A New Efficient Method for Substrate-Aware Device-Level Placement

C. Lin

Department of Electrical Engineering
Eindhoven University of Technology
Eindhoven, 5600 MB
Tel: +31 40 2473407
Fax: +31 40 2455674
e-mail: C.Lin@tue.nl

D.M.W. Leenaerts

Philips Research Labs
Prof. Holstlaan 4
Eindhoven, 5656 AA
Tel: +31 40 2744723
Fax: +31 40 2744657
e-mail: Domine.Leenaerts@philips.com

Abstract — The performance of high-frequency mixed-signal and analog designs heavily relies on the actual layout of the circuit components on device level. Substrate coupling has been recognized as one of the major physical design bottlenecks in this field. To deal with this problem in an efficient way, a new technique is proposed based on the sequence pair structure and corner stitching. Performance gain is guaranteed at the cost of $O(M)$ run-time complexity, which is optimal. Simulations with randomly generated problem instances show that practical run-times are indeed linear in the size of the problem instance, thus enabling other design optimization algorithms to incorporate the proposed method without increasing run-time complexity.

I. Introduction

Due to the increasing integration and higher operating frequencies of mixed-signal and analog circuits, the negative impact of substrate coupling is becoming more pronounced [1], [2], [3]. As a result, additional design iterations are needed to compensate for the performance deterioration to eventually satisfy the design specifications.

This paper proposes a new and efficient method based on the sequence pair structure [4] and corner stitching [5] to take into account these substrate effects and minimize them, resulting in better designs in less design iterations. Because of the efficiency of the proposed method, overhead for other design optimization algorithms that incorporate the new method will be minimal.

In Section II some features of the sequence pair structure, corner stitching and basic device-level modules are briefly discussed. Section III discusses module expansion and area enumeration; two necessary and sufficient operations to effectively and efficiently minimize the impact of substrate coupling. A simple semi-empirical model to take into account substrate coupling between topologically neighboring devices is presented in Section IV. Furthermore, a subsection deals with the problem of substrate coupling impact minimization. Section V briefly covers the steps used in the simulated annealing optimization loop, and their associated complexities. Section VI shows the practical run-times of the optimization algorithm with and without the newly proposed methods. Finally, we draw some conclusions in Section VII.

II. Sequence Pair Structure and Corner Stitching

A. Sequence Pair

Murata *et al.* showed that the Sequence Pair (SP) is a very compact and powerful representation of a packing of rectangular modules [4]. It is especially well suited for use in a stochastic search algorithm, such as simulated annealing, to find a (near) optimal solution.

A Sequence Pair (SP) consists of two ordered sequences (S_1, S_2), $S_1 = (s_{1,1}, s_{1,2}, \ldots, s_{1,M})$, and $S_2 = (s_{2,1}, s_{2,2}, \ldots, s_{2,M})$, where the sequence elements $s_{i,j}$ are unique integers from $\Gamma = \{0, 1, 2, ..., M-1\}$. These integers are identifiers of the modules in the placement problem.

From S_1 and S_2 four sets $(\mathcal{M}_{aa}, \mathcal{M}_{ab}, \mathcal{M}_{ba}, \mathcal{M}_{bb})$ can be derived, called the $\mathcal{M}$ sets, which define the topological relationships "right of", "below", "above", and "left of", respectively. The definition of these sets is:

$$\mathcal{M}_{aa}(n) = \cup_{i=S_1^{-1}(n)+1}^{M} S_1(i) \bigcap \cup_{i=S_2^{-1}(n)+1}^{M} S_2(i), \quad (1)$$

$$\mathcal{M}_{ab}(n) = \cup_{i=S_1^{-1}(n)+1}^{M} S_1(i) \bigcap \cup_{i=1}^{S_2^{-1}(n)-1} S_2(i), \quad (2)$$

$$\mathcal{M}_{ba}(n) = \cup_{i=1}^{S_1^{-1}(n)-1} S_1(i) \bigcap \cup_{i=S_2^{-1}(n)+1}^{M} S_2(i), \quad (3)$$

$$\mathcal{M}_{bb}(n) = \cup_{i=1}^{S_1^{-1}(n)-1} S_1(i) \bigcap \cup_{i=1}^{S_2^{-1}(n)-1} S_2(i). \quad (4)$$

