Pin Placer Configuration Files

If you're using the step `Odb.CustomIOPlacement` the variable `FP_PIN_ORDER_CFG` asks you to create a `.cfg` file that can be used to configure this placement step.

Parsing and Grammar

The grammar is expressed in antlr4 and is available at the repository efabless/ioplace_parser.

Instructions

The document is a **whitespace-delimited** set of instructions.

Sections

There are four sections, `N`, `S`, `E`, and `W`, corresponding to the four cardinal directions.

The sections are marked using a `#` instruction. For example, `#N` marks the beginning of the section for the pins on the north of the chip.

Pins

You can capture one or more pins by writing a regular expression.

For example, a line with simply `y` will match a pin named y, meanwhile, a line with `x\[\d+\]` will match `x[0]`, `x[1]`, `x[2]`, ..., `x[10]`, et cetera.

> **Warning**
>
> A line with just `x[0]` is still a regular expression which will match the string `x0`. Be sure to escape all special regex characters: `x\[0\]` would match `x[0]`.

Annotations

Some statements can be preceded with `@`s and are referred to as annotations.

Annotations placed before declaring any sections are known as **global annotations** and affect default behaviors of all sections.

Annotations placed inside a section only affect the specific section.

Minimum Distance

You can set the minimum distance between two pins using the `@min_distance=<distance>` annotation, where the distance is in microns.

If set globally, the minimum distance applies to all sections, unless another `@min_distance` is declared inside a section.

Be advised, if this distance is less than the legal minimum distance for a section, the legal minimum will be used.

Regex Sorting

The results of regular expressions can be sorted according to two ways.

Like minimum distance, setting these annotations globally applies to all sections unless another annotation is specified inside a section.

`@bus_major` (default)

If two buses are matched with the same regular expression, the buses are sorted in a bus-major fashion: i.e. alphabetically by any found bus name(s) and then ascendingly by any found bit number(s).

For example, for a regular expression `l.*` matching pins:

- `lemon[1]`
- `lime[0]`
- `lemon[0]`
- `lime[1]`

The order returned would be:

- `lemon[0]`
- `lemon[1]`
- `lime[0]`
- `lime[1]`

`@bit_major`

If two buses are matched with the same regular expression, the buses are sorted in a bit-major fashion: i.e., ascendingly by any found bit number(s) then alphabetically by any found bus name(s).

For example, for a regular expression `1.*` matching pins:

- `lemon[1]`
- `lime[0]`
- `lemon[0]`
- `lime[1]`

The order returned would be:

- `lemon[0]`
- `lime[0]`
- `lemon[1]`
- `lime[1]`.

Virtual Pins

You can add one or more “virtual pins” (i.e. pins that occupy a slot but do not actually exist) using the `$` instruction, where `$1` adds one virtual pin, `$2` adds two, and so on.

Example

Here is the `.cfg` file for the SPM design:

```
#N
@min_distance=0.1
x.*

#S
$1
rst

#E
clk

#W
p
y
```