The Symbolic To Real Mask Translation Approach in the Alliance CAD System

Frédéric Pétrot and Franck Wajsbürt
Équipe CAO-VLSI du laboratoire MASI
Université Pierre et Marie Curie
4, Place Jussieu, 75252 Paris cedex 5

Abstract

This report aims at explaining the way to parameterize the symbolic layout to target process layout tool of the Alliance CAD system. Because of the human investment required for the design of physical VLSI libraries, a process independent approach to layout is a must. Introducing a new symbolic approach to layout, Alliance requires also an efficient way to translate the process independent layouts into masks suitable for the silicon foundry. We present here the various mecanisms that are used during the symbolic to target process translation along with informations that allows the technologist to parameterize the tool for a new process.

Résumé

Ce rapport a pour but d'expliquer la paramétrisation de l'outil de transformation de la représentation des masques symboliques en masques réels de la chaîne de CAO Alliance. L'investissement en temps pour le dessin de la vue physique des librairies de cellules pour les VLSI est considérable, et un tel investissement ne peut s'envisager que si le résultat est indépendant d'un procédé de fabrication donné. Alliance met en avant l'utilisation d'une nouvelle approche de dessin symbolique, il faut donc aussi se doter d'un outil automatique qui permette de fournir des masques idoines pour une manufacture donnée. Nous présentons ici les divers mécanismes mis en œuvre pendant l'opération de transformation, ainsi que les informations nécessaires au technologue pour qu'il paramètre l'outil pour de nouveaux procédés de fabrication.

The Symbolic To Real Mask Translation Approach in the Alliance CAD System

Frédéric Pétrot and Franck Wajsbürt

Équipe CAO-VLSI du laboratoire MASI

Tour 55-65, 201

Université Pierre et Marie Curie

4, Place Jussieu, 75252 Paris cedex 5

1 Introduction

Designing ASICs with the **Alliance CAD** system ensures the access to a large number of components: standard cell libraries for usual place and route tools and dedicated datapath structures, custom and semi-custom (datapath dedicated) generators and soon macrocells.

The time investment for the design of more than 600 leaf cells is very important, and therefore the layout can simply not be dedicated to a single fabrication process. Since module generation is the most time consuming task, a simple and efficient approach to the pitch matching problem through technological retargetting must be enforced.

The goal of this paper is to describe the method that produces a process dedicated layout that does not violate foundry rules from a validated symbolic layout. It also explains how to find the values needed to parameterize the translation tool `s2r` using the symbolic design rules and the process design rules.

We use a symbolic method based upon the statistical analysis of a group of processes. Careful examination of over twenty different processes ranging from 2 μm to 0.5 μm has shown that while minimum widths and spacings are quite variable throughout the different samples, the pitches — axis to axis distances — vary more homogeneously. From these technologies and the fact that the design is mostly done using minimum width wires, a generic set of process independent symbolic design rules is defined. These rules are expressed using a unit called the λ. For a given process, the value of the λ unit is close to the minimum length of the transistor gate.

This method can be though of as a generalization of the λ as described in [**?**]. It is nevertheless quite different:

- the design is done using structured objects called *symbols*, with strong associated semantic. It is not possible to draw a transistor with polysilicon over diffusion, for example. The symbols are either defined by a single point, like contact primitives, connectors, and references, or by two points, like segments and transistors. They are placed on the nodes of an isotropic grid with a spacing of 1 λ in both directions. The layout is drawn using a simplified set of layers and symbolic rules expressed with small integers.

- the symbolic to real translation process allows to independently adjust the spacing between the objects and the size of the objects. For example, the spacing between two wires depends upon the value of the λ, while the minimum width of each wire is defined by a technological parameter different for each process layer.

- each symbol is macro-generated using technological constraints. For example, the translation of a piece of diffusion will be very different if it is a wire, a transistor drain, or part of a contact primitive.

Three different grids are useful to explain the translations.

- the previously seen symbolic grid, with a step of 1 λ in both directions.
- the 'scaled symbolic grid', the homothetic image of the symbolic grid, with a $n \times \mu m$ step in both directions, where $n \times \mu m$ is the value of the λ for the given process. This value is computed from the set of symbolic and micron rules.
- the physical grid step, the minimum resolution of the fabrication line. It is one order of magnitude smaller than the λ. It is a technological constraint of the fabrication process.

Symbolic layout is translated into process dependent layout with the following steps:

- shrink the size of the symbolic grid to the λ value chosen for the process to obtain the 'scaled symbolic grid'. Since the grid changes homothetically, the relative positions of the centers of the symbols are kept.
- translate the symbolic objects in a collection of rectangles.
- place the instances, as this is a hierarchy preserving process.
- post process to erase notches and merge implant areas.

If the coordinates of a rectangle are not aligned on a physical grid step, then their width and height are bloated up so to snap to the physical grid.

Since the 'center' of a rectangle is snapped to a physical grid step, the extension on each side of the 'center' must be an integer number of physical grid steps. This implies that the minimum real width of a layer cannot be an odd number of physical grid steps. If such a situation occurs, the value of the width will be rounded up to the nearest greater integer number of physical grid steps. A half physical grid step translation of the center of all the rectangles is not possible because the translation is hierarchical.

This paper is organized as follows: Section 3 assumes that all the translation tool parameters are known and explains how the translation is done. This Section is of general interest and applies for both **CMOS** and **GaAs** layouts. Section 4 assumes that the translation formulae are known and explains how to find the parameterization values for the translation tool. This Section is valid for **CMOS** layout only.

2 Notations

We will use the following notations throughout the paper. The result of the transformation of a symbolic layout object is one or several user defined rectangles. A rectangle is made of:

1. a couple of coordinates, the lower left corner of the rectangle. They are called x_r, y_r.
2. dx_r, always positive, is the size of the rectangle in the x direction.
3. dy_r, always positive, is the size of the rectangle in the y direction.
4. a physical layer.

Index r indicates a 'real' value in μm and an index s indicates a 'symbolic' value in λ.
p is a pitch distance, d is an edge to edge distance, and W^i s the width of the fabrication layer i. $O^{j/i}$ indicates an overlap rule of layer j over layer i, and $D^{j\ i}$ indicate a spacing rule between layer i and layer j. The min index indicates that the minimum width rule for the considered layer is used, either for a real rectangle or a symbolic segment.

Values that belong to the technology file are in upper case letters, and variables are in lower case letters.

3 Transformation formulae

This defines the formulae used for the translation of each symbolic object.

