NRG: Global and Detailed Placement

Majid Sarrafzadeh Maogang Wang
Department of Electrical and Computer Engineering, Northwestern University
majid@ece.nwu.edu

Abstract

We present a new approach to the placement problem. The proposed approach consists of analyzing the input circuit and deciding on a two-dimensional global grid *for that particular input.*

After determination of the grid size, the placement is carried out in three steps: global placement, detailed placement, *and* final optimization. *We will show that the output of the global placement can also serve as a fast and accurate predictor. Current implementation is based on simulated annealing.*

We have put all algorithms together in a placement package called NRG (pronounced N-er-G). In addition to area minimization, NRG can perform timing-driven placement. Experimental results are strong. We improve TimberWolf's results (version 1.2, the commercial version which is suppose to be better than all university versions including version 7) by about 5 %. Our predictor can estimate the wirelength within 10-20 % accuracy offering 2-20x speedup compared with the actual placement algorithm.

1 Introduction

Performance-driven, issues related to deep submicron, reliability and many more factors make the placement problem a real challenge. In this paper we show that even the classical (area minimization) placement problem is yet unsolved. We demonstrate some general issues that need to be considered in designing a placement algorithm. We will make use of them in designing a new placement package called NRG.

In addition to good placement algorithms, we need good predictors that can quickly estimate the placement cost function.

Numerous approaches to the placement problem has been proposed. For example, force directed [2, 7, 16] simulated annealing [11, 13], and partitioning-based placement [4, 3, 6]. A constructive placement method that employs resistive networks as a working domain was proposed in [5]. In [15], there is a comparison between linear and quadratic cost functions. The linear cost function used in Gordian**L** placement tool achieves results with up to 20% less area than the quadratic cost function of the original Gordian procedure. Therefore, in this paper, we will use the half-perimeter cost function for area minimization.

Timing-driven placement problem has also been studied. The notion of zero-slack was introduced in [9]. In [8] criticality was used as a guide to select cells from a cell library. For a history of the timing-driven placement problem see [1, 14, 10].

In an attempt to simplify and possibly speed up the placement process, this paper presents a new formulation that splits the domain of the placement problem. We define two new problems known as the global placement and detailed placement problems. The intuition behind our strategy is that we should be able to quickly determine a good global placement of gates for a circuit without devoting too much attention to details. After performing the global placement, detailed placement can concentrate further on the exact position and timing of a gate given that its final general vicinity is favorable and known.

One of the main issues in performing global placement is determination of the global grid size. We will show an arbitrary grid in the global placement problem is not necessarily good. Indeed, we show examples where a wrong choice of the grid produces very bad placement results. We propose an input (circuit) analysis scheme, based on breadth-first search of the circuit, followed by a correlation analysis to decide on the size of the global grid.

Once the grid size has bee determined global placement phase starts. by performing iterative improvement. Finally, a detailed placement followed by local optimization is performed.

This paper is organized as follows: In Section 2 we introduce some terminology and modeling of the problem. In Section 3 a motivating example is provided followed by a summary of the proposed NRG placement tool in Section 4. In Section 5, details of the proposed placement tool is discussed followed by experiments in Section 6. We conclude the paper in Section 7.

2 Terminology

In this paper we assume that we are given a synchronous circuit denoted $\mathcal{C}(\mathcal{M},\mathcal{N})$, which consists of a set of modules $\mathcal{M} = \{M_i | i = 1, \ldots, |\mathcal{M}|\}$, and a set of nets $\mathcal{N} = \{N_i | i = 1, \ldots, |\mathcal{N}|\}$. The set of modules consists of four subsets: primary inputs $\mathcal{I}$, primary outputs $\mathcal{O}$, storage elements $\mathcal{R}$, and combinational elements $\mathcal{A}$, i.e., $\mathcal{M} = \mathcal{I} \cup \mathcal{O} \cup \mathcal{R} \cup \mathcal{A}$. A sample circuit is given in Figure 1.

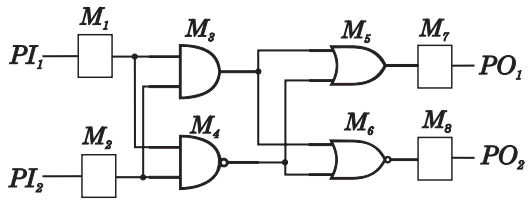

Figure 1: Sample circuit.

For this work, we use a row-based placement model, which cover standard cell and sea of gates layout styles. The concepts developed here can be applied to other styles, however, we have not explored them yet.

