Layout Optimizations in VLSI Designs
B. Lu, D.-Z. Du, and S. S. Sapatnekar (Eds.) pp. 155 - 190

Interconnect Modeling and Design With Consideration of Inductance

Lei He
Department of Electric and Computer Engineering
University of Wisconsin, Madison, WI 53706
E-mail: lhe@ece.wisc.edu

Contents

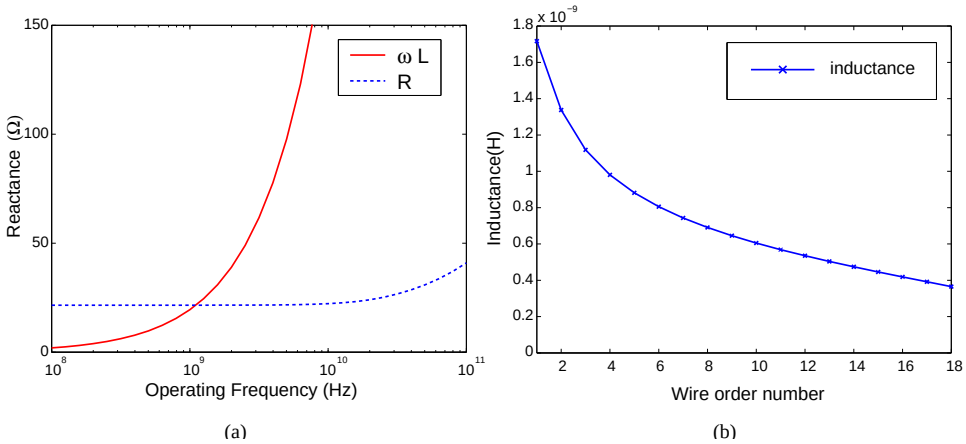


Figure 1: (a)Characteristics of resistance and reactance due to inductance; (b) Characteristics of self inductance and mutual inductance.

1 Introduction

Given the growing importance of interconnects in performance, reliability, cost, and power dissipation for high-performance circuits and systems, interconnect modeling and optimization has been an active research area [1]. However, most existing work on interconnect modeling and optimization assumes an RC interconnect model, which becomes increasingly inadequate as the on-chip inductive effect gains prominence in gigahertz designs. A simple rule of thumb is that the inductance should be considered if resistance R and reactance ωL have similar values, where L is inductance and $\omega = 2\pi f$ with f being the operating frequency. In Figure 1(a), we compare R and ωL under different operating frequencies. We used the three-dimensional electromagnetic field solver FastHenry [2] to compute R and ωL for a typical global interconnect, which is $0.8\mu m$ wide, $2\mu m$ tall, and $2000\mu m$ long. One may easily see that ωL starts to outweigh the resistance at the operating frequency of approximate one gigahertz. As the operating frequency

is larger than the clock frequency due to the harmonic effect,[1] on-chip inductance should be considered in the layout design for circuits of gigahertz clock frequencies.

Furthermore, we compare the self inductance and mutual inductance in Figure 1(b). We designed an eighteen-bit signal bus sandwiched between two coplanar power/ground nets, where all wires are $0.8\mu m$ wide, $2\mu m$ tall, $2000\mu m$ long, and are separated by $0.8\mu m$. We computed the loop inductance for these wires using FastHenry. In Figure 1(b), the leftmost data-point stands for the self inductance of the leftmost signal net, and the rest of the data-points are mutual inductance between the leftmost signal net and the remainder of the signal nets. Clearly, the inductance has a "long-range" effect in the sense that the mutual inductance between non-adjacent nets cannot be ignored when compared to the self inductance. Note that the capacitance is a "short-range" effect in the sense that the mutual capacitance between non-adjacent nets is negligible, so that interconnect modeling and design under the RC model needs to consider only the net under study and its two adjacent nets, or in most cases, to consider just the net under study by assuming the worst-case mutual capacitance to the adjacent nets. This single-net based approach no longer works for the RLC model due to the long-range inductive effect. Hence, interconnect modeling and design under the RLC model should consider multiple coupled nets simultaneously, and is *inherently* more difficult compared to interconnect modeling and design under the RC model.

The remainder of this chapter is organized as follows: In Section 2, we present an efficient approach to compute inductance from the interconnect geometry, and use the resulting inductance model and SPICE simulation to evaluate the impact of layout optimization techniques such as wire sizing and spacing, buffer insertion and shield insertion. In Section 3, we describe the formulation and algorithm for the simultaneous shield insertion and net ordering problem. Discussions about recent progress are included in each section.

[1] The operating frequency is decided by the signal rise time t_r. The knee frequency can be defined as $F_{knee} = 0.5/t_r$ and be used as the operating frequency to compare ωL and R, as "the behavior of a circuit at (operating) frequencies above F_{knee} hardly affects digital performance" [3]. Similar conclusion was drawn in [4] using the concept "significant frequency". More sophisticated rules to judge the significance of inductance can be found in [5, 6, 7].

2 Inductance Extraction

Inductance extraction computes inductance from complex 3-dimensional (3D) interconnect structures. It is considered as a very difficult problem, because the inductance in essence is defined for the current loop but the current return path is not well defined for on-chip interconnects. The key idea to solve the problem is the *partial element equivalent circuit (PEEC)* model. It is developed in [8] and was introduced to the circuit design field in [9, 10]. The inductance in the PEEC model, in short, the *partial inductance* for a wire segment is defined as the portion of the loop inductance for the wire segment, assuming the wire segment forms a loop with the infinity. Numerical field solvers have been developed based on the PEEC model [10, 2, 11].

However, extraction of parasitic parameters (resistance, capacitance and inductance) via numerical field solvers is hard to support during iterative procedures of simulation and optimization for on-chip interconnects. Accurate and efficient extractions of resistance and capacitance without using numerical methods have been achieved recently. Examples include the 2 1/2-D capacitance extraction methodology [12], and the statistically-based worst-case RC model [13]. Both are table-based approaches, and are suitable for iterative simulation and optimization purposes. In the following, we will first present two foundations concerning the inductance extraction. These two foundations are based on the PEEC model, and will lead to an accurate and efficient table-based inductance extraction methodology. We then apply the table-based inductance model to compute the loop inductance for a coplanar waveguide to show the link between the loop inductance and partial inductance model, and apply the inductance model to study the impact of layout optimization.

2.1 Foundations for Inductance Extraction

There are multiple metal layers in a VLSI technology. We assume that wire traces on adjacent layers are orthogonal, and extract the inductance for a block, which contains n parallel traces (T_1, T_2,..., T_n) of same length on the same layer (see Figure 2). In addition, we also assume that the two most outside traces, T_1 and T_n, are dedicated power/ground traces. When the block size is three, it is a coplanar-waveguide, which is one of the three basic forms for transmission line, and is often used for clock tree in high-speed designs. When the block size is large, it models the bus structure

with outside power/ground traces that can be used for shielding only or for shielding and power supply at the same time. Because traces are orthogonal on adjacent layers, traces on layer $N+1$ and layer $N-1$ will not affect the inductance of traces on the current layer N [4].

Figure 2: The cross-section view for a block of n traces, where T_1 and T_n are dedicated power/ground traces. The widths for the traces are W_1, W_2, ..., and W_n, respectively. The spacings between them are S_1, S_2, ... and S_{n-1}, respectively.

Note that the capacitive effect is a "short-range" effect in the sense that for a block, only the mutual capacitance between adjacent traces are important, and the rest of the mutual capacitance can be ignored. Therefore, for any trace, it is sufficient to solve the trace and its two adjacent traces via numerical extraction. In other words, we are able to reduce the n-trace capacitance problem to a number of 3-trace subproblems [12]. The inductive effect, however, is a "long-range" effect. For example, in Figure 3, we compute the inductance for a block with size n=5 by assuming that the wire thickness is $2.0\mu m$, wire width W1=$4\mu m$, W2=W3=W4=$0.8\mu m$, W5=$2\mu m$, all spacings are $0.8\mu m$, and the length is $4000\mu m$. We specify that T_1 and T_5 are power/ground traces that serve as current return paths, and run a 3D inductance tool RI3 in Raphael [11] to compute loop inductance. The result is the loop inductance in a 3x3 matrix, where current is assumed to return via the two power/ground traces T_1 and T_5. In the matrix, diagonal elements are self inductance, and off-diagonal elements are mutual inductance. The mutual inductance between T_2 and T_4 can not be ignored even though there is T_3 between them.

In general, there is a significant mutual inductance between any traces within a block (e.g., even for a block of size n=32). Due to this "long-range" effect, even though we assume that all signal traces have identical widths, and the spacings are identical, the brute-force way to build inductance tables will have large table sizes. The table for self inductance has six dimensions: two widths for power/ground traces, one width for signal traces, the trace location, and uniform spacing and length. Note that the trace length is

Figure 3: Loop inductance (nH) by specifying that T_1 and T_5 are current return paths.

needed because the inductance is not a linear function of trace length. At the same time, the table for mutual inductance needs locations for two traces, which leads to seven-dimension tables. We may not afford to consider different widths and spacings for different traces.

