Estimation of Wiring Area for Hierarchical Design

Bernd Schürmann
Gerhard Zimmermann

SFB-124
Report-No. 24/92

Fachbereich Informatik
Universität Kaiserslautern
Erwin-Schrödinger-Straße
D-6750 Kaiserslautern

Estimation of Wiring Area for Hierarchical Design

Abstract

The estimation of the area of cells before layout, especially of shape functions, is very important in a hierarchical design process and a necessity for top down design styles. The major problem is the estimation of wiring space. For the internal wiring between the subcells relative good models exist. Problems occur for the wiring space to connect the pins of a cell with its subcells. This paper gives a mathematical description of the wiring models for both cases and especially treats the pin connection problem. Our goal is to find the simplest model that explains the experiments and to introduce only a small number of model parameters. A set of values for these parameters is an important result which is supported by new measurements for standard cell blocks.

1. Introduction

The focus of interest in CAD for VLSI has shifted from individual algorithms to complete CAD systems. This shift has resulted from increasing demands on the complexity, predictability, and efficiency of the design process and on the quality of the products. Due to the complexity of the task the design process has been split into steps between levels of a part-of hierarchy and between description domains, roughly the behavioral, structural, and physical domain /GaK83/. These steps correspond to individual tools and algorithms within the tools. One requirement to combine the tools into a CAD system is the CAD framework. Because of the strong dependency between the individual design steps, there is a second requirement for a CAD system, which we will call the “glue”.

We can view the design of a digital system as a planning process with stepwise refinement of an initial specification /GiO84/. In each refinement step decisions are made on the basis of the state of the current design (as a result of design decisions of the past) and on the basis of a prediction of future design decisions. This prediction is the glue between past and future and thus “glues” all steps of the design process together. If the future design steps would be fully predictable, as for example the size of an unfolded PLA, the glue process would be a - not necessarily simple - calculation with known parameters. In the VLSI design process this is typically not the case. As long as there is free design space, future decisions are not fully predictable and have to be estimated, as for example the result of placement and the resulting wiring space.

Because of this, the glue plays an important role in a multi-step design process, more so in top-down than in bottom-up design styles. For example, high-level synthesis is currently hampered by very unreliable cost and performance abstractions as objective functions. The goal of our research is an improvement in all cases where glue is necessary. So far we have specialized on area estimations /Zim88/ and the work reported here is progress in this direction. Our next goal is performance estimation and has, in the case of CMOS, to rely heavily on precise area and wire length estimations.

Estimation cannot be precise in the sense that the final result and the estimation fully match. There will always be a tolerance. The more predictable the future design process is, the smaller the tolerance will be. Thus a reduction of prediction tolerances can only be achieved by making the design steps more predictable. In addition to the tolerance, estimation errors are possible. Our goal is the reduction of these errors. A second goal is the quantitative determination of estimation tolerances and errors. For the user of the glue, a designer or a tool, it is important to know how reliable the estimations are. The reliability can be either expressed by upper and lower bounds or by probabilistic measures. This is a new but important field of research where we can report only preliminary results.

Many different approaches have been published for area estimation. Reis was one of the first with an empirical method. Here we do not want to go into further details.

Experience with our shape function generator /Zim88/ has unveiled some interesting aspects. A comparison between predicted and real floorplan shapes led us to the belief that the predicted area is close to the lower bound of all the floorplans that can be generated /Zim90/. Just by changing the orientation of the first cut line of floorplans, changes in area of over 70% have been found. Other dependencies of this kind will be shown for standard cell blocks in this paper. We therefore have to be careful about what we want to predict.

Predictions can also be used in the sense of “what-if” tools. A designer may not want to complete a design, but evaluate different options. The knowledge in this case is again determined by the past decisions and by “default” assumptions about the future. Such assumptions also play an important role in glue applications of predictions and the default parameters we found are therefore, besides the estimation method, an important result.

2. The Area Estimation Model

2.1 Shape Functions

The goal of our method is the estimation of shape functions of rectangular layouts. Shape functions form the border between feasible and infeasible shapes. Figure 1 shows the shape function of a fixed macrocell A. The black dot (corner) defines the x and y dimensions of the rectangular cell A. All other points on the curve and in the hatched area are made feasible by adding empty space to A on any of the four sides or within A. It can be used for wiring.

