Xilinx OpenCV User Guide

UG1233 (v2017.1) July 18, 2017

XILINX
ALL PROGRAMMABLE™

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/20/2017	2017.1	Initial Xilinx release
07/18/2017	2017.1	Updated instructions in Using the xfOpenCV Library section.

Table of Contents

Chapter 1

Overview

This document describes the FPGA device optimized OpenCV library, called the Xilinx® xfOpenCV library and is intended for application developers using Zynq®-7000 All Programmable SoC and Zynq UltraScale+™ MPSoC devices. xfOpenCV library has been designed to work in the SDx™ development environment, and provides a software interface for computer vision functions accelerated on an FPGA device. xfOpenCV library functions are mostly similar in functionality to their OpenCV equivalent. Any deviations, if present, are documented.

NOTE: *For more information on the xfOpenCV library prerequisites, see the Prerequisites. To familiarize yourself with the steps required to use the xfOpenCV library functions, see the Using the xfOpenCV Library.*

Basic Features

All xfOpenCV lilbrary functions follow a common format. The following properties hold true for all the functions.

- All the functions are designed as templates and all arguments that are images, must be provided as `xF::Mat`.
- Some of the major template arguments are:
  - Maximum size of the image to be processed
  - Datatype defining the properties of each pixel
  - Number of pixels to be processed per clock cycle
  - Other compile-time arguments relevent to the functionality.

The xfOpenCV library contains enumerated datatypes which enables you to configure `xF::Mat`. For more details on `xF::Mat`, see the xF::Mat Image Container Class.

xfOpenCV Kernel on the reVISION Platform

The xfOpenCV library is designed to be used with the SDx™ development environment. xfOpenCV kernels are evaluated on the reVISION™ platform.

The following steps describe the general flow of an example design, where both the input and the output are image files.

1. Read the image using `cv::imread()`.
2. Copy the data to `xF::Mat`.
3. Call the processing function(s) in xfOpenCV.
4. Copy the data from `xF::Mat` to `cv::Mat`.
5. Write the output to image using `cv::imwrite()`.

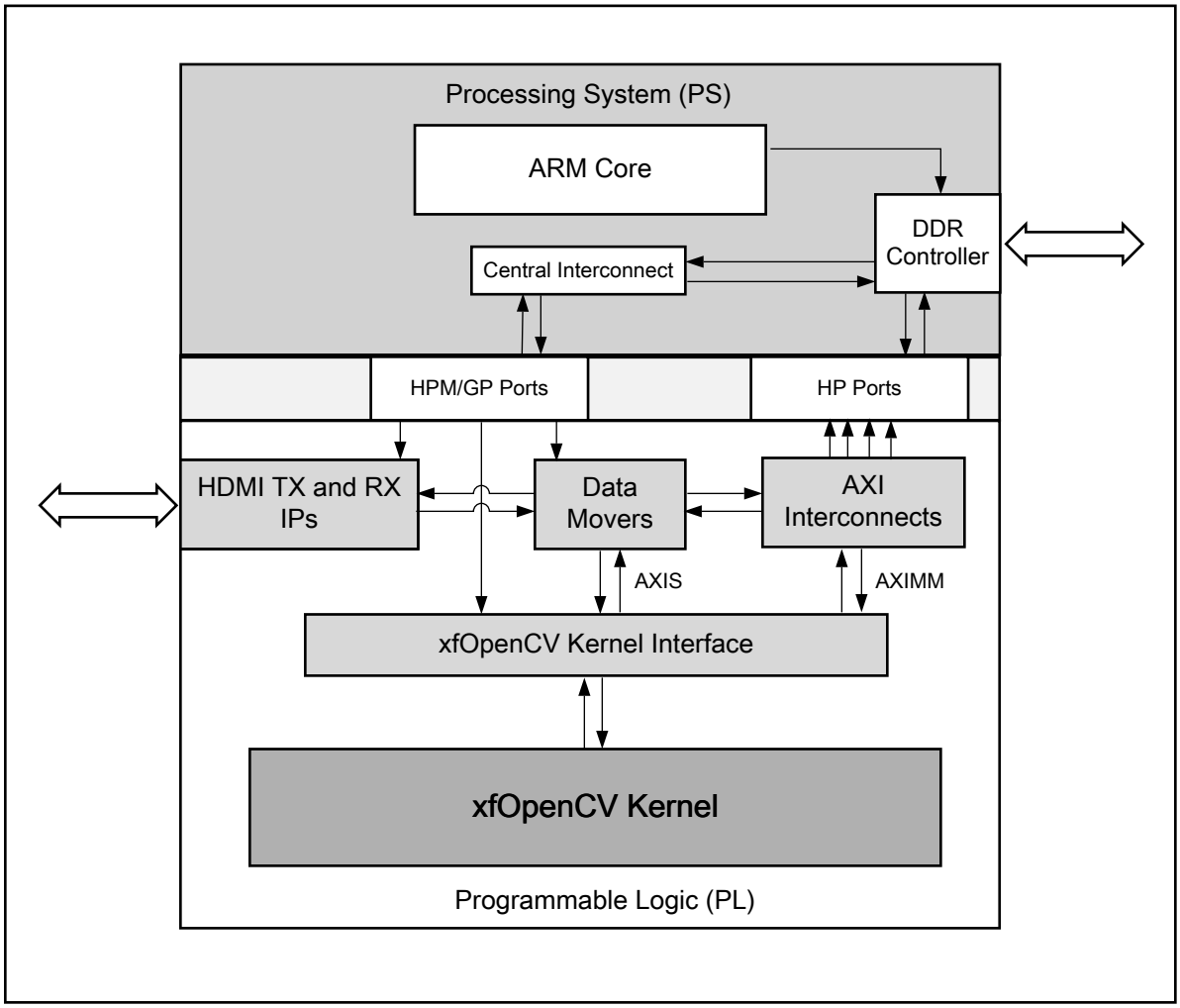
The entire code is written as the host code for the pipeline , from which all the calls to xfOpenCV functions are moved to hardware. Functions from OpenCV are used to read and write images in the memory. The image containers for xfOpenCV library functions are `xF::Mat` objects. For more information, see the xF::Mat Image Container Class.

The reVISION platform supports both live and file input-output (I/O) modes. For more details, see the reVISION Getting Started Guide.

- File I/O mode enables the controller to transfer images from SD Card to the hardware kernel. The following steps describe the file I/O mode.
    1. Processing system (PS) reads the image frame from the SD Card and stores it in the DRAM.
    2. The xfOpenCV kernel reads the image from the DRAM, processes it and stores the output back in the DRAM memory.
    3. The PS reads the output image frame from the DRAM and writes it back to the SD Card.
- Live I/O mode enables streaming frames into the platform, processing frames with the xfOpenCV kernel, and streaming out the frames through the appropriate interface. The following steps describe the live I/O mode.
    1. Video capture IPs receive a frame and store it in the DRAM.
    2. The xfOpenCV kernel fetches the image from the DRAM, processes the image, and stores the output in the DRAM.
    3. Display IPs read the output frame from the DRAM and transmits the frame through the appropriate display interface.

Following figure shows the reVISION platform with the xfOpenCV kernel block:

Figure 1: xfOpenCV Kernel on the reVISION Platform



NOTE: *For more information on the PS-PL interfaces and PL-DDR interfaces, see the Zynq UltraScale+ MPSoC Technical Reference Manual (UG1085).*

Chapter 2

Getting Started

This chapter provides the informaiton you need to bring up your design using the xfOpenCV library functions.

Prerequisites

This section lists the prerequisites for using the xfOpenCV library functions.

- Download and install the SDx development environment according to the directions provided in *SDx Environments Release Notes, Installation, and Licensing Guide* (UG1238). Before launching the SDx development environment on Linux, set the `$SYSROOT` environment variable to point to the Linux root file system, delivered with the reVISION platform. For example:

```
export SYSROOT = <local folder>/zcu102_es2_reVISION/sw/
aarch64-linux-gnu/sysroot
```

- Download the Zynq® UltraScale+™ MPSoC Embedded Vision Platform zip file and extract its contents. Create the SDx development environment workspace in the `zcu102_es2_reVISION` folder of the extracted design file hierarchy. For more details, see the reVISION Getting Started Guide.
- Set up the ZCU102 evaluation board. For more details, see the reVISION Getting Started Guide.
- Download the xfOpenCV library. The library is made available through github. Run the following `git clone` command to clone the xfOpenCV repository to your local disk:

```
git clone https://github.com/Xilinx/xfopencv.git
```

xfOpenCV Library Contents

The contents of the xfOpenCV library are detailed in the following table:

Folder	Details
`include`	Contains the header files required by the library.
`common`	Contains the common library infrastructure headers, such as types specific to the library.
`core`	Contains the core library functionality headers, such as the `math` functions.
`features`	Contains the feature extraction kernel function definitions. For example, `Harris`.
`imgproc`	Contains all the kernel function definitions, except the ones available in the `features` folder.
`examples`	Contains the sample test bench code to facilitate running unit tests. The `examples/` folder contains the folders with algorithm names. Each algorithm folder contains host files , `.json` file, `data` folder and `include` folder.

Using the xfOpenCV Library

This section describes using the xfOpenCV library in the SDx development environment.

NOTE: *The instructions in this section assume that you have downloaded and installed all the required packages. For more information, see the Prerequisites .*

Use the following steps to run a unit test for bilateral filter on zcu102_es2_reVISION:

1. Launch the SDx development environment using the desktop icon or the **Start** menu.

   The **Workspace Launcher** dialog appears.

2. Click **Browse** to enter a workspace folder used to store your projects (you can use workspace folders to organize your work), then click **OK** to dismiss the **Workspace Launcher** dialog.

   NOTE: *Before launching the SDx IDE on Linux, ensure that you use the same shell that you have used to set the `$SYSROOT` environment variable. This is usually the file path to the Linux root file system.*

   The SDx development environment window opens with the **Welcome** tab visible when you create a new workspace. The **Welcome** tab can be closed by clicking the **X** icon or minimized if you do not wish to use it.

3. Select **File** →**New** →**Xilinx SDx Project** from the SDx development environment menu bar.

   The **New Project** dialog box opens.

4. Specify the name of the project. For example **Bilateral**.
5. Click **Next**.

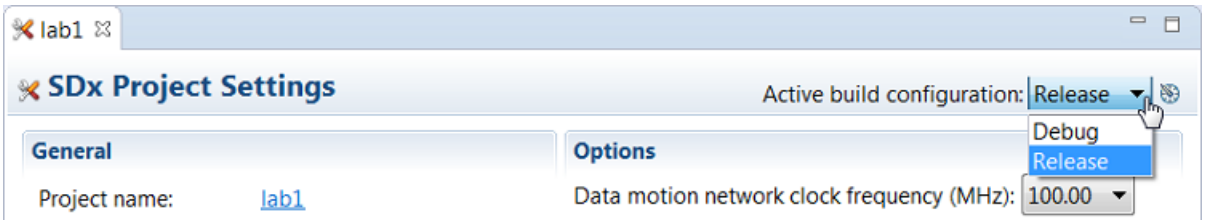
   The the **Choose Hardware Platform** page appears.

6. From the **Choose Hardware Platform** page, click the **Add Custom Platform** button.
7. Browse to the directory where you extracted the reVISION platform files. Ensure that you select the `zcu102_es2_reVISION` folder.
8. From the **Choose Hardware Platform** page, select **zcu102_es2_reVISION (custom)**.
9. Click **Next**.

   The **Templates** page appears, containing source code examples for the selected platform.
10. From the list of application templates, select **bilateral - File I/O** and click **Finish**.
11. Click the **Active build configurations** drop-down from the **SDx Project Settings** window, to select the active configuration or create a build configuration.

    The standard build configurations are Debug and Release. To get the best runtime performance, switch to use the **Release** build configuration as it uses a higher compiler optimization setting than the Debug build configuration.

Figure 2: SDx Project Settings - Active Build Configuration



12. Set the **Data motion network clock frequency (MHz)** to the required frequency, on the **SDx Project Settings** page.
13. Select the **Generate bitstream** and **Generate SD card image** check boxes.
14. Right-click the project and select **Build Project** or press Ctrl+B keys to build the project, in the **Project Explorer** view.
15. Copy the contents of the newly created `sd_card` folder to the SD card.

    The `sd_card` folder contains all the files required to run designs on the `ZCU102` board.
16. Insert the SD card in the `ZCU102` board card slot and switch it ON.

    NOTE: *A serial port emulator (Teraterm/ minicom) is required to interface the user commands to the board.*
17. Upon successful boot, run the following command in the Teraterm terminal (serial port emulator):

```
#cd /media/card
#remount
```

18. Run the `.elf` file for the respective functions.

    For more information, see the Using the xfOpenCV Library Functions on Hardware.

Building a Project Using the Example Makefiles on Linux

Use the following steps to build a project using the example makefiles on the Linux platform:

1. Open a terminal.
2. Set the environment variable `SYSROOT` to the `<path to revision platform on local machine>/sw/aarch64-linux-gnu/sysroot` folder.
3. Copy the bilateralfilter example from the `<path to xfOpenCV git folder on local machine>/xfOpenCV/examples/bilateralfilter` folder to the `<path to revision platform on local machine>/sw/aarch64-linux-gnu/sysroot` folder.
4. Change the directory to `<path to reVISION platform on local machine>/samples/bilateral/example`.

   ```
   cd <path to reVISION platform>/samples/bilateral/example
   ```

5. Set the environment variables to run SDx development environment.
   - For c shell:

     ```
     source <SDx tools install path>/settings.csh
     ```

   - For bash shell:

     ```
     source <SDx tools install path>/settings.sh
     ```

6. Type the `make` command in the terminal.

   The `sd_card` folder is created and can be found in the `<path to reVISION platform on local machine>/samples/bilateral/examples` folder.

Changing the Hardware Kernel Configuration

Use the following steps to change the hardware kernel configuration:

1. Update the `<path to xfOpenCV git folder>/xfOpenCV/examples/<function>/xf_config_params.h` file.
2. Update the makefile along with the `xf_config_params.h` file:
   a. Find the line with the function name in the makefile. For bilateral filter, the line in the makefile will be `xFBilateralFilter<3,1,0,1080,1920,1>`.
   b. Update the template parameters in the makefile to reflect changes made in the `xf_config_params.h` file. For more details, see the xfOpenCV Library API Reference.

Using the xfOpenCV Library Functions on Hardware

The following table lists the xfOpenCV library functions and their usage on hardware:

Table 1: Using the xfOpenCV Library Function on Hardware

Example	Function Name	Usage on Hardware
Accumulate	xFaccumulate	./<executable name>.elf <path to input image 1> <path to input image 2>
accumulatesquared	xFaccumulateSquare	./<executable name>.elf <path to input image 1> <path to input image 2>
accumulateweighted	xFaccumulateWeighted	./<executable name>.elf <path to input image 1> <path to input image 2>
Arithm	xFabsdiff, xFadd, xFsubtract, xFbitwise_and, xFbitwise_or, xFbitwise_not, xFbitwise_xor	./<executable name>.elf <path to input image 1> <path to input image 2>
Bilateralfilter	xFBilateralFilter	./<executable name>.elf <path to input image>
Boxfilter	xFboxfilter	./<executable name>.elf <path to input image>
Canny	xFcanny	./<executable name>.elf <path to input image>
channelcombine	xFmerge	./<executable name>.elf <path to input image 1> <path to input image 2> <path to input image 3> <path to input image 4>
Channelextract	xFextractChannel	./<executable name>.elf <path to input image>
Convertbitdepth	xFconvertTo	./<executable name>.elf <path to input image>
Customconv	xFfilter2D	./<executable name>.elf <path to input image>
cvtcolor IYUV2NV12	xFiyuv2nv12	./<executable name>.elf <path to input image 1> <path to input image 2> <path to input image 3>
cvtcolor IYUV2RGBA	xFiyuv2rgba	./<executable name>.elf <path to input image 1> <path to input image 2> <path to input image 3>
cvtcolor IYUV2YUV4	xFiyuv2yuv4	./<executable name>.elf <path to input image 1> <path to input image 2> <path to input image 3> <path to input image 4> <path to input image 5> <path to input image 6>
cvtcolor NV122IYUV	xFnv122iyuv	./<executable name>.elf <path to

Example	Function Name	Usage on Hardware
		input image 1> <path to input image 2>
cvtcolor NV122RGBA	xFnv122rgba	./<executable name>.elf <path to input image 1> <path to input image 2>
cvtcolor NV122YUV4	xFnv122yuv4	./<executable name>.elf <path to input image 1> <path to input image 2>
cvtcolor NV212IYUV	xFnv212iyuv	./<executable name>.elf <path to input image 1> <path to input image 2>
cvtcolor NV212RGBA	xFnv212rgba	./<executable name>.elf <path to input image 1> <path to input image 2>
cvtcolor NV212YUV4	xFnv212yuv4	./<executable name>.elf <path to input image 1> <path to input image 2>
cvtcolor RGBA2YUV4	xFrgba2yuv4	./<executable name>.elf <path to input image>
cvtcolor RGBA2IYUV	xFrgba2iyuv	./<executable name>.elf <path to input image>
cvtcolor RGBA2NV12	xFrgba2nv12	./<executable name>.elf <path to input image>
cvtcolor RGBA2NV21	xFrgba2nv21	./<executable name>.elf <path to input image>
cvtcolor UYVY2IYUV	xFuyvy2iyuv	./<executable name>.elf <path to input image>
cvtcolor UYVY2NV12	xFuyvy2nv12	./<executable name>.elf <path to input image>
cvtcolor UYVY2RGBA	xFuyvy2rgba	./<executable name>.elf <path to input image>
cvtcolor YUYV2IYUV	xFyuyv2iyuv	./<executable name>.elf <path to input image>
cvtcolor YUYV2NV12	xFyuyv2nv12	./<executable name>.elf <path to input image>
cvtcolor YUYV2RGBA	xFyuyv2rgba	./<executable name>.elf <path to input image>
Dilation	xFdilate	./<executable name>.elf <path to input image>
Erosion	xFerode	./<executable name>.elf <path to input image>
Fast	xFFAST	./<executable name>.elf <path to input image>
Gaussianfilter	xFGaussianBlur	./<executable name>.elf <path to input image>

Example	Function Name	Usage on Hardware
Harris	xFCornerHarris	./<executable name>.elf <path to input image>
Histogram	xFcalcHist	./<executable name>.elf <path to input image>
Histequialize	xFequalizeHist	./<executable name>.elf <path to input image>
Hog	xFHOGDescriptor	./<executable name>.elf <path to input image>
Integralimg	xFIntegralImage	./<executable name>.elf <path to input image>
Lkdensepyrof	xFDensePyrOpticalFlow	./<executable name>.elf <path to input image 1> <path to input image 2>
Lknpyroflow	xFDenseNonPyrLKOpticalFlow	./<executable name>.elf <path to input image 1> <path to input image 2>
Lut	xFLUT	./<executable name>.elf <path to input image>
Magnitude	xFmagnitude	./<executable name>.elf <path to input image>
meanshifttracking	xFMeanShift	./<executable name>.elf <path to input input video/input image files> <Number of objects to track>
meanstddev	xFmeanstd	./<executable name>.elf <path to input image>
medianblur	xFMedianBlur	./<executable name>.elf <path to input image>
Minmaxloc	xFminMaxLoc	./<executable name>.elf <path to input image>
otsuthreshold	xFOtsuThreshold	./<executable name>.elf <path to input image>
Phase	xFphase	./<executable name>.elf <path to input image>
Pyrdown	xFPyrDown	./<executable name>.elf <path to input image>
Pyrup	xFPyrUp	./<executable name>.elf <path to input image>
remap	xFRemap	./<executable name>.elf <path to input image> <path to mapx data> <path to mapy data>
Resize	xFResize	./<executable name>.elf <path to input image>
scharrfilter	xFScharr	./<executable name>.elf <path to input image>
sobelfilter	xFSobel	./<executable name>.elf <path to

Example	Function Name	Usage on Hardware
		input image>
stereopipeline	xFStereoPipeline	./<executable name>.elf <path to left image> <path to right image>
stereolbm	xFStereoBM	./<executable name>.elf <path to left image> <path to right image>
Svm	xFSVM	./<executable name>.elf
threshold	xFThreshold	./<executable name>.elf <path to input image>
warpaffine	xFwarpAffine	./<executable name>.elf <path to input image>
warpperspective	xFperspective	./<executable name>.elf <path to input image>
warptransform	xFWarpTransform	./<executable name>.elf <path to input image>

Chapter 3

xfOpenCV Library API Reference

To facilitate local memory allocation on FPGA devices, the xfOpenCV library functions are provided in templates with compile-time parameters. Data is explicitly copied from `cv::Mat` to `xF::Mat` and is stored in physically contiguous memory to achieve the best possible performance. After processing, the output in `xF::Mat` is copied back to `cv::Mat` to write it into the memory.

xF::Mat Image Container Class

`xF::Mat` is a template class that serves as a container for storing image data and its attributes.

NOTE: *The `xF::Mat` image container class is similar to the `cv::Mat` class of the OpenCV library.*

Class Definition

```
template<int TYPE, int ROWS, int COLS, int NPC>
class Mat {
public:
        Mat();                          // default constructor
        Mat(int _rows, int _cols);
       Mat(int _rows, int _cols, void *_data);
        Mat(int _size, int _rows, int _cols);
        ~Mat();
        void init(int _rows, int _cols);
        void copyTo(XF_PTSNAME(T,NPC)* fromData);
        XF_PTSNAME(T,NPC)* copyFrom();
        int rows, cols, size;               // actual image size
#ifndef __SYNTHESIS__
        XF_TNAME(T,NPC)*data;
#else
        XF_TNAME(T,NPC) data[ROWS*(COLS>> (XF_BITSHIFT(NPC)))];
#endif
};
```

Parameter Descriptions

The following table lists the `xF::Mat` class parameters and their descriptions:

Table 2: xF::Mat Class Parameter Descriptions

Parameter	Description
rows	The number of rows in the image or height of the image.
cols	The number of columns in the image or width of the image.
size	The number of words stored in the data member. The value is calculated using `rows*cols/(number of pixels packed per word).`
*data	The pointer to the words that store the pixels of the image.