3 A Motivating Example: Linear Placement

Consider the linear placement problem as shown in Figure 2a. Consider a circuit $\mathcal{C}_{lin}$ containing n modules. For simplicity, assume all modules have the same width. Size of the solution space is $n!$, for there are $n!$ permutations of the modules.

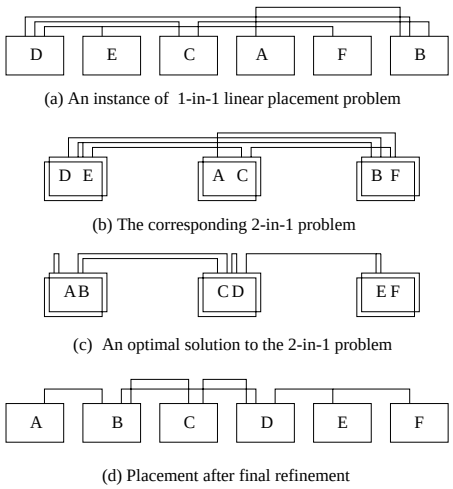

Figure 2: Example of the linear placement problem.

Now, let's conduct an experiment. Suppose that instead of n positions for the modules, we can use only $n/2$ positions, with exactly two modules to each position, as shown in Figure 2b. We will call the new problem the 2-in-1 problem (meaning 2 modules in one position) and the original problem 1-in-1. The objective is to solve the 2-in-1 problem optimally as shown in Figure 2c. Then refine it, that is, to "locally" move the modules to their final position, as shown in Figure 2d. The hope is that the two step approach can find a good solution to the original problem. Thus, we need to investigate the size of the reduced solution space and its quality.

Fact 1 *The search space in the 2-in-1 problem is reduced by a factor of* $2^{n/2}$ *with respect to the original 1-in-1 problem.*

Proof: There are a total of n modules. In the 2-in-1 problem $\binom{n}{2}$ of them can be placed in the leftmost position. Once that is decided, we are left with $n-2$ modules. Thus, $\binom{n-2}{2}$ modules can be placed in the second position and so on. Thus, the total number of solutions is:

$$\binom{n}{2} \times \binom{n-2}{2} \times \binom{n-4}{2} \times \ldots \times \binom{2}{2}$$

$$= \frac{n \times (n-1)}{2} \times \frac{(n-2) \times (n-3)}{2} \times \ldots \frac{2 \times 1}{2}$$

$$= \frac{n!}{2^{n/2}}$$

□

Given that the solution space is reduced by a "large" factor, there is a hope to obtain an optimal solution to the 2-in-1 problem very quickly and then refine it to obtain a good solution to the original problem. We call the 2-in-1 problem the *global placement problem* (GP) and the subsequent refinement, the *detailed placement problem* (DP) and the combined step the global-detailed placement (or GDP) problem. The main question that remains is the quality of GDP. To answer this question, we have done experiments on a number of benchmarks. Lets look at the MCNC benchmark *highway2*. Results are summarized in Table 1. For quick prototyping, all methods used in the table are simulated annealing (SA) based. In summary, Table 1 shows that in 4.1 seconds (1.4 plus 2.7) we can obtain results that are of comparable quality to the best solution. Therefore, **we can reduce the search space without sacrificing the quality.** Certainly, this is just one small example. We will show in later sections that that same behavior can be observed in all benchmarks.

Method	WireLength	RunTime (sec)
1-in-1	14712	18
2-in-1 GP	15065	1.4
DP	14790	2.7

Table 1: Experiments with highway2 (62 cells).

Note that we can generalize the idea to 3-in-1 GP, 4-in-1, etc. Results of quality versus GP size is summarized in Table 2. As can be observed, not every GP size is suitable. Indeed, some produce very poor results. We will make use of this observation in the NRG placement algorithm.

	2-in-1	4-in-1	8-in-1	16-in-1
GP	15065	15887	19535	19385
Time	1.4	1.3	1.3	1.1
DP	14790	15556	20180	17940
Time	2.7	2.8	3.1	3.3

Table 2: Various GP grids in highway2.

4 Summary of NRG

The main objective of global and detailed placement (GDP) is to reduce the search space and thus obtain better results faster. Two major advantages are:

- It will be much faster to get a similar (or better) placement results.
- It can be used as a very good placement predictor. This feature will be very useful when used in a logic-synthesis or high-level synthesis environment.

