Custom IC Creator

Carsten Wulff

Jun 22, 2025

CONTENTS:

cIcCreator is a compiler for integrated circuits.

cIcCreator reads a JSON object definition file, technology rule file and a SPICE netlist (assumes same name as object definition file) and outputs a cic description file (.cic).

Use cicpy (https://github.com/wulffern/cicpy) to transpile the .cic file to other formats.

Download PDF.

CHAPTER

ONE

GETTING STARTED

First clone the repo

```
git clone https://github.com/wulffern/ciccreator
cd ciccreator
git clone https://github.com/wulffern/ciccreator-bin release
```

In the release folder, you hopefully find the binary for your platform.

There are three input files needed to run the compiler

- Technology file
- Object definition file
- Netlist

The description below is for MacOS. For other OS replace the .darwin with the binary that applies to your platform.

```
release/cic.darwin-latest examples/SAR_ESSCIRC16_28N.json examples/tech.json SAR_TEST
```

The output of the compiler will be SAR_TEST.cic

To view the output, do

```
release/cic-gui.darwin-latest SAR_TEST.cic examples/tech.json
```

The examples/routes.json show some examples of the routings, and other options

```
release/cic.darwin-latest examples/routes.json examples/tech.json routes
release/cic-gui.darwin-latest routes.cic examples/tech.json
```

1.1 Basics

1.1.1 Technology File

The technology file has sections: layers, technology, rules.

An example of a technology file for Skywater 130 nm

layers

cic expect to find some common layers, like "OD" (diffusion, active, the stuff that is under the poly gates), "PO" (the gate material), "M1" "M2" (meta layers).

cic does not care that much about the number, or datatype anymore, but they still need to be there. The important parts are the material, previous, next, and pin.

The color is used by cic-gui.

```
"PO"  : { "number" : 7,          //SKILL layer number
          "datatype" : 0,        //SKILL layer datatype
          "alias" : "poly",      //Name in Skywater 130 nm Magic VLSI
          "material" : "poly",   //Type of layer [poly,cut,metal,marker,implant]
          "previous" : "",       //Previous layer in the connect stack. For M1 it would␣
↪be CO
          "next" : "CO",         //Layer to used to connect to the next routing layer
          "pin" : "PO_pin" ,     //Layer to be used for pins in this layer (i.e port text)
          "color" : "red"        //Color of layer in cic-gui
        }
```

technology

Most of the technology file is irrelevant for cic, but the rule file still expects to find gamma, grid, spiceunit, devices (name,devicetype,ports).

The "gamma" keyword is the unit used to translate from "rules" into real numbers (Ångstrøm). The default gamma is 50 Ångstrøm.

The "grid" is the minimum snap distance, for example all units will be rounded to "5 nm"

The "spiceunit" is no longer applicable, but has not been removed from "core/rules.cpp"

```
"gamma" : 500,
"grid" : 5
"spiceunit": 1,
```

rules

All rules are defined as a multiplum of the gamma.

```
"M2"   : {                      // Layer to apply rules for
    "space" : 1,                // rule name : multiplum of gamma
    "width" : 1
 },
```

Common rule names are

- space : distance between two rectangles
- width : default width
- height : default height, common for cut layers
- enclosure : default enclosure of for example OD around this layer
- XXXenclosure : enclosure of current layer around layer XXX

- XXXencOpposite : enclosure of current layer around layer XXX, used when there is more than two cuts on a via

A special rule is the 'ROUTE' layer, which does not correspond to a layer. It's used to set the grid for the PatternTile objects, and only has 'verticalgrid' and 'horizontal grid' rule names.

```
"ROUTE": { "horizontalgrid": 18, "verticalgrid" : 22 }
```

1.1.2 Object Definition File

The object definition file contains the almost everything except for the placement information, and the connectivity, both of which are in the Netlist.

An example object definition file can be seen at ip.json

```
{
  "options" :
  {
      "ignoreSetYoffsetHalf" : false,
      "prefix" : "SUNTR_"
  },
  "include" : [
      "dmos_sky130nm_core.json",
      "dcap.json",
      "tr.json",
      "resistors.json",
      "dig.json",
      "components.json",
      "capacitor.json"

  ],
  "cells" : [ ]
}
```

Comments

Any line starting with *s*// (space followed by //), will be ignored by the compiler.

Options

Options are instructions to cic. For example, "ignoreSetYoffsetHalf" is a global parameter to override overlap of source/drain (custom feature).

Prefix is the name to prepend to all cells, which is useful for technologies where one need to have multiple, slightly different, versision of the design.

Include

The include sections link to other object definition files, and is mostly to avoid having everything in one big file. The array is read sequentially, as such, the order of the files are important. For example, if *tr.json* has an instance of a MOSFET from *dmos_sky130nm_core.json* the MOSFET must be created first.

Cells

`cells` is an array of all the cells in the design. The order of the cells are important, since a cell can't be referenced until it's been created.

An example of cell definition can be seen in dig.json

Each of the cells are read by cic, one by one, in sequence. cic will create an instance of the class, and run methods on that class.

```
"cells":
  [
      { "name": "TAPCELLB_CV",
        "class" : "Layout::LayoutDigitalCell",
        "meta" : {
            "symbol" : "cic_wbulk/tap"
        },

        "boundaryIgnoreRouting" : 1,
        "beforeRoute" : {
            "addDirectedRoutes" : [ ["M1","AVSS","MN1:B->MN1:G"],
                                    ["M1","AVSS","MN1:G-|--MN1:S"],
                                    ["M1","AVSS","MN1:G-|--MN1:D"],
                                    ["M1","AVDD","MP:S-|--MP:G"],
                                    ["M1","AVDD","MP:D-|--MP:G"],
                                    ["M1","AVDD","MP:G->MP:B"]
                                  ]

        }

      }
  ]
```

Name

All cells must have a name. The name is used to search the spice file (assumed to be called *dig.spi* for a *dig.json* file). The name will also be used for the cell in the output file

Class

The class correspond to a class inside cic

1.1.3 Netlist

The netlist is written in standard SPICE, and the placement is based on location of items in spice.

1.2 Cells

There are more fundamental building blocks in cIcCreator than a [[Cell]], however, unless you plan to hack the source, you probably won't use Rect, Port, Text, etc.

1.2.1 Cell

Cell is the fundamental object of all objects in the layout (Ok, some object, like Port, inherit only Rect, but most inherit Cell). Cell can be found in `cic-core/src/core/cell.cpp`

Properties

In the object definition file the Cell can contain certain properties

- name [mandatory] : Name of the cell, must be uniq
- class [optional] : will use `LayoutCell` if it's not specified. In `cic-core/src/core/design.cpp` constructor there is a list of allowedcells
- inherit [optional] : Cell from which this cell will inherit all instructions
- leech [optional] : Cell from which this cell will copy all instructions, but will not follow the inheritance hierarchy. In other words it only cares about the first parent, no grandparents.
- comment [optional] : can occur anywhere, and is ignored by the compiler

Action

Each cell will trigger actions in a specific sequence. The sequence is as follows:

- Create object if the class exists
- Set name
- Run `afterNew` of all parents, starting with the oldest
- Run `afterNew` of the current cell
- Set all properties, and run all custom functions on the object

- Run `beforePlace` of all parents, starting with the oldest
- Run `beforePlace` of the current cell
- Call place() on the object
- Run `afterPlace` of all parents, starting with the oldest
- Run `afterPlace` of the current cell
- Run `beforeRoute` of all parents, starting with the oldest
- Run `beforeRoute` of the current cell
- Call route()
- Run `afterRoute` of all parents, starting with the oldest
- Run `afterRoute` of the current cell
- Call addAllPorts() on the object, to place remaining Ports.
- Run `beforePaint` of all parents, starting with the oldest
- Run `beforePaint` of the current cell
- Call paint() on the object
- Run `afterPaint` of all parents, starting with the oldest
- Run `afterPaint` of the current cell
- Add the object as a child of Design
- Add to the static list of cells

1.2.2 PatternTile

PatternTile is the base functions for all ASCII to Layout objects. PatternTile can be found in `cic-core/src/core/patterntile.cpp`

Properties

- yoffset : number [optional] : Vertical grid offset of the origin
- xoffset : number [optional] : Horizontal grid offset of the origin
- widthoffset : number [optional] : Reduce the width by X grid

Functions

fillCoordinatesFromString

The argument is an array of arrays that contain the ASCII

```
[
  [ "Layer Name",
   "rectangle definitions",
  ]
]
```

for example

```
[
    [ "M1",
    "--xxxQxxxxx---",
    "----------x---",
    "--xxxxxkx-x---",
    "--------x-D---",
    "--xxxxxxx-x---",
    "--------x-x---",
    "--xxxxxkx-x---",
    "----------x---",
    "--xxxQxxxxx---"
    ]
]
```

The possible rectangle definitions are

- '-' : Empty rectangle
- 'x' : Fill rectangle completely
- 'X' :
- 'm' : Fill rectangle horizontally, but use "mingatelength" rule for height
- 'w' : Fill rectangle horizontally, but use "width" rule for height
- 'D,G,S,B,A' : Add Port
- 'c' : Add cut in the center of the current grid
- 'C' : Add cut aligned on the left edge of the current grid
- 'K' : Add two cuts with the first cut aligend on the left edge of the current grid
- 'k' : Add two cuts with the first cut aligend in the center of the current grid
- 'Q' : Add two cuts centered in the center of the current grid
- 'r' : Add metal resistor

1.2.3 PatternCapacitor

PatternCapacitor extends PatternTile, and can create capacitors with two, and three terminals. PatternCapacitor can be found in `cic-core/src/core/patterncapacitor.cpp`

Properties

- yoffset : number [optional] : Vertical grid offset of the origin
- xoffset : number [optional] : Horizontal grid offset of the origin
- widthoffset : number [optional] : Reduce the width by X grid

Functions

1.2.4 PatternTransistor

1.2.5 PatternResistor

1.2.6 LayoutCell

LayoutCell extends Cell, it's the most basic cell. It's the default class unless `class` parameter is defined. LayoutCell can be found in `cici-core/src/core/layoutcell.cpp`

setYoffsetHalf

If called it will calculate the height of the cell, and adjust the height of the cell to half. It's used to overlap transistor drain/source

noPowerRoute : 1 | 0

At paint() the power will be routedd unless `noRoutePower : 1`

addDirectedRoute : Array

Adds a route from start rectangles to stop rectangles. The rectangles are determined based on the route command.

Arguments

```
[ "M1",  //Layer name
  "CKN", //Net name
  "XA1:MP0:D-|--XA2:MP0:G", //Route command
  "offsetlowend" //Route Options [optional]
]
```

Route command The routecommand is split with the following regex

```
^([^-\|<>]*)([-\|<>]+)([^-\|<>]*)$
```

A route is defined by the characters -|<>, everything before is put into a "start rectangle" path regex, and everything after is put into a "stop rectangle" path regex.

The path regex uses [[Cell::findAllRectangles(pathRegex,layer)|Cell]]

See [[Route]] for the route definitions and options

addConnectivityRoute: Array

Adds a route based on the connectivity of the subcircuit.

```
[ "M1",     //Layer name
  "^ENO$", //Regular expression for net match
  "-|--",  //Route
  "onTopL,offsetlow",  //Route options [optional]
  "",                  //Number of cuts to use [optional]
  "IVX|STATE"          //Instances to include [optional]
]
```

addPortOnRect: Array

Define which rectangles to add ports on

```
[
 "CMP_OP",   //Port name
 "M1",       //Layer name
 "XA4:A"     //Path regex
 ]
```

addVia: Array

Add a via at an horizontal offset to a rectangle defined by a path regex

```
[ "M3",          //Start layer
  "M4",          //Stop layer
  "MP1:D",       //Path regex to find rectangle
  2,             //Horizontal cuts
  1,             //Vertical cuts       [optional]
  8,             //Horizontal offset   [optional]
  "CUST_VREF"    //Custom name for via [optional]
]
```

addConnectivityVia: Array

Add a via on ports defined by a net name regular epression

```
["M4",      //Start layer
 "M5",      //Stop layer
 "C16",     //Path regex
 1000,      //Grid overrride, if 0 it's equal the grid is a cut width
 2,         //Vertical cuts
 1,         //Horizontal cuts    [optional]
 15,        //Horizontal offset  [optional]
 -0.5,      //Vertical offset    [optional]
 "CUST_C16" //Custom name, searchable by path regex [optional]
 ],
```

addPortVia: Array

Adds a via, and places a port on the via

```
[ "M2",          //Start layer
  "M4",          //Stop layer
  "RESN",        //Port name
  "X9$:MN1$:D", //Path regex
  1,             //Vertical cuts
  2,             //Horizontal cuts
  -3,            //Horizontal offset, multiplum of via width
  -1,            //Vertical offset, multiplum of via height
  "CUST_RESN"    //Custom name, searchable by path regex [optional]
]
```

addVerticalrect: Array

Adds a custom rectangle for the height of the module

```
["M5",           //Layer
 "CUST_C16",     //Path regex
 1               //Cuts, default 0, if 0 then use rectangle width
]
```

1.3 Routing

Routing in ciccreator is done using functions in LayoutCell, addDirectedRoutes and addConnectivityRoutes. One should start with addConnectivityRoutes, because that is easiest to use, however, it can only access the ports on instances in a subcircuit.

See https://github.com/wulffern/ciccreator/tree/master/examples/routes.spi and https://github.com/wulffern/ciccreator/tree/master/examples/routes.json for the example source code used in this wiki.

