Towards Integration of Quadratic Placement and Pin Assignment

Jurjen Westra, Patrick Groeneveld

ES Reports
ISSN 1574-9517

ESR-2005-03
15 March 2005

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems

"Great ability develops and reveals itself increasingly with every new assignment."
Baltasar Gracian

http://www.es.ele.tue.nl/esreports
esreports@es.ele.tue.nl

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems
PO Box 513
NL-5600 MB Eindhoven
The Netherlands

Towards Integration of Quadratic Placement and Pin Assignment

Jurjen Westra Patrick Groeneveld

Eindhoven University of Technology
Den Dolech 2
Eindhoven, The Netherlands
{jwestra,patrick}@ics.ele.tue.nl

ABSTRACT

Pins serve as both the logical and physical interface between two levels in a hierarchical flow. Pin assignment is the placement of pins on the boundary of a chip or macro. Proper pin placement has a large influence on wire length. We will present experiments indicating that the spread in wire length is typically 8%, but can be more than 20% for specific cases.

To address the pin assignment problem, a modification to the well-known and widely used quadratic placement is introduced. This modification allows for the integration between pin assignment and global placement. Wire length within macros is minimized, while top-level considerations such as the relative position of macro and clusters of cells are taken into account in the form of a side assignment.

The results presented in this paper indicate that integration during the beginning of the placement phase does indeed help *on average.* The exceptions, however, show that it is still not understood *how* circuit structure and pin assignment impact wire length. It is argued that further integration such as presented in this paper is possible and necessary to obtain better wire length and thereby reducing congestion.

1. INTRODUCTION AND MOTIVATION

Pin assignment is the placement of pins on the boundary of a chip or macro. Although it is an important step during the floorplanning stage, the effect of pin assignment on placement quality has not been studied thoroughly. Most of the well-known EDA Physical Design or Floorplanning packages have some form of pin assign command, but little has been published about it.

In a hierarchical flow, pins serve as the interface between the *inside* of a macro and the *outside* of the macro at the top-level. On one hand, pin positions should be such that the smallest wire length in the macro is achieved, while on the other hand, physical top-level synthesis should not suffer from badly placed pins. The first issue is addressed by formulating internal wire length as an objective. The second issue is addressed by formulating *pin constraints.*

The contributions of this paper are as follows. First, we run thousands of experiments on a well-known benchmark suite in order to find out in how far pin assignment influences placement quality. The spread between best and worst pin assignment is typically 8%, but can be as much as 20%. Secondly, we introduce a modification to the widely used quadratic placement formulation[10, 9, 4]. The modification enables the integration between pin assignment and large-scale VLSI standard cell placement in a natural way. Finally, pin assignment experiments based on the new formulation indicate that integration does indeed typically improve pin assignment quality.

1.1 Hierarchical flow

Since the first integrated circuits in 1963, design sizes have been growing exponentially. The methodological way to deal with design sizes approaching one billion transistors is through the use of *hierarchy.* In a hierarchical flow, the design is partitioned into smaller parts in *top-down* fashion. Such a partition could be re-used in the form of hard or soft IP. Designing soft macros occurs in *bottom-up* fashion. The process of the design of different macros should be independent to enable parallelism. Then, at the floorplanning stage, the macros are "glued" together to complete the design. The pins on the macro boundaries serve as the interface between the inside and the outside of the macro, one hierarchical level up. Fig. 1 shows a typical large hierarchical design. The focus of this paper is on assigning positions to the pins of *soft macros*[1]. In this respect, a complete chip can be seen as a macro, but without the environment.

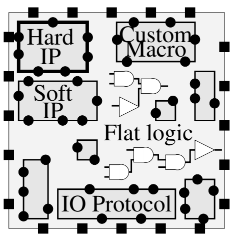

Figure 1: A typical large hierarchical design contains many macros at the boundary of the chip.