Member Functions Description

The following table lists the member functions and their descriptions:

Table 3: xF::Mat Member Function Descriptions

Member Functions	Description
Mat()	This default constructor initializes the Mat object sizes, using the template parameters ROWS and COLS.
Mat(int _rows, int _cols)	This constructor initializes the Mat object using arguments _rows and _cols.
Mat(int _rows, int _cols, void *_data)	This constructor initializes the Mat object using arguments _rows, _cols, and _data. The *data member of the Mat object points to the memory allocated for _data argument, when this constructor is used. No new memory is allocated for the *data member.
copyTo(* fromData)	Copies the data from Data pointer into physically contiguous memory allocated inside the constructor.
copyFrom()	Returns the pointer to the first location of the *data member.
~Mat()	This is a default destructor of the Mat object.

Template Parameter Descriptions

Template parameters of the `xF::Mat` class are used to set the depth of the pixel, number of channels in the image, number of pixels packed per word, maximum number of rows and columns of the image. The following table lists the template parameters and their descriptions:

Table 4: xF::Mat Template Parameter Descriptions

Parameters	Description
TYPE	Type of the pixel data. For example, XF_8UC1 stands for 8-bit unsigned and one channel pixel. More types can be found in `includes/common/xf_params.h`.
HEIGHT	Maximum height of an image.
WIDTH	Maximum width of an image.
NPC	The number of pixels to be packed per word. For instance, XF_NPPC1 for 1 pixel per word; and XF_NPPC8 for 8 pixels per word.

Pixel-Level Parallelism

The amount of parallelism to be implemented in a function from xfOpenCV is kept as a configurable parameter. In most functions, there are two options for processing data.

- Single-pixel processing
- Processing eight pixels in parallel

The following table describes the options available for specifying the level of parallelism required in a particular function:

Table 5: Options Available for Specifying the Level of Parallelism

Option	Description
XF_NPPC1	Process 1 pixel per clock cycle
XF_NPPC2	Process 2 pixels per clock cycle
XF_NPPC8	Process 8 pixels per clock cycle

Macros to Work With Parallelism

There are two macros that are defined to work with parallelism.

- The `XF_NPIXPERCYCLE(flags)` macro resolves to the number of pixels processed per cycle.
  - `XF_NPIXPERCYCLE(XF_NPPC1)` resolves to 1
  - `XF_NPIXPERCYCLE(XF_NPPC2)` resolves to 2
  - `XF_NPIXPERCYCLE(XF_NPPC8)` resolves to 8
- The `XF_BITSHIFT(flags)` macro resolves to the number of times to shift the image size to right to arrive at the final data transfer size for parallel processing.
  - `XF_BITSHIFT(XF_NPPC1)` resolves to 0
  - `XF_BITSHIFT(XF_NPPC2)` resolves to 1
  - `XF_BITSHIFT(XF_NPPC8)` resolves to 3

Pixel Types

Parameter types will differ, depending on the combination of the depth of pixels and the number of channels in the image. The generic nomenclature of the parameter is listed below.

```
XF_<Number of bits per pixel><signed (S) or unsigned (U) or float
(F)>C<number of channels>
```

For example, for an 8-bit pixel - unsigned - 1 channel the data type is `XF_8UC1`.

The following table lists the available data types for the `xF::Mat` class:

Table 6: xf::Mat Class - Available Data Types

Option	Number of bits per Pixel	Unsigned/ Signed/ Float Type	Number of Channels
XF_8UC1	8	Unsigned	1
XF_16UC1	16	Unsigned	1
XF_16SC1	16	Signed	1
XF_32UC1	32	Unsigned	1
XF_32FC1	32	Float	1
XF_32SC1	32	Signed	1
XF_8UC2	8	Unsigned	2
XF_8UC4	8	Unsigned	4
XF_2UC1	2	Unsigned	1

Manipulating Data Type

Based on the number of pixels to process per clock cycle and the type parameter, there are different possible data types. The xfOpenCV library uses these datatypes for internal processing and inside the `xF::Mat` class. The following are a few supported types:

- `XF_TNAME(TYPE,NPPC)` resolves to the data type of the data member of the `xF::Mat` object. For instance, `XF_TNAME(XF_8UC1,XF_NPPC8)` resolves to `ap_uint<64>`.
- `Word width = pixel depth * number of channels * number of pixels to process per cycle (NPPC).`
- `XF_DTUNAME(TYPE,NPPC)` resolves to the data type of the pixel. For instance, `XF_DTUNAME(XF_32FC1,XF_NPPC1)` resolves to `float`.
- `XF_PTSNAME(TYPE,NPPC)` resolves to the 'c' data type of the pixel. For instance, `XF_PTSNAME (XF_16UC1,XF_NPPC2)` resolves to `unsigned short`.

NOTE: *`ap_uint<>`, `ap_int<>`, `ap_fixed<>`, and `ap_ufixed<>` types belong to the high-level synthesis (HLS) library. For more information, see the Vivado Design Suite User Guide: High-Level Synthesis (UG902).*

Sample Illustration

The following code illustrates the configurations that are required to build the gaussian filter on an image, using SDSoC tool for Zynq® UltraScale™ platform.

NOTE: *In case of a real-time application, where the video is streamed in, it is recommended that the location of frame buffer is `xF::Mat` and is processed using the library function. The resultant location pointer is passed to display IPs.*

`xf_config_params.h`

```
#define FILTER_SIZE_3 1
#define FILTER_SIZE_5 0
#define FILTER_SIZE_7 0
```

`xf_gaussian_filter_tb.cpp`

```
int main(int argc, char **argv)
{
                cv::Mat in_img, out_img, ocv_ref;
        cv::Mat in_gray, in_gray1, diff;
in_img = cv::imread(argv[1], 1); // reading in the color image
                extractChannel(in_img, in_gray, 1);

xF::Mat<XF_8UC1, HEIGHT, WIDTH, XF_NPPC1>
imgInput(in_gray.rows,in_gray.cols);
xF::Mat<XF_8UC1, HEIGHT, WIDTH,XF_NPPC1>
imgOutput(in_gray.rows,in_gray.cols);
imgInput.copyTo(in_gray.data);
xFGaussianBlur<FILTER, XF_BORDER_CONSTANT,XF_8UC1, HEIGHT, WIDTH,
XF_NPPC1>(imgInput, imgOutput, sigma);
```

```
out_img.data = imgOutput.copyFrom();

            imwrite("output_hls.png", out_img);
}
```

xf_gaussian_filter.hpp

```
#pragma SDS data data_mover("_src.data":AXIDMA_SIMPLE)
#pragma SDS data data_mover("_dst.data":AXIDMA_SIMPLE)
#pragma SDS data access_pattern("_src.data":SEQUENTIAL)
#pragma SDS data copy("_src.data"[0:"_src.size"])
#pragma SDS data access_pattern("_dst.data":SEQUENTIAL)
#pragma SDS data copy("_dst.data"[0:"_dst.size"])

template<int FILTER_SIZE, int BORDER_TYPE, int SRC_T, int ROWS, int
COLS,int NPC = 1>
void xFGaussianBlur(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src,
xF::Mat<SRC_T, ROWS, COLS, NPC> & _dst, float sigma)
{
//function body
}
```

The design fetches data from external memory (with the help of SDSoC data movers) and is transferred to the function in 8-bit or 64-bit packets, based on the configured mode. Assuming 8-bits per pixel, 8 pixels can be packed into 64-bits. Therefore, 8 pixels are available to be processed in parallel.

Enable the `FILTER_SIZE_3_1` and the `NO` macros in the `xf_config_params.h` file. The `#define FILTER_SIZE_3 1` macro is used to set the filter size to `3x3` and `#define NO 1` macro enables 1 pixel parallelism.

Specify the SDSoC tool specific pragmas, in the `xf_gaussian_filter.hpp` file.

```
#pragma SDS data data_mover("_src.data":AXIDMA_SIMPLE)
#pragma SDS data data_mover("_dst.data":AXIDMA_SIMPLE)
#pragma SDS data access_pattern("_src.data":SEQUENTIAL)
#pragma SDS data copy("_src.data"[0:"_src.size"])
#pragma SDS data access_pattern("_dst.data":SEQUENTIAL)
#pragma SDS data copy("_dst.data"[0:"_dst.size"])
```

NOTE: *For more information on the pragmas used for hardware accelerator functions in SDSoC, see the SDSoC Environment User Guide (UG1027).*

xfOpenCV Library Functions

The xfOpenCV library is a set of select OpenCV functions optimized for Zynq-7000 All Programmable SoC and Zynq UltraScale+ MPSoC devices. The supported xfOpenCV library functions are listed in the following table:

Computations	Input Processing	Filters	Other
Absolute Difference	Bit Depth Conversion	Bilateral Filter	Canny Edge Detection
Accumulate	Channel Combine	Box Filter	FAST Corner Detection
Accumulate Squared	Channel Extract	Custom Convolution	Harris Corner Detection
Accumulate Weighted	Color Conversion	Dilate	Histogram Computation
Atan2	Histogram Equalization	Erode	Dense Pyramidal LK Optical Flow
Bitwise AND, Bitwise NOT, Bitwise OR, Bitwise XOR	Look Up Table	Gaussian Filter	Dense Non-Pyramidal LK Optical Flow
Gradient Magnitude	Remap	Sobel filter	MinMax Location
Gradient Phase	Resolution Conversion (Resize)	Median Blur Filter	Thresholding
Integral Image		Scharr Filter	WarpAffine
Inverse (Reciprocal)			WarpPerspective
Pixel-Wise Addition			SVM
Pixel-Wise Multiplication			Otsu Threshold
Pixel-Wise Subtraction			Mean Shift Tracking
Square Root			HOG
Mean and Standard Deviation			Stereo Local Block Matching
			WarpTransform
			Pyramid Up
			Pyramid Down

The functions that are not supported on Zynq-7000 All Programmable SoC devices, when configured to use 128-bit interfaces in 8 pixel per cycle mode are listed in the following table:

Computations	Input Processing	Filters
Accumulate	Bit Depth Conversion	Box Filter: signed 16-bit pixel type, and unsigned 16-bit pixel type
Accumulate Squared		Custom Convolution: signed 16-bit output pixel type
Accumulate Weighted		Sobel filter
Gradient Magnitude		Scharr Filter
Gradient Phase		
Pixel-Wise Addition: signed 16-bit pixel type, and unsigned 16-bit pixel type		
Pixel-Wise Multiplication: signed 16-bit pixel type, and unsigned 16-bit pixel type		
Pixel-Wise Subtraction: signed 16-bit pixel type, and unsigned 16-bit pixel type		

Absolute Difference

The `xFabsdiff` function finds the pixel wise absolute difference between two input images and returns an output image. The input and the output images must be the XF_8UC1 type.

$$I_{out}(x, y) = |I_{in1}(x, y) - I_{in2}(x, y)|$$

Where,

- $I_{out}(x, y)$ is the intensity of output image at (x,y) position.
- $I_{in1}(x, y)$ is the intensity of first input image at (x,y) position.
- $I_{in2}(x, y)$ is the intensity of second input image at (x,y) position.

API Syntax

```
template<int SRC_T, int ROWS, int COLS, int NPC=1>
void xFabsdiff(
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC>  src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC>  src2,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC>  dst )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 7: xFabsdiff Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado® HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 8: xFabsdiff Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	0	62	67	17
8 pixel	150	0	0	67	234	39

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 9: xFabsdiff Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.69

Deviation from OpenCV

There is no deviation from OpenCV, except that the `xFabsdiff` function supports 8-bit pixels.

Accumulate

The `xFaccumulate` function adds an image (src1) to the accumulator image (src2), and generates the accumulated result image (dst).

$$dst(x, y) = src1(x, y) + src2(x, y)$$

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFaccumulate (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int DST_T, int ROWS, int COLS, int NPC> dst )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 10: xFaccumulate Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_T	Output pixel type. Only 16-bit, unsigned, 1 channel is supported (XF_16UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 11: xFaccumulate Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 pixel	300	0	0	62	55	12
8 pixel	150	0	0	389	285	61

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 12: xFaccumulate Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.7

Deviation from OpenCV

In OpenCV the accumulated image is stored in the second input image. The src2 image acts as both input and output, as shown below:

$$src2(x,\ y) = src1(x,\ y) + src2(x,\ y)$$

Whereas, in the xfOpenCV implementation, the accumulated image is stored separately, as shown below:

$$dst(x,\ y) = src1(x,\ y) + src2(x,\ y)$$

Accumulate Squared

The `xFaccumulateSquare` function adds the square of an image (src1) to the accumulator image (src2) and generates the accumulated result (dst).

$$dst(x,\ y) = src1(x,\ y)^2 + src2(x,\ y)$$

The accumulated result is a separate argument in the function, instead of having src2 as the accumulated result. In this implementation, having a bi-directional accumulator is not possible as the function makes use of streams.

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFaccumulateSquare (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int DST_T, int ROWS, int COLS, int NPC> dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 13: xFaccumulateSquare Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_T	Output pixel type. Only 16-bit, unsigned, 1 channel is supported (XF_16UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 14: xFaccumulateSquare Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 pixel	300	0	1	71	52	14
8 pixel	150	0	8	401	247	48

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 15: xFaccumulateSquare Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.6

Deviation from OpenCV

In OpenCV the accumulated squared image is stored in the second input image. The src2 image acts as input as well as output.

$$src2(x, y) = src1(x, y)^2 + src2(x, y)$$

Whereas, in the xfOpenCV implementation, the accumulated squared image is stored separately. $$dst(x, y) = src1(x, y)^2 + src2(x, y)$$

Accumulate Weighted

The `xFaccumulateWeighted` function computes the weighted sum of the input image (src1) and the accumulator image (src2) and generates the result in dst.

$$dst(x, y) = alpha*src1(x, y) + (1 - alpha)*src2(x, y)$$

The accumulated result is a separate argument in the function, instead of having src2 as the accumulated result. In this implementation, having a bi-directional accumulator is not possible, as the function uses streams.

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFaccumulateWeighted (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int DST_T, int ROWS, int COLS, int NPC> dst,
float alpha )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 16: xFaccumulateWeighted Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_T	Output pixel type. Only 16-bit, unsigned, 1 channel is supported (XF_16UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image
alpha	Weight applied to input image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 17: xFaccumulateWeighted Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	5	295	255	52
8 pixel	150	0	19	556	476	88

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 18: xFaccumulateWeighted Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.7

Deviation from OpenCV

The resultant image in OpenCV is stored in the second input image. The src2 image acts as input as well as output, as shown below:

$$src2(x, y) = alpha*src1(x, y) + (1 - alpha)*src2(x, y)$$

Whereas, in xfOpenCV implementation, the accumulated weighted image is stored separately.

$$dst(x, y) = alpha*src1(x, y) + (1 - alpha)*src2(x, y)$$

Bilateral Filter

In general, any smoothing filter smoothens the image which will affect the edges of the image. To preserve the edges while smoothing, a bilateral filter can be used. In an analogous way as the Gaussian filter, the bilateral filter also considers the neighboring pixels with weights assigned to each of them. These weights have two components, the first of which is the same weighing used by the Gaussian filter. The second component takes into account the difference in the intensity between the neighboring pixels and the evaluated one.

The bilateral filter applied on an image is:

$$BF[I]_p = \frac{1}{W_p} \sum_{q \epsilon S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(\| I_p - I_q \|) I_q$$

Where

$$W_p = \sum_{q \epsilon S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(\| I_p - I_q \|)$$

and G_σ is a gaussian filter with variance σ.

The gaussian filter is given by: $G_\sigma = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$

API Syntax

```
template<int FILTER_SIZE, int BORDER_TYPE, int TYPE, int ROWS, int COLS,
int NPC=1>
void xFBilateralFilter (
xF::Mat<int TYPE, int ROWS, int COLS, int NPC> src,
xF::Mat<int TYPE, int ROWS, int COLS, int NPC> dst,
float sigma_space, float sigma_color )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 19: xFBilateralFilter Function Parameter Descriptions

Parameter	Description
FILTER_SIZE	Filter size. Filter size of 3 (XF_FILTER_3X3), 5 (XF_FILTER_5X5) and 7 (XF_FILTER_7X7) are supported
BORDER_TYPE	Border type supported is XF_BORDER_CONSTANT
TYPE	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; this function supports only XF_NPPC1 or 1 pixel per cycle operations.
src	Input image
dst	Output image
sigma_space	Standard deviation of filter in spatial domain
sigma_color	Standard deviation of filter used in color space

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to progress a grayscale HD (1080x1920) image.

Table 20: xFBilateralFilter Resource Utilization Summary

Operating Mode	Filter Size	Operating Frequency (MHz)	Utilization Estimate			
			BRAM_18K	DSP_48Es	FF	LUT
1 pixel	3x3	300	6	22	4934	4293
	5x5	300	12	30	5481	4943
	7x7	300	37	48	7084	6195

Performance Estimate

The following table summarizes a performance estimate of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 21: xFBilateralFilter Function Performance Estimate Summary

Operating Mode	Filter Size	Latency Estimate
		168 MHz
		Max (ms)
1 pixel	3x3	7.18
	5x5	7.20
	7x7	7.22

Deviation from OpenCV

Unlike OpenCV, xfOpenCV only supports filter sizes of 3, 5 and 7.

Bit Depth Conversion

The `xFConvertBitDepth` function converts the input image bit depth to the required bit depth in the output image.

API Syntax

```
template <int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFconvertTo(xF::Mat<SRC_T, ROWS, COLS, NPC> &_src_mat,
xF::Mat<DST_T, ROWS, COLS, NPC> &_dst_mat, ap_uint<4> _convert_type, int
_shift)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 22: xFConvertBitDepth Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. 8-bit, unsigned, 1 channel (XF_8UC1), 16-bit, unsigned, 1 channel (XF_16UC1), 16-bit, signed, 1 channel (XF_16SC1), 32-bit, unsigned, 1 channel (XF_32UC1) 32-bit, signed, 1 channel (XF_32SC1) are supported.
DST_T	Output pixel yype. 8-bit, unsigned, 1 channel (XF_8UC1), 16-bit, unsigned, 1 channel (XF_16UC1), 16-bit, signed, 1 channel (XF_16SC1), 32-bit, unsigned, 1 channel (XF_32UC1) 32-bit, signed, 1 channel (XF_32SC1) are supported.
ROWS	Height of input and output images
COLS	Width of input and output images
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_mat	Output image
_convert_type	This parameter specifies the type of conversion required. (See XF_convert_bit_depth_e enumerated type in file `xf_params.h` for possible values.)
_shift	Optional scale factor

Possible Conversions

The following table summarizes supported conversions. The rows are possible input image bit depths and the columns are corresponding possible output image bit depths (U=unsigned, S=signed).

Table 23: xFConvertBitDepth Function Supported Conversions

INPUT/OUTPUT	U8	U16	S16	U32	S32
U8	NA	yes	yes	NA	yes
U16	yes	NA	NA	NA	yes
S16	yes	NA	NA	NA	yes
U32	NA	NA	NA	NA	NA
S32	yes	yes	yes	NA	NA

Resource Utilization

The following table summarizes the resource utilization of the xFConvertBitDepth function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 24: xFConvertBitDepth Function Resource Utilization Summary For XF_CONVERT_8U_TO_16S Conversion

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	8	581	523	119
8 pixel	150	0	8	963	1446	290

Table 25: xFConvertBitDepth Function Resource Utilization Summary For XF_CONVERT_16U_TO_8U Conversion

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	8	591	541	124
8 pixel	150	0	8	915	1500	308

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 26: xFConvertBitDepth Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency
1 pixel operation (300 MHz)	6.91 ms
8 pixel operation (150 MHz)	1.69 ms

Bitwise AND

The `xFbitwise_and` function performs the bitwise AND operation for each pixel between two input images, and returns an output image.

$$I_{out}(x, y) = I_{in1}(x, y) \ \& \ I_{in2}(x, y)$$

Where,

- $I_{out}(x, y)$ is the intensity of output image at (x, y) position
- $I_{in1}(x, y)$ is the intensity of first input image at (x, y) position
- $I_{in2}(x, y)$ is the intensity of second input image at (x, y) position

API Syntax

```
template<int SRC_T, int ROWS, int COLS, int NPC=1>
void xFbitwise_and (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> dst )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 27: xFbitwise_and Function Parameter Descriptions

Parameter	Description
SRC_T	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 28: xFbitwise_and Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	0	62	44	10
8 pixel	150	0	0	59	72	13

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 29: xFbitwise_and Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.7

Bitwise NOT

The `xFbitwise_not` function performs the pixel wise bitwise NOT operation for the pixels in the input image, and returns an output image. $I_{out}(x, y) = \sim I_{in}(x, y)$

Where,

- $I_{out}(x, y)$ is the intensity of output image at (x, y) position
- $I_{in}(x, y)$ is the intensity of input image at (x, y) position

API Syntax

```
template<int SRC_T, int ROWS, int COLS, int NPC=1>
void xFbitwise_not (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> dst )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 30: xFbitwise_not Function Parameter Descriptions

Parameter	Description
SRC_T	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 31: xFbitwise_not Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	0	97	78	20
8 pixel	150	0	0	88	97	21

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 32: xFbitwise_not Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.7

Bitwise OR

The `xFbitwise_or` function performs the pixel wise bitwise OR operation between two input images, and returns an output image. $I_{out}(x, y) = I_{in1}(x, y) \mid I_{in2}(x, y)$

Where,

- $I_{out}(x, y)$ is the intensity of output image at (x, y) position
- $I_{in1}(x, y)$ is the intensity of first input image at (x, y) position
- $I_{in2}(x, y)$ is the intensity of second input image at (x, y) position

API Syntax

```
template<int SRC_T, int ROWS, int COLS, int NPC=1>
void xFbitwise_or (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> dst )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 33: xFbitwise_or Function Parameter Descriptions

Parameter	Description
SRC_T	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 34: xFbitwise_or Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	0	62	44	10
8 pixel	150	0	0	59	72	13

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 35: xFbitwise_or Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.7

Bitwise XOR

The `xFbitwise_xor` function performs the pixel wise bitwise XOR operation between two input images, and returns an output image, as shown below:

$$I_{out}(x, y) = I_{in1}(x, y) \oplus I_{in2}(x, y)$$

Where,

- $I_{out}(x, y)$ is the intensity of output image at (x, y) position
- $I_{in1}(x, y)$ is the intensity of first input image at (x, y) position
- $I_{in2}(x, y)$ is the intensity of second input image at (x, y) position

API Syntax

```
template<int SRC_T, int ROWS, int COLS, int NPC=1>
void xFbitwise_xor(
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> dst )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 36: xFbitwise_xor Function Parameter Descriptions

Parameter	Description
SRC_T	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image:

Table 37: xFbitwise_xor Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	0	62	44	10
8 pixel	150	0	0	59	72	13

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image:

Table 38: xFbitwise_xor Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.7

Box Filter

The `xFboxfilter` function performs box filtering on the input image. Box filter acts as a low-pass filter and performs blurring over the image. The `xFboxfilter` function or the box blur is a spatial domain linear filter in which each pixel in the resulting image has a value equal to the average value of the neighboring pixels in the image.

$$K_{box} = \frac{1}{(ksize*ksize)}\begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 1 \\ 1 & \dots & 1 \end{bmatrix}$$

API Syntax

