Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement

Peter Spindler and Frank M. Johannes
Institute for Electronic Design Automation, Technische Universitaet Muenchen, Munich, Germany

Abstract— **This paper presents a fast and accurate routing demand estimation called RUDY and its efficient integration in a force-directed quadratic placer to optimize placements for routability.**

RUDY is based on a *R*ectangular *U*niform wire *D*ensit*Y* per net and accurately models the routing demand of a circuit as determined by the wire distribution after final routing. Unlike published routing demand estimation, RUDY depends neither on a bin structure nor on a certain routing model to estimate the behavior of a router. Therefore RUDY is independent of the router.

Our fast and robust force-directed quadratic placer is based on a generic demand-and-supply model and is guided by the routing demand estimation RUDY to optimize placements for routability. This yields a placer which simultaneously reduces the routing demand in congested regions and increases the routing supply there. Therefore our placer fully utilizes the potential to optimize the routability. This results in the best published routed wirelength of the IBMv2 benchmark suite until now. In detail, our approach outperforms mPL, ROOSTER, and APlace by 9%, 8%, and 5%, respectively. Compared by the CPU times, which ROOSTER needs to place this benchmark, our routability optimization placer is eight times faster.

1. Introduction

Physical design produces the geometrical data for the fabrication of a VLSI circuit based on the netlist of the circuit. Traditionally, this process is partitioned into two steps: placement and routing. In order to place routable circuits, these two steps have to be combined by estimating the routability during placement. Since the next generation VLSI circuits will have tens of millions modules, (i) fast and accurate techniques are needed to estimate the routing demand in placement and (ii) efficient methods are needed to optimize routability in placement.

Different solutions for both problems (i) and (ii) appeared in the last few years:

(i) Estimation of the Routing Demand
All published methods to estimate the routing demand divide the chip area into bins and estimate the routing demand in each bin.

a) Routing model
A common technique to estimate the routing demand is to use a certain routing model, which provides possible routes for each net. The number of possible routes crossing the border of a bin reflects its routing demand. The authors of [1] present a simple routing model which just uses the border of the bounding box of a net as possible routes. Since the results in [2] show that this simple model does not correlate with the behavior of a router, a more accurate routing model is proposed in [3], which breaks down multi-pin nets into two-pin connections and models two-pin connections by routes with different number of bends. This probabilistic routing model is improved in [4], and [5] by adjusting its result to the result of a router. The authors of [6] state that one- and two-bend routes between a two-pin connection are enough.

b) Pin density
The estimation technique based on a routing model has problems if a net does not pass the border of a bin. Therefore [7] and [8] improve the estimation of routing demand by utilizing the number of pins within a bin.

c) Rent's Rule
A method to estimate the routing demand without a routing model is proposed in [9], [10], and [11] by applying Rent's rule.

d) Distribution of number of nets per bin
Another technique to estimate the routing demand is the analysis of the distribution of the number of nets per bin [12].

(ii) Routability optimization in placement
Unroutable circuits have congested regions where the routing demand of the nets is higher than the supply by the routing layers. Hence there exist two main approaches to optimize placement for routability. The direct approach reduces routing demand in congested regions by moving those modules which are connected to the nets causing the overflow in the routing demand. The indirect approach is based on the fact that the lowest routing layer is usually blocked by the modules. Thus the indirect approach increases the routing supply in congested regions by reducing the module density there.

The direct approach is often used as a post-process to tune an already placed circuit for routability. A post-process utilizing Simulated Annealing is described in [1], [11], and [13]. A flow-based method is presented in [2], and [14]. Linear programming is used in [15].

The indirect approach to optimize routability is mostly used during placement. In [7] and [16], a quadratic placer is described which inflates the modules in congested regions. The authors of [17] present a quadratic placer which reduces module density in congested regions by growing these regions. In [18], a min-cut placer is shown, which allocates white space, i.e. reduces module density, during top-down placement in congested regions.

State-of-the-art placers with routability optimization
mPL [19] is a multilevel analytical placer based on non-linear optimization and estimates the routing demand based on a two-pin connection routing model developed in [20]. Routability is achieved in global placement by moving certain modules to reduce the routing demand. In final placement, a white space allocation (WSA) method is used based on recursively partitioning the placement area and shifting the cut lines according to the routing demand. Thus mPL utilizes the direct approach during placement and the indirect approach after placement.

ROOSTER [21] as a feature of Capo 10 is a min-cut placer. It models nets by Rectilinear Steiner Minimal Trees [22], estimates the routing demand by a probabilistic routing model [6] and utilizes the white space allocation (WSA) method of [19] during top-down placement and in final placement. Therefore ROOSTER applies the indirect approach to optimize routability.

APlace [23] is a multilevel analytical placer based on non-linear optimization and estimates the routing demand by a probabilistic routing model [4]. Routability is optimized during placement by decreasing module density in congested areas, i.e. by the indirect approach.

Our estimation of the routing demand based on RUDY, as presented in this paper, is characterized by the following enhancements to other estimation techniques:

- RUDY is defined per net by a *R*ectangular *U*niform wire *D*ensit*Y*.
- RUDY models the wire distribution over the chip area.
- RUDY depends neither on a bin structure nor on a certain routing model.
- RUDY estimates the real routing demand very accurately.
- RUDY needs very low CPU time.