By $S(i)$ we denote the element on position i of sequence S, with a count index starting at 1. By $S^{-1}(n)$ we mean the index of module (identifier) n in sequence S. If the upper index value is lower than the lower index value then the union operator gives $\emptyset$. Furthermore, we define $S(0) = S(M+1) = \emptyset$. For example, if $S_1 = (3, 4, 1, 2)$ then $S_1(2) = 4$ and $S_1^{-1}(4) = 2$. In addition, if $S_2 = (2, 3, 1, 4)$ then $\mathcal{M}_{ba}(1) = \{3, 4\} \cap \{4\} = \{4\}$.

Definition 1: A packing is a minimum-area placement of rectangular modules with respect to a given SP.

By construction of a packing by means of the longest path algorithm, there are essentially four ways to obtain a packing. We will, without loss of generality, adhere to the LD-packing (hereafter simply referred to as packing) which is a placement in which every module is shifted to the **l**eft and **d**own as much as possible, preserving the $\mathcal{M}$ sets. Thus, with the SP we are able to represent *any* packing of rectangular modules.

B. Corner Stitching

Several important analog aspects must be considered during placement and routing; substrate coupling is the effect which we focus on in this paper. Substrate coupling can be described as a local phenomenom and, as such, one needs local information to minimize it. Deriving the set of modules which are topologically located around a given module (its neighbors), directly from the $\mathcal{M}$ sets requires $O(M^2)$, on average; not tolerable for practical purposes.

Ousterhout [5] introduced the corner stitching (CS) data-structure for use in computer aided layout tools. An important feature of CS is that it efficiently stores and retrieves *any* (empty or non-empty) rectangular module in an arbitrarily[1] split up 2D area. Corner stitching operations such as `module search`, `module search with hint`, `area enumeration`, `add module`, `delete module`, and `neighbor enumeration` all have worst-case complexity $O(M)$, except for the last one with $O(\delta)$ worst-case complexity. $O(\delta)$ indicates that the complexity does not scale with the instance size M, but depends on the number of modules that are of direct interest to the operation, which is often some kind of constant. In case of an optimization process, where clearly average-case scenarios are of interest, the complexities reduce to $O(\delta)$ or $O(1)$ [5].

Corner stitching (CS) provides a means to exactly determine all neighbors within a given window using localized `area enumeration`. Furthermore, the worst-case complexity is $O(M')$ for checking a single neighborhood [5], where M' is the number of modules within the window, which is usually much smaller than M. Also note that M' does not scale with M if the window size is kept constant. Thus, in practice the complexity can be bounded by a constant. This implies that computation of all neighborhood sets using CS has worst-case complexity $O(M' \times M) = O(\delta \cdot M) = O(M)$. An

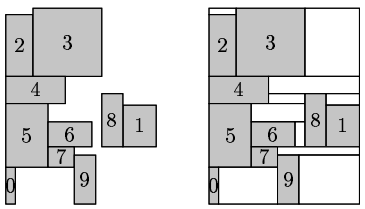


Fig. 1. The packing of 10 modules corresponding to SP $((2,3,4,5,6,8,1,7,0,9),(0,5,7,9,6,4,2,3,8,1))$ (a) without and (b) with explicit representation of empty space using corner stitching.

important feature of CS is the explicit representation of empty tiles; i.e. horizontal strips of slack space. Fig. 1(a) shows a packing corresponding to $(S_1, S_2) = ((2,3,4,5,6,8,1,7,0,9),(0,5,7,9,6,4,2,3,8,1))$. Fig. 1(b) depicts the packing which explicitly includes empty modules.

[1] Manhattan-style

C. A basic module

The atomic elements that can be manipulated during optimization are rectangular modules. A module has connecting pins on all four sides, which means that modules are adequate to represent devices such as transistors, capacitors, inductors and resistors. The space occupied by a rectangular module can be subdivided into: core module space, routing space, and expansion space. Fig. 2 depicts the basic module and the necessary data-structure elements. The data-structure of the basic module we use consists of:

- $(xpos, ypos)$: the lower left coordinate of the enclosing module,
- (w, h): the width and height of the enclosing module,
- (cx, cy): x and y offset of the core module's lower left corner,
- (cw, ch): width and height of the core module,
- (rt, rr, rb, rl): top, right, bottom, and left routing space width.

