ザイリンクス OpenCV ユーザー ガイド

UG1233 (v2017.1) 2017 年 6 月 20 日

この資料は表記のバージョンの英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。資料によっては英語版の更新に対応していないものがあります。日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

XILINX
ALL PROGRAMMABLE™

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	改訂内容
2017 年 7 月 18 日	2017.1	「xfOpenCV ライブラリの使用」セクションのインストラクションをアップデート。
2017 年 6 月 20 日	2017.1	初版。

目次

概要

基本的な機能 4
reVISION プラットフォーム上の xfOpenCV カーネル 5

はじめに

使用要件 7
xfOpenCV ライブラリの内容 8
xfOpenCV ライブラリの使用 8
サンプル makefile を使用した Linux でのプロジェクトの構築 9
ハードウェア カーネル コンフィギュレーションの変更 10
xfOpenCV ライブラリ関数のハードウェアでの使用 11

xfOpenCV ライブラリ API リファレンス

xF::Mat 画像コンテナー クラス 15
xfOpenCV ライブラリ関数 21

その他のリソースおよび法的通知

参考資料 157
お読みください: 重要な法的通知 158

XILINX
ALL PROGRAMMABLE™

第 1 章

概要

この資料では、ザイリンクス xfOpenCV ライブラリと呼ばれる FPGA デバイス用に最適化された OpenCV ライブラリについて説明します。ザイリンクス Zynq®-7000 All Programmable SoC および Zynq UltraScale+™ MPSoC デバイスを使用するアプリケーション開発者向けに記述されています。xfOpenCV ライブラリは、SDx™ 開発環境で使用するために設計されており、FPGA デバイス上でアクセラレーションされるコンピューター ビジョン関数用のソフトウェア インターフェイスを提供します。xfOpenCV ライブラリ関数の機能は、OpenCV の等価関数とほぼ同じです。違いがある場合は、このユーザー ガイドに記述されています。

注記: xfOpenCV ライブラリの使用要件は、使用要件を参照してください。xfOpenCV ライブラリ関数を使用するための手順の詳細は、xfOpenCV ライブラリの使用を参照してください。

基本的な機能

xfOpenCV ライブラリ関数は、すべて一般的なフォーマットに従っています。すべての関数に次の特徴があります。

- すべての関数はテンプレートとして設計されており、画像である引数はすべて `xF::Mat` として供給する必要があります。
- 次に、主なテンプレート引数を示します。
  - 処理する画像の最大サイズ
  - 各ピクセルのプロパティを定義するデータ型
  - クロック サイクルごとに処理されるピクセル数
  - 機能に関連するその他のコンパイル時引数。

xfOpenCV ライブラリには列挙データ型が含まれており、ユーザーが `xF::Mat` を設定できます。`xF::Mat` の詳細は、xF::Mat 画像コンテナー クラス を参照してください。

reVISION プラットフォーム上の xfOpenCV カーネル

xfOpenCV ライブラリは、SDx™ 開発環境で使用できるように設計されています。xfOpenCV カーネルは、reVISION™ プラットフォームで評価されています。

次に、入力と出力の両方が画像ファイルであるサンプル デザインの一般的なフローを示します。

1. `cv::imread()` を使用して画像を読み込みます。
2. データを `xF::Mat` にコピーします。
3. xfOpenCV で処理関数を呼び出します。
4. データを `xF::Mat` から `cv::Mat` にコピーします。
5. `cv::imwrite()` を使用して出力を画像に書き出します。

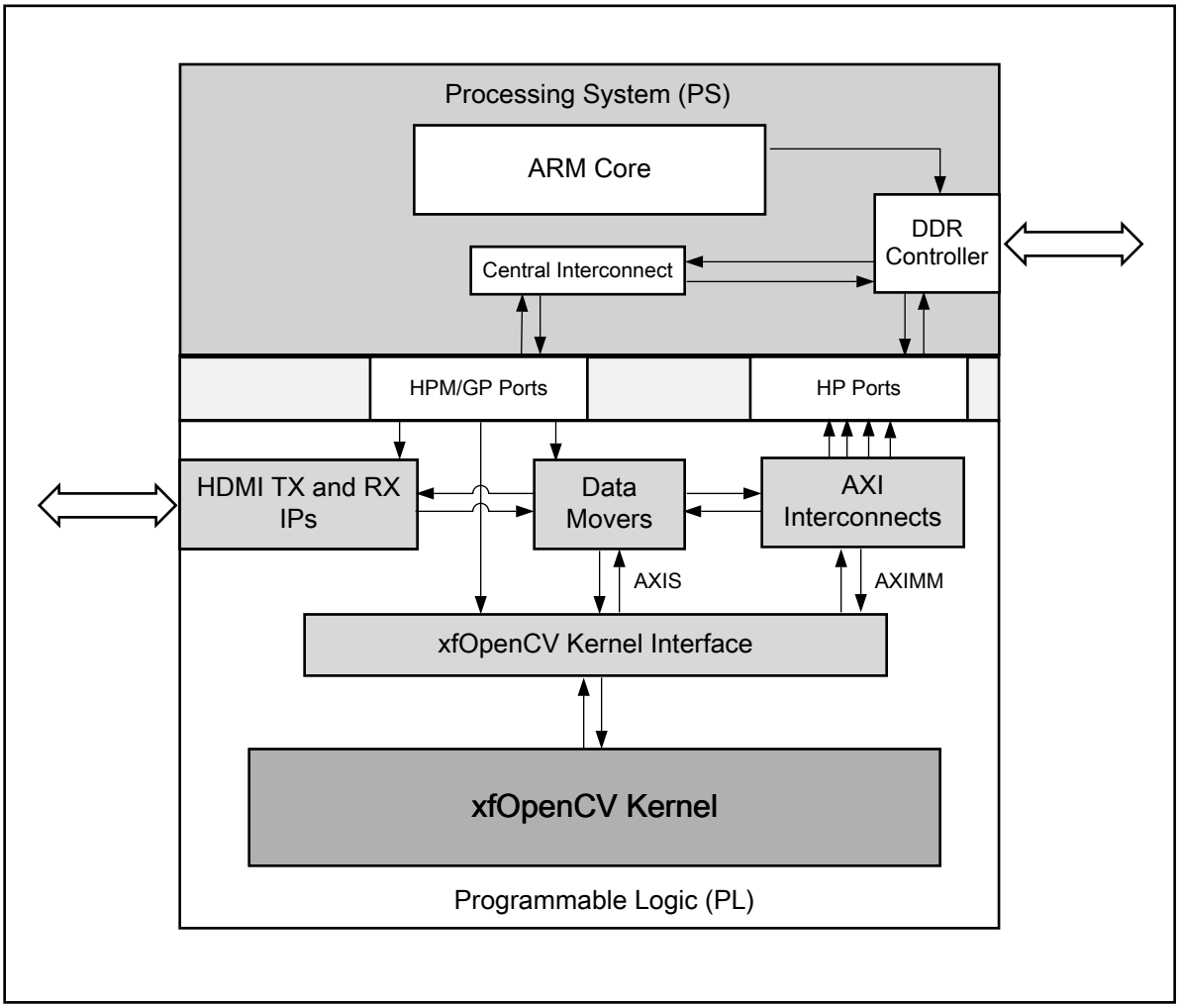
コード全体はパイプラインのホスト コードとして書き出され、そこから xfOpenCV 関数へのすべての呼び出しがハードウェアに移動されます。OpenCV からの関数を使用して、メモリ内の画像の読み出しおよび書き込みを実行します。xfOpenCV ライブラリ関数の画像コンテナーは、`xF::Mat` オブジェクトです。詳細は、xF::Mat 画像コンテナー クラスを参照してください。

reVISION プラットフォームでは、ライブおよびファイル入力/出力 (I/O) モードがサポートされます。詳細は、「reVISION 入門ガイド」を参照してください。

- ファイル I/O モードでは、コントローラーで画像が SD カードからハードウェア カーネルに転送されます。ファイル I/O モードの手順は、次のとおりです。
  1. プロセッシング システム (PS) で SD カードから画像フレームを読み出し、DRAM に格納します。
  2. xfOpenCV カーネルで DRAM から画像を読み出して処理し、出力を DRAM メモリに戻します。
  3. PS で DRAM から出力画像フレームを読み出し、SD カードに戻します。
- ライブ I/O モードでは、フレームをプラットフォームにストリーミングし、xfOpenCV カーネルでフレームを処理して、適切なインターフェイスを介してフレームをストリーミング出力します。ライブ I/O モードの手順は、次のとおりです。
  1. ビデオ キャプチャ IP でフレームを受信し、DRAM に格納します。
  2. xfOpenCV カーネルで DRAM から画像をフェッチして処理し、出力を DRAM に格納します。
  3. 表示 IP で DRAM から出力フレームを読み出し、画像インターフェイスを介してフレームを出力します。

次の図に、xfOpenCV ブロックを含む reVISION プラットフォームを示します。

図 1: reVISION プラットフォーム上の xfOpenCV カーネル



注記: PS-PL インターフェイスおよび PL-DDR インターフェイスの詳細は、『Zynq UltraScale+ MPSoC テクニカル リファレンス マニュアル』(UG1085: 英語版、日本語版) を参照してください。

XILINX
ALL PROGRAMMABLE™

第 2 章

はじめに

この章では、xfOpenCV ライブラリ関数を使用してデザインを作成するために必要な情報を示します。

使用要件

このセクションでは、xfOpenCV ライブラリ関数を使用するための要件を示します。

- SDSoC 環境のダウンロードおよびインストール方法については、『SDx 環境リリース ノート、インストールおよびライセンス ガイド』(UG1238) を参照してください。SDx 開発環境を起動する前に、`$SYSROOT` 環境変数で reVISION プラットフォームで提供される Linux ルート ファイル システムを指定するよう設定してください。例:

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

- Zynq® UltraScale+™ MPSoC エンベデッド ビジョン プラットフォーム ZIP ファイルをダウンロードして解凍します。解凍したデザイン ファイル階層の `zcu102_es2_reVISION` フォルダーに SDx 開発環境ワークスペースを作成します。詳細は、「reVISION 入門ガイド」を参照してください。
- ZCU102 評価ボードを設定します。詳細は、「reVISION 入門ガイド」を参照してください。
- xfOpenCV ライブラリをダウンロードします。ライブラリは github から入手できます。次の `git clone` コマンドを実行し、xfOpenCV リポジトリをローカル ディスクにクローンします。

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

xfOpenCV ライブラリの内容

次の表に、xfOpenCV ライブラリの内容を示します。

フォルダー	詳細
include	ライブラリで必要なヘッダー ファイルが含まれます。
common	ライブラリ特有のタイプなど、共通するライブラリの構造ヘッダーが含まれます。
core	math 関数などのコア ライブラリの機能ヘッダーが含まれます。
features	特徴抽出カーネル関数定義が含まれます (例: Harris)。
imgproc	features フォルダーで提供されるものを除くすべてのカーネル関数定義が含まれます。
examples	ユニット テストを実行するサンプル テストベンチ コードが含まれます。examples/ フォルダーには、アルゴリズム名のフォルダーが含まれます。各アルゴリズム フォルダーには、ホスト ファイルの .json ファイル、data フォルダー、および include フォルダーが含まれます。

xfOpenCV ライブラリの使用

このセクションでは、SDx 開発環境での xfOpenCV ライブラリの使用方法を説明します。

注記: このセクションの手順は、必要なパッケージをすべてダウンロードおよびインストールしていることを前提としています。詳細は、使用要件を参照してください。

zcu102_es2_reVISION でバイラテラル フィルターのユニット テストを実行するには、次の手順に従います。

1. デスクトップ アイコンをダブルクリックするか [スタート] メニューを使用して、SDx 開発環境を起動します。

   [Workspace Launcher] ダイアログ ボックスが表示されます。

2. [Browse] をクリックしてプロジェクトを保存するワークスペース フォルダーを選択し、[OK] をクリックします。

   注記: Linux で SDx IDE を起動する前に、$SYSROOT 環境変数を設定したのと同じシェルを使用していることを確認します。これは通常、Linux ルート ファイル システムへのファイル パスです。

   SDx 開発環境のメイン ウィンドウが表示されます。新しいワークスペースを作成した場合は、[Welcome] タブが表示されます。[Welcome] タブは、[X] をクリックして閉じるか、[Minimize] アイコンをクリックして最小化できます。

3. SDx 開発環境のメニュー バーから [File] → [New] → [Xilinx SDx Project]. をクリックします。

   [New Project] ダイアログ ボックスが開きます。

4. プロジェクト名を指定します。たとえば、「Bilateral」と入力します。
5. [Next] をクリックします。

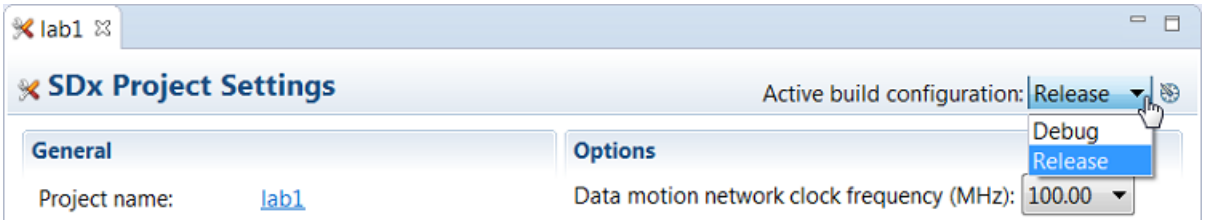
[Choose Hardware Platform] ページが開きます。

6. [Choose Hardware Platform] ページで [Add Custom Platform] をクリックします。
7. reVISION プラットフォーム ファイルを解凍したディレクトリに移動します。`zcu102_es2_reVISION` フォルダーを選択します。
8. [Choose Hardware Platform] ページで [zcu102_es2_reVISION (custom)] を選択します。
9. [Next] をクリックします。

   選択したプラットフォーム用のソース コード例をリストする [Templates] ページが表示されます。
10. [Available Templates] から [bilateral – File I/O] を選択し、[Finish] をクリックします。
11. [SDx Project Settings] で [Active build configurations] ドロップダウン リストをクリックしてアクティブ コンフィギュレーションを選択するか、ビルド コンフィギュレーションを作成します。

    標準ビルド コンフィギュレーションは [Debug] および [Release] です。最高のランタイム パフォーマンスを得るには、[Release] ビルド コンフィギュレーションを選択します。このビルド コンフィギュレーションでは、Debug ビルド コンフィギュレーションよりも高いコンパイラ最適化設定が使用されます。

図 2: [SDx Project Settings] – [Active build configuration] ドロップダウン リスト



12. [SDx Project Settings] で [Data motion network clock frequency (MHz)] を必要な周波数に設定します。
13. [Generate bitstream] および [Generate SD card Image] チェック ボックスをオンにします。
14. [Project Explorer] ビューでプロジェクトを右クリックして [Build Project] をクリックするか、Ctrl + B キーを押してプロジェクトを構築します。
15. 作成された `sd_card` フォルダーの内容を SD カードにコピーします。

    `sd_card` フォルダーには、`ZCU102` ボードでデザインを実行するのに必要なファイルがすべて含まれます。
16. `ZCU102` ボードのカード スロットに SD カードを挿入し、スイッチをオンにします。

    注記: ユーザー コマンドをボードに渡すには、シリアル ポート エミュレーター (Teraterm/Minicom) が必要です。
17. 正しくブートしたら、Teraterm ターミナル (シリアル ポート エミュレーター) で次のコマンドを実行します。

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

18. 対応する関数の `.elf` ファイルを実行します。

    詳細は、xfOpenCV ライブラリ関数のハードウェアでの使用を参照してください。

サンプル makefile を使用した Linux でのプロジェクト

の構築

Linux プラットフォームでサンプル makefile を使用してプロジェクトを構築するには、次の手順に従います。

1. ターミナルを開きます。
2. SYSROOT 環境変数を <path to revision platform on local machine>/sw/aarch64-linux-gnu/sysroot フォルダーに設定します。
3. <path to xfOpenCV git folder on local machine>/xfOpenCV/examples/bilateralfilter フォルダーから bilateralfilter サンプルを <path to revision platform on local machine>/sw/aarch64-linux-gnu/sysroot フォルダーにコピーします。
4. ディレクトリを <path to reVISION platform on local machine>/samples/bilateral/example に変更します。

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

5. SDx 開発環境を実行する環境変数を設定します。
   - C シェル:

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

   - bash シェル:

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

6. ターミナルに make コマンドを入力します。

   <path to reVISION platform on local machine>/samples/bilateral/examples フォルダーに sd_card フォルダーが作成されます。

ハードウェア カーネル コンフィギュレーションの変更

ハードウェア カーネル コンフィギュレーションを変更するには、次の手順に従います。

1. <path to xfOpenCV git folder>/xfOpenCV/examples/<function>/xf_config_params.h ファイルをアップデートします。
2. makefile と xf_config_params.h ファイルをアップデートします。
   a. makefile で関数名を含む行を見つけます。バイラテラル フィルターの場合、makefile 内の行は xFBilateralFilter<3,1,0,1080,1920,1> です。
   b. xf_config_params.h ファイルに加えた変更を反映するため makefile のテンプレート パラメーターをアップデートします。詳細は、xfOpenCV ライブラリ API リファレンスを参照してください。

xfOpenCV ライブラリ関数のハードウェアでの使用

次の表に、xfOpenCV ライブラリ関数とそれらのハードウェアでの使用方法を示します。

表 1: xfOpenCV ライブラリ関数のハードウェアでの使用

例	関数名	ハードウェアでの使用方法
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 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	xFnv122yuv4	./<executable name>.elf <path to input image

例	関数名	ハードウェアでの使用方法
NV122YUV4		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>
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

例	関数名	ハードウェアでの使用方法
		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 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>

例	関数名	ハードウェアでの使用方法
warpperspective	xFperspective	./<executable name>.elf <path to input image>
warptransform	xFWarpTransform	./<executable name>.elf <path to input image>

XILINX
ALL PROGRAMMABLE

第 3 章

xfOpenCV ライブラリ API リファレンス

FPGA デバイス上のローカル メモリ割り当てを円滑にするため、xfOpenCV ライブラリ関数はコンパイル時パラメーターを持つテンプレートとして提供されています。データは `cv::Mat` から `xF::Mat` に明示的にコピーされ、できるだけ高いパフォーマンスを達成するため物理的に隣接したメモリに保存されます。処理後、`xF::Mat` の出力が `cv::Mat` にコピーされ、メモリに書き出されます。

xF::Mat 画像コンテナー クラス

`xF::Mat` は、画像データとその属性を格納するコンテナーとして動作するテンプレート クラスです。

注記: `xF::Mat` 画像コンテナー クラスは OpenCV ライブラリの `cv::Mat` クラスと類似しています。

クラス定義

```
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
};
```

パラメーターの説明

次の表に、xF::Mat クラスのパラメーターと説明を示します。

表 2: xF::Mat クラスのパラメーターと説明

パラメーター	説明
rows	画像内の行数または画像の高さ。
cols	画像内の列数または画像の幅。
size	データ メンバーに格納されるワード数。この値は rows*cols/(number of pixels packed per word) で計算されます。
*data	画像のピクセルを格納するワードへのポインター。

メンバー関数の説明

次の表に、メンバー関数とその説明を示します。

表 3: xF::Mat メンバー関数の説明

メンバー関数	説明
Mat()	デフォルトのコンストラクター。ROWS および COLS テンプレート パラメーターを使用して Mat オブジェクトのサイズを初期化します。
Mat(int _rows, int _cols)	_rows および _cols 引数を使用して Mat オブジェクトを初期化します。
Mat(int _rows, int _cols, void *_data)	_rows、_cols、および _data.引数を使用して Mat オブジェクトを初期化します。このコンストラクターを使用すると、Mat オブジェクトの *data メンバーで _data 引数に割り当てられたメモリが指定されます。*data メンバー用に新しいメモリが割り当てられることはありません。
copyTo(*fromData)	データ ポインターからのデータをコンストラクター内で割り当てられた物理的に隣接したメモリにコピーします。
copyFrom()	ポインターを *data メンバーの最初の位置に戻します。
~Mat()	Mat オブジェクトのデフォルト デストラクターです。

テンプレート パラメーターの説明

xF::Mat クラスのテンプレート パラメーターは、ピクセルの深さ、画像のチャネル数、各ワードごとにパックされるピクセル数、画像の行および列の最大数を設定するのに使用します。次の表に、テンプレート パラメーターとその説明を示します。