3.1 Segment transformation

A symbolic segment is defined by:

1. a couple of coordinates, the segment starting point. They are called x_s and y_s.
2. a direction, `UP`, `DOWN`, `RIGHT`, `LEFT`, called $DIRECTION$.
3. the symbolic length, l_s, always positive.
4. the symbolic width, w_s, always positive.
5. a layer.

A symbolic segment starts on (x_s, y_s), has a length l_s, in the direction $DIRECTION$, and a width of w_s. The x_s, y_s, l_s items define the axis of the segment. The rectangle has not to be centered on the axis.

To have a simpler image of the translation process, two distances are introduced, l_r and w_r, that are respectively the size of the rectangle in the axis direction, and the size of the rectangle perpendicular to the segment axis. These values are useful because they do not depend upon the segments $DIRECTION$. One will become dx_r, the other one dy_r depending on $DIRECTION$.

Each physical rectangle comes from a symbolic segment by applying a specific formula on the segment items. Two types of rectangles are necessary for the symbolic to real translation of a segment for the **CMOS** and **GaAs** target technologies we aim at.

variable width rectangles that are called `VW` in the technology file. an `VW` rectangle is used when a real rectangle has a width that depends upon the symbolic width of the segment it comes from. This is by far the most common case.

constant width rectangles. These are needed when a rectangle has a constant size regarding the segment symbolic width. Practically, this formula is needed only to compute diffusion and active areas of transistors on each side of the transistor grid. To express rectangles on both sides of the symbolic segment, two types of constant width segments are needed: the ones to be drawn at the right hand side of the symbolic segments considered as a vector, and the ones at its left hand side. They are respectively called `RCW` and `LCW`.

Note that in both cases the coordinates of the rectangle are functions of the symbolic width.

The first formula for variable width rectangles needs three user defined parameters:

DLR

physical length extension. The length of the real rectangle in the axis direction is the symbolic length times the λ plus twice DLR.

$$l_r = l_s\lambda + 2DLR$$

DWR

physical width oversize. The real width is proportional to the symbolic width times the λ, plus an algebraic user defined constant, DWR.

$$w_r = w_s\lambda + DWR$$

This is a rewritten version of equation 1, where $DWR = W^i_{r\ min} - W^i_{s\ min}\lambda$.

$OFFSET$

physical offset, needed to shift the center of the resulting rectangle orthogonally to the segment axis. Generaly, when rectangles are centered on the segment axis, this value is zero.

The second formula for constant width rectangles also needs three user defined parameters.

DLR

physical length extension. The length of the real rectangle in the axis direction is the symbolic length times the λ plus twice DLR.

$$l_r = l_s\lambda + 2DLR$$

WR

physical width. This parameters is independent of the symbolic width of the segment.

$OFFSET$

physical offset, needed to shift the center of the resulting rectangle orthogonally to the segment axis.

The translation formulae based upon the use of these parameters are detailed in the following tables. A visual explanation of the formulae is given in the figures with each table. We are considering the symbolic to real translation of one segment with the $DIRECTION$ `UP`, and $w_s = 1.5\lambda$, the user parameters being $DWR = 1.5, DLR = 1.5, OFFSET = 2$ in all the cases. We assume the λ — noted as 'l' on the drawing — is 2 μm.

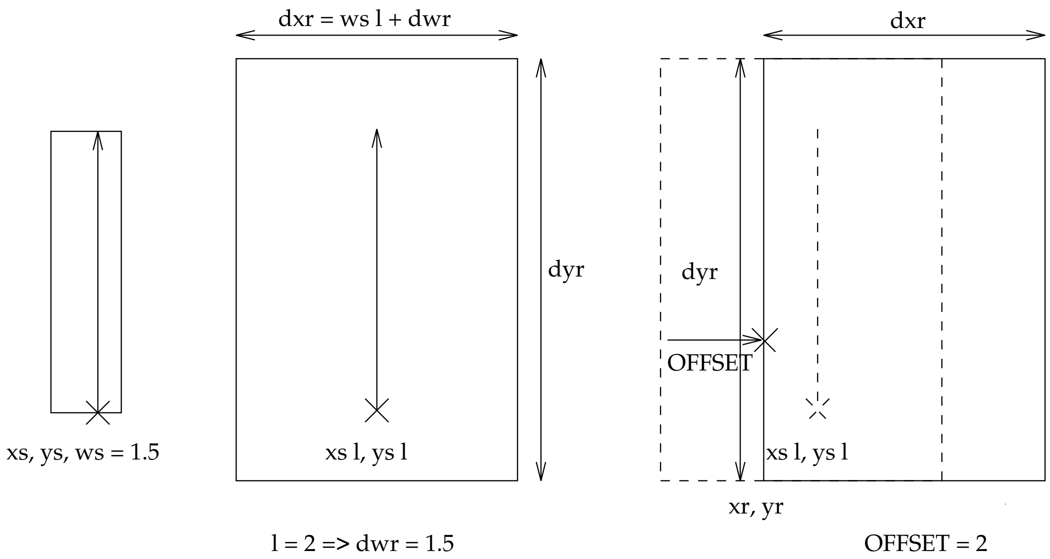
VW

$$
\begin{array}{llcl}
\texttt{UP} & x_r & = & x_s\lambda - (w_s/2)\lambda - DWR/2 + OFFSET \\
 & y_r & = & y_s\lambda - DLR \\
 & dx_r & = & w_s\lambda + DWR \\
 & dy_r & = & l_s\lambda + 2DLR
\end{array}
$$

$$
\begin{array}{llcl}
\texttt{DOWN} & x_r & = & x_s\lambda - (w_s/2)\lambda - DWR/2 - OFFSET \\
 & y_r & = & y_s\lambda - l_s\lambda - DLR \\
 & dx_r & = & w_s\lambda + DWR \\
 & dy_r & = & l_s\lambda + 2DLR
\end{array}
$$

$$
\begin{array}{llcl}
\texttt{RIGHT} & x_r & = & x_s\lambda - DLR \\
 & y_r & = & y_s\lambda - (w_s/2)\lambda - DWR/2 - OFFSET \\
 & dx_r & = & l_s\lambda + 2DLR \\
 & dy_r & = & w_s\lambda + DWR
\end{array}
$$