We divide the circuit placement problem into five phases: input analysis, grid determination, GP, DP and final optimization. In the next section, we will elaborate on each step.

5 Details of NRG

In this section, we will discuss each phase of NRG.

5.1 Input Analysis

Consider a circuit $\mathcal{C}$. It can be modeled as a hypergraph $G_{\mathcal{C}}$: Vertices associated with primary inputs of $\mathcal{C}$ are called *PI vertices*. *PO vertices* are symmetrically defined. We perform a breadth-first search (BFS) of $G_{\mathcal{C}}$: PI vertices will be placeed in level 1, all vertices connected to vertices in level i will be placed in level $i+1$. *Level number* of a circuit $\mathcal{C}$ is the number of levels in BFS of the corresponding graph $G_{\mathcal{C}}$. Intuitively, given two circuits with the same number of modules, the one with a higher level number is less tightly connected and thus easier to place. To account the number of vertices, we define the *BFS number* of $G_{\mathcal{C}}$ as n divided by the level-number, where n is the total number of modules. BFS number of a circuit will be used in determination of the GP grid.

5.2 Determination of the GP Grid

As observed before (in the linear placement experiment), size of the global grid makes a big difference in the result. Lets look at a similar experiment done with Primary1. As shown in Table 3 different GP grids produce very different placement results. Note that we have run the algorithm very fast to obtain these results. Thus, the final results is not as good as the best we have obtained. The point is, even in this case, we can get a sense of which grid size can potentially produce good solutions. As can be seen, GP grids of size 16×6 and 16×8 produces good placement results.

	2×2	4×4	8×8	16×2
GP	838800	919700	968700	870800
DP	1373500	1091600	1062600	1143800
	16×4	16×6	16×8	16×10
GP	925400	963700	970900	991600
DP	1032100	982700	982500	1004200
	16×12	16×14	16×16	16×18
GP	1022300	1022200	1033400	1037200
DP	1025700	1029000	1011000	992600

Table 3: Grid sizes in GP for Primary 1.

BFS number suggests whether a larger GP grid is most suitable for the problem or a smaller GP grid. To find the exact grid to choose, we perform a *correlation analysis*. It basically determines the how well is the cost function obtained during GP is correlated to DP. The difference of wirelength between GP and DP is shown is Figure 3. As can be seen the Δ function is convex. Therefore we can perform a binary search to find the right grid size. To find Δ for a given grid size, we perform a very fast GP followed by a very fast DP. Since simulated annealing is our current implementation, so a "fast" GP or DP can be easily obtained by reducing the number of random moves at each tempture.

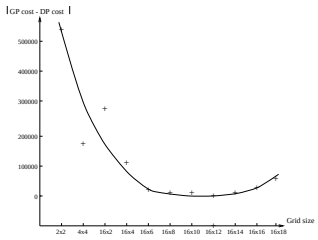

Figure 3: Correlation between GP and DP in Primary1

5.3 The Global Placement Problem

Consider an $a \times b$ GP grid imposed on the chip area. A total of ab disjoint *bins* is obtained,

A good solution to the Global Placement Problem will manage to keep a fairly even distribution while minimizing the given cost function.

In this paper we propose the following **Global Placement Problem:**

Given a standard cell circuit and a GP grid, assign the modules to GP regions so as to minimize a cost function based on the wirelength Furthermore, assure that the modules are distributed evenly among GP regions.

These two factors can be combined and expressed as a singular objective cost function:

$$f(x) = P_{WIRELENGTH} + \lambda P_{GBCD} \qquad (1)$$

where x is a certain placement needs to be evaluated, $P_{WIRELENGTH}$ is the sum of the bounding box of all nets and and P_{GBCD} is the GP region distribution penalty. λ is a simple scaling factor.

We chose to use a simulated annealing approach in iteratively improving the GP results. We defined our move set as follows: pick two cells in two different bins and evaluate the cost of swapping them with each other. We set all parameters using trial-and-error and making use of published results, e.g., [12].

5.4 The Detailed Placement Problem

Given a "good" global placement, we must then create a "good" final placement. This step is called the detailed placement stage. Its goal is to take the global bin assignments and produce a legal placement while trying to minimize its objective function. We implement DP using a low tempreture annealer. The objective function is a combination of wirelength, overlap penalty and row penalty. Contact the anthor for more details.

5.5 Final Optimization

Althouth we put the overlapping penalty in the cost function of DP, there still might be some overlapping exist in the final placement of DP. So the first thing we need to do is to perform a "clean up" operation to shift cells to eliminate overlapping. After this is done, we flip the cells. We chose to implement the same algorithm reported in [11]. For details of this phase, see [11].

