Sensitivity Guided Net Weighting for Placement Driven Synthesis

Haoxing Ren
ECE Department, UT Austin
& IBM Corporation
Austin, TX 78758
haoxing@us.ibm.com

David Z. Pan
ECE Department
University of Texas at Austin
Austin, TX 78712
dpan@ece.utexas.edu

David S. Kung
IBM T.J. Watson Research
Center
Yorktown Heights, NY 10598
kung@us.ibm.com

ABSTRACT

Net weighting is a key technique in large scale timing driven placement, which plays a crucial role for deep submicron physical synthesis and timing closure. A popular way to assign net weight is based on the slack of the nets, trying to minimize the worst negative slack (WNS) for the entire circuit. 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. In this paper, we perform a comprehensive analysis of the slack and FOM sensitivities to the net weight, and propose a new net weighting scheme based on the slack and FOM sensitivities. Such sensitivity analysis implicitly takes potential physical synthesis effect into consideration. Experiment results on a set of industrial circuits are promising for both stand-alone timing driven placement and physical synthesis afterwards.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuit, Design Aids**]: Placement and Routing

General Terms

Algorithms, Design, Performance

Keywords

Timing Driven Placement, Physical Synthesis, Net Weight, Interconnect, Sensitivity Analysis

1. INTRODUCTION

As the very large scale integrated (VLSI) circuits are scaled into nanometer dimensions and operate in giga-hertz frequencies, it is well understood that the interconnect has become the dominant factor in determining the overall circuit performance and complexity. The global wiring delay can easily be a factor of ten or hundred times of a logic gate delay, even with repeater insertion [1]. 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). While minimizing the total wire length is important (as done by the traditional placement), timing driven placement also attempts to minimize the wire length that are on the critical paths so that the interconnect delays on the 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]. The net weights may be iteratively updated for multiple placement runs [7] [9] [11] [12]. However, in a modern ASIC physical synthesis flow that needs to deal with millions of placeable cells, global placement is usually performed only two or three times for acceptable turn around time.

Therefore, an effective net weight assignment will be critical to the success of the physical synthesis flow. A popular way to assign net weight is based on the slack of the net, which aims to minimize the worst negative slack (WNS) for the entire circuit [8][13] [10] [14]. 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 2 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 previous work uses the total negative slack (TNS) to measure the quality of timing driven placement. TNS is a special case of FOM with zero slack target. But [12] did not use the TNS during placement step to guide it. In this paper, we explicitly use FOM metric to guide the placement.

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 [15] [16] [17]. 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. Higher net weights

ISPD'04, April 18–21, 2004, Phoenix, Arizona, USA.

for timing critical nets ideally lead to shorter wire length, less delay and better FOM. However, it is not clear how much weight change a critical net shall have and what is its potential impact on the slack and FOM.

This paper presents a comprehensive sensitivity analysis on the impact of net weight to slack and FOM. Based on these sensitivities, we propose a new net weighting algorithm with consideration of both FOM and slack sensitivities. 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 2 describes background information on the quadratic placement, static timing analysis and delay models used to illustrate the sensitivity analysis. Section 3 derives the slack and FOM sensitivities to net weight. Section 4 presents a new net weighting algorithm based on the sensitivity analysis in section 3. The experimental placement flow and results are shown in section 5, followed by the conclusion in section 6.

2. PRELIMINARIES

In this section, we give the preliminaries for timing driven placement and use a hybrid quadratic programming and partitioning approach [13] [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, and can be solved by quadratic programming techniques. Following each quadratic solution, cells will be partitioned and assigned into smaller bins, with an optional repartitioning step to further improve the result. The quadratic programming, partitioning and repartitioning process will 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 will be assigned to timing critical nets. A static timing analysis engine will be called to identify such nets. For each timing point t, its arrival time $Arr(t)$, required arrival time $Req(t)$, and slack $Slk(t)$ can be computed as follows:

$$Arr(t) = \begin{cases} T_i(t) & t \in P_i \\ max_{(s,t)\in E}\{Arr(s)+d(s,t)\} & otherwise \end{cases} \quad (1)$$

where E is the set of timing arcs, $d(s,t)$ is the delay between timing point s and 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.

$$Req(t) = \begin{cases} T_o(t) & t \in P_o \\ min_{(t,s)\in E}\{Req(s)-d(t,s)\} & otherwise \end{cases} \quad (2)$$

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.

$$Slk(t) = Req(t) - Arr(t) \quad (3)$$

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.