```
template<int BORDER_TYPE,int FILTER_TYPE, int SRC_T, int ROWS, int
COLS,int NPC=1>
void xFboxfilter(xF::Mat<SRC_T, ROWS, COLS, NPC> &
_src_mat,xF::Mat<SRC_T, ROWS, COLS, NPC> & _dst_mat)
```

Parameter Descriptions

The following table describes the template and the function parameters.:

Table 39: xFboxfilter Function Parameter Descriptions

Parameter	Description
FILTER_SIZE	Filter size. Filter size of 3(XF_FILTER_3X3), 5(XF_FILTER_5X5) and 7(XF_FILTER_7X7) are supported
BORDER_TYPE	Border Type supported is XF_BORDER_CONSTANT
SRC_T	Input and output pixel type. 8-bit, unsigned, 16-bit unsigned and 16-bit signed, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_mat	Output image

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 40: xFboxfilter Function Resource Utilization Summary

Operating Mode	Filter Size	Operating Frequency (MHz)	Utilization Estimate				
			BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	3x3	300	3	1	545	519	104
	5x5	300	5	1	876	870	189
	7x7	300	7	1	1539	1506	300
8 pixel	3x3	150	6	8	1002	1368	264
	5x5	150	10	8	1576	3183	611
	7x7	150	14	8	2414	5018	942

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image:

Table 41: xFboxfilter Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Filter Size	Latency Estimate
			Max (ms)
1 pixel	300	3x3	7.2
	300	5x5	7.21
	300	7x7	7.22
8 pixel	150	3x3	1.7
	150	5x5	1.7
	150	7x7	1.7

Canny Edge Detection

The Canny edge detector finds the edges in an image or video frame. It is one of the most popular algorithms for edge detection. Canny algorithm aims to satisfy three main criteria:

1. Low error rate: A good detection of only existent edges.
2. Good localization: The distance between edge pixels detected and real edge pixels have to be minimized.
3. Minimal response: Only one detector response per edge.

In this algorithm, the noise in the image is reduced first by applying a Gaussian mask. The Gaussian mask used here is the average mask of size 3x3. Thereafter, gradients along x and y directions are computed using the Sobel gradient function. The gradients are used to compute the magnitude and phase of the pixels. The phase is quantized and the pixels are binned accordingly. Non-maximal suppression is applied on the pixels to remove the weaker edges.

Edge tracing is applied on the remaining pixels to draw the edges on the image. In this algorithm, the canny up to non-maximal suppression is in one kernel and the edge linking module is in another kernel. After non-maxima suppression, the output is represented as 2-bit per pixel, Where:

- `00` - represents the background
- `01` - represents the weaker edge
- `11` - represents the strong edge

The output is packed as 8-bit (four 2-bit pixels) in 1 pixel per cycle operation and packed as 16-bit (eight 2-bit pixels) in 8 pixel per cycle operation. For the edge linking module, the input is 64-bit, such 32 pixels of 2-bit are packed into a 64-bit. The edge tracing is applied on the pixels and returns the edges in the image.

API Syntax

The API syntax for `xFcanny` is:

```
template<int FILTER_TYPE,int NORM_TYPE,int SRC_T,int DST_T, int ROWS, int
COLS,int NPC,int NPC1>
void xFcanny(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat,xF::Mat<DST_T,
ROWS, COLS, NPC1> & _dst_mat,unsigned char _lowthreshold,unsigned char
_highthreshold)
```

The API syntax for `xFEdgeTracing` is:

```
template<int SRC_T, int DST_T, int ROWS, int COLS,int NPC_SRC,int NPC_DST>
void xFEdgeTracing(xF::Mat<SRC_T, ROWS, COLS, NPC_SRC> &
_src,xF::Mat<DST_T, ROWS, COLS, NPC_DST> & _dst)
```

Parameter Descriptions

The following table describes the `xFcanny` template and function parameters:

Table 42: xfcanny Function Parameter Descriptions

Parameter	Description
FILTER_TYPE	The filter window dimensions. The options are 3 and 5.
NORM_TYPE	The type of norm used. The options for norm type are L1NORM and L2NORM.
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_T	Output pixel type. The output in 1pixel case is 8-bit and packing 4 2-bit pixel values into 8-bit. The Output in 8pixel case is 16-bit, 8-bit, 2-bit pixel values are packing into 16-bit.
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_mat	Output image
_lowthreshold	The lower value of threshold for binary thresholding.
_highthreshold	The higher value of threshold for binary thresholding.

The following table describes the `xFEdgeTracing` template and function parameters:

Table 43: xFEdgeTracing Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type
DST_T	Output pixel type
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC_SRC	Number of pixels to be processed per cycle. Fixed to XF_NPPC32.
NPC_DST	Number of pixels to be written to destination. Fixed to XF_NPPC8.
_src	Input image
_dst	Output image

Resource Utilization

The following table summarizes the resource utilization of `xFcanny` and `xFEdgeTracing` in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image for Filter size is 3.

Table 44: xFcanny and xFEdgeTracing Function Resource Utilization Summary

Name	Resource Utilization					
	1 pixel	1 pixel	8 pixel	8 pixel	Edge Linking	Edge Linking
	L1NORM,FS:3	L2NORM,FS:3	L1NORM,FS:3	L2NORM,FS:3		
	300 MHz	300 MHz	150 MHz	150 MHz	300 MHz	150 MHz
BRAM_18K	22	18	36	32	84	84
DSP48E	2	4	16	32	3	3
FF	3027	3507	4899	6208	17600	14356
LUT	2626	3170	6518	9560	15764	14274
CLB	606	708	1264	1871	2955	3241

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image for L1NORM, filter size is 3 and including the edge linking module.

Table 45: xFcanny and xFEdgeTracing Function Performance Estimate Summary

Operating Mode	Latency Estimate	
	Operating Frequency (MHz)	Latency (in ms)
1 pixel	300	10.2
8 pixel	150	8

Deviation from OpenCV

Gaussian filter is not available in OpenCV. Edge linking module is not available in the xfOpenCV library.

Channel Combine

The `xFmerge` function, merges single channel images into a multi-channel image. The number of channels to be merged should be four.

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFmerge(xF::Mat<SRC_T, ROWS, COLS, NPC> &_src1, xF::Mat<SRC_T, ROWS,
COLS, NPC> &_src2, xF::Mat<SRC_T, ROWS, COLS, NPC> &_src3, xF::Mat<SRC_T,
ROWS, COLS, NPC> &_src4, xF::Mat<DST_T, ROWS, COLS, NPC> &_dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 46: xFmerge Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 4 channel is supported (XF_8UC1)
DST_T	Output pixel type. Only 8-bit, unsigned,1 channel is supported (XF_8UC4)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 for 1 pixel operation.
_src1	Input single-channel image
_src2	Input single-channel image
_src3	Input single-channel image
_src4	Input single-channel image
_dst	Output multi-channel image

Resource Utilization

The following table summarizes the resource utilization of the xFmerge function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process 4 single-channel HD (1080x1920) images.

Table 47: xFmerge Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	8	494	386	85

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process 4 single channel HD (1080x1920) images.

Table 48: xFmerge Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency
1 pixel operation (300 MHz)	6.92 ms

Channel Extract

The `xFextractChannel` function splits a multi-channel array (32-bit pixel-interleaved data) into several single-channel arrays and returns a single channel. The channel to be extracted is specified by using the channel argument.

The value of the channel argument is specified by macros defined in the `xf_channel_extract_e` enumerated data type. The following table summerizes the possible values for the `xf_channel_extract_e` enumerated data type:

Table 49: xf_channel_extract_e Enumerated Data Type Values

Channel	Enumerated Type
Unknown	XF_EXTRACT_CH_0
Unknown	XF_EXTRACT_CH_1
Unknown	XF_EXTRACT_CH_2
Unknown	XF_EXTRACT_CH_3
RED	XF_EXTRACT_CH_R
GREEN	XF_EXTRACT_CH_G
BLUE	XF_EXTRACT_CH_B
ALPHA	XF_EXTRACT_CH_A
LUMA	XF_EXTRACT_CH_Y
Cb/U	XF_EXTRACT_CH_U
Cr/V/Value	XF_EXTRACT_CH_V

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFextractChannel(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat,
xF::Mat<DST_T, ROWS, COLS, NPC> & _dst_mat, uint16_t _channel)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 50: xFextractChannel Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 4channel is supported (XF_8UC4)
DST_T	Output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 for 1 pixel operation.
_src_mat	Input multi-channel image
_dst_mat	Output single channel image
_channel	Channel to be extracted (See xf_channel_extract_e enumerated type in file `xf_params.h` for possible values.)

Resource Utilization

The following table summarizes the resource utilization of the xfextractChannel function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a 4 channel HD (1080x1920) image.

Table 51: xFextractChannel Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	8	508	354	96

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a 4 channel HD (1080x1920) image.

Table 52: xFextractChannel Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.92

Color Conversion

The color conversion functions convert one image format to another image format, for the combinations listed in the following table. The rows represent the input formats and the columns represent the output formats. Supported conversions are discussed in the following sections.

Table 53: Supported Color Conversions

I/O Formats	RGBA	NV12	NV21	IYUV	UYVY	YUYV	YUV4
RGBA	N/A	For details, see the RGBA to NV12	For details, see the RGBA to NV21	For details, see the RGBA to IYUV			For details, see the RGBA to YUV4
NV12	For details, see the NV12 to RGBA	N/A		For details, see the NV12 to IYUV			For details, see the NV12 to YUV4
NV21	For details, see the NV21 to RGBA		N/A	For details, see the NV21 to IYUV			For details, see the NV21 to YUV4
IYUV	For details, see the IYUV to RGBA	For details, see the IYUV to NV12		N/A			For details, see the IYUV to YUV4
UYVY	For details, see the UYVY to RGBA	For details, see the UYVY to NV12		For details, see the UYVY to IYUV	N/A		
YUYV	For details, see the YUYV to RGBA	For details, see the YUYV to NV12		For details, see the YUYV to IYUV		N/A	
YUV4							N/A

RGB to YUV Conversion Matrix

Following is the formula to convert RGB data to YUV data:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 & 16 \\ -0.148 & -0.291 & 0.439 & 128 \\ 0.439 & -0.368 & -0.071 & 128 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \\ 1 \end{bmatrix}$$

YUV to RGB Conversion Matrix

Following is the formula to convert YUV data to RGB data:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0 & 1.596 \\ 1.164 & -0.391 & -0.813 \\ 1.164 & 2.018 & 0 \end{bmatrix} \begin{bmatrix} (Y - 16) \\ (U - 128) \\ (V - 128) \end{bmatrix}$$

Source: http://www.fourcc.org/fccyvrgb.php

RGBA to YUV4

The `xFrgba2yuv4` function converts a 4-channel RGBA image to YUV444 format. The function outputs Y, U, and V streams separately.

API Syntax

```
template <int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFrgba2yuv4(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<DST_T,
ROWS, COLS, NPC> & _y_image, xF::Mat<DST_T, ROWS, COLS, NPC> & _u_image,
xF::Mat<DST_T, ROWS, COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 54: xFrgba2yuv4 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 4-channel is supported (XF_8UC4).
DST_T	Output pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input Y plane of size (ROWS, COLS).
_y_image	Output Y image of size (ROWS, COLS).
_u_image	Output U image of size (ROWS, COLS).
_v_image	Output V image of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of RGBA to YUV4 for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 55: xFrgba2yuv4 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	9	589	328	96

Performance Estimate

The following table summarizes the performance of RGBA to YUV4 for different configurations, as generated using the Vivado HLS 2017.1 version for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 56: xFrgba2yuv4 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	1.89

RGBA to IYUV

The `xFrgba2iyuv` function converts a 4-channel RGBA image to IYUV (4:2:0) format. The function outputs Y, U, and V planes separately. IYUV holds subsampled data, Y is sampled for every RGBA pixel and U,V are sampled once for 2row and 2column(2x2) pixels. U and V planes are of (rows/2)*(columns/2) size, by cascading the consecutive rows into a single row the planes size becomes (rows/4)*columns.

API Syntax

```
template <int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFrgba2iyuv(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<DST_T,
ROWS, COLS, NPC> & _y_image, xF::Mat<DST_T, ROWS/4, COLS, NPC> &
_u_image, xF::Mat<DST_T, ROWS/4, COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 57: xFrgba2iyuv Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit,unsigned, 4-channel is supported (XF_8UC4).
DST_T	Output pixel type. Only 8-bit,unsigned, 1-channel is supported (XF_8UC1).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input Y plane of size (ROWS, COLS).
_y_image	Output Y image of size (ROWS, COLS).
_u_image	Output U image of size (ROWS/4, COLS).
_v_image	Output V image of size (ROWS/4, COLS).

Resource Utilization

The following table summarizes the resource utilization of RGBA to IYUV for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 58: xFrgba2iyuv Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	9	816	472	149

Performance Estimate

The following table summarizes the performance of RGBA to IYUV for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 59: xFrgba2iyuv Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	1.8

RGBA to NV12

The `xFrgba2nv12` function converts a 4-channel RGBA image to NV12 (4:2:0) format. The function outputs Y plane and interleaved UV plane separately. NV12 holds the subsampled data, Y is sampled for every RGBA pixel and U, V are sampled once for 2row and 2columns (2x2) pixels. UV plane is of (rows/2)*(columns/2) size as U and V values are interleaved.

API Syntax

```
template <int SRC_T, int Y_T, int UV_T, int ROWS, int COLS, int NPC=1>
void xFrgba2nv12(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<Y_T,
ROWS, COLS, NPC> & _y, xF::Mat<UV_T, ROWS/2, COLS/2, NPC> & _uv)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 60: xFrgba2nv12 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit,unsigned, 4-channel is supported (XF_8UC4).
Y_T	Output pixel type. Only 8-bit,unsigned, 1-channel is supported (XF_8UC1).
UV_T	Output pixel type. Only 8-bit,unsigned, 2-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input RGBA image of size (ROWS, COLS).
_y	Output Y image of size (ROWS, COLS).
_uv	Output UV image of size (ROWS/2, COLS/2).

Resource Utilization

The following table summarizes the resource utilization of RGBA to NV12 for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 61: xFrgba2nv12 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	9	802	452	128

Performance Estimate

The following table summarizes the performance of RGBA to NV12 for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 62: xFrgba2nv12 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	1.8

RGBA to NV21

The `xFrgba2nv21` function converts a 4-channel RGBA image to NV21 (4:2:0) format. The function outputs Y plane and interleaved VU plane separately. NV21 holds subsampled data, Y is sampled for every RGBA pixel and U, V are sampled once for 2 row and 2 columns (2x2) RGBA pixels. UV plane is of (rows/2)*(columns/2) size as V and U values are interleaved.

API Syntax

```
template <int SRC_T, int Y_T, int UV_T, int ROWS, int COLS, int NPC=1>
void xFrgba2nv21(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<Y_T,
ROWS, COLS, NPC> & _y, xF::Mat<UV_T, ROWS/2, COLS/2, NPC> & _uv)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 63: xFrgba2nv21 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 4-channel is supported (XF_8UC4).
Y_T	Output pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Output pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input RGBA image of size (ROWS, COLS).
_y	Output Y image of size (ROWS, COLS).
_uv	Output UV image of size (ROWS/2, COLS/2).

Resource Utilization

The following table summarizes the resource utilization of RGBA to NV21 for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 64: xFrgba2nv21 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	9	802	453	131

Performance Estimate

The following table summarizes the performance of RGBA to NV21 for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 65: xFrgba2nv21 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	1.89

YUYV to RGBA

The `xFyuyv2rgba` function converts a single-channel YUYV (YUV 4:2:2) image format to a 4-channel RGBA image. YUYV is a sub-sampled format, a set of YUYV value gives 2 RGBA pixel values. YUYV is represented in 16-bit values where as, RGBA is represented in 32-bit values.

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFyuyv2rgba(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<DST_T,
ROWS, COLS, NPC> & _dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 66: xFyuyv2rgba Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 16-bit, unsigned, 1-channel is supported (XF_16UC1).
DST_T	Output pixel type. Only 8-bit, unsigned, 4-channel is supported (XF_8UC4).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input image of size (ROWS, COLS).
_dst	Output image of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of YUYV to RGBA for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 67: xFyuyv2rgba Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	6	765	705	165

Performance Estimate

The following table summarizes the performance of UYVY to RGBA for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 68: xFyuyv2rgba Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9

YUYV to NV12

The `xFyuyv2nv12` function converts a single-channel YUYV (YUV 4:2:2) image format to NV12 (YUV 4:2:0) format. YUYV is a sub-sampled format, 1 set of YUYV value gives 2 Y values and 1 U and V value each.

API Syntax

```
template<int SRC_T,int Y_T,int UV_T,int ROWS,int COLS,int NPC=1,int
NPC_UV=1>
void xFyuyv2nv12(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src,xF::Mat<Y_T,
ROWS, COLS, NPC> & _y_image,xF::Mat<UV_T, ROWS/2, COLS/2, NPC_UV> &
_uv_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 69: xFyuyv2nv12 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 16-bit, unsigned, 1-channel is supported (XF_16UC1).
Y_T	Output pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Output UV image pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
NPC_UV	Number of UV image Pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC4 for 1 pixel and 8 pixel operations respectively.
_src	Input image of size (ROWS, COLS).
_y_image	Output Y plane of size (ROWS, COLS).
_uv_image	Output U plane of size (ROWS/2, COLS/2).

Resource Utilization

The following table summarizes the resource utilization of YUYV to NV12 for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 70: xFyuyv2nv12 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	0	831	491	149
8pixel	150	0	0	1196	632	161

Performance Estimate

The following table summarizes the performance of YUYV to NV12 for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 71: xFyuyv2nv12 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9
8 pixel operation(150Mhz)	1.7

YUYV to IYUV

The `xFyuyv2iyuv` function converts a single-channel YUYV (YUV 4:2:2) image format to IYUV(4:2:0) format. Outputs of the function are separate Y, U, and V planes. YUYV is a sub-sampled format, 1 set of YUYV value gives 2 Y values and 1 U and V value each. U, V values of the odd rows are dropped as U, V values are sampled once for 2 rows and 2 colums in the IYUV(4:2:0) format.

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFyuyv2iyuv(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<DST_T,
ROWS, COLS, NPC> & _y_image, xF::Mat<DST_T, ROWS/4, COLS, NPC> &
_u_image, xF::Mat<DST_T, ROWS/4, COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 72: xFyuyv2iyuv Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 16-bit, unsigned,1 channel is supported (XF_16UC1).
DST_T	Output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input image of size (ROWS, COLS).
_y_image	Output Y plane of size (ROWS, COLS).
_u_image	Output U plane of size (ROWS/4, COLS).
_v_image	Output V plane of size (ROWS/4, COLS).

Resource Utilization

The following table summarizes the resource utilization of YUYV to IYUV for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 73: xFyuyv2iyuv Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	0	835	497	149
8pixel	150	0	0	1428	735	210

Performance Estimate

The following table summarizes the performance of YUYV to IYUV for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 74: xFyuyv2iyuv Function Performance Estimate

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9
8 pixel operation(150Mhz)	1.7

UYVY to IYUV

The `xFuyvy2iyuv` function converts a UYVY (YUV 4:2:2) single-channel image to the IYUV format. The outputs of the functions are separate Y, U, and V planes. UYVY is sub sampled format. 1 set of UYVY value gives 2 Y values and 1 U and V value each.

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFuyvy2iyuv(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<DST_T,
ROWS, COLS, NPC> & _y_image,xF::Mat<DST_T, ROWS/4, COLS, NPC> & _u_image,
xF::Mat<DST_T, ROWS/4, COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 75: xFuyvy2iyuv Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 16-bit, unsigned, 1-channel is supported (XF_16UC1).
DST_T	Output pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input image of size (ROWS, COLS).
_y_image	Output Y plane of size (ROWS, COLS).
_u_image	Output U plane of size (ROWS/4, COLS).
_v_image	Output V plane of size (ROWS/4, COLS).

Resource Utilization

The following table summarizes the resource utilization of UYVY to IYUV for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image..

Table 76: xFuyvy2iyuv Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	0	835	494	139

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
8pixel	150	0	0	1428	740	209

Performance Estimate

The following table summarizes the performance of UYVY to IYUV for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 77: xFuyvy2iyuv Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9
8 pixel operation(150Mhz)	1.7

UYVY to RGBA

The `xFuyvy2rgba` function converts a UYVY (YUV 4:2:2) single-channel image to a 4-channel RGBA image. UYVY is sub sampled format, 1set of UYVY value gives 2 RGBA pixel values. UYVY is represented in 16-bit values where as RGBA is represented in 32-bit values.

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFuyvy2rgba(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<DST_T,
ROWS, COLS, NPC> & _dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 78: xFuyvy2rgba Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 16-bit, unsigned, 1-channel is supported (XF_16UC1).
DST_T	Output pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input image of size (ROWS, COLS).
_dst	Output image of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of UYVY to RGBA for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 79: xFuyvy2rgba Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	6	773	704	160

Performance Estimate

The following table summarizes the performance of UYVY to RGBA for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 80: xFuyvy2rgba Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.8

UYVY to NV12

The `xFuyvy2nv12` function converts a UYVY (YUV 4:2:2) single-channel image to NV12 format. The outputs are separate Y and UV planes. UYVY is sub sampled format, 1 set of UYVY value gives 2 Y values and 1 U and V value each.

