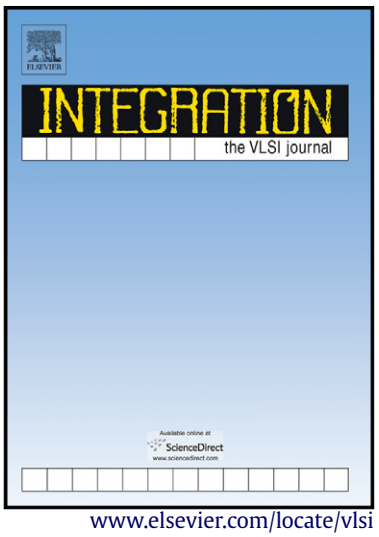


www.elsevier.com/locate/vlsi

Timing-constrained Power Minimization in VLSI Circuits by Simultaneous Multilayer Wire Spacing

Konstantin Moiseev, Shmuel Wimer, Avinoam Kolodny

PII: S0167-9260(14)00016-9
DOI: http://dx.doi.org/10.1016/j.vlsi.2014.03.002
Reference: VLSI1060

To appear in: *INTEGRATION, the VLSI journal*

Cite this article as: Konstantin Moiseev, Shmuel Wimer, Avinoam Kolodny, Timing-constrained Power Minimization in VLSI Circuits by Simultaneous Multilayer Wire Spacing, *INTEGRATION, the VLSI journal,* http://dx.doi.org/10.1016/j.vlsi.2014.03.002

Timing-constrained Power Minimization in VLSI Circuits by Simultaneous Multilayer Wire Spacing

Konstantin Moiseev[a*], Shmuel Wimer[b], Avinoam Kolodny[c]

[a]Intel Israel (74) Ltd., Technology and Manufacturing Group, Haifa 31015, Israel

[b]Bar-Ilan University, Engineering Faculty, Ramat-Gan 52900, Israel and with Technion, Electrical Engineering Faculty, Haifa 32000, Israel

[c]Technion, Electrical Engineering Faculty, Haifa 32000, Israel

[*]Corresponding author. Tel.: +972 4865 1537.

Email: kostya.moiseev@gmail.com

Email: konstantin.moiseev@intel.com

Abstract

Reduction of interconnect delay and interconnect power has become a primary design challenge in recent CMOS technology generations. Spacing between wires can be modified so that line-to-line capacitances will be optimized for minimal power under timing constraints. In this paper, we present a novel algorithm for simultaneous multilayer interconnect spacing that minimizes the total dynamic power dissipation caused by an interconnect, while maximum delay constraints are satisfied. A multi-dimensional visibility graph is used to represent the problem, and a layout partitioning technique is applied to solve the problem efficiently. The algorithm was evaluated on an industrial microprocessor designed using the 32 nanometer technology, and it achieved a 5-12% reduction in interconnect switching power.

Highlights

- Multi-layer interconnect power optimization under timing constraints is described.
- Related global optimization problem is formulated.
- An algorithm for solving optimization problem is described and implemented.
- A mathematical relation to similar optimization problems is developed.
- 5–12% dynamic interconnect power reduction on industrial cases is demonstrated.

Keywords

Interconnect sizing and spacing, power-delay optimization, constrained optimization

I. Introduction

Minimization of power dissipation has become a primary design challenge due to a combination of technology scaling, the prevalence of mobile battery-operated electronic products, and growing awareness of environmental heating. To develop power-efficient VLSI products, the power optimization is applied at all design stages, starting from the architecture through the circuit implementation and down to the layout design. Meanwhile, the circuit performance remains an important design objective, so that the power optimization must not ignore the timing requirements imposed on the circuits. Any power optimization must therefore be timing-constrained.

One of the largest contributions to power dissipation in CMOS VLSI processors is incurred by the charging and discharging of the interconnect capacitances [3]. The relative contribution of the line-to-line capacitances within the same metal layer grows with the technology progression due to the nonuniform scaling [7], as the aspect ratio between the wire thickness and the width continuously increases. Consequently, the cross-coupling capacitances between the adjacent wires that reside on the same metal layer have a major effect on both the circuit timing and the power. The high-level metal layers in the interconnect stack are the most important contributors to this capacitance.

The cross-coupling capacitances can be decreased and, therefore, the switching power can be reduced by increasing the inter-wire spaces as long as the timing constraints are not violated and the chip area is not increased, which is the goal of this paper. We claim that the inter-wire spacing has become an important resource in the physical design: the large spaces should be allocated to those wires that are more likely to switch rather than to wires that are typically inactive. Our technique is designated for use in the late pre-tapeout design stages. We therefore assume that the interconnects have been routed (manually or automatically), and their relative locations are not subject to change (i.e., the layout topology is unchanged). The wire widths are assumed to have been set to satisfy the signal delay and the other design goals such as reliability, and the shield wires have been employed to eliminate the crosstalk noise on the sensitive nodes. Hence, our only purpose is to modify the wire-to-wire capacitance densities across the whole layout so that the DFM rules and design timing constraints are not violated. Figure 1 illustrates a layout before and after optimization. The inter-wire space is reallocated according to the wire switching activity. The wires with higher switching activity are allocated the larger spaces, while the wires with lower activities are allocated the smaller spaces. Notwithstanding, the space reallocation must not violate the wire delay constraints. To preserve the layout connectivity, the orthogonal wires located on the neighboring layers are shortened or prolonged accordingly so that the vias can be landed safely. The creation of new jogs is avoided by using the technique described in Section IV. Some design rule violations may still appear in the resulting layout. The design rule violations are fixed manually at a later stage of the design.

Layout optimization by wire spacing has been discussed in the literature for yield improvement [4], cross-coupling noise reduction [5, 6], timing optimization [8, 9, 10, 11, 33], power optimization [12], and a combination of timing and power [1]. The authors of [6] employed a net-by-net heuristic for crosstalk noise reduction, which may change the layout topology and yield non-optimal results. In [12], the switching power of the wires bundled in a bus was heuristically optimized. In [5], all the wires in a layer were simultaneously spaced for delay optimization, but the power was ignored because the power was not a primary concern at that time. Net-by-net optimization is applicable for timing but less suited for power because the interconnect power is accumulated from all of the wires, while timing optimization concerns only the most critical wire delays. In [1], a combined power-delay optimization was simultaneously performed for many wires using a Weighted Power-Delay Sum (WPDS). This method still suffers from a major drawback: Although the method allows a tradeoff between the power and the delay by setting their relative weights appropriately, the method does not guarantee the satisfaction of the timing constraints. Displacement of a wire might violate the max delay constraint of its neighbor. To circumvent these cases, such wires in [1] were not allowed to move at the expense of some power reduction potential. The wire fixing is too conservative in many cases. Some relaxation could still reduce the power without violating the max delay constraints. This paper takes full

advantage of such relaxation, where the wires are allowed to be displaced as far as permitted by the timing constraints.

Another limitation in the previous work is that either they aim to optimize a single net with all of its interconnecting wires residing on the various layers, or their scope is limited to simultaneous optimization of all of the wires residing on same layer. Those solutions were iteratively employed net-by-net or layer-by-layer, which yielded sub-optimal solutions. All of these studies rely on the convexity of the delay and the power. However, working net by net does not guarantee that the global minimum will be reached, as stated in [28]. The illustrative example shown in Figure 2 demonstrates that in the presence of constraints (solid curve), the one-wire-at-a-time downhill approach does not always converge to the global minimum. The optimization path (the staircase line) can leave the region dominated by the global optimum P^* (dashed in the picture); to reach the optimum point from the feasible region boundary, locally non-optimal moves are required (the dashed arrow). In this paper, we show that the multi-layer multi-net power minimization problem under the delay constraints is convex, and we propose an optimization algorithm that considers all the nets and layers simultaneously, guaranteeing an optimal solution.

The rest of this paper is organized as follows. In the next section, we present the layout model and define the power-delay optimization problem. In section III, we solve the problem and present an implementation of the algorithm. Practical considerations of the power-delay optimization are discussed in Sections IV and V. Examples and experimental results are shown in Section VI. Section VII concludes the paper. The relationship between our method and the WPDS optimization of [1] is discussed in the Appendix.

II. INTERCONNECT MODELING AND PROBLEM DEFINITION

The following notation is used for describing the problem:

N - *Total number of routed nets*

L - *Total number of metal layers*

N_l - *Total number of wires residing at layer* l

A_l - *Width of total routing area at layer* l

σ_i - *The* i^{th} *net*

Q_i - *Total number of effective loads (pins) of the* i^{th} *net*

W_i - *Total number of wire segments belonging to the* i^{th} *net*

I_i^l - *The* i^{th} *wire residing on layer* l

$M = \sum_{i=1}^{N} Q_i$ - *Total number of effective loads (pins)*

d_{ij}^l - *Length of the common span of wires* i *and* j *residing on layer* l *where they are visible to each other.*

s_{ij}^l -*Spacing between wires* i *and* j *residing on layer* l

The high-metal layers in modern VLSI circuits (e.g., metal 5 and above) are used for long distance interconnect routing, typically spanning distances from hundreds to thousands of microns across the chip. A schematic example of this type of interconnect is shown in Figure 3, where all the wire segments are numbered.

Because the signal connect devices lie underneath the stack of metal layers, a small portion of the routing takes place on the lower metal layers (e.g., metal 1 up to metal 4) that are not considered in this work due to their minor impact. Still, the models of the effective driver and the effective receiver load consider the local routing, so that the resistance of these wires is included in the driver model, and their capacitance is included in the load model. In the following text, we assume that all layout changes are performed on the global metal layers, with no effect on the interconnects or the cells in the low metal layers, so that their contributions to the total power and delay remain unchanged.

