IRWIN AND JOAN JACOBS
CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES

Timing aware power minimization in VLSI circuits by simultaneous multilayer wire

K. Moiseev, A. Kolodny and S. Wimer

CCIT Report #775
September 2010

Electronics
Computers
Communications

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL

CCIT Report #775 September 2010

Timing aware power minimization in VLSI circuits by simultaneous multilayer wire spacing

K. Moiseev, A. Kolodny and S. Wimer

Abstract

Due to continuous technology scaling, the interconnect delay and power reduction is becoming one of the most important design challenges. In this paper, we present a novel algorithm for simultaneous multilayer interconnect spacing so that on the one hand the total power of interconnect is reduced and on the other hand, delay constraints are not violated. We first present an optimization problem and show that it is convex and therefore has unique optimal solution. Then, we develop algorithm which solves the optimization problem. The optimization we propose can be applied to individual nets as well as to large layouts due to smart layout partitioning scheme applied. We demonstrate algorithm effectiveness by showing power reduction of 5-12% of interconnect power on clips from real industrial layout of 32 nm technology node. In addition, we show relation of new optimization technique with previously reported Weighted Power-Delay Sum optimization (WPDS).

1. Introduction

The continuous scaling of technology process, trend towards mobile battery-operated electronic products and growing awareness to environmental heating have caused power minimization to become an important design challenge. Today, in order to produce power-efficient and high-speed VLSI products, power optimizations should be performed at all stages of the design flow: starting from architecture through RTL and circuit implementation and up to layout design. Generally, every opportunity to contribute to power saving is considered. On the other hand, circuit performance remains the most important design objective and power optimization cannot neglect timing constraints imposed on circuits. Therefore, any power optimization should be timing-aware, which is the topic of this paper.

One of the largest power components in VLSI processors is interconnect power, i.e. the power dissipated due to charging and discharging of wire capacitances [3]. These capacitances generally are divided in two groups: capacitances between wires on the same metal layer and capacitances between wires on different metal layers. The contribution of the former, especially at high metal layers, grows with the nonuniform scaling of technology [7], because wire aspect ratios (thickness/width) tend to grow. Thus, cross-coupling capacitances between adjacent wires at the same metal layer have a major effect on both circuit timing and power. The cross-coupling capacitances can be decreased by increasing of inter-wire spaces. In this paper we show how the total power dissipation can be improved by reallocation of inter wire spaces without violating delay constraints, while maintaining the same layout area. We assume that interconnects have been routed (manually or automatically), and their relative locations are not subject to any change (i.e. layout topology is unchanged). It is also assumed that wire widths have been set to satisfy signal delay and other design goals such as reliability, and shield wires have been inserted to eliminate crosstalk noise on sensitive nodes. Hence, the method aims only to modify wire-to-wire capacitance densities across the whole layout in each of the top level interconnect layers. For example, a schematic layout before and after optimization is shown at Figure 1. The inter-wire space is reallocated according to wire switching activities. The wires with higher activities are allocated larger spaces while the wires with lower activities are allocated smaller spaces. Notwithstanding, the space reallocation doesn't violate wire delay constraints.

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]], power optimization [12], or combination of timing and power [1] . The authors in [5],[6],[12] used local optimization, optimizing only single net for single objective function. However, since

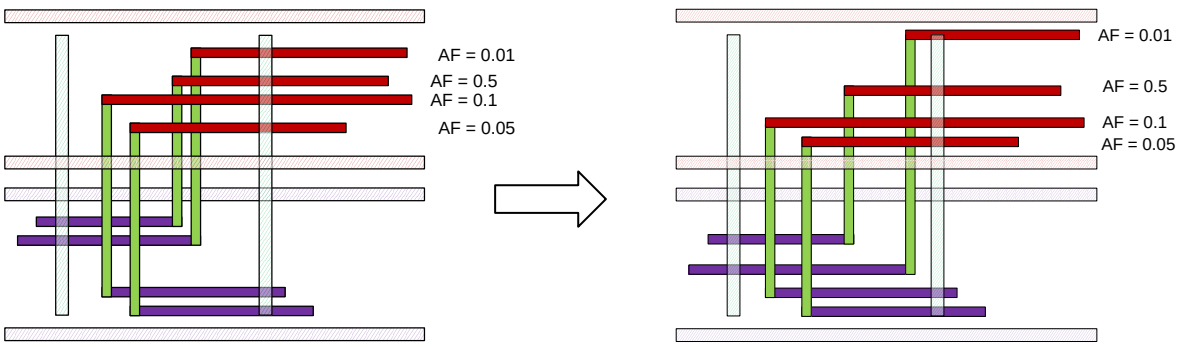

Figure 1. Example of layout with 4 nets routed on 3 metal layers before and after spacing optimization. The dashed rectangles denote wires bounding routing regions, AF stands for activity factor (net's switching activity). The wires are distributed according to activitiy factors of corresponding nets. For example, the wires of net with AF = 0.5 are allocated larger spaces than the wires of net with AF = 0.05.

the power consumption is a cumulative effect, this approach is unacceptable for power minimization. In [1] simultaneous power-delay optimization is performed for many wires together. However, our method has several advantages as compared to [1], which are listed below.

First, [1] uses Weighted Power-Delay Sum (WPDS) as an objective function. This method allows tradeoffs between delay minimization and power minimization, but doesn't guarantee satisfaction of timing constraints. During wire movement, delays of neighbor wires may increase and one of the neighbors can violate its maximal allowed delay. Another method proposed in [1] is wire "freezing", which means that a wire is not moved if such a movement might cause violation of timing constraints of one of its neighbors. This approach indeed prevents violation of timing constraints; however, it does not fully exploit available positive delay slack of neighbor wires. We propose an improvement to the wire freezing method: the wire is moved until its delay or its neighbor delay reaches their maximum allowed values. Such approach fully utilizes available slacks, while it guarantees satisfaction of timing constraints.

Second, the earlier works in this area either optimized a single net residing in several metal layers, or simultaneously optimized many nets residing on the same metal layer. We have made an observation that in order to achieve the global minimum both these methods should be combined. Optimization layer-by-layer neglects interaction between wires residing in the different metal layers. On the other hand, optimization net-by-net doesn't take into account relations between neighbor wires on the same metal layer. Thus, these two optimization problems, being convex, don't exploit the full optimization space. In this paper we show that simultaneous multi-layer multi-net optimization problem is also convex and achieves a global minimum. This formulation allows a global view of interconnect power and interconnect delay, while taking into account the delay criticality of individual wires.

From the design point of view, both WPDS and power optimization under delay constraints are essential optimization techniques which can be used in different stages of the design flow. The former is used in earlier stages of the design, when no individual delay requirements are imposed on nets. Its accuracy is not very high, but it can tune the interconnect design to the right corner in terms of power-delay optimality. The latter can be used in later design stages when exact delay requirements are known and delay violations are not possible

because of high sensitivity of the design in its final stages. The power optimization under delay constraints doesn't have to be applied to the whole layout and can be used only for tuning most delay critical or most power consuming nets.

The rest of this paper is organized as follows. In the next section the layout model is presented and optimization problem is defined. The solution of optimization problem and implementation of the algorithm are presented in Section 3. The relation of the problem to Weighted Power-Delay Sum (WPDS) optimization is discussed in section 4. Algorithm complexity is analyzed in Section 5. Practical considerations of power-delay optimization are discussed in Sections 6 and 7. The examples and experimental results are brought in Section 8. Finally, section 9 concludes the paper.

