHIFTS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(7), __identifier("INPUT_PIXEL_OFFSETS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(8), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void lbp_cascade (vec08u ∗apOut, vec32u ∗apInII, int aOutStride, int aInStride, int aTileWidth, int aTileHeight, int16u aLineIndex, vec32u ∗apWindowBuffer, const APEX_lbpFeature ∗apcFeatures, int aStageCount, const APEX_lbpStage ∗apcStages, const int08u ∗apcXshifts, const int08u ∗apcXoffsets, int skipOdd)

    Local Binary Pattern object detection.

Variables

- const int **STAGE_FRACTIONAL_BITS** = 28
- const int **STAGE_FIXED_POINT_MULTIPLIER** = (1 << STAGE_FRACTIONAL_BITS)

5.44.2 Function Documentation

5.44.2.1 KERNEL_INFO apu_lbp_cascade (" apu_lbp_cascade " , 9 , __port(__index(0), __identifier("INPUT_INTEGRAL_IMA↩GE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("LINE_INDEX"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("WINDOW_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(20+1, 32)) , __port(__index(3), __identifier("INPUT_CASCADE_SIZES_AND_SKIP"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(3, 1)) , __port(__index(4), __identifier("INPUT_CASCADE_FEATURES"), __attributes(ACF_ATTR_↩SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(10000, 1)) , __port(__index(5), __identifier("INPUT_CASCADE_STAGES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(200, 1)) , __port(__index(6), __identifier("INPUT_PIXEL_SHIFTS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(64, 1)) , __port(__index(7), __identifier("INPUT_PIXEL_OFFSETS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(64, 1)) , __port(__index(8), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

LBP cascade object detection kernel metadata.

Parameters

LBP_CASC_KN	Define for Kernel name
9	Number of ports
Port LBP_CASC_I↩N_INTEGR_IMG	Define for name of input integral image (unsigned 32bit)
Port LBP_CAS↩C_LINE_IDX	Define for name of line index buffer (unsigned 16bit)
Port LBP_CAS↩C_WND_BUF	Define for name of integral image window buffer (unsigned 32bit)
Port LBP_CAS↩C_IN_CASC_↩SZ_AND_SKIP	Define for name of number of features and stages + whetherthe odd rows and columns should be skipped (3 16-bit values) (unsigned 16bit)
Port LBP_CASC_I↩N_CASC_FEAT	Define for name of array of Haar-like features (unsigned 8bit)
Port LBP_CAS↩C_IN_CASC_↩STAGES	Define for name of array of cascade stages (unsigned 8bit)
Port LBP_CASC_I↩N_PIX_SHFT	Define for name of LUT containing tile shifts needed for horizontal pixel offsets (unsigned 8bit)
Port LBP_CASC_I↩N_PIX_OFFS	Define for name of LUT containing in-tile offsets needed for horizontal pixel offsets (unsigned 8bit)

Port LBP_CAS↩ C_OUT	Define for name of output image (unsigned 8bit)

5.44.2.2 void lbp_cascade (vec08u ∗ *apOut,* vec32u ∗ *apInII,* int *aOutStride,* int *aInStride,* int *aTileWidth,* int *aTileHeight,* int16u *aLineIndex,* vec32u ∗ *apWindowBuffer,* const APEX_lbpFeature ∗ *apcFeatures,* int *aStageCount,* const APEX_lbpStage ∗ *apcStages,* const int08u ∗ *apcXshifts,* const int08u ∗ *apcXoffsets,* int *skipOdd*)

Local Binary Pattern object detection.

Detects objects using local binary pattern feature cascades.

The algorithm searches for 24x24-pixel objects using a local binary pattern (LBP) classifier provided by the user. For each input pixel, it outputs 255 if the pixel is a lower left corner of an object and 0 otherwise. The classifier consists of a number of stages, each stage contains multiple LBP features. The window has to pass every stage in order to be classified as containing an object.

See Shengcai Liao, Xiangxin Zhu, Zhen Lei, Lun Zhang and Stan Z. Li. Learning Multi-scale Block Local Binary Patterns for Face Recognition
http://www.nlpr.labs.gov.cn/2007papers/gjhy/gh98.pdf
The format of the classifier stage is:

```
struct APEX_lbpStage
{
  uint8_t featureCount;
  int32s threshold;
};
```

Where featureCount is a number of LBP features included in the stage.
threshold is a fixed point threshold (28 fractional bits) which has to be exceeded by the sum of the feature test results in order for the stage to pass.

An LBP feature is a rectangle subdivided into a uniform 3x3 grid. Pixel values in each of the 9 sub-areas are summed. The sums of the outer areas are compared to the sum of the central area resulting in a 8-bit code (starting in the upper left area and going clockwise). This code is then used as an index to 256-bit table and the resulting bit tells whether the feature passed or not.

```
struct APEX_lbpFeature
{
  int32_t values[8];
  int32s leafValuesFixed[2];
  uint8_t x, y, width, height;
};
```

values[8] is a 256-bit classifying table used to determine whether the LBP feature with a given code should pass
Where threshold is a fixed-point number (13 fractional bits) which has to be exceeded by the weighted sum of the feature rectangles in order for the feature to pass.
leafValuesFixed are fixed-point values (28 fractional bits), leafValuesFixed[0] is added to the stage sum if the feature passes, leafValuesFixed[1] is added to the stage sum if the feature doesn't pass.
x, y define the position of the feature's upper left corner
width, height define the size of the feature

The kernel expects an integral image. The implementation retains a rolling window for the integral image, therefore sufficiently sized buffer has to be supplied.

Parameters

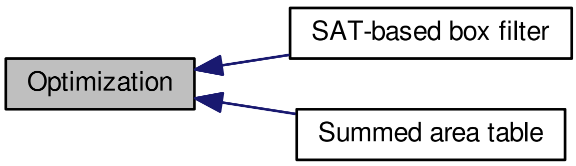
apOut	- [Output] pointer to the destination buffer

apInII	- [Input] pointer to the source integral image buffer
aOutStride	- [Input] line stride of the destination data
aInStride	- [Input] line stride of the source data
aTileWidth	- [Input] width of one data tile
aTileHeight	- [Input] height of one data tile
aLineIndex	- [Input] index of currently processed line
apWindowBuffer	- [Buffer] pointer to the input integral image window buffer
apcFeatures	- [Input] pointer to the LBP feature array
aStageCount	- [Input] number of LBP feature cascade stages
apcStages	- [Input] pointer to the LBP feature cascade stages
apcXshifts	- [Input] pointer to LUT containing tile shifts needed for horizontal pixel offsets
apcXoffsets	- [Input] pointer to LUT containing in-tile offsets needed for horizontal pixel offsets
skipOdd	- [Input] 1 if even columns and rows should be skipped, 0 otherwise

5.45 Optimization

5.45.1 Detailed Description

Collaboration diagram for Optimization:



Modules

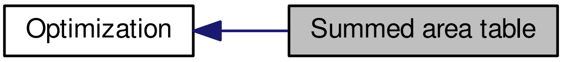
- Summed area table
  Summed area table (integral image) computation.
- SAT-based box filter
  SAT-based box filter.

5.46 Summed area table

5.46.1 Detailed Description

Summed area table (integral image) computation.

Collaboration diagram for Summed area table:



Functions

- KERNEL_INFO apu_sat (" apu_sat ", 3, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_ROW"), __attributes(ACF_ATTR_VEC_OUT_STATIC), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- void sat32 (vec32u ∗apDest, vec32u ∗apPrevRow, const vec08u ∗apcSrc, int aSourceStride, int aDestinationStride, int aBlockWidth, int aBlockHeight, int08u aFirstTile)

    Summed area table.

5.46.2 Function Documentation

5.46.2.1 KERNEL_INFO apu_sat (" apu_sat " , 3 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_ROW"), __attributes(ACF_ATTR_VEC_OUT_STATIC), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Summed area table kernel metadata.

Parameters

SAT_KN	Define for Kernel name
3	Number of ports
Port SAT_KN_IN	Define for name of input image (unsigned 8bit)
Port SAT_KN_OUT	Define for name of output image (unsigned 32bit)
Port SAT_KN_OUT_ROW	Define for name of last row of previous tile buffer (unsigned 32bit)

5.46.2.2 void sat32 (vec32u ∗ *apDest,* vec32u ∗ *apPrevRow,* const vec08u ∗ *apcSrc,* int *aSourceStride,* int *aDestinationStride,* int *aBlockWidth,* int *aBlockHeight,* int08u *aFirstTile*)

Summed area table.

Computes summed area table (integral image).

out(i,j) = sum(in(k,l)) where k,l := (0,0) to (i,j) inclusive

Parameters

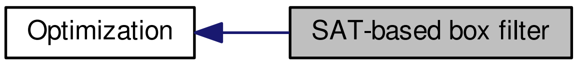
apDest	- [Output] 32bit the integral image of the input
apPrevRow	- [Output] 32bit the last integral row in the tile.
apcSrc	- [Input] 8bit source block pointer
aSourceStride	- [Input] Source block width (in bytes) including padding
aDestination← Stride-	[Input] Destination block width (in bytes) including padding
aBlockWidth	- [Input] Width of one data tile
aBlockHeight	- [Input] Height of one data tile
aFirstTile	- [Input]Boolean. True, if the first tile is computed.

5.47 SAT-based box filter

5.47.1 Detailed Description

SAT-based box filter.

Collaboration diagram for SAT-based box filter:



Functions

- KERNEL_INFO apu_sat_box_filter (" apu_sat_box_filter ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(BOX_SIZE+1, BOX_SIZE, BOX_SIZE+1, BOX_SIZE), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void sat_box_filter_impl (vec08u ∗apDest, const vec32u ∗apcSrc, int aBlockWidth, int aBlockHeight, int aSourceStride, int aDestStride)

    Sum of values over one patch the input image is a SAT image (i.e. integral computed with the sat32() function.

Variables

- const int **BOX_SIZE**
- const int **BOX_AREA**
- const int **BOX_SIZE**
- const int **BOX_AREA**

5.47.2 Function Documentation

5.47.2.1 KERNEL_INFO apu_sat_box_filter (" apu_sat_box_filter " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(BOX_SIZE+1, BOX_SIZE, BOX_SIZE+1, BOX_SIZE), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Box filter using summed area table kernel metadata.

Parameters

SAT_BOX_FILTER_KN	Define for Kernel name
2	Number of ports
Port SAT_BOX_FILTER_IN	Define for name of input summed area table image (unsigned 32bit)

Port SAT_BOX↩_FILTER_OUT	Define for name of output image (unsigned 8bit)

5.47.2.2 void sat_box_filter_impl (vec08u ∗ *apDest*, const vec32u ∗ *apcSrc*, int *aBlockWidth*, int *aBlockHeight*, int *aSourceStride*, int *aDestStride*)

Sum of values over one patch the input image is a SAT image (i.e. integral computed with the sat32() function.

Applies a box filter (== sum over that patch) to the image using its summed area table (integral image).

out(i,j) = in(i - box_size - 1, j - box_size - 1) + in(i + box_size, j + box_size) - in(i - box_size - 1, j + box_size) - in(i + box_size, j - box_size - 1)

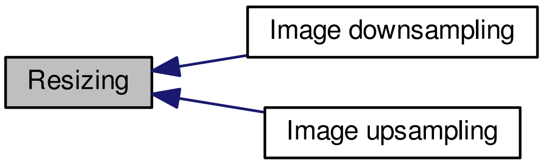
Parameters

apDest	- [Output] 8bit destination image containing sum of values over one patch
apcSrc	- [Input] 32bit source block pointer
aBlockWidth	- [Input] Block width
aBlockHeight	- [Input] Block height
aSourceStride	- [Input] Source block width (in bytes) including padding
aDestination↩Stride-	[Input] Destination block width (in bytes) including padding

5.48 Resizing

5.48.1 Detailed Description

Collaboration diagram for Resizing:



Modules

- Image downsampling

  Image downsampling.
- Image upsampling

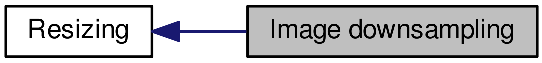
  Image upsampling.

5.49 Image downsampling

5.49.1 Detailed Description

Image downsampling.

Collaboration diagram for Image downsampling:



Functions

- KERNEL_INFO apu_downsample (" apu_downsample ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_downsample_16u (" apu_downsample_16u ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_downsample_gauss (" apu_downsample_gauss ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- void downsample (vec08u ∗apDest, const vec08u ∗apcSrc, int aOutBlockWidth, int aOutBlockHeight, int aInBlockStride, int aOutBlockStride)

  x2 downsampling.
- void downsample_16u (vec16u ∗apDest, const vec16u ∗apcSrc, int aInBlockWidth, int aInBlockHeight, int aOutBlockWidth, int aOutBlockHeight, int aInBlockStride, int aOutBlockStride)

  x2 downsampling, 16-bit.
- void downsample_gauss (vec08u ∗apDest, const vec08u ∗apcSrc, int32s aOutBlockWidth, int32s aOutBlockHeight, int32s aInBlockStride, int32s aOutBlockStride)

  x2 downsampling using Gaussian blur.

5.49.2 Function Documentation

5.49.2.1 KERNEL_INFO apu_downsample (" apu_downsample " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Image 2x downsampling kernel metadata.

Parameters

DOWNSAMPL←E_KN	Define for Kernel name
2	Number of ports
Port DOWNSA←MPLE_KN_IN	Define for name of input image (unsigned 8bit)
Port DOWNSAMPL←E_KN_OUT	Define for name of output image (unsigned 8bit)

5.49.2.2 KERNEL_INFO apu_downsample_16u (" apu_downsample_16u " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

Image 2x downsampling kernel (16-bit version) metadata.

Parameters

DOWNSAMPL←E_16u_KN	Define for Kernel name
2	Number of ports
Port DOWNSA←MPLE_KN_IN	Define for name of input image (unsigned 16bit)
Port DOWNSAMPL←E_KN_OUT	Define for name of output image (unsigned 16bit)

5.49.2.3 KERNEL_INFO apu_downsample_gauss (" apu_downsample_gauss " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Image 2x downsampling using Gaussian blur kernel metadata.

Parameters

DOWNSAMPL←E_GAUSS_KN	Define for Kernel name
2	Number of ports
Port DOWNSA←MPLE_KN_IN	Define for name of input image (unsigned 8bit)
Port DOWNSAMPL←E_KN_OUT	Define for name of output image (unsigned 8bit)

5.49.2.4 void downsample (vec08u ∗ *apDest*, const vec08u ∗ *apcSrc*, int *aOutBlockWidth*, int *aOutBlockHeight*, int *aInBlockStride*, int *aOutBlockStride*)

x2 downsampling.

Downsamples the image by two.

Parameters

apDest	- [Output] pointer to the destination buffer
apcSrc	- [Input] pointer to the source buffer
aOutBlockWidth	- [Input] width of one output data tile
aOutBlockHeight	- [Input] height of one output data tile
aInBlockStride	- [Input] line stride of the source data
aOutBlockStride	- [Input] line stride of the destination data

5.49.2.5 void downsample_16u (vec16u ∗ *apDest*, const vec16u ∗ *apcSrc*, int *aInBlockWidth*, int *aInBlockHeight*, int *aOutBlockWidth*, int *aOutBlockHeight*, int *aInBlockStride*, int *aOutBlockStride*)

x2 downsampling, 16-bit.

Downsamples the image by two. 16-bit version

Parameters

apDest	- [Output] pointer to the destination buffer
apcSrc	- [Input] pointer to the source buffer
aOutBlockWidth	- [Input] width of one output data tile
aOutBlockHeight	- [Input] height of one output data tile
aInBlockStride	- [Input] line stride of the source data
aOutBlockStride	- [Input] line stride of the destination data

5.49.2.6 void downsample_gauss (vec08u ∗ *apDest*, const vec08u ∗ *apcSrc*, int32s *aOutBlockWidth*, int32s *aOutBlockHeight*, int32s *aInBlockStride*, int32s *aOutBlockStride*)

x2 downsampling using Gaussian blur.

Downsamples the image by two using Gaussian blur.

Parameters

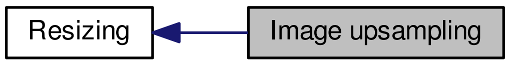
apDest	- [Output] pointer to the destination buffer
apcSrc	- [Input] pointer to the source buffer
aOutBlockWidth	- [Input] width of one output data tile
aOutBlockHeight	- [Input] height of one output data tile
aInBlockStride	- [Input] line stride of the source data
aOutBlockStride	- [Input] line stride of the destination data

5.50 Image upsampling

5.50.1 Detailed Description

Image upsampling.

Collaboration diagram for Image upsampling:



Functions

- KERNEL_INFO apu_upsample (" apu_upsample ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)))
- void upsample (vec08u ∗apDest, const vec08u ∗apcSrc, int aInBlockWidth, int aInBlockHeight, int aInBlockStride, int aOutBlockStride)

  x2 upsampling.

5.50.2 Function Documentation

5.50.2.1 KERNEL_INFO apu_upsample (" apu_upsample " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)))

Image 2x upsampling kernel metadata.

Parameters

UPSAMPLE_KN	Define for Kernel name
2	Number of ports
Port UPSAMPLE_KN_IN	Define for name of input image (unsigned 8bit)
Port UPSAMPLE_KN_OUT	Define for name of output image (unsigned 8bit)

5.50.2.2 void upsample (vec08u ∗ *apDest*, const vec08u ∗ *apcSrc*, int *aInBlockWidth*, int *aInBlockHeight*, int *aInBlockStride*, int *aOutBlockStride*)

x2 upsampling.

Upsamples the image by two.

Parameters

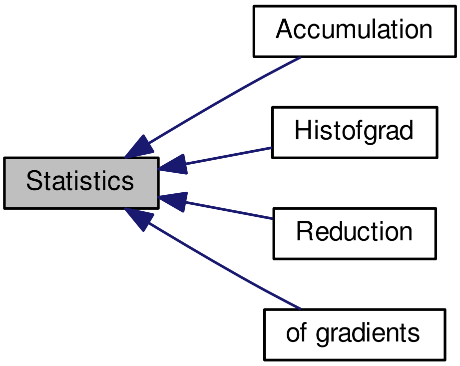
apDest	- [Output] pointer to the destination buffer
apcSrc	- [Input] pointer to the source buffer
aInBlockWidth	- [Input] width of one input data tile
aInBlockHeight	- [Input] height of one input data tile
aInBlockStride	- [Input] line stride of the source data
aOutBlockStride	- [Input] line stride of the destination data

5.51 Statistics

5.51.1 Detailed Description

sum of columns thru each row

Collaboration diagram for Statistics:



Modules

- Accumulation

  Element value accumulation.
- Histofgrad

  Histogram of Gradient.
- of gradients

  Histogram.
- Reduction

  Vector to scalar reduction.

Functions

- KERNEL_INFO apu_columns_sum (" apu_columns_sum ", 6, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 10)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(2), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("INPUT_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_2"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("INPUT_3"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

5.51.2 Function Documentation

5.51.2.1 KERNEL_INFO apu_columns_sum (" apu_columns_sum " , 6 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 10)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(2), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(3), __identifier("INPUT_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("INPUT_2"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(5), __identifier("INPUT_3"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Histogram kernel metadata.

Parameters

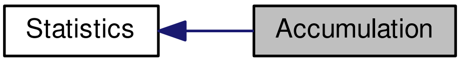
HISTOGRAM_↩ KN	Define for Kernel name
2	Number of ports
Port HISTOGR↩ AM_KN_IN	Define for name of input image (unsigned 8bit)
Port HISTOGR↩ AM_KN_OUT	Define for name of histogram vector output, CU count X 256 (unsigned 32bit)

5.52 Accumulation

5.52.1 Detailed Description

Element value accumulation.

Collaboration diagram for Accumulation:



Macros

- #define **ACCUM_TILE_SIZE_X** 10
- #define **ACCUM_TILE_SIZE_Y** 10

Functions

- KERNEL_INFO apu_accumulation (" apu_accumulation ", 6, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(10, 10)), __port(__index(1), __identifier("Output_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("ACCUM_XOFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("ACCUM_YOFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("ACCUM_XWIDTH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("ACCUM_YHEIGHT"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- void accumulation_in32s_filter (vec32s ∗dst, vec32s ∗srcA, int16s sstr, int16s xOffs, int16s yOffs, int16s xAccWidth, int16s yAccHeight)

  Accumulates all values in a chunk (signed 32bit).
- void accumulation_in32u_filter (vec32u ∗lpvOut, vec32u ∗lpvIn, int16s strideWidth, int16s xOffs, int16s yOffs, int16s xAccWidth, int16s yAccHeight)

  Accumulates all values in a chunk (unsigned 32bit).

5.52.2 Function Documentation

5.52.2.1 void accumulation_in32s_filter (vec32s ∗ *dst*, vec32s ∗ *srcA*, int16s *sstr*, int16s *xOffs*, int16s *yOffs*, int16s *xAccWidth*, int16s *yAccHeight*)

Accumulates all values in a chunk (signed 32bit).

Accumulates the sum of all signed 32bit values in a chunk. Input is a vector/matrix, output is one sum-value

Parameters

dst	- [Output] pointer to output accumulation value
srcA	- [Input] Source block pointer of img A
sstr	- [Input] Source block width in elements (including padding)
xOffs	- [Input] X Offset where to start accumulation
yOffs	- [Input] Y Offset where to start accumulation
xAccWidth	- [Input] Width inside block for which accumulation has to be performed
yAccHeight	- [Input] Height inside block for which accumulation has to be performed

5.52.2.2 void accumulation_in32u_filter (vec32u ∗ *lpvOut,* vec32u ∗ *lpvIn,* int16s *strideWidth,* int16s *xOffs,* int16s *yOffs,* int16s *xAccWidth,* int16s *yAccHeight*)

Accumulates all values in a chunk (unsigned 32bit).

Accumulates the sum of all unsigned 32bit values in a chunk. Input is a vector/matrix, output is one sum-value

Parameters

dst	- [Output] pointer to output accumulation value
srcA	- [Input] Source block pointer of img A
sstr	- [Input] Source block width in elements (including padding)
xOffs	- [Input] X Offset where to start accumulation
yOffs	- [Input] Y Offset where to start accumulation
xAccWidth	- [Input] Width inside block for which accumulation has to be performed
yAccHeight	- [Input] Height inside block for which accumulation has to be performed

5.52.2.3 KERNEL_INFO apu_accumulation (" apu_accumulation " , 6 , __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(10, 10)) , __port(__index(1), __identifier("Output_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("ACCUM_XOFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("ACCUM_YOFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("ACCUM_XWIDTH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(5), __identifier("ACCUM_YHEIGHT"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

Accumulation kernel metadata. Accumulates the pixels in one a certain neighborhood (XWIDTH, YHEIGHT) of signed 32bit image according to the chunk related xoffs and yoffs Outputs 16bit signed interpolation vector having as many elements as number of chunks .

the accumulation is not crossing the chunk limits.

Parameters

ACCUM_KN	Define for Kernel name
6	Number of ports
Port ACCUM_IN	Define for name offirst input image (signed 32bit)
Port ACCUM_OUT	Define for name of interpolation result of the input image (signed 32bit).
Port ACCUML←_XOFFS	Define for name of signed 16bit x offset vector (has one element for each chunk of the image) (signed 16bit)

Port ACCUM_↩ YOFFS	Define for name of signed 16bit y offset vector (has one element for each chunk of the image) (signed 16bit)
Port ACCUM_↩ XWIDTH	Define for name of scalar value defining the accumulation width (has to be defined by the user) (signed 16bit)
Port ACCUM_↩ YHEIGHT	Define for name of scalar value defining the accumulation height (has to be defined by the user) (signed 16bit)

5.53 Column_sum

5.53.1 Detailed Description

Functions

- void column_sum (vec08u ∗lpvIn0, vec32u ∗lpvOutDown, vec32u ∗lpvOutUp, bool isFirstTile, int lStrideIn0, int chunkWidth, int chunkHeight, int outChunkWidth, int priorityDown, int priorityUp, int indexOfTileStart)

  Elementwise unsigned 8bit addition => unsigned 16bit.

5.53.2 Function Documentation

5.53.2.1 void column_sum (vec08u ∗ *lpvIn0,* vec32u ∗ *lpvOutDown,* vec32u ∗ *lpvOutUp,* bool *isFirstTile,* int *lStrideIn0,* int *chunkWidth,* int *chunkHeight,* int *outChunkWidth,* int *priorityDown,* int *priorityUp,* int *indexOfTileStart*)

Elementwise unsigned 8bit addition => unsigned 16bit.

Parameters

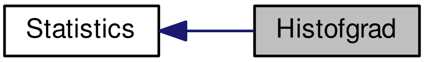
lpvIn0	- [Input] 8bit source block pointer of img 0
lpvOutDown	- [Output] 32 unsigned bit destination block pointer
lpvOutUp	- [Output] 32 unsigned bit destination block pointer
isFirstTile	- [Input] boolean: true if the algorithm is at its first tile
lStride0	- [Input] Source block width (in bytes) including padding
chunkWidth	- [Input] Block width
chunkHeight	- [Input] Block height
outChunkWidth	- [Input] Block width
priorityDown	- [Input] Priority for lower part of input image
priorityUp	- [Input] Priority for upper part of input image
indexOfTileStart	- [Input] Index of tile start

5.54 Histofgrad

5.54.1 Detailed Description

Histogram of Gradient.

Collaboration diagram for Histofgrad:



Functions

- KERNEL_INFO apu_histofgrad (" apu_histofgrad ", 4, __port(__index(0), __identifier("HOG_InGradX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 4, 0, 4), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(4, 4)), __port(__index(1), __identifier("HOG_InGradY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 4, 0, 4), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(4, 4)), __port(__index(2), __identifier("HOG_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size((16 ∗4 ∗1), 1), __ek_size(1, 1)), __port(__index(3), __identifier("HOG_OUT_BlNorm"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

5.54.2 Function Documentation

5.54.2.1 KERNEL_INFO apu_histofgrad (" apu_histofgrad " , 4 , __port(__index(0), __identifier("HOG_InGradX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 4, 0, 4), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(4, 4)) , __port(__index(1), __identifier("HOG_InGradY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 4, 0, 4), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(4, 4)) , __port(__index(2), __identifier("HOG_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size((16 ∗4 ∗1), 1), __ek_size(1, 1)) , __port(__index(3), __identifier("HOG_OUT_BlNorm"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

Histogram of Gradient kernel metadata.

Parameters

HISTOFGRAD_KN	Define for Kernel name
3	Number of ports
Port HOG_KN_INGradX	Define for name of input gradientX image (signed 32bit)
Port HOG_KN_INGradX	Define for name of input gradientY image (signed 32bit)
Port HOG_KN_OUT	Define for name of histofgrad vector output, CU count X 64 bins of 4bits each (i.e. 2 x unsigned 32bit)

5.55 of gradients

5.55.1 Detailed Description

Histogram.

Collaboration diagram for of gradients:

Macros

- #define **HOG_NR_FEATURES_PER_HISTO** 16
- #define **HOG_NR_SCALES** 1
- #define **HOG_NR_OVERLAPPING_WINDOWS** 4
- #define **HOG_NR_FEATURES_PER_BOX** (HOG_NR_FEATURES_PER_HISTO ∗ HOG_NR_OVERLAP↩ PING_WINDOWS ∗ HOG_NR_SCALES)
- #define **HOG_OVERLAP** 2
- #define **HOG_LAT_DEPENDENCY** 4
- #define **HOG_WND_SZ** 4

Functions

- void hog (vec08s ∗lpvInGradX, vec08s ∗lpvInGradY, vec16u ∗lpvOut, vec32u ∗lpvOutBlNorm, int lStrideIn, int chunkWidth, int chunkHeight, int lStrideOut)

  Elementwise unsigned 8bit addition => unsigned 16bit.
- KERNEL_INFO apu_histogram (" apu_histogram ", 2, __port(__index(0), __identifier("INPUT_0"), __↩ attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_↩ size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_STATIC_F↩ IXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)))
- void hist (vec08u ∗lpvIn0, vec32u ∗lpvOut0, bool isFirstTile, int lStrideIn0, int chunkWidth, int chunkHeight, int outChunkWidth)

  Elementwise unsigned 8bit addition => unsigned 16bit.

5.55.2 Function Documentation

5.55.2.1 KERNEL_INFO apu_histogram (" apu_histogram " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)))

Histogram kernel metadata.

Parameters

HISTOGRAM_↩ KN	Define for Kernel name
2	Number of ports
Port HISTOGR↩ AM_KN_IN	Define for name of input image (unsigned 8bit)
Port HISTOGR↩ AM_KN_OUT	Define for name of histogram vector output, CU count X 256 (unsigned 32bit)

5.55.2.2 void hist (vec08u ∗ *lpvIn0,* vec32u ∗ *lpvOut0,* bool *isFirstTile,* int *lStrideIn0,* int *chunkWidth,* int *chunkHeight,* int *outChunkWidth*)

Elementwise unsigned 8bit addition => unsigned 16bit.

Histogram computation of an input image

Parameters

lpvIn0	- [Input] 8bit source block pointer of img 0
lpvOut0	- [Output] 32 unsigned bit destination block pointer
isFirstTile	- [Input] boolean: true if the algorithm is at its first tile
lStride0	- [Input] Source block width (in bytes) including padding
chunkWidth	- [Input] Block width
chunkHeight	- [Input] Block height

5.55.2.3 void hog (vec08s ∗ *lpvInGradX,* vec08s ∗ *lpvInGradY,* vec16u ∗ *lpvOut,* vec32u ∗ *lpvOutBlNorm,* int *lStrideIn,* int *chunkWidth,* int *chunkHeight,* int *lStrideOut*)

Elementwise unsigned 8bit addition => unsigned 16bit.

Histogram computation of an input image

Parameters

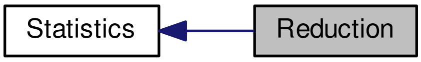
lpvInGradX	- [Input] signed integral image of gradient X
lpvInGradY	- [Input] signed integral image of gradient Y
lpvOut	- [Output] histogram of gradients of 4x4 blocks on 2 different scales stored on 4 32bit words
isFirstTile	- [Input] boolean: true if the algorithm is at its first tile
lStrideIn	- [Input] Source block width (in elements) including padding
chunkWidth	- [Input] Block width
chunkHeight	- [Input] Block height
lStrideOut	- [Input] Destination block width (in elements) including padding

5.56 Reduction

5.56.1 Detailed Description

Vector to scalar reduction.

Collaboration diagram for Reduction:



Functions

- KERNEL_INFO apu_reduction (" apu_reduction ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)))
- void reduc (vec32u ∗lpvIn0, int32s ∗lpOut0, bool isLastTile, int16s lFirstCuId, int16s lTileWidthInChunks, int lChunkWidth, int lChunkHeight, int lChunkSpanIn0, int lChunkSpanOut0)

    Elementwise unsigned 8bit addition => unsigned 16bit.
- KERNEL_INFO apu_reduction_for_clmn (" apu_reduction_for_clmn ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(192, 1)), __port(__index(3), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(192, 1)))
- void reduc (vec32u ∗lpvIn0, vec32u ∗lpvIn1, int32s ∗lpOut0, int32s ∗lpOut1, bool isLastTile, int16s lFirstCuId, int16s lTileWidthInChunks, int lChunkWidth, int lChunkHeight, int lChunkSpanIn0, int lChunkSpanOut0)

    Elementwise unsigned 8bit addition => unsigned 16bit.

5.56.2 Function Documentation

5.56.2.1 KERNEL_INFO apu_reduction (" apu_reduction " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)))

256 vectors to 256 scalars accumulation reduction.

Parameters

REDUCTION_KN	Define for Kernel name

2	Number of ports
Port REDUCTI↩ ON_KN_IN	Define for name of vector input (unsigned 32bit)
Port REDUCTI↩ ON_KN_OUT	Define for name of scalar output (unsigned 32bit)

5.56.2.2 KERNEL_INFO apu_reduction_for_clmn (" apu_reduction_for_clmn " , 4 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(192, 1)) , __port(__index(3), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(192, 1)))

Parameters

REDUCTION_↩ KN	Define for Kernel name
2	Number of ports
Port REDUCTI↩ ON_KN_IN	Define for name of vector input (unsigned 32bit)
Port REDUCTI↩ ON_KN_OUT	Define for name of scalar output (unsigned 32bit)

5.56.2.3 void reduc (vec32u ∗ *lpvIn0,* vec32u ∗ *lpvIn1,* int32s ∗ *lpOut0,* int32s ∗ *lpOut1,* bool *isLastTile,* int16s *lFirstCuId,* int16s *lTileWidthInChunks,* int *lChunkWidth,* int *lChunkHeight,* int *lChunkSpanIn0,* int *lChunkSpanOut0*)

Elementwise unsigned 8bit addition => unsigned 16bit.

Reduce an input vector/image by summing up the corresponding elements

Parameters

lpvIn0	- [Input] 32unsigned bit source block pointer of img 0
lpOut0	- [Output] 32signed bit destination block pointer
isLastTile	- [Input] boolean: is true, if the last image tile is being processed
lFirstCuId	- [Input] the id of the first CU
lTileWidthIn↩ Chunks	- [Input] number of used chunks/CUs
chunkWidth	- [Input] Block width
chunkHeight	- [Input] Block height
lChunkSpanIn0-	[Input] Source block width (in bytes) including padding
lChunkSpan↩ Out0-	[Input] Destination block width (in bytes) including padding

5.56.2.4 void reduc (vec32u ∗ *lpvIn0,* int32s ∗ *lpOut0,* bool *isLastTile,* int16s *lFirstCuId,* int16s *lTileWidthInChunks,* int *lChunkWidth,* int *lChunkHeight,* int *lChunkSpanIn0,* int *lChunkSpanOut0*)

Elementwise unsigned 8bit addition => unsigned 16bit.

Reduce an input vector/image by summing up the corresponding elements

Parameters

lpvIn0	- [Input] 32unsigned bit source block pointer of img 0
lpOut0	- [Output] 32signed bit destination block pointer
isLastTile	- [Input] boolean: is true, if the last image tile is being processed
lFirstCuId	- [Input] the id of the first CU
lTileWidthIn←Chunks	- [Input] number of used chunks/CUs
chunkWidth	- [Input] Block width
chunkHeight	- [Input] Block height
lChunkSpanIn0-	[Input] Source block width (in bytes) including padding
lChunkSpan←Out0-	[Input] Destination block width (in bytes) including padding

Chapter 6

Class Documentation

6.1 APEX_HaarCascadeFeature Struct Reference

Public Attributes

- int16u **threshold**
- int16u **leftVal**
- int16u **rightVal**
- int16u **x5y5w5_1**
- int08u **h5type3**

The documentation for this struct was generated from the following file:

- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩
sdk/kernels/apu/sample_object_detection_kernels/src/haar_cascade_apu.h

6.2 APEX_HaarCascadeStage Struct Reference

Public Attributes

- int08u **featureCount**
- int16u **thresholdFixed**

The documentation for this struct was generated from the following file:

- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩
sdk/kernels/apu/sample_object_detection_kernels/src/haar_cascade_apu.h

6.3 APEX_lbpFeature Struct Reference

Public Attributes

- int32_t **values** [8]
- STAGE_FIXED_POINT_TYPE_SCALAR **leafValuesFixed** [2]
- uint8_t **x**
- uint8_t **y**
- uint8_t **width**

- uint8_t **height**

The documentation for this struct was generated from the following file:

- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩
  sdk/kernels/apu/sample_object_detection_kernels/src/lbp_cascade_apu.h

6.4 APEX_lbpStage Struct Reference

Public Attributes

- uint8_t **featureCount**
- STAGE_FIXED_POINT_TYPE_SCALAR **threshold**

The documentation for this struct was generated from the following file:

- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩
  sdk/kernels/apu/sample_object_detection_kernels/src/lbp_cascade_apu.h

6.5 point_t Struct Reference

Public Attributes

- int **x**
- int **y**

The documentation for this struct was generated from the following file:

- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩
  sdk/kernels/apu/apexcv_pro_remap/src/common_types.h

6.6 rect_t Struct Reference

Public Attributes

- int16u **left**
- int16u **top**
- int16u **width**
- int16u **height**

The documentation for this struct was generated from the following file:

- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩
  sdk/kernels/apu/apexcv_pro_remap/src/common_types.h

6.7 RESIZE_DESCRIPTOR Struct Reference

Struct for Resize descriptor.

```
#include <resize_definitions_apu.h>
```

Public Attributes

- int32_t **dst_bh**
- int32_t dst_bw
- int32_t **src_current_line**
- int32_t unused
- int32_t **src_offs**
- int32_t scl_fact
- int32_t **phases**
- int32_t taps
- int32_t **out_scale**
- int32_t out_round
- const int16_t ∗ fbnk

6.7.1 Detailed Description

Struct for Resize descriptor.