The following properties distinguish our routability-driven placement approach from other approaches:

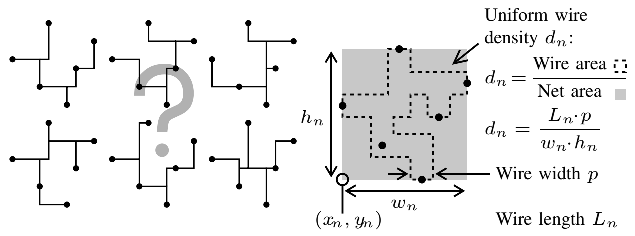
- Simultaneous application of the direct and indirect technique to optimize routability, i.e. our placer concurrently reduces the routing demand as well as it increases the routing supply in congested regions.

- Convenient and efficient implementation of routability as a demand-and-supply model in a fast and robust force-directed quadratic placer.

The rest of the paper is organized as follows: section 2 describes the routing demand estimation based on RUDY. In section 3 we explain our force-directed quadratic placer based on a generic demand-and-supply model. Our routability-driven placement approach together with some general statements about routability optimization during placement is described in section 4. Experimental results are provided in section 5, followed by the conclusion in section 6.

2. Estimation of the Routing Demand

The most common approach to estimate the routing demand is to utilize a routing model which provides possible routes of a net. Figure 1(a) shows different routes of a six-pin net if a Rectilinear Steiner Minimal Tree (RSMT) is used as a routing model. Since all of these routes have the same minimal length, it is difficult to predict which one a router will use. Moreover a router will use a totally different route if one net interferes with another net. Therefore the estimation of the routing demand based on a routing model depends highly on how good the actual router is modeled. In addition, published approaches to estimate the routing demand are based on a bin structure and therefore have to cope with local uncertainty which arises if a net does not cross a bin border.


Uniform wire density d_n:
$d_n = \frac{\text{Wire area}}{\text{Net area}}$
$d_n = \frac{L_n \cdot p}{w_n \cdot h_n}$
h_n
Wire width p
w_n
(x_n, y_n)
Wire length L_n

(a) Six possible rectilinear Steiner minimal trees (b) RUDY of the net n

Fig. 1. Six-pin net: (a) six possible rectilinear Steiner minimal trees and (b) our estimation of the routing demand by RUDY (*R*ectangular *U*niform wire *D*ensit*Y*)

To solve these problems of traditional estimation approaches, we present RUDY. RUDY stands for *R*ectangular *U*niform wire *D*ensit*Y* and is based on two assumptions: (1) All routers will try hard to route each net within the rectangle enclosing all its pins. (2) Since there are thousands and even millions of nets in a modern VLSI circuit, the enclosing rectangle of one net is small compared to the chip's dimension and the routing demand of one net is marginal compared to the routing demand of the circuit. Thus it is not necessary for a single net to predict its routing demand accurately within its enclosing rectangle.

In detail, the RUDY is defined per net $n = 1, 2, 3, ..., N$ by a uniform wire density d_n within the enclosing rectangle of net n. This wire density d_n is the ratio of the wire area WA_n and the net area NA_n:

$$d_n = \frac{WA_n}{NA_n} \tag{1}$$

The RUDY of one single six-pin net n is displayed in figure 1(b).

The enclosing rectangle of the net n is characterized by the lower left corner (x_n, y_n), the width w_n and the height h_n. Thus the net area NA_n is defined by $NA_n = w_n \cdot h_n$.

The wire area WA_n is calculated by the wire length L_n times the wire width p: $WA_n = L_n \cdot p$. The wire width p is defined by the average wire-to-wire pitch $\bar{p}$ and number of routing layers l: $p = \frac{\bar{p}}{l}$. The wire length L_n is the estimated routed wire length of the net and can be calculated for example by the Half Perimeter Wirelength (HPWL) or by the length of the Rectilinear Steiner Minimal Tree (RSMT). Using the above mentioned definition of the enclosing rectangle of the net n, the HPWL of the net n is $w_n + h_n$. Based on the observation in section 4 (see also figure 3) that the HPWL correlates to the routed wirelength as good as the RSMT length does, but that the HPWL is determined much faster [22], we will use the HPWL as the estimation for the routed wirelength L_n.

To calculate the estimation of the routing demand based on the RUDY, a rectangle function $R(x, y; x_{ll}, y_{ll}, w, h)$ is needed, which is defined in the x-y-plane and has the parameters lower left corner (x_{ll}, y_{ll}), the width w, and the height h:

$$R(x, y; x_{ll}, y_{ll}, w, h) = \begin{cases} 1 & \text{if } 0 \le x - x_{ll} \le w \wedge 0 \le y - y_{ll} \le h \\ 0 & \text{else} \end{cases} \tag{2}$$

The estimation of the routing demand $D^{dem}_{rout}(x, y)$ of N nets using RUDY is the superposition of the rectangle functions of all nets, weighted by the wire density d_n of each net:

$$\textit{RUDY:} \qquad D^{dem}_{rout}(x, y) = \sum_{n=1}^{N} d_n \cdot R(x, y; x_n, y_n, w_n, h_n) \tag{3}$$

To compare RUDY with another routing demand estimation approach, we used the utility "CongestionMaps Plotter" of the UMICH package [24]. This utility is an implementation of the common routing demand estimation approach of Westra et al. [6], which breaks down multi-pin nets into two-pin connections and models the possibles routes between two-pin connections by one- and two-bend routes. To evaluate the quality of RUDY and the quality of Westras' approach, we calculated the error between both estimations and the real routing demand. The real routing demand is defined by the wire distribution after final routing with Cadence WarpRoute. The estimation error was determined by overlaying the estimated and the real routing demand by a fine grid, computing the wire usage in every bin and subtracting the estimated wire usage from the real wire usage in every bin. Table 1 shows the comparison between RUDY and Westras' approach. Since the standard deviation σ_{RUDY} and σ_{Westra} of the estimation errors of both approaches are almost the same, RUDY estimates the real routing demand as good as the estimation approach of Westra et al. This demonstrates that it is not necessary to predict the route of each single net as done by Westras' approach but it is sufficient to model the routing demand of each net by a *R*ectangular *U*niform wire *D*ensit*Y*, i.e. RUDY. Therefore the above mentioned assumptions (1) and (2) of RUDY are justified by experiments. Moreover table 1 shows that RUDY is almost $11\times$ faster than the estimation by Westras' approach.

