Sensitivity Guided Net Weighting for Placement Driven Synthesis

Haoxing Ren, David Z. Pan, and David S. Kung

***Abstract*— Net weighting is a key technique in timing driven placement (TDP), which plays a crucial role for deep submiron VLSI physical synthesis and timing closure. A popular way to assign net weight is based on its slack, such that the worst negative slack (WNS) of the entire circuit may be minimized. While WNS is an important optimization metric, another figure of merit (FOM), defined as the total slack difference compared to a certain slack threshold for all timing end points, is of equivalent importance to measure the overall timing closure result for highly complex modern ASIC and microprocessor designs. Moreover, to optimally assign net weight for timing closure, the effect of net weighting on timing should be carefully studied. In this paper, we perform a comprehensive analysis of the wire length, slack and FOM sensitivities to the net weight, and propose a new net weighting scheme based on those sensitivities. Such sensitivity analysis implicitly takes potential physical synthesis effect into consideration. The experiments on a set of industrial circuits show promising results for both stand-alone timing driven placement and physical synthesis afterwards.**

***Index Terms*— Timing Driven Placement, Physical Synthesis, Net Weighting, Interconnect, Sensitivity Analysis.**

I. Introduction

It has been widely recognized that interconnect becomes a dominant factor in determining the overall performance and complexity for deep submicron VLSI circuits. The global wiring delay can easily be a factor of ten or hundred times of a logic gate delay, even with repeater insertion [1]. Since interconnect length is roughly determined by the placement step, which decides where the logic and memory elements shall be located while satisfying the layout constraints (e.g., non-overlapping), many timing driven placement techniques have been developed to minimize the wire length that are on the critical paths so that interconnect delays on timing critical paths are under control.

Existing timing driven placement algorithms can be divided into two groups: path-based and net-based. Path-based algorithms [2][3][4] consider every path simultaneously in their placement models. Path-based approaches in general have higher complexity, especially for high end ASIC designs with millions of placeable objects. For net-based approaches, one way is to assign wire length bounds to critical nets [5][6]. However, placement algorithms are usually not well suited to honor these bounds. Another popular net-based placement approach is to assign higher net weights to the more timing critical nets [7][5][6][8][9][10]. Although net weights can be iteratively updated during multiple placement runs [7][9][11][12], for acceptable turn around time, the global placement for a modern ASIC design with millions of placeable instances is usually performed only two or three times. Therefore, an effective net weight assignment is extremely important.

It shall also be noted that timing driven placement is *not* the end point of a physical synthesis flow. After placement, physical synthesis tools, such as buffer insertion/sizing and gate sizing, will be used extensively to further improve timing on the critical paths [17][18][19][16]. Thus, timing driven placement should provide a good starting point for the physical synthesis engine, and the net weighting should consider the potential effect of it. A popular way to assign net weight is based on its slack, which aims to minimize the worst negative slack (WNS) for the entire circuit [8][14][10][15]. While WNS is an important optimization metric, modern physical synthesis also uses another metric, so called *figure of merit* (FOM) to measure the quality of results for timing driven placement and physical synthesis. The FOM is defined as the total slack difference compared to a certain slack threshold for all *timing end points* (see section II for its formal definition). It can be interpreted as the amount of work left for the physical synthesis engine or to the designers for manual fix if the optimization engine alone cannot close the timing. A special case of FOM with zero slack target is the total negative slack (TNS), which was used to measure the quality of timing driven placement [12], but not explicitly used to guide TDP. In this paper, we explicitly use FOM metric to guide the placement.

Higher net weights for timing critical nets ideally lead to shorter wire lengths and less delays for critical nets, and better FOM for the overall design as well. However, it was not clear how much weight change a critical net shall have and what its potential impact on the slack and FOM is. Blindly assigning net weights without predicting their impact on the final wire length and timing characteristic such as WNS and FOM could lead to inferior results. In this paper, we present a comprehensive sensitivity analysis on the impact of net weight to wire length, slack and FOM. Although our analysis is by no means perfect, this is the first analytical study on net weighting impact on timing, i.e. slack and FOM, to our best knowledge. Based on these sensitivities, we propose a new net weighting algorithm with consideration of both FOM and slack sensitivities. In our experiment flow, to be efficient for large circuits, instead of iteratively updating net weights during TDP,

This work was supported by the IBM Degree Work Study Program and IBM Faculty Award.

H. Ren is with IBM Corporation, Austin, TX 78759 and ECE Department, University of Texas at Austin, Austin, TX 78712, email: haoxing@us.ibm.com

D. Z. Pan is with ECE Department, University of Texas at Austin, Austin, TX 78712, email: dpan@ece.utexas.edu

D. S. Kung is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, email: kung@us.ibm.com

we apply our algorithm once to obtain net weights for TDP from an initial wire length driven placement. Experimental results on a set of industrial circuits show that by adding slack and FOM sensitivities, we are able to obtain better results for not just timing-driven placement, but also the physical synthesis optimization after it. In particular, considering the FOM sensitivity to explicitly guide the net weight generation, we can further improve the final FOM measurement without deteriorating the worst slack and wire length.

The rest of the paper is organized as follows. Section II describes background information on the quadratic placement, static timing analysis and delay models used to illustrate the sensitivity analysis. Section III derives the wire length, slack and FOM sensitivities to net weight. Section IV presents a new net weighting algorithm based on the sensitivity analysis in section III. The experimental placement flow and results are shown in section V, followed by the conclusion in section VI. A preliminary version of this paper was presented in ISPD 2004 [13].

II. Preliminaries

In this section, we give the preliminaries for timing driven placement and use a hybrid quadratic programming and partitioning approach [14][9] to illustrate the net weighting process. Let x_i and y_i be the x- and y-coordinates of the center of cell i, respectively. The weighted cost of an edge (i, j) is its quadratic wire length multiplied by its weight, i.e., $w_{ij}((x_i-x_j)^2+(y_i-y_j)^2)$. The overall objective function sums up the weighted cost of all edges, as shown in the following equation.

$$Min \sum_{(i,j)\in E} w_{ij}[(x_i-x_j)^2+(y_i-y_j)^2] \tag{1}$$

where E is the edge set of the entire circuit. Since x and y are decoupled, we can write the objective function separately for each direction as follows.

$$Min \sum_{(i,j)\in E} w_{ij}(x_i-x_j)^2 \tag{2}$$

$$Min \sum_{(i,j)\in E} w_{ij}(y_i-y_j)^2 \tag{3}$$

Equations (2) (3) can be solved by various quadratic programming techniques. Following each quadratic solution, cells may be partitioned and assigned into smaller bins, with an optional repartitioning step to further improve the result. The quadratic programming, partitioning and repartitioning process may be run iteratively until the bin size is small enough. After the global placement, detailed placement (also called legalization) is done to move cells locally and remove overlaps.