Furthermore, the loop inductance in Figure 3 assumes that all current returns via the two power/ground traces, which may not be true for high frequency. For example, when only one trace is switching, its current may return from adjacent quiet traces. The right way to extract inductance for a block is to run RI3 [11] without specifying any power/ground traces as current return paths. Then, for the block in Figure 3, we obtain the partial inductance in a 5x5 matrix (see Figure 4(a)). Again, the diagonal elements are self inductance, and off-diagonal elements are mutual inductance. An important observation is that now the self inductance of a trace depends only on the trace itself, and the mutual inductance of two traces depends only on the two traces themselves. For example, in Figure 4(b), we compute the self inductance L_{11} for T_1 with other traces removed, and obtain the same L_{11} as in Figure 4(a). In Figure 4(c), we compute the mutual inductance L_{15} for T_1 and T_5 with T_2, T_3 and T_4 removed, and obtain the same L_{15} as in Figure 4(a).

When we do not specify which traces are power/ground traces, we compute partial inductance (denoted as L_p) under the PEEC model. In general, we have the following foundations:

Foundation 2.1 *Self L_p of a trace is* solely *decided by the trace (its length, width and thickness).*

Foundation 2.2 *Mutual L_p of two traces is* solely *decided by the two traces (their lengths, widths and thicknesses, and the spacing between them).*

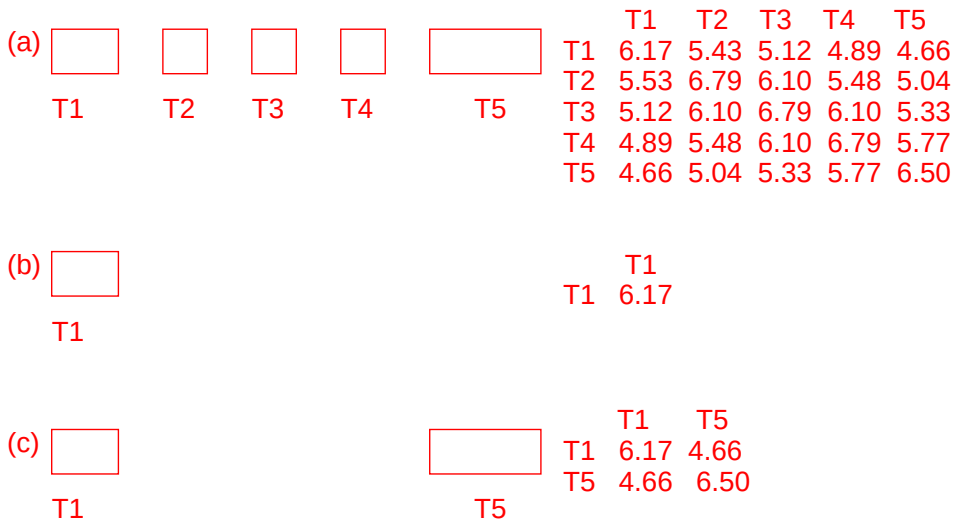


Figure 4: Partial inductance (nH) without specifying any current return path: (a) a block of size $n = 5$, (b) only trace T_1 (with other traces removed), and (c) only two traces T_1 and T_5 (with traces T_2-T_4 removed).

2.2 Validation of Foundations

In order to validate the two foundations, the following illustrates the inductance extraction procedure under the PEEC model. The PEEC model was introduced in [9, 10], and has been widely used in numerical inductance extraction tools (for example, [11, 2]). Because the inductance is defined only for closed loops, the partial inductance of a trace can be viewed as the inductance of the trace as it forms a loop with infinity. If the current density is uniform in traces T_k and T_m, according to [10], the mutual inductance $L_{p_{km}}$ under the PEEC model is:

$$L_{p_{km}} = \frac{\mu}{4\pi}\frac{1}{a_k a_m}\int_{b_k}^{c_k}\int_{a_k}\int_{b_m}^{c_m}\int_{a_m}\frac{dl_k \cdot dl_m}{r_{km}} da_k \cdot da_m \qquad (1)$$

where a_k and a_m are cross-sectional areas, b_k and b_m are starting points, c_k and c_m are ending points, all for traces T_k and T_m, respectively. In addition, r_{km} is the distance between dl_k and dl_m, which represents differential

elements of length for traces T_k and T_m, respectively. When $k = m$, Eqn. (1) gives the self L_p of a trace.

In the case where the current is not uniform in a trace, the trace can be divided into rectangular filaments (see Figure 5). The current is assumed to flow along the length of each filament with a constant density within each filament. Therefore, Eqn. (1) may be used for each filament. It is easy to see that Foundations 2.1 and 2.2 hold for each filament with respect to Eqn. (1). I.e., the self L_p of a filament is *solely* decided by the filament, and the mutual L_p between two filaments is *solely* decided by the two filaments. The conclusions hold for cases of a single trace and multiple traces.

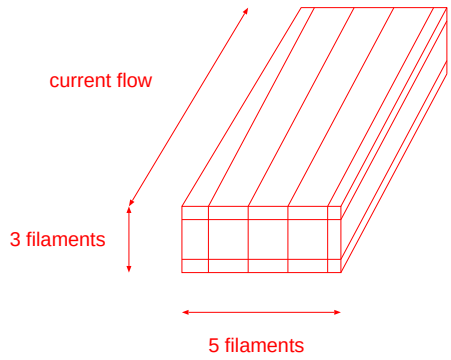


Figure 5: A trace is divided into 3×5 filaments. It is assumed that the current density is the same within a filament.

If we assume that trace T_k has P filaments, and trace T_m Q filaments, then $L_{p_{km}}$ is given by

$$L_{p_{km}} = \sum_{i=1}^{P} \sum_{j=1}^{Q} L_{p_{ij}} \tag{2}$$

where $L_{p_{ij}}$ is the mutual L_p between filament i of T_k and filament j of T_m. Again, when $k = m$, Eqn. (2) computes the self L_p for a trace. Because Foundations 2.1 and 2.2 hold for $L_{p_{ij}}$, it is easy to see that Foundations 2.1 and 2.2 still hold after using Eqn. (2) to compute L_p for traces.

2.3 Table-based Inductance Extraction

The two foundations enable us to reduce the n trace inductance problem into 1-trace subproblems to solve the self L_p, and into 2-trace subproblems to solve the mutual L_p. There is no loss of accuracy during the reduction. As given in [14], the self inductance may be solved by

$$L(nH) = 2l[\ln \frac{2l}{w+t} + 0.5 - k] \quad (3)$$

where $k = f(w,t)$ and $0 < k < 0.0025$, and l, w, and t are length, width and thickness of the trace in unit of cm. The mutual inductance for two traces of same width and length is

$$L(nH) = \frac{\mu_0 l}{2\pi}[\ln \frac{2l}{s} - 1 + \frac{s}{l}] \quad (4)$$

where s is the spacing between two traces, again in unit of cm.

length	$1000\mu m$		
width(μm)	1	1.2	2
Self L_p(nH)	1.480(0.0%)	1.461(-1.2%)	1.400(-5.4%)
Mutual L_p(nH)	1.101(0.0%)	1.100(-0.1%)	1.096(-0.5%)
length	$4000\mu m$		
width(μm)	1	1.2	2
Self L_p(nH)	7.028(0.0%)	6.951(-1.1%)	6.709(-4.5%)
Mutual L_p(nH)	5.551(0.0%)	5.508(-0.1%)	5.490(-1.1%)

Table 1: On-chip self and mutual inductance computed via numerical extraction for two parallel wire traces with pitch-spacing being $3\mu m$, and the thickness being $1\mu m$. For each length, three wire widths are assumed, namely, $1\mu m$, $1.2\mu m$ (i.e., the width has 20% variation compared to width=$1\mu m$), and $2\mu m$ (i.e., the wire is up-sized by 100% compared to width=$1\mu m$). Percentages of inductance changes with respect to inductance for width=$1\mu m$ are also given.

These equations give us two insights: First, the inductance for on-chip interconnects is not linearly scalable. Both self and mutual inductance are super-linear functions of the trace length. Secondly, because of the logarithmic operation of $\frac{2l}{w+t}$ and $\frac{l}{s}$, both mutual and self inductance is less sensitive

to variations of trace width and spacing as the capacitance and resistance are. The two insights are also verified by experiments with numerical inductance tools. For example, in Table 1, we report both self and mutual inductance for two parallel wire traces with pitch-spacing being $3\mu m$. For each length, three wire widths are assumed, namely, $1\mu m$, $1.2\mu m$ (i.e., the width has 20% variation compared to width=$1\mu m$), and $2\mu m$ (i.e., the wire is up-sized by 100% compared to width=$1\mu m$). One can see the following from the table: First, the inductance is not linearly scalable with respect to the wire length. For example, when the wire width is $1\mu m$ and the wire length increases from $1000\mu m$ to $4000\mu m$ (an increasing factor of 4x), the self inductance increases by a factor of 4.74x rather than 4x. Second, the inductance variation is only around 1% for the 20% variation of wire width. Therefore, there is no need to consider the impact of process variation for inductance extraction, even though the impact must be considered for resistance and capacitance extractions as in [13]. Finally, when we up-size the wire width by 100%, the inductance changes by up to 5.4% for self inductance, and by 4.5% for mutual inductance. Therefore, interconnect sizing and spacing might *not* be an effective way to change inductance[2].

