Kraftwerk2—A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model

Peter Spindler, Ulf Schlichtmann, *Member, IEEE*, and Frank M. Johannes

***Abstract*—The force-directed quadratic placer "Kraftwerk2," as described in this paper, is based on two main concepts. First, the force that is necessary to distribute the modules on the chip is separated into the following two components: a hold force and a move force. Both components are implemented in a systematic manner. Consequently, Kraftwerk2 converges such that the module overlap is reduced in each placement iteration. The second concept of Kraftwerk2 is to use the "Bound2Bound" net model, which accurately represents the half-perimeter wirelength (HPWL) in the quadratic cost function. Aside from these features, this paper presents additional details about Kraftwerk2. An approach to remove halos (free space) around large modules is described, and a method to control the module density is presented. In order to choose the important tradeoff between runtime and quality, a systematic quality control is shown. Furthermore, plots demonstrating the convergence of Kraftwerk2 are presented. Results using various benchmark suites demonstrate that Kraftwerk2 offers both high quality and excellent computational efficiency.**

***Index Terms*—Bound2Bound, force-directed, half-perimeter wirelength (HPWL), Kraftwerk2, quadratic placement.**

I. Introduction

THE QUALITY of an integrated circuit design is significantly determined by its physical implementation and, in particular, by the quality of the placement. Consequently, research activities on placement have proceeded continuously over the past several decades, and significant progress has been achieved. However, with the continuous growth in the complexity of integrated circuits, there is still an urgent need for fast placers offering high quality.

Fig. 1 categorizes various placers according to which of the three main placement techniques they use. Most placers are iterative and solve the placement problem in a series of placement iterations.

Stochastic placers often utilize simulated annealing, e.g., Timberwolf [1], and they are theoretically able to find the global optimum. However, the high CPU times may limit their applicability.

Min-cut placers recursively divide the netlist and the chip area. The netlist is cut based on minimizing a cost function, e.g., the number of nets connecting adjacent partitions. Examples of min-cut placers are Capo [2], Dragon [3], and FengShui [4].

Manuscript received December 14, 2007; revised February 8, 2008. This paper was recommended by Associate Editor L. Scheffer.

The authors are with the Institute for Electronic Design Automation, Technische Universität München, 80333 Munich, Germany (e-mail: peter.spindler@tum.de; ulf.schlichtmann@tum.de; frank.johannes@tum.de).

Digital Object Identifier 10.1109/TCAD.2008.925783

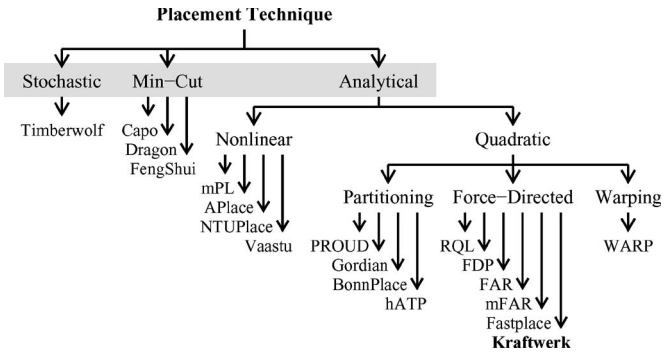

Fig. 1. Three main placement techniques and various placers.

Analytical placers define a suitable analytical cost function for the placement problem and minimize the cost function through numerical optimization methods. Depending on the cost function, analytical placers can be subdivided into the following two categories: nonlinear and quadratic.

Nonlinear placers are based on a nonlinear cost function, e.g., by expressing the netlength in a log-sum-exp function [5]. Because nonlinear optimization needs high CPU times, placers based on this optimization technique usually reduce the complexity by a multilevel approach. Here, the netlist is clustered over a few levels during the coarsening phase. In the refinement phase, the coarsest netlist is placed, unclustered, and then the next finer netlist is placed. This is done until the original flat netlist is reached. Examples of nonlinear placers are APlace [6], mPL [7], NTUPlace [8], and Vaastu [9].

Quadratic placers formulate the netlength in a quadratic cost function, which can be minimized quite efficiently by solving systems of linear equations. Because minimizing just the netlength may result in considerable module overlap, quadratic placers need a method to reduce the overlap. Depending on this method, quadratic placers can be subdivided into the following three categories: partitioning based, force directed, and warping based. Partitioning-based quadratic placers like PROUD [10], Gordian [11], BonnPlace [12], and hATP [13] recursively partition the circuit and the chip area and minimize the quadratic cost function in each level of partitioning in order to place the modules. Force-directed quadratic placers like FAR [14], FastPlace [15], mFAR [16], FDP [17], RQL [18], and Eisenmann's Kraftwerk [19] utilize a force to spread the modules over the chip. Warping-based quadratic placers, as presented in [20], [21], and [22], deform the chip area and move the modules indirectly.

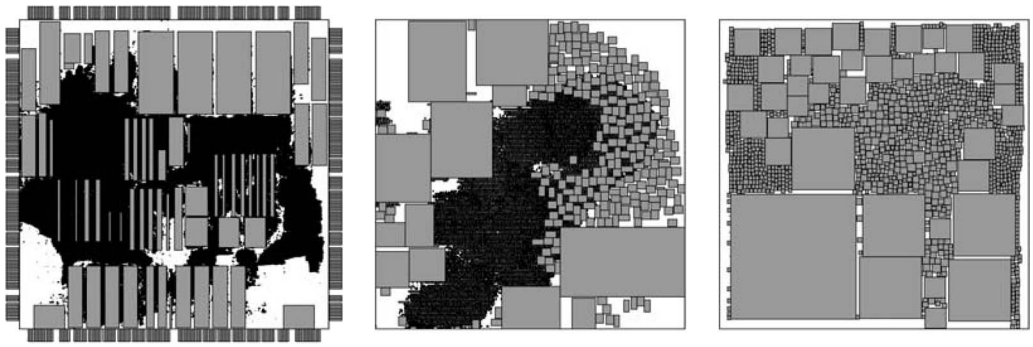

Fig. 2. Different placement types supported by Kraftwerk2. Final placements with no module overlap are displayed. (a) Standard cell placement: Millions of small movable and big modules are fixed. (b) Mixed-size placement: Small and big modules are movable. (c) Floorplacing: Some modules of various dimensions.

This paper focuses on the force-directed quadratic placement technique, as it offers both low CPU time and high placement quality. Several approaches have appeared in recent years to determine and model the force that is necessary for this placement technique. Eisenmann's Kraftwerk [19] represents the module density in a Poisson potential. At the start of each placement iteration, the gradient of the potential is determined at the position of each module. The scaled gradients are accumulated over the placement iterations and give a constant additional force. The force is constant as it does not depend on the module positions computed during each placement iteration. FDP [17] utilizes a similar approach to Eisenmann's Kraftwerk. Furthermore, FDP uses two more forces to stabilize the algorithm and to improve the netlength. Using a constant force is one way to model a force. The authors of FAR [14] showed that another way to model a force is to assign a fixed point to each module and then connect the module to its fixed point by an elastic spring. This spring then creates the force. FAR [14] and mFAR [16] utilize two forces. The controlling force in both placers is given by achieving force equilibrium and is modeled by controlling fixed points. The perturbing force of FAR is constant, and it is determined by a potential that is similar to Eisenmann's Kraftwerk. The perturbing force of mFAR is given by a local bin utilization and is modeled by perturbing fixed points. FastPlace [15] and RQL [18] utilize one force, the spreading force, which is given by a local bin utilization and is modeled by spreading fixed points. In addition, RQL uses a force vector modulation technique and removes the spreading force of single modules if the force is too high. Like nonlinear placers, most force-directed quadratic placers use a multilevel approach to cope with the complexity of modern circuits.

This paper presents Kraftwerk2, which is based on [23] and [24]. Similar to other approaches, Kraftwerk2 utilizes a Poisson potential and separates the force into the following two components: a hold force and a move force. The enhancements of Kraftwerk2 over other force-directed quadratic placement approaches are as follows.

1) The move force is modeled by target points, and the locations of the target points are directly given by the gradient of a Poisson potential. The Poisson potential is created by a generic supply and demand system.
2) The hold force is modeled as a constant force, not by fixed points. Due to the hold force, no force accumulation over the placement iterations is necessary, and each placement iteration is decoupled from the preceding one.
3) A deterministic quality control is used to choose the important tradeoff between runtime and quality.
4) An advanced method for module demand is applied to handle modules of different dimensions and to prevent halos around big modules.
5) To control the module density, an advanced approach for module supply is utilized.
6) A flat placement approach is followed, which means that Kraftwerk2 considers the complete netlist in each placement iteration.

Fig. 2 shows the following different placement types that are supported by Kraftwerk2: circuits with millions of standard cells, circuits with movable macros, and circuits consisting of some modules with various dimensions.

Kraftwerk2 and all other quadratic placers rely on a net model to represent the nets by two-pin connections. The sum of the quadratic length of all two-pin connections gives the quadratic cost function. Traditionally, a clique net model is used, which utilizes all possible two-pin connections of a net. Another common net model is the star net model, which introduces one star pin per net and connects each pin with the star pin. Viswanathan and Chu [15] demonstrated the equivalence of the star net model and the clique net model. Hence, both net models (clique and star) can be used interchangeably. Furthermore, different approaches exist to determine the weight of the two-pin connections [11], [15], [25], [26].

The enhancements of the Bound2Bound net model, as presented in this paper, over other net models are as follows.

1) Accurate representation of the half-perimeter wirelength (HPWL) in the quadratic cost function.
2) Compared with that of the clique net model, the number of two-pin connections is lower.
3) Compared with that of the star net model, no additional star pins are necessary.
4) Based on experimental results, the Bound2Bound net model offers lower runtime and better netlength than a hybrid clique/star net model.

The rest of this paper is organized as follows. Section II gives an introduction to net models in quadratic placement. Then, Section III presents the new Bound2Bound net model and compares different net models. Section IV describes the

background of force-directed quadratic placement. Section V presents the force-directed placer Kraftwerk2. Section VII describes the quality control used. Sections VIII and IX present the approaches for the module demand and the module supply, respectively. Section X is about convergence. Section XI presents experimental results. Finally, Section XII concludes the paper.

II. Net Models

Integrated circuits consist of modules (set $\mathcal{M}$), the modules have pins (set $\mathcal{P}$), and the pins are connected by nets (set $\mathcal{N}$). The term "modules" denotes standard cells and macros in this paper. In quadratic placement, the nets are modeled by two-pin connections. This modeling is done by a net model and results in the representation of each net $n \in \mathcal{N}$ by a set $\mathcal{E}_n$ of two-pin connections. One two-pin connection $e = (p, q)$ connects pins p and q. Each pin (p for example) is located at $(x_p^{\text{pin}}, y_p^{\text{pin}})$. The sum of the (weighted) quadratic Euclidean length of all two-pin connections gives the quadratic cost function Γ. Thus, Γ represents the netlength of the circuit

$$\Gamma = \frac{1}{2} \sum_{n \in \mathcal{N}} \sum_{e \in \mathcal{E}_n} w_{x,pq} \left(x_p^{\text{pin}} - x_q^{\text{pin}}\right)^2 + w_{y,pq} \left(y_p^{\text{pin}} - y_q^{\text{pin}}\right)^2 \tag{1}$$

$$= \sum_{n \in \mathcal{N}} \Gamma_{n,x} + \Gamma_{n,y} = \Gamma_x + \Gamma_y. \tag{2}$$

This cost function Γ can be separated into the x- and y-directions, and the cost $\Gamma_{n,x}$ is the cost of net n in the x-direction. In the following, the focus is on $\Gamma_{n,x}$.