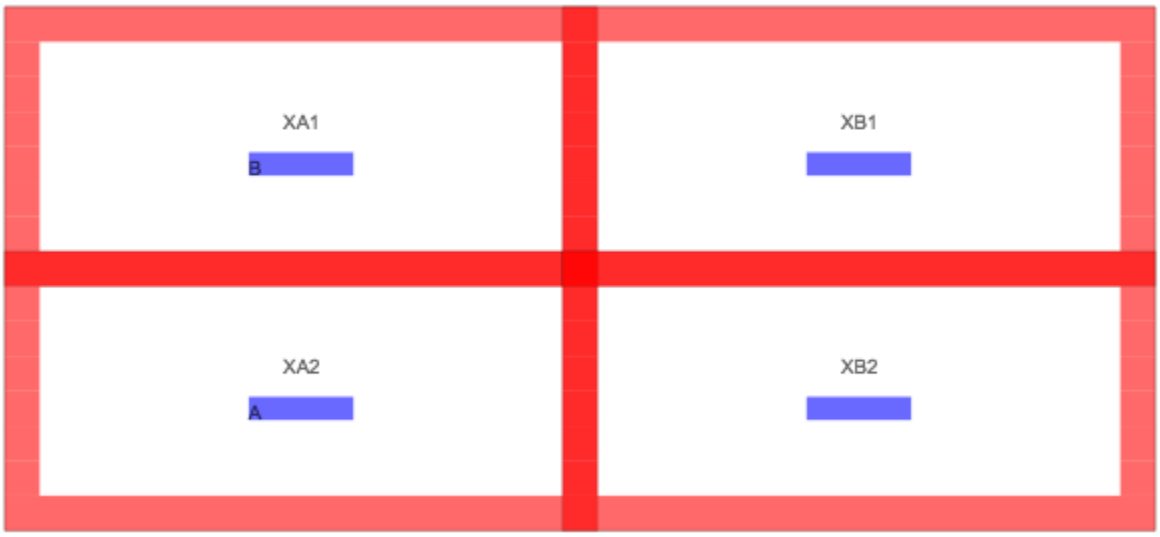
1.3.1 Connectivity Route

ConnectivityRoute uses regular expressions to match the net names. For example, take the spice circuit below

```
.subckt TEST A B
XA1 B DDD
XA2 A DDD
XB1 B DDD
XB2 A DDD
.ends
```

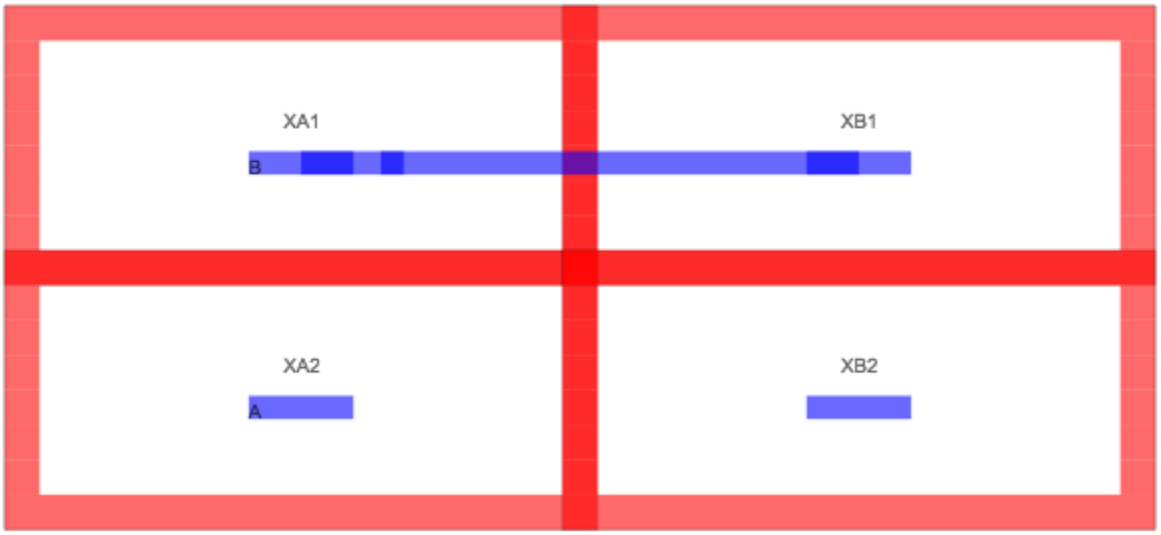
Without any routes, we'll get

```
{ "name" : "TEST", "class" : "cIcCore::LayoutCell"}
```



We can add a route by matching for example 'B' with

```
{ "name" : "Connectivity_B_-|--",
  "inherit": "TEST",
  "beforeRoute": [
    {"addConnectivityRoutes" : [["M1","B","-|--"]]}
  ]
}
```



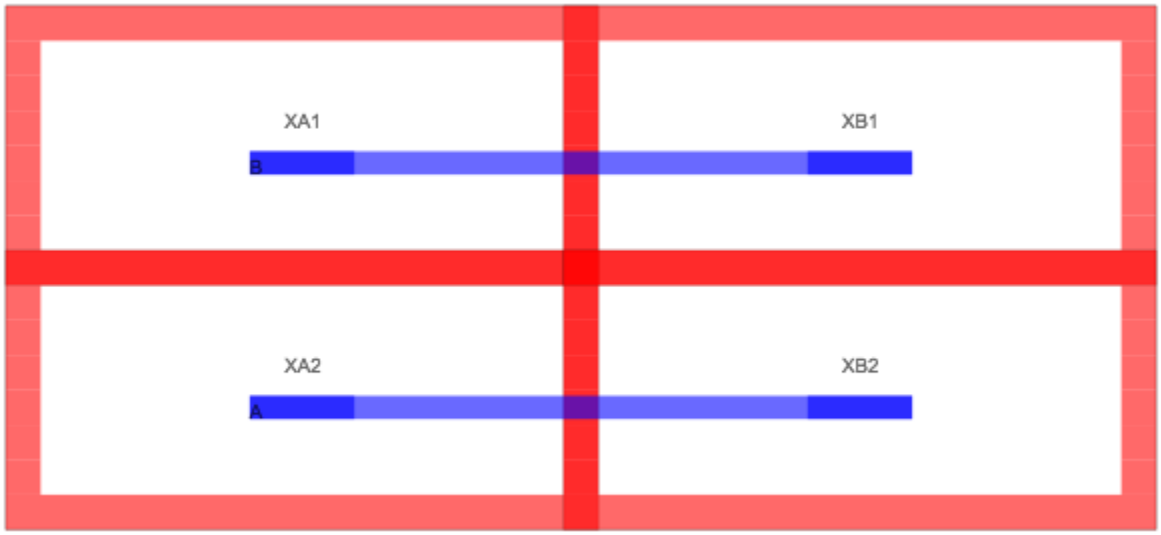
Or even matching both A and B

```
{ "name" : "Connectivity_A|B_-",
  "inherit": "TEST",
  "beforeRoute": [
      {"addConnectivityRoutes" : [["M1","A|B","-"]]}
  ]
```

(continues on next page)

(continued from previous page)

```
}
```



Even though we match both nets, the compiler knows it should not try and route A->B. But the Router is very dum, it does not know if it's created a short.

1.3.2 Directed Route

DirectedRoute is for the cases were you want to access a port deep in the hierarchy, but it's not available on top level via a port. The

DirectedRoute should be avoided if at all possible, because it's hard linked to the hierarchy, so if you change instance names, or the hierarchy, they will no longer work. But it's perfect for those tricky situations where ConnectivityRoute falls short.

The arguments for directed route are layer, net, and 'route command'.

Route Command

A route command looks like this

```
"XA1:S-|--XB2:S"
```

The route command contains three parts, start rectangles path regex, route type, and stop rectangles path regex. See addDirectedRoute in [[LayoutCell]] for details.

A 'path regex' looks like this 'XA1:XB2:D,XA2:XB2:D', the colon denotes a hierarchy border, while the stuff between are the regular expressions used to match instance names. The last 'D' is the port name of an instance. Multiple paths can be specified in one routecommand, separated by a comma.

The -|– is the [[Route Type]] of this particular route command.

With the spice

```
.subckt TEST A B
XA1 B DDD
```

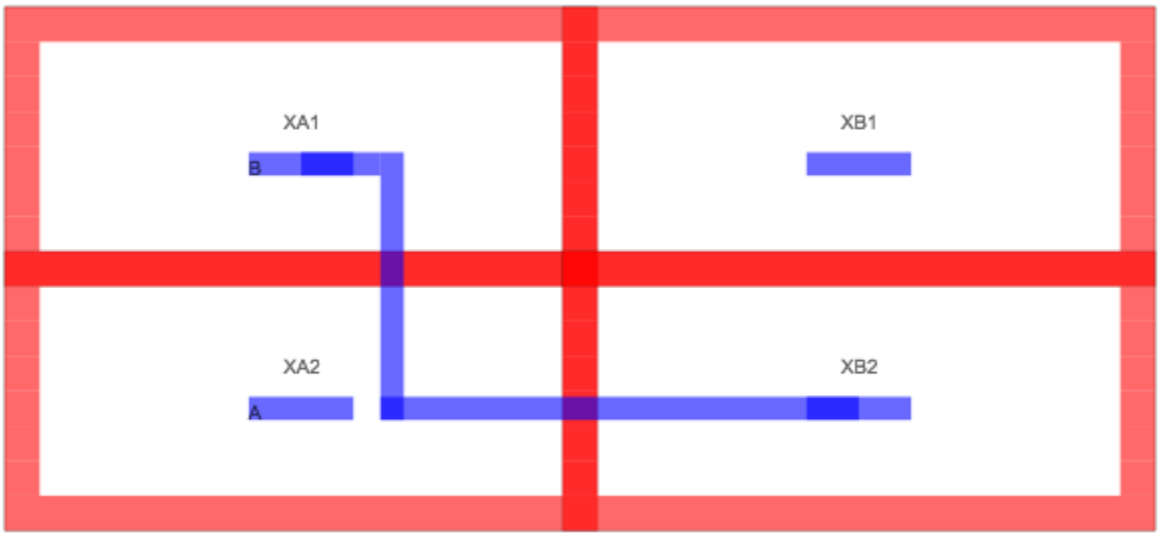
(continues on next page)

(continued from previous page)

```
XA2 A DDD
XB1 B DDD
XB2 A DDD
.ends
```

```
{ "name" : "Directed_StartLeft_Left(-|--)",
  "inherit": "TEST",
  "beforeRoute": [
    {"addDirectedRoutes" : [["M1","S","XA1:S-|--XB2:S"]]}
  ]
}
```

Will produce



1.3.3 Via/Cut

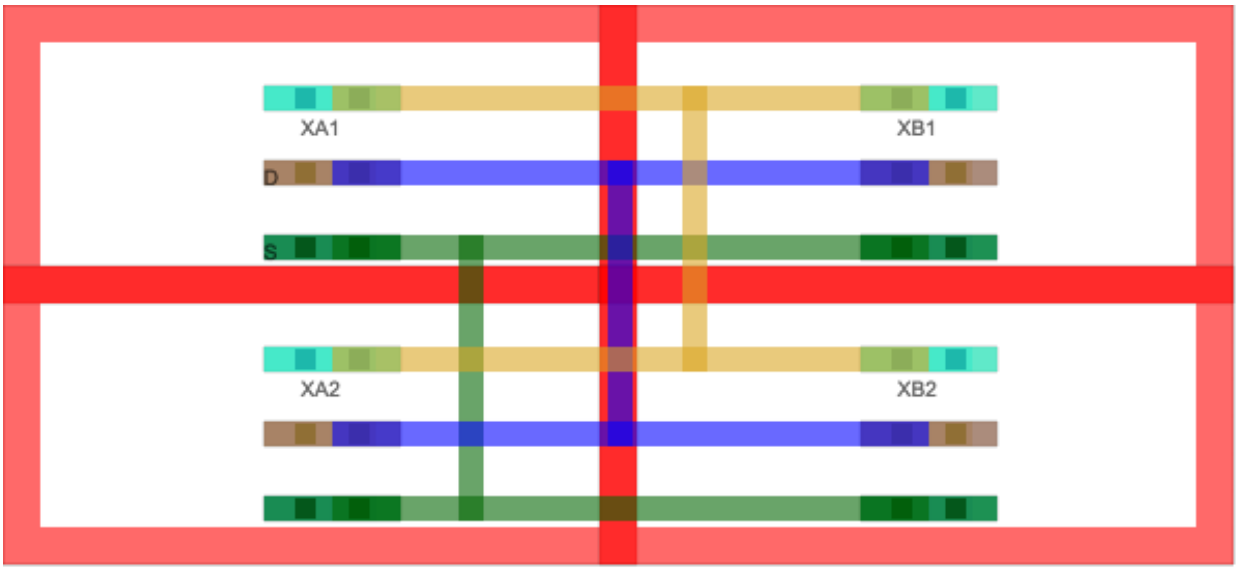
For most routes the vias will be automatically inserted. It will automatically find the layer for the start/stop rectangles, and add a via up to the routing layer specified.