There are limitations of applying the two equations however. First, they do not consider the skin depth and internal inductance; Second, widths are not considered for mutual inductance. Therefore, we will propose to build tables via numerical inductance extraction for self and mutual inductance.

There are two parts in the table-based inductance extraction. One is to pre-compute inductance tables. We assume that each layer has a nominal thickness, and build tables for different layers. The self inductance table has two dimensions: width and length. The mutual inductance table has four dimensions: widths for two traces, the spacing between them, and the length. The 3D inductance extraction tool RI3 [11] is invoked to solve a block of two traces for different combinations of lengths, widths, and spacings. The resulting self and mutual inductance is stored in tables. Note that only 2-trace subproblems need to be solved, because results to 1-trace subproblems are parts of results to 2-trace subproblems. In addition, the inductance depends on the skin depth, which is a function of frequency. We run RI3 [11] under the significant frequency. The significant frequency is defined as $0.17/t_r$, where t_r is the minimum rising/falling time [4]. It captures the

[2] On the other hand, because the coupling capacitance depends strongly on the spacing, the optimal interconnect sizing and spacing is effective to reduce RC delay (as well as capacitive coupling).

majority of high-frequency components for the signal with rising/falling time t_r.

The other part of the table-based inductance extraction is table lookup. For each trace in a block, we obtain a self inductance from tables for a given layer, length and width. For any combination of two traces T_i and T_j, we obtain a mutual inductance from tables for a given layer, widths, and spacing between T_i and T_j. A bicubic spline algorithm [15] will be used to compute inductance that is not given in the table.

2.4 Applications of Efficient Inductance Model

2.4.1 L_{eff} for Coplanar-waveguide

The coplanar-waveguide structure (a block of size $n = 3$, see Figure 6) is often used for on-chip clock trees in high-speed designs. Not to consider the inductive effect will lead to a significant underestimate of delay and noise. Therefore, the effective loop inductance (L_{eff}) of the signal trace needs to be computed in order to use the transmission line theory. In the following, we derive L_{eff} as a function of L_p for the three traces T_1, T_2 and T_3, where T_2 is the signal trace, and T_1 and T_3 are coplanar power/ground traces.

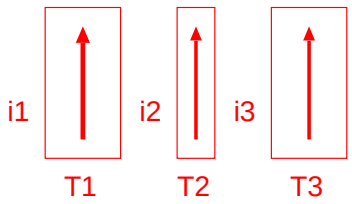


Figure 6: The top view of a coplanar-waveguide. T_1 and T_3 are dedicated power/ground traces.

L_{eff} is defined for the current loop that has two segments: the first segment is current i_2 through T_2; the second segment has two parallel branches, i.e., i_1 through T_1 and i_3 through T_3. According to the definition of L_{eff}, we have

$$\begin{aligned} \Delta V &= L_{eff}\frac{di_2}{dt} \\ &= L_{p_{22}}\frac{di_2}{dt} + L_{p_{21}}\frac{di_1}{dt} + L_{p_{23}}\frac{di_3}{dt} - L_{p_{11}}\frac{di_1}{dt} - L_{p_{12}}\frac{di_2}{dt} - L_{p_{13}}\frac{di_3}{dt} \end{aligned} \tag{5}$$

and the two power/ground traces have the same voltage drop

$$L_{p_{11}}\frac{di_1}{dt}+L_{p_{12}}\frac{di_2}{dt}+L_{p_{13}}\frac{di_3}{dt} \;=\; L_{p_{13}}\frac{di_1}{dt}+L_{p_{23}}\frac{di_2}{dt}+L_{p_{33}}\frac{di_3}{dt} \qquad (6)$$

finally, the current is conservative according to KCL

$$i_1+i_2+i_3 \;=\; 0 \qquad (7)$$

L_{eff} can be derived by simultaneously solving (5)-(7). When T_1 and T_3 are symmetric with respect to T_2, L_{eff} is

$$L_{eff} \;=\; L_{p_{22}}-2L_{p_{23}}+\frac{L_{p_{11}}}{2}+\frac{L_{p_{13}}}{2} \qquad (8)$$

Experiments show that (8) using our partial inductance tables gives results almost identical to L_{eff} obtained by 3D extractions using RI3 [11]. For example, we compute the L_{eff} for a coplanar-waveguide structure with all three traces $10\mu m$ wide, $2000\mu m$ long, and separated by $2\mu m$. The L_{eff} given by our formula (8) using table-based partial inductance is 0.839nH, while L_{eff} given by RI3 is 0.840nH. The difference can be almost ignored.

2.4.2 Impact of Layout Optimization

We have shown that wire sizing and spacing are not effective to reduce the inductance. However, as wire sizing and spacing are effective to change the coupling capacitance, wire sizing and spacing are still effective to reduce the coupling noise between adjacent wires.

The above inductance model can be used to generate the RLC circuit model. In the following, we apply the resulting RLC model to study the impact of buffer insertion and shield insertion. We use a bus structure as an example. It has 18 signal traces, and two fat power-traces outside the signal traces. The wire thickness is $2.0\mu m$, and spacing is $0.8\mu m$, both for all traces. The width is $0.8\mu m$ for all signal traces, and is $16\mu m$ for two fat power-traces. We assume the following signal pattern: all signal traces are simultaneously switching up with rising time of 80ps, except that one of the two central signal traces is the quiet victim. We also assume that all devices, including drivers, buffers and receivers, are 40x of the minimum inverter in a representative $0.18\mu m$ CMOS technology. Based on SPICE simulations, we will show how buffer insertion and shielding insertion reduce the noise under the RLC model.

A. Buffer insertion

It is well known that buffer insertion is effective to reduce the RC delay and capacitive noise. It is also very effective to reduce the inductive noise, since the original longer current return loop becomes shorter, as the current returns through the inserted buffers. Furthermore, as we discussed in the above (see Table 1), the inductance is a super-linear function of the wire length, more precisely, the DC-connected wire length between devices such as drivers, buffers and receivers. Therefore, while inserting a buffer at the middle point of a long wire reduces the coupling capacitance by a half for each DC-connected wire, it reduces the mutual inductance by more than a half for each DC-connected wire.

In the following example via SPICE simulation under the RLC model, we assume that for all traces, the wire length is $4000\mu m$. We measure the noise at the far-end of the victim trace (the input node of receiver) for three cases: no buffer, one buffer, and three buffers inserted for each single trace, respectively. These buffers are inserted uniformly. Therefore, the DC-connected wire lengths are $4000\mu m$, $2000\mu m$ and $1000\mu m$ for three cases, respectively. As shown in Table 2, compared to the case of no buffer insertion, inserting one buffer reduces the noise by 21.1%, and inserting three buffers reduces the noise by 42.3%. Note that the noise is measured at the inputs of devices for the victim trace. Much less of noise will be observed if the measurement is made at the output of devices.

number of buffer inserted	0	1	3
wire length (μm)	4000	2000	1000
Noise (V)	0.71(0.0%)	0.56(-21.1%)	0.41(-42.3%)

Table 2: Comparison of noise between different buffer insertion solutions.

B. Shield insertion

A shielding trace is a wire directly (without through devices) connected to power or ground networks.[3] It can provide the dedicated current return

[3]The most convenient way to connect random shields is to add vias between shields and P/G wires in orthogonal routing layers. In this chapter, we assume that there is a connection between the P/G structure and every $200\mu m$-long shield.

path to reduce the inductive noise. In the following experiment, we assume that for all traces, the length is $4000\mu m$, and no buffers are inserted. We again measure the noise at the far-end of the victim trace (the input node of receiver) via SPICE simulation. We will insert shielding traces to make the far-end noise of the victim trace less than 0.25V.

When there is no shielding traces, the noise of victim trace is 0.71V. Then, we insert a shielding trace for every six signal traces, and increase the width W_s for shielding traces from $0.8\mu m$ to $2.4\mu m$. The noise is 0.22V with $W_s = 2.4\mu m$. Finally, we insert a shielding trace for every three traces. The noise is 0.17V when $W_s = 0.8\mu m$. As shown in Table 3, there is a clear trade-off between area and noise: we may reduce the noise by a factor of 4.2x while the total width, the sum of the total wire width and total spacing, of the bus structure is increased by 13%, and the total wire width is increased by 8.8%.