Traditionally, the clique net model or the star net model is used in quadratic placement. The clique net model utilizes all possible two-pin connections of a net. The star net model introduces an additional star pin per net and connects each pin of the net to the star pin. With P representing the number of pins in net n, the clique model is equivalent to the star model in the quadratic cost if the clique cost is scaled with $1/P$ [15]. Due to this equivalence of both net models, the focus is on the clique net model in the following. The quadratic cost of the clique model is

$$\Gamma_{n,x} = \frac{1}{2} \sum_{p=1}^{P} \sum_{q=p+1}^{P} w_{x,pq} \left(x_p^{\text{pin}} - x_q^{\text{pin}}\right)^2. \tag{3}$$

Different approaches exist for the connection weight $w_{x,pq}$. GordianL [25] uses the following technique:

$$w_{x,pq}^{\text{GordianL}} = \frac{1}{P} \frac{2}{P} \frac{4}{\left|x_p^{\text{pin}} - x_q^{\text{pin}}\right|}. \tag{4}$$

The first factor $1/P$ adapts the clique model to the star model. The second factor $2/P$ adjusts the number of connections of the clique to the number of connections in the corresponding spanning tree. With the factor $1/|x_p^{\text{pin}} - y_q^{\text{pin}}|$, the quadratic distance between both pins p and q is linearized.

The (quadratic) clique length (3) is one metric for the netlength. The ideal metric for the netlength would be the routed wirelength, as given after final routing. However, placement is done iteratively, and in each iteration, the circuit would have to be routed to obtain the routed wirelength, which would take enormous CPU time. A good estimation of the routed wirelength is the length of the rectilinear minimal Steiner tree (RSMT) [27]. However, constructing the RSMT takes some CPU time. A fast estimation of the routed wirelength is the HPWL. The HPWL Γ_n^{HPWL} of net n is defined by the width w_n and height h_n of the smallest rectangle enclosing all $p = 1, \ldots, P$ pins of the net

$$w_n = \max\left(x_p^{\text{pin}}\right) - \min\left(x_p^{\text{pin}}\right)$$

$$h_n = \max\left(y_p^{\text{pin}}\right) - \min\left(y_p^{\text{pin}}\right) \tag{5}$$

$$\Gamma_n^{\text{HPWL}} = w_n + h_n. \tag{6}$$

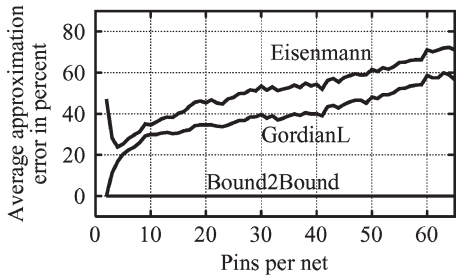

Fig. 3. Approximation error between the quadratic cost function and the HPWL using different approaches for the connection weight $w_{x,pq}$. The statistic is based on 5.6 million nets in the ISPD 2005 contest benchmark suite.

Using GordianL's connection weight (4), the approximation error between the quadratic clique length $\Gamma_{n,x}$ and $\Gamma_{n,x}^{\text{HPWL}}$ is shown in Fig. 3. For two-pin nets, GordianL's approach results in no approximation error. This is due to the factor of four in the last numerator in (4). However, with increasing numbers of pins per net, the approximation error increases. On average, the approximation error is about 30% and is too high to reflect the HPWL precisely in the quadratic cost function Γ.

An unpublished approach by Eisenmann uses the following two-pin connection weight:

$$w_{x,pq}^{\text{Eisenmann}} = \frac{1}{P} \frac{2}{P} \frac{10}{10 + w_n}. \tag{7}$$

Fig. 3 shows that the average approximation error with this approach also depends on the number of pins per net. In addition, Eisenmann's approach has a higher approximation error than GordianL's approach.

III. Bound2Bound Net Model

In the clique net model, there is a high approximation error between the length of the clique net and the HPWL, independent of the different approaches for the connection weights $w_{x,pq}$. The basic problem in the clique model is the existence of connections between inner pins, and the lengths of these inner connections contribute to the clique length but are ignored in the HPWL metric; the HPWL is just the distance between the boundary pins. This problem in the clique net model is shown in Fig. 4(a). Here, the boundary pins are those pins with the highest or lowest coordinate; all other pins are inner pins.

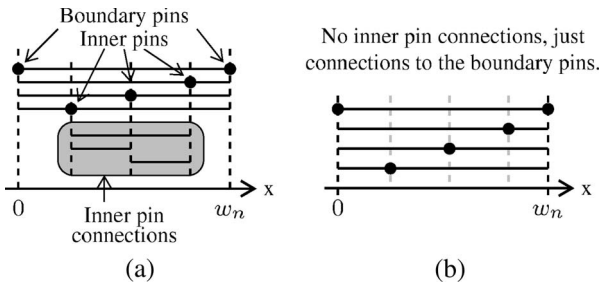

Fig. 4. Traditional clique net model and our Bound2Bound net model. (a) Clique. (b) Bound2Bound.

The new Bound2Bound net model is based on the idea of removing all inner two-pin connections and utilizing only connections to the boundary pins. With this, the boundary pins span the net, and the property of the HPWL netlength being the distance between the boundary pins is emulated. An example of a Bound2Bound net model is shown in Fig. 4(b). The two-pin connection weight $w_{x,pq}^{\mathrm{B2B}}$ of the Bound2Bound net model is determined as follows:

$$w_{x,pq}^{\mathrm{B2B}} = \begin{cases} 0, & \text{if pin } p \text{ and pin } q \text{ are inner pins} \\ \frac{2}{P-1}\frac{1}{\left|x_p^{\mathrm{pin}} - x_q^{\mathrm{pin}}\right|}, & \text{else.} \end{cases} \tag{8}$$

With this connection weight, the quadratic cost function $\Gamma_{n,x}$ (3) of the net is exactly the HPWL in the x-direction

$$\Gamma_{n,x} = \frac{1}{2}\sum_{p=1}^{P}\sum_{q=p+1}^{P} w_{x,pq}^{\mathrm{B2B}}\left(x_p^{\mathrm{pin}} - x_q^{\mathrm{pin}}\right)^2 \tag{9}$$

$$= \frac{1}{2}\frac{2}{P-1}\left[\left|x_1^{\mathrm{pin}} - x_2^{\mathrm{pin}}\right| + \sum_{q=3}^{P}\left|x_1^{\mathrm{pin}} - x_q^{\mathrm{pin}}\right| + \sum_{q=3}^{P}\left|x_2^{\mathrm{pin}} - x_q^{\mathrm{pin}}\right|\right] \tag{10}$$

$$= \frac{1}{P-1}\left[w_n + (P-2)w_n\right] \tag{11}$$

$$= w_n. \tag{12}$$

In (10), the linearization factor $1/|x_p^{\mathrm{pin}} - x_q^{\mathrm{pin}}|$ is multiplied with the quadratic distance $(x_p^{\mathrm{pin}} - x_q^{\mathrm{pin}})^2$, which gives the linear distance $|x_p^{\mathrm{pin}} - x_q^{\mathrm{pin}}|$. Furthermore, all possible two-pin connections are separated into the following three categories: connections between the two boundary pins ($p = 1$, $q = 2$), connections between the "left" boundary pin 1 and the inner pins ($p = 1$, $q \geq 3$), and connections between the "right" boundary pin 2 and the inner pins ($p = 2$, $q \geq 3$). The inner two-pin connections ($p \geq 3$, $q > 3$) are not considered, as they have a connection weight of zero (8). With $w_n = |x_1^{\mathrm{pin}} - x_2^{\mathrm{pin}}|$, (11) is given. At last, (12) expresses that the quadratic cost function $\Gamma_{n,x}$ is exactly the HPWL in the x-direction w_n. By using similar operations for the y-direction, it can be shown that the Bound2Bound net model exactly represents the HPWL in the cost function Γ_n of each net. Thus, the approximation error is zero in the Bound2Bound net model (independent of the number of pins per net), which is shown in Fig. 3.

TABLE I
COMPARISON BETWEEN THE NEW BOUND2BOUND NET MODEL AND TWO APPROACHES (GORDIANL AND EISENMANN) FOR THE CONNECTION WEIGHTS IN A CLIQUE/STAR NET MODEL. RESULTS REPRESENT LEGAL PLACEMENTS OF THE ISPD 2005 CONTEST BENCHMARK SUITE. CPU IS IN SECONDS AND REPRESENTS THE RUNTIME TO PLACE EACH CIRCUIT. HPWL IS IN METERS AND REPRESENTS THE COMPLETE NETLENGTH OF EACH CIRCUIT

	Bound2Bound		**GordianL**		**Eisenmann**	
Circuit	HPWL	CPU	HPWL	CPU	HPWL	CPU
adaptec1	82.43	262	87.96	303	87.63	321
adaptec2	92.85	349	99.63	403	98.54	385
adaptec3	227.22	713	239.97	852	239.05	745
adaptec4	199.43	709	212.31	829	213.32	721
bigblue1	97.67	407	104.81	484	107.23	441
bigblue2	154.74	559	165.27	590	165.60	606
bigblue3	343.32	2070	370.00	2367	389.58	2220
bigblue4	852.40	4147	942.06	5491	958.44	4758
Average	**1.000**	**1.00**	**1.073**	**1.17**	**1.084**	**1.10**

A. Comparison

For nets with two and three pins, the new Bound2Bound net model has the same number of two-pin connections as the clique net model. For all other nets, the clique net model has the most two-pin connections—with a quadratic complexity. The Bound2Bound net model has just a linear complexity in the number of two-pin connections, but it has more two-pin connections than the star net model. However, no additional star pins are introduced in the Bound2Bound net model. In an average circuit, most of the nets have two or three pins, and nets with lots of pins are rare. Based on such a circuit, the number of all two-pin connections is about 75% lower in the Bound2Bound net model than in the clique net model.

In [15], the hybrid usage of the clique net model and the star net model in a circuit is presented. For small nets (i.e., nets with a low number of pins), the clique net model is used. For big nets (i.e., nets with a high number of pins), the star net model is used in order to reduce the number of two-pin connections. The disadvantage of increasing the number of pins by the additional star pins is accepted here because there are just a few big nets in an average circuit. Compared with such a hybrid clique/star net model, the number of all two-pin connections is about the same as in the Bound2Bound net model.