Figures 2(a) and (c) show the estimation of the routing demand by RUDY of two circuits of the IBMv2 benchmark suite [25]. The real routing demand of these circuits are displayed in figures 2(b) and (d). Comparing the figures of RUDY with the figures of the real routing demand demonstrates that the regions with high routing demand (dark color) and the regions with low routing demand (light color) are well predicted by RUDY.

In summary, RUDY is a novel, very efficient, and accurate estimation of the routing demand, which depends neither on a certain routing model nor on a bin structure. Moreover RUDY is a generic estimation method for the routing demand, which can be integrated in a placer, as described in section 4, as well as it can be integrated in a router to drive routing by congestion similar to the approach described in [26].

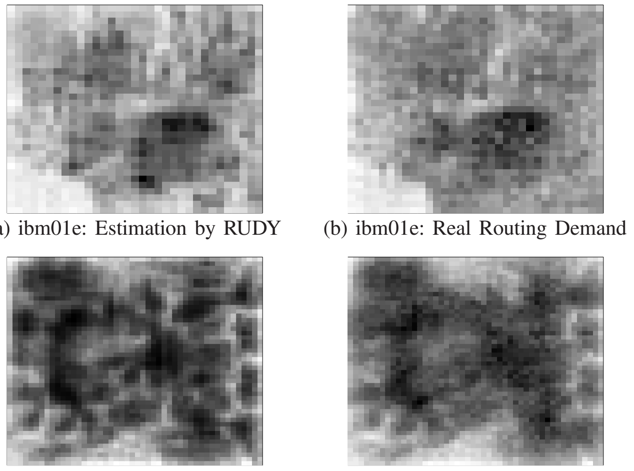
3. Force-directed Quadratic Placement

We use the force-directed quadratic placer as described in [27] because it is fast, robust and can be extended in a convenient way to optimize circuits for routability.

Quadratic placers in general are based on the representation of the circuit's netlist by a binary graph $B(\mathcal{M}, \mathcal{E})$, with the set of edges $\mathcal{E}$ connecting pairs of modules in set $\mathcal{M}$. The Euclidean length of each edge $e \in \mathcal{E}$ is weighted, squared and added up to the cost function

		RUDY		Westra et al. [6]	
Circuit	# Nets	σ_{RUDY}	CPU [s]	σ_{Westra}	CPU [s]
ibm01e/h	11.753	0.23	0.05	0.23	0.46
ibm02e/h	18.429	0.29	0.08	0.24	0.83
ibm07e/h	44.394	0.18	0.15	0.18	1.84
ibm08e/h	47.944	0.18	0.20	0.17	2.18
ibm09e/h	50.393	0.19	0.19	0.20	2.03
ibm10e/h	64.227	0.19	0.29	0.19	2.86
ibm11e/h	67.016	0.19	0.24	0.19	2.59
ibm12e/h	68.376	0.19	0.29	0.18	3.25
	Average	**1.00**	**1.00**	**0.97**	**10.66**

Table 1. Routing demand estimation by RUDY and by the approach of Westra et al. [6]. σ_{RUDY} and σ_{Westra} represent the standard deviation of the estimation error of RUDY and of Westras' approach, respectively. All results are based on circuits of the IBMv2 benchmark suite [25]. Please note that this benchmark suite has no circuits ibm03e/h-ibm06e/h. The CPU times were determined on a Pentium 4 running at 3.2 GHz.


(a) ibm01e: Estimation by RUDY (b) ibm01e: Real Routing Demand

(c) ibm12h: Estimation by RUDY (d) ibm12h: Real Routing Demand

Fig. 2. Estimated routing demand by RUDY (a) and (c), the real routing demand (b) and (d), based on the wire distribution after final routing. White represents low demand, black represents high demand. The circuits ibm01e and ibm12h are from the IBMv2 benchmark.

Γ of quadratic placement. Thus the cost function Γ reflects the total net length in a quadratic metric. Collecting the positions (x_m, y_m), $m = 1, 2, 3, ..., M$ of the M movable modules in vectors $\mathbf{x}$ and $\mathbf{y}$ gives the matrix-vector notation of the cost function Γ [28]:

$$\Gamma = \frac{1}{2}\mathbf{x}^\mathbf{T}\mathbf{C_x}\mathbf{x} + \mathbf{x}^\mathbf{T}\mathbf{d_x} + \frac{1}{2}\mathbf{y}^\mathbf{T}\mathbf{C_y}\mathbf{y} + \mathbf{y}^\mathbf{T}\mathbf{d_y} + const \tag{4}$$

A net model is applied in quadratic placement in order to represent the nets of a circuit by just two-pin connections. Traditionally the clique net model is used, which represents a net by all its possible two-pin connections. In contrast, we use the BoundingBox net model of [27], which uses just those two-pin connections of a net, which are joined to the bounds of the net's box, i.e. to the bounds of the rectangle enclosing the net's pins. The BoundingBox net model has the major advantage that it expresses the total HPWL accurately in the cost function Γ of quadratic placement.