2. Interconnect modeling and problem definition

The following notation is used through the paper. We summarize it here in order to simplify reading:

N - *Total number of routed nets*

L - *Total number of metal layers*

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

A_l - *Total routing area at layer* l

σ_i - *The* $i-th$ *net*

$I_{i(p),l}$ - *The* $i-th$ *wire routed on layer* l *and belonging to* $p-th$ *net*

Q_i - *Total number of effective loads (pins) of* $i-th$ *net*

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

W_i - *Total number of wire segments belonging to* $i-th$ *net*

$d_{ij,l}$ - *Length of the common span of wires* i*and* j *routed on layer* l

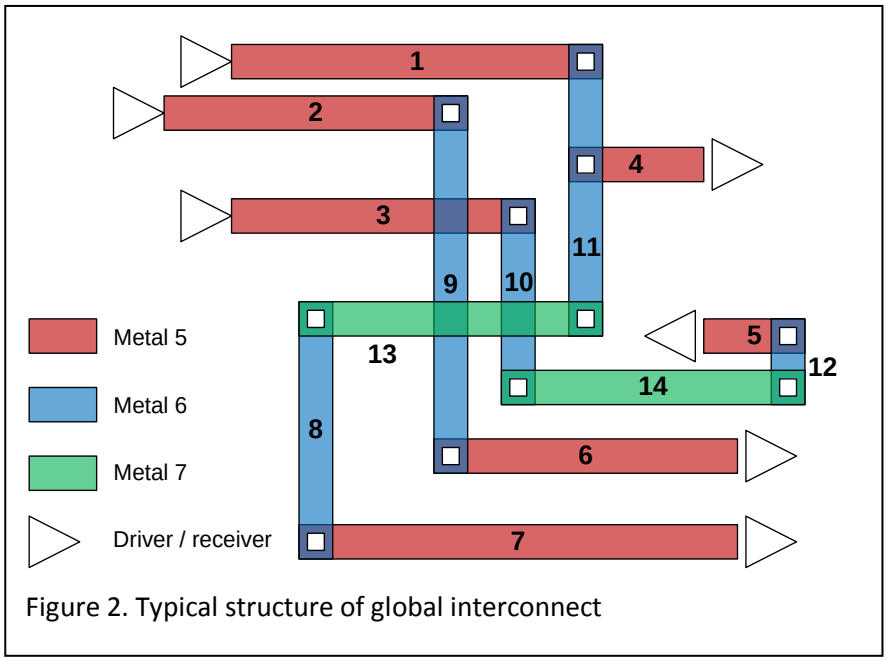

Figure 2. Typical structure of global interconnect

$s_{ij,l}$ -*Spacing between wires* i *and* j *routed on layer* l

The high metal layers in modern VLSI circuits (e.g. layers 5, 6, 7, 8, 9, …) are typically used for long-range interconnect routing. The wires at these "global routing" levels span distances of hundreds or thousands of microns across the chip. An example of such interconnect network is schematically shown in Figure 2, where all wire segments are numbered.

Since the signals connect devices residing at the bottom of the interconnect stack, some portion of the routing is made on the lower metal layers (1, 2, 3, 4). The **effective driver** and **effective receiver load** are used to represent the local routing, such that resistance of these wires is included in the driver model, and their capacitance is included in the load model.

Let N be a total number of routed nets $\sigma_1,\ldots\sigma_N$. We assume that each net is assigned an **activity factor** α_i, $1 \le i \le n$, which denotes the amount of signal's switching relative to the clock signal. It can range from $\alpha_i = 0$ if the signal never switches (e.g., shields or power delivery wires) to $\alpha_i = 1$ if it toggles twice at every cycle (e.g., clocks). Signal activity factors are derived by using an industrial power simulator which checks the signal activity in different scenarios, and then averaging activities over all cases [13],[14].

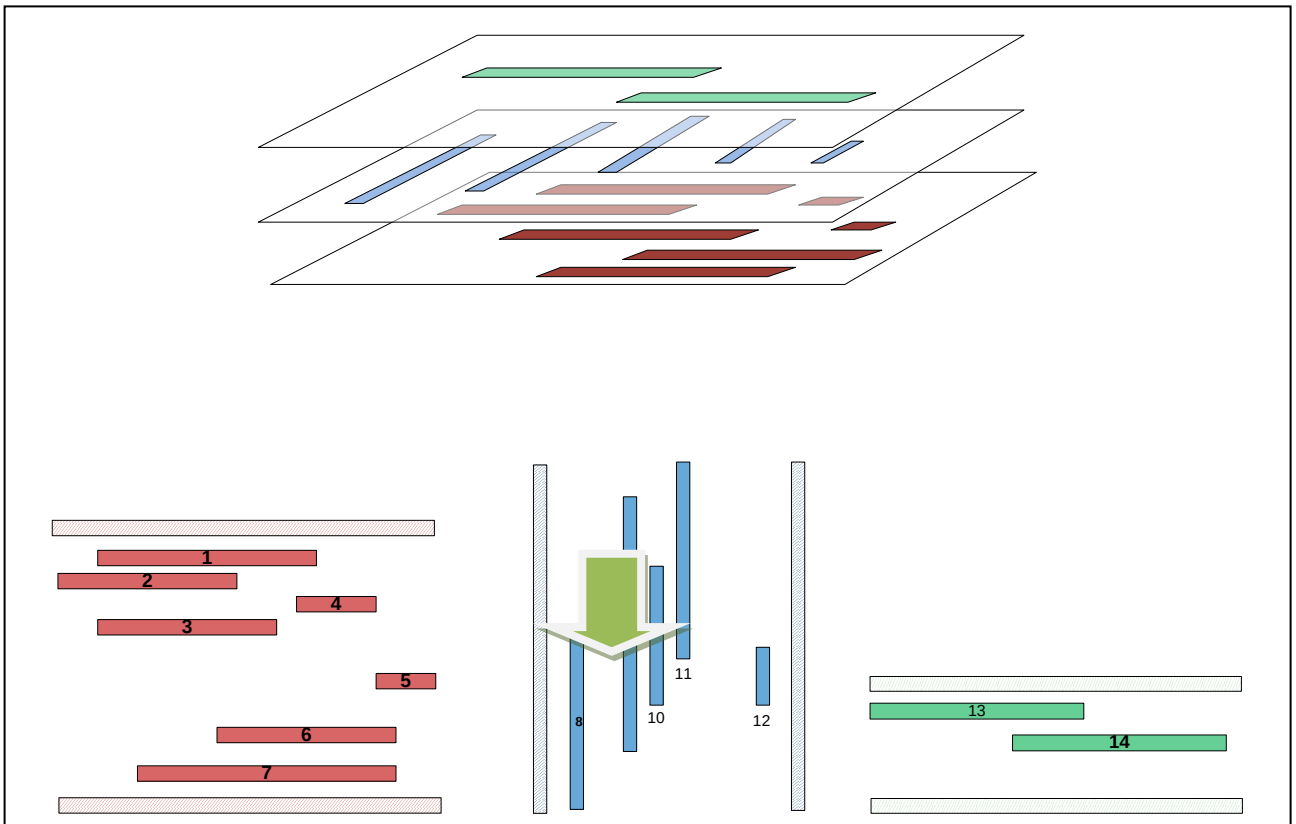

Figure 3. Representation the wire segments from Figure 2 as a collection of three two-dimensional planes.

The three-dimensional structure of global interconnect consisting of L routing layers can be represented as a collection of two-dimensional planes, each of which includes all wire segments routed on the same metal layer l, $1 \le l \le L$. We look at wires in areas within each layer which are bounded by two power grid wires connected to constant voltages. These supply wires serve as "walls" of the routing area. An example of such representation is shown in Figure 1. The "wall" wires are dashed.