Table I shows experimental results comparing the Bound2Bound net model with the hybrid clique/star net model. The results represent legal placements of modern circuits and are obtained with the placer Kraftwerk2. The placer is described in the rest of this paper. In the hybrid clique/star net model, GordianL and Eisenmann's approaches for the two-pin connection weights are used. To obtain the best CPU times for the hybrid clique/star net model when using Kraftwerk2, all nets with up to five pins are modeled as cliques; the remaining nets are modeled as stars. Table I demonstrates that the new Bound2Bound net model offers the best results in HPWL netlength and CPU time. Eisenmann's approach increases the HPWL by about 8% and the CPU time by about 10%. By using GordianL's approach, the HPWL is increased by about 7%, and the CPU time is increased by about 17%. The Bound2Bound net model has the best HPWL because it accurately models the HPWL in the quadratic cost function.

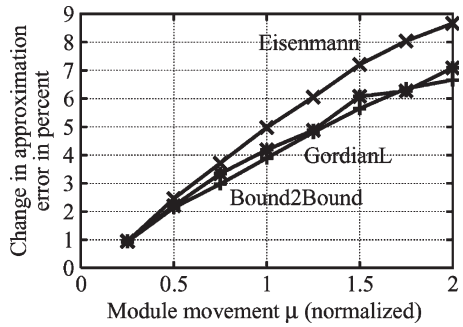

Fig. 5. Change in approximation error due to the module movement for different net models. Results are based on the bigblue1 circuit of the ISPD 2005 contest benchmark suite. The module movement is normalized to the movement, which gives a good tradeoff between runtime and quality (see Section VII).

The Bound2Bound net model has the lowest CPU time because no additional star pins are used.

B. Approximation Error Versus Module Movement

In each placement iteration of quadratic placement, the quadratic cost function Γ is minimized, and the modules are moved to the minimum. Before minimization, a net model is applied to represent the netlength in Γ and to determine the connection weight $w_{x,pq}$. There, $w_{x,pq}$ depends on the pin positions and, thus, on the module positions. Once the connection weights are determined, they remain constant, i.e., neither they are updated during minimization nor after the modules have been moved.[1] Hence, there is an inherent approximation error $\hat{\epsilon}$ between the quadratic cost function and the HPWL netlength after the modules have been moved. ϵ is the approximation error before the modules have been moved, i.e., right at the point where the net model is applied. Based on the statements in the previous section, $\epsilon = 0$ in the Bound2Bound net model.

Fig. 5 shows the change in the approximation error $\Delta\epsilon = |\epsilon - \hat{\epsilon}|$, depending on the average module movement μ for three approaches: the Bound2Bound net model and the hybrid clique/star net model with both GordianL and Eisenmann's approaches for the connection weights. In general, $\Delta\epsilon$ increases with the module movement, and there is no essential difference between the net models and between the approaches for the connection weights. Thus, the Bound2Bound net model also has an inherent approximation error $\hat{\epsilon}$ in the case where the two-pin connection weights $w_{x,pq}^{\mathrm{B2B}}$ remain constant while the modules are moved. Aside from this, Fig. 5 shows that the lowest $\Delta\epsilon$ and, consequently, the best placements are obtained if the modules are moved as little as possible during each placement iteration. This is of interest in Section VII addressing quality control.

IV. Force-Directed Quadratic Placement

In quadratic placement, the quadratic cost function Γ represents the netlength. The placement (i.e., the module positions) with minimal netlength is obtained by minimizing Γ. Because Γ is separable in the x- and y-directions, this paper focuses on the x-direction. The y-direction is obtained similarly. The formulation of Γ_x in (1) depends on the pin positions and not on the module positions. Hence, a transformation from pin position to module position is necessary. To do this transformation, the function $\pi(p) = m$ maps the pin $p \in \mathcal{P}$ to the module $m \in \mathcal{M}$ according to the relation between module m and pin p

$$\begin{aligned} \pi &: (\mathcal{P}) \rightarrow \mathcal{M} \\ \pi(p) = m &: \text{pin } p \in \mathcal{P} \text{ belongs to module } m \in \mathcal{M}. \end{aligned} \quad (13)$$

Assuming that the pins are located at the center of the modules, the position x_p^{pin} of pin p is the center position x_i of the corresponding module $i = \pi(p) : x_p^{\mathrm{pin}} = x_{\pi(p)}$. During placement, the modules are separated into movable modules (cardinality M) and fixed modules (cardinality F) because only the positions of the movable modules have to be determined. The x-positions of the M movable modules are represented in vector $\mathbf{x}$

$$\mathbf{x} = (x_1, x_2, x_3, \ldots, x_M)^{\mathrm{T}}. \quad (14)$$

With (13) and (14), the quadratic cost function Γ_x that is represented as a sum in (1) can be transformed into a matrix–vector notation

$$\Gamma_x = \frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{C}_{\mathbf{x}}\mathbf{x} + \mathbf{x}^{\mathbf{T}}\mathbf{d}_{\mathbf{x}} + \text{const}. \quad (15)$$

The matrix $\mathbf{C}_{\mathbf{x}}$ represents the connectivity between movable modules, and the vector $\mathbf{d}_{\mathbf{x}}$ reflects the connections between movable and fixed modules. $\mathbf{C}_{\mathbf{x}}$ is of dimension $M \times M$, and has entry c_{ij} in row i, and column j. $\mathbf{d}_{\mathbf{x}}$ is of dimension M and has entry d_i in row i.

The contribution of one two-pin connection $e = (p, q) \in \mathcal{E}$ (with $\pi(p) = i$, $\pi(q) = j$, and a connection weight $w_{x,e}$) on $\mathbf{C}_{\mathbf{x}}$ and $\mathbf{d}_{\mathbf{x}}$ is as follows. If both modules i and j are movable, then $w_{x,e}$ is added to the diagonal matrix entries c_{ii} and c_{jj}, and $w_{x,e}$ is subtracted from the off-diagonal entries c_{ij} and c_{ji}. If one module is fixed, e.g., module j, then $w_{x,e}$ is added to c_{ii}, and $w_{e,x} \cdot x_j$ is subtracted from the vector entry d_i. If both modules are fixed, then $\mathbf{C}_{\mathbf{x}}$ and $\mathbf{d}_{\mathbf{x}}$ do not change.

With no modules fixed, the matrix $\mathbf{C}_{\mathbf{x}}$ is positive semidefinite [28]. With some modules fixed, the matrix is positive definite [17]. In both cases, Γ_x is convex, and its minimum is obtained by setting its derivative to zero. The derivative in the x-direction is described by the nabla operator $\boldsymbol{\nabla}_{\boldsymbol{x}} = (\partial/\partial x_1, \partial/\partial x_2, \ldots, \partial/\partial x_M)^{\mathrm{T}}$

$$\boldsymbol{\nabla}_x \Gamma = \mathbf{C}_{\mathbf{x}}\mathbf{x} + \mathbf{d}_{\mathbf{x}} = 0. \quad (16)$$

Solving the system of linear equations (16) with respect to $\mathbf{x}$ gives the x-positions of the modules with minimal netlength.

In quadratic placement, each two-pin connection can be viewed as an elastic spring, and the cost function Γ represents the total energy of the spring system. As the derivative of an energy is a force, the derivative of the cost function (16) is the net force $\mathbf{F}_{\mathbf{x}}^{\mathbf{net}}$, which is created by the springs (representing the nets) between the pins

$$\mathbf{F}_{\mathbf{x}}^{\mathbf{net}} = \boldsymbol{\nabla}_x \Gamma_x = \mathbf{C}_{\mathbf{x}}\mathbf{x} + \mathbf{d}_{\mathbf{x}}. \quad (17)$$

[1] The weights are determined/updated in the next placement iteration.

If just the net force is acting on the modules, then the modules are strongly contracted somewhere on the chip, resulting in a lot of module overlap. Force-directed quadratic placers utilize an additional force to spread the modules over the chip, and this is done in a sequence of placement iterations.

V. KRAFTWERK2

Kraftwerk2 is a fast force-directed quadratic placer, which separates the additional force into a hold force and a move force. Both forces are implemented in a novel and systematic way. In the following, the force implementation is described on the basis of one placement iteration. At the start of the iteration, the modules are located at $\mathbf{x}'$. The new positions of the modules are represented in vector $\mathbf{x}$. $\mathbf{\Delta x}$ is the change in the module positions

$$\mathbf{\Delta x} = \mathbf{x} - \mathbf{x}'. \tag{18}$$

Next, the force implementation is formally presented. After that, an illustration of one placement iteration is given (see Section V-C).

A. Move Force

The move force moves the modules in the placement iteration in order to reduce the module overlap and to spread the modules over the chip. To determine the move force, the placement is represented in a supply and demand system D

$$D(x,y) = D^{\text{dem}}_{\text{mod}}(x,y) - D^{\text{sup}}_{\text{mod}}(x,y). \tag{19}$$

The demand $D^{\text{dem}}_{\text{mod}}(x,y)$ represents the modules, and the supply $D^{\text{sup}}_{\text{mod}}(x,y)$ is the placement area given by the chip. As a prerequisite, the supply and demand system has to be balanced, i.e., the integral over the demand has to equal the integral over the supply. This is necessary to adapt the demand completely to the supply

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} D^{\text{dem}}_{\text{mod}}(x,y)dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} D^{\text{sup}}_{\text{mod}}(x,y)dxdy. \tag{20}$$

To formulate the demand of the modules, a rectangle function R is used. R is one for all points (x,y) inside a given rectangle, and R is zero outside. The rectangle is defined by its lower left corner (x_{ll}, y_{ll}), its width w, and its height h

$$R(x,y;x_{ll},y_{ll},w,h) = \begin{cases} 1, & \text{if } 0 \le x - x_{ll} \le w \\ & \wedge 0 \le y - y_{ll} \le h \\ 0, & \text{else.} \end{cases} \tag{21}$$

The demand of one module i is

$$D^{\text{dem}}_{\text{mod},i}(x,y) = d_{\text{mod},i} \cdot R\left(x,y;x'_i - \frac{w_i}{2}, y'_i - \frac{h_i}{2} w_i, h_i\right). \tag{22}$$

Here, module i is located at (x'_i, y'_i), has a width w_i, a height h_i, and an area $A_{\text{mod},i} = w_i \cdot h_i$. In the following, the individual module density $d_{\text{mod},i}$ is set to one. However, Section VIII presents an advanced approach for scaling $d_{\text{mod},i}$ to control unwanted halos, i.e., to remove unwanted free space, around large modules. The module demand $D^{\text{dem}}_{\text{mod}}$ for all M movable and F fixed modules is the sum of all single module demands $D^{\text{dem}}_{\text{mod},i}$

$$D^{\text{dem}}_{\text{mod}}(x,y) = \sum_{i=1}^{M+F} D^{\text{dem}}_{\text{mod},i}(x,y). \tag{23}$$

With $d_{\text{mod},i} = 1$, the value of $D^{\text{dem}}_{\text{mod}}$ at point (x,y) reflects the number of modules covering this point. In the module demand (23), there is no fundamental difference between small and large modules or between fixed and movable modules. Thus, it can be used to place various circuit types like standard cell circuits with millions of small modules, mixed-size circuits with small and big modules, and circuits with fixed modules.