To obtain the module positions with minimal total net length, Γ is differentiated with respect to x and to y and set to zero, which yields two linear equations for x- and y-direction. The linear equation in x-direction is:

$$\mathbf{C_x}\mathbf{x} + \mathbf{d_x} = \mathbf{0} \tag{5}$$

The y-direction is formulated accordingly and therefore not given separately in the following.

The two-pin connections between the modules can be interpreted as elastic springs. Thus (5) is the net force $\mathbf{F_x^{net}} = \mathbf{C_x}\mathbf{x} + \mathbf{d_x}$, which attracts the modules resulting in a lot of overlap between the modules. To reduce the overlap and to spread the modules over the chip area, force-directed quadratic placement utilizes additional forces. Our approach utilizes two additional forces [27]: a hold force $\mathbf{F_x^{hold}}$ and a move force $\mathbf{F_x^{move}}$. Furthermore, the modules are spread iteratively over the chip area. This means that the modules are moved from the old position $\mathbf{x}'$ to the new position $\mathbf{x}$ in each iteration by the change in the module positions of $\mathbf{\Delta x} = \mathbf{x} - \mathbf{x}'$. The hold force $\mathbf{F_x^{hold}}$ is defined by $\mathbf{F_x^{hold}} = -\left(\mathbf{C_x}\mathbf{x}' + \mathbf{d_x}\right)$. The move force $\mathbf{F_x^{move}}$ is determined by the distribution $D(x, y)$ representing for example the distribution of the modules on the chip area. Based on this distribution $D(x, y)$, a potential Φ is calculated by Poisson's equation:

$$\triangle\Phi\left(x, y\right) = -D(x, y) \tag{6}$$

It can be stated that the potential Φ is high in regions where the distribution $D(x, y)$ is high and vice versa. Hence the derivative of the potential Φ can be used for the move force $\mathbf{F_x^{move}}$ to move the modules away from high density regions towards low density regions and thereby reduce the module overlaps.

To calculate the move force $\mathbf{F_x^{move}}$ the derivative of the potential Φ is determined at each module position:

$$\mathbf{\Phi_x} = \left(\frac{\partial}{\partial x}\Phi\Big|_{(x_1', y_1')}, \frac{\partial}{\partial x}\Phi\Big|_{(x_2', y_2')}, ..., \frac{\partial}{\partial x}\Phi\Big|_{(x_M', y_M')}\right)^T \tag{7}$$

Then target points $\mathring{\mathbf{x}}$ are calculated for each module by $\mathring{\mathbf{x}} = \mathbf{x}' - \mathbf{\Phi_x}$. Each module is connected to its target point by a two-pin connection with weight $\mathring{w}_m$, $m = 1, 2, 3, ..., M$. These weights $\mathring{w}_m$ can be used to control the placement process. Altogether, the move force $\mathbf{F_x^{move}}$ is calculated by $\mathbf{F_x^{move}} = \mathring{\mathbf{C}}_\mathbf{x}\left(\mathbf{x} - \mathring{\mathbf{x}}\right)$. The matrix $\mathring{\mathbf{C}}_\mathbf{x}$ is a diagonal matrix with the weights $\mathring{w}_m$ of the target point connections as the entries $\mathring{\mathbf{C}}_\mathbf{x} = diag(\mathring{w}_m)$.

The sum of the net force $\mathbf{F_x^{net}}$ and the two additional forces $\mathbf{F_x^{hold}}$ and $\mathbf{F_x^{move}}$ gives the total force $\mathbf{F_x} = \mathbf{F_x^{net}} + \mathbf{F_x^{hold}} + \mathbf{F_x^{move}}$. Setting this total force to zero yields the following linear system of equations:

$$\left(\mathbf{C_x} + \mathring{\mathbf{C}}_\mathbf{x}\right)\mathbf{\Delta x} + \mathring{\mathbf{C}}_\mathbf{x}\mathbf{\Phi_x} = \mathbf{0} \tag{8}$$

Solving (8) for $\mathbf{\Delta x}$ gives the new module positions in the current iteration and can be done numerically by the conjugate-gradient method.

As mentioned above, the move force $\mathbf{F_x^{move}}$ is determined by the distribution $D(x, y)$. This distribution reflects a generic demand-and-supply system:

$$D(x, y) = D^{dem}(x, y) - D^{sup}(x, y) \tag{9}$$

If the demand-and-supply is balanced, i.e. the integral over the distribution is zero

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} D(x, y)\ dx\, dy = 0 \tag{10}$$

and the demand can be moved by our force-directed quadratic placer, we can prove that in each iteration the demand is more adapted to the supply. Thus our placement algorithm converges to a state where the demand is completely adapted to the supply at each position (x, y): $D^{dem}(x, y) = D^{sup}(x, y)$ for all (x, y).