表 4: xF::Mat テンプレート パラメーターの説明

パラメーター	説明
TYPE	ピクセル データのタイプ。たとえば、XF_8UC1 は 8 ビットの符号なしの 1 チャネル ピクセルを意味します。その他のタイプは includes/common/xf_params.h を参照してください。
HEIGHT	画像の最大高さ。
WIDTH	画像の最大幅。
NPC	ワードごとにパックされるピクセル数。たとえば、XF_NPPC1 は毎ワード 1 ピクセル、XF_NPPC8 は毎ワード 8 ピクセルを意味します。

ピクセル レベルの並列処理

xfOpenCV から関数にインプリメントされる並列処理の量は、設定可能なパラメーターです。ほとんどの関数では、2 つのデータ処理オプションがあります。

- 1 ピクセル処理
- 8 ピクセルの並列処理

次の表に、特定の関数で必要な並列処理のレベルを指定するオプションを示します。

表 5: 並列処理のレベルを指定するのに使用できるオプション

オプション	説明
XF_NPPC1	クロック サイクルごとに 1 ピクセルを処理
XF_NPPC2	クロック サイクルごとに 2 ピクセルを処理
XF_NPPC8	クロック サイクルごとに 8 ピクセルを処理

並列処理に使用するマクロ

次の 2 つのマクロは、並列処理で使用できるよう定義されています。

- XF_NPIXPERCYCLE(flags) マクロ: サイクルごとに処理するピクセルの数を返します。
  - XF_NPIXPERCYCLE(XF_NPPC1) は 1 を返します。
  - XF_NPIXPERCYCLE(XF_NPPC2) は 2 を返します。
  - XF_NPIXPERCYCLE(XF_NPPC8) は 8 を返します。

- XF_BITSHIFT(flags) マクロ: 並列処理で最終画像データ画像サイズを得るために画像サイズを右にシフトする回数を返します。
  - XF_BITSHIFT(XF_NPPC1) は 0 を返します。
  - XF_BITSHIFT(XF_NPPC2) は 1 を返します。
  - XF_BITSHIFT(XF_NPPC8) は 3 を返します。

ピクセル タイプ

パラメーター タイプは、画像内のピクセルの深さおよびチャネル数の組み合わせによって異なります。このパラメーターの一般的な命名方法は、次のとおりです。

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

たとえば、8 ビット ピクセル、符号なし、1 チャネルのデータ型は XF_8UC1 です。

次の表に、xF::Mat クラスの使用可能なデータ型を示します。

表 6: xf::Mat クラス – 使用可能なデータ型

オプション	ピクセルごとのビット数	符号なし/符号付き/浮動小数点型	チャネル数
XF_8UC1	8	符号なし	1
XF_16UC1	16	符号なし	1
XF_16SC1	16	符号付き	1
XF_32UC1	32	符号なし	1
XF_32FC1	32	浮動小数点	1
XF_32SC1	32	符号付き	1
XF_8UC2	8	符号なし	2
XF_8UC4	8	符号なし	4
XF_2UC1	2	符号なし	1

データ型の操作

クロック サイクルごとに処理するピクセル数と型パラメーターに基づき、異なるデータ型を使用できます。xfOpenCV ライブラリでは、これらのデータ型が内部処理および xF::Mat クラス内で使用されます。次に、サポートされるデータ型をいくつか示します。

- XF_TNAME(TYPE,NPPC): xF::Mat オブジェクトのデータ メンバーのデータ型を返します。たとえば、XF_TNAME(XF_8UC1,XF_NPPC8) は ap_uint<64> を返します。
- ワード幅 = ピクセルの深さ * チャネル数 * サイクルごとに処理するピクセル数 (NPPC)
- XF_DTUNAME(TYPE,NPPC): ピクセルのデータ型を返します。たとえば、XF_DTUNAME(XF_32FC1,XF_NPPC1) は float を返します。
- XF_PTSNAME(TYPE,NPPC): ピクセルの c データ型を返します。たとえば、XF_PTSNAME (XF_16UC1,XF_NPPC2) は short を返します。

注記: ap_uint<>、ap_int<>、ap_fixed<>、および ap_ufixed<> 型は、高位合成 (HLS) ライブラリに含まれます。詳細は、『Vivado Design Suite ユーザー ガイド: 高位合成』(UG902) を参照してください。

コード例

次のコードは、Zynq® UltraScale™ プラットフォームの場合に SDSoC ツールを使用して、画像にガウシアン フィルターを構築するのに必要な設定を示しています。

注記: ビデオがストリーム入力されるリアルタイム アプリケーションの場合、フレーム バッファーの位置が xF::Mat でライブラリ関数を使用して処理されるようにしてください。結果の位置ポインターが渡されて IP が表示されるようになります。

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
```

```
}
```

データが SDSoC データ ムーバーを使用して外部メモリからフェッチされて、設定モードによって 8 ビットまたは 64 ビット パケットの関数に転送されます。毎ピクセル 8 ビットの場合、8 ピクセルを 64 ビットにパックできます。この結果、8 ピクセルが並列処理に使用できるようになります。

xf_config_params.h ファイルで FILTER_SIZE_3_1 と NO マクロをイネーブルにします。#define FILTER_SIZE_3 1 マクロでフィルター サイズが 3x3 に設定され、#define NO 1 マクロで 1 ピクセルの並列処理がイネーブルになります。

xf_gaussian_filter.hpp ファイルで SDSoC 特有のプラグマを指定します。

```
#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"])
```

注記: SDSoC のハードウェア アクセラレータ関数で使用されるプラグマについては、『SDSoC 環境ユーザー ガイド』(UG1027) を参照してください。

xfOpenCV ライブラリ関数

xfOpenCV ライブラリは、Zynq-7000 All Programmable SoC および Zynq UltraScale+ MPSoC デバイス用に最適化された OpenCV 関数のセットです。次の表に、サポートされる xfOpenCV ライブラリ関数を示します。

計算	入力処理	フィルター	その他
絶対差分	ビット深度変換	バイラテラル フィルター	Canny 法によるエッジ検出
累算	チャネル結合	ボックス フィルター	FAST コーナー検出
累積二乗	チャネル抽出	カスタムたたみ込み	Harris コーナー検出
累積重み	色変換	膨張	ヒストグラム計算
Atan2	ヒストグラム均一化	収縮	高密度ピラミッド型 LK オプティカル フロー
ビット単位 AND、ビット単位 NOT、ビット単位 OR, ビット単位 XOR	ルックアップ テーブル	ガウシアン フィルター	高密度非ピラミッド型 LK オプティカル フロー
勾配の大きさ	リマップ	Sobel フィルター	MinMax ロケーション
勾配位相	解像度変換/リサイズ	メジアン フィルター	しきい値処理
積分画像		Scharr フィルター	WarpAffine
インバース/逆数			WarpPerspective
ピクセル加算			SVM
ピクセル乗算			Otsu 法のしきい値処理
ピクセル減算			平均値シフト トラッキング
平方根			HOG
平均および標準偏差			ステレオ ローカル ブロック マッチング
			WarpTransform
			ピラミッド アップ
			ピラミッド ダウン

次の表の関数は、サイクルごとに 8 ピクセル モードで 128 ビット インターフェイスを使用するように設定した場合、Zynq-7000 All Programmable SoC デバイスでサポートされません。

計算	入力処理	フィルター
累算	ビット深度変換	ボックス フィルター: 符号付き 16 ビット ピクセル タイプおよび符号なし 16 ビット ピクセル タイプ
累積二乗		カスタムたたみ込み: 符号付き 16 ビット出力ピクセル タイプ
累積重み (xFaccumulateWeighted)		Sobel フィルター
勾配の大きさ		Scharr フィルター
勾配位相		
ピクセル加算: 符号付き 16 ビット ピクセル タイプおよび符号なし 16 ビット ピクセル タイプ		
ピクセル乗算: 符号付き 16 ビット ピクセル タイプおよび符号なし 16 ビット ピクセル タイプ		
ピクセル減算: 符号付き 16 ビット ピクセル タイプおよび符号なし 16 ビット ピクセル タイプ		

絶対差分 (xFabsdiff)

xFabsdiff 関数は、2 つの入力画像のピクセル単位の違いを検出し、出力画像を返します。入力画像および出力画像は、XF_8UC1 型である必要があります。

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

説明:

- $I_{out}(x, y)$: 出力画像の (x,y) 位置での強度。
- $I_{in1}(x, y)$: 1 つ目の入力画像の (x,y) 位置での強度。
- $I_{in2}(x, y)$: 2 つ目の入力画像の (x,y) 位置での強度。

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 7: xFabsdiff 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado® HLS 2017.1 ツールを使用して生成された異なる設定でのリソース使用量を示します。

表 8: xFabsdiff 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	62	67	17
8 ピクセル	150	0	0	67	234	39

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 9: xFabsdiff 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.69

OpenCV との違い

xFabsdiff で 8 ビット ピクセルがサポートされることを除き、OpenCV との違いはありません。

累積 (xFaccumulate)

xFaccumulate 関数は、画像 (src1) をアキュムレータ画像 (src2) に追加し、累積結果 (dst) を生成します。

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

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 10: xFaccumulate 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのリソース使用量を示します。

表 11: xFaccumulate 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	62	55	12
8 ピクセル	150	0	0	389	285	61

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 12: xFaccumulate 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

OpenCV との違い

OpenCV では、累積画像は 2 番目の入力画像に格納されます。src2 画像は、次に示すように入力および出力の両方として使用されます。

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

xfOpenCV インプリメンテーションでは、累積画像は次に示すように別に格納されます。

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

累積二乗 (xFaccumulateSquare)

`xFaccumulateSquare` 関数は、画像 (src1) をアキュムレータ画像 (src2) に追加し、累積結果 (dst) を生成します。

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

src2 を累積結果とするのではなく、別の引数が使用されます。このインプリメンテーションではストリームが使用されるので、双方向アキュムレータは使用できません。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 13: xFaccumulateSquare 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのリソース使用量を示します。

表 14: xFaccumulateSquare 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	1	71	52	14
8 ピクセル	150	0	8	401	247	48

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 15: xFaccumulateSquare 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.6

OpenCV との違い

OpenCV では、累積二乗画像は 2 番目の入力画像に格納されます。src2 画像は入力および出力の両方として使用されます。

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

xfOpenCV インプリメンテーションでは、累積二乗画像は別に格納されます。

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

累積重み (xFaccumulateWeighted)

`xFaccumulateWeighted` 関数は、入力画像 (src1) とアキュムレータ画像 (src2) の重みの和を計算し、結果を dst に生成します。

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

src2 を累積結果とするのではなく、別の引数が使用されます。このインプリメンテーションではストリームが使用されるので、双方向アキュムレータは使用できません。

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 16: xFaccumulateWeighted 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像
alpha	入力画像に適用される重み

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのリソース使用量を示します。

表 17: xFaccumulateWeighted 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	5	295	255	52
8 ピクセル	150	0	19	556	476	88

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 18: xFaccumulateWeighted 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

OpenCV との違い

OpenCV の結果の画像が 2 つ目の入力画像に格納されます。src2 画像は次に示すように入力および出力の両方として使用されます。

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

xfOpenCV でインプリメンテーションでは、累積重み画像は別に格納されます。

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

バイラテラル フィルター (xFBilateralFilter)

通常、平滑化フィルターは画像のエッジに影響する画像を平滑化します。平滑化処理中にエッジを保持するには、バイラテラル フィルターを使用できます。バイラテラル フィルターでは、ガウシアン フィルターと同様に、近傍ピクセルが考慮され、各ピクセルには重みが設定されます。これらの重みには 2 つの要素があります。1 つ目の要素は、ガウシアン フィルターで使用される重みと同じです。2 つ目の要素では、近傍ピクセルと評価されるピクセルの強度の違いが考慮されます。

画像に適用されるバイラテラル フィルターは、次のとおりです。

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

ここで、

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

および G_σ は分散 σ のガウシアン フィルターです。

ガウシアン フィルターは、次の式で求められます: $G_\sigma = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 19: xFBilateralFilter 関数パラメーターの説明

パラメーター	説明
FILTER_SIZE	フィルター サイズ。サポートされるフフィルター サイズは 3 (XF_FILTER_3X3)、5 (XF_FILTER_5X5)、および 7 (XF_FILTER_7X7)。
BORDER_TYPE	サポートされる境界タイプは XF_BORDER_CONSTANT。
TYPE	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。XF_NPPC1 (サイクルごとに 1 ピクセルの操作) のみサポート。
src	入力画像
dst	出力画像
sigma_space	空間領域のフィルターの標準偏差
sigma_color	色空間で使用されるフィルターの標準偏差

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 20: xFBilateralFilter 関数のリソース使用量のサマリ

動作モード	フィルター サイズ	動作周波数 (MHz)	使用量の見積もり			
			BRAM_18K	DSP_48E	FF	LUT
1 ピクセル	3x3	300	6	22	4934	4293
	5x5	300	12	30	5481	4943
	7x7	300	37	48	7084	6195

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 21: xFBilateralFilter 関数のパフォーマンス見積もりのサマリ

動作モード	フィルター サイズ	レイテンシ見積もり
		168 MHz
		最大 (ms)
1 ピクセル	3x3	7.18
	5x5	7.20
	7x7	7.22

OpenCV との違い

OpenCV とは異なり、xfOpenCV ではフィルター サイズ 3、5、および 7 のみがサポートされます。

ビット深度変換 (xFConvertBitDepth)

xFConvertBitDepth 関数は、入力画像のビット深度を出力画像で必要なビット深度に変換します。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 22: xFConvertBitDepth 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1)、 16 ビット、符号なし、1 チャネル (XF_16UC1)、 16 ビット、符号付き、1 チャネル (XF_16SC1)、 32 ビット、符号なし、1 チャネル (XF_32UC1)、 32 ビット、符号付き、1 チャネル (XF_32SC1) をサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1)、 16 ビット、符号なし、1 チャネル (XF_16UC1)、 16 ビット、符号付き、1 チャネル (XF_16SC1)、 32 ビット、符号なし、1 チャネル (XF_32UC1)、 32 ビット、符号付き、1 チャネル (XF_32SC1) をサポート。
ROWS	入力および出力画像の高さ。
COLS	入力および出力画像の幅。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_mat	出力画像
_convert_type	必要な変換タイプを指定。有効な値は、`xf_params.h` ファイルの XF_convert_bit_depth_e 列挙型を参照。
_shift	オプションのスケール係数

可能な変換

次の表は、サポートされる変換を示しています。可能な入力画像ビット深度を行に、可能な出力画像ビット深度を列に示します (U = 符号なし、S = 符号付き)。

表 23: xFConvertBitDepth 関数のサポートされる変換

入力/出力	U8	U16	S16	U32	S32
U8	なし	可	可	なし	可
U16	可	なし	なし	なし	可
S16	可	なし	なし	なし	可
U32	なし	なし	なし	なし	なし
S32	可	可	可	なし	なし

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された xFConvertBitDepth 関数のリソース使用量を示します。

表 24: XF_CONVERT_8U_TO_16S 変換のための xFConvertBitDepth 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	8	581	523	119
8 ピクセル	150	0	8	963	1446	290

表 25: XF_CONVERT_16U_TO_8U 変換のための xFConvertBitDepth 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	8	591	541	124
8 ピクセル	150	0	8	915	1500	308

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 26: xFConvertBitDepth 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ
1 ピクセル動作 (300 MHz)	6.91 ms
8 ピクセル動作 (150 MHz)	1.69 ms

ビット単位 AND (xFbitwise_and)

xFbitwise_and 関数は、2 つの画像の各ピクセルに対してビット単位の AND 演算を実行し、出力画像を返します。

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

説明:

- $I_{out}(x, y)$: 出力画像の (x, y) 位置での強度
- $I_{in1}(x, y)$: 1 つ目の入力画像の (x,y) 位置での強度
- $I_{in2}(x, y)$: 2 つ目の入力画像の (x,y) 位置での強度

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 27: xFbitwise_and 関数パラメーターの説明

パラメーター	説明
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのリソース使用量を示します。

表 28: xFbitwise_and 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	62	44	10
8 ピクセル	150	0	0	59	72	13

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 29: xFbitwise_and 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

ビット単位 NOT (xFbitwise_not)

`xFbitwise_not` 関数は、入力画像のピクセルに対してビット単位の NOT 演算を実行し、出力画像を返します。 $I_{out}(x, y) = \sim I_{in}(x, y)$

説明:

- $I_{out}(x, y)$: 出力画像の (x, y) 位置での強度
- $I_{in}(x, y)$: 入力画像の (x,y) 位置での強度

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 30: xFbitwise_not 関数パラメーターの説明

パラメーター	説明
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src	入力画像
dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのリソース使用量を示します。

表 31: xFbitwise_not 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	97	78	20
8 ピクセル	150	0	0	88	97	21

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 32: xFbitwise_not 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

ビット単位 OR (xFbitwise_or)

`xFbitwise_or` 関数は、2 つの入力画像のビット単位の OR 演算を実行し、出力画像を返します。

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

説明:

- $I_{out}(x, y)$: 出力画像の (x, y) 位置での強度
- $I_{in1}(x, y)$: 1 つ目の入力画像の (x,y) 位置での強度
- $I_{in2}(x, y)$: 2 つ目の入力画像の (x,y) 位置での強度

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 33: xFbitwise_or 関数パラメーターの説明

パラメーター	説明
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのリソース使用量を示します。

表 34: xFbitwise_or 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	62	44	10
8 ピクセル	150	0	0	59	72	13

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 35: xFbitwise_or 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

ビット単位 XOR (xFbitwise_xor)

`xFbitwise_xor` 関数は、2 つの入力画像のビット単位の XOR 演算を実行し、出力画像を返します。

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

説明:

- $I_{out}(x, y)$: 出力画像の (x, y) 位置での強度
- $I_{in1}(x, y)$: 1 つ目の入力画像の (x,y) 位置での強度
- $I_{in2}(x, y)$: 2 つ目の入力画像の (x,y) 位置での強度

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 36: xFbitwise_xor 関数パラメーターの説明

パラメーター	説明
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのリソース使用量を示します。

表 37: xFbitwise_xor 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	62	44	10
8 ピクセル	150	0	0	59	72	13

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 38: xFbitwise_xor 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

ボックス フィルター (xFboxfilter)

xFboxfilter 関数は、入力画像に対してボックス フィルター処理を実行します。ボックス フィルターローパス フィルターとして機能し、画像にぼかしを適用します。xFboxfilter 関数 (ボックスぼかし) は空間領域線形フィルターで、処理後の画像の各ピクセルは近傍ピクセルの平均値となります。

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 39: xFboxfilter 関数パラメーターの説明

パラメーター	説明
FILTER_SIZE	フィルター サイズ。サポートされるフィルター サイズは 3 (XF_FILTER_3X3)、5 (XF_FILTER_5X5)、および 7 (XF_FILTER_7X7)。
BORDER_TYPE	サポートされる境界タイプは XF_BORDER_CONSTANT。
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、16 ビット符号なし、および 16 ビット符号付き 1 チャネル (XF_8UC1) をサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_mat	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 40: xFboxfilter 関数のリソース使用量のサマリ

動作モード	フィルター サイズ	動作周波数 (MHz)	使用量の見積もり				
			BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	3x3	300	3	1	545	519	104
	5x5	300	5	1	876	870	189
	7x7	300	7	1	1539	1506	300
8 ピクセル	3x3	150	6	8	1002	1368	264
	5x5	150	10	8	1576	3183	611
	7x7	150	14	8	2414	5018	942

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 41: xFboxfilter 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	フィルター サイズ	レイテンシ見積もり 最大 (ms)
1 ピクセル	300	3x3	7.2
	300	5x5	7.21
	300	7x7	7.22
8 ピクセル	150	3x3	1.7
	150	5x5	1.7
	150	7x7	1.7

Canny 法によるエッジ検出 (xFcanny)

Canny 法によるエッジ検出では、画像またはビデオ フレームのエッジを検出します。エッジ検出で最もよく使用されるアルゴリズムの 1 つです。Canny アルゴリズムでは、次の 3 つの主な条件を満たすことが目的とされます。

1. 低エラー レート: 既存のエッジのみを検出。
2. 局所化: 検出されたエッジ ピクセルと実際のエッジ ピクセルとの距離を最小化。
3. 最低限の応答: エッジごとに 1 つの検出器のみが応答。

このアルゴリズムでは、まずガウシアン マスクが適用されて画像のノイズが削減されます。ここで使用されるガウシアン マスクは、3x3 サイズの平均マスクです。その後、Sobel 勾配関数を使用して x および y 方向の勾配が計算されます。勾配は、ピクセルの大きさと位相を計算するために使用されます。位相は量子化され、それに応じてピクセルがビンに分類されます。ピクセルに NMS (Non-Maximal Suppression) が適用され、弱いエッジは除去されます。

残りのエッジにエッジ トレースが適用され、画像のエッジが描画されます。このアルゴリズムでは、Canny から NMS までは 1 つのカーネルに含まれ、エッジ リンキング モジュールは別のカーネルに含まれます。NMS の後、出力はピクセルごとに 2 ビットで表されます。

- 00: 背景
- 01: 弱いエッジ
- 11: 強いエッジ

出力は、サイクルごとに 1 ピクセルの操作では 8 ビット (4 つの 2 ビット ピクセル) としてパックされ、サイクルごとに 8 ピクセルの操作では 16 ビット (8 つの 2 ビット ピクセル) としてパックされます。エッジ リンキング モジュールでは、入力は 64 ビットであり、32 個の 2 ビット ピクセルが 64 ビットにパックされます。ピクセルにエッジ トレースが適用され、画像のエッジが返されます。

API 構文

xFcanny の API 構文は次のとおりです。