Let nets $\sigma_1, \ldots \sigma_N$ be given. Each net is assigned an activity factor α_i, quantifying the amount of the signal switching relative to the clock signal. This factor can range from $\alpha_i = 0$ if the signal never switches (e.g., the shields or the power delivery wires) to $\alpha_i = 1$ if the switch toggles twice in every cycle (e.g., clocks). The signal activity factors used in this paper have been obtained by an industrial power simulator that calculates its average activity based on different scenarios [13, 14], so that the power calculations reflect the realistic operation of the circuit.

The multilayer structure of global interconnects can be represented as a collection of planes, each of which includes all wire segments routed on the corresponding metal layer l, $1 \le l \le L$. Routing areas A_l within each layer are bounded by a fixed grid of the power supply wires. These wires serve as "walls" of the routing area, as shown in Figure 4.

In modern VLSI technologies, the routing layers contain the wires that are either vertical or horizontal with only a few, usually very small, jogs. The influence of the jogs on the power and the delay is negligible, and we therefore ignore the jogs in the analysis.

The multilayer interconnect structures shown in Figures 3 and 4 are represented by a *multilayer visibility graph* $G(V, E)$ as follows. For each wire I_i^l we associate a vertex $v_i^l \in V$. The vertices v_0^l and $v_{N_l+1}^l$ correspond to the "wall" wires in the layer l. There are two types of edges in the graph. Two vertices v_i^l and u_j^l that correspond to wires I_i^l and I_j^l, which are visible to each other, define a **visibility** edge. Two vertices, $v_i^{l_1}$ and $u_j^{l_2}$, with $l_1 \ne l_2$ and that are physically connected to each other define a **connectivity** edge. A similar visibility graph structure is described in [34] in the context of a layout migration. An example of a multilayer visibility graph is shown in Figure 5. The relative locations of wires are maintained using the visibility graph. Because wires in both the vertical and horizontal directions can move simultaneously, the visibility relationships between the wires may change because, as the spaces between wires residing on some layer change, the wires residing in the neighboring layers are stretched or contracted. However, these changes are usually very small in comparison to the wire lengths, so corresponding changes in the cross coupling capacitances, the ground capacitances and the resistances can be neglected.

Different models exist for the cross-coupling capacitance between two adjacent wires [2, 15, 16]. The coupling capacitance per unit length $g\left(s_{ij}^l\right)$ between the adjacent wires monotonically decreases with s_{ij}^l. The nominal line-to-line capacitance associated with I_i^l and I_j^l is

$$c_{ij}^l = \kappa d_{ij}^l \cdot g\left(s_{ij}^l\right). \qquad (2)$$

The only assumption made about g is that it is a convex function, which conforms with the commonly used model $g\left(s_{ij}^l\right) = 1 \Big/ \left(s_{ij}^l\right)^{\gamma}$, where $\gamma \ge 1$. If wires I_i^l and I_j^l are not visible to each

other, then $d_{ij}^{l}=0$ and the cross capacitance is negligible. We also set $d_{ij}^{l}=0$ for the case where I_i^l and I_j^l belong to the same net.

The cross coupling capacitance between two neighbors also depends on their mutual switching activity [17, 18], known as the Miller Coupling Factor (MCF). Their simultaneous switching in opposite directions consumes four times the power consumed by a single logical transition of one of the wires. Simultaneous switching in the same direction consumes no power. The average factor per wire is therefore 1 (averaging 4/2, 1 and 0). Assuming equal probability for those switching patterns, it is legitimate to assume MCF=1 when considering the cross coupling capacitance contribution to the switching power.

Under this assumption, the power contributed by each cross capacitance is proportional to the product of its nominal value by the sum of the activity factors of neighbor wires. Therefore, the contribution of each cross capacitance to the total power can be divided between two neighboring wires, as shown below.

The dynamic power corresponding to wire I_i^l is expressed by:

$$\begin{aligned} P_i &= \alpha_i V_{dd}{}^2 f\left(C_i^a + C_i^{ll}\right) = \\ &= \alpha_i C_i^a V_{dd}{}^2 f + \alpha_i C_i^{ll} V_{dd}{}^2 f = P^{self} + P^{cross} \end{aligned}, \quad (3)$$

where α_i is the activity factor of the net to which I_i^l belongs, V_{dd} is the voltage swing, and f is the clock frequency. C_i^a is the total wire self capacitance contributed by the capacitors formed between I_i^l and the layers above and below. C_i^{ll} is the total effective cross-capacitance formed by I_i^l and its visible wires. The corresponding power portions are denoted P^{self} and P^{cross} [1]. We assume that the widths of the wires are not subject to change in the spacing optimization. Adding wire widths as optimization variables could further reduce the power at the expense of complicating the solution. In full-custom design practice, wire widths are set very early in the design flow according to the signal propagation delay specifications and are not changed in the late stages where spacing optimization is applied. We also assume that, although below and above layers are not ground planes, their influence on the cross capacitance changes is negligible on average because the wires on these layers are perpendicular to those in the layer of interest. Thus, both P^{self} and P^{cross} are used in the total power calculations, but only P^{cross} in (3) is of interest for the power minimization by the wire spacing. The same holds for delay calculations, which are shown below.

It follows from (2) that the power contributed by the cross-capacitances of I_i^l is:

$$P_i^{cross\,l} = \alpha_i k' \sum_{j=1, j\neq i}^{N_l} d_{ij}^{l} g\left(s_{ij}^{l}\right) \quad (4)$$

where the coefficient k' incorporates the supply voltage, the clock frequency and the technology-dependent constants. The total power contributed by all of the wires routed on all of the metal layers is then expressed by:

$$P^{cross} = k' \sum_{l=1}^{L} \sum_{i=1}^{N_l - 1} \sum_{j=i+1}^{N_l} d_{ij}^{l} \left(\alpha_i + \alpha_j\right) g\left(s_{ij}^{l}\right) \quad (5)$$

Cross coupling capacitance affects delay as well. A net is represented as a rooted interconnect tree, as illustrated in Figure 6. The interconnect tree comprises three types of wire segments: a wire

[1] Instead of breakdown to self and cross capacitance, it is possible to consider a single total capacitance as a function of spaces to neighbors.

connected to the near-end driver at the root, wires connected to the receiving gates at the far-end leaves, and wires corresponding to the internal nodes of the tree. Denote by W_p the number of wire segments of net σ_p. Let Q_p of those, denoted by $Ircv_k$, be connected at the far end, and let C_k^{el}, $1 \le k \le Q_p$, be their corresponding effective loads. $Idrv$ is connected to the driver, and the rest are internal wires $Iint_k$, $1 \le k \le W_p - Q_p - 1$. For the sake of the calculations, the wire segments are divided into smaller pieces with homogeneous adjacencies on their two sides. The visibility between the same two wires may define several capacitors in the case where the common span of the two wires experiences interference from small wire segments between them. We denote by $C_{i-j,k}$ the $k-^{\text{th}}$ capacitance between I_i and I_j. For example, on the left side of Figure 7, wire segment 9 is divided into four parts, forming five cross coupling capacitances: one with segment 11 denoted C_{9-11}, two with segment 8 denoted $C_{9-8,1}$ and $C_{9-8.2}$, and two $C_{9-10,1}$ and $C_{9-10,2}$ with segment 10. Each wire segment or part of a segment is modeled as a π-load. The decoupled line-to-line capacitance is counted along with the self-capacitance of the segment. The right side of Figure 7 illustrates the modeling of the driver-receiver path for a single net, comprising segments 2, 9, and 6.

We use the Elmore delay estimation to calculate net delays. Although it is not accurate, its high fidelity property has previously been shown to allow its use in optimization algorithms [30, 31]. By Elmore's model, the delay of an interconnect path is a convex function of the spaces to visible wires along its traversal from driver to receiver, given by the linear sum of the RC delays occurring along the driver to receiver path.

The Elmore delay expression depends on the various spaces to the visible wires. Let $\mathbf{s}$ denote the vector of the involved spaces. The delay from the driver to the receiver C_k^{el} is expressed by:

$$T_k = h(\mathbf{s}), \tag{6}$$

where h is a convex function in each one of the spaces. In the above discussion, we disregarded via resistances. Including via resistances in the Elmore delay formula does not change the functional form of the delay dependence on the cross-capacitance. Therefore, via resistances are neglected for the sake of expression simplicity.

The delay expression also holds for the more accurate models, such as those proposed in [15]. We experimented with those models and found that the convexity assumption does hold. Only the convexity of $h(\mathbf{s})$ is required for the proposed algorithm to work. While MCF=1 is used for power, MCF=2 is used for delay calculations, representing the worst-case coupling scenario. Modern timing analysis tools are able to calculate the individual MCF for each net. The technique presented is general and allows the definition of an individual MCF for each net segment. The uniform MCF factor of 2 is used for simplicity and also because this factor represents the worst-case coupling. Thus, the convex function in (6) already includes this factor. Finding the spaces s_{ij} that minimize the total power in (5) is subject to a number of constraints, listed below.

