Post-Layout Timing-Driven Cell Placement Using an Accurate Net Length Model with Movable Steiner Points*

Amir H. Ajami and Massoud Pedram
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089
{aajami, massoud}@zugros.usc.edu

ABSTRACT – *This paper presents a new algorithm for timing-driven cell placement using the notion of movable Steiner points that capture the net topology. The proposed algorithm improves the timing closure at the backend of the EDA design flow. Unlike conventional flows that perform placement and routing in two separate steps and use rough estimates of the net lengths during placement, our algorithm uses accurate net lengths by considering the net topologies during the Elmore delay calculation step and dynamically updates the routing during the concurrent placement of Steiner points and cells. The simultaneous placement and routing problem is formulated as a mathematical program with a small number of variables and solved by the Han-Powell method. Experimental results demonstrate the effectiveness of the new approach compared to the conventional flows.*

1 Introduction

The strong demand for complex high performance digital circuits motivates a continuous reduction of the minimum feature size in VLSI process technologies, which in turn introduces new challenges. Specifically, the interconnect delay has become a dominant factor in delay calculations. On the other hand, the strong demand for faster clock speeds calls for more aggressive timing-driven EDA tools. Inconsistency of the delay models that are used in different stages of the EDA flow causes the timing-closure problem (also known as the solution oscillation problem) in conventional flows. In contrast, unification-based approaches, which combine different stages of optimization flow into one integrated step, solve the timing closure problem by using unified timing and data models. To-date, the unification-based approaches have mostly focused on the timing-closure problem between the front-end (synthesis) and back-end (layout) of EDA flows [1].

This paper presents a unification-based approach to improve the timing-closure inside the back-end of an EDA flow, which is caused by the inconsistency between the delay calculations performed during the placement and routing stages. In the past, this inconsistency was ignored. However as the interconnect delays become more dominant compared to the gate delays, this conflict cannot be ignored anymore. Timing–driven placement has been studied extensively in the literature. Existing techniques may be classified into two major categories: net-based and path-based.

*This work was supported in part by the SRC under contract number 98-DJ-606.

In the net-based approach, after assigning weights to nets and updating these weights based on their timing criticality, the placement algorithm seeks to minimize the total weighted net length by placing the cells in an iterative manner [2] [3] [4] [5] [6] [7]. In the path-based approach, the placement algorithm chooses a fixed number of critical paths after performing timing analysis on the circuit netlist and then seeks to minimize the delay of these paths by placing the cells [8] [9] [10] [11] [12] [13]. Path-based approaches formulate the timing-driven placement problem more accurately than net-based approaches. Since path-based approaches do not know the net topologies prior to the routing step, they often approximate the net lengths using models such as the minimum bounding-box or the source-sink edge model [12] [14].

By performing global routing after the placement step, the exact topology of each net is determined due to the construction of its Steiner routing tree. There are a number of different Steiner routers based on the objective function used. Earlier works tried to minimize the cost (i.e. the total edge length) of the resulting Steiner routing tree [15]. More recent works attempt to simultaneously minimize the cost and the radius (the longest source-sink path length). In timing-driven global routing (which is our concern in this paper), the objective is to keep the critical-sink (CS) arrival time delay to a minimum while making the routing cost of the net as low as possible [16].

Standard backend flow performs timing-driven placement followed by timing-driven global routing. In this flow, the net length model used during placement is based on the bounding-box model or clique model whereas during global routing it is based on the CS-Steiner routing trees. Due to this inconsistency in determining the net lengths, after performing the global routing the delay information of critical paths may significantly change. Hence it becomes likely that newly created critical paths will be introduced, which in turn must be handled by another level of timing-driven placement. This solution oscillation happens to be costly in terms of the design time. Worst of all, there is no guarantee that it will eventually converge.