```
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)
```

xFEdgeTracing の API 構文は次のとおりです。

```
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)
```

パラメーターの説明

次の表に、xFcanny のテンプレートと関数パラメーターを説明します。

表 42: xfcanny 関数パラメーターの説明

パラメーター	説明
FILTER_TYPE	フィルター ウィンドウの大きさ。有効な値は 3 および 5。
NORM_TYPE	使用するノルムのタイプ。有効なノルム タイプは L1NORM および L2NORM。
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセル タイプ。1 ピクセル操作での出力は 8 ビットで、4 つの 2 ビット ピクセル値が 8 ビットにパックされます。8 ピクセル操作での出力は 16 ビットで、8 つの 2 ビット ピクセル値が 16 ビットにパックされます。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_mat	出力画像
_lowthreshold	バイナリしきい値処理の下限しきい値。
_highthreshold	バイナリしきい値処理の上限しきい値。

次の表に、xFEdgeTracing のテンプレートと関数パラメーターを説明します。

表 43: xFEdgeTracing 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ
DST_T	出力ピクセル タイプ
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC_SRC	サイクルごとに処理されるピクセル数。XF_NPPC32 に固定。
NPC_DST	処理後のピクセル数。XF_NPPC8 に固定。
_src	入力画像
_dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像をフィルター サイズ 3 で処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での xFcanny および xFEdgeTracing のリソース使用量を示します。

表 44: xFcanny および xFEdgeTracing 関数のリソース使用量のサマリ

名前	リソース使用量					
	1 ピクセル	1 ピクセル	8 ピクセル	8 ピクセル	エッジ リンキング	エッジ リンキング
	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

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を L1NORM、フィルター サイズ 3、エッジ リンキング モジュールを使用して処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 45: xFcanny および xFEdgeTracing 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり	
	動作周波数 (MHz)	レイテンシ (ms)
1 ピクセル	300	10.2
8 ピクセル	150	8

OpenCV との違い

ガウシアン フィルターは、OpenCV には含まれません。エッジ リンキング モジュールは xfOpenCV ライブラリには含まれません。

チャネル結合 (xFmerge)

`xFmerge` 関数では、シングルチャネル画像がマルチチャネル画像に統合されます。統合されるチャネル数は 4 です。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 46: xFmerge 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC4) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合、可能なオプションは XF_NPPC1。
_src1	入力シングルチャネル画像
_src2	入力シングルチャネル画像
_src3	入力シングルチャネル画像
_src4	入力シングルチャネル画像
_dst	出力マルチチャネル画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で 4 つのシングル チャネル HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された xFmerge 関数のリソース使用量を示します。

表 47: xFmerge 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	8	494	386	85

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 で 4 つのシングル チャネル HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 48: xFmerge 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ
1 ピクセル動作 (300 MHz)	6.92 ms

チャネル抽出 (xFextractChannel)

xFextractChannel 関数は、マルチチャネル配列 (32 ビットのピクセルがインターリーブされたデータ) を複数シングルチャネル配列に分割し、シングル チャネルを返します。抽出されるチャネルは、チャネル引数を使用して指定します。

チャネル引数の値は、xf_channel_extract_e 列挙データ型で定義されたマクロにより指定します。次の表に、xf_channel_extract_e 列挙データ型に可能な値を示します。

表 49: xf_channel_extract_e 列挙データ型の値

チャネル	列挙型
不明	XF_EXTRACT_CH_0
不明	XF_EXTRACT_CH_1
不明	XF_EXTRACT_CH_2
不明	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 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 50: xFextractChannel 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合、可能なオプションは XF_NPPC1。
_src_mat	入力マルチチャネル画像
_dst_mat	出力シングルチャネル画像
_channel	抽出するチャネル (有効な値は、`xf_params.h` ファイルの xf_channel_extract_e 列挙型を参照)

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で 4 チャネル HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された xfextractChannel 関数のリソース使用量を示します。

表 51: xFextractChannel 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	8	508	354	96

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 で 4 チャネル HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 52: xFextractChannel 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.92

色変換

色変換関数は、ある画像フォーマットを別の画像フォーマットに変換します。次の表に、変換可能な画像フォーマットの組み合わせを示します。行が入力フォーマットで、列が出力フォーマットです。次のセクションで、サポートされる変換について説明します。

表 53: サポートされる色変換

入力/出力フォーマット	RGBA	NV12	NV21	IYUV	UYVY	YUYV	YUV4
RGBA	N/A	RGBA to NV12 (xFrgba2nv12) を参照	RGBA to NV21 (xFrgba2nv21) を参照	RGBA to IYUV (xFrgba2iyuv) を参照			RGBA to YUV4 (xFrgba2yuv4) を参照
NV12	NV12 to RGBA (xFnv122rgba) を参照	N/A		NV12 to IYUV (xFnv122iyuv) を参照			NV12 to YUV4 (xFnv122yuv4) を参照
NV21	NV21 to RGBA (xFnv212rgba) を参照		N/A	NV21 to IYUV (xFnv212iyuv) を参照			NV21 to YUV4 (xFnv212yuv4) を参照
IYUV	IYUV to RGBA (xFiyuv2rgba) を参照	IYUV to NV12 (xFiyuv2nv12) を参照		N/A			IYUV to YUV4 (xFiyuv2yuv4) を参照
UYVY	UYVY to RGBA (xFuyvy2rgba) を参照	UYVY to NV12 (xFuyvy2nv12) を参照		UYVY to IYUV (xFuyvy2iyuv) を参照	N/A		
YUYV	YUYV to RGBA (xFyuyv2rgba) を参照	YUYV to NV12 (xFyuyv2nv12) を参照		YUYV to IYUV (xFyuyv2iyuv) を参照		N/A	
YUV4							N/A

RGB から YUV への変換行列

次に、RGB データから YUV データへの変換式を示します。

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

YUV から RGB への変換行列

次に、YUV データから RGB データへの変換式を示します。

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

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

RGBA to YUV4 (xFrgba2yuv4)

`xFrgba2yuv4` は、4 チャネルの RGBA 画像を YUV444 フォーマットに変換します。出力は Y、U、および V ストリームです。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 54: xFrgba2yuv4 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)。
COLS	入力および出力画像の最大幅 (8 の倍数で指定)。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) の入力 Y プレーン。
_y_image	サイズ (ROWS, COLS) の出力 Y 画像。
_u_image	サイズ (ROWS, COLS) の出力 U 画像。
_v_image	サイズ (ROWS, COLS) の出力 V 画像。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での RGBA to YUV4 のリソース使用量を示します。

表 55: xFrgba2yuv4 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	9	589	328	96

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での RGBA to YUV4 のパフォーマンス見積もりを示します。

表 56: xFrgba2yuv4 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	1.89

RGBA to IYUV (xFrgba2iyuv)

xFrgba2iyuv 関数は、4 チャネルの RGBA 画像を IYUV (4:2:0) フォーマットに変換します。出力は Y、U、および V プレーンです。IYUV にはサブサンプリングされたデータが格納されるので、Y は RGBA ピクセルごとにサンプリングされ、U と V は 2 行および 2 列 (2x2) のピクセルごとにサンプリングされます。U および V プレーンのサイズは (rows/2)*(columns/2) であり、連続する行を 1 つの行にカスケード接続することにより、プレーンのサイズが (rows/4)*columns になります。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 57: xFrgba2iyuv 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) の入力 Y プレーン。
_y_image	サイズ (ROWS, COLS) の出力 Y 画像。
_u_image	サイズ (ROWS/4, COLS) の出力 U 画像。
_v_image	サイズ (ROWS/4, COLS) の出力 V 画像。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での RGBA to IYUV のリソース使用量を示します。

表 58: xFrgba2iyuv 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	9	816	472	149

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での RGBA to IYUV のパフォーマンス見積もりを示します。

表 59: xFrgba2iyuv 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	1.8

RGBA to NV12 (xFrgba2nv12)

`xFrgba2nv12` 関数は、4 チャネルの RGBA 画像を NV12 (4:2:0) フォーマットに変換します。出力は Y プレーンおよびインターリーブされた UV プレーンです。NV12 にはサブサンプリングされたデータが格納されるので、Y は RGBA ピクセルごとにサンプリングされ、U および V は 2 行と 2 列 (2x2) のピクセルごとにサンプリングされます。UV プレーンのサイズは (rows/2)*(columns/2) で、U と V の値がインターリーブされます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 60: xFrgba2nv12 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
Y_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	出力ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) の入力 RGBA 画像。
_y	サイズ (ROWS, COLS) の出力 Y 画像。
_uv	サイズ (ROWS/2, COLS/2) の出力 UV 画像。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での RGBA to NV12 のリソース使用量を示します。

表 61: xFrgba2nv12 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	9	802	452	128

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での RGBA to NV12 のパフォーマンス見積もりを示します。

表 62: xFrgba2nv12 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	1.8

RGBA to NV21 (xFrgba2nv21)

xFrgba2nv21 関数は、4 チャネルの RGBA 画像を NV21 (4:2:0) フォーマットに変換します。出力は Y プレーンおよびインターリーブされた VU プレーンです。NV21 にはサブサンプリングされたデータが格納されるので、Y は RGBA ピクセルごとにサンプリングされ、U および V は 2 行と 2 列 (2x2) の RGBA ピクセルごとにサンプリングされます。UV プレーンのサイズは (rows/2)*(columns/2) で、V と U の値がインターリーブされます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 63: xFrgba2nv21 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
Y_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)。
COLS	入力および出力画像の最大幅 (8 の倍数で指定)。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) の入力 RGBA 画像。
_y	サイズ (ROWS, COLS) の出力 Y 画像。
_uv	サイズ (ROWS/2, COLS/2) の出力 UV 画像。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での RGBA to NV21 のリソース使用量を示します。

表 64: xFrgba2nv21 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	9	802	453	131

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での RGBA to NV21 のパフォーマンス見積もりを示します。

表 65: xFrgba2nv21 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	1.89

YUYV to RGBA (xFyuyv2rgba)

xFyuyv2rgba 関数は、1 チャネルの YUYV (YUV 4:2:2) の画像フォーマットを 4 チャネルの RGBA 画像に変換します。YUYV はサブサンプリング フォーマットで、RGBA が 2 つ得られます。YUYV は 16 ビット値、RGBA は 32 ビット値です。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 66: xFyuyv2rgba 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) の入力画像
_dst	サイズ (ROWS, COLS) の出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での YUYV to RGBA のリソース使用量を示します。

表 67: xFyuyv2rgba 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 ピクセル	300	0	6	765	705	165

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での UYVY to RGBA のパフォーマンス見積もりを示します。

表 68: xFyuyv2rgba 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9

YUYV to NV12 (xFyuyv2nv12)

`xFyuyv2nv12` 関数は、1 チャネルの YUYV (YUV 4:2:2) の画像フォーマットを NV12 (YUV 4:2:0) フォーマットに変換します。YUYV はサブサンプリング フォーマットで、YUYV 値の 1 セットから Y 値が 2 つと U 値および V 値が 1 つずつ得られます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 69: xFyuyv2nv12 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
Y_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	出力 UV 画像ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
NPC_UV	サイクルごとに処理される UV 画像ピクセル数。1 ピクセルの操作の場合は XF_NPPC1、8 ピクセルの操作の場合は XF_NPPC4。
_src	サイズ (ROWS, COLS) の入力画像
_y_image	サイズ (ROWS, COLS) 出力 Y プレーン
_uv_image	サイズ (ROWS/2, COLS/2) の出力 U プレーン

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での YUYV to NV12 のリソース使用量を示します。

表 70: xFyuyv2nv12 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48Es	FF	LUT	CLB
1 ピクセル	300	0	0	831	491	149
8 ピクセル	150	0	0	1196	632	161

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での YUYV to NV12 のパフォーマンス見積もりを示します。

表 71: xFyuyv2nv12 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

YUYV to IYUV (xFyuyv2iyuv)

xFyuyv2iyuv は、1 チャネルの YUYV (YUV 4:2:2) の画像フォーマットを IYUV(4:2:0) フォーマットに変換します。関数の出力は、Y、U、V プレーンです。YUYV はサブサンプリング フォーマットで、YUYV 値の 1 セットから Y 値が 2 つと U 値および V 値が 1 つずつ得られます。U および V 値は IYUV(4:2:0) フォーマットで 2 行と 2 列 (2x2) ごとにサンプリングされるので、奇数行の U および V 値は破棄されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 72: xFyuyv2iyuv 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) の入力画像
_y_image	サイズ (ROWS, COLS) 出力 Y プレーン
_u_image	サイズ (ROWS/4, COLS) の出力 U プレーン
_v_image	サイズ (ROWS/4, COLS) の出力 V プレーン

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での YUYV to IYUV のリソース使用量を示します。

表 73: xFyuyv2iyuv 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	835	497	149
8 ピクセル	150	0	0	1428	735	210

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での YUYV to IYUV のパフォーマンス見積もりを示します。

表 74: xFyuyv2iyuv 関数のパフォーマンス見積もり

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

UYVY to IYUV (xFuyvy2iyuv)

xFuyvy2iyuv 関数は、UYVY (YUV 4:2:2) の 1 チャネル画像を IYUV フォーマットに変換します。出力は Y、U、V プレーンです。UYVY はサブサンプリング フォーマットです。UYVY 値の 1 セットから Y 値が 2 つと U 値および V 値が 1 つずつ得られます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 75: xFuyvy2iyuv 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)。
COLS	入力および出力画像の最大幅 (8 の倍数で指定)。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) の入力画像。
_y_image	サイズ (ROWS, COLS) 出力 Y プレーン。
_u_image	サイズ (ROWS/4, COLS) の出力 U プレーン。
_v_image	サイズ (ROWS/4, COLS) の出力 V プレーン。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での UYVY to IYUV のリソース使用量を示します。

表 76: xFuyvy2iyuv 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	835	494	139
8 ピクセル	150	0	0	1428	740	209

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での UYVY to IYUV のパフォーマンス見積もりを示します。

表 77: xFuyvy2iyuv 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

UYVY to RGBA (xFuyvy2rgba)

xFuyvy2rgba 関数は、UYVY (YUV 4:2:2) の 1 チャネル画像を 4 チャネルの RGBA 画像に変換します。UYVY はサブサンプリング フォーマットで、UYVY 値の 1 セット値から 2 つの RGBA ピクセル値が得られます。UYVY は 16 ビット値、RGBA は 32 ビット値です。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 78: xFuyvy2rgba 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)。
COLS	入力および出力画像の最大幅 (8 の倍数で指定)。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) の入力画像。
_dst	サイズ (ROWS, COLS) の出力画像。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での UYVY to RGBA のリソース使用量を示します。

表 79: xFuyvy2rgba 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	6	773	704	160

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での UYVY to RGBA のパフォーマンス見積もりを示します。

表 80: xFuyvy2rgba 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.8

UYVY to NV12 (xFuyvy2nv12)

`xFuyvy2nv12` 関数は、UYVY (YUV 4:2:2) の 1 チャネル画像を NV12 フォーマットに変換します。出力は Y および UV プレーンです。UYVY はサブサンプリング フォーマットで、1 つの UYVY のセット値から Y 値が 2 つと U 値および V 値が 1 つずつ得られます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 81: xFuyvy2nv12 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。16 ビット、符号なし、1 チャネル (XF_16UC1) のみサポート。
Y_T	出力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	出力 UV 画像ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)。
COLS	入力および出力画像の最大幅 (8 の倍数で指定)。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
NPC_UV	サイクルごとに処理される UV 画像ピクセル数。1 ピクセルの操作の場合は XF_NPPC1、8 ピクセルの操作の場合は XF_NPPC4。
_src	サイズ (ROWS, COLS) の入力画像。
_y_image	サイズ (ROWS, COLS) 出力 Y プレーン。
_uv_image	サイズ (ROWS/2, COLS/2) の出力 U プレーン。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での UYVY to NV12 のリソース使用量を示します。

表 82: xFuyvy2nv12 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	831	488	131
8 ピクセル	150	0	0	1235	677	168

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での UYVY to NV12 のパフォーマンス見積もりを示します。

表 83: xFuyvy2nv12 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

IYUV to RGBA (xFiyuv2rgba)

xFiyuv2rgba 関数は、1 チャネルの IYUV (YUV 4:2:0) の画像を 4 チャネルの RGBA 画像に変換します。入力は Y、U、V プレーンです。IYUV はサブサンプリング フォーマットで、U および V 値は RGBA ピクセルの 2 行および 2 列に一度サンプリングされます。サイズ (columns/2) の連続行のデータを組み合わせることにより、サイズ (columns) の 1 行が形成されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 84: xFiyuv2rgba 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_u	サイズ (ROWS/4, COLS) の入力 U プレーン。
src_v	サイズ (ROWS/4, COLS) の入力 V プレーン。
_dst0	サイズ (ROWS, COLS) の出力 RGBA 画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での IYUV to RGBA のリソース使用量を示します。

表 85: xFiyuv2rgba 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	2	5	1208	728	196

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での IYUV to RGBA のパフォーマンス見積もりを示します。

表 86: xFiyuv2rgba 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9

IYUV to NV12 (xFiyuv2nv12)

xFiyuv2nv12 関数は、1 チャネル IYUV 画像を NV12 フォーマットに変換します。入力は U および V プレーンです。Y プレーンはどちらのフォーマットでも同じなので、Y プレーンを処理する必要はありません。U 値および V 値は、プレーン インターリーブからピクセル インターリーブに変換されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 87: xFiyuv2nv12 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	出力ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
NPC_UV	サイクルごとに処理される UV ピクセル数。1 ピクセルの場合は XF_NPPC1、4 ピクセルの場合は XF_NPPC4。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_u	サイズ (ROWS/4, COLS) の入力 U プレーン。
src_v	サイズ (ROWS/4, COLS) の入力 V プレーン。
_y_image	サイズ (ROWS, COLS) の出力 V プレーン。
_uv_image	サイズ (ROWS/2, COLS/2) の出力 UV プレーン。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での IYUV to NV12 のリソース使用量を示します。

表 88: xFiyuv2nv12 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	12	907	677	158
8 ピクセル	150	0	12	1591	1022	235

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での IYUV to NV12 のパフォーマンス見積もりを示します。

表 89: xFiyuv2nv12 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

IYUV to YUV4 (xFiyuv2yuv4)

`xFiyuv2yuv4` 関数は、1 チャネル IYUV 画像を YUV444 フォーマットに変換します。Y プレーンはどちらのフォーマットでも同じです。入力は IYUV 画像の U および V プレーンで、出力は YUV4 画像の U および V プレーンです。IYUV には、サブサンプリングされた U および V 値が格納されます。YUV フォーマットには、各ピクセルの U および V 値が格納されます。同じ U および V 値が 2 行および 2 列 (2x2) のピクセルに複製され、YUV444 フォーマットに必要なデータが取得されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 90: xFiyuv2yuv4 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_u	サイズ (ROWS/4, COLS) の入力 U プレーン。
src_v	サイズ (ROWS/4, COLS) の入力 V プレーン。
_y_image	サイズ (ROWS, COLS) の出力 Y 画像
_u_image	サイズ (ROWS, COLS) の出力 U 画像
_v_image	サイズ (ROWS, COLS) の出力 V 画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での IYUV to YUV4 のリソース使用量を示します。

表 91: xFiyuv2yuv4 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	1398	870	232
8 ピクセル	150	0	0	2134	1214	304

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での IYUV to YUV4 のパフォーマンス見積もりを示します。

表 92: xFiyuv2yuv4 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	13.8
8 ピクセル動作 (150 MHz)	3.4

NV12 to IYUV (xFnv122iyuv)

xFnv122iyuv 関数は、NV12 フォーマットを IYUV フォーマットに変換します。入力はインターリーブされた UV プレーンのみであり、出力は U および V プレーンです。Y プレーンはどちらのフォーマットでも同じなので、Y プレーンを処理する必要はありません。U 値および V 値は、ピクセル インターリーブからプレーン インターリーブに変換されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 93: xFnv122iyuv 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	入力ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
NPC_UV	サイクルごとに処理される UV 画像ピクセル数。1 ピクセルの操作の場合は XF_NPPC1、4 ピクセルの操作の場合は XF_NPPC4。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_uv	サイズ (ROWS/2, COLS/2) の入力 UV プレーン。
_y_image	サイズ (ROWS, COLS) 出力 Y プレーン。
_u_image	サイズ (ROWS/4, COLS) の出力 U プレーン。
_v_image	サイズ (ROWS/4, COLS) の出力 V プレーン。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での NV12 to IYUV のリソース使用量を示します。

表 94: xFnv122iyuv 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	1	1344	717	208
8 ピクセル	150	0	1	1961	1000	263

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での NV12 to RGBA のパフォーマンス見積もりを示します。

表 95: xFnv122iyuv 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

NV12 to RGBA (xFnv122rgba)

`xFnv122rgba` 関数は、NV12 画像フォーマットを 4 チャネル RGBA 画像に変換します。入力は Y および UV プレーンです。NV12 にはサブサンプリングされたデータが格納されており、Y プレーンがユニット レートでサンプリングされ、2x2 の Y 値ごとに U 値と V 値が 1 つずつサンプリングされます。RGBA データを生成するため、各 U および V 値が 2x2 回複製されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 96: xFnv122rgba 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	入力ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)。
COLS	入力および出力画像の最大幅 (8 の倍数で指定)。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_uv	サイズ (ROWS/2, COLS/2) の入力 UV プレーン。
_dst0	サイズ (ROWS, COLS) の出力 RGBA 画像。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での NV12 to RGBA のリソース使用量を示します。

表 97: xFnv122rgba 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	2	5	1191	708	195

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での NV12 to RGBA のパフォーマンス見積もりを示します。

表 98: xFnv122rgba 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9

NV12 to YUV4 (xFnv122yuv4)

xFnv122yuv4 関数は、NV12 画像フォーマットを YUV444 フォーマットに変換します。出力は U および V プレーンです。Y プレーンはどちらの画像フォーマットでも同じです。UV プレーンは 2x2 回複製され、YUV444 画像フォーマットの 1 つの U プレーンと V プレーンを表します。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 99: xFnv122yuv4 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	入力ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
NPC_UV	サイクルごとに処理される UV 画像ピクセル数。1 ピクセルの操作の場合は XF_NPPC1、4 ピクセルの操作の場合は XF_NPPC4。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_uv	サイズ (ROWS/2, COLS/2) の入力 UV プレーン。
_y_image	サイズ (ROWS, COLS) 出力 Y プレーン。
_u_image	サイズ (ROWS, COLS) の出力 U プレーン。
_v_image	サイズ (ROWS, COLS) の出力 V プレーン。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での NV12 to YUV4 のリソース使用量を示します。

表 100: xFnv122yuv4 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	1383	832	230
8 ピクセル	150	0	0	1772	1034	259

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での NV12 to YUV4 のパフォーマンス見積もりを示します。

表 101: xFnv122yuv4 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	13.8
8 ピクセル動作 (150 MHz)	3.4

NV21 to IYUV (xFnv212iyuv)

xFnv212iyuv 関数は、NV21 画像フォーマットを IYUV 画像フォーマットに変換します。入力はインターリーブされた VU プレーンのみであり、出力は U および V プレーンです。Y プレーンはどちらのフォーマットでも同じなので、Y プレーンを処理する必要はありません。U 値および V 値は、ピクセル インターリーブからプレーン インターリーブに変換されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 102: xFnv212iyuv 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	入力ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
NPC_UV	サイクルごとに処理される UV 画像ピクセル数。1 ピクセルの操作の場合は XF_NPPC1、4 ピクセルの操作の場合は XF_NPPC4。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_uv	サイズ (ROWS/2, COLS/2) の入力 UV プレーン。
_y_image	サイズ (ROWS, COLS) 出力 Y プレーン。
_u_image	サイズ (ROWS/4, COLS) の出力 U プレーン。
_v_image	サイズ (ROWS/4, COLS) の出力 V プレーン。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での NV21 to IYUV のリソース使用量を示します。

表 103: xFnv212iyuv 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	1	1377	730	219
8 ピクセル	150	0	1	1975	1012	279

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での NV21 to RGBA のパフォーマンス見積もりを示します。

表 104: xFnv212iyuv 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

NV21 to RGBA (xFnv212rgba)

xFnv212rgba 関数は、NV21 画像フォーマットを 4 チャネル RGBA 画像に変換します。入力は Y および VU プレーンです。NV12 にはサブサンプリングされたデータが格納されており、Y プレーンがユニット レートでサンプリングされ、2x2 の Y 値ごとに U 値と V 値が 1 つずつサンプリングされます。RGBA データを生成するため、各 U および V 値が 2x2 回複製されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 105: xFnv212rgba 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	入力ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
