

RQL: Global Placement via Relaxed Quadratic Spreading and Linearization *

Natarajan Viswanathan^{1,2}, Gi-Joon Nam¹, Charles J. Alpert¹,
Paul Villarrubia¹, Haoxing Ren¹, Chris Chu²

¹IBM Corporation, 11501 Burnet Road, Austin, TX 78758
{nviswan, gnam, alpert, pgvillar, haoxing}@us.ibm.com

² Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011
{nataraj, cnchu}@iastate.edu

ABSTRACT

This paper describes a simple and effective quadratic placement algorithm called *RQL*. We show that a good quadratic placement, followed by local wirelength-driven spreading can produce excellent results on large-scale industrial ASIC designs. As opposed to the current top performing academic placers [4, 7, 11], *RQL* does not embed a linearization technique within the solver. Instead, it only requires a simpler, pure quadratic objective function in the spirit of [8, 10, 23].

Experimental results show that *RQL* outperforms all available academic placers on the ISPD-2005 placement contest benchmarks. In particular, *RQL* obtains an average wirelength improvement of 2.8%, 3.2%, 5.4%, 8.5%, and 14.6% versus *mPL6* [5], *NTUPlace3* [7], *Kraftwerk* [20], *APlace2.0* [11], and *Capo10.2* [18], respectively. In addition, *RQL* is three, seven, and ten times faster than *mPL6*, *Capo10.2*, and *APlace2.0*, respectively. On the ISPD-2006 placement contest benchmarks, on average, *RQL* obtains the best scaled wirelength among all available academic placers.

Categories and Subject Descriptors

B.7.2 [Hardware, Integrated Circuits, Design Aids]: Placement and routing

General Terms

Algorithms, Design

Keywords

Analytical Placement, Force-vector Modulation

1. INTRODUCTION

Global placement is a fundamental and still highly relevant problem in VLSI CAD because the quality of the placement significantly impacts the ability of a physical synthesis

*This work was partially supported by IBM custom funding through SRC Task ID 1206 and NSF under grant CCF-0540998.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2007, June 4–8, 2007, San Diego, California, USA.

Copyright 2007 ACM 978-1-59593-627-1/07/0006 ...\$5.00.

tool (or designer) to achieve design closure. An industry-strength placer must be reasonably fast, yet still obtain high-quality solutions. It should also be scalable with design size. The last few years have seen significant advances in the quality of global placement algorithms (e.g., [5, 7, 11, 20] etc.) in large part due to the recent availability of large, challenging testcases [13, 14].

Existing placement algorithms can be broadly classified into three categories: *simulated annealing* [19, 21], *top-down partitioning* [1, 3, 6, 25] and *analytic placement* (which includes force-directed approaches) [4, 7, 8, 10–12, 15, 20, 23].

In recent years analytical placement techniques have generated a great deal of attention because they have generated the best results at the two recent ISPD placement contests [13, 14]. These analytic placers can be further divided into those that utilize a *quadratic objective* [8, 10, 12, 15, 20, 23] or a *non-linear objective* [4, 7, 11].

This work presents a placer that not only surpasses the quality of existing state-of-the-art placers, but also has the additional advantage of being (relatively) simple. It uses a pure quadratic function in the solver, and does not require the popular log-sum-exponential function of Naylor *et al.* [16]. The primary advantages and features of our approach are:

- Unlike [4, 7, 11], the solver actually uses a pure quadratic wirelength objective and does not rely on the patent of Naylor *et al.*, which prohibits its widespread adoption by the EDA industry.
- *RQL* utilizes a new linearization technique called *Force-vector Modulation* that restructures the placement at a global scale to minimize the wirelength without sacrificing the degree of spreading.
- *RQL* exploits efficient *Density-aware Module Spreading* and wirelength-driven *Local Spreading* techniques to spread the modules over the placement region.
- Finally, *RQL* utilizes a simple, but effective multi-level global placement framework incorporating the above techniques to yield a high-quality placement algorithm.

