Three-Phase Chip Planning
- An Improved Top-Down Chip Planning Strategy -

Bernd Schürmann Joachim Altmeyer Gerhard Zimmermann

University of Kaiserslautern
D-6750 Kaiserslautern, Germany

Abstract

The most important precondition for top-down chip planning is a good area estimation. However, each estimation has tolerances which result in differences of the estimated shapes in the floorplan and the final layouts.

This paper introduces an improved top-down chip planning method that reduces the effects of these deviations. In a fully recursive approach, each cell is planned several times with different presumptions. Bottom-up adjustment steps use refined shape functions instead of rigid dimensions. Although we perform such bottom-up adjustment steps, the general direction is top-down. The convergence of our procedure can be ensured.

Within the paper, we describe our method in detail and provide some experimental results. Several real big test designs (the largest example has nearly 300.000 standard cells) have been performed with our PLAYOUT design system to compare the pure top-down approach with our new method.

1. Introduction

The design of large VLSI circuits needs a hierarchical approach. Because of their complexity these circuits cannot be designed in a flat fashion. There are two possible hierarchical approaches, *top-down* and *bottom-up*.

A bottom-up design system first generates the layouts of the leaf cells with minimal area. When all layouts are available, they have to be composed bottom-up [17]. Because of their fixed shapes more or less empty space will be the result. Many current design systems try to avoid this empty space by allowing several shapes for the leaf cells [6], [1], [4], [18].

However, if each cell has only a small number of different shapes, empty space cannot be avoided. In addition, the fixed pin positions result in long wiring nets. The most recent design systems avoid both problems by applying a top-down design strategy [19], [21], [7], [11]. Here we have a top-down chip planning step before the layouts will be composed bottom-up with respect to the top-down computed floorplan. Within the chip planning phase the subcells have flexible shapes and free pin positions. The sizes of the subcells were computed by a preceding area estimation step. The result of the estimation is a shape function that represents the minimal area of a cell for all aspect ratios.

It is obvious that the quality of a top-down design depends very much on the quality of the estimation phase. However, it is not possible to estimate the area of a cell exactly. All estimation methods have tolerances because they do not have all informations the layout generators will have.

The analysis of many test designs has shown that even small changes in the shape of a (sub)cell after planning may result in a large amount of empty space. These differences between estimation and final layout make an improved top-down chip planning method necessary. We developed such a new planning strategy which we call *Three-Phase Chip Planning*. Within an iterative planning method each cell will be planned several times. There are three different planning phases. The first phase will be applied to each cell only once. Within this phase, the initial floorplan will be computed, whose topology will be kept fixed for phases two and three. Both remaining phases are adjustment steps. While phase two adjusts the floorplan to a new input frame that is computed top-down by a further planning of the supercell, the third phase adjusts the floorplan to more precise subcell informations (refined shape functions).

The rest of this paper is divided into four chapters. Chapter 2 briefly describes the typical top-down design. In chapter 3 we describe an experiment that shows that there exist unavoidable estimation tolerances which cannot be reduced by improved estimation methods. These tolerances make the improved planning strategy necessary. Chapter 4 describes the Three-Phase Chip Planning strategy in detail. Finally, chapter 5 contains several experimental results.

0-8186-3010-8/92 $03.00

2. Hierarchical Top-Down Design

Because of its complexity, the design of circuits with one million basic cells or more has to be performed hierarchically. The whole design will be divided into a part-of hierarchy and into design domains which are shown by the design plane in figure 1. The design process traverses the design plane from left to right. Within each domain, the process can be performed top-down (e.g. chip planning), bottom-up (e.g. chip assembly), or in a mixed manner. Future technologies will need three or more hierarchy levels which are currently handled by only few design systems [19], [21].

Figure 2 shows the kernel of our PLAYOUT design system which we will use for describing the basic design flow. PLAYOUT [21] is a prototype design system that operates on any number of hierarchy levels beginning at the domain behavior up to the domain masklayout.

The first step in designing a circuit is the *structure synthesis*. This step can be performed automatically by using a synthesis system like MIMOLA [8] or by a schematic entry tool. Generally, the output of these tools is a netlist which is coded in a hardware description language.

In general, structure synthesis tools do not generate a hierarchy which can be used directly for the physical design. In most cases, we have too many (several hundred) register transfer blocks as part of a processor netlist while the register transfer blocks are built up by a deep hierarchy with a few modules at each level. The cells at system level also consist of only a small number of subcells. On the other hand, physical design tools generate good results only if the number of cells is within a certain range. The circuit has to be *repartitioned* into a physical hierarchy for the geometrical design steps.