Aside from the module demand, a module supply $D^{\text{sup}}_{\text{mod}}(x,y)$ is necessary for the supply and demand system D. This supply is given by the chip area

$$D^{\text{sup}}_{\text{mod}}(x,y) = d_{\text{sup}} \cdot R(x,y;x_{\text{Chip}},y_{\text{Chip}},w_{\text{Chip}},h_{\text{Chip}}). \tag{24}$$

$(x_{\text{Chip}}, y_{\text{Chip}})$ is the lower left corner of the chip, w_{Chip} and h_{Chip} are the dimensions of the chip, and A_{Chip} is the area of the chip. The module supply density d_{sup} is determined by using (20) and (23): $d_{\text{sup}} = \sum_{i=1}^{M+F}(d_{\text{mod},i}A_{\text{mod},i})/A_{\text{Chip}}$. With (24), the modules are spread over the whole chip area. Section IX presents an advanced module supply to spread the modules according to a user-given module density.

After defining the module demand $D^{\text{dem}}_{\text{mod}}$ and the module supply $D^{\text{sup}}_{\text{mod}}$, the supply and demand system D (19) is given. D is interpreted as a charge distribution and creates an electrostatic potential Φ by Poisson's equation

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\Phi(x,y) = -D(x,y). \tag{25}$$

This differential equation is solved, for example, with the geometric multigrid solver DiMEPACK [29]. The usage of a potential is similar to other force-directed quadratic placement approaches. However, Kraftwerk2 utilizes a hold force besides the move force, and the gradients of the potential are not accumulated over the placement iterations in Kraftwerk2. Rather, the gradients of the potential are directly used to determine the location $\overset{\circ}{x}_i$ of the target point of each module i

$$\overset{\circ}{x}_i = x'_i - \frac{\partial}{\partial x}\Phi(x,y)\bigg|_{(x'_i,y'_i)}. \tag{26}$$

The move force $F^{\text{move}}_{x,i}$ of module i is created by a spring connection between the module and its target point

$$F^{\text{move}}_{x,i} = \overset{\circ}{w}_i\,(x_i - \overset{\circ}{x}_i). \tag{27}$$

Based on the move force, the modules, represented in the demand, are moved away from high-density regions (i.e., regions with a lot of demand) to low-density regions (i.e., regions with remaining supply). In other words, in each placement iteration, the demand is adapted further to the supply. $\overset{\circ}{w}_i$ in (27) is the spring constant of the move force and affects the distance that a module i is moved during one placement iteration: With a high

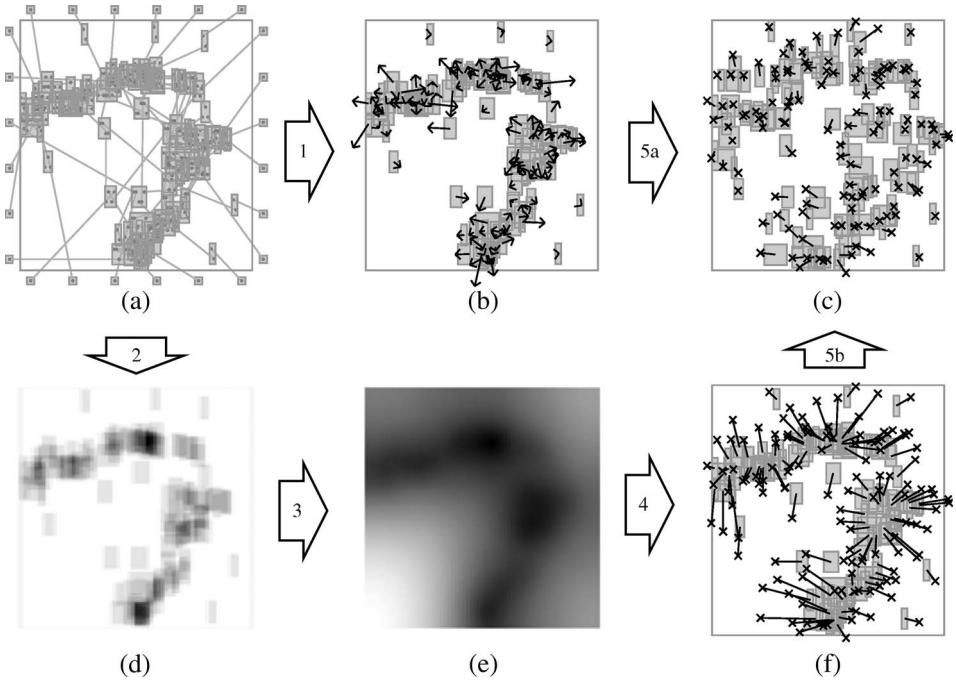

Fig. 6. Illustration of one placement iteration. The numbers in the big arrows represent the sequence of the steps executed in each placement iteration. (d), (e) Density plots, with white and black colors representing low and high densities, respectively. (a) Starting placement. (b) Hold force. (c) Resulting placement. (d) Supply and demand system D. (e) Potential Φ. (f) Target points and move force.

$\overset{\circ}{w}_i$, the move force of module i pulls a lot on its module, and the module will be moved a long distance. The opposite is true for a small $\overset{\circ}{w}_i$.

By using target points for the move force, each module can be moved at most up to its target point during one placement iteration, i.e., the module movement is limited. This enforces the convergence of Kraftwerk2.

To represent the move force (27) in a matrix–vector notation, the weights of the move force are collected in the diagonal matrix $\overset{\circ}{\mathbf{C}} = \operatorname{diag}(\overset{\circ}{w}_i)$. The gradients of the potential are collected in the vector $\Phi = ((\partial/\partial x)\Phi|_{(x'_1,y'_1)}, (\partial/\partial x)\Phi|_{(x'_2,y'_2)}, \ldots, (\partial/\partial x)\Phi|_{(x'_M,y'_M)})^{\mathrm{T}}$. All target points are represented in the vector $\overset{\circ}{\mathbf{x}} = \mathbf{x}' - \mathbf{\Phi}_{\mathbf{x}}$. Therefore, the move force $\mathbf{F}_{\mathbf{x}}^{\mathbf{move}}$ in matrix–vector notation is

$$\mathbf{F}_{\mathbf{x}}^{\mathbf{move}} = \overset{\circ}{\mathbf{C}}_{\mathbf{x}} (\mathbf{x} - \overset{\circ}{\mathbf{x}}). \tag{28}$$

The generic supply and demand system D gives the target points $\overset{\circ}{\mathbf{x}}$ of the move force (25) and (26). D is used in (19) to represent the modules and the placement area. However, D can be extended to consider other supply and demand systems. For example, it can be extended by the routing supply and demand system in order to optimize routability during placement [30], or D can be used to optimize the temperature profile of a chip [31].

B. Hold Force

To spread the modules iteratively over the chip, the move force is used. However, besides the move force, the net force is acting on the modules and minimizes the netlength. Thus, the net force has to be compensated for at the start of each placement iteration. Otherwise, the modules would collapse back to the initial placement, where the netlength is minimal but the modules overlap too much. The compensation of the net force is done by the hold force. Hence, the hold force $\mathbf{F}_{\mathbf{x}}^{\mathbf{hold}}$ equals the negative net force

$$\mathbf{F}_{\mathbf{x}}^{\mathbf{hold}} = -(\mathbf{C}_{\mathbf{x}}\mathbf{x}' + \mathbf{d}_{\mathbf{x}}). \tag{29}$$

By using only the hold force as one additional force, the modules will not collapse back but will stay at their current position. In other words, the change in module position $\mathbf{\Delta x}$ is zero. This can be shown by $\mathbf{F}_{\mathbf{x}}^{\mathbf{net}} + \mathbf{F}_{\mathbf{x}}^{\mathbf{hold}} = \mathbf{0} \Leftrightarrow \mathbf{C}_{\mathbf{x}}\mathbf{\Delta x} = \mathbf{0} \Leftrightarrow \mathbf{\Delta x} = \mathbf{0}$. It should be noted here that the hold force equals the net force only at the start of the placement iteration, where the modules are located at $\mathbf{x}'$. Moreover, the hold force is a constant force, as it does not depend on $\mathbf{x}$.

The result of the hold force is that no force accumulation is necessary, and each placement iteration is decoupled from the previous one. Therefore, the placement algorithm can be restarted at any iteration, and the engineering change order is supported best.

C. Illustration

The previous sections formally described how the move force and the hold force are determined and how they are modeled. This section gives an illustration of one placement iteration, particularly of the forces.

The placement iteration starts with a given placement, and Fig. 6(a) shows such a placement. Ignoring the move force, only the net force is acting on the modules, and the modules are contracted. To compensate for this, the hold force is used, which preserves the given placement. The hold forces are shown as arrows in Fig. 6(b).

Based on the module positions, the supply and demand system D is created, which represents the local module density and the chip area. Fig. 6(d) shows D of the given placement. D is treated as a charge distribution and creates the electrostatic potential Φ shown in Fig. 6(e). Comparing Fig. 6(d) with Fig. 6(e) reveals that the potential Φ can be viewed as a smoothed representation of the supply and demand system D. Moreover, in regions where D is low, the potential Φ is low, and in regions where D is high, the potential Φ is high.

The gradients of the potential determine the target points, and the target points are shown as crosses in Fig. 6(e). The move force is created by spring connections between the modules and their target points. The target points move the modules away from high-density regions [black regions in Fig. 6(d) and (e)] to low-density regions [white regions in Fig. 6(d) and (e)].

Therefore, the following three forces are acting on the modules in each placement iteration: the net force, the hold force, and the move force. These forces move the modules until the sum of the forces is zero. The placement where the sum of all three forces is zero is the resulting placement of one placement iteration, and Fig. 6(c) shows such a placement. Comparing Fig. 6(c) with Fig. 6(f) shows that the modules are moved toward the target points, the modules are spread more over the placement area, and the module overlap is reduced.

D. Core of Kraftwerk2

In summary, the following three forces are used by Kraftwerk2: the net force $\mathbf{F_x^{net}}$, and two additional forces: the move force $\mathbf{F_x^{move}}$ and the hold force $\mathbf{F_x^{hold}}$. Setting the sum of the three forces to zero gives the core system of linear equations used in Kraftwerk2's iterative placement process

$$(\mathbf{C_x} + \overset{\circ}{\mathbf{C}}_\mathbf{x})\mathbf{\Delta x} = -\overset{\circ}{\mathbf{C}}_\mathbf{x}\,\Phi_\mathbf{x}. \tag{30}$$

Solving (30) with respect to $\mathbf{\Delta x}$ and updating $\mathbf{x}'$ by $\mathbf{\Delta x}$ give the new module position $\mathbf{x}$ in the current placement iteration. Because the matrix $\mathbf{C_x} + \overset{\circ}{\mathbf{C}}_\mathbf{x}$ is symmetric, positive definite, and highly sparse, solving (30) can be done efficiently, e.g., with the conjugate-gradient method. Based on (30), Kraftwerk2 has the following three degrees of freedom: first, the cost function Γ, represented in $\mathbf{C_x}$; second, the supply and demand system D, represented in $\mathbf{\Phi_x}$; and third, the weights of the move force $\overset{\circ}{w}_i$, represented in $\overset{\circ}{\mathbf{C}}_x$. The degrees of freedom are used to minimize the netlength, spread the modules over the placement area, and control the tradeoff between quality and runtime.