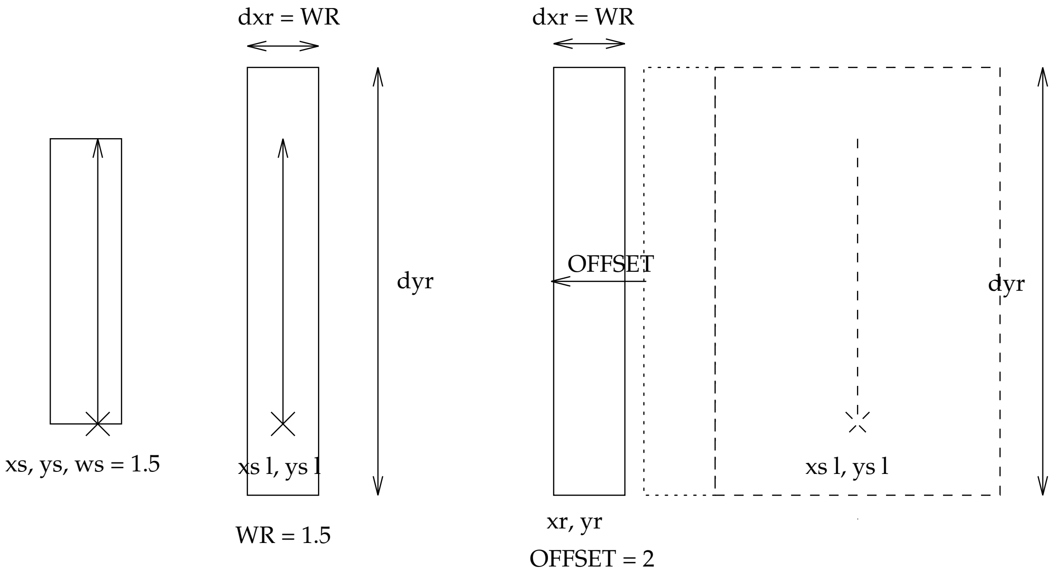
$$
\begin{array}{llcl}
\texttt{LEFT} & x_r & = & x_s\lambda - l_s\lambda - DLR \\
 & y_r & = & y_s\lambda - (w_s/2)\lambda - DWR/2 + OFFSET \\
 & dx_r & = & l_s\lambda + 2DLR \\
 & dy_r & = & w_s\lambda + DWR
\end{array}
$$



For a symbolic to target process layout translation, it’s highly probable that only this type of rectangles will be needed.

LCW

$$
\begin{array}{llcl}
\texttt{UP} & x_r & = & x_s\lambda - (w_s/2)\lambda - WR - OFFSET \\
 & y_r & = & y_s\lambda - DLR \\
 & dx_r & = & WR \\
 & dy_r & = & l_s\lambda + 2DLR \\
\\
\texttt{DOWN} & x_r & = & x_s\lambda + (w_s/2)\lambda + OFFSET \\
 & y_r & = & y_s\lambda - l_s\lambda - DLR \\
 & dx_r & = & WR \\
 & dy_r & = & l_s\lambda + 2DLR \\
\\
\texttt{RIGHT} & x_r & = & x_s\lambda - DLR \\
 & y_r & = & y_s\lambda + (w_s/2)\lambda + OFFSET \\
 & dx_r & = & l_s\lambda + 2DLR \\
 & dy_r & = & WR \\
\\
\texttt{LEFT} & x_r & = & x_s\lambda - l_s\lambda - DLR \\
 & y_r & = & y_s\lambda - (w_s/2)\lambda - WR - OFFSET \\
 & dx_r & = & l_s\lambda + 2DLR \\
 & dy_r & = & WR
\end{array}
$$



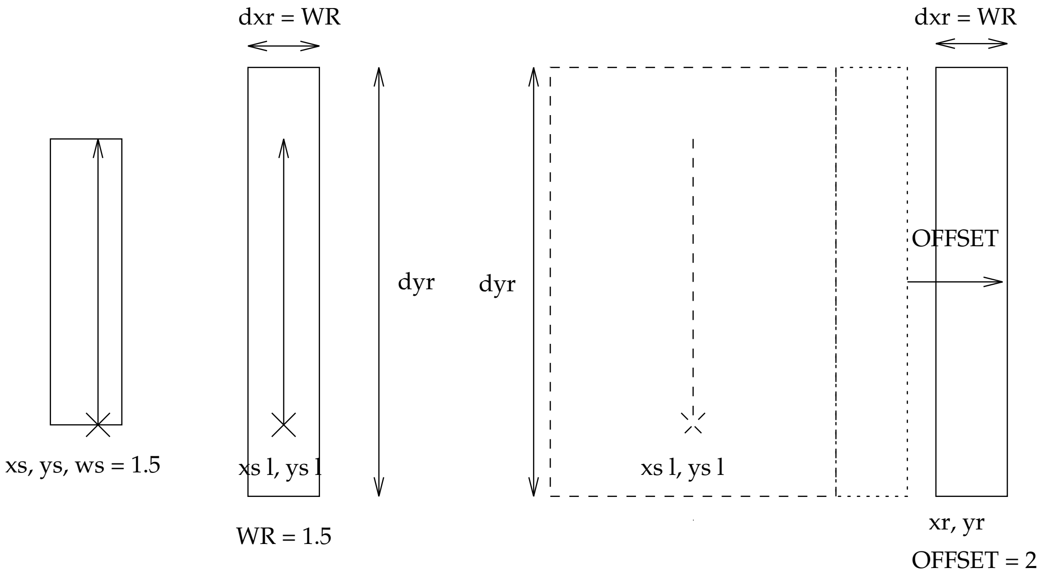
RCW

$$
\begin{array}{llcl}
\texttt{UP} & x_r & = & x_s\lambda + (w_s/2)\lambda + OFFSET \\
 & y_r & = & y_s\lambda - DLR \\
 & dx_r & = & WR \\
 & dy_r & = & l_s\lambda + 2DLR
\end{array}
$$

$$
\begin{array}{llcl}
\texttt{DOWN} & x_r & = & x_s\lambda - (w_s/2)\lambda - WR - OFFSET \\
 & y_r & = & y_s\lambda - l_s\lambda - DLR \\
 & dx_r & = & WR \\
 & dy_r & = & l_s\lambda + 2DLR
\end{array}
$$

$$
\begin{array}{llcl}
\texttt{RIGHT} & x_r & = & x_s\lambda - DLR \\
 & y_r & = & y_s\lambda - (w_s/2)\lambda - WR - OFFSET \\
 & dx_r & = & l_s\lambda + 2DLR \\
 & dy_r & = & WR
\end{array}
$$

$$
\begin{array}{llcl}
\texttt{LEFT} & x_r & = & x_s\lambda - l_s\lambda - DLR \\
 & y_r & = & y_s\lambda + (w_s/2)\lambda + OFFSET \\
 & dx_r & = & l_s\lambda + 2DLR \\
 & dy_r & = & WR
\end{array}
$$



This type of rectangle is required for symbolic extraction purposes, as the drain and source diffusions must not be connected. It is more or less reserved to symbolic **CMOS** 'pseudo-technologies' and **GaAs** processes.

