Dynamic Bound Generation for Constraint-Driven Routing

Enrico Malavasi† and Alberto Sangiovanni-Vincentelli

Department of Electrical Engineering and Computer Sciences,
University of California, Berkeley

† *On leave from* Dipartimento di Elettronica ed Informatica
Università di Padova, Italy

Abstract

We propose a technique to update dynamically the bounds used during constraint-driven routing. Moderate bound violations are allowed as long as no constraint violations are induced. Adaptive net scheduling is made possible during routing, so that nets requiring an implementation with large parasitics can take advantage of the margin made available to them by other parameters maintained within their own bounds. The user is provided with a quantitative evaluation of the effectiveness of the tool in enforcing the set of constraints on the given design.

1. Introduction

The constraint-driven approach to layout [1] is based on the translation of performance constraints into a set of bounds for all layout parameters. A tool able to enforce all these bounds guarantees that high-level performance constraints are met. In general, an infinite number of feasible sets of bounds exist, and in order to determine the most suitable one, an optimization problem must be solved, usually referred to as "constraint generation" or "constraint translation". The constraint-driven approach has been employed successfully in several fields of layout design for enforcement of timing [2] and analog constraints [3][4]. However, it suffers from two main limitations. The first is the fact that constraint generation is usually based on heuristic assumptions and incomplete information, and consequently bounds are not always feasible. If a violation occurs, the tool is not be able to finish its task and requires the generation of a new set of bounds. The second limitation is with sequential algorithms, such as area routing, whose result depends strongly on the order in which their steps (e.g. net scheduling) are performed. A bound might become feasible or infeasible depending on the order in which it is enforced.

All constraint-driven routers suffer from a strong dependence on net scheduling. The ones which try to enforce directly the bounds, such as [5], can only route nets which can be implemented without bound violations. Weight-driven tools, such as ANAGRAM [6] and ROAD [7], allow bound violations, since their target is the minimization of a cost function rather than direct bound enforcement. However, their dependence on net scheduling is still considerable, and the performance constraints are not guaranteed to be met. A solution meeting all specs must be searched iteratively, either through aggressive rip-up and re-route, or through repeated routing with different sets of automatically generated weights.

In this paper we propose a technique to update dynamically the bounds during constraint-driven area routing. It applies to all the performance constraints which are continuous and sufficiently regular to use sensitivity analysis. Dynamic bound generation requires negligible computational overhead, and guarantees constraint enforcement with maximum flexibility and little dependence on net scheduling. For each parasitic maintained within its bound, we take into account the corresponding margin made available to performance degradation. Net scheduling is redefined dynamically during routing, so that nets requiring an implementation with large amounts of parasitics are routed after enough margin has been cumulated. The user is provided with a quantitative evaluation of the tool's suitability to enforce the given design constraints.

In Section 2 the problem of adaptive bound generation is introduced. In Section 3 the balance between constraint margins and bound violations is described. Section 4 describes the implementation of adaptive net scheduling. A test case example is shown in Section 5, and conclusions are reported in Section 6.

2. Dynamic bound generation

Notation

In what follows, N_p denotes the number of layout parasitics $p_j (j = 1, \ldots, N_p)$, $p_j^{(0)}$ is the *nominal* value of p_j, and $p_j^{(b)}$ is its upper *bound*. N_k is the number of performance functions, denoted as $K_i (i = 1, \ldots, N_k)$. All parasitics are subject to variations with respect to their nominal values, and ΔK_i is the corresponding degradation induced on K_i. The sensitivity of K_i with respect to p_j is indicated as $S_{i,j}$. We assume that performance functions are continuous and sufficiently regular in an interval around their nominal value, and therefore degradations can be linearized as in [1]:

$$\Delta K_i = \sum_{j=1}^{N_p} S_{i,j} \left(p_j - p_j^{(0)} \right) . \qquad (1)$$