Sizes and Offsets are on vertical direction (Y)

NB: src_current_line, src_offs cleared by a call to get_polyphase_params();

6.7.2 Member Data Documentation

6.7.2.1 int32_t RESIZE_DESCRIPTOR::dst_bw

Destination block width and height

6.7.2.2 const int16_t ∗ RESIZE_DESCRIPTOR::fbnk

Polyphase Filter Bank

6.7.2.3 int32_t RESIZE_DESCRIPTOR::out_round

Values for scaling and rounding

6.7.2.4 int32_t RESIZE_DESCRIPTOR::scl_fact

Source offset - Scale Factor

6.7.2.5 int32_t RESIZE_DESCRIPTOR::taps

Number of taps and phases

6.7.2.6 int32_t RESIZE_DESCRIPTOR::unused

Line number of the top left corner of the source (input) block.

The documentation for this struct was generated from the following files:

- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩ sdk/kernels/apu/apexcv_pro_resize/src/resize_definitions_apu.h
- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩ sdk/kernels/apu/sample_resizing_kernels/src/resize_definitions.h

6.8 resize_descriptor Struct Reference

Public Attributes

- int32_t **src_offs**
- int32_t **scl_fact**
- int16_t **phases**
- int16_t **taps**
- int32_t **out_scale**
- int32_t **out_round**
- int16_t * **fbnk**

The documentation for this struct was generated from the following file:

- /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_↩ sdk/kernels/apu/apexcv_pro_resize/src/resize_definitions.h

Chapter 7

File Documentation

7.1 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_gdc_ldw2/src/cg_kernel.h File Reference

Contains the prototypes for the APU kernels found in cg_kernel.a.

7.1.1 Detailed Description

Contains the prototypes for the APU kernels found in cg_kernel.a.

7.2 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_remap/src/cg_kernel.h File Reference

Contains the prototypes for the APU kernels found in cg_kernel.a.

7.2.1 Detailed Description

Contains the prototypes for the APU kernels found in cg_kernel.a.

7.3 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_resize/src/cg_kernel.h File Reference

Contains the prototypes for the APU kernels found in cg_kernel.a.

7.3.1 Detailed Description

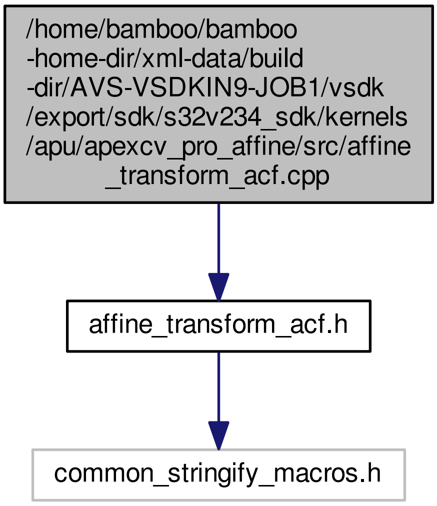
Contains the prototypes for the APU kernels found in cg_kernel.a.

7.4 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_affine/src/affine_transform_acf.cpp File Reference

ACF metadata and wrapper function for the affine transformation.

`#include "affine_transform_acf.h"`

Include dependency graph for affine_transform_acf.cpp:



Functions

- KERNEL_INFO affine_bilinear_interpolate (" affine_bilinear_interpolate ", 5, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MATRIX"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(6, 1), __ek_size(1, 1)), __port(__index(2), __identifier("IMAGE_WIDTH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("IMAGE_HEIGHT"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

  ACF metadata for the bilinear interpolation.

7.4.1 Detailed Description

ACF metadata and wrapper function for the affine transformation.

7.4.2 Function Documentation

7.4.2.1 KERNEL_INFO affine_bilinear_interpolate ("affine_bilinear_interpolate" , 5 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("MATRIX"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(6, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("IMAGE_WIDTH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("IMAGE_HEIGHT"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

ACF metadata for the bilinear interpolation.

See also

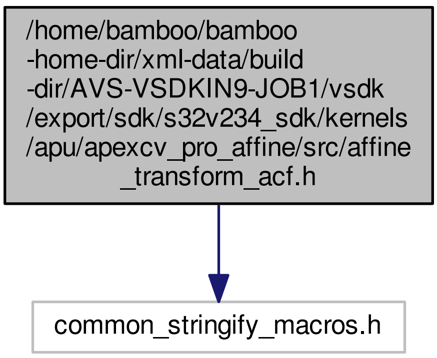
UG-10267-03 ACF User Guide, Section 3.2.2

7.5 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234↩ _sdk/kernels/apu/apexcv_pro_affine/src/affine_transform_acf.h File Reference

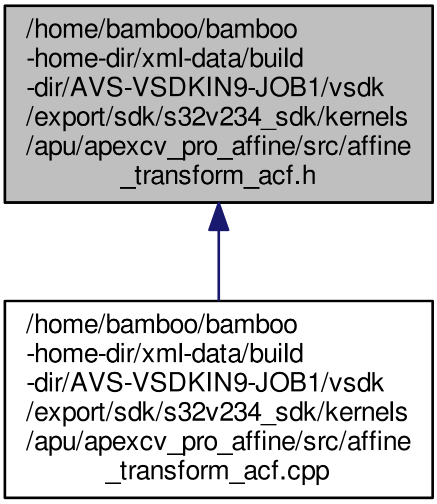
ACF metadata and wrapper function for the affine transformation.

`#include "common_stringify_macros.h"`

Include dependency graph for affine_transform_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **AFFINE_BILINEAR_INTERPOLATE_K** affine_bilinear_interpolate
- #define **AFFINE_BILINEAR_INTERPOLATE_KN** XSTR(AFFINE_BILINEAR_INTERPOLATE_K)
- #define **INPUT** "INPUT"
- #define **MATRIX** "MATRIX"
- #define **IMAGE_WIDTH** "IMAGE_WIDTH"
- #define **IMAGE_HEIGHT** "IMAGE_HEIGHT"
- #define **OUTPUT** "OUTPUT"

Functions

- **extKernelInfoDecl** (affine_bilinear_interpolate)

7.5.1 Detailed Description

ACF metadata and wrapper function for the affine transformation.

7.6 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_affine/src/affine_transform_apu.cpp File Reference

ACF Affine Transform Wrapper.

7.6.1 Detailed Description

ACF Affine Transform Wrapper.

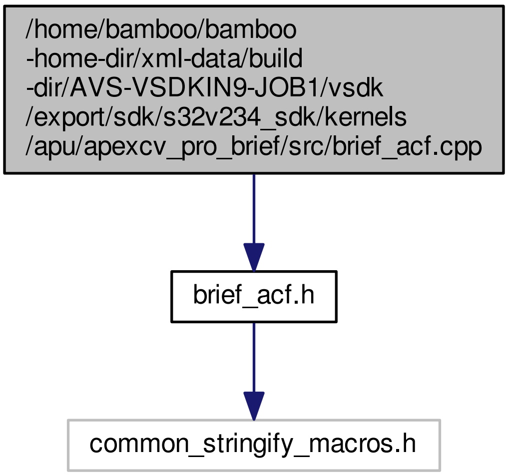
APU affine transform kernel.

7.7 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_brief/src/brief_acf.cpp File Reference

ACF metadata and wrapper function for the BRIEF.

`#include "brief_acf.h"`

Include dependency graph for brief_acf.cpp:



Functions

- KERNEL_INFO compute_brief_descriptor (" compute_brief_descriptor ", 11, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(36, 36)), __port(__index(1), __identifier("FILTER_TYPE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("SMPL_PACKET"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(2048, 1)), __port(__index(3), __identifier("NR_PACKETS_UL"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("NR_PACKETS_UR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("NR_PACKETS_LL"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(6), __identifier("NR_PACKETS_LR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(7), __identifier("PATCH_SIZE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(8), __identifier("DESC_SIZE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(9), __identifier("COUNT"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(10), __identifier("DESCRIPTOR_OUT"), __attributes(ACF_ATTR_VEC_OUT_FIFO_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(64, 1)))

ACF metadata for the computation of BRIEF descriptors.

7.7.1 Detailed Description

ACF metadata and wrapper function for the BRIEF.

7.7.2 Function Documentation

7.7.2.1 KERNEL_INFO compute_brief_descriptor (" compute_brief_descriptor " , 11 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(36, 36)) , __port(__index(1), __identifier("FILTER_TYPE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("SMPL_PACKET"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(2048, 1)) , __port(__index(3), __identifier("NR_PACKETS_UL"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(4), __identifier("NR_PACKETS_UR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(5), __identifier("NR_PACKETS_LL"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(6), __identifier("NR_PACKETS_LR"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(7), __identifier("PATCH_SIZE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(8), __identifier("DESC_SIZE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(9), __identifier("COUNT"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(10), __identifier("DESCRIPTOR_OUT"), __attributes(ACF_ATTR_VEC_OUT_FIFO_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(64, 1)))

ACF metadata for the computation of BRIEF descriptors.

See also

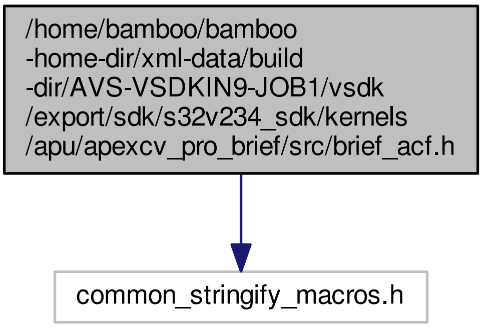
UG-10267-03 ACF User Guide, Section 3.2.2

7.8 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_brief/src/brief_acf.h File Reference

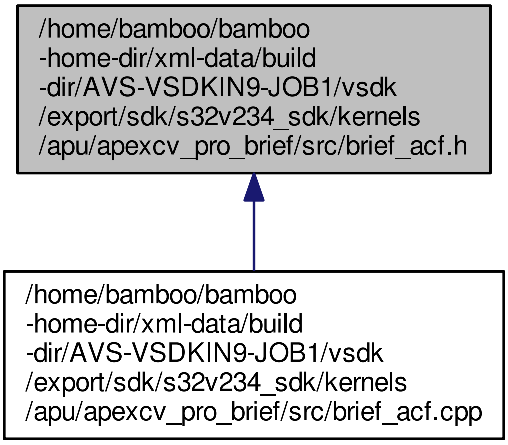
ACF metadata for the BRIEF.

```
#include "common_stringify_macros.h"
```

Include dependency graph for brief_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **COMPUTE_BRIEF_DESCRIPTOR_K** compute_brief_descriptor
- #define **COMPUTE_BRIEF_DESCRIPTOR_KN** XSTR(COMPUTE_BRIEF_DESCRIPTOR_K)
- #define **INPUT** "INPUT"
- #define **FILTER_TYPE** "FILTER_TYPE"
- #define **SMPL_PACKET** "SMPL_PACKET"
- #define **NR_PACKETS_UL** "NR_PACKETS_UL"
- #define **NR_PACKETS_UR** "NR_PACKETS_UR"
- #define **NR_PACKETS_LL** "NR_PACKETS_LL"

- #define **NR_PACKETS_LR** "NR_PACKETS_LR"
- #define **PATCH_SIZE** "PATCH_SIZE"
- #define **DESC_SIZE** "DESC_SIZE"
- #define **COUNT** "COUNT"
- #define **DESCRIPTOR_OUT** "DESCRIPTOR_OUT"

Functions

- **extKernelInfoDecl** (compute_brief_descriptor)

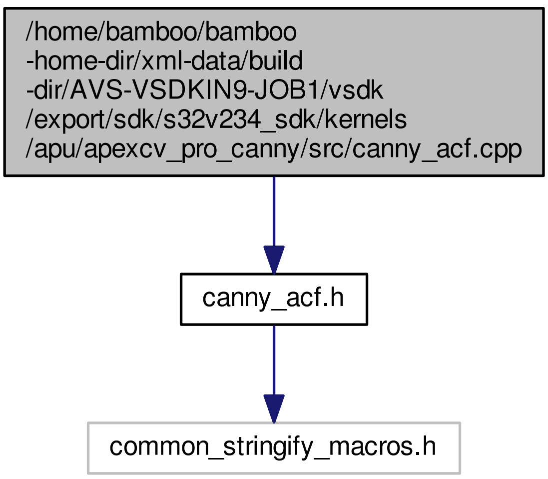
7.8.1 Detailed Description

ACF metadata for the BRIEF.

7.9 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234↩ _sdk/kernels/apu/apexcv_pro_canny/src/canny_acf.cpp File Reference

ACF Metadata and wrapper function for Canny edge detector.

`#include "canny_acf.h"`
Include dependency graph for canny_acf.cpp:



Functions

- KERNEL_INFO canny_non_maxima_suppress (" canny_non_maxima_suppress ", 4, __port(__index(0), ↩ __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(1), __identifier("LOW_THRESH"), __attributes(ACF_A↩ TTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek↩ _size(1, 1)), __port(__index(2), __identifier("HIGH_THRESH"), __attributes(ACF_ATTR_SCL_IN_STATIC↩ _FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩

index(3), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_↩ data_type(d08u), __e0_size(1, 1), __ek_size(2, 1)))

ACF metadata for the non-maxima suppression kernel.

- KERNEL_INFO canny_nms_promote (" canny_nms_promote ", 4, __port(__index(0), __identifier("INPU↩ T"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(1), __identifier("LOW_THRESH"), __attributes(ACF_ATTR_SCL_IN_ST↩ ATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(↩ __index(2), __identifier("HIGH_THRESH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_↩ dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("O↩ UTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_↩ size(1, 1), __ek_size(2, 1)))

  ACF metadata for the non-maxima suppression & edge promotion kernel.

- KERNEL_INFO canny_promote_edges (" canny_promote_edges ", 2, __port(__index(0), __identifier("IN↩ PUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __↩ spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)))

  ACF metadata for the edge promotion kernel.

- KERNEL_INFO canny_promote_edges_full (" canny_promote_edges_full ", 2, __port(__index(0), __↩ identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_V↩ EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)))

  ACF metadata for the internal edge promotion kernel.

- KERNEL_INFO canny_create_image (" canny_create_image ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __↩ ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial↩ _dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)))

  ACF metadata for the create image kernel.

7.9.1 Detailed Description

ACF Metadata and wrapper function for Canny edge detector.

7.9.2 Function Documentation

7.9.2.1 KERNEL_INFO canny_create_image (" canny_create_image " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)))

ACF metadata for the create image kernel.

7.9.2.2 KERNEL_INFO canny_nms_promote (" canny_nms_promote " , 4 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 1)) , __port(__index(1), __identifier("LOW_THRESH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("HIGH_THRESH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 1)))

ACF metadata for the non-maxima suppression & edge promotion kernel.

7.9.2.3 KERNEL_INFO canny_non_maxima_suppress (" canny_non_maxima_suppress " , 4 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 1)) , __port(__index(1), __identifier("LOW_THRESH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("HIGH_THRESH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 1)))

ACF metadata for the non-maxima suppression kernel.

7.9.2.4 KERNEL_INFO canny_promote_edges (" canny_promote_edges " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)))

ACF metadata for the edge promotion kernel.

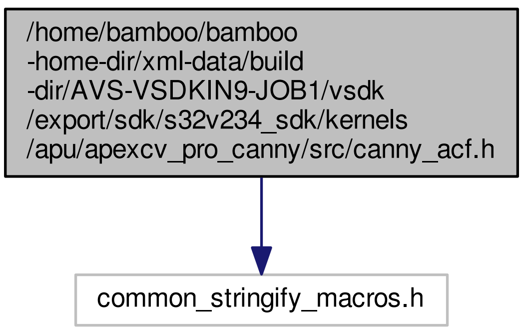
7.9.2.5 KERNEL_INFO canny_promote_edges_full (" canny_promote_edges_full " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)))

ACF metadata for the internal edge promotion kernel.

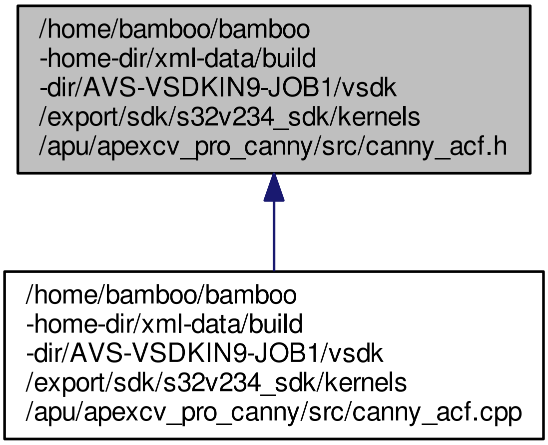
7.10 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234 _sdk/kernels/apu/apexcv_pro_canny/src/canny_acf.h File Reference

ACF Metadata for Canny edge detector.

`#include "common_stringify_macros.h"`
Include dependency graph for canny_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **CANNY_NON_MAXIMA_SUPPRESS_K** canny_non_maxima_suppress
- #define **CANNY_NON_MAXIMA_SUPPRESS_KN** XSTR(CANNY_NON_MAXIMA_SUPPRESS_K)
- #define **CANNY_NMS_PROMOTE_K** canny_nms_promote
- #define **CANNY_NMS_PROMOTE_KN** XSTR(CANNY_NMS_PROMOTE_K)
- #define **CANNY_PROMOTE_EDGES_K** canny_promote_edges
- #define **CANNY_PROMOTE_EDGES_KN** XSTR(CANNY_PROMOTE_EDGES_K)
- #define **CANNY_PROMOTE_EDGES_FULL_K** canny_promote_edges_full
- #define **CANNY_PROMOTE_EDGES_FULL_KN** XSTR(CANNY_PROMOTE_EDGES_FULL_K)
- #define **CANNY_CREATE_IMAGE_K** canny_create_image
- #define **CANNY_CREATE_IMAGE_KN** XSTR(CANNY_CREATE_IMAGE_K)
- #define **INPUT** "INPUT"
- #define **LOW_THRESH** "LOW_THRESH"
- #define **HIGH_THRESH** "HIGH_THRESH"
- #define **OUTPUT** "OUTPUT"

Functions

- **extKernelInfoDecl** (canny_non_maxima_suppress)
- **extKernelInfoDecl** (canny_nms_promote)
- **extKernelInfoDecl** (canny_promote_edges)
- **extKernelInfoDecl** (CANNY_PROMOTE_EDGES_ITERATIONS_K)
- **extKernelInfoDecl** (canny_promote_edges_full)
- **extKernelInfoDecl** (canny_create_image)

7.10.1 Detailed Description

ACF Metadata for Canny edge detector.

7.11 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_canny/src/canny_apu.cpp File Reference

Canny Edge Detection Kernels.

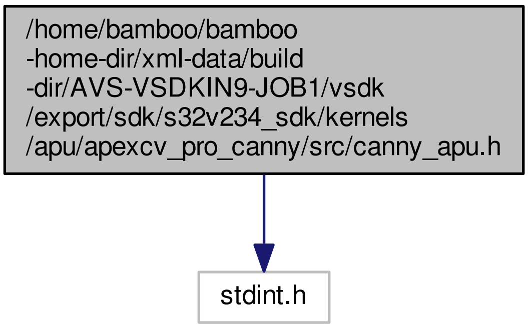
7.11.1 Detailed Description

Canny Edge Detection Kernels.

7.12 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_canny/src/canny_apu.h File Reference

Canny Edge Detection Kernels.

`#include <stdint.h>`
Include dependency graph for canny_apu.h:



Macros

- #define **MAX_BLK_SIZE** 96
- #define **MAX_STACK_SIZE** (MAX_BLK_SIZE/2)
- #define **ROW_BIT_OFFSET** 8

Functions

- void apu_canny_suppress (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh, int16u low, int16u high)
- void **apu_canny_suppress_promote** (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh, int16u low, int16u high)
- void apu_canny_connect_edges (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh)
- void apu_canny_connect_edges_full (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh)
- void apu_canny_create_image (vec08u ∗dst, int dstr, const vec08u ∗edge, int estr, int bw, int bh)

7.12.1 Detailed Description

Canny Edge Detection Kernels.

7.12.2 Function Documentation

7.12.2.1 void apu_canny_connect_edges (vec08u ∗ *dst,* int *dstr,* const vec08u ∗ *src,* int *sstr,* int *bw,* int *bh*)

Performs edge connection between blocks. This is used where the block size does not change between kernel outputs

Parameters

dst	- [Output] Destination edge map buffer
dstr	- [Input] Destination edge map stride
src	- [Input] Source edge map buffer
sstr	- [Input] Source edge map stride
bw	- [Input] Block width
bh	- [Input] Block height

7.12.2.2 void apu_canny_connect_edges_full (vec08u ∗ *dst,* int *dstr,* const vec08u ∗ *src,* int *sstr,* int *bw,* int *bh*)

Performs edge connection between blocks. This is used where the block size changes between kernel outputs

Parameters

dst	- [Output] Destination edge map buffer
dstr	- [Input] Destination edge map stride
src	- [Input] Source edge map buffer
sstr	- [Input] Source edge map stride
bw	- [Input] Block width
bh	- [Input] Block height

7.12.2.3 void apu_canny_create_image (vec08u ∗ *dst,* int *dstr,* const vec08u ∗ *edge,* int *estr,* int *bw,* int *bh*)

Outputs the *src* image with *edge* as a mask.

Parameters

dst	- [Output] Destination image buffer
dstr	- [Input] Destination image stride
src	- [Input] Source image buffer
sstr	- [Input] Source image stride
edge	- [Input] Source edge map buffer
estr	- [Input] Source edge map stride
bw	- [Input] Block width
bh	- [Input] Block height

7.12.2.4 void apu_canny_suppress (vec08u ∗ *dst,* int *dstr,* const vec08u ∗ *src,* int *sstr,* int *bw,* int *bh,* int16u *low,* int16u *high*)

Performs Canny Non-Maxima Suppression and Edge Hysteresis

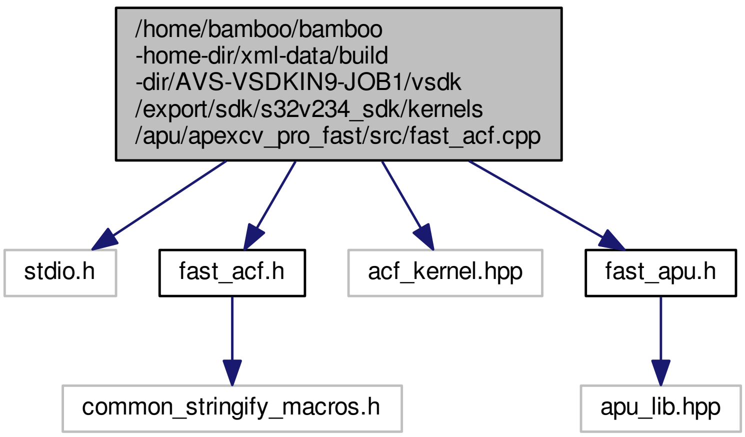
Parameters

dEdges	- [Output] Destination edge map buffer
estr	- [Input] Destination edge map stride
sMag	- [Input] Source magnitude buffer
mstr	- [Input] Source magnitude stride
dx	- [Input] Source X gradient buffer
xstr	- [Input] Source X gradient stride
dy	- [Input] Source Y gradient buffer
xstr	- [Input] Source Y gradient stride
bw	- [Input] Block width
bh	- [Input] Block height
low	- [Input] Low threshold used in edge hysteresis
high	- [Input] High threshold used in edge hysteresis

7.13 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_fast/src/fast_acf.cpp File Reference

```
#include <stdio.h>
#include "fast_acf.h"
#include "acf_kernel.hpp"
#include "fast_apu.h"
```

Include dependency graph for fast_acf.cpp:



Functions

- KERNEL_INFO **fast_offset** (" fast_offset ", 3, __port(__index(0), __identifier("OUTPUT_OFFSETS"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(1), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("CIRCUMFERENCE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO **fast** (" fast ", 4, __port(__index(0), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_↩ VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(↩ __index(1), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_↩ data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("THRESHOLD"), __↩ attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OFFSET_TABLE"), __attributes(ACF_ATTR_↩ SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(16, 1)))

- KERNEL_INFO **fast_serialized** (" fast_serialized ", 6, __port(__index(0), __identifier("INPUT"), __↩ attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), ↩ __ek_size(1, 1)), __port(__index(1), __identifier("THRESHOLD"), __attributes(ACF_ATTR_SCL_IN_↩ STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OFFSET_TABLE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), ↩ __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(3), __identifier("OUT_PACKED"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(8192, 1)), __port(__index(4), __identifier("COUNTE↩ R"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("OUT_MAX_SIZE"), __attributes(ACF↩ _ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO **fast_nms** (" fast_nms ", 4, __port(__index(0), __identifier("OUTPUT_0"), __attributes(AC↩ F_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), ↩ __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("THRESHOLD"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __↩ e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OFFSET_TABLE"), __attributes(ACF_AT↩ TR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_↩ size(16, 1)))

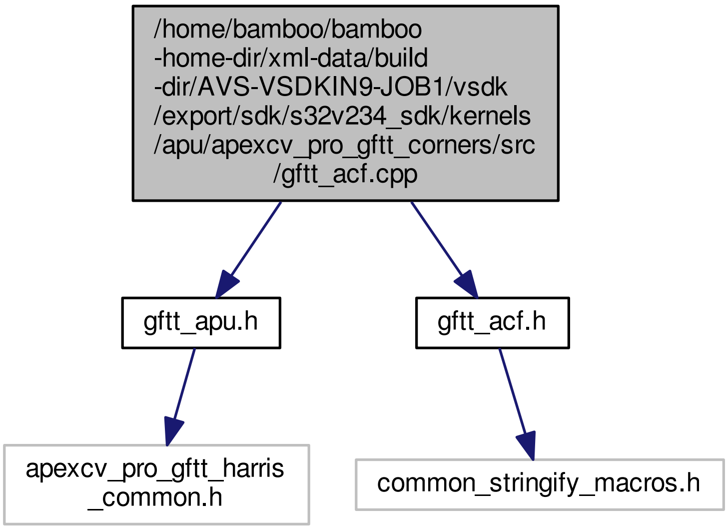
- KERNEL_INFO nms3x3 (" nms3x3 ", 2, __port(__index(0), __identifier("input"), __attributes(ACF_ATTR_↩ VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩ index(1), __identifier("output"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data↩ _type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.14 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_gftt_corners/src/gftt_acf.cpp File Reference

ACF metadata and wrapper function for the Good Features To Track.

```
#include "gftt_apu.h"
#include "gftt_acf.h"
```

Include dependency graph for gftt_acf.cpp:



Functions

- KERNEL_INFO **gftt_wrapper_box7_nms5** (" gftt_wrapper_box7_nms5 ", 12, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(6, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("SVXX"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_7+1)), __port(__index(4), __identifier("SVXY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_7+1)), __port(__index(5), __identifier("SVYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_7+1)), __port(__index(6), __identifier("NMS_X"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_NMS_R-1, MAX_NMS_R+1)), __port(__index(7), __identifier("NMS"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_NMS_R+1)), __port(__index(8), __identifier("BXX"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_7-1, 1)), __port(__index(9), __identifier("BXY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_7-1, 1)), __port(__index(10), __identifier("BYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_7-1, 1)), __port(__index(11), __identifier("MAX_EIGEN"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO **gftt_wrapper_box5_nms5** (" gftt_wrapper_box5_nms5 ", 12, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(6,

1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("S↩VXX"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_↩type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_5+1)), __port(__index(4), __identifier("SVXY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __↩e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_5+1)), __port(↩__index(5), __identifier("SVYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y↩_5+1)), __port(__index(6), __identifier("NMS_X"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MA↩X_NMS_R-1, MAX_NMS_R+1)), __port(__index(7), __identifier("NMS"), __attributes(ACF_ATTR_VEC_O↩UT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_↩BLOCK_WIDTH, MAX_NMS_R+1)), __port(__index(8), __identifier("BXX"), __attributes(ACF_ATTR_VEC↩_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(M↩AX_BLOCK_WIDTH+MAX_FILTER_X_5-1, 1)), __port(__index(9), __identifier("BXY"), __attributes(A↩CF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_5-1, 1)), __port(__index(10), __identifier("BYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_5-1, 1)), __port(__index(11), __↩identifier("MAX_EIGEN"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO **gftt_wrapper_box3_nms5** (" gftt_wrapper_box3_nms5 ", 12, __port(__index(0), __↩identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("PARAMS"), __attributes(ACF_ATTR_↩SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(6, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("S↩VXX"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_↩type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_3+1)), __port(__index(4), __identifier("SVXY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __↩e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_3+1)), __port(↩__index(5), __identifier("SVYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y↩_3+1)), __port(__index(6), __identifier("NMS_X"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MA↩X_NMS_R-1, MAX_NMS_R+1)), __port(__index(7), __identifier("NMS"), __attributes(ACF_ATTR_VEC_O↩UT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_↩BLOCK_WIDTH, MAX_NMS_R+1)), __port(__index(8), __identifier("BXX"), __attributes(ACF_ATTR_VEC↩_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(M↩AX_BLOCK_WIDTH+MAX_FILTER_X_3-1, 1)), __port(__index(9), __identifier("BXY"), __attributes(A↩CF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_3-1, 1)), __port(__index(10), __identifier("BYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_3-1, 1)), __port(__index(11), __↩identifier("MAX_EIGEN"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO **gftt_extract** (" gftt_extract ", 8, __port(__index(0), __identifier("COORD"), __attributes(A↩CF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(MAX_CORNERS, 1)), __port(__index(1), __identifier("STREN"), __attributes(ACF_ATTR_SCL↩_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(MA↩X_CORNERS, 1)), __port(__index(2), __identifier("COUNT"), __attributes(ACF_ATTR_SCL_OUT_STATI↩C_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩index(3), __identifier("LOCAL_COORD"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial↩_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_CORNER_PER_CHUNK, 1)), ↩__port(__index(4), __identifier("LOCAL_STREN"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), ↩__spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_CORNER_PER_CH↩UNK, 1)), __port(__index(5), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(6), __identifier("MAX_EIGE↩

N"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(7), __identifier("PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(6, 1)))

- KERNEL_INFO **gftt_sort_and_filter** (" gftt_sort_and_filter ", 7, __port(__index(0), __identifier("FEATURE"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_CORNERS *2, 1)), __port(__index(1), __identifier("FEAT_COUNT"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INDEX"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_CORNERS, 1)), __port(__index(3), __identifier("COORD"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(MAX_CORNERS, 1)), __port(__index(4), __identifier("STREN"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(MAX_CORNERS, 1)), __port(__index(5), __identifier("COUNT"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(6), __identifier("PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(6, 1)))

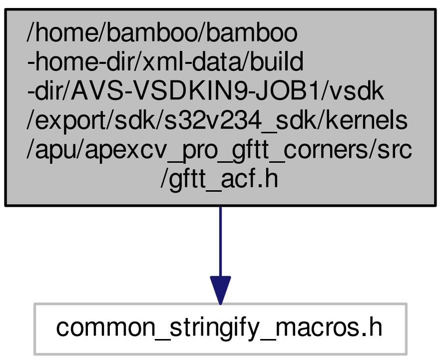
7.14.1 Detailed Description

ACF metadata and wrapper function for the Good Features To Track.

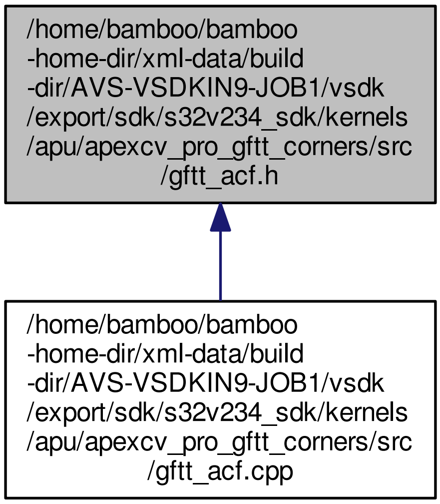
7.15 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_gftt_corners/src/gftt_acf.h File Reference

ACF metadata for the Good Features To Track.

`#include "common_stringify_macros.h"`
Include dependency graph for gftt_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **GFTT_WRAPPER_BOX7_NMS5_K** gftt_wrapper_box7_nms5
- #define **GFTT_WRAPPER_BOX7_NMS5_KN** XSTR(GFTT_WRAPPER_BOX7_NMS5_K)
- #define **GFTT_WRAPPER_BOX5_NMS5_K** gftt_wrapper_box5_nms5
- #define **GFTT_WRAPPER_BOX5_NMS5_KN** XSTR(GFTT_WRAPPER_BOX5_NMS5_K)
- #define **GFTT_WRAPPER_BOX3_NMS5_K** gftt_wrapper_box3_nms5
- #define **GFTT_WRAPPER_BOX3_NMS5_KN** XSTR(GFTT_WRAPPER_BOX3_NMS5_K)
- #define **GFTT_EXTRACT_K** gftt_extract
- #define **GFTT_EXTRACT_KN** XSTR(GFTT_EXTRACT_K)
- #define **GFTT_SORT_AND_FILTER_K** gftt_sort_and_filter
- #define **GFTT_SORT_AND_FILTER_KN** XSTR(GFTT_SORT_AND_FILTER_K)
- #define **INPUT** "INPUT"
- #define **PARAMS** "PARAMS"
- #define **OUTPUT** "OUTPUT"
- #define **SVXX** "SVXX"
- #define **SVXY** "SVXY"
- #define **SVYY** "SVYY"
- #define **NMS_X** "NMS_X"
- #define **NMS** "NMS"
- #define **BXX** "BXX"
- #define **BXY** "BXY"
- #define **BYY** "BYY"
- #define **MAX_EIGEN** "MAX_EIGEN"
- #define **COORD** "COORD"
- #define **STREN** "STREN"
- #define **COUNT** "COUNT"
- #define **LOCAL_COORD** "LOCAL_COORD"
- #define **LOCAL_STREN** "LOCAL_STREN"

- #define **SRC** "SRC"
- #define **FEATURE** "FEATURE"
- #define **FEAT_COUNT** "FEAT_COUNT"
- #define **INDEX** "INDEX"

Functions

- **extKernelInfoDecl** (gftt_wrapper_box7_nms5)
- **extKernelInfoDecl** (gftt_wrapper_box5_nms5)
- **extKernelInfoDecl** (gftt_wrapper_box3_nms5)
- **extKernelInfoDecl** (gftt_extract)
- **extKernelInfoDecl** (gftt_sort_and_filter)

7.15.1 Detailed Description

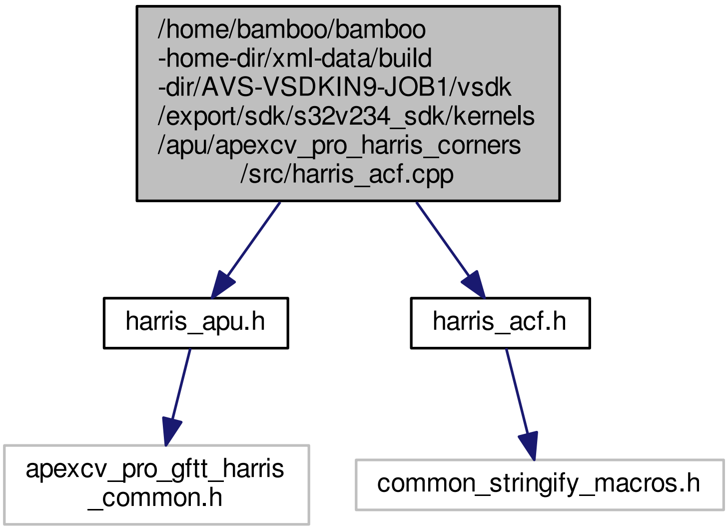
ACF metadata for the Good Features To Track.