In modern VLSI circuits at each routing layer almost all wires are routed in the same direction (i.e. either vertical or horizontal). The only exceptions are "jogs" – short wire segments going in the direction perpendicular to the main layer direction. Since the number of jogs is usually small and they are short, their influence on the power and delay is minor, therefore we neglect jogs in the following discussion. Let's denote wires routed on the same metal layer l by $I_{i(p),l}$, $1 \le i \le N_l$, where N_l is number of wires routed at layer l. The index p in brackets means that

the wire I_i is a part of net p, $1 \le p \le N$. We denote by $w_{i,l}$ and $d_{i,l}$ the width and the length of wire $I_{i(p),l}$, and by $s_{ij,l}$ and $d_{ij,l}$ the spacing and the common span of wires $I_{i(p),l}$ and $I_{j(q),l}$. We call two wires routed on the same metal layer **visible** to each other if their common span $d_{ij,l}$ is non-zero and can be connected by the line not crossing other wires.

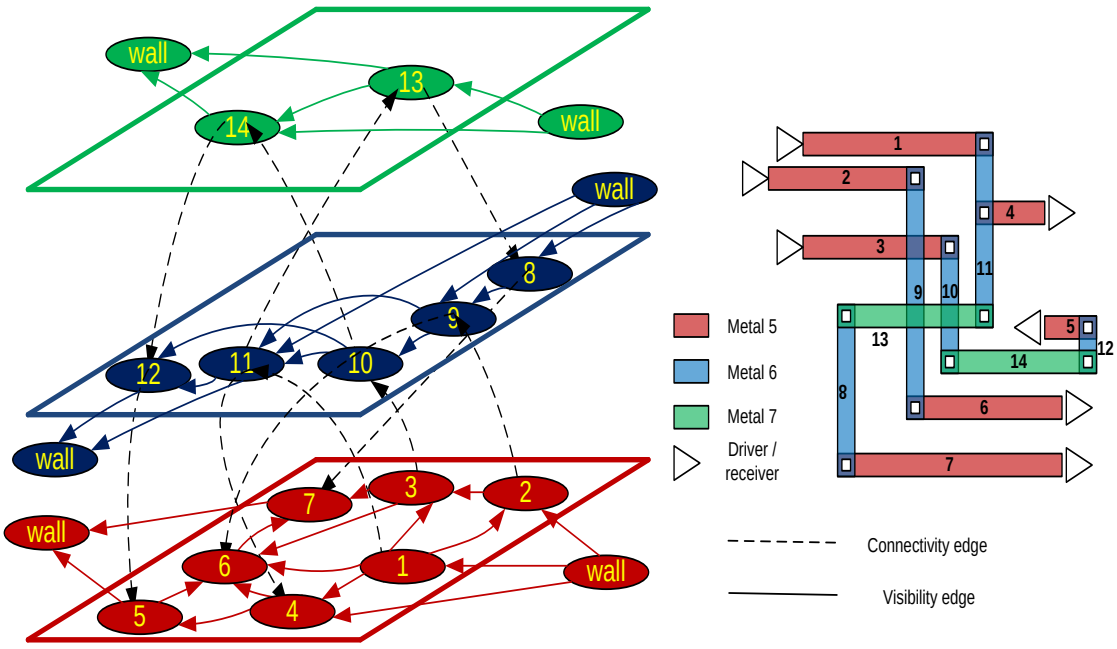

Figure 4. Multidimensional visibility graph on the left for the layout shown on the right. The solid arrows correspond to visibility edges, the dashed arrows correspond to connectivity edges

The 3-dimensional interconnect structure shown in Figures 2 and 3 is represented by a *multilayer visibility graph* $G(V,E)$ as follows. For each wire $I_{i,l}$ we associate vertex $v_{i,l}$ in the graph G. The vertices $v_{0,l}$ and $v_{N_l+1,l}$ corresponds to "wall" wires of each routing layer l. There are two kinds of edges in the graph. Each two vertices $v_{i,l}$ and $u_{j,l}$ which correspond to wires $I_{i,l}$ and $I_{j,l}$ visible to each other are connected by a **visibility** edge. Each two vertices v and u which are physically connected to each other (they are usually routed on different layers) are connected by **connectivity** edge. The example of a multilayer visibility graph is shown in Figure 4.

Different models exist for modeling cross-coupling capacitances between two adjacent wires [2],[15],[16]. Most of them involve spacing between wires as well as widths and heights of coupling wires. It is commonly accepted that coupling capacitance between adjacent wires is a

monotonically decreasing function of inter-wire spacing and is directly proportional to the length of common wire span. Denoting by $f\left(s_{ij,l}\right)$ a monotonically decreasing function of inter-wire spacing between two visible wires $I_{i(p),l}$ and $I_{j(q),l}$ and by $d_{ij,l}$ the length of common span between wires $I_{i(p),l}$ and $I_{j(q),l}$, the following expression represents line-to-line capacitance associated with $I_{i(p),l}$ and $I_{j(q),l}$:

$$c_{ij,l} = \kappa d_{ij,l} f\left(s_{ij,l}\right) \tag{2}$$

It is commonly accepted that $f\left(s_{ij,l}\right) \sim 1/s_{ij}^{\gamma}$, where $\gamma \geq 1$. In this paper we don't make any assumption on the special form of function $f\left(s_{ij,l}\right)$ except for being convex in $s_{ij,l}$ (which is true for the model above). By definition, if wires $I_{i(p),l}$ and $I_{j(q),l}$ are not visible to each other, then $d_{ij,l} = 0$, so actually only cross capacitance between visible wires is non-zero in our model.

For a given wire, line-to-line capacitances to its visible wires can influence the dynamic power or delay associated with the wire's net. The dynamic power of the net σ_i can be expressed by:

$$P_i = P^{self} + P^{cross} = \alpha_i C_i^a V_{dd}{}^2 f + \alpha_i C_i^{ll} V_{dd}{}^2 f \ . \tag{3}$$

In (3), α_i denotes the net's activity factor, V_{dd} is voltage swing and f is clock frequency. C_i^a is the total net's self capacitance, contributed by capacitors formed by the net's wires and lower and upper layout layers, and C_i^{ll} is the total net's cross-capacitance, formed by spaces between net's wires and their neighbors on the same metal layer. The associated power consumption parts are denoted P^{self} and P^{cross} correspondingly. We assume that the widths of the wires are set by wire sizing optimization performed earlier to satisfy timing requirements [19] and thus are not subject to change in the spacing optimization. This assumption matches VLSI design practice, where wire widths are set very early in the design flow according to signal propagation delay goals. Thus, only P^{cross} part in (3) is of interest. The effective cross-coupling capacitance C_i^{ll} depends on neighbor wire mutual activities and is usually modeled using a Miller factor [17],[18]. However, since power is a cumulative metric and the exact mutual behavior of any two neighbor wires is not known, we assume a Miller factor of 1 in the rest of the paper.

Using (2), given wire $I_{i(p),l}$, the power contributed by cross-capacitances to its neighbors can be expressed by:

$$P_{i(p),l}^{cross} = \alpha_p k' \sum_{j=1, j\neq i}^{N_l} d_{ij,l} f\left(s_{ij,l}\right) \tag{4}$$

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

$$P^{cross} = k' \sum_{l=1}^{L} \sum_{i=1}^{N_l} \sum_{j=1}^{N_l} d_{ij,l} \left(\alpha_p + \alpha_q\right) f\left(s_{ij,l}\right) \tag{5}$$

Let's denote by Q_p the number of driven receivers $C_{j(p)}^{el}$, $1 \le j \le Q_p$ of net σ_p and by W_p the total number of wire segments $I_{j(p)}$, $1 \le j \le W_p$ of net σ_p.