The rest of this paper is organized as follows: In Section 2 we describe our global placement algorithm in detail. In Section 3, we present the Force-vector Modulation technique. Experimental results are reported in Section 4 followed by conclusions in Section 5.

2. THE RQL ALGORITHM

The quadratic placement approach models the connectivity of the circuit using a system of springs by transforming the netlist hypergraph into a set of two-pin connections. The objective is to minimize the total squared wirelength (EQ 1) of all the two-pin connections, which corresponds to the minimum potential energy of the spring system.

$$\frac{1}{2} \sum_{\forall(i,j)} W_{ij} [(x_i - x_j)^2 + (y_i - y_j)^2] \quad (1)$$

EQ (1) can be written in matrix notation [9] and the problem can be efficiently solved using SOR or conjugate gradient based methods. However, the resulting placement has significant module overlap. Hence, quadratic placement algorithms typically follow an iterative procedure, where in each iteration, the overlap is progressively reduced.

In this section, we first give an overview of our iterative quadratic global placement algorithm. We then describe the individual components of the flow in more detail.

2.1 Multilevel Global Placement Algorithm

Algorithm 1 The RQL Algorithm

```

1: Phase 0: Clustering
2:   initial_number_of_modules ← module_count_in_flat_netlist
3:   while number_of_modules > target_number_of_modules do
4:     cluster netlist using the Best-choice clustering algorithm
5:   end while
6: end
7: Phase 1: Coarsened netlist placement
8:   solve initial quadratic program (QP)
9:   while max_bin_util > target_util_threshold do
10:    perform Density-aware Module Spreading
11:    calculate spreading forces for all the modules
12:    rank modules based on the spreading force magnitude
13:    modulate the spreading force for the top x% of modules
14:    add spreading forces to QP formulation
15:    solve the quadratic program
16:   end while
17:   repeat
18:     perform wirelength-driven Local Spreading
19:     perform Density-aware Module Spreading
20:   until max_bin_util ≤ 1.0
21:   uncluster movable macro-blocks
22:   legalize and fix movable macro-blocks
23: end
24: Phase 2: Refinement
25:   while number_of_modules < initial_number_of_modules do
26:     uncluster netlist
27:     perform wirelength-driven Local Spreading
28:   end while
29: end

```

Figure 1: The RQL placement algorithm.

Figure 1 gives the overall flow of the *RQL* algorithm. The key components of *RQL* are:

1. Clustering: We use a multilevel approach to improve the efficiency and scalability of our placer and use the Best-choice clustering algorithm [15] to reduce the placement problem size. In addition, clustering also has an indirect linearization effect. Clustered modules are often placed in close proximity and this helps reduce the gap between quadratic and linear wirelength.
2. Coarsened netlist placement: We then perform quadratic placement on the coarse netlist. Quadratic placement is an iterative procedure comprising of three steps: (a) solving the quadratic program (QP) (b) Density-aware

Module Spreading to determine the spreading forces (Sec 2.2) (c) addition of spreading forces to the subsequent QP formulation to account for the spreading.

3. Force-vector Modulation: During quadratic placement, we modulate the spreading force on the modules to restructure the placement at a global scale to further minimize the wirelength (Sec 3).
4. Local Spreading: After quadratic placement, we use a Local Spreading technique to further spread the modules while simultaneously minimizing the half-perimeter wirelength (Sec 2.4).
5. Refinement: Once we obtain a placement of the coarsened netlist, we perform a series of unclustering and *Local Spreading*-based refinement steps until we obtain a placement of the original flat netlist.

2.2 Density-aware Module Spreading (DMS)

Solving the quadratic objective with only the netlist based forces results in a placement with significant module overlap. To reduce this overlap and distribute the modules over the placement region we use an efficient *Density-aware Module Spreading (DMS)* algorithm.

In [17] Ren *et al.* proposed a diffusion based spreading algorithm for placement legalization. Based on the density map of the placement region, their scheme uses the diffusion process to move the modules from high to low concentration regions. The technique proposed in [17] can be viewed as global diffusion, because modules will spread as long as there exists any density gradient among the bins in the current density map. However, using such an approach within global placement will cause excessive spreading and adversely impact the wirelength.