API Syntax

```
template<int SRC_T, int Y_T, int UV_T, int ROWS, int COLS, int NPC=1, int
NPC_UV=1>
void xFuyvy2nv12(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src,xF::Mat<Y_T,
ROWS, COLS, NPC> & _y_image,xF::Mat<UV_T, ROWS/2, COLS/2, NPC_UV> &
_uv_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 81: xFuyvy2nv12 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 16-bit, unsigned, 1-channel is supported (XF_16UC1).
Y_T	Output pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Output UV image pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
NPC_UV	Number of UV image Pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC4 for 1 pixel and 8 pixel operations respectively.
_src	Input image of size (ROWS, COLS).
_y_image	Output Y plane of size (ROWS, COLS).
_uv_image	Output U plane of size (ROWS/2, COLS/2).

Resource Utilization

The following table summarizes the resource utilization of UYVY to NV12 for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 82: xFuyvy2nv12 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	0	831	488	131
8pixel	150	0	0	1235	677	168

Performance Estimate

The following table summarizes the performance of UYVY to NV12 for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 83: xFuyvy2nv12 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9
8 pixel operation(150Mhz)	1.7

IYUV to RGBA

The `xFiyuv2rgba` function converts single channel IYUV (YUV 4:2:0) image to a 4-channel RGBA image. The inputs to the function are separate Y, U, and V planes. IYUV is sub sampled format, U and V values are sampled once for 2 rows and 2 columns of the RGBA pixels. The data of the consecutive rows of size (columns/2) is combined to form a single row of size (columns).

API Syntax

```
template<int SRC_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFiyuv2rgba(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y, xF::Mat<SRC_T,
ROWS/4, COLS, NPC> & src_u,xF::Mat<SRC_T, ROWS/4, COLS, NPC> & src_v,
xF::Mat<DST_T, ROWS, COLS, NPC> & _dst0)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 84: xFiyuv2rgba Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
DST_T	Output pixel type. Only 8-bit, unsigned, 4-channel is supported (XF_8UC4).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_u	Input U plane of size (ROWS/4, COLS).
src_v	Input V plane of size (ROWS/4, COLS).
_dst0	Output RGBA image of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of IYUV to RGBA for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 85: xFiyuv2rgba Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	2	5	1208	728	196

Performance Estimate

The following table summarizes the performance of IYUV to RGBA for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 86: xFiyuv2rgba Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9

IYUV to NV12

The `xFiyuv2nv12` function converts single channel IYUV image to NV12 format. The inputs are separate U and V planes. There is no need of processing Y plane as both the formats have a same Y plane. U and V values are rearranged from plane interleaved to pixel interleaved.

API Syntax

```
template<int SRC_T, int UV_T, int ROWS, int COLS, int NPC =1, int
NPC_UV=1>
void xFiyuv2nv12(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y, xF::Mat<SRC_T,
ROWS/4, COLS, NPC> & src_u,xF::Mat<SRC_T, ROWS/4, COLS, NPC> &
src_v,xF::Mat<SRC_T, ROWS, COLS, NPC> & _y_image, xF::Mat<UV_T, ROWS/2,
COLS/2, NPC_UV> & _uv_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 87: xFiyuv2nv12 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Output pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
NPC_UV	Number of UV Pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC4 for 1 pixel and 4-pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_u	Input U plane of size (ROWS/4, COLS).
src_v	Input V plane of size (ROWS/4, COLS).
_y_image	Output V plane of size (ROWS, COLS).
_uv_image	Output UV plane of size (ROWS/2, COLS/2).

Resource Utilization

The following table summarizes the resource utilization of IYUV to NV12 for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image..

Table 88: xFiyuv2nv12 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	12	907	677	158
8pixel	150	0	12	1591	1022	235

Performance Estimate

The following table summarizes the performance of IYUV to NV12 for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 89: xFiyuv2nv12 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9
8 pixel operation(150Mhz)	1.7

IYUV to YUV4

The `xFiyuv2yuv4` function converts a single channel IYUV image to a YUV444 format. Y plane is same for both the formats. The inputs are separate U and V planes of IYUV image and the outputs are separate U and V planes of YUV4 image. IYUV stores subsampled U,V values. YUV format stores U and V values for every pixel. The same U, V values are duplicated for 2 rows and 2 columns (2x2) pixels in order to get the required data in the YUV444 format.

API Syntax

```
template<int SRC_T, int ROWS, int COLS, int NPC=1>
void xFiyuv2yuv4(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y, xF::Mat<SRC_T,
ROWS/4, COLS, NPC> & src_u,xF::Mat<SRC_T, ROWS/4, COLS, NPC> &
src_v,xF::Mat<SRC_T, ROWS, COLS, NPC> & _y_image, xF::Mat<SRC_T, ROWS,
COLS, NPC> & _u_image, xF::Mat<SRC_T, ROWS, COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 90: xFiyuv2yuv4 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_u	Input U plane of size (ROWS/4, COLS).
src_v	Input V plane of size (ROWS/4, COLS).
_y_image	Output Y image of size (ROWS, COLS).
_u_image	Output U image of size (ROWS, COLS).
_v_image	Output V image of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of IYUV to YUV4 for different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 91: xFiyuv2yuv4 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	0	1398	870	232
8pixel	150	0	0	2134	1214	304

Performance Estimate

The following table summarizes the performance of IYUV to YUV4 for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 92: xFiyuv2yuv4 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	13.8
8 pixel operation(150Mhz)	3.4

NV12 to IYUV

The `xFnv122iyuv` function converts NV12 format to IYUV format. The function inputs the interleaved UV plane and the outputs are separate U and V planes. There is no need of processing the Y plane as both the formats have a same Y plane. U and V values are rearranged from pixel interleaved to plane interleaved.

API Syntax

```
template<int SRC_T, int UV_T, int ROWS, int COLS, int NPC=1, int NPC_UV=1>
void xFnv122iyuv(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y, xF::Mat<UV_T,
ROWS/2, COLS/2, NPC_UV> & src_uv,xF::Mat<SRC_T, ROWS, COLS, NPC> &
_y_image,xF::Mat<SRC_T, ROWS/4, COLS, NPC> & _u_image,xF::Mat<SRC_T, ROWS/
4, COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 93: xFnv122iyuv Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Input pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
NPC_UV	Number of UV image Pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC4 for 1 pixel and 4-pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_uv	Input UV plane of size (ROWS/2, COLS/2).
_y_image	Output Y plane of size (ROWS, COLS).
_u_image	Output U plane of size (ROWS/4, COLS).
_v_image	Output V plane of size (ROWS/4, COLS).

Resource Utilization

The following table summarizes the resource utilization of NV12 to IYUV for different configurations, as generated in the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 94: xFnv122iyuv Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	1	1344	717	208
8pixel	150	0	1	1961	1000	263

Performance Estimate

The following table summarizes the performance of NV12 to IYUV for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 95: xFnv122iyuv Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9
8 pixel operation(150Mhz)	1.7

NV12 to RGBA

The `xFnv122rgba` function converts NV12 image format to a 4-channel RGBA image. The inputs to the function are separate Y and UV planes. NV12 holds sub sampled data, Y plane is sampled at unit rate and 1 U and 1V value each for every 2x2 Y values. To generate the RGBA data, each U and V value is duplicated (2x2) times.

API Syntax

```
template<int SRC_T, int UV_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFnv122rgba(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y,xF::Mat<UV_T,
ROWS/2, COLS/2, NPC> & src_uv,xF::Mat<DST_T, ROWS, COLS, NPC> & _dst0)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 96: xFnv122rgba Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Input pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
DST_T	Output pixel type. Only 8-bit,unsigned,4channel is supported (XF_8UC4).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_uv	Input UV plane of size (ROWS/2, COLS/2).
_dst0	Output RGBA image of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of NV12 to RGBA for different configurations, as generated in the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 97: xFnv122rgba Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	2	5	1191	708	195

Performance Estimate

The following table summarizes the performance of NV12 to RGBA for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 98: xFnv122rgba Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9

NV12 to YUV4

The `xFnv122yuv4` function converts a NV12 image format to a YUV444 format. The function outputs separate U and V planes. Y plane is same for both the image formats. The UV planes are duplicated 2x2 times to represent one U plane and V plane of the YUV444 image format.

API Syntax

```
template<int SRC_T,int UV_T, int ROWS, int COLS, int NPC=1, int NPC_UV=1>
void xFnv122yuv4(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y, xF::Mat<UV_T,
ROWS/2, COLS/2, NPC_UV> & src_uv,xF::Mat<SRC_T, ROWS, COLS, NPC> &
_y_image, xF::Mat<SRC_T, ROWS, COLS, NPC> & _u_image,xF::Mat<SRC_T, ROWS,
COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 99: xFnv122yuv4 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Input pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
NPC_UV	Number of UV image Pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC4 for 1 pixel and 4-pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_uv	Input UV plane of size (ROWS/2, COLS/2).
_y_image	Output Y plane of size (ROWS, COLS).
_u_image	Output U plane of size (ROWS, COLS).
_v_image	Output V plane of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of NV12 to YUV4 for different configurations, as generated in the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 100: xFnv122yuv4 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	0	1383	832	230
8pixel	150	0	0	1772	1034	259

Performance Estimate

The following table summarizes the performance of NV12 to YUV4 for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 101: xFnv122yuv4 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	13.8
8 pixel operation(150Mhz)	3.4

NV21 to IYUV

The `xFnv212iyuv` function converts a NV21 image format to an IYUV image format. The input to the function is the interleaved VU plane only and the outputs are separate U and V planes. There is no need of processing Y plane as both the formats have same the Y plane. U and V values are rearranged from pixel interleaved to plane interleaved.

API Syntax

```
template<int SRC_T, int UV_T, int ROWS, int COLS, int NPC=1,int NPC_UV=1>
void xFnv212iyuv(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y, xF::Mat<UV_T,
ROWS/2, COLS/2, NPC_UV> & src_uv,xF::Mat<SRC_T, ROWS, COLS, NPC> &
_y_image, xF::Mat<SRC_T, ROWS/4, COLS, NPC> & _u_image,xF::Mat<SRC_T,
ROWS/4, COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 102: xFnv212iyuv Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Input pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
NPC_UV	Number of UV image Pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC4 for 1 pixel and 4-pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_uv	Input UV plane of size (ROWS/2, COLS/2).
_y_image	Output Y plane of size (ROWS, COLS).
_u_image	Output U plane of size (ROWS/4, COLS).
_v_image	Output V plane of size (ROWS/4, COLS).

Resource Utilization

The following table summarizes the resource utilization of NV21 to IYUV for different configurations, as generated in the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 103: xFnv212iyuv Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	1	1377	730	219
8pixel	150	0	1	1975	1012	279

Performance Estimate

The following table summarizes the performance of NV21 to IYUV for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 104: xFnv212iyuv Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9
8 pixel operation(150Mhz)	1.7

NV21 to RGBA

The `xFnv212rgba` function converts a NV21 image format to a 4-channel RGBA image. The inputs to the function are separate Y and VU planes. NV21 holds sub sampled data, Yplane is sampled at unit rate and 1 U and 1V value each for every 2x2 Yvalues. To generate the RGBA data, each U and V value is duplicated (2x2) times.

API Syntax

```
template<int SRC_T, int UV_T, int DST_T, int ROWS, int COLS, int NPC=1>
void xFnv212rgba(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y, xF::Mat<UV_T,
ROWS/2, COLS/2, NPC> & src_uv,xF::Mat<DST_T, ROWS, COLS, NPC> & _dst0)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 105: xFnv212rgba Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Input pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
DST_T	Output pixel type. Only 8-bit, unsigned, 4-channel is supported (XF_8UC4).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_uv	Input UV plane of size (ROWS/2, COLS/2).
_dst0	Output RGBA image of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of NV21 to RGBA for different configurations, as generated in the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 106: xFnv212rgba Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	2	5	1170	673	183

Performance Estimate

The following table summarizes the performance of NV12 to RGBA for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 107: xFnv212rgba Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	6.9

NV21 to YUV4

The `xFnv212yuv4` function converts an image in the NV21 format to a YUV444 format. The function outputs separate U and V planes. Y plane is same for both formats. The UV planes are duplicated 2x2 times to represent one U plane and V plane of YUV444 format.

API Syntax

```
template<int SRC_T, int UV_T, int ROWS, int COLS, int NPC=1,int NPC_UV=1>
void xFnv212yuv4(xF::Mat<SRC_T, ROWS, COLS, NPC> & src_y, xF::Mat<UV_T,
ROWS/2, COLS/2, NPC_UV> & src_uv, xF::Mat<SRC_T, ROWS, COLS, NPC> &
_y_image, xF::Mat<SRC_T, ROWS, COLS, NPC> & _u_image, xF::Mat<SRC_T,
ROWS, COLS, NPC> & _v_image)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 108: xFnv212yuv4 Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1).
UV_T	Input pixel type. Only 8-bit, unsigned, 2-channel is supported (XF_8UC2).
ROWS	Maximum height of input and output image (must be a multiple of 8).
COLS	Maximum width of input and output image (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
NPC_UV	Number of UV image Pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC4 for 1 pixel and 4-pixel operations respectively.
src_y	Input Y plane of size (ROWS, COLS).
src_uv	Input UV plane of size (ROWS/2, COLS/2).
_y_image	Output Y plane of size (ROWS, COLS).
_u_image	Output U plane of size (ROWS, COLS).
_v_image	Output V plane of size (ROWS, COLS).

Resource Utilization

The following table summarizes the resource utilization of NV21 to YUV4 for different configurations, as generated in the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a HD (1080x1920) image.

Table 109: xFnv212yuv4 Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	300	0	0	1383	817	233
8pixel	150	0	0	1887	1087	287

Performance Estimate

The following table summarizes the performance of NV21 to YUV4 for different configurations, as generated using the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 110: xFnv212yuv4 Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300Mhz)	13.8
8 pixel operation(150Mhz)	3.5

Custom Convolution

The `xFfilter2D` function performs convolution over an image using a user-defined kernel.

Convolution is a mathematical operation on two functions *f* and *g*, producing a third function, The third function is typically viewed as a modified version of one of the original functions, that gives the area overlap between the two functions to an extent that one of the original functions is translated.

The filter can be unity gain filter or a non-unity gain filter. The filter must be of type AU_16SP. If the co-efficients are floating point, it must be converted into the Qm.n and provided as the input as well as the shift parameter has to be set with the 'n' value. Else, if the input is not of floating point, the filter is provided directly and the shift parameter is set to zero.

API Syntax

```
template<int BORDER_TYPE,int FILTER_WIDTH,int FILTER_HEIGHT, int
SRC_T,int DST_T, int ROWS, int COLS,int NPC=1>
void xFfilter2D(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat,xF::Mat<DST_T,
ROWS, COLS, NPC> & _dst_mat,short int
filter[FILTER_HEIGHT*FILTER_WIDTH],unsigned char _shift)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 111: xFfilter2D Function Parameter Descriptions

Parameter	Description
BORDER_TYPE	Border Type supported is XF_BORDER_CONSTANT
FILTER_HEIGHT	Number of rows in the input filter
FILTER_WIDTH	Number of columns in the input filter
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_T	Output pixel type.8-bit unsigned single channel (XF_8UC1) and 16-bit signed single channel (XF_16SC1) supported.
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_mat	Output image
filter	The input filter of any size, provided the dimensions should be an odd number. The filter co-efficients either a 16-bit value or a 16-bit fixed point equivalent value.
_shift	The filter must be of type XF_16SP. If the co-efficients are floating point, it must be converted into the Qm.n and provided as the input as well as the shift parameter has to be set with the 'n' value. Else, if the input is not of floating point, the filter is provided directly and the shift parameter is set to zero.

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 112: xFfilter2D Function Resource Utilization Summary

Operating Mode	Filter Size	Operating Frequency (MHz)	Utilization Estimate				
			BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	3x3	300	3	9	1701	1161	269
	5x5	300	5	25	3115	2144	524
8 pixel	3x3	150	6	72	2783	2768	638
	5x5	150	10	216	3020	4443	1007

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 113: xFfilter2D Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Filter Size	Latency Estimate Max (ms)
1 pixel	300	3x3	7
	300	5x5	7.1
8 pixel	150	3x3	1.86
	150	5x5	1.86

Dilate

During a dilation operation, the current pixel intensity is replaced by the maximum value of the intensity in a 3x3 neighborhood of the current pixel.

$$dst(x, y) = \max_{\substack{x-1 \le x' \le x+1 \\ y-1 \le y' \le y+1}} src(x', y')$$

API Syntax

```
template<int BORDER_TYPE, int SRC_T, int ROWS, int COLS,int NPC=1>
void xFdilate(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat, xF::Mat<SRC_T,
ROWS, COLS, NPC> & _dst_mat)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 114: xFdilate Function Parameter Descriptions

Parameter	Description
BORDER_TYPE	Border Type supported is XF_BORDER_CONSTANT
SRC_T	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)

Parameter	Description
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_mat	Output image

Resource Utilization

The following table summarizes the resource utilization of the Dilation function for 1 pixel operation and 8 pixel operation, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 115: xFdilate Function Resource Utilization Summary

Name	Resource Utilization	
	1 pixel per clock operation	**8 pixel per clock operation**
	300 MHz	**150 MHz**
BRAM_18K	3	6
DSP48E	0	0
FF	339	644
LUT	350	1325
CLB	81	245

Performance Estimate

The following table summarizes a performance estimate of the Dilation function for Normal Operation (1 pixel) and Resource Optimized (8 pixel) configurations, generated using Vivado HLS 2017.1 tool for Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 116: xFdilate Function Performance Estimate Summary

Operating Mode	Latency Estimate	
	Min (ms)	**Max (ms)**
1 pixel (300 MHz)	7.0	7.0
8 pixel (150 MHz)	1.87	1.87

Erode

The `xFerode` function finds the minimum pixel intensity in the 3x3 neighborhood of a pixel and replaces the pixel intensity with the minimum value.

$$dst(x, y) = \min_{\substack{x-1 \le x' \le x+1 \\ y-1 \le y' \le y+1}} src(x', y')$$

API Syntax

```
template<int BORDER_TYPE, int SRC_T, int ROWS, int COLS,int NPC=1>
void xFerode(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat, xF::Mat<SRC_T,
ROWS, COLS, NPC> & _dst_mat)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 117: xFerode Function Parameter Descriptions

Parameter	Description
BORDER_TYPE	Border type supported is XF_BORDER_CONSTANT
SRC_T	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_mat	Output image

Resource Utilization

The following table summarizes the resource utilization of the Erosion function generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 118: xFerode Function Resource Utilization Summary

Name	Resource Utilization	
	1 pixel per clock operation	**8 pixel per clock operation**
	300 MHz	**150 MHz**
BRAM_18K	3	6
DSP48E	0	0
FF	342	653
LUT	351	1316
CLB	79	230

Performance Estimate

The following table summarizes a performance estimate of the Erosion function for Normal Operation (1 pixel) and Resource Optimized (8 pixel) configurations, generated using Vivado HLS 2017.1 tool for Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 119: xFerode Function Performance Estimate Summary

Operating Mode	Latency Estimate	
	Min (ms)	Max (ms)
1 pixel (300 MHz)	7.0	7.0
8 pixel (150 MHz)	1.85	1.85

FAST Corner Detection

Features from accelerated segment test (FAST) is a corner detection algorithm, that is faster than most of the other feature detectors.

The `xFFAST` function picks up a pixel in the image and compares the intensity of 16 pixels in its neighborhood on a circle, called the Bresenham's circle. If the intensity of 9 contiguous pixels is found to be either more than or less than that of the candidate pixel by a given threshold, then the pixel is declared as a corner. Once the corners are detected, the non-maximal suppression is applied to remove the weaker corners.

This function can be used for both still images and videos. Initially, you have to specify the maximum number of corners. The total number of corners returned will never exceed that maximum value. If the actual number of corners in the image is more than that specified , the threshold can be increased to reduce the number of corners.

API Syntax

```
template<int NMS,int MAXPNTS,int SRC_T,int ROWS, int COLS,int NPC=1>
void xFFAST(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat,ap_uint<32>
list[MAXPNTS],unsigned char _threshold,uint32_t *nCorners)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 120: xFFAST Function Parameter Descriptions

Parameter	Description
NMS	If NMS == 1, non-maximum suppression is applied to detected corners (keypoints). The value should be 0 or 1.
MAXPNTS	Maximum number of corners that can be detected by the kernel.
SRC_T	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1)
ROWS	Maximum height of input image (must be a multiple of 8)
COLS	Maximum width of input image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
list	List of corners. The corners are packed in 32-bit format. The lower 16-bits provides the column index and the upper 16-bits indicates the row index.
_threshold	Threshold on the intensity difference between the center pixel and its neighbors. Usually it is taken around 20.
nCorners	The number of corners detected in the input image, which is the output of kernel.

Resource Utilization

The following table summarizes the resource utilization of the kernel for different configurations, generated using Vivado HLS 2017.1 for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image for 1024 corners with NMS.

Table 121: xFFAST Function Resource Utilization Summary

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	10	28
DSP48E	0	0
FF	2695	7310
LUT	3792	20956
CLB	769	3519

Performance Estimate

The following table summarizes the performance of kernel for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image for 1024 corners with non-maximum suppression (NMS).

Table 122: xFFAST Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Filter Size	Latency Estimate Max (ms)
1 pixel	300	3x3	7
8 pixel	150	3x3	1.86

Gaussian Filter

The `xFGaussianBlur` function applies Gaussian blur on the input image. Gaussian filtering is done by convolving each point in the input image with a Gaussian kernel.

$$G_0(x, y) = e^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}}$$

Where μ_x, μ_y are the mean values and σ_x, σ_y are the variances in x and y directions respectively. In the GaussianBlur function, values of μ_x, μ_y are considered as zeroes and the values of σ_x, σ_y are equal.