DST_T	出力ピクセル タイプ。8 ビット、符号なし、4 チャネル (XF_8UC4) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)。
COLS	入力および出力画像の最大幅 (8 の倍数で指定)。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_uv	サイズ (ROWS/2, COLS/2) の入力 UV プレーン。
_dst0	サイズ (ROWS, COLS) の出力 RGBA 画像。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での NV21 to RGBA のリソース使用量を示します。

表 106: xFnv212rgba 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	2	5	1170	673	183

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での NV12 to RGBA のパフォーマンス見積もりを示します。

表 107: xFnv212rgba 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9

NV21 to YUV4 (xFnv212yuv4)

xFnv212yuv4 関数は、NV12 画像フォーマットを YUV444 フォーマットに変換します。出力は U および V プレーンです。Y プレーンはどちらのフォーマットでも同じです。UV プレーンは 2x2 回複製され、YUV444 フォーマットの 1 つの U プレーンと V プレーンを表します。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 108: xFnv212yuv4 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
UV_T	入力ピクセル タイプ。8 ビット、符号なし、2 チャネル (XF_8UC2) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
NPC_UV	サイクルごとに処理される UV 画像ピクセル数。1 ピクセルの操作の場合は XF_NPPC1、4 ピクセルの操作の場合は XF_NPPC4。
src_y	サイズ (ROWS, COLS) の入力 Y プレーン。
src_uv	サイズ (ROWS/2, COLS/2) の入力 UV プレーン。
_y_image	サイズ (ROWS, COLS) 出力 Y プレーン。
_u_image	サイズ (ROWS, COLS) の出力 U プレーン。
_v_image	サイズ (ROWS, COLS) の出力 V プレーン。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での NV21 to YUV4 のリソース使用量を示します。

表 109: xFnv212yuv4 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	1383	817	233
8 ピクセル	150	0	0	1887	1087	287

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定での NV21 to YUV4 のパフォーマンス見積もりを示します。

表 110: xFnv212yuv4 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	13.8
8 ピクセル動作 (150 MHz)	3.5

カスタムたたみ込み (xFfilter2D)

xFfilter2D 関数は、ユーザー定義のカーネルを使用して画像に対してたたみ込みを実行します。

たたみ込みは、2 つの関数 f および g に対する数学演算で、3 つ目の関数を生成します。通常 3 つ目の関数は元の関数の 1 つを変更したバージョンと考慮され、元の関数の 1 つが変換された量までの 2 つの関数のエリア オーバーラップを示します。

フィルターには、ユニティ ゲイン フィルターまたは非ユニティ ゲイン フィルターを使用できます。フィルターは、AU_16SP 型である必要があります。係数が浮動小数点の場合は、Qm.n に変換して入力として供給し、シフト パラメーターを n 値に設定する必要があります。入力が浮動小数点でない場合は、フィルターは直接供給され、シフト パラメーターは 0 に設定されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 111: xFfilter2D 関数パラメーターの説明

パラメーター	説明
BORDER_TYPE	サポートされる境界タイプは XF_BORDER_CONSTANT。
FILTER_HEIGHT	入力フィルターの行数
FILTER_WIDTH	入力フィルターの列数
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセルのタイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) および 16 ビット、符号付き、1 チャネル (XF_16SC1) をサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_mat	出力画像
filter	任意のサイズ (奇数) の入力フィルター。フィルターの係数は 16 ビット値または 16 ビット固定小数点の等価値。
_shift	フィルターは、XF_16SP 型である必要があります。係数が浮動小数点の場合は、Qm.n に変換して入力として供給し、シフト パラメーターを n 値に設定する必要があります。入力が浮動小数点でない場合は、フィルターは直接供給され、シフト パラメーターは 0 に設定されます。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 112: xFfilter2D 関数のリソース使用量のサマリ

動作モード	フィルター サイズ	動作周波数 (MHz)	使用量の見積もり				
			BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	3x3	300	3	9	1701	1161	269
	5x5	300	5	25	3115	2144	524
8 ピクセル	3x3	150	6	72	2783	2768	638
	5x5	150	10	216	3020	4443	1007

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 113: xFfilter2D 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	フィルター サイズ	レイテンシ見積もり 最大 (ms)
1 ピクセル	300	3x3	7
	300	5x5	7.1
8 ピクセル	150	3x3	1.86
	150	5x5	1.86

膨張 (xFdilate)

膨張演算は、現在のピクセル強度をその近傍 3x3 ピクセルで最大の強度値に置き換えます。

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 114: xFdilate 関数パラメーターの説明

パラメーター	説明
BORDER_TYPE	サポートされる境界タイプは XF_BORDER_CONSTANT。
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_mat	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で Vivado HLS 2017.1 バージョン ツールを使用して生成された、1 ピクセル操作と 8 ピクセル操作における膨張関数のリソース使用量を示します。

表 115: xFdilate 関数のリソース使用量のサマリ

名前	リソース使用量	
	クロックごとに 1 ピクセル	クロックごとに 8 ピクセル
	300 MHz	150 MHz
BRAM_18K	3	6
DSP48E	0	0
FF	339	644
LUT	350	1325
CLB	81	245

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で Vivado HLS 2017.1 ツールを使用して生成された、通常動作 (1 ピクセル) およびリソース最適化 (8 ピクセル) 設定での膨張関数のパフォーマンス見積もりを示します。

表 116: xFdilate 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり	
	最小 (ms)	最大 (ms)
1 ピクセル (300 MHz)	7.0	7.0
8 ピクセル (150 MHz)	1.87	1.87

収縮 (xFerode)

`xFerode` 関数は、現在のピクセル強度をその近傍 3x3 ピクセルで最小の強度値に置き換えます。

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 117: xFerode 関数パラメーターの説明

パラメーター	説明
BORDER_TYPE	サポートされる境界タイプは XF_BORDER_CONSTANT です。
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_mat	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で Vivado HLS 2017.1 バージョン ツールを使用して生成された収縮関数のリソース使用量を示します。

表 118: xFerode 関数のリソース使用量のサマリ

名前	リソース使用量	
	クロックごとに 1 ピクセル	クロックごとに 8 ピクセル
	300 MHz	150 MHz
BRAM_18K	3	6
DSP48E	0	0
FF	342	653
LUT	351	1316
CLB	79	230

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で Vivado HLS 2017.1 ツールを使用して生成された、通常動作 (1 ピクセル) およびリソース最適化 (8 ピクセル) 設定での収縮関数のパフォーマンス見積もりを示します。

表 119: xFerode 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり	
	最小 (ms)	最大 (ms)
1 ピクセル (300 MHz)	7.0	7.0
8 ピクセル (150 MHz)	1.85	1.85

FAST コーナー検出 (xFFAST)

FAST (Features from Accelerated Segment Test) は、ほかのほとんどの特徴検出より高速のコーナー検出アルゴリズムです。

xFFAST 関数は、画像内のピクセルを 1 つ選び、Bresenham の円と呼ばれる円上の近傍ピクセル 16 個の強度を比較します。9 個の連続するピクセルの強度が候補のピクセルよりも指定のしきい値分大きいか小さい場合、ピクセルをコーナーとみなします。コーナーが検出されると NMS (Non-Maximal Suppression) が適用され、弱いコーナーが削除されます。

この関数は、静止画像およびビデオの両方に使用できます。最初に、コーナーの最大数を指定する必要があります。返されるコーナーの合計数は、この最大値以下である必要があります。画像内の実際のコーナー数が指定の最大値を超える場合は、しきい値を増加してコーナー数を削減できます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 120: xFFAST 関数パラメーターの説明

パラメーター	説明
NMS	NMS == 1 の場合、検出されたコーナー (キーポイント) に NMS が適用されます。有効な値は 0 または 1 です。
MAXPNTS	カーネルで検出可能な最大コーナー数。
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力画像の最大高さ (8 の倍数で指定)
COLS	入力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
list	コーナーのリスト。コーナーは 32 ビット フォーマットにパックされます。下位 16 ビットは列インデックス、上位 16 ビットは行インデックスを示します。
_threshold	中央ピクセルと近傍ピクセルとの強度の差のしきい値。通常 20 前後の値が使用されます。
nCorners	入力画像で検出されるコーナー数 (カーネルの出力)。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像で NMS を使用して 1024 個のコーナーを処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 121: xFFAST 関数のリソース使用量のサマリ

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	10	28
DSP48E	0	0
FF	2695	7310
LUT	3792	20956
CLB	769	3519

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像で NMS を使用して 1024 個のコーナーを処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 122: xFFAST 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	フィルター サイズ	レイテンシ見積もり 最大 (ms)
1 ピクセル	300	3x3	7
8 ピクセル	150	3x3	1.86

ガウシアン フィルター (xFGaussianBlur)

xFGaussianBlur 関数は、入力画像にガウシアンぼかしを適用します。ガウシアン フィルター処理は、入力画像の各点をガウシアン カーネルでたたみ込むことにより実行されます。

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

ここで、μ_x および μ_y は平均値、σ_x および σ_y はそれぞれ x 方向と y 方向の分散です。GaussianBlur 関数では、μ_x および μ_y は 0、σ_x および σ_y は等しいと考慮されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 123: xFGaussianBlur 関数パラメーターの説明

パラメーター	説明
FILTER_SIZE	フィルター サイズ。サポートされるフィルター サイズは 3 (XF_FILTER_3X3)、5 (XF_FILTER_5X5)、および 7 (XF_FILTER_7X7) です。
BORDER_TYPE	サポートされる境界タイプは XF_BORDER_CONSTANT。
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。

パラメーター	説明
src	入力画像
dst	出力画像
sigma	ガウシアン フィルターの標準偏差

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのガウシアン フィルターのリソース使用量を示します。

表 124: xFGaussianBlur 関数のリソース使用量のサマリ

動作モード	フィルター サイズ	動作周波数 (MHz)	使用量の見積もり				
			BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	3x3	300	3	17	3641	2791	610
	5x5	300	5	27	4461	3544	764
	7x7	250	7	35	4770	4201	894
8 ピクセル	3x3	150	6	52	3939	3784	814
	5x5	150	10	111	5688	5639	1133
	7x7	150	14	175	7594	7278	1518

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es11 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのガウシアン フィルターのパフォーマンス見積もりを示します。

表 125: xFGaussianBlur 関数のパフォーマンス見積もりのサマリ

動作モード	フィルター サイズ	レイテンシ見積もり
		最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	3x3	7.01
	5x5	7.03
	7x7	7.06
8 ピクセル動作 (150 MHz)	3x3	1.6
	5x5	1.7
	7x7	1.74

勾配の大きさ (xFmagnitude)

xFmagnitude 関数では、画像の大きさが計算されます。入力画像は、16S 型の X 勾配および Y 勾配画像です。出力画像は、入力画像と同型になります。

L1NORM 正規化の場合、大きさの計算された画像は、次のように X 勾配と Y 勾配の絶対値のピクセル加算された画像になります。

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

L2NORM 正規化の場合、画像の大きさは次のように計算されます。

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 126: xFmagnitude 関数パラメーターの説明

パラメーター	説明
NORM_TYPE	正規化タイプは、L1 または L2 正規化にでき、値は XF_L1NORM または XF_L2NORM です。
SRC_T	入力ピクセル タイプ。16 ビット、符号付き、1 チャネル (XF_16SC1) のみサポート。
DST_T	出力ピクセル タイプ。16 ビット、符号付き、1 チャネル (XF_16SC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_matx	最初の入力、X 勾配画像。
_src_maty	2 つ目の入力、Y 勾配画像。
_dst_mat	出力、大きさの計算された画像。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像で L2 正規化の処理をするために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 127: xFmagnitude 関数のリソース使用量のサマリ

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	0	0
DSP48E	2	16
FF	707	2002
LUT	774	3666
CLB	172	737

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像で L2 正規化の処理をするために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 128: xFmagnitude 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり
		最大 (ms)
1 ピクセル	300	7.2
8 ピクセル	150	1.7

勾配位相 (xFphase)

xFphase 関数では、2 つの画像の極角が計算されます。入力画像は、16S 型の X 勾配および Y 勾配画像です。出力画像は、入力画像と同型になります。

ラジアン (radians) の場合:

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

度 (degrees) の場合:

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 129: xFphase 関数パラメーターの説明

パラメーター	説明
RET_TYPE	出力フォーマットは、ラジアン (radians) か度 (degrees) のいずれかにできます。オプションは XF_RADIANS または XF_DEGREES です。 ・ XF_RADIANS: Q4.12 フォーマットの結果が返されます。出力範囲は (0, 2 pi) です。 ・ XF_DEGREES: 結果が Q10.6 度で返されます。出力範囲は (0, 360) です。
SRC_T	入力ピクセル タイプ。16 ビット、符号付き、1 チャネル (XF_16SC1) のみサポート。
DST_T	出力ピクセル タイプ。16 ビット、符号付き、1 チャネル (XF_16SC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_matx	最初の入力、X 勾配画像。
_src_maty	2 つ目の入力、Y 勾配画像。
_dst_mat	出力、位相計算された画像。

1. XF_RADIANS オプションを選択すると、xFphase API により結果が Q4.12 フォーマットで返されます。出力範囲は (0, 2 pi) です。
2. XF_DEGREES オプションを選択すると、xFphase API により結果が Q10.6 度で返されます。出力範囲は (0, 360) です。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 130: xFphase 関数のリソース使用量のサマリ

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	6	24
DSP48E	6	19
FF	873	2396
LUT	753	3895
CLB	185	832

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 131: xFphase 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり (ms)
1 ピクセル	300	7.2
8 ピクセル	150	1.7

OpenCV との違い

xFphase インプリメンテーションでは、出力が固定小数点フォーマットで返されます。XF_RADIANS オプションを選択すると、xFphase API により結果が Q4.12 フォーマットで返されます。出力範囲は (0, 2 pi) です。XF_DEGREES オプションを選択すると、xFphase API により結果が Q10.6 度で返されます。出力範囲は (0, 360) です。

Harris コーナー検出 (xFCornerHarris)

Harris コーナー検出を理解するには、グレースケール画像を考慮します。ウィンドウ `w(x,y)` (x 方向に `u`、`y` 方向に `v` 移動) をスイープし、`I` で `w(x,y)` の強度の変動を計算します。

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

説明:

- `w(x,y)`: (x,y) のウィンドウ位置
- `I(x,y)`: (x,y) での強度
- `I(x+u,y+v)`: 移動したウィンドウ `(x+u,y+v)` での強度

コーナーのあるウィンドウを探しているので、強度の変動が大きいウィンドウを見つけます。上記の式、特に次の項を最大化する必要があります。

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

テーラー展開を使用します。

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

式を展開して `I(x,y)` を `-I(x,y)` で相殺します。

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

上記の式を行列で表すと、次のようになります。

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

最終的な式は、次のようになります。

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

各ウィンドウに対してスコアが計算され、コーナーを含むかどうかが判断されます。

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

説明:

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

NMS (Non-Maximum Suppression):

NMS では、範囲が 1 の場合境界ボックスは 2*r+1 = 3 です。

中央ピクセルの近傍 3x3 ピクセルを考慮します。中央ピクセルが周辺ピクセルより大きい場合、これはコーナーと考えられます。範囲内の周辺ピクセルと比較されます。

範囲 = 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

しきい値:

3x3、5x5、および 7x7 フィルターに対して、それぞれしきい値 442、3109、および 566 が使用されます。このしきい値は、40 を超える画像のセットで検証されます。しきい値は、アプリケーションによって異なります。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 132: xFCornerHarris 関数パラメーターの説明

パラメーター	説明
MAXCORNERS	カーネルで検出可能な最大コーナー数。
FILTER_SIZE	Sobel フィルターのサイズ。有効な値は 3、5、7。
BLOCK_WIDTH	ボックス フィルターのサイズ。有効な値は 3、5、7。
NMS_RADIUS	NMS で考慮される範囲。有効な値は 1 および 2。
TYPE	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力画像の最大高さ (8 の倍数で指定)
COLS	入力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src	入力画像
points	コーナーのリスト。コーナーは 32 ビット フォーマットにパックされます。下位 16 ビットは列インデックス、上位 16 ビットは行インデックスを示します。
threshold	コーナー計測に適用されるしきい値。
k	Harris 検出器パラメーター
nCorners	入力画像で検出されたコーナー数を示すカーネルからの出力

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像で 1024 個のコーナーを処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での Harris コーナー検出のリソース使用量を示します。

次の表に、Sobel フィルター = 3、ボックス フィルター = 3、NMS_RADIUS = 1 のリソース使用量を示します。

表 133: リソース使用量のサマリ: Sobel フィルター = 3、ボックス フィルター = 3、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	33	74
DSP48E	13	83
FF	3254	9330
LUT	3522	13222
CLB	731	2568

次の表に、Sobel フィルター = 3、ボックス フィルター = 5、NMS_RADIUS = 1 のリソース使用量を示します。

表 134: リソース使用量のサマリ: Sobel フィルター = 3、ボックス フィルター = 5、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	45	98
DSP48E	13	83
FF	5455	12459
LUT	5675	24594
CLB	1132	4498

次の表に、Sobel フィルター = 3、ボックス フィルター = 7、NMS_RADIUS = 1 のリソース使用量を示します。

表 135: リソース使用量のサマリ: Sobel フィルター = 3、ボックス フィルター = 7、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	57	122
DSP48E	13	83
FF	8783	16593
LUT	9157	39813
CLB	1757	6809

次の表に、Sobel フィルター = 5、ボックス フィルター = 3、NMS_RADIUS = 1 のリソース使用量を示します。

表 136: リソース使用量のサマリ: Sobel フィルター = 5、ボックス フィルター = 3、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	200 MHz
BRAM_18K	35	78

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	200 MHz
DSP48E	13	83
FF	4656	11659
LUT	4681	17394
CLB	1005	3277

次の表に、Sobel フィルター = 5、ボックス フィルター = 5、NMS_RADIUS = 1 のリソース使用量を示します。

表 137: リソース使用量のサマリ: Sobel フィルター = 5、ボックス フィルター = 5、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	47	102
DSP48E	13	83
FF	6019	14776
LUT	6337	28795
CLB	1353	5102

次の表に、Sobel フィルター = 5、ボックス フィルター = 7、NMS_RADIUS = 1 のリソース使用量を示します。

表 138: リソース使用量のサマリ: Sobel フィルター = 5、ボックス フィルター = 7、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	59	126
DSP48E	13	83
FF	9388	18913
LUT	9414	43070
CLB	1947	7508

次の表に、Sobel フィルター = 7、ボックス フィルター = 3、NMS_RADIUS = 1 のリソース使用量を示します。

表 139: リソース使用量のサマリ: Sobel フィルター = 7、ボックス フィルター = 3、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	37	82
DSP48E	14	91
FF	6002	13880

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
LUT	6337	25573
CLB	1327	4868

次の表に、Sobel フィルター = 7、ボックス フィルター = 5、NMS_RADIUS = 1 のリソース使用量を示します。

表 140: リソース使用量のサマリ: Sobel フィルター = 7、ボックス フィルター = 5、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	49	106
DSP48E	14	91
FF	7410	17049
LUT	8076	36509
CLB	1627	6518

次の表に、Sobel フィルター = 7、ボックス フィルター = 7、NMS_RADIUS = 1 のリソース使用量を示します。

表 141: リソース使用量のサマリ: Sobel フィルター = 7、ボックス フィルター = 7、NMS_RADIUS = 1

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	61	130
DSP48E	14	91
FF	10714	21137
LUT	11500	51331
CLB	2261	8863

次の表に、Sobel フィルター = 3、ボックス フィルター = 3、NMS_RADIUS = 2 のリソース使用量を示します。

表 142: リソース使用量のサマリ: Sobel フィルター = 3、ボックス フィルター = 3、NMS_RADIUS = 2

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	41	90
DSP48E	13	83
FF	5519	10714
LUT	5094	16930
CLB	1076	3127

リソース使用量: Sobel フィルター = 3、ボックス フィルター = 5、NMS_RADIUS = 2

表 143: リソース使用量のサマリ

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	53	114
DSP48E	13	83
FF	6798	13844
LUT	6866	28286
CLB	1383	4965

次の表に、Sobel フィルター = 3、ボックス フィルター = 7、NMS_RADIUS = 2 のリソース使用量を示します。

表 144: リソース使用量のサマリ: Sobel フィルター = 3、ボックス フィルター = 7、NMS_RADIUS = 2

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	65	138
DSP48E	13	83
FF	10137	17977
LUT	10366	43589
CLB	1940	7440

次の表に、Sobel フィルター = 5、ボックス フィルター = 3、NMS_RADIUS = 2 のリソース使用量を示します。

表 145: リソース使用量のサマリ: Sobel フィルター = 5、ボックス フィルター = 3、NMS_RADIUS = 2

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	43	94
DSP48E	13	83
FF	5957	12930
LUT	5987	21187
CLB	1244	3922

次の表に、Sobel フィルター = 5、ボックス フィルター = 5、NMS_RADIUS = 2 のリソース使用量を示します。

表 146: リソース使用量のサマリ: Sobel フィルター = 5、ボックス フィルター = 5、NMS_RADIUS = 2

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	55	118
DSP48E	13	83
FF	5442	16053
LUT	6561	32377
CLB	1374	5871

次の表に、Sobel フィルター = 5、ボックス フィルター = 7、NMS_RADIUS = 2 のリソース使用量を示します。

表 147: リソース使用量のサマリ: Sobel フィルター = 5、ボックス フィルター = 7、NMS_RADIUS = 2

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	67	142
DSP48E	13	83
FF	10673	20190
LUT	10793	46785
CLB	2260	8013

次の表に、Sobel フィルター = 7、ボックス フィルター = 3、NMS_RADIUS = 2 のリソース使用量を示します。

表 148: リソース使用量のサマリ: Sobel フィルター = 7、ボックス フィルター = 3、NMS_RADIUS = 2

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	45	98
DSP48E	14	91
FF	7341	15161
LUT	7631	29185
CLB	1557	5425

次の表に、Sobel フィルター = 7、ボックス フィルター = 5、NMS_RADIUS = 2 のリソース使用量を示します。

表 149: リソース使用量のサマリ: Sobel フィルター = 7、ボックス フィルター = 5、NMS_RADIUS = 2

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	57	122

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
DSP48E	14	91
FF	8763	18330
LUT	9368	40116
CLB	1857	7362

次の表に、Sobel フィルター = 7、ボックス フィルター = 7、NMS_RADIUS = 2 のリソース使用量を示します。

表 150: リソース使用量のサマリ: Sobel フィルター = 7、ボックス フィルター = 7、NMS_RADIUS = 2

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	69	146
DSP48E	14	91
FF	12078	22414
LUT	12831	54652
CLB	2499	9628

パフォーマンス見積もり

次の表に、ザイリンクス Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での Harris コーナー検出のパフォーマンス見積もりを示します。

表 151: xFCornerHarris 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	設定			レイテンシ見積もり
		Sobel	ボックス	NMS 範囲	レイテンシ (ms)
1 ピクセル	300 MHz	3	3	1	7
1 ピクセル	300 MHz	3	5	1	7.1
1 ピクセル	300 MHz	3	7	1	7.1
1 ピクセル	300 MHz	5	3	1	7.2
1 ピクセル	300 MHz	5	5	1	7.2
1 ピクセル	300 MHz	5	7	1	7.2
1 ピクセル	300 MHz	7	3	1	7.22
1 ピクセル	300 MHz	7	5	1	7.22
1 ピクセル	300 MHz	7	7	1	7.22
8 ピクセル	150 MHz	3	3	1	1.7
8 ピクセル	150 MHz	3	5	1	1.7
8 ピクセル	150 MHz	3	7	1	1.7
8 ピクセル	150 MHz	5	3	1	1.71
8 ピクセル	150 MHz	5	5	1	1.71
8 ピクセル	150 MHz	5	7	1	1.71
8 ピクセル	150 MHz	7	3	1	1.8
8 ピクセル	150 MHz	7	5	1	1.8
8 ピクセル	150 MHz	7	7	1	1.8
1 ピクセル	300 MHz	3	3	2	7.1
1 ピクセル	300 MHz	3	5	2	7.1
1 ピクセル	300 MHz	3	7	2	7.1
1 ピクセル	300 MHz	5	3	2	7.21
1 ピクセル	300 MHz	5	5	2	7.21
1 ピクセル	300 MHz	5	7	2	7.21
1 ピクセル	300 MHz	7	3	2	7.22
1 ピクセル	300 MHz	7	5	2	7.22
1 ピクセル	300 MHz	7	7	2	7.22
8 ピクセル	150 MHz	3	3	2	1.8
8 ピクセル	150 MHz	3	5	2	1.8
8 ピクセル	150 MHz	3	7	2	1.8
8 ピクセル	150 MHz	5	3	2	1.81
8 ピクセル	150 MHz	5	5	2	1.81
8 ピクセル	150 MHz	5	7	2	1.81
8 ピクセル	150 MHz	7	3	2	1.9
8 ピクセル	150 MHz	7	5	2	1.91
8 ピクセル	150 MHz	7	7	2	1.92

OpenCV との違い

xfOpenCV にはしきい値と NMS が含まれますが、OpenCV には含まれません。xfOpenCV では、すべての点が固定小数点でインプリメントされます。OpenCV では、すべてのブロックが浮動小数点でインプリメントされます。

ヒストグラム計算 (calcHist)

`calcHist` 関数は、入力画像のヒストグラムを計算します。

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

ここで、H は 256 個の要素の配列です。

API 構文

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

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 152: calcHist 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数
_src	入力画像
histogram	256 個の要素の出力配列

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で Vivado HLS 2017.1 ツールを使用して生成された、通常動作 (1 ピクセル モードで 300 MHz) およびリソース最適化 (8 ピクセル モードで 150 MHz) 設定での calcHist 関数のリソース使用量を示します。

表 153: calcHist 関数のリソース使用量のサマリ

名前	リソース使用量	
	通常動作 (1 ピクセル)	リソース最適化 (8 ピクセル)
BRAM_18K	2	16
DSP48E	0	0
FF	196	274
LUT	240	912
CLB	57	231

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で Vivado HLS 2017.1 ツールを使用して生成された、通常動作 (1 ピクセル モードで 300 MHz) およびリソース最適化 (8 ピクセル モードで 150 MHz) 設定での calcHist 関数のパフォーマンス見積もりを示します。

表 154: calcHist 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大 (ms)
通常動作	6.9
リソース最適化	1.7

ヒストグラム均一化 (xFequalizeHist)

`equalizeHist` 関数は、入力画像またはビデオに対してヒストグラム均一化処理を実行します。画像のコントラストを改善し、強度範囲を拡張します。この関数は、1 つの分布 (ヒストグラム) を別の分布 (より幅が広く強度値がより均一に分配されている) にマップし、強度が範囲全体に拡張されるようにします。

ヒストグラム H[i] に対し、累積分布 H'[i] は次のようになります。

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

均一化された画像の強度は次のように計算されます。

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 155: xFequalizeHist 関数パラメーターの説明

パラメーター	説明
SRC_T	入力および出力のピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数
_src	入力画像
_src1	入力画像
_dst	出力画像

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で Vivado HLS 2017.1 ツールを使用して生成された、通常動作 (1 ピクセル モードで 300 MHz) およびリソース最適化 (8 ピクセル モードで 150 MHz) 設定での equalizeHist 関数のリソース使用量を示します。

表 156: xFequalizeHist 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	4	5	3492	1807	666
8 ピクセル	150	25	5	3526	2645	835

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA で Vivado HLS 2017.1 ツールを使用して生成された、通常動作 (1 ピクセル モードで 300 MHz) およびリソース最適化 (8 ピクセル モードで 150 MHz) 設定での equalizeHist 関数のパフォーマンス見積もりを示します。

表 157: xFequalizeHist 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大 (ms)
クロックごとに 1 ピクセル	13.8
クロックごとに 8 ピクセル	3.4

HOG (xFHOGDescriptor)

HOG (Histogram of Oriented Gradients) は、コンピューター ビジョンで物体検出のために使用される特徴ディスクリプターです。この方法で生成された特徴ディスクリプターは、歩行者の検出に広く使用されます。

この方法では、画像中の局所領域内における勾配方向の発生回数をカウントします。HOG は均一間隔のセルの高密度グリッドで計算され、精度を向上するためオーバーラップしたブロックが正規化されます。HOG は、画像内の物体の外観と形状を強度勾配またはエッジ方向の分布で記述できるという概念に基づいています。

関数には RGB とグレー入力の両方を入力できます。RGB モードでは、各プレーンに対して勾配が個別に計算され、大きいものが選択されます。指定された設定では、ウィンドウのサイズは 64x128、ブロックのサイズは 16x16 です。

API 構文