VI. Overview of the Placement Algorithm

Fig. 7 shows the complete placement algorithm of Kraftwerk2. At first, an initial placement is computed by minimizing the quadratic cost function Γ (15) over a few iterations. In each iteration, the Bound2Bound net model is applied to adjust the two-pin connection weights. The initial placement has a minimal netlength. However, the modules are concentrated somewhere on the chip (mostly at the center), and the modules may overlap a lot.

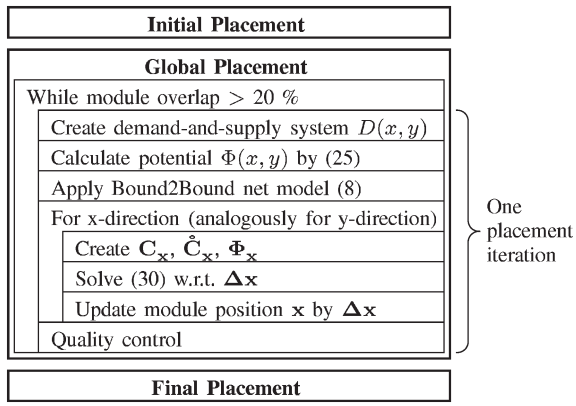

Fig. 7. Complete algorithm of Kraftwerk2.

In global placement, the modules are spread iteratively over the chip. Each placement iteration starts with determining the supply and demand system D and with computing the potential Φ. Then, the Bound2Bound net model is applied to determine the weights of the two-pin connections. Here, (8) is utilized for the x-direction, and a similar equation is used for the y-direction. Once the two-pin connection weights are determined, they remain constant for the rest of the placement iteration. The next step in each placement iteration is to solve the system of linear equations (30) for the x-direction and a similar one for the y-direction. Then, the module positions are updated. Solving the systems of linear equations and updating the module positions are done once per placement iteration. At the end of each placement iteration, a quality-control procedure (presented in Section VII) is called to adjust the weights of the move force. The global placement is stopped if the module overlap Ω is below a certain limit, e.g., below 20%. The definition of Ω is given in (35).

After global placement, final placement is done. Here, the modules are legalized first, i.e., the remaining overlap is removed, and the modules are aligned to rows if necessary. Kraftwerk2 utilizes an approach similar to Tetris [32] for legalizing standard cells. Macros are legalized with an approach similar to that published in [33] and [34]. Legalizing a placement with 20% overlap is done in short CPU time (about 5% of the CPU time of global placement), and the netlength increases by around 1%. After legalization, a simple greedy detailed placement method is applied to improve the legal placement: Single modules are rotated, or pairs of neighboring modules are exchanged. This improves the netlength by about 2%. Most of the runtime of Kraftwerk2 is spent for global placement and to solve the systems of linear equations.

VII. Quality Control

The weights $\overset{\circ}{w}_i$ (where $i = 1, 2, 3, \ldots, M$) of the move force (28) are 1 degree of freedom in Kraftwerk2. They are utilized to control the iterative global placement process and to control the quality of placement. The weight $\overset{\circ}{w}_i$ of module i is initialized at the beginning of the global placement process with

$$\overset{\circ}{w}_i = \frac{A_{\mathrm{mod},i}}{A_{\mathrm{avg}}} \cdot \frac{1}{M}. \tag{31}$$

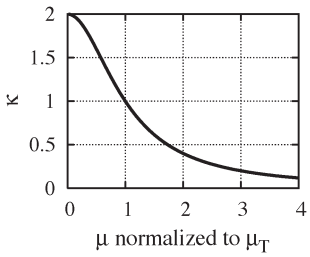

Fig. 8. Scale factor κ, depending on the module movement μ and the target module movement μ_T.

A_{avg} represents the average module area, and M is the number of movable modules. With the factor $A_{\text{mod},i}/A_{\text{avg}}$, the move force (27) of module i is proportional to its module area $A_{\text{mod},i}$. Consequently, big modules are moved farther in each placement iteration than small modules. Thus, big modules are moved away from small modules, and over all iterations, small modules have to be moved less to obtain an overlap-free placement. This improves the netlength, particularly in mixed-size placements, where most of the modules are small and where most of the nets interconnect small modules.

Based on Rent's rule [35], with increasing M, there are more connections between movable modules than connections to fixed modules.[2] Hence, with increasing M and by minimizing the netlength, the movable modules are more contracted, the module overlap is higher in the initial placement, and the target points are farther away from the modules. Consequently, the move force (27) would increase with M if its weight $\overset{\circ}{w}_i$ remains constant. However, $\overset{\circ}{w}_i$ is scaled with $1/M$ in (31). With this, the move force is not increasing with M.

To control the quality during the placement process, the characteristics presented in Section III-B and shown in Fig. 5 are used. There, $\Delta\epsilon$, which is the inherent change in the approximation error between the quadratic cost function Γ and the real objective, depends mainly on the module movement μ. To obtain a high-quality placement, $\Delta\epsilon$ should be low, and thus, μ should be low. However, with a low μ, a high number of placement iterations are necessary to spread the modules over the chip. Consequently, high-quality placements need a high CPU time and vice versa. Thus, there is a tradeoff between quality and runtime, and this tradeoff is controlled by the user in setting a target module movement μ_T. The regulation of the module movement μ according to the target movement μ_T is done by updating the weights $\overset{\circ}{w}_i$ of the move force in the quality-control procedure. This procedure is called at the end of each placement iteration (see Fig. 7) and is implemented as follows. First, the average movement μ of all modules is calculated. Then, a scale factor κ is determined on the basis of μ and μ_T: If $\mu < \mu_T$, then $\kappa > 1$; if $\mu > \mu_T$, then $\kappa < 1$; else, $\kappa = 1$. Fig. 8 shows a suitable function $\kappa(\mu) = 1 + \tanh(\ln(\mu_T/\mu))$.

After the scale factor κ is determined on the basis of μ and μ_T, the weights $\overset{\circ}{w}_i$ of the move force are multiplied with κ

$$\overset{\circ}{w}_i \leftarrow \overset{\circ}{w}_i \cdot \kappa. \tag{32}$$

[2]Fixed modules means fixed macros and fixed I/O pins.

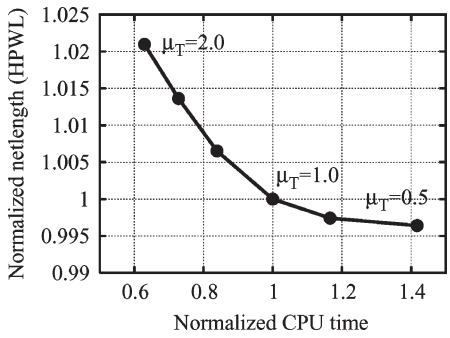

Fig. 9. Tradeoff between runtime and quality based on μ_T. μ_T is normalized to the average module dimension. Results are based on all eight circuits of the ISPD 2005 contest benchmark suite.

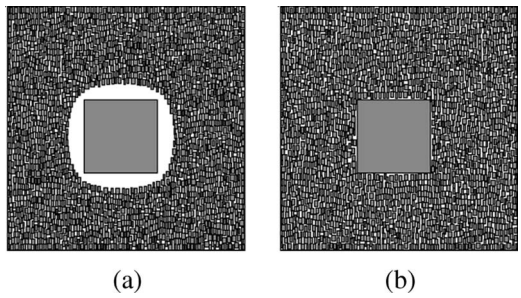

Fig. 10. Impact of scaling down the module density $d_{\text{mod},i}$ for large modules. Global placements are displayed here. (a) With $d_{\text{mod},i} = 1$: Halo around the large module. (b) With scaling down $d_{\text{mod},i}$ for large modules: No halo.

Fig. 9 shows the tradeoff between runtime (CPU time) and quality (netlength in HPWL). The tradeoff is achieved with the presented quality control and is determined by the user parameter μ_T. With a low μ_T, the CPU time is high, and the netlength is good. The opposite is true for a high μ_T. To choose a suitable target movement μ_T, the average module size is a good rule of thumb.

VIII. Module Demand

In Section V-A, the individual module density $d_{\text{mod},i}$ of the module demand was set to one for simplicity: $d_{\text{mod},i} = 1$. However, this results in a halo, i.e., free space, around each module. Fig. 10(a) shows such halos, in particular around the large module at the center. This halo around the large module is not wanted because the small modules are "pushed away" from the large module, and the netlength is increased. A better placement with no halo around the large module is shown in Fig. 10(b). This placement is achieved by scaling down $d_{\text{mod},i}$ for large modules. This section describes details about this approach for preventing unwanted halos around large modules.

The reason for the halos is the potential Φ and, thus, the supply and demand system D. Kraftwerk2 converges to a placement where each module i is in an exclusive region R_i, and no module other than i is in this region R_i. In the region R_i, the total demand $(A_{\text{mod},i} \cdot d_{\text{mod},i})$ equals the total supply $(A_{R,i} \cdot d_{\text{sup}})$. Hence, the area $A_{R,i}$ of the region R_i depends on

the module supply density d_{sup}, the individual module density $d_{\text{mod},i}$, and the module area $A_{\text{mod},i}$

$$A_{R,i} = \frac{d_{\text{mod},i}}{d_{\text{sup}}} A_{\text{mod},i}. \tag{33}$$

In Fig. 10(a), $d_{\text{sup}} = 0.5$ and $d_{\text{mod},i} = 1$. Thus, $A_{R,i} = 2 \cdot A_{\text{mod},i}$, and the exclusive region for the large module in the center is quite big. Consequently, there is free space, i.e., a halo, around the large module. To prevent the halo, $d_{\text{mod},i}$ has to be scaled down, depending on the module area $A_{\text{mod},i}$. A good approach for $d_{\text{mod},i}$ is

$$d_{\text{mod},i} = \begin{cases} 1, & \text{if } A_{\text{mod},i} < A_{\text{large}} \\ \sqrt{\frac{A_{\text{large}}}{A_{\text{mod},i}}}(1 - d_{\text{sup}}) + d_{\text{sup}}, & \text{else.} \end{cases} \tag{34}$$

Here, the individual module density $d_{\text{mod},i}$ stays one for small modules ($A_{\text{mod},i} < A_{\text{large}}$). This conserves the halos around small modules because these halos are necessary to spread the small modules over the placement area. For large modules ($A_{\text{mod},i} \geq A_{\text{large}}$), $d_{\text{mod},i}$ is scaled down with increasing module area $A_{\text{mod},i}$. In addition, $d_{\text{mod},i}$ is bound from below by the supply density d_{sup}. If $d_{\text{mod},i}$ is not bound this way, the placement algorithm would not achieve convergence to an overlap-free placement. A good value for the reference area A_{large} used in (34) is 50 A_{avg}, with A_{avg} denoting the average module area.