Automatic via

Routes will find the necessary transisitons for routes.

```
.subckt TESTVIA S D B
XA1 S D B DDMVIA
XA2 S D B DDMVIA
XB1 S D B DDMVIA
XB2 S D B DDMVIA
.ends
```

```
{ "name" : "TESTVIA",
  "class" : "cIcCore::LayoutCell",
  "beforeRoute": [
  {"addConnectivityRoutes" : [
    ["M4","S","-|--","track0"],
    ["M1","D","-|--","track4"],
    ["M2","B","-|--","track6"]
  ]}
]
```



1.3.4 Route Type

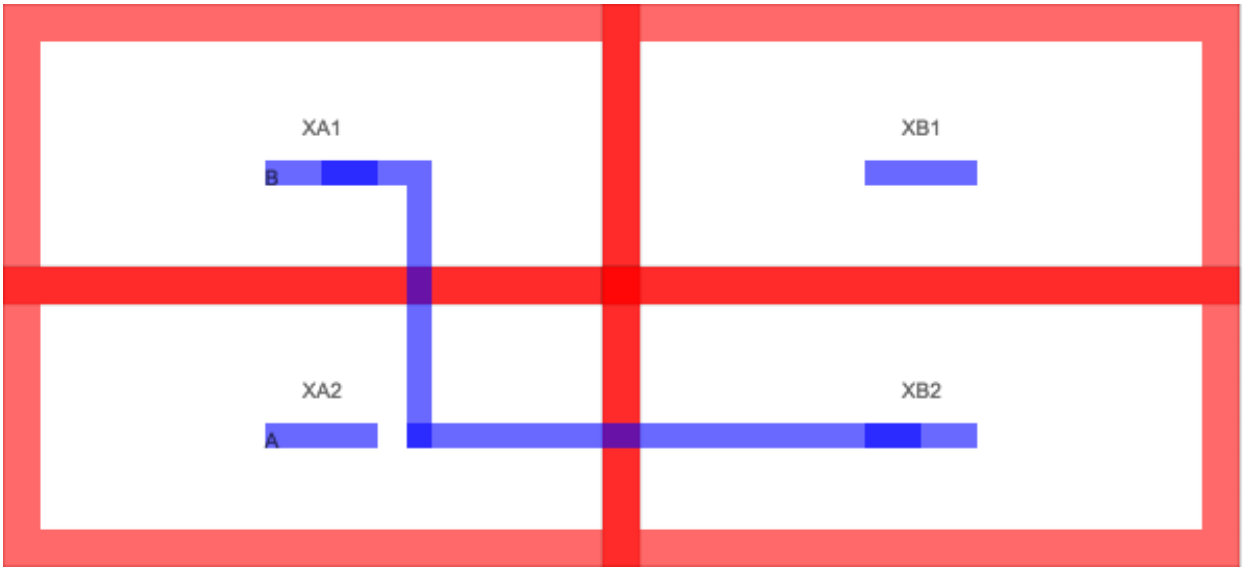
The following route types can be used, all of them have a short-hand notation.

Left (-|–)

The 'Left' route will go one metal space to the left, then up, and or down, and finish the route. The type of route will depend on whether the start rectangle is to the left, or the right of the finish.

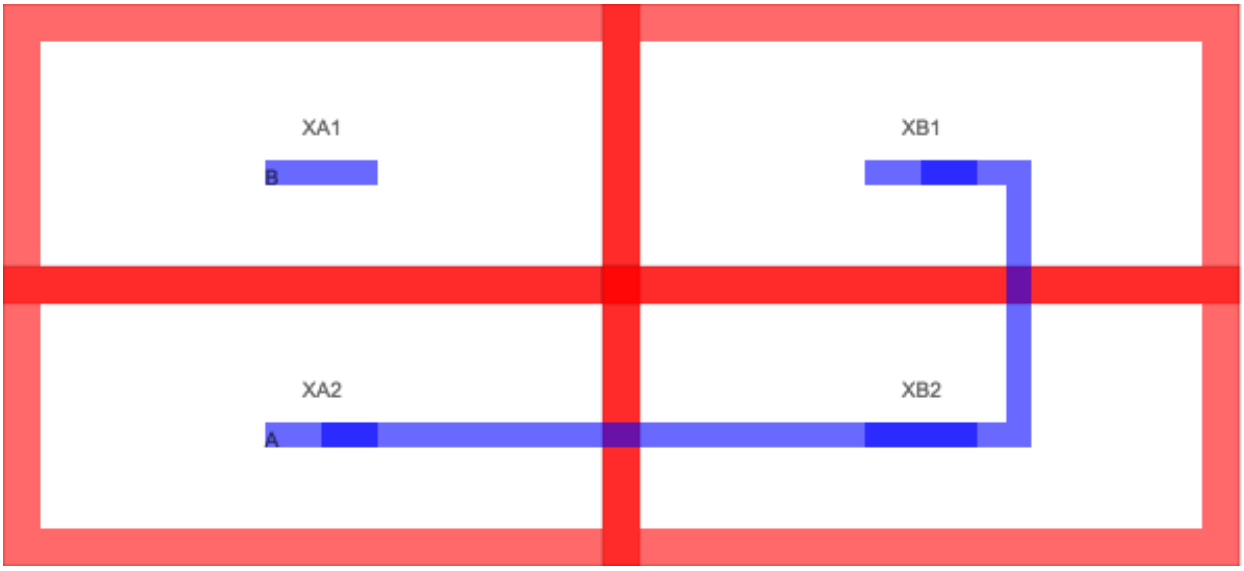
Starting from the left

```
{ "name" : "Directed_StartLeft_Left(-|--)",
  "inherit": "TEST",
  "beforeRoute": [
    {"addDirectedRoutes" : [["M1","S","XA1:S-|--XB2:S"]]}
  ]
}
```



Starting from the right

```
{ "name" : "Directed_StartRight_Left(-|--)",
  "inherit": "TEST",
  "beforeRoute": [
    {"addDirectedRoutes" : [["M1","S","XB1:S-|--XA2:S"]]}
  ]
}
```

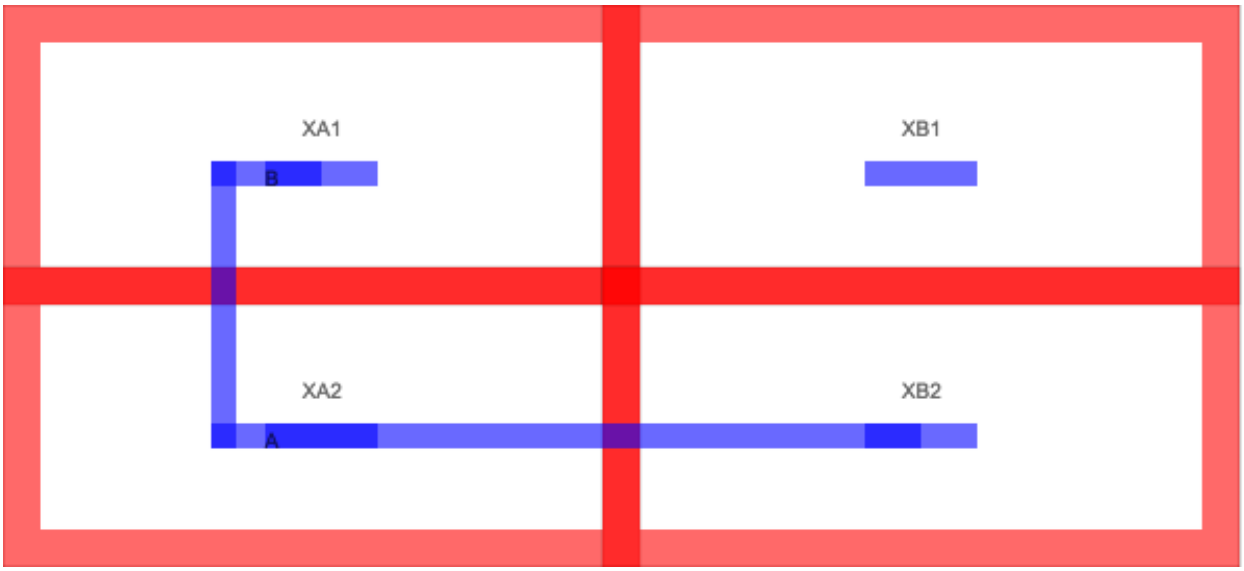


Right (–|-)

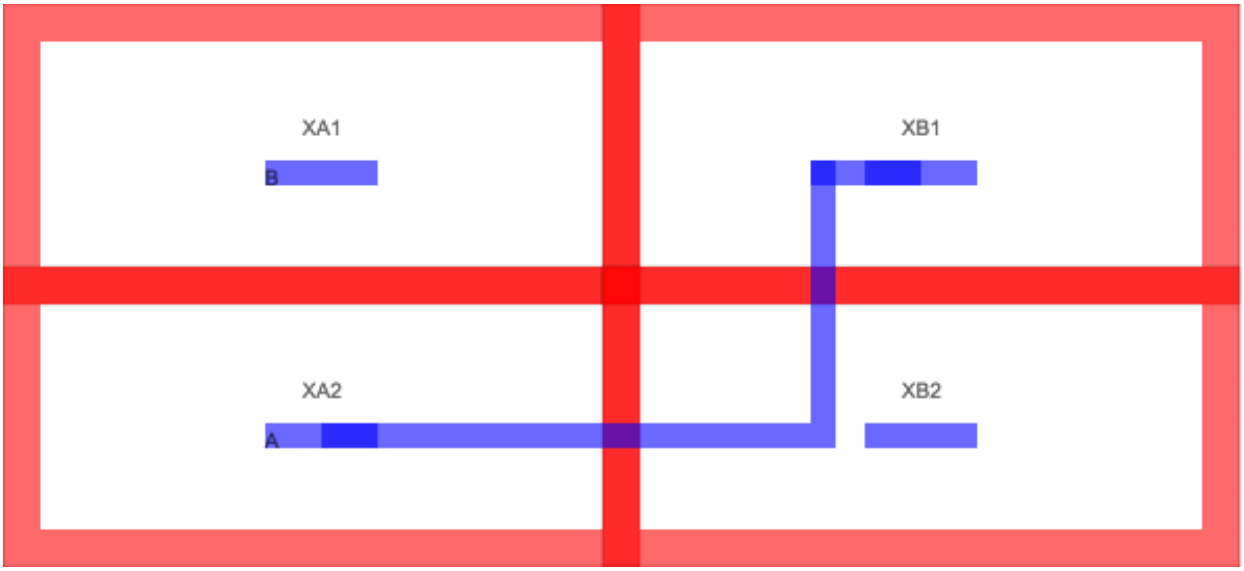
The 'Right' route will go one metal spacing to the right, then up, and or down, and finish the route.

Starting from the left

```
{ "name" : "Directed_StartLeft_Right(--|-)",
  "inherit": "TEST",
  "beforeRoute": [
    {"addDirectedRoutes" : [["M1","S","XB1:S--|-XB2:S"]]}
  ]
}
```



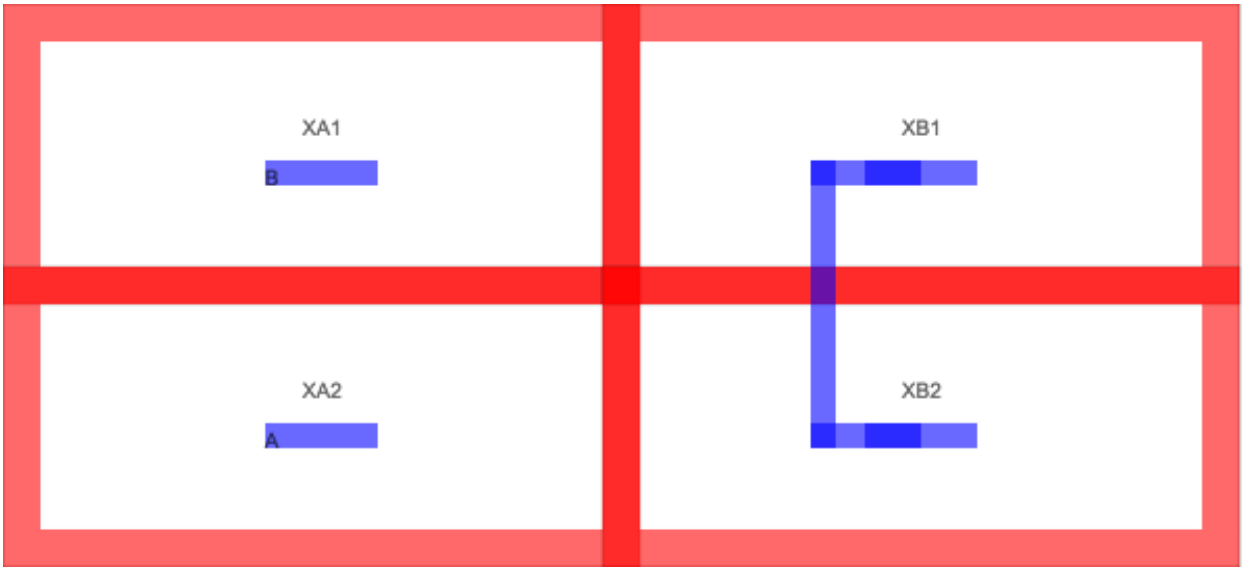

```
{ "name" : "Directed_StartRight_Right(--|-)",
  "inherit": "TEST",
  "beforeRoute": [
    {"addDirectedRoutes" : [["M1","S","XB1:S--|-XA2:S"]]}
  ]
}
```



U left route (|-)

The U route will go one metal spacing left, up or down, and back again.