To guide timing driven placement, higher net weights can be assigned to timing critical nets based on static timing analysis. For each timing point t, its arrival time $Arr(t)$, required arrival time $Req(t)$, and slack $Slk(t)$ can be computed as follows:

$$Arr(t) = \begin{cases} T_i(t) & t \in P_i \\ max_{(s,t)\in E}\{Arr(s)+d(s,t)\} & otherwise \end{cases} \tag{4}$$

where E is the set of timing arcs, $d(s,t)$ is the delay from timing point s to t, P_i is the set of timing begin points, i.e., primary inputs (PIs) and output pins of memory elements, and $T_i(t)$ is the asserted arrival time at the timing begin point t.

$$Req(t) = \begin{cases} T_o(t) & t \in P_o \\ min_{(t,t')\in E}\{Req(t')-d(t,t')\} & otherwise \end{cases} \tag{5}$$

where P_o is the set of timing end points, i.e., primary outputs (POs) and input pins of memory elements, and $T_o(t)$ is the asserted required arrival time at timing end point t.

$$Slk(t) = Req(t) - Arr(t) \tag{6}$$

The slack of a net is the slack at its source pin. To achieve timing closure, all nets should have non-negative slacks. For nanometer designs with growing variability, one may even set the slack target to be a positive value to safe guard the process variations. The *figure of merit* (FOM) can be defined accordingly as follows.

$$FOM = \sum_{Slk(t)<Slk_t}^{t\in P_o} (Slk(t)-Slk_t) \tag{7}$$

where Slk_t is the slack target for the entire design. If $Slk_t = 0$, the FOM is reduced to the TNS metric as used to measure the quality of results in [12].

To perform sensitivity analysis of slack and FOM (to be explained in the next section), the switch level RC device model and the Elmore delay model [20] are used to illustrate the concept since analytical formulae with intuitive explanation can be obtained. To guide placement, these models shall be adequate since there are many other uncertainties like the routing topology during the placement evaluation step. Our sensitivity analysis, however, can be extended to handle more general delay models [21] if necessary. For an interconnect with wire resistance R_w and capacitance C_w, let R_d be the effective output resistance of its driver, C_l be the load capacitance for its receiver, then the Elmore delay T from the driver to the receiver (through the interconnect) is

$$T = R_d(C_w+C_l)+C_lR_w+C_wR_w/2 \tag{8}$$

Let the unit length wire resistance and capacitance be r and c respectively. Then $R_w = rL$, $C_w = cL$, and (8) can be rewritten as:

$$T = \frac{rc}{2}L^2+(cR_d+rC_l)L+R_dC_l \tag{9}$$

For nets with multiple sinks, since the interconnect topology is unknown during the placement stage, we can estimate the delay from source to sink using the Elmore delay approximation.

$$T_j = R_dC_{total}+rC_lL_j+\frac{rc}{2}L_j^2 \tag{10}$$

where T_j is the source to sink delay for sink j, L_j is the source to sink distance for sink j, and C_{total} is the total capacitive load to the source. The capacitive load to the source can be estimated through the half-perimeter bounding box or the sum of total source-to-sink direct connections since the actual wiring topology is unknown at the global placement stage. For example of a net with two sinks, the delay from the source to

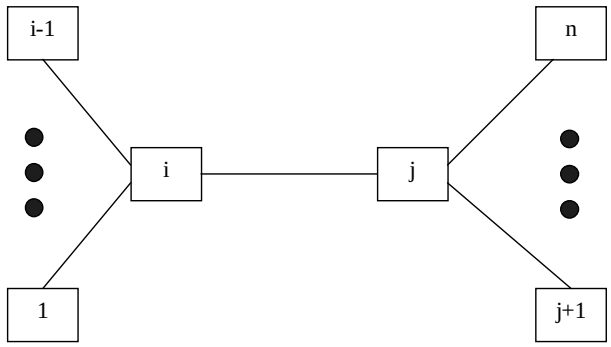


Fig. 1. Sample circuit for wire length to net weight sensitivity.

sink 1 can be estimated as follows under direct source-to-sink assumption:

$$T_1 = R_d(C_{w1}+C_{l1}+C_{w2}+C_{l2})+C_{l1}R_{w1}+C_{w1}R_{w1}/2 \quad (11)$$

where C_{w1}, R_{w1}, C_{l1} and C_{w2}, R_{w2}, C_{l2} are the capacitance, resistance and load of wire segment to sinks 1 and 2, respectively. Similar to (9), we can rewrite (11) as follows:

$$T_1 = \frac{rc}{2}L_1^2+(cR_d+rC_{l1})L_1+cR_dL_2+R_d(C_{l1}+C_{l2}) \quad (12)$$

where L_1 and L_2 are the lengths of wire segment 1 and 2, respectively.

III. Net Weight Sensitivity Analysis

In this section, we will derive the relationship of slack and FOM sensitivities to net weight. The question that we want to answer is that given an initial placement from an initial net weighting scheme, if we increase the net weight of net i by a nominal amount, how much improvement net i will get for its slack and the overall FOM.

We first derive and validate the wire length sensitivity to net weight. Then we analyze the slack sensitivity to net weight, and derive the FOM sensitivity to net weight.

A. Wire Length Sensitivity to Net Weight

We define the wire length sensitivity to net weight $S_W^L(i)$ as:

$$S_W^L(i) = \frac{\Delta L(i)}{\Delta W(i)} \quad (13)$$

where $L(i)$ is the wire length for net i, and $W(i)$ is the net weight of net i. $S_W^L(i)$ implicates how much wire length would change when there is a small amount of net weight change.

As shown in Fig. 1, suppose net ij is the net with Δw change. It connects cell i and j. Cell 1 to $i-1$ are the cells connected to i, and cell $j+1$ to n are the cells connected to j. Assuming only the net weight of net ij change and the locations of cell 1 to $i-1$ and cell $j+1$ to n remain constant [1], we can obtain the following two equations for the x-dimension from (2).

$$\sum_{k=1}^{i-1} w_{ki}(x_k-x_i)+w_{ji}(x_j-x_i) = 0 \quad (14)$$

$$\sum_{k=j+1}^{n} w_{kj}(x_k-x_j)+w_{ji}(x_i-x_j) = 0 \quad (15)$$

where w_{ki} is the net weight of net connecting cell k and i. Similar results can be obtained for y-dimension. These equations show that cell i shall be placed at the weighted center of cells 1 to $i-1$ and cell j; meanwhile, cell j shall be placed at the center of cells $j+1$ to n and cell i. Multiplying (14) with $\sum_{k=j+1}^{n} w_{kj}$, then subtracting it by (15) multiplied with $\sum_{k=1}^{i-1} w_{ki}$, we can get the wire length x_j-x_i to net weight w_{ij} relationship as:

$$x_j-x_i = \frac{\sum_{k=1}^{i-1} w_{ki}\sum_{k=j+1}^{n} w_{kj}x_k-\sum_{k=j+1}^{n} w_{kj}\sum_{k=1}^{i-1} w_{ki}x_k}{(\sum_{k=1}^{i-1} w_{ki}+\sum_{k=j+1}^{n} w_{kj})w_{ij}+\sum_{k=j+1}^{n} w_{kj}\sum_{k=1}^{i-1} w_{ki}} \quad (16)$$

To simplify the notation, we can rewrite (16) as:

$$x_j-x_i = \frac{A}{B\cdot w_{ij}+C} \quad (17)$$

where

$$\begin{aligned} A &= \sum_{k=1}^{i-1} w_{ki}\sum_{k=j+1}^{n} w_{kj}x_k-\sum_{k=j+1}^{n} w_{kj}\sum_{k=1}^{i-1} w_{ki}x_k \\ B &= \sum_{k=1}^{i-1} w_{ki}+\sum_{k=j+1}^{n} w_{kj} \\ C &= \sum_{k=j+1}^{n} w_{kj}\sum_{k=1}^{i-1} w_{ki} \end{aligned}$$