One way to control the degree of spreading within diffusion is to reduce the number of time steps [17] taken during module movement. In the limit, the module movement can be restricted to only its neighboring bins. Such a technique can be considered as a localized version of the diffusion algorithm. In principle, this technique is similar to one that equalizes the densities of adjacent bins by migrating the modules between them. Hence, the localized version of diffusion can also be viewed as the cell-shifting [23] technique.

Density-aware Module Spreading is an improved version of the cell-shifting algorithm with better handling of fixed blocks and density target constraints. Note *DMS* does not physically move a module to a new location. It only provides a “target location” for the module based on the current density map of the placement region. The actual location of the module is determined by the subsequent quadratic optimization step.

2.2.1 Handling Fixed Blocks

During spreading, fixed blocks can cause the following serious issues: (a) there will be abrupt transitions in the density map at the boundaries of the fixed blocks, and (b) quadratic optimization might place numerous modules on top of large fixed blocks. Hence, spreading algorithms may find it difficult to (a) enable the modules to “cross over” the fixed blocks to find better locations, and (b) move the modules out of large fixed blocks in an effective manner.

To aid the spreading algorithm in achieving these objectives we use a smoothing transform to remove the sharp

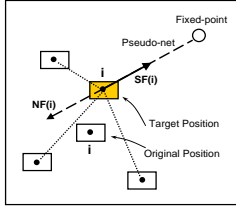


Figure 2: Fixed-point and spreading force.

edges in the density map corresponding to the fixed blocks. We initially construct a density map comprising of only the fixed blocks. A 3×3 Laplacian matrix is then applied as a smoothing filter over the entire density map for a pre-defined number of iterations. This generates a smoothed version of the fixed block density map. This density map is updated with the area utilized by the movable modules to generate the complete density map of the current placement solution. Essentially, smoothing facilitates the modules to cross over any fixed block if necessary or slide down its slope to be moved out of it.

2.2.2 Handling the density target constraint

To handle the density target constraint in an effective manner, for every bin (j, k) , we determine a scaled density value $s_{j,k}$. The scaled density is defined as the average density of a $x \times y$ window of bins around the bin under consideration. If $s_{j,k}$ is lower than the density target for the bin, it is blocked during *DMS*. Otherwise, the entire $x \times y$ window of bins is considered for spreading. This bin blocking prevents unnecessary spreading in regions that are already below the density target, which in turn improves the placement wirelength.

2.3 Addition of Fixed-points

Since *DMS* only gives a target location for a module based on the bin density, we need to add a spreading force to the module to actually move it during the subsequent quadratic optimization. This is done by connecting the module to an associated fixed-point via a pseudo-net. This is shown in Figure 2, where an empty and a shaded box represents the location of a module before and after *DMS* respectively. For a module i if we define:

- $NF(i)$: The *Native Force* on i due to its connections with the other modules in the netlist.
- $SF(i)$: The *Spreading Force* imposed on i to move it to its target location obtained from *DMS*.

The *Spreading Force* vector on the module is equal in magnitude and opposite in direction to the *Native Force* vector experienced by the module in its target location.

The location of the fixed-point and the weight on the pseudo-net are key elements that affect the stability of the spreading and the placement wirelength. If the fixed-point is too close to the module, then the spreading force will dominate the native force during quadratic optimization. This will result in extremely slow spreading and also severely degrade the wirelength. Adding the fixed-point to the chip boundary for large designs makes the spreading force behave as a “constant-force”. Constant forces are hard to control and may lead to a “blow-up” of the placement - creating “donuts” or empty regions in parts of the placement.