These transformations can be seen as taking place in two steps, the first one generates a rectangle with a given size, and the second one makes a translation of the lower left corner of the rectangle from a given $OFFSET$.

3.2 Contact transformation

A symbolic contact is defined by:

- a couple of coordinates which are the center of the contact, called x_s and y_s.
- a type.

The transformation of a contact generates generally 2 to 5 physical rectangles. A size, $SIZE$, and a layer are needed for each of these rectangles. Their coordinates are computed as follows:

$$\begin{aligned} x_r &= x_s - SIZE/2 \\ y_r &= y_s - SIZE/2 \\ dx_r &= SIZE \\ dy_r &= SIZE \end{aligned}$$

Only square contacts, or related macros such as corners for `L` and `S` shaped transistors, are allowed with this approach.

3.3 Connector transformation

A symbolic connector is defined by:

1. a couple of coordinates, called x_s and y_s.
2. an orientation, `NORTH`, `SOUTH`, `EAST`, `WEST`, called $ORIENTATION$.
3. a symbolic width, w_s, that is always positive.
4. a layer.

A symbolic connector produces a single rectangle. Three parameters are needed for this transformation:

- a physical layer.
- the physical width oversize, called DWR.
- the physical extension on each side of the abutment box, DER.

The formulae that apply to the connectors are, regarding the $ORIENTATION$ value:

$$
\begin{array}{llcl}
\texttt{NORTH} & x_r & = & x_s\lambda - (w_s\lambda + DWR)/2 \\
 & y_r & = & y_s\lambda - DER \\
 & dx_r & = & w_s\lambda + DWR \\
 & dy_r & = & 2DER \\
\texttt{SOUTH} & x_r & = & x_s\lambda - (w_s\lambda + DWR)/2 \\
 & y_r & = & y_s\lambda - DER \\
 & dx_r & = & w_s\lambda + DWR \\
 & dy_r & = & 2DER \\
\texttt{EAST} & x_r & = & x_s\lambda - DER \\
 & y_r & = & y_s\lambda - (w_s\lambda + DWR)/2 \\
 & dx_r & = & 2DER \\
 & dy_r & = & w_s\lambda + DWR \\
\texttt{WEST} & x_r & = & x_s\lambda - DER \\
 & y_r & = & y_s\lambda - (w_s\lambda + DWR)/2 \\
 & dx_r & = & 2DER \\
 & dy_r & = & w_s\lambda + DWR
\end{array}
$$

The connector transformation formulae are derived from the segment transformation formulae since a connector is almost always placed at a segment extremity.

3.4 Instance transformation

A symbolic instance is defined by:

1. a couple of coordinates, the lower left corner of the instance. This point is (x_s, y_s).
2. a geometrical operation. Eight operations are possible and operation compositions give a result in the same set of operations. Operations are done 'in place', ie (x_s, y_s) indicates the lower left corner of the instanciated figure *after* the operation has been applied.
3. an instance name.
4. a model name.

The transformation of a symbolic instance into a real instance does not need any user input but the value of the λ. The coordinates in the micron world are the scaled equivalent of the symbolic ones, with a possible offset due to the different ways to represent symbolic geometrical operations, where they are done 'in place', and in the **cif** or **gdsII** format, where they are done regarding the origin, point $(0, 0)$, of the instanciated figure. This semantic change is due to the non-existence of the abutment box concept in micron layout. Let (x_1, y_1) be the lower left corner of the abutment box of the model of the instance, and (x_2, y_2) its upper right corner. x_r and y_r are the coordinates of instanciation in micron.

$$
\begin{array}{lrcl}
\texttt{NOSYM} & x_r & = & (x_s - x_1)\lambda \\
 & y_r & = & (y_s - y_1)\lambda \\
\texttt{SYM\_X} & x_r & = & (x_s + x_2)\lambda \\
 & y_r & = & (y_s - y_1)\lambda \\
\texttt{SYM\_Y} & x_r & = & (x_s - x_1)\lambda \\
 & y_r & = & (y_s + y_2)\lambda \\
\texttt{SYMXY} & x_r & = & (x_s + x_2)\lambda \\
 & y_r & = & (y_s + y_2)\lambda \\
\texttt{ROT\_P} & x_r & = & (x_s + y_2)\lambda \\
 & y_r & = & (y_s - x_1)\lambda \\
\texttt{ROT\_M} & x_r & = & (x_s - y_1)\lambda \\
 & y_r & = & (y_s + x_2)\lambda \\
\texttt{SY\_RP} & x_r & = & (x_s - y_1)\lambda \\
 & y_r & = & (y_s - x_1)\lambda \\
\texttt{SY\_RM} & x_r & = & (x_s + y_2)\lambda \\
 & y_r & = & (y_s + x_2)\lambda
\end{array}
$$

4 How are computed the values for a given technology?

General rules can be applied to choose the right values of the formulae parameters for a given process. A program, called `prol`, is available to produce the technology file for the symbolic to real translation tool using the physical grid step, the minimum widths, spacings and overlaps of the target process. The equations used in this program are described here.

4.1 Computation of the λ value

The first and most important work is to determine the value of the λ, as it will define the size and overall design rule correctness of the circuit.

The λ depends on the values of the process design rules. As stated in Section 1, the important distances in our approach are the pitches, so the value of the λ is calculated regarding centerline to centerline distances between two minimal adjacent segments. The easiest way to compute this value is to use a table with symbolic design rules and target design rules as entries and compute the ratio of 'target' over 'symbolic' for each pitch rule. The maximum value resulting from these computations gives the value of the λ. Note that all the values resulting from a symbolic to real translation are multiples of the 'physical grid' value — the resolution of the fabrication line —, and so does the λ. So each entry has to be rounded towards the nearest multiple of the physical grid.

The pitch values are for minimum width runs, as we assume that with the symbolic to micron translation formula we use, if some pitch distance rule is true for minimum width segments, then the resulting edge to edge distance rule is also true for non minimal segments.