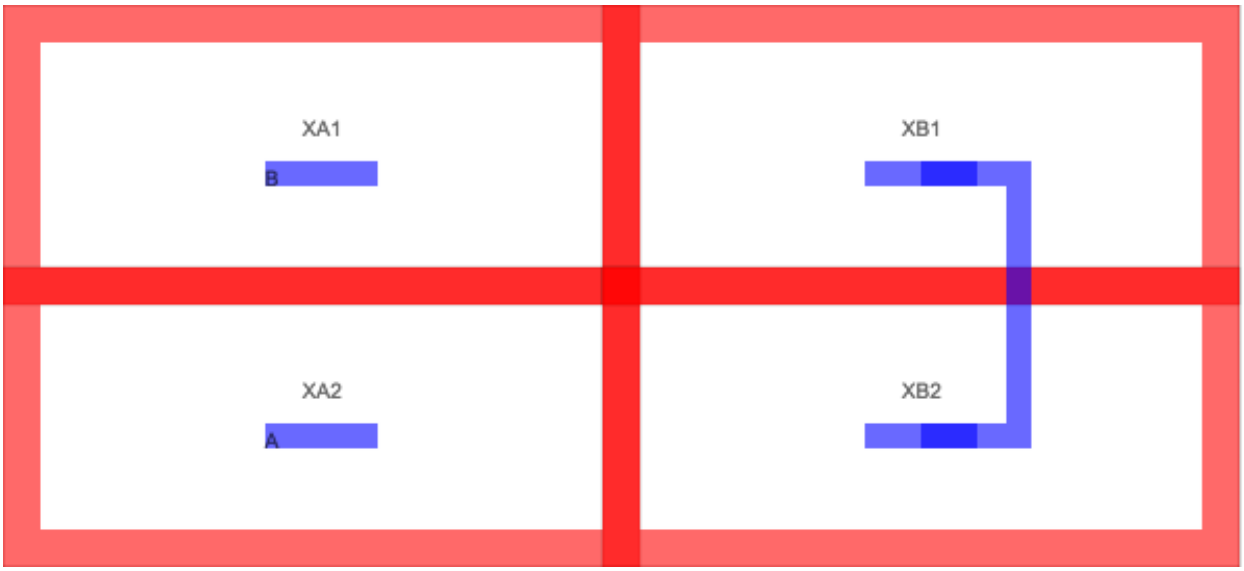
```
{ "name" : "Directed_U_Left(|-)",
  "inherit": "TEST",
  "beforeRoute": [
    {"addDirectedRoutes" : [["M1","S","XB1:S|-XB2:S"]]}
  ]
}
```



U right route (-|)

The U route will go one metal spacing right, up or down, and back again.

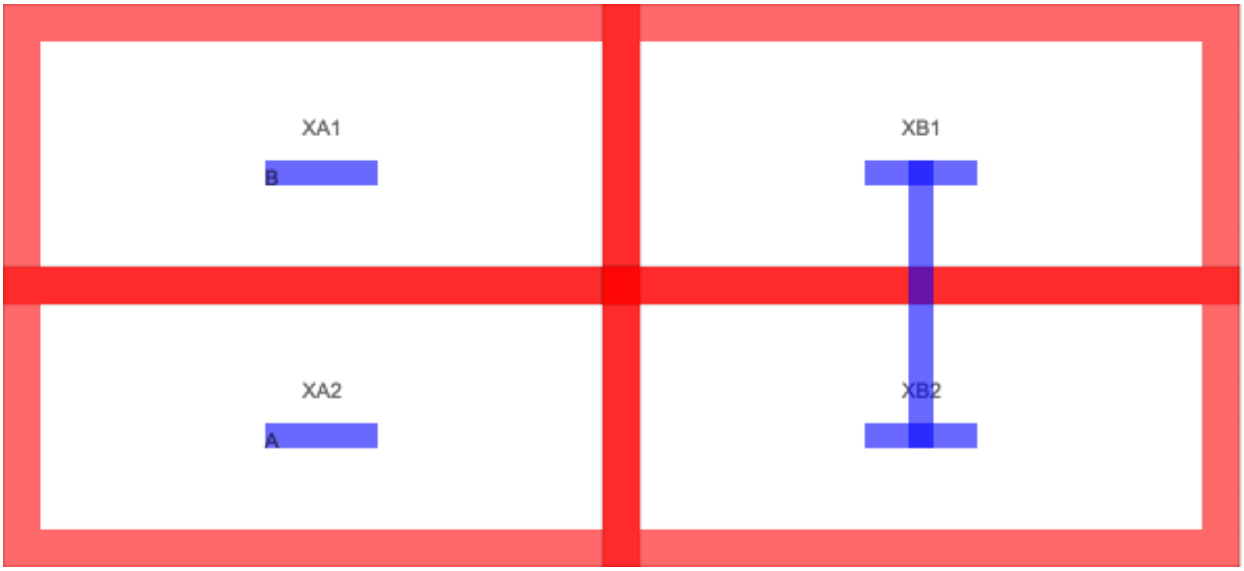
```
{ "name" : "Directed_U_Right(-|)",
  "inherit": "TEST",
  "beforeRoute": [
    {"addDirectedRoutes" : [["M1","S","XB1:S-|XB2:S"]]}
  ]
}
```



Vertical route (||)

The vertical route will find the center of the start rectangles and draw a straight vertical line up, or down to the stop rectangles.

```
{ "name" : "Directed_Vertical(||)",
  "inherit": "TEST",
  "beforeRoute": [
     {"addDirectedRoutes" : [["M1","S","XB1:S||XB2:S"]]}
  ]
}
```



1.3.5 Route Options

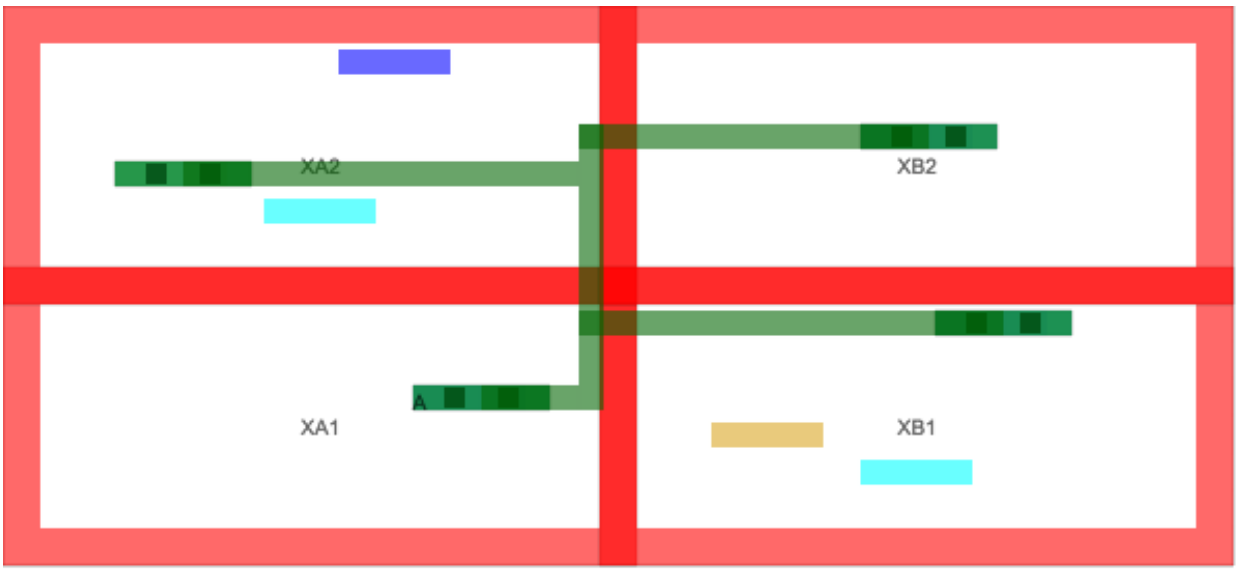
The router has some options to ease routing. Both connectivity routes and directed routes have options. Options are separated by a comma.

Sort options

These options specify which rectangle to route from. They are useful for connectivity route where the start rectangles are not specified.

