NAME

xil_roi_create, xil_roi_create_copy, xil_roi_destroy – create or destroy ROIs

SYNOPSIS

#include <xil/xil.h>

XilRoi xil_roi_create (XilSystemState *State***);**

XilRoi xil_roi_create_copy (XilRoi *roi***);**

void xil_roi_destroy (XilRoi *roi***);**

DESCRIPTION

These routines create and destroy region of interest (ROI) objects.

xil_roi_create () creates an *XilRoi* object. It is initially empty. You can use **xil_roi_add_rect**(3), **xil_roi_add_region**(3), or **xil_roi_add_image**(3) to add rectangles to an ROI. ROIs exist in the coordinate system of the image storage. That is, an ROI coordinate of (0.0, 0.0) always refers to the upper left pixel in an image, regardless of the image’s origin.

xil_roi_create_copy () returns a copy of the specified ROI. The name of a copy is initially empty (NULL).

xil_roi_destroy () destroys the specified ROI.

XIL and Regions of Interest

ROIs provide a way to limit operations to a specific part of image data. ROIs are attributes of images; they specify what part of an image may be used.

As a destination attribute, the ROI functions as a "write mask" for the destination image. If the ROI is valid (non-zero) for a particular pixel, that pixel may be modified by the operation; otherwise, the pixel is not written.

For a source image, the ROI defines what part of the source image may go toward modifying the destination. In the case of some of the geometric operators, this means a rectangular ROI in the source maps to a nonrectangular area of modification in the destination.

Area operations, such as interpolated geometric zooms, convolution, and erosion, may use data outside the source ROI in creating their output pixel. In the case of geometric operators, the source pixel is generated if the backward-mapped subpixel position lies in the ROI; pixels used in the interpolation may be outside the ROI. For convolution and erosion/dilation, the source pixel is used if it is inside the source ROI; the surrounding pixels used in generating the convolution may be outside the ROI. In the destination, only the output pixel is tested against the destination ROI for writability.

If more than one image in an operation has an ROI attribute, the intersection of all the ROIs (with source ROIs transformed into the destination space) is used to mask the destination.

Although they are image attributes, ROIs attached to an image are not modified along with the image. Destination images retain their own ROIs and do not adopt the ROI of the source image. Copying an image does not copy the ROI attribute; it must be copied explicitly. In addition, creating a child image from an image with an ROI does not cause the child to inherit the portion of the parent's ROI covering it. Installation of an ROI on a child image must be performed explicitly.

The coordinate space of the ROI is conceptually tied to the image storage. That is, the location of the ROI with respect to image data is not changed by changing the image origin.

Operations on ROIs may be performed by retrieving the ROI as a 1-bit image, passing the image to the appropriate XIL operator, then reinstalling the image as an ROI. Several functions exist to operate directly on ROIs without having to first convert them into an external format. This probably provides better performance for these supported operators. A list of ROI operations and their corresponding man pages is given below.

Get an ROI	**xil_get_roi**(3)
Set an ROI	**xil_set_roi**(3)
Add a binary image to an ROI	**xil_roi_add_image**(3)
Add a rectangle to an ROI	**xil_roi_add_rect**(3)
Add an X region to an ROI	**xil_roi_add_region**(3)
Create and return a copy of an ROI	**xil_roi_create_copy**(3)
Destroy an ROI	**xil_roi_destroy**(3)
Get an image version of an ROI	**xil_roi_get_as_image**(3)
Get an X region version of an ROI	**xil_roi_get_as_region**(3)
Find the intersection of two ROIs	**xil_roi_intersect**(3)
Subtract a rectangle from an ROI	**xil_roi_subtract_rect**(3)
Translate an ROI	**xil_roi_translate**(3)
Find the union of two ROIs	**xil_roi_unite**(3)

ERRORS

For a complete list of XIL error messages by number, consult Appendix B of the *XIL Programmer's Guide.*

EXAMPLES

Create an ROI, add a rectangle to it (beginning at (10,20) and ending at (109,69), associate it with an image, and then destroy it:

```
XilSystemState State;
XilImage image;
XilRoi roi;
long  xstart=10, ystart=20, width=100, height=50;

roi = xil_roi_create (State);
xil_roi_add_rect (roi, xstart, ystart, width, height);
xil_set_roi (image, roi);
xil_roi_destroy (roi);
```

SEE ALSO

xil_roi_add_image(3), **xil_roi_add_rect**(3), **xil_roi_add_region**(3), **xil_roi_get_as_image**(3), **xil_roi_get_as_region**(3), **xil_roi_intersect**(3), **xil_roi_subtract_rect**(3), **xil_roi_translate**(3), **xil_roi_unite**(3), **xil_get_roi**(3), **xil_set_roi**(3).