In this paper we use a placed and routed circuit as the initial solution. In contrast to the conventional techniques, which estimate the length of the nets, our algorithm computes the net delays based on the exact length of each net by considering its topology as defined by the relative arrangement of the Steiner points in its routing tree. Next, by simultaneously moving the Steiner points and the cells on the most critical paths while preserving the topology of each critical net, our algorithm minimizes the arrival times of the primary outputs by considering the k most critical paths. Optimizing the cycle time is subject to satisfying delay constraints on the fanout branches connected to the set of critical

paths so as to reduce the probability of introducing new critical paths in the process. Our algorithm performs topological fix-ups during the optimization routine to ensure the correct construction of a minimum critical-sink routing tree and limit the potential increase in the total cost of the Steiner trees.

The remainder of the paper is as follows. Section 2 covers background issues such as a review of currently used delay models in timing-driven placers and the delay model used in this paper. Section 3 introduces the problem formulation, while section 4 describes the proposed algorithm and optimization techniques. Experimental results and concluding remarks are given in sections 5 and 6, respectively.

2 Background

2.1 Wire length Model

Path-based timing-driven placers attempt to minimize the delay of the critical path, which in turn is dependent on the length of the nets on the path. They often approximate this length using the minimum bounding box model. Consider net_i with the source S_0 and n sinks $\{S_1, S_2, \ldots, S_n\}$ (Figure 1):

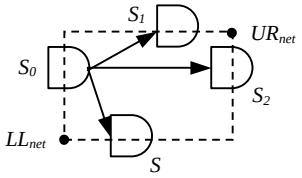


Figure 1: Minimum-Bounding box for net_i.

UR_{net} and LL_{net} denote the upper-right corner and lower-left corners of the minimum-bounding box, respectively. The length of the bounding box is defined as the Manhattan distance between these two corners times a variable ρ which is a function of the number of sinks contained in the bounding box and is used to adjust the estimation error of this interconnect model [17]. The length of each net can be written as:

$$\ell_{net} = \rho \cdot [(x_{UR} - x_{LL}) + (y_{UR} - y_{LL})] \quad (1)$$

Using the bounding box model, however, gives rise to high inaccuracy of delay estimates. This fact motivates us to use a more detailed net length model, which takes the topology of the net into account. We know the exact topology of the Steiner routing tree and can write the exact length of each source-to-sink path based on the locations of the source node, the Steiner points and the sink node. Assume the routing tree of net_i and its equivalent lumped *RC* model as shown in Figure 2:

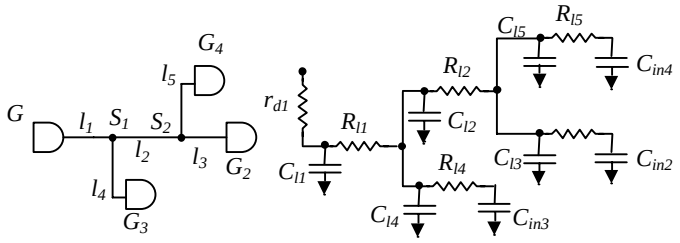


Figure 2: A sample Steiner tree of net_i.

The exact net length l_k between gate G_i and Steiner point S_j can be computed as:

$$\ell_k = \left| x_{G_i} - x_{S_j} \right| + \left| y_{G_i} - y_{S_j} \right| \quad (2)$$

The lumped resistance and capacitance for net_i are defined as:

$$R_{\ell_i} = \overline{r}_x \cdot \ell_{x_i} + \overline{r}_y \cdot \ell_{y_i} \ , \ C_{\ell_i} = \overline{c}_x \cdot \ell_{x_i} + \overline{c}_y \cdot \ell_{y_i} \quad (3)$$

where $\overline{c}_x, \overline{c}_y, \overline{r}_x, \overline{r}_y$ are capacitance per unit length in x and y directions and resistance per unit length in x and y directions, respectively. We use the Elmore delay model to calculate the delay between two gates and model the interconnect by using a lumped circuit model. In Figure 2, r_d is the output resistance of the driver gate, C_{in}'s are the input capacitance of sinks and C_l's and R_l's are the lumped capacitance and resistance of each interconnect segment, respectively.