Therefore, we use an on-chip fixed-point (shown by the hollow circle in Figure 2 whose location is in between the

two cases. We set the fixed-point distance from the module to be proportional to the distance moved by the module during *DMS*. Specifically, the fixed-point distance $D_{FP} = K1 + K2 \times f(module_displacement)$, where $K1 = K2 = 0.25(chip_diagonal)$ and $module_displacement$ is the distance between the original and target location of a module as shown in Figure 2. If the fixed-point falls outside the chip boundary, we add it at the intersection of the spreading force vector with the chip boundary. This ensures that modules are always placed within the placement region during the subsequent quadratic optimization.

The reason behind a constant and variable term in the fixed-point distance is two-fold: (a) if a module does not move during spreading, we still have the constant term to ensure that it stays in its current location and (b) if a module moves by a large distance during spreading, then the distance of the fixed-point from the module should be proportional to the displacement, to ensure this large displacement during the subsequent quadratic optimization step.

2.4 Local Spreading

Once quadratic placement yields a good initial placement on the coarsened netlist, we invoke a *Local Spreading* technique to further refine the placement solution. Our local spreading technique is similar to the Iterative Local Refinement technique of [23,24]. During Local Spreading, we initially bin the placement region. We determine the *density* of each bin and the modules contained within the bin. We then use a scoring function similar to the one described in [24] to re-locate the modules in a local region.

3. PLACEMENT RESTRUCTURING VIA FORCE-VECTOR MODULATION

Spreading techniques within analytical placement, typically retain the relative ordering of the modules as obtained by solving the initial unconstrained non-linear program. This property simplifies the problem of spreading, but adversely affects the wirelength. To offset this disadvantage, local optimization techniques [10, 23] have been proposed, that change the relative ordering of the modules to improve the wirelength. By nature, such techniques do not have a global view of the placement and can only improve the wirelength in a local region. To re-order the modules and restructure the placement at a global scale, we propose an efficient and effective *Force-vector Modulation* technique, that can be used within any analytical placer (i.e., irrespective of the objective function).

3.1 Spreading Forces in Quadratic Placement

In force-directed quadratic placement, a module is being acted upon by two conflicting forces: the *Native Force* that tries to bring it closer to connected modules and the *Spreading Force* that tries to pull it to the sparse areas within the placement region. During quadratic placement, a careful balance between the *Native* and *Spreading* forces is required to achieve a good trade-off between the objectives of spreading and wirelength minimization.

To understand the relationship between the spreading forces and the wirelength, we observed the magnitude of the spreading force for all the modules over successive iterations of quadratic placement. A plot for one of the iterations is shown in Figure 3.

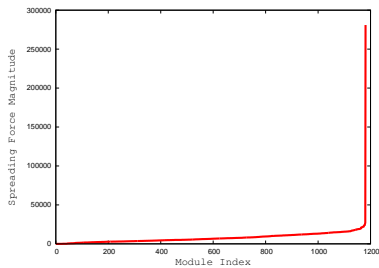


Figure 3: Spreading force magnitude.

From the plots, we see that a small fraction of the modules have an extremely high spreading force magnitude. Since the spreading force is directly proportional to the native force, such high magnitudes imply that these modules either belong to high fanout nets or they are connected to distant modules like boundary I/Os. We experimentally observed that retaining the full spreading force for these modules resulted in a placement with a very high wirelength. Instead, by carefully controlling the spreading force magnitudes for these modules, we were able to substantially decrease the wirelength without sacrificing the degree of spreading.

3.2 Force-vector Modulation

The Force-vector Modulation technique, modulates the spreading force vectors within quadratic placement. Modulation of spreading forces results in a modified distribution of the spreading force magnitudes.

One method for modulation that we propose is to sort the modules in a non-decreasing order of their spreading force magnitude during each iteration of quadratic placement. We then pick a fraction of the modules having a very high spreading force and nullify the spreading force for the subsequent quadratic optimization step. Typically only a small fraction of around 5 – 10% of the modules are picked for nullification. The plot of the spreading force magnitude after nullification is shown in Figure 4.