$$FOM = \sum_{Slk(t)<Slk_t}^{t\in P_o} (Slk(t)-Slk_t) \quad (4)$$

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 their 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 [18] 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 [19] 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

$$T = R_d(C_w+C_l)+C_lR_w+C_wR_w/2 \quad (5)$$

Let the unit length wire resistance and capacitance be r and c respectively. Then $R_w = rL$, $C_w = cL$, and (5) can be rewritten as:

$$T = \frac{rc}{2}L^2+(cR_d+rC_l)L+R_dC_l \quad (6)$$

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.

$$T_j = R_dC_{total}+rC_lL_j+\frac{rc}{2}L_j^2 \quad (7)$$

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. For example of a net with two sinks, the delay from the source to one of the sinks can be estimated as:

$$T_1 = R_d(C_{w1}+C_{l1}+C_{w2}+C_{l2})+C_{l1}R_{w1}+C_{w1}R_{w1}/2 \quad (8)$$

where T_1 is the delay from the source to sink 1. C_{w1},R_{w1},C_{l1} and C_{w2},R_{w2},C_{l2} are the capacitance, resistance and load of wire segment 1 and 2, respectively. Here we assume direct wiring from source to each sink, because we do not know the actual wiring topology at this point. Similarly we can write (8) as :

$$T_1 = \frac{rc}{2}L_1^2+(cR_d+rC_{l1})L_1+cR_dL_2+R_d(C_{l1}+C_{l2}) \quad (9)$$

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

3. NET WEIGHT SENSITIVITY ANALYSIS

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

3.1 Slack Sensitivity to Net Weight

The slack sensitivity to net weight is defined as:

$$S_W^{Slk}(i) = \frac{\Delta Slk(i)}{\Delta W(i)} \tag{10}$$

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,

$$S_W^{Slk}(i) = -\frac{\Delta T(i)}{\Delta W(i)} \tag{11}$$

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 Eqn. (11) into the following two terms.

$$S_W^{Slk}(i) = -S_L^T(i) S_W^L(i) \tag{12}$$

where $S_L^T(i)$ is the net delay sensitivity to wire length, and $S_W^L(i)$ is the wire length sensitivity to net weight:

$$S_L^T(i) = \frac{\Delta T(i)}{\Delta L(i)} \tag{13}$$

$$S_W^L(i) = \frac{\Delta L(i)}{\Delta W(i)} \tag{14}$$

where $L(i)$ is the wire length for net i. From (6), we can obtain for net i the delay sensitivity to its wire length change as follows:

$$S_L^T(i) = \frac{\Delta T(i)}{\Delta L(i)} = rcL(i) + cR_d + rC_l \tag{15}$$

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 (9), and assuming that lengths to two different sinks j and k of the same logical net i are independent variables, we have:

$$S_{L_j}^{T_k}(i) = \frac{\Delta T_k(i)}{\Delta L_j(i)} = cR_d \tag{16}$$

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 (15).

$$S_{L_j}^{T_j}(i) = \frac{\Delta T_j(i)}{\Delta L_j(i)} = rcL_j(i) + cR_d + rC_{lj} \tag{17}$$

To derive the wire length sensitivity to net weight change $S_W^L(i)$, we use the result from [6] to estimate the relationship of weight and wire length, which can be written in the following equation:

$$\Delta W(i) = \frac{-\Delta L(i)/[L(i) + \Delta L(i)]}{\frac{1}{W_{src}(i)} + \frac{1}{W_{sink}(i)} - \frac{2W(i)}{W_{src}(i)W_{sink}(i)}} \tag{18}$$

where $L(i)$ is the initial wire length of net i, $W(i)$ is the initial weight of net i, $W_{src}(i)$ is the total initial weight on the driver cell of net i (simply the summation of net weights of those nets that intersect with the driver), and $W_{sink}(i)$ is the total initial weight on the receiver cell of net i. For sensitivity analysis, $\Delta L(i)$ is very small compared to $L(i)$. We can then rewrite (18) as:

$$\Delta W(i) = \frac{-\Delta L(i)/L(i)}{\frac{1}{W_{src}(i)} + \frac{1}{W_{sink}(i)} - \frac{2W(i)}{W_{src}(i)W_{sink}(i)}} \tag{19}$$

Substituting (19) into (14) yields

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

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 $W(i)$.