2.2 Delay Calculation

Consider a net with a source node and k sink nodes. Elmore delay seen from the source to the i^{th} sink is computed as follows:

$$delay_i = d_{\text{int}} + \sum_{j=1}^{m} R_{ij} \cdot \sum_{k=1}^{n} C_{jk} \quad (4)$$

where m is the number of intermediate nodes on the path from the source to the i^{th} sink, R_{ij} is the total sum of resistances from the source node up to the j^{th} node on the path from the source to the i^{th} sink, and C_{jk} is the capacitance seen at node j on the path i plus the summation of downstream capacitances seen at n side-branches going out of the j^{th} node. We denote the intrinsic delay of the source gate as d_{int}. To use the exact value of both lumped capacitances and resistances for individual segments of nets, we need to consider the topology of each net, which is known after the routing phase. We consider two forms of delay calculation based on their complexity and accuracy.

2.2.1 Delay Formulation I

Assume we want to write the delay between source gate G_1 and sink gate G_2 in Figure 2. By rearranging the delay equation, we obtain:

$$d(G_1, G_2) = r_{d_1} \cdot (\sum_{i=1}^{5} C_{\ell_i} + \sum_{i=2}^{4} C_{in_i}) + R_{\ell_1} \cdot (\sum_{i=2}^{5} C_{\ell_i} + \sum_{i=2}^{4} C_{in_i})$$
$$+ R_{\ell_2} \cdot (C_{\ell_3} + C_{\ell_5} + C_{in_2} + C_{in_4}) + R_{\ell_3} \cdot C_{in_2} + d_{\text{int}} \quad (5)$$

To simplify the notation we only consider the *1-D* space (extension to both x and y directions is straightforward). In general, by considering Figure 2, if there are k movable Steiner points on the path between two gates G_i and G_j, the number of variable length net segments between them has a lower bound of $2k+1$ and an upper bound of $3k+1$, depending on the number of fanouts going out of each Steiner point. If we assume that there are p variable-length segments, then we can rewrite the propagation delay $d(i,j)$ between gates G_i and G_j based on the segment lengths by using equations (3) and (4) as follows:

$$d(i, j) = \sum_{i=1}^{p} \alpha_i \cdot \ell_i + \sum_{i=1}^{p} \kappa_i \cdot \ell_i \cdot (\sum_{j=1}^{p} M_{ij} \cdot \ell_j) + K \quad (6)$$

where α_{1xp}, κ_{1xp}, $\mathbf{M}_{pxp}$ *and* K_{1x1} are two constant vectors, a constant matrix, and a constant scalar value, respectively. If we allow k movable Steiner points on the path between every pair of cells, the number of variables used in equation (6) will be $k+2$ ($2k+4$ in *2-D* space).

2.2.2 Delay Formulation II

We make an important observation with respect to the computation of the $d(i,j)$ between G_i and G_j. In practice r_d is much larger than R_l's where C_l's are comparable to C_{in} values. Consequently, it is more important to have an accurate value for C_l's than for R_l's. For this reason, we use the Steiner points to obtain accurate values for the capacitance of partial nets (C_l's), but use the minimum bounding box to derive the resistance of the whole net. In this way the delay equation (5) is simplified to:

$$d(G_1, G_2) = r_{d_1} \cdot (\sum_{i=1}^{5} C_{\ell_i} + \sum_{i=2}^{4} C_{in_i}) + R_{net} \cdot \sum_{i=2}^{4} C_{in_i} + d_{\text{int}} \qquad (7)$$

where R_{net} is computed using the length of the minimum bounding box as in equation (1) and C_l's are calculated from equation (3) in which each l_i is calculated from equation (2). By introducing new notation, the propagation delay derived in equation (7) can be simply written as:

$$d(i, j) = \sum_i \alpha_i \cdot \ell_i + \beta \cdot \ell_{net_i} + \kappa \qquad (8)$$

where α_i, β and κ are constants and l_{net} is calculated from equation (1). If we allow k movable Steiner points on the path between every two cells, the number of variables used in equation (8) will be $k+4$ ($2k+8$ in *2-D* space).