7.16 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_harris_corners/src/harris_acf.cpp File Reference

ACF metadata and wrapper function for the harris corner.

```
#include "harris_apu.h"
#include "harris_acf.h"
```

Include dependency graph for harris_acf.cpp:



Functions

- KERNEL_INFO **harris_corners_box7_nms5** (" harris_corners_box7_nms5 ", 11, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 1, 1), __e0_data_type(d08u),

__e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("PARAMS"), __attributes(ACF_ATTR_↩
SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(7, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("S↩
VXX"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_↩
type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_7+1)), __port(__index(4), __identifier("SVXY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __↩
e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_7+1)), __port(↩
__index(5), __identifier("SVYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y↩
_7+1)), __port(__index(6), __identifier("NMS_X"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MA↩
X_NMS_R-1, MAX_NMS_R+1)), __port(__index(7), __identifier("NMS"), __attributes(ACF_ATTR_VEC_O↩
UT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_↩
BLOCK_WIDTH, MAX_NMS_R+1)), __port(__index(8), __identifier("BXX"), __attributes(ACF_ATTR_VEC↩
_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(M↩
AX_BLOCK_WIDTH+MAX_FILTER_X_7-1, 1)), __port(__index(9), __identifier("BXY"), __attributes(A↩
CF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_7-1, 1)), __port(__index(10), __identifier("BYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_7-1, 1)))

- KERNEL_INFO **harris_corners_box5_nms5** (" harris_corners_box5_nms5 ", 11, __port(__index(0), _↩
_identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("PARAMS"), __attributes(ACF_ATTR_↩
SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(7, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("S↩
VXX"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_↩
type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_5+1)), __port(__index(4), __identifier("SVXY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __↩
e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_5+1)), __port(↩
__index(5), __identifier("SVYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y↩
_5+1)), __port(__index(6), __identifier("NMS_X"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MA↩
X_NMS_R-1, MAX_NMS_R+1)), __port(__index(7), __identifier("NMS"), __attributes(ACF_ATTR_VEC_O↩
UT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_↩
BLOCK_WIDTH, MAX_NMS_R+1)), __port(__index(8), __identifier("BXX"), __attributes(ACF_ATTR_VEC↩
_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(M↩
AX_BLOCK_WIDTH+MAX_FILTER_X_5-1, 1)), __port(__index(9), __identifier("BXY"), __attributes(A↩
CF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_5-1, 1)), __port(__index(10), __identifier("BYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_5-1, 1)))
- KERNEL_INFO **harris_corners_box3_nms5** (" harris_corners_box3_nms5 ", 11, __port(__index(0), _↩
_identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("PARAMS"), __attributes(ACF_ATTR_↩
SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(7, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("S↩
VXX"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_↩
type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_3+1)), __port(__index(4), __identifier("SVXY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __↩
e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y_3+1)), __port(↩
__index(5), __identifier("SVYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH, MAX_FILTER_Y↩
_3+1)), __port(__index(6), __identifier("NMS_X"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MA↩

X_NMS_R-1, MAX_NMS_R+1)), __port(__index(7), __identifier("NMS"), __attributes(ACF_ATTR_VEC_O↩ UT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_↩ BLOCK_WIDTH, MAX_NMS_R+1)), __port(__index(8), __identifier("BXX"), __attributes(ACF_ATTR_VEC↩ _OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(M↩ AX_BLOCK_WIDTH+MAX_FILTER_X_3-1, 1)), __port(__index(9), __identifier("BXY"), __attributes(A↩ CF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_3-1, 1)), __port(__index(10), __identifier("BYY"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_BLOCK_WIDTH+MAX_FILTER_X_3-1, 1)))

- KERNEL_INFO **harris_extract** (" harris_extract ", 7, __port(__index(0), __identifier("COORD"), __↩ attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(MAX_CORNERS, 1)), __port(__index(1), __identifier("STREN"), __attributes(A↩ CF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(MAX_CORNERS, 1)), __port(__index(2), __identifier("COUNT"), __attributes(ACF_ATTR_SC↩ L_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("COORD1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(MAX_CORNER_PER_CHU↩ NK, 1)), __port(__index(4), __identifier("STREN1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(MAX_CORNER_PER_C↩ HUNK, 1)), __port(__index(5), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(6), __identifier("PARAM↩ S"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(7, 1)))

- KERNEL_INFO **harris_sort_and_filter** (" harris_sort_and_filter ", 7, __port(__index(0), __identifier("FE↩ ATURE"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_↩ type(d32s), __e0_size(1, 1), __ek_size(MAX_CORNERS ∗2, 1)), __port(__index(1), __identifier("FEA↩ T_OUT"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_↩ type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("COORD"), __attributes(ACF↩ _ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek↩ _size(MAX_CORNERS, 1)), __port(__index(3), __identifier("STREN"), __attributes(ACF_ATTR_SCL_IN_↩ STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(MAX_COR↩ NERS, 1)), __port(__index(4), __identifier("INDEX"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(MAX_CORNERS, 1)), __port(↩ __index(5), __identifier("COUNT"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(6), __identifier("PARAM↩ S"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(7, 1)))

7.16.1 Detailed Description

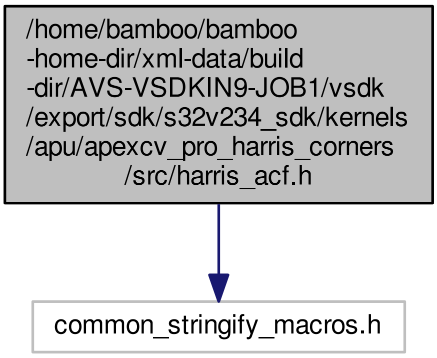
ACF metadata and wrapper function for the harris corner.

7.17 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_harris_corners/src/harris_acf.h File Reference

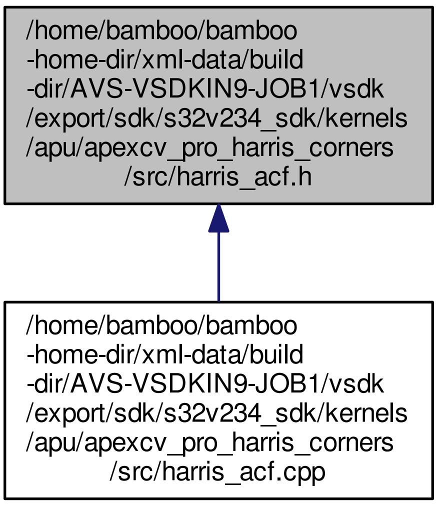
ACF metadata for the harris corner.

`#include "common_stringify_macros.h"`

Include dependency graph for harris_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **HARRIS_CORNERS_BOX7_NMS5_K** harris_corners_box7_nms5
- #define **HARRIS_CORNERS_BOX7_NMS5_KN** XSTR(HARRIS_CORNERS_BOX7_NMS5_K)
- #define **HARRIS_CORNERS_BOX5_NMS5_K** harris_corners_box5_nms5
- #define **HARRIS_CORNERS_BOX5_NMS5_KN** XSTR(HARRIS_CORNERS_BOX5_NMS5_K)
- #define **HARRIS_CORNERS_BOX3_NMS5_K** harris_corners_box3_nms5
- #define **HARRIS_CORNERS_BOX3_NMS5_KN** XSTR(HARRIS_CORNERS_BOX3_NMS5_K)

- #define **HARRIS_EXTRACT_K** harris_extract
- #define **HARRIS_EXTRACT_KN** XSTR(HARRIS_EXTRACT_K)
- #define **HARRIS_SORT_AND_FILTER_K** harris_sort_and_filter
- #define **HARRIS_SORT_AND_FILTER_KN** XSTR(HARRIS_SORT_AND_FILTER_K)
- #define **INPUT** "INPUT"
- #define **PARAMS** "PARAMS"
- #define **OUTPUT** "OUTPUT"
- #define **SVXX** "SVXX"
- #define **SVXY** "SVXY"
- #define **SVYY** "SVYY"
- #define **NMS_X** "NMS_X"
- #define **NMS** "NMS"
- #define **BXX** "BXX"
- #define **BXY** "BXY"
- #define **BYY** "BYY"
- #define **COORD** "COORD"
- #define **STREN** "STREN"
- #define **COUNT** "COUNT"
- #define **COORD1** "COORD1"
- #define **STREN1** "STREN1"
- #define **SRC** "SRC"
- #define **FEATURE** "FEATURE"
- #define **FEAT_OUT** "FEAT_OUT"
- #define **INDEX** "INDEX"

Functions

- **extKernelInfoDecl** (harris_corners_box7_nms5)
- **extKernelInfoDecl** (harris_corners_box5_nms5)
- **extKernelInfoDecl** (harris_corners_box3_nms5)
- **extKernelInfoDecl** (harris_extract)
- **extKernelInfoDecl** (harris_sort_and_filter)

7.17.1 Detailed Description

ACF metadata for the harris corner.

7.18 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_hog/src/hog_apu.h File Reference

HOG APU kernel implementation.

7.18.1 Detailed Description

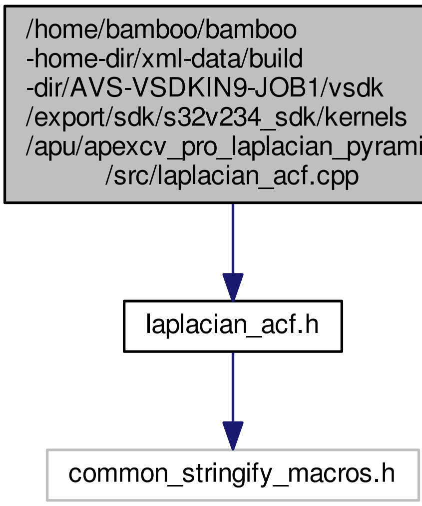
HOG APU kernel implementation.

7.19 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_laplacian_pyramid/src/laplacian_acf.cpp File Reference

ACF metadata and wrapper function for the Laplacian Pyramid.

```
#include "laplacian_acf.h"
```

Include dependency graph for laplacian_acf.cpp:



Functions

- KERNEL_INFO horizontal_gaus_laplacian_mid (" horizontal_gaus_laplacian_mid ", 2, __port(__index(0), ↩ __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_V↩ EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 2)))

  ACF metadata for the 5x1 horizontal Gaussian operation.
- KERNEL_INFO **horizontal_gaus_laplacian_last** (" horizontal_gaus_laplacian_last ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_V↩ EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO vertical_gaus_laplacian_mid (" vertical_gaus_laplacian_mid ", 4, __port(__index(0), __↩ identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), ↩ __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("INPUT_GAUSS"), __attributes(ACF_A↩ TTR_VEC_IN), __spatial_dep(0, 0, 2, 2), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 2)), __↩ port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), _↩ _e0_data_type(d16s), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(3), __identifier("OUTPUT_NEXT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), _↩ _ek_size(1, 1)))

  ACF metadata for the 1x5 vertical Gaussian and Laplacian output mid level.
- KERNEL_INFO vertical_gaus_laplacian_last (" vertical_gaus_laplacian_last ", 4, __port(__index(0), __↩ identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), ↩ __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_GAUSS"), __attributes(ACF_A↩

TTR_VEC_IN), __spatial_dep(0, 0, 2, 2), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __↩
port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), ↩
__e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUTPUT_REC"),
__attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), _↩
_ek_size(1, 1)))

ACF metadata for the 1x5 vertical Gaussian and Laplacian output last level.

7.19.1 Detailed Description

ACF metadata and wrapper function for the Laplacian Pyramid.

7.19.2 Function Documentation

7.19.2.1 KERNEL_INFO horizontal_gaus_laplacian_mid (" horizontal_gaus_laplacian_mid " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 2)))

ACF metadata for the 5x1 horizontal Gaussian operation.

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.19.2.2 KERNEL_INFO vertical_gaus_laplacian_last (" vertical_gaus_laplacian_last " , 4 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("INPUT_GAUSS"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 2, 2), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(3), __identifier("OUTPUT_REC"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

ACF metadata for the 1x5 vertical Gaussian and Laplacian output last level.

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.19.2.3 KERNEL_INFO vertical_gaus_laplacian_mid (" vertical_gaus_laplacian_mid " , 4 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(1), __identifier("INPUT_GAUSS"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 2, 2), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(3), __identifier("OUTPUT_NEXT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

ACF metadata for the 1x5 vertical Gaussian and Laplacian output mid level.

See also

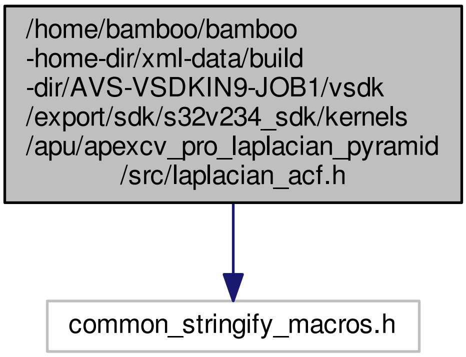
UG-10267-03 ACF User Guide, Section 3.2.2

7.20 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_laplacian_pyramid/src/laplacian_acf.h File Reference

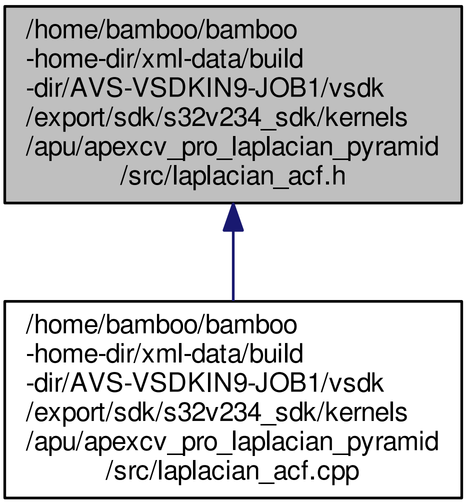
ACF metadata for the Laplacian Pyramid.

```
#include "common_stringify_macros.h"
```

Include dependency graph for laplacian_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **HORIZONTAL_GAUS_LAPLACIAN_MID_K** horizontal_gaus_laplacian_mid

- #define **HORIZONTAL_GAUS_LAPLACIAN_MID_KN** XSTR(HORIZONTAL_GAUS_LAPLACIAN_MID_K)
- #define **HORIZONTAL_GAUS_LAPLACIAN_LAST_K** horizontal_gaus_laplacian_last
- #define **HORIZONTAL_GAUS_LAPLACIAN_LAST_KN** XSTR(HORIZONTAL_GAUS_LAPLACIAN_LAS←T_K)
- #define **VERTICAL_GAUS_LAPLACIAN_MID_K** vertical_gaus_laplacian_mid
- #define **VERTICAL_GAUS_LAPLACIAN_MID_KN** XSTR(VERTICAL_GAUS_LAPLACIAN_MID_K)
- #define **VERTICAL_GAUS_LAPLACIAN_LAST_K** vertical_gaus_laplacian_last
- #define **VERTICAL_GAUS_LAPLACIAN_LAST_KN** XSTR(VERTICAL_GAUS_LAPLACIAN_LAST_K)
- #define **INPUT** "INPUT"
- #define **OUTPUT** "OUTPUT"
- #define **INPUT_GAUSS** "INPUT_GAUSS"
- #define **OUTPUT_NEXT** "OUTPUT_NEXT"
- #define **OUTPUT_REC** "OUTPUT_REC"

Functions

- **extKernelInfoDecl** (horizontal_gaus_laplacian_mid)
- **extKernelInfoDecl** (horizontal_gaus_laplacian_last)
- **extKernelInfoDecl** (vertical_gaus_laplacian_mid)
- **extKernelInfoDecl** (vertical_gaus_laplacian_last)

7.20.1 Detailed Description

ACF metadata for the Laplacian Pyramid.

7.21 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_laplacian_pyramid/src/laplacian_apu.cpp File Reference

APU Laplacian Pyramid Implementation.

7.21.1 Detailed Description

APU Laplacian Pyramid Implementation.

7.22 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_laplacian_pyramid/src/laplacian_apu.h File Reference

APU Laplacian Pyramid Header.

Functions

- void apu_pyr_horizontal_gaus_laplacian (const vec08u ∗lpvIn, int16u lStrideIn, vec16u ∗lpvOut, int16u l←StrideOut, int16u lChunkWidth, int16u lChunkHeight)

  Apply a 5x1 horizontal gaussian filter to an image.

- void apu_pyr_vertical_gaus_laplacian_mid (const vec08u ∗lpvIn, int16u lStrideIn, const vec16u ∗lpvIn_gauss, int16u lStrideIn_gauss, vec16s ∗lpvOut_lap, int16u lStrideOut_lap, vec08u ∗lpvOut_nex, int16u lStrideOut↩_nex, int16u lChunkWidth, int16u lChunkHeight)

  Apply a 1x5 vertical gaussian filter to an image and generate laplacian and reduced outputs.
- void apu_pyr_vertical_gaus_laplacian_last (const vec08u ∗lpvIn, int16u lStrideIn, const vec16u ∗lpvIn_gauss, int16u lStrideIn_gauss, vec16s ∗lpvOut_lap, int16u lStrideOut_lap, vec16s ∗lpvOut_out, int16u lStrideOut_↩out, int16u lChunkWidth, int16u lChunkHeight)

  Apply a 1x5 vertical gaussian filter to an image and generate laplacian and filtered outputs.

7.22.1 Detailed Description

APU Laplacian Pyramid Header.

image pyramid creation implementation for the APU.

7.22.2 Image Pyramid Creation

There are two common kinds of image pyramids: Gaussian pyramids and Laplacian pyramids. Here, we present the implementation of Laplacian pyramid creation.

To downsample an image (pyramid down), first the source image is convolved with a 5x5 Gaussian kernel, and then every even-numbered row and column is removed. As a result, the area is reduced to exactly one-quarter the area of the source image.

Laplacian output for each pyramid level is obtained by subtracting the convoluted image from the source image. The convoluted image is also used for the final output, which is necessasry for the input image reconstruction.

7.22.3 Function Documentation

7.22.3.1 void apu_pyr_horizontal_gaus_laplacian (const vec08u ∗ *lpvIn,* int16u *lStrideIn,* vec16u ∗ *lpvOut,* int16u *lStrideOut,* int16u *lChunkWidth,* int16u *lChunkHeight*)

Apply a 5x1 horizontal gaussian filter to an image.

5x1 horizontal gaussian filter is applied.

Parameters

lpvIn	Pointer to the source image. The source image is assumed to be padded according to the filter size. However, `lpvIn` points the top left corner of the *unpadded* image region.
lStrideIn	Stride of the padded source image.
lpvOut	Pointer to the destination image.
lStrideOut	Stride of the destination image.
lChunkWidth	Chunk width.
lChunkHeight	Chunk height.

7.22.3.2 void apu_pyr_vertical_gaus_laplacian_last (const vec08u ∗ *lpvIn,* int16u *lStrideIn,* const vec16u ∗ *lpvIn_gauss,* int16u *lStrideIn_gauss,* vec16s ∗ *lpvOut_lap,* int16u *lStrideOut_lap,* vec16s ∗ *lpvOut_out,* int16u *lStrideOut_out,* int16u *lChunkWidth,* int16u *lChunkHeight*)

Apply a 1x5 vertical gaussian filter to an image and generate laplacian and filtered outputs.

1x5 horizontal gaussian filter is applied. The filtered image is one of the outputs. Then the filered image is subtracted from de input image to generate the laplacian output.

Parameters

lpvIn	Pointer to the source image.
lStrideIn	Stride of the source image.
lpvIn_gauss	Pointer to the horizontally filtered source image. The image is assumed to be padded according to the filter size. However, `lpvIn` points the top left corner of the *unpadded* image region.
lStrideIn_gauss	Stride of the padded filtered source image.
lpvOut_lap	Pointer to the destination image - laplacian pyramid.
lStrideOut_lap	Stride of the destination image - laplacian pyramid.
lpvOut_out	Pointer to the destination image - last level output.
lStrideOut_out	Stride of the destination image - last level output.
lChunkWidth	Chunk width.
lChunkHeight	Chunk height.

7.22.3.3 void apu_pyr_vertical_gaus_laplacian_mid (const vec08u ∗ *lpvIn,* int16u *lStrideIn,* const vec16u ∗ *lpvIn_gauss,* int16u *lStrideIn_gauss,* vec16s ∗ *lpvOut_lap,* int16u *lStrideOut_lap,* vec08u ∗ *lpvOut_nex,* int16u *lStrideOut_nex,* int16u *lChunkWidth,* int16u *lChunkHeight*)

Apply a 1x5 vertical gaussian filter to an image and generate laplacian and reduced outputs.

1x5 horizontal gaussian filter is applied. Then the filered image is subtracted from de input image to generate laplacian output and it is reduced to 1/2 to generate input for the next pyramid level.

Parameters

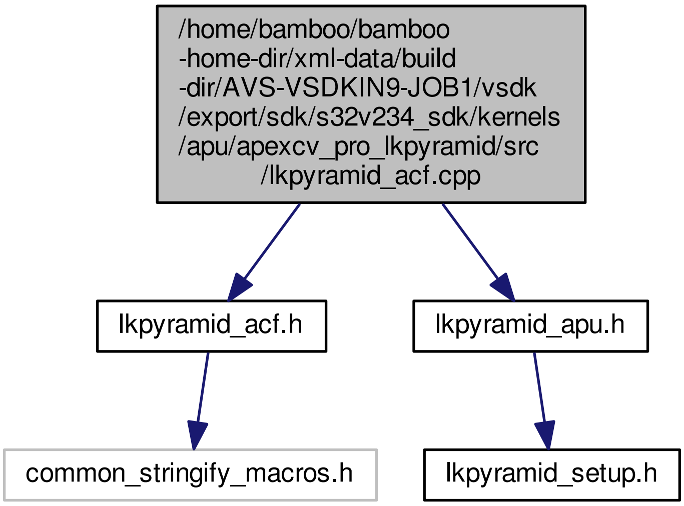
lpvIn	Pointer to the source image.
lStrideIn	Stride of the source image.
lpvIn_gauss	Pointer to the horizontally filtered source image. The image is assumed to be padded according to the filter size. However, `lpvIn` points the top left corner of the *unpadded* image region.
lStrideIn_gauss	Stride of the padded filtered source image.
lpvOut_lap	Pointer to the destination image - laplacian pyramid.
lStrideOut_lap	Stride of the destination image - laplacian pyramid.
lpvOut_nex	Pointer to the destination image - input for next level.
lStrideOut_nex	Stride of the destination image - input for next level.
lChunkWidth	Chunk width.
lChunkHeight	Chunk height.

7.23 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_lkpyramid/src/lkpyramid_acf.cpp File Reference

ACF metadata and wrapper function for the L-K optical flow pyramid.

```
#include "lkpyramid_acf.h"
#include "lkpyramid_apu.h"
```

Include dependency graph for lkpyramid_acf.cpp:



Functions

- KERNEL_INFO lkpyramid_templ_bilinear_08u_7x7 (" lkpyramid_templ_bilinear_08u_7x7 ", 3, __port(__↩
index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __↩
e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 10)), __port(__index(1), __identifier("INPUT_1"), __↩
attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1),
__ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXE↩
D), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)))

  ACF metadata for 7x7 bilinear interpolation.
- KERNEL_INFO lkpyramid_ht_centraldxdy_7x7 (" lkpyramid_ht_centraldxdy_7x7 ", 4, __port(__index(0), _↩
_identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_↩
type(d08u), __e0_size(1, 1), __ek_size(12, 9)), __port(__index(1), __identifier("INPUT_1"), __attributes(A↩
CF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), ↩
__ek_size(8, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXE↩
D), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)), __port(__index(3),
__identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_↩
data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)))

  ACF metadata for 7x7 X gradient using 3x3 centraldx/dy.
- KERNEL_INFO lkpyramid_core_7x7 (" lkpyramid_core_7x7 ", 13, __port(__index(0), __identifier("INPUT↩
_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0↩
_size(1, 1), __ek_size(20, 20)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VE↩
C_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)), __↩
port(__index(2), __identifier("INPUT_2"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0,
0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)), __port(__index(3), __identifier("INPUT_3"), ↩
__attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1,
1), __ek_size(8, 7)), __port(__index(4), __identifier("INPUT_4"), __attributes(ACF_ATTR_VEC_IN_FIXE↩
D), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)), __port(__index(5),
__identifier("INPUT_5"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data↩
_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(6), __identifier("INPUT_6"), __attributes(A↩

CF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(8, 1)), __port(__index(7), __identifier("INPUT_7"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)), __port(__index(8), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)), __port(__index(9), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(10), __identifier("OUTPUT_2"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(11), __identifier("OUTPUT_3"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(8, 7)), __port(__index(12), __identifier("OUTPUT_4"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)))

ACF metadata for 7x7 L-K pyramid core iteration.

7.23.1 Detailed Description

ACF metadata and wrapper function for the L-K optical flow pyramid.

-
-

7.23.2 Function Documentation

7.23.2.1 KERNEL_INFO lkpyramid_core_7x7 (" lkpyramid_core_7x7 " , 13 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(20, 20)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)) , __port(__index(2), __identifier("INPUT_2"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)) , __port(__index(3), __identifier("INPUT_3"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)) , __port(__index(4), __identifier("INPUT_4"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)) , __port(__index(5), __identifier("INPUT_5"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(6), __identifier("INPUT_6"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(8, 1)) , __port(__index(7), __identifier("INPUT_7"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)) , __port(__index(8), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)) , __port(__index(9), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(10), __identifier("OUTPUT_2"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(11), __identifier("OUTPUT_3"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(8, 7)) , __port(__index(12), __identifier("OUTPUT_4"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)))

ACF metadata for 7x7 L-K pyramid core iteration.

**

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.23.2.2 KERNEL_INFO lkpyramid_ht_centraldxdy_7x7 (" lkpyramid_ht_centraldxdy_7x7 " , 4 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(8, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)) , __port(__index(3), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)))

ACF metadata for 7x7 X gradient using 3x3 centraldx/dy.

**

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.23.2.3 KERNEL_INFO lkpyramid_templ_bilinear_08u_7x7 (" lkpyramid_templ_bilinear_08u_7x7 " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 10)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)))

ACF metadata for 7x7 bilinear interpolation.

**

See also

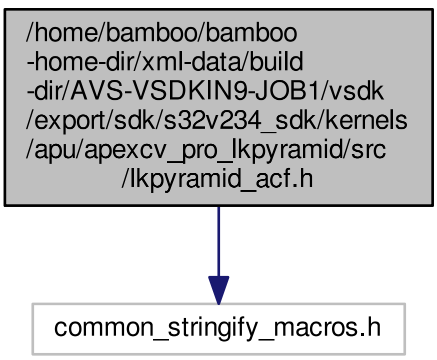
UG-10267-03 ACF User Guide, Section 3.2.2

7.24 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_lkpyramid/src/lkpyramid_acf.h File Reference

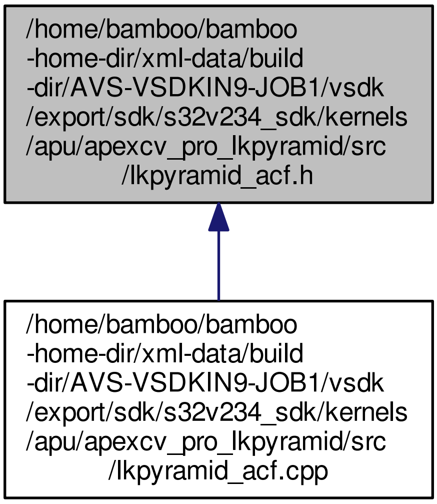
ACF metadata for the L-K optical flow pyramid.

```
#include "common_stringify_macros.h"
```

Include dependency graph for lkpyramid_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **LKPYRAMID_TEMPL_BILINEAR_08U_7X7_K** lkpyramid_templ_bilinear_08u_7x7
- #define **LKPYRAMID_TEMPL_BILINEAR_08U_7X7_KN** XSTR(LKPYRAMID_TEMPL_BILINEAR_08U_7↩X7_K)
- #define **LKPYRAMID_HT_CENTRALDXDY_7X7_K** lkpyramid_ht_centraldxdy_7x7
- #define **LKPYRAMID_HT_CENTRALDXDY_7X7_KN** XSTR(LKPYRAMID_HT_CENTRALDXDY_7X7_K)
- #define **LKPYRAMID_CORE_7X7_K** lkpyramid_core_7x7
- #define **LKPYRAMID_CORE_7X7_KN** XSTR(LKPYRAMID_CORE_7X7_K)
- #define **INPUT_0** "INPUT_0"
- #define **INPUT_1** "INPUT_1"
- #define **OUTPUT_0** "OUTPUT_0"
- #define **OUTPUT_1** "OUTPUT_1"
- #define **INPUT_2** "INPUT_2"
- #define **INPUT_3** "INPUT_3"
- #define **INPUT_4** "INPUT_4"
- #define **INPUT_5** "INPUT_5"
- #define **INPUT_6** "INPUT_6"
- #define **INPUT_7** "INPUT_7"
- #define **OUTPUT_2** "OUTPUT_2"
- #define **OUTPUT_3** "OUTPUT_3"
- #define **OUTPUT_4** "OUTPUT_4"

Functions

- **extKernelInfoDecl** (lkpyramid_templ_bilinear_08u_7x7)
- **extKernelInfoDecl** (lkpyramid_ht_centraldxdy_7x7)
- **extKernelInfoDecl** (lkpyramid_core_7x7)

7.24.1 Detailed Description

ACF metadata for the L-K optical flow pyramid.

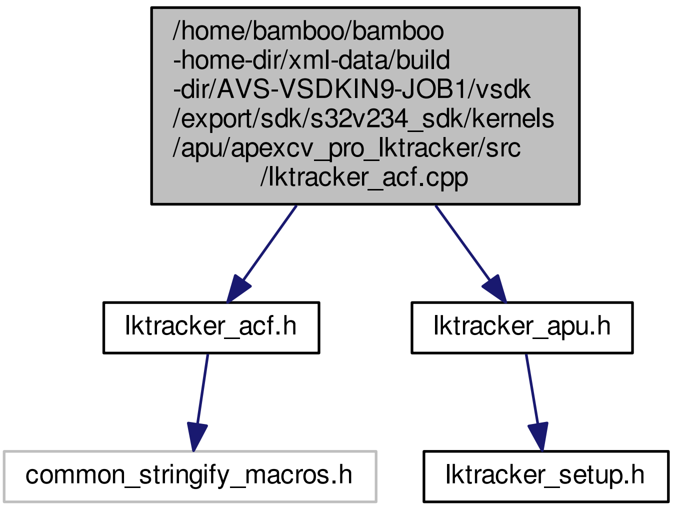
-
-

7.25 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_lktracker/src/lktracker_acf.cpp File Reference

ACF metadata and wrapper function for the L-K optical flow tracker.

```
#include "lktracker_acf.h"
#include "lktracker_apu.h"
```

Include dependency graph for lktracker_acf.cpp:



Functions

- KERNEL_INFO lktracker_templ_bilinear_08u_7x7 (" lktracker_templ_bilinear_08u_7x7 ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 10)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)))

  ACF metadata for 7x7 bilinear interpolation.
- KERNEL_INFO lktracker_ht_centraldxdy_7x7 (" lktracker_ht_centraldxdy_7x7 ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(12, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)), __port(__index(3),

__identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)))

ACF metadata for 7x7 X gradient using 3x3 centraldx/dy.

- KERNEL_INFO lktracker_core_7x7 (" lktracker_core_7x7 ", 9, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(20, 20)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)), __port(__index(2), __identifier("INPUT_2"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)), __port(__index(3), __identifier("INPUT_3"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)), __port(__index(4), __identifier("INPUT_4"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)), __port(__index(5), __identifier("INPUT_5"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(12, 1)), __port(__index(6), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)), __port(__index(7), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(8, 7)), __port(__index(8), __identifier("OUTPUT_2"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)))

  ACF metadata for 7x7 L-K tracker core iteration.

7.25.1 Detailed Description

ACF metadata and wrapper function for the L-K optical flow tracker.

7.25.2 Function Documentation

7.25.2.1 KERNEL_INFO lktracker_core_7x7 (" lktracker_core_7x7 " , 9 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(20, 20)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)) , __port(__index(2), __identifier("INPUT_2"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)) , __port(__index(3), __identifier("INPUT_3"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)) , __port(__index(4), __identifier("INPUT_4"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)) , __port(__index(5), __identifier("INPUT_5"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(12, 1)) , __port(__index(6), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)) , __port(__index(7), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(8, 7)) , __port(__index(8), __identifier("OUTPUT_2"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)))

ACF metadata for 7x7 L-K tracker core iteration.

**

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.25.2.2 KERNEL_INFO lktracker_ht_centraldxdy_7x7 (" lktracker_ht_centraldxdy_7x7 " , 4 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(12, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)) , __port(__index(3), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(8, 7)))

ACF metadata for 7x7 X gradient using 3x3 centraldx/dy.

**

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.25.2.3 KERNEL_INFO lktracker_templ_bilinear_08u_7x7 (" lktracker_templ_bilinear_08u_7x7 " , 3 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 10)) , __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(4, 1), __ek_size(1, 1)) , __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(12, 9)))

ACF metadata for 7x7 bilinear interpolation.

**

See also

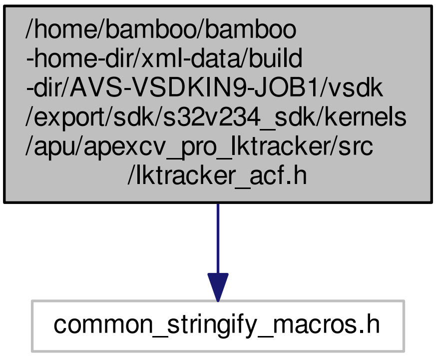
UG-10267-03 ACF User Guide, Section 3.2.2

7.26 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_lktracker/src/lktracker_acf.h File Reference

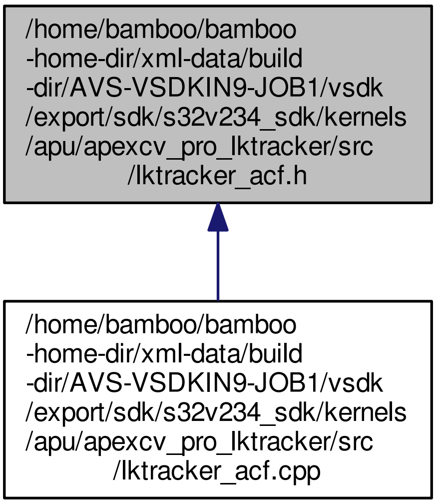
ACF metadata for the L-K optical flow tracker.

```
#include "common_stringify_macros.h"
```

Include dependency graph for lktracker_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **LKTRACKER_TEMPL_BILINEAR_08U_7X7_K** lktracker_templ_bilinear_08u_7x7
- #define **LKTRACKER_TEMPL_BILINEAR_08U_7X7_KN** XSTR(LKTRACKER_TEMPL_BILINEAR_08U_↩ 7X7_K)
- #define **LKTRACKER_HT_CENTRALDXDY_7X7_K** lktracker_ht_centraldxdy_7x7
- #define **LKTRACKER_HT_CENTRALDXDY_7X7_KN** XSTR(LKTRACKER_HT_CENTRALDXDY_7X7_K)
- #define **LKTRACKER_CORE_7X7_K** lktracker_core_7x7

- #define **LKTRACKER_CORE_7X7_KN** XSTR(LKTRACKER_CORE_7X7_K)
- #define **INPUT_0** "INPUT_0"
- #define **INPUT_1** "INPUT_1"
- #define **OUTPUT_0** "OUTPUT_0"
- #define **OUTPUT_1** "OUTPUT_1"
- #define **INPUT_2** "INPUT_2"
- #define **INPUT_3** "INPUT_3"
- #define **INPUT_4** "INPUT_4"
- #define **INPUT_5** "INPUT_5"
- #define **OUTPUT_2** "OUTPUT_2"

Functions

- **extKernelInfoDecl** (lktracker_templ_bilinear_08u_7x7)
- **extKernelInfoDecl** (lktracker_ht_centraldxdy_7x7)
- **extKernelInfoDecl** (lktracker_core_7x7)

7.26.1 Detailed Description

ACF metadata for the L-K optical flow tracker.

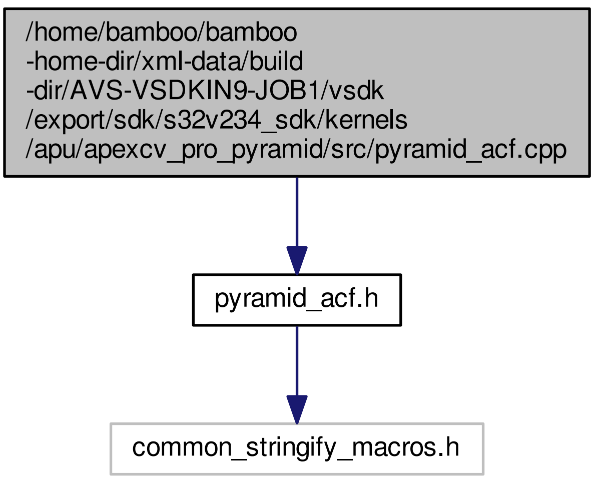
-
-

7.27 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_pyramid/src/pyramid_acf.cpp File Reference

ACF metadata and wrapper function for the image pyramid creation.

```
#include "pyramid_acf.h"
```

Include dependency graph for pyramid_acf.cpp:



Functions

- KERNEL_INFO horizontal_gaus (" horizontal_gaus ", 2, __port(__index(0), __identifier("INPUT"), __↩attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek↩_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_↩dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

  ACF metadata for the 5x1 horizontal Gaussian operation.
- KERNEL_INFO horizontal_gaus_and_expand (" horizontal_gaus_and_expand ", 2, __port(__index(0), _↩_identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_V↩EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(2, 2)))

  ACF metadata for the 5x1 horizontal Gaussian and expand operation.
- KERNEL_INFO vertical_gaus (" vertical_gaus ", 2, __port(__index(0), __identifier("INPUT"), __attributes(A↩CF_ATTR_VEC_IN), __spatial_dep(0, 0, 2, 2), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

  ACF metadata for the 1x5 vertical Gaussian operation.
- KERNEL_INFO vertical_gaus_and_reduce (" vertical_gaus_and_reduce ", 2, __port(__index(0), __↩identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 2, 2), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_V↩EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

  ACF metadata for the 1x5 vertical Gaussian and reduce operation.