The difficulty of a hierarchical flow as sketched above is that all macros are assigned time, power and area budgets that serve as constraints for (physical) synthesis, e.g. timing assertions on the pins of a macro. Typically, iterations are necessary because too little is known at the start of the flow. An iterative methodology "weighs" between considerations at the different hierarchical levels.

In the case of pin assignment, the danger of a bottom-up flow is that synthesis assigns pins bottom-up to a position,

[1]Soft macros are macros without fixed outline or pins. They can be bought as soft IP or developed in-house.

totally ignoring the environment of the macro and the destination of the signal going through the pin (Fig. 2). This problem is solved in a two-step strategy: first top-down *side assignment* determines which side of a macro a pin will be on, and then *bottom-up* combined placement and pin assignment assigns the exact position to the pin. At any point, the pins can be *legalized* to a set of positions, and traditional global placement with fixed pins can continue. Thus, the potential for pin assignment to realize low wire lengths inside the macro is exploited, while avoiding difficulties at the top-level.

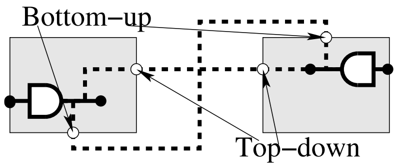

Figure 2: Bottom-up pin assignment may cause detours at the top-level.

1.2 Previous work

Pin assignment has not received much attention in the recent literature. A nice overview is given in [3] and the references therein. Experiments with an industrial (analytical) placer, where pin optimiztion is performed by reversing the role of pins and cells, and a partitioning-based placer are performed. Wire length is optimized by *alternating* between placement and pin optimization (reversing the role of pins and cells in that paper). Additionally, results of experiments with a partitioning-based placer are given. Two relatively small designs are used. In [8], pre-placement pin assignment is based on circuit structure and shown to effective for designs with up to several hundreds of cells. Pin assignment for macros only is described in [12]. The order of pins on the macro is fixed, and in-macro wire length is ignored. Much early work on pin assignment for routability has lost its importance due to the fact that aggressive over-the-cell routing is commonplace nowadays.

2. QUADRATIC PLACEMENT FORMULATION WITH PIN ASSIGNMENT

Many modern vlsi placement techniques[4, 9, 10], and also our state-of-the art placer[6] are based on the iterative solution and perturbation of a classical quadratic placement (QP) formulation[5]. In this formulation, multi-pin nets are modeled with a set of two-pin nets, usually a clique[4, 9] or a star-configuration[7, 9]. Each of the two-pin nets (i, j) has a cost, based on quadratic wire length, associated with it:

$$K_{(i,j)} = K^x_{(i,j)} + K^y_{(i,j)} \text{ with } K^x_{(i,j)} = \frac{1}{2} w_{i,j}(x_i - x_j)^2, \quad (1)$$

and $K^y_{(i,j)}$ similarly. i and j are the indices of the cells, x_i and y_i represent the x- and y-positions of the cells or fixed pins, and $w_{i,j}$ the weight of the net. The sum over all nets can be written in matrix notation:

$$K_{tot} = K^x_{tot} + K^y_{tot} \text{ with } K^x_{tot} = \frac{1}{2}\mathbf{x}^T C\mathbf{x} + \mathbf{f_x}^T\mathbf{x} + c_x, \quad (2)$$

and K^y_{tot} similarly. The vectors $\mathbf{x}$ and $\mathbf{y}$ hold the x- and y-coordinates of the movable cells. The square $n \times n$ positive-definite matrix C (with n the number of movable cells) and the vector $\mathbf{f_x}$ hold the connectivity, and c_x is a constant due to fixed pins. Two-pin nets between cells i and j with weight w "stamp" patterns in the matrix and vectors: w is added to $C(i, i)$ and $C(j, j)$, and $-w$ to $C(i, j)$ and $C(j, i)$. A connection between movable cell i and a fixed pin at (x_f, y_f) adds w to $C(i, i)$, $-2wx_f$ to $\mathbf{f_x}(i)$, and $-2wy_f$ to $\mathbf{f_y}(i)$. The total cost is minimized[5] by solving the system

$$\begin{aligned} C \cdot \mathbf{x} + \mathbf{f_x} &= \mathbf{0} \\ C \cdot \mathbf{y} + \mathbf{f_y} &= \mathbf{0}. \end{aligned} \quad (3)$$

The problem above is equivalent to calculating the equilibrium of a spring system, where all two-pin nets represent springs. Note that without fixed pins the system would have a trivial solution at $\mathbf{0}$. As noted in [3], this makes it hard to integrate pin assignment with QP-based placement.