The formula for segment transformation can be basically expressed as

$$W_r^i = (W_s^i - W_{s\ min}^i)\lambda + W_{r\ min}^i \tag{1}$$

For minimum symbolic width wires, it can be rewritten as $W_r = W_{r\ min}$ where it can be noticed that the value of the λ is not used.

The λ will be determined from the ratio of a real pitch distance over a symbolic pitch distance, $\lambda = p_r/p_s$. The above expression can be rewritten, using edge to edge distance and minimum width:

$$\lambda = \frac{p_r}{p_s} = \frac{d_r + W_{r\ min}^i/2 + W_{r\ min}^j/2}{d_s + W_{s\ min}^i/2 + W_{s\ min}^j/2} \tag{2}$$

Since this should also be true for all segments, we express the ratio of equation 2 using the width formula for non minimal segments of equation 1

$$\begin{aligned}
\lambda &= \frac{d_r + W_r^i/2 + W_r^j/2}{d_s + W_s^i/2 + W_s^j/2} \\
&= \frac{d_r + (W_s^i - W_{s\ min}^i)\lambda/2 + W_{r\ min}^i/2 + (W_s^j - W_{s\ min}^j)\lambda/2 + W_{r\ min}^j/2}{d_s + W_s^i/2 + W_s^j/2}
\end{aligned}$$

Moving λ out of the expression yields to

$$\lambda d_s + \lambda W_s^i/2 + \lambda W_s^j/2 =$$
$$d_r + \lambda W_s^i/2 - \lambda W_{s\ min}^i/2 + W_{r\ min}^i/2 + \lambda W_s^j/2 - \lambda W_{s\ min}^j/2 + W_{r\ min}^j/2$$

Eliminating the redundant expression on each side of the equation gives

$$\lambda d_s = d_r - \lambda W_{s\ min}^i/2 + W_{r\ min}^i/2 - \lambda W_{s\ min}^j/2 + W_{r\ min}^j/2$$

Factorizing with λ leads to

$$\lambda(d_s + W_{s\ min}^i/2 + W_{s\ min}^j/2) = d_r + W_{r\ min}^i/2 + W_{r\ min}^j/2 \tag{3}$$

Expression 3 is strictly equivalent to equation 2, thus showing that the values found for minimum geometries are applicable in the general case.

In fact this is correct only if the $W_{r\ min}^i$ and the λ are even numbers of physical grid steps. If rounding is necessary, some specific technics explained in Section 4 solve the problem.

An example of the computation for the generic **prol12** process is given in the table below. The **edge** column displays the edge to edge symbolic design rules. The **pitch** column displays the symbolic pitch with minimum geometries. The **prol12** column contains the process edge to edge design rules. The p_r/p_s column display the pitch ratio formula. And finally, the λ column contains the resulting value rounded to the physical grid, .15 μm for this process.

rules	edge	pitch	prol12	p_r/p_s	λ
nwell	4		6		
nwell-nwell (cold)	12	12	6	$(6+6)/12 = 1$	1.05
dif	2		1.2		
dif-dif	3	5	2.4	$(1.2+2.4)/5 = 0.72$	0.75
(dif/cont)-dif	3	6	2.4	$(0.6+2.4+1.5)/6 = 0.75$	0.75
(dif/cont)-(dif/cont)	3	6	2.4	$(3+2.4)/6 = 0.9$	0.9
dif-nwell	7.5	11	3.6	$(3+3.6+0.6)/11 = 0.65$	0.75
nwell/dif	0.5		3.6		
difN-(difP/nwell)	9	10	7.2	$(1.2+7.2)/10 = 0.84$	0.9
poly	1		1.2		
Ngate	1		1.2		
Pgate	1		1.2		
poly-poly	2	3	1.8	$(1.2+1.8)/3 = 1$	1.05
(poly/cont)-poly	2	4	1.8	$(0.6+1.8+1.5)/4 = 0.97$	1.05
(poly/cont)-(poly/cont)	2	5	1.8	$(3+1.8)/5 = 0.96$	1.05
poly-dif	1	3	0.6	$(0.6+0.6+0.6)/3 = 0.6$	0.6
gate/dif	1.5		1.2		
dif/gate	2		1.8		
cont	1		1.5		
cont-cont	3	4	1.5	$(1.5+1.5)/4 = 0.75$	0.75
cont-gate	2	3	1.05	$(0.75+1.05+0.6)/3 = 0.8$	0.9
cont-dif	2	4	1.05	$(0.75+1.05+0.6)/4 = 0.6$	0.6
dif/cont	1		0.75		
poly/cont	1		0.75		
mt1	1		1.8		
mt1-mt1	2.5	4	1.8	$(1.8+1.8)/4 = 0.9$	0.9
(mt1/cont)-mt1	2.5	4	1.8	$(0.9+1.8+1.5)/4 = 1.05$	1.05
(mt1/cont)-(mt1/cont)	2.5	5	1.8	$(3+1.8)/5 = 0.96$	1.05
(mt1/via)-mt1	2.5	4	1.8	$(0.9+1.8+1.5)/4 = 1.05$	1.05
(mt1/via)-(mt1/via)	2.5	5	1.8	$(3+1.8)/5 = 0.96$	1.05
mt1/cont	0.5		0.75		
mt1/via	0.5		0.75		
via	1		1.5		
via-via	3	4	1.5	$(1.5+1.5)/4 = 0.75$	0.75
via-poly	2	3	1.65	$(0.75+1.65+0.6)/3 = 1$	1.05
via-(poly/cont)	2	4	1.65	$(0.75+1.65+1.5)/4 = .975$	1.05
via-gate	2	3	1.65	$(0.75+1.65+0.6)/3 = 1$	1.05
via-cont	2	3	1.5	$(1.5+1.5)/3 = 1$	1.05
mt2	2		1.8		
mt2-mt2	2	4	1.8	$(1.8+1.8)/4 = 0.9$	0.9
(mt2/via)-mt2	2	5	1.8	$(0.9+1.8+1.5)/5 = 0.841$	0.9
(mt2/via)-(mt2/via)	2	5	1.8	$(3+1.8)/5 = 0.96$	1.05
mt2/via	1		0.75		

The maximum value of the last column is 1.05, so the value of the λ for this process is 1.05. Note that this value, or a very close value, should appear for the metals rules and less important the polysilicon rules, as this approach implicitly states that the size of a circuit is primary due to the metals-metals pitch rules. If it's not the case, our fixed grid approach is not well suited for the target process.