Since we assume the locations of cells 1 to $i-1$ and $j+1$ to n do not change, A, B and C are constants. Thus we can obtain the partial derivative of x_j-x_i to w_{ij} as:

$$\begin{aligned} \frac{\partial(x_j-x_i)}{\partial w_{ij}} &= -\frac{A\cdot B}{(B\cdot w_{ij}+C)^2} \\ &= -\frac{(x_j-x_i)(B\cdot w_{ij}+C)\cdot B}{(B\cdot w_{ij}+C)^2} \\ &= -\frac{(x_j-x_i)\cdot B}{B\cdot w_{ij}+C} \end{aligned} \quad (18)$$

Without loss of generality, let cells i and j be the driver (source) and receiver (sink) cells of net ij, respectively. We can use the $W_{src}(ij)$ and $W_{sink}(ij)$ to substitute $\sum_{k=1}^{i-1} w_{ki}+w_{ij}$ and $\sum_{k=j+1}^{n} w_{kj}+w_{ij}$. That is, $W_{src}(ij)$ is the total weight on the driver cell i (simply the summation of net weights of those nets that intersect with the driver), and $W_{sink}(ij)$ is the total weight on the receiver cell j. Then we can rewrite B and C as:

$$\begin{aligned} B &= W_{src}(ij)+W_{sink}(ij)-2w_{ij} \\ C &= W_{src}(ij)W_{sink}(ij)-(W_{src}(ij)+W_{sink}(ij))w_{ij}+w_{ij}^2 \end{aligned}$$

Substituting B and C in (18), we have

$$\frac{\partial(x_j-x_i)}{\partial w_{ij}} = -(x_j-x_i)\frac{(W_{src}(ij)+W_{sink}(ij)-2w_{ij})}{W_{src}(ij)W_{sink}(ij)-w_{ij}^2} \quad (19)$$

[1] Cells 1 to $i-1$ and $j+1$ to n might move as w_{ij} is changed. Since the sensitivity analysis tries to catch the first order effect of net weighting, we consider each net separately and assume others do not change. The experiment results show that the estimation errors are quite small.

To simplify the notation, we just use net i to denote net ij as cell i is the driver of this net. Therefore, for any net i, we have the following important wire length versus net weight relationship.

$$S_W^L(i) = \frac{\Delta L(i)}{\Delta W(i)} = -L(i)\frac{W_{src}(i)+W_{sink}(i)-2W(i)}{W_{src}(i)W_{sink}(i)-W(i)^2} \quad (20)$$

where $L(i) = x_j - x_i$. Intuitively, (20) implies that if the initial wire length $L(i)$ is longer, for the same amount of nominal net weight change, it expects to see bigger wire length change. Meanwhile, if the initial net weight $W(i)$ is bigger, for the same amount of nominal net weight change, it expects to see a smaller amount of wire length change, since $W_{src}(i)+W_{sink}(i)-2W(i)$ is constant while $W_{src}(i)W_{sink}(i)$ is bigger for a larger $W(i)$.

In [6], a similar relationship of weight and wire length was presented by computing wire length with matrix approximation, also assuming that weighting increase of a particular net has minor effects on other nets. It is written in the following equation:

$$\Delta W(i) = \frac{-\Delta L(i)/[L(i)+\Delta L(i)]}{\frac{1}{W_{src}(i)}+\frac{1}{W_{sink}(i)}-\frac{2W(i)}{W_{src}(i)W_{sink}(i)}} \quad (21)$$

When $\Delta L(i)$ is relatively small than $L(i)$, $S_W^L(i)$ can be estimated as:

$$S_W^L(i) = -L(i)\frac{W_{src}(i)+W_{sink}(i)-2W(i)}{W_{src}(i)W_{sink}(i)} \quad (22)$$

The main difference between (22) and our model (20) is the denominator. This is mainly due to the different approximations used while deriving (20) and (22). Equation (22) is derived with the inverse coefficient matrix approximation of the quadratic objective function using two iterations of the Jacobi method [6], while our model (20) is derived directly from the necessary condition of the optimal solution, which is more intuitive than [6]. In fact, [6] did not show in detail how they obtained (22).

It can be shown that our model is also slightly more accurate than the one from [6]. We run several experiments on a circuit with 72K cells (ckt2 in Table II) to verify the accuracy of wire length sensitivity to net weight. Since the sensitivity is directly derived from the wire length estimation (17), we only need to compare the wire length estimation with the actual wire length.

Fig. 2 shows the wire length of a net in ckt2 with different net weight. L^* is the actual wire length after the quadratic placement, L' is the estimated wire length using (17), L'' is the estimated wire length using (21). The initial net weight for all the nets are 10. The initial wire length of this net is about 1150. When the net weight increases from 10 to 30, the actual wire length L^* decreases from 1150 to 550. As shown in Fig. 2, both L' and L'' track L^* quite well. However, L' is closer to the actual wire length L^* than L'', i.e., our model is more accurate.

To further validate the wire length sensitivity estimation, we randomly pick 1% of nets and increase their net weights by 10%. Then we compute the estimation error $E = (L'-L^*)/L^*$. The average estimation error is only 0.455% with a standard deviation of 6.08%. So we verify that wire length sensitivity

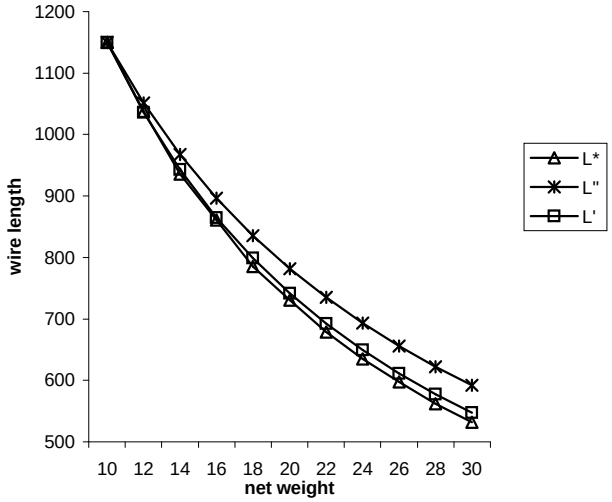


Fig. 2. The wire length of a net in ckt2 with different net weights. L^* is the actual wire length, L' is the estimated wire length using (17), L'' is the estimated wire length using (21).

TABLE I
ESTIMATION ERRORS WITH REAL NET WEIGHT DISTRIBUTION

Our Model		Tsay's Model	
Average	Variation	Average	Variation
1.8%	26%	3.2%	37%

to net weighting formula in (20) works reasonably well even with a number of nets simultaneously changing their weights. Moreover, the net weights generated by our net weighting formula presented in section V can be as large as 6 times the original weights, and 20% of nets might change their weights simultaneously. Therefore, we apply another set of experiments which randomly pick 20% of nets and increase their net weights randomly from 0 to 600% to match the real net weight distribution. Table I reports the average estimation errors and variations for our model (20) and Tsay's model (22). It shows that our model has less estimation error than Tsay's model.