7.27.1 Detailed Description

ACF metadata and wrapper function for the image pyramid creation.

7.27.2 Function Documentation

7.27.2.1 KERNEL_INFO horizontal_gaus (" horizontal_gaus " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

ACF metadata for the 5x1 horizontal Gaussian operation.

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.27.2.2 KERNEL_INFO horizontal_gaus_and_expand (" horizontal_gaus_and_expand " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(2, 2)))

ACF metadata for the 5x1 horizontal Gaussian and expand operation.

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.27.2.3 KERNEL_INFO vertical_gaus (" vertical_gaus " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 2, 2), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

ACF metadata for the 1x5 vertical Gaussian operation.

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.27.2.4 KERNEL_INFO vertical_gaus_and_reduce (" vertical_gaus_and_reduce " , 2 , __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 2, 2), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(2, 2)) , __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

ACF metadata for the 1x5 vertical Gaussian and reduce operation.

See also

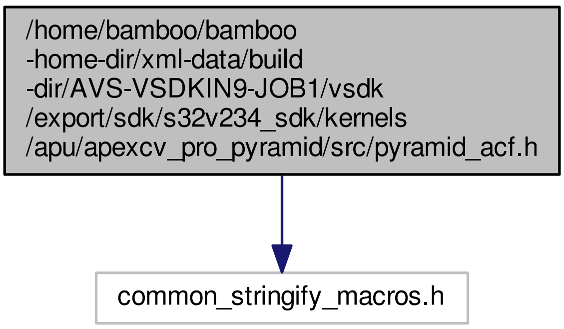
UG-10267-03 ACF User Guide, Section 3.2.2

7.28 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_pyramid/src/pyramid_acf.h File Reference

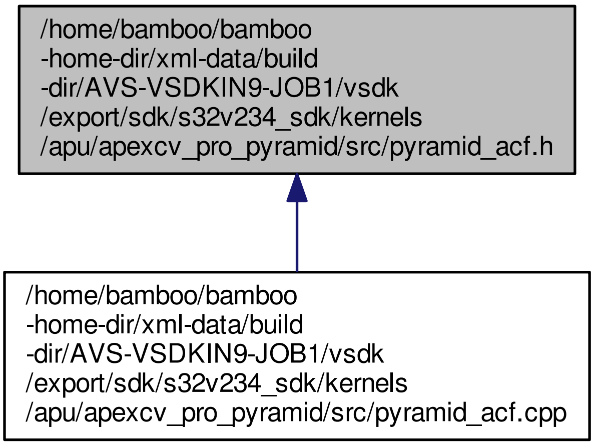
ACF metadata for the image pyramid creation.

`#include "common_stringify_macros.h"`

Include dependency graph for pyramid_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **HORIZONTAL_GAUS_K** horizontal_gaus
- #define **HORIZONTAL_GAUS_KN** XSTR(HORIZONTAL_GAUS_K)
- #define **HORIZONTAL_GAUS_AND_EXPAND_K** horizontal_gaus_and_expand
- #define **HORIZONTAL_GAUS_AND_EXPAND_KN** XSTR(HORIZONTAL_GAUS_AND_EXPAND_K)
- #define **VERTICAL_GAUS_K** vertical_gaus
- #define **VERTICAL_GAUS_KN** XSTR(VERTICAL_GAUS_K)
- #define **VERTICAL_GAUS_AND_REDUCE_K** vertical_gaus_and_reduce
- #define **VERTICAL_GAUS_AND_REDUCE_KN** XSTR(VERTICAL_GAUS_AND_REDUCE_K)
- #define **INPUT** "INPUT"
- #define **OUTPUT** "OUTPUT"

Functions

- **extKernelInfoDecl** (horizontal_gaus)
- **extKernelInfoDecl** (horizontal_gaus_and_expand)
- **extKernelInfoDecl** (vertical_gaus)
- **extKernelInfoDecl** (vertical_gaus_and_reduce)

7.28.1 Detailed Description

ACF metadata for the image pyramid creation.

7.29 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234 _sdk/kernels/apu/apexcv_pro_pyramid/src/pyramid_apu.cpp File Reference

apu Image Pyramid Implementation.

7.29.1 Detailed Description

apu Image Pyramid Implementation.

7.30 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_pyramid/src/pyramid_apu.h File Reference

APU pyramid creation algorithm implementation.

7.30.1 Detailed Description

APU pyramid creation algorithm implementation.

7.31 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_remap/src/remap_apu.h File Reference

Declaration of kernel functions for Remap.

Functions

- void remap_bilinear_rgb (vec32u __cmem ∗dst, vec32u __cmem ∗src, vec16u __cmem ∗offset, vec08u __cmem ∗delta, int sstride, int dstride, int bw, int bh)

  Performs remap of RGBA pixel data..
- void remap_bilinear_grayscale (vec08u __cmem ∗dst, vec08u __cmem ∗src, vec16u __cmem ∗offset, vec08u __cmem ∗delta, int sstride, int dstride, int bw, int bh)

  Performs remap of 8-bit grayscale pixel data.
- void **remap_block_size** (vec32u ∗map, int32_t chunk_width, int32_t chunk_height, int32_t tile_width, vec16u ∗t_width, vec16u ∗t_height, vec16u ∗f_width, vec16u ∗f_height, vec16u ∗e_width, vec16u ∗e_height, vec16u ∗s_width, vec16u ∗s_height)

7.31.1 Detailed Description

Declaration of kernel functions for Remap.

7.31.2 Function Documentation

7.31.2.1 void remap_bilinear_grayscale (vec08u __cmem ∗ *dst,* vec08u __cmem ∗ *src,* vec16u __cmem ∗ *offset,* vec08u __cmem ∗ *delta,* int *sstride,* int *dstride,* int *bw,* int *bh*)

Performs remap of 8-bit grayscale pixel data.

Returns

None.

7.31.2.2 void remap_bilinear_rgb (vec32u __cmem * *dst*, vec32u __cmem * *src*, vec16u __cmem * *offset*, vec08u __cmem * *delta*, int *sstride*, int *dstride*, int *bw*, int *bh*)

Performs remap of RGBA pixel data..

Returns

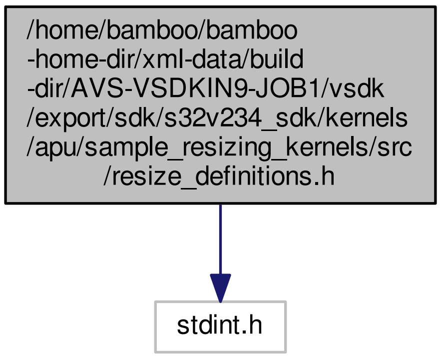
None.

7.32 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_resizing_kernels/src/resize_definitions.h File Reference

General definitions and declarations (Standard C), for Vertical Resizer.

`#include <stdint.h>`

Include dependency graph for resize_definitions.h:



Classes

- struct RESIZE_DESCRIPTOR

  Struct for Resize descriptor.

Macros

- #define RSZ_DECIMAL_SCL 16
- #define FLT_BANK_SIZE 32
- #define FLT_UP_SAMPLING
- #define FLT_DOWN_SAMPLING1

  8 Phases, 4 Taps
- #define FLT_DOWN_SAMPLING2

  4 Phases, 8 Taps
- #define RESIZE_DESC_SIZE (POLYPHASE_FLT_SIZE*2 + (10*4))

  Size of descriptors in Bytes.

7.32.1 Detailed Description

General definitions and declarations (Standard C), for Vertical Resizer.

7.32.2 Macro Definition Documentation

7.32.2.1 #define FLT_BANK_SIZE 32

Polyphase filter bank size

7.32.2.2 #define FLT_DOWN_SAMPLING1

Value:

```
{28, 200,  28, 0,\
                            15, 194,  47, 0,\
                             7, 178,  71, 0,\
                             2, 156,  98, 0,\
                             0, 128, 128, 0,\
                             0,  98, 156, 2,\
                             0,  71, 178, 7,\
                             0,  47, 194, 15}
```

8 Phases, 4 Taps

7.32.2.3 #define FLT_DOWN_SAMPLING2

Value:

```
{2, 32, 94,  94, 32,  2, 0, 0,\
                            0, 19, 80, 105, 47,  5, 0, 0,\
                            0, 11, 64, 106, 64, 11, 0, 0,\
                            0,  5, 47, 105, 80, 19, 0, 0}
```

4 Phases, 8 Taps

7.32.2.4 #define FLT_UP_SAMPLING

Value:

```
{0 , 256,   0,   0,\
                        -11, 245,  23,  -1,\
                        -17, 220,  58,  -5,\
                        -18, 184, 100, -10,\
                        -15, 143, 143, -15,\
                        -10, 100, 184, -18,\
                         -5,  58, 220, -17,\
                         -1,  23, 245, -11}
```

Filter banks 8 Phases, 4 Taps

7.32.2.5 #define RESIZE_DESC_SIZE (POLYPHASE_FLT_SIZE∗2 + (10∗4))

Size of descriptors in Bytes.

Did not use sizeof to prevent probable size mismatch of same types on different platforms sizeof(RESIZE_DESC↩
RIPTOR) in bytes

7.32.2.6 #define RSZ_DECIMAL_SCL 16

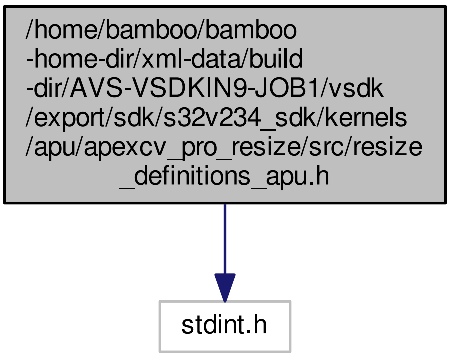
Fixed-point scaler

7.33 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/apexcv_pro_resize/src/resize_definitions_apu.h File Reference

General definitions and declarations (Standard C), for Vertical Resizer.

`#include <stdint.h>`

Include dependency graph for resize_definitions_apu.h:



Classes

- struct RESIZE_DESCRIPTOR

  Struct for Resize descriptor.

Macros

- #define RSZ_DECIMAL_SCL 16
- #define FLT_BANK_SIZE 32
- #define FLT_UP_SAMPLING
- #define FLT_DOWN_SAMPLING1

  8 Phases, 4 Taps
- #define FLT_DOWN_SAMPLING2

  4 Phases, 8 Taps
- #define RESIZE_DESC_SIZE (POLYPHASE_FLT_SIZE∗2 + (10∗4))

  Size of descriptors in Bytes.

7.33.1 Detailed Description

General definitions and declarations (Standard C), for Vertical Resizer.

7.33.2 Macro Definition Documentation

7.33.2.1 #define FLT_BANK_SIZE 32

Polyphase filter bank size

7.33.2.2 #define FLT_DOWN_SAMPLING1

Value:

```
{28, 200,  28, 0,\
                            15, 194,  47, 0,\
                             7, 178,  71, 0,\
                             2, 156,  98, 0,\
                             0, 128, 128, 0,\
                             0,  98, 156, 2,\
                             0,  71, 178, 7,\
                             0,  47, 194, 15}
```

8 Phases, 4 Taps

7.33.2.3 #define FLT_DOWN_SAMPLING2

Value:

```
{2, 32, 94,  94, 32,  2, 0, 0,\
                            0, 19, 80, 105, 47,  5, 0, 0,\
                            0, 11, 64, 106, 64, 11, 0, 0,\
                            0,  5, 47, 105, 80, 19, 0, 0}
```

4 Phases, 8 Taps

7.33.2.4 #define FLT_UP_SAMPLING

Value:

```
{0 , 256,   0,   0,\
                        -11, 245,  23,  -1,\
                        -17, 220,  58,  -5,\
                        -18, 184, 100, -10,\
                        -15, 143, 143, -15,\
                        -10, 100, 184, -18,\
                         -5,  58, 220, -17,\
                         -1,  23, 245, -11}
```

Filter banks 8 Phases, 4 Taps

7.33.2.5 #define RESIZE_DESC_SIZE (POLYPHASE_FLT_SIZE∗2 + (10∗4))

Size of descriptors in Bytes.

Did not use sizeof to prevent probable size mismatch of same types on different platforms sizeof(RESIZE_DESC↩
RIPTOR) in bytes

7.33.2.6 #define RSZ_DECIMAL_SCL 16

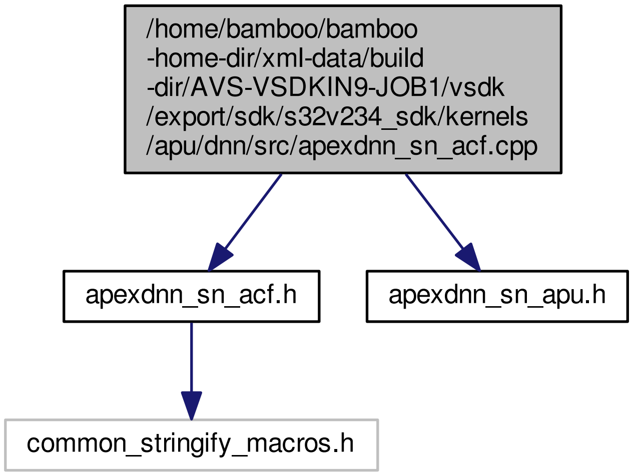
Fixed-point scaler

7.34 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/dnn/src/apexdnn_sn_acf.cpp File Reference

ACF metadata and wrapper function for the Convolutional Neural Networks.

```
#include "apexdnn_sn_acf.h"
#include "apexdnn_sn_apu.h"
```

Include dependency graph for apexdnn_sn_acf.cpp:



Functions

- KERNEL_INFO apu_conv3x3mps1_module_nhcw_forward (" apu_conv3x3mps1_module_nhcw_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 4)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(32, 1)), __port(__index(3), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(12, 1)), __port(__index(4), __identifier("OUTPUT_CONV3X3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(5), __identifier("OUTPUT_MP"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(6), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)))

  ACF metadata for APEXDNN library.
- KERNEL_INFO **apu_e1e3s1_module_nhcw_forward** (" apu_e1e3s1_module_nhcw_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(32, 1)), __port(__index(3), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1),

__ek_size(12, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_↵STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), ↵__port(__index(5), __identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __↵spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(6), __↵identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_↵data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)))

- KERNEL_INFO **apu_e1e3mps1_module_nhcw_forward** (" apu_e1e3mps1_module_nhcw_forward ", 9, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_↵dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("I↵NPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_↵data_type(d08s), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(2), __identifier("INPUT_PARAMS"), _↵_attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0↵_size(1, 1), __ek_size(32, 1)), __port(__index(3), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(A↵CF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(12, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_↵OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(5), __identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(6), __↵identifier("OUTPUT_MP1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(7), __identifier("OUTPUT_MP3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(8), __identifier("OUTPUT_S1"), __attributes(ACF_AT↵TR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)))
- KERNEL_INFO **apu_e1e3mp_module_nhcw_forward** (" apu_e1e3mp_module_nhcw_forward ", 8, __↵port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 4)), __port(__index(1), __identifier("INPU↵T_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data↵_type(d08s), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(2), __identifier("INPUT_PARAMS"), __↵attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_↵size(1, 1), __ek_size(32, 1)), __port(__index(3), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(A↵CF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(12, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_↵STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), ↵__port(__index(5), __identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __↵spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(6), __↵identifier("OUTPUT_MP1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0↵_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(7), __identifier("OUTPUT_MP3"), _↵_attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)))
- KERNEL_INFO **apu_e1e3_module_nhcw_forward** (" apu_e1e3_module_nhcw_forward ", 6, __port(__↵index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(1), __identifier("INPUT_WEIGH↵T"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(AC↵F_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), ↵__ek_size(32, 1)), __port(__index(3), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR↵_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_↵size(12, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), ↵__spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(5), _↵_identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_↵data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)))
- KERNEL_INFO **apu_eltmulcred_module_nhcw_forward** (" apu_eltmulcred_module_nhcw_forward ", 6, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial↵_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __↵identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 0, 0), __↵e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_BIAS"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1,

1), __ek_size(1, 1)), __port(__index(3), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(32, 1)), __port(__index(4), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("OUTPUT"), __attributes(ACF_ATTR_SCL_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO **apu_e1ap_module_nhcw_forward** (" apu_e1ap_module_nhcw_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(2), __identifier("INPUT_ACCM"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(32, 1)), __port(__index(4), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(12, 1)), __port(__index(5), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(6), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

7.34.1 Detailed Description

ACF metadata and wrapper function for the Convolutional Neural Networks.

-
-

7.34.2 Function Documentation

7.34.2.1 KERNEL_INFO apu_conv3x3mps1_module_nhcw_forward (" apu_conv3x3mps1_module_nhcw_forward " , 7 , __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 4)) , __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 1)) , __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(32, 1)) , __port(__index(3), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(12, 1)) , __port(__index(4), __identifier("OUTPUT_CONV3X3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)) , __port(__index(5), __identifier("OUTPUT_MP"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)) , __port(__index(6), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(2, 1)))

ACF metadata for APEXDNN library.

See also

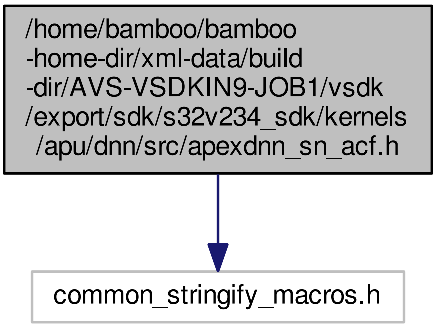
UG-10267-03 ACF User Guide, Section 3.2.2

7.35 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/dnn/src/apexdnn_sn_acf.h File Reference

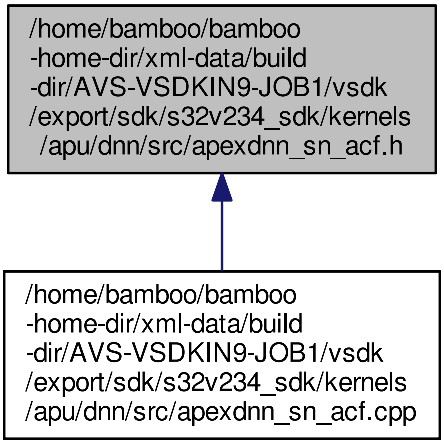
ACF metadata and wrapper function for the APEXDNN Library.

```
#include "common_stringify_macros.h"
```

Include dependency graph for apexdnn_sn_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **APU_CONV3X3MPS1_MODULE_NHCW_FORWARD_K** apu_conv3x3mps1_module_nhcw_↵ forward
- #define **APU_CONV3X3MPS1_MODULE_NHCW_FORWARD_KN** XSTR(APU_CONV3X3MPS1_MODU↵ LE_NHCW_FORWARD_K)
- #define **APU_E1E3S1_MODULE_NHCW_FORWARD_K** apu_e1e3s1_module_nhcw_forward
- #define **APU_E1E3S1_MODULE_NHCW_FORWARD_KN** XSTR(APU_E1E3S1_MODULE_NHCW_FOR↵ WARD_K)
- #define **APU_E1E3MPS1_MODULE_NHCW_FORWARD_K** apu_e1e3mps1_module_nhcw_forward

- #define **APU_E1E3MPS1_MODULE_NHCW_FORWARD_KN** XSTR(APU_E1E3MPS1_MODULE_NHCW_FORWARD_K)
- #define **APU_E1E3MP_MODULE_NHCW_FORWARD_K** apu_e1e3mp_module_nhcw_forward
- #define **APU_E1E3MP_MODULE_NHCW_FORWARD_KN** XSTR(APU_E1E3MP_MODULE_NHCW_FORWARD_K)
- #define **APU_E1E3_MODULE_NHCW_FORWARD_K** apu_e1e3_module_nhcw_forward
- #define **APU_E1E3_MODULE_NHCW_FORWARD_KN** XSTR(APU_E1E3_MODULE_NHCW_FORWARD_K)
- #define **APU_ELTMULCRED_MODULE_NHCW_FORWARD_K** apu_eltmulcred_module_nhcw_forward
- #define **APU_ELTMULCRED_MODULE_NHCW_FORWARD_KN** XSTR(APU_ELTMULCRED_MODULE_NHCW_FORWARD_K)
- #define **APU_E1AP_MODULE_NHCW_FORWARD_K** apu_e1ap_module_nhcw_forward
- #define **APU_E1AP_MODULE_NHCW_FORWARD_KN** XSTR(APU_E1AP_MODULE_NHCW_FORWARD_K)
- #define **INPUT_IMAGE** "INPUT_IMAGE"
- #define **INPUT_WEIGHT** "INPUT_WEIGHT"
- #define **INPUT_PARAMS** "INPUT_PARAMS"
- #define **OUTPUT_SCRATCH_BUF** "OUTPUT_SCRATCH_BUF"
- #define **OUTPUT_CONV3X3** "OUTPUT_CONV3X3"
- #define **OUTPUT_MP** "OUTPUT_MP"
- #define **OUTPUT_S1** "OUTPUT_S1"
- #define **OUTPUT_E1** "OUTPUT_E1"
- #define **OUTPUT_E3** "OUTPUT_E3"
- #define **OUTPUT_MP1** "OUTPUT_MP1"
- #define **OUTPUT_MP3** "OUTPUT_MP3"
- #define **INPUT_BIAS** "INPUT_BIAS"
- #define **OUTPUT** "OUTPUT"
- #define **INPUT_ACCM** "INPUT_ACCM"

Functions

- **extKernelInfoDecl** (apu_conv3x3mps1_module_nhcw_forward)
- **extKernelInfoDecl** (apu_e1e3s1_module_nhcw_forward)
- **extKernelInfoDecl** (apu_e1e3mps1_module_nhcw_forward)
- **extKernelInfoDecl** (apu_e1e3mp_module_nhcw_forward)
- **extKernelInfoDecl** (apu_e1e3_module_nhcw_forward)
- **extKernelInfoDecl** (apu_eltmulcred_module_nhcw_forward)
- **extKernelInfoDecl** (apu_e1ap_module_nhcw_forward)

7.35.1 Detailed Description

ACF metadata and wrapper function for the APEXDNN Library.

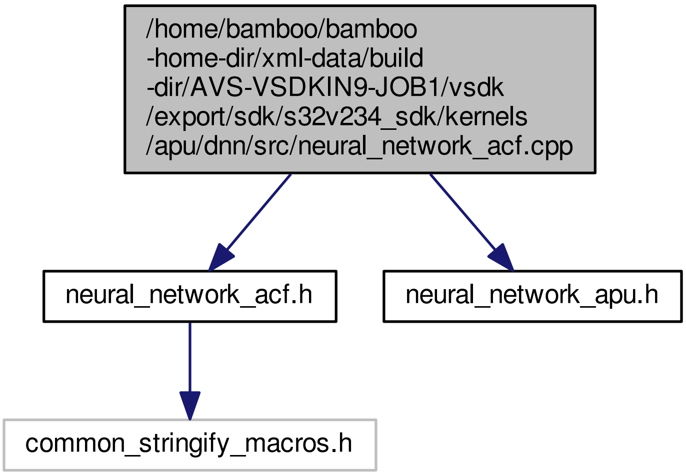
-
-

7.36 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/dnn/src/neural_network_acf.cpp File Reference

ACF metadata and wrapper function for the Convolutional Neural Networks.

```
#include "neural_network_acf.h"
#include "neural_network_apu.h"
```

Include dependency graph for neural_network_acf.cpp:



Functions

- KERNEL_INFO apu_fire2_conv1mps1_forward (" apu_fire2_conv1mps1_forward ", 8, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 2, 3), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(700, 12)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 7)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(112, 1)), __port(__index(4), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(9, 1)), __port(__index(5), __identifier("OUTPUT_CONV1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(222, 3)), __port(__index(6), __identifier("OUTPUT_MP"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(112, 1)), __port(__index(7), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 3)))

  ACF metadata for Convolutional Neural Networks.
- KERNEL_INFO **apu_fire3_e1e3s1_forward** (" apu_fire3_e1e3s1_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(912, 5)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 5)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek

_size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(5), __identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(6), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 5)))

- KERNEL_INFO **apu_fire4_e1e3s1_forward** (" apu_fire4_e1e3s1_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(912, 5)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 5)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(5), __identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(6), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 5)))
- KERNEL_INFO **apu_fire5_e1e3mps1_forward** (" apu_fire5_e1e3mps1_forward ", 10, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1808, 4)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 16)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(4), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(9, 1)), __port(__index(5), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(112, 3)), __port(__index(6), __identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(112, 3)), __port(__index(7), __identifier("OUTPUT_MP1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(8), __identifier("OUTPUT_MP3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(9), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(28, 2)))
- KERNEL_INFO **apu_fire6_e1e3s1_forward** (" apu_fire6_e1e3s1_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(912, 3)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 16)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(28, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(5), __identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)), __port(__index(6), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(28, 3)))
- KERNEL_INFO **apu_fire7_e1e3s1_forward** (" apu_fire7_e1e3s1_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1360, 3)), __port(__index(1), __identifier("INPUT_WEIGHT"),

__attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __↩ e0_size(1, 1), __ek_size(64, 31)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_↩ ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek↩ _size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OU↩ T_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(28, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __↩ spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(84, 1)), __port(__index(5), __↩ identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(84, 1)), __port(__index(6), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(28, 3)))

- KERNEL_INFO **apu_fire8_e1e3s1_forward** (" apu_fire8_e1e3s1_forward ", 7, __port(__index(0), __↩ identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0_↩ data_type(d08s), __e0_size(1, 1), __ek_size(1360, 3)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __↩ e0_size(1, 1), __ek_size(64, 31)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_↩ ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek↩ _size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OU↩ T_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(28, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __↩ spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(84, 1)), __port(__index(5), __↩ identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(84, 1)), __port(__index(6), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(28, 3)))
- KERNEL_INFO **apu_fire9_1st_e1e3mp_forward** (" apu_fire9_1st_e1e3mp_forward ", 8, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0↩ _data_type(d08s), __e0_size(1, 1), __ek_size(1808, 2)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __↩ e0_size(1, 1), __ek_size(64, 45)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_A↩ TTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_↩ size(16, 1)), __port(__index(3), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_↩ OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(9, 1)), __port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __↩ spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(112, 3)), __port(__index(5), _↩ _identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(112, 3)), __port(__index(6), __identifier("OUTPUT_M↩ P1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0↩ _size(1, 1), __ek_size(56, 1)), __port(__index(7), __identifier("OUTPUT_MP3"), __attributes(ACF_ATTR_↩ VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 1)))
- KERNEL_INFO **apu_fire9_2nd_s1_forward** (" apu_fire9_2nd_s1_forward ", 5, __port(__index(0), __↩ identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_↩ data_type(d08s), __e0_size(1, 1), __ek_size(7184, 1)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __↩ e0_size(1, 1), __ek_size(64, 9)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_A↩ TTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_↩ size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_↩ STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(14, 1)), __↩ port(__index(4), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(14, 1)))
- KERNEL_INFO **apu_fire10_1st_e1e3_forward** (" apu_fire10_1st_e1e3_forward ", 6, __port(__index(0), ↩ __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 1, 1), __e0↩ _data_type(d08s), __e0_size(1, 1), __ek_size(912, 2)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __↩ e0_size(1, 1), __ek_size(64, 45)), __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_A↩ TTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_↩ size(16, 1)), __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_↩ STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(14, 1)), __↩

port(__index(4), __identifier("OUTPUT_E1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 2)), __port(__index(5), __identifier("OUTPUT_E3"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 2)))

- KERNEL_INFO **apu_fire10_2nd_conv10ap_top_forward** (" apu_fire10_2nd_conv10ap_top_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(7168, 6)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 9)), __port(__index(2), __identifier("INPUT_ACCM"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(4), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(14, 1)), __port(__index(5), __identifier("OUTPUT_CONV10"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(14, 1)), __port(__index(6), __identifier("OUTPUT_AP"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO **apu_fire10_2nd_conv10ap_bottom_forward** (" apu_fire10_2nd_conv10ap_bottom_forward ", 7, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(7168, 7)), __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 9)), __port(__index(2), __identifier("INPUT_ACCM"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(16, 1)), __port(__index(4), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(14, 1)), __port(__index(5), __identifier("OUTPUT_CONV10"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(14, 1)), __port(__index(6), __identifier("OUTPUT_AP"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

7.36.1 Detailed Description

ACF metadata and wrapper function for the Convolutional Neural Networks.

-
-

7.36.2 Function Documentation

7.36.2.1 KERNEL_INFO apu_fire2_conv1mps1_forward (" apu_fire2_conv1mps1_forward " , 8 , __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_SCL_IN_FIXED), __spatial_dep(0, 0, 2, 3), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(700, 12)) , __port(__index(1), __identifier("INPUT_WEIGHT"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(64, 7)) , __port(__index(2), __identifier("INPUT_PARAMS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(16, 1)) , __port(__index(3), __identifier("OUTPUT_ROW_BUF"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(112, 1)) , __port(__index(4), __identifier("OUTPUT_SCRATCH_BUF"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(9, 1)) , __port(__index(5), __identifier("OUTPUT_CONV1"), __attributes(ACF_ATTR_VEC_OUT_STATI↩C_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(222, 3)) , __port(__index(6), __identifier("OUTPUT_MP"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(112, 1)) , __port(__index(7), __identifier("OUTPUT_S1"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(56, 3)))

ACF metadata for Convolutional Neural Networks.

**

See also

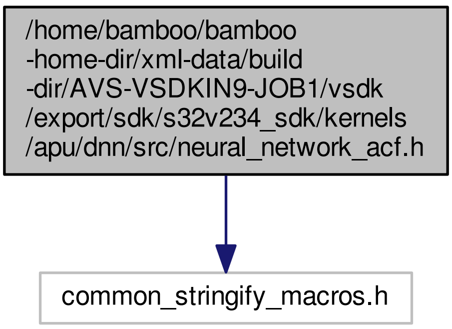
UG-10267-03 ACF User Guide, Section 3.2.2

7.37 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/dnn/src/neural_network_acf.h File Reference

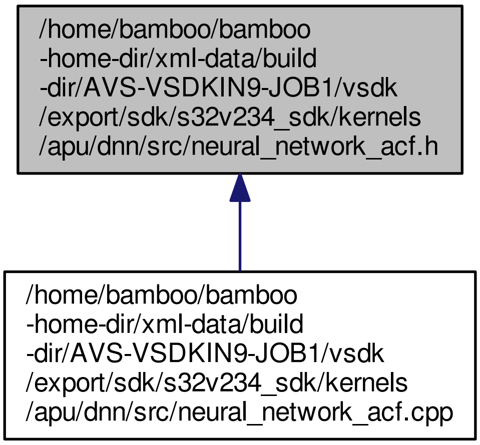
ACF metadata and wrapper function for the Convolutional Neural Networks.

```
#include "common_stringify_macros.h"
```

Include dependency graph for neural_network_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **APU_FIRE2_CONV1MPS1_FORWARD_K** apu_fire2_conv1mps1_forward
- #define **APU_FIRE2_CONV1MPS1_FORWARD_KN** XSTR(APU_FIRE2_CONV1MPS1_FORWARD_K)
- #define **APU_FIRE3_E1E3S1_FORWARD_K** apu_fire3_e1e3s1_forward
- #define **APU_FIRE3_E1E3S1_FORWARD_KN** XSTR(APU_FIRE3_E1E3S1_FORWARD_K)
- #define **APU_FIRE4_E1E3S1_FORWARD_K** apu_fire4_e1e3s1_forward
- #define **APU_FIRE4_E1E3S1_FORWARD_KN** XSTR(APU_FIRE4_E1E3S1_FORWARD_K)
- #define **APU_FIRE5_E1E3MPS1_FORWARD_K** apu_fire5_e1e3mps1_forward
- #define **APU_FIRE5_E1E3MPS1_FORWARD_KN** XSTR(APU_FIRE5_E1E3MPS1_FORWARD_K)
- #define **APU_FIRE6_E1E3S1_FORWARD_K** apu_fire6_e1e3s1_forward
- #define **APU_FIRE6_E1E3S1_FORWARD_KN** XSTR(APU_FIRE6_E1E3S1_FORWARD_K)
- #define **APU_FIRE7_E1E3S1_FORWARD_K** apu_fire7_e1e3s1_forward
- #define **APU_FIRE7_E1E3S1_FORWARD_KN** XSTR(APU_FIRE7_E1E3S1_FORWARD_K)
- #define **APU_FIRE8_E1E3S1_FORWARD_K** apu_fire8_e1e3s1_forward
- #define **APU_FIRE8_E1E3S1_FORWARD_KN** XSTR(APU_FIRE8_E1E3S1_FORWARD_K)
- #define **APU_FIRE9_1ST_E1E3MP_FORWARD_K** apu_fire9_1st_e1e3mp_forward
- #define **APU_FIRE9_1ST_E1E3MP_FORWARD_KN** XSTR(APU_FIRE9_1ST_E1E3MP_FORWARD_K)
- #define **APU_FIRE9_2ND_S1_FORWARD_K** apu_fire9_2nd_s1_forward
- #define **APU_FIRE9_2ND_S1_FORWARD_KN** XSTR(APU_FIRE9_2ND_S1_FORWARD_K)
- #define **APU_FIRE10_1ST_E1E3_FORWARD_K** apu_fire10_1st_e1e3_forward
- #define **APU_FIRE10_1ST_E1E3_FORWARD_KN** XSTR(APU_FIRE10_1ST_E1E3_FORWARD_K)
- #define **APU_FIRE10_2ND_CONV10AP_TOP_FORWARD_K** apu_fire10_2nd_conv10ap_top_forward
- #define **APU_FIRE10_2ND_CONV10AP_TOP_FORWARD_KN** XSTR(APU_FIRE10_2ND_CONV10AP_TOP_FORWARD_K)
- #define **APU_FIRE10_2ND_CONV10AP_BOTTOM_FORWARD_K** apu_fire10_2nd_conv10ap_bottom_forward
- #define **APU_FIRE10_2ND_CONV10AP_BOTTOM_FORWARD_KN** XSTR(APU_FIRE10_2ND_CONV10AP_BOTTOM_FORWARD_K)
- #define **INPUT_IMAGE** "INPUT_IMAGE"
- #define **INPUT_WEIGHT** "INPUT_WEIGHT"

- #define **INPUT_PARAMS** "INPUT_PARAMS"
- #define **OUTPUT_ROW_BUF** "OUTPUT_ROW_BUF"
- #define **OUTPUT_SCRATCH_BUF** "OUTPUT_SCRATCH_BUF"
- #define **OUTPUT_CONV1** "OUTPUT_CONV1"
- #define **OUTPUT_MP** "OUTPUT_MP"
- #define **OUTPUT_S1** "OUTPUT_S1"
- #define **OUTPUT_E1** "OUTPUT_E1"
- #define **OUTPUT_E3** "OUTPUT_E3"
- #define **OUTPUT_MP1** "OUTPUT_MP1"
- #define **OUTPUT_MP3** "OUTPUT_MP3"
- #define **INPUT_ACCM** "INPUT_ACCM"
- #define **OUTPUT_CONV10** "OUTPUT_CONV10"
- #define **OUTPUT_AP** "OUTPUT_AP"

Functions

- **extKernelInfoDecl** (apu_fire2_conv1mps1_forward)
- **extKernelInfoDecl** (apu_fire3_e1e3s1_forward)
- **extKernelInfoDecl** (apu_fire4_e1e3s1_forward)
- **extKernelInfoDecl** (apu_fire5_e1e3mps1_forward)
- **extKernelInfoDecl** (apu_fire6_e1e3s1_forward)
- **extKernelInfoDecl** (apu_fire7_e1e3s1_forward)
- **extKernelInfoDecl** (apu_fire8_e1e3s1_forward)
- **extKernelInfoDecl** (apu_fire9_1st_e1e3mp_forward)
- **extKernelInfoDecl** (apu_fire9_2nd_s1_forward)
- **extKernelInfoDecl** (apu_fire10_1st_e1e3_forward)
- **extKernelInfoDecl** (apu_fire10_2nd_conv10ap_top_forward)
- **extKernelInfoDecl** (apu_fire10_2nd_conv10ap_bottom_forward)

7.37.1 Detailed Description

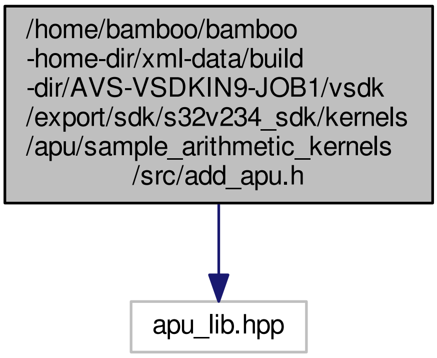
ACF metadata and wrapper function for the Convolutional Neural Networks.