First, DFM rules impose limits on the allowed distance between two wires. Thus, each space s_{ij} should satisfy a minimum spacing rule associated with every layer,

$$s_{ij}^l \ge s_{\min}^l . \tag{7}$$

Second, the circuit timing requirements should not be violated. If D_j is the required signal arrival time at the receiver j, $1 \le j \le M$, then

$$T_j \le D_j , \tag{8}$$

Third, the wire position cannot exceed the location of the fixed wall boundaries. Let us denote by Ω_l the set of all paths in the visibility graph between the vertices corresponding to the wall wires at layer l consisting of visibility edges only—without the loss of generality, paths of the vertically routed layers extending between the source and the target vertices corresponding to the left and right walls, respectively. Similarly, the paths of the horizontally routed layers extend between the source and the target vertices corresponding to the bottom and top walls, respectively). Let $\omega = \left(w_{i_1}, s_{i_2}, w_{i_3}, s_{i_4}, \ldots\right)$ be a path of alternating widths and spaces corresponding to a path in G. Then:

$$\sum_{\substack{w_i \in \omega \\ s_j \in \omega}} w_i + s_j \le A_l, \ \forall \omega \in \Omega, \ \forall 1 \le l \le L, \qquad (9)$$

The set of constraints (9) is impractical because of the very large size, resolved by introducing new variables. Let us denote by x_i^l the coordinates of the centerlines of wires I_i^l. The relationship between variables x_i^l and s_{ij}^l is expressed by

$$s_{ij}^l = x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \qquad (10)$$

Taking into account that the wall wire coordinates are $x_0^l = 0$ and $x_{N_l+1}^l = A_l$ and their widths are zero, the constraints (7) can be rewritten as:

$$\begin{aligned} & x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \ge s_{\min}^l, \\ & \forall \ 1 \le l \le L, \ 0 \le i, j \le N_l + 1 \ \wedge \ d_{ij} > 0 \end{aligned} \qquad (11)$$

Equation (11) also contains boundary constraints. Indeed, summing the constraints (11) on some path $\omega \in \Omega_l$, we obtain $A_l - \sum_{w_i^l \in \omega} w_i^l \ge |\omega| \cdot s_{\min}^l$, which is always true if the problem is feasible.

Using (5), (8) and (11), the optimization problem can be formulated as follows:

Program *PODC (Power Optimization under Delay Constraints)*

$$\begin{aligned} & \underset{x_i}{\text{minimize}} \ P^{cross} \\ & \text{s.t.} \\ & T_j \le D_j, \ 1 \le j \le M \\ & x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \ge s_{\min}^l, \forall \ 1 \le l \le L, \ 0 \le i, j \le N_l + 1 \ \wedge \ d_{ij} > 0 \end{aligned}$$

Program PODC is closely related to the Weighted Power Delay Sum (WPDS) optimization problem [1, 25]:

Program *WPDS*

$$\begin{aligned} & \underset{x_i}{\text{minimize}} \left(P^{cross} + \sum_{i=1}^{M} k_i T_i \right) \\ & \text{s.t.} \\ & x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \ge s_{\min}^l, \forall 1 \le l \le L, \ 0 \le i, j \le N_l + 1 \ \wedge \ d_{ij} > 0 \end{aligned}$$

In both PODC and WPDS, the delays T_i are calculated according to the Elmore delay model presented earlier. The coefficients k_i are non-negative numbers representing delay criticalities. k_i, $1 \le i \le M$ are

set in advance. The WPDS optimizes the power contributed by cross-capacitances, weighted by the net delays. While the PODC can be used for design tuning when the exact delay constraints for each net are known, WPDS can be useful in the early design stage when specific delay requirements are not yet available.

The relationship between the PODC and the WPDS is discussed in the Appendix. The following sections are dedicated to solving the PODC problem.

III. ALGORITHM FOR SOLUTION OF THE OPTIMAL SPACING PROBLEM

Theorem. Program PODC is convex.

Proof. Both the objective function and the delay inequality constraints are convex in s_{ij} by their definitions. The transformation in (10) is linear; therefore, the transformation preserves convexity [20]. The location constraints are linear in x and are thus convex. Consequently, the optimization problem is convex.

The convexity of the PODC allows us to apply Newton's method directly, provided that a step of the cost reduction does not fall out of the feasibility region. To ensure that a step of the cost reduction does not fall out of the feasibility region, we use the interior-point method [20, 27]. For our optimization problem, we introduce an additional variable $\eta > 0$ and form the following *log-barrier* function:

$$LB(\mathbf{x};\eta) = -\eta\left(\sum_{\substack{1\le l\le L\\ 0\le i,j\le N_l+1,\ i\ne j}} \log\left[x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 - s_{\min}^l\right]\right) - \eta\left(\sum_{1\le j\le M}\log\left(D_j - T_j\right)\right) \qquad (12)$$

To apply the log-barrier approach, assume that the initial design is at a feasible point. Such a point always exists because usually the design process is iterative so that at the end of each iteration, all timing constraints are satisfied.

The domain of function (12) is the set of points that satisfy the inequality constraints of the PODC strictly. The logarithmic barrier grows without bound if any of the inequality constraints approaches equality. The new objective function is obtained by

$$P^{cross}{}'(\mathbf{x};\eta) = P^{cross}(\mathbf{x}) + LB(\mathbf{x};\eta) \qquad (13)$$

and the new optimization problem becomes the following unconstrained program:

Program *PODC - LB*

$$\min\left\{P^{cross} - \eta\left(\sum_{\substack{1\le l\le L\\ 0\le i,j\le N_l+1,i\ne j}} \log\left[x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 - s_{\min}^l\right] + \sum_{1\le j\le M}\log\left(D_j - T_j\right)\right)\right\}$$

The PODC-LB program is only an approximation of the PODC program, and its quality improves as the parameter η decreases [20]. Denote by $\mathbf{x}^*(\eta)$ the solution of PODC –LB for a given η. One can show that $\mathbf{x}^*(\eta)$ converges to solution $\mathbf{x}^*$ of the PODC problem as $\eta \to 0$ [20]. The solution of the PODC is obtained by solving a sequence of PODC-LB problems with decreasing values of η (in every

iteration η is multiplied by some $0 < \tau < 1$). Each iteration starts at the solution of the problem for the previous value of η. Figure 8 shows the pseudocode of the procedure.

PODC-LB is an unconstrained convex optimization problem, solved by Newton's method as follows. Given the initial feasible point $\mathbf{x}$, a direction of a step is calculated by $\Delta\mathbf{x}_N = -\nabla^2 P^{cross}{}'(\mathbf{x})^{-1} \cdot \nabla P^{cross}{}'(\mathbf{x})$. The location is obtained by $\mathbf{x} = \mathbf{x} + t \cdot \Delta\mathbf{x}_N$, where t is a step size calculated for every iteration by line search along direction $\Delta\mathbf{x}$. Although Newton's method is known for its fast convergence, the calculation and storage of the Hessian $\nabla^2 P^{cross}{}'(\mathbf{x})$ and its inverse is not always possible for real cases involving thousands of variables. Even if the Hessian of the original functions $\nabla^2 P^{cross}(\mathbf{x})$ is sparse, the log-barrier operation usually causes the Hessian to be dense, which makes the calculation of its inverse impossible. Therefore, we use the L-BFGS quasi-Newton method [21] that has, on the one hand, a super-linear rate of convergence and on the other hand, does not require the calculation of the full Hessian inverse. According to this method, the inverse of the original Hessian matrix is replaced by the inverse of the Hessian approximation matrix, which is recalculated in every iteration based on its value from the previous iteration. Denoting the gradient change $\nabla P^{cross}{}'(\mathbf{x}_{k+1}) - \nabla P^{cross}{}'(\mathbf{x}_k)$ by $\Delta\mathbf{g}$ and the variable vector change $\mathbf{x}_{k+1} - \mathbf{x}_k$ by $\Delta\mathbf{x}$, the inverse of the Hessian approximation matrix in the $k+1-$iteration is calculated by

$$\mathbf{H}_{k+1} = \left(\mathbf{I} - \frac{\Delta\mathbf{x} \cdot \Delta\mathbf{g}^T}{\Delta\mathbf{g}^T \Delta\mathbf{x}}\right) \mathbf{H}_k \left(\mathbf{I} - \frac{\Delta\mathbf{g} \cdot \Delta\mathbf{x}^T}{\Delta\mathbf{g}^T \Delta\mathbf{x}}\right) + \frac{\Delta\mathbf{x} \cdot \Delta\mathbf{x}^T}{\Delta\mathbf{g}^T \Delta\mathbf{x}} \quad (14)$$

Notice that the calculation of $\mathbf{H}_{k+1}$ involves only scalar products of vectors or matrices by vector multiplications. The value of $\mathbf{H}_0$ is chosen to be as close as possible to the original Hessian inverse. The choice of

$$\mathbf{H}_0 = \frac{\Delta\mathbf{g}^T \Delta\mathbf{x}}{\Delta\mathbf{x}^T \Delta\mathbf{x}} \mathbf{I} \quad (15)$$

is reported to be the most successful in practice [22] and is therefore used in our implementation.

The storage required for $\mathbf{H}_k$ may still be expensive for real design cases. Instead of storing the full matrix $\mathbf{H}_k$, we save only a few pairs of $\{\Delta\mathbf{x}; \Delta\mathbf{g}\}$ from the most recent iterations. These pairs are used to construct the inverse Hessian approximation. The curvature information from earlier iterations that is less relevant to the Hessian behavior in the current iteration is discarded. The optimization procedure based on this method is processed as follows. At each iteration, the initial matrix $\mathbf{H}_k^0$ is first calculated by (15) based on the most recent values of $\Delta\mathbf{x}$ and $\Delta\mathbf{g}$. Then, the product of the inverse Hessian approximation by the gradient vector $\mathbf{H}_k \nabla P^{cross}{}'(\mathbf{x}_k)$ is calculated from $\mathbf{H}_k^0$ by a recursive procedure using pairs of $\{\Delta\mathbf{x}; \Delta\mathbf{g}\}$ stored for the last m iterations. Now, the new location is calculated by $\mathbf{x}_{k+1} = \mathbf{x}_k - t \cdot \mathbf{H}_k \nabla P^{cross}{}'(\mathbf{x}_k)$. Finally, new values of $\Delta\mathbf{x}_{k+1}$ and $\Delta\mathbf{g}_{k+1}$ are calculated and replace the least recent pair $\{\Delta\mathbf{x}_{k-m+1}, \Delta\mathbf{g}_{k-m+1}\}$. The algorithm for solving the PODC-LB is shown in Figure 9.