API Syntax

```
template<int FILTER_SIZE, int BORDER_TYPE, int SRC_T, int ROWS, int COLS,
int NPC =  1>
void xFGaussianBlur(xF::Mat<SRC_T, ROWS, COLS, NPC> & src, xF::Mat<SRC_T,
ROWS, COLS, NPC> & dst, float sigma)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 123: xFGaussianBlur Function Parameter Descriptions

Parameter	Description
FILTER_SIZE	Filter size. Filter size of 3 (XF_FILTER_3X3), 5 (XF_FILTER_5X5) and 7 (XF_FILTER_7X7) are supported.
BORDER_TYPE	Border type supported is XF_BORDER_CONSTANT
SRC_T	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible values are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.

Parameter	Description
src	Input image
dst	Output image
sigma	Standard deviation of Gaussian filter

Resource Utilization

The following table summarizes the resource utilization of the Gaussian Filter in different configurations, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to progress a grayscale HD (1080x1920) image.

Table 124: xFGaussianBlur Function Resource Utilization Summary

Operating Mode	Filter Size	Operating Frequency (MHz)	Utilization Estimate				
			BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	3x3	300	3	17	3641	2791	610
	5x5	300	5	27	4461	3544	764
	7x7	250	7	35	4770	4201	894
8 pixel	3x3	150	6	52	3939	3784	814
	5x5	150	10	111	5688	5639	1133
	7x7	150	14	175	7594	7278	1518

Performance Estimate

The following table summarizes a performance estimate of the Gaussian Filter in different configurations, as generated using Vivado HLS 2017.1 tool for Xilinx Xczu9eg-ffvb1156-1-i-es11 FPGA, to process a grayscale HD (1080x1920) image.

Table 125: xFGaussianBlur Function Performance Estimate Summary

Operating Mode	Filter Size	Latency Estimate
		Max Latency (ms)
1 pixel operation (300 MHz)	3x3	7.01
	5x5	7.03
	7x7	7.06
8 pixel operation (150 MHz)	3x3	1.6
	5x5	1.7
	7x7	1.74

Gradient Magnitude

The `xFmagnitude` function computes the magnitude for the images. The input images are x-gradient and y-gradient images of type 16S. The output image is of same type as the input image.

For L1NORM normalization, the magnitude computed image is the pixel-wise added image of absolute of x-gradient and y-gradient, as shown below:.

$$g = |g_x| + |g_y|$$

For L2NORM normalization, the magnitude computed image is as follows:

$$g = \sqrt{(g_x^2 + g_y^2)}$$

API Syntax

```
template< int NORM_TYPE ,int SRC_T,int DST_T, int ROWS, int COLS,int
NPC=1>
void xFmagnitude(xF::Mat<SRC_T, ROWS, COLS, NPC> &
_src_matx,xF::Mat<DST_T, ROWS, COLS, NPC> & _src_maty,xF::Mat<DST_T,
ROWS, COLS, NPC> & _dst_mat)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 126: xFmagnitude Function Parameter Descriptions

Parameter	Description
NORM_TYPE	Normalization type can be either L1 or L2 norm. Values are XF_L1NORM or XF_L2NORM
SRC_T	Input pixel type. Only 16-bit, signed, 1 channel is supported (XF_16SC1)
DST_T	Output pixel type. Only 16-bit, signed,1 channel is supported (XF_16SC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible values are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_matx	First input, x-gradient image.
_src_maty	Second input, y-gradient image.
_dst_mat	Output, magnitude computed image.

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image and for L2 normalization.

Table 127: xFmagnitude Function Resource Utilization Summary

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	0	0
DSP48E	2	16
FF	707	2002
LUT	774	3666
CLB	172	737

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image and for L2 normalization.

Table 128: xFmagnitude Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate
		Max (ms)
1 pixel	300	7.2
8 pixel	150	1.7

Gradient Phase

The `xFphase` function computes the polar angles of two images. The input images are x-gradient and y-gradient images of type 16S. The output image is of same type as the input image.

For radians:

$$angle(x,\ y) = atan2(g_y,\ g_x)$$

For degrees:

$$angle(x,\ y) = atan2(g_y,\ g_x) * \frac{180}{\pi}$$

API Syntax

```
template<int RET_TYPE ,int SRC_T,int DST_T, int ROWS, int COLS,int NPC=1 >
void xFphase(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_matx,xF::Mat<DST_T,
ROWS, COLS, NPC> & _src_maty,xF::Mat<DST_T, ROWS, COLS, NPC> & _dst_mat)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 129: xFphase Function Parameter Descriptions

Parameter	Description
RET_TYPE	Output format can be either in radians or degrees. Options are XF_RADIANS or XF_DEGREES. • If the XF_RADIANS option is selected, xFphase API will return result in Q4.12 format. The output range is (0, 2 pi) • If the XF_DEGREES option is selected, xFphaseAPI will return result in Q10.6 degrees and output range is (0, 360)
SRC_T	Input pixel type. Only 16-bit, signed, 1 channel is supported (XF_16SC1)
DST_T	Output pixel type. Only 16-bit, signed, 1 channel is supported (XF_16SC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_matx	First input, x-gradient image.
_src_maty	Second input, y-gradient image.
_dst_mat	Output, phase computed image.

1. If XF_RADIANS options is selected, xFphase API will return result in Q4.12 format. The output range is (0, 2pi)
2. If XF_DEGREES options is selected, xFphase API will return result in Q10.6 degrees and output range is (0, 360)

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es11 FPGA, to process a grayscale HD (1080x1920) image.

Table 130: xFphase Function Resource Utilization Summary

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	6	24
DSP48E	6	19
FF	873	2396
LUT	753	3895
CLB	185	832

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 131: xFphase Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate (ms)
1 pixel	300	7.2
8 pixel	150	1.7

Deviation from OpenCV

In xFphase implementation, the output is returned in a fixed point format. If XF_RADIANS option is selected, xFphase API will return result in Q4.12 format. The output range is (0, 2 pi). If XF_DEGREES option is selected, xFphase API will return result in Q10.6 degrees and output range is (0, 360);

Harris Corner Detection

In order to understand Harris Corner Detection, let us consider a grayscale image. Sweep a window `w(x,y)` (with displacements `u` in the x-direction and `v` in the y-direction), `I` calculates the variation of intensity `w(x,y)`.

$$E(u, v) = \sum w(x, y)[I(x+u, y+v) - I(x, y)]^2$$

Where:

- `w(x,y)` is the window position at (x,y)
- `I(x,y)` is the intensity at (x,y)
- `I(x+u,y+v)` is the intensity at the moved window `(x+u,y+v)`.

Since we are looking for windows with corners, we are looking for windows with a large variation in intensity. Hence, we have to maximize the equation above, specifically the term:

$$[I(x+u, y+v) - I(x, y)]^2$$

Using Taylor expansion:

$$E(u, v) = \sum [I(x, y) + uI_x + vI_y - I(x, y)]^2$$

Expanding the equation and cancelling `I(x,y)` with `-I(x,y)`:

$$E(u, v) = \sum u^2 I_x^2 + 2uvI_x I_y + v^2 I_y^2$$

The above equation can be expressed in a matrix form as:

$$E(u, v) = [u\ v]\left(\sum w(x, y)\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}\right)\begin{bmatrix} u \\ v \end{bmatrix}$$

So, our equation is now:

$$E(u, v) = [u\ v]M\begin{bmatrix} u \\ v \end{bmatrix}$$

A score is calculated for each window, to determine if it can possibly contain a corner:

$$R = det(M) - k(trace(M))^2$$

Where,

- $det(M) = \lambda_1 \lambda_2$
- $trace(M) = \lambda_1 + \lambda_2$

Non-Maximum Suppression:

In non-maximum suppression (NMS) if radius = 1, then the bounding box is 2*r+1 = 3.

In this case, consider a 3x3 neighborhood across the center pixel. If the center pixel is greater than the surrounding pixel, then it is considered a corner. The comparison is made with the surrounding pixels, which are within the radius.

Radius = 1

x-1, y-1	x-1, y	x-1, y+1
x, y-1	x, y	x, y+1
x+1, y-1	x+1, y	x+1, y+1

Threshold:

A threshold=442, 3109 and 566 is used for 3x3, 5x5, and 7x7 filters respectively. This threshold is verified over 40 sets of images. The threshold can be varied, based on the application.

API Syntax

```
template<int MAXCORNERS, int FILTER_SIZE, int BLOCK_WIDTH, int
NMS_RADIUS, int TYPE, int ROWS, int COLS, int NPC=1>
void xFCornerHarris ( xF::Mat<int TYPE, int ROWS, int COLS, int NPC>
_src,
ap_uint<32> points[MAXCORNERS], uint16_t threshold, uint16_t  k, uint32_t
* nCorners)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 132: xFCornerHarris Function Parameter Descriptions

Parameter	Description
MAXCORNERS	Maximum number of corners that can be detected by the kernel.
FILTER_SIZE	Size of the Sobel filter. 3, 5, and 7 supported.
BLOCK_WIDTH	Size of the box filter. 3, 5, and 7 supported.
NMS_RADIUS	Radius considered for non-maximum suppression. Values supported are 1 and 2.
TYPE	Input pixel type. Only 8-bit, unsigned, 1-channel is supported (XF_8UC1)
ROWS	Maximum height of input image (must be a multiple of 8)
COLS	Maximum width of input image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src	Input image
points	List of corners. The corners are packed in 32-bit format. Lower 16-bits provides the column index and upper 16-bits indicates the row index.
threshold	Threshold applied to the corner measure.
k	Harris detector parameter
nCorners	Output from kernel providing number of corners detected in the input image.

Resource Utilization

The following table summarizes the resource utilization of the Harris corner detection in different configurations, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image for 1024 corners.

The following table summarizes the resource utilization for Sobel Filter = 3, Box filter=3 and NMS_RADIUS =1

Table 133: Resource Utilization Summary - For Sobel Filter = 3, Box filter=3 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	33	74
DSP48E	13	83
FF	3254	9330
LUT	3522	13222
CLB	731	2568

The following table summarizes the resource utilization for Sobel Filter = 3, Box filter=5 and NMS_RADIUS =1

Table 134: Resource Utilization Summary - Sobel Filter = 3, Box filter=5 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	45	98
DSP48E	13	83
FF	5455	12459
LUT	5675	24594
CLB	1132	4498

The following table summarizes the resource utilization for Sobel Filter = 3, Box filter=7 and NMS_RADIUS =1

Table 135: Resource Utilization Summary - Sobel Filter = 3, Box filter=7 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	57	122
DSP48E	13	83
FF	8783	16593
LUT	9157	39813
CLB	1757	6809

The following table summarizes the resource utilization for Sobel Filter = 5, Box filter=3 and NMS_RADIUS =1

Table 136: Resource Utilization Summary - Sobel Filter = 5, Box filter=3 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	200 MHz
BRAM_18K	35	78

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	200 MHz
DSP48E	13	83
FF	4656	11659
LUT	4681	17394
CLB	1005	3277

The following table summarizes the resource utilization for Sobel Filter = 5, Box filter=5 and NMS_RADIUS =1

Table 137: Resource Utilization Summary - Sobel Filter = 5, Box filter=5 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	47	102
DSP48E	13	83
FF	6019	14776
LUT	6337	28795
CLB	1353	5102

The following table summarizes the resource utilization for Sobel Filter = 5, Box filter=7 and NMS_RADIUS =1

Table 138: Resource Utilization Summary - Sobel Filter = 5, Box filter=7 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	59	126
DSP48E	13	83
FF	9388	18913
LUT	9414	43070
CLB	1947	7508

The following table summarizes the resource utilization for Sobel Filter = 7, Box filter=3 and NMS_RADIUS =1

Table 139: Resource Utilization Summary - Sobel Filter = 7, Box filter=3 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	37	82
DSP48E	14	91
FF	6002	13880

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
LUT	6337	25573
CLB	1327	4868

The following table summarizes the resource utilization for Sobel Filter = 7, Box filter=5 and NMS_RADIUS =1

Table 140: Resource Utilization Summary - Sobel Filter = 7, Box filter=5 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	49	106
DSP48E	14	91
FF	7410	17049
LUT	8076	36509
CLB	1627	6518

The following table summarizes the resource utilization for Sobel Filter = 7, Box filter=7 and NMS_RADIUS =1

Table 141: Resource Utilization Summary - Sobel Filter = 7, Box filter=7 and NMS_RADIUS =1

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	61	130
DSP48E	14	91
FF	10714	21137
LUT	11500	51331
CLB	2261	8863

The following table summarizes the resource utilization for Sobel Filter = 3, Box filter=3 and NMS_RADIUS =2

Table 142: Resource Utilization Summary - Sobel Filter = 3, Box filter=3 and NMS_RADIUS =2

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	41	90
DSP48E	13	83
FF	5519	10714
LUT	5094	16930
CLB	1076	3127

Resource utilization: For Sobel Filter = 3, Box filter=5 and NMS_RADIUS =2

Table 143: Resource Utilization Summary

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	53	114
DSP48E	13	83
FF	6798	13844
LUT	6866	28286
CLB	1383	4965

The following table summarizes the resource utilization for Sobel Filter = 3, Box filter=7 and NMS_RADIUS =2

Table 144: Resource Utilization Summary - Sobel Filter = 3, Box filter=7 and NMS_RADIUS =2

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	65	138
DSP48E	13	83
FF	10137	17977
LUT	10366	43589
CLB	1940	7440

The following table summarizes the resource utilization for Sobel Filter = 5, Box filter=3 and NMS_RADIUS =2

Table 145: Resource Utilization Summary - Sobel Filter = 5, Box filter=3 and NMS_RADIUS =2

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	43	94
DSP48E	13	83
FF	5957	12930
LUT	5987	21187
CLB	1244	3922

The following table summarizes the resource utilization for Sobel Filter = 5, Box filter=5 and NMS_RADIUS =2

Table 146: Resource Utilization Summary - Sobel Filter = 5, Box filter=5 and NMS_RADIUS =2

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	55	118
DSP48E	13	83
FF	5442	16053
LUT	6561	32377
CLB	1374	5871

The following table summarizes the resource utilization for Sobel Filter = 5, Box filter=7 and NMS_RADIUS =2

Table 147: Resource Utilization Summary - Sobel Filter = 5, Box filter=7 and NMS_RADIUS =2

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	67	142
DSP48E	13	83
FF	10673	20190
LUT	10793	46785
CLB	2260	8013

The following table summarizes the resource utilization for Sobel Filter = 7, Box filter=3 and NMS_RADIUS =2

Table 148: Resource Utilization Summary - Sobel Filter = 7, Box filter=3 and NMS_RADIUS =2

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	45	98
DSP48E	14	91
FF	7341	15161
LUT	7631	29185
CLB	1557	5425

The following table summarizes the resource utilization for Sobel Filter = 7, Box filter=5 and NMS_RADIUS =2

Table 149: Resource Utilization Summary - Sobel Filter = 7, Box filter=5 and NMS_RADIUS =2

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	57	122

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
DSP48E	14	91
FF	8763	18330
LUT	9368	40116
CLB	1857	7362

The following table summarizes the resource utilization for Sobel Filter = 7, Box filter=7 and NMS_RADIUS =2

Table 150: Resource Utilization Summary - Sobel Filter = 7, Box filter=7 and NMS_RADIUS =2

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	69	146
DSP48E	14	91
FF	12078	22414
LUT	12831	54652
CLB	2499	9628

Performance Estimate

The following table summarizes a performance estimate of the Harris corner detection in different configurations, as generated using Vivado HLS 2017.1 tool for Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 151: xFCornerHarris Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Configuration			Latency Estimate
		Sobel	Box	NMS Radius	Latency(In ms)
1 pixel	300 MHz	3	3	1	7
1 pixel	300 MHz	3	5	1	7.1
1 pixel	300 MHz	3	7	1	7.1
1 pixel	300 MHz	5	3	1	7.2
1 pixel	300 MHz	5	5	1	7.2
1 pixel	300 MHz	5	7	1	7.2
1 pixel	300 MHz	7	3	1	7.22
1 pixel	300 MHz	7	5	1	7.22
1 pixel	300 MHz	7	7	1	7.22
8 pixel	150 MHz	3	3	1	1.7
8 pixel	150 MHz	3	5	1	1.7
8 pixel	150 MHz	3	7	1	1.7
8 pixel	150 MHz	5	3	1	1.71
8 pixel	150 MHz	5	5	1	1.71
8 pixel	150 MHz	5	7	1	1.71
8 pixel	150 MHz	7	3	1	1.8
8 pixel	150 MHz	7	5	1	1.8
8 pixel	150 MHz	7	7	1	1.8
1 pixel	300 MHz	3	3	2	7.1
1 pixel	300 MHz	3	5	2	7.1
1 pixel	300 MHz	3	7	2	7.1
1 pixel	300 MHz	5	3	2	7.21
1 pixel	300 MHz	5	5	2	7.21
1 pixel	300 MHz	5	7	2	7.21
1 pixel	300 MHz	7	3	2	7.22
1 pixel	300 MHz	7	5	2	7.22
1 pixel	300 MHz	7	7	2	7.22
8 pixel	150 MHz	3	3	2	1.8
8 pixel	150 MHz	3	5	2	1.8
8 pixel	150 MHz	3	7	2	1.8
8 pixel	150 MHz	5	3	2	1.81
8 pixel	150 MHz	5	5	2	1.81
8 pixel	150 MHz	5	7	2	1.81
8 pixel	150 MHz	7	3	2	1.9
8 pixel	150 MHz	7	5	2	1.91
8 pixel	150 MHz	7	7	2	1.92

Deviation from OpenCV

In xfOpenCV thresholding and NMS are included, but in OpenCV they are not included. In xfOpenCV, all the blocks are implemented in fixed point. Whereas,in OpenCV, all the blocks are implemented in floating point.

Histogram Computation

The `calcHist` function computes the histogram of given input image.

$$H[src(x, y)] = H[src(x, y)] + 1$$

Where, H is the array of 256 elements.

API Syntax

```
template<int SRC_T,int ROWS, int COLS,int NPC=1>
void calcHist(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, uint32_t *histogram)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 152: calcHist Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle
_src	Input image
histogram	Output array of 256 elements

Resource Utilization

The following table summarizes the resource utilization of the calcHist function for Normal Operation (1 pixel) and Resource Optimized (8 pixel) configurations, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA at 300 MHz for 1 pixel case and at 150 MHz for 8 pixel mode.

Table 153: calcHist Function Resource Utilization Summary

Name	Resource Utilization	
	Normal Operation (1 pixel)	Resource Optimized (8 pixel)
BRAM_18K	2	16
DSP48E	0	0
FF	196	274
LUT	240	912
CLB	57	231

Performance Estimate

The following table summarizes a performance estimate of the calcHist function for Normal Operation (1 pixel) and Resource Optimized (8 pixel) configurations, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA at 300 MHz for 1 pixel and 150 MHz for 8 pixel mode.

Table 154: calcHist Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max (ms)
Normal Operation	6.9
Resource Optimized	1.7

Histogram Equalization

The `equalizeHist` function performs histogram equalization on input image or video. It improves the contrast in the image, to stretch out the intensity range. This function maps one distribution (histogram) to another distribution (a wider and more uniform distribution of intensity values), so the intensities are spread over the whole range.

For histogram H[i], the cumulative distribution H'[i] is given as:

$$H'[i] = \sum_{0 \le j < i} H[j]$$

The intensities in the equalized image are computed as:

$$dst(x, y) = H'(src(x, y))$$

API Syntax

```
template<int SRC_T, int ROWS, int COLS, int NPC = 1>
void xFequalizeHist(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src,xF::Mat<SRC_T,
ROWS, COLS, NPC> & _src1,xF::Mat<SRC_T, ROWS, COLS, NPC> & _dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 155: xFequalizeHist Function Parameter Descriptions

Parameter	Description
SRC_T	Input and output pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle
_src	Input image
_src1	Input image
_dst	Output image

Resource Utilization

The following table summarizes the resource utilization of the equalizeHist function for Normal Operation (1 pixel) and Resource Optimized (8 pixel) configurations, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA at 300 MHz for 1 pixel and 150 MHz for 8 pixel mode.

Table 156: xFequalizeHist Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	4	5	3492	1807	666
8 pixel	150	25	5	3526	2645	835

Performance Estimate

The following table summarizes a performance estimate of the equalizeHist function for Normal Operation (1 pixel) and Resource Optimized (8 pixel) configurations, generated using Vivado HLS 2017.1version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA at 300 MHz for 1 pixel and 150 MHz for 8 pixel mode.

Table 157: xFequalizeHist Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max (ms)
1 pixel per clock operation	13.8
8 pixel per clock operation	3.4

HOG

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision for the purpose of object detection. The feature descriptors produced from this approach is widely used in the pedestrian detection.

The technique counts the occurrences of gradient orientation in localized portions of an image. HOG is computed over a dense grid of uniformly spaced cells and normalized over overlapping blocks, for improved accuracy. The concept behind HOG is that the object appearance and shape within an image can be described by the distribution of intensity gradients or edge direction.

Both RGB and gray inputs are accepted to the function. In the RGB mode, gradients are computed for each plane separately, but the one with the higher magnitude is selected. With the configurations provided, the window dimensions are 64x128, block dimensions are 16x16.

API Syntax

```
template<int WIN_HEIGHT, int WIN_WIDTH, int WIN_STRIDE, int BLOCK_HEIGHT,
int BLOCK_WIDTH, int CELL_HEIGHT, int CELL_WIDTH, int NOB, int ROWS, int
COLS, int SRC_T, int DST_T, int DESC_SIZE, int NPC = XF_NPPC1, int
IMG_COLOR, int OUTPUT_VARIANT>
void xFHOGDescriptor(xF::Mat<SRC_T, ROWS, COLS, NPC> &_in_mat,
xF::Mat<DST_T, 1, DESC_SIZE, NPC> &_desc_mat
```

Parameter Descriptions

The following table describes the template parameters.

Table 158: xFHOGDescriptor Template Parameter Descriptions

PARAMETERS	DESCRIPTION
WIN_HEIGHT	The number of pixel rows in the window. It is fixed at 128.
WIN_WIDTH	The number of pixel cols in the window. It is fixed at 64.
WIN_STIRDE	The pixel stride between two adjacent windows. It is fixed at 8.
BLOCK_HEIGHT	Height of the block. It is fixed at 16.
BLOCK_WIDTH	Width of the block. It is fixed at 16.
CELL_HEIGHT	Number of rows in a cell. It is fixed at 8.
CELL_WIDTH	Number of cols in a cell. It is fixed at 8.
NOB	Number of histogram bins for a cell. It is fixed at 9
ROWS	Number of rows in the image being processed. (Should be a multiple of 8)
COLS	Number of columns in the image being processed. (Should be a multiple of 8)
SRC_T	Input pixel type. Must be either XF_8UC1 or XF_8UC4, for gray and color respectively.
DST_T	Ouput descriptor type. Must be XF_32UC1.
DESC_SIZE	The size of the output descriptor.
NPC	Number of pixels to be processed per cycle; this function supports only XF_NPPC1 or 1 pixel per cycle operations.
IMG_COLOR	The type of the image, set as either XF_GRAY or XF_RGB
OUTPUT_VARIENT	Must be either XF_HOG_RB or XF_HOG_NRB

The following table describes the function parameters.

Table 159: xFHOGDescriptor Function Parameter Descriptions

PARAMETERS	DESCRIPTION
_in_mat	Input image, of xF::Mat type
_desc_mat	Output descriptors, of xF::Mat type

Where,

- NO is normal operation (single pixel processing)
- RB is repetitive blocks (descriptor data are written window wise)
- NRB is non-repetitive blocks (descriptor data are written block wise, in order to reduce the number of writes).

NOTE: *In the RB mode, the block data is written to the memory taking the overlap windows into consideration. In the NRB mode, the block data is written directly to the output stream without consideration of the window overlap. In the host side, the overlap must be taken care.*

Resource Utilization

The following table shows the resource utilization of `xFHOGDescriptor` function for normal operation (1 pixel) mode as generated in Vivado HLS 2017.1 version tool for the part Xilinx Xczu9eg-ffvb1156-1-i-es1 at 300 MHz to process an image of 1920x1080 resolution.

