How Accurately Can We Model Timing In A Placement Engine?

Amit Chowdhary, Karthik Rajagopal, Satish Venkatesan, Tung Cao, Vladimir Tiourin, Yegna Parasuram[†], Bill Halpin[‡]

Intel Corporation, Santa Clara, CA
[†]Sierra Design Automation, Santa Clara, CA
[‡]Synplicity, Inc., Sunnyvale, CA

{amit.chowdhary, karthik.rajagopal, satish.venkatesan, tung.d.cao, vladimir.tiourin}@intel.com, yegna@sierra-da.com, bhalpin@synplicity.com

ABSTRACT

This paper presents a novel placement algorithm for timing optimization based on a new and powerful concept, which we term differential timing analysis. Recognizing that accurate optimization requires timing information from a signoff static timing analyzer, we propose an incremental placement algorithm that uses timing information from a signoff static timing engine. We propose a set of differential timing analysis equations that accurately capture the effect of placement perturbations on changes in timing from the signoff timer. We have formulated an incremental placement optimization problem based on differential timing analysis as a single linear programming (LP) problem which is solved to generate the new timing-optimized placement.

Our experiments show that the worst negative slack (WNS) improves by an average of 30% and the total negative slack (TNS) improves by 33% on average for a set of circuits from a 3.0 GHz microprocessor that were already synthesized and placed by a leading industrial physical synthesis tool. We also show that multiple iterations of our engine give further TNS improvements – an average improvement of 51%, which implies that our placer will significantly speed up timing convergence.

Categories and Subject Descriptors

B.7.2 [**Integrated circuits**]: Design aids – *placement and routing.*

General Terms

Algorithms, Design, Performance.

Keywords

Timing-driven placement, static timing analysis, linear programming, differential timing analysis.

1. INTRODUCTION

Placement is an integral part of a timing convergence flow. It determines the length of nets on timing-critical paths, which directly affects the delay of cells and nets on these critical paths. The problem of timing driven placement is extremely complicated due to the fact that it is difficult to accurately model timing as a function of the placement of cells. We briefly explain how timing is modeled in existing placement algorithms.

- Existing global placement engines convert timing information into net weights or net constraints which are then used in the global placement formulation [3][4]. Few methods [6] interleave timing analysis and global placement. Kahng *et al.* [6] use a min-max timing optimization approach that moves cells in order to minimize the maximum of all weighted edge delays, where an edge is the combination of a net and its driver cell; edge delay is modeled by Elmore model. The min-max timing optimization step is interleaved with global placement based on recursive bisection. The timing model in min-max optimization calculates weights on edges using timing information of current placement, but does not model the change in arrival times and hence slacks, which are needed to accurately model timing similar to a static timing analysis engine.

- Existing incremental placement engines [5][7] improve timing by moving cells on a few timing-critical paths. Choi and Bazargan [7] iteratively assign net constraints on nets of top few critical paths and move cells to meet these constraints. Ajami and Pedram [5] present an iterative technique that models nets with movable Steiner points in a static timing analysis based incremental placement framework. However, this formulation is non-convex, which can only be solved for top few paths, even after approximation.

Existing placers cannot accurately model timing for more than a few paths. Also, their timing models do not capture the complexities of static timing such as arrival time propagation, slope (transition time) effects, transparent latches, etc. As a result, these methods leave much room for further optimization. Recent work has shown that there is significant scope for improvement in the state of the art in placement technology [8].

We found that expert designers can easily improve timing of a placement generated by state of the art timing-driven placement engines. Designers do so by moving a few critical cells, because they have a good knowledge of the impact of cell movement on the overall timing of the design. During these timing improvement steps, designers do not worry about removal of cell overlaps, because moving a small number of cells results is a small amount of overlaps that could be removed by a legalization step at the end. Designers need to run a timing analysis engine after moving

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

a few cells to find the new timing. The main drawback of manual incremental placement by designers is that they can focus on only a few cells at a time. Therefore, multiple iterations are needed to converge timing and every iteration requires timing analysis making such a manual flow very time-consuming. It would be very beneficial to accurately automate such a placement step.