Note that both models estimate the wire length and net weight relationship after a single global quadratic programming. It is not aimed to estimated the wire length after the entire placement flow, which usually includes partitioning, recursive cutting, or other heuristics. However, for sensitivity analysis, it is still reasonable to use it since the global quadratic placement roughly determines the overall cell distribution.

In the preliminary version of this paper appeared in ISPD 2004 [13], we used Tsay's model (22) to estimate wire length sensitivity to net weight. Although the new model (20) produces slightly better wire length estimation, the timing driven placement results of both models are almost the same. We present the derivation of this model in this section to demonstrate the physical meanings of wire length sensitivity to net weight.

B. Slack Sensitivity to Net Weight

The slack sensitivity to net weight is defined as:

$$S_W^{Slk}(i) = \frac{\Delta Slk(i)}{\Delta W(i)} \tag{23}$$

where $Slk(i)$ and $W(i)$ are the slack and weight of net i respectively. Since only net i is changed, the slack change of net i comes from the delay change of net i. Then,

$$S_W^{Slk}(i) = -\frac{\Delta T(i)}{\Delta W(i)} \tag{24}$$

where $\Delta T(i)$ is the nominal delay change of net i. Because smaller net delay ($\Delta T(i) < 0$) corresponds to larger slack ($\Delta Slk(i) > 0$), there is a negative sign in the above equation. Since higher net weight for net i will ideally result in shorter wire length $L(i)$, which in turn, will cause less delay, naturally we can decompose (24) into the following two terms.

$$S_W^{Slk}(i) = -S_L^T(i) S_W^L(i) \tag{25}$$

where $S_L^T(i)$ is the net delay sensitivity to wire length:

$$S_L^T(i) = \frac{\Delta T(i)}{\Delta L(i)} \tag{26}$$

We can compute $S_W^L(i)$ via (20). The remaining job is to compute $S_L^T(i)$. From (9), we can obtain for net i the delay sensitivity to its wire length change as follows:

$$S_L^T(i) = \frac{\Delta T(i)}{\Delta L(i)} = rcL(i) + cR_d + rC_l \tag{27}$$

It implies that for a given technology (fixed r and c), the delay of a long wire (larger $L(i)$) with a weak driver (larger R_d) and large capacitive load (larger C_l) will be more sensitive to the same amount of nominal wire length change (larger $S_L^T(i)$).

For nets that have multiple sinks, since wire length change of one sink may also change the delays on other sinks due to the change of the capacitance load seen by the driver, we need to evaluate the sensitivities of delays on other sinks to the wire length of this sink as well. From (12), and assuming that lengths to two different sinks j and k of the same logical net i are independent variables, we have:

$$S_{L_j}^{T_k}(i) = \frac{\Delta T_k(i)}{\Delta L_j(i)} = cR_d \tag{28}$$

where $k \neq j$, and $T_k(i)$ is the delay to sink k, $L_j(i)$ is the wire length from driver to sink j. As expected, the sensitivity in this case is only contributed through the driver. When $k = j$, it is the same as in (27).

$$S_{L_j}^{T_j}(i) = \frac{\Delta T_j(i)}{\Delta L_j(i)} = rcL_j(i) + cR_d + rC_{lj} \tag{29}$$

C. FOM Sensitivity to Net Weight

In this section, we will derive the FOM sensitivity to net weight, defined as follows:

$$S_W^{FOM}(i) = \Delta FOM / \Delta W(i) \tag{30}$$

Note that FOM improvement comes from the delay improvement of this net, (30) can be decomposed into:

$$S_W^{FOM}(i) = \frac{\Delta FOM}{\Delta T(i)} \cdot \frac{\Delta T(i)}{\Delta W(i)} \tag{31}$$

We define another sensitivity, FOM sensitivity to net delay as:

$$S_T^{FOM}(i) = \Delta FOM / \Delta T(i) \tag{32}$$

From (26), (13) and (32), $S_W^{FOM}(i)$ can be written as:

$$S_W^{FOM}(i) = S_T^{FOM}(i) S_L^T(i) S_W^L(i) \tag{33}$$

We have already shown how to compute $S_L^T(i)$ and $S_W^L(i)$ in the previous sections. In this section, we will illustrate how to compute $S_T^{FOM}(i)$. A trivial way to compute S_T^{FOM} is to run static timing analysis for the entire circuit after each $T(i)$ changed, however, this is too time consuming. If there are N nets, assuming the complexity of static timing analysis is $O(N)$, the complexity of computing all the S_T^{FOM} are $O(N^2)$. An important contribution of this work is a fast and novel algorithm to compute S_T^{FOM}. It is based on the following theorem.

Theorem 1: $S_T^{FOM}(i)$ of a two-pin net i is equal to the negative of the number of critical timing end points whose slacks are influenced by net i with a nominal $\Delta T(i)$.

Proof: Suppose there is a nominal delay change $\Delta T(i)$ on net i, it will affect the arrival time of the sink of net i by $\Delta T(i)$, and may propagate to its downstream timing points. Assume k is an immediate downstream timing point of sink j of net i. The new arrival time of k can be computed using (4). $Arr(k)$ will change if and only if $d(j,k)$ is the most critical timing arc for all timing arcs to k. For a nominal (very small) $\Delta T(i)$, the $\Delta Arr(k)$ is exactly $\Delta Arr(j)$. Continue this propagation process we can see that if any timing point m is changed, the amount of change to $Arr(m)$ will be equal to $\Delta T(i)$.

$$\Delta Arr(m) = \Delta T(i) \quad \text{if } Arr(m) \text{ is changed} \tag{34}$$

Suppose the number of critical timing end points whose arrival times will be influenced by net i is $K(i)$. It is also the number of critical timing end points whose slack will be influenced. The sensitivity of FOM to the delay change of net i is:

$$\begin{aligned} S_T^{FOM}(i) &= \sum_{m \in M} \Delta Slk(m) / \Delta T(i) \\ &= \sum_{m \in M} -\Delta Arr(m) / \Delta T(i) \\ &= -(K(i) \Delta T(i)) / \Delta T(i) \\ &= -K(i) \end{aligned} \tag{35}$$

where M is the set of timing end points whose slack will change due to the delay change of net i. This equation shows that $S_T^{FOM}(i)$ is the negative of the number of critical timing end points influenced by net i. ■

For nets with multiple sinks, we can view them as several driver-to-sink two-pin nets to do the sensitivity analysis.

Theorem 2: The FOM sensitivity of the sink j delay of net i can be computed by the following equation:

$$S_{T_j}^{FOM}(i) = -\sum_{m \in S(i)} K_m(i) \frac{S_{L_j}^{T_m}(i)}{S_{L_j}^{T_j}(i)} \tag{36}$$

where $S(i)$ is the set of sinks of net i, $K_m(i)$ is the number of influenced critical timing end points for sink m of net i.