Table 160: xFHOGDescriptor Function Resource Utilization Summary

Resource	Utilization (at 300 MHz) of 1 pixel operation			
	NRB		RB	
	Gray	RGB	Gray	RGB
BRAM_18K	43	49	171	177
DSP48E	34	46	36	48
FF	15365	15823	15205	15663
LUT	12868	13267	13443	13848

Performance Estimate

The following table shows the performance estimates of xFHOGDescriptor() function for different configurations as generated in Vivado HLS 2017.1 version tool for the part Xilinx Xczu9eg-ffvb1156-1-i-es1 to process an image of 1920x1080p resolution.

Table 161: xFHOGDescriptor Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate	
		Min (ms)	Max (ms)
NRB-Gray	300	6.98	8.83
NRB-RGBA	300	6.98	8.83
RB-Gray	300	176.81	177
RB-RGBA	300	176.81	177

Deviations from OpenCV

Listed below are the deviations from the OpenCV:

1. Border care

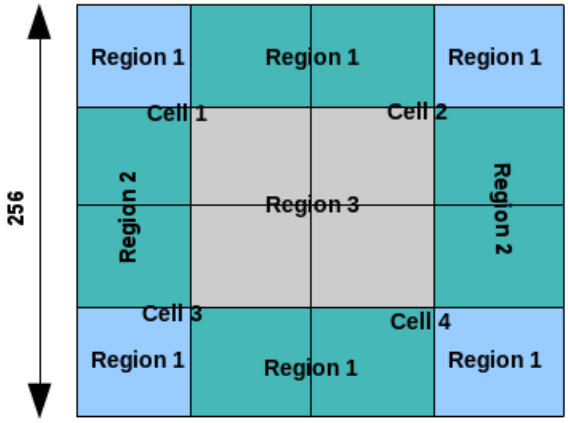
   The border care that OpenCV has taken in the gradient computation is BORDER_REFLECT_101, in which the border padding will be the neighboring pixels' reflection. Whereas, in the Xilinx implementation, BORDER_CONSTANT (zero padding) was used for the border care.

2. Gaussian weighing

   The Gaussian weights are multiplied on the pixels over the block, that is a block has 256 pixels, and each position of the block are multiplied with its corresponding Gaussian weights. Whereas, in the HLS implementation, gaussian weighing was not performed.

3. Cell-wise interpolation

   The magnitude values of the pixels are distributed across different cells in the blocks but on the corresponding bins.



   Pixels in the region 1 belong only to its corresponding cells, but the pixels in region 2 and 3 are interpolated to the adjacent 2 cells and 4 cells respectively. This operation was not performed in the HLS implementation.

4. Output handling

   The output of the OpenCV will be in the column major form. In the HLS implementation, output will be in the row major form. Also, the feature vector will be in the fixed point type Q0.16 in the HLS implementation, while in the OpenCV it will be in floating point.

Limitations

1. The configurations are limited to Dalal's implementation. [1]
2. Image height and image width must be a multiple of cell height and cell width respectively.

1. N. Dalal, B. Triggs: *Histograms of oriented gradients for human detection,* IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

Pyramid Up

The `xFPyrUp` function is an image up-sampling algorithm. It first inserts zero rows and zero columns after every input row and column making up to the size of the output image. The output image size is always $(2*rows \times 2*columns)$ .The zero padded image is then smoothened using Gaussian image filter. Gaussian filter for the pyramid-up function uses a fixed filter kernel as given below:

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

However, to make up for the pixel intensity that is reduced due to zero padding, each output pixel is multiplied by 4.

API Syntax

```
template<int TYPE, int ROWS, int COLS, int NPC>
void xFPyrUp (xF::Mat<TYPE, ROWS, COLS, NPC> & _src, xF::Mat<TYPE, ROWS,
COLS, NPC> & _dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 162: xFPyrUp Function Parameter Descriptions

Parameter	Description
TYPE	Pixel type. XF_8UC1 is the only supported pixel type.
ROWS	Maximum Height or number of output rows to build the hardware for this kernel
COLS	Maximum Width or number of output columns to build the hardware for this kernel
NPC	Number of pixels to process per cycle. Currently, the kernel supports only 1 pixel per cycle processing (XF_NPPC1).
_src	Input image stream
_dst	Output image stream

Resource Utilization

The following table summarizes the resource utilization of xFPyrUp for 1 pixel per cycle implementation, for a maximum input image size of 1920x1080 pixels. The results are after synthesis in Vivado HLS 2017.1 for the Xilinx xczu9eg-ffvb1156-1-i-es1 FPGA at 300 MHz.

Table 163: xFPyrUp Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate			
		LUTs	FFs	DSPs	BRAMs
1 Pixel per Clock Cycle	300	1124	1199	0	10

Performance Estimate

The following table summarizes performance estimates of xFPyrUp function on Vivado HLS 2017.1 for the Xilinx xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 164: xFPyrUp Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Input Image Size	Latency Estimate
			Max (ms)
1pixel	300	1920x1080	27.82

Pyramid Down

The `xFPyrDown` function is an image down-sampling algorithm which smoothens the image before down-scaling it. The image is smoothened using a Gaussian filter with the following kernel:

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Down-scaling is performed by dropping pixels in the even rows and the even columns. The resulting image size is $\left(\frac{rows+1}{2} \quad \frac{columns+1}{2}\right)$

API Syntax

```
template<int TYPE, int ROWS, int COLS, int NPC>
void xFPyrDown (xF::Mat<TYPE, ROWS, COLS, NPC> & _src, xF::Mat<TYPE,
ROWS, COLS, NPC> & _dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 165: xFPyrDown Function Parameter Descriptions

Parameter	Description
TYPE	Pixel type. XF_8UC1 is the only supported pixel type.
ROWS	Maximum Height or number of input rows to build the hardware for this kernel
COLS	Maximum Width or number of input columns to build the hardware for this kernel
NPC	Number of pixels to process per cycle. Currently, the kernel supports only 1 pixel per cycle processing (XF_NPPC1).
_src	Input image stream
_dst	Output image stream

Resource Utilization

The following table summarizes the resource utilization of xFPyrDown for 1 pixel per cycle implementation, for a maximum input image size of 1920x1080 pixels. The results are after synthesis in Vivado HLS 2017.1 for the Xilinx xczu9eg-ffvb1156-1-i-es1 FPGA at 300 MHz.

Table 166: xFPyrDown Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate			
		LUTs	FFs	DSPs	BRAMs
1 Pixel	300	1171	1238	1	5

Performance Estimate

The following table summarizes performance estimates of xFPyrDown function in Vivado HLS 2017.1 for the Xilinx xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 167: xFPyrDown Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Input Image Size	Latency Estimate Max (ms)
1 pixel	300	1920x1080	6.99

Integral Image

The `xFIntegralImage` function omputes an integral image of the input. Each output pixel is the sum of all pixels above and to the left of itself.

$$dst(x, y) = sum(x, y) = sum(x, y) + sum(x - 1, y) + sum(x, y - 1) - sum(x - 1, y - 1)$$

API Syntax

```
template<int SRC_TYPE,int DST_TYPE, int ROWS, int COLS, int NPC=1>
void xFIntegralImage(xF::Mat<SRC_TYPE, ROWS, COLS, NPC> & _src_mat,
xF::Mat<DST_TYPE, ROWS, COLS, NPC> & _dst_mat)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 168: xFIntegralImage Function Parameter Descriptions

Parameter	Description
SRC_TYPE	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_TYPE	Output pixel type. Only 32-bit,unsigned,1 channel is supported(XF_32UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; this function supports only XF_NPPC1 or 1 pixel per cycle operations.
_src_mat	Input image
_dst_mat	Output image

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 169: xFIntegralImage Function Resource Utilization Summary

Name	Resource Utilization
	1 pixel
	300 MHz
BRAM_18K	4
DSP48E	0
FF	613
LUT	378
CLB	102

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 170: xFIntegralImage Function Performance Estimate Summary

Operating Mode	Latency Estimate	
	Operating Frequency (MHz)	Latency(in ms)
1pixel	300	7.2

Dense Pyramidal LK Optical Flow

Optical flow is the pattern of apparent motion of image objects between two consecutive frames, caused by the movement of object or camera. It is a 2D vector field, where each vector is a displacement vector showing the movement of points from first frame to second.

Optical Flow works on the following assumptions:

- Pixel intensities of an object do not have too many variations in consecutive frames
- Neighboring pixels have similar motion

Consider a pixel I(x, y, t) in first frame. (Note that a new dimension, time, is added here. When working with images only, there is no need of time). The pixel moves by distance (dx, dy) in the next frame taken after time dt. Thus, since those pixels are the same and the intensity does not change, the following is true:

$$I(x,\ y,\ t) = I(x + dx,\ y + dy,\ t + dt)$$

Taking the Taylor series approximation on the right-hand side, removing common terms, and dividing by dt gives the following equation:

$$f_x u + f_y v + f_t = 0$$

Where $f_x = \frac{\delta f}{\delta x}$, $f_y = \frac{\delta f}{\delta x}$, $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$.

The above equation is called the Optical Flow equation, where, f_x and f_y are the image gradientsand f_t is the gradient along time. However, (u, v) is unknown. It is not possible to solve this equation with two unknown variables. Thus, several methods are provided to solve this problem. One method is Lucas-Kanade. Previously it was assumed that all neighboring pixels have similar motion. The Lucas-Kanade method takes a patch around the point, whose size can be defined through the 'WINDOW_SIZE' template parameter. Thus, all the points in that patch have the same motion. It is possible to find (f_x, f_y, f_t) for these points. Thus, the problem now becomes solving 'WINDOW_SIZE * WINDOW_SIZE' equations with two unknown variables,which is over-determined. A better solution is obtained with the "least square fit" method. Below is the final solution, which is a problem with two equations and two unknowns:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum f_{x_i}^2 & \sum f_{x_i} f_{y_i} \\ \sum f_{x_i} f_{y_i} & \sum f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum f_{x_i} f_{t_i} \\ -\sum f_{y_i} f_{t_i} \end{bmatrix}$$

This solution fails when a large motion is involved and so pyramids are used. Going up in the pyramid, small motions are removed and large motions become small motions and so by applying Lucas-Kanade, the optical flow along with the scale is obtained.

API Syntax

```
template< int NUM_PYR_LEVELS, int NUM_LINES, int WINSIZE, int FLOW_WIDTH,
int FLOW_INT, int TYPE, int ROWS, int COLS, int NPC>
void xFDensePyrOpticalFlow(
xF::Mat<TYPE,ROWS,COLS,NPC> & _current_img,
xF::Mat<TYPE,ROWS,COLS,NPC> & _next_image,
xF::Mat<XF_32UC1,ROWS,COLS,NPC> & _streamFlowin,
xF::Mat<XF_32UC1,ROWS,COLS,NPC> & _streamFlowout,
const int level, const unsigned char scale_up_flag, float scale_in )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 171: xFDensePyrOpticalFlow Function Parameter Descriptions

Parameter	Description
NUM_PYR_LEVELS	Number of Image Pyramid levels used for the optical flow computation
NUM_LINES	Number of lines to buffer for the remap algorithm – used to find the temporal gradient
WINSIZE	Window Size over which Optical Flow is computed
FLOW_WIDTH, FLOW_INT	Data width and number of integer bits to define the signed flow vector data type. Integer bit includes the signed bit. The default type is 16-bit signed word with 10 integer bits and 6 decimal bits.
TYPE	Pixel type of the input image. XF_8UC1 is only the supported value.
ROWS	Maximum Height or number of rows to build the hardware for this kernel
COLS	Maximum Width or number of columns to build the hardware for this kernel
NPC	Number of pixels the hardware kernel must process per clock cycle. Only XF_NPPC1, 1 pixel per cycle, is supported.
_curr_img	First input image stream
_next_img	Second input image to which the optical flow is computed with respect to the first image
_streamFlowin	32-bit Packed U and V flow vectors input for optical flow. The bits from 31-16 represent the flow vector U while the bits from 15-0 represent the flow vector V.
_streamFlowout	32-bit Packed U and V flow vectors output after optical flow computation. The bits from 31-16 represent the flow vector U while the bits from 15-0 represent the flow vector V.
level	Image pyramid level at which the algorithm is currently computing the optical flow.
scale_up_flag	Flag to enable the scaling-up of the flow vectors. This flag is set at the host when switching from one image pyramid level to the other.
scale_in	Floating point scale up factor for the scaling-up the flow vectors. The value is (previous_rows-1)/(current_rows-1). This is not 1 when switching from one image pyramid level to the other.

Resource Utilization

The following table summarizes the resource utilization of xFDensePyrOpticalFlow for 1 pixel per cycle implementation, with the optical flow computed for a window size of 11 over an image size of 1920x1080 pixels. The results are after implementation in Vivado HLS 2017.1 for the Xilinx xczu9eg-ffvb1156-2L-e FPGA at 300 MHz.

Table 172: xFDensePyrOpticalFlow Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate			
		LUTs	FFs	DSPs	BRAMs
1 Pixel	300	32231	16596	52	215

Performance Estimate

The following table summarizes performance figures on hardware for the xFDensePyrOpticalFlow function for 5 iterations over 5 pyramid levels scaled down by a factor of two at each level. This has been tested on the zcu102 evaluation board.

Table 173: xFDensePyrOpticalFlow Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Image Size	Latency Estimate
			Max (ms)
1 pixel	300	1920x1080	49.7
1 pixel	300	1280x720	22.9
1 pixel	300	1226x370	12.02

Dense Non-Pyramidal LK Optical Flow

Optical flow is the pattern of apparent motion of image objects between two consecutive frames, caused by the movement of object or camera. It is a 2D vector field, where each vector is a displacement vector showing the movement of points from first frame to second.

Optical Flow works on the following assumptions:

- Pixel intensities of an object do not have too many variations in consecutive frames
- Neighboring pixels have similar motion

Consider a pixel I(x, y, t) in first frame. (Note that a new dimension, time, is added here. When working with images only, there is no need of time). The pixel moves by distance (dx, dy) in the next frame taken after time dt. Thus, since those pixels are the same and the intensity does not change, the following is true:

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Taking the Taylor series approximation on the right-hand side, removing common terms, and dividing by dt gives the following equation:

$$f_x u + f_y v + f_t = 0$$

Where $f_x = \frac{\delta f}{\delta x}$, $f_y = \frac{\delta f}{\delta x}$, $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$.

The above equation is called the Optical Flow equation, where, f_x and f_y are the image gradientsand f_t is the gradient along time. However, (u, v) is unknown. It is not possible to solve this equation with two unknown variables. Thus, several methods are provided to solve this problem. One method is Lucas-Kanade. Previously it was assumed that all neighboring pixels have similar motion. The Lucas-Kanade method takes a patch around the point, whose size can be defined through the 'WINDOW_SIZE' template parameter. Thus, all the points in that patch have the same motion. It is possible to find (f_x, f_y, f_t) for these points. Thus, the problem now becomes solving 'WINDOW_SIZE * WINDOW_SIZE' equations with two unknown variables,which is over-determined. A better solution is obtained with the "least square fit" method. Below is the final solution, which is a problem with two equations and two unknowns:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum f_{x_i}^2 & \sum f_{x_i} f_{y_i} \\ \sum f_{x_i} f_{y_i} & \sum f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum f_{x_i} f_{t_i} \\ -\sum f_{y_i} f_{t_i} \end{bmatrix}$$

API Syntax

```
template<int TYPE, int ROWS, int COLS, int NPC, int WINDOW_SIZE>
void xFDenseNonPyrLKOpticalFlow (xF::Mat<TYPE, ROWS, COLS, NPC> & frame0,
xF::Mat<TYPE, ROWS, COLS, NPC> & frame1, xF::Mat<XF_32FC1, ROWS, COLS,
NPC> & flowx, xF::Mat<XF_32FC1, ROWS, COLS, NPC> & flowy)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 174: xFDenseNonPyrLKOpticalFlow Function Parameter Descriptions

Parameter	Description
Type	pixel type. The current supported pixel value is XF_8UC1, unsigned 8 bit.
ROWS	Maximum number of rows of the input image that the hardware kernel must be built for.
COLS	Maximum number of columns of the input image that the hardware kernel must be built for.
NPC	Number of pixels to process per cycle. Supported values are XF_NPPC1 (=1) and XF_NPPC2(=2).
WINDOW_SIZE	Window size over which optical flow will be computed. This can be any odd positive integer.
frame0	First input images.
frame1	Second input image. Optical flow is computed between frame0 and frame1.
flowx	Horizontal component of the flow vectors. The format of the flow vectors is XF_32FC1 or single precision.
flowy	Vertical component of the flow vectors. The format of the flow vectors is XF_32FC1 or single precision.

Resource Utilization

The following table summarizes the resource utilization of xFDenseNonPyrLKOpticalFlow for a 4K image, as generated in the Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA at 300 MHz.

Table 175: xFDenseNonPyrLKOpticalFlow Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate			
		BRAM_18K	DSP_48Es	FF	LUTs
1 pixel	300	182	44	25336	21603
2-pixel	300	264	82	25740	17216

Performance Estimate

The following table summarizes performance estimates of the xFDenseNonPyrLKOpticalFlow function for a 4K image, generated using Vivado HLS 2017.1 version tool for the Xilinx xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 176: xFDenseNonPyrLKOpticalFlow Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate
		Max (ms)
1 pixel	300	28.01
2-pixel	300	14.01

Mean and Standard Deviation

The `xFMeanStddev` function computes the mean and standard deviation of input image. The output Mean value is in fixed point Q8.8 format, and the Standard Deviation value is in Q8.8 format. Mean and standard deviation are calculated as follows:

$$\mu = \frac{\sum_{y=0}^{height}\sum_{x=0}^{width} src(x, y)}{(width*height)}$$

$$\sigma = \sqrt{\frac{\sum_{y=0}^{height}\sum_{x=0}^{width} (\mu - src(x, y))^2}{(width*height)}}$$

API Syntax

```
template<int SRC_T,int ROWS, int COLS,int NPC=1>
void xFmeanstd(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src,unsigned short*
_mean,unsigned short* _stddev)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 177: xFmeanstd Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. 8-bit, unsigned, 1 channel (XF_8UC1) is supported.
ROWS	Number of rows in the image being processed.
COLS	Number of columns in the image being processed.
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input image
_mean	16-bit data pointer through which the computed mean of the image is returned.
_stddev	16-bit data pointer through which the computed standard deviation of the image is returned.

Resource Utilization

The following table summarizes the resource utilization of the xFmeanstd function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 178: xFmeanstd Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	6	896	461	121
8 pixel	150	0	13	1180	985	208

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 179: xFmeanstd Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency
1 pixel operation (300 MHz)	6.9 ms
8 pixel operation (150 MHz)	1.69 ms

Median Blur Filter

The function xFMedianBlur performs a median filter operation on the input image. The median filter acts as a non-linear digital filter which improves noise reduction. A filter size of N would output the median value of the NxN neighborhood pixel values, for each pixel.

API Syntax

```
template<int FILTER_SIZE, int BORDER_TYPE, int TYPE, int ROWS, int COLS,
int NPC>
void xFMedianBlur (xF::Mat<TYPE, ROWS, COLS, NPC> & _src, xF::Mat<TYPE,
ROWS, COLS, NPC> & _dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 180: xFMedianBlur Function Parameter Descriptions

Parameter	Description
FILTER_SIZE	Window size of the hardware filter for which the hardware kernel will be built. This can be any odd positive integer greater than 1.
BORDER_TYPE	The way in which borders will be processed in the hardware kernel. Currently, only XF_BORDER_REPLICATE is supported.
TYPE	Type of input pixel. XF_8UC1 is supported.
ROWS	Number of rows in the image being processed.
COLS	Number of columns in the image being processed.
NPC	Number of pixels to be processed in parallel. Options are XF_NPPC1 (for 1 pixel processing per clock), XF_NPPC8 (for 8 pixel processing per clock
_src	Input image.
_dst	Output image.

Resource Utilization

The following table summarizes the resource utilization of the xFMedianBlur function for XF_NPPC1 and XF_NPPC8 configurations, generated using Vivado HLS 2017.1 version tool for the Xilinx Xc7z020clg484-1 FPGA.

Table 181: xFMedianBlur Function Resource Utilization Summary

Operating Mode	FILTER_SIZE	Operating Frequency (MHz)	Utilization Estimate			
			LUTs	FFs	DSPs	BRAMs
1 pixel	3	300	1197	771	0	3
8 pixel	3	150	6559	1595	0	6
1 pixel	5	300	5860	1886	0	5

Performance Estimate

The following table summarizes performance estimates of xFMedianBlur function on Vivado HLS 2017.1 version tool for the Xilinx xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 182: xFMedianBlur Function Performance Estimate Summary

Operating Mode	FILTER_SIZE	Operating Frequency (MHz)	Input Image Size	Latency Estimate
				Max (ms)
1 pixel	3	300	1920x1080	6.99
8 pixel	3	150	1920x1080	1.75
1 pixel	5	300	1920x1080	7.00

MinMax Location

The `xFminMaxLoc` function finds the minimum and maximum values in an image and location of those values.

$$minVal = \min_{\substack{0 \le x' \le width \\ 0 \le y' \le height}} src(x', y')$$

$$maxVal = \max_{\substack{0 \le x' \le width \\ 0 \le y' \le height}} src(x', y')$$

API Syntax

```
template<int SRC_T,int ROWS,int COLS,int NPC>
void xFminMaxLoc(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src,int32_t
*max_value, int32_t *min_value,uint16_t *_minlocx, uint16_t *_minlocy,
uint16_t *_maxlocx, uint16_t *_maxlocy )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 183: xFminMaxLoc Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. 8-bit, unsigned, 1 channel (XF_8UC1), 16-bit, unsigned, 1 channel (XF_16UC1), 16-bit, signed, 1 channel (XF_16SC1), 32-bit, signed, 1 channel (XF_32SC1) are supported.
ROWS	Number of rows in the image being processed.
COLS	Number of columns in the image being processed.
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input image
max_val	Maximum value in the image, of type int.
min_val	Minimum value in the image, of type int.
_minlocx	x-coordinate location of the first minimum value.
_minlocy	y-coordinate location of the first maximum value.
_maxlocx	x-coordinate location of the first minimum value.
_maxlocy	y-coordinate location of the first maximum value.