As illustrated by Fig. 3, the solution of Eq. 3 typically yields considerable overlap. To reduce overlap, small perturbations, usually based on (local) cell densities are made to $\mathbf{f}$ and/or C[4, 9], and with the new values new cell positions are calculated. This process continues until a stop criterion (usually based on the amount of overlap) is met. The remaining overlap is removed by detailed placement (an entirely different algorithm).

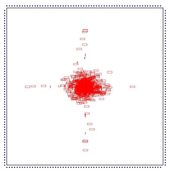

Figure 3: Typical situation after pure quadratic wirelength minimization.

2.1 Unfi xingthe pins

As outlined in [3], integrating pin assignment with *partitioning-based* placement follows naturally, but QP needs fixed pins to avoid the trivial solution. Pin assignment can be viewed as *one dimensional* placement, while the placement problem as described above is *two dimensional*. Looking at Eq. 3, one can see that during cell placement, the problem is solved *for each dimension separately*. This suggests that it is possible to treat a pin as a cell and include it during placement in one dimension, and treat it as in the original formulation for the other dimension.

Fig. 4 illustrates our solution. During the calculation of x-coordinates, the pins that can move *horizontally* (the North and South pins) are "unfixed"[2]. The pins with fixed x-coordinates (the East and West pins) now span the system. When the y-positions are calculated, the North and South pins are fixed, and the East and West pins are unfixed in turn.

Essentially, the system that is to be solved becomes

$$\begin{aligned} C_x \cdot \mathbf{x}' + \mathbf{f}'_\mathbf{x} &= \mathbf{0} \\ C_y \cdot \mathbf{y}' + \mathbf{f}'_\mathbf{y} &= \mathbf{0}. \end{aligned} \quad (4)$$

C_x, C_y, $\mathbf{x}'$, $\mathbf{y}'$, $\mathbf{f}'_\mathbf{x}$ and $\mathbf{f}'_\mathbf{y}$ are slightly larger than their counterparts in Eq. 3 since they must also accomodate the now unfixed pins. For example, $\mathbf{x}' = [\ \mathbf{x}^T \quad \mathbf{x_{uf}}^T\ ]^T$, where $\mathbf{x}$ holds the x-coordinates of the cells as before, and $\mathbf{x_{uf}}$ holds the

[2]Unfixing means the pin's net is first removed from C and $\mathbf{f}$, and then added again with the difference that the pin is treated as a movable cell.

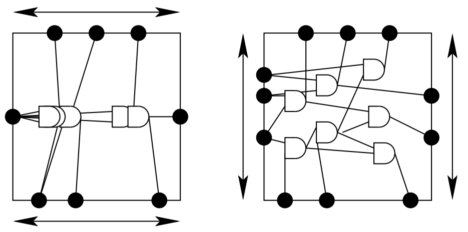

Figure 4: Initially, East and West pins are fixed, while North and South pin positions are unfixed and optimized together with the cell positions. Next, the roles are reversed.

x-coordinates of the (for this dimension) unfixed pins. Typically, $|\mathbf{x_{uf}}| << |\mathbf{x}|$. C_x and C_y can be calculated from C by adding/removing the few stamps involved in fixing/unfixing pins, and $\mathbf{f_x}$ and $\mathbf{f_y}$ can be precalculated. Since the dimension of the problem changes only little, runtime and memory overhead should be small.