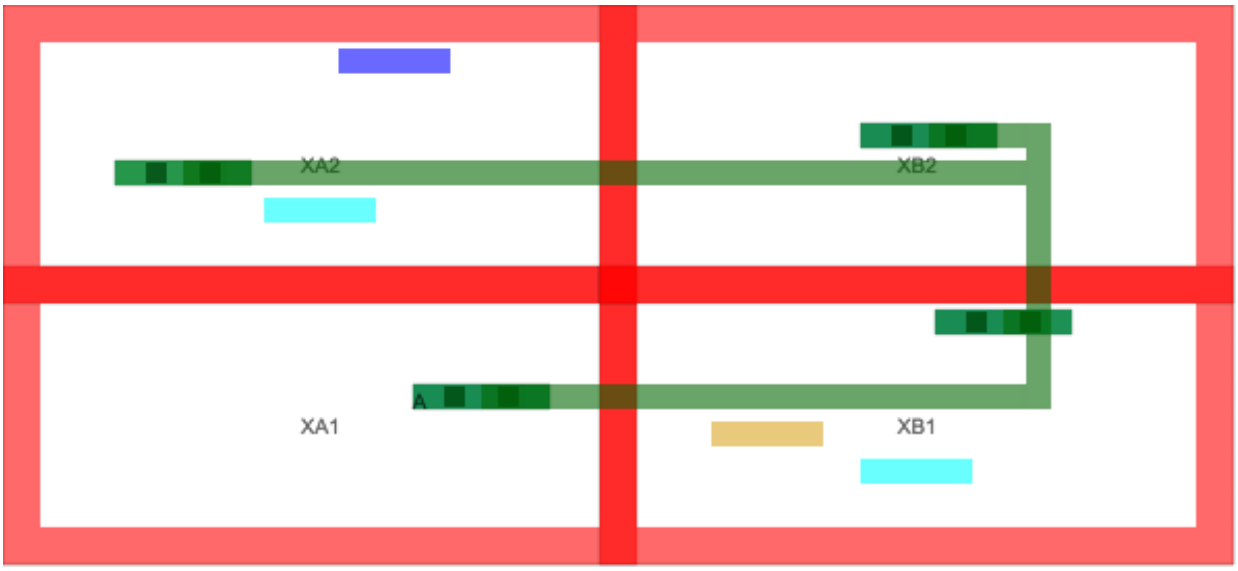
onTopB

Put the bottom rectangle as start rectangle



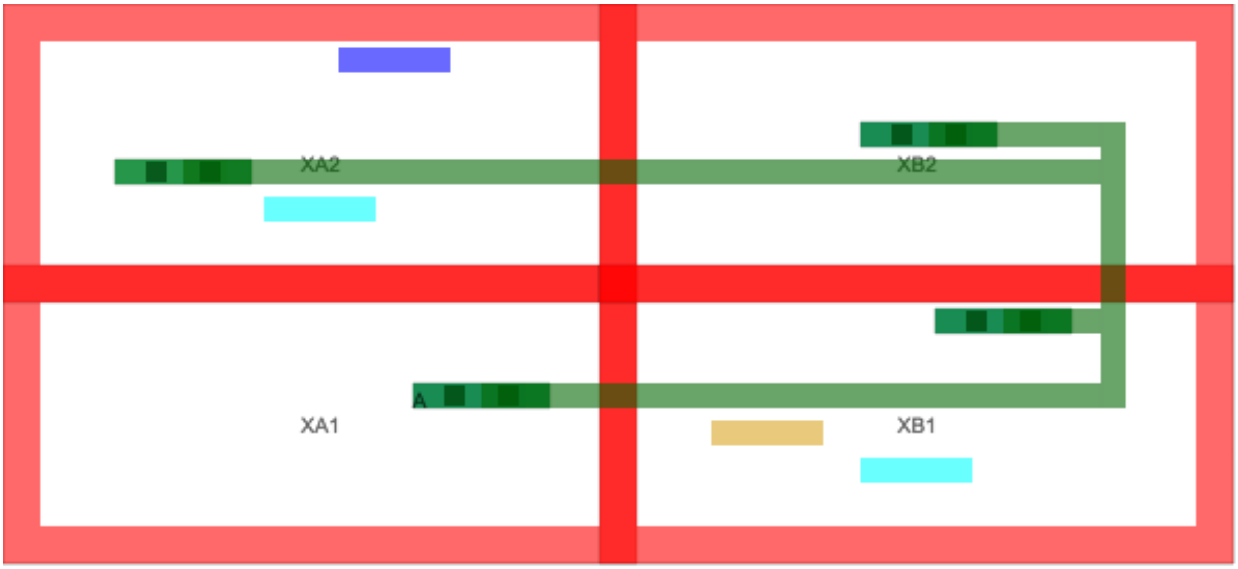
onTopT

Put the top rectangle as start rectangle



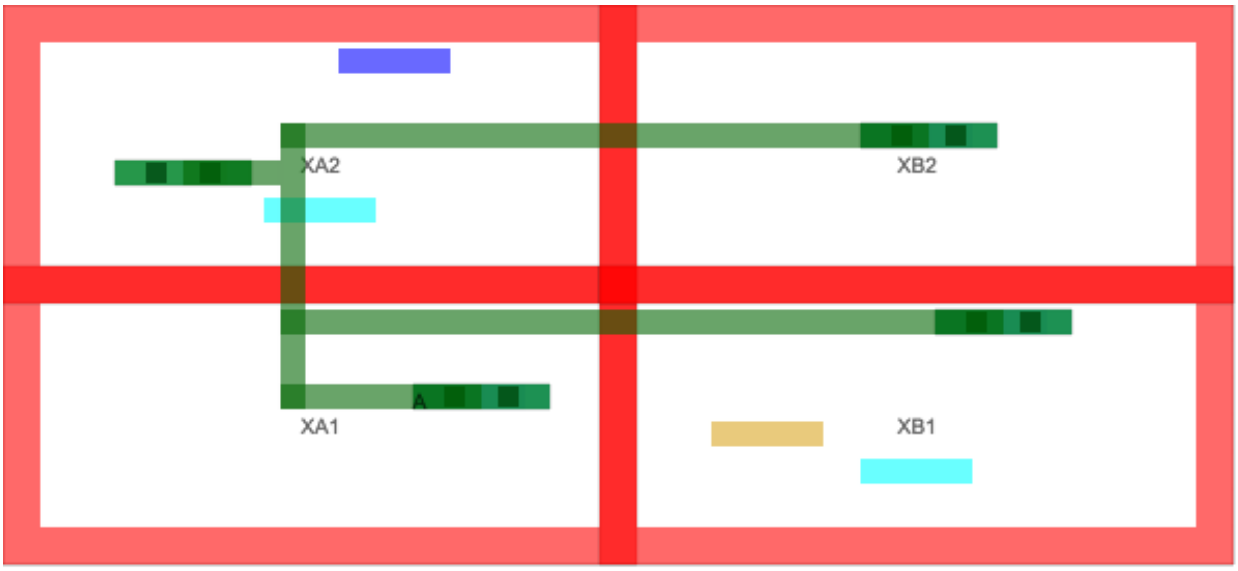
onTopR

Put the right rectangle as start rectangle



onTopL

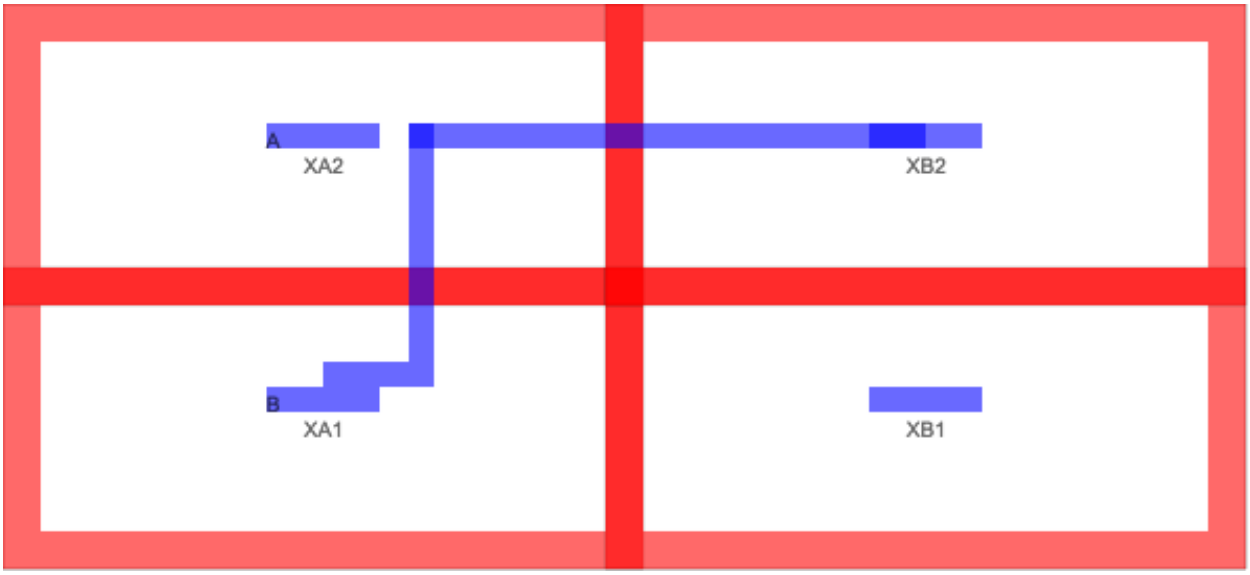
Put the left rectangle as start rectangle



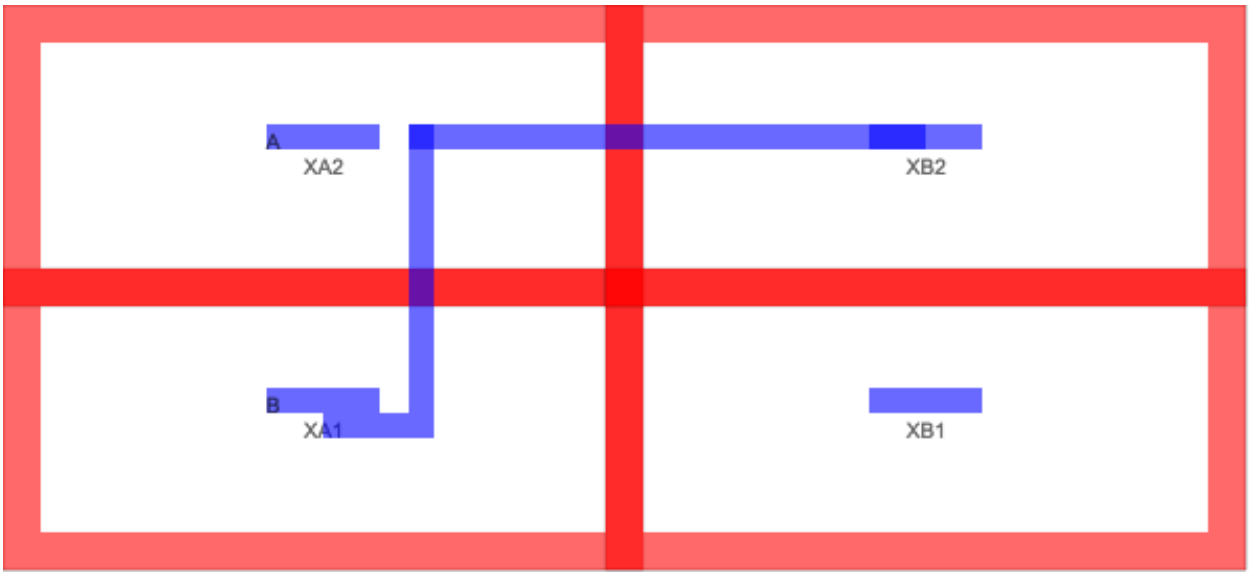
Offset

These options can offset the routing rectangle to avoid nearby routings

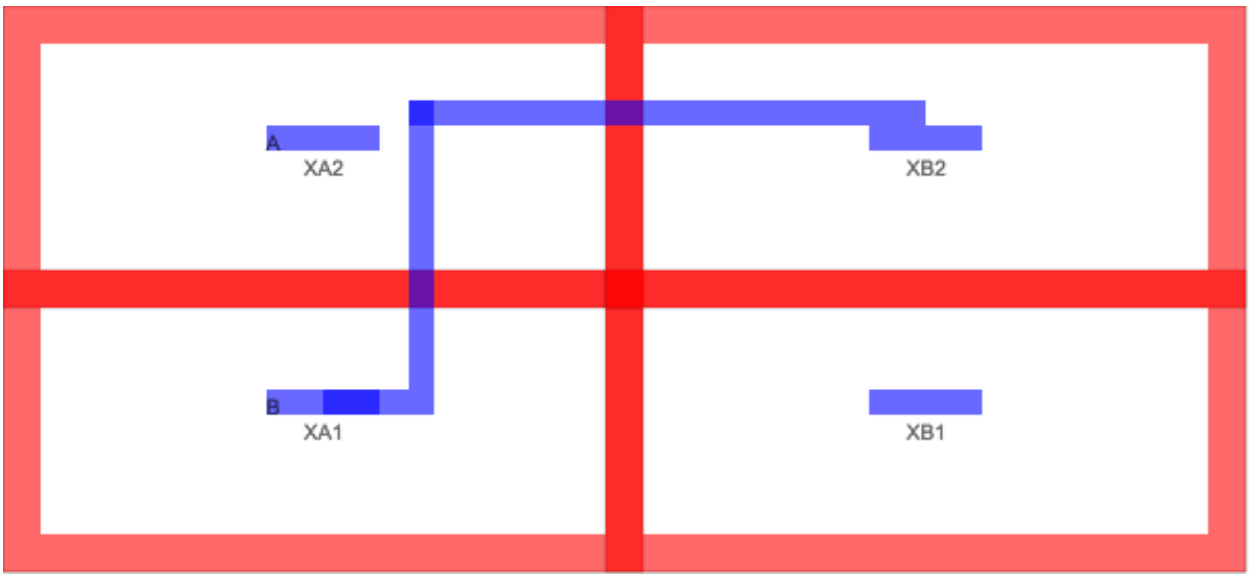
offsethigh



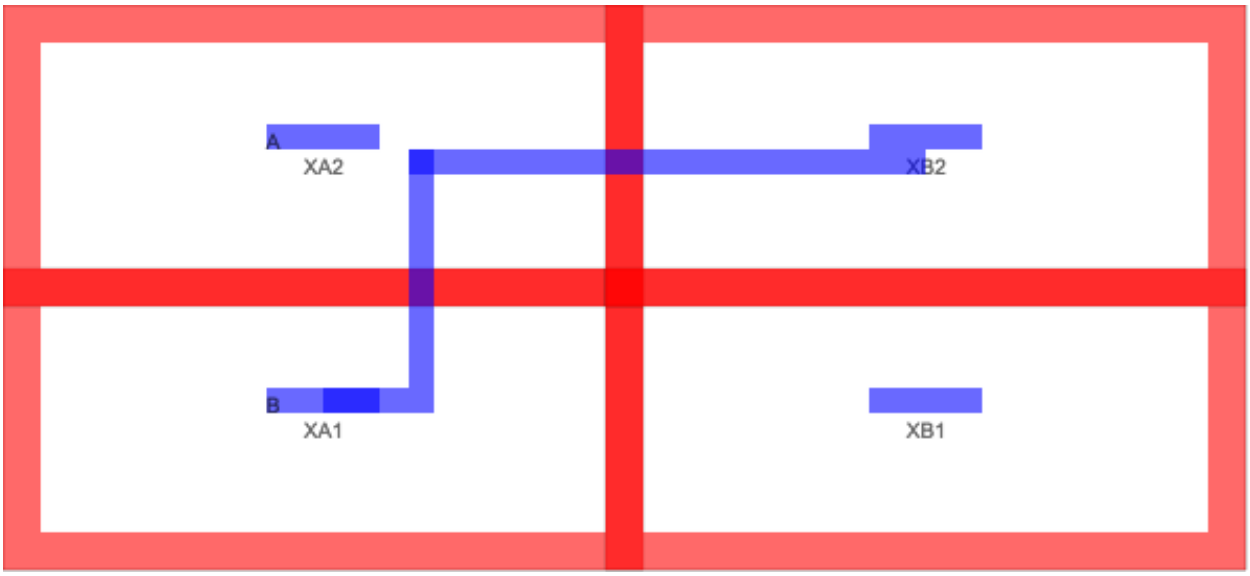
offsetlow



offsethighend



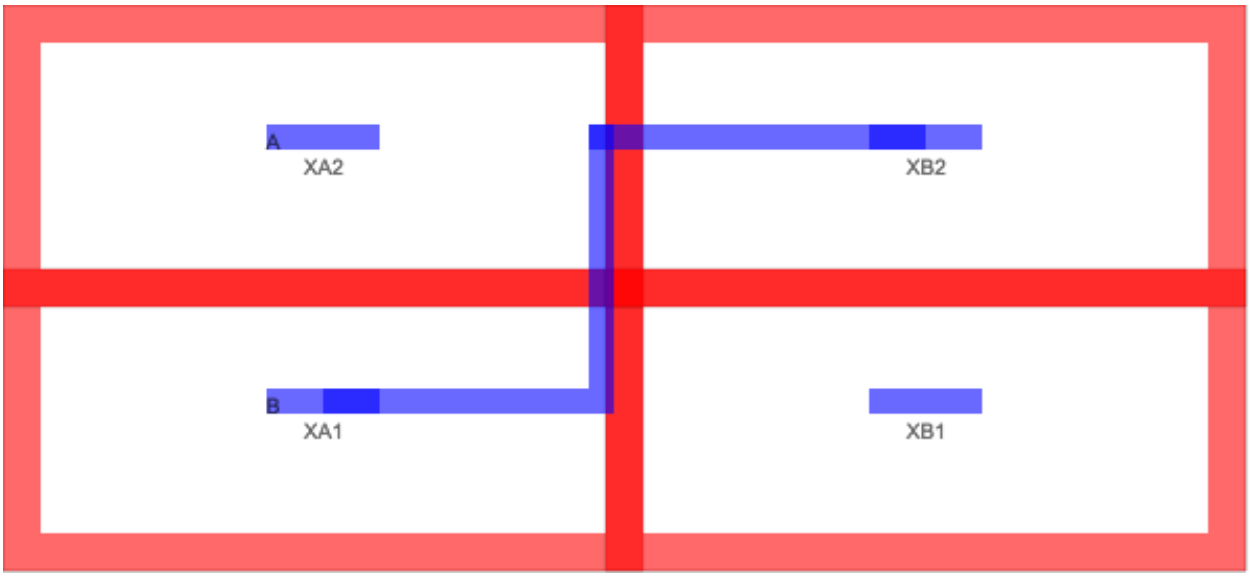
offsetlowend



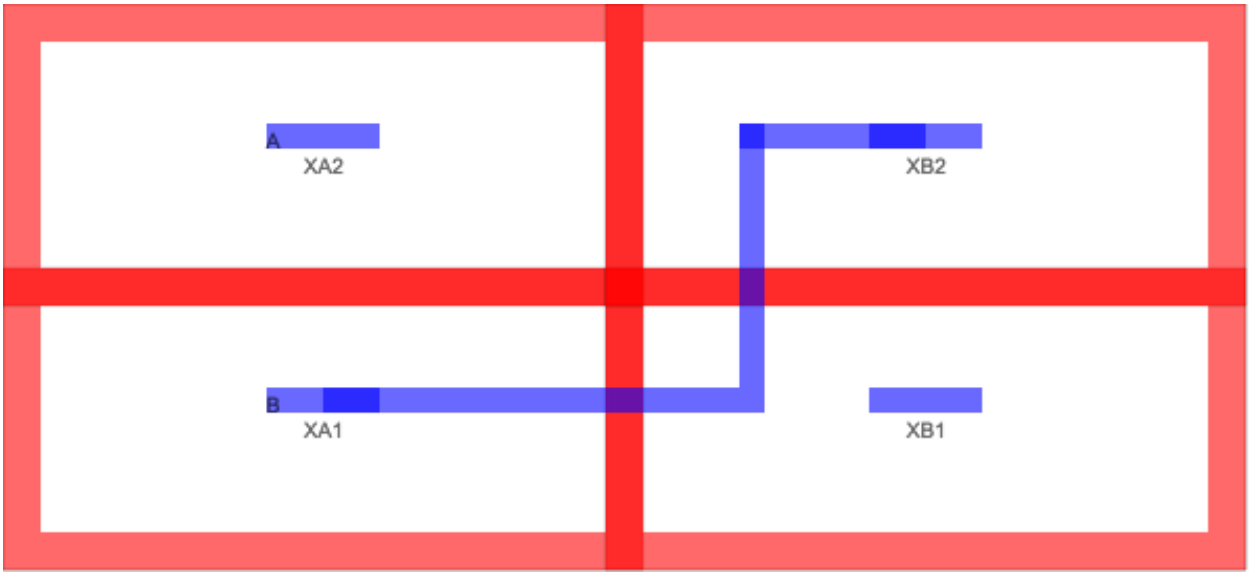
Track

'track' can be used to specifiy how many grids the route is offset. It uses "ROUTE" and "horizontalgrid".