2.3 Timing Calculation in the Circuit

Let a directed graph $G(V,E)$ represent the circuit netlist in which vertex set V is a one-to-one correspondence with the gates on the netlist, and edge set E represents the directed connections between vertices. Associated with each arc between two vertices i,j , there is a $d(i,j)$ value which denotes the propagation delay between i,j. Also for each gate G_i on the net there is an associated arrival time a_i and a required time r_i. The worst-case arrival time a_j and the required arrival time r_i at the two end points of an arbitrary arc (i,j) on the graph are given by:

$$a_j = \max\{a_i + d(i, j) \mid \forall (i, j) \in E\} \qquad (9)$$

$$r_i = \min\{(r_j - d(i, j) \mid \forall (i, j) \in E) \qquad (10)$$

Given the arrival times at the circuit inputs, T_{start}, and required times at the circuit outputs, T_{req}, we can easily compute the arrival and required times for all the gates in the vertex set V. Based on these values, a slack s_i for vertex i is defined as $s_i = r_i - a_i$. A negative slack represents a timing violation over that vertex. A critical path Γ is a sequence of vertices $(v_s,\ldots v_e)$ along a path from primary input v_s to primary output v_e where all vertices have negative slacks.

3 Mathematical Problem Formulation

We start with a placed and routed circuit, so we are given the initial coordinates for all cells and Steiner points along the critical path. We also know all the arrival times at the primary inputs (*PI*'s) and the required times at the primary outputs (*PO*'s). By performing timing analysis on the circuit, we identify the timing-critical primary output (*CPO*) by finding the *PO* with the most negative slack. We also know the arrival time at the output of the last gate on each path branching into the critical path and the required time at the input of the first gate on each path branching out of the critical path. Let M define the set of movable objects (which are cells and Steiner points that are on the critical path) and F_{in} and F_{out} denote the set of all immediate fanin and fanout nodes of the critical path, respectively (notice that nodes may be gates or Steiner points). The mathematical programming formulation of the problem is as follows:

$$\begin{cases} \textbf{\textit{Maximize}} \quad (r_{CPO} - a_{CPO}) & (11) \\ s.t. & \\ a_i + d(i, j) - a_j \le 0 & \forall (i, j) \in E \ : \ i \ , j \in M \cup F_{in} \\ r_j - d(i, j) - r_i \ge 0 & \forall (i, j) \in E \ : \ i \ , j \in M \\ r_i - a_i \ge r_{CPO} - a_{CPO} & \forall i \in F_{out} \\ a_i \ge T_{start} & \forall i \in PI \\ r_i \le T_{req} & \forall i \in PO \\ x_{\min} \le x_i \le x_{\max} & \forall i \in M \\ y_{\min} \le y_i \le y_{\max} & \forall i \in M \end{cases}$$

where a_{CPO} and r_{CPO} are the arrival and required times at the CPO and x_{min}, x_{max}, y_{min} and y_{max} are the coordinates of the lower left and upper right corners of the chip. As we stated before, the arrival times at output of fanin gates and required times at input of fanout gates and are known values after a timing analysis pass (Figure 3). The first and second sets of constraints simply describe the equations used to capture the timing calculation in the circuit. The third set of constraints states that the slack on any path branching out of the current critical path should be no more negative than the slack of the critical path itself. This constraint therefore minimizes the chance that a new critical path will be generated after optimizing the current critical path. Notice that this constraint cannot guarantee that no new critical path will be created because of the possibility that the fanout branches may reconverge in the circuit after they leave the current critical path. The remaining sets of constraints describe the boundary conditions for timing calculation and for placement.