2.2 Pin legalization

Equivalently to cells, pins are modeled as points in our combined placement and pin assignment formulation. Near the boundary of macros, where the pins are located, congestion and layer-changes are more likely. A bit of room around the pins is useful and can be obtained by specifying a *pin pitch*, yielding a limited set of positions available to pin assignment. Due to the large size of the pins of complete chips (*pads*), only a limited number of positions is available. Cells should be spread over the macro area, so equivalently, the pins connecting them to the outside world should be spread over the macro boundary. If the pins spanning the system are not well-spread, it becomes difficult for the placement algorithm to spread the cells over the placement area. If more positions than pins are available, a well-spread subset should be chosen. The remaining positions can be used to fine-tune the pin assignment *after* placement[3].

In our legalization scheme, we assign the pins to predefined positions, derived from a pin pitch or a limited set of available pad positions. Key is to respect the order they are in after the combined pin assignment-placement phase (Fig. 5). Obviously, such a legalization scheme is common-sense, but it also makes sense in light of the following observation. If pins swap, cells connected to them may also have to swap relative positions for an optimal solution. However, it is well-understood that QP-based methods have difficulty with swapping, so (most likely), longer wire length will result.

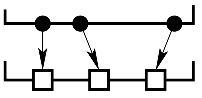

Figure 5: Pins are legalized to predefined positions according to their order.

3. RESULTS

In our experiments, the well-known ISPD02 placement benchmarks[1] were used. Unfortunately, this is a suite with large macros that is unsuitable for standard cell placers. In [11], this problem is tackled by removing pins and macros[3], something that obviously cannot be done with pin optimization. Instead, the approach of [9] is taken: the height of the macros is scaled to the height of one cellrow, and the width to four times the average cell-width. Although the size of the macros have been altered, the benchmarks suite remains a *mixed size* benchmark suite. The benchmarks are ported to a standard cell library that comes with our EDA software, and everything is scaled such that the row utilization is 90%. The benchmark suite is summarized in Table 1.

[3]The predecessor of our benchmarks suite is used in [11].

Table 1: The benchmark suite

Chip	cells	pins	nets	Chip	cells	pins	nets
ibm01	12506	246	14111	ibm10	68685	744	75196
ibm02	19342	259	19584	ibm11	70152	406	81454
ibm03	22853	283	27401	ibm12	70439	637	77240
ibm04	27220	287	31970	ibm13	83709	490	99666
ibm05	28146	1201	28446	ibm14	147088	517	152772
ibm06	32332	166	34826	ibm15	161187	383	186608
ibm07	45639	287	48117	ibm16	182980	504	190048
ibm08	51023	286	50513	ibm17	184752	743	189581
ibm09	53110	285	60902	ibm18	210341	272	201920

3.1 Experimental setup

In order to obtain statistics about the effect of pin assignment on wirelength, 100 different random pin assignments were created for each design in the benchmark suite. The positions of random pins on the same side were swapped, and a benchmark suite of 1800 designs was created[4]. Note that 100 is small in comparison to the amount of permutations ($p!$ for p pins), but running more experiments was not feasible because of limited compute power. Wire length was measured after global and detailed placement in a normal flow.

For each benchmark, also a pin assigned version was created with the flow of Fig. 6. Because we did not have access to the source code of a state-of-the-art QP-based placer[5], we applied the modified formulation only to the first iteration and fed the result to our industrial placer. The first iteration is equivalent in all QP-based placers, but we do *not* achieve full integration this way. We did not strictly follow the scheme of Sec. 2. The position of an optimized pin is always the projection of the cell it is connected to on the pin's side. Effectively, by unfixing the pin is removed from the netlist. The pin is fixed at the projection of the center-of-gravity of the remaining cells on the pin-net[6], thus optimizing quadratic wire length. This method is less error-prone and saves optimization variables. Pin legalization followed as described, and the normal placement flow was employed to aquire wire length results.

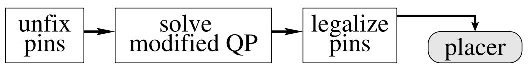

Figure 6: The placement flow with pin assignment.