4.2 Contacts

The parameters for the contacts involving a cut layer are easily calculated. First compute the size of the cut, where index i is a cut layer.

$$SIZE^i = W^i_{r\ min}$$

Then, for the other layer involved in a contact, use the minimum real overlap of each layer, j, over the cut, i.

$$SIZE^j = O^{j/i}_{r\ min}$$

The $SIZE^i$ of the transistor corners — to draw `L` and `S` shaped transistors — are fixed by the rules used for a minimum width — electrical channel length — transistor.

4.3 Segments

A symbolic segment is a unique object, but when used for symbolic to real translation, we have to consider four types of segments:

- the runs, metals and polysilicon lines.
- the transistors and diffusions.
- the surrounding segments, the wells.

The computations must be done in that order, because each step puts constraints on the following step. We will only focus on the `VW` rectangles, as they are the only ones useful for **CMOS** processes.

Concerning the λ, two cases must be distinguished. If the λ is an even number of physical grid steps, half λ extension on each side of a segment gives an integer number of physical grid steps. If the λ is an odd number of physical grid steps, the rounding operation may create a design rule violation. When translating a non minimal segment having a width that is an odd number of λs of the most constraining rules, an edge to edge spacing error can occur because the rectangle coordinates are rounded by excess to a physical grid step, as explained Section 1. Solving this problem is done by substracting a physical grid step to the width of all the rectangles when the λ is an odd number of physical grid steps. If the number of λs of a segment is even, the number of physical grid steps on each side of a segment is always integer, and the substracted half physical grid step on each side of the segment will be provided by the rounding function. If the λ is an odd number of physical grid steps, the substracted half physical grid step on each side of the segment will give a integer number of physical grid steps without rounding. This — always non minimal — run will have a width a physical grid step smaller than expected.

Runs

Run parameters are computed using the minimum symbolic and real width of the runs.

$$
\begin{aligned}
DLR^i &= \frac{1}{2} W^i_{r\ min} \\
DWR^i &= W^i_{r\ min} - W^i_{s\ min} \lambda
\end{aligned}
$$

DWR^i may have a negative value. In that case, two abutted colinear symbolic segments may not be connected anymore after translation. The post processing described in Section 4.5 merges them back.

These rules are correct for the symbolic layers used in wires that have a direct equivalent in the process, namely `POLY`, `ALU1`, `ALU2`, and `ALU3`.

diffusions

DWR^i is computed as follows for the active area in runs — `NDIF`, `PDIF`, `NTIE` and `PTIE`—. The choice is to have a 3 λ run that is as large as a symbolic diffusion contact.

$$DWR^{act} = W^{con}_{r\ min} + 2\, O^{act/con}_{r\ min} - \lambda$$

DLR^{act} is based on the constraints of the transistor.

Transistors

Please note that when speaking of a transistor, the term length refers to it electrical channel length. The same applies to its width. Nevertheless, when the minimum width of a layer is required, we refer to the technological constraint.

DLR^{gat} is fixed to 0, because the computation of the λ value makes this hypothesis. One can see on layout (1) of Figure 1 that DLR must satisfy

$$-\frac{1}{2}W^{pol}_{r\ min} \leq DLR^{gat\ in\ tr} \leq \frac{1}{2}W^{pol}_{r\ min}$$

DWR^{gat} is the same as in the runs:

$$DWR^{i} = W^{i}_{r\ min} - W^{i}_{s\ min}\lambda$$

This is required in order to use the minimum gate length of the process.

$DLR^{act\ in\ tr}$ has to be computed from five constraints. It is always a negative value.

- the ratio W_P/W_N must be the same than in symbolic if possible, so $W_r = n\lambda + 2DLR^{act\ in\ tr}$ must be equal to $W_s = (n-3)\lambda$.

  $$n\lambda - 2|DLR^{act\ in\ tr}| = (n-3)\lambda$$

  leads to $DLR^{act\ in\ tr} = \frac{3}{2}\lambda$.

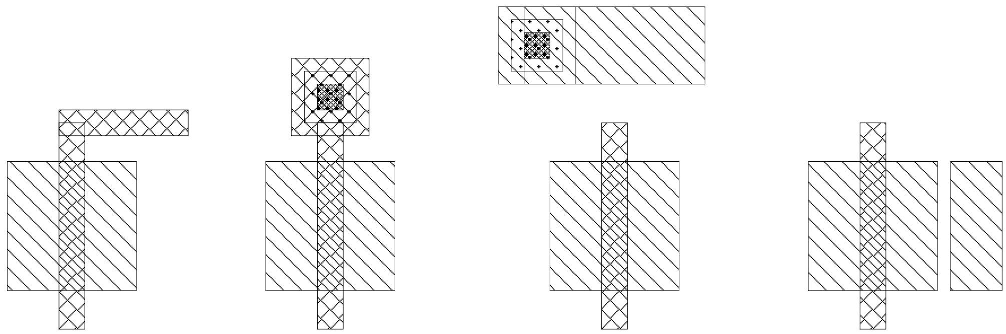
- the spacing between polysilicon and active must be respected. This spacing is illustrated in layout (1) of Figure 1.

  $$|DLR^{act\ in\ tr}| \geq \frac{1}{2}W^{pol}_{r} + D^{pol\ act}_{r\ min}$$

- the spacing between contact of poly and active must be respected. This spacing is shown in layout (2) of Figure 1.

  $$|DLR^{act\ in\ tr}| \geq \frac{1}{2}W^{con}_{r\ min} + O^{pol/con}_{r\ min} + D^{pol\ act}_{r\ min} - \lambda$$

Finally, $|DLR^{act\ in\ tr}|$ is the maximum of these values.



(1) (2) (3) (4)

Figure 1: Configurations for the transistor parameters choice.

Two other spacing distances between active in a transistor and active in symbolic contacts and segments must be respected, as shown on layout (3) of Figure 1. Since it is not possible to express these distances using $DLR^{act\ in\ tr}$, this will force the choice of DLR^{act} in wires. If we can prove the rules for the active wires, then we implicitly prove them for the active into transistors. The symbolic transistor has a length $W_s = (n-3)\lambda$, so we express DLR^{act} as

$$DLR^{act} = 2\lambda + DLR^{act\ in\ tr}$$

This ensures that an active wire with a length of $n \times \lambda$ is as long as the active area of a transistor with a width of $(n+4) \times \lambda$, as on layout (4), Figure 1. (This behavior is the same as the one of symbolic rules.) To verify this constraint, we must check that DLR^{act} is smaller than $\frac{1}{2}W^{act}_{r\ min}$, the value that is used when computing the λ.