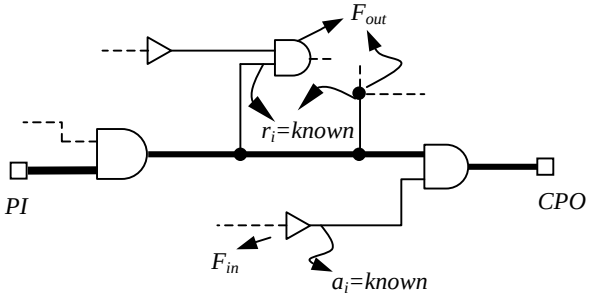


Figure 3: Fanin and Fanout constraints (critical path is shown in thick lines).

Consider a circuit with N nodes (including *PI*'s, *PO*'s, internal gates, and Steiner points for the nets connecting these gates). Suppose that the current critical path has n nodes (a *PI*, a *PO*, and $n-2$ movable nodes in between) and that the cardinalities of the corresponding F_{in} and F_{out} sets are p and q, respectively. Thus the mathematical optimization problem formulation described by system of equations (11) only has $n+p$ arrival time variables, n required time variables, and $2n$ position variables.

As we saw in section 2.2, $d(i,j)$ can be written as a polynomial of partial sums of resistances and capacitances on the intermediate nodes of the path connecting gates G_i and G_j. These partial sums are functions of net segment lengths that comprise the arc (v_i,v_j) and its fanout branches. In general $d(i,j)$ is a function of the physical coordinates of gate G_i, gate G_j and all intermediate Steiner points over the arc (v_i,v_j). Note that $d(i,j)$ is not a polynomial (because of the absolute values presented by the Manhattan distances between two intermediate nodes).

Observation: Problem formulation (11) is a non-convex program. Computing the Hessian matrix of the timing constraints proves this. More precisely, we can show that the Hessian is not positive-semi definite and hence the system of equations (11) is a non-convex optimization problem. Consequently, using the simplex method or any kind of quadratic programming technique cannot solve this problem formulation.

Notice that by using (8) in section 2.2.2, we introduce more variables into formulation (11), so using (8) instead of (6) has little impact on reducing the computation time for solving the optimization problem. For these reasons we will use delay formulation I of section 2.2.1 for calculating the delay between two gates. Each length l_i is the absolute value of the difference between the x-coordinates of the two endpoints of the corresponding net segment. Note that one or both of the x-coordinates can be variables. We approximate the absolute value function with a differentiable smooth function as follows:

$$\ell_k = \left| x_i - x_j \right| \cong \sqrt{(x_i - x_j)^2 + \beta_k} \qquad (12)$$

β_k is called the *regularization factor*, which has a very small magnitude and is set based on the required precision of the final results [18]. By substituting equation (12) into equation (6), the constraints in problem formulation (11) will become a function of arrival times and required times at each gate on the critical path and its first neighbors and the physical coordinates of the Steiner points and the cells on the critical path.

4 Problem Optimization

We first provide the theoretical background to motivate the way we solve problem formulation (11).

4.1 Background

Theorem 1: (Kuhn-Tucker's first order necessary condition) Consider the following problem:

$$\begin{aligned} &\textit{Min} \quad f(x) \\ &\textit{s.t.} \quad g(x) \le 0 \end{aligned} \qquad (13)$$

Let α be a relative minimum point for problem (13) and suppose α is a regular point for the constraints. There is a vector $\lambda \in E^M$ such that:

$$\begin{aligned} &\nabla f(\alpha) + \lambda^T \nabla g(\alpha) = 0 \\ &\lambda^T g(\alpha) = 0 \end{aligned} \qquad (14)$$

The Lagrangian for system of equations (14) can be defined as $L(x)=f(x)+\lambda^T g(x)$. As noted before problem formulation (11) is non-convex, i.e. the Hessian of its Lagrangian is not positive definite. For this reason we try to "convexify" the Lagrangian in the local sub-space and find a descent direction toward the global minimum. One common way is to define a *merit function* that is defined for the purpose of measuring the progress toward the global optimum. The merit function must be defined so as to be a minimum at the solution of the original problem, and at the same time, its value to decrease at each optimization step. Another effective method approximates the Lagrangian matrix such that the new matrix is positive semi-definite and updates it iteratively during optimization steps.