For delay calculation purposes, each net is represented as an interconnect tree. Each wire segment or part of the segment is modeled as a π-load so that its total capacitance is equally divided between two resistor ends. The decoupled line-to-line capacitance is counted together with the segment self capacitance (Figure 5).

It can usually be accepted that the delay of interconnect path is convex function of spacing of wire segments appearing on the path. For example, using Elmore delay as the simplest approximation, the delay from net driver to receiver $C_{j(p)}^{el}$ can be expressed as $T_{j(p)} = \sum_{i=1}^{W_p} C_i R_{ik}$, where C_i is the capacitance tied at junction i of the interconnect tree and R_{ik} is the total resistance of the common part of (unique) path from net effective driver to the receiver $C_{j(p)}^{el}$ and (unique) path from net effective driver to junction i. Since capacitance values C_i incorporate also coupling capacitance terms (which are convex in spaces as clamed earlier), the delay from effective driver of net σ_p to receiver $C_{j(p)}^{el}$ can be expressed as

$$T_{j(p)} = g\left(\boldsymbol{s}\right) \tag{6}$$

where $\boldsymbol{s} = (s_1, s_2, \ldots)$ - vector of inter wire spaces between wires of net p and their neighbors and

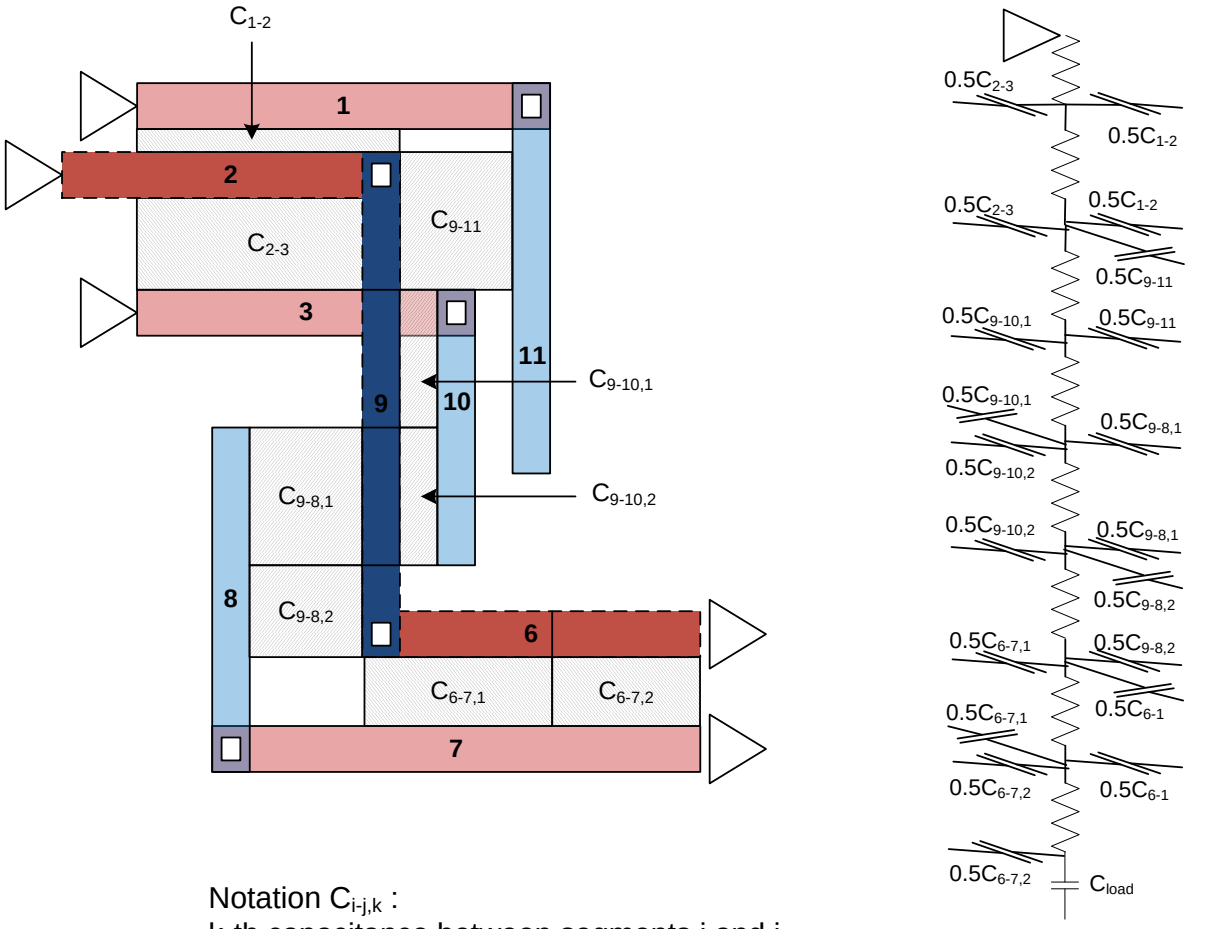

Figure 5. RC tree modeling of the layout. On the left: layout of net (emphasized) with neighbors (only relevant neighbors are shown). On the right: corresponding RC model. Only cross-soupling capacitances are shown.

g is convex function in each one of $s_1, s_2, \ldots$.

Notice that the expression (6) remains true even if much more accurate models [15] are used for net delay estimation. We used models developed in [15] to check this assumption and found that for typical design conditions the convexity assumption is true. Again, we don't make any assumption on the specific form of function g except for being convex.

We are aiming at finding a set of spaces s_{ij} so that total power (5) is minimized. This optimization problem is subjected to number of constraints, which are listed below.

First, each space s_{ij} should satisfy

$$s_{ij,l} \geq s_{\min,l}, \tag{7}$$

i.e. any two wires should be apart of each other by at least $s_{\min,l}$, which is called *minimum spacing rule*. Minimum spacing depends on routing layer and can be different in different routing layers.

Second, circuit timing requirements should not be violated. Let's denote by $D_{j(p)}$ the required arrival time at the receiver j of net σ_p. Then the following constraints should be satisfied:

$$T_{j(p)} \leq D_{j(p)}, \tag{8}$$

i.e. the delay at each receiver should not exceed the corresponding required time. In the aggressive design technology the constraint (8) may have form

$$T_{j(p)} + \Delta \leq D_{j(p)},$$

where Δ is a positive margin taken into account due to silicon variatons.

Third, the spacing optimization should be performed within boundaries of each routing layer. Let's denote by Ω_l a set of all paths in the visibility graph between vertices corresponding to wall wires of layer l consisting of visibility edges only. Let's denote by $\omega_l = \left(w_{i_1}, s_{i_2}, w_{i_3}, s_{i_4}, \ldots\right)$ a single path in Ω_l and by $\left|\omega_l\right|$ the number of wires in ω_l. Then the following set of constraints should be satisfied:

$$\sum_{\substack{w_i \in \omega_l \\ s_j \in \omega_l}} w_i + s_j \leq A_l \text{ for all } \omega \in \Omega \text{ and for all } l, \tag{9}$$

where summation is done over all wire widths w_i and inter-wire spaces s_j for all wires lying on the path ω_l.

Using (5)-(9), the optimization problem can be naturally formulated as follows:

Program *P1*

$\min P^{cross}$

s.t.