After completing the physical hierarchy, the geometrical (physical) design takes place. This design phase is divided into the *chip planning* and the *chip assembly*

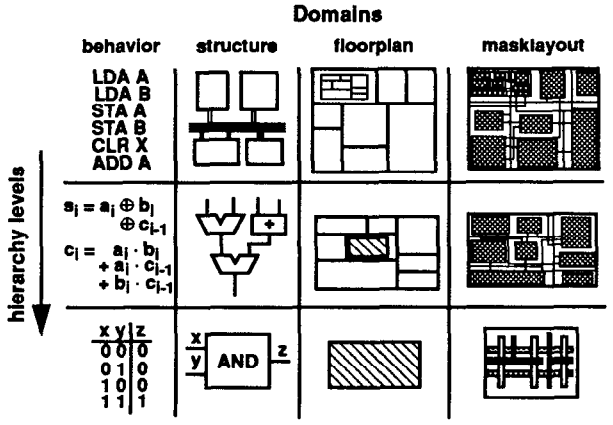

Figure 1: The design plane.

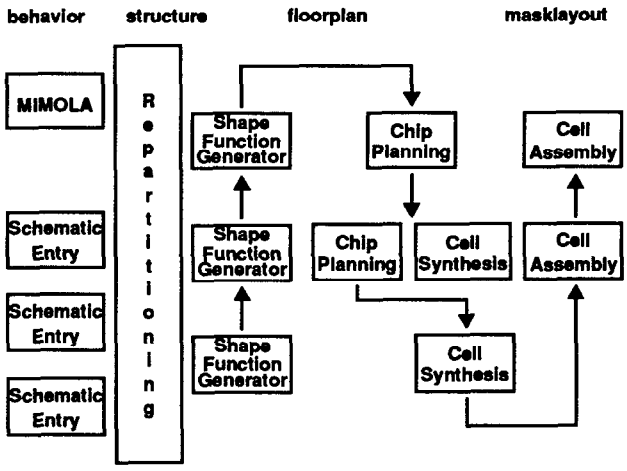

Figure 2: PLAYOUT toolboxes.
Most toolboxes are used recursively at several hierarchy levels. The arrows show the general physical design flow.

steps. The former step is further divided. A bottom-up shape function generation (*area estimation*) is necessary for the following top-down chip planning (figure 2). These steps will be described later in more detail.

The *chip assembly* completes the final layout. The layouts of the leaf cells are synthesized by the *cell synthesis*. Cells on higher hierarchy levels are assembled with respect to the top-down computed floorplan (*cell assembly*). This plan has to be revised each time the estimated width of a channel is different from the detailed routing result. The layout of a cell is finished when all channels are routed correctly [3].

2.1 Top-Down Chip Planning

An important part of the top-down VLSI design is the (top-down) chip planning. As described above, this step is divided into a bottom-up area estimation and a following top-down planning phase (figure 3).

The bottom-up area estimation is the basis of the geometrical top-down design. The estimation is input to the chip planning, and it is useful for an early area prediction of the whole chip. The estimated area is available before any floorplan or even a layout is computed. For each *cell under design* (CUD) we compute a shape function by using the shape functions of its subcells.

There are three different kinds of cells which are abstracted by shape functions: *macros*, *multi-macros*, and *flexible cells*. Macros have one fixed layout. They are abstracted by shape functions with only one corner point that represents the dimensions of the layout. If a cell has several layouts with different shapes, we call these multi-macros. The corresponding shape function has corner points for all layout alternatives which determine the lower bound. For flexible cells, only estimated shape functions can exist. Each corner point has tolerances in x and y

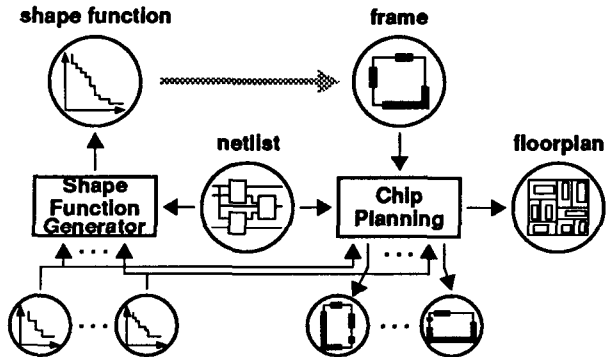

Figure 3: Area estimation and chip planning.

dimensions. In chapter 5 we will extend these three shape function classes by so-called *refined shape functions* which describe the possible shapes of a floorplan with still flexible subcells.