Proof: Suppose the wire length change on net i's sink j is $\Delta L_j(i)$. This wire length change will cause the delay change on each sink of net i. From (28), we can compute the delay change on sink m due to $\Delta L_j(i)$ as:

$$\Delta T_m(i) = S_{L_j}^{T_m}(i)\Delta L_j(i) \tag{37}$$

At each sink, we can use Theorem 1. Thus, we can compute the total ΔFOM due to $\Delta L_j(i)$ as:

$$\begin{aligned}\Delta FOM &= -\sum_{m\in S(i)} K_m(i)\Delta T_m(i) \\ &= -\sum_{m\in S(i)} K_m(i) S_{L_j}^{T_m}(i)\Delta L_j(i)\end{aligned}$$

Then $S_{T_j}^{FOM}(i)$ can be derived from above equation:

$$\begin{aligned}S_{T_j}^{FOM}(i) &= \frac{\Delta FOM}{\Delta T_j(i)} \\ &= -\sum_{m\in S(i)} K_m(i) S_{L_j}^{T_m}(i)\frac{\Delta L_j(i)}{\Delta T_j(i)} \\ &= -\sum_{m\in S(i)} K_m(i)\frac{S_{L_j}^{T_m}(i)}{S_{L_j}^{T_j}(i)}\end{aligned}$$

■

To compute the number of the influenced critical timing end points $K_m(i)$ for each sink m of each net i, we have the following efficient algorithm. This algorithm can give the number of the influenced critical timing end points $K(i)$ for net i at the same time.

Algorithm 1 Counting the number of influenced timing critical end points for each sink and each net

1: initialize $K(i) = 0$ for all nets and $K_m(i) = 0$ for each sink m of net i
2: sort all nets in topological order from timing end points to timing start points
3: **for all** P_o pin t **do**
4: set $K_t(i)$ to be 1 if t is timing critical (i.e., $Slk(t) < Slk_t$); otherwise set $K_t(i)$ to be 0
5: **for all** net i in the above topologically sorted order **do**
6: **for all** sink pin j of net i **do**
7: $K(i) = K(i) + K_j(i)$
8: propagate $K(i)$ of net i to the most critical input pin l of the cell driving i; pin l is a sink of net p: $K_l(p) = K_l(p) + K(i)$; other input pins of the driver will not be propagated because they are not on the critical path of net i, thus cannot influence the timing end points from net i

Algorithm 1 backward traverses the netlist in the topological order. When it traverses through a gate, only the most timing

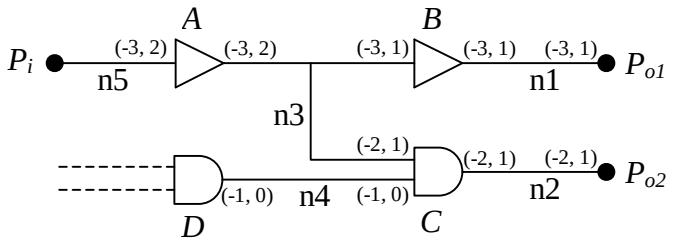


Fig. 3. Counting the number of influenced timing end points.

critical input pin[2] gets the propagated K from its downstream nets. Since each gate and net will be traversed only once, we have the following theorem for Algorithm 1:

Theorem 3: The complexity of algorithm 1 is $O(N)$, where N is the total number of nets in the design.

As an example, Fig. 3 shows two paths from a timing begin point P_i to timing end points P_{o1} and P_{o2}. In the figure notation such as (-3, 1), the first number is the slack (in ns), and the second number is the K value. Since the slacks at P_{o1} and P_{o2} are -3ns and -2ns, respectively, worse than the slack target of 0, the K values for P_{o1} and P_{o2} are both 1. We can see how the K val are propagated from PO to PI. Note that for gate C, the upper input pin has slack of -2ns, while the lower input pin has slack of -1ns, thus the upper pin is the most timing critical input pin of gate C, and will influence the slack of P_{o2}. The lower input pin of C does not influence P_{o2}, meaning that even if the wire length of net n4 is shortened, it will not improve the FOM.

IV. Sensitivity Guided Net Weighting

In this section, we will use the sensitivities derived from the previous section to guide the net weight generation for slack and FOM optimization. Again the net weighting scheme is that we start from a set of initial net weights and compute a new set of net weights that would maximize the WNS and FOM gain. Since the sensitivity analysis works best when the net weights vary little from their initial values, we also add a constant of total change to bound the net weights. We formulate the net weighting problem as the following constrained optimization problem:

$$\begin{aligned}\max_{\Delta\mathbf{W}} \quad & \textstyle\sum_{i=n_1}^{i=n_k}[(Slk_t - Slk(i))\Delta Slk(i) + \alpha\Delta FOM(i)] \\ s.t. \quad & \textstyle\sum_{i=n_1}^{i=n_k}[\Delta W(i)]^2 \le C\end{aligned} \tag{38}$$

where n_1,...,n_k are critical nets, $\Delta\mathbf{W} = \{\Delta W(i)\}$, C is a constant to bound the total weight change. The multiplier for $\Delta Slk(i)$ is its relative slack to the slack target Slk_t, since we want more $\Delta Slk(i)$ for more critical nets. The constant α on each ΔFOM is the same, which is used to balance the FOM and slack. The quadratic sum constraint of $\Delta W(i)$ helps to produce smooth

[2]When there are two or more input pins with the same negative slack, the K value propagated by this algorithm might be not exactly equal to the influenced timing end points. This is because those two or more input pins might or might not come from the same source net. Our experiments show that either propagate for all the most critical pins or not does not make much difference on the final timing result. Therefore, if there are multiple most critical pins with the same slack, we do not propagate K for any of them.

distribution of net weights. Replacing $\Delta Slk(i)$ and $\Delta FOM(i)$ with $S_W^{Slk}(i)$, $S_W^{FOM}(i)$ and $\Delta W(i)$, we have:

$$\begin{array}{ll} \max\limits_{\Delta \mathbf{W}} & \sum_{i=n_1}^{i=n_k}[(Slk_t - Slk(i))S_W^{Slk}(i) + \alpha S_W^{FOM}(i)]\Delta W(i) \\ s.t. & \sum_{i=n_1}^{i=n_k}[\Delta W(i)]^2 \le C \end{array} \quad (39)$$

We can use *Lagrange multiplier* method to solve this nonlinear programming problem. Let

$$\begin{aligned} L(\Delta \mathbf{W}, \lambda) &= \sum_{i=n_1}^{i=n_k}[(Slk_t - Slk(i))S_W^{Slk}(i) + \alpha S_W^{FOM}(i)] \cdot \Delta W(i) \\ &\quad + \lambda \cdot (C - \sum_{i=n_1}^{i=n_k}[\Delta W(i)]^2) \end{aligned} \quad (40)$$

where λ is a non-negative Lagrange multiplier. The solution $\Delta \mathbf{W}^*$ and λ^* should satisfy:

$$\left\{ \begin{array}{lcl} \frac{\partial L(\Delta \mathbf{W}, \lambda)}{\partial \Delta W(i)}(\Delta \mathbf{W}^*, \lambda^*) &=& 0 \\ \frac{\partial L(\Delta \mathbf{W}, \lambda)}{\partial \lambda}(\Delta \mathbf{W}^*, \lambda^*) &=& 0 \end{array} \right. \text{ for each net } i \in (n_1, ..., n_k) \quad (41)$$

Thus we have

$$\Delta W^*(i) = \beta\{[Slk_t - Slk(i)]S_W^{Slk}(i) + \alpha S_W^{FOM}(i)\} \quad (42)$$

where,

$$\beta = \sqrt{\frac{C}{\sum_{i=n_1}^{i=n_k}[(Slk_t - Slk(i))S_W^{Slk}(i) + \alpha S_W^{FOM}(i)]^2}} \quad (43)$$

is a constant for all nets, which absorbs the effect of C and determines how much weight change is allowed. The other constant parameter α balances the weighting of critical slack and FOM. In the real implementation, we also linearly scale $(Slk_t - Slk(i))S_W^{Slk}(i)$ and $S_W^{FOM}(i)$ to $[0,1]$ in order to precisely control the weighting scale via α and β.