IX. Module Supply

In Section V-A, the whole placement area provides supply for the modules. This results in the modules being spread over the whole placement area [see Fig. 11(a)]. To lower the netlength, it may be allowed to pack the modules to a user-given target density td and not to spread them over the whole placement area [see Fig. 11(b)–(d)].

This section presents an approach to control the module density and, thus, to control the netlength. As Kraftwerk2 adapts the demand to the supply, and the modules are represented in the demand, the supply can be used to control the module density. The creation of the module supply is done in two steps. First, an initial module supply $D^{\text{sup}}_{\text{mod,init}}(x, y)$ with the value td is created at each point (x, y), where the module demand is greater than zero: $D^{\text{dem}}_{\text{mod}}(x, y) > 0$. Second, an additional module supply $D^{\text{sup}}_{\text{mod,add}}(x, y)$ with the value td is created around the initial module supply. With this, the initial module supply is framed by the additional module supply, the frame has a constant width, and the supply density inside the frame is td and constant. The sum of the initial and the additional module supply gives the module supply: $D^{\text{sup}}_{\text{mod}} = D^{\text{sup}}_{\text{mod,init}} + D^{\text{sup}}_{\text{mod,add}}$. Because the potential is solved by numeric methods, a grid structure is used to represent the potential and the supply and demand system. Hence, the grid structure can be used to create the initial and the additional module supply.

X. Convergence

Due to the systematic force implementation, in particular using a potential formulation, target points, and a constant hold force, Kraftwerk2 converges such that the module overlap is reduced in each placement iteration. This can be analyzed in theory, but various assumptions and a lot of formulas are necessary. Hence, this section focuses more on the practical issues of the convergence. Fig. 12 shows the results of one typical experiment and demonstrates the convergence of Kraftwerk2. Here, a circuit with 0.2 million small movable modules and some big fixed modules is placed over a few placement iterations.

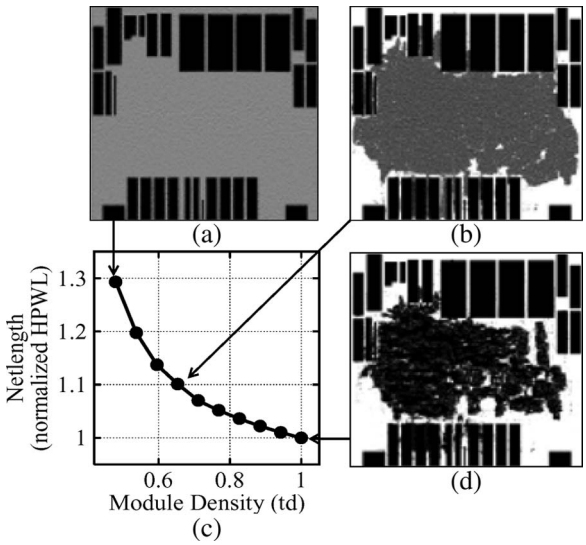

Fig. 11. Control of the module density. Module density plots (a), (b), and (d) represent a low density with white color and a high density with black color. The big black rectangles represent fixed big modules. Based on a circuit with 0.2 million small movable modules and some big fixed modules.

Fig. 12(a) shows that the module overlap Ω is continuously decreasing over all placement iterations and best shows the convergence. The module overlap Ω can be calculated on the basis of $D^{\text{dem}}_{\text{mod}}$ (23) using $d_{\text{mod},i} = 1$. With a function $\omega(x, y) = \{1 \text{ if } D^{\text{dem}}_{\text{mod}}(x, y) \geq 1; 0 \text{ else}\}$, the area $A_{\cup}$ of the union of all modules is given: $A_{\cup} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \omega(x, y) dx dy$. With $A_{\text{mod}} = \sum_{i=1}^{M+F} A_{\text{mod},i}$ being the sum of the area of all modules, Ω is

$$\Omega = 1 - \frac{A_{\cup}}{A_{\text{mod}}}. \tag{35}$$

If no modules are overlapping, $A_{\cup}$ equals A_{mod}, and Ω is zero. If there is some module overlap, $A_{\cup}$ is smaller than A_{mod}, and Ω is between zero and one.

The parameter δ, as shown in Fig. 12(b), represents the average length of the potential's gradient Φ

$$\delta = \frac{1}{M} \sum_{i=1}^{M} \left| \nabla \Phi(x, y)|_{(x'_i, y'_i)} \right|, \qquad i = 1, 2, 3, \ldots, M. \tag{36}$$

Comparing Fig. 12(a) and (b) shows that, similar to Ω, δ is also continuously decreasing. This is because the modules are spread more and more over the placement area during the iterations. With this, the peaks in the module demand $D^{\text{dem}}_{\text{mod}}$ are decreasing, and the supply and demand system D is getting more and more even. As the potential Φ represents D by Poisson's equation (25), the average length δ of the potential's

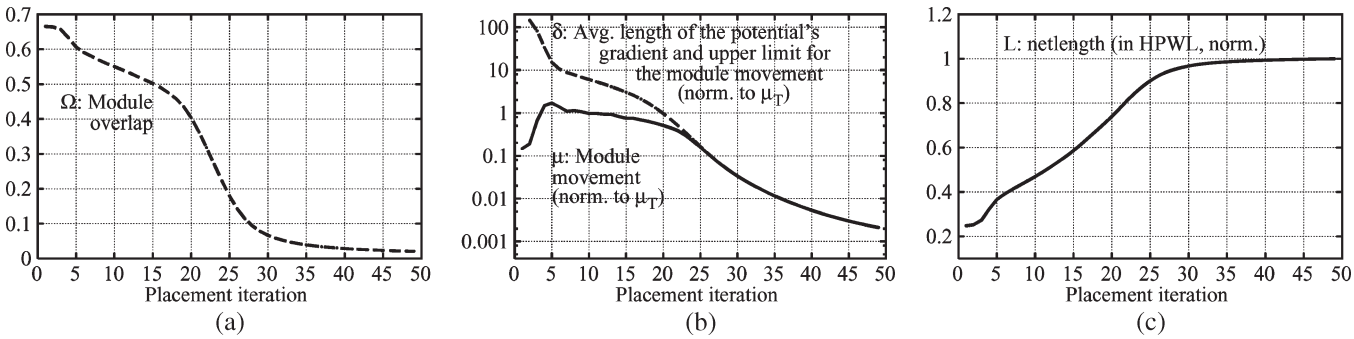

Fig. 12. Demonstration of Kraftwerk2's convergence based on the smooth and continuous progress of some characteristic parameters. Circuit: adaptec1 of the ISPD 2005 contest benchmark suite. (a) Module overlap. (b) Module movement. (c) Netlength.

gradient is decreasing continuously, as shown in Fig. 12(b). The parameter δ is also the upper limit of the module movement μ. This can be shown by approximating first the matrix $\mathbf{C_x}$ with a diagonal matrix $\mathbf{A_x} = \mathrm{diag}(\alpha_{x,i})$. This approximation is valid because $\mathbf{C_x}$ is diagonal dominant, and the average approximation error[3] between $\mathbf{C_x}$ and $\mathbf{A_x}$ is about 12% for various circuits. Thus, the ith equation of the system of linear equations (30) is

$$(\alpha_{x,i} + \overset{\circ}{w}_i)\Delta x_i = -\overset{\circ}{w}_i \left.\frac{\partial}{\partial x}\Phi\right|_{(x_i', y_i')}. \tag{37}$$

With $\beta_{x,i} = \overset{\circ}{w}_i / (\alpha_{x,i} + \overset{\circ}{w}_i)$, (37) becomes

$$\Delta x_i = -\beta_{x,i} \left.\frac{\partial}{\partial x}\Phi\right|_{(x_i', y_i')}, \qquad 0 < \beta_{x,i} \leq 1. \tag{38}$$

Based on (38), the movement $\mu_{x,i}$ of module i in the x-direction is limited by the potential's gradient in one placement iteration

$$\mu_{x,i} = |\Delta x_i| \leq \left| \left.\frac{\partial}{\partial x}\Phi\right|_{(x_i', y_i')} \right| = \delta_{x,i}. \tag{39}$$

Thus, the average movement of all modules μ is limited by δ

$$\mu = \frac{1}{M} \sum_{i=1}^{M} \left| (\Delta x_i, \Delta y_i)^{\mathrm{T}} \right| \leq \delta. \tag{40}$$

This relation between μ and δ, i.e., μ is bound from above by δ, is shown in Fig. 12(b). Moreover, the progress of μ has three characteristics. μ is small in the first placement iteration because the weights of the target points $\overset{\circ}{w}_i$ are initialized with a small value (31). Then, μ is increasing and is around the target movement μ_T because of the quality control described in Section VII. After placement iteration 20, μ is continuously decreasing, as it reached its upper limit δ, and δ is continuously decreasing over all placement iterations.

Fig. 12(c) shows that the netlength L is continuously and steadily increasing up to around placement iteration 20. This is because the module movement μ is almost constant in these iterations. Then, L increases with a lower rate and is almost not changing after iteration 30. This is also due to the module movement μ, which is decreasing after iteration 20 and has a very low value after iteration 30.

The global placement, as shown in Fig. 12(a)–(c), is stopped at around iteration 25, as the module overlap Ω is below 20% there.

Some limitations of the convergence of Kraftwerk2 should be noted. First, if two modules are exactly on top of each other, then they must have different adjacent modules. If not, these critical stacked modules will always be moved in the same way, and the overlap between them will not be removed. However, in all performed experiments, such critical stacked modules were never detected. Another limitation of Kraftwerk2 is the number of placement iterations that is necessary to obtain a complete overlap-free placement. This number of iterations is infinity in theory. However, the module overlap is reduced in each iteration. In addition, Kraftwerk2 is a global placer, and it is stopped if there is little module overlap remaining (e.g., $\Omega < 20\%$). These almost overlap-free placements are obtained in about 25 placement iterations.

XI. Experimental Results

This section demonstrates the high quality and very low CPU time of Kraftwerk2 using various benchmark suites. All benchmark suites were placed on an AMD Opteron 248 machine running at 2.2 GHz and using one CPU core. The memory usage of the biggest benchmark was below 4 GB. To compare with the CPU times of other placers, their CPU times are scaled according to the SPEC CPU2000 benchmark [36]. This scaling factor will be noted as "CPU scaling" in the following. All HPWL netlengths are expressed in meters. The CPU times are in seconds. The results of NTUPlace3 are taken from [8], using a CPU scaling of 1.1. The results of FastPlace3 are taken from [37], and the CPU scaling is 1.2. The results of RQL are taken from [18], with a CPU scaling of 1.2.

Section VII presented the quality control used in Kraftwerk2 to choose the tradeoff between quality and CPU time. The results of Kraftwerk2 presented in the following are obtained at a good tradeoff point. In principle, the results can be improved in quality by about 0.5% with an increase in the CPU time by about a factor of two. On the other hand, the CPU time can be improved by about a factor of two with a decline in quality of about 2%.

In all benchmark suites, the chip area of the circuits is given.