4.2 Optimization Technique

The basis of our optimization technique is the quasi-Newton method, which is a structured, modified Newton algorithm using an approximation and successive updates of the Lagrangian matrix. The iterative process is stated as follows:

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} - \alpha \begin{bmatrix} \mathrm{B_k} & \nabla g(x_k)^T \\ \nabla g(x_k) & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nabla L(x_k, \lambda_k)^T \\ g(x_k) \end{bmatrix} \qquad (15)$$

where $\mathrm{B_k}$ is a positive semi-definite approximation for Lagrangian L and will be updated at each iteration. By solving the above linear system of equations, we find the values of x and λ for the next step. This process is then repeated. α is a factor defined by the specified merit function to improve the convergence rate of the iteration. In the quasi-Newton method, the value of α is usually set to 1. The value of matrix **B** can be updated as follows:

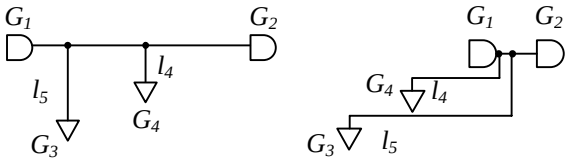
$$\begin{aligned} &\mathrm{B_{k+1}} = \mathrm{B_k} + \frac{q_k q_k^T}{q_k^T p_k} - \frac{\mathrm{B_k} p_k p_k^T \mathrm{B_k}}{p_k^T \mathrm{B_k} p_k} \\ &p_k = x_{k+1} - x_k \ , \ q_k = \nabla L(x_{k+1}, \lambda_{k+1})^T - \nabla L(x_k, \lambda_{k+1})^T \end{aligned} \qquad (16)$$

It has been shown that if the initial guess is sufficiently close to the solution (x,λ), this method converges to this solution super linearly. However, it is obvious that this "closeness" condition is unsatisfactory when looking for a general method to solve system (13). Also notice that after updating Matrix **B**, there will be no guarantee that the matrix remains positive semi-definite.

The Han-Powell method is a variant of the quasi-Newton method in which a quadratic merit function is used to update α as well as **B**. It can be shown that during the Han-Powell method, the matrix **B** always remains positive semi-definite and that the Han-Powell method is a globally convergent method since the search direction given by the merit function is a descent direction [19].

4.3 Topology Correction

Due to the movements of Steiner points along the critical path during the optimization steps, it is possible to find that some of the fanout net lengths become large. This may occur, for example, in order to shorten a particular source to sink path in the routing tree at the expense of increasing the overall routing length. This situation is described by an example in Figure 4. Assume that gate G_2 has a very large input capacitance in comparison to the other gates, which is possible when using a rich library of gates. At the same time assume that because of some physical constraints imposed by other fanins of gate G_2 or because we may increase the output load of G_2, it is not possible to move gate G_2 toward gate G_1. Having relatively small driver strength for gate G_1 forces us to reduce the length of the path between gate G_1 and G_2 as much as possible, which in turn forces gate G_1 to move toward G_2 and hence increases l_4 and l_5 (Figure 4b).



(*a*) (*b*)

Figure 4: (*a*) The original net topology and (*b*) the net topology after moving cells and Steiner points.

Notice that the topology of the routing tree subsequently needs to be changed to improve the delay and wire length. Indeed by overlapping l_4 and l_5 routes and removing the shared part of one of them and introducing a new Steiner point we will improve the delay of the critical path as well as reduce the total wire lengths (Figure 5). In general we can do this kind of fix-up when it does not violate the timing constraints at the outputs of gates G_3 and G_4.

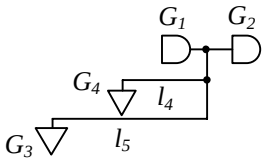


Figure 5: Routing tree after topology correction.