2. MOTIVATION

This research is motivated by the realization that optimizations need to be based on an accurate signoff timing engine to be successful. We start with timing report from a state of the art timing analysis engine which models all the complexities of modern design. Reference timing from an accurate timer is the basis for our placement optimization. Rather than fully modeling static timing analysis, we use the accurate timing information and a novel differential timing analysis model to direct placement optimization. A timing model in the placer that calculates changes in timing with respect to an accurate reference timing, which we call differential timing, will be much more precise than timing models used in current placers to estimate absolute timing numbers. We bound the movement of cells to improve the accuracy of our differential timing model.

Key contributions of our approach are:

- A differential timing analyzer that computes differences in arrival and required times at all pins of a circuit, relative to a reference static timing analysis, given changes in cell placement. This differential analyzer is almost exact in the neighborhood of the reference static timing, including modeling of setup time and latch transparency.
- A linear programming formulation of the differential timing model that optimizes timing of the input placement. To maintain the validity of our differential timing model, we limit the placement changes to a local neighborhood.

3. PROBLEM STATEMENT

The problem of incremental timing-driven placement can be stated as follows: Given an initial placement, its timing information from a static timing analysis engine and a critical subcircuit, find a new placement of cells in the subcircuit such that overall timing is improved. Timing of a circuit is measured in terms of two metrics: worst negative slack (WNS) and total negative slack (TNS). Slack at any pin of a standard cell or any pad of the circuit is defined as the difference between the time signal is required (required time) and the time signal arrives (arrival time). A negative slack implies that signal is arriving later than required. WNS is defined as the worst slack among all timing endpoints of the circuit, where a timing endpoint is either the data input pin of a latch or a flip-flop, or an output pad of the circuit. TNS is the total sum of negative slacks at the timing endpoints (positive slacks are ignored). We select a critical subcircuit of the input circuit for incremental placement. Figure 1 illustrates a small subcircuit that we use as an example of input to incremental placement. Cells B, C, D, E, F and G are movable cells. We consider combinational as well as sequential cells as movable cells in our incremental placement approach. Fixed cells (or pads) that drive movable cells are called start cells. Cells A and H, and input pad I are start cells. Fixed cells (or pads) driven by movable cells are called end cells. Output pad J and cell K are end cells.

4. Proposed Algorithm

We now describe differential timing analysis that models changes in timing as a function of changes in cell locations. The incremental placement problem can be naturally modeled as a linear programming (LP) problem using differential timing analysis, as we show next. Current LP solvers can optimally and quickly solve very large LP problems [2]. We now describe the differential timing analysis and the resulting LP problem formulation of incremental placement. We first describe the modeling of changes in net length and load capacitance, and then use these changes to describe our differential timing model.

4.1 Model for net length and load capacitance

We define x_i and y_i variables for new x and y locations of cell i for every movable cell. Length of a net is modeled as half-perimeter of the bounding box of all cells connected to it. We define variables $leftx_j$, $rightx_j$, $lowery_j$ and $uppery_j$ for the four boundaries of the bounding box of net j. For every cell i connected to net j,

$$leftx_j = \min_i(x_i);\ rightx_j = \max_i(x_i)$$

$$lowery_j = \min_i(y_i);\ uppery_j = \max_i(y_i)$$

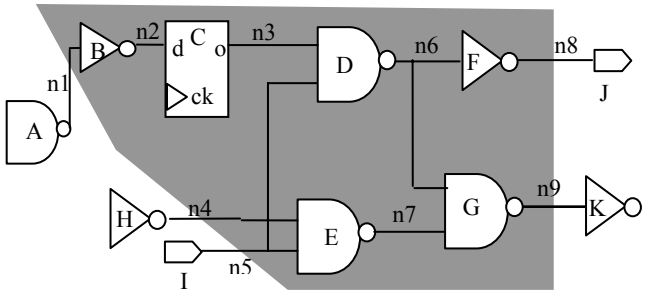

Fig. 1. A small subcircuit used to explain our linear programming formulation. Cells B, C, D, E, F, G are movable.