If $rl + cw + rr = w$ and $rb + ch + rt = h$ then we call the enclosing module tight, otherwise the enclosing module is loose and we have expansion space. Hereafter, routing issues are disregarded (at least the details); we included them here for completeness.

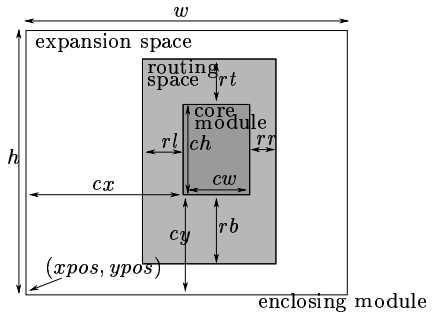


Fig. 2. A basic module.

III. Expansion and Area Enumeration

In [4] a method called *gridding* was introduced to map a packing to an SP. However, the described approach is quite ambiguous, in the sense that the packing can be mapped to more than one SP. Furthermore, if modules are placed with cutting zones of slack space such as in Fig. 1, then it is impossible to apply Murata's gridding procedure. With a slightly adapted version of operation `area enumeration` [5] it is possible to uniquely map a packing to an SP, which means that after packing the found SP again, we obtain the original packing. However, this is only true if we perform procedure **expansion** in advance, which is described hereafter. It should be noted that the SP depends on the size of the modules, as can be seen from Fig. 3, possibly causing ambiguity.

Definition 2: An equivalent packing (EP) is a packing that is obtained after expansion.

Property 1: An EP uniquely maps to an SP.

Proof: Evident from the deterministic `area enumeration` procedure [5]. □

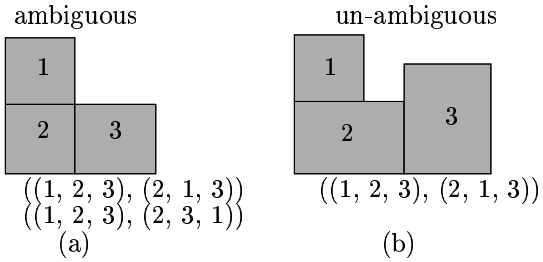


Fig. 3. The mapping from packing to SP can be ambiguous depending on the physical dimensions of the modules.

The expansion procedure[2] is needed in order to make optimal usage of available slack space. Procedure **expansion** consists of the following steps.

1. $Q = \{0, 1, 2, ..., M-1\}$, $m \in Q$.
2. Check if all touching rectangles, found by `neighbor enumeration`, to the right of m are empty.
3. If so, then expand m with the horizontally smallest rectangle size.
4. If not, then expansion is not possible for module m (the module is tight).
5. $Q = Q - m$. If $Q \neq \emptyset$ go to step 2, else STOP.

Fig. 4 shows the packing of 10 modules after (horizontal) expansion, which is the equivalent packing of Fig. 1. In almost all cases there is no slack space left after full

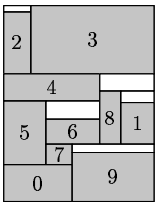


Fig. 4. A (horizontally) expanded packing of 10 modules with SP ((2, 3, 4, 5, 6, 8, 1, 7, 0, 9), (0, 5, 7, 9, 6, 4, 2, 3, 8, 1)).

expansion of a packing. This implies that an equivalent packing has maximal flexibility in shifting the modules to minimize the influence of substrate coupling.

IV. Substrate Coupling Model

A. A Semi-Empirical Model

For our purpose, a simple substrate model is required which can be quickly evaluated. We have chosen the model shown in Fig. 5 [1]. This model holds for guarded structures, but application to unguarded structures is justified [7] if we only want to *minimize* the influence of substrate coupling. Resistances R_1, R_2, R_4, R_5 and junction capacitances C_1 and C_2 will be disregarded in this treatment for simplicity, but without loss of generality. Resistance R_3 is relevant if we change the distance between two modules perpendicular to the orientation of the closest module edges. Substrate coupling between two modules i

[2]We only consider expansion in horizontal direction here without loss of generality.