4.4 Flow of the Proposed Method

The complete flow of the proposed algorithm will be as follows:

1. Start with an initial solution produced by a timing-driven placement and global routing tool.
2. Perform timing analysis on the circuit to extract the *k* most critical paths.
3. Construct the problem formulation (11) using the physical location of each cell and Steiner point on the critical path and the arrival times at each gate as the system variables.
4. Perform one step of the quasi-Newton or Han-Powell iteration methods.
5. Correct the topology of fanout nets as described in section 4.3.
6. Update problem formulation (11) to reflect any net topology changes in step 5.
7. Go back to step 4 and repeat unless a satisfactory result has been generated.
8. Stop if all the critical path delays are optimized; otherwise, go back to step 2 and repeat.

5 Experimental Results

To show the effectiveness of the proposed algorithm, we applied it to a number of benchmark circuits. The experiments were done on a 700MHz P-III with 256MB of memory. The results are reported on Table 1. In the first three columns the netlist information of each circuit is given. To test our algorithm we first place the circuits with a simulated annealing based placer (TimberWolf 1.0) followed by a timing-driven placement step [12] that uses the bounding-box estimation for each net. Then we perform a C-ERT global routing over the circuit [15]. The delay (*ns*), area (mm^2) and running time (*sec*) after these steps are reported under P&R columns of the Table 1. Note that the reported delay is the propagation delay after placement and routing using the topology of the constructed Steiner trees and based on the exact net lengths. Our results after doing timing-driven placement with movable Steiner points are given in the TPGR columns. As one can see, there is a timing performance improvement between 9 to 14 percent, at the cost of a slight increase in the chip area. As was expected, the run time is higher for TPGR due to the nonlinearity of the optimization objective and the iterative nature of the Han-Powell method.

Even though the running time of one pass of P&R flow is shorter, performing the global routing stage will change the timing information of the circuit due to the inconsistency of the delay model in the placement and routing stages. Hence, for obtaining the same performance improvement as TPGR, we need to perform multiple iterations of P&R, which can be more costly in terms of running time than TPGR and there is no guarantee that this iterative process converges at the end. In contrast, the TPGR flow is a one-shot optimization process based on a stable and rather efficient optimization method.

6 Conclusion

This paper presented a new algorithm for performing timing-driven placement with global routing information using the notion of movable Steiner points. The proposed algorithm uses accurate net lengths by considering the net topologies during the Elmore delay calculation step and dynamically updates the routing during the concurrent placement of Steiner points and cells. The simultaneous placement and routing problem was formulated as a mathematical program with a small number of variables and solved by the Han-Powell method. Experimental results demonstrated the effectiveness of the new approach compared to the conventional flows. Future work consists of integrating this algorithm with a post-layout buffer insertion technique.

Circuit	*#cells*	*#cells CP*	*P & R*			*TPGR*			*Delay Imp. %*	*Area Inc.%*
			area	*delay*	*runtime*	*area*	*delay*	*runtime*		
C499	385	13	2.82	12.05	26.7	2.90	10.36	327	13.8	2.9
C2670	811	15	6.53	14.45	40.1	6.69	13.1	411	9.34	1.8
C1908	453	21	3.21	19.60	32.7	3.34	17.24	342	12.01	4.2
C5315	1720	17	8.14	23.17	109.2	8.31	20.73	1112	10.5	2.1
C3540	1149	22	7.15	26.53	81.2	7.28	23.55	615	11.2	1.9
C7552	2158	24	27.41	29.2	152.2	27.82	26.50	2120	9.23	1.5
des	3059	21	33.21	21.8	270.8	34.53	19.03	3910	11.3	4.0
biomed	6417	22	52.08	32.1	501.5	53.69	28.88	5214	10.03	3.2

Table 1: Comparing TPGR with conventional P&R flow.