3.2 FOM Sensitivity to Net Weight

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

$$S_W^{FOM}(i) = \Delta FOM/\Delta W(i) \tag{21}$$

Note that FOM improvement comes from the delay improvement of this net, equation (21) can be decomposed into:

$$S_W^{FOM}(i) = \frac{\Delta FOM}{\Delta T(i)} \cdot \frac{\Delta T(i)}{\Delta W(i)} \tag{22}$$

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

$$S_T^{FOM}(i) = \Delta FOM/\Delta T(i) \tag{23}$$

From (13), (14) and (23), $S_W^{FOM}(i)$ can be written as:

$$S_W^{FOM}(i) = S_T^{FOM}(i) S_L^T(i) S_W^L(i) \tag{24}$$

We have already shown how to compute $S_L^T(i)$ and $S_W^L(i)$ in the previous subsection. In this subsection, 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 (1). $Arr(k)$ 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 $\Delta T(i)$.

$$\Delta Arr(m) = \Delta T(i) \quad \text{if } Arr(m) \text{ is changed} \tag{25}$$

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:

$$\begin{aligned} S_T^{FOM}(i) &= \sum_{m\in M} \Delta Slk(m)/\Delta T(i) \\ &= \sum_{m\in M} -\Delta Arr(m)/\Delta T(i) \\ &= -(K(i)\Delta T(i))/\Delta T(i) \\ &= -K(i) \end{aligned} \tag{26}$$

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 point 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:*

$$S_{T_j}^{FOM}(i) = -\sum_{m\in S(i)} K_m(i)\frac{S_{L_j}^{T_m}(i)}{S_{L_j}^{T_j}(i)} \tag{27}$$

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 (16), we can compute the delay change on sink m due to $\Delta L_j(i)$ as:

$$\Delta T_m(i) = S_{L_j}^{T_m}(i)\Delta L_j(i) \tag{28}$$

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

$$\begin{aligned} \Delta FOM &= -\sum_{m\in S(i)} K_m(i)\Delta T_m(i) \\ &= -\sum_{m\in S(i)} K_m(i) S_{L_j}^{T_m}(i)\Delta L_j(i) \end{aligned} \tag{29}$$

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

$$\begin{aligned} S_{T_j}^{FOM}(i) &= \frac{\Delta FOM}{\Delta T_j(i)} \\ &= -\sum_{m\in S(i)} K_m(i) S_{L_j}^{T_m}(i)\frac{\Delta L_j(i)}{\Delta T_j(i)} \\ &= -\sum_{m\in S(i)} K_m(i)\frac{S_{L_j}^{T_m}(i)}{S_{L_j}^{T_j}(i)} \end{aligned} \tag{30}$$

□

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 backward traverses the netlist in the topological order. When it traverses through a gate, only the most timing critical input pin gets the propagated K from its downstream netw. Since each gate and net will be traversed only once, we have the following theorem for Algorithm 1:

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

```
initialize $K(i) = 0$ for all nets and $K_m(i) = 0$ for each sink $m$ of net $i$
sort all nets in topological order from timing end points to timing start points
for all $P_o$ pin t do
    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
for all net $i$ in the above topologically sorted order do
    for all sink pin $j$ of net $i$ do
        $K(i) = K(i) + K_j(i)$
    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 $i$
```

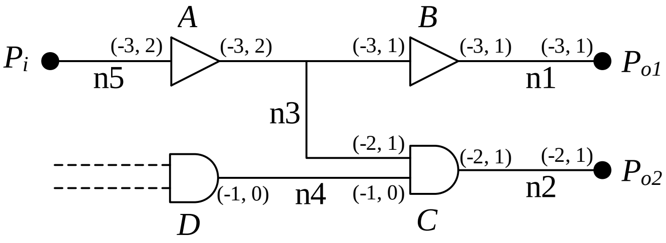

Figure 1: Counting the number of influenced timing end points.

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

As an example, Figure 1 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 values 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 pin to gate C, and will influence the slack of P_{o2}. The lower 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.

4. SENSITIVITY GUIDED NET WEIGHT ASSIGNMENT

In this section, we will use the sensitivities derived from the previous section to guide the net weight generation for slack and FOM optimization. To balance the slack and FOM, we formulate the net weighting problem as the following constrained optimization problem:

$$\begin{aligned} \max_{\Delta\mathbf{W}} \quad & \textstyle\sum_{i=n_1}^{i=n_k}[(Slk_t - Slk(i))\Delta Slk(i) + \alpha\Delta FOM(i)] \\ s.t. \quad & \textstyle\sum_{i=n_1}^{i=n_k}[\Delta W(i)]^2 \leq C \end{aligned} \tag{31}$$