The maximum allowed degradation of K_i is denoted as $\overline{\Delta K_i}$, hence performance constraints are expressed as

$$\Delta K_i \leq \overline{\Delta K_i}, \qquad \text{for } i = 1, \ldots, N_k$$

Problem formulation

As shown in [1], we can assume without loss of generality that $S_{i,j} \geq 0,\ \forall i, j$, and $p_j^{(b)} > p_j^{(0)},\ \forall j$. Halfway through a routing session, after some but not all of the nets have been routed, a number l of parasitics p_j are known, while $N_p - l$ are still not known. Without loss of generality, we assume that parasitics are sorted so that p_j is known if $j \leq l$, unknown otherwise. The only information available about the unknown parameters is their bound $p_j^{(b)}$. Hence, the worst case degradation of K_i is

$$\Delta K_i = \sum_{j=1}^{l} S_{i,j} \left(p_j - p_j^{(0)}\right) + \sum_{j=l+1}^{N_p} S_{i,j} \left(p_j^{(b)} - p_j^{(0)}\right). \quad (2)$$

The difference $\delta_l(K_i) = \overline{\Delta K_i} - \Delta K_i$ is called **constraint violation** if negative, and **constraint margin** if positive. Its expression can be computed from (1) and (2):

$$\delta_l(K_i) = \sum_{j=1}^{l} S_{i,j} \left(p_j^{(b)} - p_j\right). \quad (3)$$

The layout is correct only if $\delta_l(K_i) \geq 0,\ \forall i$. If $\delta_l(K_i) > 0,\ \forall i$, it is possible to increase at least one of the bounds of the remaining $N_p - l$ parasitics, thus enhancing the flexibility of the tool. The modified bounds $p_j^{(b')}$ are defined for $j > l$ in such a way that the modified worst-case degradation

$$\Delta K_i{}' = \sum_{j=1}^{l} S_{i,j} \left(p_j - p_j^{(0)}\right) + \sum_{j=l+1}^{N_p} S_{i,j} \left(p_j^{(b')} - p_j^{(0)}\right)$$

becomes equal to $\overline{\Delta K_i}$. This condition is expressed for all i by the following equation:

$$\sum_{j=1}^{l} S_{i,j} \left(p_j^{(b)} - p_j\right) + \sum_{j=l+1}^{N_p} S_{i,j} \left(p_j^{(b)} - p_j^{(b')}\right) = 0 \quad (4)$$

That is, we want to solve a new constraint generation problem for parasitics $p_{l+1}, \ldots, p_{N_p}$, given the values of all parasitics $p_1, \ldots, p_l$. The modified bounds must meet the following

Requirements:

(i) The modified bounds must be independent from the net routing order. In particular, if we first route a net, then rip it up, bounds should be unchanged.

(ii) The ratios between bounds must remain approximately the same, so that the bounds of critical parasitics (i.e. with large sensitivities) remain lower than those of the non-critical ones.

(iii) Computation needs to be *fast*, because it must be repeated after every modification of the layout.

Proposed solution

For sake of simplicity, consider first the case where only one performance constraint $\Delta K_i \leq \overline{\Delta K_i}$ is specified. After each layout step, the modified bounds for the unknown parasitics are defined as follows:

$$p_j^{(b')} = p_j^{(b)} + \rho \left(p_j^{(b)} - p_j^{(0)}\right), \quad (5)$$

where $\rho \geq 1$ is a parameter called **margin factor**. We use the notation ρ_l to denote explicitly the value of ρ when l parasitics are known, and $N_p - l$ are unknown.

From (3), (4), and (5): $\rho_l = \dfrac{\delta_l}{D_l}$

where $D_l = \sum_{j=l+1}^{N_p} S_{i,j} \left(p_j^{(b)} - p_j^{(0)}\right)$.

Since $p_j^{(b)} > p_j^{(0)}$ and $S_{i,j} \geq 0$ for all j, $D_l = 0$ with $l < N_p$ if and only if all the sensitivities of K_i with respect to the unknown parasitics are null. This means that for the i-th performance constraint every value of those parasitics is acceptable. Therefore, if $D_l = 0$ we define $\rho_l = \infty$.