N_s	W_s	Noise (V)	total width (μm)	wire width (μm)
18	–	0.71	61.6	46.4
6	0.8	0.38	64.8	46.0
6	1.6	0.27	64.4	49.6
6	2.4	0.22	68.0	51.2
3	0.8	0.17	69.6	50.4

Table 3: Comparison of noise between different shielding insertion solutions. Column one (N_s) is the number of signal traces between two shielding traces, and column 2 (W_s) is the width for the shielding traces. Column 3 is the total width (including the total spacing) of the the bus structure, and column 4 is the total wire width.

2.5 Conclusions and Discussions

In this section, we have present an efficient table-based inductance model for parallel coplanar wires. Several papers have extended this model. Our table-based inductance model is not applicable to strip lines and micro striplines with power/ground planes. In [16], two new foundations have been developed to compute the loop inductance for strip lines and micro striplines, in a fashion similar to Foundations 2.1 and 2.2. Our table-based inductance model is not applicable to random wires either. Different from parallel wires

that have the same lengths, and are aligned and routed in the same metal layer, random wires may have different lengths, and may routed in the different layers. In [17, 18], formulae have been developed to compute the inductance for random wires based on the table-based inductance model for parallel wires.

Further, we have studied the impact of layout optimization based on SPICE simulation using the RLC circuit model. We show that both buffer insertion and shield insertion are effective to reduce noise. Related studies and tutorials can be found in [19, 4, 20, 21]. In the next section, we will discuss algorithms to leverage the shield insertion with simultaneous net ordering.

3 Simultaneous Shield Insertion and Net Ordering

Interconnect synthesis has been extensively studied for RC interconnect models, including routing topology generation, wire sizing, wire spacing, device sizing, buffer insertion, net ordering, and combinations of these design components. A comprehensive survey of these work can be found in [1].

There has been some preliminary work using the RLC interconnect models as well, including routing topology generation [22], wire sizing [23] and buffer insertion [24]. However, all of them assume a single RLC net, which is not adequate as discussed in Section 1. Below, we present the formulation and algorithm for simultaneous shield insertion and net ordering (SINO) problem. It is one of the earliest work that studies the interconnect synthesis for multiple RLC nets.

3.1 Preliminaries

A. Coplanar Interconnect Structures

Again, we consider only parallel coplanar interconnect structures with all wires having the same lengths. These are characterized as a number of signal traces and power/ground traces which run parallel in the same layer. We give an example of this structure in Figure 7. In the figure, P and G represent the power and ground grids (P/G grids), s represents signal wires (denoted as s-wires), and g is a shield wire connected to P/G grids. Both P/G grids and shield wires provide dedicated current return paths

for signals, and are denoted as g-wires in this chapter. We use the terms, "trace", "wire" and "net" interchangeably.

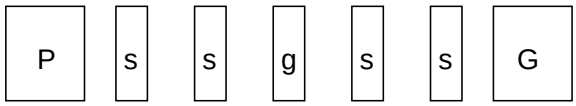


Figure 7: A cross-section view of a coplanar interconnect structure with a shield inserted.

A coplanar interconnect structure can be represented by a string, where each symbol stands for an s-wire or a g-wire. For example, the interconnect structure in Figure 7 can be represented by $gssgssg$ if we do not distinguish these s-wires (or alternatively, g2sg2sg). If we label the s-wires from left to right as s_1, s_2, s_3 and s_4, then the string $gs_1s_2gs_3s_4g$ is a unique representation of a net ordering and shield insertion solution (referred to as a SINO solution or a SINO string). In this chapter, a SINO string implicitly includes a shield trace (the power and ground grids) as its first and last element. These P/G grids are shield resources, but are not considered in solution size computations as they are present generally, with or without noise considerations. The "size" of a SINO solution can be determined directly from the length of the SINO string. As an example, consider the following: $< g > s_1s_3gs_2s_4s_5gs_7s_6s_0 < g >$. This string represents an eight (signal) bit interconnect structure with two g-wires (plus two implicit g-wires for the P/G grids denoted as $< g >$). Note that we can apply our formulations and algorithms to be presented to any group of wires which may contain pre-routed g-wires more than just a pair of P/G grids. We call the group of wires sandwiched between adjacent g-wires a block, and the number of s-wires in a block as the block size. A block can be represented as a substring of a SINO string (i.e. block 0 of the above string would be written as s_1s_3). As in the original SINO string, the g-wires on each end are implicit with the substring.

B. Characteristics and Model of Inductive Coupling

We illustrate the characteristics of inductive coupling in Figure 8, where we show the mutual inductance from the leftmost s-wire to all other s-wires. Cases (a) and (b) in the figure show two interconnect structures, $g18sg$ and

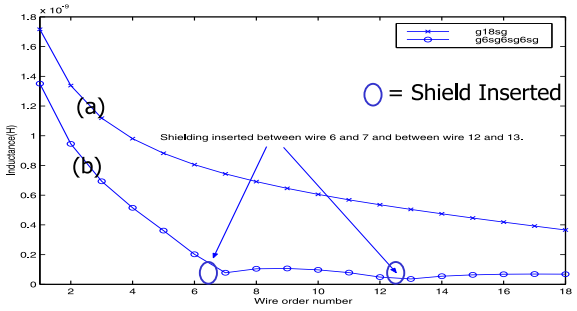


Figure 8: Illustration of mutual inductive coupling from a signal wire (s_1) to other signal traces for two coplanar structures. Structures (a) and (b) show coupling from s_1 to all other s-wires as a function of wire-order number for the g18sg and g6sg6sg6sg structures, respectively.

$g6sg6sg6sg$, respectively. As shown by (a) where there is no shield between s-wires, the mutual inductance decreases slowly from left to right. This implies that net ordering is not effective to reduce inductive coupling. In comparing (a) with (b) in the figure, the mutual inductance between wires separated by shields becomes much smaller compared with coupling to wires within the same block. Therefore, shields are effective to reduce inductive coupling.

In order to efficiently model inductive coupling between two arbitrary s-wires, we use the coupling coefficient between two s-wires as the figure of merit. The coupling coefficient between two nets s_i and s_j is defined as

$$K_{ij} = \frac{m_{ij}}{\sqrt{l_i \cdot l_j}}$$

where m_{ij} is the mutual inductance between s_i and s_j, and l_i and l_j is self inductance for s_i and s_j, respectively. We use an "effective coupling model", i.e. K_{eff} model, in this chapter in order to efficiently compute the coupling coefficient. In this model, when nets i and j are in different blocks, the coupling coefficient is $K_{ij} = 0$.[4] When the two nets are in the same block, as shown in Figure 9 where N_i and N_j are track ordering numbers for the two s-wires, and g_l and g_r are track ordering numbers for the two edge

[4] This assumption may occasionally lead to under-estimating inductive coupling, which will be discussed in Section 3.5.

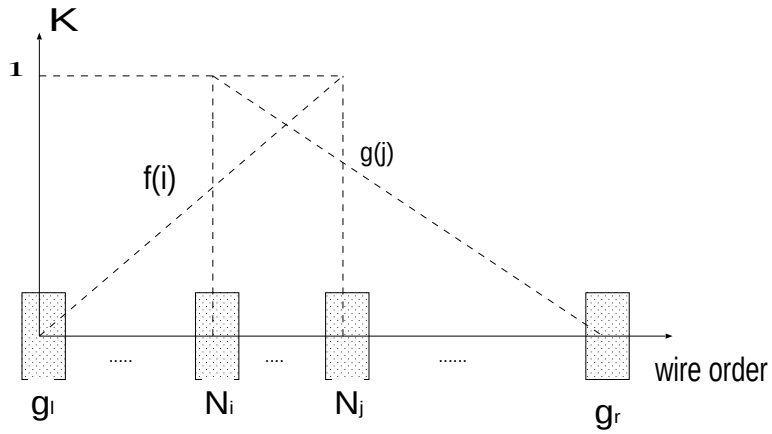


Figure 9: Illustration of K_{eff} computation. N_i and N_j are two signal wires in the same block sandwiched by ground wires g_l and g_r. $f(i)$ and $g(j)$ are two linear functions of the wire order number i, j as shown by the sloping dotted line. The mutual inductance coupling is given by the mean of $f(i)$ and $g(j)$.

g-wires, the coupling coefficient is computed as

$$K_{i,j} = \frac{(f(i) + g(j))}{2}$$

where $f(i)$ and $g(j)$ are two functions defined as follows: We assume that the function f is 0 at g_l, and is 1 at N_j; therefore $f(i)$ is the linear interpolation, i.e., $f(i) = \frac{(N_i - g_l)}{(N_j - g_l)}$. Similarly, we can compute $g(j) = \frac{(g_r - N_j)}{(g_r - N_i)}$.