Based on (42), we propose the following sensitivity guided net weighting scheme

$$W(i) = \left\{ \begin{array}{ll} W_{org}(i) & Slk(i) > Slk_t \\ W_{org}(i) + \Delta W^*(i) & Slk(i) \le Slk_t \end{array} \right. \quad (44)$$

where $W_{org}(i)$ is the original net weight, $\Delta W^*(i)$ is net weight adjustment from (42).

Algorithm 2 An example of timing driven placement flow using sensitivity guided net weighting

1: run wire length driven placement with uniform weight W_{min}, i.e., $W_{org}(i) = W_{min}$ for all nets
2: run static timing analysis
3: compute S_W^{FOM}, S_W^{Slk} for each net
4: compute weight $W(i)$ for each net i based on (44)
5: run timing driven placement with new net weight

V. Experimental Flow and Results

A. Experimental Flow and Setup

Our sensitivity-based net weighting algorithm can be used to guide timing driven placement, by either iteratively updating net weights gradually (e.g., using very small α and β parameters) or generating a set of new net weights in one shot. Iteratively updating net weights might get us the best results, but it requires many placement, timing analysis, and net weighting runs. It may take too much run time for modern large-scale ASIC chips, with hundreds of thousands to millions of placeable objects. In Algorithm 2, we show an example of a practical, industrial strength timing driven placement flow which only runs global placement twice and generates sensitivity-based net weights once. This flow is used in our experiments since we are mostly interested in large designs.

It shall be pointed that many timing-driven quadratic placement engine uses a clique model for multiple-pin nets. Then each edge of a net shares the same net weight.[3] We can still compute the sensitivities for each edge, then assign a net weight to the entire net. An alternative approach is to model the multiple-pin net as a lumped net. Instead of decomposing a multiple-sink net into a set of edges when computing the sensitivities, we use a lumped, single sink net to approximate the net weight in our experiments. The wire length of this lumped net is the half perimeter length of the bounding box of the original multi-sink net. The sink of the new net is the one with the worst slack in the original net because what matters most is the most critical sink. Since most nets in real designs have only one or two sinks, the half perimeter length of the bounding box can approximate the total wire length reasonably accurately. From Algorithm 1, the influenced timing end points for each multiple-pin net is simply the summation of that for its sinks. Note that the lumped net approximation is only used for computing net weight sensitivities. It is not used for the static timing analysis to obtain the slack for each net and pin.

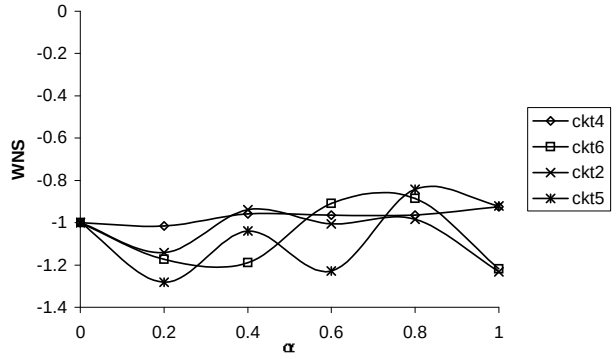
The net weighting algorithm is implemented in C++ language and tested on the IBM AIX 43P-S85 servers. The placement tool used in our timing driven placement flow is the IBM CPlace [22]. CPlace has been used in the design and production of hundreds of ASIC chips and several microprocessors. It includes several placement engines. In our experiment, we uses the quadratic placement engine called QPS. The placement result of this engine is relatively stable compared to the pure partition-based engine. CPlace is also integrated with the IBM Placement Driven Synthesis design closure tool. Instead of using the old MCNC or ISPD'98 benchmarks, we test our algorithm on a set of real industry circuits (ASIC chips and cores), with circuit size up to 444K placeable cells using IBM CMOS technologies [23]. We use a state-of-the-art static timing analyzer Einstimer from IBM to perform the timing report. The test circuit characteristics are summarized in Table II. The slack target Slk_t is set to be 0.3ns for all circuits in our experiments. We compare the following four algorithms:

- WL: **w**ire **l**ength driven placement with uniform weight
- TS: **t**iming driven placement using **s**lack
- TSS: **t**iming driven placement using **s**lack **s**ensitivity
- TSF: **t**iming driven placement using both **s**lack and **F***OM*

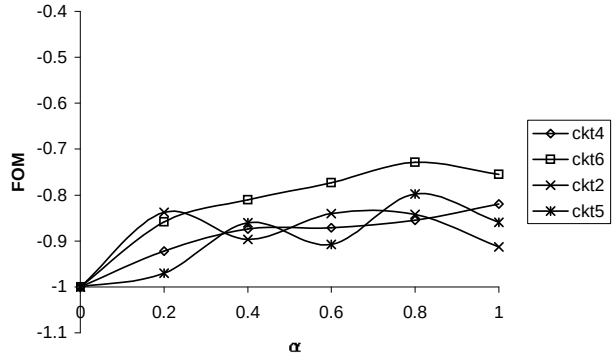
[3] With a clique model, one may still be able to add additional edge-based net weighting by creating artificial two-pin nets between the driver and its sinks. But it would work better in an incremental placement and net weighting flow. Since our flow only runs global placement twice and net weighting once, we do not add these artificial two-pin nets.

TABLE II
TESTCASE SIZE AND TECHNOLOGY

Design	cells	nets	technology
ckt1	57K	58K	0.13um
ckt2	72K	64K	0.18um
ckt3	159K	157K	0.25um
ckt4	216K	203K	0.25um
ckt5	252K	257K	0.18um
ckt6	303K	328K	0.18um
ckt7	444K	395K	0.18um



(a) *WNS*



(b) *FOM*

Fig. 4. *WNS* and *FOM* after timing driven placement with different α.

sensitivity

All our placement results are legal, i.e., there is no cell overlapping. We report two set of results, one is from timing driven placement alone, and the other is from physical synthesis after the timing driven placement to show that it is important to have a good placement starting point for physical synthesis to work on.

B. Timing Driven Placement

Based on timing driven placement flow described in Algorithm 2, we first run CPlace with $W_{min} = 10$. Then we compute the S_W^{Slk} and $_W^{FOM}$ for each net. We also run Einstimer to perform the static timing analysis and get the slack for each net. Before we generate net weights using (44), we need to select α and β. After a set of experiments, we set $\beta = 60$, i.e., the maximum ΔW generated by slack sensitivity will be 60. To evaluate the impact of FOM sensitivity, we run net weighting algorithm with different α ranging from 0 to 1. Figs. 4(a) and 4(b) show the WNS and FOM of a set of testcases after the timing driven placement with net weights generated by different α. The FOM and WNS are scaled to -1 for the comparison among different testcases. As shown in the figure, WNS does not have a consistent trend, but FOM gets better as α increases from 0 to 0.8 for most testcases. Thus adding FOM sensitivity helps to achieve better FOM. We use $\alpha = 0.8$ for TSF in the rest experiments.