In all experiments, the `query measure wirelength` command[6] was used to measure wire lengths. This command is an accurate estimate for post-routing wire length and uses both bounding box and pin counts.

[4]Actually, during this project even more benchmarks were created, especially for the smaller designs. All observations were similar.

[5]Recently, the source code of [10] became available.

[6]All but few pin-nets are two-pin nets. In that case, the entire net is removed.

3.2 Random pin assignment statistics

The difference between the best found and worst found pin assignment for a design is called the *spread* of the design. The larger the spread, the greater the importance of good pin assignment. Fig. 7 shows the spread for the benchmarks after global placement. On average, the spread is 8%, but it can be as large as 20%. These numbers should be considered large since the same placer is used on exactly the same benchmarks except for the pin positions. In congested designs, differences in the order of a few percent can make or break routability. This study proves that pin assignment is an important consideration.

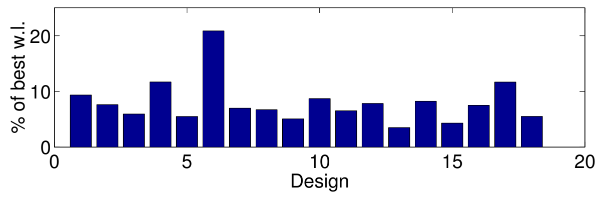

Figure 7: The *spread* of the designs as a percentage.

The wire length distribution of a design indicates how likely it is that a random pin assignment will result in good wire length. Because of space limitations, we can not print all the distributions, but Fig. 8 is examplary for the results: with a bit of fantasy the familiar bell-shape of a normal distribution can be recognized in the distribution on the left. As shown on the right, there are examples with a large "tail" on one side. One hundred experiments per design is perhaps too little to draw definitive conclusions, but the results indicate that a random pin assignment has the largest probability to be close to the middle between the best and worst possible.

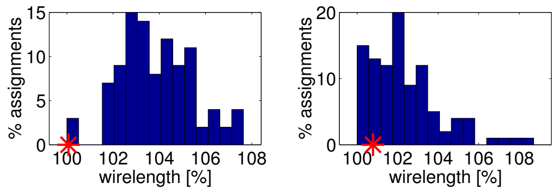

Figure 8: Wire length distribution of ibm02 (left) and ibm10 (right) after global placement. The red starts denote the wire lengths obtained with our pin assignment method.

3.3 Bottom-up pin assignment results

Fig. 9 shows the results for the pin assignment experiment. If a design has a result close to 0%, this means that the pin assigned wire length is close to the best found wire length (e.g. ibm02). Equivalently, if the result is close to 100%, the pin assigned wire length is close to the worst found wire length. For most designs, the results are (very) good, or average, but for a few designs, the results are not so good. It can be seen however that typically pin assignment improves wire length. On average, the pin assigned result is at 38% of the spread, which is much closer to the best pin assignment than to the worst assignment. Since including pins in the QP formulas is relatively cheap in terms of runtime, including pin assignment as described may be worth it[7].

[7]Contrary to for full integration, the described method does not need adjustment and tuning of the placement engine.

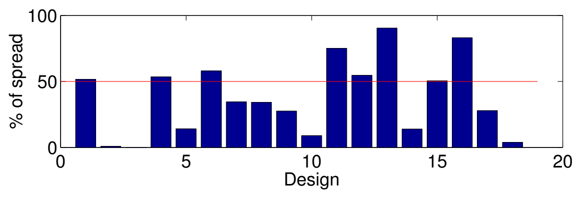

Figure 9: Wire length results for the pin assigned designs as a percentage of the spread.