The K_{eff} model is independent of the width, length and spacing of s-wires and g-wires. As an illustration in Figure 10, we compare the coupling coefficients given by the K_{eff} model and the three-dimensional field solver FastHenry [2]. In the figure, the x-axis indicates the coupling coefficient given by FastHenry and the y-axis indicates the coupling coefficient given by the k_{eff} model. We use coplanar interconnect structures $g(6sg)^2 6sg$ (i.e., $g6sg6sg6sg$) and $g(3sg)^5 3sg$. The wire width is $0.8\mu m$, length is $1000\mu m$, thickness is $2\mu m$ and the spacing between wires is $0.8\mu m$. The three dashed lines in Figure 10 indicate differences of 0%, +15% and −15%, respectively. One can easily see that most of the data points lie within the ±15% error range.

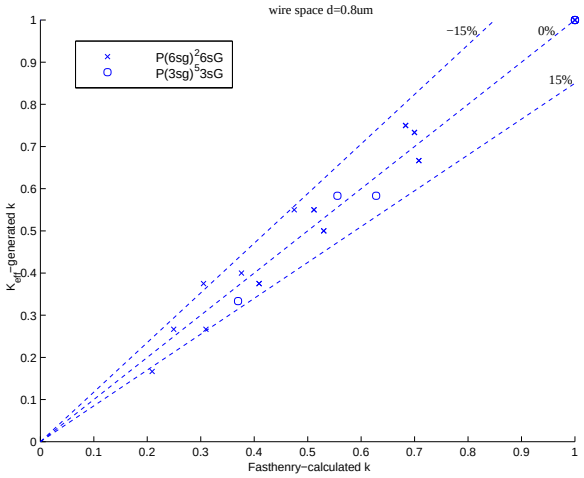


Figure 10: Accuracy of K_{eff} model. The x-axis is the mutual inductance coefficient as computed by FastHenry. The y-axis is the K_i value calculated by the K_{eff} model. Error lines for +15%, 0%, and -15% difference between the models indicate the quality of the K_{eff} model.

C. Net Sensitivity

We define two nets s_1 and s_2 to be sensitive to each other if a switching signal on s_1 will cause s_2 to malfunction (due to extraordinary crosstalk or delay variation) or vice-versa. Net sensitivity is depicted graphically in Figure 11.

The sensitivity for all s-wires in a given problem can be represented compactly with a sensitivity matrix S of size $n \times n$ (where n is the number of s-wires) as shown in Figure 12. The graphical representation of the sensitivity matrix (essentially an undirected graph structure), is also shown in Figure 12. An entry of 1,0 in location (i, j) indicates that s_i and s_j are sensitive or not sensitive, respectively, to one another. By definition, the matrix must be symmetric since the underlying graph is undirected (i.e. $S_{ij} = S_{ji}$). For all formulations, we assume that an appropriate sensitivity matrix indicating design parameters and net relationship semantics is given a priori.

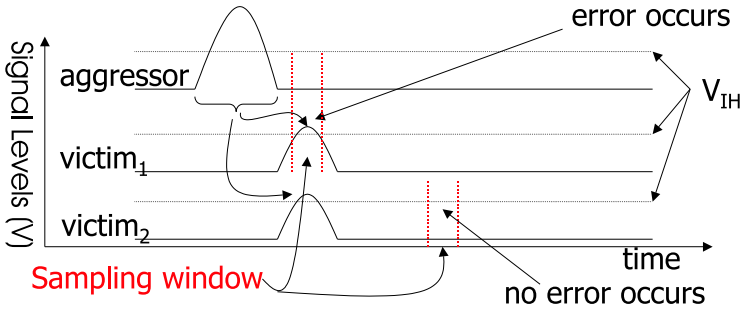


Figure 11: Illustration of net sensitivity. The y-axis indicates signal (voltage) level and the x-axis indicates time. The switching event on the aggressor induces a noise voltage in the two victims as shown. For $victim_1$, the noise pulse occurs during a sampling window–hence the aggressor and $victim_1$ are sensitive. For $victim_2$, the noise pulse does not occur during a sampling window–hence the aggressor and $victim_2$ are not sensitive.

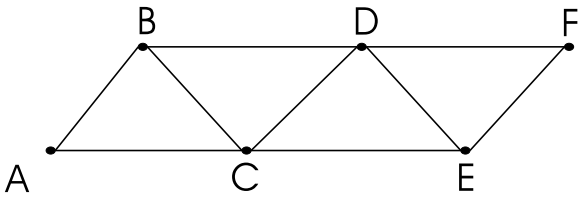
3.2 Formulation of Optimal SINO Problems

We say that an s-wire s_i is capacitive noise free if it is not directly adjacent to any other s-wire s_j that is sensitive to it. Similarly, we say that s_i is inductive noise free if it does not share a block with any other sensitive wire s_j. We say a placement P (or equivalently, a SINO solution, or a SINO string) is noise free if, and only if, all nets s_i within P are free of both capacitive noise and inductive noise. With respect to these concepts, we define the following SINO problem to eliminate Cx and Lx noise and call it the noise free SINO problem (SINO/NF).

Formulation 3.1 *(Optimal SINO/NF problem) For a given placement P, find a new placement P' by simultaneous shield insertion and net re-ordering such that P' is noise free and the total area of P' is minimal.*

In general, the SINO/NF problem is over-constrained and may lead to over-designed solutions as shown in Section 3.5. To address more realistic design constraints, we define the following SINO problem to meet a given noise bound and call it the noise bounded SINO problem (SINO/NB).

Formulation 3.2 *(Optimal SINO/NB problem) For a given placement P, find a placement P' with the minimum area by simultaneous shield insertion*



	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	1	1	0	0
C	1	1	0	1	1	0
D	0	1	1	0	1	1
E	0	0	1	1	0	1
F	0	0	0	1	1	0

Figure 12: Illustration of a sensitivity graph and the corresponding sensitivity matrix. The nodes in the graph correspond to signal nets, with an arc between the nodes indicating that two nets are sensitive to one-another. The sensitivity matrix is an equivalent representation for the sensitivity graph shown.

and net re-ordering such that any s_i in P' is free of capacitive noise and its inductive coupling to all sensitive wires s_j is less than a given value.

In this chapter, we use the K_{eff} model as the figure of merit for inductive coupling. If s-wire s_i is a net sensitive to s-wire s_j, we may compute the total amount of inductive coupling for s_i as:

$$K_i = \sum_{j \neq i} S_{ij} \cdot K_{ij}$$

Therefore, we solve the following optimal SINO/NB-K_{eff} problem: For a given placement P, find a new placement P' by simultaneous shield insertion and net re-ordering such that P' is free of capacitive coupling and the inductive coupling K_i satisfies $K_i \leq k_i$ for all K_i where k_i is given a priori and is a measure of inductive noise that can be tolerated in an s-wire to maintain correct operation. Throughout the rest of this chapter, we use a uniform value of k_i denoted as K_{thresh} (a noise threshold for K). However, our formulation, algorithms, and implementation are all able to handle non-uniform k_{thresh}. Note that we only consider insertion of minimum width

shields because it has already been shown that, in general, using additional shields is more effective than increasing the shield wire width.

We attempt to solve both the SINO/NF problem and SINO/NB-K_{eff} problem in Section 3.3. For simplicity of presentation, we use SINO/NB as shorthand for SINO/NB-K_{eff} throughout the remainder of the chapter. Note that our formulation and algorithms to be presented are applicable to inductive noise models more accurate than the K_{eff} model.

3.3 SINO Properties

Theorem 3.3 *The Optimal SINO/NF problem is NP-hard.*

Theorem 3.4 *The optimal SINO/NB problem is NP-hard.*

Given that the SINO/NF and SINO/NB problems are NP-hard, we will focus on developing heuristic/approximate algorithms to solve the problems with satisfactory results. Before we present these algorithms, we first introduce the following lower bound for the optimal SINO/NF solution:

Theorem 3.5 *The maximum clique size in the sensitivity graph is a lower bound of the number of blocks required in all optimal SINO/NF solutions.*

The theorem follows directly from the formulation of the sensitivity graph and the definition of the maximum clique in a graph [25]. To illustrate this conclusion and further show that the maximum clique size is not an upper bound for the total number of blocks in optimal SINO/NF solutions, we present two examples in Figure 13. For the sensitivity graph in Figure 13(a), it is easy to verify that the graph has a maximum clique size of two. An optimal SINO/NF solution is $ABgCDE$, and there are two blocks in the solution. The sensitivity graph in Figure 13(b) is nearly the same as in Figure 13(a), except that an edge is added between C and D, i.e. nets C and D are sensitive to each other. The reader may easily verify that indeed this graph still only has a maximum clique size of two, but it is not possible to find a SINO/NF solution for this graph with two blocks. One optimal solution, consisting of three blocks, is $AEgCDgB$. Therefore, the maximum clique size is not a strict upper bound of the number of shields required in an optimal SINO/NF solution. Comparisons between the lower bound and the number of shield wires used will be presented in Section 3.5 to illustrate the quality of our approximation algorithms.