As a reference, we also report the result from zero wire model (ZW), i.e., assuming zero wire resistance and capacitance. The timing under ZW model is the best timing that any timing driven placement algorithm can possibly achieve. Furthermore, we run a simple slack based net weighting algorithm (TS) to compare our result with. This net weighting algorithm linearly assigns net weights based on net slacks. To ensure a fair comparison, we make TS generate the same average weight as TSS does.

Table III compares the FOM and WNS results from ZW, WL, TS, TSS, and TSF. Since we can not compare directly with other timing driven placement algorithms due to the uniqueness of our flow, we report under the improvement columns for TS, TSS and TSF in these tables the *optimization potential* over WL relative to ZW, which is also used in [12]. The *optimization potential* is defined as the percentage of timing improvement by timing driven placement versus wire length driven placement, compared to an upper bound for such improvement. The timing (WNS or FOM) difference of the zero-wire load model versus the wire length driven placement is used as an upper bound. For example in Table III, the improvement potential of TSS for $ckt1$ can be computed by $(41650-26093)/(41650-9134) = 48\%$. We can see that algorithm TSS and TSF improve FOM and WNS by a large margin (on average from 37% to 58%) compared with uniform net weighting (i.e. wire length driven) placement WL. The algorithm TSF (with both slack and FOM sensitivities) further improves the TSS (with only slack sensitivity) results, from 49% to 58% for FOM. The WNS also gets slight improvement from 37% to 40% using the TSF algorithm. Compared to the pure slack based algorithm TS, TSS and TSF have better FOM and WNS improvement. Actually the FOM optimization of TS is just slightly worse than TSS, however, the average WNS of TS is significantly worse. The degradation is mainly caused by some long wire delays in the largest design $ckt7$. Since TS does not use sensitivity to guide net weighting, the low driving strengths of the drivers of those wire are not considered. Therefore TS produces inferior results than those of TSS and TSF.

Note that we did not report the timing improvement in terms of cycle time. This is because the chips we tested all have multiple clock signals with different cycles times. If we only use the master clock for comparison, the average improvement of cycle time for these circuits is 13%.

In a recent paper, [12] reported an average TNS (total negative slack, which is a special case of our FOM when the slack threshold is zero) improvement of 47.6% and WNS

TABLE III

FOM AND *WNS* COMPARISON AFTER TIMING DRIVEN PLACEMENT.

Design	*FOM*(*ns*)					Improvement			*WNS*(*ns*)					Improvement		
	ZW	*WL*	*TS*	*TSS*	*TSF*	*TS*	*TSS*	*TSF*	*ZW*	*WL*	*TS*	*TSS*	*TSF*	*TS*	*TSS*	*TSF*
ckt1	-9134	-41650	-28295	-26093	-25602	41%	48%	49%	-1.702	-6.274	-3.720	-3.392	-4.254	55%	63%	44%
ckt2	0	-6966	-3359	-4102	-3454	52%	41%	50%	0.248	-2.977	-1.375	-1.784	-1.754	50%	37%	38%
ckt3	-535	-13711	-6466	-6468	-5595	55%	55%	62%	-0.55	-4.997	-3.474	-3.684	-3.788	34%	30%	27%
ckt4	-322	-8057	-3673	-4024	-3440	57%	52%	60%	-0.941	-7.218	-5.112	-3.736	-3.605	34%	55%	58%
ckt5	-114	-28527	-13826	-15334	-12229	52%	46%	57%	-0.102	-3.575	-2.418	-2.379	-2.002	33%	34%	45%
ckt6	-142	-20257	-14200	-9417	-9536	30%	54%	53%	-0.508	-5.47	-4.135	-5.484	-4.856	27%	-0%	12%
ckt7	-4	-452	-243	-248	-131	47%	46%	72%	0.16	-1.135	-2.351	-0.66	-0.432	-94%	37%	54%
Average						48%	49%	58%						20%	37%	40%

TABLE IV

TOTAL WIRE LENGTH COMPARISON AFTER PLACEMENT

Design	*TWL* ($\times 10^6$)				change		
	WL	*TS*	*TSS*	*TSF*	*TS*	*TSS*	*TSF*
ckt1	10.30	11.24	10.89	11.10	9.14%	5.79%	7.86%
ckt2	14.93	16.20	15.87	16.54	8.50%	6.28%	10.78%
ckt3	40.05	42.59	41.04	42.41	6.35%	2.49%	5.91%
ckt4	49.44	50.20	49.59	50.07	1.52%	0.30%	1.26%
ckt5	59.98	63.92	64.39	63.59	6.57%	7.36%	6.02%
ckt6	134.64	136.03	136.01	135.96	1.03%	1.01%	0.98%
ckt7	126.60	127.07	126.22	126.34	0.37%	-0.30%	-0.21%
Average					4.78%	3.28%	4.66%

improvement of 63.6%. Since we have no access to those test circuits in [12], we cannot make direct comparison with those numbers. Also, it should be noted that our test circuits are significantly bigger than those used in [12] (the largest circuit in [12] is only 6K, while ours is over 440K). Yet it is interesting to observe that our *TSF* algorithm gets 58% *FOM* improvement with a single non-iterative net weighting (as opposed to [12] which iteratively updates placement and net weighting).

Table IV compares the total wire length (TWL) from the four algorithms *WL*, *TS*, *TSS* and *TSF*. We can see that *TSS* and *TSF* only increase TWL by a small percentage, and they are better than *TS*.

C. Post Physical Synthesis Result

For deep submicron timing closure, tremendous amount of optimizations such as buffer insertion, gate sizing, pin swapping will be done after placement [19][18]. A good timing driven placement should provide a good starting point for the follow-on physical synthesis. We run an industry physical synthesis tool PDS [17][16] to further improve *WNS* and *FOM* based on the placement results from *WL*, *TSS* and *TSF* algorithms. We did not run *TS* again because *TSS* and *TSF* are the best results we have after TDP.

Table V compares the *FOM* and *WNS* after PDS for algorithms *WL*, *TSS* and *TSF*. Again, we see a consistent significant improvement of *TSS* and *TSF* over *WL*. The explicit *FOM* guided algorithm *TSF* still has the best *FOM* after PDS (on average 7% better than the improvement by *TSS* over *WL*). Note that the *WNS* improvement of *TSF* after PDS is slightly smaller than that of *TSS*, 45% vs. 47%, while the *WNS* improvement of *TSF* after placement is higher than that of *TSS*. It shows that a placement with better *WNS* does not necessarily end up with better *WNS* after PDS. But the placements with better *FOM* in general still have better *FOM* after PDS. This demonstrates the importance of optimizing *FOM* explicitly during the placement.

Table VI shows the total wire length of each circuit after PDS. It can be seen that the wire length difference becomes smaller compared to Table IV after PDS. So we are able to achieve significant improvement in timing with little degradation in the wire length metric. It also shows the average total wire length of *TSF* is only 2 percent worse that of *TSS*, which means timing driven placement with *FOM* sensitivity trades off little wire length for a much better *FOM*.