We assume that all layout geometries can be approximated by slicing topologies without loosing much of the quality. These slicing topologies can be represented by binary slicing trees. If shape functions for the leaves of a tree are known they can be easily added up the tree and thus for the root node which represents the cell of interest [10].

The resulting shape function estimates the total area only if no additional wiring space is required. This wiring area cannot be computed precisely and must be estimated [20], [5]. Measurements on a large number of layouts have shown that this wiring area estimation depends very much on the quality of the placements and the CUD pin positions [16]. Since these locations are not known during the bottom-up area estimation, we estimate the area which is needed by a good placement without any CUD pin position restrictions. The increase in area that is due to the pin position restrictions must be computed during the following top-down chip planning.

Pure top-down chip planning starts at the topmost hierarchy level. For each cell, a floorplan will be computed that is based on the estimated shape functions of its subcells. In addition, the shapes of the flexible subcells will be computed which are input for the chip planning at the next lower hierarchy level. The locations of the subcell pins are restricted due to the global routing.

Each chip planning consists of three main steps: the placement of the subcells, the global wiring, and the estimation of the subcell shapes. For the first two steps, many algorithms have already been published. The estimation of the subcell shapes is based on the bottom-up area estimation as described above. Since the planning process determines the positions of the subcell pins, we now have to estimate the increase in area that is due to the pin position restrictions.

In the next chapter, we show that it is not possible to estimate the subcell area more precisely than 10 - 15%. We must choose estimation parameters for the average case such that the sum of all subcell areas plus the wiring area needed to connect the subcells should diverge from the CUD layout not more than 10%. However, because of the inaccuracies of the area estimations, the subcells do not fit accurately into the planned spaces of the floorplan. The empty space increases and so the overall cell area. To reduce this empty space, we developed an improved chip planning method that is described in chapter 4.

3. Limits of the Area Estimation

As described above, the kernel of a top-down design system is a good area estimation technique [16], [20]. Extensive measurements with our design system yielded that the average deviations between the estimated area and the final layout of a cell are about 10-15%. The occurring estimation tolerances can be categorized in two different classes: *avoidable errors* and *unavoidable deviations*.

Avoidable Errors

Although our underlying model for shape function estimation [20] results in good area predictions, an improvement of the quality and an adaptation to different design methods and technologies are future topics of our research. A discussion of useful estimation techniques are outside the scope of this paper. Interesting examinations can be found in [16].

Unavoidable Deviations

Beside the previously mentioned errors, there are estimation tolerances which we *cannot* avoid. A simple experiment demonstrates the existence of such insecurities. Figure 4 shows three different floorplan frames of a subcell with pin assignments.

For estimating the area, we have to examine the entry direction of the nets into the cell. Nets which are entering a cell in horizontal direction will widen the height of the cell. So, the height of cell A is larger than for B and C because all nets are entering the cell at the left and right sides.

But if we look at B and C, the locations of the pin intervals are identical. For both cells we have 32 wires at each of the left, right, and bottom sides. The amount of the area for the entering nets should be identical. The size of a cell should not change by only permuting the pin to interval association.

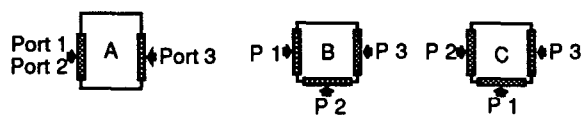

Figure 4: Three different frames of one cell.
(All ports have 32-bit pins.)

Permutation	Area			
	TimberWolfSC		PLAYOUT	
permut1	1.51 mm^2	100 %	1.61 mm^2	103 %
permut2	1.64 mm^2	109 %	1.59 mm^2	101 %
permut3	1.70 mm^2	113 %	1.80 mm^2	115 %
permut4	1.72 mm^2	114 %	1.57 mm^2	100 %

Table 1: Deviations in area due to permutations of the pin assignment.

We performed a simple experiment which showed that our assumption is not true. We synthesized the standard cell block layout of a multiplexer with 8 input ports with 32 bits each. The standard cell placement has been performed by our PLAYOUT synthesis tool and the TimberWolfSC system (Version 6.0; [12]) which are both based on simulated annealing.

In our experiment, we computed four layouts with each synthesis tool. For the TimberWolf measurement, each port was restricted to one side of the frame while we used smaller intervals for our synthesis tool. The shape of the given frame was the same for all placements. We only permuted the association of the ports to the intervals. The total number of wires of a particular interval did not change in all experiments (similar to cells B and C in figure 4).