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 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:

$$\begin{array}{ll} \max\limits_{\Delta\mathbf{W}} & \Sigma_{i=n_1}^{i=n_k}[(Slk_t - Slk(i))S_W^{Slk}(i) + \alpha S_W^{FOM}(i)]\Delta W(i) \\ s.t. & \qquad\qquad \Sigma_{i=n_1}^{i=n_k}[\Delta W(i)]^2 \le C \end{array} \tag{32}$$

Let

$$\begin{aligned} L(\Delta\mathbf{W},\lambda) &= \sum_{i=n_1}^{i=n_k}[(Slk_t - Slk(i))S_W^{Slk}(i) + \alpha S_W^{FOM}(i)] \cdot \Delta W(i) \\ &\quad + \lambda \cdot (C - \sum_{i=n_1}^{i=n_k}[\Delta W(i)]^2) \end{aligned} \tag{33}$$

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

$$\left\{ \begin{array}{lll} \frac{\partial L(\Delta\mathbf{W},\lambda)}{\partial \Delta W(i)}(\Delta\mathbf{W}^*,\lambda^*) & = & 0 \\ \frac{\partial L(\Delta\mathbf{W},\lambda)}{\partial\lambda}(\Delta\mathbf{W}^*,\lambda^*) & = & 0 \end{array} \right. \text{for each net } i \in (n_1,\ldots,n_k) \tag{34}$$

Thus we have

$$\Delta W^*(i) = \beta\{[Slk_t - Slk(i)]S_W^{Slk}(i) + \alpha S_W^{FOM}(i)\} \tag{35}$$

where,

$$\beta = \sqrt{\frac{C}{\Sigma_{i=n_1}^{i=n_k}[(Slk_t - Slk(i))S_W^{Slk}(i) + \alpha S_W^{FOM}(i)]^2}} \tag{36}$$

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.

Based on (35), we propose the following sensitivity guided net weighting scheme

$$W(i) = \left\{ \begin{array}{ll} W_{org}(i) & Slk(i) > Slk_t \\ W_{org}(i) + \Delta W^*(i) & Slk(i) \le Slk_t \end{array} \right. \tag{37}$$

where $W_{org}(i)$ is the original net weight, $\Delta W^*(i)$ is net weight adjustment from (35).

5. EXPERIMENTAL FLOW AND RESULTS

5.1 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 theoretically may 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 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.

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

```
run wire length driven placement with uniform weight $W_{min}$,
  i.e., $W_{org}(i) = W_{min}$ for all nets
run static timing analysis
compute $S_W^{FOM}$, $S_W^{Slk}$ for each net
compute weight $W(i)$ for each net $i$ based on (37)
run timing driven placement with new net weight
```

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.[1] 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. 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 since 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 [20]. 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 [21]. 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 1. The slack target Slk_t is set to be 0.3ns for all circuits in our experiments.

Table 1: 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

We compare the following three algorithms:

[1] 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.

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

5.2 Timing Driven Placement

Based on timing driven placement flow described in Algorithm 2, we first run CPlace with $W_{min} = 10$. Then we run Einstimer to perform the static timing analysis. Before generating net weight based on (37), we linearly scale $(Slk_t - Slk(i))S_W^{Slk}(i)$ and $S_W^{FOM}(i)$ to $[0,1]$. We set $\beta = 60$, i.e., the maximum ΔW generated by slack sensitivity will be 60. To evaluate the impact of FOM sensitivity, we have the following two cases: one is TS ($\alpha = 0$), and the other is TSF with FOM sensitivity ($\alpha = 0.8$). The maximum ΔW due to FOM sensitivity is $\beta \cdot \alpha = 48$. As a reference, we also report the result from zero wire model (ZW), i.e., assuming zero wire resistance and capacitance.

Table 2: FOM comparison after placement.

Design	FOM				Improvement	
	ZW	WL	TS	TSF	TS	TSF
ckt1	-9134	-41650	-26093	-25602	48%	49%
ckt2	0	-6966	-4102	-3454	41%	50%
ckt3	-535	-13711	-6468	-5595	55%	62%
ckt4	-322	-8057	-4024	-3440	52%	60%
ckt5	-114	-28527	-15334	-12229	46%	57%
ckt6	-142	-20257	-9417	-9536	54%	53%
ckt7	-4	-452	-248	-131	46%	72%
Average					49%	58%

Table 3: WNS comparison after placement.