3.4 Detailed placement results

The results presented so far are based on wire length after global placement. Detailed placement has totally different algorithms that act more *locally* than global placement. The primary goal of detailed placement is removing overlap, but meanwhile both wire length and congestion are minimized as well. Potentially, the lower wire lengths could be achieved at the cost of congestion, i.e. good pin assignment improves wire length estimates by making the bounding boxes of the nets small, but the design is not routable, or needs large detours, because of congestion. Our experiments indicate that this is not the case. The wire length distributions are similar to the ones found after global placement. The spread is 9% on average (8% for global placement), and the pin assigned results are similar at 40% of the spread (38% for global placement).

4. DISCUSSION AND FUTURE WORK

The experiments show clearly that pin assignment deserves more attention than it currently receives. On average, the difference in wire length between a good and a bad pin assignment is about 8%, but in extreme cases, it may be more than 20%. The wire length distribution pictures show that a more or less random pin assignment will most likely yield an average wire length, leaving significant room to improve. On average, the tested method produces better-than-average wire lengths, but this is not the case for all benchmarks.

4.1 Spread

The design with the largest spread is ibm06. This also happens to be the design with (by far) the lowest pin count. Other designs with large spread are ibm04 and ibm17. The first does not have an out-of-the-ordinary pin count, and the second has in fact a quite *large* pin count. Also looking at the designs with *low* spread, we conclude that (perhaps surprising), pin count has little to do with the spread.

Another interesting conclusion that can be drawn from Fig. 7 is that spread *does not depend on circuit size.* One would perhaps expect that larger designs are less sensitive to pin positions since the number of pins is lower relative to the number of cells. According to the experiments, this is not the case. Except perhaps for much smaller designs, the experiments indicate that pin assignment is equally important for designs of any size.

4.2 Pin assignment results

The pin assignment experiment does not consistently outperform the average, but on average it is better. Ideally, we would fully integrate pin assignment and placement, but as an alternative, the pins are *pre-placed* based on knowledge about the placement algorithm. In fact, the first placement iteration is performed twice, once by the pin assigner, and once by the actual placement engine[8]. In the tested method, only quadratic wire length is taken into account, and important issues such as overlap, net-weights and cell-sizes are totally ignored. Since basically only circuit-structure is taken into account, much randomness in the results is to be expected.

Comparing Fig. 9 and Table 1, there seems to be little relation between the success of our approach and pin counts. Most of the poor results are achieved with the larger benchmarks, indicating that size may be of importance here. We noted during the experiments that the global placer needed more iterations than usual, especially on the larger benchmarks. In this case, initial co-placement of pins and cells has less influence. If more iterations are needed, spreading the cells over the placement area is apparently difficult. If the "spreading" objective dominates wire length, it is to be expected that a method that focusses entirely on wirelength is less effective. Perhaps, for "easy" designs, the method is more effective than for "difficult" designs. Note that the method was tested on difficult designs with a utilization of 90% only.

In the method, pins are legalized. As a consequence, many pins are moved relatively large distances. If initially the pins are more "spread out" over the macro's boundary, pin legalization has less influence. As shown in Fig. 10, pins are clustered around the center of their respective side right before legalization. We visually inspected many of those pictures, and found little or no correlation between the effectivity of the approach and initial pin spreading.

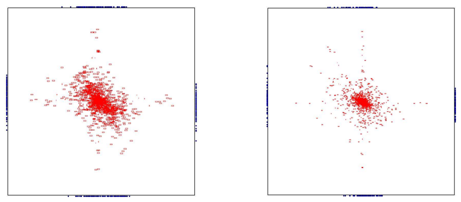
Figure 10: Initially, the pins of ibm11 (left) and ibm18 (right) are equally spread over the chip boundary, but eventually pin assignment proves to be far more effective for the second than for the first benchmark.

Comparison with the results of [3] is difficult since the spread of those benchmarks is not known. We note that improvements of 14% and 5% are achieved, but the designs are much smaller and initial pin placement quality is not known.

In short, instead of fully integrating pin assignment in a placer, the described method might be useful. On average, wire length is improved, but the produced results should not be considered "sacred" since the method may produce suboptimal results. Also, post-placement pin optimization has been shown to be effective in reducing wire length further[3].