These min and max functions are converted to linear constraints below.

$$leftx_j \le x_i;\quad rightx_j \ge x_i$$

$$lowery_j \le y_i;\ uppery_j \ge y_i$$

Even though these constraints allow $leftx_j$ to be much less than $min_i(x_i)$, the final LP solution that optimizes *TNS* will guarantee that $leftx_j$ is set to $min_i(x_i)$.

We model net by half-perimeter of its bounding box. The change in the length of net j is given below.

$$\Delta l_j = (rightx_j - leftx_j) + (uppery_j - lowery_j) - \text{old\_l}_j$$

Here, old_l_j is the length of net j in the initial placement.

The load capacitance $cload_i$ of cell i is the sum of the interconnect capacitance and the total pin capacitance cpin_j of all receiver pins connected to the net j driven by cell i. Here, c is the interconnect capacitance per unit length and l_j is the total length of the net.

$$cload_i = \text{c} \cdot l_j + \text{cpin}_j$$

The change in load capacitance is then a linear function of change in net length. We currently use a single value for c regardless of the metal layer on which the net is routed.

$$\Delta cload_i = \mathrm{c} \cdot \Delta l_j$$

The maximum load capacitance that can be driven by cell *i* is bounded by $Cmax_i$. The Cmax constraint is linear as given below. The only variable in this constraint is Δl_j.

$$\mathrm{c} \cdot \mathrm{old\_l_j} + \mathrm{c} \cdot \Delta l_j + \mathrm{cpin_j} \le \mathrm{Cmax_i}$$

We limit cell movement by M to reduce placement perturbation and to improve accuracy of differential timing model. Here, old_x_i and old_y_i are x and y locations of cell *i* in the initial placement.

$$\mathrm{old\_x_i} - \mathrm{M} \le x_i \le \mathrm{old\_x_i} + \mathrm{M}$$
$$\mathrm{old\_y_i} - \mathrm{M} \le y_i \le \mathrm{old\_y_i} + \mathrm{M}$$

4.2 Model for differential timing analysis

4.2.1 Delay and slope (transition time) across cells

The delay from an input pin *k* to the output pin of a cell *i* can be modeled as a linear function of the load capacitance at the output pin and the slope (transition time) at the input pin, with a reasonably high degree of accuracy. The slope at the output pin of cell *i* can be defined by a linear function in a similar fashion.

$$delay_{i,k} = \mathrm{A_0} + \mathrm{A_1} \cdot cload_i + \mathrm{A_2} \cdot inslope_{i,k}$$
$$slope_{i,k} = \mathrm{B_0} + \mathrm{B_1} \cdot cload_i + \mathrm{B_2} \cdot inslope_{i,k}$$

Here, $inslope_{i,k}$ is the slope at the input pin *k* of cell *i* and $cload_i$ is the load capacitance at the output pin of cell *i*. The constants A_0, A_1, A_2, B_0, B_1, B_2 are determined by characterization of the standard cell library. We define delay and output slope constraints for every feasible signal transition for the cell. For example, an inverter has only two transitions – (input rise, output fall) and (input fall, output rise), while a two-input XOR has all four possible transitions. To simplify our discussion in this paper, we write constraints hereafter for only one transition for a cell, but our LP formulation includes all possible transitions for every cell.

Change in delay and slope can be modeled by linear constraints.

$$\Delta delay_{i,k} = \mathrm{A_1} \cdot \Delta cload_i + \mathrm{A_2} \cdot \Delta inslope_{i,k}$$
$$\Delta slope_{i,k} = \mathrm{B_1} \cdot \Delta cload_i + \mathrm{B_2} \cdot \Delta inslope_{i,k}$$

It is very important to note that linear modeling of $\Delta delay$ and $\Delta slope$ has a higher accuracy than linear modeling of absolute *delay* and *slope*. Thus, the use of differential timing analysis with respect to reference timing from an accurate static timer is more precise than directly using static timing model. Prior work has used a simple, but inaccurate, modeling of absolute timing.