Resource Utilization

The following table summarizes the resource utilization of the xFminMaxLoc function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 184: xFminMaxLoc Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	3	451	398	86
8 pixel	150	0	3	1049	1025	220

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 185: xFminMaxLoc Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency
1 pixel operation (300 MHz)	6.9 ms
8 pixel operation (150 MHz)	1.69 ms

Mean Shift Tracking

Mean shift tracking is one of the basic object tracking algorithms. Mean-shift tracking tries to find the area of a video frame that is locally most similar to a previously initialized model. The object to be tracked is represented by a histogram. In object tracking algorithms target representation is mainly rectangular or elliptical region. It contain target model and target candidate. Color histogram is used to characterize the object. Target model is generally represented by its probability density function (pdf). Weighted RGB histogram is used to give more importance to object pixels.

Mean-shift algorithm is an iterative technique for locating the maxima of a density function. For object tracking, the density function used is the weight image formed using color histograms of the object to be tracked and the frame to be tested. By using the weighted histogram we are taking spatial position into consideration unlike the normal histogram calculation. This function will take input image pointer, top left and bottom right coordinates of the rectangular object, frame number and tracking status as inputs and returns the centroid using recursive mean shift approach.

API Syntax

```
template <int ROWS, int COLS, int OBJ_ROWS, int OBJ_COLS, int MAXOBJ, int
MAXITERS, int SRC_T, int NPC=1>
void xFMeanShift(xF::Mat<SRC_T, ROWS, COLS, NPC> &_in_mat, uint16_t* x1,
uint16_t* y1, uint16_t* obj_height, uint16_t* obj_width, uint16_t* dx,
uint16_t* dy, uint16_t* status, uint8_t frame_status, uint8_t no_objects,
uint8_t no_iters );
```

Template Parameter Descriptions

The following table describes the template parameters.

Table 186: xFMeanShift Template Parameters

PARAMETERS	DESCRIPTION
ROWS	Maximum height of the image
COLS	Maximum width of the image
OBJ_ROWS	Maximum Height of the object to be tracked
OBJ_COLS	Maximum width of the object to be tracked
MAXOBJ	Maximum number of objects to be tracked
MAXITERS	Maximum iterations for convergence
SRC_T	Type of the input xf::Mat, must be XF_8UC4, 8-bit data with 4 channels
NPC	Number of pixels to be processed per cycle; this function supports only XF_NPPC1 or 1 pixel per cycle operations.

Function Parameter Description

The following table describes the function parameters.

Table 187: xFMeanShift Function Parameters

PARAMETERS	DESCRIPTION
_in_mat	Input xF Mat
x1	Top Left corner x-coordinate (row number) of all the objects
y1	Top Left corner y-coordinate (col number) of all the objects
obj_height	Height of all the objects (must be odd)
obj_width	Width of all the objects (must be odd)
dx	Centers x-coordinate of all the objects returned by the kernel function
dy	Centers y-coordinate of all the objects returned by the kernel function
status	Track the object only if the status of the object is true (i.e.) if the object goes out of the frame, status is made zero
frame_status	Set as zero for the first frame and one for other frames
no_objects	Number of objects we want to track
no_iters	Number of iterations for convergence

Resource Utilization and Performance Estimate

The following table summarizes the resource utilization of xFMeanShift() for normal (1 pixel) configuration as generated in Vivado HLS 2017.1 release tool for the part xczu9eg-ffvb1156-i-es1 at 300 MHz to process a RGB image of resolution,1920x1080, and for 10 objects of size of 250x250 and 4 iterations.

Table 188: xFMeanShift Function Resource Utilization and Performance Estimate Summary

Configuration	Max. Latency	BRAMs	DSPs	FFs	LUTs
1 pixel	19.28	76	14	13198	10064

Limitations

The maximum number of objects that can be tracked is 10.

Otsu Threshold

Otsu threshold is used to automatically perform clustering-based image thresholdingor the reduction of a gray-level image to a binary image. The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes.

Otsu method is used to find the threshold which can minimize the intra class variance which separates two classes defined by weighted sum of variances of two classes.

$$\sigma_w^2(t) = w_1\,\sigma_1^2(t) + w_2\,\sigma_2^2(t)$$

Where, w_1is the class probability computed from the histogram.

$$w_1 = \sum_{1}^{t} p(i) \qquad\qquad w_2 = \sum_{t+1}^{I} p(i)$$

Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance

$$\sigma_b^2 = \sigma - \sigma_w^2$$

$$\sigma_b^2 = w_1\,w_2\,(\mu_b - \mu_f)^2$$

Where, $\mu_b = \left[\sum_{1}^{t} p(i)x(i)\right] \Big/ w_1$, $\mu_f = \left[\sum_{t+1}^{I} p(i)x(i)\right] \Big/ w_2$ is the class mean.

API Syntax

```
template<int SRC_T, int ROWS, int COLS,int NPC=1> void
xFOtsuThreshold(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat, uint8_t
&_thresh)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 189: xFOtsuThreshold Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_thresh	Output threshold value after the computation

Resource Utilization

The following table summarizes the resource utilization of the xFOtsuThreshold function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 190: xFOtsuThreshold Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	8	49	2239	3353	653
8 pixel	150	22	49	1106	3615	704

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 191: xFOtsuThreshold Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation(300 MHz)	6.92 ms
8 pixel operation(150 MHz)	1.76 ms

Pixel-Wise Addition

The `xFadd` function performs the pixel-wise addition between two input images and returns the output image.

$I_{out}(x, y) = I_{in1}(x, y) + I_{in2}(x, y)$

Where:

- $I_{out}(x, y)$ is the intensity of the output image at (x, y) position
- $I_{in1}(x, y)$ is the intensity of the first input image at (x, y) position
- $I_{in2}(x, y)$ is the intensity of the second input image at (x, y) position.

XF_CONVERT_POLICY_TRUNCATE: Results are the least significant bits of the output operand, as if stored in two's complement binary format in the size of its bit-depth.

XF_CONVERT_POLICY_SATURATE: Results are saturated to the bit depth of the output operand.

API Syntax

```
template<int POLICY_TYPE, int SRC_T, int ROWS, int COLS, int NPC=1>
void xFadd (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> dst )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 192: xFadd Function Parameter Descriptions

Parameter	Description
POLICY_TYPE	Type of overflow handling. It can be either, XF_CONVERT_POLICY_SATURATE or XF_CONVERT_POLICY_TRUNCATE.
SRC_T	pixel type. Options are XF_8UC1 and XF_16SC1.
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 193: xFadd Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	0	62	55	11
8 pixel	150	0	0	65	138	24

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 194: xFadd Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150MHz)	1.7

Pixel-Wise Multiplication

The `xFmultiply` function performs the pixel-wise multiplication between two input images and returns the output image.

$I_{out}(x, y) = I_{in1}(x, y) * I_{in2}(x, y) * scale\_val$

Where:

- $I_{out}(x, y)$ is the intensity of the output image at (x, y) position
- $I_{in1}(x, y)$ is the intensity of the first input image at (x, y) position
- $I_{in2}(x, y)$ is the intensity of the second input image at (x, y) position
- scale_val is the scale value.

XF_CONVERT_POLICY_TRUNCATE: Results are the least significant bits of the output operand, as if stored in two's complement binary format in the size of its bit-depth.

XF_CONVERT_POLICY_SATURATE: Results are saturated to the bit depth of the output operand.

API Syntax

```
template<int POLICY_TYPE, int SRC_T,int ROWS, int COLS, int NPC=1>
void xFmultiply (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int SRC_T int ROWS, int COLS, int NPC> dst,
float scale)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 195: xFmultiply Function Parameter Descriptions

Parameter	Description
POLICY_TYPE	Type of overflow handling. It can be either, XF_CONVERT_POLICY_SATURATE or XF_CONVERT_POLICY_TRUNCATE.
SRC_T	pixel type. Options are XF_8UC1 and XF_16SC1.
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image
scale_val	Weighing factor within the range of 0 and 1

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 196: xFmultiply Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	2	124	59	18
8 pixel	150	0	16	285	108	43

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 197: xFmultiply Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.6

Pixel-Wise Subtraction

The `xFsubtract` function performs the pixel-wise subtraction between two input images and returns the output image.

$I_{out}(x, y) = I_{in1}(x, y) - I_{in2}(x, y)$

Where:

- $I_{out}(x, y)$ is the intensity of the output image at (x, y) position
- $I_{in1}(x, y)$ is the intensity of the first input image at (x, y) position
- $I_{in2}(x, y)$ is the intensity of the second input image at (x, y) position.

XF_CONVERT_POLICY_TRUNCATE: Results are the least significant bits of the output operand, as if stored in two's complement binary format in the size of its bit-depth.

XF_CONVERT_POLICY_SATURATE: Results are saturated to the bit depth of the output operand.

API Syntax

```
template<int POLICY_TYPE int SRC_T, int ROWS, int COLS, int NPC=1>
void xFsubtract (
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src1,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> src2,
xF::Mat<int SRC_T, int ROWS, int COLS, int NPC> dst )
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 198: xFsubtract Function Parameter Descriptions

Parameter	Description
POLICY_TYPE	Type of overflow handling. It can be either, XF_CONVERT_POLICY_SATURATE or XF_CONVERT_POLICY_TRUNCATE.
SRC_T	pixel type. Options are XF_8UC1 and XF_16SC1.
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
src1	Input image
src2	Input image
dst	Output image

Resource Utilization

The following table summarizes the resource utilization in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 199: xFsubtract Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	0	0	62	53	11
8 pixel	150	0	0	59	13	21

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 200: xFsubtract Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency (ms)
1 pixel operation (300 MHz)	6.9
8 pixel operation (150 MHz)	1.7

Remap

The `xFRemap` function takes pixels from one place in the image and relocates them to another position in another image. Two types of interpolation methods are used here for mapping the image from source to destination image.

$$dst = src(map_x(x, y), map_y(x, y))$$

API Syntax

```
template<int WIN_ROWS, int SRC_T, int MAP_T, int DST_T, int ROWS, int
COLS, int NPC = 1>

void xFRemap (xF::Mat<SRC_T, ROWS, COLS, NPC> &_src_mat,
                  xF::Mat<DST_T, ROWS, COLS, NPC> &_remapped_mat,
                  xF::Mat<MAP_T, ROWS, COLS, NPC> &_mapx_mat,
                  xF::Mat<MAP_T, ROWS, COLS, NPC> &_mapy_mat,
              int interpolation=XF_INTERPOLATION_NN);
```

Parameter Descriptions

The following table describes the template parameters.

Table 201: xFRemap template Parameter Descriptions

Parameter	Description
WIN_ROWS	Number of input image rows to be buffered inside. Must be set based on the map data. For instance, for left right flip, 2 rows are sufficient.
SRC_T	Input image type. Grayscale image of type 8-bits and single channel. XF_8UC1.
MAP_T	Map type. Single channel float type. XF_32FC1.
DST_T	Output image type. Grayscale image of type 8-bits and single channel. XF_8UC1.
ROWS	Height of input and output images
COLS	Width of input and output images
NPC	Number of pixels to be processed per cycle; this function supports only XF_NPPC1 or 1 pixel per cycle operations.

The following table describes the function parameters.

Table 202: xFRemap Function Parameter Descriptions

PARAMETERS	DESCRIPTION
_src_mat	Input xF Mat
_remapped_mat	Output xF Mat
_mapx_mat	mapX Mat of float type
_mapy_mat	mapY Mat of float type
interpolation	Type of interpolation, either XF_INTERPOLATION_NN (nearest neighbor) or XF_INTERPOLATION_BILINEAR (linear interpolation)

Resource Utilization

The following table summarizes the resource utilization of xFRemap, for HD (1080x1920) images generated in the Vivado HLS 2017.1 version tool for the Xilinx xczu9eg-ffvb1156-i-es1 FPGA at 300 MHz, with WIN_ROWS as 64.

Table 203: xFRemap Function Resource Utilization Summary

Name	Resource Utilization
BRAM_18K	128
DSP48E	14
FF	2064
LUT	2277
CLB	500

Performance Estimate

The following table summarizes the performance of xFRemap(), for HD (1080x1920) images generated in the Vivado HLS 2017.1 version tool for the Xilinx xczu9eg-ffvb1156-i-es1 FPGA at 300 MHz, with WIN_ROWS as 64.

Table 204: xFRemap Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate Max latency (ms)
1 pixel mode	300	7.2

Resolution Conversion (Resize)

Resolution Conversion is the method used to resize the source image to the size of the destination image. Different types of interpolation techniques can be used in resize function, namely: Nearest-neighbor, Bilinear, and Area interpolation. The type of interpolation can be passed as a template parameter to the API. The following enumeration types can be used to specify the interpolation type:

- XF_INTERPOLATION_NN - For Nearest-neighbor interpolation
- XF_INTERPOLATION_BILINEAR - For Bilinear interpolation
- XF_INTERPOLATION_AREA - For Area interpolation

NOTE: *Scaling factors greater than or equal to 0.25 are supported in down-scaling and values less than or equal to 8 are supported for up-scaling.*

API Syntax

```
template<int INTERPOLATION_TYPE, int TYPE, int SRC_ROWS, int SRC_COLS,
int DST_ROWS, int DST_COLS, int NPC>
void xFResize (xF::Mat<TYPE, SRC_ROWS, SRC_COLS, NPC> & _src,
xF::Mat<TYPE, DST_ROWS, DST_COLS, NPC> & _dst)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 205: xFResize Function Parameter Descriptions

Parameter	Description
INTERPOLATION_TYPE	Interpolation type. The different options possible are • XF_INTERPOLATION_NN – Nearest Neighbor Interpolation • XF_INTERPOLATION_BILINEAR – Bilinear interpolation • XF_INTERPOLATION_AREA – Area Interpolation
TYPE	Number of bits per pixel. Only XF_8UC1 is supported.
SRC_ROWS	Maximum Height of input image for which the hardware kernel would be built.
SRC_COLS	Maximum Width of input image for which the hardware kernel would be built. (must be a multiple of 8).
DST_ROWS	Maximum Height of output image for which the hardware kernel would be built.
DST_COLS	Maximum Width of output image for which the hardware kernel would be built. (must be a multiple of 8).
NPC	Number of pixels to be processed per cycle. Possible options are XF_NPPC1 (1 pixel per cycle) and XF_NPPC8 (8 pixel per cycle)
_src	Input Image
_dst	Output Image

Resource Utilization

The following table summarizes the resource utilization of Resize function in Resource Optimized (8 pixel) mode and Normal mode, as generated in the Vivado HLS 2017.1 tool for the Xilinx xczu9eg-ffvb1156-2-i-es2 FPGA, to downscale a grayscale HD(1080x1920) image to SD(640x480); and to upscale a HD(1920x1080) image to a 4K(3840x2160) image.

Table 206: xFResize Function Resource Utilization Summary

Operating Mode	Utilization Estimate							
	1 Pixel (at 300 MHz)				8 Pixel (at 150MHz)			
	LUTs	FFs	DSPs	BRAMs	LUTs	FFs	DSPs	BRAMs
Downscale Nearest Neighbor	800	1315	8	4	5079	5720	8	11
Downscale Bilinear	1067	1580	10	7	6511	4863	14	23
Downscale Area	2558	2995	42	30	324991	17726	42	79
Upscale Nearest Neighbor	803	1115	8	8	1599	1636	8	20
Upscale Bilinear	1231	1521	10	15	3588	2662	14	37
Upscale Area	1461	2107	16	25	5861	3611	36	40

Performance Estimate

The following table summarizes the performance estimation of Resize for various configurations, as generated in the Vivado HLS 2017.1 tool for the xczu9eg-ffvb1156-2-i-es2 FPGA at 300 MHz to resize a grayscale image from 1080x1920 to 480x640 (downscale); and to resize a greyscale image from 1080x1920 to 2160x3840 (upscale). This table also shows the latencies obtained for different interpolation types.

Table 207: xFResize Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate (ms)					
		Downscale NN	Downscale Bilinear	Downscale Area	Upscale NN	Upscale Bilinear	Upscale Area
1 pixel	300	6.94	6.97	7.09	27.71	27.75	27.74

Scharr Filter

The `xFScharr` function computes the gradients of input image in both x and y direction by convolving the kernel with input image being processed.

For Kernel size 3x3:

- GradientX:

$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} * I$$

- GradientY:

$$G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} * I$$

API Syntax

```
template<int BORDER_TYPE, int SRC_T,int DST_T, int ROWS, int COLS,int
NPC=1>
void xFScharr(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat,xF::Mat<DST_T,
ROWS, COLS, NPC> & _dst_matx,xF::Mat<DST_T, ROWS, COLS, NPC> & _dst_maty)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 208: xFScharr Function Parameter Descriptions

Parameter	Description
BORDER_TYPE	Border Type supported is XF_BORDER_CONSTANT
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_T	Output pixel type. Only 16-bit,signed,1 channel is supported (XF_16SC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_matx	X gradient output image.
_dst_maty	Y gradient output image.

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 209: xFScharr Function Resource Utilization Summary

Name	Resource Utilization	
	1 pixel	**8 pixel**
	300 MHz	**150MHz**
BRAM_18K	3	6
DSP48E	0	0
FF	728	1434
LUT	812	2481
CLB	171	461

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 210: xFScharr Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate (ms)
1pixel	300	7.2
8pixel	150	1.7

Sobel filter

The `xFSobel` function Computes the gradients of input image in both x and y direction by convolving the kernel with input image being processed.

- For Kernel size 3x3
  - GradientX:

$$G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} * I$$

  - GradientY:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I$$

- For Kernel size 5x5
  - GradientX:

$$G_x = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix} * I$$

  - GradientY:

$$G_y = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} * I$$

- For Kernel size 7x7
  - GradientX:

$$G_x = \begin{bmatrix} -1 & -4 & -5 & 0 & 5 & 4 & 1 \\ -6 & -24 & -30 & 0 & 30 & 24 & 6 \\ -15 & -60 & 75 & 0 & 75 & 60 & 15 \\ -20 & -80 & -100 & 0 & 75 & 60 & 15 \\ -15 & -60 & -75 & 0 & 75 & 60 & 15 \\ -6 & -24 & -30 & 0 & 30 & 24 & 6 \\ -1 & -4 & -5 & 0 & 5 & 4 & 1 \end{bmatrix} *I$$

  - GradientY:

$$G_y = \begin{bmatrix} -1 & -6 & -15 & -20 & -15 & -6 & -1 \\ -4 & -24 & -60 & -80 & -60 & -24 & -4 \\ -5 & -30 & -75 & -100 & -75 & -30 & -5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 30 & 75 & 100 & 75 & 30 & 5 \\ 4 & 24 & 60 & 80 & 60 & 24 & 4 \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{bmatrix} *I$$

API Syntax

```
template<int BORDER_TYPE,int FILTER_TYPE, int SRC_T,int DST_T, int ROWS,
int COLS,int NPC=1>
void xFSobel(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src_mat,xF::Mat<DST_T,
ROWS, COLS, NPC> & _dst_matx,xF::Mat<DST_T, ROWS, COLS, NPC> & _dst_maty)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 211: xFSobel Function Parameter Descriptions

Parameter	Description
FILTER_TYPE	Filter size. Filter size of 3(XF_FILTER_3X3), 5(XF_FILTER_5X5) and 7(XF_FILTER_7X7) are supported
BORDER_TYPE	Border Type supported is XF_BORDER_CONSTANT
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_T	Output pixel type. Only 16-bit, signed, 1 channel is supported (XF_16SC1) for filter size 3 and 5. Only 32 bit, signed 1 channel is supported for filter size 7.
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_matx	X gradient output image.
_dst_maty	Y gradient output image.

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 212: xFSobel Function Resource Utilization Summary

Operating Mode	Filter Size	Operating Frequency (MHz)	Utilization Estimate				
			BRAM_18K	DSP_48Es	FF	LUT	CLB
1pixel	3x3	300	3	0	609	616	135
	5x5	300	5	0	1133	1499	308
	7x7	300	7	0	2658	3334	632
8pixel	3x3	150	6	0	1159	1892	341
	5x5	150	10	0	3024	5801	999

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 213: xFSobel Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Filter Size	Latency Estimate (in ms)
1pixel	300	3x3	7.5
	300	5x5	7.5
	300	7x7	7.5
8pixel	150	3x3	1.7
	150	5x5	1.71

Stereo Local Block Matching

Stereo block matching is a method to estimate the motion of the blocks between the consecutive frames, called stereo pair. The postulate behind this idea is that, considering a stereo pair, the foreground objects will have disparities higher than the background. Local block matching uses the information in the neighboring patch based on the window size, for identifying the conjugate point in its stereo pair. While, the techniques under global method, used the information from the whole image for computing the matching pixel, providing much better accuracy than local methods. But, the efficiency in the global methods are obtained with the cost of resources, which is where local methods stands out.

Local block matching algorithm consists of preprocessing and disparity estimation stages. The preprocessing consists of Sobel gradient computation followed by image clipping. And the disparity estimation consists of SAD (Sum of Absolute Difference) computation and obtaining the disparity using winner takes all method (least SAD will be the disparity). Invalidity of the pixel relies upon its uniqueness from the other possible disparities. And the invalid pixels are indicated with the disparity value of zero.

API Syntax

```
template<int WSIZE, int NDISP, int NDISP_UNITS, int SRC_T, int DST_T, int
ROWS, int COLS, int NPC =  1>
void xFFindStereoCorrespondenceBM (xF::Mat<SRC_T, ROWS, COLS, NPC> &
left_image, xF::Mat<SRC_T, ROWS, COLS, NPC> & right_image, xF::Mat<DST_T,
ROWS, COLS, NPC> & disparity_image, xF::SBMState< WSIZE,NDISP,NDISP_UNITS
> &sbmstate)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 214: xFFindStereoCorrespondenceBM Function Parameter Descriptions

Parameter	Description
WSIZE	Size of the window used for disparity computation
NDISP	Number of disparities
NDISP_UNITS	Number of disparities to be computed in parallel.
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
DST_T	Output type. This is XF_16UC1, where the disparities are arranged in Q12.4 format.
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 only.
left_image	Image from the left camera
right_image	Image from the right camera
disparity_image	Disparities output in the form of an image.
sbmstate	Class object consisting of various parameters regarding the stereo block matching algorithm. 1. preFilterCap: default value is 31, can be altered by the user, value ranges from 1 to 63 2. minDisparity: default value is 0, can be altered by the user, value ranges from 0 to (imgWidth-NDISP) 3. uniquenessRatio: default set to 15, but can be altered to any non-negative integer. 4. textureThreshold: default set to 10, but can be modified to any non-negative integer.

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to progress a grayscale HD (1080x1920) image.

The configurations are in the format: imageSize_SADwinSize_NDisp_NDispUnits.