For example 'track4', gives



For example 'track8', gives

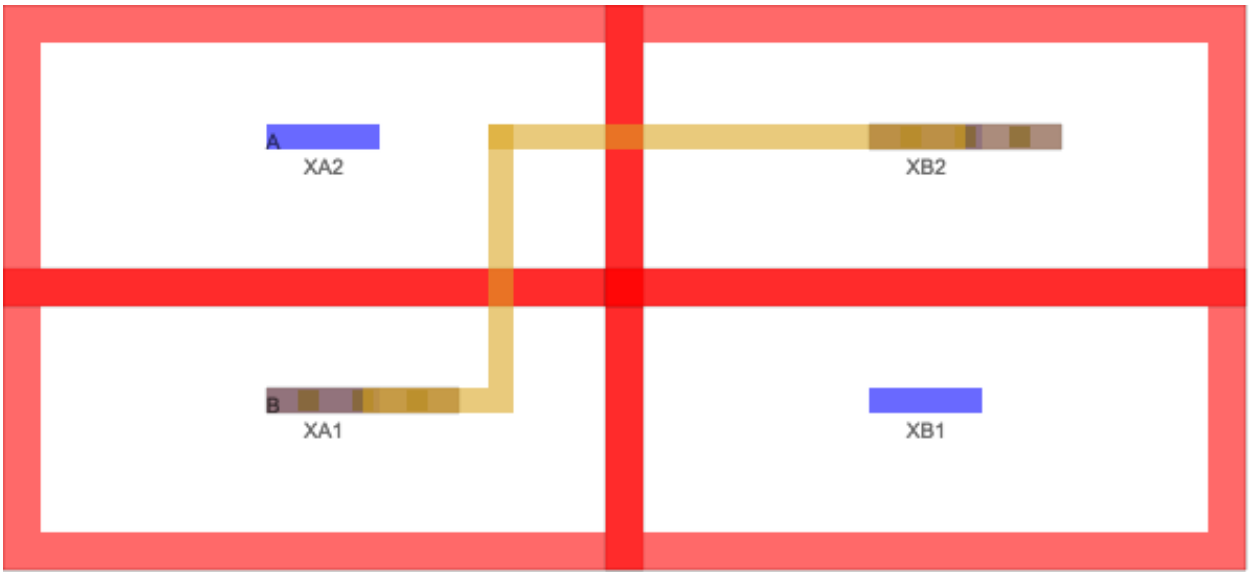


Cuts

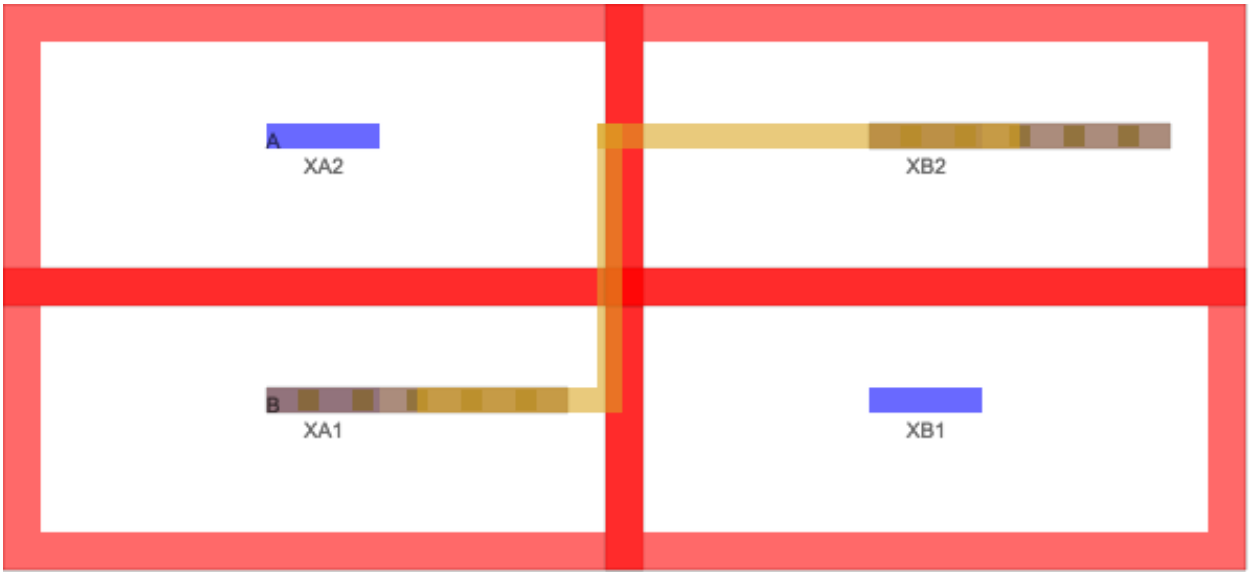
The number of cuts used by the route can be modified. Default is two cuts for the horizontal via.

\d+cuts

For example ‘3cuts’



And for example ‘5cuts’



1.4 API

1.4.1 Main

Layer

class **Layer**

Defines a layer, data for this object is usually loaded from the "layers" section in the technology file

Public Types

enum **MATERIAL_TYPE**

Values:

enumerator **diffusion**

enumerator **poly**

enumerator **metal**

enumerator **cut**

enumerator **metalres**

enumerator **other**

enumerator **marker**

enumerator **implant**

Public Functions

Layer()

Public Members

QString **name**

Name of layer, for example “M1”.

int **number**

GDS layer number.

int **datatype**

GDS layer datatype.

QMap<QString, int> **datatypes**

MATERIAL_TYPE **material**

Type of material.

QString **previous**

Previous layer in routing stack, i.e CO for M1.

QString **next**

Next layer in routing stack, i.e VIA1 for M1.

QString **pin**

Name of pin layer for this layer, i.e. M1_pin.

QString **res**

Name of resistor layer for this layer, i.e. M1_res.

QString **color**

Color of this layer to use in GUI, QColor names can be used.

bool **nofill**

Fill rectangle of this layer in GUI.

bool **visible**

Whether this layer is visible in GUI.

Rules

class **Rules** : public QObject

Public Functions

Rules()

inline **~Rules**()

QList<*Layer**> **getConnectStack**(QString layer1, QString layer2)

Device ***getDevice**(QString dev)

qreal **get**(QString layer, QString rule)

bool **hasRule**(QString layer, QString rule)

double **spiceUnit**()

QString **layerToColor**(QString name)

int **layerToNumber**(QString name)

int **layerToDataType**(QString name)

inline int **gamma**()

inline int **grid**()

void **setRules**(QJsonObject job)

Layer ***getLayer**(QString string)

inline QMap<QString, *Layer**> **layers**()

bool **isLayerBeforeLayer**(QString layer1, QString layer2)

QString **getNextLayer**(QString lay)

QString **getPreviousLayer**(QString lay)

double **toMicron**(int val)

QString **removeDataType**(QString layer)

QString **getDataType**(QString layer)

Public Static Functions

static void **loadRules**(QString path)

static inline *Rules* ***getRules**()

Design

class **Design** : public cIcCore::*Cell*

Public Functions

Design()

bool **read**(QString filename)

bool **readCells**(QString filename)

inline QList<QString> **cellNames**()

cIcSpice::*Subckt* ***getSpiceSubckt**(QJsonObject jobj, QList<QJsonObject> *reverse_parents, QString name)

void **runMethod**(QJsonValue v, QMetaMethod m, *Cell* *c)

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

QJsonObject **readJson**(QString filename)

void **readJsonFile**(QString filename)

void **writeJsonFile**(QString filename, QJsonObject info)

void **addIncludePath**(QString path)

void **setPrefix**(QString prefix)

bool **hasTopCells**()

Console

class **Console**

Public Functions

Console(int argc, char *argv[])

~Console()

virtual void **initialize**(int argc, char *argv[])

virtual void **initialize**()

void **addOption**(Option *o)

Option ***option**(string name)

1.4.2 Basics

Point

class **Point**

Public Functions

inline **Point**()

inline **Point**(int xa, int ya)

inline void **setPoint**(*Point* p)

inline void **setPoint**(int xa, int ya)

inline void **rotate**(int org_x, int org_y, int angle)

inline void **translate**(int dx, int dy)

inline bool **leftOf**(const *Point* &p)

inline bool **over**(const *Point* &p)

inline int **swapX**(int x2)

inline int **swapY**(int y2)

inline QString **toString**()

inline bool **operator==**(*Point* p)

Public Members

int **x**

int **y**

Rect

class **Rect** : public QObject, public cIcCore::SimpleRect

Subclassed by *cIcCore::Cell*, *cIcCore::Port*, *cIcCore::Text*

Public Functions

inline **Rect**()

inline **~Rect**()

inline **Rect**(QString layer, int left, int bottom, int width, int height)

inline **Rect**(const SimpleRect &r)

inline **Rect**(QString layer, const SimpleRect &r)

inline explicit **Rect**(*Rect* *r)

Copy rectangle from pointer (r)

bool **isHorizontal**()

bool **isVertical**()

Rect ***getCopy**()

Return a copy of this rectangle, creates a new rectangle.

Rect ***getCopy**(QString layer)

Get a copy of this rectangle, but with in a different layer.

QString **layer**()

Name of GDS layer.

inline QString **net**()

Net name of this net.

inline void **setNet**(QString net)

inline *Rules* ***getRules**()

Get the *cIcCore::Rules* object.

inline virtual void **translate**(int ax, int ay)

Move this rectangle by ax and ay.

inline virtual void **moveTo**(int x, int y)

Move this rectangle to x and y.

inline virtual void **moveCenter**(int x, int y)

Place center of this rectangel at x and y.

inline virtual void **adjust**(int dx)

Increase the size of this rectangle by dx on all sides.

inline virtual void **adjust**(int dx1, int dy1, int dx2, int dy2)

Rect ***adjustedOnce**(int xp1)

Get a rectangele where each side is moved by xp1. Useful for generating a rectangle to enclose this rectangle by a certain amount

virtual void **mirrorX**(int ax)

Mirror around ax, will send “updated()”.

virtual void **mirrorY**(int ay)

Mirror arround ay, will send “updated()”.

bool **isRect**()

bool **isInstance**()

Check if this is an *cIcCore::Instance* object.

bool **isRoute**()

Check if this is a cIcCore::Routeobject.

bool **isCut**()

Check if this is a *cIcCore::Cut* object.

bool **isCell**()

Check if this is a *cIcCore::Cell* object.

bool **isLayoutCell**()

Check if this is a *cIcCore::LayoutCell* object.

bool **isPort**()

Check if this is a *cIcCore::Port* object.

bool **isText**()

Check if this is a *cIcCore::Text* object.

bool **abutsLeft**(*Rect* *r)

Check if a rectangle is exactly to the left of of this rectangle.

bool **abutsRight**(*Rect* *r)

Check if a rectangle is exactly to the right of of this rectangle.

bool **abutsTop**(*Rect* *r)

Check if a rectangle is exactly to the top of of this rectangle.

bool **abutsBottom**(*Rect**)

Check if a rectangle is exactly to the bottom of of this rectangle.

Rect ***parent**(*Rect* *rect = 0)

Get the rectangle of this.

int **snap**(int x)

Snap to grid, defult 5 ångstrøm.

virtual QString **toString**()

Convert a rectangle to a string that can be printed to console, useful for debug.

void **rotate**(int i)

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

void **setPrefix**(QString prefix)

inline void **setRect**(const SimpleRect &rect)

inline void **setRect**(int x, int y, int width, int height)

Public Slots

void **setLayer**(QString layer)
: Set GDS layer name.

inline void **setLeft**(int left)
: Set the left coordinate (x1)

inline void **setRight**(int right)
: Set the right coordinate (x2)

inline void **setTop**(int top)
: Set the top coordinate (y2)

inline void **setBottom**(int bottom)
: Set the bottom coordinate (y1)

inline void **setHeight**(int height)
: Set height, moves y2.

inline void **setWidth**(int width)
: Set width, moves x2.

inline void **setRect**(QString layer, int x, int y, int width, int height)
: Set coordinates based on rect.

Signals

void **updated**()
: Notify listeners when the rectangle has moved.