The overlap of the active area around the transistor gate must be bigger than its minimum,

$$DWR_{act\ in\ tr} \geq W^{gat}_{r\ min} + 2\,O^{act/gat}_{r\ min}$$

but may be as big as its symbolic value, which is 5 λ.

The implant area and well sizes can be directly calculated. The minimum overlap rule around the active area is used. This gives:

$$\begin{aligned} DLR^{j} &= DLR^{i} + O^{j/i}_{r\ min} \\ DWR^{j} &= DWR^{i}\lambda + O^{j/i}_{r\ min} \end{aligned}$$

where j is the implant or well layer, and i is the active layer, either in a transistor or in a wire.

Wells

The determination of well parameters is a little complicated. This is because it is a segment that surrounds other segments. It is based on the exhaustive analysis of the different objects it may contain. The most constraining configurations will determine the extensions.

Figures 2 and 3 show the possible symbolic configurations for a technology where the well has been automatically generated with the minimum technological value.

First, DLR.

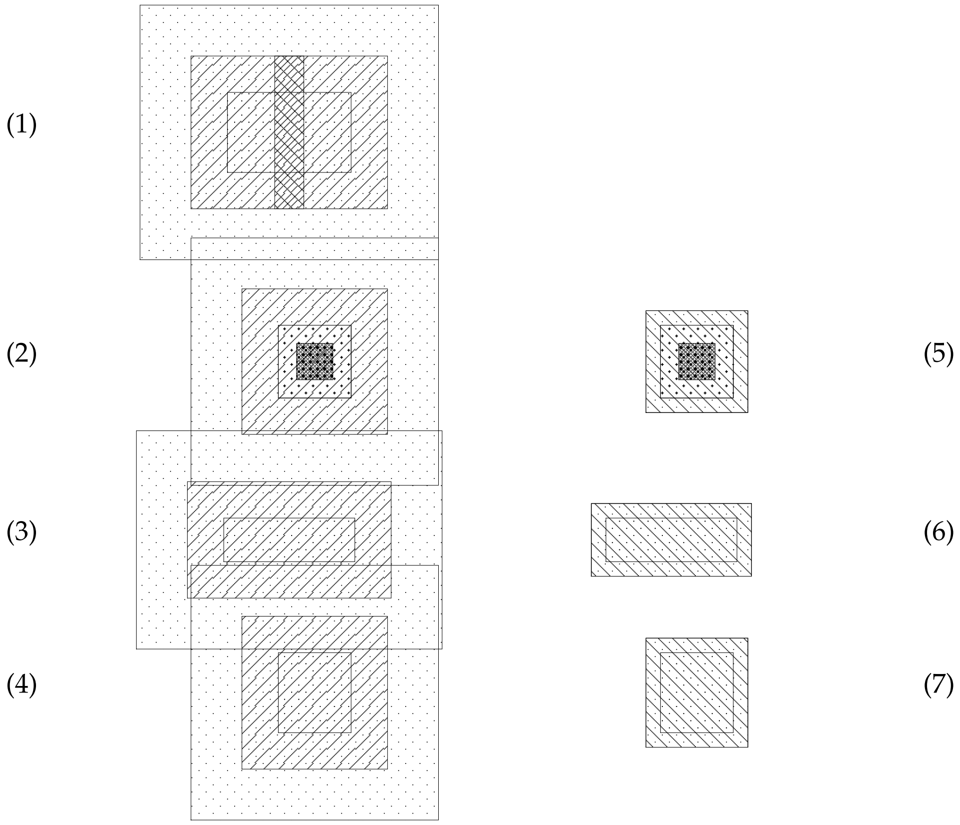


Figure 2: Configurations for well DLR choice. All the well segments are drawn vertically.

(1) transistor is orthogonal to well:

$$O_{r\ min}^{wel/imp} + O_{r\ min}^{imp/act} - \frac{1}{2}\lambda$$

(2) contact diffusion is into well:

$$\frac{1}{2}W_{r\ min}^{con} + O_{r\ min}^{acti/con} + O_{r\ min}^{wel/imp} + O_{r\ min}^{imp/act} - 2\lambda$$

(3) diffusion run is orthogonal to well:

$$\frac{1}{2}W^{act} + O_{r\ min}^{wel/imp} + O_{r\ min}^{imp/act} - \frac{3}{2}\lambda$$

(4) diffusion run or transistor is parallel to well:

$$O_{r\ min}^{wel/imp} + O_{r\ min}^{imp/act} - DLR^{act} - \lambda$$

Configurations (5), (6) and (7) are the equivalent of configurations (2), (3) and (4), when the active is used for well polarization purposes. DLR^{wel} is given by the maximum of these values. On Figure 2, configuration (3) is the most constraining one.

Then DWR.

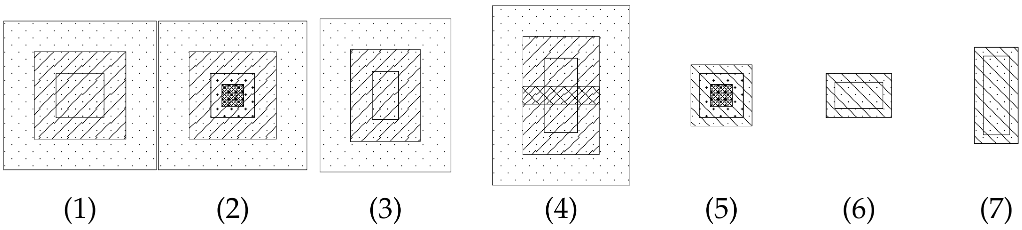


Figure 3: Configurations for well DWR choice. All the well segments are drawn horizontally.

(1) transistor or diffusion run is orthogonal to well:

$$2DLR^{act} + 2\,O_{r\ min}^{imp/act} + 2\,O_{r\ min}^{wel/imp} - 2\lambda$$

(2) contact diffusion is into well:

$$W_{r\ min}^{con} + D_{r\ min}^{con\ act} + 2\,O_{r\ min}^{imp/act} + 2\,O_{r\ min}^{wel/imp} - 4\lambda$$

(3) diffusion run is parallel to well:

$$W_{r\ min}^{act} + 2\,O_{r\ min}^{imp/act} + 2\,O_{r\ min}^{wel/imp} - 3\lambda$$

(4) transistor parallel is to well:

$$2\,O^{imp/act}_{r\ min} + 2\,O^{wel/imp}_{r\ min} - \lambda$$

Configurations (5), (6) and (7) are the equivalent of configurations (2), (3) and (4), when the active is used for well polarization purposes. As for DLR^{wel}, DWR^{wel} is given by the maximum of these values. Configuration (4) is the most constraining on Figure 2,

4.4 Connectors

The two needed parameters are computed as follows:

$$\begin{aligned} DER^i &= \frac{1}{2} W^i_{r\ min} \\ DWR^i &= W^i_{r\ min} - W^i_{s\ min} \lambda \end{aligned}$$

Note that these values are the same as the segment's ones, and that connectors are supposed to end runs only. In case connectors are required for other type of segments, the segment's DWR should be used.