7.38 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/add_apu.h File Reference

element-wise addition implementation for APEX

`#include "apu_lib.hpp"`

Include dependency graph for add_apu.h:



Functions

- void add (vec16u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit addition => unsigned 16bit.
- void add_in16s_out32s (vec32s ∗dst, vec16s ∗srcImage0, vec16s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 16bit addition => signed 32bit.
- void add_in32s_out32s (vec32s ∗dst, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit addition => signed 32bit.
- void add_in32u_out32u (vec32u ∗dst, vec32u ∗srcImage0, vec32u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 32bit addition => unsigned 32bit.
- void add_in64s_out64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcImage0_high, vec32u ∗srcImage0_low, vec32s ∗srcImage1_high, vec32u ∗srcImage1_low, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 64bit addition => signed 64bit.
- void add_in64u_out64u (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗srcImage0_high, vec32u ∗srcImage0_low, vec32u ∗srcImage1_high, vec32u ∗srcImage1_low, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 64bit addition => unsigned 64bit.
- void add_in32Q3_28_out32Q3_28 (vec32s ∗dstInt, vec32s ∗dstFrac, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise 32bit Q3.28 addition => 64bit in [Q3.28] (integer part) and {Q3.28} (fractional part) format.

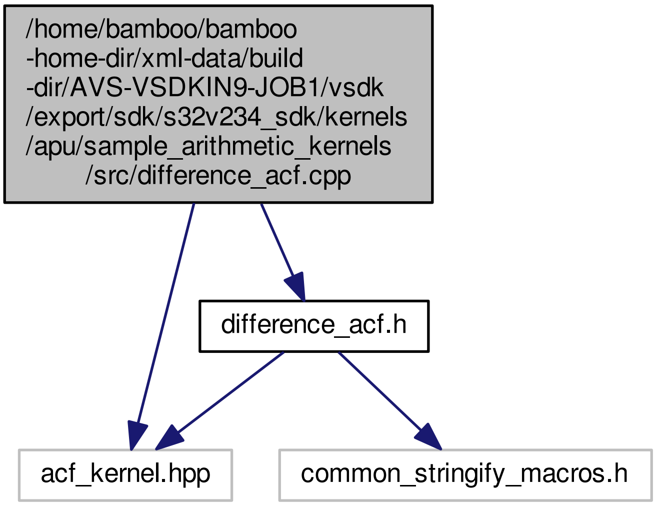
7.38.1 Detailed Description

element-wise addition implementation for APEX

7.39 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/difference_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "difference_acf.h"
```

Include dependency graph for difference_acf.cpp:



Functions

- KERNEL_INFO apu_diff_in08u_out16s (" apu_diff_in08u_out16s ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_diff_in16s_out16s (" apu_diff_in16s_out16s ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_diff_in16s_out32s (" apu_diff_in16s_out32s ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_diff_in32s_out32s (" apu_diff_in32s_out32s ", 3, __port(__index(0), __identifier("INPUT_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

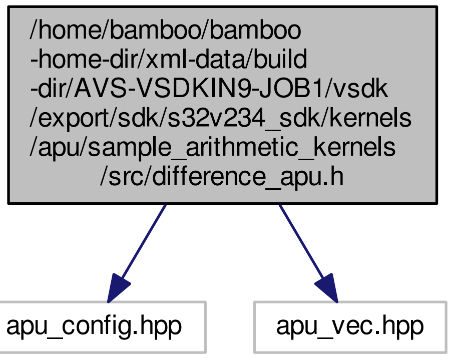
- KERNEL_INFO apu_diff_in64s_out64s (" apu_diff_in64s_out64s ", 6, __port(__index(0), __identifier("INPUT_A_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_A_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_B_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("INPUT_B_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("OUTPUT_HIGH"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("OUTPUT_LOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

7.40 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/difference_apu.h File Reference

Computes the pixelwise difference btw. two images.

```
#include "apu_config.hpp"
#include "apu_vec.hpp"
```

Include dependency graph for difference_apu.h:



Functions

- void difference_filter_in08u_out16s (vec16s ∗dst, vec08u ∗srcA, vec08u ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Pixel-wise difference between two unsigned 8bit images => signed 16bit.
- void difference_filter_in16s_out16s (vec16s ∗dst, vec16s ∗srcA, vec16s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Pixel-wise difference between two signed 16bit images => signed 16bit.
- void difference_filter_in16s_out32s (vec32s ∗dst, vec16s ∗srcA, vec16s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Pixel-wise difference between two signed 16bit images => signed 32bit.
- void difference_filter_in32s_out32s (vec32s ∗dst, vec32s ∗srcA, vec32s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Pixel-wise difference between two signed 16bit images => signed 32bit.

- void difference_filter_in32u_out32s (vec32s ∗dst, vec32u ∗srcA, vec32u ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Pixel-wise difference between two unsigned 32bit images => signed 32bit.

- void difference_filter_in32s_out64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA, vec32s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Pixel-wise difference between two signed 32bit images => signed 64bit.

- void difference_filter_in64s_out64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA_high, vec32u ∗srcA_low, vec32s ∗srcB_high, vec32u ∗srcB_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Pixel-wise difference between two signed 64bit images => signed 64bit.

- void difference_filter_in64u_out64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32u ∗srcA_high, vec32u ∗srcA_low, vec32u ∗srcB_high, vec32u ∗srcB_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

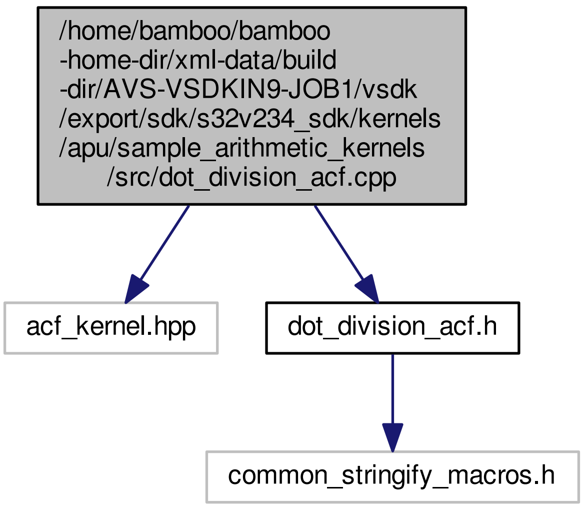
    Pixel-wise difference between two unsigned 64bit images => signed 64bit.

7.40.1 Detailed Description

Computes the pixelwise difference btw. two images.

7.41 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/dot_division_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "dot_division_acf.h"
```
Include dependency graph for dot_division_acf.cpp:



Functions

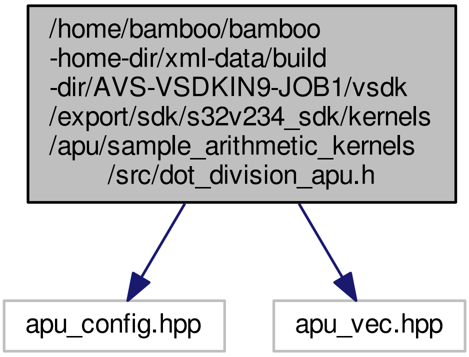
- KERNEL_INFO apu_dot_division (" apu_dot_division ", 3, __port(__index(0), __identifier("INPUT_A"), __↩ attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_↩ size(1, 1)), __port(__index(1), __identifier("INPUT_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPU↩ T"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_log2 (" apu_dot_log2 ", 2, __port(__index(0), __identifier("Log2_input"), __↩ attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek↩ _size(1, 1)), __port(__index(1), __identifier("Log2_fact"), __attributes(ACF_ATTR_VEC_OUT), __spatial_↩ dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_inv_NewtonRaphson (" apu_dot_inv_NewtonRaphson ", 4, __port(__index(0), __↩ identifier("Inv_Inverse_divisor"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data↩ _type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("Inv_divisor"), __attributes(A↩ CF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), ↩ __port(__index(2), __identifier("Inv_log2fact"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("Inv_shiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __↩ e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_division_N64s_D32s_Q64s (" apu_dot_division_N64s_D32s_Q64s ", 6, __port(↩ __index(0), __identifier("Div_N64s_NOM_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("Div_N64s_N↩ OM_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Div_N64s_DIVISOR"), __attributes(ACF_ATT↩ R_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__↩ index(3), __identifier("Div_N64s_OUT_HIGH"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("Div_N64s_O↩ UT_LOW"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_↩ size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("Div_N64s_OUT_REM"), __attributes(ACF_AT↩ TR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

7.42 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/dot_division_apu.h File Reference

Element-wise division of two vectors/matrices. If divisor is zero, a value of zero is returned (in order not to influence following arithmetics).

```
#include "apu_config.hpp"
#include "apu_vec.hpp"
```

Include dependency graph for dot_division_apu.h:



Functions

- void dot_division_filter (vec32s ∗res, vec32s ∗numerator, vec32s ∗denominator, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise division of signed 32bit integers.
- void computeLog2 (vec08u ∗log2Fact, vec32s ∗input, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  res[i] = log2(abs(input[i])) + 1, where input is a signed 32bit integer
- void computeLog2u (vec08u ∗log2Fact, vec32u ∗input, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  res[i] = log2(input[i]) + 1, where input is a unsigned 32bit integer
- void compute64bitLog2 (vec08u ∗log2Fact, vec32s ∗input_high, vec32u ∗input_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  res[i] = log2(abs(input[i])) + 1, where input is a signed 64bit integer
- void compute64bitLog2u (vec08u ∗log2Fact, vec32u ∗input_high, vec32u ∗input_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  res[i] = log2(input[i]) + 1, where input is a unsigned 64bit integer
- void computeInv_NewtonRaphson (vec32s ∗invDiv, vec32s ∗div, vec08u ∗log2Fact, const int08s shiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise inverse of a signed integer vector using the NewtonRaphson algorithm.
- void dot_division_filter_N64s_D32s_Q64s (vec32s ∗dst_high, vec32u ∗dst_low, vec32u ∗dst_rem, vec32s ∗nom_high, vec32u ∗nom_low, vec32s ∗divisor, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise division of a 64bit nominator by a 32bit divisor.

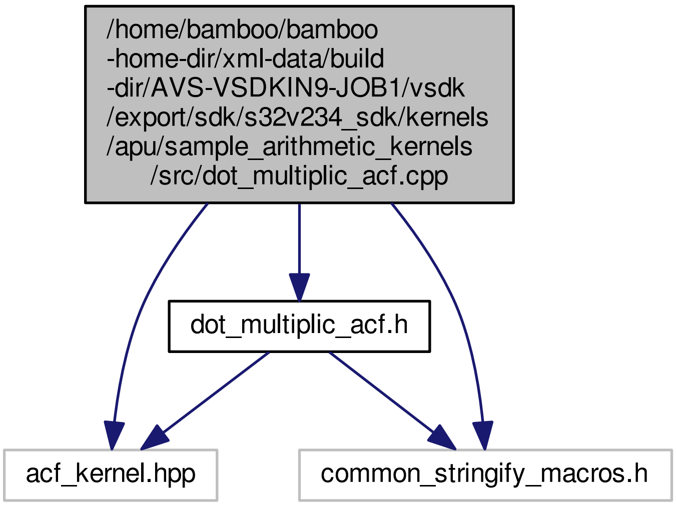
7.42.1 Detailed Description

Element-wise division of two vectors/matrices. If divisor is zero, a value of zero is returned (in order not to influence following arithmetics).

7.43 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/dot_multiplic_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "common_stringify_macros.h"
#include "dot_multiplic_acf.h"
```

Include dependency graph for dot_multiplic_acf.cpp:



Functions

- KERNEL_INFO apu_dot_mult_in16s_out32s (" apu_dot_mult_in16s_out32s ", 3, __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_in32s_out32s (" apu_dot_mult_in32s_out32s ", 3, __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_in32s_out64s (" apu_dot_mult_in32s_out64s ", 4, __port(__index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("DotMultKn_OUT_High"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("DotMultKn_OUT_Low"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO apu_dot_mult_in32s_in16s_out32s (" apu_dot_mult_in32s_in16s_out32s ", 3, __port(__↩ index(0), __identifier("DotMultKn_IN_A"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), _↩ _e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DotMultKn_IN_B"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __↩ ek_size(1, 1)), __port(__index(2), __identifier("DotMultKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), _↩ _spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_scalar_in08u_out16s (" apu_dot_mult_scalar_in08u_out16s ", 3, __port(_↩ _index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __↩ e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultScalKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), _↩ _ek_size(1, 1)), __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL↩ _IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_mult_scalar_in32s_out32s (" apu_dot_mult_scalar_in32s_out32s ", 3, __port(_↩ _index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __↩ e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultScalKn_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), _↩ _ek_size(1, 1)), __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL↩ _IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_left_shift_in16u_out16s (" apu_dot_left_shift_in16u_out16s ", 3, __port(__index(0), __identifier("MultScalKn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data↩ _type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultScalKn_OUT"), __↩ attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), _↩ _ek_size(1, 1)), __port(__index(2), __identifier("MultScalKn_IN_SCALAR"), __attributes(ACF_ATTR_SCL↩ _IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_right_shift_in64s_out64s (" apu_dot_right_shift_in64s_out64s ", 5, __port(__↩ index(0), __identifier("RightShift_64bit_InHigh"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("RightShift_↩ 64bit_InLOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0↩ _size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("RightShift_64bit_OutHigh"), __attributes(A↩ CF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("RightShift_64bit_OutLOW"), __attributes(ACF_ATTR_VEC_OUT), __↩ spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __↩ identifier("RghtShft_64bit_ShiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_right_shift_in64s_out32s (" apu_dot_right_shift_in64s_out32s ", 4, __port(__↩ index(0), __identifier("RightShift_64bit_InHigh"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("RightShift_↩ 64bit_InLOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __↩ e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("RightShift_32bit_Out"), __attributes(ACF↩ _ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("RghtShft_64bit_ShiftFact"), __attributes(ACF_ATTR_SCL_IN_STATIC_FI↩ XED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_lsh1_in32s_out32s (" apu_dot_lsh1_in32s_out32s ", 2, __port(__index(0), _↩ _identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_↩ type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultBy2Kn_OUT"), __↩ attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), _↩ _ek_size(1, 1)))
- KERNEL_INFO apu_dot_lsh1_in32s_out64s (" apu_dot_lsh1_in32s_out64s ", 3, __port(__index(0), _↩ _identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_↩ type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultBy2Kn_Output_high"), ↩ __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), _↩ _ek_size(1, 1)), __port(__index(2), __identifier("MultBy2Kn_Output_low"), __attributes(ACF_ATTR_VEC_↩ OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_lsh1_in32s_Q3_28_out64s (" apu_dot_lsh1_in32s_Q3_28_out64s ", 3, __port(↩ __index(0), __identifier("MultBy2Kn_IN"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), _↩ _e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MultBy2_Q3_28↩ _Kn_Output_int"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s),

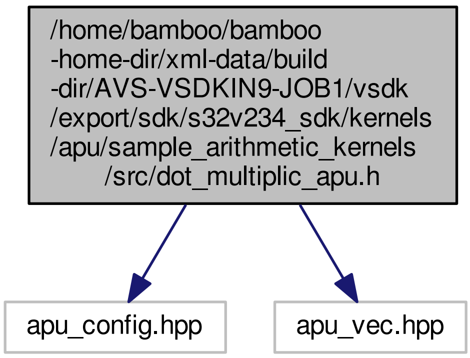
__e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("MultBy2_Q3_28_Kn_Output_frac"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)))

7.44 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/dot_multiplic_apu.h File Reference

Element-wise multiplication of two vectors/matrices.

```
#include "apu_config.hpp"
#include "apu_vec.hpp"
```

Include dependency graph for dot_multiplic_apu.h:



Functions

- vbool hasSign (vec32s &a, vec32s &b)

  sign(a) ∗ sign(b) == -1
- void change64bitSign (vec32s &highWord, vec32u &lowWord)

  In place sign change of a 64bit integer, i.e. a = -a.
- void dot_mult_in16s_out32s_filter (vec32s ∗dst, vec16s ∗srcA, vec16s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 16bit multiplication => signed 32bit.
- void dot_mult_in32s_out32s_filter (vec32s ∗dst, vec32s ∗srcA, vec32s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit multiplication => signed 32bit.
- void dot_mult_in32s_in16s_out32s_filter (vec32s ∗dst, vec32s ∗srcA, vec16s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit and 16bit multiplication => signed 32bit.
- void dot_mult_in32s_out64s_filter (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA, vec32s ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit multiplication => signed 64bit.
- void dot_mult_in32u_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗srcA, vec32u ∗srcB, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

Elementwise unsigned 32bit multiplication => unsigned 64bit.

- void dot_mult_in64s_out64s_filter (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA_high, vec32u ∗srcA_↩ low, vec32s ∗srcB_high, vec32u ∗srcB_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 64bit multiplication => signed 64bit.
- void dot_mult_in64u_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗srcA_high, vec32u ∗srcA_↩ low, vec32u ∗srcB_high, vec32u ∗srcB_low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise unsigned 64bit multiplication => unsigned 64bit.
- void dot_mult_scalar_in08u_out16s_filter (vec16s ∗dst, vec08u ∗srcA, int32s scalar, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise unsigned 8bit multiplication with a fixed scalar => signed 16bit.
- void dot_mult_scalar_in32s_out32s_filter (vec32s ∗dst, vec32s ∗srcA, int32s scalar, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Elementwise signed 32bit multiplication with a fixed scalar => signed 32bit.
- void lsh_in16u_out16s_filter (vec16s ∗upShifted, vec16u ∗src, vec16s ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 16bit left shift with a signed 16bit shift vector => signed 16bit.
- void lsh_in32u_out32u_filter (vec32u ∗upShifted, vec32u ∗src, vec08u ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 32bit left shift with an unsigned 8bit shift vector => unsigned 32bit.
- void lsh_in32s_out32s_filter (vec32s ∗upShifted, vec32s ∗src, vec08u ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 32bit left shift with an unsigned 8bit shift vector => signed 32bit.
- void lsh_in32s_out64s_filter (vec32s ∗upShifted_high, vec32u ∗upShifted_low, vec32s ∗src, vec08u ∗left↩ ShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 32bit left shift with an unsigned 8bit shift vector => signed 64bit.
- void lsh_in32u_out64u_filter (vec32u ∗upShifted_high, vec32u ∗upShifted_low, vec32u ∗src, vec08u ∗left↩ ShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 32bit left shift with an unsigned 8bit shift vector => unsigned 64bit.
- void lsh_in32s_Q3_28_out64s_filter (vec32s ∗upShifted_int, vec32s ∗upShifted_frac, vec32s ∗src, vec08u ∗leftShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 32bit left shift of a 32bit matrix in Q3_28 format with an unsigned 8bit shift vector => signed 64bit.
- void rsh_in32u_out32u_filter (vec32u ∗downShift, vec32u ∗src, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise unsigned 32bit right shift with an unsigned 8bit shift vector => unsigned 32bit.
- void rsh_in32s_out32s_filter (vec32s ∗downShift, vec32s ∗src, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 32bit right shift with a signed 8bit shift vector => signed 32bit.
- void rsh_in64s_out64s_filter (vec32s ∗dst_high, vec32u ∗dst_low, vec32s ∗in_high, vec32u ∗in_low, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 64bit.
- void rsh_in64u_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗in_high, vec32u ∗in_low, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 64bit.
- void rsh_in64s_out32s_filter (vec32s ∗dst, vec32s ∗in_high, vec32u ∗in_low, vec08u ∗rightShiftFact, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

  Chunk-wise signed 64bit right shift with a signed 8bit shift vector => signed 32bit.

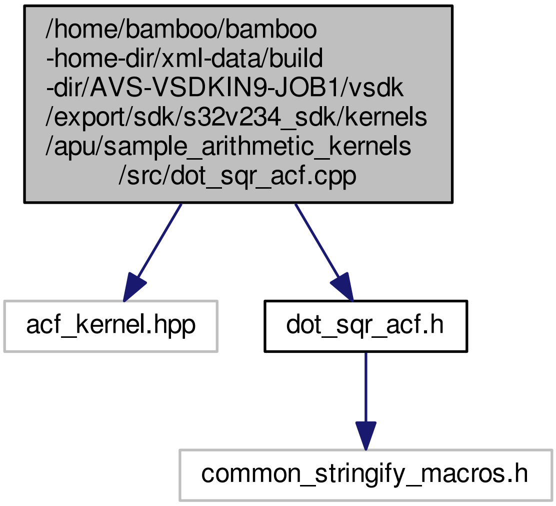
7.44.1 Detailed Description

Element-wise multiplication of two vectors/matrices.

7.45 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/dot_sqr_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "dot_sqr_acf.h"
```

Include dependency graph for dot_sqr_acf.cpp:



Functions

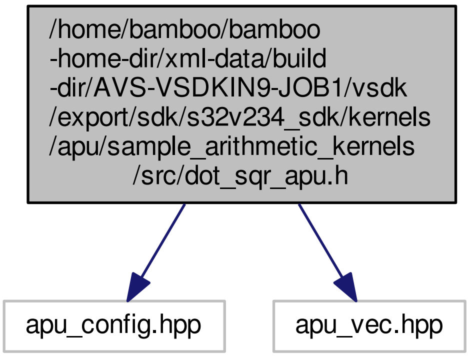
- KERNEL_INFO apu_dot_sqr_in16s_out32u (" apu_dot_sqr_in16s_out32u ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_sqr_in32s_out32u (" apu_dot_sqr_in32s_out32u ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_dot_sqr_in32s_out64u (" apu_dot_sqr_in32s_out64u ", 3, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_High"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_Low"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

7.46 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/dot_sqr_apu.h File Reference

Element-wise square of the input matrix.

```
#include "apu_config.hpp"
#include "apu_vec.hpp"
```

Include dependency graph for dot_sqr_apu.h:



Functions

- vec32u vsqrt_32 (vec32u a)
- void dot_sqr_in16s_out32u_filter (vec32u ∗dst, vec16s ∗srcA, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Elementwise signed 16bit square => unsigned 32bit.
- void dot_sqr_in32s_out32u_filter (vec32u ∗dst, vec32s ∗srcA, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Elementwise signed 32bit square => unsigned 32bit.
- void dot_sqr_in32s_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Elementwise signed 32bit square => unsigned 64bit.
- void dot_sqr_in32u_out64u_filter (vec32u ∗out_high, vec32u ∗out_low, vec32u ∗srcA, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Elementwise unsigned 32bit square => unsigned 64bit.
- void dot_sqr_in64s_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32s ∗srcA_high, vec32u ∗srcA_↩ low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Elementwise signed 64bit square => unsigned 64bit.
- void dot_sqr_in64u_out64u_filter (vec32u ∗dst_high, vec32u ∗dst_low, vec32u ∗srcA_high, vec32u ∗srcA_↩ low, int16s bw, int16s bh, int16s inStrideWidth, int16s outStrideWidth)

    Elementwise unsigned 64bit square => unsigned 64bit.

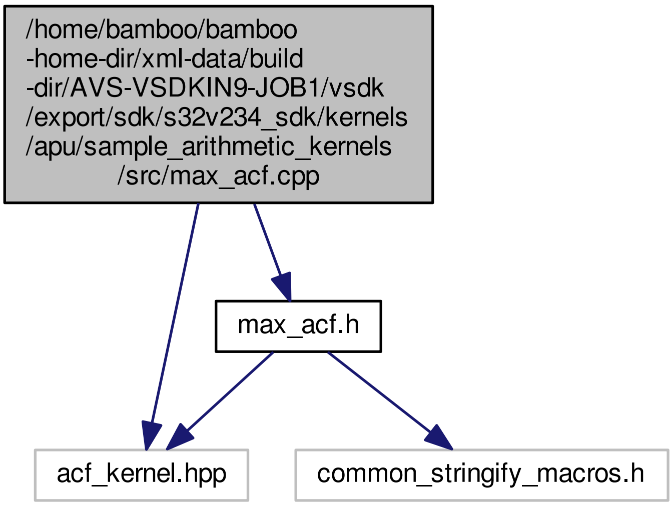
7.46.1 Detailed Description

Element-wise square of the input matrix.

7.47 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/max_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "max_acf.h"
```

Include dependency graph for max_acf.cpp:



Functions

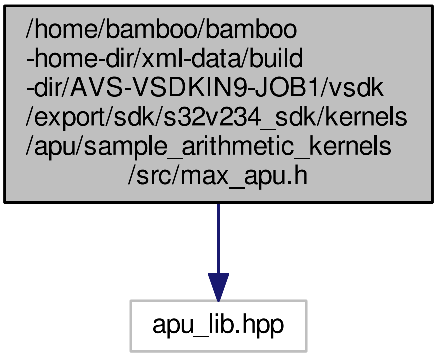
- KERNEL_INFO apu_max (" apu_max ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.48 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_arithmetic_kernels/src/max_apu.h File Reference

element-wise maximum implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for max_apu.h:



Functions

- void max (vec08u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

    Element-wise maximum.

7.48.1 Detailed Description

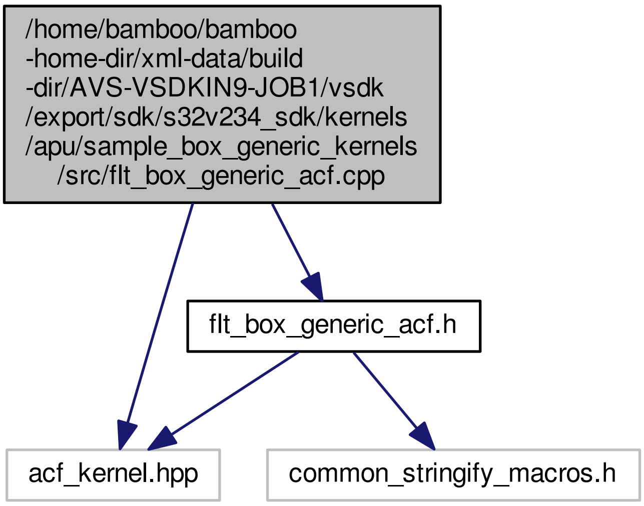
element-wise maximum implementation for APEX

7.49 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_box_generic_kernels/src/flt_box_generic_acf.cpp File Reference

ACF metadata and wrapper function for the harris corner.

```
#include "acf_kernel.hpp"
#include "flt_box_generic_acf.h"
```

Include dependency graph for flt_box_generic_acf.cpp:



Functions

- KERNEL_INFO apu_flt_box_generic_16s (" apu_flt_box_generic_16s ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 8)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 8)))

  ACF metadata for pixel summation.
- KERNEL_INFO harris_box_5x5_16s (" harris_box_5x5_16s ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 8)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 8)))

  ACF metadata for pixel summation.

7.49.1 Detailed Description

ACF metadata and wrapper function for the harris corner.

7.49.2 Function Documentation

7.49.2.1 KERNEL_INFO apu_flt_box_generic_16s (" apu_flt_box_generic_16s " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 8)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 8)))

ACF metadata for pixel summation.

See also

UG-10267-03 ACF User Guide, Section 3.2.2

7.49.2.2 KERNEL_INFO harris_box_5x5_16s (" harris_box_5x5_16s " , 2 , __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 8)) , __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(4, 8)))

ACF metadata for pixel summation.

See also

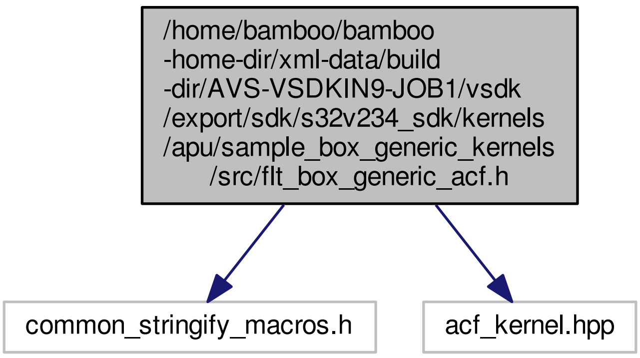
UG-10267-03 ACF User Guide, Section 3.2.2

7.50 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_box_generic_kernels/src/flt_box_generic_acf.h File Reference

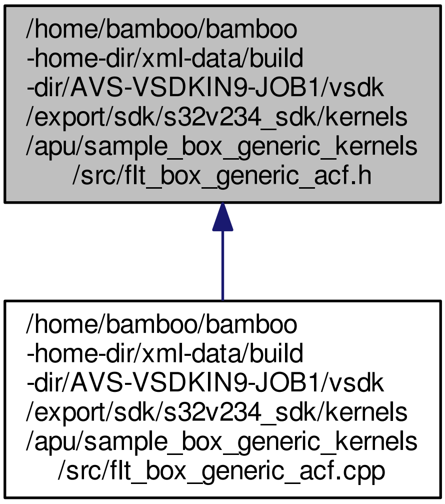
ACF metadata and wrapper function for the harris corner.

```
#include "common_stringify_macros.h"
#include "acf_kernel.hpp"
```

Include dependency graph for flt_box_generic_acf.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **FLT_BOX_GENERIC_K** apu_flt_box_generic_16s
- #define **FLT_BOX_GENERIC_KN** XSTR(FLT_BOX_GENERIC_K)
- #define **HARRIS_BOX_5X5_K** harris_box_5x5_16s
- #define **HARRIS_BOX_5X5_KN** XSTR(HARRIS_BOX_5X5_K)
- #define **FLT_BOX_GENERIC_K_IN** "INPUT_0"
- #define **FLT_BOX_GENERIC_K_OUT** "OUTPUT_0"
- #define **HARRIS_BOX_K_IN** "INPUT_0"
- #define **HARRIS_BOX_K_OUT** "OUTPUT_0"

Functions

- void **apu_flt_box_generic_16s** (kernel_io_desc in0, kernel_io_desc out0)
- void **harris_box_5x5_16s** (kernel_io_desc in0, kernel_io_desc out0)
- **extKernelInfoDecl** (apu_flt_box_generic_16s)
- **extKernelInfoDecl** (harris_box_5x5_16s)

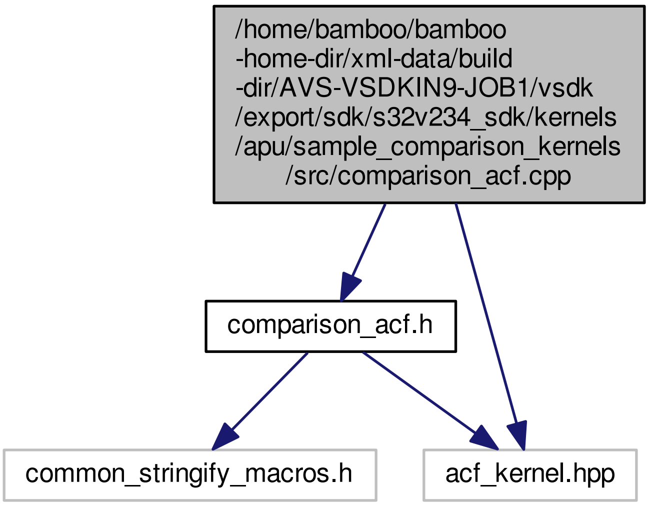
7.50.1 Detailed Description

ACF metadata and wrapper function for the harris corner.

7.51 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_comparison_kernels/src/comparison_acf.cpp File Reference

```
#include "comparison_acf.h"
#include "acf_kernel.hpp"
```

Include dependency graph for comparison_acf.cpp:



Functions

- KERNEL_INFO apu_lower (" apu_lower ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_scalar (" apu_lower_scalar ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_Scalar"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_in16s (" apu_lower_in16s ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_in32s (" apu_lower_in32s ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2),

__identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO apu_abs_lower_in32s (" apu_abs_lower_in32s ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_abs_lower_in32s_scalar16u (" apu_abs_lower_in32s_scalar16u ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lowerEqual_in32s (" apu_lowerEqual_in32s ", 3, __port(__index(0), __identifier("LOWER_KN_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_mask8b (" apu_mask8b ", 3, __port(__index(0), __identifier("MASK_IN_IMG"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MASK_IN_MASK"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("MASK_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_and (" apu_and ", 3, __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_and_in16u_out16u (" apu_and_in16u_out16u ", 3, __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_and_3Pt_in16u_out16u (" apu_and_3Pt_in16u_out16u ", 4, __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("AND_IN_2"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_and_in08u_out16u (" apu_and_in08u_out16u ", 3, __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_and_in08u_in16u_out16u (" apu_and_in08u_in16u_out16u ", 3, __port(__index(0), __identifier("AND_IN_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("AND_IN_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("AND_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO apu_lower_in64s (" apu_lower_in64s ", 5, __port(__index(0), __identifier("LOWER_KN_I↩N_0_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0↩_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_0_LOW"), __attributes(A↩CF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_IN_1_HIGH"), __attributes(ACF_ATTR_VEC_IN), __↩spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __↩identifier("LOWER_KN_IN_1_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_↩data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_lower_in64u (" apu_lower_in64u ", 5, __port(__index(0), __identifier("LOWER_KN↩_IN_0_HIGH"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __↩e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LOWER_KN_IN_0_LOW"), __attributes(A↩CF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LOWER_KN_IN_1_HIGH"), __attributes(ACF_ATTR_VEC_IN), __↩spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __↩identifier("LOWER_KN_IN_1_LOW"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_↩data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("LOWER_KN_OUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.52 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_comparison_kernels/src/comparison_apu.cpp File Reference

compare functions implementation for APEX

7.52.1 Detailed Description

compare functions implementation for APEX

Author

Anca Dima

Version

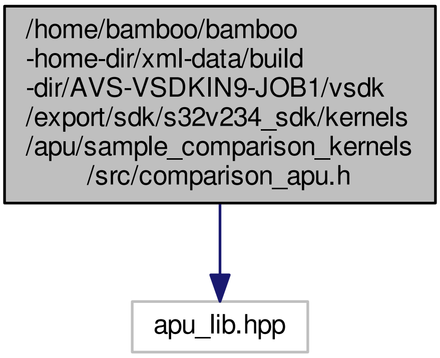
Date

7.53 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_comparison_kernels/src/comparison_apu.h File Reference

element-wise comparison implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for comparison_apu.h:



Functions

- void lower (vbool ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit "<" operation => bool.
- void lower_scalar (vec08u ∗dst, vec08u ∗srcImage, unsigned char scalar, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit "<" operation => bool.
- void lower_in16s (vbool ∗dst, vec16s ∗srcImage0, vec16s ∗srcImage1, int bw, int bh, int inStrideW, int out←StrideW)

  Elementwise signed 16bit "<" operation => bool.
- void lower_in32s (vbool ∗dst, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int out←StrideW)

  Elementwise signed 32bit "<" operation => bool.
- void absLower_in32s (vbool ∗dst, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit "<" operation between the absolute values of the operands => bool.
- void absLower_in32s_scalar16u (vbool ∗dst, vec32s ∗srcImage0, int16u compVal, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit "<" operation between an image and a fixed unsigned 16bit scalar => bool.
- void lowerEqual_in32s (vbool ∗dst, vec32s ∗srcImage0, vec32s ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 32bit "<=" operation => bool.
- void lower_in64u (vbool ∗dst, vec32u ∗srcImage0_high, vec32u ∗srcImage0_low, vec32u ∗srcImage1_high, vec32u ∗srcImage1_low, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 64bit "<" operation => bool.
- void lower_in64s (vbool ∗dst, vec32s ∗srcImage0_high, vec32u ∗srcImage0_low, vec32s ∗srcImage1_high, vec32u ∗srcImage1_low, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise signed 64bit "<" operation => bool.
- void mask_kn (vec08u ∗dst, vec08u ∗srcImage, vec08u ∗srcMask, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit mask operation => unsigned 8bit.
- void and_kn (vec08u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int out←StrideW)

Elementwise unsigned 8bit "&&" operation => unsigned 8bit.

- void [and_in16u_out16u](#) (vec16u ∗dst, vec16u ∗srcImage0, vec16u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

    Elementwise unsigned 16bit "&&" operation => unsigned 16bit.

- void [and_in08u_out16u](#) (vec16u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int inStrideW, int outStrideW)

    Elementwise unsigned 8bit "&&" operation => unsigned 16bit.

- void [and_in08u_in16u_out16u](#) (vec16u ∗dst, vec08u ∗srcImage0, vec16u ∗srcImage1, int bw, int bh, int in↩ StrideW, int outStrideW)

    Elementwise unsigned 8bit "&&" unsigned 16bit operation => unsigned 16bit.

- void [and_3Pt_in16u_out16u](#) (vec16u ∗dst, vec16u ∗srcImage0, vec16u ∗srcImage1, vec16u ∗srcImage2, int bw, int bh, int inStrideW, int outStrideW)

    Elementwise unsigned 3-point 16bit "&&" operation => unsigned 16bit.

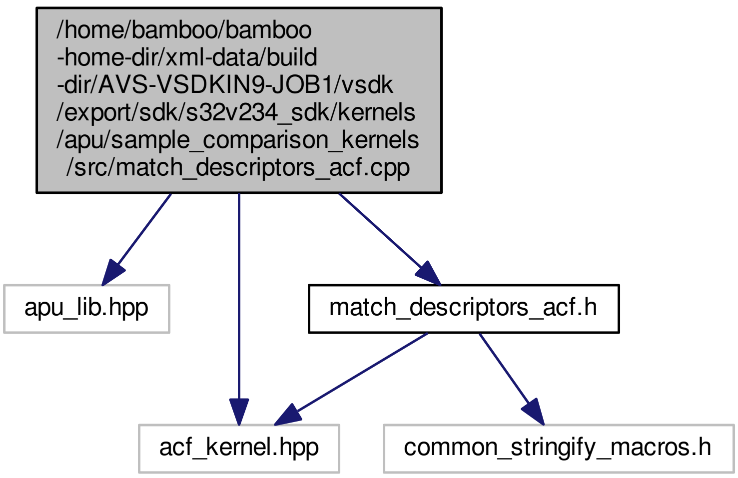
7.53.1 Detailed Description

element-wise comparison implementation for APEX

7.54 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_comparison_kernels/src/match_descriptors_acf.cpp File Reference