4.3 Side Assignment

The modified quadratic placement formulation focusses on minimization of wirelength *within* a macro. It is based on a *side assignment* that should reflect the position of the macro in its environment. A side assignment can be obtained by simply unfixing the pins from a traditional pin assignment flow[9], but it is preferable to have methods to automatically extract the side assignment from the positions of macros and clusters of cells, information that is available after floorplanning.

[8]There is a (small) difference: during pin assignment, the pins are unfixed, while during placement the pins are fixed at optimized positions.

As shown in Fig. 2, the main issue side assignment has to address is top-level wire length. The main idea is that if nets connect nearby macros, pins should be placed on the nearest sides. Sometimes a pin has multiple possibilites, and in this case a decision has to be made. The first proposed algorithm bases this decision on *distance*, and the second proposed algorithm bases it on *abuttability*.

Minimum Spanning Tree Side Assignment

The side assignment of pins on macros should depend on the environment of the macro. The lower bound on top-level wire length is the minimum rectilinear steiner tree length, but a good approximation is Minimum Spanning Tree (MST) length. Obviously, exact positions of pins are not determined yet, but as a first approximation, they are placed *at the center* of their macro (Fig. 11). Approximate positions of cells at the top-level and macros should be known after floorplanning, and based on these positions, an MST is calculated. From the positions where MST edges cut macro boundaries side assignments are derived. More than one side may be cut for a pin (denoted by the question mark in Fig. 11). This is a source of freedom that may be used to *balance* pin counts on different sides of a macro. The exact positions of the cut may also be used as the starting point in an alternating pin assignment-placement flow such as in [3], perhaps after pin legalization.

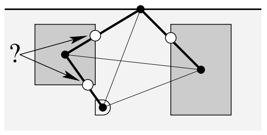

Figure 11: MST edges determine the side assignment.

MST side assignment makes sure that potentially close pins will indeed end up close to eachother. The method is *flexible* in the sense that designers can easily fix pins or remove potential MST edges in order to steer the algorithm.

Abuttability-based Side Assignment

If two pins on a net can potentially be placed on facing macro boundaries, they are called *abuttable*. Abuttability-based side assignment *maximizes* the amount of actually abutted pin-pairs as illustrated in Fig. 12. In an *abuttability graph*, each node represents a pin and an edge exists between abuttable pins[10]. The problem is to find the largest subset of edges such that no edge in the subset is adjacent to the same node as another edge in the subset, since edges represent abuttable pin-pairs, and no pin can be abutted to more than one other pin. This problem is known as a *maximum cardinality matching*[2] problem, and is easily solved.

Matched pin-pairs will be connected by a wire. Unmatched pins in turn can abut to these wires again (Fig. 12, right). Alternatively, pins can be assigned to the side closest to the center-of-gravity of the net.

[9]In many modern floorplanning packages, pins are initially assigned to random positions.

[10]If pins are not allowed on a side, or may not abut to some other pin, this is easily incorporated

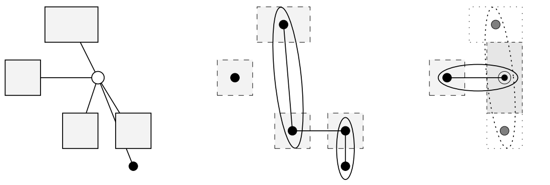

Figure 12: Abuttability of pins (left) is translated into an *Abuttability Graph* (center). A *Maximum Matching* algorithm selects edges, yielding a side assignment. In a next step, pins can also be abutted agains *wires* created for a selected edge in the previous step (right).

Weights can be used to steer the matching process. They can be set by a designer, or can be based on distance such as in MST side assignment. In all cases, the matching problem can be solved efficiently[11] in $O(|V||E|)$[2].

Finally, another application of our abuttability-based method is shown in Fig. 13. On the left, the pins are ill-alligned, creating longer wires and making the design more congested. By calculating the maximal matching of the abuttability graphs, the situation on the right, with lower wire length *and lower congestion* can be created.