Table 215: xFFindStereoCorrespondenceBM Function Resource Utilization Summary

Configurations	Frequency (MHz)	Resource Utilization			
		BRAM_18k	DSP48E	FF	LUT
HD_5_16_2	300	37	20	6856	7181
HD_9_32_4	300	45	20	9700	10396
HD_11_32_32	300	49	20	34519	31978

Configurations	Frequency (MHz)	Resource Utilization			
		BRAM_18k	DSP48E	FF	LUT
HD_15_128_32	300	57	20	41017	35176
HD_21_64_16	300	69	20	29853	30706

Performance Estimate

The following table summarizes a performance estimate of the Stereo local block matching in different configurations, as generated using Vivado HLS 2017.1 tool for Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

The configurations are in the format: imageSize_SADwinSize_NDisp_NDispUnits.

Table 216: xFFindStereoCorrespondenceBM Function Performance Estimate Summary

Configurations	Frequency (MHz)	Latency (ms)	
		Min	Max
HD_5_16_2	300	55.296	55.296
HD_9_32_4	300	55.296	55.296
HD_11_32_32	300	6.912	6.912
HD_15_48_16	300	20.736	20.736
HD_15_128_32	300	27.648	27.648
HD_21_64_16	300	27.648	27.648

SVM

The `xFSVM` function is the SVM core operation, which performs dot product between the input arrays. The function returns the resultant dot product value with its fixed point type.

API Syntax

```
template<int SRC1_T, int SRC2_T, int DST_T, int ROWS1, int COLS1, int
ROWS2, int COLS2, int NPC=1, int N>
void xFSVM(xF::Mat<SRC1_T, ROWS1, COLS1, NPC> &in_1, xF::Mat<SRC2_T,
ROWS2, COLS2, NPC> &in_2, uint16_t idx1, uint16_t idx2, uchar_t frac1,
uchar_t frac2, uint16_t n, uchar_t *out_frac,
ap_int<XF_PIXELDEPTH(DST_T)> *result)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 217: xFSVM Function Parameter Descriptions

Parameters	Description
SRC1_T	Input pixel type. 16-bit, signed, 1 channel (XF_16SC1) is supported.
SRC2_T	Input pixel type. 16-bit, signed, 1 channel (XF_16SC1) is supported.
DST_T	Output data Type. 32-bit, signed, 1 channel (XF_32SC1) is supported.
ROWS1	Number of rows in the first image being processed.
COLS1	Number of columns in the first image being processed.
ROWS2	Number of rows in the second image being processed.
COLS2	Number of columns in the second image being processed.
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1.
N	Max number of kernel operations
in_1	First Input Array.
in_2	Second Input Array.
idx1	Starting index of the first array.
idx2	Starting index of the second array.
frac1	Number of fractional bits in the first array data.
frac2	Number of fractional bits in the second array data.
n	Number of kernel operations.
out_frac	Number of fractional bits in the resultant value.
result	Resultant value

Resource Utilization

The following table summarizes the resource utilization of the xFSVM function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 218: xFSVM Function Resource Utilization Summary

Operating Frequency (MHz)	Utilization Estimate (ms)				
	BRAM_18K	DSP_48Es	FF	LUT	CLB
300	0	1	27	34	12

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 219: xFSVM Function Performance Estimate Summary

Operating Frequency (MHz)	Latency Estimate	
	Min (cycles)	Max (cycles)
300	204	204

Thresholding

The `xFThreshold` function performs thresholding operation on the input image. Thresholding are of two types, binary and range.

In the Binary thresholding, the threshold value will be set and the depending upon the threshold value the output is set to either 0 or 255. The _binary_thresh_val has to be set to the threshold value when the binary thresholding is selected, as well as the _upper_range and _lower_range has to be set to 0.

$$dst(x, y) = \begin{cases} 255, & if \ \ src(x, y) > \ threshold \\ 0, & Otherwise \end{cases}$$

In the Range thresholding, the upper and the lower threshold values are set and depending upon those values the output is set either 0 or 255. The _upper_range and _lower_range values has to be set to the upper and the lower threshold values respectively, when the range thresholding is selected, as well as in this case the _binary_thresh_val has to be set to 0.

$$dst(x, y) = \begin{cases} 0, & if \ src(x, y) > upper \\ 0, & if \ src(x, y) < lower \\ 255, & otherwise \end{cases}$$

API Syntax

```
template<int THRESHOLD_TYPE, int SRC_T, int ROWS, int COLS,int NPC=1>
void xFThreshold(xF::Mat<SRC_T, ROWS, COLS, NPC> &
_src_mat,xF::Mat<SRC_T, ROWS, COLS, NPC> & _dst_mat,short
thresh_val,short thresh_upper,short thresh_lower)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 220: xFThreshold Function Parameter Descriptions

Parameter	Description
THRESHOLD_TYPE	Type of thresholding. It can be either binary thresholding or range thresholding. Options are XF_THRESHOLD_TYPE_BINARY or XF_THRESHOLD_TYPE_RANGE.
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; this function supports only XF_NPPC1 or 1 pixel per cycle operations.
_src_mat	Input image
_dst_mat	Output image

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 221: xFThreshold Function Resource Utilization Summary

c	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150MHz
BRAM_18K	0	0
DSP48E	3	3
FF	410	469
LUT	277	443
CLB	72	103

Performance Estimate

The following table summarizes the performance of the kernel in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 222: xFThreshold Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate (ms)
1pixel	300	7.2
8pixel	150	1.7

WarpAffine

Affine Transformation is used to express rotation, scaling and translation operations.

`xFWarpAffine` takes inverse transformation as input and transforms the input image by matrix multiplication with 2x3 matrix.Calculate the input position corresponding to every output pixel in the following way.

$$\begin{bmatrix} x_{input} \\ y_{input} \end{bmatrix} = \begin{bmatrix} M_{0,0} & M_{0,1} & M_{0,2} \\ M_{1,0} & M_{1,1} & M_{1,2} \end{bmatrix} \begin{bmatrix} x_{output} \\ y_{output} \\ 1 \end{bmatrix}$$

$$x_{input} = M_{0,0} * x_{output} + M_{0,1} * y_{output} + M_{0,2}$$

$$y_{input} = M_{1,0} * x_{output} + M_{1,1} * y_{output} + M_{1,2}$$

$$output(x_{output}, y_{output}) = input(x_{input}, y_{input})$$

NOTE: *Prior scaling of greater than or equal to 0.25 and less than or equal to 8 are supported and throws an assertion if the transformation matrix passed is having unsupported scaling factors.*

API Syntax

```
template<int INTERPOLATION_TYPE, int SRC_T, int ROWS, int COLS, int NPC=1>
void xFwarpAffine(xF::Mat<SRC_T, ROWS, COLS, XF_NPPC8> & _src,
xF::Mat<SRC_T, ROWS, COLS, XF_NPPC8> & _dst, float* transformation_matrix)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 223: xFwarpAffine Function Parameter Descriptions

Parameter	Description
INTERPOLATION_TYPE	Interpolation technique to be used XF_INTERPOLATION_NN - Nearest Neighbor XF_INTERPOLATION_BILINEAR - Bilinear
SRC_T	Input pixel type.
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input Image pointer
_dst	Output Image pointer
transformation_matrix	Inverse Affine Transformation matrix

Resource Utilization

The following table shows the resource utilization of xFWarpAffine in different configurations as generated in Vivado HLS 2017.1 version tool for the part Xilinx Xczu9eg-ffvb1156-1-i-es1.

Table 224: xFwarpAffine Function Resource Utilization Summary

Name	Utilization (at 250 MHz)			
	Normal Operation (1 pixel) Mode		Resource Optimized (8 pixel) Mode	
	Nearest Neighbour	Bilinear Inerpolation	Nearest Neighbour	Bilinear Interpolation
BRAM	200	194	200	200
DSP48E	121	125	139	155
FF	9523	10004	8070	9570
LUT	9928	10592	13296	13894
CLB	2390	2435	2932	2829

NOTE: *NO is single pixel processing and RO is 4-pixel processing.*

Performance Estimate

The following table shows the performance of Affine Transformation in different configurations, as generated in Vivado HLS 2017.1 version tool for the part Xilinx Xczu9eg-ffvb1156-1-i-es1.

Table 225: xFwarpAffine Function Performance Estimate Summary

Latency Estimate			
1pixel(300 MHz)		8pixel(150 MHz)	
Nearest Neighbour	Bilinear Interpolation	Nearest Neighbour	Bilinear Interpolation
Max(in ms)	Max(in ms)	Max(in ms)	Max(in ms)
10.4	19.3	10.7	10.4

WarpPerspective

The `xFperspective` function performs the perspective transformation on the image. It takes inverse transformation as input and applies perspective transformation with a 3x3 matrix.

Calculate the input position corresponding to every output pixel as following:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} M_{0,0} & M_{0,1} & M_{0,2} \\ M_{1,0} & M_{1,1} & M_{1,2} \\ M_{2,0} & M_{2,1} & M_{2,2} \end{bmatrix} \begin{bmatrix} x_{output} \\ y_{output} \\ 1 \end{bmatrix}$$

$$x = M_{0,0} * x_{output} + M_{0,1} * y_{output} + M_{0,2}$$

$$y = M_{1,0} * x_{output} + M_{1,1} * y_{output} + M_{1,2}$$

$$z = M_{2,0} * x_{output} + M_{2,1} * y_{output} + M_{2,2}$$

$$output(x_{output}, y_{output}) = input(x/z, y/z)$$

NOTE: *Prior scaling of greater than or equal to 0.25 and less than or equal to 8 are supported and throws an assertion if the transformation matrix passed is having unsupported scaling factors.*

API Syntax

```
template< int INTERPOLATION_TYPE ,int SRC_T, int ROWS, int COLS,int NPC>
void xFperspective(xF::Mat<SRC_T, ROWS, COLS, XF_NPPC8> &
_src_mat,xF::Mat<SRC_T, ROWS, COLS, XF_NPPC8> & _dst_mat,float
*transformation_matrix)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 226: xFperspective Function Parameter Descriptions

Parameter	Description
INTERPOLATION_TYPE	Interpolation technique to be used XF_INTERPOLATION_NN - Nearest Neightbour XF_INTERPOLATION_BILINEAR - Bilinear
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image (must be a multiple of 8)
COLS	Maximum width of input and output image (must be a multiple of 8)
NPC	Number of pixels to be processed per cycle; possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src_mat	Input image
_dst_mat	Output image
transformation_matrix	Inverse Perspective Transformation matrix

Resource Utilization

The following table summarizes the resource utilization of the kernel in different configurations, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image for Bilinear interpolation.

Table 227: xFperspective Function Resource Utilization Summary

Name	Resource Utilization	
	1 pixel	8 pixel
	300 MHz	150 MHz
BRAM_18K	223	233
DSP48E	191	293
FF	17208	14330
LUT	14458	18969
CLB	3230	3876

Performance Estimate

The following table summarizes the performance of kernel for different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image for Bilinear interpolation.

Table 228: xFperspective Function Performance Estimate Summary

Operating Mode	Operating Frequency (MHz)	Latency Estimate Max (ms)
1 pixel	300	19.3
8 pixel	150	15.5

Atan2

The `xFAtan2LookupFP` function finds the arctangent of y/x. It returns the angle made by the vector $\begin{bmatrix} x \\ y \end{bmatrix}$ with respect to origin. The angle returned by atan2 will also contain the quadrant information.

The function xFAtan2LookupFP is a fixed point version of the standard atan2 function. This function implements the atan2 using a lookup table approach. The values in the look up table are represented in Q4.12 format and so the values returned by this function are in Q4.12. A maximum error of 0.2 degrees is present in the range of 89 to 90 degrees when compared to the standard atan2 function available in glibc. For the other angles (0 to 89) the maximum error is in the order of 10-3. This function returns 0 when both xs and ys are zeroes.

API Syntax

```
short xFAtan2LookupFP(short xs, short ys, int M1,int N1,int M2, int N2)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 229: xFAtan2LookupFP Function Parameter Descriptions

Parameter	Description
xs	16-bit signed value x in fixed point format of QM1.N1
ys	16-bit signed value y in fixed point format of QM2.N2
M1	Number of bits to represent integer part of x.
N1	Number of bits to represent fractional part of y. Must be equal to 16-M1.
M2	Number of bits to represent integer part of y.
N2	Number of bits to represent fractional part of y. Must be equal to 16-N1.
Return	Return value is in radians. Its range varies from -pi to +pi in fixed point format of Q4.12

Resource Utilization

The following table summarizes the resource utilization of the xFAtan2LookupFP function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 230: xFAtan2LookupFP Function Resource Utilization Summary

Operating Frequency (MHz)	Utilization Estimate				
	BRAM_18K	DSP_48Es	FF	LUT	CLB
300	4	2	275	75	139

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 231: xFAtan2LookupFP Function Performance Estimate Summary

Operating Frequency (MHz)	Latency Estimate	
	Min (cycles)	Max (cycles)
300	1	15

Inverse (Reciprocal)

The `xFInverse` function computes the reciprocal of a number x. The values of 1/x are stored in a look up table of 2048 size. The index for picking the 1/x value is computed using the fixed point format of x. Once this index is computed, the corresponding 1/x value is fetched from the look up table and returned along with the number of fractional bits needed to represent this value in fixed point format.

API Syntax

```
unsigned int xFInverse(unsigned short x,int M,char *N)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 232: xFInverse Function Parameter Descriptions

Parameter	Description
x	16-bit unsigned value x in fixed point format of QM.(16-M)
M	Number of bits to represent integer part of x.
N	Pointer to a char variable which stores the number of bits to represent fractional part of 1/x. This value is returned from the function.
Return	1/x value is returned in 32-bit format represented by a fixed point format of Q(32-N).N

Resource Utilization

The following table summarizes the resource utilization of the xFInverse function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 233: xFInverse Function Resource Utilization Summary

Operating Frequency (MHz)	Utilization Estimate (ms)				
	BRAM_18K	DSP_48Es	FF	LUT	CLB
300	4	0	68	128	22

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 234: xFInverse Function Performance Estimate Summary

Operating Frequency (MHz)	Latency Estimate	
	Min (cycles)	Max (cycles)
300	1	8

Look Up Table

The `xFLUT` function performs the table lookup operation. Transforms the source image into the destination image using the given look-up table. The input image must be of depth AU_8UP and the output image of same type as input image.

$I_{out}(x, y) = LUT [I_{in1}(x, y)]$

Where:

- $I_{out}(x, y)$ is the intensity of output image at (x, y) position
- $I_{in}(x, y)$ is the intensity of first input image at (x, y) position
- LUT is the lookup table of size 256 and type unsigned char.

API Syntax

```
template <int SRC_T, int ROWS, int COLS,int NPC=1>
void xFLUT(xF::Mat<SRC_T, ROWS, COLS, NPC> & _src, xF::Mat<SRC_T, ROWS,
COLS, NPC> & _dst,unsigned char* _lut)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 235: xFLUT Function Parameter Descriptions

Parameter	Description
SRC_T	Input pixel type. 8-bit, unsigned, 1 channel (XF_8UC1) is supported.
ROWS	Number of rows in the image being processed.
COLS	Number of columns in the image being processed.
NPC	Number of pixels to be processed in parallel. Possible options are XF_NPPC1 and XF_NPPC8 for 1 pixel and 8 pixel operations respectively.
_src	Input image of size (ROWS, COLS) and type 8U.
_dst	Output image of size (ROWS, COLS) and same type as input.
_lut	Input lookup Table of size 256 and type unsigned char.

Resource Utilization

The following table summarizes the resource utilization of the xFLUT function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 236: xFLUT Function Resource Utilization Summary

Operating Mode	Operating Frequency (MHz)	Utilization Estimate				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 pixel	300	1	0	937	565	137
8 pixel	150	9	0	1109	679	162

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1, to process a grayscale HD (1080x1920) image.

Table 237: xFLUT Function Performance Estimate Summary

Operating Mode	Latency Estimate
	Max Latency
1 pixel operation (300 MHz)	6.92 ms
8 pixel operation (150 MHz)	1.66 ms

Square Root

The `xFSqrt` function computes the square root of a 16-bit fixed point number using the non-restoring square root algorithm. The non-restoring square root algorithm uses the two's complement representation for the square root result. At each iteration the algorithm can generate exact result value even in the last bit.

Input argument D must be 16-bit number, though it is declared as 32-bit. The output sqrt(D) is 16-bit type. If format of D is QM.N (where M+N = 16) then format of output is Q(M/2).N

To get a precision of 'n' bits in fractional part, you can simply left shift the radicand (D) by '2n' before the function call and shift the solution right by 'n' to get the correct answer. For example, to find the square root of 35 (011000112) with one bit after the decimal point, that is, N=1:

1. Shift the number (01100011002) left by 2
2. Shift the answer (10112) right by 1. The correct answer is 101.1, which is 5.5.

API Syntax

```
int xFSqrt(unsigned int D)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 238: xFSqrt Function Parameter Descriptions

Parameter	Description
D	Input data in a 16-bit fixed-point format.
Return	Output value in short int format.

Resource Utilization

The following table summarizes the resource utilization of the xFSqrt function, generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 239: xFSqrt Function Resource Utilization Summary

Operating Frequency (MHz)	Utilization Estimate				
	BRAM_18K	DSP_48Es	FF	LUT	CLB
300	0	0	8	6	1

Performance Estimate

The following table summarizes the performance in different configurations, as generated using Vivado HLS 2017.1 tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA.

Table 240: xFSqrt Function Performance Estimate Summary

Operating Frequency (MHz)	Latency Estimate	
	Min (cycles)	Max (cycles)
300	18	18

WarpTransform

The `xFWarpTransform` function is designed to perform the perspective and affine geometric transformations on an image. The type of transform is a compile time parameter to the

function.

The function uses a streaming interface to perform the transformation. Due to this and due to the fact that geometric transformations need access to many different rows of input data to compute one output row, the function stores some rows of the input data in BRAMs. The number of rows the function stores can be configured by the user by modifying a template parameter. Based on the transformation matrix, you can decide on the number of rows to be stored. You can also choose when to start transforming the input image in terms of the number of rows of stored image.

Affine Transformation

The transformation matrix consists of size parameters, and is as shown:

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \end{bmatrix}$$

Affine transformation is applied in the xFWarpTransform function following the equation:

$$dst\begin{pmatrix} x \\ y \end{pmatrix} = M^{*}\ src\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Perspective Transformation

The transformation matrix is a 3x3 matrix as shown below:

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

Perspective transformation is applied in xFWarpTransform following the equation:

$$dst^{1}\begin{pmatrix} x \\ y \\ n \end{pmatrix} = M^{*}\ src\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

The destination pixel is then computed by dividing the first two dimensions of the dst1 by the third dimension

$$dst^{1}\begin{pmatrix} x \\ y \\ n \end{pmatrix} = M^{*}\ src\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

API Syntax

```
template<int STORE_LINES, int START_ROW, int TRANSFORMATION_TYPE, int
INTERPOLATION_TYPE, int SRC_T, int ROWS, int COLS, int NPC=1>
void xFWarpTransform(xF::Mat<SRC_T, ROWS, COLS, NPC> & src,
xF::Mat<SRC_T, ROWS, COLS, NPC> & dst, float *transformation_matrix)
```

Parameter Descriptions

The following table describes the template and the function parameters.

Table 241: xFWarpTransform Function Parameter Descriptions

Parameter	Description
STORE_LINES	Number of lines of the image that need to be buffered locally on FPGA.
START_ROW	Number of the input rows to store before starting the image transformation. This must be less than or equal to STORE_LINES.
TRANFORMATION_TYPE	Affine and perspective transformations are supported. Set this flag to '0' for affine and '1' for perspective transformation.
INTERPOLATION_TYPE	Set flag to '1' for bilinear interpolation and '0' for nearest neighbor interpolation.
SRC_T	Input pixel type. Only 8-bit, unsigned, 1 channel is supported (XF_8UC1)
ROWS	Maximum height of input and output image.
COLS	Maximum width of input and output image.
NPC	Number of pixels to be processed per cycle; only one-pixel operation supported (XF_NPPC1).
src	Input image
dst	Output image
transformation_matrix	Transformation matrix that is applied to the input image.

Resource Utilization

The following table summarizes the resource utilization of the Warp transform, generated using Vivado HLS 2017.1 version tool for the Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to progress a grayscale HD (1080x1920) image.

Table 242: xFWarpTransform Function Resource Utilization Summary

Transformation	INTERPOLATION_TYPE	STORE_LINES	START_ROW	Operating Frequency (MHz)	Utilization Estimate			
					LUTs	FFs	DSPs	BRAMs
Perspective	Bilinear	100	50	300	7468	9804	61	112
Perspective	Nearest Neighbor	100	50	300	4514	6761	35	104
Affine	Bilinear	100	50	300	6139	5606	40	124
Affine	Nearest Neighbor	100	50	300	4611	4589	18	112

Performance Estimate

The following table summarizes a performance estimate of the Warp transform, as generated using Vivado HLS 2017.1 tool for Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA, to process a grayscale HD (1080x1920) image.

Table 243: xFWarpTransform Function Performance Estimate Summary

Transformation	INTERPOLATION_TYPE	STORE_LINES	START_ROW	Operating Frequency (MHz)	Latency Estimate Max (ms)
Perspective	Bilinear	100	50	300	7.46
Perspective	Nearest Neighbor	100	50	300	7.31
Affine	Bilinear	100	50	300	7.31
Affine	Nearest Neighbor	100	50	300	7.24

XILINX
ALL PROGRAMMABLE

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips

References

These documents provide supplemental material useful with this guide:

1. *SDx Environments Release Notes, Installation, and Licensing Guide* (UG1238)
2. *SDSoC Environment User Guide* (UG1027)
3. *SDSoC Environment Optimization Guide* (UG1235)
4. *SDSoC Environment Tutorial: Introduction* (UG1028)
5. *SDSoC Environment Platform Development Guide* (UG1146).
6. SDSoC Development Environment web page
7. *UltraFast Embedded Design Methodology Guide* (UG1046)
8. *ZC702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC User Guide* (UG850)
9. *Vivado Design Suite User Guide: High-Level Synthesis* (UG902)
10. *PetaLinux Tools Documentation: Workflow Tutorial* (UG1156)
11. Vivado® Design Suite Documentation
12. *Vivado Design Suite User Guide: Creating and Packaging Custom IP* (UG1118)

Please Read: Important Legal Notices

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx’s limited warranty, please refer to Xilinx’s Terms of Sale which can be viewed at www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx’s Terms of Sale which can be viewed at www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS “XA” IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE (“SAFETY APPLICATION”) UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD (“SAFETY DESIGN”). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.