4.2.2 Delay and slope across net segments

For a net with *m* receiver pins, we individually consider timing for *m* net segments, where a net segment is the connection from the driver pin to a receiver pin of the net. We use Elmore model [1] for estimating delay across a net segment *j* of length l_j.

$$delay_j = \mathrm{K_D} \cdot \mathrm{r} \cdot l_j \cdot \left(\frac{\mathrm{c} \cdot l_j}{2} + \mathrm{cpin_j}\right)$$

Here, r is the interconnect resistance per unit length, K_D is a constant with a value of 0.69, and $cpin_j$ is the pin capacitance of the receiver pin of the net segment *j*. For lack of simple modeling, we do not consider the capacitance of side branches when modeling delay from driver pin to a receiver pin. We need to enhance our timing model to include capacitance of side receivers, at least receivers close to the receiver in question.

Similarly, slope at the receiver pin *k* of a net segment with driver cell *i1* and receiver cell *i2* is given below, where K_S is a constant with a value of 2.2 for transition from 10% to 90% of V_{DD}.

$$inslope_{i2,k} = \mathrm{K_S} \cdot \mathrm{r} \cdot l_j \cdot \left(\frac{\mathrm{c} \cdot l_j}{2} + \mathrm{cpin_j}\right) + slope_{i1}$$

We model the change in length of a net segment, similar to the modeling of change in length of a net. When the length of net segment changes by Δl_j, we derive the change in delay and slope as a function of Δl_j as given below.

$$\Delta delay_j = \mathrm{K_D} \cdot \left(\mathrm{r} \cdot \left(\mathrm{c} \cdot \mathrm{old\_l_j} + \mathrm{cpin_j}\right) \cdot \Delta l_j + \frac{\mathrm{r} \cdot \mathrm{c}}{2} \cdot \left(\Delta l_j\right)^2\right)$$

$$\Delta inslope_{i2,k} = \mathrm{K_S} \cdot \left(\mathrm{r} \cdot \left(\mathrm{c} \cdot \mathrm{old\_l_j} + \mathrm{cpin_j}\right) \cdot \Delta l_j + \frac{\mathrm{r} \cdot \mathrm{c}}{2} \cdot \left(\Delta l_j\right)^2\right) + \Delta slope_{i1}$$

The above equations are linear, except for a quadratic term $(\Delta l_j)^2$. Because $(\Delta l_j)^2$ is a convex function, we can linearize it using a set of linear constraints as shown in Fig. 2. We bound the change in wirelength by L in order to make the linear approximation $sq\_\Delta l_j$ close to the quadratic term $(\Delta l_j)^2$.

$$sq\_\Delta l_j \ge \pm \frac{\mathrm{L}}{2} \cdot \Delta l_j$$
$$sq\_\Delta l_j \ge \pm \frac{3\mathrm{L}}{2} \cdot \Delta l_j - \frac{\mathrm{L}^2}{2}$$

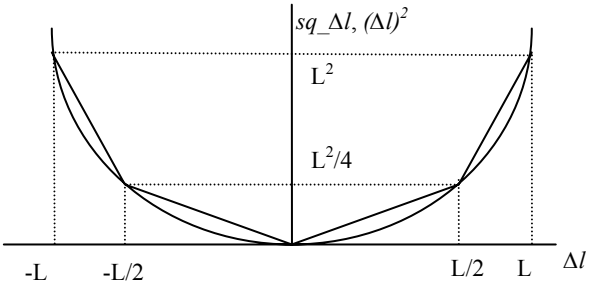

Fig. 2. Linear approximation $sq\_\Delta l$ of the squared change in wirelength $(\Delta l)^2$, given a bound L on change in wirelength.

Even though these constraints allow $sq\_\Delta l_j$ to be larger than the smallest value from these constraints, the optimal LP solution will ensure that $sq\_\Delta l_j$ is set to the smallest value from these constraints. Here, we have approximated $(\Delta l_j)^2$ by a set of four linear constraints. We can improve accuracy of linear approximation by using a larger set of linear constraints. In our experiments, we have approximated this quadratic function by a set of 20 linear constraints without any significant impact on runtime of LP solver.