5.6 Comparison of NRG with Other Placement Algorithms

TimberWolf's commercial placement tool has a hierarchical methodology which shows more or less the similiar idea as NRG. The similarity between NRG and TimberWolf's hierarchical placement is: *Both algorithms try to simplify the placement problem by trying to determine an approximate location for each module first.*

Although they have the same goal in general, they take different approaches. First of all, the major difference lies between the idea of *global* vs. the idea of *clustering*. Hierarchical placement is based on the idea of clustering, i.e., by combining a certain number of cells to form a bigger cell(cluster) to reduce the problem size. The main problem is once a cluster is formed in an iteration it cannot be broken during an iteration. Thus, modules do not have the required moving flexibility. So the problem arises that a move might be good for only some modules in that moving cluster but In NRG, we do not cluster the modules. So, potentially, a move takes place only if it is good for all modules.

6 Experimental Results

NRG placement tool were implemented in C++ and the experiments were run on a Sun Spark-20 workstation. Five MCNC benchmark standcell circuits are used for experiments. Table 4 lists the characteristics of all 5 test circuits.

TestCase	# Cells	# IO	# Nets	# Rows
highway2	62	11	87	4
fract	125	24	163	6
struct	1888	64	1920	16
Primary1	833	107	1266	16
Primary2	3014	132	3817	32

Table 4: Testing benchmark circuits.

The first experiments we did was to compare NRG with TimberWolfSC1.2.6 (the commercial version) us-

ing the half-perimeter cost function. We ran TimberWolf three times on each benchmark and have reported the best result of the three runs in Table 5. NRG results have been obtained also by running the algorithm three times. Since, NRG also use simulated annealing in parts of it, different outputs is obtained in different runs. However, NRG's output is more deterministic than that of TimberWolf. Results are summarized in Table 5. As can be seen, NRG produces results better than that of TimberWolf in most cases. This is a surprising result, given that TimberWolf is a "mature" tools and has many details not yet implemented in NRG. It shows that better algorithms can improve the placement quality.

Test Case	TW	NRG	% improvement
highway2	9221	8646	6.2%
fract	26773	25602	4.4%
struct	314762	287631	8.6%
Primayr1	900745	894545	0.7%
Primary2	3401378	3412195	-0.3%

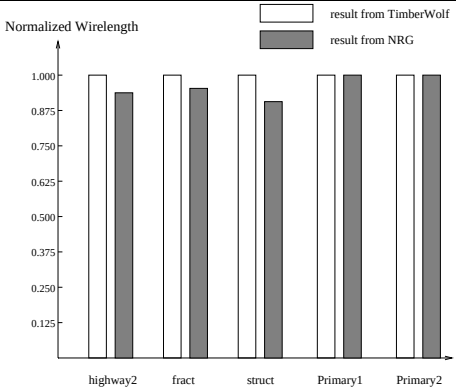

Table 5: Wirelength result: NRG vs. TimberWolf 1.2.6

Table 6 shows how well GP can predict the cost function. As can be seen the prediction error is within 3-20%. Therefore, in addition to providing a good input to DP, GP can be used as a good predictor (e.g, in a high-level synthesis system).

Table 7 shows the speed-up offered by GDP versus the flat approach and the quality tradeoff. In summary, by sacrificing 1-3 % in quality a speed-up of 2x is obtained.

7 Conclusions

We proposed a two-step (GP followed by DP) approach to the classical placement problem. It is very important how the the placement problem is divided between these two step. Algorithms, based on input analysis, for determination of this division along with techniques for performing GP and DP have been proposed.

Test Case	NRG	GP predict.	% accur.	spdup
highway2	8646	9009	4.2%	19
fract	25602	28494	11.3%	4.9
struct	287631	349399	21.4%	2.3
Primayr1	894545	918435	2.7%	7.3
Primary2	3412195	3941040	15.4%	5.1

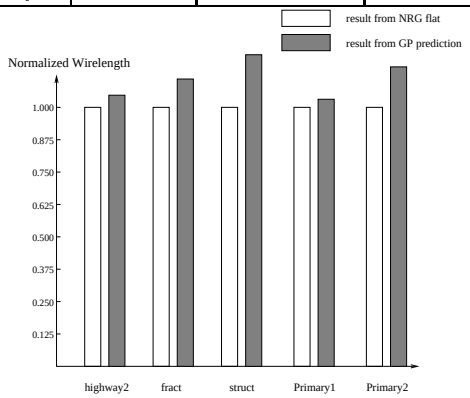

Table 6: Wirelength result: NRG vs. GP prediction

The two step approach allows for fast exploration of the search space. It results in good quality solution with about 3x speed-up. The notion of global and detailed placement will be more important as design problems get harder. We also showed that GP alone can serve as a good predictor if the corresponding grid has been carefully determined.