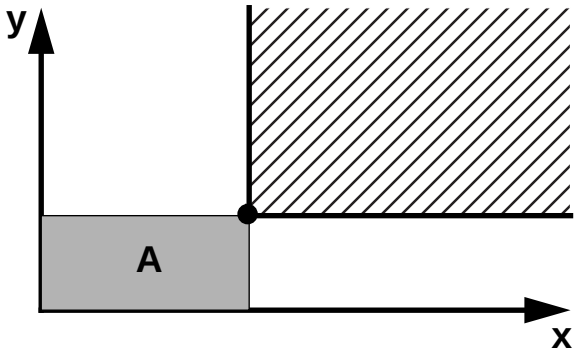

Figure 1: Macro shape function

IF A has different shapes, for example (A_1, ..., A_4) by controlling the layout of A with different parameters, then a multi-macro shape function as in figure 2 evolves. Per definition as a lower area bound, the shape function is monotonic and A_4 could be constructed by adding empty area to A_2 or A_3. The shape function in figure 2 is therefore fully defined by the corner coordinates A_1, A_2, A_3.

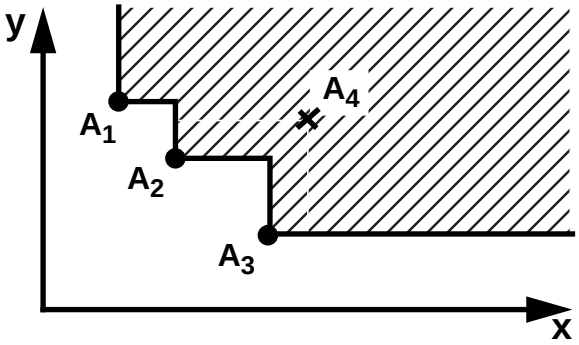

Figure 2: Multi-macro shape function

A_4 may have properties that neither A_2 or A_3 have. For example, A_4 may have all pins on one side, whereas A_2 and A_3 have pins on all four sides. Thus, in a specific floorplan, A_4 may, despite larger area, produce a smaller floorplan or a shorter critical delay path. Therefore, we do not always enforce monotony.

For flexible cells, only estimated shape functions can exist, if the layout style offers design decisions that influence the layout area. Thus each corner point has tolerances in x and y and the shape function has a tolerance band around the average curve as shown in figure 3. In the extreme the shape function can be a continuous curve, but due to the tolerances it can always be approximated by a suitable number of corners. Module generators can be either modeled by multi-macro shape functions if they generate fully predictable layouts or by flexible cell shape functions otherwise.

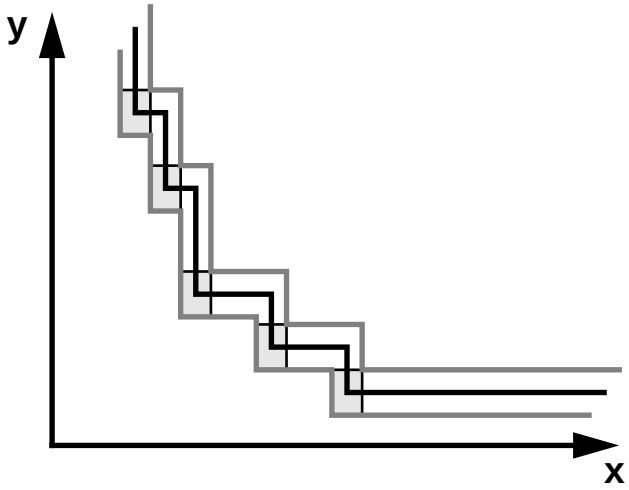

Figure 3: Flexible cell shape function

If we assume that all layout geometries for which we seek shape functions can be approximated by slicing topologies, these can be represented by slicing trees. If shape functions for the leaves of a tree are known and the orientations of all slices in the tree are determined, the shape functions can be easily added up the tree and thus for the root node which represents the cell of interest /Ott83/. Figure 4 shows this correspondence.

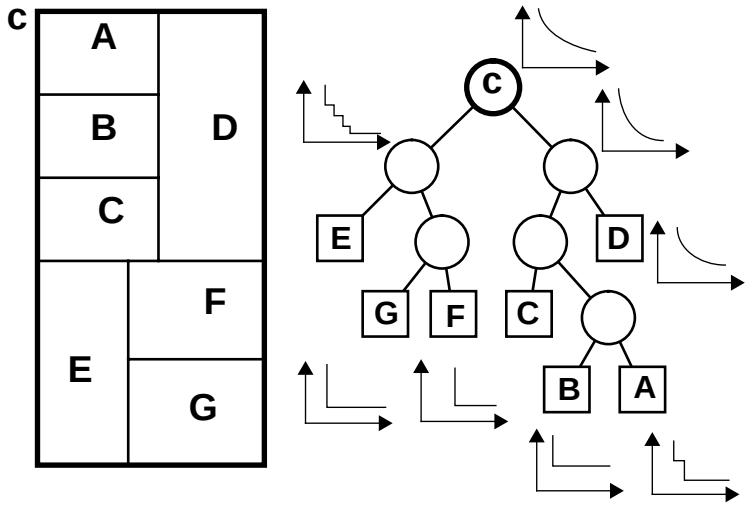

Figure 4: Slicing structure and tree

If the orientations of the slices are not known, optimal orientations can be determined by adding monotonic shape functions horizontally and vertically and selecting the corners with minimal area of both results. This results in a monotonic shape function at each internal node of the tree with the attribute vertical or horizontal at each corner.