References:

[1] A. Salek, J. Lou, M. Pedram, "A Simultaneous Routing Tree and Fanout Optimization Algorithm," *Proc. Intl. Conf. on CAD*, pp. 625-630, 1988.

[2] A.E. Dunlop, V.D. Agrawal, D.N. Deutsch, M.F. Jukl, P. Kozak, M. Weisel, "Chip Layout Optimization using Critical Path Weighting," *Proc. Design Automation Conf.*, pp.133-136, 1984.

[3] P. Hauge, R. Nair, E. Yoffa, "Circuit Placement for Predictable Performance," *Proc. Intl. Conf. on CAD*, pp. 88-91, 1987.

[4] S. Ou, M. Pedram, "Timing-Driven Placement Based on Partitioning with Dynamic Cut-Net Control," *Proc. Design Automation Conf*, pp. 472-476, 2000.

[5] T. Gao, P.M. Vidya, C.L. Liu, "A Performance Driven Macro-Cell Placement Algorithm," *Proc. Design Automation Conf.*, pp. 147-152, 1992.

[6] M. Marek-Sadowska, S. Lin, "Timing Driven Placement," *Proc. Int. Conf. On CAD*, pp. 94-97, 1989.

[7] M. Sarrafzadeh, D. Knol, G. Tellez, "Unification of Budgetting and Placement," *Proc. Design Automation Conference*, pp. 758-761, 1997.

[8] W. Donath, R. Norman, B. Agrawal, S. Bello, S. Han, J. Kurtzberg, P. Lowy, R. MacMillan, "Timing Driven Placement using Complete Path Delays," *Proc. Design Automation Conf.*, pp. 84-89, 1990.

[9] W. Swartz, C. Sechen, "Timing Driven Placement for Large Standard Cell Circuits," *Proc. Design Automation Conf.*, pp. 211-215, 1995.

[10] A. Srinivasan, "An Algorithm for Performance-Driven Initial Placement for Small-Cell ICs," *Proc. Int. Conf. on CAD*, pp. 94-97, 1989.

[11] M.A.B. Jackson, E.S. Kuh, "Performance-Driven Placement of Cell Based IC's," *Proc. Design Automation Conf.*, pp. 370-375, 1989.

[12] T. Koide, M. Ono, S. Wakabayashi, Y. Nishimaru, N. Yoshida, "A New Performance Driven Placement Method with Elmore Delay Model for Row Based VLSI," *Proc. Asia and South Pacific DAC*, pp. 405-412, 1995.

[13] J.M. Kelinhans, G. Sigl, F.M. Johannes, K. Anterich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," *IEEE Trans. Computer-Aided Design*, vol.10, No.3, pp. 356.365, Mar 1991.

[14] B.M. Reiss, G.C. Ettelt, "SPEED: Fast and Efficient Timing Driven Placement," *Proc. Intl Symp. of Circuits and Systems* pp. 377-380, 1995.

[15] A.B. Kahng, G. Robbins, "A New Class of Iterative Steiner Tree Heuristics with Good Performance," *IEEE Trans. Computer-Aided Design*, vol.11, pp. 893-902, July 1992.

[16] K.D. Boese, A.B. Kahng, B.A. McCoy, G. Robins, "Near-Optimal Critical Sink Routing Tree Construction," *IEEE Trans. Computer-Aided Design*, vol.14, No.12, pp. 1417-1436, 1995.

[17] W. Chen, C. Hsieh, M. Pedram, "Simultaneous Gate Sizing and Placement," *IEEE Trans. Computer-Aided Design*, vol. 19, No.2, pp. 206-214, Feb. 2000.

[18] C.J. Alpert, T.F. Chan, A.B. Kahng, I.L. Markov, P. Mulet, "Faster Minimization of Linear Wirelength for Global Placement," *IEEE Trans. Computer-Aided Design*, vol.17, No.1, Jan. 1998

[19] D. Luenberger, Linear and Nonlinear Programming, 2nd edition, *Addison-Wesley Pub. Company*, 1984.