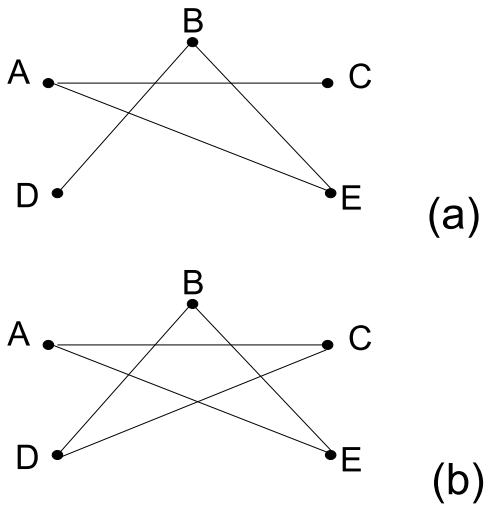


Figure 13: Sensitivity graphs: (a) Both the maximum clique size and the number of blocks are two; (b) The maximum clique size is two, but the minimum number of required blocks is three.

3.4 SINO Algorithms

We develop four approximate algorithms for solving the SINO/NB problem: greedy-based shield insertion (SI) algorithm, net ordering for minimizing Cx noise followed by SI algorithm (NO+SI algorithm), graph-coloring based SINO algorithm (SINO/GC algorithm), and simulated-annealing based SINO algorithm (SINO/SA algorithm). We solve the SINO/NF problem by using the SINO algorithms and setting the noise bound to zero. We will compare different problem formulations and algorithms in Section 3.5. We consider Lx and Cx noise explicitly in all algorithms. In order to more succinctly describe the algorithms and not clutter them with trivial details, we also define the following operations and quantities:

- $Insert\_Shield(a)$: Given a placement P of size m, move all wires (s-wires and g-wires) at locations $a, a+1, ..., m-1$ to locations $a+1, a+2, ..., m$ in the placement, creating a new placement P'. Then, insert a g-wire at location a in P'. Finally set $P = P'$.

- $Compute\_Coupling(s_i)$: Given a placement P of size m, for each $s_k \neq$

s_i in the current block in P, if s_i is sensitive to s_k compute the K_{eff} between s_i and s_k. Sum the K_{eff} over all s_k.

- $Compute\_Block\_Coupling(s_i)$: For each s_i in the current block, find the maximal K_{eff} computed by $Compute\_Coupling(s_i)$ and return it (the maximal K_i for any s-wire within the block).

- $Max\_Clique(S)$: Compute the maximum clique in sensitivity graph S.

- $Compute\_Placement\_Cost()$: Compute the cost for a placement. Details of this are given in Section 3.4.

With these definitions in place, we can succinctly describe our SINO/NB algorithms. The SI and SINO/GC algorithms can be described easily and intuitively. SINO/SA is slightly more complicated, hence we assume that the reader is familiar with SA (for a discussion of SA in other VLSI design contexts, see [26], [27]).

A. Greedy Based Shield Insertion Algorithm

In Figure 14, we present the greedy-based shield insertion (SI) algorithm. The essence of the algorithm is the following: Run through the given placement P. If we encounter two adjacent sensitive nets, insert a shield wire between them. Also, at each location in the placement, calculate the maximum value of K_i that would exist in the current block if we allowed net s_i to become a member. If K_i is greater than K_{thresh}, then create a new block.

```
SI Algorithm: Given a placement P
for each s-wire s_i in P at location a:
        if s_j at location P(a − 1) is sensitive to s_i
                Insert_Shield(a)
        BC = Compute_Block_Coupling(s_i)
        if (BC > K_thresh)
                Insert_Shield(a)
                P(a + 1) = s_i
endfor
```

Figure 14: Greedy shield insertion algorithm (SI)

Because one shield wire is needed for every pair of adjacent sensitive wires, the solution given by the SI algorithm depends on the initial placement. Obviously, the number of shield wires can be reduced by first running existing net ordering algorithms to re-order nets so that no sensitive nets are adjacent to each other, then invoking the SI algorithm. This leads to the NO+SI algorithm.

B. Graph-Coloring Based SINO Algorithm

```
Graph Coloring (GC) Algorithm:
Given an initial sensitivity graph S of signal nets:
MC = Max_Clique(S)
P = new placement with MC blocks
for each s-wire s_i
      for each block b in P (sorted from largest to smallest)
            if Compute_Block_Coupling(s_i) == 0
                  Insert s_i into b
                  next s_i
      endfor
      for each block b in P
            if Compute_Block_Coupling(s_i) < K_thresh
                  Insert s_i into b (adjacent to non-sensitive nets)
                  next s_i
      endfor
      Insert_Shield(end_of_placement)
      Insert s_i into the new block
endfor
```

Figure 15: Graph-Coloring SINO Algorithm (SINO/GC)

In Figure 15, we present the graph-coloring based SINO (SINO/GC) algorithm. This algorithm attempts to approximate a solution to the weighted graph coloring problem by using the maximum clique as a starting point. It works in the following way: First, determine the maximum clique in the sensitivity graph S. We have shown that this is a lower bound for the number of shields required in the SINO/NF problem. Let m be the maximum clique size. We first create a placement with m blocks. We then take each net s_i and try to place it into a block with no other sensitive wires. If we cannot do this, we insert s_i into a block where inserting s_i will cause the

least amount of noise to be added but without causing a noise violation. If we cannot place s_i without creating a K_{thresh} violation, simply create a new block and place s_i into it.

Note that the SINO/NF problem can be mapped into the graph-coloring problem as outlined in the proof of Theorem 3.5, and then be solved using the graph coloring algorithms given in [28], [29].

B. Simulated Annealing Based SINO Algorithm

In Figure 16, we present the simulated-annealing based SINO (SINO/SA) algorithm. We give the details of our SINO/SA algorithm in the following subsections:

Cost Function $Compute\_Cost(P)$ computes the cost for a placement P. The cost is the weighted sum of the following components: (i) *Cap_violation*: total number of nets that are adjacent to their sensitive nets in P; (ii) *Area*: total number of g-wires present in P; (iii) *Ind_Noise*: total number of $K_i > K_{thresh}$ violations in P; and (iv) *Inductance Violation Figure*. It is computed for a placement as shown in Figure 17. The purpose of the Inductance Violation Figure is to penalize a placement for the magnitude of K_{thresh} violations. Its usage (as opposed to simply forbidding placements P' that have K_{thresh} violations) allows the algorithm to potentially trade inductive noise violations for smaller overall placement size depending on the result desired, and can be useful in different SINO formulations. The weighting factor for each cost component can be tuned for different design objectives. In this chapter our stated goal is to minimize placement size without violating K_{thresh} noise constraints, hence weighting factors were chosen to help us achieve these goals with maximal efficiency.

Random Moves $Random\_Move(P, P')$ performs one of the following changes to placement P to generate a new placement P': (i) Combine two random blocks in P, (ii) Swap two random s-wires in P, (iii) Move a single random s-wire to a new and random location, (iv) Insert a g-wire at a random location in P. It is worthwhile to note that combining two random blocks in a placement P is also equivalent to removing a g-wire if the two blocks are adjacent. Moves which create two adjacent g-wires in a placement are categorically rejected and a new move is tried.

```
Simulated Annealing Algorithm: Given a placement P:
Repeat
        Temp = Initial_Temperature;
        Repeat
                Random_Move(P, P');
                Candidate_Cost = Compute_Cost(P');
                ds = Candidate_Cost − Compute_Cost(P);
                if (ds < 0)
                        P = P';
                else
                        r = RANDOM(0, 1);
                        if (r < exp(−ds/Temp))
                                P = P';
        Until equilibrium at Temp is reached;
        Temp = Temp * Temperature_Adjustment;
        /*(0 < Temperature_Adjustment < 1)*/
Until Temp == Freezing_Point;
```

Figure 16: Simulated Annealing SINO Algorithm (SINO/SA)

```
Total_Violation_Figure = 0;
for each s_i in placement P
        if Compute_Coupling(s_i) > K_thresh
                Total_Violation_Figure+ = (1 + K_eff − K_thresh)^3 − 1
end
```

Figure 17: Computation of the Inductance_Violation_Figure

Temperature Adjustment and Stopping Criterion The method of temperature adjustment is shown in Figure 16. We use a simple multiplicative constant of the current temperature. At each temperature step, the variance of the current placement cost from its previous value is taken and averaged over several random moves to determine the stability of the system at each temperature. When the variance is less than a set threshold, we move to the next temperature step. The starting temperature, freezing point, temperature adjustment, and variance threshold factors were all determined experimentally.