The resulting shape function estimates the total area only if no additional wiring space is required. We therefore extended the model by trying to estimate this space /Zim88/. This led to a five color model.

2.2 The Five Color Model

Let us look at two levels of a multi-level part-of hierarchy. This involves three levels of cells, $l-1$, l, and $l+1$. The level of a cell c is denoted by a superscript and the cell instance by a subscript. The order of levels is such that c^{l+1} is composed of cells c_i^l which are composed of cells c_{ij}^{l-1} (figure 5). Let us further assume that we seek the shape function of c^l. Each cell c_i^l has pins at its perimeter which we denote $p_{i,k}^l$. Nets n_m are globally defined for all cell levels. A net can therefore exist at many levels. For the purpose of estimation we divide nets into segments that are fully contained in one level only. Such a segment of net n_m of cell c_i^l is denoted by $n_m c_i^l$ and is defined as a set of pins $\{p_{i,k}^l\}\{p_{ij,o}^{l-1}\}$

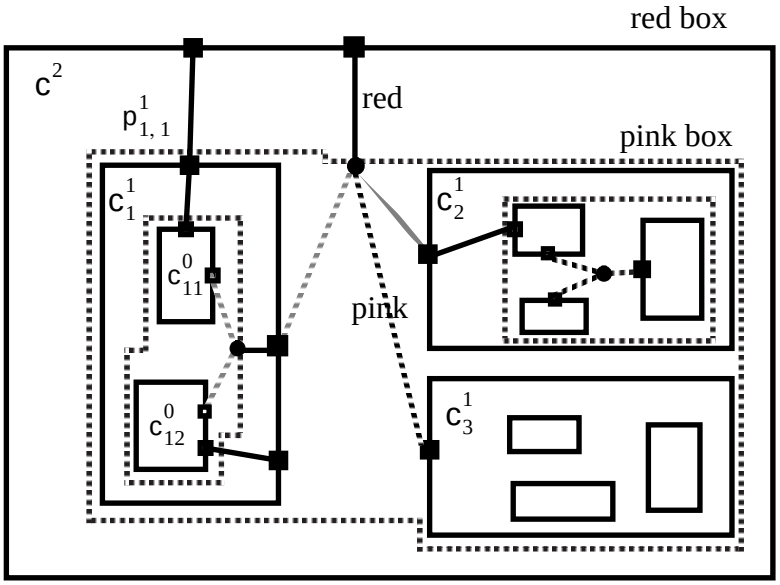

Figure 5: Two level hierarchy example

A segment can be further divided into a pink part $n_m^p c_i^l$ and a red part $n_m^r c_i^l$ (figure 5). The pink part connects only the pins of the subcells c_{ij}^{l-1} (dotted lines). Pink segments are internal connections of a cell and exist if more than one internal pin exists. Red segments connect pink segments or the only one internal pin of a net to an external pin (solid lines). Seen from cell c_i^l, the whole segment $n_m c^{l+1}$ of the supercell c^{l+1} can be seen as external to c_i^l. We assign the color green to external net segments.

With this notion of colors of nets we define wiring area $w^p c_i^l, w^r c_i^l$ and $w^g c_i^l$ for cell c_i^l. Accordingly we define enclosing rectangles for each color, called the pink, red, and green box

(figure 5). In addition we define a blue box for the case that all nets of the current level are disregarded.

Each box has an area, for example $a^p c_i^l$, and x and y dimensions $x^p c_i^l$, $y^p c_i^l$. In addition we have empty (black) area, denoted $a^e c_i^l$. If we assume red subcells c_{ij}^{l-1}, the areas of the four box types of cell c_i^l are:

$$a^b c_i^l = \sum_j a^r c_{ij}^{l-1} + a^e c_i^l \tag{1}$$

$$a^p c_i^l = a^b c_i^l + w^p c_i^l \tag{2}$$

$$a^r c_i^l = a^p c_i^l + w^r c_i^l \tag{3}$$

$$a^g c_i^l = a^r c_i^l + w^g c_i^l \tag{4}$$

w^g has to be explained in more detail. w^g is the share of cell c_i^l of the pink and red wiring area of the supercell c^{l+1}. We need this notion during floorplanning of cell c^{l+1}, because $w^r c^{l+1}$ and $w^p c^{l+1}$ are useless parameters for the placement of cells c_i^l.