Table 1 shows the result of our experiments. In contrast to our assumption, the deviation of the layout areas was great, up to 15%. It is therefore not possible to estimate the area of this cell more precisely than 15% by using the interface description only, i.e. without inspecting the internal structure of the cell.

4. Three-Phase Chip Planning

As we have seen in the previous chapter, the average tolerance of the estimation is about 10-15%. However, in the worst case the deviation can be substantially greater. In the pure top-down chip planning, we have no chance to compensate these differences. The only possibility to react upon deviations is to return to the shape function estimation phase and to change estimation parameters or to use realizations (layouts) of critical cells (using macros or multi-macros instead of flexible cells).

On the other hand, if we are not planning the top-most cell, it may be possible to compensate the deviation of the current cell shape with the deviations of the sibling cells. The area of the supercell should not change if the sum of all area estimations of sibling cells is similar to the sum of the areas of the realizations. Furthermore, all planned cells are slightly flexible because their subcells are still flexible. This flexibility increases the chance of a good balancing.

So, it is useful, after computing a floorplan of the CUD, to perform an *adjustment planning step* for the supercell before continuing the top-down planning process at the subcell level. The deviations at the current level can be compensated at the supercell level. Figure 5 depicts the methodical proceeding of an adjustment process. We assume that the CUD is at the hierarchy level i. There are three planning phases which we denote by the greek letters α, β, and γ. After planning the subcells at level i+1 (phase α), we perform an adjustment step at level i (phase γ) before continuing the top-down planning of all subcells at level i+1 (phase β).

Of course, the cell at level i itself is part of an adjustment process for its supercell on the level i-1. Thus, the three fundamental planning phases α, β, and γ are performed with each cell. Therefore, we call our planning method *Three-Phase Planning*.

4.1 Description of the three phases

The following actions are performed in a particular planning phase:

Phase α (*initial floorplan*):
- placement with flexible, macro, and multi-macro subcells (using the frame description from the supercell planning as in the pure top-down approach)
- global wiring
- wiring area estimation
- legalization (computing a correct geometry and a refined shape function for the CUD)
- computation of pin constraints for the CUD

The result of phase α is a floorplan from which we use the topology for further planning steps and the pin constraints of the CUD for an adjustment step at the supercell level (phase γ). Since the subcells are still flexible many floorplans with different shapes but the same topology are feasible. We generate a *refined shape function* that describes all possible shapes of the same topology. This shape function is a second, very important input to the supercell adjustment. Refined shape functions are more precise than shape functions of flexible cells because they rely on a particular topology and a global routing (not only on a rough wiring area estimation). We call these cells *semiflexible* because of

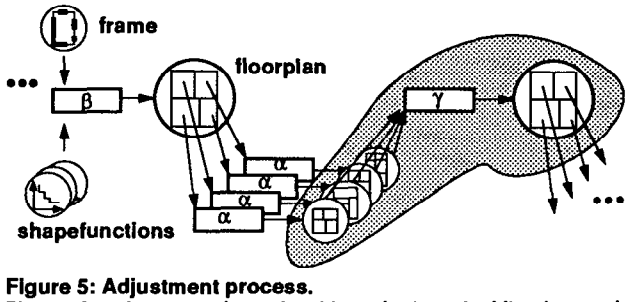

Figure 5: Adjustment process.
Phases β and γ are performed at hierarchy level i while phase α is applied at level i+1.

their restricted flexibility.

The actions of phase α are similar to the pure top-down chip planning. In the pure top-down planning, no constraints for the CUD are needed but the constraints for the subcells are computed (see also phases β and γ)

Phase β (*adjustment to new frame*):

- global wiring (using the placement of phase α and the new frame description from the supercell adjustment step γ)
- wiring area estimation
- legalization (selecting subcell shapes from their shape functions)
- computation of pin constraints for the flexible subcells

In the adjustment step γ of the supercell, a new frame of the CUD was computed based on its refined shape function and its pin constraints of phase α. In phase β, we adjust the floorplan from phase α to this new frame.

As in the pure top-down planning strategy, we now compute the pin constraints for the subcells. The subcells can then be planned in phase α (see above) which results in semiflexible cells with new constraints (refined shape functions and pin positions). These constraints are input to the CUD adjustment planning phase γ.

Phase γ (*adjustment to more precise subcell data*):

- correction of the global wiring (because subcell pins can "change" their sides. We have to build a consistent description of the wiring information.)
- wiring area estimation
- selection of subcell shapes using the refined shape functions of the subcells (from phase α)
- computation of pin constraints for the subcells

The resulting subcell constraints (area, shape, and pin positions) of phase γ are used as input to phases β and γ of the subcells.