```
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
```

パラメーターの説明

次の表では、テンプレート パラメーターについて説明します。

表 158: xFHOGDescriptor テンプレート パラメーターの説明

パラメーター	説明
WIN_HEIGHT	ウィンドウのピクセル行の数。128 に固定されています。
WIN_WIDTH	ウィンドウのピクセル列の数。64 に固定されています。
WIN_STIRDE	2 つの隣接するウィンドウ間のピクセル幅。8 に固定されています。
BLOCK_HEIGHT	ブロックの高さ。16 に固定されています。
BLOCK_WIDTH	ブロックの幅。16 に固定されています。
CELL_HEIGHT	セルの行数。8 に固定されています。
CELL_WIDTH	セルの列数。8 に固定されています。
NOB	セルのヒストグラム ビンの数。9 に固定されています。
ROWS	処理される画像の行数。8 の倍数で指定します。
COLS	処理される画像の列数。8 の倍数で指定します。
SRC_T	入力ピクセル タイプ。グレーの場合は XF_8UC1、カラーの場合は XF_8UC4。
DST_T	出力ディスクリプター タイプ。XF_32UC1 に設定する必要があります。
DESC_SIZE	出力ディスクリプターのサイズ。
NPC	サイクルごとに処理されるピクセル数。XF_NPPC1 (サイクルごとに 1 ピクセルの操作) のみサポート。
IMG_COLOR	画像のタイプ。XF_GRAY または XF_RGB に設定します。
OUTPUT_VARIENT	XF_HOG_RB または XF_HOG_NRB に設定する必要があります。

次の表では、関数パラメーターについて説明します。

表 159: xFHOGDescriptor 関数パラメーターの説明

パラメーター	説明
_in_mat	xF::Mat タイプの入力画像
_desc_mat	xF::Mat タイプの出力ディスクリプター

説明:

- NO: 通常動作 (1 ピクセル処理)
- RB: 繰り返しブロック (ディスクリプター データはウィンドウ単位で記述される)
- NRB: 非繰り返しブロック (書き込み数を削減するため、ディスクリプター データはウィンドウ単位で記述される)。

注記: RB モードでは、ブロック データはオーバーラップ ウィンドウを考慮してメモリに書き込まれます。NRB モードでは、ブロック データはウィンドウ オーバーラップを考慮せずに出力ストリームに直接書き込まれます。ホスト側でオーバーラップを処理する必要があります。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 で解像度 1920x1080 の画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された通常動作 (1 ピクセル) での `xFHOGDescriptor` 関数のリソース使用量を示します。

表 160: xFHOGDescriptor 関数のリソース使用量のサマリ

リソース	1 ピクセル動作 (300 MHz) での使用量			
	NRB		RB	
	グレー	RGB	グレー	RGB
BRAM_18K	43	49	171	177
DSP48E	34	46	36	48
FF	15365	15823	15205	15663
LUT	12868	13267	13443	13848

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 で解像度 1920x1080p の画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定での xFHOGDescriptor() 関数のリソース使用量を示します。

表 161: xFHOGDescriptor 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり	
		最小 (ms)	最大 (ms)
NRB-グレー	300	6.98	8.83
NRB-RGBA	300	6.98	8.83
RB-グレー	300	176.81	177
RB-RGBA	300	176.81	177

OpenCV との違い

OpenCV との違いは、次のとおりです。

1. 境界の処理

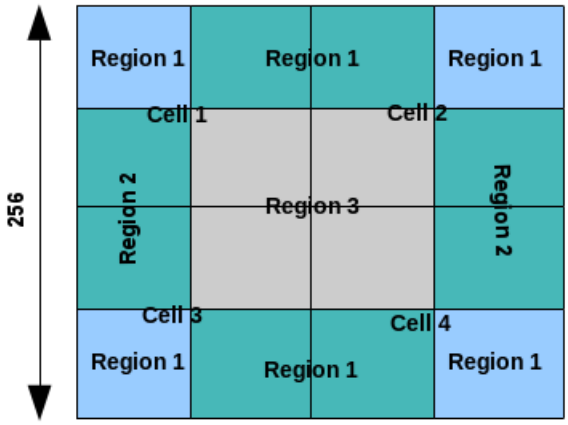
   OpenCV で勾配の計算に使用される境界の処理は BORDER_REFLECT_101 であり、境界のパディングは近傍ピクセルの反射です。ザイリンクス インプリメンテーションでは、境界の処理に BORDER_CONSTANT (0 パディング) が使用されます。

2. ガウシアン重み付け

   ガウシアン重みはブロック全体でピクセルに乗算されます。ブロックには 256 ピクセルあり、ブロックの各位置が対応するガウシアン重みで乗算されます。HLS インプリメンテーションでは、ガウシアン重み付けは実行されません。

3. セル単位の補間

   ピクセルの強度値は、対応するビンのブロック内の異なるセルに分配されます。



   領域 1 のピクセルは対応するセルにのみ属し、領域 2 および 3 のピクセルはそれぞれ隣接する 2 つのセルおよび 4 つのセルに保管されます。HLS インプリメンテーションではこの操作は実行されません。

4. 出力の処理

   OpenCV の出力は列優先形式です。HLS インプリメンテーションでは、出力は行優先形式です。また、特徴ベクターは HLS インプリメンテーションでは固定小数点型 Q0.16 ですが、OpenCV では浮動小数点です。

制限

1. 設定は Dalal のインプリメンテーションに制限されます。[1]
2. 画像の高さおよび幅は、それぞれセルの高さおよび幅の倍数である必要があります。

1. N. Dalal、B. Triggs 著:『Histograms of oriented gradients for human detection』、IEEE Computer Society Conference on Computer Vision and Pattern Recognition、2005 年

ピラミッド アップ (xFPyrUp)

xFPyrUp 関数は、画像のアップサンプリング アルゴリズムです。まず、各入力行および列の後に 0 行と 0 列を挿入して、出力画像のサイズを作成します。出力画像サイズは、常に $(2*rows \times 2*columns)$ になります。この後、0 でパディングされた画像がガウシアン画像フィルターを使用して滑らかにされます。ピラミッド アップ関数のガウシアン フィルターでは、次のような固定フィルター カーネルが使用されます。

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

ただし、0 パディングによって削減されるピクセル強度を補強するため、各出力ピクセルは 4 で乗算されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 162: xFPyrUp 関数パラメーターの説明

パラメーター	説明
TYPE	ピクセル タイプ。サポートされるピクセル タイプは XF_8UC1 のみ。
ROWS	このカーネルのハードウェアを構築するための最大高さまたは出力行数。
COLS	このカーネルのハードウェアを構築するための最大幅または出力列数。
NPC	サイクルごとに処理されるピクセル数。現時点では、カーネルでは各サイクル 1 ピクセルのみの処理 (XF_NPPC1) をサポート。
_src	入力画像ストリーム
_dst	出力画像ストリーム

リソース使用量

次の表に、最大入力画像サイズ 1920x1080 ピクセルの場合の各サイクル 1 ピクセルのインプリメンテーションでの xFPyrUp のリソース使用量を示します。これは 300 MHz の xczu9eg-ffvb1156-1-i-es1 FPGA の Vivado HLS 2017.1 での合成後の結果です。

表 163: xFPyrUp 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり			
		LUT	FF	DSP	BRAM
クロック サイクルごとに 1 ピクセル	300	1124	1199	0	10

パフォーマンス見積もり

次の表に、xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された xFPyrUp のパフォーマンス見積もりを示します。

表 164: xFPyrUp 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	入力画像サイズ	レイテンシ見積もり
			最大 (ms)
1 ピクセル	300	1920x1080	27.82

ピラミッド ダウン (xFPyrDown)

xFPyrDown 関数は、画像をダウンスケールする前に画像を滑らかにする画像のダウンサンプリング アルゴリズムです。画像は、次のカーネルのガウシアン フィルターを使用して滑らかにされます。

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

ダウンスケールは、偶数行と偶数列のピクセルが破棄されることで実行されます。結果の画像サイズは次のようになります。 $\left(\frac{rows+1}{2} \quad \frac{columns+1}{2}\right)$

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 165: xFPyrDown 関数パラメーターの説明

パラメーター	説明
TYPE	ピクセル タイプ。サポートされるピクセル タイプは XF_8UC1 のみ。
ROWS	このカーネルのハードウェアを構築するための最大高さまたは入力行数。
COLS	このカーネルのハードウェアを構築するための最大幅または入力列数。
NPC	サイクルごとに処理されるピクセル数。現時点では、カーネルでは各サイクル 1 ピクセルのみの処理 (XF_NPPC1) をサポート。
_src	入力画像ストリーム
_dst	出力画像ストリーム

リソース使用量

次の表に、最大入力画像サイズ 1920x1080 ピクセルの場合の各サイクル 1 ピクセルのインプリメンテーションでの xFPyrDown のリソース使用量を示します。これは 300 MHz の xczu9eg-ffvb1156-1-i-es1 FPGA の Vivado HLS 2017.1 での合成後の結果です。

表 166: xFPyrDown 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり			
		LUT	FF	DSP	BRAM
1 ピクセル	300	1171	1238	1	5

パフォーマンス見積もり

次の表に、xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された xFPyrDown のパフォーマンス見積もりを示します。

表 167: xFPyrDown 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	入力画像サイズ	レイテンシ見積もり 最大 (ms)
1 ピクセル	300	1920x1080	6.99

積分画像 (xFIntegralImage)

xFIntegralImage 関数では、入力の積分画像が計算されます。各出力ピクセルは、そのピクセルの上および左にあるすべてのセルの合計です。

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 168: xFIntegralImage 関数パラメーターの説明

パラメーター	説明
SRC_TYPE	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_TYPE	出力ピクセル タイプ。32 ビット、符号なし、1 チャネル (XF_32UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。XF_NPPC1 (サイクルごとに 1 ピクセルの操作) のみサポート。
_src_mat	入力画像
_dst_mat	出力画像

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 169: xFIntegralImage 関数のリソース使用量のサマリ

名前	リソース使用量
	1 ピクセル
	300 MHz
BRAM_18K	4
DSP48E	0
FF	613
LUT	378
CLB	102

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 170: xFIntegralImage 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり	
	動作周波数 (MHz)	レイテンシ (ms)
1 ピクセル	300	7.2

高密度ピラミッド型 LK オプティカル フロー (xFDensePyrOpticalFlow)

オプティカル フローは、オブジェクトまたはカメラの動きのため発生する 2 つの連続するフレーム間における画像オブジェクトの動きのパターンです。これは 2D ベクター フィールドであり、各ベクターは 1 つ目のフレームから 2 つ目のフレームの点の移動を表す変位ベクターです。

オプティカル フローは、次を前提として実行されます。

- オブジェクトのピクセル強度は、連続するフレーム間でそれほど変動しない
- 近傍ピクセルも同じように動く

最初のフレームにピクセル I(x, y, t) があるとします。3 つ目の次元として時間 (t) が追加されています。画像のみを処理する場合は、時間は必要ありません。時間 dt 後の次のフレームで、ピクセルが距離 (dx, dy) だけ移動します。これらのピクセルは同じであり強度は変化しないので、次のことが言えます。

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

右辺のテイラー級数近似を求め、共通項を取り除き、dt で割ると、次の式が得られます。

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

$f_x = \frac{\delta f}{\delta x}$ 、$f_y = \frac{\delta f}{\delta x}$ 、$u = \frac{dx}{dt}$ 、および $v = \frac{dy}{dt}$ です。

上記の式をオプティカル フローと呼びます。f_x および f_y は画像勾配で、f_t は時間の経過に伴う勾配です。ただし、(u, v) は不明です。これらの不明な変数のためこの方程式を解くことはできないので、この問題を解くために複数の方法が提供されています。1 つの方法は Lucas-Kanade (LK) です。先ほど近傍のすべてのピクセルが同じように動くと想定しました。Lucas-Kanade 法では、点を囲む WINDOW_SIZE テンプレート パラメーターで指定されたサイズのパッチが使用されます。つまり、パッチ内のすべての点が同じように動くと想定されます。これらの点に対しては、(f_x, f_y, f_t) を求めることが可能です。これで、問題は 2 つの不明な変数を含む WINDOW_SIZE * WINDOW_SIZE 方程式を解くことになり、これは重複決定されます。より良い方法は、最小二乗適合法を使用する方法です。次に、2 つの方程式と 2 つの不明な変数を含む問題の最終的な解を示します。

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

このソリューションは、大きな動きがあるためにピラミッドが使用される場合はエラーとなります。ピラミッドを上に登ると小さな動きが削除され、大きな動きが小さな動きとなりますので、Lucas-Kanade 法を適用し、そのスケールでのオプティカル フローを取得します。

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 171: xFDensePyrOpticalFlow 関数パラメーターの説明

パラメーター	説明
NUM_PYR_LEVELS	オプティカル フローの計算に使用する画像のピラミッド レベル数
NUM_LINES	一時的な勾配を見つけるために使用されるリマップ アルゴリズム用に格納する行数
WINSIZE	オプティカル フローを計算するウィンドウ サイズ。
FLOW_WIDTH、FLOW_INT	符号付きフロー ベクター データ型を定義するデータ幅と整数ビット数。整数ビットには、符号付きビットが含まれます。 デフォルトのデータ型は、10 個の整数ビットと 6 個の小数ビットを持つ 16 ビット符号付きワードです。
TYPE	入力画像のピクセル タイプ。XF_8UC1 のみをサポート。
ROWS	このカーネルのハードウェアを構築するための最大高さまたは行数。
COLS	このカーネルのハードウェアを構築するための最大幅または列数。
NPC	カーネルでクロック サイクルごとに処理する必要のあるピクセル数。サイクルごとに 1 ピクセル (XF_NPPC1) のみをサポート。
_curr_img	1 つ目の入力画像ストリーム
_next_img	1 つ目の画像に対してオプティカル フローを計算する 2 つ目の入力画像
_streamFlowin	オプティカル フロー用の 32 ビットにパックされた U および V フロー ベクター入力。31 ～ 16 のビットはフロー ベクター U を表し、15 ～ 0 のビットはフロー ベクター V を表します。
_streamFlowout	オプティカル フローの計算後に 32 ビットにパックされた U および V フロー ベクター出力。31 ～ 16 のビットはフロー ベクター U を表し、15 ～ 0 のビットはフロー ベクター V を表します。
level	アルゴリズムが現在オプティカル フローを計算している画像のピラミッド レベル。
scale_up_flag	フロー ベクターのスケール アップをイネーブルにするフラグ。1 つの画像ピラミッド レベルから別のピラミッド レベルに切り替える際にホストで設定されます。
scale_in	フロー ベクターをスケール アップするための浮動小数点スケール アップ係数。 この値は (previous_rows-1)/(current_rows-1) です。1 つの画像ピラミッド レベルから別のピラミッド レベルに切り替える際は、1 ではない値になります。

リソース使用量

次の表に、1920x1080 ピクセルの画像サイズに対してウィンドウ サイズ 11 で オプティカル フローを計算する場合の、各サイクル 1 ピクセルのインプリメンテーションにおける xFDensePyrOpticalFlow のリソース使用量を示します。これは、300 MHz のザイリンクス xczu9eg-ffvb1156-2L-e FPGA を Vivado HLS 2017.1 でインプリメンテーションした結果です。

表 172: xFDensePyrOpticalFlow 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり			
		LUT	FF	DSP	BRAM
1 ピクセル	300	32231	16596	52	215

パフォーマンス見積もり

次の表に、各レベルで係数 2 でスケール ダウンして 5 つのピラミッド レベルで 5 回のイテレーションを実行した場合の、ハードウェアでの xFDensePyrOpticalFlow 関数のパフォーマンス見積もりを示します。これは、zcu102 評価ボードでテストされています。

表 173: xFDensePyrOpticalFlow 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	画像サイズ	レイテンシ見積もり
			最大 (ms)
1 ピクセル	300	1920x1080	49.7
1 ピクセル	300	1280x720	22.9
1 ピクセル	300	1226x370	12.02

高密度非ピラミッド型 LK オプティカル フロー (xFDenseNonPyrLKOpticalFlow)

オプティカル フローは、オブジェクトまたはカメラの動きのため発生する 2 つの連続するフレーム間における画像オブジェクトの動きのパターンです。これは 2D ベクター フィールドであり、各ベクターは 1 つ目のフレームから 2 つ目のフレームの点の移動を表す変位ベクターです。

オプティカル フローは、次を前提として実行されます。

- オブジェクトのピクセル強度は、連続するフレーム間でそれほど変動しない
- 近傍ピクセルも同じように動く

最初のフレームにピクセル I(x, y, t) があるとします。3 つ目の次元として時間 (t) が追加されています。画像のみを処理する場合は、時間は必要ありません。時間 dt 後の次のフレームで、ピクセルが距離 (dx, dy) だけ移動します。これらのピクセルは同じであり強度は変化しないので、次のことが言えます。

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

右辺のテイラー級数近似を求め、共通項を取り除き、dt で割ると、次の式が得られます。

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

$f_x = \frac{\delta f}{\delta x}$ 、$f_y = \frac{\delta f}{\delta x}$ 、$u = \frac{dx}{dt}$ 、および $v = \frac{dy}{dt}$ です。

上記の式をオプティカル フローと呼びます。f_x および f_y は画像勾配で、f_t は時間の経過に伴う勾配です。ただし、(u, v) は不明です。これらの不明な変数のためこの方程式を解くことはできないので、この問題を解くために複数の方法が提供されています。1 つの方法は Lucas-Kanade (LK) です。先ほど近傍のすべてのピクセルが同じように動くと想定しました。Lucas-Kanade 法では、点を囲む WINDOW_SIZE テンプレート パラメーターで指定されたサイズのパッチが使用されます。つまり、パッチ内のすべての点が同じように動くと想定されます。これらの点に対しては、(f_x, f_y, f_t) を求めることが可能です。これで、問題は 2 つの不明な変数を含む WINDOW_SIZE * WINDOW_SIZE 方程式を解くことになり、これは重複決定されます。より良い方法は、最小二乗適合法を使用する方法です。次に、2 つの方程式と 2 つの不明な変数を含む問題の最終的な解を示します。

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 174: xFDenseNonPyrLKOpticalFlow 関数パラメーターの説明

パラメーター	説明
Type	ピクセル タイプ。サポートされるピクセル値は符号なし 8 ビット (XF_8UC1)。
ROWS	入力画像の最大行数。ハードウェア カーネルはこの行数用に構築する必要があります。
COLS	入力画像の最大列数。ハードウェア カーネルはこの列数用に構築する必要があります。
NPC	サイクルごとに処理されるピクセル数。XF_NPPC1 (=1) および XF_NPPC2(=2) をサポート。
WINDOW_SIZE	オプティカル フローを計算するウィンドウ サイズ。奇数の正の整数を指定可能。
frame0	1 つ目の入力画像。
frame1	2 つ目の入力画像。オプティカル フローは、frame0 と frame1 の間で計算されます。
flowx	フロー ベクターの水平要素。フロー ベクターのフォーマットは XF_32FC1 (単精度)。
flowy	フロー ベクターの垂直要素。フロー ベクターのフォーマットは XF_32FC1 (単精度)。

リソース使用量

次の表に、300 MHz の Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された 4K 画像の xFDenseNonPyrLKOpticalFlow のリソース使用量を示します。

表 175: xFDenseNonPyrLKOpticalFlow 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり			
		BRAM_18K	DSP_48E	FF	LUT
1 ピクセル	300	182	44	25336	21603
2 ピクセル	300	264	82	25740	17216

パフォーマンス見積もり

次の表に、xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された、4K 画像の xFDenseNonPyrLKOpticalFlow 関数のパフォーマンス見積もりを示します。

表 176: xFDenseNonPyrLKOpticalFlow 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり
		最大 (ms)
1 ピクセル	300	28.01
2 ピクセル	300	14.01

平均および標準偏差 (xFMeanStddev)

`xFMeanStddev` 関数では、入力画像の平均偏差と標準偏差が計算されます。出力される平均値のフォーマットは固定小数点の Q8.8 で、標準偏差値のフォーマットは Q8.8 です。平均および標準偏差は次のように計算されます。

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

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 177: xFmeanstd 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) をサポート。
ROWS	処理される画像の行数。
COLS	処理される画像の列数。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	入力画像
_mean	画像の平均値の計算結果を示す 16 ビットのデータ ポインター。
_stddev	画像の標準偏差の計算結果を示す 16 ビットのデータ ポインター。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された xFmeanstd 関数のリソース使用量を示します。

表 178: xFmeanstd 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	6	896	461	121
8 ピクセル	150	0	13	1180	985	208

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールで生成された異なる設定でのパフォーマンス見積もりを示します。

表 179: xFmeanstd 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ
1 ピクセル動作 (300 MHz)	6.9 ms
8 ピクセル動作 (150 MHz)	1.69 ms

メジアン フィルター (xFMedianBlur)

xFMedianBlur は、入力画像のメジアン フィルター処理を実行する関数です。メジアン フィルターは、ノイズ除去を改善する非線形のデジタル フィルターです。N サイズのフィルターの場合、各ピクセルごとに NxN 近隣ピクセル値の中央値が出力されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 180: xFMedianBlur 関数パラメーターの説明

パラメーター	説明
FILTER_SIZE	ハードウェア カーネルを構築するハードウェア フィルターのウィンドウ サイズ。1 を超える奇数の正の整数を指定可能。
BORDER_TYPE	ハードウェア カーネルで処理される境界タイプ。現時点でサポートされているのは XF_BORDER_REPLICATE のみ。
TYPE	入力ピクセルのタイプ。XF_8UC1 をサポート。
ROWS	処理される画像の行数。
COLS	処理される画像の列数。
NPC	並列で処理されるピクセル数。オプションは、XF_NPPC1 (毎クロック 1 ピクセルの処理の場合)、XF_NPPC8 (毎クロック 8 ピクセルの処理の場合)
_src	入力画像。
_dst	出力画像。

リソース使用量

次の表に、Xc7z020clg484-1 FPGA で Vivado HLS 2017.1 バージョン ツールを使用して生成された、XF_NPPC1 および XF_NPPC8 設定の xFMedianBlur 関数のリソース使用量を示します。

表 181: xFMedianBlur 関数のリソース使用量のサマリ

動作モード	FILTER_SIZE	動作周波数 (MHz)	使用量の見積もり			
			LUT	FF	DSP	BRAM
1 ピクセル	3	300	1197	771	0	3
8 ピクセル	3	150	6559	1595	0	6
1 ピクセル	5	300	5860	1886	0	5

パフォーマンス見積もり

次の表に、xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 バージョン ツールを使用して生成された xFMedianBlur のパフォーマンス見積もりを示します。

表 182: xFMedianBlur 関数のパフォーマンス見積もりのサマリ

動作モード	FILTER_SIZE	動作周波数 (MHz)	入力画像サイズ	レイテンシ見積もり
				最大 (ms)
1 ピクセル	3	300	1920x1080	6.99
8 ピクセル	3	150	1920x1080	1.75
1 ピクセル	5	300	1920x1080	7.00

MinMax ロケーション (xFminMaxLoc)

xFminMaxLoc 関数では、画像の最小値および最大値と、それらの値のロケーションが検出されます。

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

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

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 183: xFminMaxLoc 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1)、16 ビット、符号なし、1 チャネル (XF_16UC1)、16 ビット、符号付き、1 チャネル (XF_16SC1)、32 ビット、符号付き、1 チャネル (XF_32SC1) をサポート。
ROWS	処理される画像の行数。
COLS	処理される画像の列数。
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	入力画像
max_val	int 型の画像の最大値。
min_val	int 型の画像の最小値。
_minlocx	最初の最小値の X 軸の位置。
_minlocy	最初の最大値の Y 軸の位置。
_maxlocx	最初の最小値の X 軸の位置。
_maxlocy	最初の最大値の Y 軸の位置。

リソース使用量

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された xFminMaxLoc 関数のリソース使用量を示します。

表 184: xFminMaxLoc 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	3	451	398	86
8 ピクセル	150	0	3	1049	1025	220

パフォーマンス見積もり

次の表に、Xilinx Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 185: xFminMaxLoc 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ
1 ピクセル動作 (300 MHz)	6.9 ms
8 ピクセル動作 (150 MHz)	1.69 ms

平均値シフト トラッキング (xFMeanShift)

平均値シフト トラッキングは、基本的なオブジェクト トラッキング アルゴリズムの 1 つで、前に初期化されたモデルと局所的に最も類似したビデオ フレームのエリアを検索します。トラッキングされるオブジェクトは、ヒストグラムで示されます。オブジェクト トラッキング アルゴリズムでは、ターゲットは主に矩形または楕円形の領域で示され、ターゲット モデルとターゲット候補が含まれます。オブジェクトの特性を示すために、カラー ヒストグラムが使用されます。ターゲット モデルは通常その確率密度関数 (PDF) で示されます。重み付き RGB ヒストグラムが使用され、オブジェクト ピクセルにさらに重要度が追加されます。

平均値シフト アルゴリズムは、密度関数の最大値を見つける反復手法です。オブジェクト トラッキングの場合、密度関数はトラッキングされるオブジェクトとテストされるフレームのカラー ヒストグラムを使用して形成される重み付き画像です。重み付きヒストグラムを使用すると、通常のヒストグラム計算とは異なり、空間的な位置も考慮されます。この関数では、入力画像ポインター、短形オブジェクトの左上と右下の座標、フレーム数、およびトラッキング ステータスが入力として使用され、反復平均値シフト方法を使用して重心が返されます。

API 構文