Since

$$w^r c^{l+1} + w^p c^{l+1} = \sum_i w^g c_i^l, \tag{5}$$

we derive by applying equation (3) to c^{l+1} and using equations (1), (4), and (5)

$$a^r c^{l+1} = \sum_i a^g c_i^l + a^e c^{l+1}. \tag{6}$$

Thus, besides the empty space $a^e c^{l+1}$, the green areas of the cells c_i^l completely fill the floorplan of c^{l+1}.

The red boxes are the outlines of the cells after layout with pins at the frame. The red boxes can be defined recursively using equations (1), (2) and (3):

$$a^r c_i^l = \sum_j a^r c_{ij}^{l-1} + w^p c_i^l + w^r c_i^l + a^e c_i^l \tag{7}$$

The disadvantage of this definition is the fact that a net may pass through several levels of the hierarchy without connecting more than one internal pin, that means without pink segments. No good model is known for the estimation of a chain of red segments over several levels. Figure 6 demonstrates this problem.

At this point an explanation of the hierarchy is necessary. We started with a part-of hierarchy with two or three levels at most, and a hundred to a thousand cell instances at each level. With this notion we can easily handle 10^5 to 10^7 primitive cells. This number of levels would not justify the above definitions. Our shape function estimation procedure, as already mentioned,

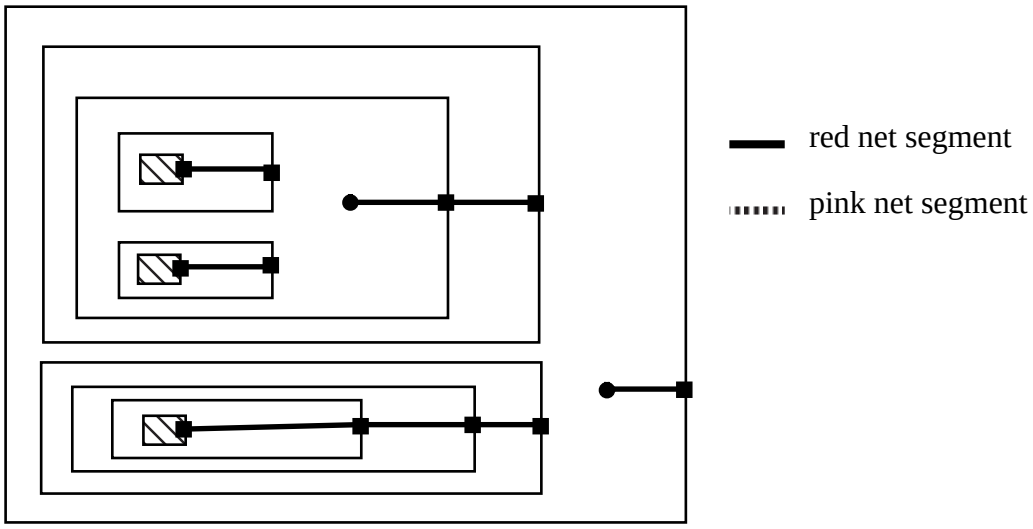

Figure 6: Chain of red net segments

makes use of a slicing structure which can be expressed by the slicing tree. This tree also defines a hierarchy of cells with two subcells per cell in our method or up to five as reported by other authors to allow for pin wheels. If we extend the original hierarchy by the slicing hierarchy, the assumption of many levels is justified.

2.3 The Pink Model

Because of the disadvantage of the recursive red model, we use the following recursive pink model. In order to get rid of the red net segments, we add all red segments to the next pink segment on a higher level. Figure 7 explains this notion. The pink net segments and the pink boxes are shown by dotted lines. The extension of the pink segment yields an increase of the pink wiring area per net. We denote this new area by $\tilde{w}^p$.

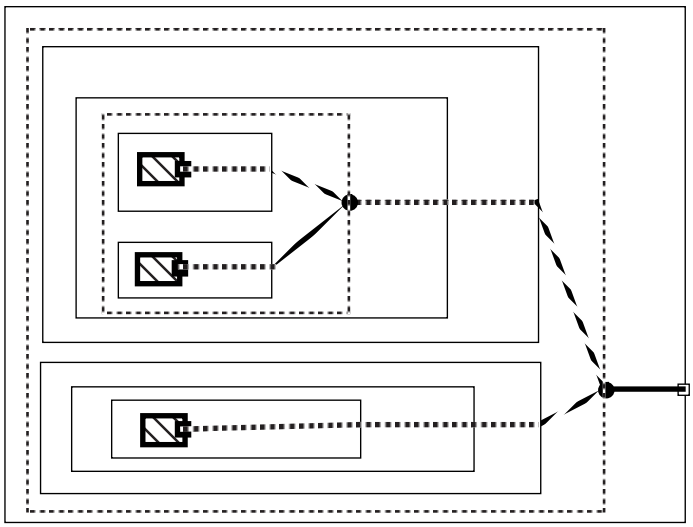

Figure 7: Pink net model

The pink recursion can be described by the area equation

$$\tilde{a}^p c_i^l = \sum_j \tilde{a}^p c_{ij}^{l-1} + \tilde{w}^p c_i^l + a^e c_i^l \quad (8)$$