$s_{ij,l} \geq s_{\min,l}$, *for all* $1 \leq l \leq L$ *and* $1 \leq i, j \leq N_l$ *such that* I_i *and* I_j *are visible*

$T_{j(p)} \leq D_{j(p)}$, *for all* $1 \leq p \leq N$ *and* $1 \leq j \leq Q_p$

$\sum_{\omega_l} w_i + s_j \leq A_l$, *for all* $1 \leq l \leq L$ *and* $\omega \in \Omega$ *where* $w_i \in \omega_l$ *and* $s_j \in \omega_l$

3. Solution of the optimal spacing problem

Observation: Program 1 is convex.

Proof: □ Objective function and delay inequality constraints are convex in s_{ij} by definition. Minimum spacing and boundary constraints are linear and thus convex. Therefore, the optimization problem is convex. ■

Although program 1 is convex, it is impractical in its current formulation, since the number of boundary constraints equals to number of paths in Ω which can be exponential in the number of wires. The difficulty can be handled by the classical technique of introducing new variables and partitioning constraints on a path into constraints on path components. Let's denote by $x_{i,l}$ coordinates of centerlines of wires $I_{i,l}$. The relation 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 wall wire coordinates are $x_{0,l} = 0$ and $x_{N_l+1,l} = A$ and their widths are zero, it is easy to see that the following program is equivalent to Program P1:

Program *P2*

$\min P^{cross}$

s.t.

$T_{j(p)} \leq D_{j(p)}$, *for all* $1 \leq p \leq N$ *and* $1 \leq j \leq Q_p$

$x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 \geq s_{\min,l}$, *for all* $1 \leq l \leq L$ *and* $0 \leq i, j \leq N_l + 1$ *such that* I_i *and* I_j *are visible*

Notice that mutual location constraints also contain boundary constraints. Indeed, summing up mutual location constraints on some path $\omega_l \in \Omega_l$, we obtain $A_l - \sum_{w_{i,l} \in \omega_l} w_{i,l} \geq |\omega_l| \cdot s_{\min,l}$, which is always true if the problem is feasible.

The convexity of P2 allows us to directly apply Newton's method, providing the Newton step doesn't take the solution out of the feasibility region. To ensure this, we use the interior-point method [20]. The interior point method has been used in the past for transistor sizing [27]. For current optimization problem the method is applied as follows. We introduce an additional variable $\eta > 0$ and form a **log-barrier** function:

$$LB(x;\eta) = -\eta \sum_{\substack{1 \leq l \leq L \\ 0 \leq i,\, j \leq N_l + 1}} \log\left(x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 - s_{\min,l}\right) - \sum_{\substack{1 \leq p \leq N \\ 1 \leq j \leq Q_p}} \log\left(D_{j(p)} - T_{j(p)}\right) \qquad (11)$$

The domain of function (11) is the set of points which satisfy inequality constraints of P2 strictly. The logarithmic barrier grows without bound if any of the inequality constraints come close to equality. The new objective function is formed by

$$P^{cross'}(x;\eta) = P^{cross}(x) + LB(x;\eta) \qquad (12)$$

and the new optimization problem is simply an unconstrained program

Program *P3*

$$\min\left(P^{cross} - \eta \sum_{\substack{1 \leq l \leq L \\ 0 \leq i,\, j \leq N_l + 1}} \log\left(x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 - s_{\min,l}\right) - \eta \sum_{\substack{1 \leq p \leq N \\ 1 \leq j \leq Q_p}} \log\left(D_{j(p)} - T_{j(p)}\right) \right)$$

The program P3 is only an approximation of program P2 and its quality improves as the parameter η decreases [20]. Let's denote by $x^*(\eta)$ the solution of P3 for given η. The **central path** associated with the problem P2 is defined as the set of points $x^*(\eta)$ for $\eta > 0$. One can show that the central path $x^*(\eta)$ converges to solution of the problem P2 x^* as $\eta \to 0$ and that $x^*(\eta)$ is $k\eta$ – suboptimal [20] (k denotes total number of inequality constraints). The problem P2 is solved by solving sequence of problems P3 for decreasing values of η, starting each

iteration at the solution of the problem for the previous value of η. The algorithm for solving P2 through sequential solving of P3 is shown at Figure 6.

The problem P3 is an unconstrained convex optimization problem and can be solved by Newton's method as follows. Given initial feasible point x, the Newton step direction is calculated by $\Delta x_N = -\nabla^2 P^{cross'}(x)^{-1} \cdot \nabla P^{cross'}(x)$. Then the location is updated by $x = x + t \cdot \Delta x_N$, where t is a step size calculated by line search along direction Δx. Although Newton method is known for its fast convergence, the calculation and storage of Hessian $\nabla^2 P^{cross'}(x)$ and its inverse is not always possible for real cases involving thousands of optimization variables. Even if Hessian of original function $\nabla^2 P^{cross}(x)$ is sparse or close to sparse, log-barrier operation usually turns the Hessian to dense which makes impossible calculation of its inverse. Therefore, we use L-BFGS quasi – Newton method [21] which has on the one hand, super linear rate of convergence, and on the other hand doesn't require calculation of full Hessian inverse, but the small number of vector-by-vector or vector-by-matrix multiplications. According to it, the inverse of original Hessian matrix is replaced by inverse of Hessian approximation matrix, which is recalculated at each iteration based on its previous iteration's value. Denoting by Δg gradient change $\nabla P^{cross}{}'(x_{k+1}) - \nabla P^{cross}{}'(x_k)$ and by Δx variable vector change $x_{k+1} - x_k$, the inverse of Hessian approximation matrix in $k+1-$iteration is calculated by

$$H_{k+1} = \left(I - \frac{\Delta x \cdot \Delta g^T}{\Delta g^T \Delta x} \right) H_k \left(I - \frac{\Delta g \cdot \Delta x^T}{\Delta g^T \Delta x} \right) + \frac{\Delta x \cdot \Delta x^T}{\Delta g^T \Delta x} \tag{14}$$

Notice that the calculation of H_{k+1} involves only vector-by-vector or matrix-by-vector

Algorithm 1 : *Sequential Power Delay aware Optimization (SPDO)*

1. Set $x_{current} = x_{initial}$
2. Set $\eta = \eta_{initial}$
3. Repeat
4. Find x_{new} by solving P3 for given η starting at $x_{current}$
5. Stop if $k\eta < \varepsilon$
6. Update $x_{current} = x_{new}$
7. Update η

Figure 6. Algorithm for sequential solving of P2

multiplications. The value of H_0 is chosen to be as close as possible to original Hessian inverse. The choice of

$$H_0 = \frac{\Delta g^T \Delta x}{\Delta x^T \Delta x} I \tag{15}$$

is reported to be the most successful in practice [22] and therefore used in our implementation. The scaling factor $\frac{\Delta g^T \Delta x}{\Delta x^T \Delta x}$ attempts to estimate the size of the true Hessian matrix along the most recent search direction.