It is immediate to verify that definition (5) meets requirements *(i)* and *(ii)*. Req. *(iii)* is satisfied too, because ρ can be computed incrementally. In fact, consider a routing step bringing the number of known parasitics from l_1 to l_2. If $l_2 > l_1$ (for instance, after routing a new net), the margin factor is

$$\rho_{l_2} = \frac{1}{D_{l_2}} \left[\rho_{l_1} D_{l_1} + \sum_{j=l_1+1}^{l_2} S_{i,j} \left(p_j^{(b)} - p_j\right)\right]$$

where $D_{l_2} = D_{l_1} - \sum_{j=l_1+1}^{l_2} S_{i,j} \left(p_j^{(b)} - p_j^{(0)}\right)$.

Analogously, if $l_2 < l_1$ (for instance, after a rip-up):

$$\rho_{l_2} = \frac{1}{D_{l_2}} \left[\rho_{l_1} D_{l_1} - \sum_{j=l_2+1}^{l_1} S_{i,j} \left(p_j^{(b)} - p_j\right)\right]$$

where now p_j is the value of parasitic p_j **before** the layout step (i.e. when p_j was known), and

$$D_{l_2} = D_{l_1} - \sum_{j=l_2+1}^{l_1} S_{i,j} \left(p_j^{(b)} - p_j^{(0)}\right).$$

If $(N_k > 1)$ performance constraints are specified, it is necessary to compute $\rho = \rho(K_i)$ for each of them, using the expressions described above. The value used for ρ is the minimum:

$$\rho = \min_{i=1,\ldots,N_k} \rho(K_i)$$

3. Violation control

As long as no violations occur, ρ assumes only non-negative values. Every time a parasitic p_j becomes known, if $p_j < p_j^{(b)}$, it adds a positive contribution to $\rho(K_i)$, for all the K_i such that $S_{i,j} > 0$. As a consequence, the performance constraint margin $\delta(K_i)$ grows. In principle, a bound violation $p_j > p_j^{(b)}$ is acceptable if it induces no constraint violations, that is if $\Delta K_i{}' \leq \overline{\Delta K_i}, \forall i$. In practice, however, large bound violations are not recommended, because they lead to an erosion of the available performance degradation margin and to a set of very tight modified bounds, which might become infeasible. It is a good idea to accept bound violations only as long as $\rho \geq 0$, which guarantees that none of the modified bounds is tighter than the default initial bounds, i.e. $p_j^{(b')} \geq p_j^{(b)}, \quad \forall j$.

During routing, the margin factor is an indicator of how well the tool is doing with respect to the given bounds, or equivalently, of how tight the bounds are for the given tool. If ρ is negative, some of the bound were too tight, and some others needed to be tightened to prevent constraint violations. If ρ is positive, but close to 0, the tool is keeping most of the critical parameters (i.e. those with large sensitivities) near their upper bound, which means that bounds are well defined, and performance constraints are tight. If on the contrary $\rho \gg 0$, it means that the bounds are loose, and constraints could be tightened. The value of ρ during a routing session can be used to validate the combination tool/bounds for a given constrained routing problem.

4. Adaptive net scheduling

With area routing, layout quality depends critically on net scheduling. With an inappropriate net order, even a feasible layout might not be implemented correctly, i.e. within the constraint specifications. Unfortunately, there is no valid criterion to determine the best net order with a general set of constraints [8]. While numerous heuristic approaches have been developed, none of them has proved successful and reliable in general, and most area routers must employ some rip-up and re-route strategy to achieve correct layouts with reasonable density. The strategy must be targeted not only to achieve 100% routing, but also to meet the constraint specifications, and to the best of our knowledge, no techniques have been proposed so far to address this issue.