This means that the pink area of cell c_i^l does not depend on the nets without a pink segment at level l. If we need the red area of a cell, equation (3) has to be modified to

$$a^r c_i^l = \tilde{a}^p c_i^l + \tilde{w}^r c_i^l \quad (9)$$

$\tilde{w}^r c_i^l$ now includes all red segments of nets at all lower levels of the hierarchy that were left out in $\tilde{a}^p c_i^l$. Also, the blue box must be newly defined:

$$\tilde{a}^b c_i^l = \sum_j \tilde{a}^p c_{ij}^{l-1} + a^e c_i^l \quad (10)$$

We model the pink wiring areas of a net as x and y additions to the cell dimensions:

$$x\tilde{w}^p n_m c_i^l = t_x^p \cdot \alpha_x n_m \quad (11)$$

t is the track demand factor, meaning the portion of a track required for a single wire net. t_x is the factor for vertical tracks. t_x^p is further split into t_{xv}^p and t_{xh}^p for the cases of a vertical or horizontal slicing line direction. These cases require different amount of wiring space. We now have the four factors t_{xv}^p, t_{xh}^p, t_{yv}^p, and t_{yh}^p . αx is the geometric width of the net n.The y dimensions are handled equivalently.

Figure 8 defines the used dimensions. As long as we do not know the orientation of cells, we cannot distinguish between left/right and top/bottom (l/r, t/b). We therefore use $x\tilde{w} = xl + xr$ and $y\tilde{w} = yt + yb$.

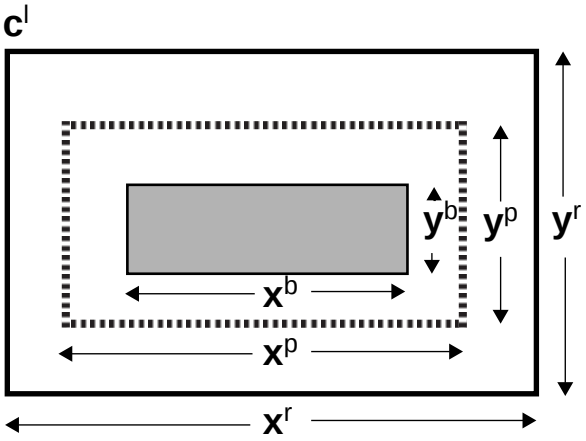

Figure 8: Dimensions

We can calculate xw and yw by building the sum of all individual contributions of nets that are counted. For the pink wiring area we only count the nets that have pink segments in the current cell. We call this set of nets $N^p c_i^l$.

$$\tilde{xw}^p c_i^l = \sum_{m \in N^p c_i^l} \tilde{xw}^p n_m c_i^l \qquad (12)$$

The y dimensions are defined accordingly. The dimensions of the blue box are the result of the sum of the shape functions $\tilde{y}^b = f(\tilde{x}^b)$ of the subcells c_{ij}^{l-1} of cell c_i^l, resulting in

$$\tilde{a}^b c_i^l = \tilde{x}^b c_i^l \cdot \tilde{y}^b c_i^l \qquad (13)$$

Two difficulties arise. First, leafcells in the tree are red and second, the empty space a^e has to be handled. Let us deal with the first problem first. There are two solutions. In the first solution we reduce the red dimensions to pink by subtracting the appropriate red wiring area. The disadvantage of this solution is the fact that we throw the exact dimensions of the leaf cells away and replace them by dimensions based on the red wiring area estimation.

The other solution is a second set of track demand factors for the case of red subcells. We call these factors t_{xv}^r, t_{xh}^r, t_{yv}^r, and t_{yh}^r. These will be smaller than the pink ones because all red segments of nets within the subcells are already included in the cell area. Now the question arises, which color has a cell composed of two red subcells and so forth. We defined a “redness” factor between 0 and 1 which defines the proportion of the pink and red track demand factors used. Red cells have the factor 1. Each time, two cells are composed, the average of the factors is taken and a reduction factor (<1) is subtracted until 0 is reached. We implemented this solution, which also has its disadvantages. It is very difficult to find the right relation between red and pink factors and the value for the red reduction factor. We therefore favor the first solution now.

2.4 Feedthroughs

The second problem, the empty space, occurs because of the corners in the shape functions. If we compose two cells on top of each other as shown in figure 9, the composite cell gets the maximum of both x dimensions and the sum of the y dimensions. The difference in x results in empty space. This empty space could be propagated up in the model.