To evaluate the memory and run-time complexity, let us denote by $N_{l,\max}$ the maximum number of wire segments routed at one routing layer. Because each layer of the visibility graph is a planar graph, the number of location constraints (equal to number of spaces) N_s can be bounded by $3N_{l,\max} - 6$, according to the Euler-Poincaré characteristic. Thus, the total number of location constraints is

bounded by $LN_{l,\max} = L\left(3N_{l,\max} - 6\right) = O(LN_{l,\max})$. The total number of delay constraints is bounded by the number of output pins, which is equal to the number of wire segments in the worst case and, therefore, also $O\left(LN_{l,\max}\right)$. The L-BFGS method requires storing only m pairs of $\{\Delta\mathbf{x};\Delta\mathbf{g}\}$, as well as the visibility graph and coefficients for the objective function and the constraints calculation, which altogether sums to $O\left(mLN_{l,\max}\right)$. The run-time complexity depends on the number of internal and external iterations and on the complexity of a single L-BGFS iteration. The latter is dominated by step 5 in Figure 9. It is shown in [22] that step 5 can be performed with $4mLN_{l,\max} + LN_{l,\max} = O\left(mLN_{l,\max}\right)$ multiplications. Assume that the number of iterations of the L-BFGS algorithm (i.e., internal iterations) is N_{int}. There is no closed form expression for N_{int}. However, L-BFGS has a super-linear rate of convergence, meaning that if N_{int} is the number of iterations and ε is the required accuracy, then $N_{\text{int}}^{N_{\text{int}}} = O\left(1/\varepsilon\right)$. Thus, the L-BFGS method requires a much smaller number of iterations than the gradient descent method but is slower than the Newton method. The desired accuracy of the log-barrier method is achieved after $\left\lceil \dfrac{\log\left(LN_{l,\max}/\varepsilon\eta_{initial}\right)}{\log\left(1/\tau\right)} \right\rceil$ iterations [20]. Thus, the run-time complexity of the algorithm is $O\left(mLN_{l,\max}N_{\text{int}} \left\lceil \dfrac{\log\left(LN_l/\varepsilon\eta_{initial}\right)}{\log\left(1/\tau\right)} \right\rceil \right)$, meaning that the algorithm storage is linear in the total number of wire segments, and the run-time has an $O(n\log n)$ order of growth in the total number of wire segments. The latter, however, is greatly affected by coefficient values, such as the required accuracy ε, the initial value of the log-barrier term multiplier $\eta_{initial}$, its update τ and the number of vector pairs stored by L-BFGS algorithm m.

IV. PRACTICAL CONSIDERATIONS

In real designs, there are always special nets (such as clock network nets) that are not likely to be moved. Other wires may be required to stay "frozen" for a variety of reasons (noise, delay, slope, etc.). Others may be required to keep a predefined distance from their neighbors. The formulation of the PODC as a convex optimization problem with constraints is very convenient for such practical cases. All such cases can be handled by defining additional constraints on the wires. For example, if wire I_i must have a fixed location X_i, then this limitation can be handled by defining two additional constraints: $x_i - X_i \le 0$ and $-x_i + X_i \le 0$, both of which are convex and can be incorporated in the log-barrier function. Another example is the avoidance of jogs, i.e., when two wires should be kept with a constant distance between them (in particular, zero). Jogs complicate the layout and introduce extra delay and extra power that should be taken into account. Consider the layout in Figure 12(a). The wire segments 1 and 3, as well as segments 4 and 5, represent pairs of segments of the same physical wires. Because each wire segment is treated independently, the optimization can end with the segments shifted relative to each other, which will result in adding jogs and the complication of the layout. To avoid the jogs, such pairs of wires might be required to be treated as a single wire by the algorithm, achieved by adding four linear constraints: $x_4 - x_5 \le 0$, $x_5 - x_4 \le 0$, $x_1 - x_3 \le 0$, $x_3 - x_1 \le 0$. In general, any condition that is convex in the optimization variables (i.e., the wire coordinates) can easily be handled by the algorithm.

V. LAYOUT SEPARATION

The optimization method described in the previous sections can be applied to a clip of the layout bounded at all metal layers by fixed-position wires that are not allowed to move ("walls"), as shown in Figure 4. The full layout of the VLSI circuit can consist of several such clips. The power grids or other wires fixed in their place can serve as such wall wires. Each one of the clips can be optimized independently, thus decreasing the number of optimization variables and constraints that must be handled simultaneously. In the following text, we describe how such natural separation is found and used in the optimization process.

We call two nets **visible** if they have visible wires on some of the routing layers. We build a **net visibility graph** by assigning a vertex to each net and assigning an edge between each pair of nets so that the nets are visible to each other. According to this definition, the layout of Figure 3 is represented by a fully connected graph with three vertices and three edges because there are visible wire segments between any two nets.

Denote by **active vertex** the vertex representing a net with at least one movable wire. An **inactive vertex** is a vertex representing a net where all of its wires are fixed. Inactive vertices may represent power grid nets, shield nets or nets that were selected by design engineers to remain in fixed positions. Inactive vertices can form separation groups with respect to groups of active nets. For example, in Figure 10 the inactive vertices (shown by the dashed boundary) separate the whole graph into three groups of active vertices (shown by a solid boundary). Each one of the groups can be optimized independently and does not affect the optimization accuracy of the other groups. The partitioning into groups can easily be achieved by a Union-Find algorithm [23]. Assume that there are N active vertices in the graph. Then, Algorithm 3 (Figure 11) finds independent groups as follows. First, the individual group G_i is assigned for each active vertex $1 \le i \le N$. Then, vertices corresponding to the visible nets are merged into single group. At the end of the algorithm, the remaining groups G_i will hold separated groups of vertices.

The layout separation can significantly improve the total algorithm runtime by optimizing the separated parts in parallel. The natural separation formed by the power grid lines and other obstacles might not be uniform to allow reasonable runtime gain; therefore, artificial separation might be needed, where a minimal separating set of active nets is found and used for the separation of the rest of the active nodes. An efficient algorithm for such a vertex separation is described in [24]. Our experiments show that natural separation usually results in one very large group, including approximately 90% of the segments and several small groups containing the other 10% of the segments. Such separation cannot significantly improve run-time, so artificial separation was needed. Applying the algorithm from [24], we succeeded in finding better partitions that resulted in three groups of ~30%, ~30% and 40% of the segments. Thus, theoretically, the performance could be improved ~3X.

VI. EXAMPLES AND EXPERIMENTAL RESULTS

Algorithms 1, 2 and 3 were implemented in C++ and tested on a Pentium M 1.7 GHz processor system with 768 MB of memory. We first demonstrate the operation on the small example layout depicted in Figure 12. The layout consists of two nets including 9 wire segments (all segments are numbered) as shown in Figure 12(a). The dotted net has a driver at one end of wire 1 and receivers tied at the ends of wires 2, 4 and 6; the plaid-patterned net has a driver at the end of wire 9 and a receiver tied at the end of wire 7. The drivers are shown schematically, and in reality the drivers may be located far from the end points of the global interconnects. The corresponding layouts of the individual layers are shown in Figure 12 (b) and (c), and the multi-layer visibility graph is shown in Figure 12(d), where dotted edges designate connectivity relationships, and visibility relationships are shown by solid edges. The activity factors are 0.1 for the net with segments 1, 2, 3, 4, 5, 6 and 1 for the net with segments 7, 8 and 9. We performed two tests with this layout. These tests exemplify the difference between the optimization with and without delay constrains and show how delay awareness affects the optimization results. First, the required arrival times at receivers 2, 4, 6 and 7 were relaxed so that the optimization

was guided only by the layout topology (the mutual location constraints). In the second test, the required time of receiver 6 was tightened. This tightening caused the corresponding delay constraint to reach its bound and, as a result, prevented further movement of some wires. The optimization results for both cases are presented in Table I, and the resulting layouts are shown in Figure 13. Power, delay and coordinates are shown in relative units. In both cases, the optimization causes a significant reduction in the interconnect power. In the second case, the optimization impact is smaller than in the first case, and the slack at receiver 6 reaches zero.

Power reduction was applied to industrial test cases using clips of the real layout from the state-of-the-art 32 nm processor design. The original layout was generated by Synopsys ICC, which is the industry standard signoff P&R tool [29]. The layout completed the entire design flow and was in pre-tapeout readiness when our algorithm was applied. The layout of the metal layers (5, 6, 7, 8) was processed by the algorithm, while wire segments on the lower metal layers were modeled by modifying the corresponding effective drivers and receivers. Two reasons for this choice of the layers for processing exist. First, in the 32 nm process technology, the wires on the higher layers are allowed to be spaced almost freely (with only min and max bounds), while the spacing of wires on the lower layers is limited strictly to a predefined set of values (e.g., X, 2X and 3X, where X is the minimum spacing rule). Second, according to the design methodology of a given industrial design, the lower metal layers are enclosed in functional blocks only and are not available in the late project stages. In the implementation, we used the capacitance models presented in [2], which are consistent with our assumptions regarding cross-coupling capacitance. For the delay estimation, we used the Elmore delay formulation with the $\pi-$models for the individual net segments. Although the Elmore delay is not very accurate, it is computationally efficient, and its high fidelity property [30] allows its use as a delay metric for the optimization algorithm. To cope with the inaccuracy of the Elmore delay, the Elmore model was also used for constraint generation. In this way, both measured and required delays were calculated consistently with each other, and as a result, the Elmore delay inaccuracy was not a concern.

The results for several layout clips are presented in Table II. The numbers representing the power were calculated using an in-house power estimation tool and are given in relative units; the real numbers cannot be revealed because of their sensitivity. The cross-coupling interconnect power is reduced by 8% on average, varying among the test cases from 5% to 12.6%. Obviously, these values and their variances reflect the density and quality of the initial design in the different layout clips, and they demonstrate the practical potential benefit of the power-aware layout generation. To validate the satisfaction of the timing constraints, an in-house timing tool was used. The graphs of the slack distribution before and after optimization for one of blocks are shown in Figure 14. There are delay violations in approximately 5% of the nets after the optimization. The delay violations can be explained by the inaccuracy of the Elmore delay model. These violations can be fixed in a post-design stage by applying other optimization methods such as gate sizing [32] or by manual work.