3.5 Experimental Results

	SINO/NF	SINO/NB			
K_{thresh}	GC	SI	NO+SI	GC	SA
Net Sensitivity Rate: 10%					
0.5	3.5(2.0)	8.0	4.9	2.6	2.2
1.0		6.1	3.3	2.2	2.0
1.5		5.1	2.2	2.0	1.0
2.0		5.0	1.8	2.0	1.0
Net Sensitivity Rate: 30%					
0.5	6.5(4.0)	13.4	6.9	5.0	4.5
1.0		12.6	4.8	4.2	3.8
1.5		12.1	3.6	4.0	2.9
2.0		11.9	3.0	4.0	2.0
Net Sensitivity Rate: 60%					
0.5	12.0(9.0)	22.2	9.0	9.9	7.9
1.0		22.1	6.0	9.0	5.8
1.5		22.0	4.9	9.0	4.8
2.0		22.0	4.0	9.0	4.0

Table 4: Number of shields inserted by SINO algorithms. The numbers in parenthesis are lower bounds on the number of shields required in SINO/NF solutions.

We have implemented all algorithms in the C programming language, and have tested our implementations using a large number of examples. In this section, we first compare results obtained by different approximate algorithms to the SINO/NB formulation, and then compare results given by two formulations (SINO/NF versus SINO/NB). We also report the running time for different formulations and algorithms.

We use a coplanar interconnect structure containing 32 s-wires as an example to determine the performance of the algorithms for different combinations of K_{thresh} and sensitivity rate. We consider the following values for K_{thresh}: 0.5, 1.0, 1.5 and 2.0. When $K_{thresh} = 0.5$, the total of coupling coefficients for a net is less than 0.5 in the target SINO solution. We iterate the following sensitivity rates: 10%, 30% and 60%. When the sensitivity rate is 10%, each net is sensitive to 10% of all nets, and these sensitive nets are picked randomly for the given s-wire. The number in parenthesis is the SINO/NF lower bound on the number of shields.

For each combination of K_{thresh} and sensitivity rate (see Table 4), we report the resulting number of shields for different formulations and algorithms. To make the comparisons "fair" among the different algorithms, we run each algorithm on the same initial placement for 20 different random placements/sensitivities. The average of these 20 runs is shown in Table 4. Note that there is no entry in the tables for Cx noise because there was not any in all cases. Finally, it is worthwhile to point out that we did not tune our SINO/SA algorithm for different examples.

3.5.1 Comparison between Different SINO/NB Algorithms

We compare the following approximate SINO/NB solutions given by the following algorithms: greedy-based shield insertion (SI), net ordering followed by SI (NO+SI), graph-coloring based SINO (SINO/GC), and simulated-annealing based SINO (SINO/SA). One may easily see from Table 4: First, SI is always significantly worse than all of the other solutions in terms of the number of shields inserted. In the worst case, SI yields a result about 550% worse than SA in terms of the number of shields inserted (see sensitivity rate of 60% and K_{thresh} of 2.0). Furthermore, performing net ordering before SI, i.e., NO+SI significantly outperforms SI only because we need not insert shields for Cx violations which may be present without performing net ordering. On the other hand, at lower sensitivity rates (10% and 30%) and low K_{thresh} values, SINO/GC performs significantly better than NO+SI in terms of shields inserted (ranging from 46% better to 9% better) because SINO/GC can consider global placement of signal nets to minimize noise, whereas NO+SI cannot. As we move to higher sensitivity rate (60%) and K_{thresh}, we see that NO+SI begins to overtake SINO/GC because SINO/GC cannot generate solutions smaller than the maximum clique in the graph by design, hence not fully exploiting the noise bound. NO+SI also approaches the performance of SINO/SA under these same circumstances because in terms of probability, for high sensitivity rates and noise bounds, the initial random placement (used in NO+SI) is likely to distribute the many sensitive nets fairly uniformly giving less relative benefit from the flexibility to rearrange all nets and shields in SINO/SA. For a concrete illustration, at sensitivity rate of 60% and $K_{thresh} = 1.0$, NO+SI performs 33% better than SINO/GC and 4% worse than SINO/SA.

As we expect, SINO/SA always performs the best for any given setting. In terms of the number of shield wires, compared to NO+SI, its improvement increases as sensitivity rates and K_{thresh} decrease, as explained previously.

It is 4% better at sensitivity rate of 60% and $K_{thresh} = 1.0$, and is 55% better at sensitivity rate of 10% and $K_{thresh} = 0.5$. Therefore, it is important to consider simultaneous net ordering and shield insertion, rather than separated net ordering and shield insertion. The improvement of SINO/SA over SINO/GC does not depend much on the sensitivity rate, but increases when K_{thresh} goes up, for the same reason NO+SI outperforms SINO/GC at higher K_{thresh}. At sensitivity rate of 60%, it is 12% better for $K_{thresh} = 0.5$, and is 56% better for $K_{thresh} = 2.0$.

3.5.2 Comparison between SINO/NB and SINO/NF Formulations

We first compare the SINO/NF and SINO/NB solutions. We base our comparison on the results produced by the best SINO/NB algorithm (SINO/SA) and the GC algorithm for the SINO/NF formulation. For the smallest K_{thresh} value of 0.5 (in order to make SINO/NB most closely approximate SINO/NF), compared to the SINO/NF formulation, the SINO/NB formulation uses about 35% fewer shield wires for all sensitivity rates. Because our GC algorithm provides an approximate to the SINO/NF formulation, we computed the average lower bound of shield wires via average maximum clique size (based on Theorem 3.5), and present these lower bounds between parentheses in the column for GC algorithm, SINO/NF formulation. Compared to these lower bounds, the SINO/NB solutions still use up to 12% fewer shield wires for the lowest K_{thresh}=0.5.

	SINO/NF	SINO/NB			
	GC	SI	NO+SI	GC	SA
Runtime	0.1sec	0.1sec	0.1sec	0.3sec	1.5sec

Table 5: Approximate run times for SINO algorithms with sensitivity rate of 30%.

Finally, we report runtime in Table 5, where the times are for a single run for a single interconnect structure. Note that the running time for the NO+SI algorithm does not include the time for net ordering–we simply applied the SI algorithm to initial placements that are free of Cx noise. As verified in our experiments, whether the initial placement is free of Cx noise does not affect the quality of solution given by SINO/GC and SINO/SA algorithms. The machine used to collect running times has a 450MHz Intel Pentium II processor. All algorithms finished the test examples in a few

seconds. Therefore, large interconnect structures can be solved easily by formulations and algorithms we have proposed.

3.5.3 Fidelity of K_{eff} Model

The K_{eff} model is easy to compute and convenient to use at a higher design level or in an earlier design stage. However, it assumes that the current returns from the nearest shield, which is not true in general. The current often returns from quiet wires within the current block if there are plenty of quiet wires in this block. On the other hand, the current often returns from shields or quiet wires outside the current block when multiple wires in the current block switch simultaneously.

Despite the K_{eff} model is far away from accurate, we observe that it has a high fidelity in the following sense: an optimal SINO solution with a higher coupling value under the K_{eff} model also has a higher SPICE-computed noise over a distributed RLC circuit model using the partial inductance model. Our observation is based on the following experiment. We first generate a large number of optimal SINO solutions using our SINO algorithms, then rank these solutions by both the coupling value under K_{eff} model and SPICE-computed noise using an accurate RLC circuit model. We then compute the absolute difference between the two rankings for each optimal SINO solution. If the ranking difference is small enough, the K_{eff} model has a high fidelity. The similar ranking techniques have been used to study the fidelity of the Elmore delay model in [30, 31].

Further, we calculate the SPICE-computed noise difference with respect to the average ranking difference in the following way: let the average ranking difference be d. For a SINO/NB-v solution whose SPICE-computed noise ranking is i, we compute the relative difference between the (i+d)-th and i-th SPICE-computed noise, as well as that between the (i-d)-th and i-th SPICE-computed noise. Between the two values, the one with the larger absolute value divided by the i-th SPICE-computed noise value is defined as *noise difference* for the solution. In other words, it shows how much the noise difference is introduced if we use K_{eff} model to approximate the SPICE-computed noise voltage under an accurate RLC circuit model.

In our SPICE simulation, we generate a RLC segment for each $100\mu m$ wire segment, and a mutual inductance between any pair of two segments even though they belong to the same wire. We use the partial inductance model in [10, 32], without assuming any current return path. The parameters for the SPICE simulation of the detailed RLC circuit model are sum-

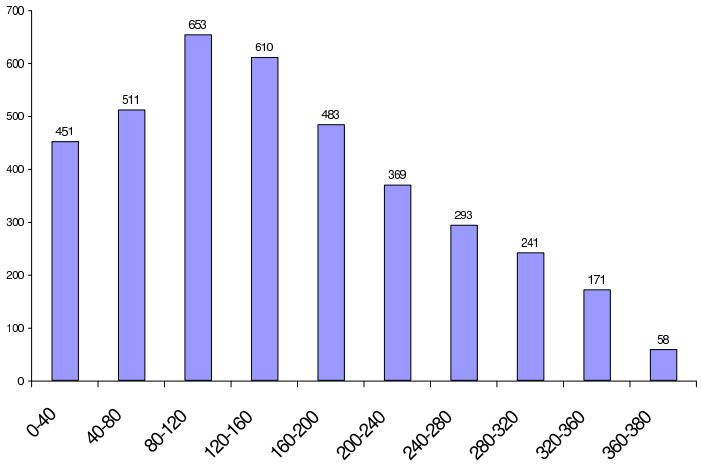


Figure 18: The distribution of the ranking difference for 3,840 SINO solutions.

marized in Table 6. We report in Figure 18 the distribution of the ranking differences for SINO solutions. The average ranking difference is only 109.387 over 3,840 SINO solutions (i.e., the relative error is 2.8486%). The noise difference is 5.3592% with respect to the average ranking difference.