In figure 6 the three planning phases with the top-down and bottom-up data interchange are outlined. In contrast to our refined shape functions, all other hierarchical top-down design systems use at most one fixed shape for an

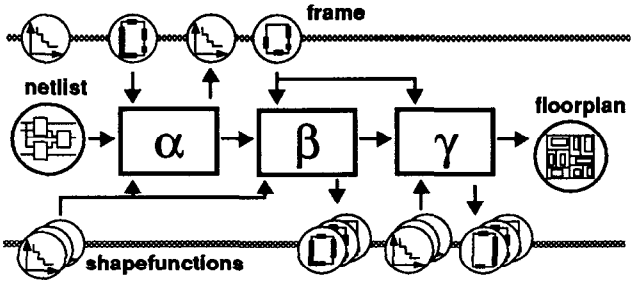

Figure 6: Chip planner input/output during the three Phases. (A floorplan is also passed from phase α to β and from β to γ)

adjustment [19]. They do not provide any top-down adjustment like the phase β.

Figure 7 shows an abstract notation of a hierarchical planning over three levels. Chip planning steps at one hierarchy level are combined to one bar.

Even though there are bottom-up movements, it is obvious that the global direction of the design process is still top-down. At the top level, the pad frame is created (e.g. by a graphical *pad frame editor*; *pfe*) and at the lowest hierarchy level we perform the cell synthesis *syn* (e.g. a standard cell block layout computation).

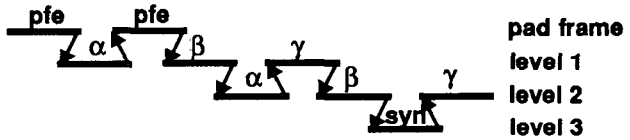

Figure 7: Three Level Chip Planning.

For simplification, we will only consider the chip planning in the rest of this paper. The management of the cell synthesis is equal to the management of the planning phase α. While the inputs are the same, the synthesis returns a final layout instead of a floorplan.

4.2 Stepwise Refinement

So far we assumed that plannings α of all subcells will be performed in parallel and independent of each other. Figure 8 shows a primitive floorplan with three modules. We assume that the floorplan of module A becomes larger than its estimation while module B becomes smaller. Module C should be estimated correctly. After planning all three subcells in phase α, each of them returns a refined shape function. The floorplan of figure 8a shows the result of the adjustment phase γ. The existing floorplans are represented by gray rectangles. Using the refined shape functions, the adjustment γ tries to balance the overall floorplan. Unfortunately, because of its fixed topology, module B cannot be realized smaller. So, the total floorplan area increases a little while the area above module B is unused.

Balancing the floorplan in phase γ can be improved by building the floorplans of the subcells step by step. Figure 8b illustrates such a stepwise refinement. For the adjustment γ_1 only subcell C has already been planned because this module may determine the whole floorplan width alone. A and B remain flexible. There exist pin constraints from C to A and B. Step γ_2 tries to balance the floorplans of A and C which influence the still flexible subcell B. The shape of module C is similar to its estimation and module A became larger. The larger frame of A results in a new shape of B that becomes higher and smaller. The whole floorplan is now larger than after phase β. Step γ_3 finally inserts the smaller floorplan of module B that now has a

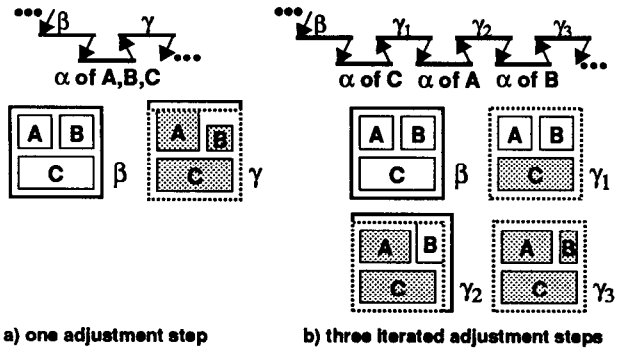

Figure 8: Iterative Three-Phase Planning.
The dotted rectangles show the frame of phase β.

fitting aspect ratio (in contrast to figure 8a). After this adjustment step, we continue with phase β for all subcells.

Another possible strategy is to plan all subcells (perhaps with different topologies for each subcell) and leave the final configuration decision to the algorithms of the chip planner toolbox. Within the adjustment phase γ of the CUD, the chip planner chooses the most fitting floorplan alternative of each cell or it may be desirable to insert all critical cells first (e.g. cells which do not fit to the estimations) and to leave the remaining cells flexible which can balance the whole floorplan in a following adjustment step.