```
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 );
```

テンプレート パラメーターの説明

次の表では、テンプレート パラメーターについて説明します。

表 186: xFmeanstd テンプレート パラメーター

パラメーター	説明
ROWS	画像の最大高さ
COLS	画像の最大幅
OBJ_ROWS	トラッキングされるオブジェクトの最大高さ
OBJ_COLS	トラッキングされるオブジェクトの最大幅
MAXOBJ	トラッキングされるオブジェクトの最大数
MAXITERS	最大反復数
SRC_T	入力 xf::Mat のタイプで、XF_8UC4 (4 チャネルの 8 ビット データ)
NPC	サイクルごとに処理されるピクセル数。XF_NPPC1 (サイクルごとに 1 ピクセルの操作) のみサポート。

関数パラメーターの説明

次の表では、関数パラメーターについて説明します。

表 187: xFMeanShift 関数パラメーター

パラメーター	説明
_in_mat	入力 xF Mat
x1	すべてのオブジェクトの左上角の X 座標 (行番号)
y1	すべてのオブジェクトの左上角の Y 座標 (列番号)
obj_height	すべてのオブジェクトの高さ (奇数)
obj_width	すべてのオブジェクトの幅 (奇数)
dx	カーネル関数で戻されたすべてのオブジェクトの X 軸の中点
dy	カーネル関数で戻されたすべてのオブジェクトの Y 軸の中点
status	オブジェクトのステータスが true の場合にのみオブジェクトをトラッキング。オブジェクトがフレーム外の場合、ステータスは 0。
frame_status	最初のフレームは 0、その他のフレームは 1 に設定。
no_objects	トラッキングするオブジェクト数。
no_iters	反復数。

リソース使用量およびパフォーマンスの見積もり

次の表に、300 MHz の xczu9eg-ffvb1156-i-es1 で解像度 1920x1080、10 個のオブジェクト (サイズ 250x250、4 反復) の RGB 画像を処理するために、Vivado HLS 2017.1 リリース ツールを使用して生成された標準 (1 ピクセル) 設定の xFMeanShift() のリソース使用量を示します。

表 188: xFMeanShift 関数のリソース使用量とパフォーマンス見積もりのサマリ

設定	最大レイテンシ	BRAM	DSP	FF	LUT
1 ピクセル	19.28	76	14	13198	10064

制限

トラッキングできるオブジェクトの最大数は 10 です。

Otsu 法のしきい値処理 (xFOtsuThreshold)

Otsu 法のしきい値処理では、クラスタリング ベースの画像のしきい値処理またはグレーレベル画像のバイナリ画像への縮小が自動的に実行されます。このアルゴリズムでは、2 クラスのピクセルの後に、バイモーダル ヒストグラム (前景ピクセルと背景ピクセル) が含まれ、その 2 つのクラスを分離する最適なしきい値が計算されます。

Otsu 法では、2 つのクラスを分離するクラス内分散 (2 つのクラスの分散の加重和で定義) を最小にできるしきい値を見つけます。

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

w_1 はヒストグラムから計算されたクラス確率です。

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

Otsu 法では、クラス内分散の最小化がクラス間分散の最大化と同じであることが示されます。

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

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

$$\mu_b = \left[\sum_{1}^{t} p(i)x(i)\right] \Big/ w_1 \quad , \qquad \mu_f = \left[\sum_{t+1}^{I} p(i)x(i)\right] \Big/ w_2$$

はクラス平均です。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 189: xFOtsuThreshold 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_thresh	計算後の出力しきい値

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された xFOtsuThreshold 関数のリソース使用量を示します。

表 190: xFOtsuThreshold 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	8	49	2239	3353	653
8 ピクセル	150	22	49	1106	3615	704

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 191: xFOtsuThreshold 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.92 ms
8 ピクセル動作 (150 MHz)	1.76 ms

ピクセル加算 (xFadd)

xFadd 関数は、2 つの入力画像間でピクセル単位の加算を実行して、出力画像を返します。

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

説明:

- $I_{out}(x, y)$: 出力画像の (x,y) 位置での強度
- $I_{in1}(x, y)$: 最初の入力画像の (x,y) 位置での強度
- $I_{in2}(x, y)$: 2 つ目の入力画像の (x,y) 位置での強度

XF_CONVERT_POLICY_TRUNCATE: 結果は出力オペランドの LSB で、そのビット深さのサイズの 2 の補数バイナリ フォーマットで保存されます。

XF_CONVERT_POLICY_SATURATE: 結果は出力オペランドのビットの深さに飽和されます。

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 192: xFadd 関数パラメーターの説明

パラメーター	説明
POLICY_TYPE	オーバーフロー処理のタイプ。XF_CONVERT_POLICY_SATURATE または XF_CONVERT_POLICY_TRUNCATE のいずれかに設定可能。
SRC_T	ピクセル タイプ。オプションは XF_8UC1 および XF_16SC1。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのリソース使用量を示します。

表 193: xFadd 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	62	55	11
8 ピクセル	150	0	0	65	138	24

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 194: xFadd 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

ピクセル乗算 (xFmultiply)

`xFmultiply` 関数は、2 つの入力画像間でピクセル単位の乗算を実行して、出力画像を返します。

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

説明:

- $I_{out}(x, y)$: 出力画像の (x,y) 位置での強度
- $I_{in1}(x, y)$: 最初の入力画像の (x,y) 位置での強度
- $I_{in2}(x, y)$: 2 つ目の入力画像の (x,y) 位置での強度
- scale_val: スケール値

XF_CONVERT_POLICY_TRUNCATE: 結果は出力オペランドの LSB で、そのビット深さのサイズの 2 の補数バイナリ フォーマットで保存されます。

XF_CONVERT_POLICY_SATURATE: 結果は出力オペランドのビットの深さに飽和されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 195: xFmultiply 関数パラメーターの説明

パラメーター	説明
POLICY_TYPE	オーバーフロー処理のタイプ。XF_CONVERT_POLICY_SATURATE または XF_CONVERT_POLICY_TRUNCATE のいずれかに設定可能。
SRC_T	ピクセル タイプ。オプションは XF_8UC1 および XF_16SC1。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像
scale_val	0 ～ 1 の重み係数

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのリソース使用量を示します。

表 196: xFmultiply 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	2	124	59	18
8 ピクセル	150	0	16	285	108	43

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 197: xFmultiply 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.6

ピクセル減算 (xFsubtract)

xFsubtract 関数は、2 つの入力画像間でピクセル単位の減算を実行して、出力画像を返します。

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

説明:

- $I_{out}(x, y)$: 出力画像の (x,y) 位置での強度
- $I_{in1}(x, y)$: 最初の入力画像の (x,y) 位置での強度
- $I_{in2}(x, y)$: 2 つ目の入力画像の (x,y) 位置での強度

XF_CONVERT_POLICY_TRUNCATE: 結果は出力オペランドの LSB で、そのビット深さのサイズの 2 の補数バイナリ フォーマットで保存されます。

XF_CONVERT_POLICY_SATURATE: 結果は出力オペランドのビットの深さに飽和されます。

API 構文