4.2.3 Arrival time propagation

Changes in delay and slope across cells and net segments affects the arrival time at pins of these cells. We now define the change in arrival time at input and output pins of all cells in the critical subcircuit. The arrival time at an input pin *k* of a cell *i2* is calculated from the arrival time at the output pin of the driving

cell *i1* and the delay across the net segment *j* connecting the two cells. We define two different arrival times at every pin – one for rising and another for falling transition. However, we state only one arrival time constraint here for ease of discussion.

$$arrival_{i2,k} = arrival_{i1} + delay_j$$
$$\Delta arrival_{i2,k} = \Delta arrival_{i1} + \Delta delay_j$$

The arrival time at the output pin of a cell *i2* depends on the last arriving signal amongst all input pins of cell *i2*.

$$arrival_{i2} = \max_k (arrival_{i2,k} + delay_{i2,k})$$
$$\Delta arrival_{i2} = \max_k (\text{old\_arrival}_{i2,k} + \Delta arrival_{i2,k} + \text{old\_delay}_{i2,k} + \Delta delay_{i2,k}) - \text{old\_arrival}_{i2}$$

The above max constraint can be linearized as follows.

$$\Delta arrival_{i2} \geq \text{old\_arrival}_{i2,k} + \Delta arrival_{i2,k} + \text{old\_delay}_{i2,k} + \Delta delay_{i2,k} - \text{old\_arrival}_{i2}, \qquad \forall k$$

The $\text{old\_arrival}_{i2,k}$, old_arrival_{i2} and $\text{old\_delay}_{i2,k}$ are respectively the arrival time at input pin *k*, arrival time at output pin and delay from input pin *k* to output pin of cell *i2* from the reference timing determined by a state of the art static timing analysis engine.

4.2.4 Sequential cells

We allow sequential cells to move during incremental placement. Movement of sequential cells can give large improvements in timing, because it allows tradeoff of slack between paths ending and starting at the sequential cells. We define variables for x and y locations of sequential cells (latches and flip-flops), similar to the combinational cells. However, we treat a sequential cell as a start cell as well as an end cell.

We consider the data input pin of a flip-flop or a closed latch as a timing endpoint in our formulation. The setup time of a given sequential cell can be modeled as a linear function of the input slope at the data and clock pins, and the load at the output pin. We assume an ideal clock, which translates to the slope at the clock pin to be unchanged, i.e. $\Delta inslope_{i,ck} = 0$. Change in setup time results in an equal and opposite change in the required time at the data input pin of the sequential cell.

$$setup_i = S_0 + S_1 \cdot cload_i + S_2 \cdot inslope_{i,d} + S_3 \cdot inslope_{i,ck}$$
$$\Delta setup_i = S_1 \cdot \Delta cload_i + S_2 \cdot \Delta inslope_{i,d}$$
$$\Delta required_i = -\Delta setup_i$$

We consider the clock input pin as the timing startpoint in our formulation, thus modeling the change in clock-to-out delay due to the movement of sequential cell.

We treat the special case of transparent latch different from a closed latch. We consider a transparent latch as a combinational cell with a timing arc going from data input pin to output pin. Thus, we model the change in delay from data pin to output pin of transparent latches. We assume that a transparent latch stays transparent during a single iteration of our incremental placer. The modeling of transparent latches allows our placer to optimize paths that span one or more transparent latches.

4.2.5 Boundary constraints

For a start cell *i*, we set the change in input slope as well as arrival time at all input pins to 0. Even though the start cells are fixed, the delay from input to output pin can change due to the change in its load capacitance. The change in delay for a start cell then changes the arrival time at the output pin of the start cell. We set the following boundary constraints for all input pins of start cells.

$$\Delta inslope_{i,k} = 0$$
$$\Delta arrival_{i,k} = 0$$

For an end cell *i*, the required time at every input pin is assumed to be unchanged. Thus, the change in slack of an input pin of an end cell or a pad *k* is simply given by the negative of the change in its arrival time.