Table III represents a comparison of the method presented with the technique described in [1], i.e., layer-by-layer optimization. For comparison, we modified the algorithm so that, each time, only the wires of a single layer are allowed to move. After optimizing the single layer, the delays were measured, and the delay constraints for the following layer were modified correspondingly. As the comparison shows, the simultaneous optimization of all layers resulted in 10%-40% better power reduction results, which can be explained by the more successful exploitation of the available delay slack. The layer-by-layer optimization is much faster.

VII. SUMMARY

Inter-wire spacing is a physical design resource that must be allocated judiciously in modern technologies because spacing determines cross-capacitances between nets, and these capacitances dominate interconnect power and delay. Previous power / delay or noise optimization techniques that rely on wire spacing work iteratively, either layer-by-layer or net-by-net. Such methods cannot fully exploit the whole optimization space and reach the global minimum because they do not take into account all imposed constraints and the interdependencies among them.

In this paper, we demonstrated an efficient method for power reduction by the simultaneous spacing of wires residing on different routing layers so that the net delay constraints were not violated. Our

method outperformed the existing techniques in the sense that it could reach the global minimum power and satisfy various (delay and layout) constraints. This result is achieved by simultaneously considering all of the nets being optimized and all of their wire segments. To reflect all relations between the wire segments, layout constraints and delay constraints, we use a novel data abstraction called the multi-layer visibility graph. An interior-point method (with the L-BFGS algorithm as an inner iteration) was used to solve the optimization problem. To cope with the scale of the problem, we applied layout partitioning based on a union-find algorithm. We demonstrated the effectiveness of the algorithm on real industrial cases and achieved a 5-12% dynamic interconnect power reduction relative to the initial layout that was generated by commercial tools by post-processing the layout and reallocating the inter-wire spaces without increasing the total area.

The proposed method treats wire spaces as continuous variables. In the up-to-date technology processes (28 nm and below), the design rules dictate the discrete locations and widths of wires so that the inter-wire spaces are also discrete (mostly at the lower layers). The application of additional methods is required to obtain discrete solutions from the continuous, such as the ILP (Integer Linear Programming) or the dynamic programming. Both approaches are beyond the scope of this paper and a matter for further research.

ACKNOWLEDGMENTS

The authors are thankful for the useful reviewers' comments, which significantly helped to improve the manuscript.

APPENDIX. DUAL PROBLEM AND RELATION TO WEIGHTED POWER-DELAY SUM (WPDS) OPTIMIZATION PROBLEM

Here, we show the relationship between the PODC problem and the Weighted Power-Delay Sum (WPDS) optimization problem described in [25], where the delay weighting was used for simultaneous gate and wire sizing for power.

The WPDS problem was discussed thoroughly in [1]. As mentioned in section II, WPDS optimizes the power contributed by the cross-capacitances, weighted by the net delays. In WPDS, the question of how to set delay criticalities k_i optimally remained open in both [1] and [25]. The theorem below provides an answer to that question by showing the relationship between WPDS and PODC solved in Section 3.

Theorem. *WPDS is the relaxation of PODC. The optimized delay criticality weights k_i in WPDS are equal to optimal values of the Lagrangian dual variables in the corresponding PODC.*

Proof. We prove the theorem by relaxing PODC and solving the dual of the relaxed problem, showing that this process obtains WPDS. We then compare WPDS to the solution of the dual of the original PODC problem.

First, relax the original problem PODC. The simplest relaxation of the program PODC would be

Program

$$\min P^{cross}$$

s.t.

$$T_i \le \delta_i, 1 \le i \le M$$

$$x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \ge s_{\min}^l, \ \forall 1 \le l \le L, 0 \le i, j \le N_l + 1 \wedge d_{ij} > 0$$

where δ_i are optimization variables (the delay constraints can as well be written as $T_i \le D_i + \delta_i$, which is equivalent). This formulation is equivalent to optimization with no delay constrains at all. To reflect delay constrains in optimization, δ_i can be incorporated into the objective function as follows:

Program *PODC - R*

$$\min\left(\alpha P^{cross} + \sum_{i=1}^{M} \beta_i \cdot \delta_i\right)$$

s.t.

$$T_i \le \delta_i, 1 \le i \le M$$

$$x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \ge s_{\min}^l, \ \forall 1 \le l \le L, 0 \le i, j \le N_l + 1 \wedge d_{ij} > 0$$

The optimization variables of the PODC-R are x_i and δ_i. The delay awareness is reflected by including δ_i in the objective function. The meaning of PODC-R is the optimization of the power under delay constraints, without explicitly specifying the delay requirement for each receiver. The delay criticality is defined by the relationship between the weights α and β_i. PODC-R is always feasible, while PODC might be infeasible, following from the PODC-R convexity and from the satisfaction of Slater's condition [20] with respect to the delay constraints that non-negative numbers $\lambda_i, 1 \le i \le M$ (Lagrange multipliers) exist so that the solution of the program PODC-R is equivalent to the solution of the following dual program PODC-RD:

Program *PODC - RD*

$$\min\left(\alpha P^{cross} + \sum_{i=1}^{M} \beta_i \cdot \delta_i + \sum_{i=1}^{M} \lambda_i \left(T_i - \delta_i\right)\right)$$

s.t.

$$x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \ge s_{\min}^l, \ 1 \le l \le L, 0 \le i, j \le N_l + 1 \wedge d_{ij} > 0$$

Solving KKT conditions [20] for PODC-RD with respect to δ_i obtains:

$$\frac{\partial}{\partial \boldsymbol{\delta}}\left(\alpha P^{cross} + \sum_{i=1}^{M} \beta_i \cdot \delta_i + \sum_{i=1}^{M} \lambda_i \left(T_i - \delta_i\right)\right) = \qquad (16)$$

$$= \beta_i - \lambda_i = 0 \Rightarrow \lambda_i = \beta_i$$

Substituting (16) into the objective function of PODC-RD and setting $\beta_i' = \beta_i / \alpha$ transforms PODC-RD into:

Program $PODC-RD'$

$$\min\left(P^{cross}+\sum_{i=1}^{M}\beta_i' T_i\right)$$

s.t.

$$x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \geq s_{\min}^l,\ \forall\, 1\leq l\leq L,\ 0\leq i,j\leq N_l+1 \wedge d_{ij}>0$$

PODC-RD' is clearly equivalent to the WPDS problem. Thus, we have shown that WPDS is the relaxation of PODC. Now, solving KKT conditions for PODC-RD' with respect to x_i yields:

$$\frac{\partial}{\partial \mathbf{x}}\left(P^{cross}+\sum_{i=1}^{M}\beta_i' T_i\right)=\nabla P^{cross}+\sum_{i=1}^{M}\beta_i'\nabla T_i=0 \quad (17)$$

Assume that λ_i^* are values of dual variables for delay constraints in the optimal point of the PODC. It is then equivalent to:

Program $D\text{-}PODC$

$$\min\left(P^{cross}+\sum_{i=1}^{M}\lambda_i^*\cdot\left(T_i-D_i\right)\right)$$

s.t.

$$x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \geq s_{\min}^l,\ \forall\, 1\leq l\leq L,\ 0\leq i,j\leq N_l+1 \wedge d_{ij}>0$$

Solving KKT conditions for D-PODC with respect to x_i results in:

$$\frac{\partial}{\partial \mathbf{x}}\left(P^{cross}+\sum_{i=1}^{M}\lambda_i^*\cdot\left(T_i-D_i\right)\right)=\nabla P^{cross}+\sum_{i=1}^{M}\lambda_i^*\nabla T_i=0 \quad (18)$$

Comparing (17) and (18), to have the same solution, the criticality weights β_i must be equal to the optimal Lagrange multipliers, λ_i^*.

The above theorem can be used to set the delay criticality weights, k_i. We use the PODC-LB interior point approximation of PODC and the fact that WPDS is the relaxation of PODC.

The optimality condition for PODC-LB (the approximation of PODC) is $\nabla P^{cross}{}'\left(\mathbf{x}^*\right)=0$, i.e.,

$$\nabla P^{cross}\left(\mathbf{x}^*\right)-\eta\sum_i\sum_j\frac{e_i}{x^{*l}_j - x^{*l}_i - \left(w_i^l+w_j^l\right)/2 - s_{\min}^l}- -\eta\sum_{i=1}^{M}\frac{\nabla T_i\left(\mathbf{x}^*\right)}{T_i\left(\mathbf{x}^*\right)-D_i}=0 \quad (19)$$

Summation is performed only for $d_{ij}>0$, and e_i is a $i-^{\text{th}}$ standard basis vector. In this sum, each term appears twice – one time for the right side and one time for the left side of each of the inter-wire spaces. The effect of the delay constraints is shown in the third term of (19). Denoting $\lambda^*{}_i(\eta)=-\frac{\eta}{T_i\left(x^*\right)-D_i}>0$ it turns into $\sum_{i=1}^{M}\lambda^*{}_i(\eta)\nabla T_i\left(x^*\right)$. Comparing to (18), we conclude that the Lagrange dual variables at the optimum (serving also as the optimal delay weights in WPDS) can be

approximated by $\lambda^*_i(\eta)$, which is inversely proportional to the wire delay slacks. In critical wires where $T_i(x^*) \approx D_i$ (small slack), λ^*_i is large indeed, while for the less critical wires where $T_i(x^*) < D_i$ (large slack), λ^*_i is smaller indeed. In general, recalling that $\eta = \lambda^*_i(D_i - T_i(x^*))$, let T_i^0 be the initial receiver delays in WPDS. Because WPDS is applied at the early design stages, specific required times D_i are yet unknown. Let D be a global required time (usually a fraction of the clock period) that all wire delays are tuned to. Then, the delay weights for WPDS are set to:

$$k_i = \frac{\eta}{D - T_i^0}, \qquad (20)$$

where η is proportionality coefficient. Thus, the WPDS problem is modified to:

Program $WPDS-1$

$$\text{minimize}\left(P^{cross} + \eta\sum_{i=1}^{M}\frac{T_i}{D-T_i^0}\right)$$

s.t.

$$x_j^l - x_i^l - \left(w_i^l + w_j^l\right)/2 \geq s_{\min}^l,\ \forall\ 1 \leq l \leq L,\ 0 \leq i, j \leq N_l + 1\ \wedge\ d_{ij} > 0$$

In the early design stages, the WPDS can be used with the delay criticalities set proportionally to the initial wire delay slacks. This process will not guarantee the satisfaction of the timing constraints after the optimization but will push the optimization in the right direction. Then, in the later design states, the transition to the PODC can be performed. Setting WPDS coefficients as in (20) (or a similar expression providing the coefficients inversely proportional to net delay criticalities) through the design lifetime will guarantee that such a transition will not result in large changes in the design and will not harm the design stability and convergence.