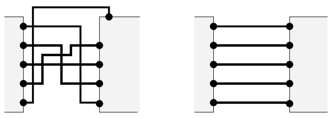

Figure 13: Using abuttability can solve detours and congestion problems.

4.4 Further integration

The results in the previous section clearly show that further integration between placement and pin assignment is necessary to obtain better, and more stable results. The modified QP formulation as presented in this paper enables this. A possible global placement flow with pin assignment is shown in Fig. 14. It will probably be useful to fix the pins at one point since convergence is questionable if both pins and cells are allowed to change position. Preliminary results with a placer inspired by [9] indicate that with unfixed pins, spreading goes faster. The results of [3] indicate that further integration will lead to better wire length. The authors achieve integration by alternating placement and pin assignment. Up to 14 iterations are necessary to achieve the optimal result, and oscillations are common. With large designs, such an approach is clearly undesirable.

When implementing a pin assigner/placer based on the described ideas, it is important to realize placers require a lot of tuning. Current implementations are based on problems with *fixed* pins that span the system. Through Rent's rule, the number of pins is related to the problem size. In the modified formulation, about half of the pins is unfixed during the calculation of x- and y-positions, making new tuning necessary.

5. CONCLUSIONS

In this paper, it was shown using several thousands of large standard cell placement benchmarks that pin assignment has a major impact on wire length. A modification to

[11]Pin counts are relatively low, so this is no real issue.

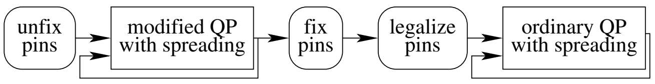

Figure 14: With the modified formulation, tighter integration between pin assignment and placement is possible.

the well-known and widely used quadratic placement formulation that enables full integration between pin assignment and global placement was introduced. Wire length within macros remains the primary objective, but the relative position of macros is taken into account through side assignment of pins.

Experiments on a flow where pin assignment is based only on the first iteration of any QP-based placer, indicate that on average wire length improves. The method does not consistently outperform the average, which is probably due to the fact that only a single iteration is integrated and the designs are difficult. No single factor seems to predict the results, except perhaps circuit size. Better understanding of the influence of pin assignment on wire length is necessary, but our guess is that only further integration will guarantee good and stable results. The methods and formulas presented in this paper can be applied and integrated with many academic and industrial global placers.

6. REFERENCES

[1] S. Adya and I. L. Markov. Combinatorial techniques for mixed-size placement. *Trans. Design Automation of Electronic Systems*, 2004.

[2] N. Blum. Maximum matching in nonbipartite graphs without explicit consideration of blossoms. Technical report, Universitat Bonn, 1999.

[3] A. Caldwell, A. Kahng, S. Mantik, and I. Markov. Implications of area-array i/o for row-based placement methodology. 1998.

[4] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. Design Automation Conference*, 1998.

[5] K. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17, 1970.

[6] Magma Design Automation. *Blast Chip 4.0 User Guide.*

[7] F. Mo, A. Tabbara, and R. K. Brayton. A force-directed macro-cell placer. In *Proc. International Conference on Computer-Aided Design*, 2000.

[8] M. Pedram, K. Chaudhary, and E. S. Kuh. I/o pad assignment based on the circuit structure. In *Proc. International Conference on Computer Design*, 1991.

[9] N. Viswanathan and C. C.-N. Chu. Fastplace: efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In *Proc. International Symposium on Physical Design*, 2004.

[10] K. Vorwerk, A. Kennings, and A. Vannelli. Engineering details of a stable force-directed placer. In *Proc. International Conference on Computer-Aided Design*, 2004.

[11] M. Wang, X. Yang, and M. Sarrafzadeh. Dragon2000: standard-cell placement tool for large industry circuits. In *Proc. International Conference on Computer-Aided Design*, 2000.

[12] X. Yao. A new approach to the pin assignment problem. In *Proc. Design Automation Conference*, 1988.