Public Static Functions

static QList<*Rect**> **sortLeftOnTop**(QList<*Rect**> rects)

static QList<*Rect**> **sortRightOnTop**(QList<*Rect**> rects)

static QList<*Rect**> **sortBottomOnTop**(QList<*Rect**> rects)

static QList<*Rect**> **sortTopOnTop**(QList<*Rect**> rects)

static *Rect* ***getScaled**(*Rect* *r, int unit)

static *Rect* ***getHorizontalRectangleFromTo**(QString layer, int x1, int x2, int y, int height)

static *Rect* ***getVerticalRectangleFromTo**(QString layer, int x, int y1, int y2, int width)

Port

class **Port** : public cIcCore::*Rect*

Port is a net name attached to a rectangle

Subclassed by *cIcCore::InstancePort*

Public Functions

Port()

~Port()

explicit **Port**(QString name)

QString **name**()
Net name.

void **setName**(QString name)

QString **pinLayer**()

virtual void **set**(*Rect* *r)

virtual *Rect* ***get**()

virtual *Rect* ***get**(QString layer)

virtual QList<*Rect**> **getAll**(QString layer)

virtual void **mirrorX**(int ay) override
Mirror around ax, will send "updated()".

virtual void **mirrorY**(int ax) override
Mirror arround ay, will send "updated()".

virtual QString **toString**() override
Convert a rectangle to a string that can be printed to console, useful for debug.

void **add**(*Rect* *r)

inline bool **isInstancePort**()

virtual QJsonObject **toJson**() override

virtual void **fromJson**(QJsonObject o) override

Public Members

bool **spicePort**

Public Slots

void **updateRect**()

InstancePort

class **InstancePort** : public cIcCore::*Port*

Public Functions

InstancePort(QString name, *Port* *p, *Rect* *parent)

~InstancePort()

virtual void **mirrorX**(int ay) override
Mirror around ax, will send "updated()".

virtual void **mirrorY**(int ax) override
Mirror arround ay, will send "updated()".

inline *Rect* ***parent**()

QString **childName**()

Text

class **Text** : public cIcCore::*Rect*
Text is just a text

Public Functions

Text()

~Text()

explicit **Text**(QString name)

QString **name**()
Net name.

void **setName**(QString name)

virtual QJsonObject **toJson**() override

virtual void **fromJson**(QJsonObject o) override

1.4.3 Basic Cells

Cell

class **Cell** : public cIcCore::*Rect*

Base class for all cells, usually inherited to provide specialization. The methods of this class is called in the following order when a cell is created: *place()*; *route()*; *paint()*;

Subclassed by *cIcCore::Cut*, *cIcCore::Design*, *cIcCore::Guard*, *cIcCore::Instance*, *cIcCore::LayoutCell*, *cIcCore::PatternTile*, *cIcCore::Route*, *cIcCore::RouteRing*

Public Functions

Cell()

~Cell()

Rect ***getRect**(QString layer)

Find the first rectangle in this cell that uses layer.

virtual void **add**(*Rect* *rect)

Add a rectangle to the cell, hooks *updated()* of the child to updateBoundingRect.

virtual void **add**(QList<*Rect**> rects)

virtual void **translate**(int dx, int dy) override

Move this cell, and all children by dx and dy.

virtual void **mirrorX**(int ax) override

Mirror this cell, and all children around ax.

virtual void **mirrorY**(int ay) override

Mirror this cell, and all children around ay.

virtual void **moveTo**(int ax, int ay) override

Move this cell, and all children to ax and ay.

virtual void **moveCenter**(int ax, int ay) override

Center this cell, and all children on ax and ay.

void **meta**(QJsonObject obj)

void **boundaryIgnoreRouting**(QJsonValue obj)

Center this cell, and all children on ax and ay.

void **setBoundaryIgnoreRouting**(bool bir)

bool **boundaryIgnoreRouting**()

void **addPort**(QString name, *Rect* *r)

Shortcut for adding ports.

void **mirrorCenterX**()

Mirror this cell, and all children around horizontal center point (basically flip horizontal)

void **mirrorCenterY**()

virtual SimpleRect **calcBoundingRect**()
Calculate the extent of this cell. Should be overriden by children.

virtual QString **toString**() override
Convert cell to a human readable format, useful for debug.

inline bool **isPhysicalOnly**()
Mark as a physcial only cell.

inline bool **setPhysicalOnly**(bool val)

inline QString **name**()
Name of this cell.

inline QString **setName**(QString val)

inline void **setLibCell**(bool isLibCell)

inline void **setLibPath**(QString path)

inline QString **libPath**()

inline void **setUsed**(bool isUsed)

inline bool **isUsed**()

Port ***getPort**(QString name)
Get the port linked to net name (name)

Port ***getCellPort**(QString name)

QList<*Port**> **ports**()
Get all ports on this cell.

QMap<QString, QList<*Port**>> **allports**()

QList<QString> **allPortNames**()

Port ***updatePort**(QString name, *Rect* *r)
Update rectangle of port, if port does not exist a new one is created

inline cIcSpice::*Subckt* ***subckt**()
Spice subcircuit object.

inline cIcSpice::*Subckt* ***setSubckt**(cIcSpice::*Subckt* *val)

inline QList<*Rect**> **children**()
Get list of all children.

bool **isASpicePort**(QString name)

virtual void **place**()
Place children.

virtual void **route**()
Route children.

virtual void **paint**()
Paint children, useful with a method after route.

virtual void **addAllPorts**()

Automatically add remaing ports.

virtual QList<*Rect**> **findRectanglesByRegex**(QString regex, QString layer)

Find all rectangles by regular expression.

virtual void **findRectangles**(QList<*Rect**> &rects, QString name, QString layer)

virtual QList<*Rect**> **findAllRectangles**(QString regex, QString layer)

virtual QJsonObject **toJson**() override

virtual void **fromJson**(QJsonObject o) override

virtual *Rect* ***cellFromJson**(QJsonObject co)

QList<*Rect**> **getChildren**(QString type)

void **addEnclosingLayers**(QList<QString> layers)

virtual void **updateUsedChildren**()

Public Slots

void **updateBoundingRect**()

Properties

bool physicalOnly

Mark a cell as physical only.

Accessors:\n isPhysicalOnly(), setPhysicalOnly()

Public Static Functions

static SimpleRect **calcBoundingRect**(QList<*Rect**> children)

static SimpleRect **calcBoundingRect**(QList<*Rect**> children, bool ignoreBoundaryRouting)

static bool **isEmpty**(*Cell* *c)

static inline bool **hasCell**(QString cell)

Check if this cell contains a cell with name cell.

static inline *Cell* ***getCell**(QString cell)

Get a named cell, returns empty cell if it does not exist, so you should check that the cell exists in this cell first

static inline QList<*Cell**> **getAllCells**()

Get a list of all cells in this design.

static inline *Cell* ***addCell**(QString cell, *Cell* *c)

Add a cell to the list of all cells.

static inline *Cell* ***addCell**(*Cell* *c)

Add a cell, and use the cell->*name()* as key.

Cut

class **Cut** : public cIcCore::*Cell*

Public Functions

Cut(QString layer1, QString layer2, int horizontal_cuts, int vertical_cuts)

Cut(QString layer1, QString layer2, *Rect* *r)

~Cut()

Public Static Functions

static QString **makeName**(QString layer1, QString layer2, int horizontal_cuts, int vertical_cuts)

static *Instance* ***getInstance**(QString layer1, QString layer2, int horizontal_cuts, int vertical_cuts)

static QList<*Rect**> **getCutsForRects**(QString layer1, QList<*Rect**>, int horizontal_cuts, int vertical_cuts)

static QList<*Rect**> **getCutsForRects**(QString layer1, QList<*Rect**>, int horizontal_cuts, int vertical_cuts, bool alignLeft)

static QList<*Rect**> **getVerticalFillCutsForRects**(QString layer1, QList<*Rect**> rects, int horizontal_cuts)

static QList<*Cut**> **getCuts**()

Instance

class **Instance** : public cIcCore::*Cell*

Subclassed by *cIcCore::InstanceCut*

Public Functions

Instance()

~Instance()

inline *Cell* ***cell**()

inline QString **instanceName**()

inline QString **id**()

virtual SimpleRect **calcBoundingRect**() override

Calculate the extent of this cell. Should be overriden by children.

inline QString **angle**()

void **setAngle**(QString angle)

inline void **setCell**(*Cell* *cell)

inline cIcSpice::*SubcktInstance* ***subcktInstance**()

void **setSubcktInstance**(cIcSpice::*SubcktInstance* *inst)

virtual QList<*Rect**> **findRectanglesByRegex**(QString regex, QString layer) override
Find all rectangles by regular expression.

QList<*Rect**> **findRectanglesByNode**(QString node, QString filterChild)

virtual QString **toString**() override
Convert cell to a human readable format, useful for debug.

void **transform**(*Rect* *r)

Rect ***getRect**(QString layer)

Point ***getCellPoint**()

virtual QJsonObject **toJson**() override

virtual void **fromJson**(QJsonObject o) override

void **setCell**(QString cell)

virtual void **updateUsedChildren**() override

Public Static Functions

static *Instance* ***getInstance**(QString cell)

InstanceCut

class **InstanceCut** : public cIcCore::*Instance*

Public Functions

inline **InstanceCut**()

inline **~InstanceCut**()

1.4.4 Pattern Cells

PatternTile

class **PatternTile** : public cIcCore::*Cell*

Subclassed by *cIcCore::PatternCapacitor*, *cIcCore::PatternHighResistor*, cIcCore::PatternHighResistorNoBulk, *cIcCore::PatternResistor*, *cIcCore::PatternTransistor*, cIcCore::ResistorCell

Public Functions

PatternTile()

PatternTile(const *PatternTile*&)

~PatternTile()

void **fillCoordinatesFromString**(QJsonArray ar)

fillCoordinatesFromString

void **verticalMultiplyVector**(QJsonArray ar)

verticalMultiplyVector

EXPERIMENTAL! Vector length must be the same length as the number of rows in fillCoordinatesFromString. Multiplies the height of a cell with the number in the vector multiplier

void **getRuleForHorizontalGrid**(QJsonArray ar)

void **getRuleForVerticalGrid**(QJsonArray ar)

void **copyColumn**(QJsonObject obj)

Copy a column set of a fillCoordinateFromStrings

A CopyColumn object consists of the following { "count" : int, "offset" : int, "length" : int, ["position" : int] }

- count is the number of times to copy the column set
- offset is the column set index from left edge of string
- length is the length of the column set
- position is optional, and if not set will be equal to offset. If position is given, then the copies of a column is inserted at that point

Parameters

QJsonObject – Array of CopyColumn objects

void **copyRow**(QJsonObject obj)

Copy a row set of a fillCoordinateFromStrings

A CopyRows object consists of the following { "count" : int, "offset" : int, "length" : int, ["position" : int] }

- count is the number of times to copy the row set
- offset is the row set index from top edge of string
- length is the length of the row set
- position is optional, and if not set will be equal to offset. If position is given, then the copies of a row set is inserted at that point

Parameters
QJsonObject – Array of CopyRows objects

void **copyLayer**(QJsonArray ar)

void **addEnclosure**(QJsonArray ar)

void **addEnclosureByRectangle**(QJsonArray ar)

void **addEnclosuresByRectangle**(QJsonArray ar)

virtual QMap<QString, QVariant> **initFillCoordinates**()

inline virtual void **onFillCoordinate**(QChar, QString, int, int, QMap<QString, QVariant>&)

virtual void **onPaintEnclosure**(*Rect**)

inline virtual void **endFillCoordinate**(QMap<QString, QVariant>&)

inline virtual void **paintRect**(*Rect**, QChar, int, int)

virtual void **paint**() override
Paint children, useful with a method after route.

virtual SimpleRect **calcBoundingRect**() override
Calculate the extent of this cell. Should be overriden by children.

inline qreal **minPolyLength**()

inline qreal **setMinPolyLength**(qreal val)

inline bool **metalUnderMetalRes**()

inline void **setMetalUnderMetalRes**(bool val)

inline qreal **verticalGrid**()

inline qreal **setVerticalGrid**(qreal val)

inline double **verticalGridMultiplier**()

inline double **setVerticalGridMultiplier**(double val)

inline int **horizontalGrid**()

inline int **setHorizontalGrid**(int val)

inline double **horizontalGridMultiplier**()

inline double **setHorizontalGridMultiplier**(double val)

inline qreal **widthoffset**()

inline qreal **setWidthoffset**(qreal widthoffset)

inline qreal **heightoffset**()

inline qreal **setHeightoffset**(qreal heightoffset)

inline qreal **xoffset**()

inline qreal **setXoffset**(qreal xoffset)

inline int **polyWidthAdjust**()

inline int **setPolyWidthAdjust**(int val)

inline qreal **yoffset**()

inline qreal **setYoffset**(qreal yoffset)

inline int **mirrorPatternString**()

inline int **setMirrorPatternString**(int mirrorPatternString)

Public Members

QMap<QChar, PatternData*> **Pattern**

Properties

qreal minPolyLength

minimum poly length

Accessors:\n minPolyLength(), setMinPolyLength()

qreal widthoffset

offset to add to the width

Accessors:\n widthoffseth(), setWidthOffset()

qreal heightoffset

offset to add to the width

Accessors:\n height(), setHeightOffset()

`qreal verticalGrid`

Override vertical grid.

Accessors:\n verticalGrid(), setVerticalGrid()

`qreal horizontalGrid`

Override horizontal grid.

Accessors:\n horizontalGrid(), setHorizontalGrid()

`double verticalGridMultiplier`

multiply vertical grid by an number

Accessors:\n verticalGridMultiplier(), setVerticalGridMultiplier()

`double horizontalGridMultiplier`

multiply horizontal grid by an number

Accessors:\n horizontalGridMultiplier(), setHorizontalGridMultiplier()

`qreal yoffset`

add offset to Y origin coordinate

Accessors:\n yoffset(), setYoffset()

`qreal xoffset`

add offset to X origin coordinate

Accessors:\n xoffset(), setXoffset()

`int mirrorPatternString`

Mirror the pattern string after creation.