Based on the above empirical evidence, we conclude that the K_{eff} model has a high fidelity over SPICE-computed noise using an accurate circuit model. Note that the above fidelity holds for optimal SINO solutions, because the SINO algorithms are able to distribute shields and quite wires evenly both spatially and temporally.

Vdd	clock	wire width	wire thickness	wire spacing	rising time	driver res	load cap
1.05V	3GHz	$1.0\mu m$	$1.1\mu m$	$0.8\mu m$	$33ps$	150Ω	60fF

Table 6: Experiment settings for fidelity study based on ITRS predicted $0.10\mu m$ technology

3.6 Conclusions and Discussions

In this section, we have formulated and solved the simultaneous shield insertion and net ordering (SINO) problem under the K_{eff} model. We have shown that the K_{eff} model has a high fidelity versus SPICE-computed noise

for SINO solutions.

In this chapter, we assume that the shields can be placed onto any track, without consideration of pre-routed P/G wires. In [33], a set of formulae has been developed to estimate the number of shields needed by optimal SINO solutions for given noise bound. Further, the simultaneous signal and power routing (SPR) problem has been formulated. The problem determines a regular P/G structure and finds a SINO solution with respect to the P/G structure such that the total routing area and number of shields are minimized. An efficient two-phase algorithm has been developed: a P/G structure is first speculated using the formula-based shield estimation, then the optimal SPR solution is found within the very limited neighborhood of the speculated P/G structure. Experiment has shown that the SPR solution reduces the total routing area by 1/3 compared to the min-area solution using a regular P/G structure without allowing any shields.

The K_{eff} model has been used in this chapter and [33]. A conservative RLC noise model has been developed in [34]. The noise model applies table-based models for capacitance and partial inductance, an inductance screening rule [35] to decide the scope of inductive coupling, and a five-pole model to compute noise voltage. Further, the SINO problem has been solved under this conservative model. SPICE simulations using an accurate RLC circuit model show that the new SINO algorithm leads to solutions with a smooth tradeoff between the number of shields and targeted noise bound.

4 Acknowledgment

The author would like to thank the following persons for valuable contributions to results presented in this chapter: Dr. Norman Chang, Dr. Shen Lin and Dr. Weize Xie at Apache Design Solutions, Dr. O. Sam Nakagawa at HP Labs, Mr. Kevin M. Lepak, and Mr. James Ma at University of Wisconsin.

A Web-based tool has been developed based on the inductance model presented in this chapter. This tool was developed by Mr. Min Xu at University of Wisconsin and is available at http://eda.ece.wisc.edu. New progress related to SINO and other interconnect synthesis techniques will be made available at the website too.

References

[1] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal*, vol. 21, pp. 1–94, 1996.

[2] M. Kamon, M. Tsuk, and J. White, "Fasthenry: a multipole-accelerated 3d inductance extraction program," *IEEE Trans. on MTT*, 1994.

[3] H. Johnson and M. Graham, *High-Speed Digital Design – A Handbook of Black Magic*. Prentice Hall, 1993.

[4] J. Lillis, C. Cheng, S. Lin, and N. Chang, *High-performance interconnect analysis and synthesis*. John Wiley, 1999.

[5] D. Zhou, F. P. Preparata, and S. M. Kang, "Interconnection delay in very high-speed VLSI," *IEEE Trans. on CAS*, July 1991.

[6] A. Deutsch *et al.*, "When are transmission-line effects important for on-chip wires," *IEEE Trans. on MTT*, 1997.

[7] Y. I. Ismail, E. G. Friedman, and J. L. Neves, "Figures of merit to characterize the importance of on-chip inductance," in *Proc. Design Automation Conf*, pp. 560–565, 1998.

[8] E. Rosa, "The self and mutual inductance of linear conductors," *Bulletin of the National Bureau of Standards*, pp. 301–344, 1908.

[9] A. Ruehli, "Inductance calculation in a complex integrated circuit environment," *IBM Journal of Res. and Dev.*, 1972.

[10] A. Ruehli, "Equivalent circuit models for three-dimensional multiconductor systems," *IEEE Trans. on MTT*, 1974.

[11] "Raphael user manual," *Avant! Corporation*.

[12] J. Cong, L. He, A. B. Kahng, D. Noice, N. Shirali, and S. H.-C. Yen, "Analysis and justification of a simple, practical 2 1/2-d capacitance extraction methodology," in *Proc. Design Automation Conf*, pp. 627–632, 1997.

[13] N. Chang, V. Kanevsky, O. S. Nakagawa, K. Rahmat, and S.-Y. Oh, "Fast generation of statistically-based worst-case modeling of on-chip interconnect," in *IEEE ICCD*, 1997.

[14] F. W. Grover, *Inductance Calculations: working formulas and tables.* Dover Publications.

[15] W. Press, S. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C.* Cambridge University Press, 1992.

[16] N. Chang, S. Lin, L. He, O. S. Nakagawa, and W. Xie, "Clocktree RLC extraction with efficient inductance modeling," in *Design Automation and Test in Europe*, March 2000.

[17] X. Qi, G. Wang, Z. Yu, R. Dutton, T. Young, and N. Chang, "On-chip inductance modeling and RLC extraction of VLSI interconnects for circuit simulation," in *Proc. IEEE Custom Integrated Circuits Conference*, May 2000.

[18] M. Xu and L. He, "An efficient model for frequency-dependent on-chip inductance," in *Proc. the Sixth Great Lakes Symp. on VLSI*, 2001.

[19] M. W. Beattie and L. Pileggi, "IC analyses including extracted inductance model," in *Proc. Design Automation Conf*, 1999.

[20] L. He and S. Lin, "Interconnect modeling and design for gigascale systems-on-chip with consideration of inductance," in *IEEE International ASIC/SOC Conference*, 2000.

[21] Y. Cao, C. M. Hu, X. Huang, A. B. Kahng, S. Muddu, D. S. oobandt, and D. Sylvester, "Effects of global interconnect optimizations on performance estimation of deep submicron design," in *Proc. Int. Conf. on Computer Aided Design*, 2000.

[22] J. Cong and C.-K. Koh, "Interconnect layout optimization under higher-order RLC model," in *Proc. Int. Conf. on Computer Aided Design*, 1997.

[23] T. Xue, E. S. Kuh, and Q. Yu, "A sensitivity-based wiresizing approach to interconnect optimization of lossy transmission line topologies," in *Proc. IEEE Multi-Chip Module Conf.*, pp. 117–121, 1996.

[24] Y. I. Ismail and E. G. Friedman, "Effects of inductance on the propagation delay and repeater insertion in VLSI circuits," in *Proc. Design Automation Conf*, pp. 721–724, 1999.

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability.* W. H. Freeman, San Francisco, 1979.

[26] C. Sechen, "An improved simulated annealing algorithm for row-based placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 478–481, 1997.

[27] N. Sherwani, *Algorithms for VLSI Physical Design Automation.* Kluwer Academic Publishers, 1999.

[28] O. Coudert, "Exact coloring of real-life graphs is easy," in *Proc. Design Automation Conf*, 1997.

[29] D. Kirovski and M. Potkonjak, "Efficient coloring of a large spectrum of graphs," in *Proc. Design Automation Conf*, 1998.

[30] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Rectilinear Steiner trees with minimum Elmore delay," in *Proc. Design Automation Conf*, pp. 381–386, 1994.

[31] J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," *ACM Trans. on Design Automation of Electronics Systems*, vol. 1, pp. 478–511, Oct. 1996.

[32] L. He, N. Chang, S. Lin, and O. S. Nakagawa, "An efficient inductance modeling for on-chip interconnects," in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 457–460, May 1999.

[33] J. D. Ma and L. He, "Simultaneous signal and power routing under k_{eff} model," in *The 3rd Intl. Workshop on System-Level Interconnect Prediction*, 2001.

[34] K. M. Lepak, I. Luwandi, and L. He, "Simultaneous shield insertion and net ordering under explicit noise constraint," in *Proc. Design Automation Conf*, 2001.

[35] S. Lin, N. Chang, and O. S. Nakagawa, "Quick on-chip self- and mutual-inductance screen," in *International Symposium on Quality of Electronic Design*, March 2000.