```
#include <apu_lib.hpp>
#include <acf_kernel.hpp>
#include "match_descriptors_acf.h"
```

Include dependency graph for match_descriptors_acf.cpp:



Functions

- KERNEL_INFO apu_match_descriptors (" apu_match_descriptors ", 5, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_CONFIG"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(512, 1)), __port(__index(4), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(512, 1)))

7.55 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_comparison_kernels/src/match_descriptors_apu.cpp File Reference

Binary descriptor matching implementation for APEX.

7.55.1 Detailed Description

Binary descriptor matching implementation for APEX.

Author

Igor Aleksandrowicz

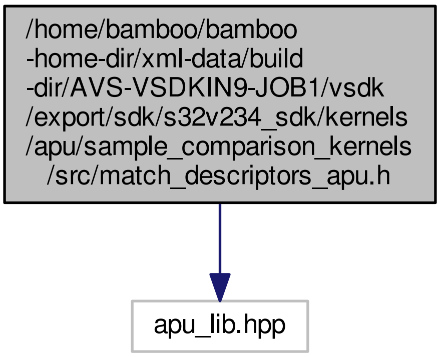
Version

7.56 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_comparison_kernels/src/match_descriptors_apu.h File Reference

binary descriptor matching

```
#include "apu_lib.hpp"
```

Include dependency graph for match_descriptors_apu.h:



Functions

- void Match (const vec08u ∗apcDescriptors0, unsigned int aDescriptor0Count, const vec08u ∗apc←
Descriptors1, unsigned int aDescriptor1Count, int16s ∗apMatches0, int16s ∗apMatches1, int08u aThreshold, int08u aRangeCheck)

    Matches binary descriptors.

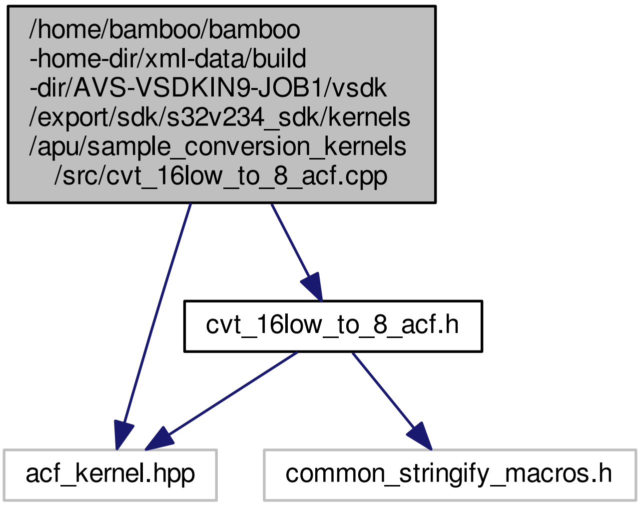
7.56.1 Detailed Description

binary descriptor matching

7.57 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/cvt_16low_to_8_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "cvt_16low_to_8_acf.h"
```

Include dependency graph for cvt_16low_to_8_acf.cpp:



Functions

- KERNEL_INFO apu_16low_to_8 (" apu_16low_to_8 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("APU_16LOWTO8_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.58 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/cvt_16low_to_8_apu.cpp File Reference

16low_to_8 implementation for APEX

7.58.1 Detailed Description

16low_to_8 implementation for APEX

Author

Igor Aleksandrowicz

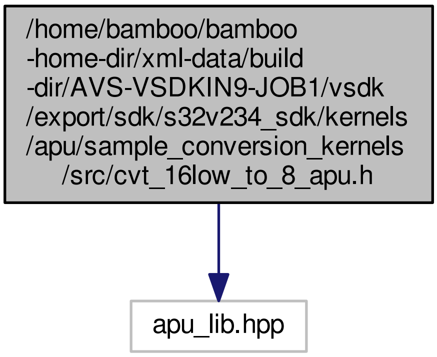
Version

Date

7.59 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/cvt_16low_to_8_apu.h File Reference

```
#include "apu_lib.hpp"
```

Include dependency graph for cvt_16low_to_8_apu.h:



Functions

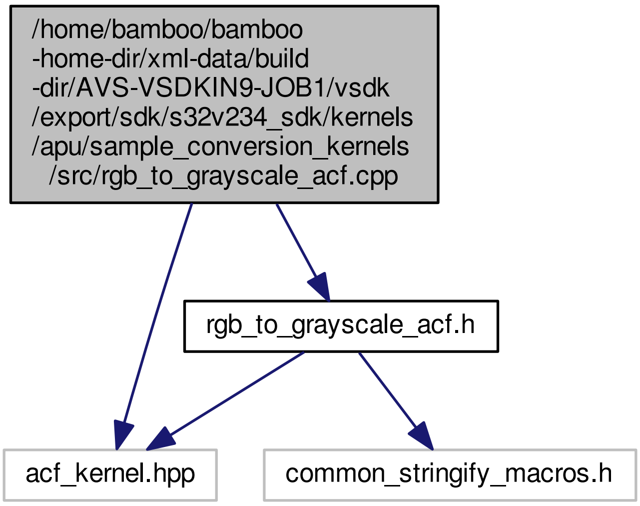
- void f16low_to_8 (vec08u ∗dst, vec16u ∗src, int bw, int bh)

  Extracts the lower bytes.

7.60 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/rgb_to_grayscale_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "rgb_to_grayscale_acf.h"
```

Include dependency graph for rgb_to_grayscale_acf.cpp:



Functions

- KERNEL_INFO apu_rgb_to_grayscale (" apu_rgb_to_grayscale ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.61 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/rgb_to_grayscale_apu.cpp File Reference

rgb_to_grayscale implementation for APEX

7.61.1 Detailed Description

rgb_to_grayscale implementation for APEX

Author

Igor Aleksandrowicz

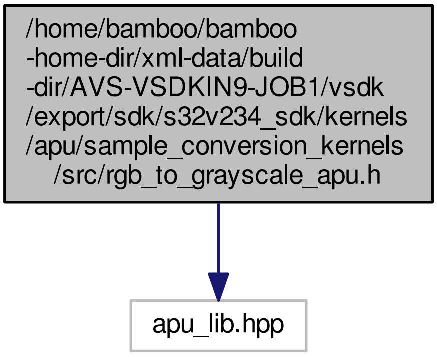
Version

Date

7.62 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/rgb_to_grayscale_apu.h File Reference

```
#include "apu_lib.hpp"
```

Include dependency graph for rgb_to_grayscale_apu.h:



Functions

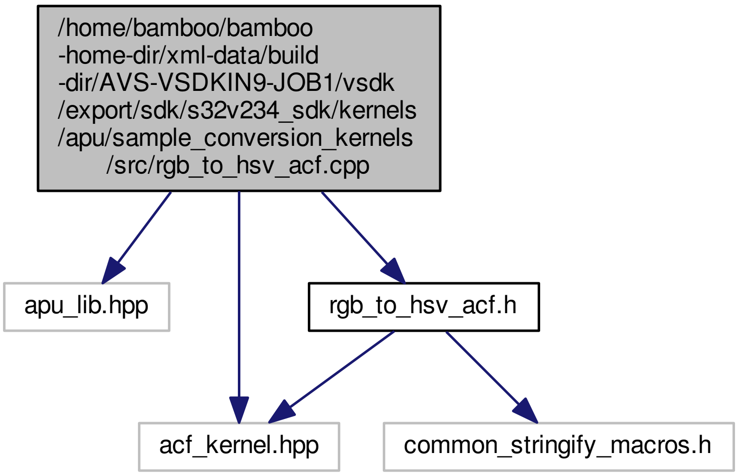
- void rgb_to_grayscale (vec08u ∗apDest, const vec08u ∗apcSrc, int aBlockWidth, int aBlockHeight, int a↩
OutputSpan, int aInputSpan)

  Transforms RGB to grayscale.

7.63 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/rgb_to_hsv_acf.cpp File Reference

```
#include "apu_lib.hpp"
#include "acf_kernel.hpp"
#include "rgb_to_hsv_acf.h"
```

Include dependency graph for rgb_to_hsv_acf.cpp:



Functions

- KERNEL_INFO apu_rgb_to_hsv_sat (" apu_rgb_to_hsv_sat ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_rgb_to_hsv_hue_sat (" apu_rgb_to_hsv_hue_sat ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_HUE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_rgb_to_hsv_hue_sat_grey (" apu_rgb_to_hsv_hue_sat_grey ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_HUE"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUT_GREY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_rgb_to_hsv_svr (" apu_rgb_to_hsv_svr ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_SAT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_VAL"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUT_RED"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.64 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/rgb_to_hsv_apu.cpp File Reference

rgb_to_grayscale implementation for APEX

7.64.1 Detailed Description

rgb_to_grayscale implementation for APEX

Author

Igor Aleksandrowicz

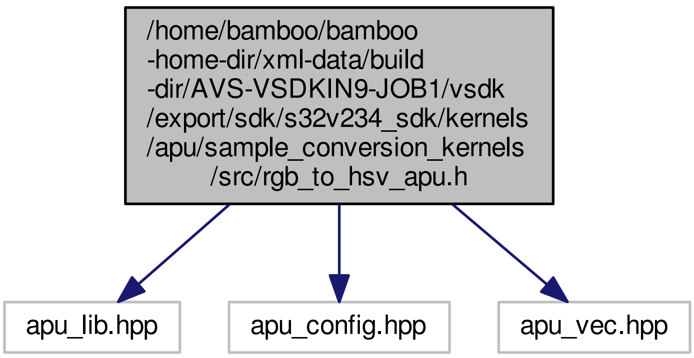
Version

Date

7.65 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_conversion_kernels/src/rgb_to_hsv_apu.h File Reference

```
#include "apu_lib.hpp"
#include "apu_config.hpp"
#include "apu_vec.hpp"
```

Include dependency graph for rgb_to_hsv_apu.h:



Functions

- void rgb_to_hsv_sat (vec08u ∗apSat, const vec08u ∗apcSrc, int aBlockWidth, int aBlockHeight, int aOutputSpan, int aInputSpan)

Transforms RGB to HSV => S.

- void rgb_to_hsv_hue_sat (vec16u ∗apHue, vec08u ∗apSat, const vec08u ∗apcSrc, int aBlockWidth, int a↩ BlockHeight, int aOutputSpan, int aInputSpan)

  Transforms RGB to HSV => (Hue,Sat)

- void rgb_to_hsv_hue_sat_grey (vec16u ∗apHue, vec08u ∗apSat, vec08u ∗grey, const vec08u ∗apcSrc, int aBlockWidth, int aBlockHeight, int aOutputSpan, int aInputSpan)

  Transforms RGB to HSV,Grey => (Hue,Sat, Grey)

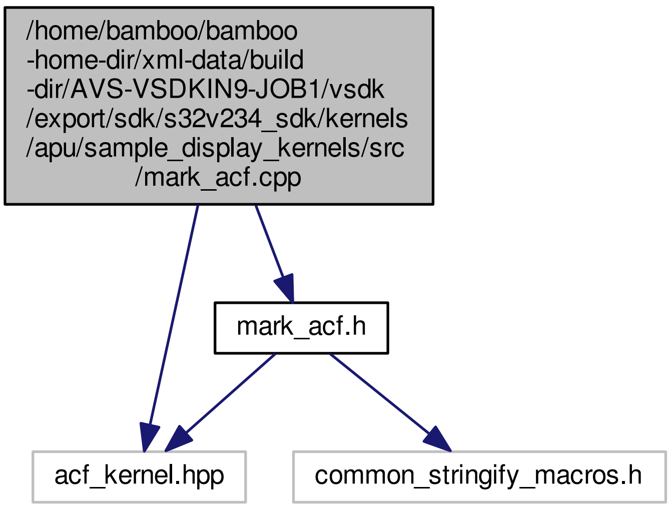
- void rgb_to_hsv_svr (vec08u ∗apSat, vec08u ∗apVal, vec08u ∗apRed, const vec08u ∗apcSrc, int aBlock↩ Width, int aBlockHeight, int aOutputSpan, int aInputSpan)

  Transforms RGB to HSV => (S,V,Red)

7.66 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_display_kernels/src/mark_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "mark_acf.h"
```

Include dependency graph for mark_acf.cpp:



Functions

- KERNEL_INFO apu_mark (" apu_mark ", 3, __port(__index(0), __identifier("INPUT_IMAGE"), __↩ attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), _↩ _ek_size(1, 1)), __port(__index(1), __identifier("INPUT_MARKER"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_↩ type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.67 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_display_kernels/src/mark_apu.cpp File Reference

mark implementation for APEX

7.67.1 Detailed Description

mark implementation for APEX

Author

Igor Aleksandrowicz

Version

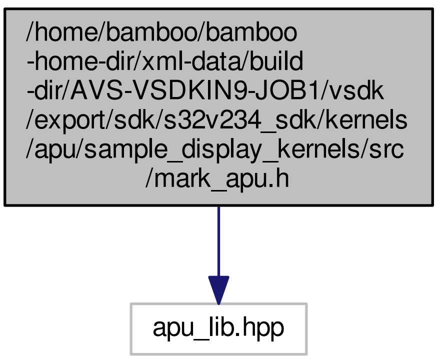
Date

7.68 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_display_kernels/src/mark_apu.h File Reference

image marking implementation for APEX

`#include "apu_lib.hpp"`

Include dependency graph for mark_apu.h:



Functions

- void mark (vec08u ∗dst, vec08u ∗srcImage, vec08u ∗srcMarker, int bw, int bh, int sstride, int mstride)

  Marks the image.

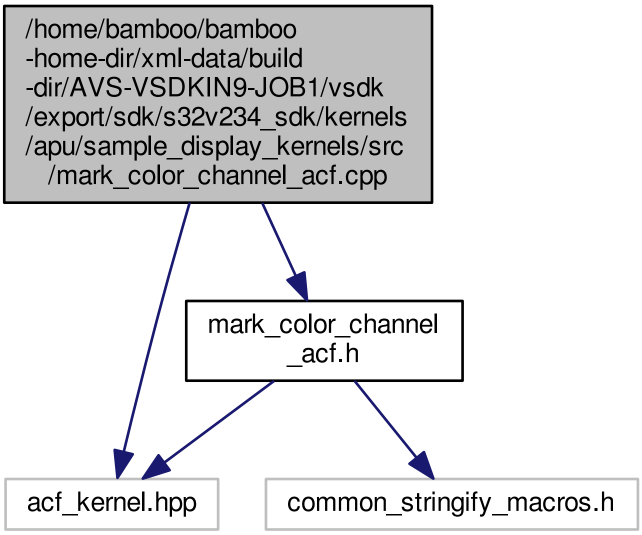
7.68.1 Detailed Description

image marking implementation for APEX

7.69 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_display_kernels/src/mark_color_channel_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "mark_color_channel_acf.h"
```

Include dependency graph for mark_color_channel_acf.cpp:



Functions

- KERNEL_INFO apu_mark_color_channel (" apu_mark_color_channel ", 4, __port(__index(0), __identifier("INPUT_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_MARKER"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("INPUT_CHANNEL_INDEX"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(3, 1), __ek_size(1, 1)))

7.70 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_display_kernels/src/mark_color_channel_apu.cpp File Reference

mark_color_channel implementation for APEX

7.70.1 Detailed Description

mark_color_channel implementation for APEX

Author

Igor Aleksandrowicz

Version

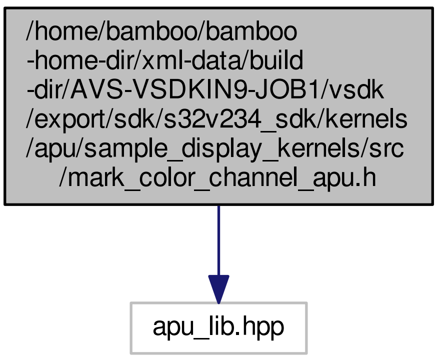
Date

7.71 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_display_kernels/src/mark_color_channel_apu.h File Reference

color channel image marking implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for mark_color_channel_apu.h:



Functions

- void mark_color_channel (vec08u ∗dst, vec08u ∗srcImage, vec08u ∗srcMarker, int bw, int bh, int08u channel, int inStride, int inMarkerStride, int outStride)

Marks a color channel of the image.

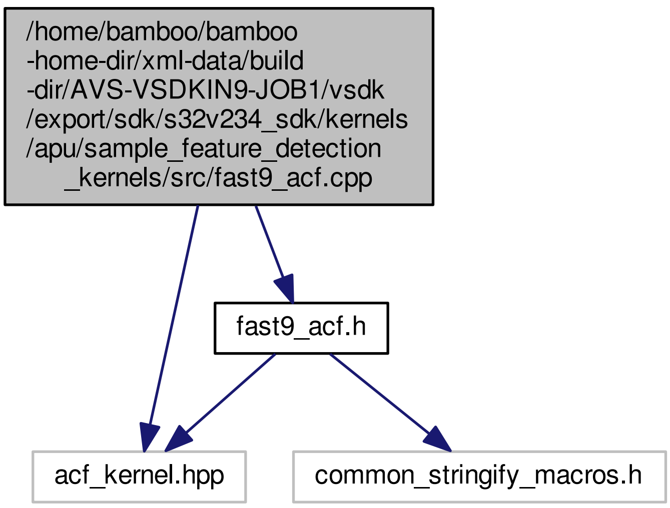
7.71.1 Detailed Description

color channel image marking implementation for APEX

7.72 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_feature_detection_kernels/src/fast9_acf.cpp File Reference

```
#include <acf_kernel.hpp>
#include "fast9_acf.h"
```

Include dependency graph for fast9_acf.cpp:



Functions

- KERNEL_INFO apu_fast9 (" apu_fast9 ", 3, __port(__index(0), __identifier("IN_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("IN_Thr"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.73 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_feature_detection_kernels/src/fast9_apu.cpp File Reference

FAST 9 corner detection implementation for APEX.

7.73.1 Detailed Description

FAST 9 corner detection implementation for APEX.

Author

Igor Aleksandrowicz

Version

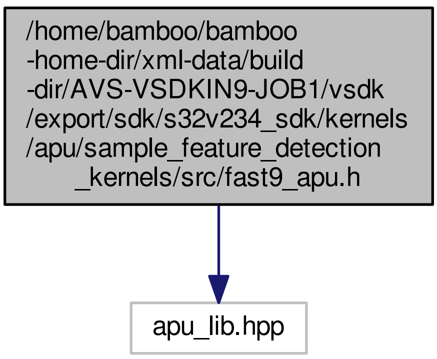
Date

7.74 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_feature_detection_kernels/src/fast9_apu.h File Reference

FAST 9 corner detection implementation for APEX.

`#include "apu_lib.hpp"`

Include dependency graph for fast9_apu.h:



Functions

- void apu_fast9_unsuppressed_score (const vec08u ∗apcSrc, vec08u ∗apDst, int aSourceStride, int a←
DestinationStride, int aBlockWidth, int aBlockHeight, int08u aThreshold)

    FAST9 corner detection.

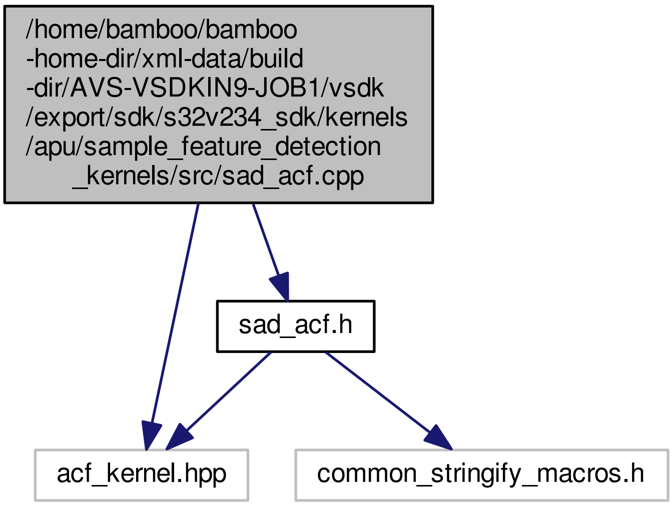
7.74.1 Detailed Description

FAST 9 corner detection implementation for APEX.

7.75 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_feature_detection_kernels/src/sad_acf.cpp File Reference

```
#include <acf_kernel.hpp>
#include "sad_acf.h"
```

Include dependency graph for sad_acf.cpp:



Functions

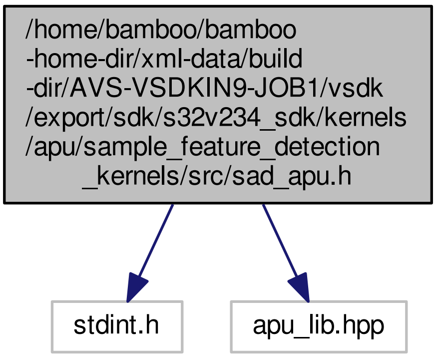
- KERNEL_INFO apu_sad (" apu_sad ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 4)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(8, 8)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(4, 1), __ek_size(1, 1)))

7.76 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_feature_detection_kernels/src/sad_apu.h File Reference

SAD - Sum of absolute Differences implementation for APEX.

```
#include <stdint.h>
#include "apu_lib.hpp"
```

Include dependency graph for sad_apu.h:



Functions

- void apu_sad_impl (vec08u *lpvIn0, int16_t lStrideIn0, int16_t lChunkWidthIn0, int16_t lChunkHeight↩In0, vec08u *lpvIn1, int16_t lStrideIn1, int16_t lChunkWidthIn1, int16_t lChunkHeightIn1, vec32u *lpvOut0, int16_t lStrideOut0, int16_t lChunkWidthOut0, int16_t lChunkHeightOut0)

  Sum of absolute differences. Store the minimum of all differences and the location of the minimum in image0.

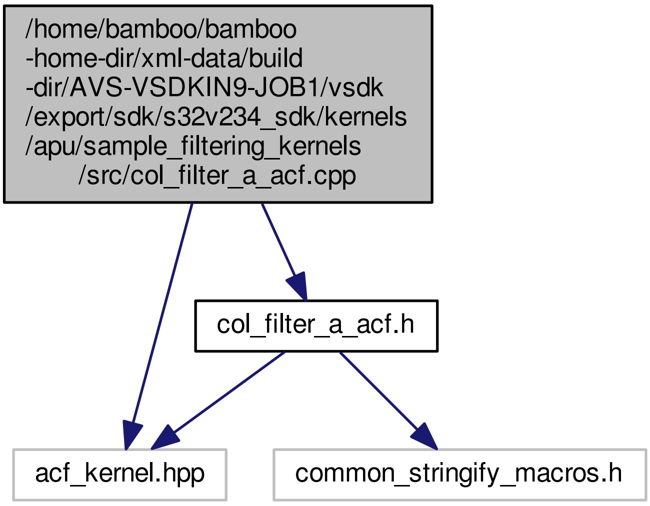
7.76.1 Detailed Description

SAD - Sum of absolute Differences implementation for APEX.

7.77 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/col_filter_a_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "col_filter_a_acf.h"
```

Include dependency graph for col_filter_a_acf.cpp:



Functions

- KERNEL_INFO col_filter (" col_filter ", 3, __port(__index(0), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(COL_PADDING, COL_PADDING, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("COEFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(FILTER_COLS, 1)), __port(__index(2), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Variables

- const int FILTER_COLS = 3
- const int COL_PADDING = FILTER_COLS >> 1

7.78 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/col_filter_a_apu.cpp File Reference

Filtering with general filter image columns - implementation for APEX.

7.78.1 Detailed Description

Filtering with general filter image columns - implementation for APEX.

Author

CGV

Version

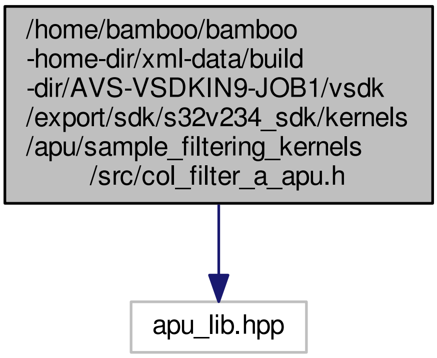
Date

7.79 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/col_filter_a_apu.h File Reference

A column filter implementation for the APU.

`#include "apu_lib.hpp"`

Include dependency graph for col_filter_a_apu.h:



Functions

- void col_filter (vec08u ∗dst, int dstStride, const vec08u ∗src, int srcStride, int rows, int cols, const unsigned char ∗filter, int filterSize, int filterQ)

  Apply a column filter to an image.

7.79.1 Detailed Description

A column filter implementation for the APU.

Filter an image column with a certain filter.

7.79.2 The Column Filter

A column filter is a 1-dimension filter applied to an image where each pixel becomes a weighted sum of itself and neighbouring pixels in the same row. The weighted sum is determined by a set of filter coefficients. The filter are pixel-centered (i.e. the middle coefficient lines up with the pixel under consideration), so the number of coefficients must be odd. For example, consider the column filter with $N_c = 5$ shown below

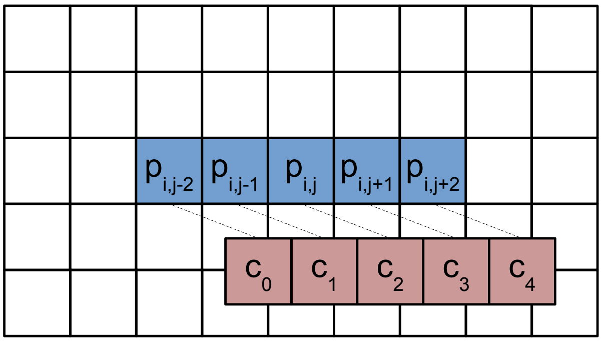


Figure 7.1: A 5-coefficient column filter centered on pixel p_ij

The filter, shown in red, has five columns and a single row of coefficients. If we apply this filter to pixel p_{ij}, the first filter coefficient c_0 lines up with $p_{i,j-2}$, the second coefficient c_1 lines up with $p_{i,j-1}$, and so on. So the weighted sum is

$$p'_{ij} \equiv c_0 p_{i,j-2} + c_1 p_{i,j-1} + c_2 p_{i,j} + c_3 p_{i,j+1} + c_4 p_{i,j+2}$$

In order to filter pixels at the left and right edges of the image, we must pad the image. The number of columns padded on either side of the image is

$$N_{col\ padding} = \text{floor}(N_c/2)$$

and so the total number of columns in the padded image is $N_{cols} + 2N_{col\ padding}$, where N_{cols} is the original number of image columns. Note that each row of the image is filtered independently of the others and the no row padding is required.

The filter coefficients are expressed as 8-bit unsigned fixed point numbers with Q fractional bits, where Q is chosen by the user.

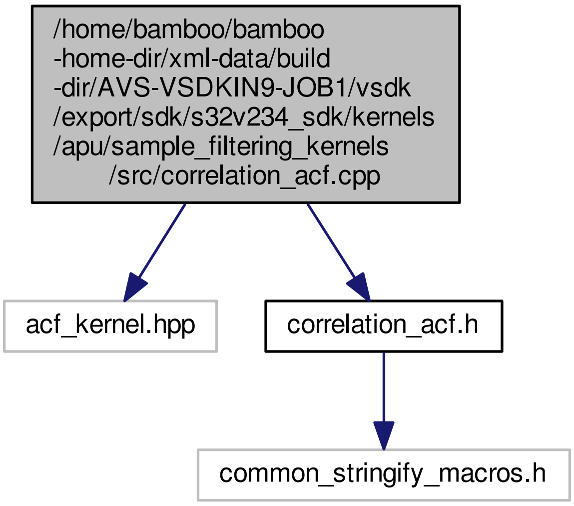
See also

pagFixedPoint

7.80 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/correlation_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "correlation_acf.h"
```

Include dependency graph for correlation_acf.cpp:



Functions

- KERNEL_INFO apu_gradient_x (" apu_gradient_x ", 2, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_gradient_y (" apu_gradient_y ", 2, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_scharr_x (" apu_scharr_x ", 2, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_scharr_y (" apu_scharr_y ", 2, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation (" apu_correlation ", 5, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 3)), __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation_1x3 (" apu_correlation_1x3 ", 5, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_V↩

EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 3)), __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

- KERNEL_INFO apu_correlation_3x1 (" apu_correlation_3x1 ", 5, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 1)), __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation_3x3 (" apu_correlation_3x3 ", 5, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(3, 3)), __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation_5x5 (" apu_correlation_5x5 ", 5, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(5, 5)), __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_correlation_7x7 (" apu_correlation_7x7 ", 5, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(3, 3, 3, 3), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("Corr_IN_Filter"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(7, 7)), __port(__index(3), __identifier("INPUT_FilterScale"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_FiltSymmFl"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

7.81 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/correlation_apu.cpp File Reference

Convolution with general filter 1D/2D.

7.81.1 Detailed Description

Convolution with general filter 1D/2D.

Author

Anca Dima

Version

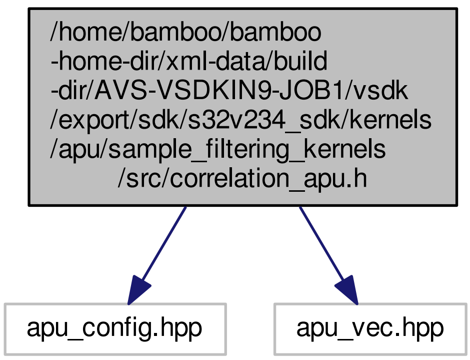
Date

7.82 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/correlation_apu.h File Reference

Correlation with general filter 1D/2D.

```
#include "apu_config.hpp"
#include "apu_vec.hpp"
```

Include dependency graph for correlation_apu.h:



Macros

- #define **inputFiltUpScale** 3
- #define **scharrFiltUpscale** 1

Typedefs

- typedef void(∗ **corrKernelPtr**)(vec16s ∗dst, vec08u ∗src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s ∗filterCoefs)

Functions

- void initFilters ()

  Initializes the array of filter function pointers.
- void performCorrelation (int16u filterFlags, vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s scaleFact, const int16s *filter↩ Coefs)

  Computes time optimized the correlation with a general filter.
- void correlation_filter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filterCoefs)
- void correlation__antisymXfilter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filterCoefs)

  Computes the correlation with an anti-symmetric X filter.
- void correlation__symXfilter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filterCoefs)

  Computes the correlation with a symmetric X filter.
- void correlation__symYfilter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filterCoefs)

  Computes the correlation with a symmetric Y filter.
- void correlation__antisymYfilter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filterCoefs)

  Computes the correlation with an anti-symmetric Y filter.
- void correlation__symXYfilter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filterCoefs)

  Computes the correlation with a symmetric in X and Y filter.
- void correlation__symXantisymYfilter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filter↩ Coefs)

  Computes the correlation with a symmetric in X and anti-symmetric in Y filter.
- void correlation__symXantisymYfilter_16s (vec16s *dst, vec16s *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s scaleFact, const int16s *filter↩ Coefs)

  Computes the 16bit correlation with a symmetric in X and anti-symmetric in Y filter.
- void correlation__antisymXsymYfilter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filter↩ Coefs)

  Computes the correlation with an anti-symmetric in X and symmetric in Y filter.
- void correlation__antisymXsymYfilter_16s (vec16s *dst, vec16s *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filter↩ Coefs)

  Computes the 16bit correlation with an anti-symmetric in X and symmetric in Y filter.
- void correlation__antisymXYfilter (vec16s *dst, vec08u *src, int16s sstr, int16s bw, int16s bh, int16s destBw, int16s xSkip, int16s ySkip, int16u filtWidth, int16u filtHeight, int16s filtScaling, const int16s *filterCoefs)

  Computes the correlation with an anti-symmetric in X and anti-symmetric in Y filter.

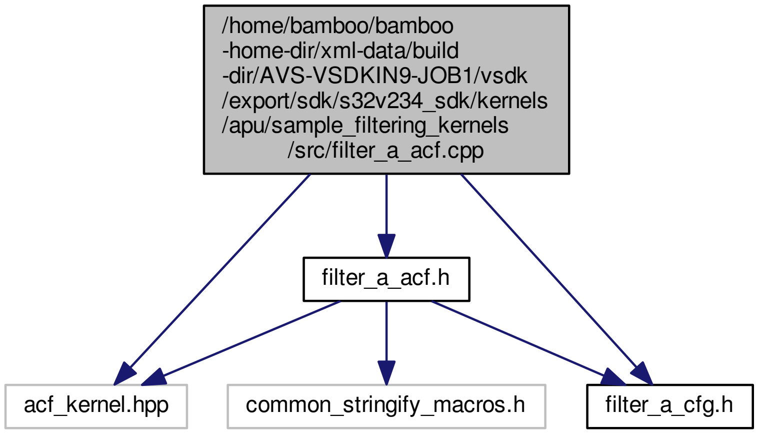
7.82.1 Detailed Description

Correlation with general filter 1D/2D.

7.83 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filter_a_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "filter_a_acf.h"
#include "filter_a_cfg.h"
```

Include dependency graph for filter_a_acf.cpp:



Functions

- KERNEL_INFO apu_filter_a (" apu_filter_a ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(A↩
CF_ATTR_VEC_IN), __spatial_dep(3 >> 1, 3 >> 1, 3 >> 1, 3 >> 1), __e0_data_type(d08u), __e0_size(1,
1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_COEF"), __attributes(ACF_ATTR_SCL_IN_S↩
TATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(3 ∗3, 1)), _↩
_port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0),
__e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.84 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filter_a_apu.cpp File Reference

Filtering with general filter 1D/2D implementation for APEX.

7.84.1 Detailed Description

Filtering with general filter 1D/2D implementation for APEX.

Author

Version

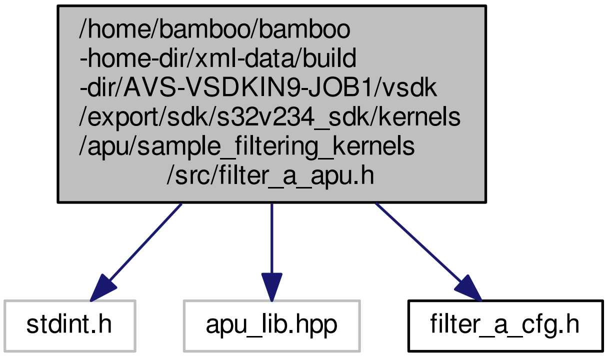
Date

7.85 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filter_a_apu.h File Reference

filtering with general filter 1D/2D

```
#include <stdint.h>
#include "apu_lib.hpp"
#include "filter_a_cfg.h"
```

Include dependency graph for filter_a_apu.h:



Functions

- void apu_filter_a_impl (vec08u ∗src, int16_t sstr, uint8_t ∗coef, vec08u ∗dst, int16_t dstr, int16_t bw, int16_t bh)

  General filtering function with a 1D/2D-filter.