An extreme possibility of the stepwise refinement strategy is a depth-first run through the hierarchy tree. The advantage of this method is that we have the freedom to build critical modules (subtrees) step by step first. The disadvantage of a depth-first strategy is that we loose the whole parallelism. No planning processes can be performed concurrently.

4.3 Convergence

In total, there are many possible strategies although we use only three different phases: phase α for the initial floorplan, phase β for the top-down adjustment, and phase γ for the bottom-up adjustment. In an extreme strategy, it is possible to descend and to ascend the hierarchy tree in a yo-yo fashion. So, we have to ask whether our procedure terminates.

The answer is *YES*. In each bottom-up adjustment step γ we always replace at least one flexible subcell by a floorplan or a layout. A floorplan will be replaced by a layout only. When all flexible subcells are replaced by a layout, the final cell assembly terminates the adjustment procedure.

While the procedure terminates for all strategies, we can see a convergence behavior when executing the stepwise refinement strategy. Here, we replace the inexact flexible cells with more exact rigid cells step by step. With each replacement, one inexact component has been removed and with that the tolerances will become smaller. The floorplan geometries converge to the final layout.

4.4 Restrictions

In spite of all the freedoms we have, we are not permitted to build a cell and one of its subcells in parallel. Before we can make an adjustment step γ_k, we have to stop all subcell planning processes and have to collect the current results as input for γ_k. After γ_k, we can go on with the subcell plannings in consideration of the new constraints from γ_k. If we do not stop the subcell processes, we build divergent bottom-up and top-down frame descriptions.

Experiments with our PLAYOUT design system have shown that it is not easy to meet a correct design flow when designing large circuits. An automatic flow management will be necessary that keeps track about the current design state and all allowed actions. For that we developed a set of rules which may be the basis of an automatic design management that controls a correct design flow. The rules are described in an extended version of this paper [15].

5. Results

In the recent past, we performed one large and several smaller test designs using the improved planning strategy. The large example is a circuit with nearly 300.000 standard cells. The design was broken down into three hierarchy levels and is described in [14]. Although the size of such a circuit is beyond the current technology (we used the Siemens 1.25µm technology and achieved a chip of approximately $10x10cm^2$), the goal of that design was a demonstration of the feasibility of designs in future technologies by using top-down design systems like our PLAYOUT system.

The structure was generated by the high level synthesis system MIMOLA [8] which resulted in a circuit with a high connectivity. The relative wiring area is very large compared to other benchmarks. This large amount of wiring area yielded large estimation tolerances which made the Three-Phase Chip Planning strategy necessary.

The top-level cell consists of 12 macro cells and 20 flexible modules which are composed of 40-50 blocks each. So, we had two planning and one cell synthesis levels. The whole circuit was realized by performing the Three-Phase Chip Planning strategy. In total, we performed nearly 80 planning steps (e.g. 9 adjustments at top level) and more than 600 cell synthesis steps.

Table 2 shows a comparison between the Three-Phase Chip Planning and the pure top-down planning of the top level cell (XLII) and six cells of the second hierarchy level. At the top level, the adjustments by using refined shape functions resulted in a 7% smaller floorplan compared

name	Top-Down Planning		3-Phase Planning		gain
	size [mm]	area [mm^2]	size [mm]	area [mm^2]	
XLII	101.5 x 91.0	9.229	98.1 x 87.9	8.616	7%
PE1.1	22.2 x 15.0	333	19.9 x 15.2	303	10%
PE1.2	26.3 x 13.3	350	---	---	---
PE1.3	23.1 x 15.0	347	21.2 x 15.1	320	8%
PE2.1	20.5 x 17.2	353	19.4 x 17.4	338	4%
PE2.2	23.2 x 17.5	406	21.4 x 17.5	375	8%
PE2.3	17.6 x 23.4	412	---	---	---

Table 2: Gain of the Three-Phase Chip Planning.

with a pure top-down planning.

At the second hierarchy level, we performed our stepwise refinement method only. Figure 9 depicts a part of one block of this level. The figure shows the success of the iterative planning strategy very well. Figure 9a shows the result of phase β. The subcell shapes are based on our estimation method. Figure 9b demonstrates the pure top-down planning and contains the layouts of the standard cell blocks. The layouts were computed with respect to the frames of figure 9a. The results of the cell synthesis pointed out that the alu's at the left side (u6alu, u7alu, u9alu, u11alu) were estimated too small while the register files in the bottom-right corner (sioo3x16thl1, sioo3x16thl3, sioo3x16thl4) became smaller than the estimation.