We use a different solution. We assume that empty space increases the transparency of a cell. Transparency is modeled by vertical and horizontal feedthroughs. Standard cells, for example, have built-in feedthroughs perpendicular to the row. We measure vertical feedthroughs by the wiring space xf in the x-dimension of a cell. The empty space a^e of cell B in figure 9 is modelled as wiring space x^e and thus

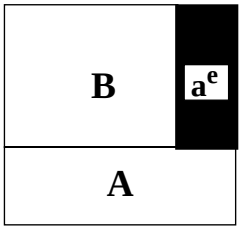

Figure 9: Empty space

$$xf\hat{c}_B = xfc_B + x^e c_B \qquad (14)$$

$\hat{c}$ denotes the cell with empty space added. The question now is: What is the vertical transparency of AB. We use the weighted sum of two extremes. One extreme takes the average of the transparencies of AB, the other the minimum. This results in

$$xfc_{AB} = w \cdot \frac{xf\hat{c}_A + xf\hat{c}_B}{2} + (1-w) \cdot \min(xf\hat{c}_A, xf\hat{c}_B) \quad . \qquad (15)$$

The horizontal transparency in this case is simply, because no empty space has to be considered:

$$yfc_{AB} = yfc_A + yfc_B. \qquad (16)$$

The case of cells with a vertical slicing line is handled appropriately. In the calculation of wiring space the feedthrough dimensions are subtracted up to the additional wiring space for nets. Any remaining feedthrough space is propagated up in the slicing tree.

This pink model has proved its quality for standard cell and flexible cell examples. The results for standard cells are shown in chapter 3. Our shape function generator therefore strictly uses the pink model for the complete slicing and hierarchy trees. In order to use the shape function, the additional red wiring space has to be added at the level where the shape function is used. We call this additional space the red shift.

2.5 Red Shift

The red shift is handled as an addition to the x and y dimensions of the pink cell. Let π be the number of one wire pins of cell c_i^l (corresponding to the number of nets N^r). If we assume an equal distribution of the pins on the four sides of the cell, we can calculate four values π_l, π_r, π_t, and π_b. This is similar to the assumption that the pins can freely float around the perimeter of the cell. If we have more knowledge about the positions of the pins, for example after global routing during floorplanning, the four values can be determined more precisely.

If the capacity of the sides of the red box is larger than the associated number of pins, we can calculate the dimensions of the red box as:

$$x^r = x^p + \max(\lceil (\pi_t + \pi_b) \cdot t^r_{xv} + (\pi_l + \pi_r) \cdot t^r_{xh} - xf \rceil, 0) \qquad (17)$$

For brevity we have eliminated the cell identifier c^l_i and have assumed a unit grid. t^r_{xv} is the red shift track demand factor for vertical nets and t^r_{xh} is the factor for the vertical part of horizontal nets. The feedthroughs xf are only subtracted up to a value which leaves the red shift positive (max-function). The y-dimension is handled accordingly.

The capacity of a side depends on the design style. Let us assume blocks of horizontal standard cell rows as an example. It is reasonable to have no pins to the side of cell rows. Therefore, the capacity of the vertical sides is equal to the y-dimension minus the accumulated height of the rows. The latter is equivalent to y^b of the blue box.

Therefore, we have the additional conditions for y^r :

$$y^r \geq \pi_l + y^b \quad \text{and} \quad y^r \geq \pi_r + y^b \qquad (18)$$

For the x-dimension the conditions are different. We assume that the internal pin positions of the nets entering from the bottom are equally distributed. For R rows, $\pi_b \cdot (R-1)/R$ nets cross the bottom row. The number of feedthroughs of this row is assumed to limit the capacity. None of the built-in feedthroughs of the standard cells of this row is used for internal wiring. Let T_x be the average transparency for vertical wires of all standard cells in c^l_i, defined as number of vertical feedthroughs divided by the x-dimension of the cell, then we arrive at the conditions:

$$x^r \geq \frac{\pi_b \cdot (R-1)}{T_x \cdot R} \quad \text{and} \quad x^r \geq \frac{\pi_t \cdot (R-1)}{T_x \cdot R} \qquad (19)$$

In the case the conditions are not met, we add one grid unit to the x- or y-dimension for each pin that violates the condition.

3. Experimental Results

Here we restrict ourselves to standard cell blocks. The examples are taken from a large design with nearly 300 000 standard cells. The design was broken down into three levels of a hierarchy and the examples are taken from the lowest level. The structure of the design was generated with the high-level synthesis system MIMOLA which resulted in a high connectivity. Therefore the relative wiring area is very large compared to other benchmarks. We used a commercial standard cell library ACMOS 4H from Siemens AG. This is a 1.25µm CMOS technology with an average transparency $T_x = 0.6$.