REFERENCES

[1] K. Moiseev, A. Kolodny and S. Wimer, "Power-delay Optimization in VLSI Microprocessors by Wire Spacing", *TODAES*, vol. 14, issue 4, 2009

[2] F. Stellari and A.L. Lacaita, "New Formulas of Interconnect Capacitances Based on Results of Conformal Mapping Method", *IEEE Transactions on Electron Devices*, vol.47, no,1, January 2000

[3] N. Magen, A. Kolodny, U. Weiser and N. Shamir, "Interconnect power dissipation in a microprocessor", *proceedings of 2004 international workshop on System Level Interconnect Prediction*, pp. 7-13, 2004

[4] V. K. R. Chiluvuri and I. Koren, "Layout-synthesis techniques for yield enhancement", *IEEE Transactions On Semiconductor Manufactoring*, Vol. 8, Issue 2, pp. 178-187, 1995

[5] K. Chanundhary, A. Onozawa and E. Kuh, "A spacing algorithm for performance enhancement and cross-talk reduction", *Proceedings of IEEE / ACM International Conference on CAD*, pp. 697-702, 1993

[6] P. Saxena and C. L. Liu, "An algorithm for crosstalk-driven wire perturbation", *IEEE Transactions on CAD of Integrated Circuits and Systems*, Vol. 19, No. 6, pp. 691-702, 2000

[7] International technology roadmap for semiconductors, 2009

[8] J. Cong, L. He, C. K. Koh and Z. Pan, "Interconnect Sizing and Spacing with Consideration of Coupling Capacitance", *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1164-1169, 2001

[9] J. –A. He and H. Kobayashi, "Simultaneous wire sizing and wire spacing in post-layout performance optimization", *Proceedings of ASP-DAC*, pp. 378-378, 1998

[10] S. Wimer, S. Michaely, K. Moiseev and A. Kolodny, "Optimal Bus Sizing in Migration of Processor Design", *IEEE Transactions on Circuits and Systems*, vol. 53, no.5, pp. 1089-1100, 2006

[11] N. Hanchate and N. Ranganathan, "A linear time algorithm for wire sizing with simultaneous optimization of interconnect delay and crosstalk noise", *Proceedings of the 19th International Conference on VLSI Design*, pp. 283-290, 2006

[12] E. Macii, M. Poncino and S. Salerno, "Combining Wire Swapping and Spacing for Low-Power Deep-Submicron Buses", *Proceedings of the 13th ACM Great Lakes Symposium on VLSI*, pp. 198-202, 2003

[13] H. Bakoglu, Circuits, Interconnects and Packaging for VLSI. *Addison-Wesley*, 1990.

[14] D. Genossar and N, Shamir, "Intel ® Pentium ® M Processor Power Estimation, Budgeting, Optimization and Validation", *Intel Technology Journal*, vol. 7, pp. 43-50, 2003

[15] S.-C. Wong, G.-Y. Lee and D. – J. Ma, "Modeling of Interconnect Capacitance, Delay and Crosstalk in VLSI", *IEEE Transactions on Semiconductor Manufactoring*, vol. 13, no. 1, 2000

[16] C. P. Yuan and T. N. Trick, "A Simple Formula for the Estimation of the Capacitance of Two-Dimensional Interconnects in VLSI Circuits", *IEEE Electronic Device Letters*, Vol. 3, No. 12, pp. 391-393, 1982

[17] A. Kahng, S. Muddu and E. Sarto, "On Switch Factor Based Analysis of Coupled RC Interconnects", *Proc. Of IEEE Design Automation Conference*, pp. 79-84, 2000
[18] P. Gupta, A. Kahng and S. Muddu, "Quantifying Error in Dynamic Power Estimation of CMOS Circuits", *Analog Integrated Circuits and Signals Porcessing*, vol. 42, pp. 253-264, 2005
[19] C.- K. Cheng, J. lillis, S. Lin and N. Chang, Interconnect Analysis and Synthesis. *Wiley-Interscience*, 1999
[20] S. Boyd and L. Vandenberghe, Convex Optimization, *Cambridge University Press*, 2004.
[21] L. Luksan and J. Vlcek, "Efficient methods for large-scale unconstrained optimization", *Nonconvex Optimization and Its Applications*, vol. 83, pp. 185-210, 2006
[22] J. Nocedal and S. Wright, Numerical Optimization, *Springer*, 2006
[23] T. Cormen, C. Leiserson, R. Rivest and C. Stein, Introduction to Algorithms, *The MIT Press*, 2005.
[24] J. Liu, "A Graph Partitioning Algorithm by Node Separators", *ACM Transactions on Mathematical Software*, vol. 15, no. 3, pp. 198-219, 1989
[25] J. Cong and C. Koh, "Simultaneous Driver and Wire Sizing for Performance and Power Optimization", *IEEE Transactions on VLSI*, vol. 2, no.4, 1994
[26] N. Gould, D. Orban and P. Toint, "Numerical methods for Large-Scale Nonlinear Optimization", *Acta Numerica*, 14, pp. 299-361, 2005
[27] S. Sapatnekar, V. Rao, P. Vaidya and S.-M. Kang, "An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization", *IEEE Transactions on CAD of VLSI*, vol. 12, no. 11, 1993
[28] R. Kay and L. Pillegi, "EWA: Efficient Wiring-Sizing Algorithm for Signal Nets and Clock Nets", *IEEE Transactions on CAD of VLSI*, vol. 17, no. 1, 1998
[29] IC Compiler – the next generation physical design system, Synopsys. Available online: http://www.synopsys.com/Tools/Implementation/PhysicalImplementation/Documents/iccompiler_ds.pdf
[30] K.D. Boese, A.B. Kahng, B.A. McCoy and G. Robins, "Fidelity and Near Optimality of Elmore-Based Routing Constructions", *proceedings of IEEE International Comference on Computer Design*, pp. 81-84, 1993
[31] A.I. Abou-Seido, B. Nowak, C. Chu, "Fitted Elmore delay: a simple and accurate interconnect delay model", *IEEE Transactions on VLSI*, Vol. 12, No. 7, pp. 691-696, 2004
[32] C.-P. Chen, C.C.N Chu and D.F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation", *IEEE Transactions on CAD*, Issue 7, pp. 691-696, 1999
[33] K. Moiseev, S. Wimer and A. Kolodny, "On Optimal Ordering of Signals in Parallel Wire Bundles", *Integration – the VLSI Journal,* vol. 41(2), pp. 253-268, 2008
[34] E. Shapir, R.Y, Pinter and S. Wimer, "Cell-based Interconnect Migration by Hierarchical Optimization", *Integration – the VLSI Journal*, vol. 47, pp.161-174, 2014

Figures

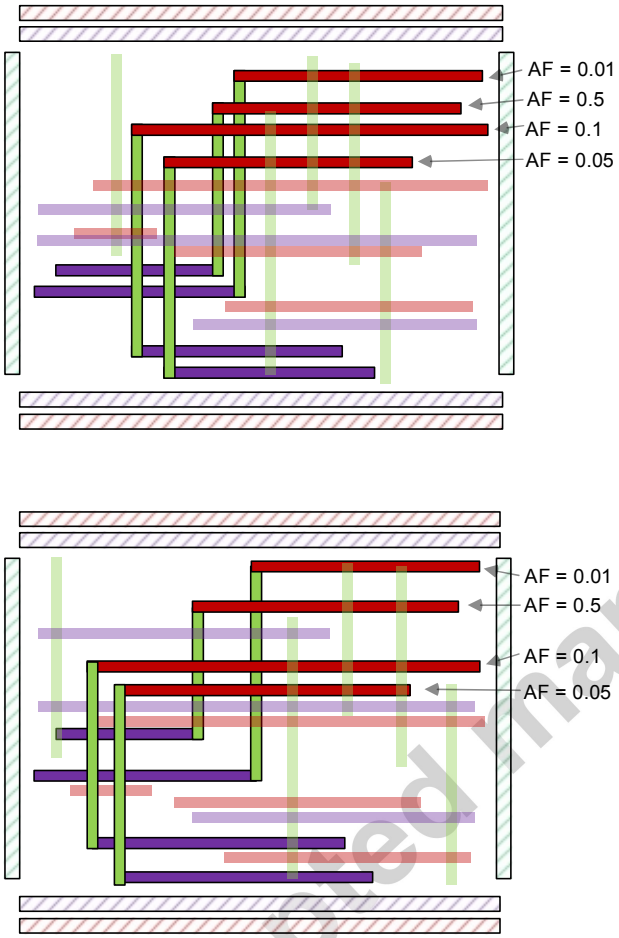


Figure 1. Example of a layout with 4 nets routed on 3 metal layers before and after spacing optimization. The shaded rectangles denote wires bounding routing regions that are not allowed to move, AF stands for activity factor. The wires are spaced according to activitiy factors of the corresponding nets. For example, the wires of the net with AF = 0.5 are allocated larger spaces than those with AF = 0.05.