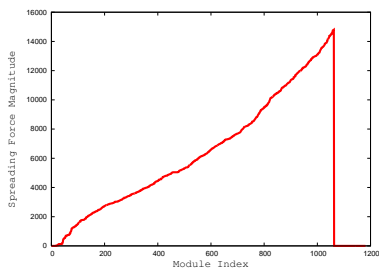


Figure 4: Nullify top $x\%$ of spreading forces.

By nullifying the *Spreading Forces* on a small fraction of the modules, we emphasize the *Native Forces* acting on them. As a result, the locations of these modules after the subsequent quadratic optimization step will be based only on their connections to the other modules in the netlist. This is equivalent to optimizing the wirelength objective with no spreading constraints on these modules. As a result, these modules will be placed in their quadratically optimal locations.

3.2.1 Advantages of Modulation

- Modulation of spreading forces results in a placement solution with a better wirelength since the modulated

modules are placed at their quadratically optimal locations.

- Modulation of spreading forces results in a re-ordering of the modules at a global scale. Since we use the quadratic optimization to perform the re-ordering, it is done on a global scale. Also, modules no longer retain their relative ordering as obtained after the initial quadratic optimization step. We believe that this change in the relative ordering of the modules is the fundamental reason for the significant decrease in the final global placement wirelength.
- Modulation does not impact the degree of spreading. Since we pick only a small fraction of the modules for modulation, the unmodulated modules will still shift towards their target locations as determined during *DMS* and hence contribute towards spreading the placement.
- Modulation can be used within any analytical placer, irrespective of the type of objective function, i.e., either quadratic or non-linear optimization based.

4. EXPERIMENTAL RESULTS

The *RQL* algorithm is implemented within the *CPlace* [2] industrial placement framework. We use the ISPD-2005 [14] and ISPD-2006 [13] placement contest benchmarks for the results presented in this section. All runtimes are reported on a 2.6 GHZ AMD Opteron 252 machine with 8 GB RAM.

4.1 Effect of Force-vector Modulation

Table 1 shows the effect of force-vector modulation on the placement wirelength. We report legalized results for two versions of our placer: (i) with force-vector modulation and (ii) without modulation. In both cases, the placer was run to satisfy the same density target. To show the full effect of modulation, we ran the flat version of our placer and did not perform wirelength minimization using the Local Spreading technique. It can be seen that force-vector modulation achieves upto $1.95\times$ reduction in the legalized wirelength without impacting the degree of spreading.

Circuit	Half-Perimeter Wirelength ($\times 10e6$)		
	With Modulation (Case I)	No Modulation (Case II)	$\frac{Case II}{Case I}$
adaptec1	128.84	252.11	1.95
bigblue1	114.47	168.74	1.47

Table 1: Effect of Force-vector Modulation on the HPWL.

4.2 ISPD 05 Placement Contest Benchmarks

Table 3 compares the half-perimeter wirelength (HPWL) of *RQL* in *default* mode with other state-of-the-art academic placers. It is divided in two sections: Section A corresponds to running the other placers in their default modes. Section B corresponds to the HPWL obtained by the top three placers during the ISPD 2005 placement contest. Please note that during the contest, all the placers were given the circuits in advance. There was no limit on the CPU time and the placers were allowed to have *separate* parameters for each individual circuit to obtain the best possible results.

SECTION A: In default mode, ***RQL* obtains the best HPWL results on all eight circuits of the ISPD-2005**

benchmark suite. In particular, *RQL* obtains an average wirelength improvement of 2.8%, 8.5%, 14.6%, 3.2% and 5.4% versus *mPL6*, *APlace2.0*, *Capo10.2*, *NTUPlace3* and *Kraftwerk* respectively.

SECTION B: *RQL* obtains an average wirelength improvement of 1.5% as compared to *APlace*, which generated the best wirelength results during the ISPD-2005 placement contest. Till date, the *APlace* contest wirelength (reproduced in column eight of Table 3) was the best reported results in literature on these circuits.