The notion of GDP can be coupled with most existing algorithms to speed them up and possibly produce better quality solutions. Future work include theoretical study of the quality of GP as a function of the corresponding GP grid and inclusion of non-uniform grids in GP.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant MIP-9527389.

References

[1] "Issues in Timing Driven Layout". pages 1–24, 1993. M. Sarrafzadeh and D. T. Lee, editors.

[2] H. Anway, G. Farnham, and R. Reid. "Plint Layout System for VLSI Chip Design". In

Test Case	HQ NRG	Fast NRG	% deg.	spdup
highway2	8646	8877	2.7%	2.1
fract	25602	26075	1.8%	2.0
struct	287631	291078	1.2%	1.3
Primayr1	894545	904520	1.1%	2.0
Primary2	3412195	3512926	3.0%	1.3

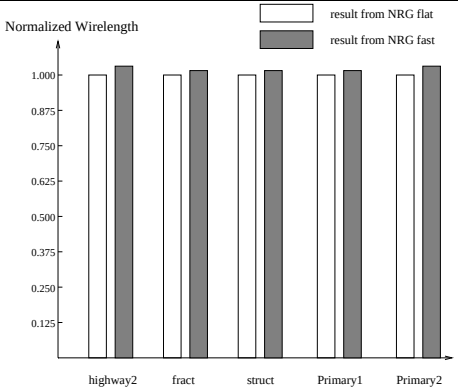

Table 7: Wirelength result: Fast NRG vs. high-quality

Design Automation Conference, pages 449–452. IEEE/ACM, 1985.

[3] M. A. Breuer. "A Class of Min-cut Placement Algorithms". In *Design Automation Conference*, pages 284–290. IEEE/ACM, 1977.

[4] M. A. Breuer. "Min-cut Placement". *J. Design Automation and Fault-Tolerant Computing*, 1(4):343–382, 1977.

[5] C. K. Cheng and E. S. Kuh. "Module Placement Based on Resistive Network Optimization". *IEEE Transactions on Computer Aided Design*, 3(3):218–225, 1984.

[6] A. E. Dunlop and B. W. Kernighan. "A Procedure for Placement of Standard Cell VLSI Circuits". *IEEE Transactions on Computer Aided Design*, 4(1):92–98, January 1985.

[7] C. P. Hsu et al. "APLS2: A Standard Cell Layout System for Double-layer Metal Technology". In *Design Automation Conference*, pages 443–448. IEEE/ACM, 1985.

[8] S.P. Lin, M. Marek-Sadowska, and E.S. Kuh. "Delay and Area Optimization in Standard Cell Design". In *Design Automation Conference*, pages 349–352. IEEE/ACM, 1990.

[9] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa. "Generation of Performance Constraints for Layout". *IEEE Transactions on Computer Aided Design*, CAD-8(8):860–874, August 1989.

[10] M. Sarrafzadeh and C. K. Wong. *An Introduction to VLSI Physical Design*. McGrwa Hill, 1996.

[11] C. Sechen. *VLSI Placement and Global Routing Using Simulated Annealing*. Kluwer, B. V., Deventer, The Netherlands, 1988.

[12] C. Sechen and K. W. Lee. "An Improved Simulated Annealing Algorithm for Row–Based Placement". In *Design Automation Conference*, pages 180–183. IEEE/ACM, 1988.

[13] C. Sechen and A. Sangiovanni-Vincentelli. "TimberWolf3.2: A New Standard Cell Placement and Global Routing Package". In *Design Automation Conference*, pages 432–439. IEEE/ACM, 1986.

[14] N. A. Sherwani. *Algorithms For VLSI Physical Design Automation*. Kluwer Academic Publishers, 1993.

[15] G. Sigl, K. Doll, and F. M. Johannes. "Analytical Placement: A Linear or a Quadratic Objective Function". In *Design Automation Conference*, pages 427–431. IEEE/ACM, 1991.

[16] T. Ohtsuki T. Sudo and S. Goto. "CAD Systems for VLSI in Japan". In *Information and Control*, volume 59, 1983.