TABLE VI

TOTAL WIRE LENGTH COMPARISON AFTER PHYSICAL SYNTHESIS

Design	*TWL* ($\times 10^6$)			change	
	WL	*TSS*	*TSF*	*TSS*	*TSF*
ckt1	10.55	11.14	11.34	5.59%	7.46%
ckt2	15.23	16.07	16.78	5.53%	10.21%
ckt3	56.24	57.15	58.99	1.62%	4.89%
ckt4	49.62	49.70	50.19	0.16%	1.14%
ckt5	60.06	64.42	63.60	7.27%	5.90%
ckt6	144.97	146.16	146.38	0.82%	0.98%
ckt7	133.17	126.12	133.65	-5.30%	0.36%
Average				2.24%	4.42%

Table VII compares the total cell area after PDS for algorithms *WL*, *TSS* and *TSF*. It shows that the total cell area difference is negligible among these three algorithms. In fact, *TSS* and *TSF* even have slightly smaller area than *WL*, for example of *ckt*1, which has a 1.3 percent area reduction. So a better placement starting point may need less aggressive gate sizing.

TABLE VII

TOTAL CELL AREA COMPARISON AFTER PHYSICAL SYNTHESIS

Design	*Area* ($\times 10^5$)		
	WL	*TSS*	*TSF*
ckt1	6.85	6.76	6.76
ckt2	11.50	11.46	11.47
ckt3	24.79	24.84	24.82
ckt4	153.26	153.20	153.21
ckt5	37.08	36.98	36.98
ckt6	269.31	269.20	269.17
ckt7	99.87	99.78	99.83

VI. CONCLUSIONS

In this paper, we first derive a set of sensitivity analysis for wire length, slack and *FOM* due to a nominal change of net weighting. We then propose a new net weighting scheme that incorporates both slack and *FOM* sensitivities. The net weighting algorithm is implemented in an industrial strength timing driven placement and physical synthesis flow. Experimental results show by adding slack and *FOM* sensitivities, we are able to obtain better results for not just timing-driven placement (58% in *FOM* and 40% in *WNS*), but also the

TABLE V

FOM AND *WNS* COMPARISON AFTER PHYSICAL SYNTHESIS.

Design	*FOM*(*ns*)			Improvement		*WNS*(*ns*)			Improvement	
	WL	*TSS*	*TSF*	*TSS*	*TSF*	*WL*	*TSS*	*TSF*	*TSS*	*TSF*
ckt1	-7829	-6086	-5170	22%	34%	-0.834	-0.743	-0.739	11%	11%
ckt2	-2059	-384	-631	81%	69%	-0.705	-0.011	-0.073	98%	90%
ckt3	-1854	-405	-422	78%	77%	-0.701	-0.139	-0.19	80%	73%
ckt4	-2537	-1844	-1770	27%	30%	-2.156	-1.908	-1.9	12%	12%
ckt5	-4732	-2726	-1819	42%	62%	-0.472	-0.443	-0.341	6%	28%
ckt6	-1481	-541	-266	63%	82%	-0.36	-0.293	-0.351	19%	3%
ckt7	-94	-8	0	91%	100%	-0.097	0	0	100%	100%
Average				58%	65%				47%	45%

physical synthesis optimization after it (65% in *FOM* and 45% in *WNS*). Adding the *FOM* sensitivity to guide the net weight generation, we can further improve the *FOM* without deteriorating the worst slack and wire length.

Since physical synthesis transforms such as buffering and gate sizing could change the timing of a netlist significantly, we plan to consider their impact on net weighting explicitly in the future.

VII. ACKNOWLEDGMENT

The authors would like to thank Chuck Alpert, Ruchir Puri, Lakshmi Reddy, Louise Trevillyan, and Paul Villarrubia at IBM Corporation for their helpful discussions.

REFERENCES

[1] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, 2003, http://public.itrs.net/.

[2] M. A. B. Jackson and E. S. Kuh, "Performance-driven placement of cell based ic's," in *Proc. Design Automation Conf.*, pp. 370–375, 1989.

[3] W. E. Donath, R. J. Norman, B. K. Agrawal, S. E. Bello, S. Y. Han, J. M. Kurtzberg, P. Lowy, and R. I. McMillan, "Timing driven placemeng using complete path delays," in *Proc. Design Automation Conf.*, pp. 84–89, 1990.

[4] A. Srinivasan, K. Chaudhary, and E. S. Kuh, "Ritual: A performance driven placement algorithm for small cell ics," in *Proc. Int. Conf. on Computer Aided Design*, pp. 48–51, 1991.

[5] T. Gao, P. M. Vaidya, and C. L. Liu, "A performance driven macro-cell palcement algorithm," in *Proc. Design Automation Conf.*, pp. 147–152, 1992.

[6] R. S. Tsay and J. Koehl, "An analytic net weighting approach for performance optimization in circuit placement," in *Proc. Design Automation Conf.*, pp. 636–639, 1991.

[7] B. M. Riess and G. G. Ettelt, "SPEED: fast and efficient timing driven placement," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 377–380, 1995.

[8] M. Marek-Sadowska and S. P. Lin, "Timing driven placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 94–97, 1989.

[9] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. Design Automation Conf.*, pp. 269–274, 1998.

[10] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," in *ACM Symposium on FPGAs*, pp. 203–213, 2000.

[11] S. Ou and M. Pedram, "Timing-driven placement based on partitioning with dynamic cut-net control," in *Proc. Design Automation Conf.*, pp. 472–476, 2000.

[12] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary, and B. Halpin, "Timing driven force directed placement with physical net constraints," in *Proc. Int. Symp. on Physical Design*, pp. 60–66, Apr. 2003.

[13] H. Ren, D. Z. Pan and D. S. Kung, "Sensitivity guided net weighting for placement driven synthesis," in *Proc. Int. Symp. on Physical Design*, pp. 10–17, Apr. 2004.

[14] J. Kleinhans, G. Sigl, F. M. Johannes, and K. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-10, pp. 356–365, Mar. 1991.

[15] T. Kong, "A novel net weighting algorithm for timing-driven placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 172–176, 2002.

[16] L. Trevillyan, D. Kung, R. Puri, L. N. Reddy, and M. A. Kazda,"An integrated environment for technology closure of deep-submicron IC designs," in *IEEE Design & Test of Computers*, vol. 21, pp. 14–22, Jan.-Feb. 2004.

[17] W. Donath, P. Kudva, L. Stok, P. Villarrubia, L. Reddy, A. Sullivan, and K. Chakraborty, "Transformational placement and synthesis," in *Proc. Design, Automation and Test in Eurpoe*, Mar. 2000.

[18] D. S. Kung, "A fast fanout optimization algorithm for near-continuous buffer libraries," in *Proc. Design Automation Conf.*, pp. 352–355, 1998.

[19] C. J. Alpert, A. Devgan, and S. Quay, "Buffer insertion for noise and delay optimization," in *Proc. Design Automation Conf.*, pp. 362–7, June 1998.

[20] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *Journal of Applied Physics*, vol. 19, pp. 55–63, Jan. 1948.

[21] 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.

[22] P. Villarrubia, G. Nusbaum, R. Masleid, and P. T. Patel, "IBM RISC chip design methodology," in *Proc. IEEE Int. Conf. on Computer Design*, pp. 143–147, 1989.

[23] *http://www-306.ibm.com/chips/services/foundry/technologies/cmos.html*.