Table 2 compares the runtime of *RQL* with *mPL6*, *APlace2.0* and *Capo10.2*. Unfortunately, the authors of *NTUPlace3* were unable to provide their binary because of potential patent infringement issues regarding Naylor’s linearization work [16]; hence we could not directly compare runtimes. On average, *RQL* is 3.09 times, 10.22 times and 6.99 times faster than *mPL6*, *APlace2.0* and *Capo10.2* respectively.

Circuit	Runtime			
	RQL (sec)	$\frac{mPL6}{RQL}$	$\frac{APlace2.0}{RQL}$	$\frac{Capo10.2}{RQL}$
adaptec1	626	3.22	9.42	6.57
adaptec2	1039	2.17	8.83	5.44
adaptec3	2004	3.58	11.11	6.31
adaptec4	1747	3.87	14.39	6.59
bigblue1	967	2.83	8.81	6.93
bigblue2	1884	4.14	10.64	7.06
bigblue3	4053	2.59	9.31	9.38
bigblue4	10342	2.33	9.30	7.62
Average		3.09×	10.22×	6.99×

Table 2: Runtime comparison of *RQL* with *mPL6*, *APlace2.0* and *Capo10.2*.

4.3 ISPD 06 Placement Contest Benchmarks

Tables 4 and 5 give the HPWL and scaled HPWL comparison of *RQL* with other academic placers. The scaled HPWL (S_HPWL) is defined as: $S_HPWL = HPWL \times (1 + density_over_flow_penalty)$ [13]. The results for *NTUPlace3* [7] and *FastPlace3* [24] are reported from the respective publications. All other results are those reported during the ISPD-2006 placement contest.

From Table 4, on average *RQL* is 12%, 5% and 4% better in HPWL as compared to *Kraftwerk*, *mPL6* and *NTUPlace2* respectively, which were the top three placers during the ISPD-2006 placement contest. The most current results of *NTUPlace3*, shows an average improvement of 1% over *RQL*.

From Table 5, on average *RQL* is 7%, 1% and 2% better in terms of S_HPWL as compared to *Kraftwerk*, *mPL6* and *NTUPlace2* respectively. The average S_HPWL of *NTUPlace3* is comparable to that of *RQL*, but looking at individual results, *RQL* obtains better S_HPWL on 5/8 circuits as compared to *NTUPlace3*.

5. CONCLUSIONS

Currently, a majority of the top performing academic placers use the patented log-sum-exponential function [16] to minimize linear wirelength. In fact, literature [4] indicates that a placement algorithm with a pure quadratic wirelength objective cannot produce competitive results compared to

one using the log-sum approximation. This paper shows that it is possible to surpass the quality of state-of-the-art placers without infringing on the patent via a new force-directed global placement algorithm called *RQL*. The *RQL* algorithm relies on simple quadratic wirelength optimization with fixed-point based module spreading. Two new techniques are used to obtain high-quality placement solutions: (1) force-vector modulation and (2) density-aware module spreading (*DMS*). Force-vector modulation re-orders the modules at a global scale to improve the wirelength without impacting the degree of spreading. *DMS* prevents unnecessary spreading in low density regions. Experimental results show that *RQL* in default mode obtains the best published wirelength results on the ISPD-2005 benchmarks and also the best scaled wirelength results on the ISPD-2006 benchmarks with significant speed-up.

We are confident that there is room for further improvement both in terms of quality and runtime. For example, we can employ more clustering to reduce the problem size. We are also looking at deploying *RQL* within a physical synthesis framework to understand its interaction with timing analysis and optimization.