4.5 Post processing

A straight translation using the values described above leads to an incorrect layout. Abutted colinear symbolic segments may not be connected anymore after this translation, notches have been created, and implant areas need to be merged.

A hierarchical merging phase needs to be performed in order to obtain a correct layout. In order to erase the notches, the rectangles are bloated, merged, and then shrunk. Two rectangles that are closer than the minimum spacing must be merged. Therefore the oversize value, OSV is computed as

$$OSV^i = \frac{1}{2} D^{i\ i}_{r\ min} - PGS$$

where PGS is the physical grid step.

When merging, the tool may create rectangles. As these rectangles should no be smaller than the physical design rule, the minimum width of each layer must be provided. The minimum layer width, MLW, is

$$MLW^i = W^i_{r\ min}$$

Finally, since this is a hierarchical process, some informations must be available throughout the hierarchies. Only a ring, per layer, is necessary to correct the notches on cell boundaries. The width of this ring, BRW is given by

$$BRW^i = D^{i\ i}_{r\ min}$$

Note that this BRW^i is applied with respect to the abutment box of the cell currently post processed and the physical envelops of its neighboring cells.

5 Conclusion

The approach described here provides a valid technological file for all the technologies we have tried. If that were not the case, or if some configuration resulting of the translation does not 'looks' fine, it is a good idea to slightly modify the values in the parameterization file, run the translation tool, and check the layout.

We mainly tune the file to have a nice rectangular layout, with as little notches as possible.

6 Custom cells replacement

It is possible to substitute custom cells, designed for a given process, to symbolic cells into a symbolic layout at translation time. Two needs lead to custom cell insertions: necessity to use some foundry specific objects, such as process dedicated pads, or small changes in a traduced symbolic layout to correct known DRC violations on a highly used cell that may lead to great density improvement — a *RAM* memory point in a matrix is a good example —.

1. a symbolic phantom cell must be drawn so as to abut it correctly with its neighbors, from a size and connector location point of view. This cell is the one being used for the whole symbolic design.

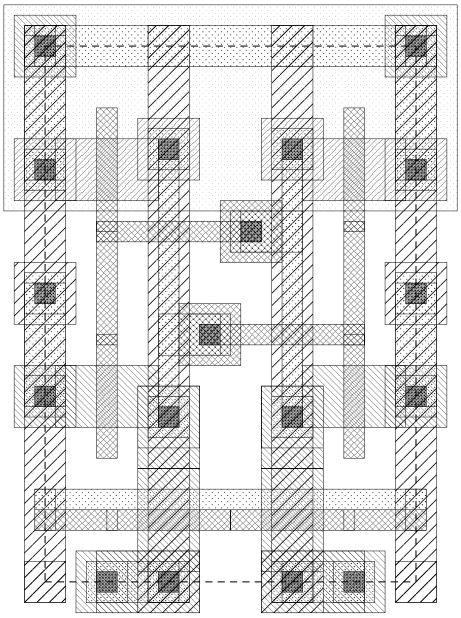


Figure 4: Symbolic RAM memory point.

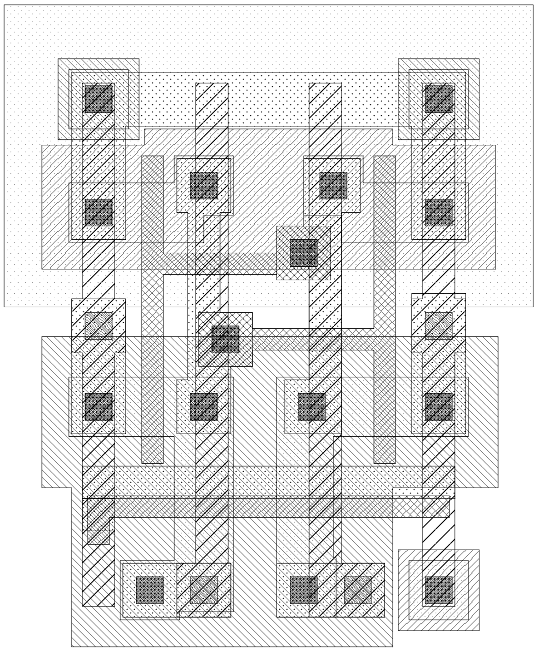


Figure 5: Custom RAM memory point.

Lets take the symbolic memory point of Figure 4 as example. This cells violates two symbolic design rules: metal to metal spacing in the center of the cell. Also there is no substrate polarization inside the cell. We know that this cell must abut with itself after a symmetry around the y axis.

2. since the symbolic coordinates are snapped to a 1 λ grid, the 'real' cell will probably need to be slightly modified to ensure that its connectors will match the ones of its phantom once transformed. COmputing connector locations can be done either manually with the formulae given Section 3.3 or by running the translation tool parameterized for the process on the phantom. In any case, it is a good idea to design the symbolic cell with the lower left corner of the abutment box in $(0, 0)$ in order for the coordinates to be directly, ie without translation, usable as micron coordinate. Nevertheless, this is not required.

   Since no abutment box is available in 'real', and since we use the abutment box of a model to compute the placement of its instances, the 'real' design must have its *virtual abutment box* lower left corner located in $(0, 0)$. In general, this *virtual abutment box* is the abutment box of the cell scaled by the value of the λ.

   For the memory point, we wanted to put a substrate polarization contact in the lower right corner. No space was available for that, but since we know the cell abutment properties, we know we can modify the east side of the cell without interacting on its symbolic neighbors. The resulting — DRC correct — cell is shown Figure 5.

3. add an entry in the catalog file to indicated the translation tool to substitute this cell: `phantomcell G`. The `G` stands for **gdsII** layout.
4. run the translation tool on the whole design.

Care must be taken to ensure that no notches nor that any DRC rule is violated when designing the custom cell, since no post processing will occur on either the model nor its instances.