```
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 )
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 198: xFsubtract 関数パラメーターの説明

パラメーター	説明
POLICY_TYPE	オーバーフロー処理のタイプ。XF_CONVERT_POLICY_SATURATE または XF_CONVERT_POLICY_TRUNCATE のいずれかに設定可能。
SRC_T	ピクセル タイプ。オプションは XF_8UC1 および XF_16SC1。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
src1	入力画像
src2	入力画像
dst	出力画像

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのリソース使用量を示します。

表 199: xFsubtract 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	0	0	62	53	11
8 ピクセル	150	0	0	59	13	21

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 200: xFsubtract 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ (ms)
1 ピクセル動作 (300 MHz)	6.9
8 ピクセル動作 (150 MHz)	1.7

リマップ (xFRemap)

xFRemap 関数は、画像内の 1 つの位置からピクセルを取り出して、別の画像の別の位置に移動します。移動前の画像から移動後の画像へのマップには、2 種類の補間方法が使用されます。

$$dst = src\left(map_x(x, y), map_y(x, y)\right)$$

API 構文

```
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);
```

パラメーターの説明

次の表では、テンプレート パラメーターについて説明します。

表 201: xFRemap テンプレート パラメーターの説明

パラメーター	説明
WIN_ROWS	内部でバッファリングされる入力画像の行数。マップ データに基づいて設定する必要あり。たとえば、左右反転には 2 行で十分。
SRC_T	入力画像タイプ。8 ビットで 1 チャネルのグレースケール画像。XF_8UC1。
MAP_T	マップ タイプ。1 チャネルの浮動小数点型。XF_32FC1。
DST_T	出力画像タイプ。8 ビットで 1 チャネルのグレースケール画像。XF_8UC1。
ROWS	入力および出力画像の高さ。
COLS	入力および出力画像の幅。
NPC	サイクルごとに処理されるピクセル数。XF_NPPC1 (サイクルごとに 1 ピクセルの操作) のみサポート。

次の表では、関数パラメーターについて説明します。

表 202: xFRemap 関数パラメーターの説明

パラメーター	説明
_src_mat	入力 xF Mat
_remapped_mat	出力 xF Mat
_mapx_mat	浮動小数点型の mapX Mat
_mapy_mat	浮動小数点型の mapY Mat
補間	補間タイプは XF_INTERPOLATION_NN (最近傍補間) または XF_INTERPOLATION_BILINEAR (バイリニア補間) のいずれか。

リソース使用量

次の表に、xczu9eg-ffvb1156-i-es1 FPGA 用に 300 MHz、WIN_ROWS 64 で Vivado HLS 2017.1 ツールを使用して生成された HD (1080x1920) 画像の xFRemap のリソース使用量を示します。

表 203: xFRemap 関数のリソース使用量のサマリ

名前	リソース使用量
BRAM_18K	128
DSP48E	14
FF	2064
LUT	2277
CLB	500

パフォーマンス見積もり

次の表に、xczu9eg-ffvb1156-i-es1 FPGA 用に 300 MHz、WIN_ROWS 64 で Vivado HLS 2017.1 ツールを使用して生成された HD (1080x1920) 画像の xFRemap() のパフォーマンス見積もりを示します。

表 204: xFRemap 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり 最大レイテンシ (ms)
1 ピクセル モード	300	7.2

解像度変換/リサイズ (xFResize)

解像度変換は、元の画像のサイズをターゲットの画像サイズに変換するのに使用される方法です。リサイズ関数には、最近傍補間、バイリニア補間、エリア補間など、さまざまな補間方法を使用できます。補間のタイプは、API にテンプレート パラメーターとして渡すことができます。補間タイプを指定するには、次の列挙型を使用できます。

- XF_INTERPOLATION_NN – 最近傍補間
- XF_INTERPOLATION_BILINEAR – バイリニア補間
- XF_INTERPOLATION_AREA – エリア補間

注記: ダウンスケールには 0.25 以上、アップスケールには 8 以下のスケール係数がサポートされます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 205: xFResize 関数パラメーターの説明

パラメーター	説明
INTERPOLATION_TYPE	補間タイプ。使用可能なオプションは次のとおり。 ・ XF_INTERPOLATION_NN - 最近傍補間 ・ XF_INTERPOLATION_BILINEAR - バイリニア補間 ・ XF_INTERPOLATION_AREA - エリア補間
TYPE	ピクセルごとのビット数。XF_8UC1 のみサポート。
SRC_ROWS	ハードウェア カーネルを構築する入力画像の最大高さ。
SRC_COLS	ハードウェア カーネルを構築する入力画像の最大幅(8 の倍数を指定)
DST_ROWS	ハードウェア カーネルを構築する出力画像の最大高さ。
DST_COLS	ハードウェア カーネルを構築する出力画像の最大幅(8 の倍数を指定)
NPC	サイクルごとに処理されるピクセル数。可能なオプションは XF_NPPC1 (各サイクル 1 ピクセル) および XF_NPPC8 (各サイクル 8 ピクセル)。
_src	入力画像
_dst	出力画像

リソース使用量

次の表に、xczu9eg-ffvb1156-2-i-es2 FPGA でグレースケール HD (1080x1920) 画像を SD(640x480) にダウンスケールし、HD(1920x1080) 画像を 4K(3840x2160) 画像にアップスケールするために Vivado HLS 2017.1 ツールを使用して生成されたリソース最適化 (8 ピクセル) モードおよび通常モードでのリサイズ関数のリソース使用量を示します。

表 206: xFResize 関数のリソース使用量のサマリ

動作モード	使用量の見積もり							
	1 ピクセル (300 MHz)				8 ピクセル (150MHz)			
	LUT	FF	DSP	BRAM	LUT	FF	DSP	BRAM
最近傍ダウンスケール	800	1315	8	4	5079	5720	8	11
バイリニア ダウンスケール	1067	1580	10	7	6511	4863	14	23
エリア ダウンスケール	2558	2995	42	30	324991	17726	42	79
最近傍アップスケール	803	1115	8	8	1599	1636	8	20
バイリニア アップスケール	1231	1521	10	15	3588	2662	14	37
エリア アップスケール	1461	2107	16	25	5861	3611	36	40

パフォーマンス見積もり

次の表に、300 MHz の xczu9eg-ffvb1156-2-i-es2 FPGA でグレースケール画像を 1080x1920 から 480x640 にリサイズ (ダウンスケール) し、1080x1920 から 2160x3840 にリサイズ (アップスケール) するために Vivado HLS 2017.1 ツールを使用して生成されたさまざまな設定のリサイズ関数のパフォーマンス見積もりを示します。補間タイプ別のレイテンシも示します。

表 207: xFResize 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり (ms)					
		ダウンスケール 最近傍補間 (NN)	ダウンスケール バイリニア補間	ダウンスケール エリア補間	アップスケール 最近傍補間 (NN)	アップスケール バイリニア補間	アップスケール エリア補間
1 ピクセル	300	6.94	6.97	7.09	27.71	27.75	27.74

Scharr フィルター (xFScharr)

xFScharr 関数では、処理される入力画像でカーネルをたたみ込むことで、x と y の両方の方向で入力画像の勾配が計算されます。

カーネル サイズ 3x3 の場合:

- 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 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 208: xFScharr 関数パラメーターの説明

パラメーター	説明
BORDER_TYPE	サポートされる境界タイプは XF_BORDER_CONSTANT。
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセル タイプ。16 ビット、符号付き、1 チャネル (XF_16SC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_matx	X 勾配の出力画像。
_dst_maty	Y 勾配の出力画像。

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 209: xFScharr 関数のリソース使用量のサマリ

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	3	6
DSP48E	0	0
FF	728	1434
LUT	812	2481
CLB	171	461

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 210: xFScharr 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり (ms)
1 ピクセル	300	7.2
8 ピクセル	150	1.7

Sobel フィルター (xFSobel)

xFSobel 関数では、処理される入力画像でカーネルをたたみ込むことで、x と y の両方の方向で入力画像の勾配が計算されます。

- カーネル サイズ 3x3 の場合:
  - GradientX:

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

  - GradientY:

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

- カーネル サイズ 5x5 の場合:
  - 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:

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

- カーネル サイズ 7x7 の場合:
  - 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 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 211: xFSobel 関数パラメーターの説明

パラメーター	説明
FILTER_TYPE	フィルター サイズ。サポートされるフィルター サイズは 3 (XF_FILTER_3X3)、5 (XF_FILTER_5X5)、および 7 (XF_FILTER_7X7)。
BORDER_TYPE	サポートされる境界タイプは XF_BORDER_CONSTANT。
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力ピクセル タイプ。フィルター サイズ 3 および 5 の場合は、16 ビット、符号付き、1 チャネルのみサポート。フィルター サイズ 7 の場合は、32 ビット、符号付き、1 チャネルのみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_matx	X 勾配の出力画像。
_dst_maty	Y 勾配の出力画像。

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 212: xFSobel 関数のリソース使用量のサマリ

動作モード	フィルター サイズ	動作周波数 (MHz)	使用量の見積もり				
			BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	3x3	300	3	0	609	616	135
	5x5	300	5	0	1133	1499	308
	7x7	300	7	0	2658	3334	632
8 ピクセル	3x3	150	6	0	1159	1892	341
	5x5	150	10	0	3024	5801	999

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 213: xFSobel 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	フィルター サイズ	レイテンシ見積もり (ms)
1 ピクセル	300	3x3	7.5
	300	5x5	7.5
	300	7x7	7.5
8 ピクセル	150	3x3	1.7
	150	5x5	1.71

ステレオ ローカル ブロック マッチング (xFFindStereoCorrespondenceBM)

ステレオ ブロック マッチングは、連続するフレーム (ステレオ ペア) 間のブロックの動きを推定します。この前提となるのは、ステレオ ペアでは、前景のオブジェクトの方が背景のオブジェクトよりも視差が大きいということです。ローカル ブロック マッチングでは、ウィンドウ サイズに基づく近傍パッチの情報を使用して、ステレオ ペアの共役点を特定します。グローバル マッチングでは、マッチング ピクセルを計算するのに画像全体からの情報が使用されるので、ローカル マッチングよりも高い精度が得られますが、グローバル マッチングではリソースが多く使用されるので、その点ではローカル マッチングの方が有利です。

ローカル ブロック マッチング アルゴリズムには、前処理段階と視差推定段階があります。前処理では、Sobel 勾配計算が実行された後、イメージ クリッピングが実行されます。視差推定では、SAD (絶対差の和) 計算を実行し、WTA (Winner Takes All) 方式を使用して視差を取得します (最小の SAD が視差)。ほかの可能な視差と異なる場合、ピクセルは無効になります。無効なピクセルは、視差値 0 で示されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 214: xFFindStereoCorrespondenceBM 関数パラメーターの説明

パラメーター	説明
WSIZE	視差計算に使用されるウィンドウのサイズ
NDISP	視差の数
NDISP_UNITS	並列計算される視差の数。
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
DST_T	出力タイプ。XF_16UC1 で、視差が Q12.4 フォーマットで配列されます。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。可能なオプションは XF_NPPC1 のみ。
left_image	左カメラからの画像
right_image	右カメラからの画像
disparity_image	画像の形式で出力された視差。
sbmstate	ステレオ ブロック マッチング アルゴリズムに関するさまざまなパラメーターで構成されるクラス オブジェクト。 1. preFilterCap: デフォルト値は 31 で、有効な値は 1 ～ 63 です。 2. minDisparity: デフォルト値は 0 で、有効な値は 0 ～ (imgWidth-NDISP) です。 3. uniquenessRatio: デフォルト値は 15 で、有効な値は負でない整数です。 4. textureThreshold: デフォルト値は 10 で、有効な値は負でない整数です。

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

設定のフォーマットは imageSize_SADwinSize_NDisp_NDispUnits です。

表 215: xFFindStereoCorrespondenceBM 関数のリソース使用量のサマリ

設定	周波数 (MHz)	リソース使用量			
		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
HD_15_128_32	300	57	20	41017	35176
HD_21_64_16	300	69	20	29853	30706

パフォーマンス見積もり

次の表に、ザイリンクス Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのステレオ ローカル ブロック マッチングのパフォーマンス見積もりを示します。

設定のフォーマットは imageSize_SADwinSize_NDisp_NDispUnits です。

表 216: xFFindStereoCorrespondenceBM 関数のパフォーマンス見積もりのサマリ

設定	周波数 (MHz)	レイテンシ (ms)	
		最小	最大
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 (xFSVM)

xFSVM 関数は SVM コアの演算で、入力配列間のドット積を実行します。この関数は、固定小数点型のドット積値の結果を戻します。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 217: xFSVM 関数パラメーターの説明

パラメーター	説明
SRC1_T	入力ピクセル タイプ。16 ビット、符号付き、1 チャネル (XF_16SC1) をサポート。
SRC2_T	入力ピクセル タイプ。16 ビット、符号付き、1 チャネル (XF_16SC1) をサポート。
DST_T	出力データ型。32 ビット、符号付き、1 チャネル (XF_32SC1) をサポート。
ROWS1	処理される最初の画像の行数。
COLS1	処理される最初の画像の列数。
ROWS2	処理される 2 つ目の画像の行数。
COLS2	処理される 2 つ目の画像の列数。
NPC	サイクルごとに処理されるピクセル数。可能なオプションは XF_NPPC1。
N	カーネル処理の最大数。
in_1	最初の入力配列。
in_2	2 つ目の入力配列。
idx1	最初の配列のインデックスの開始。
idx2	2 つ目の配列のインデックスの開始。
frac1	最初の配列データの小数ビット数。
frac2	2 つ目の配列データの小数ビット数。
n	カーネル処理の数。
out_frac	結果の値の小数ビット数。
result	結果の値。

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された xFSVM 関数のリソース使用量を示します。

表 218: xFSVM 関数のリソース使用量のサマリ

動作周波数 (MHz)	使用量の見積もり (ms)				
	BRAM_18K	DSP_48E	FF	LUT	CLB
300	0	1	27	34	12

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 219: xFSVM 関数のパフォーマンス見積もりのサマリ

動作周波数 (MHz)	レイテンシ見積もり	
	最小 (サイクル数)	最大 (サイクル数)
300	204	204

しきい値処理 (xFThreshold)

xFThreshold 関数は、入力画像のしきい値処理を実行する関数です。しきい値処理には、バイナリとレンジの 2 種類があります。

バイナリしきい値処理の場合、しきい値が設定され、そのしきい値によって出力が 0 か 255 のいずれかに設定されます。バイナリしきい値処理を選択したら、_binary_thresh_val はそのしきい値に、_upper_range および _lower_range は 0 に設定する必要があります。