6. REFERENCES

- [1] A. R. Agnihotri, S. Ono, C. Li, M. C. Yildiz, A. Khatkhate, C.-K. Koh, and P. H. Madden. Mixed block placement via fractional cut recursive bisection. *TCAD*, 24(5):748–761, May 2005.
- [2] C. J. Alpert, G.-J. Nam, and P. G. Villarrubia. Effective free space management for cut-based placement via analytical constraint generation. *TCAD*, 22(10):1343–1353, Oct. 2003.
- [3] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection produce routable placements. In *Proc. DAC*, pages 477–482, 2000.
- [4] T. Chan, J. Cong, and K. Sze. Multilevel generalized force-directed method for circuit placement. In *Proc. ISPD*, pages 185–192, 2005.
- [5] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie. mPL6: Enhanced multilevel mixed-size placement. In *Proc. ISPD*, pages 212–214, 2006.
- [6] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang, and Y.-W. Chang. NTUplace: A ratio partitioning based placement algorithm for large-scale mixed-size designs. In *Proc. ISPD*, pages 236–238, 2005.
- [7] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. A high-quality mixed-size analytical placer considering preplaced blocks and density constraints. In *Proc. ICCAD*, 2006.
- [8] H. Eisenmann and F. Johannes. Generic global placement and floorplanning. In *Proc. DAC*, pages 269–274, 1998.
- [9] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17:219–229, 1970.
- [10] B. Hu and M. Marek-Sadowska. Multilevel fixed-point-addition-based VLSI placement. *TCAD*, 24(8):1188–1203, Aug. 2005.
- [11] A. B. Kahng, S. Reda, and Q. Wang. Architecture and details of a high quality, large-scale analytical placer. In *Proc. ICCAD*, pages 890–897, 2005.
- [12] J. Kleinbans, G. Sigl, F. Johannes, and K. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *TCAD*, 10(3):356–365, Mar. 1991.
- [13] G.-J. Nam. ISPD 2006 placement contest: Benchmark suite and results. In *Proc. ISPD*, pages 167–167, 2006.
- [14] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. The ISPD2005 placement contest and

Circuit	Section A (Default Mode Runs)						Section B (Placer tuned for each circuit)			
	RQL	mPL6 [5]	AP2.0 [11]	CP10.2 [18]	NP3 [7]	KW [20]	AP [11]	mFAR [10]	DG [22]	mPL [4]
adaptec1	77.82	77.91	78.35	91.28	80.93	NA	NA	NA	NA	NA
adaptec2	88.51	91.96	95.70	100.75	89.95	93.84	87.31	91.53	94.72	97.11
adaptec3	210.96	214.05	218.52	228.47	214.20	NA	NA	NA	NA	NA
adaptec4	188.86	194.23	209.28	208.35	193.74	199.75	187.65	190.84	200.88	200.94
bigblue1	94.98	96.79	100.02	108.60	97.28	99.61	94.64	97.70	102.39	98.31
bigblue2	150.03	152.33	153.75	162.92	152.20	155.19	143.82	168.70	159.71	173.22
bigblue3	323.09	344.37	411.59	398.49	348.48	339.20	357.89	379.95	380.45	369.66
bigblue4	797.66	829.35	871.29	965.30	829.16	857.09	833.21	876.28	903.96	904.19
Average	1.000	1.028	1.085	1.146	1.032	1.054*	1.015*	1.079*	1.098*	1.105*

Table 3: HPWL ($\times 10e6$) comparison on the ISPD-2005 placement contest benchmarks. (* avg of 6 ckts) (AP=APlace, CP=Capo, NP=NTUPlace, KW=Kraftwerk, DG=Dragon)

	adaptec5	newblue1	newblue2	newblue3	newblue4	newblue5	newblue6	newblue7	Average
RQL	405.73	64.21	196.74	269.13	268.07	473.14	494.30	1031.33	1.00
Aplace3	449.61	73.26	197.42	273.63	377.55	545.90	522.58	1098.26	1.12
mFAR	448.43	77.36	211.65	303.58	307.73	567.65	527.36	1135.80	1.13
Dragon	500.24	80.76	259.95	524.41	340.70	613.34	572.19	1408.97	1.36
mPL6	425.12	66.90	197.53	283.80	294.43	530.67	510.40	1070.33	1.05
Capo	491.60	98.35	308.64	361.21	358.28	657.40	668.33	1518.49	1.40
NTUPlace2	404.98	62.40	201.95	291.14	284.99	494.57	504.39	1116.86	1.04
FastPlace	478.47	84.49	209.73	361.05	319.08	601.45	539.16	1173.12	1.20
Kraftwerk	444.07	78.29	205.87	279.94	311.09	555.48	537.32	1139.17	1.12
DPlace	463.95	102.37	324.07	379.19	305.78	600.11	674.39	1398.85	1.37
NTUPlace3 [7]	378.56	60.74	198.76	278.87	274.48	474.84	484.81	1056.78	0.99
FastPlace3 [24]	432.96	78.56	201.51	292.58	284.54	530.12	539.44	1124.55	1.10