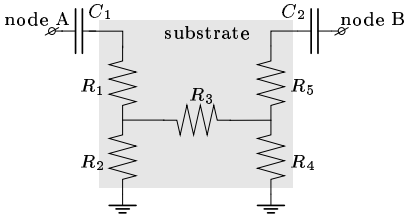


Fig. 5. A simple substrate model.

and j is inversely proportional to R_3 and is given by

$$SC_{i,j} = f_{\parallel} \cdot \frac{W}{c_1} \left\{ \frac{\pi}{2} - \text{asin}\left(\frac{c_2 W}{W + c_3 y_{i,j}} \right) \right\}^{-1}, \quad (5)$$

where the constants c_1, c_2 and c_3 depend on process parameters, and $f_{\parallel}$ is a skew function defined by

$$f_{\parallel} = H \cdot \exp\left(-\frac{1}{2} \left(\frac{x_i - \mu_j}{\sigma} \right)^2 \right), \quad (6)$$

where $W = \min(cw_i, cw_j)$, $H = \frac{1}{2}(ch_i + ch_j)$, $x_i = xpos_i + cx_i + \frac{1}{2}cw_i$, $\mu_j = xpos_j + cx_j + \frac{1}{2}cw_j$, $\sigma = \frac{1}{2}\max(cw_i, cw_j)$, and $y_{i,j}$ is the vertical component of the Manhattan distance between the core modules. Eq. (6) is used for adapting the coupling if the modules are shifted horizontally with respect to each other, which is also shown in Fig. 6(a). Fig. 6(b) shows how horizontal coupling can be changed to a vertical coupling problem by exchanging all vertically oriented variables ($ch, ypos$, etc.) with their horizontal counterpart ($cw, xpos$, etc.). After rotation, Eq. (5) can be applied.

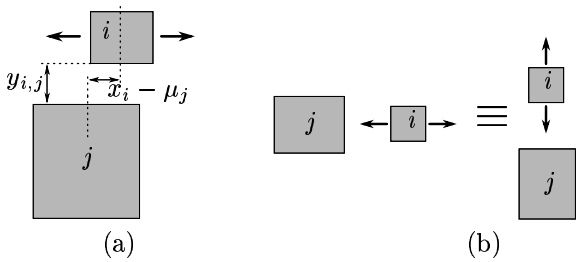


Fig. 6. The coupling model applied to general module orientations.

B. Substrate Coupling Impact Minimization

The impact (on a given performance measure) of substrate coupling from module j to module i is defined by[3]

$$\Delta P_{i,j}(x_i, y_i) = S^2_{SCi,j} \cdot SC_{i,j}(x_i, y_i) \cdot n_j, \quad (7)$$

where $S_{SCi,j} = \frac{\partial pf_i}{\partial 1/R_{3i,j}}$ is the substrate coupling sensitivity of performance function pf defined on module i. The sensitivities can, for instance, be obtained from circuit simulation, a priori. The noisiness n_i of module i depends on both amplitude and time-derivative of a predefined electrical property.

[3]Note that $S_{SCi,j} \neq S_{SCj,i}$, in general, but $SC_{i,j} = SC_{j,i}$.

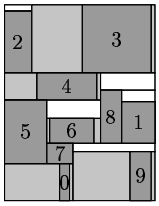


Fig. 7. The substrate-aware optimized packing of 10 modules corresponding to SP $((2,3,4,5,6,8,1,7,0,9), (0,5,7,9,6,4,2,3,8,1))$.

The problem of minimizing the impact of substrate coupling can be stated as follows.

Minimize $\Delta P_i = \sum_j \Delta P_{i,j}(x_i, y_i)$, subject to

$$\begin{aligned} xpos_i + cx_i \leq \quad & x_i \quad \leq xpos_i + w_i - rr_i - cw_i, \\ ypos_i + ch_i \leq \quad & y_i \quad \leq ypos_i + h_i - rt_i - ch_i. \end{aligned}$$

We consider the one-dimensional case of the problem for simplicity and ease of implementation, but without loss of generality. Thus, the problem can be solved by finding a global minimum within a given (integer) interval. Very efficient algorithms exist for this problem [8].