Using our stepwise refinement strategy, we first inserted these seven critical layouts in phase γ. All other modules remained flexible. They got new frames which balanced the whole cell better. In particular, two cells (pe2_1173 and pe2_1174) got totally different aspect ratios, which fitted to the layouts of the alu's and register files. Since the layouts of the remaining flexible modules did not differ much from the estimations, the result of the Three-Phase Chip Planning method (figure 9c) was a much smaller circuit than the result of the pure top-down approach.

The results of six typical cells at the second hierarchy level are also shown in table 2 (the other cells yielded similar results). There were two cells (PE1.1 and PE2.1) which consist of almost only flexible subcells. PE1.2 and PE2.2 consist of about 15 flexible and 25 multi-macro cells while PE1.3 and PE2.3 consist of 25 macros instead of the multi-macro cells.

In four of the six cases, the gain of the Three-Phase Chip Planning strategy was large (up to 10%) while for two cells (PE1.2 and PE2.3) we did not need any iteration step. These cells consist of many macros and multi-macros, respectively, which resulted in a large amount of empty space in the floorplan (already in phases α and β). The additional layout area of all subcells which became larger than their estimations could be compensated by this empty space (and by the cells which became smaller).

Finally, we observed the following interesting phenomenon: the more flexible modules a cell contains, the more adjustment steps were needed to compensate the larger tolerances. On the other hand, these cells result in smaller total area than cells built up by many macro cells.

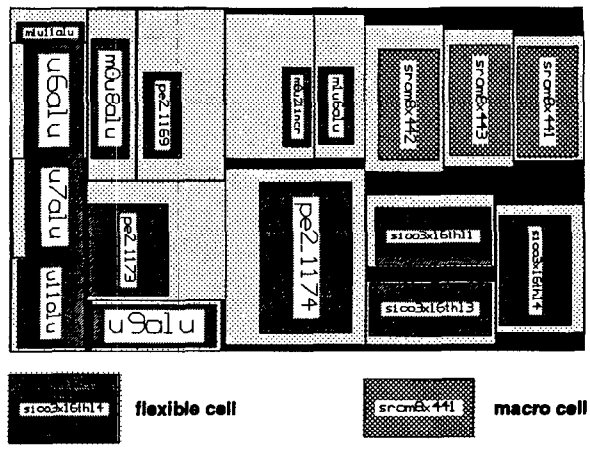

Figure 9.a: Result of Phase β
(The subcells are based on shape functions)

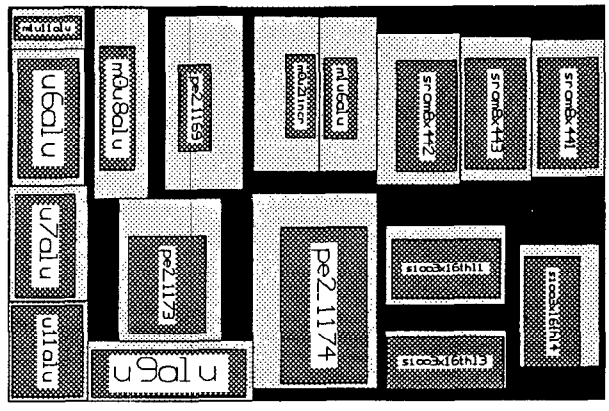

Figure 9.b: Result of the Pure Top-Down Design

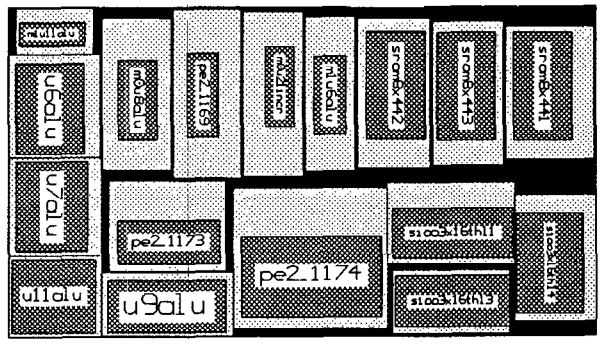

Figure 9.c: Result of the Stepwise Refinement

Figure 9: Three-Phase Planning compared with pure top-down design.