With dynamic bound generation, layout quality depends much less on net order. Furthermore, net scheduling can be dynamically adjusted to maximize the constraint margins, using the margin factor. If ρ becomes negative after a layout step, a violation has occurred due to the latest routed net. This net is ripped-up, and re-inserted in the stack of unrouted nets with a lower priority. Routing proceeds with the remaining nets. That net is routed again later during the layout, when the constraint margin has increased, the net is easier to route. At the end, if one or more nets are left, neither of which could be routed without making $\rho < 0$, they are routed in their current order. If the final value of $\delta_{N_p}(K_i)$ is non-negative for all i, all constraints are satisfied, otherwise a violation has occurred.

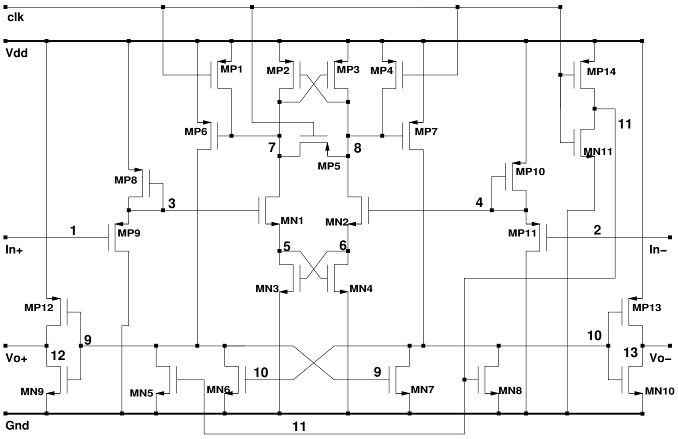

Fig. 1. *Schematic of a comparator.*

Cap.to gnd		Net-to-net cap.	
net	*Max cap.(fF)*	*Net pairs*	*Max cap.(fF)*
2	77.2	(1,3)(1,7)(1,8)	2.0
3	6.0	(1,11)	39.3
4	6.0	(2,3)(2,6)(2,8)	2.0
7	70.6	(3,7)(3,8)(3,9)	2.0
10	61.2	(3,10)	12.0
11	6.0	(3,4)	35.0
	...		...

Table 1. *Some of the most critical bounds on stray capacitances*

5. Results

The approach to adaptive bound generation and dynamic net scheduling has been implemented in the area router ROAD [7], programmed to accommodate direct bound-driven operation as well as a weight-driven cost-function. The router has been used in bound-driven mode to validate the methodology.

Figure 1 shows the schematic of a clocked comparator. The following constraints on maximum switching delay:

$$T_{LH} \leq 5ns, \quad T_{HL} \leq 5ns$$

have been used to determine a set of bounds on stray capacitances, and the circuit was routed using these bounds. More than 120 stray capacitances were identified as critical. Some of their bounds are listed in Table 1. Notice in particular the tight bounds on capacitance-to-ground for nets 3, 4 and 11. This set of bounds was computed with the constraint generator PARCAR [9], from a set of estimates of the range of feasible values for each parasitic. Without adaptive bound generation, we have not been able to find a feasible net scheduling allowing to route the circuit with this set of bounds. With all the orders we tried, at least one of the three nets 3, 4, and 11 gave a violation of the bound on ground capacitance. Although a scheduling might exist, which provides a feasible configuration according to these bounds, yet to find it (or to determine whether such an

Fig. 2. *Routed layout of the comparator.*

order exists) is a complex problem in itself. By using adaptive bounds, we have been able to find several different layouts satisfying the constraints on delay, starting with automatic as well as randomly generated default scheduling. One of these layouts is shown (before compaction) in Figure 2. In more than 90% of our tests, the layout was completed without constraint violations. For each scheduling, the first net introducing a violation large enough to make $\rho < 0$ was re-scheduled with lower priority, and in most of our tests the accumulated constraint margin grew large enough to finish successfully the layout. Diagrams reporting the evolution of the value of ρ in different experiments are shown in Figure 3. Scheduling "aut" was generated automatically using a heuristic based on net size and parasitic criticality, while the others are randomly generated. Notice that the automatic scheduling is not necessarily the best to start with. In this case, the best performance (the one maximizing δ_{N_p}) was achieved by a randomly generated scheduling denoted "rnd2". Negative-slope segments in the curves indicate violations of the bounds. Most of them occurred in the final stages of the routing, since they were due to rescheduled nets which had failed to meet their previous, tighter bounds. Rescheduled nets are those nets which, unless ripped-up, would have brought ρ to negative values.

