New Features

This document list the new features in XIL 1.2.

New Imaging Operations

XIL 1.2 contains new imaging operations that let you: store the absolute values of a source image's pixels; detect a source image's edges; and find the larger or lesser of pixel values in two source images. The new imaging operations are:

- `xil_absolute()`
- `xil_edge_detection()`
- `xil_max()`
- `xil_min()`

Using a Plane Mask for Copying Images

The `xil_copy()` function copies a source image to a destination image, copying all source-image *planes* (bits) to the destination. XIL 1.2 contains a new `xil_copy_with_planemask()` function, which lets you determine which source-image planes are copied to the destination. Copying with plane-mask control is typically used for overlaying images. It is also useful for double buffering on hardware that doesn't have separate memory buffers; double buffering is useful for animation because you can render an image in a hidden memory buffer while you display another image in a second buffer.

Multiband Lookup Tables

XIL 1.2 supports multiband images as input to `xil_lookup()`; earlier releases supported only single-band input images. With multiband input images, the `XilLookup` object contains a separate data array for each band in the input image; thus, it can be used to convert a multiband image of any data type to a multiband image of any data type.

The new multiband lookup functions are:

- `xil_lookup_create_combined()`
- `xil_lookup_get_input_nbands()`
- `xil_lookup_get_band_lookup()`

Interpolation Tables

When geometric transformations result in non-integer pixel locations, the XIL library provides nearest-neighbor, bilinear, and bicubic options for interpolating pixel values in a destination image; for these options, the interpolation filter is provided by the library. XIL 1.2 introduces a *general* interpolation option, for which your application defines the filter.

To support general interpolation, the XIL library provides a new `XilInterpolationTable` object, which is an array of 1xn kernels that represents the interpolation filter in either the horizontal or vertical direction. Interpolation tables let an application determine how many pixels are used to calculate interpolated pixel values, weight the pixels used in the calculations, and divide the space between adjacent pixels into multiple sub-locations so a destination pixel's value can be interpolated differently, depending on which sub-location it falls in. The interpolation tables are set on an `XilSystemState` object, and they affect all general interpolation operations using images created from this `XilSystemState`.

The following new functions support general interpolations:

- `xil_interpolation_table_create()`
- `xil_interpolation_table_get_data()`
- `xil_interpolation_table_get_kernel_size()`
- `xil_interpolation_table_get_subsamples()`
- `xil_interpolation_table_destroy()`
- `xil_state_get_interpolation_tables()`
- `xil_state_set_interpolation_tables()`

Warping an Image

The XIL library now contains three general functions that let you customize a geometric transformation by creating *warp* tables that displace pixels horizontally, vertically, or in both directions. The warp tables are XIL images whose pixel values define the backward mapping from a pixel in the destination to a pixel in the source.

Warp tables are typically used to stretch an image according to predefined rules and are most useful for performing nonlinear transformations. For example, they might be used to correct distortions that were imposed on an image by the equipment used to capture it. Or they might be used for cartographic projection of an image.

The new warping functions are:

- `xil_tablewarp()`
- `xil_tablewarp_horizontal()`
- `xil_tablewarp_vertical()`

XilDevice Objects

Typically when working with device images, an application creates the device image, then sets its attributes. Some device types, however, require that the device attributes be defined when the device image is created. For these device types, XIL 1.2 has a new `XilDevice` object that you can create and associate with the device type, then use to store device attributes. After storing all required attributes on the device object, you pass the object as an argument on the `xil_create_from_device()` function that creates the device image, thereby creating the device image and simultaneously setting its required attributes.

The device objects have the following new man pages:

- `xil_device_create()`
- `xil_device_destroy()`
- `xil_device_set_value()`

In addition, `xil_create_from_device()` has been modified to accept the device object as an argument on its last parameter.

Note – Device objects can be used only to initialize attributes before a device image is created. They cannot be used to modify the attributes of an existing device image. Devices that don't require initialized attributes may not recognize the device object; for these device types, you still pass `NULL` as the final argument on `xil_create_from_device()`.

Reading Kodak Photo CD Images

The XIL library includes a device handler that reads and decodes images stored in Eastman Kodak Company's Photo CD™ format. This feature was previously available as a patch to XIL 1.1.

Conditionally Compiling Code for Different XIL Versions

Beginning with release 1.2, the XIL library lets you conditionally compile code so you can take advantage of new interfaces in a current release while still supporting earlier releases of the library. To permit compile-time decisions, the library uses preprocessor directives to define symbols that identify the current XIL major and minor version numbers; these symbols—`XIL_API_MAJOR_VERSION` and `XIL_API_MINOR_VERSION`—are available to your application when you include the `xil.h` header file.

Shared Memory

XIL 1.2 supports the X Shared Memory Extension for displaying images with a new I/O device. Given an X window by `xil_create_from_window()`, the XIL library searches for an XIL I/O device for the window to associate with the image.

Previously, the XIL library used DGA to locate a device-specific I/O device, and if no device-specific I/O device could be loaded, it used an Xlib I/O device. For XIL 1.2, the XIL library tries to use a device-specific I/O device, and if no device-specific I/O device can be loaded, it attempts to load an XShm I/O device. If both the device-specific and the XShm I/O device fail, the Xlib I/O device is loaded. These changes were implemented to assure you of the following:

- The XIL library always chooses the fastest available way to display an image
- The display mechanism is network independent

Performance Enhancements

In addition to supporting the X Shared Memory Extension for speeding up image displays, many XIL functions have been optimized to improve the overall performance of XIL applications.

Installing XIL Packages

As discussed in the *Software Developer Kit Installation Guide*, the best way to install XIL packages is to use the `pkgadd/pkrm` commands. However, if you prefer, you can still use `swmtool`. You can no longer use the `swm` command to install XIL packages.