[3]Based on the Frobenius matrix norm $\|E\|_F^2 = \sum_{i,j=1}^{N} e_{ij}^2$.

TABLE II
RESULTS IN THE ISPD 2005 CONTEST BENCHMARK SUITE

Circuit	**Kraftwerk2**		**FastPlace3**		**RQL**		**NTUPlace3**		**APlace2**	**mFAR**	**Dragon**	**mPL5**	**Capo**	**FengShui**
	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	HPWL	HPWL	HPWL	HPWL	HPWL
adaptec1	82.43	262	79.38	353	77.82	751	80.93	883	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
adaptec2	92.85	349	93.08	559	88.51	1247	89.85	906	87.31	91.53	94.72	97.11	99.71	122.99
adaptec3	227.22	713	217.80	2275	210.96	2405	214.20	1944	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
adaptec4	199.43	709	201.36	1411	188.86	2096	193.74	2325	187.65	190.84	200.88	200.94	211.25	337.22
bigblue1	97.67	407	95.68	604	94.98	1160	97.28	1675	94.64	97.70	102.39	98.31	108.21	114.57
bigblue2	154.74	559	155.10	1380	150.03	2261	152.20	3352	143.82	168.70	159.71	173.22	172.30	285.43
bigblue3	343.32	2070	379.88	4642	323.09	4864	348.48	6256	357.89	379.95	380.45	369.66	382.63	471.15
bigblue4	852.40	4147	832.88	6862	797.66	12410	829.16	11308	833.21	876.28	903.96	904.19	1098.76	1040.05
Average	**1.000**	**1.00**	**1.000**	**2.00**	**0.959**	**3.12**	**0.979**	**3.48**	**0.967**	**1.028**	**1.046**	**1.053**	**1.126**	**1.438**

TABLE III
RESULTS IN THE ISPD 2006 CONTEST BENCHMARK SUITE. (a) KRAFTWERK'S RESULTS. AS REQUIRED IN THIS BENCHMARK SUITE, THE CPU FACTOR IS LIMITED TO ±10%. THE "RAW" CPU FACTORS ARE −13.50% AND −10.98%, RESPECTIVELY. (b) RESULTS OF OTHER PLACERS

Circuit	HPWL	Overflow factor	CPU	CPU factor	**Score** HPWL	HPWL+ Overflow	HPWL+ Overflow+ CPU
adaptec5	433.84	3.606%	1618	– 9.35%	1.071	1.032	0.939
newblue1	65.92	0.415%	603	– 8.38%	1.057	1.043	0.956
newblue2	203.91	1.286%	508	– 10.00%*	1.033	1.082	0.975
newblue3	278.51	0.382%	526	– 10.00%*	1.018	1.067	0.961
newblue4	304.24	1.709%	1553	– 8.63%	1.068	1.033	0.945
newblue5	548.38	2.694%	2622	– 9.50%	1.109	1.054	0.957
newblue6	528.59	1.702%	2579	– 9.89%	1.048	1.036	0.936
newblue7	1126.58	3.155%	4828	– 9.06%	1.053	1.051	0.958
Average		**1.869%**		**– 9.35%**	**1.057**	**1.050**	**0.953**

(a)

Placer	Overflow factor	CPU factor	**Score** HPWL	HPWL+ Overflow	HPWL+ Overflow+ CPU
Kraftwerk2	1.87 %	– 9.35 %	1.057	1.050	0.953
NTUPlace3	6.26 %	– 2.61 %	0.976	1.007	0.990
RQL	6.80 %	n.a. %	0.981	1.018	n.a.
Fastplace3	n.a.	– 8.17 %	n.a.	n.a.	1.040
mPL6	1.36 %	1.58 %	1.035	1.020	1.040
mFAR	2.71 %	– 0.12 %	1.108	1.107	1.108
APlace3	3.83 %	5.31 %	1.097	1.107	1.165
Dragon	0.12 %	– 5.90 %	1.331	1.300	1.232
DPlace	9.32 %	– 4.54 %	1.343	1.414	1.364
Capo	0.32 %	2.69 %	1.375	1.344	1.385

(b)

A. ISPD 2005 Contest Benchmark Suite

The ISPD 2005 contest benchmark suite [38], [39] consists of eight circuits with up to 2.2 million movable modules. The quality of placement is measured by the HPWL netlength. Table II shows the results of Kraftwerk2 and of other state-of-the-art placers. The results of APlace2, mFAR, Dragon, mPL5, Capo, and FengShui are taken from [40]. Unfortunately, in [40], no detailed CPU times and no results for the circuits adaptec1 and adaptec3 are published.

On average, Kraftwerk2 is as good as FastPlace3 in netlength but is two times faster. Compared with RQL, Kraftwerk2 has a 5.38% higher netlength but is more than three times faster. Compared with NTUPlace3, Kraftwerk2 has a 2.2% higher netlength but is more than three times faster. Relative to APlace2, Kraftwerk2 has a 3.5% higher netlength but is almost 40 times faster. Based on the netlength of the remaining other placers, Kraftwerk2 is between 2.7% and 30% better.

B. ISPD 2006 Contest Benchmark Suite

The ISPD 2006 contest benchmark suite [41] consists of eight circuits with up to 2.5 million movable modules. The quality of a placer is measured based on the following three parameters: the netlength in HPWL, the CPU factor, and an overflow factor. The overflow factor is zero if the given upper limit d_{up} for the module density is respected everywhere on the chip. Thus, the overflow factor, in combination with a low d_{up}, should assure routability. The CPU factor is derived from the logarithmic ratio between the placer's CPU time and the median over the CPU times of all placers that completed this benchmark suite. For example, a CPU factor of -4% ($+4\%$) represents that the placer's CPU time is two times smaller (greater) than the median CPU time. The three parameters are combined in the following three scoring functions: HPWL, HPWL + Overflow, and HPWL + Overflow + CPU. All three scoring functions are normalized to the best values published in [41].

Table III(a) shows the detailed results of Kraftwerk2. The low overflow factor of 1.87% demonstrates that Kraftwerk2 respects the upper limit d_{up} of the module density very well. The very low CPU factor of -9.35% reveals that Kraftwerk2's CPU time is more than four times smaller than the median CPU time. To obtain the CPU factor, the CPU times in Table III(a) are scaled with 0.86. This is because the results in [41], which are used to calculate the CPU factor, are based on a different machine.

Table III(b) summarizes the results of Kraftwerk2 and of other state-of-the-art placers. Results other than those of RQL, NTUPlace3, and FastPlace3 are taken from [41]. Unfortunately, there are no CPU times available for RQL. Based on the CPU factor, Kraftwerk2 is the fastest placer. According to the main scoring function HPWL + Overflow + CPU, Kraftwerk2 is the best placer. NTUPlace3 is the second best and has a 3.9% higher value in this scoring function. Ignoring the CPU factor, and using the scoring function HPWL + Overflow, Kraftwerk2 is the fourth best. NTUPlace3, RQL, and mPL6 are 4.1%, 3.0%, and 2.9% better, respectively. There are no scores of FastPlace3 in HPWL + Overflow and HPWL available or of RQL in HPWL + Overflow + CPU.

TABLE IV
RESULTS IN MIXED-SIZE AND FLOORPLACEMENT BENCHMARK SUITES. (a) ICCAD 2004 MIXED-SIZED BENCHMARK SUITE. (b) IBM-HB$^+$ FLOORPLACEMENT BENCHMARK SUITE

Circuit	Kraftwerk2		FDP		NTUPlace3		APlace2		mPL5	
	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU
ibm01	2.24	11	2.42	145	2.17	33	2.14	381	2.22	91
ibm02	4.90	27	5.11	284	4.63	63	4.65	872	4.68	264
ibm03	6.61	24	7.08	337	6.65	72	6.71	1015	6.86	300
ibm04	7.63	29	7.69	317	7.21	89	7.57	977	7.69	261
ibm05	9.79	33	n.a.	n.a.	9.66	160	9.69	766	10.09	130
ibm06	6.11	40	6.20	389	5.94	95	6.02	967	6.16	520
ibm07	10.42	52	10.57	607	9.90	219	10.00	1296	9.96	692
ibm08	12.97	85	13.30	719	12.29	235	12.50	1484	11.92	1133
ibm09	11.98	71	13.30	713	12.00	213	12.13	1837	13.15	1363
ibm10	30.15	232	30.70	924	28.49	351	28.83	2649	29.36	1654
ibm11	17.59	107	18.41	950	17.54	336	18.67	3814	17.87	1071
ibm12	31.42	124	36.46	1472	32.07	332	33.42	3663	33.43	1419
ibm13	22.48	147	23.60	1175	22.16	536	22.80	3845	22.52	1079
ibm14	35.13	308	37.84	2185	35.36	1274	35.92	4723	34.99	1588
ibm15	47.58	468	47.69	2468	45.38	1251	46.81	5419	50.88	4989
ibm16	54.17	527	61.27	2792	57.59	1595	54.53	6109	55.21	6200
ibm17	66.63	474	69.45	3577	66.73	2123	65.67	6635	66.96	2131
ibm18	42.36	609	44.88	4369	41.58	2874	41.99	10925	43.99	2477
Average	**1.000**	**1.00**	**1.056**	**9.02**	**0.982**	**3.25**	**0.995**	**23.93**	**1.010**	**9.67**

(a)

Circuit	Kraftwerk2		SCAMPI	
	HPWL	CPU	HPWL	CPU
ibm-HB$^+$01	2.82	10	3.4	68
ibm-HB$^+$02	5.87	26	8.0	154
ibm-HB$^+$03	9.23	16	9.5	115
ibm-HB$^+$04	9.98	21	12.3	158
ibm-HB$^+$06	8.79	12	11.0	187
ibm-HB$^+$07	14.80	16	15.7	110
ibm-HB$^+$08	21.27	19	20.5	207
ibm-HB$^+$09	17.44	18	22.2	200
ibm-HB$^+$10	47.51	47	55.2	351
ibm-HB$^+$11	25.92	23	27.8	159
ibm-HB$^+$12	51.38	43	67.6	447
ibm-HB$^+$13	34.90	23	42.2	231
ibm-HB$^+$14	63.11	42	66.4	295
ibm-HB$^+$15	92.88	46	88.2	414
ibm-HB$^+$16	95.60	54	106.2	337
ibm-HB$^+$17	148.16	96	152.7	424
ibm-HB$^+$18	73.95	52	77.8	211
Average	**1.000**	**1.00**	**1.140**	**8.03**

(b)

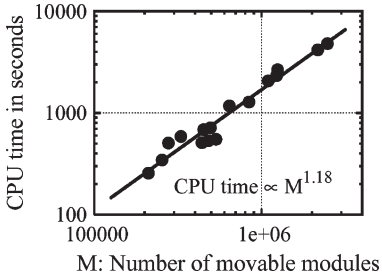

Fig. 13. CPU time of Kraftwerk2, depending on the number of movable modules M per circuit.