To spread the modules on the chip with our placement approach, the distribution $D(x, y)$ has to represent the demand of the modules and the supply for the modules. The demand $D^{dem}_{mod}(x, y)$ of the modules is the area needed by the modules. Using the rectangle function R in Eq. (2) and the information that module m has the center position (x_m, y_m), width w_m, and height h_m, the demand $D^{dem}_{mod}(x, y)$ is calculated for the M movable and F fixed modules by:

$$D^{dem}_{mod}(x, y) = \sum_{m=1}^{M+F} R\left(x, y; x_m - \frac{w_m}{2}, y_m - \frac{h_m}{2}, w_m, h_m\right) \tag{11}$$

The supply $D^{sup}_{mod}(x,y)$ for the modules is the area which the chip provides for the modules. If the chip has the lower left corner (x_{Chip}, y_{Chip}), the width w_{Chip}, and the height h_{Chip}, the supply $D^{sup}_{mod}(x,y)$ is:

$$D^{sup}_{mod}(x,y) = R(x,y;x_{Chip},y_{Chip},w_{Chip},h_{Chip}) \qquad (12)$$

Using this supply, the modules will be spread all over the whole chip. To reduce the net length in chips with low module utilization u_{mod}, it is better to spread the modules considering an upper limit $d_{mod,up} > u_{mod}$ for the module density. This can be implemented by setting the module supply $D^{dem}_{mod}(x,y)$ to $d_{mod,up}$ in regions where there is a module demand and to zero elsewhere.

The distribution $D_{mod}(x,y)$ of the modules is the module demand minus the module supply:

$$D_{mod}(x,y) = D^{dem}_{mod}(x,y) - \alpha_{mod} \cdot D^{sup}_{mod}(x,y) \qquad (13)$$

The factor α_{mod} is to fulfill (10) if $D(x,y) = D_{mod}(x,y)$, i.e. to normalize the supply to the demand.

Although our quadratic placer can produce placements without module overlap, it is stopped at an iteration where there is only some small module overlap, because it cannot arrange the modules on the chip's rows. Thus our final placer removes the remaining module overlap and arranges the modules on the chip's rows. This is done by placing each module at the next best place according to a certain cost function. In routability-driven placement we take the movement as the cost function, to change the placement as little as possible by the final placer. The final placer is quite fast and the CPU times presented in the result section 5 include the CPU time of the final placer.

4. Routability Driven Placement

A placed circuit can sometimes not be routable because there are regions on the chip where the demand of the routing is higher than the supply for routing. This problem can be solved if the modules are placed in a way that the routing demand is adapted to the routing supply.

Our placement approach provides a convenient solution by moving the modules based on a routing distribution $D_{rout}(x,y)$, which reflects the routing demand $D^{dem}_{rout}(x,y)$ and routing supply $D^{sup}_{mod}(x,y)$. An accurate and fast estimation of the routing demand $D^{dem}_{rout}(x,y)$ by RUDY in Eq. (3) is presented in section 2. The routing supply is the area of the chip:

$$D^{sup}_{rout}(x,y) = R(x,y;x_{Chip},y_{Chip},w_{Chip},h_{Chip}) \qquad (14)$$

Preplaced macros, which block some or all routing layers, can be excluded from the routing supply by subtracting them using the rectangle function R.

The routing distribution $D_{rout}(x,y)$ is the routing demand minus the routing supply:

$$D_{rout}(x,y) = D^{dem}_{rout}(x,y) - \alpha_{rout} \cdot D^{sup}_{rout}(x,y) \qquad (15)$$

The factor α_{rout} is to fulfill (10) if $D(x,y) = D_{rout}(x,y)$, i.e. to normalize the supply to the demand.

If the distribution $D(x,y)$ reflects just the routing distribution $D_{rout}(x,y)$, then the modules might not be spread over the chip because their module distribution $D_{mod}(x,y)$ is not taken into account during placement. Thus only a combination of both distributions $D_{rout}(x,y)$ and $D_{mod}(x,y)$ will produce routable chips with the modules spread over the chip:

$$D(x,y) = w_{rout} \cdot D_{rout}(x,y) + (1 - w_{rout}) \cdot D_{mod}(x,y) \qquad (16)$$

The routing weight w_{rout} in the distribution (16) reflects the importance of routability in our placement approach: with $w_{rout} = 0$ routability is not considered at all and with $w_{rout} = 1$ just routability is considered. Figure 3 is based on the circuit ibm01e of the IBMv2 benchmark suite and shows the impact of the routing weight w_{rout} to the routed wirelength, to the net length measured in HPWL and RSMT, and to the standard deviation of the routing demand distribution. This standard deviation was determined by overlaying the routing demand after final routing by a fine grid, computing the wire usage in every bin and calculating the standard deviation of these wire usages.

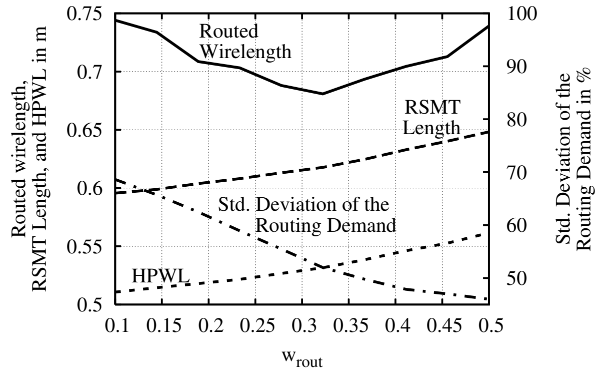


Fig. 3. Impact of the routing weight w_{rout} to the routed wirelength, to the net length measured by HPWL and by Rectilinear Minimal Steiner Tree (RSMT) length and to the standard deviation of the routing demand distribution. Results are based on circuit ibm01e of the IBMv2 benchmark suite.

Some general statements about routability optimization in placement can be derived from the figure 3:

1) *With increasing routing weight, the peaks in the routing demand distribution are reduced, represented in the decrease in the standard deviation of the routing demand distribution.*
This is because the fraction of the routing distribution in the total distribution increases and thus our placer tries harder to adapt the routing demand to the routing supply. Since the routing supply is constant over the chip area, the placer evens the routing demand distribution, resulting in a decreasing standard deviation of the routing demand distribution.

