NAME

fish - purify wire patterns on Sea-of-Gates images

SYNOPSIS

fish [-options] [<cell_name>]

OPTIONS

<cell_name>
Cell name of layout to be purified.

[no argument]
Make empty image array of given size (implies ’-o’).

-c Purify circuit cell instead of layout cell.

-H Hierarchical fish: fish all son-cells too.

options to control how to write the cell back into nelsis:

-i Produce output cell without the image.

-x <x_size>
Force horizontal array size to x_size (>= 1).

-y <y_size>
Force vertical array size to y_size (>= 1).

-v Write vias as model calls instead of boxes.

-a Perform automatic move to origin of cell.

-E <list>
Exclude the objects in <list> from being fished. Example: -E tmv13 leaves terminals, MC’s vias and layers 1 and 3.

misc. options:

-V Verbose parsing of image file: print all unknown keywords.

-o <out_name>
Write purified result to specified cell name (default: overwrite input cell).

-q Quiet mode: nothing is printed (except errors and warnings).

-h Help: print list of options.

DESCRIPTION

Fish is a layout purifier for gate-array and sea_of_gates layout. It reads a layout from the database, processes the layout, and writes a new enhanced cell back into the database. The basic idea is to allow some design-rule errors during the manual design of a gate-array cell. *Fish* corrects these errors and produces a design rule correct result. In this way the process of manual design using *dali*(1ICD) can be speeded up considerably.

Fish performs the following:

- Wire segments are rounded to the nearest grid point (legal point) of the array and obtain the correct width.
- Vias are aligned on grid points.
- Vias which were entered using a contact hole mask (e.g. the ’cps’ a and ’cos’ in c3tu) are converted into a model-call to the proper via pattern. The designer does not have to worry about the type of via used.
- The size of the image (the number of repetitions) is adjusted.
- Some warnings for ambiguity, illegal wires, etc. are given.
- Instances are snapped to grid.

- In this way instances (components) can be stamped onto the image, without having to take care of exact alignment.

To perform all of this, *fish* needs an image description file. This file is called "image.seadif" and should be present in the project directory. It contains data about the positions of grid points, the way in which the image should be repetited, the kind of via which should be used, etc. Cells containing the basic image and the vias should be available in the database.

EXAMPLE

Suppose you want to make a cell called 'nand2' in an image in the c3tu process.

First make sure that the image cell and the vias are present in the database. Import them if necessary. The name of these cells is given in the image description file "image.seadif".

Before editing it is convenient to have an empty array to indicate the transistor and grid positions. The following command will cause *fish* to create an empty array of size 10 x 1 in the cell 'nand2'.

```
% fish -x 10 -y 1 -o nand2
```

Now start editing your cell using *dali*(1ICD):

```
% dali
```

Read cell 'nand2' and expand it such that the image becomes visible. Use the 'visible' menu to turn off some useless masks such as 'nw', 'sn' and 'sp'. Using the 'append' or 'wire' command in the 'box' menu metal wires can be be added. Do not add layout in ps or od, since these layers are fixed in a gate-array image. *Fish* will delete rectangles in illegal layers.

Vias can be added in two ways:

1) By placing specific instances that represent vias. In the library via cells are present which are recognized by *fish*, such as 'Via_in_ins'.

2) A easier way is by making a rectangle in the contact hole layer at the desired position. The size of the via does not matter, since *fish* will adjust it automatically. For a via between metal1 (in) and metal2 (ins) used the 'cos' layer. In some images 'cos' vias may not be placed at some positions. In the fishbone image, for instance, 'cos' vias are not allowed at all poly connections. For ad via to the image (to 'cps' or 'od') use any of the following layers: 'cps', 'con', 'cop'. *Fish* will automatically replace any of these masks by the proper pattern. *Fish* will only distinguish between 'cop' and ('cps', 'con', 'cop'). Remember that 'stacked contacts' are not allowed.

Terminals may only be placed in a metal layer: in or ins.

If the underlying image should become too small, simply enlarge the basic image: press 'arr_par' in the 'instances' menu. Set nx or ny to the new value. Do not change the dx or dy!

When ready write the cell back to the database and quit *dali*(1ICD). Now the big trick can be performed:

```
% fish nand2
```

This command will overwrite the cell 'nand2' by the new purified cell. It is also possible to give the new cell a different name (and, with that, save the original cell):

```
% fish -o nand2_new nand2
```

This command will write a new cell in 'nand2_new'.

To produce only the metal interconnection pattern of the nand2 use:

```
% fish -i nand2
```

This nand2 can now be used as a 'stamp' for larger cells. It can be read as instance in *dali*(1ICD). You only have to make a rough placement of this instance. *Fish* will snap the instance to the nearest grid point. To perform this snapping, *fish* opens the cell and looks for vias. The first via it encounters is used for the snapping. Notice that *fish* assumes that the instances are on grid (that is, fished).

AUTHORS

Paul Stravers, Patrick Groeneveld

FILES

proj_dir/image.seadif	(technology file)