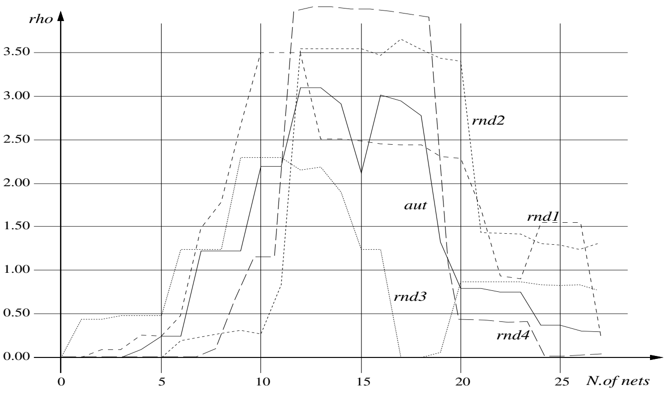

Fig. 3. *Evolution of ρ with different starting net orders.*

6. Conclusions

A technique has been illustrated, which allows to update dynamically the bounds during constraint-driven area routing. This makes it possible to introduce bound violations, while controlling the performance constraints accurately. The flexibility of the tool is increased, while performance specs are guaranteed to be met. Adaptive net scheduling is made possible by this approach, and experimental tests have shown how this decreases the dependence of routing on the starting net scheduling. By the constraint margin, the user can evaluate the effectiveness of the tool in dealing with the given design and its constraints.

Future research will focus on the application of dynamic bound generation to other layout algorithms, such as monodimensional compaction, and to irregular performance constraints, such as timing constraints, for which sensitivity analysis is not applicable in its standard form.

References

[1] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint-Based Channel Routing for Analog and Mixed-Analog Digital Circuits", in *Proc. IEEE ICCAD*, pp. 198–201, Nov. 1990.

[2] W. K. Luk, "A Fast Physical Constraint Generator for Timing Driven Layout", in *Proc. IEEE/ACM DAC*, pp. 626–631, June 1991.

[3] G. Gad-El-Karim and R. S. Gyurcsik, "Generation of Performance Sensitivities for Analog Cell Layout", in *Proc. IEEE/ACM DAC*, pp. 500–505, June 1991.

[4] E. Malavasi et al., "A Top-down, Constraint-Driven Design Methodology for Analog Integrated Circuits", in *Analog Circuit Design*, pp. 285–324, J. H. Huijsing, R. J. van der Plassche and W. Sansen Ed., Kluwer Academic Press, 1993.

[5] B. Basaran, R. A. Rutenbar and L. R. Carley, "Latchup-Aware Placement and Parasitic-Bounded Routing of Custom Analog Cells", in *Proc. IEEE ICCAD*, pp. 415–421, Nov. 1993.

[6] J. M. Cohn, D. J. Garrod, R. A. Rutenbar and L. R. Carley, "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing", *IEEE Journal of Solid State Circuits*, vol. 26, n. 3, pp. 330–342, March 1991.

[7] E. Malavasi and A. Sangiovanni-Vincentelli, "Area Routing for Analog Layout", *IEEE Trans. on CAD*, vol. 12, n. 8, pp. 1186–1197, August 1993.

[8] T. Ohtsuki, "Maze-Running and Line-Search Algorithms", in *Layout Design and Verification*, ch. 3, pp. 99–131, T. Ohtsuki Ed., North Holland, 1986.

[9] U. Choudhury and A. Sangiovanni-Vincentelli, "Automatic Generation of Parasitic Constraints for Performance-Constrained Physical Design of Analog Circuits", *IEEE Trans. on CAD*, vol. 12, n. 2, pp. 208–224, Feb. 1993.