Table 4: HPWL ($\times 10e6$) comparison on the ISPD-2006 placement contest benchmarks.

	adaptec5	newblue1	newblue2	newblue3	newblue4	newblue5	newblue6	newblue7	Average
RQL	443.28	64.43	199.60	269.33	308.75	537.49	515.69	1057.79	1.00
APlace3	520.97	73.31	198.24	273.64	384.12	613.86	522.73	1098.88	1.10
mFAR	476.28	77.54	212.90	303.91	324.40	601.27	535.96	1153.76	1.10
Dragon	500.74	80.77	260.83	524.58	341.16	614.23	572.53	1410.54	1.29
mPL6	431.14	67.02	200.93	287.05	299.66	540.67	518.70	1082.92	1.01
Capo	494.64	98.48	309.53	361.25	362.40	659.57	668.66	1518.75	1.33
NTUPlace2	432.58	63.49	203.68	291.15	305.79	517.63	532.79	1181.30	1.02
FastPlace	805.63	84.55	212.30	362.99	429.78	962.06	574.18	1236.34	1.38
Kraftwerk	457.92	78.60	208.41	280.93	315.53	569.36	545.94	1170.85	1.07
DPlace	572.98	102.75	329.92	380.14	364.45	752.08	682.87	1438.99	1.40
NTUPlace3 [7]	448.58	61.08	203.39	278.89	301.19	509.54	521.65	1099.66	1.00
FastPlace3 [24]	517.56	78.75	202.98	294.77	325.06	633.21	546.72	1139.24	1.11

Table 5: Comparison of the Scaled HPWL (S_HPWL) ($\times 10e6$) which includes the density target based overflow penalty on the ISPD-2006 placement contest benchmarks.

- benchmark suite. In *Proc. ISPD*, pages 216–220, 2005.
- [15] G.-J. Nam, S. Reda, C. J. Alpert, P. G. Villarrubia, and A. B. Kahng. A fast hierarchical quadratic placement algorithm. *TCAD*, 25(4):678–691, Apr. 2006.
- [16] W. Naylor et al. Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer, Oct. 2001.
- [17] H. Ren, D. Z. Pan, C. J. Alpert, and P. Villarrubia. Diffusion-based placement migration. In *Proc. DAC*, pages 515–520, 2005.
- [18] J. A. Roy, S. N. Adya, D. A. Papa, and I. L. Markov. Min-cut floorplacement. *TCAD*, 25(7):1313–1326, July 2006.
- [19] C. Sechen and A. L. Sangiovanni-Vincentelli. TimberWolf 3.2: A new standard cell placement and global routing package. In *Proc. DAC*, pages 432–439, 1986.
- [20] P. Spindler and F. M. Johannes. Fast and robust quadratic placement combined with an exact linear net model. In *Proc. ICCAD*, 2006.
- [21] W.-J. Sun and C. Sechen. Efficient and effective placement for very large circuits. *TCAD*, 14(5):349–359, 1995.
- [22] T. Taghavi, X. Yang, B.-K. Choi, M. Wang, and M. Sarrafzadeh. Dragon2005: Large-scale mixed-size placement tool. In *Proc. ISPD*, pages 245–247, 2005.
- [23] N. Viswanathan and C. C.-N. Chu. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. *TCAD*, 24(5):722–733, May 2005.
- [24] N. Viswanathan, M. Pan, and C. Chu. Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In *Proc. ASP-DAC*, pages 135–140, 2007.
- [25] M. Wang, X. Yang, and M. Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. ICCAD*, pages 260–263, 2000.