The storage of H_k still can be expensive for real design cases. Instead of storing full matrix H_k, we save only a few pairs of $\{\Delta x; \Delta g\}$ from the most recent iterations. These pairs are used to construct the inverse Hessian approximation. Curvature information from earlier iterations which is less relevant to the Hessian behavior in current iteration is discarded. The optimization procedure based on this method is processed as follows. In each iteration, first the initial matrix H_k^0 is calculated by (15) based on most recent values of Δx and Δg. Then, the product of inverse Hessian approximation by gradient vector $H_k \nabla P^{cross}{}'(x_k)$ is calculated from H_k^0 by recursive procedure using pairs of $\{\Delta x; \Delta g\}$ stored for last m iterations. Now, the new location is calculated by $x_{k+1} = x_k - t \cdot H_k \nabla P^{cross}{}'(x_k)$. Finally, new values of Δx_{k+1} and Δg_{k+1} are calculated and stored

Algorithm 2 : L-BFGS for *Power Delay aware Optimization*

1. Set $x_0 = x_{current}$ from last interior point iteration
2. $k = 0$
3. Repeat
4. Calculate H_k^o according to (16)
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 7. Algorithm for solving of P3.

instead of least recent pair $\{\Delta x_{k-m+1}, \Delta g_{k-m+1}\}$. The algorithm for solving P3 is shown in Figure 7.

4. Dual problem and relation to Weighted Power-Delay Sum (WPDS) optimization problem

The native relaxation of the program P2 would be

Program *P4*

$\min P^{cross}$

s.t.

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

$x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 \ge s_{\min,l}$, *for all* $1 \le l \le L$ *and* $0 \le i, j \le N_l + 1$ *such that* I_i *and* I_j *are visible*

where $M = \sum_{i=1}^{N} Q_i$ and d_i are optimization variables (the delay constraints can as well be written as $T_i \le D_i + d_i$ which is equivalent). This formulation, however, is not very interesting and equivalent to optimization without delay constrains at all. In order to reflect delay constrains in optimization, variable d_i can be incorporated in the objective function in the following way:

Program *P4.1*

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

s.t.

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

$x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 \ge s_{\min,l}$, *for all* $1 \le l \le L$ *and* $0 \le i, j \le N_l + 1$ *such that* I_i *and* I_j *are visible*

In P4.1 the optimization is performed over the variables x_i and d_i. The delay awareness is reflected by including d_i into objective function. The meaning of P4.1 is the optimization of power under delay constraints, without explicitly specifying delay requirement for each receiver. The delay criticality is defined by relation between weights α and β_i. The program P4.1 is always feasible while the program P2 might be infeasible. Since program P4.1 is convex, and Slater's condition [20] with respect to delay constraints is always satisfied, there exist non-

negative numbers $\lambda_i, 1 \le i \le M$ (Lagrange multipliers) so that solution of the program P4.1 is equivalent to solution of the following dual program D4.1:

Program *D4.1*

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

s.t.

$$x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 \ge s_{\min,l}, \quad \textit{for all } 1 \le l \le L \textit{ and } 0 \le i, j \le N_l + 1 \textit{ such that } I_i \textit{ and } I_j \textit{ are visible}$$

Solving KKT conditions [20] for D4.1 with respect to variables d_i , obtain:

$$\frac{\partial}{\partial d_i}\left(\alpha P^{cross} + \sum_{i=1}^{M} \beta_i \cdot d_i + \sum_{i=1}^{M} \lambda_i \left(T_i - d_i\right)\right) = \beta_i - \lambda_i = 0 \Rightarrow \beta_i = \lambda_i \tag{16}$$

Substituting (16) into objective function of D4.1, it's transformed to:

Program *D4.2*

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

s.t.

$$x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 \ge s_{\min,l}, \quad \textit{for all } 1 \le l \le L \textit{ and } 0 \le i, j \le N_l + 1 \textit{ such that } I_i \textit{ and } I_j \textit{ are visible}$$

Now, solving KKT conditions for D4.2 with respect to variables x_i , we have:

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

On the other hand, assume that λ_i^* are values of dual variables for delay constraints in the optimal point of the problem P2. Then, the problem P2 is equivalent to:

Program *D2*

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

s.t.

$$x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 \ge s_{\min,l}, \quad \textit{for all } 1 \le l \le L \textit{ and } 0 \le i, j \le N_l + 1 \textit{ such that } I_i \textit{ and } I_j \textit{ are visible}$$

Solving KKT conditions for D2 with respect to variables x_i results in:

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

Comparing (17) and (18) it can be seen that in order they have the same solution, it is required that the criticality weights β_i are equal to optimal Lagrangian multipliers λ_i^*. It is clear also that the program D4.2 is equivalent (up to power weight α) to the following Weighted Power-Delay Sum (WPDS) optimization problem:

Program *WPDS*

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

s.t.

$$x_{j,l} - x_{i,l} - \left(w_{i,l} + w_{j,l}\right)/2 \geq s_{\min,l}, \quad \textit{for all } 1 \leq l \leq L \textit{ and } 0 \leq i, j \leq N_l + 1 \textit{ such that } I_i \textit{ and } I_j \textit{ are visible}$$

Similar formulation can be found in [25], where delay weighting was used for simultaneous gate and wire sizing for power. The WPDS problem in its current formulation was discussed in deep in [1]. It represents optimization of total power contributed by cross-capacitances, weighted by delays of net pins. Such an optimization is essential in the early stages of the design, when no individual delay requirements available. The coefficients k_i are non-negative numbers representing pin delay criticalities: for more critical pin the larger weight is set. The criticality coeficients are set in advance and are constant through the optimization process. The question of how to set coefficients k_i optimally remained open in both [1] and [25]. The following theorem summarizes above developments and claims about values of optimal delay criticality weights.

Theorem. *The weighted power-delay optimization (program WPDS) is the relaxation of the power optimization under delay constraints (program P2). The optimal delay criticality weights k_i in weighted power-delay optimization are equal to optimal values of Lagrangian dual variables of corresponding delay constraints in power optimization under delay constraints*.

From the practical point of view, if one uses WPDS approach for optimization, it is still unclear how to set delay criticality weights k_i in advance. In the following we use interior point approximation P3 of the P2 and the fact that WPDS is relaxation of P2 to show how these weights may be initially set.

The optimality conditions for P3 (which is approximation of P2) is $\nabla P^{cross'}\left(x^*\right) = 0$, i.e

$$\nabla P^{cross}\left(x^*\right)-\eta\sum_i\sum_j\frac{e_i}{x^*_{j,l}-x^*_{i,l}-\left(w_{i,l}+w_{j,l}\right)/2-s_{\min,l}}-\eta\sum_{i=1}^{M}\frac{\nabla T_i\left(x^*\right)}{T_i\left(x^*\right)-D_i}=0 \qquad (19)$$

In the double sum the summation is done only on such i and j so that I_i and I_j are visible and e_i is a $i-$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 inter-wire space. In the second sum, by denoting $\lambda^*_i=-\frac{\eta}{T_i\left(x^*\right)-D_i}>0$ the third term of (19) turns to be $\sum_{i=1}^{M}\lambda^*_i\nabla T_i\left(x^*\right)$. Comparing to (18), it can be said that Lagrangian dual variables (which serve also as delay weights in WPDS optimization), are inversely proportional to wire delay slacks. For highly critical wires $T_i\left(x^*\right)\approx D_i$ (small slack) and λ^*_i is large, for less critical wires, $T_i\left(x^*\right)<D_i$ (large slack) and λ^*_i is smaller. In general, it can be seen that $\lambda^*_i\left(D_i-T_i\left(x^*\right)\right)=\eta$. Let's denote by T_i^0 initial receiver delays in WPDS. Since WPDS is applied in early design stages, no individual required times D_i available. Assume that D is the single required time (usually clock period) that all wire delays are tuned to. Then the delay weights for WPDS can be set as:

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

where η is proportionality coefficient. Thus, the WPDS problem can be re-written in the following way:

Program $WPDS-1$

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

s.t.