6. Conclusions

We have shown that every area estimation for a top-down planning process has a certain degree of inaccuracy which we cannot completely avoid. In this paper, we described a new top-down planning method for compensating the deviations between an estimation and the final layout. The main idea is that we make small bottom-up adjustment steps (between two hierarchy levels by using refined shape functions) within a top-down chip planning process. The convergence of the planning process for a cell is secured by the permanent refinement of the resulting floorplan descriptions during the different planning steps.

Our experiments indicated that top-down designs seem to result smaller layouts than the bottom-up approach. However, the top-down results can be improved by the presented Three-Phase Chip Planning method. The measured gain was up to 10%.

In addition, our experiments have shown that the effort for designing realistic circuits is large - even for the pure top-down approach. Using a planning method with adjustment steps, the design management will not become easier. For our large design, we used the VLSI CAD system PLAYOUT with its data management [13]. Without this support the design would not have been possible. However, all design decisions were performed by the designers. An improved design management tool (e. g. an expert system) in which we can describe complex design flows in a more general way and which gives intelligent advises for all management decisions is a main subject of our future research.

7. References

[1] W.W. Dai, B. Eschermann, E.S. Kuh, M. Pedram, "Hierarchical Placement and Floorplanning in Bear", Trans. on CAD, 1989

[2] K. Glasmacher, A. Hess, G. Zimmermann, "A Genetic Algorithm for Global Improvement of Macrocell Layouts", Proc. Int. Conference on Computer Design, Cambridge, 1991

[3] K. Glasmacher, G. Zimmermann, "Chip Assembly in the PLAYOUT VLSI Design System", Proc. European Design Automation Conference, Hamburg, 1992

[4] A. Herrigel, "GRCA: A Global Approach for Floorplanning Synthesis in VLSI Macrocell Design", Proc. Int. Conference of Computer Aided Design, 1990

[5] F.J. Kurdahi, A.C. Parker, "Techniques for Area Estimation of VLSI Layouts", Trans. on CAD, 1989

[6] D. LaPotin, S. Director, "Mason: A global floorplanning approach for VLSI design", Trans. on CAD, 1986

[7] T. Lengauer, R. Mueller, "A Robust Framework for Hierarchical Floorplanning with Integrated Global Wiring", Proc. Int. Conference of Computer Aided Design, 1990

[8] P. Marwedel, "A New Synthesis Algorithm for the MIMOLA Software System", Proc. 23rd Design Automation Conference, 1986

[9] C. Masson, R. Escassut, D. Barbier, et. al., "Object-Oriented Lisp Implementation of the CHEOPS VLSI Floorplanning and Routing System", Proc. 28th Design Automation Conference, 1991

[10] R. Otten, "Efficient Floorplan Optimization", Proc. Int. Conf. on Computer Design, Port Chester, 499-503, 1983

[11] M. Pedram, B. Preas, "A Hierarchical Floorplanning Approach", Proc. Int. Conference on Computer Design, Cambridge, 1990

[12] C. Sechen, K. Lee, "A Improved Simulated Annealing Algorithm for Row-Based Placement", Proc. Int. Conference of Computer Aided Design, 1987

[13] E. Siepmann, "A Data Management Interface as Part of the Framework of an Integrated VLSI-Design System", Proc. Int. Conference on Computer-Aided Design, 1989

[14] B. Schuermann, "A 280.00 Standard Cells Test Design - First Experiences with the PLAYOUT Top-Down Design System", Proc. Int. Workshop on Layout Synthesis, Research Triangle Park, NC, 1992

[15] B. Schuermann, J. Altmeyer, G. Zimmermann, "Three-Phase Chip Planning", technical report, University of Kaiserslautern, 1992

[16] B. Schuermann, G. Zimmermann, "Estimation of Wiring Area for Hierarchical Design", Proc. Int. Workshop on Layout Synthesis, Research Triangle Park, NC, 1992

[17] W. Swartz, C. Sechen, "New Algorithms for the Placement and Routing of Macro Cells", Proc. Int. Conference of Computer Aided Design, 1990

[18] T.-C. Wang, D.F. Wong, "An Optimal Algorithm for Floorplan Area Estimation", Proc. 27th Design Automation Conference, 1990

[19] C.-S. Ying, J.S.-L. Wong, "An Analytical Approach to Floorplanning for Hierarchical Building Blocks", Trans. on CAD, 1989

[20] G. Zimmermann, "A New Area Shape Function Estimation Technique for VLSI Layouts", Proc. 25th Design Automation Conference, 1988

[21] G. Zimmermann, "PLAYOUT - A Hierarchical Design System", Information Processing 89, G.X. Ritter (ed.), Elsevier Science Publishers B.V. (North Holland), IFIP, 1989