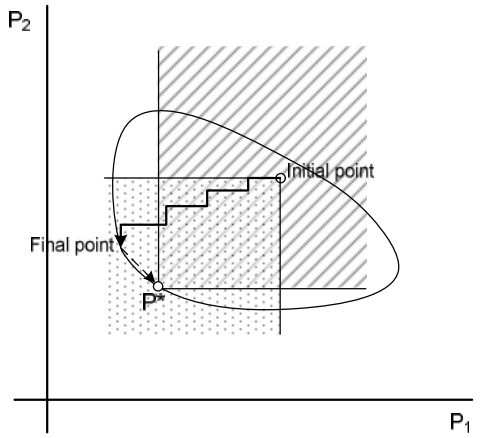


Figure 2. An artificial example of the optimization process. The optimization process of the total power is shown on the power-power plane, where P_1 and P_2 denote the power of two individual wires. The constraint is represented by the solid curve. The iterative improvement of the local objectives (polygonal line) can lead to a sub-optimal point, so that to get to the global optimum P^*, one of the local objectives (P_1 in this case) must be increased (dashed arrow).

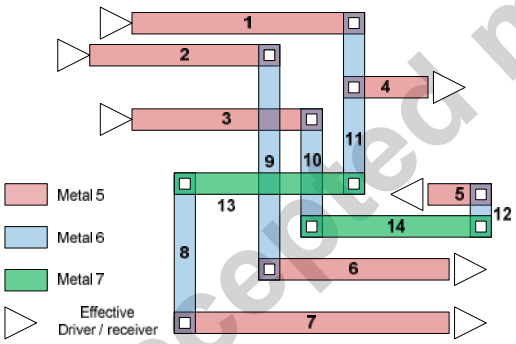


Figure 3. Typical structure of global interconnect

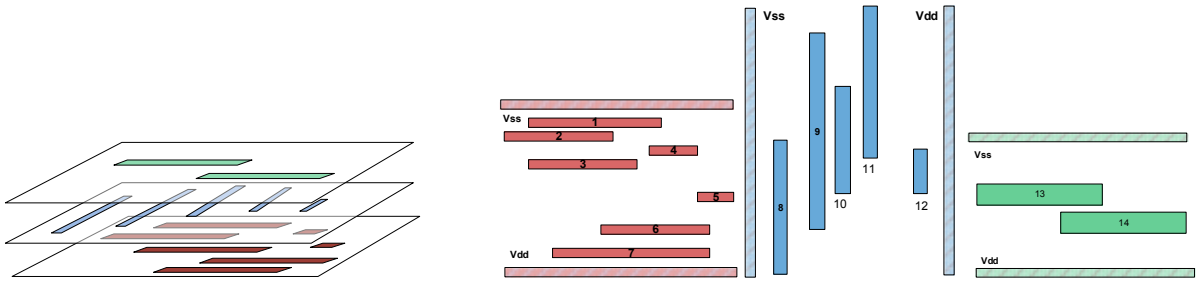


Figure 4. The wire segments from Figure 3 as a collection of three planes. On the right picture, both logic and power grid wires (walls) are shown. Power grid wires are dashed and located at left and right (top and bottom) ends of each plane.

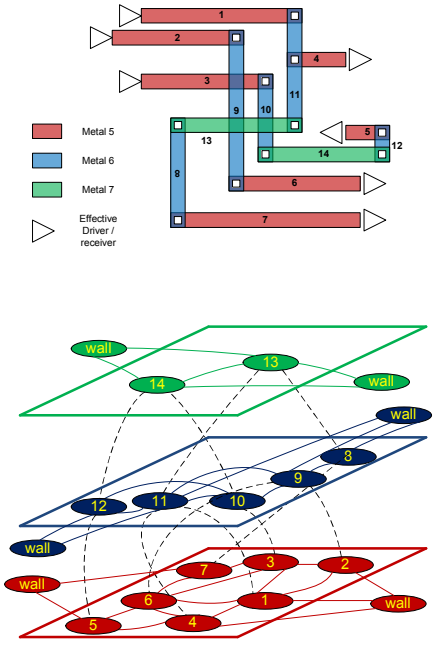


Figure 5. Clip of the layout and corresponding multidimensional visibility graph below it. The solid arrows correspond to visibility edges, the dashed arrows correspond to connectivity edges.

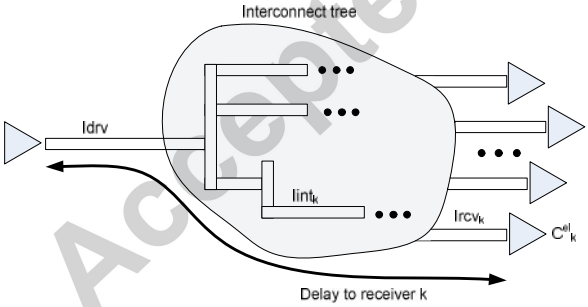


Figure 6. Interconnect tree representation for delay calculation.

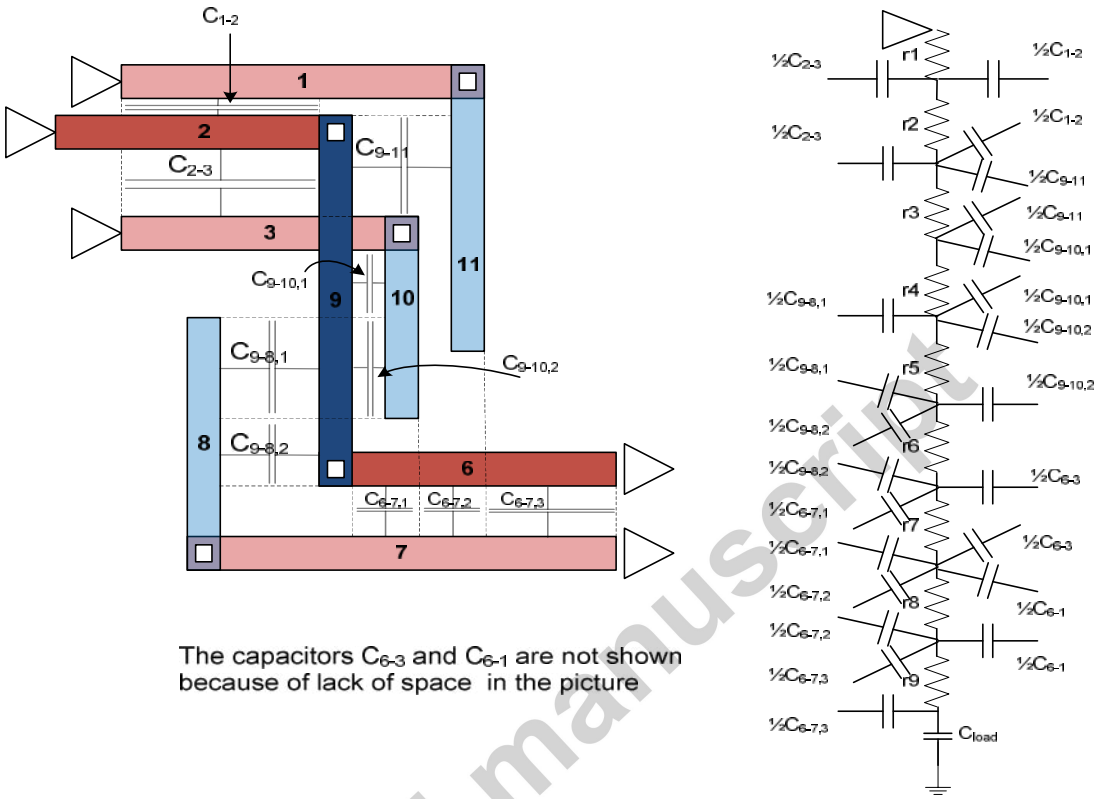


Figure 7. RC tree model of the layout. The RC model on the right represents the relevant cross coupling capacitances shown in the layout on the left.

Algorithm 1 : *Sequential Power Optimization under Delay Constrains*

1. Set $x_{current} = x_{initial}$
2. Set $\eta = \eta_{initial}$
3. Repeat
4. Find x_{new} by solving PODC-LB for given η starting at $x_{current}$
5. Stop if $k\eta < \varepsilon$
6. Update $x_{current} = x_{new}$
7. Update $\eta = \eta \cdot \tau, \quad 0 < \tau < 1$

Figure 8. Algorithm for sequential solving of PODC

Algorithm 2 : L-BFGS for *Power Optimization under Delay constraints*

1. Set $x_0 = x_{current}$ from last interior point iteration
2. $k = 0$
3. Repeat
4. Calculate H_k^o according to (15)
5. Calculate iteratively $H_k \nabla P^{cross}{}'(x_k)$
6. Update $x_{k+1} = x_k + t \cdot H_k \nabla P^{cross}{}'(x_k)$
7. If $k > m$
8. remove pair $\{\Delta x_{k-m+1}, \Delta g_{k-m+1}\}$
9. Calculate and store $\{\Delta x_{k+1}, \Delta g_{k+1}\}$
10. $k = k+1$
11. Until $\left|\nabla P^{cross}{}'\right| < \varepsilon$

Figure 9. Algorithm for solving of PODC-LB.