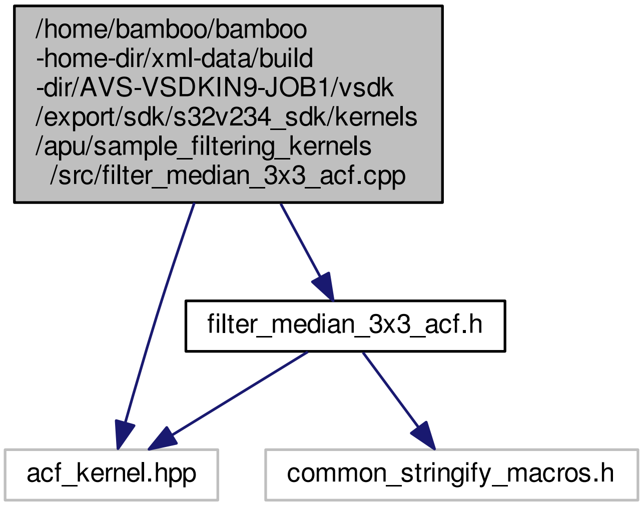
7.85.1 Detailed Description

filtering with general filter 1D/2D

7.86 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filter_median_3x3_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "filter_median_3x3_acf.h"
```

Include dependency graph for filter_median_3x3_acf.cpp:



Functions

- KERNEL_INFO median_3x3_8bpp (" median_3x3_8bpp ", 2, __port(__index(0), __identifier("DST"), __↩ attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __↩ ek_size(1, 1)), __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

  Filtering with a Median 3x3-filter.

7.87 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filter_median_3x3_apu.cpp File Reference

3x3 Median filter implementation for APEX

7.87.1 Detailed Description

3x3 Median filter implementation for APEX

Author

Igor Aleksandrowicz

Version

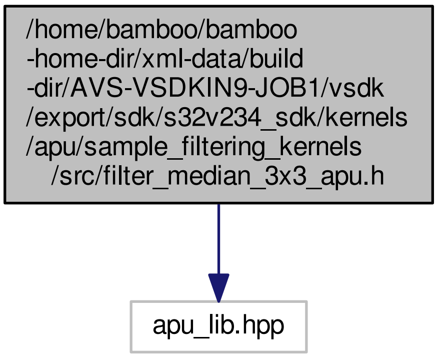
Date

7.88 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filter_median_3x3_apu.h File Reference

3x3 Median filter implementation for APEX

```
#include <apu_lib.hpp>
```

Include dependency graph for filter_median_3x3_apu.h:



Functions

- void apu_flt_median_3x3 (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh)

  Filter with a 3x3 median filter.

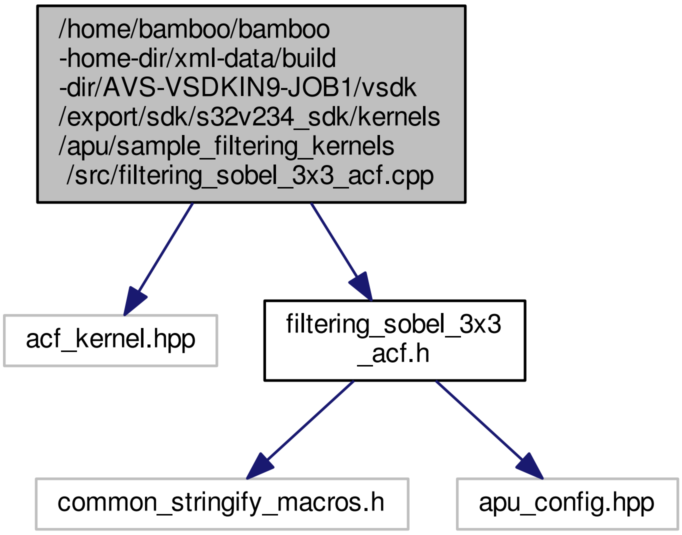
7.88.1 Detailed Description

3x3 Median filter implementation for APEX

7.89 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filtering_sobel_3x3_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "filtering_sobel_3x3_acf.h"
```

Include dependency graph for filtering_sobel_3x3_acf.cpp:



Functions

- KERNEL_INFO sobel_3x3_8bpp (" sobel_3x3_8bpp ", 2, __port(__index(0), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

  Calculate sum of absolute values of first order derivatives x and y using sobel 3x3.

7.90 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filtering_sobel_3x3_apu.cpp File Reference

3x3 Sobel filter implementation for APEX

7.90.1 Detailed Description

3x3 Sobel filter implementation for APEX

Author

Anca Dima

Version

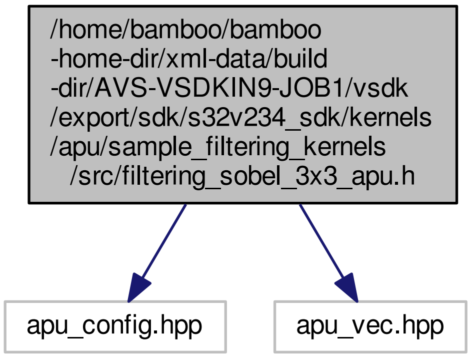
Date

7.91 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/filtering_sobel_3x3_apu.h File Reference

3x3 Sobel filter implementation for APEX

```
#include "apu_config.hpp"
#include "apu_vec.hpp"
```

Include dependency graph for filtering_sobel_3x3_apu.h:



Functions

- void apu_flt_sobel_3x3_x (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh)

  Calculate first order derivative x using sobel 3x3.
- void apu_flt_sobel_3x3_y (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh)

  Calculate first order derivative y using sobel 3x3.
- void apu_flt_sobel_3x3 (vec08u ∗dst, int dstr, const vec08u ∗src, int sstr, int bw, int bh)

  Calculate sum of absolute values of first order derivatives x and y using sobel 3x3.

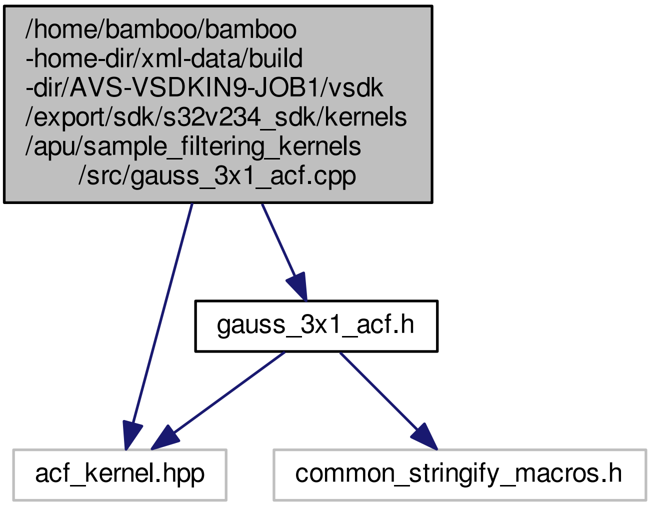
7.91.1 Detailed Description

3x3 Sobel filter implementation for APEX

7.92 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/gauss_3x1_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "gauss_3x1_acf.h"
```

Include dependency graph for gauss_3x1_acf.cpp:



Functions

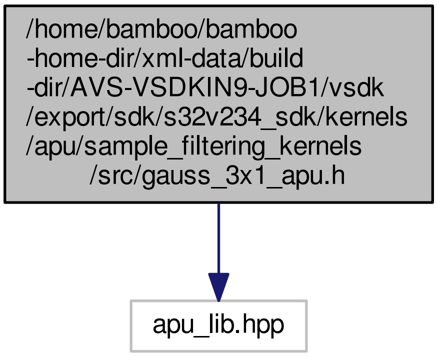
- KERNEL_INFO apu_gauss_3x1 (" apu_gauss_3x1 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.93 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/gauss_3x1_apu.h File Reference

element-wise addition implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for gauss_3x1_apu.h:



Functions

- void gauss_3x1 (vec08u ∗dst, vec08u ∗srcImage0, int bw, int bh, int inStrideW, int outStrideW)

  Elementwise unsigned 8bit addition => unsigned 16bit.

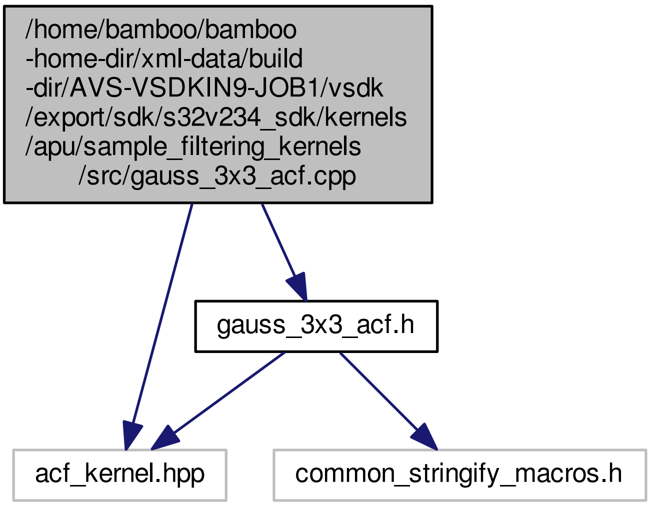
7.93.1 Detailed Description

element-wise addition implementation for APEX

7.94 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/gauss_3x3_acf.cpp File Reference

```
#include <acf_kernel.hpp>
#include "gauss_3x3_acf.h"
```

Include dependency graph for gauss_3x3_acf.cpp:



Functions

- KERNEL_INFO apu_gauss_3x3 (" apu_gauss_3x3 ", 2, __port(__index(0), __identifier("INPUT_0"), __↩ attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_↩ size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_↩ dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.95 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/gauss_3x3_apu.cpp File Reference

3x3 Gauss filter

7.95.1 Detailed Description

3x3 Gauss filter

Author

Igor Aleksandrowicz

Version

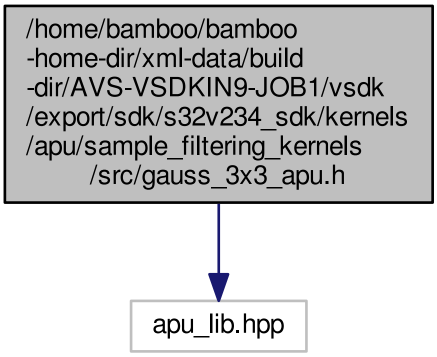
Date

7.96 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/gauss_3x3_apu.h File Reference

3x3 Gaussian filter implementation for APEX

```
#include <apu_lib.hpp>
```

Include dependency graph for gauss_3x3_apu.h:



Functions

- void apu_gauss_3x3 (vec08u ∗apOut, const vec08u ∗apcIn, int aOutStride, int aInStride, int aTileWidth, int aTileHeight)

    3x3 gaussian filter.

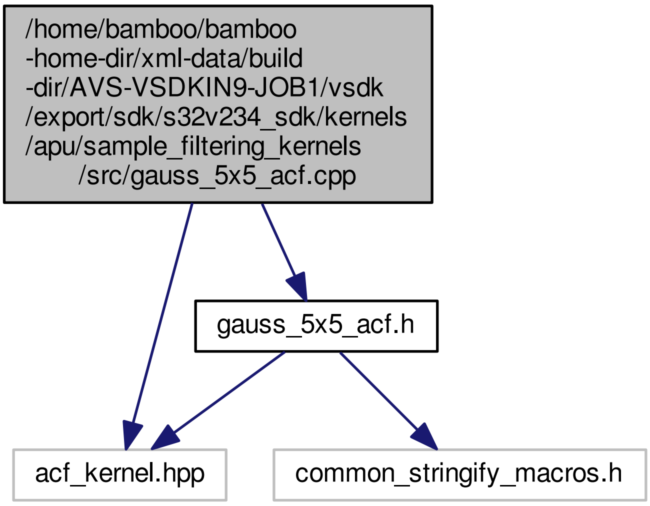
7.96.1 Detailed Description

3x3 Gaussian filter implementation for APEX

7.97 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/gauss_5x5_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "gauss_5x5_acf.h"
```

Include dependency graph for gauss_5x5_acf.cpp:



Functions

- KERNEL_INFO apu_gauss_5x5 (" apu_gauss_5x5 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.98 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/gauss_5x5_apu.cpp File Reference

5x5 Gauss filter implementation for APEX

7.98.1 Detailed Description

5x5 Gauss filter implementation for APEX

Author

Igor Aleksandrowicz

Version

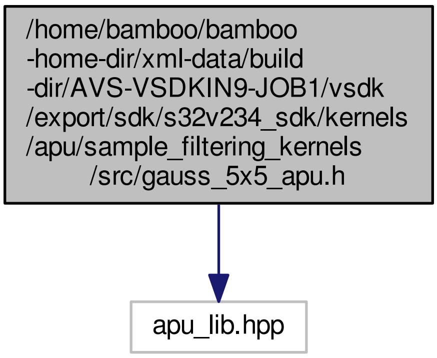
Date

7.99 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/gauss_5x5_apu.h File Reference

5x5 Gaussian filter implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for gauss_5x5_apu.h:



Functions

- void Gauss_5x5__filter (vec08u ∗dst, vec08u ∗src, int sstr, int bw, int bh)

  5x5 gaussian filter.

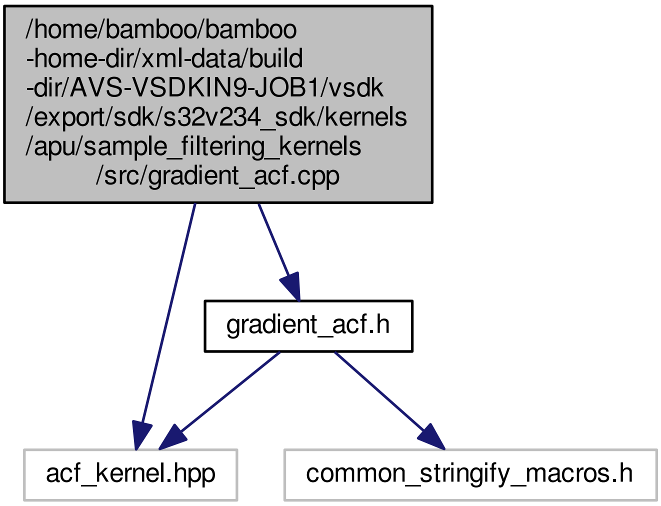
7.99.1 Detailed Description

5x5 Gaussian filter implementation for APEX

7.100 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23_sdk/kernels/apu/sample_filtering_kernels/src/gradient_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "gradient_acf.h"
```

Include dependency graph for gradient_acf.cpp:



Functions

- KERNEL_INFO apu_gradient (" apu_gradient ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_GX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_GY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO **apu_gradient_out08s** (" apu_gradient_out08s ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_GX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_GY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_gradient_abs (" apu_gradient_abs ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_GX"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUT_GY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("OUT_ABSSUM"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_gr_abs (" apu_gr_abs ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUT_ABSSUM"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.101 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_filtering_kernels/src/gradient_apu.cpp File Reference

gradient implementation for APEX

7.101.1 Detailed Description

gradient implementation for APEX

Author

Igor Aleksandrowicz

Version

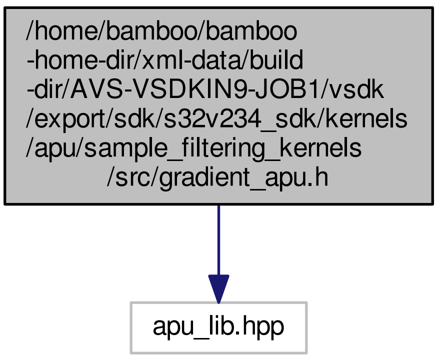
Date

7.102 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_filtering_kernels/src/gradient_apu.h File Reference

image gradient implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for gradient_apu.h:



Functions

- void apuGradient (vec16s ∗apcSobelX, vec16s ∗apcSobelY, const vec08u ∗apInput, int aBlockWidth, int a↩ BlockHeight, int aStride)

  Image gradient. ==> GradX, GradY.

- void apuGradient_out08s (vec08s ∗apcSobelX, vec08s ∗apcSobelY, const vec08u ∗apInput, int aBlockWidth, int aBlockHeight, int aStride)

  Image gradient. ==> GradX, GradY.
- void apuGradientAbs (vec08s ∗apcSobelX, vec08s ∗apcSobelY, vec08u ∗apcAbsSum, const vec08u ∗ap↩ Input, int aBlockWidth, int aBlockHeight, int aStride)

  Image gradient ==> GradX, GradY, |GradX|+|GradY|.
- void apuGradAbs (vec08u ∗apcAbsSum, const vec08u ∗apInput, int aBlockWidth, int aBlockHeight, int a↩ Stride)

  Image gradient absulute sum ==> |GradX|+|GradY|.

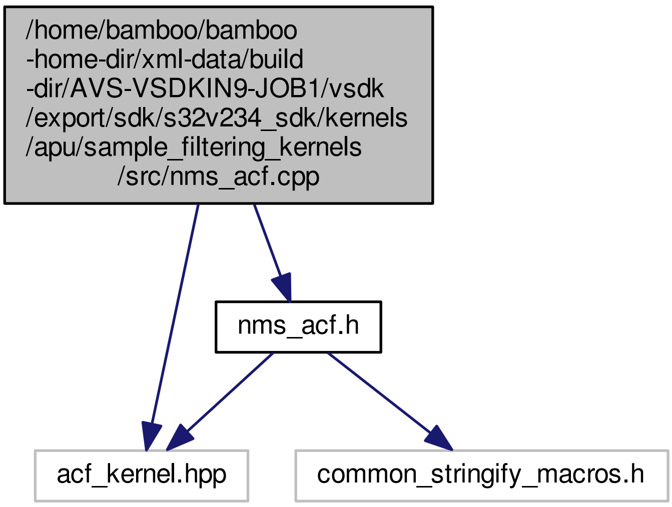
7.102.1 Detailed Description

image gradient implementation for APEX

7.103 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_filtering_kernels/src/nms_acf.cpp File Reference

```
#include <acf_kernel.hpp>
#include "nms_acf.h"
```

Include dependency graph for nms_acf.cpp:



Functions

- KERNEL_INFO apu_nms (" apu_nms ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_↩ ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __↩ port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_nms16 (" apu_nms16 ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(A↩ CF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)),

__port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

7.104 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_filtering_kernels/src/nms_apu.cpp File Reference

Non-maximum supression filter implementation for APEX.

7.104.1 Detailed Description

Non-maximum supression filter implementation for APEX.

Author

Igor Aleksandrowicz

Version

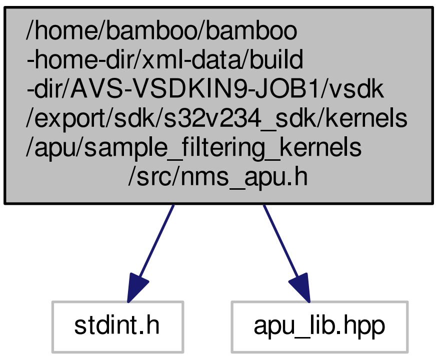
Date

7.105 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_filtering_kernels/src/nms_apu.h File Reference

non-maximum suppression implementation for APEX

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

Include dependency graph for nms_apu.h:



Functions

- void apu_nms_impl (const vec08u ∗apcIn, vec08u ∗apOut, int aInStride, int aOutStride, int aTileWidth, int aTileHeight)

    Non-maximum suppression.
- void apu_nms16 (const vec16u ∗apcIn, vec16u ∗apOut, int aInStride, int aOutStride, int aTileWidth, int a↩
TileHeight)

    Non-maximum suppression, 16-bit.

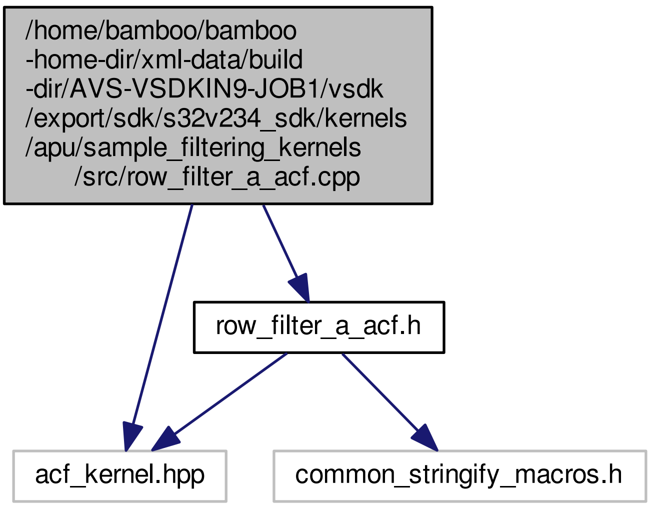
7.105.1 Detailed Description

non-maximum suppression implementation for APEX

7.106 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_filtering_kernels/src/row_filter_a_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "row_filter_a_acf.h"
```

Include dependency graph for row_filter_a_acf.cpp:



Functions

- KERNEL_INFO row_filter (" row_filter ", 3, __port(__index(0), __identifier("SRC"), __attributes(ACF_ATT↩
R_VEC_IN), __spatial_dep(0, 0, ROW_PADDING, ROW_PADDING), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("COEFFS"), __attributes(ACF_ATTR_SCL_IN_STAT↩
IC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(FILTER_ROWS, 1)), __port(__index(2), __identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Variables

- const int **FILTER_ROWS** = 5
- const int **ROW_PADDING** = FILTER_ROWS >> 1

7.107 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_filtering_kernels/src/row_filter_a_apu.cpp File Reference

Filtering with general filter image rows - implementation for APEX.

7.107.1 Detailed Description

Filtering with general filter image rows - implementation for APEX.

Author

CGV

Version

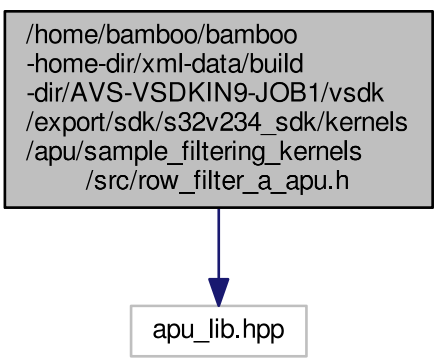
Date

7.108 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_filtering_kernels/src/row_filter_a_apu.h File Reference

Row filter implementation for APU2.

```
#include "apu_lib.hpp"
```

Include dependency graph for row_filter_a_apu.h:



Functions

- void row_filter_impl (vec08u ∗dst, int dstStride, const vec08u ∗src, int srcStride, int rows, int cols, const unsigned char ∗filter, int filterSize, int filterQ)

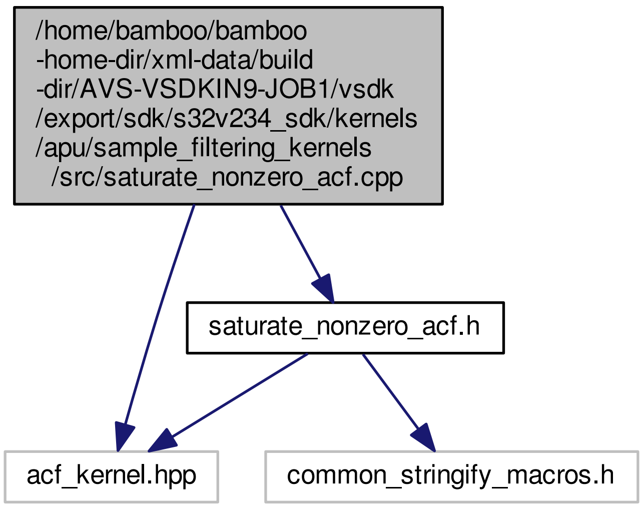
7.108.1 Detailed Description

Row filter implementation for APU2.

7.109 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_filtering_kernels/src/saturate_nonzero_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "saturate_nonzero_acf.h"
```

Include dependency graph for saturate_nonzero_acf.cpp:



Functions

- KERNEL_INFO apu_saturate_nonzero (" apu_saturate_nonzero ", 2, __port(__index(0), __identifier("INP↩UT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), ↩ __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_binarize (" apu_binarize ", 2, __port(__index(0), __identifier("BIN_IN"), __attributes(A↩CF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("BIN_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_mask (" apu_mask ", 5, __port(__index(0), __identifier("MASK_INFL"), __attributes(A↩CF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("MASK_INX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0),

__e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("MASK_INY"), ↩
__attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), _↩
_ek_size(1, 1)), __port(__index(3), __identifier("MASK_OUTX"), __attributes(ACF_ATTR_VEC_OUT), _↩
_spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), _↩
_identifier("MASK_OUTY"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_↩
type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

7.110 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23_sdk/kernels/apu/sample_filtering_kernels/src/saturate_nonzero_apu.cpp File Reference

saturate_nonzero implementation for APEX

7.110.1 Detailed Description

saturate_nonzero implementation for APEX

Author

Igor Aleksandrowicz

Version

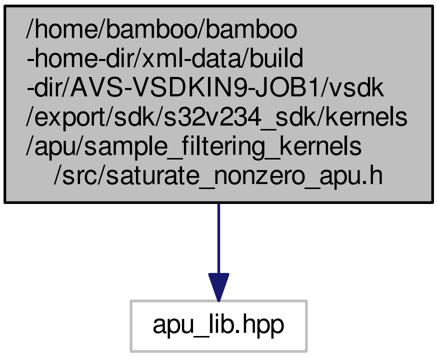
Date

7.111 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23_sdk/kernels/apu/sample_filtering_kernels/src/saturate_nonzero_apu.h File Reference

non-zero saturation implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for saturate_nonzero_apu.h:



Functions

- void saturate_nonzero (vec08u ∗dst, vec08u ∗src, int dstr, int sstr, int bw, int bh)

  Non-zero pixel saturation.

- void binarize (vec08u ∗dst, vec32u ∗src, int, int, int bw, int bh)

  Non-zero pixel binarization.

- void mask (vec16s ∗dstX, vec16s ∗dstY, vec32u ∗srcFlags, vec16s ∗inX, vec16s ∗inY, int, int, int bw, int bh)

  Masking to zero with the srcFlags.

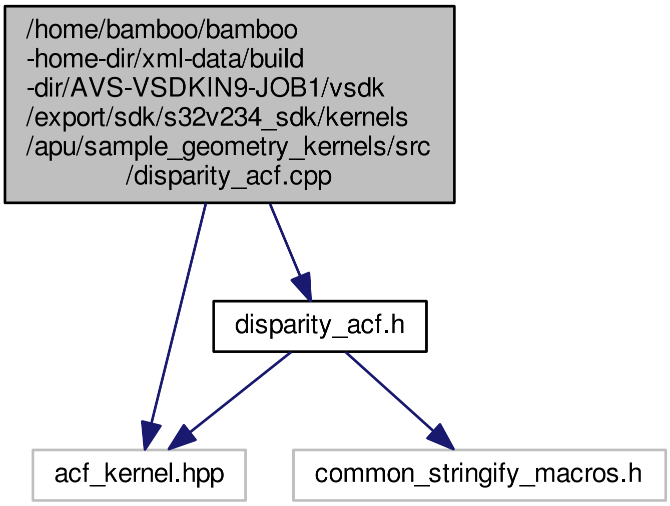
7.111.1 Detailed Description

non-zero saturation implementation for APEX

7.112 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_geometry_kernels/src/disparity_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "disparity_acf.h"
```

Include dependency graph for disparity_acf.cpp:



Functions

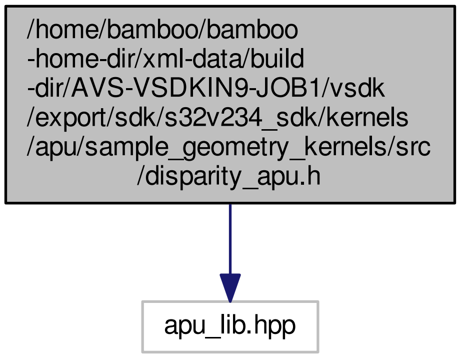
- KERNEL_INFO apu_disparity (" apu_disparity ", 3, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 64, 1, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.113 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_geometry_kernels/src/disparity_apu.h File Reference

element-wise addition implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for disparity_apu.h:



Functions

- void disparity (vec08u ∗dst, vec08u ∗srcImage0, vec08u ∗srcImage1, int bw, int bh, int cw, int ch, int in↩ StrideW0, int inStrideW1, int outStrideW)

  Elementwise unsigned 8bit addition => unsigned 16bit.

7.113.1 Detailed Description

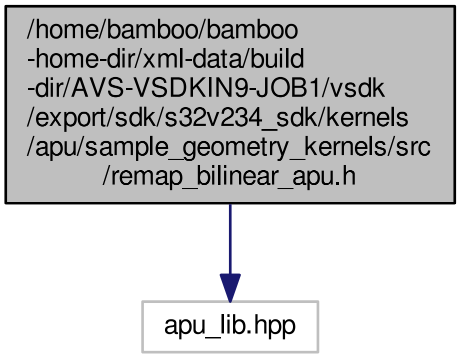
element-wise addition implementation for APEX

7.114 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234↩ _sdk/kernels/apu/sample_geometry_kernels/src/remap_bilinear_apu.h File Reference

element-wise interpolation between pixels of an image for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for remap_bilinear_apu.h:



Functions

- void remap_bilinear_rgb_impl (vec32u ∗dst, vec32u ∗src, vec16u ∗offset, vec08u ∗delta, int sstride, int dstride, int bw, int bh)

  Elementwise bilinear interpolation.

- void remap_bilinear_grayscale_impl (vec08u ∗dst, vec08u ∗src, vec16u ∗offset, vec08u ∗delta, int sstride, int dstride, int bw, int bh)

  Elementwise bilinear interpolation.

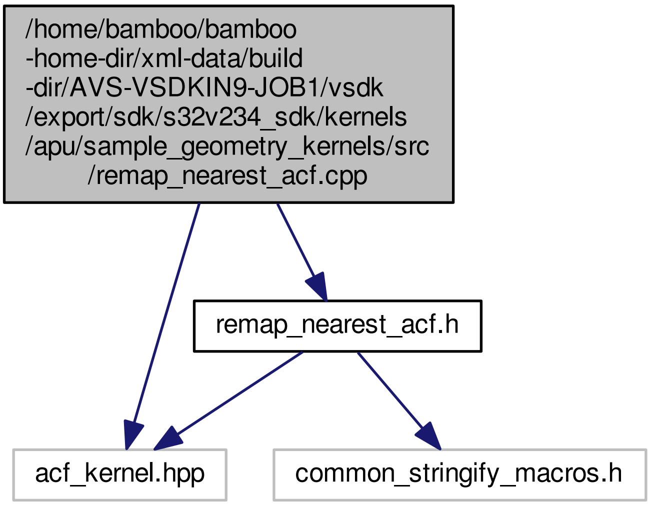
7.114.1 Detailed Description

element-wise interpolation between pixels of an image for APEX

7.115 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_geometry_kernels/src/remap_nearest_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "remap_nearest_acf.h"
```

Include dependency graph for remap_nearest_acf.cpp:



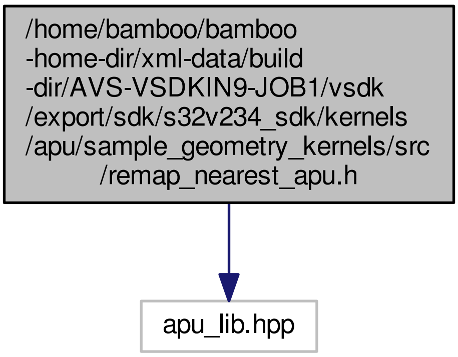
Functions

- KERNEL_INFO remap_nearest_grayscale (" remap_nearest_grayscale ", 3, __port(__index(0), __↩
identifier("DST"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("SRC"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __↩
identifier("OFFSET"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))

7.116 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_geometry_kernels/src/remap_nearest_apu.h File Reference

element-wise interpolation between pixels of an image for APEX

`#include "apu_lib.hpp"`
Include dependency graph for remap_nearest_apu.h:



Functions

- void remap_nearest_grayscale_impl (vec08u ∗dst, vec08u ∗src, vec16u ∗offset, int sstride, int dstride, int bw, int bh)

  Elementwise nearest neighbor interpolation.

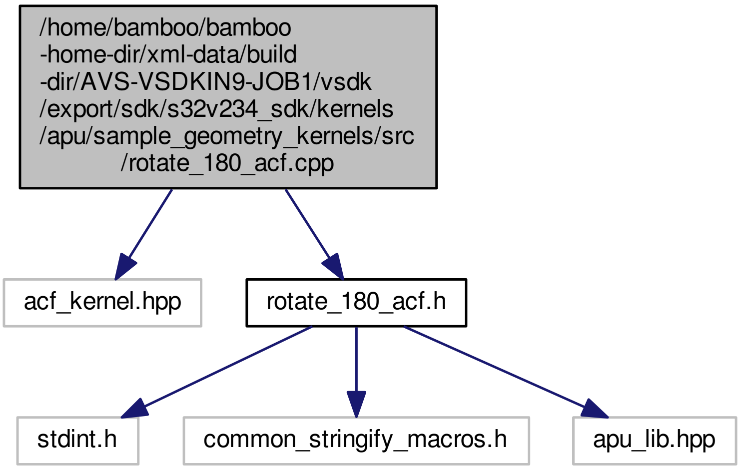
7.116.1 Detailed Description

element-wise interpolation between pixels of an image for APEX

7.117 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_geometry_kernels/src/rotate_180_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "rotate_180_acf.h"
```

Include dependency graph for rotate_180_acf.cpp:



Functions

- KERNEL_INFO apu_rotate_180 (" apu_rotate_180 ", 2, __port(__index(0), __identifier("INPUT"), __↩
attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek↩
_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_↩
dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.118 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234↩ _sdk/kernels/apu/sample_geometry_kernels/src/rotate_180_apu.cpp File Reference

Rotate an image by 180 deg implementation for APEX.

7.118.1 Detailed Description

Rotate an image by 180 deg implementation for APEX.

Author

Igor Aleksandrowicz

Version

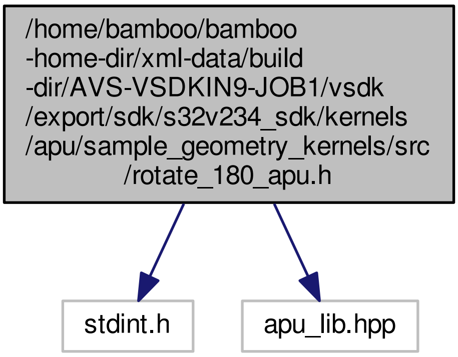
Date

7.119 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_geometry_kernels/src/rotate_180_apu.h File Reference

180-degree rotation implementation for APEX

```
#include <stdint.h>
#include "apu_lib.hpp"
```

Include dependency graph for rotate_180_apu.h:



Functions

- void rotate_180 (vec08u ∗dst, vec08u ∗src, int bw, int bh, int sstr)

  180-degree rotation.

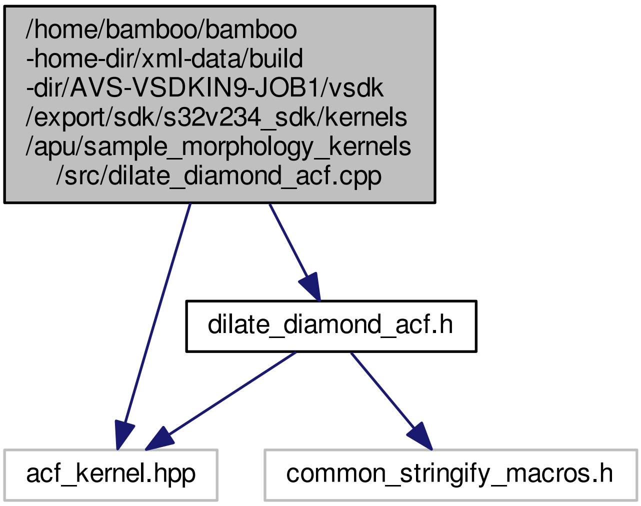
7.119.1 Detailed Description

180-degree rotation implementation for APEX

7.120 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_morphology_kernels/src/dilate_diamond_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "dilate_diamond_acf.h"
```

Include dependency graph for dilate_diamond_acf.cpp:



Functions

- KERNEL_INFO apu_dilate_diamond (" apu_dilate_diamond ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(2, 2, 2, 2), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.121 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_morphology_kernels/src/dilate_diamond_apu.cpp File Reference

dilate_diamond implementation for APEX

7.121.1 Detailed Description

dilate_diamond implementation for APEX

Author

Igor Aleksandrowicz

Version

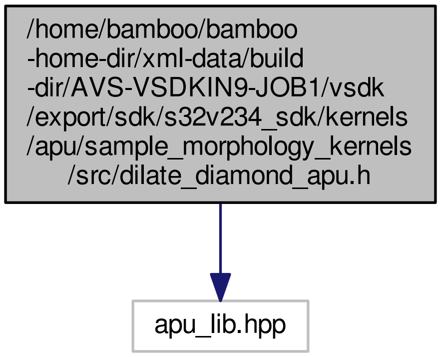
Date

7.122 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_morphology_kernels/src/dilate_diamond_apu.h File Reference

Image diamond dilation implementation for APEX.