2) *With increasing routing weight, the net length measured in HPWL and RSMT increases.*
If the routing weight is zero, routability is not considered and the modules are at their optimal places for the lowest net length measured in HPWL or RSMT. But with increasing routing weight, routability is considered more and the modules are moved away from their optimal positions, which increases the HPWL and RSMT length.

3) *There is a trade-off between an evenly distributed routing demand and a low net length.*
This trade-off can be controlled by the routing weight w_{rout} and results in an optimal routing weight w^*_{rout}, at which the routed wirelength is minimal. The optimal routing weight w^*_{rout} depends somewhat on the circuit and is $w^*_{rout} = 0.32$ in figure 3.

4) *The Rectilinear Steiner Minimal Tree (RSMT) length does not predict exactly the routed wirelength.*
Since routing and the RSMT are based on horizontal and vertical connections, the length of the RSMT is a good prediction of the routed wirelength. But there exists an inherent prediction error because of two main reasons: (i) In routing, a pin can be connected within an area called pin site. Since the RSMT is based on connecting points and not areas, the RSMT does not respect the pin sites but models the pins by points. This yields an intrinsic prediction error by the RSMT. (ii) There are always some detours necessary in routing a net because the enclosing rectangles of the nets overlap, i.e. the nets interfere with each other. But the detours increase the routed wirelength and are not respected in constructing the RSMT.

5) *The HPWL is an efficient estimation of the routed wirelength*
As there is always a prediction error of the routed wirelength by the RSMT length (see statement (4)) and the HPWL is approximately proportional to the RSMT length [1], the HPWL correlates to the routed wirelength as good as the RSMT length does. But the HPWL is determined much faster than the RSMT length [22]. Therefore the HPWL is an efficient estimation of the routed wirelength.

Statement (1) expresses that our placer utilizes the direct approach to optimize placement for routability.

But also the indirect approach is used to optimize routability by increasing the routing supply in congested regions. This can be explained by rewriting the total distribution (16) as new demand-and-supply system for the modules (Please note that we omitted (x, y) for brevity):

$$D = \hat{D}^{dem}_{mod} - \hat{D}^{sup}_{mod} \tag{17}$$

$$\hat{D}^{dem}_{mod} = (1 - w_{rout}) \cdot D^{dem}_{mod} \tag{18}$$

$$\hat{D}^{sup}_{mod} = \underbrace{(1 - w_{rout}) \cdot \alpha_{mod} \cdot D^{sup}_{mod}}_{a} - \underbrace{w_{rout} \cdot \left(D^{dem}_{rout} - \alpha_{rout} \cdot D^{sup}_{rout}\right)}_{b} \tag{19}$$

Eq. (19) shows that the new module supply $\hat{D}^{sup}_{mod}$ is reduced in congested regions: If the routing demand D^{dem}_{rout} is higher than the normalized routing supply $\alpha_{rout} \cdot D^{sup}_{rout}$, then b is greater than zero. Thus the new module supply $\hat{D}^{sup}_{mod}$ is lower than a, i.e. it is reduced. Since our placer adapts the demand to the supply, a reduced module supply in congested regions yields a lower module demand/density there. Therefore more free space is available in the lowest routing layer usually blocked by the modules. Finally this results in more routing supply in congested regions.

The utilization of the indirect approach to optimize routability by our placer is demonstrated in figure 4: in congested regions, i.e. in regions with high routing demand (black areas in (b) and (d)), the module density is low (white area in (a) and (c)).

5. Experimental Results

To validate and demonstrate the efficiency of RUDY and its integration in our placer, we placed and routed the IBMv2 benchmark suite [25] on an AMD Athlon Opteron 248 machine running at 2.2 GHz and using one of the two available CPU cores. The routing was done with Cadence WarpRoute 2.3.32.

Please note that all circuits of the IBMv2 benchmark suite have no fixed pad, which is critical for quadratic placement since no traditional initial placement can be computed. Hall [28] presents a solution by computing the initial placement by the eigenvectors of the system matrix $\mathbf{C_x}$. We use a different approach by introducing pseudo fixed pads, but this method is not the topic of this paper and will be submitted in future.

Table 2 summarizes the CPU times of placement and routing, routed wirelengths and via counts using our approach. This table also presents the routed wirelengths and via counts of ROOSTER, mPL and APlace as published in [21]. All results have no routing violations, except ibm01e and ibm01h placed by APlace.

Measuring quality by the routed wirelength, our approach to optimize routability in placement is 7.64%, 9.00%, and 5.39% better than ROOSTER, mPL, and APlace, respectively. In via count, our approach is only 0.72% worse than ROOSTER but 4.73% and 5.55% better than mPL and APlace.

The authors of [21] present only the CPU times of ROOSTER when placing some circuits of the IBMv2 benchmark. With the SPEC CPU2000 benchmark [29] to scale their CPU times on our machine configuration reveals that our approach is 8.4 times faster than ROOSTER.

All of our results are based on using the HPWL as an efficient estimation for routed wirelength. If the RSMT length is used as an estimation for the routed wirelength, then the CPU time of placement increases by 18% and the real routed wirelength is improved by only 0.04%. If routability is not considered, our placer finishes the IBMv2 benchmark suite in 0.5% shorter CPU time (compared to the CPU time of HPWL as an estimation for routed wirelength), but produces some unroutable circuits and the routed wirelength is increased by around 2.8%. Thus routability optimization based on RUDY, HPWL, and the implementation in our placer is very fast and efficient.