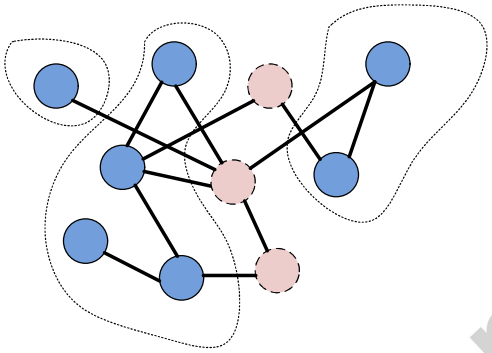
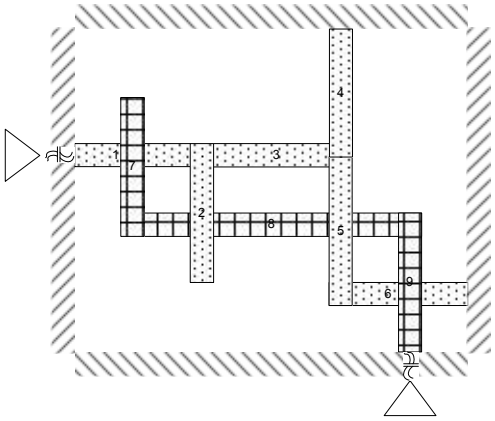


Figure 10. An example of the graph partitioning. The active vertices have a solid boundary, while the inactive vertices (separating group) have a dashed boundary. The graph is separated into three groups.

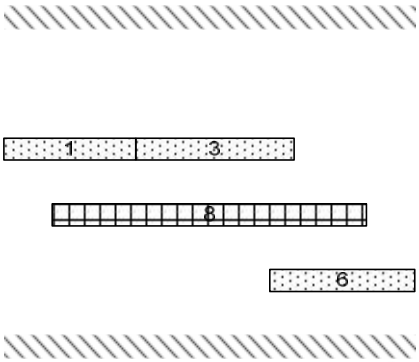
Algorithm 3 : Separation of *Node Visibility Graph* to independent groups

1. Assign $G_i = \{v_i\}, 1 \le i \le N$
2. Foreach $1 \le i \le N$
3. Foreach $i < j \le N$
4. If nets corresponding to vertices v_i and v_j are visible to each other
5. **Union** G_i and G_j
6. End
7. End
8. End

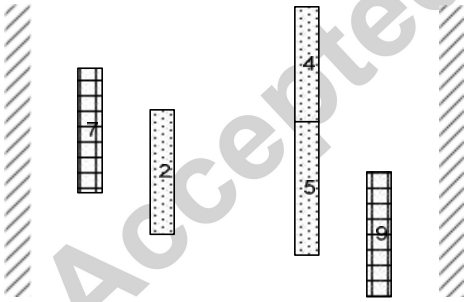
Figure 11. The algorithm for separation of node visibility graph.



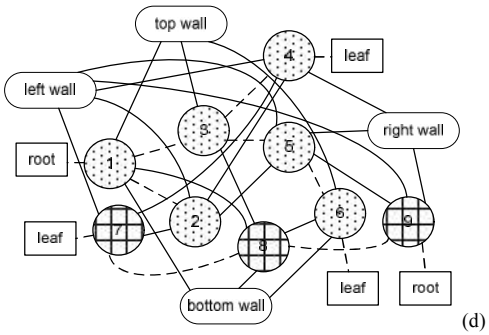
(a)



(b)



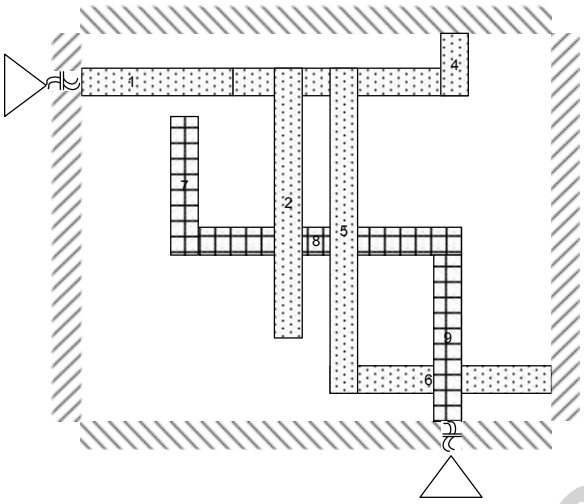
(c)



(d)

Figure 12. A small example of the layout for the multi-layer power optimization. A) The full layout including the two metal layers, two nets and 9 wire segments. B) and C) Horizontal and vertical metal layers and the wires occupying them. D) The multi-layer visibility graph corresponding to the layout from A). The visibility relationships are shown by solid edges, the connectivity relationships are shown by dashed edges.

(a)



(b)

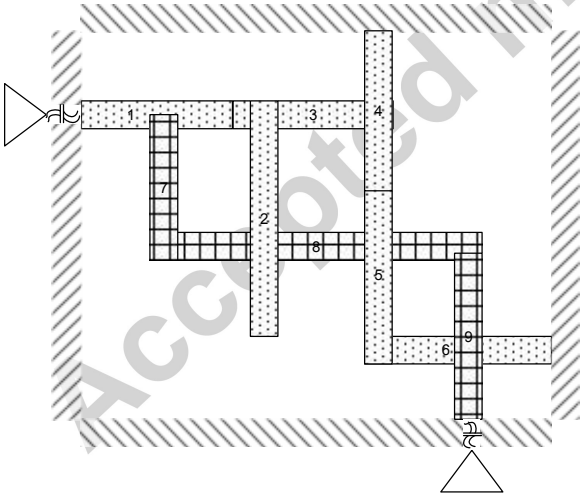


Figure 13. A) Layout after optimization without timing constraints. B) Layout after optimization with timing constraints. In the second case delay constraint prevented wires 1, 3 and 6 from moving too close to the wall as well as wires 2 and 5 from getting too close to each other.In both pictures the drivers are shown schematically. Notice that the real driver cells are not moved. The end-point segments (denoted earlier by I_{drv} and I_{rcv}) can be fixed in place by adding additional constraints, however, this was not applied in this experiment. Also, pairs of wires (1,3) and (4,5) were kept together to prevent jog creation.

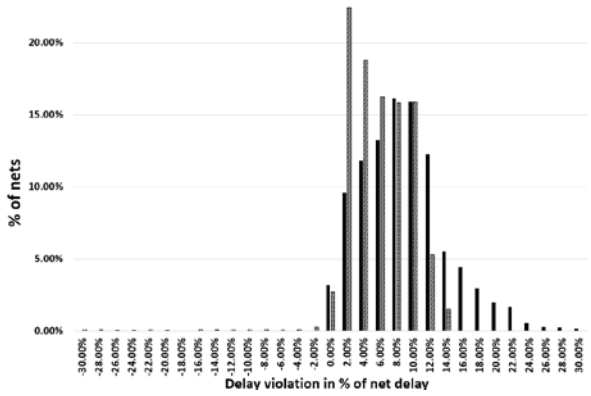


Figure 14. The distribution of the delay violation in % of net delay before (black solid) and after (dashed) optimization. A negative number signifies a delay violation, and a positive number indicates available slack. There are no delay violations in the initial state. After optimization there are some delay violations (in approximately 5% of the nets). More nets have less available slack than earlier.

Tables

Table 1 Optimization Results For The Small Example.

		Initial state	Opt. without delay constraints		Opt. with delay constraints		
Wire coordinates	1	8.50	12.18	10.89			
	2	5.50	7.46	6.69			
	3	8.50	12.43	11.25			
	4	11.50	13.47	10.95			
	5	11.50	9.50	10.21			
	6	2.50	1.56	2.42			
	7	2.50	3.71	3.07			
	8	5.50	6.53	6.23			
	9	14.50	13.25	13.85			
Total power (Improvement %)		15.37 (0%)	10.00 (35%)		11.00 (28.4%)		
Wire delay data	Rcv. number	Delay	Delay	Diff. vs. initial	Req. time	Delay	Slack or diff. vs. initial
	2	44	53	−9	–	47	−4
	4	59	67	−8	–	62	−5
	6	67	77	−10	70	70	0
	7	43	31	+12	–	35	+4

Table 2 Optimization results for real industrial layout segments.

No. of clip	Clip area in mm^2 (% of total area in brackets)	Initial Power	Final Power	Improvement, %	No. of wires (variables)	No. of spaces (location constraints)	No. of delay constraints	Run-time, sec.
1	0.52 (4.6%)	863.8504589	817.119679	5.41%	4091	21518	1427	8.8
2	1.29 (11.3%)	2723.372233	2552.8175	6.26%	37177	110962	13860	523.8
3	0.60 (5.3%)	2068.078358	1974.36814	5.67%	14403	51166	2906	1039.9
4	1.46 (12.8%)	1685.869617	1550.59565	8.02%	13397	47450	4639	70.8
5	1.77 (15.6%)	3737.549076	3306.77984	11.53%	27639	96031	7003	449.0
6	1.33 (11.7%)	3531.584387	3331.86887	5.66%	25343	89996	7161	441.3
7	2.73 (23.9%)	2058.194188	1799.12777	12.59%	22669	79838	7169	292.7
8	1.68 (14.8%)	3084.118285	2827.55122	8.32%	25537	87810	7331	365.2
Total	**11.38 (100%)**	**19752.6166**	**18160.2287**	**8.18%**	**170256**	**584771**	**51496**	**3191.5**

Table 3 Comparison of new method with THE layer-by-layer optimization – power reduction in % and run-time in sec.

Clip no.		1	2	3	4	5	6	7	8
Layer-by-layer	Power Reduction	4.9%	4.5%	5.05%	6.45%	8.67%	4.03%	11.3%	7.12%
	Run time	5.4	327.8	565.3	28.9	241.1	311.2	105.9	189.0
New method	Power Reduction	5.41%	6.26%	5.67%	8.02%	11.53%	5.66%	12.59%	8.32%
	Run time	8.8	523.8	1039.9	70.8	449.0	441.3	292.7	365.2
Improvement in %	**Power Reduction**	**10.1%**	**39.1%**	**12.3%**	**24.3%**	**33.0%**	**40.4%**	**11.4%**	**16.8%**
	Run time	**–67%**	**–59.7%**	**–84%**	**–145%**	**–86.2%**	**–41.8%**	**–176%**	**–93%**