```
#include "apu_lib.hpp"
```

Include dependency graph for dilate_diamond_apu.h:



Functions

- void dilate_diamond (vec08u ∗dst, vec08u ∗src, int dstr, int sstr, int bw, int bh)

  5x5 diamond dilation.

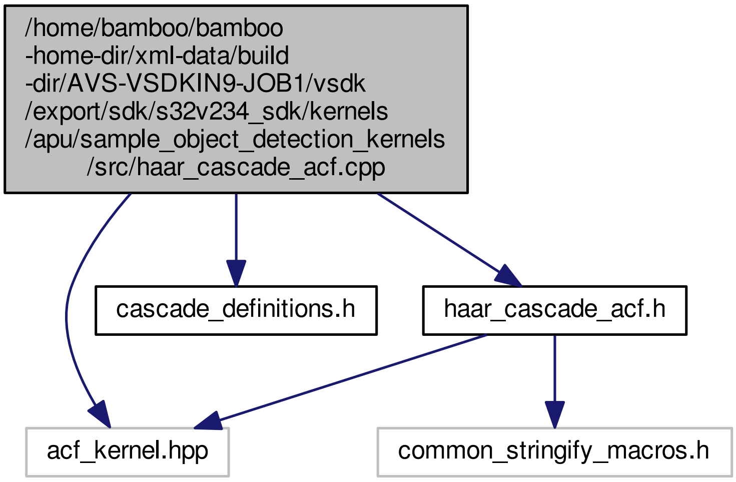
7.122.1 Detailed Description

Image diamond dilation implementation for APEX.

7.123 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_object_detection_kernels/src/haar_cascade_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "cascade_definitions.h"
#include "haar_cascade_acf.h"
```

Include dependency graph for haar_cascade_acf.cpp:



Functions

- KERNEL_INFO apu_haar_cascade (" apu_haar_cascade ", 11, __port(__index(0), __identifier("INPUT_I←NTEGRAL_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("INPUT_INTEGRAL_IMAGE_SQUARE←D"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("LINE_INDEX"), __attributes(ACF_ATTR_SCL_OUT_ST←ATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(←__index(3), __identifier("WINDOW_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __←spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(6+1, 32)), __port(__index(4), ←__identifier("WINDOW_BUFFER_SQUARED"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __←spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(6+1, 32)), __port(__index(5), __identifier("INPUT_CASCADE_SIZES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_←dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 1)), __port(__index(6), __identifier("I←NPUT_CASCADE_FEATURES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(19800, 1)), __port(__index(7), __identifier("IN←PUT_CASCADE_STAGES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(70, 1)), __port(__index(8), __identifier("INPUT_PIXE←L_SHIFTS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_←type(d08u), __e0_size(1, 1), __ek_size(32, 1)), __port(__index(9), __identifier("INPUT_PIXEL_OFFSETS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __←e0_size(1, 1), __ek_size(32, 1)), __port(__index(10), __identifier("OUTPUT"), __attributes(ACF_ATTR_V←EC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.124 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_object_detection_kernels/src/haar_cascade_apu.cpp File Reference

Haar cascade implementation for APEX.

7.124.1 Detailed Description

Haar cascade implementation for APEX.

Author

Igor Aleksandrowicz

Version

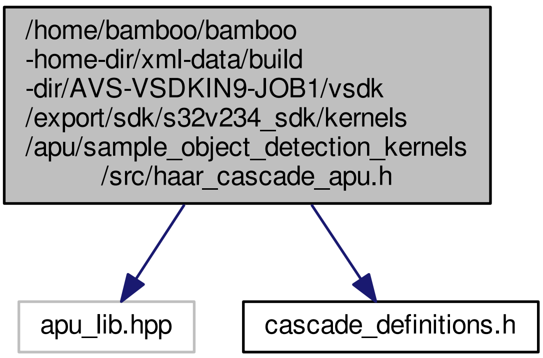
Date

7.125 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23_sdk/kernels/apu/sample_object_detection_kernels/src/haar_cascade_apu.h File Reference

Object detection based on Haar-like features implementation for APEX.

```
#include <apu_lib.hpp>
#include "cascade_definitions.h"
```

Include dependency graph for haar_cascade_apu.h:



Classes

- struct APEX_HaarCascadeFeature
- struct APEX_HaarCascadeStage

Typedefs

- typedef vec16u **FEATURE_FIXED_POINT_TYPE**
- typedef vec16u **STAGE_FIXED_POINT_TYPE**

Functions

- void haar_cascade (vec08u ∗apOut, vec32u ∗apInII, vec32u ∗apInII2, int aOutStride, int aInStride, int a↩ TileWidth, int aTileHeight, int16u aLineIndex, vec32u ∗apWindowBuffer, vec32u ∗apWindowBuffer2, const APEX_HaarCascadeFeature ∗apcFeatures, int aStageCount, const APEX_HaarCascadeStage ∗apcStages, const int08u ∗apcXshifts, const int08u ∗apcXoffsets)

  Haar object detection.

Variables

- const int **FEATURE_FRACTIONAL_BITS** = 13
- const int **FEATURE_FIXED_POINT_MULTIPLIER** = (1 << FEATURE_FRACTIONAL_BITS)
- const int **STAGE_FRACTIONAL_BITS** = 10
- const int **STAGE_FIXED_POINT_MULTIPLIER** = (1 << STAGE_FRACTIONAL_BITS)
- const int32u **featureFixedCoefficientSqrt** = 91
- const int32s **invWindowAreaScalar** = 25

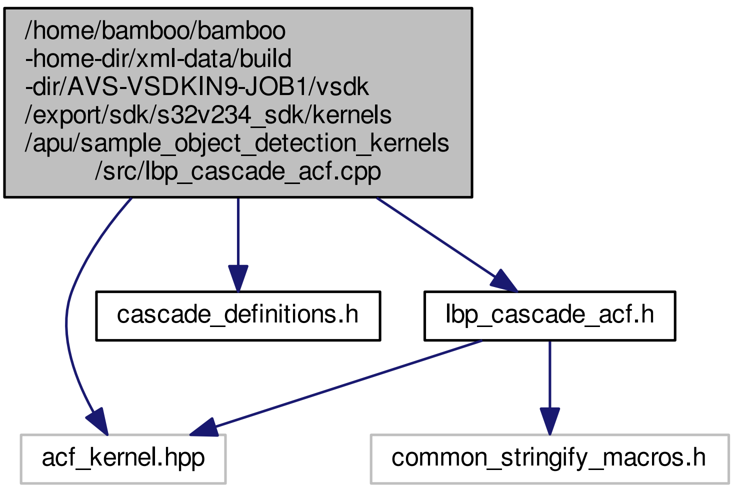
7.125.1 Detailed Description

Object detection based on Haar-like features implementation for APEX.

7.126 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_object_detection_kernels/src/lbp_cascade_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "cascade_definitions.h"
#include "lbp_cascade_acf.h"
```

Include dependency graph for lbp_cascade_acf.cpp:



Functions

- KERNEL_INFO apu_lbp_cascade (" apu_lbp_cascade ", 9, __port(__index(0), __identifier("INPUT_INTEGRAL_IMAGE"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 0, 1, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("LINE_INDEX"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("WINDOW_BUFFER"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(20+1, 32)), __port(__index(3), __identifier("INPUT_CASCADE_SIZES_AND_SKIP"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(3, 1)), __port(__index(4), __identifier("INPUT_CASCADE_FEATURES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(10000, 1)), __port(__index(5), __identifier("INPUT_CASCADE_STAGES"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(200, 1)), __port(__index(6), __identifier("INPUT_PIXEL_SHIFTS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(7), __identifier("INPUT_PIXEL_OFFSETS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(64, 1)), __port(__index(8), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.127 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23_sdk/kernels/apu/sample_object_detection_kernels/src/lbp_cascade_apu.cpp File Reference

LBP cascade implementation for APEX.

7.127.1 Detailed Description

LBP cascade implementation for APEX.

Author

Igor Aleksandrowicz

Version

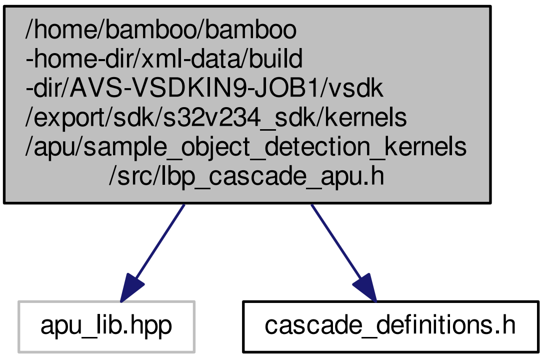
Date

7.128 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23_sdk/kernels/apu/sample_object_detection_kernels/src/lbp_cascade_apu.h File Reference

Object detection based on LBP features implementation for APEX.

```
#include <apu_lib.hpp>
#include "cascade_definitions.h"
```

Include dependency graph for lbp_cascade_apu.h:



Classes

- struct APEX_lbpFeature
- struct APEX_lbpStage

Typedefs

- typedef vec32s **STAGE_FIXED_POINT_TYPE**
- typedef int32s **STAGE_FIXED_POINT_TYPE_SCALAR**

Functions

- void lbp_cascade (vec08u ∗apOut, vec32u ∗apInII, int aOutStride, int aInStride, int aTileWidth, int aTileHeight, int16u aLineIndex, vec32u ∗apWindowBuffer, const APEX_lbpFeature ∗apcFeatures, int aStageCount, const APEX_lbpStage ∗apcStages, const int08u ∗apcXshifts, const int08u ∗apcXoffsets, int skipOdd)

  Local Binary Pattern object detection.

Variables

- const int **STAGE_FRACTIONAL_BITS** = 28
- const int **STAGE_FIXED_POINT_MULTIPLIER** = (1 << STAGE_FRACTIONAL_BITS)

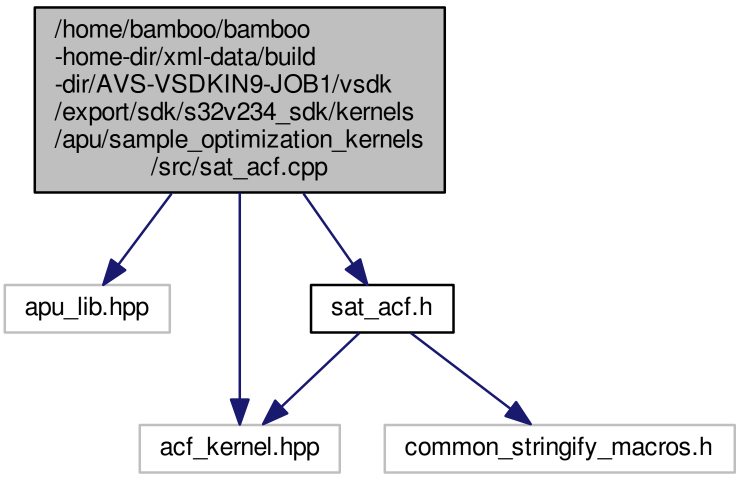
7.128.1 Detailed Description

Object detection based on LBP features implementation for APEX.

7.129 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_optimization_kernels/src/sat_acf.cpp File Reference

```
#include "apu_lib.hpp"
#include "acf_kernel.hpp"
#include "sat_acf.h"
```

Include dependency graph for sat_acf.cpp:



Functions

- KERNEL_INFO apu_sat (" apu_sat ", 3, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("OUTPUT_ROW"), __attributes(ACF_ATTR_VEC_OUT_STATIC), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

7.130 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_optimization_kernels/src/sat_apu.cpp File Reference

sat_box_filter implementation for APEX

7.130.1 Detailed Description

sat_box_filter implementation for APEX

Author

Igor Aleksandrowicz

Version

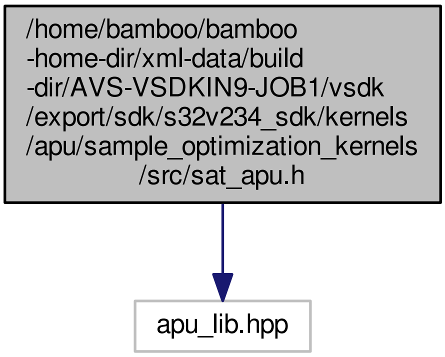
Date

7.131 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_optimization_kernels/src/sat_apu.h File Reference

Summed area table implementation for APEX.

`#include "apu_lib.hpp"`

Include dependency graph for sat_apu.h:



Functions

- void sat32 (vec32u ∗apDest, vec32u ∗apPrevRow, const vec08u ∗apcSrc, int aSourceStride, int a↩DestinationStride, int aBlockWidth, int aBlockHeight, int08u aFirstTile)

  Summed area table.

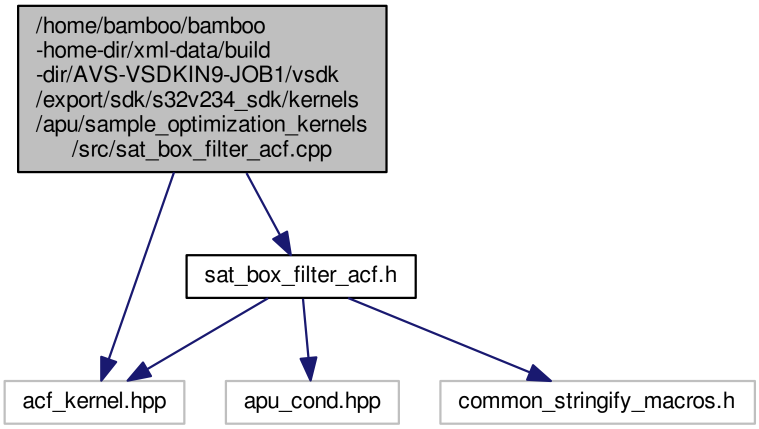
7.131.1 Detailed Description

Summed area table implementation for APEX.

7.132 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_optimization_kernels/src/sat_box_filter_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "sat_box_filter_acf.h"
```

Include dependency graph for sat_box_filter_acf.cpp:



Functions

- KERNEL_INFO apu_sat_box_filter (" apu_sat_box_filter ", 2, __port(__index(0), __identifier("INPUT"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(BOX_SIZE+1, BOX_SIZE, BOX_SIZE+1, BOX_SIZE), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

Variables

- const int **BOX_SIZE**
- const int **BOX_AREA**

7.133 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_optimization_kernels/src/sat_box_filter_apu.cpp File Reference

sat_box_filter implementation for APEX

7.133.1 Detailed Description

sat_box_filter implementation for APEX

Author

Igor Aleksandrowicz

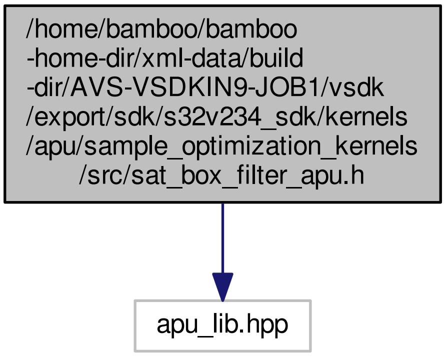
Version

7.134 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_optimization_kernels/src/sat_box_filter_apu.h File Reference

Box filter using SAT implementation for APEX.

`#include "apu_lib.hpp"`

Include dependency graph for sat_box_filter_apu.h:



Functions

- void sat_box_filter_impl (vec08u ∗apDest, const vec32u ∗apcSrc, int aBlockWidth, int aBlockHeight, int a←SourceStride, int aDestStride)

    Sum of values over one patch the input image is a SAT image (i.e. integral computed with the sat32() function.

Variables

- const int **BOX_SIZE**
- const int **BOX_AREA**

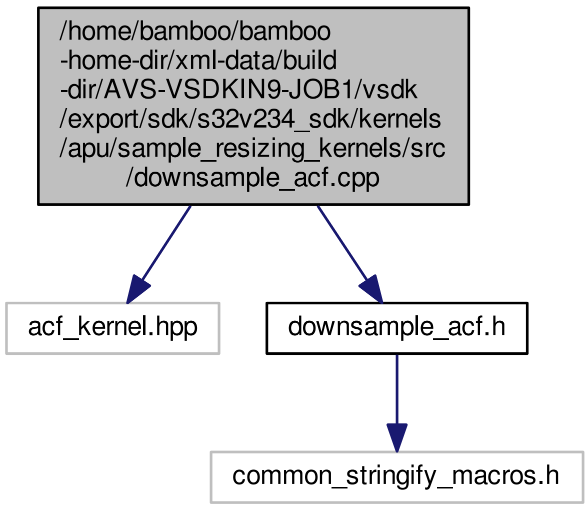
7.134.1 Detailed Description

Box filter using SAT implementation for APEX.

7.135 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_resizing_kernels/src/downsample_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "downsample_acf.h"
```

Include dependency graph for downsample_acf.cpp:



Functions

- KERNEL_INFO apu_downsample (" apu_downsample ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_downsample_16u (" apu_downsample_16u ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size(1, 1), __ek_size(1, 1)))
- KERNEL_INFO apu_downsample_gauss (" apu_downsample_gauss ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)))

7.136 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_resizing_kernels/src/downsample_apu.cpp File Reference

downsample implementation for APEX

7.136.1 Detailed Description

downsample implementation for APEX

Author

Igor Aleksandrowicz

Version

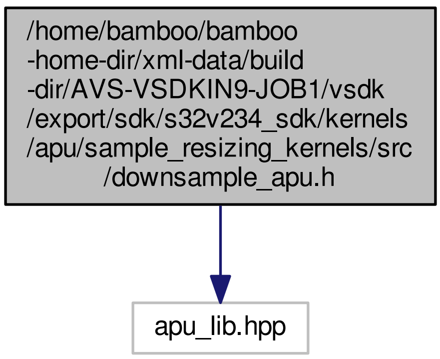
Date

7.137 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_resizing_kernels/src/downsample_apu.h File Reference

Image downsample implementation for APEX.

`#include "apu_lib.hpp"`

Include dependency graph for downsample_apu.h:



Functions

- void downsample (vec08u ∗apDest, const vec08u ∗apcSrc, int aOutBlockWidth, int aOutBlockHeight, int aInBlockStride, int aOutBlockStride)

  x2 downsampling.
- void downsample_16u (vec16u ∗apDest, const vec16u ∗apcSrc, int aInBlockWidth, int aInBlockHeight, int aOutBlockWidth, int aOutBlockHeight, int aInBlockStride, int aOutBlockStride)

  x2 downsampling, 16-bit.
- void downsample_gauss (vec08u ∗apDest, const vec08u ∗apcSrc, int32s aOutBlockWidth, int32s aOutBlockHeight, int32s aInBlockStride, int32s aOutBlockStride)

  x2 downsampling using Gaussian blur.

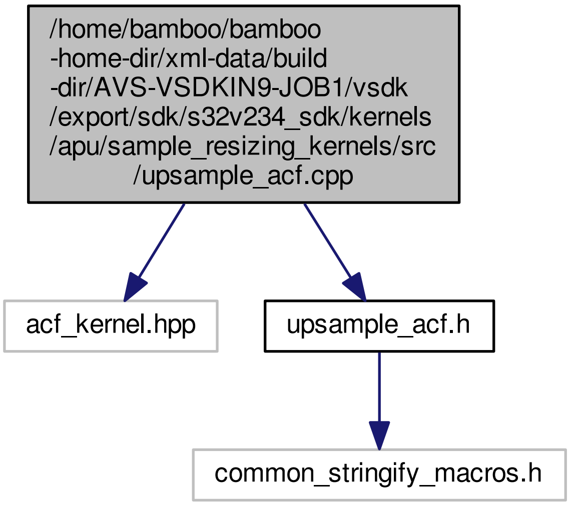
7.137.1 Detailed Description

Image downsample implementation for APEX.

7.138 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_resizing_kernels/src/upsample_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "upsample_acf.h"
```

Include dependency graph for upsample_acf.cpp:



Functions

- KERNEL_INFO apu_upsample (" apu_upsample ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(1, 1, 1, 1), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(2, 2)))

7.139 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_resizing_kernels/src/upsample_apu.cpp File Reference

upsample implementation for APEX

7.139.1 Detailed Description

upsample implementation for APEX

Author

Igor Aleksandrowicz

Version

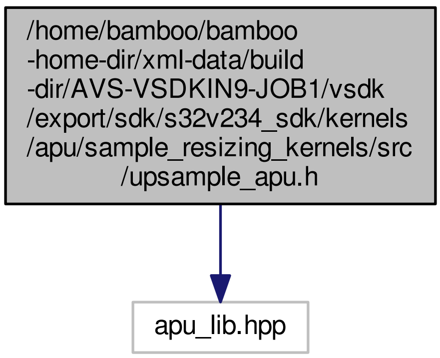
Date

7.140 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_resizing_kernels/src/upsample_apu.h File Reference

Image upsample implementation for APEX.

`#include "apu_lib.hpp"`

Include dependency graph for upsample_apu.h:



Functions

- void upsample (vec08u *apDest, const vec08u *apcSrc, int aInBlockWidth, int aInBlockHeight, int aInBlockStride, int aOutBlockStride)

    x2 upsampling.

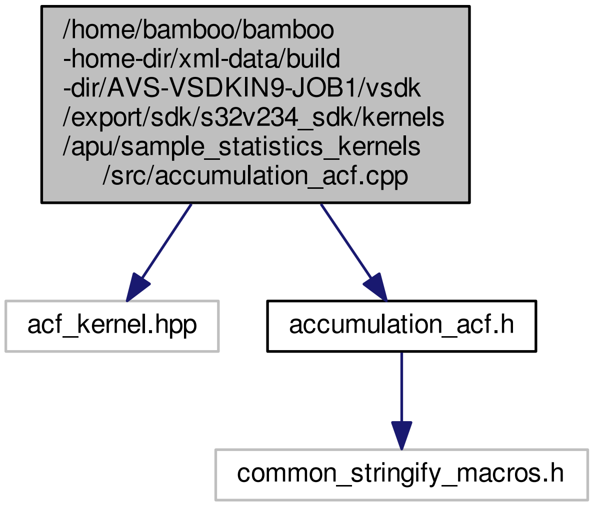
7.140.1 Detailed Description

Image upsample implementation for APEX.

7.141 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/accumulation_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "accumulation_acf.h"
```

Include dependency graph for accumulation_acf.cpp:



Macros

- #define **ACCUM_TILE_SIZE_X** 10
- #define **ACCUM_TILE_SIZE_Y** 10

Functions

- KERNEL_INFO apu_accumulation (" apu_accumulation ", 6, __port(__index(0), __identifier("INPUT_Img"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(10, 10)), __port(__index(1), __identifier("Output_Img"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(2), __identifier("ACCUM_XOFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(3), __identifier("ACCUM_YOFFS"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("ACCUM_XWIDTH"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("ACCUM_YHEIGHT"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16s), __e0_size(1, 1), __ek_size(1, 1)))

7.142 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/accumulation_apu.cpp File Reference

Builds the sum of all elements of a chunk and writes out a vector of sum values.

7.142.1 Detailed Description

Builds the sum of all elements of a chunk and writes out a vector of sum values.

Author

Anca Dima

Version

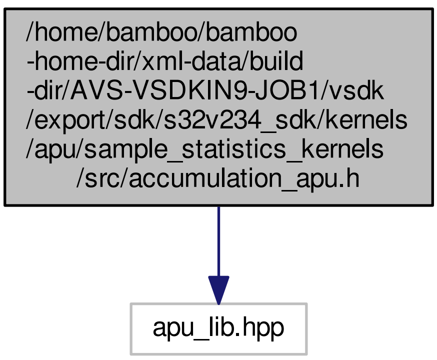
Date

7.143 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/accumulation_apu.h File Reference

accumulation of values from a chunk

```
#include "apu_lib.hpp"
```

Include dependency graph for accumulation_apu.h:



Functions

- void accumulation_in32s_filter (vec32s ∗dst, vec32s ∗srcA, int16s sstr, int16s xOffs, int16s yOffs, int16s xAccWidth, int16s yAccHeight)

Accumulates all values in a chunk (signed 32bit).

- void accumulation_in32u_filter (vec32u ∗lpvOut, vec32u ∗lpvIn, int16s strideWidth, int16s xOffs, int16s yOffs, int16s xAccWidth, int16s yAccHeight)

  Accumulates all values in a chunk (unsigned 32bit).

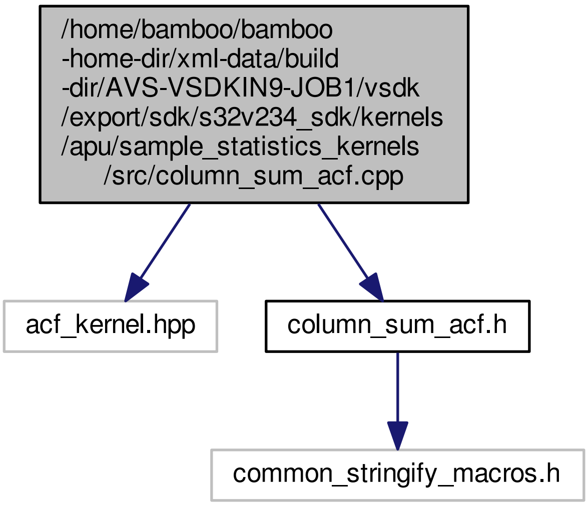
7.143.1 Detailed Description

accumulation of values from a chunk

7.144 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/column_sum_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "column_sum_acf.h"
```

Include dependency graph for column_sum_acf.cpp:



Functions

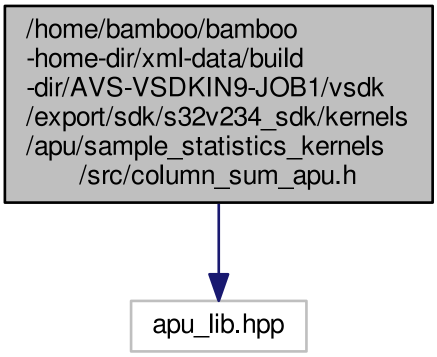
- KERNEL_INFO apu_columns_sum (" apu_columns_sum ", 6, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d08u), __e0_size(1, 1), __ek_size(4, 10)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(2), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_VEC_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(3), __identifier("INPUT_1"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(4), __identifier("INPUT_2"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)), __port(__index(5), __identifier("INPUT_3"), __attributes(ACF_ATTR_SCL_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

7.145 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_statistics_kernels/src/column_sum_apu.h File Reference

cumputing sum of columns of image for APEX ldw_v2 demo

```
#include "apu_lib.hpp"
```

Include dependency graph for column_sum_apu.h:



Functions

- void column_sum (vec08u ∗lpvIn0, vec32u ∗lpvOutDown, vec32u ∗lpvOutUp, bool isFirstTile, int lStrideIn0, int chunkWidth, int chunkHeight, int outChunkWidth, int priorityDown, int priorityUp, int indexOfTileStart)

    Elementwise unsigned 8bit addition => unsigned 16bit.

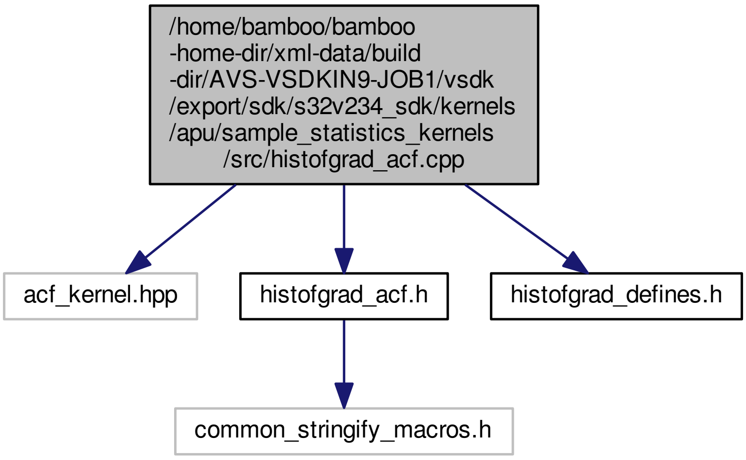
7.145.1 Detailed Description

cumputing sum of columns of image for APEX ldw_v2 demo

7.146 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/sample_statistics_kernels/src/histofgrad_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "histofgrad_acf.h"
#include "histofgrad_defines.h"
```

Include dependency graph for histofgrad_acf.cpp:



Functions

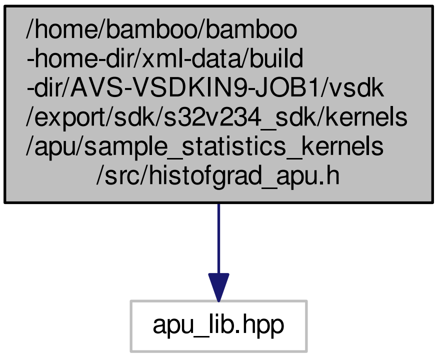
- KERNEL_INFO apu_histofgrad (" apu_histofgrad ", 4, __port(__index(0), __identifier("HOG_InGradX"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 4, 0, 4), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(4, 4)), __port(__index(1), __identifier("HOG_InGradY"), __attributes(ACF_ATTR_VEC_IN), __spatial_dep(0, 4, 0, 4), __e0_data_type(d08s), __e0_size(1, 1), __ek_size(4, 4)), __port(__index(2), __identifier("HOG_OUT"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d16u), __e0_size((16 ∗4 ∗1), 1), __ek_size(1, 1)), __port(__index(3), __identifier("HOG_OUT_BlNorm"), __attributes(ACF_ATTR_VEC_OUT), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(1, 1)))

7.147 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/histofgrad_apu.h File Reference

histogram of gradients computation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for histofgrad_apu.h:



Functions

- void hog (vec08s ∗lpvInGradX, vec08s ∗lpvInGradY, vec16u ∗lpvOut, vec32u ∗lpvOutBlNorm, int lStrideIn, int chunkWidth, int chunkHeight, int lStrideOut)

    Elementwise unsigned 8bit addition => unsigned 16bit.

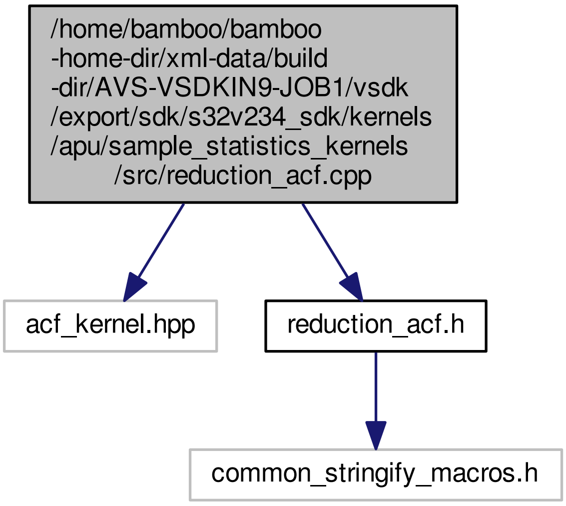
7.147.1 Detailed Description

histogram of gradients computation for APEX

7.148 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/reduction_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "reduction_acf.h"
```

Include dependency graph for reduction_acf.cpp:



Functions

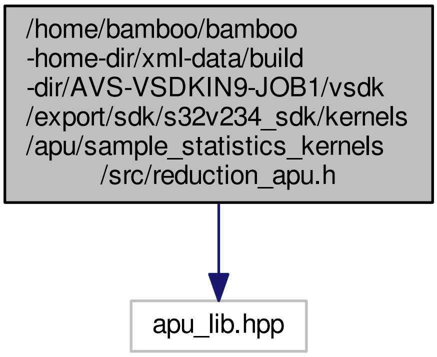
- KERNEL_INFO apu_reduction (" apu_reduction ", 2, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)), __port(__index(1), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(256, 1)))

7.149 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/reduction_apu.h File Reference

reduction of a vector/image implementation for APEX

```
#include "apu_lib.hpp"
```

Include dependency graph for reduction_apu.h:



Functions

- void reduc (vec32u ∗lpvIn0, int32s ∗lpOut0, bool isLastTile, int16s lFirstCuId, int16s lTileWidthInChunks, int lChunkWidth, int lChunkHeight, int lChunkSpanIn0, int lChunkSpanOut0)

    Elementwise unsigned 8bit addition => unsigned 16bit.

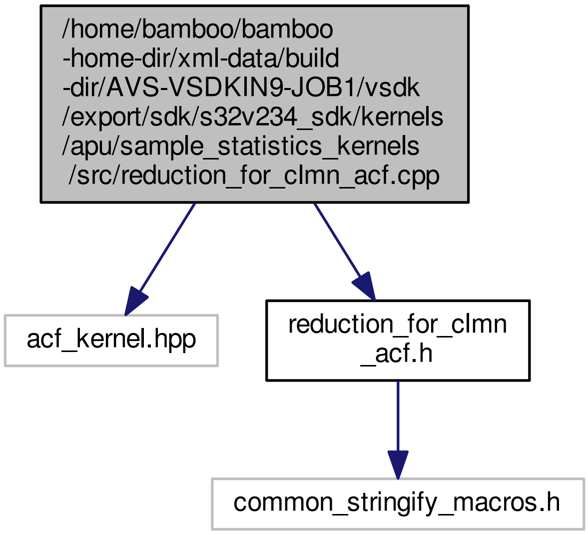
7.149.1 Detailed Description

reduction of a vector/image implementation for APEX

7.150 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/reduction_for_clmn_acf.cpp File Reference

```
#include "acf_kernel.hpp"
#include "reduction_for_clmn_acf.h"
```

Include dependency graph for reduction_for_clmn_acf.cpp:



Functions

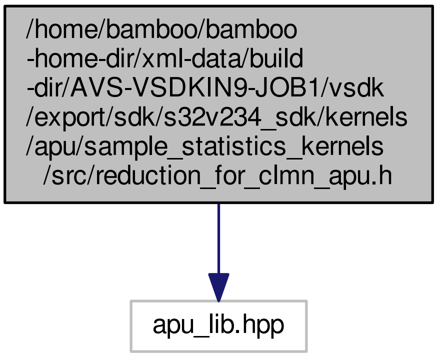
- KERNEL_INFO apu_reduction_for_clmn (" apu_reduction_for_clmn ", 4, __port(__index(0), __identifier("INPUT_0"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(1), __identifier("INPUT_1"), __attributes(ACF_ATTR_VEC_IN_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(4, 1)), __port(__index(2), __identifier("OUTPUT_0"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(192, 1)), __port(__index(3), __identifier("OUTPUT_1"), __attributes(ACF_ATTR_SCL_OUT_STATIC_FIXED), __spatial_dep(0, 0, 0, 0), __e0_data_type(d32u), __e0_size(1, 1), __ek_size(192, 1)))

7.151 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v234_sdk/kernels/apu/sample_statistics_kernels/src/reduction_for_clmn_apu.h File Reference

reduction of a vector/image implementation for APEX ldw_v2 demo

```
#include "apu_lib.hpp"
```

Include dependency graph for reduction_for_clmn_apu.h:



Functions

- void reduc (vec32u ∗lpvIn0, vec32u ∗lpvIn1, int32s ∗lpOut0, int32s ∗lpOut1, bool isLastTile, int16s lFirstCuId, int16s lTileWidthInChunks, int lChunkWidth, int lChunkHeight, int lChunkSpanIn0, int lChunkSpanOut0)

    Elementwise unsigned 8bit addition => unsigned 16bit.

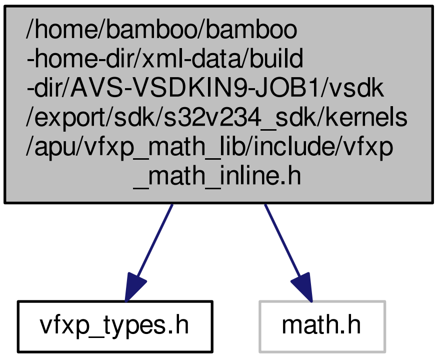
7.151.1 Detailed Description

reduction of a vector/image implementation for APEX ldw_v2 demo

7.152 /home/bamboo/bamboo-home-dir/xml-data/build-dir/AVS-VSDKIN9-JOB1/vsdk/export/sdk/s32v23 _sdk/kernels/apu/vfxp_math_lib/include/vfxp_math_inline.h File Reference

```
#include "vfxp_types.h"
#include "math.h"
```

Include dependency graph for vfxp_math_inline.h:



Macros

- #define **ALWAYS_INLINE** static inline

Index