C. ICCAD 2004 Mixed-Size Benchmark Suite

The ICCAD 2004 mixed-sized benchmark suite [42] consists of 18 circuits with up to 200 000 movable modules [see Table IV(a)]. The number of big movable modules is about 400 per circuit. The results of FDP are taken from [43], with a CPU scaling of 1.1. The results of APlace2 and mPL5 are taken from [8], with a CPU scaling of 1.1. Kraftwerk2 is the fastest placer, ranging from 3.52 times faster than NTUPlace3 to 24 times faster than APlace2. In the netlength, Kraftwerk2 is 1.0% and 5.3% better than mPL5 and FDP, respectively. Compared with APlace2 and NTUPlace3, Kraftwerk2 has a 0.5% and 1.8% higher netlength, respectively.

D. IBM-HB$^+$ Floorplacement Benchmark Suite

The IBM-HB$^+$ Floorplacement Benchmark Suite [44] consists of 17 circuits [see Table IV(b)] and is derived from the same benchmark suite (IBM/ISPD'98) as the ICCAD 2004 mixed-sized benchmark suite. However, the IBM-HB$^+$ circuits do not have standard cells but consist of about a thousand modules with various dimensions. To legalize these circuits, Kraftwerk2 uses the same technique as legalizing the big modules of the ICCAD 2004 mixed-size benchmark suite. The results of SCAMPI, which is a feature of Capo, are taken from [44], with a CPU scaling of 1.1. Aside from the results of Kraftwerk2 and SCAMPI, no results of other placers that successfully place the complete benchmark suite are available. Compared with SCAMPI, Kraftwerk2 has a 12% better netlength and is about eight times faster.

E. Average-Case Computational Complexity

Fig. 13 shows the CPU time of Kraftwerk2 versus the number M of movable modules per circuit. The results are based on the ISPD 2005/2006 contest benchmark suites. Using Fig. 13, the average-case computational complexity of Kraftwerk2 is $\Theta(M^{1.18})$ and, thus, nearly linear. Hence, the placer can cope easily with future circuits having an increasing number of modules.

XII. CONCLUSION

This paper presents the force-directed quadratic placer "Kraftwerk2," which offers high quality and excellent computational efficiency. Due to the systematic force implementation, Kraftwerk2 converges such that the module overlap is reduced in each placement iteration. By using the Bound2Bound net model, Kraftwerk2 accurately optimizes the HPWL netlength. Other features of the placer are the prevention of halos around large modules, the control of the module density, and the control of the tradeoff between runtime and quality by using one parameter.

REFERENCES

[1] W.-J. Sun and C. Sechen, "Efficient and effective placement for very large circuits," in *Proc. IEEE/ACM ICCAD*, 1993, pp. 170–177.
[2] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu, and I. L. Markov, "Capo: Robust and scalable open-source min-cut floorplacer," in *Proc. ACM/SIGDA ISPD*, 2005, pp. 224–226.
[3] T. Taghavi, X. Yang, and B.-K. Choi, "Dragon2005: Large-scale mixed-size placement tool," in *Proc. ACM/SIGDA ISPD*, 2005, pp. 245–247.
[4] A. R. Agnihorti, S. Ono, C. Li, M. C. Yildiz, A. Khathate, C.-K. Koh, and P. H. Madden, "Mixed block placement via fractional cut recursive bisection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 5, pp. 748–761, May 2005.

[5] W. Naylor, R. Donelly, and L. Sha, "Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer," U.S. Patent 6 301 693, Oct. 9, 2001.
[6] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 5, pp. 734–747, May 2005.
[7] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proc. ACM/SIGDA ISPD*, 2005, pp. 185–192.
[8] 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. IEEE/ACM ICCAD*, 2006, pp. 187–192.
[9] A. R. Agnihotri and P. H. Madden, "Fast analytic placement using minimum cost flow," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2007, pp. 128–134.
[10] R.-S. Tsay, E. S. Kuh, and C.-P. Hsu, "PROUD: A sea-of-gates placement algorithm," *IEEE Des. Test Comput.*, vol. 5, no. 6, pp. 44–56, Dec. 1988.
[11] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 3, pp. 356–365, Mar. 1991.
[12] U. Brenner and M. Struzyna, "Faster and better global placement by a new transportation algorithm," in *Proc. ACM/IEEE DAC*, 2005, pp. 591–596.
[13] G.-J. Nam, S. Reda, C. J. Alpert, P. G. Villarrubia, and A. B. Kahng, "A fast hierarchical quadratic placement algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 4, pp. 678–691, Apr. 2006.
[14] B. Hu and M. Marek-Sadowska, "FAR: Fixed-points addition & relaxation based placement," in *Proc. ACM/SIGDA ISPD*, 2002, pp. 161–166.
[15] N. Viswanathan and C. C.-N. Chu, "Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 5, pp. 722–733, May 2005.
[16] B. Hu and M. Marek-Sadowska, "Multilevel fixed-point-addition-based VLSI placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 8, pp. 1188–1203, Aug. 2005.
[17] A. Kennings and K. P. Vorwerk, "Force-directed methods for generic placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2076–2087, Oct. 2006.
[18] N. Viswanathan, G.-J. Nam, C. J. Alpert, P. Villarrubia, H. Ren, and C. Chu, "RQL: Global placement via relaxed quadratic spreading and linearization," in *Proc. ACM/IEEE DAC*, 2007, pp. 453–458.
[19] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. ACM/IEEE DAC*, Jun. 1998, pp. 269–274.
[20] Z. Xiu, J. D. Ma, S. M. Fowler, and R. A. Rutenbar, "Large-scale placement by grid-warping," in *Proc. ACM/IEEE DAC*, 2004, pp. 351–356.
[21] Z. Xiu and R. Rutenbar, "Mixed-size placement with fixed macrocells using grid-warping," in *Proc. ACM/SIGDA ISPD*, 2007, pp. 103–109.
[22] P. Chong and C. Szegedy, "A morphing approach to address placement stability," in *Proc. ACM/SIGDA ISPD*, 2007, pp. 95–102.
[23] P. Spindler and F. M. Johannes, "Fast and robust quadratic placement based on an accurate linear net model," in *Proc. IEEE/ACM ICCAD*, 2006, pp. 179–186.
[24] P. Spindler and F. M. Johannes, "Kraftwerk—A fast and robust quadratic placer using an exact linear net model," in *Modern Circuit Placement—Best Practices and Results*, 978th ed. G.-J. Nam and J. Cong, Eds. New York: Springer-Verlag, Sep. 2007, ch. 4, pp. 59–91.
[25] G. Sigl, K. Doll, and F. M. Johannes, "Analytical placement: A linear or a quadratic objective function?" in*Proc. ACM/IEEE DAC*, Francisco, CA, 1991, pp. 427–432.
[26] J. Vygen, "Algorithms for large-scale flat placement," in *Proc. ACM/IEEE DAC*, 1997, pp. 746–751.
[27] C. Chu, "FLUTE: Fast lookup table based wirelength estimation technique," in *Proc. IEEE/ACM ICCAD*, 2004, pp. 696–701.
[28] K. M. Hall, "An r-dimensional quadratic placement algorithm," *Manage. Sci.*, vol. 17, no. 3, pp. 219–229, Nov. 1970.
[29] M. Kowarschik and C. Weiß, "DiMEPACK—A cache-optimized multigrid library," in *Proc. Int. Conf. Parallel Distrib. Process. Techn. Appl.*, H. Arabnia, Ed., pp. 425–430, Jun. 2001.
[30] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proc. DATE*, Apr. 2007, pp. 1226–1231.
[31] B. Obermeier and F. M. Johannes, "Temperature-aware global placement," in *Proc. Asia South Pacific Des. Autom. Conf.*, Yokohama, Japan, Jan. 2004, vol. 1, pp. 143–148.
[32] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design," U.S. Patent 6 370 673, Apr. 9, 2002.
[33] J. Vygen, "Algorithms for large-scale flat placement," in *Proc. ACM/IEEE DAC*, 1997, pp. 746–751.
[34] J. Cong and M. Xie, "A robust detailed placement for mixed-size IC designs," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2006, pp. 188–194.
[35] B. Landman and R. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Trans. Comput.*, vol. C-20, no. 12, pp. 1469–1479, Dec. 1971.
[36] S.P.E. Corporation, *SPEC CPU 2000*. [Online]. Available: http://www.spec.org/cpu2000
[37] N. Viswanathan, M. Pan, and C. Chu, "Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2007, pp. 135–140.
[38] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ISPD2005 placement contest and benchmark suite," in *Proc. ACM/SIGDA ISPD*, May 2005, pp. 216–219.
[39] *ISPD 2005 Placement Contest*, Mar. 2005. [Online]. Available: http://www.sigda.org/ispd2005/contest.htm
[40] A. B. Kahng, S. Reda, and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," in *Proc. IEEE/ACM ICCAD*, 2005, pp. 890–897.
[41] *ISPD 2006 Placement Contest*, Mar. 2006. [Online]. Available: http://www.sigda.org/ispd2006/contest.html
[42] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov, "Unification of partitioning, placement and floorplanning," in *Proc. IEEE/ACM ICCAD*, 2004, pp. 550–557.
[43] K. Vorwerk and A. Kennings, "Mixed-size placement via line search," in *Proc. IEEE/ACM ICCAD*, 2005, pp. 899–904.
[44] A. N. Ng, R. Aggarwal, V. Rachmandran, and I. L. Markov, "Solving hard instances of floorplacement," in *Proc. ISPD*, 2006, pp. 170–177.

Peter Spindler received the B.S. and Dipl.-Ing. degrees in electrical engineering and information technology from the Technische Universität München (TUM), Munich, Germany, in 2004 and 2005, respectively, where he has been working toward the Dr.-Ing. degree in the Department of Electrical Engineering and Information Technology, Institute for Electronic Design Automation, since 2005.

Currently, he is a Research Assistant with TUM. His research interests include computer-aided design of integrated circuits and systems, with particular emphasis on layout synthesis.

Ulf Schlichtmann (S'88–M'90) received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering and information technology from the Technische Universität München (TUM), Munich, Germany, in 1990 and 1995, respectively.

From 1994 to 2003, he was with the Semiconductor Group of Siemens AG which, in 1999, became Infineon Technologies AG, where he held various technical and management positions in design automation, design libraries, IP reuse, and product development. Since 2003, he has been with TUM as a Professor and the Head of the Institute for Electronic Design Automation. His research interests are in computer-aided design of electronic circuits and systems, with special emphasis on designing robust systems.

Frank M. Johannes received the Dipl.-Ing. degree in electrical engineering from the Technische Universität Karlsruhe, Karlsruhe, Germany, in 1968, and the Dr.-Ing. degree from the Universität Erlangen–Nürnberg, Erlangen, Germany, in 1973.

From 1968 to 1975, he was with the Regional Computing Center, Erlangen, working in the fields of computer networking and computer-aided software engineering. Since 1976, he has been with the Department of Electrical Engineering and Information Technology, Institute for Electronic Design Automation, Technische Universität München (TUM), Munich, Germany. From 1993 to the present, he has been an Associate Professor with this faculty, heading a research group for synthesis methods for electronic design automation. His major research interest is on layout synthesis, with special attention on the placement problem, where his group won the ISPD placement contest in 2006.