$$slack_k = required_k - arrival_k$$
$$\Delta slack_k = -\Delta arrival_k$$

In case of a sequential end cell *i*, required time changes with the change in setup time. As a result, change in slack is given below.

$$\Delta slack_k = \Delta required_k - \Delta arrival_k$$

4.2.6 Timing metrics

We calculate the two timing metrics – *WNS* and Δ*TNS* from the change in slack at the input pins of end cells or at the output pads. *WNS* is defined as the worst new slack among all end cells (or pads). Here, old_slack_k is the slack of pin (or pad) *k* from static timing analysis based on the initial placement.

$$WNS = \min(\text{old\_slack}_k + \Delta slack_k)$$
$$WNS \leq \text{old\_slack}_k + \Delta slack_k, \qquad \forall k$$

For a pin with a negative slack of old_slack_k, the impact of this pin on TNS is bounded by $-\text{old\_slack}_k$. Even though the slack at pin *k* could improve by more than $-\text{old\_slack}_k$, the impact on TNS is still bounded by $-\text{old\_slack}_k$. We call the contribution of change in slack of pin *k* on Δ*TNS* as $\Delta negSlack_k$.

$$\Delta negSlack_k = \min(-\text{old\_slack}_k, \Delta slack_k)$$

The min function can be modeled in LP problem as follows.

$$\Delta negSlack_k \leq -\text{old\_slack}_k$$
$$\Delta negSlack_k \leq \Delta slack_k$$

The change in TNS can be simply evaluated as the sum total of change in negative slack for input pins of end cells or output pads.

$$\Delta TNS = \sum_k \Delta negSlack_k$$

4.3 Linear programming formulation

We now state the incremental placement problem as an LP problem: ***Maximize Δ*TNS*, subject to the linear constraints stated above.*** Figure 3 presents the LP problem for the subcircuit of Fig. 1. Alternate objective functions can be used, such as a combination of *WNS* and Δ*TNS*. We use a fast, commercial LP solver *cplex* from ILOG to solve the above LP problem [2].

4.4 Legalization

Output of our LP problem is a new placement with improved timing, but can have overlaps. We use a legalization engine to resolve cell overlaps in the final placement. We use two calls of the legalization engine. First, we legalize only critical cells (cells moved by our placer), while ignoring remaining cells. Next, we fix critical cells that were legalized in the first step and then

legalize remaining cells. The motivation for two-step legalization is to minimize the change in timing by limiting the movement of timing-critical cells at the expense of less critical cells.

Maximize ΔTNS subject to

Model for net length (j in $\{n1,...n9\}$)

$$\left.\begin{array}{l} leftx_j \le x_i; \quad rightx_j \ge x_i \\ lowery_j \le y_i; uppery_j \ge y_i \end{array}\right\} \quad \forall \text{cell } i \text{ connected to net } j$$

$$\Delta l_j = (rightx_j - leftx_j) + (uppery_j - lowery_j) - \text{old\_l}_\text{j}$$

Timing for cells (i in $\{A,...,I\}$, net j driven by cell i)

$$\Delta cload_i = \text{c} \cdot \Delta l_j$$

$$\left.\begin{array}{l} \Delta delay_{i,k} = \text{A}_1 \cdot \Delta cload_i + \text{A}_2 \cdot \Delta inslope_{i,k} \\ \Delta slope_i = \text{B}_1 \cdot \Delta cload_i + \text{B}_2 \cdot \Delta inslope_{i,k} \end{array}\right\} \forall \text{pin } k \text{ of cell } i$$

Pin d of flipflop C is not used in these constraints

$$\text{c} \cdot \text{old\_l}_\text{j} + \text{c} \cdot \Delta l_j + \text{cpin}_\text{j} \le \text{Cmax}_\text{i}$$