For the layout of the standard cell blocks we used our own implementation of simulated annealing similar to TimberWolf SC /SeL87/ and a heuristic channel router that could route most channels in density. Two wiring layers are assumed.

We performed three kinds of experiments. In the first experiment we compared the estimated red shape function for cells which had unrestricted pin intervals with synthesized layouts. Then, we deleted the external pins from the netlist of these cells for measuring the pink boxes. The second result was a set of layouts of a multiplexer that occurred several times on the chip. The different sizes of these layouts show that the estimation of the red shift is very important. The third experiment was the quantitative determination of the red shift for cells with restricted pin intervals.

All experiments have shown that the track demand factors strongly depend on the restrictions on the pins. In the case of no restriction (our first experiment), we found t^r_{xh} and t^r_{yv} to be zero. The routing is nearly straight from the internal pins to the cell frame. There is no need for additional tracks in the orthogonal direction. For our standard cell synthesis tool, the value $t^r_{yh} = 0.3$ seems to be reasonable. Regarding the detailed routing result, it seems that t^r_{xv} should have the same or a slightly less value. We could not measure this value exactly because the transparency xf reduced the horizontal red shift to zero in all cases.

Table 3 shows the experimental results of 8 different cells. Our circuits consist of 166 to 827 standard cells with 62 to 423 pins (table 1). We measured the pink and red boxes for a wide aspect ratio range (different number of rows). In all cases there were no restrictions on the pins.

cellname	number of subcells	nets	frame pins
alu32bit	827	877	100
alu32_ctrl	215	250	106
nmux05x32	166	269	195
nmux08x32	258	484	291
pe1_1536	530	746	423
pe2_1173	382	617	395
sioo3x16thl	602	408	62
sn32	600	686	103

Table 1: Example circuits

cut orientation	net-to-cut orientation	track demand factor
horizontal	parallel	0.5
horizontal	orthogonal	0.1
vertical	parallel	0.1
vertical	orthogonal	0.4

Table 2: Pink track demand factors

Columns 1 and 2 of table 3 show the cell name and the number of rows of our measurements. Columns 3 and 4 contain the areas of the smallest “pink layouts” (which we got by deleting the frame pins from the netlist) and the deviations of the “pink layouts” from the pink shape function. The shape function was computed by using the sizing parameters of table 2. The last two columns show the minimum areas and the deviations of the red boxes.

Except for two examples (pe1_1536 and sioo3x16thl), the errors of the red shape functions were less than 10%. This shows that we can estimate very well the smallest layouts of a cell which usually will be synthesized without pin restrictions. The two cells with the larger estimation errors were investigated in more detail. We found that these cells have many large nets with many connections. These cells cannot be handled accurately by our standard cell placement tool.

cellname	row numbers	pink boxes area$_{min}$ [mm^2]	pink boxes aver. error [%]	red boxes area$_{min}$ [mm^2]	red boxes aver. error [%]
alu32bit	2 .. 24	3.9	9.0	3.9	7.1
alu_ctrl	2 .. 15	0.8	6.8	0.9	7.6
nmux05x32	2 .. 12	0.6	23.7	0.9	8.7
nmux08x32	2 .. 15	1.0	15.9	1.3	4.5
pe1_1536	2 .. 25	2.6	12.8	2.7	13.8
pe2_1173	2 .. 18	1.8	3.8	2.3	9.4
sioo3x16thl	3 .. 30	3.9	18.0	3.9	16.2
sn32	2 .. 22	2.2	10.6	2.4	3.1

Table 3: Average area estimation error for pink boxes and red boxes of cells without pin restrictions

For instance, the smallest manual layout of cell sioo3x16thl was about 30% smaller (2.6mm^2) than the smallest layout our tool achieved. It seems that the results of synthesis tools which can handle these nets better will be closer to the shape function.

The estimation errors of the “pink layouts” are mostly due to the synthesis tool, too. This tool generally yields not very good results when no forces to the cell frame exist. For these cases the cost function should be changed. The penalty for overlapping cells must be increased. On the other hand, generating a good “pink layout” is of no practical interest.

Our second experiment shows very clearly that the layout in a top-down design depends largely on the positions of the frame pins. Figure 10 shows the red shape function and all layouts of a multiplexer (nmux05x32) which occurred several times on the chip. The shape function was computed with the track demand factor $t^r_{yh} = 0.3$ for cells without pin restrictions. The figure shows two results. First, the shape function corresponds to the lower bound of the measured values and second, the different heights of layouts with the same number of rows. The different y-dimensions for nearly the same x-value result from different pin interval restrictions as a result of floorplanning. We can see that this influence is extremely large and thus the estimation of the red shift is very important for a precise floorplanning at the next upper hierarchy level.