Placing the circuits ibm01e and ibm12h of the IBMv2 benchmark suite by our approach and routing each placement gives the module and wire distributions as displayed in figure 4(a) and (b), respectively (c) and (d). The wire distribution reflects the real routing demand determined after final routing. The comparison between the module distribution and the wire distribution, i.e the comparison between figure 4(a) and (b) for ibm01e and the comparison between figure 4(c) and (d) for ibm12h, shows that the module density is low (light color) in regions with high routing demand (dark color). Based on the fact that a low module density means that more free space is available in the lowest routing layer, this comparison demonstrates that our placer efficiently utilizes the indirect approach to optimize routability by increasing the routing supply in regions with high routing demand.

To put the improvement achieved by our approach into perspective, the module and wire distributions of the same circuits as in figure 4 but placed by ROOSTER is displayed in figure 5. The comparison between figure 5(a) and (b) and between figure 5(b) and (c) shows no high correlation between module density and routing demand.

6. Conclusion

This paper presented RUDY, which estimates fast and accurately the routing demand of a circuit independently of a certain bin structure or a routing model. The demand-and-supply system of a fast and robust force-directed quadratic placer was extended in this paper in a convenient manner by RUDY to reflect not only the distribution of the modules but also the distribution of the enclosing rectangles of the nets. As our placer is guided by the demand-and-supply system, our approach to optimize placements for routability can be interpreted in a way that we place simultaneously the modules and the nets on the chip. This yields the best published routed wirelengths of the IBMv2 benchmark suite.

7. Acknowledgments

The authors thank the reviewers for their useful suggestions, I. Markov and J. A. Roy for providing the UMICH packages “CongestionMaps Plotter” and “ROOSTER”, and U. Schlichtmann for his continuous support.

References

[1] Chih liang Eric Cheng. RISA: Accurate and efficient placement routability modeling. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 690–695, 1994.

[2] Maogang Wang, Xiaojian Yang, and Majid Sarrafzadeh. Congestion minimization during placement. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 19(10):1140–1148, oct 2000.

[3] Jinan Lou, Shankar Krishnamoorthy, and Henry S. Sheng. Estimating routing congestion using probablistic analysis. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 21(1):32–41, January 2002.

[4] Andrew B. Kahng and Xu Xu. Accurate pseudo-constructive wirelength and congestion estimation. In *International Workshop on System Level Interconnect Prediction*, pages 61–68, 2003.

[5] Mehdi Saedi, Morteza Saheb Zamani, and Ali Jahanian. Prediction and reduction of routing congestion. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 72–77, 2006.

[6] Jurjen Westra, Chris Bartels, and Patrick Groeneveld. Probabilistic congestion prediction. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 204–290, 2004.

[7] Ulrich Brenner and Andre Rohe. An effective congestion driven placement framework. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 6–11, 2002.

[8] Ke Zhong and Shantanu Dutt. Algorithms for simultaneous satisfaction of multiple constraints and objective optimization in a placement flow with application to congestion control. In *ACM/IEEE Design Automation Conference (DAC)*, volume 39, pages 854–859, 2002.

[9] Xiaojian Yang, Ryan Kastner, and Majid Sarrafzadeh. Congestion estimation during top-down placement. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 164–169, 2001.

Circuit	Our approach				ROOSTER		mPL		APlace	
	CPU [s] Place.	CPU [s] Rout.	rWL [m]	# Vias	rWL [m]	# Vias	rWL [m]	# Vias	rWL [m]	# Vias
ibm01e	34	266	0.679	120400	0.718	122873	0.718	123064	0.790*	158646
ibm01h	33	297	0.680	121757	0.725	124063	0.691	213162	0.732*	161717
ibm02e	67	325	1.850	254002	2.000	256155	1.821	250527	1.846	254713
ibm02h	68	427	1.988	266209	1.978	262022	1.897	260455	1.973	268259
ibm07e	176	566	3.601	474725	3.953	470104	4.129	492947	3.975	500574
ibm07h	183	665	3.630	488318	4.091	489067	4.240	516929	4.141	518089
ibm08e	223	682	4.041	569646	4.231	559010	4.372	579926	3.960	595528
ibm08h	220	710	3.961	575041	4.240	577879	4.280	599467	3.960	595528
ibm09e	216	504	2.901	488778	3.200	473605	3.319	488697	3.095	502455
ibm09h	211	524	2.916	490270	3.205	480961	3.454	502742	3.102	512764
ibm10e	307	905	5.808	770378	6.420	755673	6.553	777389	6.178	782942
ibm10h	314	932	5.783	773344	6.544	781897	6.474	799544	6.169	801605
ibm11e	301	685	4.405	635610	4.746	613437	4.917	633640	4.755	648044
ibm11h	308	676	4.401	636152	4.716	625654	4.912	660985	4.818	677455
ibm12e	341	1377	8.432	932513	9.333	930397	10.185	995921	8.599	921454
ibm12h	345	1386	8.469	950607	9.282	942551	9.724	976993	8.814	961296
		Average:	1.000	1.000	1.076	0.993	1.090	1.047	1.054	1.055
		Our Improvement[†]			**7.64%**	**-0.72%**	**9.00%**	**4.73%**	**5.39%**	**5.55%**

Table 2. Results of our approach: CPU times of placement and routing, routed wirelength (rWL) and number of vias (# Vias). Results of ROOSTER (as a feature of Capo 10), mPL and APlace in routed wirelength and number of vias. *means there are some routing violations. †A value greater than zero represents that our approach is better. All results are based on circuits of the IBMv2 benchmark suite.