Accessors:\n mirrorPatternString(), setMirrorPatternString()

`int polyWidthAdjust`

Adjust the poly width.

Accessors:\n polyWidthAdjust(), setPolyWidthAdjust()

bool metalUnderMetalRes

Add metal under metal-resistor layer, depends on technology.

Accessors:\n metalUnderMetalRes(), setMetalUnderMetalRes()

Public Static Attributes

static QMap<QString, QStringList> **Patterns**

PatternResistor

class **PatternResistor** : public cIcCore::*PatternTile*

Public Functions

virtual void **paintRect**(*Rect**, QChar, int, int)

PatternResistor()

~PatternResistor()

virtual QMap<QString, QVariant> **initFillCoordinates**()

virtual void **onFillCoordinate**(QChar c, QString layer, int x, int y, QMap<QString, QVariant> &data)

virtual void **endFillCoordinate**(QMap<QString, QVariant> &data)

PatternHighResistor

class **PatternHighResistor** : public cIcCore::*PatternTile*

Public Functions

virtual void **paintRect**(*Rect**, QChar, int, int)

PatternHighResistor()

~PatternHighResistor()

virtual void **onFillCoordinate**(QChar c, QString layer, int x, int y, QMap<QString, QVariant> &data)

virtual void **onPaintEnclosure**(*Rect* *r)

virtual void **endFillCoordinate**(QMap<QString, QVariant> &data)

PatternCapacitor

class **PatternCapacitor** : public cIcCore::*PatternTile*

Public Functions

virtual void **paintRect**(*Rect**, QChar, int, int)

PatternCapacitor()

~PatternCapacitor()

virtual void **onPaintEnd**()

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

virtual void **onFillCoordinate**(QChar c, QString layer, int x, int y, QMap<QString, QVariant> &data)

PatternTransistor

class **PatternTransistor** : public cIcCore::*PatternTile*

Public Functions

PatternTransistor()

~PatternTransistor()

inline QString **mosType**()

inline void **setMosType**(QString mosType)

virtual QMap<QString, QVariant> **initFillCoordinates**()

virtual void **onFillCoordinate**(QChar c, QString layer, int x, int y, QMap<QString, QVariant> &data)

virtual void **endFillCoordinate**(QMap<QString, QVariant> &data)

virtual void **paintRect**(*Rect* *r, QChar c, int x, int y)

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

Properties

QString mosType

1.4.5 Route

Route

class **Route** : public cIcCore::*Cell*

Public Functions

Route(QString net, QString layer, QList<*Rect**> start, QList<*Rect**> stop, QString options, QString routeType)

~Route()

virtual void **addStartCuts**()

virtual void **addEndCuts**()

virtual QList<*Rect**> **addCuts**(QList<*Rect**>, QList<*Rect**>&, int cuts_, int vcuts_)

virtual void **route**() override

Route children.

void **routeUHorizontal**()

void **addVertical**(int x)

void **applyOffset**(int width, *Rect* *start, Offset offset)

void **hasMatch**(QString options)

int **getIntegerFromMatch**(QString regex, QString options, int defaultValue)

QString **getQStringFromMatch**(QString regex, QString options, QString defaultValue)

Guard

class **Guard** : public cIcCore::*Cell*

Public Functions

Guard(*Rect* *r, QList<QString> layers)

~Guard()

Graph

class **Graph**

Public Functions

inline void **append**(*Port* *p)

QJsonObject **toJson**()

inline QList<*Rect**> **getRectangles**(QString excludeInstances, QString includeInstances, QString layer)

Public Members

QList<*Port**> **ports**

QString **name**

RouteRing

class **RouteRing** : public cIcCore::*Cell*

Public Functions

RouteRing()

RouteRing(QString layer, QString name, *Rect* *size, QString location, int xgrid, int ygrid, int width)

~RouteRing()

void **addRoute**(*Rect* *r, QString layer, QString options, QString location)

void **trimRouteRing**(QString location, QString whichEndToTrim)

virtual void **translate**(int dx, int dy) override
Move this cell, and all children by dx and dy.

virtual void **moveTo**(int ax, int ay) override
Move this cell, and all children to ax and ay.

Rect ***get**(QString location)

Rect ***getDefault**()

Rect ***getPointer**(QString location)

1.4.6 Layout Cells

LayoutCell

class **LayoutCell** : public cIcCore::*Cell*

Subclassed by *cIcCells::CDAC*, *cIcCells::CapCell*, cIcCells::CapCellV2, cIcCells::PhCapCell, *cIcCells::SAR*, *cIcCore::LayoutRotateCell*

Public Functions

LayoutCell()

~LayoutCell()

void **setYoffsetHalf**(QJsonValue obj)

void **noPowerRoute**(QJsonValue obj)

void **placeHorizontal**(QJsonValue obj)

void **addDirectedRoute**(QJsonArray obj)

void **addConnectivityRoute**(QJsonArray obj)

void **addPortOnRect**(QJsonArray obj)

void **addPortRectangle**(QJsonArray obj)

void **addVia**(QJsonArray obj)

void **addConnectivityVia**(QJsonArray obj)

void **addPortVia**(QJsonArray obj)

void **addVerticalRect**(QJsonArray obj)

void **addRouteRing**(QJsonArray obj)

void **parseSubckt**(QJsonObject obj)

void **addPowerRing**(QJsonArray obj)

void **addRouteConnection**(QJsonArray obj)

void **addPowerConnection**(QJsonArray obj)

void **trimRouteRing**(QJsonArray obj)

void **addRectangle**(QJsonArray obj)

void **addRouteHorizontalRect**(QString layer, QString rectpath, int x, QString name)

void **addRouteHorizontalRect**(QJsonArray obj)

void **addGuard**(QJsonArray obj)

void **addHorizontalRect**(QJsonArray obj)

void **alternateGroup**(QJsonValue obj)

void **resetOrigin**(QJsonValue obj)

void **addPortOnEdge**(QJsonArray obj)

void **addPortOnEdge**(QString layer, QString port, QString location, QString routeType, QString options)

void **setSpiceParam**(QJsonArray obj)

void **addGuard**(QString port, double gridMultiplier, QList<QString> layers)

QList<*Graph**> **getNodeGraphs**(QString regex)

void **noPowerRoute**()

Instance ***addInstance**(cIcSpice::*SubcktInstance* *ckt, int x, int y)

void **addRectangle**(QString layer, int x1, int y1, int width, int height, QString angle)

void **addPortRectangle**(QString layer, int x1, int y1, int width, int height, QString angle, QString port)

void **addConnectivityRoute**(QString layer, QString regex, QString routeType, QString options, QString cuts, QString excludeInstances, QString includeInstances)

void **trimRouteRing**(QString path, QString location, QString whichEndToTrim)

void **addRouteRing**(QString layer, QString name, QString location, int widthmult, int spacemult)

void **addRouteRing**(QString layer, QString name, QString location, int widthmult, int spacemult, bool useGridForSpace)

void **addPowerRing**(QString layer, QString name, QString location, int widthmult, int spacemult)

void **addPowerConnection**(QString name, QString includeInstances, QString location)

void **addRouteConnection**(QString path, QString includeInstances, QString layer, QString location, QString options)

void **addRouteConnection**(QString path, QString includeInstances, QString layer, QString location, QString options, QString routeTypeOverride)

Instance ***getInstanceFromInstanceName**(QString instanceName)

virtual void **place**() override

Place children.

virtual void **route**() override

Route children.

virtual void **addAllPorts**() override

Automatically add remaing ports.

virtual void **routePower**()

virtual void **paint**() override

Paint children, useful with a method after route.

QList<QString> **nodeGraphList**()

QStringList **expandBus**(QString name)

virtual QList<*Rect**> **findRectanglesByNode**(QString node, QString filterChild, QString filterInstance)

virtual void **fromJson**(QJsonObject obj) override

virtual *Rect* ***cellFromJson**(QJsonObject co) override

virtual QJsonObject **toJson**() override

void **addPowerRoute**(QString net, QString excludeInstances)

LayoutRotateCell

class **LayoutRotateCell** : public cIcCore::*LayoutCell*

Public Functions

LayoutRotateCell()

~LayoutRotateCell()

void **rotateAngle**(QJsonValue s)

void **rotateAngle**(QString s)

virtual void **place**() override
Place children.

virtual void **paint**() override
Paint children, useful with a method after route.

1.4.7 Complex Cells

CapCell

class **CapCell** : public cIcCore::*LayoutCell*

Public Functions

virtual void **place**()
Place children.

virtual SimpleRect **calcBoundingRect**()
Calculate the extent of this cell. Should be overriden by children.

void **addContacts**(QString name, QString node, int y, QList<int> array, Rect *r)

Rect ***getAvssConnectRect**(Rect *rect)

void **usem3**(QJsonValue obj)
Use Metal 3.

void **usem5**(QJsonValue obj)
Use Metal 5 shield.

void **heightIncreaseMult**(QJsonValue obj)
Increase multiplier height.

CDAC

class **CDAC** : public cIcCore::*LayoutCell*

Public Functions

virtual void **place**()
Place children.

virtual void **route**()
Route children.

virtual void **paint**()
Paint children, useful with a method after route.

SAR

class **SAR** : public cIcCore::*LayoutCell*

Public Functions

virtual void **place**()
Place children.

virtual void **route**()
Route children.

void **usem5**(QJsonValue obj)
Use Metal 5 shield.

int **getCellWidth**(SARgroup groups, QString group)

cIcCore::*Instance* ***placeAlternateMirror**(SARgroup groups, QString group, int i, int x, int y, int xoffset)

int **addSarRouting**(int y, int msw, int mw)

Public Static Functions

static bool **sortGraph**(cIcCore::*Graph* *a, cIcCore::*Graph* *b)

1.4.8 Spice

SpiceObject

class **SpiceObject** : public QObject

Subclassed by *cIcSpice::SpiceDevice*, *cIcSpice::Subckt*, *cIcSpice::SubcktInstance*

Public Functions

SpiceObject()

~SpiceObject()

QString **name**()

virtual QJsonObject **toJson**()

void **fromJson**(QJsonObject o)

virtual QString **setName**(QString val)

int **lineNumber**()

int **setLineNumber**(int val)

QList<QString> **spiceStr**()

QList<QString> **setSpiceStr**(QList<QString> val)

QStringList **nodes**()

QStringList **setNodes**(QStringList val)

QVariantMap **properties**()

QString **spiceType**()

void **setSpiceType**(QString type)

QString **deviceName**()

void **setDeviceName**(QString name)

virtual QString **toSpice**(QString instance, QStringList nodes)

virtual void **setProperty**(QString key, int val)

virtual void **setProperty**(QString key, QString val)

virtual void **setProperty**(QString key, double val)

virtual bool **hasProperty**(QString key)

virtual QString **getPropertyString**(QString key)

void **setPrefix**(QString prefix)

Properties

QString name

QStringList spiceStr

QStringList nodes

int lineNumber

SpiceDevice

class **SpiceDevice** : public cIcSpice::*SpiceObject*

Subclassed by *cIcSpice::Capacitor*, *cIcSpice::Mosfet*, *cIcSpice::Resistor*

Public Functions

SpiceDevice()

~SpiceDevice()

virtual QString **toSpice**(QString instance, QStringList nodes)

virtual QString **toSpice**()

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

Mosfet

class **Mosfet** : public cIcSpice::*SpiceDevice*

Public Functions

Mosfet()

Mosfet(const *Mosfet* &mos)

~Mosfet()

virtual QString **toSpice**(QString instance, QStringList nodes)

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

virtual QString **toSpice**()

Resistor

class **Resistor** : public cIcSpice::*SpiceDevice*

Public Functions

Resistor(QStringList n)

Resistor()

void **init**(QStringList n)

~Resistor()

virtual QString **toSpice**(QString instance, QStringList nodes)

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

virtual QString **toSpice**()

Public Members

double **width**

Capacitor

class **Capacitor** : public cIcSpice::*SpiceDevice*

Public Functions

Capacitor(QStringList n)

Capacitor()

void **init**(QStringList n)

~Capacitor()

virtual QString **toSpice**(QString instance, QStringList nodes)

virtual QJsonObject **toJson**()

Subckt

class **Subckt** : public cIcSpice::*SpiceObject*

Public Functions

Subckt()

Subckt(QList<QString> buffer)

~Subckt()

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

void **parse**(QList<QString> buffer, int line)

inline QList<*SubcktInstance**> **instances**()

inline QList<*SpiceDevice**> **devices**()

inline void **setLibPath**(QString path)

inline QString **libPath**()

inline void **add**(*SubcktInstance* *s)

inline void **add**(*SpiceDevice* *s)

SubcktInstance ***getInstance**(QString name)

inline void **addSubckt**()

Properties

QString name

Public Static Functions

static *Subckt* ***getInstanceSubckt**(*SubcktInstance**)

SubcktInstance

class **SubcktInstance** : public cIcSpice::*SpiceObject*

Public Functions

SubcktInstance()

SubcktInstance(QString buffer)

~SubcktInstance()

virtual QString **setName**(QString val)

inline QString **subcktName**()

inline QString **setSubcktName**(QString val)

inline QString **groupName**()

inline QString **groupTag**()

void **parse**(QString buffer, int line_number)

virtual QJsonObject **toJson**()

virtual void **fromJson**(QJsonObject o)

Properties

QString subcktName

SpiceParser

class **SpiceParser** : public QObject

Public Functions

SpiceParser()

~SpiceParser()

void **parseFile**(QString filename)

inline QMap<QString, *Subckt**> **subckt**()

inline *Subckt* ***getSubckt**(QString name)

void **parseSubckt**(int line_number, QList<QString> subckt_buffer)

INDEX

C