In our third experiment, we investigated this red shift for cells with restricted pin positions. We restricted ourselves to examples where the capacities of the sides were larger than the number of pins. For these samples we tried to find track demand factors for the entering nets. We restricted the pins to a single side and assigned the pins randomly to the sides. The track demand factors should be approximately double of these for unrestricted pin positions because the average net length will be twice as large due to the restrictions (if we assume a random placement for both cases). Table 4 shows the deviation of the red shape function to the sample layouts by

Figure is missing

Figure 10: Estimated shape function and layout results with different pin positions

using the track demand factor $t^r_{yh} = 0.6$. The factor t^r_{xv} is superfluous for our experiments because the built-in feedthroughs of our standard cells reduce the horizontal red shift x^r to zero.

cellname	row numbers	$area_{min}$ [mm^2]	aver. error [%]
alu32bit	2 .. 24	4.2	10.8
alu_ctrl	2 .. 15	1.0	4.7
nmux05x32	2 .. 12	0.9	22.1
nmux08x32	2 .. 15	1.5	8.2
pe1_1536	2 .. 25	4.1	12.7
pe2_1173	2 .. 18	2.8	5.8
sioo3x16thl	3 .. 30	3.9	16.5
sn32	2 .. 22	2.4	7.1

Table 4: Average area estimation error for red boxes. The pins were randomly assigned to one side each.

Experiments with further restrictions on the frame pins have shown that the cell area will not increase in all cases. We had several examples where the area slightly decreased when the pins were restricted to smaller intervals. In other cases, the area increased. For a small number of samples the variations of the cell areas were more than 30%.

We observed a similar result by another experiment, too. For this experiment we only interchanged the intervals of the pins. This was done for the nmux08x32 multiplexer. Since all input ports and the output port had the same width (32 bit), the red shift must be unchanged although we interchanged the assignment of the ports to the intervals. A red shift computation that does not analyze the contents of a cell (i.e. the netlist and/or the behavior) can only depend on the locations of the pins. In our experiment, the number of pins assigned to one interval was the same for all permutations. However, the variations of the cell areas were about 15%.

Both experiments show that it is not possible to estimate precisely the red shift of a cell with arbitrary pin restrictions. It has no sense to look for a more complex red shift model that shall improve the results of table 4.

4. Conclusion

The most difficult part of the shape function computation is an accurate wiring area estimation. The aim of this paper was to describe in detail the wiring area estimation of our shape function generation method. We introduced a mathematical five color model. Each color is related to a certain part of the wiring area. Since it is not possible to estimate accurately the wiring space of the net segments connecting the frame pins of a cell, we described a recursive pink model that considers the interconnection of the subcells only. The remaining wiring space needed to connect the frame pins has to be added only on the hierarchy level where the shape function is used. Inaccuracies of the red shift will not be multiplied with the number of hierarchy levels of a circuit.

Results taken from a large number of sample layouts have shown that the accuracy of the pink model is sufficient for an early area estimation and for floorplanning. The same accuracy was obtained for the red shape function compared to the lower bound of all layouts of a cell. However, the wiring area we need to connect the frame pins cannot be estimated with the same accuracy. The interchange of the association of the pins and intervals already results in area differences of more than 10%.

Until now we restricted our measurements to standard cell blocks where the capacities of the sides were larger than the number of pins. Our next aim is to improve our red shift model for the case that the pins are not equally distributed over the cell frame. This is necessary for the top-down floorplanning where the global router decides the pin positions. Then, we will improve our area estimation model for the upper general cell levels and we will try to determine the model parameters for these levels more precisely.

5. References

/Gak83/ D. Gajski, R. Kuhn, “New VLSI Tools”, IEEE Computer, Vol.16, No 12, 11-14, 1983

/GiO84/ L. van Ginneken, R. Otten, “Stepwise Layout Refinement”, Proc. Int. Conf. on Computer Design, Port Chester, 30-36, 1984

/Ott83/ R. Otten, “Efficient Floorplan Optimization”, Proc. Int. Conf. on Computer Design, Port Chester, 499-503, 1983

/SeL87/ C. Sechen, K. Lee, “An Improved Simulated Annealing Algorithm for Row-Based Placement”, Proc. Int. Conf. on Computer Aided Design, Santa Clara, 478-481, 1987

/Zim88/ G. Zimmermann, “A New Area Shape Function Estimation Technique for VLSI Layouts”, Proc. 25th Design Automation Conference (DAC), Anaheim, 60-65, 1988

/Zim90/ G. Zimmermann, “VLSI Area Estimation Tolerances - Shape Function Generation vs. Floorplanning”, Proc. of the EURO ASIC ‘90, Paris, 202-207, 1990