$x_{j,l}-x_{i,l}-\left(w_{i,l}+w_{j,l}\right)/2\geq s_{\min,l}$, *for all* $1\leq l\leq L$ *and* $0\leq i,j\leq N_l+1$ *such that* I_i *and* I_j *are visible*

5. Complexity analysis

The run-time complexity of the whole algorithm depends on number of interior (L-BFGS) and exterior (log-barrier) iterations.

6. Practical considerations

The formulation P2 as a convex optimization problem with constraints is very convenient for practical optimization cases. In real designs, there are always special nets (such as clock network nets) that are not likely to be moved. Other wires can be frozen in advance because of a variety of reasons (noise, delay, slope etc.). Others can be required to keep predefined distance from their neighbors. All such cases can be easily handled by defining additional constraints on the wires. For example, if wire I_i must have constant 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 log-barrier function. Another example is when two wires should be hold with constant distance between them, in particular, zero. Consider layout in Figure 10a). Wire segments 1 and 3 as well as segments 4 and 5 represent pairs of segments of the same physical wires. Since each wire segment is treated independently, the optimization can end with the segments shifted relatively to each other, which will result by adding jogs and layout complication. In aggressive layout-aware design such changes can be too disruptive and therefore such pairs of wires might be required to be treated as a single wire by the algorithm. This can be achieved by adding 4 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 which is convex in optimization variables (i.e. wire coordinates) can be easily handled by the algorithm.

7. Layout partitioning

The optimization method described in previous sections can be applied to the clip of layout bounded at all metal layers by the immovable wires ("walls"), like shown at Fig.2. However, the full layout of VLSI circuit can consist of several such clips. Power grids, shields and other wires fixed in place can serve as such wall wires. Each one of clips can be optimized independently thus decreasing number of optimization variables and constraints what should be handled simultaneously. In the following we describe how such partitioning can be performed.

We call two nets **visible** if they have visible wires on some of routing layers. We build **net visibility graph** by assigning vertex to each net and assigning edge to each pair of nets visible to each other. According to this defintion, the layout from Fig. 1 will be presented by fully connected graph with 3 vertices and 3 edges, since there are visitibile wires between any two nets.

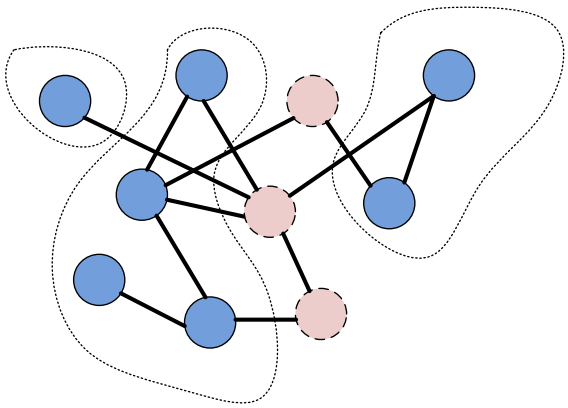

Figure 8. An example of graph partitioning. The active vertices have solid boundary, while the inactive vertices (separating group) have dashed boundary. The graph is separated to three groups.

Let's call by **active vertex** the vertex representing net with at least one movable wire and by **inactive vertex** the vertex representing net with all fixed wires. Inactive vertices can represent power grid nets, shield nets or nets which should not be moved because of some reason. Inactive vertices form separation groups with respect to groups of active nets. For example, in Fig. 8 the inactive vertices (shown by dashed boundary) separate the whole graph to three groups of active vertices (with solid boundary). Each one of the groups can be optimized independently and doesn't affect optimization accuracy of other groups. Thus, instead of optimizing single layout with 7 nodes we optimize 3 layout clips with 1, 2 and 4 nodes correspondingly. The separation to

groups can be easily done with Union-Find algorithm [23]. Assume there are N active vertices in the graph. Then Algorithm 3 finds independent 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 9. The algorithm for separation of node visibility graph.

First, individual group G_i is assigned for each active vertex $1 \le i \le N$. Then, vertices corresponding to 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 natural separation formed by power grid lines and other obstacles can be extended by artifical separation where minimal separating set of active nets is found and used for separation of the rest of active nodes. The efficient algorithm for such vertex separation is described in [24].

8. Examples and experimental results

The algorithms 1, 2 and 3 were implemented in C++ and tested on Pentium M 1.7 GHz processor system with 768 MB of memory. We first demonstrate the algorithm on small artificial layout depicted at Figure 10. The schematic layout of two nets including totally 9 wire segments (all segements are numbered) is shown at Figure 10 a). The dotted net has driver tied at end of wire 1 and receivers tied at ends of wires 2, 4 and 6; the squared net has driver tied at the end of wire 9 and receiver tied at the end of wire 7. The corresponding layouts of individual layers are shown at Figure 10 b) and c), the multi-layer visibility graph is shown at figure 10d), where by dotted edges connectivity relationships are shown while the visibility relationships are shown by solid edges. The activity factors were: 0.1 for net with segments 1, 2, 3, 4, 5, 6 and 1 for net with

segments 7, 8 and 9. We performed two tests with this layout. First, the required arrival times at receivers 2, 4, 6 and 7 were relaxed so that the optimization was guided only by mutual location constraints. In the second test, the required time of the receiver 6 was tightened so that one of 4 delay constrains was active. The optimization results for both cases are presented in Table I and the resulting layouts are shown at Figure 11. Power, delay and coordinates are shown in relative units. It can be seen that in both cases there is significant power improvement: 35% and 28.4%correspondingly. In the second case the optimization impact is smaller than in the first one and the slack at the receiver 6 reaches 0. Moreover, the delay constraint at receiver 6 influenced the rest of delay constraints at the same net so that the delays at receivers change less than in unconstrained optimization.

As large test cases we used clips of real layout of state-of-the-art 32 nm industrial design. The layout of high metal layers (5, 6, 7, 8) was preserved while parts of nets consisting of the segments on the lower metal layers were replaced by corresponding effective drivers and receivers. In implementation we used capacitance models presented in [2], which are consistent

Table I. Optimization results for artificial 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 (Impr. %)		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

with our assumptions on cross-coupling capacitance. For delay estimation we used Elmore delay model with π – model for individual net segments. Although Elmore delay is known as non-