Fig. 7 shows the result after optimizing the packing of Fig. 1, given randomly generated substrate coupling sensitivities $S_{SCi,j}$ and noisiness values n_i.

V. Simulated Annealing Loop

During the simulated annealing optimization loop, performance constraints are checked with respect to wire parasitics and substrate coupling. If a packing induces constraint violation(s) then the SP is regarded as invalid and discarded. Otherwise, it is valid and accepted with a certain probability [9]. Previously [6], this approach would lead to valid SPs being discarded because substrate coupling led to a too large performance decrease. However, with the new approach the same SP can be packed substantially better. This means that SPs will not be discarded due to substrate-induced performance deterioration, if they can be shifted within the expansion space to push the substrate coupling impact below the performance constraint level.

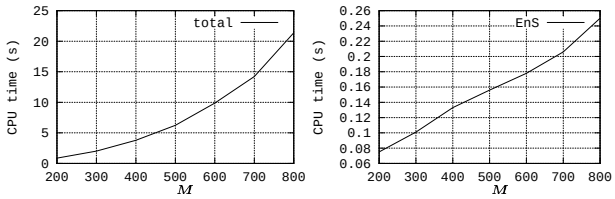
Summarizing, the following steps are needed in the simulated annealing loop:

1. packing of core modules ($O(M^2)$),
2. routing space expansion of packing ($O(M^2)$),
3. maximizing expansion space ($O(M)$),
4. optimal module placement within expansion space ($O(M)$).
5. constraint violation checking ($O(M)$).

For ease of reference we will call the proposed methods EnS (Expand and Shift).

VI. Simulation Results

We will demonstrate the efficiency of the proposed method by performing some simulations on a batch of randomly generated problem instances. Fig. 8 shows graphically the relations between the CPU times and the problem instance sizes M. The plotted CPU time is for one SA iteration, averaged over 1000 iterations. Fig. 8(a) shows a super-linear relationship between the CPU time and the number of modules M for one complete SA iteration, whereas Fig. 8(b) shows a *linear* relationship for the EnS algorithm, as expected.



(a) (b)

Fig. 8. Plots of the CPU time of one SA iteration as a function of the problem instance size M: (a) total time with EnS and (b) EnS only.

VII. Conclusions

We presented a new and efficient method, called EnS algorithm, that enables the incorporation of substrate coupling impact minimization into optimization loops. Substrate coupling has been recognized as one of the major physical design bottlenecks for high-performance high-frequency mixed-signal circuits. Therefore, minimizing the impact of substrate coupling will result in better designs in less design iterations in order to satisfy given performance specifications.

Results of simulations performed on randomly generated medium to large problem instances, clearly show that the practical run-time of the EnS algorithm is linear in the problem instance size, which is optimal.

References

[1] K. Joardar, "A simple approach to modeling cross-talk in integrated circuits," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 1212–1219, 1994.

[2] D.K. Su, M.J. Loinaz, S. Masui, and B.A. Wooley, "Experimental Results and Modeling Techniques for Substrate Noise in Mixed-Signal Integrated Circuits," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 4, pp. 420–430, April 1993.

[3] J. Catrysse, "Measured Distortion of the Output-Waveform of an Integrated Opamp Due to Substrate Noise," *IEEE Trans. on Electromagnetic Compatibility*, vol. 37, pp. 310–312, 1995.

[4] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 1518–1524, 1996.

[5] J.K. Ousterhout, "Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools," *IEEE Transactions on Computer-Aided Design*, vol. 3, no. 1, pp. 87–100, January 1984.

[6] C. Lin, "A Promising Approach to Substrate-Aware Placement Using the Sequence Pair," *unpublished*, 1999.

[7] L. Deferm, C. Claes, and G.J. Declerck, "Two- and Three-Dimensional Calculation of Substrate Resistance," *IEEE Transactions on Electron Devices*, vol. 35, no. 3, pp. 339–352, March 1988.

[8] G.E. Forsythe, M.A. Malcolm, and C.B. Moler, *Computer methods for mathematical computations*, Prentice-Hall, 1977.

[9] C. Lin, "Substrate-aware placement and routing," in *Proceedings Workshop on Circuits, Systems and Signal Processing*, 1998, pp. 379–383.