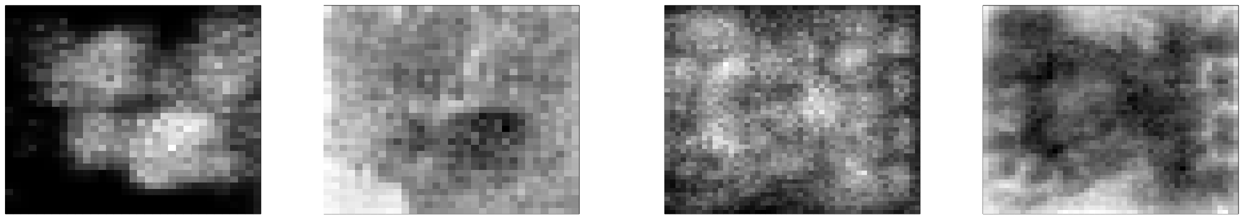

(a) ibm01e, Module Distribution (b) ibm01e, Wire Distribution (c) ibm12h, Module Distribution (d) ibm12h, Wire Distribution

Fig. 4. Our approach: module and wire distributions of the circuits ibm01e and ibm12h. White means low density, black means high density.

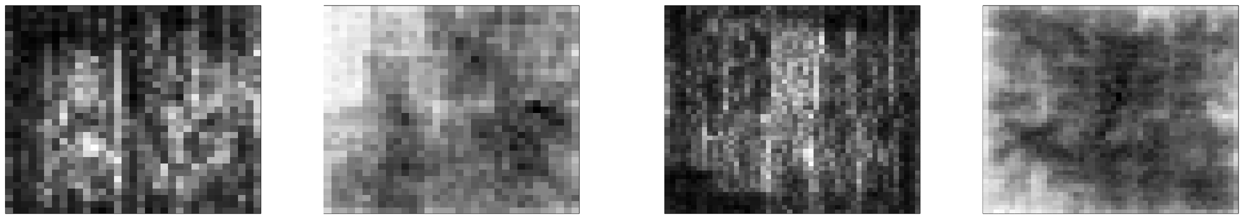

(a) ibm01e, Module Distribution (b) ibm01e, Wire Distribution (c) ibm12h, Module Distribution (d) ibm12h, Wire Distribution

Fig. 5. ROOSTER: module and wire distributions of the circuits ibm01e and ibm12h. White means low density, black means high density.

[10] Xiaojian Yang, Ryan Kastner, and Majid Sarrafzadeh. Congestion estimation during top-down placement. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 21(1):72–80, January 2002.

[11] Bo Hu and Malgorzata Marek-Sadowska. Congestion minimization during placement without estimation. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 739–745, 2002.

[12] Maogang Wang, Xiaojian Yang, Kenneth Eguro, and Majid Sarrafzadeh. Multi-center congestion estimation and minimization during placement. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 147–152, 2000.

[13] Maogang Wang and Majid Sarrafzadeh. On the behaviour of congestion minimization during placement. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 145–150, 1999.

[14] Maogang Wang and Majid Sarrafzadeh. Modeling and minimization of routing congestion. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 185–290, 2000.

[15] Zhuoyuan Li, Weimin Wu, and Xianlong Hong. Congestion driven incremental placement algorithm for standard cell layout. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 723–728, 2003.

[16] Wenting Hou, Hong Yu, Xianlong Hong, Yici Cai, Weimin Wu, Jun Gu, and William H. Kao. A new congestion-driven placement algorithm based on cell inflation. In *Asia and South Pacific Design Automation Conference*, pages 605–608, 2001.

[17] Phiroze N. Parakh, Richard B. Brown, and Karem A. Sakallah. Congestion driven quadratic placement. In *ACM/IEEE Design Automation Conference (DAC)*, pages 275–278, 1998.

[18] Xiaojian Yang, Bo-Kyung Choi, and Majid Sarrafzadeh. Routability-driven white space allocation for fixed-die standard-cell placement. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 22(4):410–419, April 2003.

[19] Chen Li, Min Xie, Cheng-Kok Koh, Jason Cong, and Patrick H. Madden. Routability-driven placement and white space allocation. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 394–401, 2004.

[20] Chin-Chih Chang, Jason Cong, Zhigang David Pan, and Xin Yuan. Physical hierarchy generation with routing congestion control. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 36–41, 2002.

[21] Jarrod A. Roy, James F. Lu, and Igor L. Markov. Seeing the forest and the trees: Steiner wirelength optimization in placement. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 78–85, 2006.

[22] Chris Chu. FLUTE: Fast lookup table based wirelength estimation technique. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 696–701, 2004.

[23] Qinke Wang Andrew B. Kahng. Implementation and extensibility of an analytic placer. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 24(05):734–747, May 2005.

[24] Ucla/umich physical design tools. `http://vlsicad.eecs.umich.edu/BK/PDtools`.

[25] Xiaojian Yang, Bo-Kyung Choi, and Majid Sarrafzadeh. Routability-driven white space allocation for fixed-die standard-cell placement. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 42–49, 2002.

[26] Raia T. Hadsell and Patrick H. Madden. Improved global routing through congestion estimation. In *ACM/IEEE Design Automation Conference (DAC)*, pages 28–31, 2003.

[27] Peter Spindler and Frank M. Johannes. Fast and robust quadratic placement based on an accurate linear net model. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2006.

[28] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, November 1970.

[29] Standard Performance Evaluation Corporation. SPEC CPU 2000. `http://www.spec.org/cpu2000`.