Design	WNS				Improvement	
	ZW	WL	TS	TSF	TS	TSF
ckt1	-1.702	-6.274	-3.392	-4.254	63%	44%
ckt2	0.248	-2.977	-1.784	-1.754	37%	38%
ckt3	-0.55	-4.997	-3.684	-3.788	30%	27%
ckt4	-0.941	-7.218	-3.736	-3.605	55%	58%
ckt5	-0.102	-3.575	-2.379	-2.002	34%	45%
ckt6	-0.508	-5.47	-5.484	-4.856	-0%	12%
ckt7	0.16	-1.135	-0.66	-0.432	37%	54%
Average					37%	40%

Table 2 and Table 3 compare the FOM and WNS results from ZW, WL, TS, 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 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 2, the improvement potential of TS for $ckt1$ can be computed by $(41650 - 26093)/(41650 - 9134) = 48\%$. We can see that algorithm TS 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 TS (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. Note that we did not compare the WNS improvement in terms of cycle time. This is because the chips we tested all have multiple clock signals with different cycles times.

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 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 4 compares the total wire length (TWL) from three algorithms WL, TS and TSF. We can see that TS and TSF only increase TWL by a small percentage.

Table 4: Total wire length comparison after placement

Design	TWL ($\times 10^6$)			change	
	WL	TS	TSF	TS	TSF
ckt1	10.30	10.89	11.10	5.79%	7.86%
ckt2	14.93	15.87	16.54	6.28%	10.78%
ckt3	40.05	41.04	42.41	2.49%	5.91%
ckt4	49.44	49.59	50.07	0.30%	1.26%
ckt5	59.98	64.39	63.59	7.36%	6.02%
ckt6	134.64	136.01	135.96	1.01%	0.98%
ckt7	126.60	126.22	126.34	-0.30%	-0.21%
Average				3.28%	4.66%

5.3 After Physical Synthesis

For deep submicron timing closure, tremendous amount of optimizations such as buffer insertion, gate sizing, pin swapping will be done after placement [17] [16]. 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 [15] to further improve WNS and FOM based on the placement results from WL, TS and TSF algorithms.

Table 5: FOM comparison after physical synthesis.

Design	FOM			Improvement	
	WL	TS	TSF	TS	TSF
ckt1	-7829	-6086	-5170	22%	34%
ckt2	-2059	-384	-631	81%	69%
ckt3	-1854	-405	-422	78%	77%
ckt4	-2537	-1844	-1770	27%	30%
ckt5	-4732	-2726	-1819	42%	62%
ckt6	-1481	-541	-266	63%	82%
ckt7	-94	-8	0	91%	100%
Average				58%	65%

Table 5 and Table 6 compare the FOM and WNS after PDS for algorithms WL, TS and TSF. Again, we see a consistent significant improvement of TS and TSF over WL. The explicit FOM guided algorithm TSF still has the best FOM after PDS (on average 7%

Table 6: *WNS* comparison after physical synthesis

Design	*WNS*			Improvement	
	WL	*TS*	*TSF*	*TS*	*TSF*
ckt1	-0.834	-0.743	-0.739	11%	11%
ckt2	-0.705	-0.011	-0.073	98%	90%
ckt3	-0.701	-0.139	-0.19	80%	73%
ckt4	-2.156	-1.908	-1.9	12%	12%
ckt5	-0.472	-0.443	-0.341	6%	28%
ckt6	-0.36	-0.293	-0.351	19%	3%
ckt7	-0.097	0	0	100%	100%
Average				47%	45%

better than the improvement by TS over WL). Note that the WNS improvement of TSF after PDS is slightly smaller than that of TS, 45% vs. 47%, while the WNS improvement of TSF after placement is higher than that of TS. 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 7 shows the total wire length of each circuit after PDS. It can be seen that the wire length difference becomes smaller compared to Table 4 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 TS, which means timing driven placement with FOM sensitivity trades off little wire length for a much better FOM.

Table 7: Total wire length comparison after physical synthesis

Design	TWL ($\times 10^6$)			change	
	WL	*TS*	*TSF*	*TS*	*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 8 compares the total cell area after PDS for algorithms WL, TS and TSF. It shows that the total cell area difference is negligible among these three algorithms. In fact, TS and TSF even have slightly smaller area than WL, for example of $ckt1$, which has a 1.3 percent area reduction. So a better placement starting point may need less aggressive gate sizing.

Table 8: Total cell area comparison after physical synthesis

Design	*Area* ($\times 10^5$)		
	WL	*TS*	*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

6. CONCLUSIONS

In this paper, we first derive a set of sensitivity analysis for 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, but also the physical synthesis optimization after it. 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.

7. 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.

8. 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] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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