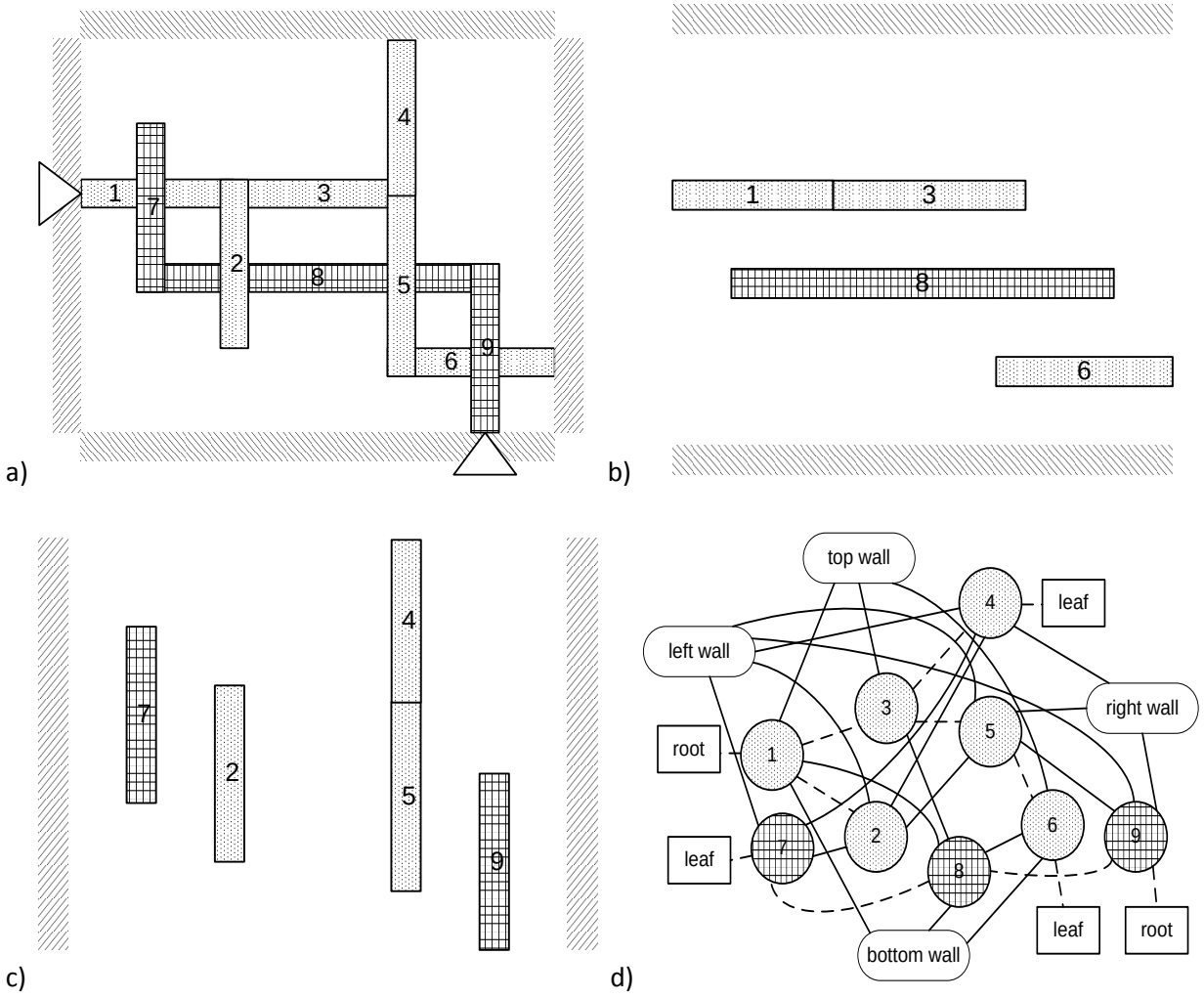

Figure 10. An artificial example of the layout for multi-layer power optimization. A) The full layout including two metal layers, two nets and 9 wire segements. B) and C) Horizontal and vertical metal layers and wires occupying them. D) The multi-layer visibility graph corresponding to layout from A). The visibility relationships are shown by solid edges, the connectivity relationships are shown by dashed edges.

exact metric, it is easy for implementation and its high fidelity property allows using it as a delay metric for optimization algorithm.

The results for different layout clips are presented in Table 2. The numbers representing power are given in relative units. It can be seen that the cross-coupling interconnect power is being reduced by 8% on average, varying from 5% to 12.6%. Notice that after optimization all nets

satisfy delay constraints. Varience of power reduction results can be explained by different density and initial stage in different layout clips.

9. Summary, conclusions and future work

In this paper we presented the problem of power optimization by simultaneous multi-layer wire spacing. Compared to existing spacing algorithms, this technique has some advantages. The spacing of the wires is performed in the way that total dynamic power is decreased, but delay constraints imposed on net receivers are not violated. Moreover, simultaneous multi-layer optimization allows full exploitation of optimization space as compared to existing layer-by-layer or net-by-net optimization techniques. We have shown that the discussed optimization problem is convex, providing the interconnect delay is convex as a function of inter wire spacing. The interior-point algorithm was developed to solve the problem. The technique of layout partitioning was used in order to simplify optimization. We demonstrated the algorithm on real industrial cases and showed 5-12% of inter-wire dynamic power reduction. In addition, the relation between weigthed power-delay optimization and power optimization under delay constraints was developed and method for chosing optimal delay weights was proposed. The presented algorithm can be extended for cases with nets initially violating delay constraints. The extended algorithm then first will repair existing delay violations and then perform power minimization under delay constraints.

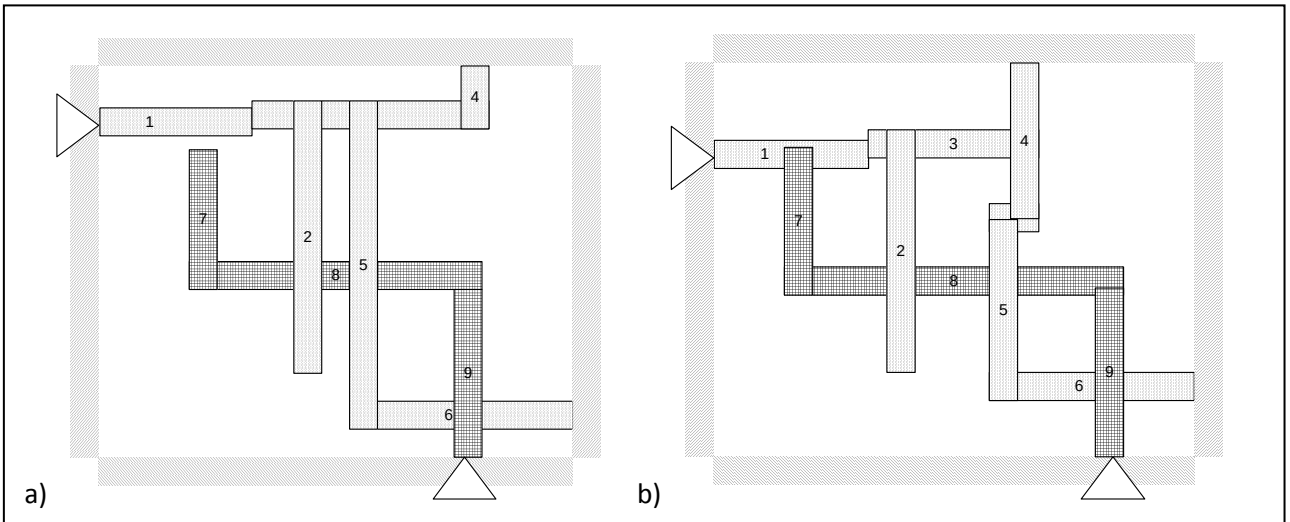

Figure 11. A) Layout after optimization without timing constraints. B) Layout after optimization with timing constraints. In the second case delay constraint prevented wires 1, 3 and 6 from moving too close to the wall as well as wires 2 and 5 from getting too close to each other.

Table II. Optimization results for real industrial layout segments

No. of clip	Initial power	Final power	Impr., %	No. of wires (variables)	No. of spaces (location constraints)	No. of delay constraints
1	863.8504589	817.119679	5.41%	4091	21518	1427
2	2723.372233	2552.8175	6.26%	37177	110962	13860
3	2068.078358	1974.36814	5.67%	14403	51166	2906
4	1685.869617	1550.59565	8.02%	13397	47450	4639
5	3737.549076	3306.77984	11.53%	27639	96031	7003
6	3531.584387	3331.86887	5.66%	25343	89996	7161
7	2058.194188	1799.12777	12.59%	22669	79838	7169
8	3084.118285	2827.55122	8.32%	25537	87810	7331
Total	**19752.6166**	**18160.2287**	**8.18%**	**170256**	**584771**	**51496**

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