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

レンジしきい値処理の場合、上限および下限のしきい値が設定され、これらの値に基づいて、出力が 0 か 255 のいずれかに設定されます。レンジしきい値処理を選択したら、_upper_range は上限しきい値に、_lower_range 値は下限しきい値に設定し、_binary_thresh_val を 0 に設定する必要があります。

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

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 220: xFThreshold 関数パラメーターの説明

パラメーター	説明
THRESHOLD_TYPE	しきい値処理のタイプ。バイナリしきい値またはレンジしきい値のいずれかにできます。オプションは XF_THRESHOLD_TYPE_BINARY または XF_THRESHOLD_TYPE_RANGE です。
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。XF_NPPC1 (サイクルごとに 1 ピクセルの操作) のみサポート。
_src_mat	入力画像
_dst_mat	出力画像

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 221: xFThreshold 関数のリソース使用量のサマリ

c	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	0	0
DSP48E	3	3
FF	410	469
LUT	277	443
CLB	72	103

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 222: xFThreshold 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり (ms)
1 ピクセル	300	7.2
8 ピクセル	150	1.7

WarpAffine (xFwarpAffine)

アフィン変換は、回転、スケーリング、平行移動などを表すために使用されます。

xFWarpAffine は、逆変換を入力として使用し、その入力画像を 2x3 行列の行列乗算で変換します。各出力ピクセルに対する入力位置は次のように計算されます。

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

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

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

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

注記: 0.25 以上または 8 以下のスケーリングを前もって実行でき、渡された変換行列にサポートされていない倍率がある場合は、アサーションが表示されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 223: xFwarpAffine 関数パラメーターの説明

パラメーター	説明
INTERPOLATION_TYPE	使用する補間方法 XF_INTERPOLATION_NN: 最近傍 (Nearest Neighbor) 補間 XF_INTERPOLATION_BILINEAR: バイリニア補間
SRC_T	入力ピクセル タイプ。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	入力画像ポインター
_dst	出力画像ポインター
transformation_matrix	逆アフィン変換行列

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 用に Vivado HLS 2017.1 ツールを使用して生成された異なる設定での xFWarpAffine のリソース使用量を示します。

表 224: xFwarpAffine 関数のリソース使用量のサマリ

名前	使用量 (250 MHz)			
	通常動作 (1 ピクセル) モード		リソース最適化 (8 ピクセル) モード	
	最近傍補間	バイリニア補間	最近傍補間	バイリニア補間
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

注記: NO は 1 ピクセル処理、RO は 4 ピクセル処理。

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 用に Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのアフィン変換のパフォーマンス見積もりを示します。

表 225: xFwarpAffine 関数のパフォーマンス見積もりのサマリ

レイテンシ見積もり			
1 ピクセル (300 MHz)		8 ピクセル (150 MHz)	
最近傍補間	バイリニア補間 補間	最近傍補間	バイリニア補間 補間
最大 (ms)	最大 (ms)	最大 (ms)	最大 (ms)
10.4	19.3	10.7	10.4

WarpPerspective (xFperspective)

xFperspective 関数は、画像の透視変換を実行します。逆変換を入力として取り込み、3x3 行列の透視変換を適用します。

各出力ピクセルに対する入力位置は、次のように計算されます。

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

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

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

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

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

注記: 0.25 以上または 8 以下のスケーリングを前もって実行でき、渡された変換行列にサポートされていない倍率がある場合は、アサーションが表示されます。

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 226: xFperspective 関数パラメーターの説明

パラメーター	説明
INTERPOLATION_TYPE	使用する補間方法 XF_INTERPOLATION_NN: 最近傍 (Nearest Neighbor) 補間 XF_INTERPOLATION_BILINEAR: バイリニア補間
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ (8 の倍数で指定)
COLS	入力および出力画像の最大幅 (8 の倍数で指定)
NPC	サイクルごとに処理されるピクセル数。1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src_mat	入力画像
_dst_mat	出力画像
transformation_matrix	逆透視変換行列

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でバイリニア補間のグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのリソース使用量を示します。

表 227: xFperspective 関数のリソース使用量のサマリ

名前	リソース使用量	
	1 ピクセル	8 ピクセル
	300 MHz	150 MHz
BRAM_18K	223	233
DSP48E	191	293
FF	17208	14330
LUT	14458	18969
CLB	3230	3876

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でバイリニア補間のグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのカーネルのパフォーマンス見積もりを示します。

表 228: xFperspective 関数のパフォーマンス見積もりのサマリ

動作モード	動作周波数 (MHz)	レイテンシ見積もり 最大 (ms)
1 ピクセル	300	19.3
8 ピクセル	150	15.5

Atan2 (xFAtan2LookupFP)

xFAtan2LookupFP 関数は、y/x の逆正接関数を計算します。ベクター $\begin{bmatrix} x \\ y \end{bmatrix}$ の原点に対する角度を返します。atan2 により返される角度には、象限情報も含まれます。

xFAtan2LookupFP 関数は、標準の atan2 関数の固定小数点バージョンです。この関数は、ルックアップ テーブル方法を使用して atan2 をインプリメントします。ルックアップ テーブルの値は Q4.12 フォーマットで表現されるので、返される値も Q4.12 フォーマットです。glibc に含まれる atan2 関数と比較すると、89 ～ 90 度の範囲で 0.2 度の最大誤差があります。それ以外の角度 (0 ～ 89) の最大誤差は 10-3 程度です。xs と ys の両方が 0 の場合は、0 が返されます。

API 構文

```
short xFAtan2LookupFP(short xs, short ys, int M1,int N1,int M2, int N2)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 229: xFAtan2LookupFP 関数パラメーターの説明

パラメーター	説明
xs	QM1.N1 の固定小数点フォーマットの 16 ビット符号なし値 x
ys	QM2.N2 の固定小数点フォーマットの 16 ビット符号なし値 y
M1	x の整数部分を表すビット数。
N1	y の分数部分を表すビット数。16-M1 にする必要があります。
M2	y の整数部分を表すビット数
N2	y の分数部分を表すビット数。16-N1 にする必要があります。
Return	戻り値 (ラジアン)。Q4.12 の固定小数点フォーマットで -pi ～ +pi の範囲です。

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された xFAtan2LookupFP 関数のリソース使用量を示します。

表 230: xFAtan2LookupFP 関数のリソース使用量のサマリ

動作周波数 (MHz)	使用量の見積もり				
	BRAM_18K	DSP_48E	FF	LUT	CLB
300	4	2	275	75	139

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 231: xFAtan2LookupFP 関数のパフォーマンス見積もりのサマリ

動作周波数 (MHz)	レイテンシ見積もり	
	最小 (サイクル数)	最大 (サイクル数)
300	1	15

インバース/逆数 (xFInverse)

xFInverse 関数では、数値 x の逆数が計算されます。1/x 値は 2048 サイズのルックアップ テーブルに格納されています。1/x 値を選択するインデックスは、x の固定小数点フォーマットを使用して計算されます。このインデックスが計算されると、ルックアップ テーブルから対応する 1/x 値がフェッチされ、この値とこの値を固定小数点で表すために必要な小数ビット数が返されます。

API 構文

```
unsigned int xFInverse(unsigned short x,int M,char *N)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 232: xFInverse 関数パラメーターの説明

パラメーター	説明
x	QM の固定小数点フォーマットの 16 ビット符号なし値 x (16-M)
M	x の整数部分を表すビット数。
N	1/x の小数部分を表すためのビット数を格納する char 変数へのポインター。この値が関数から返されます。
Return	Q(32-N).N の固定小数点フォーマットで表された 32 ビット フォーマットで返される 1/x 値

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された xFInverse 関数のリソース使用量を示します。

表 233: xFInverse 関数のリソース使用量のサマリ

動作周波数 (MHz)	使用量の見積もり (ms)				
	BRAM_18K	DSP_48E	FF	LUT	CLB
300	4	0	68	128	22

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 234: xFInverse 関数のパフォーマンス見積もりのサマリ

動作周波数 (MHz)	レイテンシ見積もり	
	最小 (サイクル数)	最大 (サイクル数)
300	1	8

ルックアップ テーブル (xFLUT)

xFLUT 関数では、表のルックアップ処理が実行されます。指定したルックアップ テーブルを使用してソース画像がデスティネーション画像に変換されます。入力画像は AU_8UP の深さで、出力画像タイプも入力画像タイプと同じである必要があります。

$I_{out}(x, y) = LUT [I_{in1}(x, y)]$

説明:

- $I_{out}(x, y)$: 出力画像の (x,y) 位置での強度
- $I_{in}(x, y)$: 最初の入力画像の (x,y) 位置での強度
- LUT: サイズ 256 および符号なしの char 型のルックアップ テーブル

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 235: xFLUT 関数パラメーターの説明

パラメーター	説明
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) をサポート。
ROWS	処理される画像の行数。
COLS	処理される画像の列数。
NPC	並列で処理されるピクセル数。使用可能なオプションは、1 ピクセルの場合は XF_NPPC1、8 ピクセルの場合は XF_NPPC8。
_src	サイズ (ROWS, COLS) および 8U 型の入力画像。
_dst	サイズ (ROWS, COLS) で入力と同型の出力画像。
_lut	サイズ 256 および符号なしの char 型のルックアップ テーブル。

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された xFLUT 関数のリソース使用量を示します。

表 236: xFLUT 関数のリソース使用量のサマリ

動作モード	動作周波数 (MHz)	使用量の見積もり				
		BRAM_18K	DSP_48E	FF	LUT	CLB
1 ピクセル	300	1	0	937	565	137
8 ピクセル	150	9	0	1109	679	162

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 237: xFLUT 関数のパフォーマンス見積もりのサマリ

動作モード	レイテンシ見積もり
	最大レイテンシ
1 ピクセル動作 (300 MHz)	6.92 ms
8 ピクセル動作 (150 MHz)	1.66 ms

平方根 (xFSqrt)

xFSqrt 関数では、引き放し法の平方根アルゴリズムを使用して 16 ビットの固定小数点の平方根が計算されます。引き放し法の平方根アルゴリズムでは、平方根結果に 2 の補数表記が使用されます。このアルゴリズムでは、イテレーションごとに最後のビットでも正確な結果値を生成できます。

入力引数 D は、32 ビットと宣言されていますが、16 ビットにする必要があります。出力 sqrt(D) は 16 ビット型です。D のフォーマットが QM.N (M+N = 16) の場合、出力フォーマットは Q(M/2).N です。

小数部の n ビットの精度は、関数呼び出し前に被開数 (D) を左に 2n シフトし、その解を右に n シフトすると取得できます。たとえば、35 (011000112) の平方根を小数点後のビット数 1 (N=1) で求める方法は次のとおりです。

1. まず 01100011002 を左に 2 シフト。
2. 得られた値 (10112) を右に 1 シフト。正しい答えは 101.1 (5.5) となります。

API 構文

```
int xFSqrt(unsigned int D)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 238: xFSqrt 関数パラメーターの説明

パラメーター	説明
D	16 ビット固定小数点フォーマットの入力データ。
Return	short int フォーマットの出力値。

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された xFSqrt 関数のリソース使用量を示します。

表 239: xFSqrt 関数のリソース使用量のサマリ

動作周波数 (MHz)	使用量の見積もり				
	BRAM_18K	DSP_48E	FF	LUT	CLB
300	0	0	8	6	1

パフォーマンス見積もり

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA 用に Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのパフォーマンス見積もりを示します。

表 240: xFSqrt 関数のパフォーマンス見積もりのサマリ

動作周波数 (MHz)	レイテンシ見積もり	
	最小 (サイクル数)	最大 (サイクル数)
300	18	18

ワープ変換 (xFWarpTransform)

`xFWarpTransform` 関数は、画像に対して透視変換およびアフィン幾何変換を実行します。変換のタイプは、関数に対するコンパイル時パラメーターです。

この関数では、ストリーミング インターフェイスを使用して変換を実行します。このことと、幾何変換で 1 つの出力行を計算するのに入力データの多数の行にアクセスする必要があるということから、入力データの一部の行がブロック RAM に保存されます。保存される行数は、テンプレート パラメーターを使用して設定できます。変換行列に基づいて、保存する行数を決定できます。また、入力画像の変換をいつ開始するかを保存される画像の行数を単位として指定できます。

アフィン変換

変換行列は、次に示すように、サイズ パラメーターで構成されます。

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \end{bmatrix}$$

xFWarpTransform 関数では、アフィン変換は次の式に従って適用されます。

$$dst\begin{pmatrix} x \\ y \end{pmatrix} = M * src\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

透視変換

変換行列は、次に示すように 3x3 行列です。

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

xFWarpTransform 関数では、透視変換は次の式に従って適用されます。

$$dst^{1}\begin{pmatrix} x \\ y \\ n \end{pmatrix} = M * src\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

その後、dst1 の最初の 2 つの次元を 3 つ目の次元で割って変換後のピクセルが計算されます。

$$dst^{1}\begin{pmatrix} x \\ y \\ n \end{pmatrix} = M * src\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

API 構文

```
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)
```

パラメーターの説明

次の表に、テンプレートと関数パラメーターを説明します。

表 241: xFWarpTransform 関数パラメーターの説明

パラメーター	説明
STORE_LINES	FPGA でローカルに格納する必要のある画像の行数。
START_ROW	画像の変換を開始する前に保存する入力行の数。STORE_LINES の値以下にする必要があります。
TRANFORMATION_TYPE	アフィン変換および透視変換がサポートされます。0 に設定するとアフィン変換、1 に設定すると透視変換が実行されます。
INTERPOLATION_TYPE	1 に設定するとバイリニア補間、0 に設定すると最近傍補間が使用されま

パラメーター	説明
	す。
SRC_T	入力ピクセル タイプ。8 ビット、符号なし、1 チャネル (XF_8UC1) のみサポート。
ROWS	入力および出力画像の最大高さ。
COLS	入力および出力画像の最大幅。
NPC	サイクルごとに処理されるピクセル数。可能なオプションは 1 ピクセル操作 (XF_NPPC1) のみ。
src	入力画像
dst	出力画像
transformation_matrix	入力画像に適用する変換行列。

リソース使用量

次の表に、Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成された異なる設定でのワープ変換のリソース使用量を示します。

表 242: xFWarpTransform 関数のリソース使用量のサマリ

変換	INTERPOLATION_TYPE	STORE_LINES	START_ROW	動作周波数 (MHz)	使用量の見積もり			
					LUT	FF	DSP	BRAM
透視変換	バイリニア補間	100	50	300	7468	9804	61	112
透視変換	最近傍補間	100	50	300	4514	6761	35	104
アフィン変換	バイリニア補間	100	50	300	6139	5606	40	124
アフィン変換	最近傍補間	100	50	300	4611	4589	18	112

パフォーマンス見積もり

次の表に、ザイリンクス Xczu9eg-ffvb1156-1-i-es1 FPGA でグレースケール HD (1080x1920) 画像を処理するために、Vivado HLS 2017.1 ツールを使用して生成されたワープ変換のパフォーマンス見積もりを示します。

表 243: xFWarpTransform 関数のパフォーマンス見積もりのサマリ

変換	INTERPOLATION_TYPE	STORE_LINES	START_ROW	動作周波数 (MHz)	レイテンシ見積もり 最大 (ms)
透視変換	バイリニア補間	100	50	300	7.46
透視変換	最近傍補間	100	50	300	7.31
アフィン変換	バイリニア補間	100	50	300	7.31
アフィン変換	最近傍補間	100	50	300	7.24

付録 A

その他のリソースおよび法的通知

ザイリンクス リソース

アンサー、資料、ダウンロード、フォーラムなどのサポート リソースは、ザイリンクス サポート サイトを参照してください。

ソリューション センター

デバイス、ツール、IP のサポートについては、ザイリンクス ソリューション センターを参照してください。デザイン アシスタント、デザイン アドバイザリ、トラブルシューティングのヒントなどが含まれます。

参考資料

このガイドの補足情報は、次の資料を参照してください。

日本語版のバージョンは、英語版より古い場合があります。

1. 『SDx 環境リリース ノート、インストールおよびライセンス ガイド』(UG1238)
2. 『SDSoC 環境ユーザー ガイド』(UG1027)
3. 『SDSoC 環境最適化ガイド』(UG1235)
4. 『SDSoC 環境チュートリアル: 入門』(UG1028)
5. 『SDSoC 環境プラットフォーム開発ガイド』(UG1146)
6. SDSoC 開発環境ウェブ ページ
7. 『UltraFast エンベデッド デザイン設計手法ガイド』(UG1046: 英語版、日本語版)
8. 『ZC702 評価ボード (Zynq-7000 XC7Z020 All Programmable SoC 用) ユーザー ガイド』(UG850)
9. 『Vivado Design Suite ユーザー ガイド: 高位合成』(UG902)
10. 『PetaLinux ツール資料: ワークフロー チュートリアル』(UG1156)
11. Vivado® Design Suite の資料
12. 『Vivado Design Suite ユーザー ガイド: カスタム IP の作成とパッケージ』(UG1118)

お読みください: 重要な法的通知

本通知に基づいて貴殿または貴社 (本通知の被通知者が個人の場合には「貴殿」、法人その他の団体の場合には「貴社」。以下同じ) に開示される情報 (以下「本情報」といいます) は、ザイリンクスの製品を選択および使用することのためにのみ提供されます。適用される法律が許容する最大限の範囲で、(1) 本情報は「現状有姿」、およびすべて受領者の責任で (with all faults) という状態で提供され、ザイリンクスは、本通知をもって、明示、黙示、法定を問わず (商品性、非侵害、特定目的適合性の保証を含みますがこれらに限られません)、すべての保証および条件を負わない (否認する) ものとします。また、(2) ザイリンクスは、本情報 (貴殿または貴社による本情報の使用を含む) に関係し、起因し、関連する、いかなる種類・性質の損失または損害についても、責任を負わない (契約上、不法行為上 (過失の場合を含む)、その他のいかなる責任の法理によるかを問わない) ものとし、当該損失または損害には、直接、間接、特別、付随的、結果的な損失または損害 (第三者が起こした行為の結果被った、データ、利益、業務上の信用の損失、その他あらゆる種類の損失や損害を含みます) が含まれるものとし、それは、たとえ当該損害や損失が合理的に予見可能であったり、ザイリンクスがそれらの可能性について助言を受けていた場合であったとしても同様です。ザイリンクスは、本情報に含まれるいかなる誤りも訂正する義務を負わず、本情報または製品仕様のアップデートを貴殿または貴社に知らせる義務も負いません。事前の書面による同意のない限り、貴殿または貴社は本情報を再生産、変更、頒布、または公に展示してはなりません。一定の製品は、ザイリンクスの限定的保証の諸条件に従うこととなるので、https://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。IP コアは、ザイリンクスが貴殿または貴社に付与したライセンスに含まれる保証と補助的条件に従うことになります。ザイリンクスの製品は、フェイルセーフとして、または、フェイルセーフの動作を要求するアプリケーションに使用するために、設計されたり意図されたりしていません。そのような重大なアプリケーションにザイリンクスの製品を使用する場合のリスクと責任は、貴殿または貴社が単独で負うものです。https://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。

自動車用のアプリケーションの免責条項

オートモーティブ製品 (製品番号に「XA」が含まれる) は、ISO 26262 自動車用機能安全規格に従った安全コンセプトまたは余剰性の機能 (「セーフティ設計」) がない限り、エアバッグの展開における使用または車両の制御に影響するアプリケーション (「セーフティ アプリケーション」) における使用は保証されていません。顧客は、製品を組み込むすべてのシステムについて、その使用前または提供前に安全を目的として十分なテストを行うものとします。セーフティ設計なしにセーフティ アプリケーションで製品を使用するリスクはすべて顧客が負い、製品の責任の制限を規定する適用法令および規則にのみ従うものとします。

© Copyright 2017 Xilinx, Inc. Xilinx、Xilinx のロゴ、Artix、ISE、Kintex、Spartan、Virtex、Vivado、Zynq、およびこの文書に含まれるその他の指定されたブランドは、米国およびその他各国のザイリンクス社の商標です。OpenCL および OpenCL のロゴは Apple Inc. の商標であり、Khronos による許可を受けて使用されています。PCI、PCIe、および PCI Express は PCI-SIG の商標であり、ライセンスに基づいて使用されています。すべてのその他の商標は、それぞれの保有者に帰属します。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。フィードバックは日本語で入力可能です。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。