Timing for nets (segment j from driver $i1$ to pin k of receiver $i2$)

$$\Delta delay_j = \text{K}_\text{D} \cdot \left(\text{r} \cdot \left(\text{c} \cdot \text{old\_l}_\text{j} + \text{cpin}_\text{j}\right) \cdot \Delta l_j + sq\_\Delta l_j\right)$$

$$\Delta inslope_{i2,k} = \text{K}_\text{S} \cdot \left(\text{r} \cdot \left(\text{c} \cdot \text{old\_l}_\text{j} + \text{cpin}_\text{j}\right) \cdot \Delta l_j + \frac{\text{r} \cdot \text{c}}{2} \cdot sq\_\Delta l_j\right) + \Delta slope_{i1}$$

$$sq\_\Delta l_j \ge \pm \frac{3\text{L}}{2^{i+1}} \cdot \Delta l_j - \frac{\text{L}^2}{2^{2i+1}}, \forall 0 \le i \le 8$$

$$sq\_\Delta l_j \ge \pm \frac{\text{L}}{2^9} \cdot \Delta l_j$$

Arrival time propagation (cell $i2$ in $\{A,...,K\}$, $i1$ is driver of $i2$)

$$\Delta arrival_{i2,k} = \Delta arrival_{i1} + \Delta delay_j \; \forall \text{input pin } k \text{ of cell } i2$$

$$\Delta arrival_{i2} \ge \text{old\_arrival}_{\text{i2,k}} + \Delta arrival_{i2,k} + \text{old\_delay}_{\text{i2,k}} + \Delta delay_{i2,k} - \text{old\_arrival}_{\text{i2}}, \forall \text{input pin } k$$

Sequential cell

$$\Delta setup_C = \text{S}_1 \cdot \Delta cload_C + \text{S}_2 \cdot \Delta inslope_{C,d}$$

$$\Delta required_C = -\Delta setup_C$$

Boundary constraints (start cell i in $\{A,C,H,I\}$, end cell j in $\{C,J,K\}$)

$$\Delta inslope_{i,k} = 0$$

$$\Delta arrival_{i,k} = 0$$

$$\Delta slack_j = \Delta required_j - \Delta arrival_j$$

Timing metrics (input pin k of end cells C,J,K)

$$\left.\begin{array}{l} \Delta negSlack_k \le -\text{old\_slack}_\text{k} \\ \Delta negSlack_k \le \Delta slack_k \\ WNS \le \text{old\_slack}_\text{k} + \Delta slack_k \end{array}\right\} \forall \text{input pin } k \text{ of end cells C, J, K}$$

$$\Delta TNS = \sum_k \Delta negSlack_k$$

Fig. 3. LP problem for incremental placement of subcircuit of Fig. 1.

5. Accuracy of our differential timing model

Differential timing model in our engine is modeled strictly on the concept of static timing analysis, which results in significant timing improvements shown later by our experiments. We list below several limitations of our timing model, and discuss techniques we use to overcome these limitations.

- Final legalization will worsen timing to some extent. However, we bound cell movement and work on a small subcircuit, resulting in only a few cell overlaps. Also, we first legalize critical cells followed by non-critical cells to reduce impact on timing.
- The bounding box model of net length may not correlate well with final routing of the net. We are working on using net parameters, such as aspect ratio and fanout, to more accurately model net length.
- The quadratic term in the change in Elmore delay of net can not be precisely modeled in the LP problem. However, we closely approximate the quadratic term by a large set of linear constraints. We also bound the change in net length to help in the linear approximation of the quadratic term.
- In case of long nets, actual load seen by a cell is smaller than the total load capacitance due to resistive shielding. We should use effective capacitance, instead of total load capacitance, while evaluating cell delay. We are working on a heuristic that models change of effective capacitance as a linear function of change in net length.

6. Experimental Results

We have implemented the algorithm for formulating incremental timing-driven placement as a linear programming problem in C++ on LINUX. We solve the LP problem using a leading industrial LP solver *cplex* from ILOG [2]. The solution of LP problem gives the new improved placement, which could have overlaps. We then remove overlaps by a two-step legalization using an internally-developed legalizer engine.

Instead of using MCNC benchmarks, we used circuits from a recent microprocessor, since the effect of incremental timing-driven placement on circuit timing is more accurately studied by using data from a recent manufacturing process and standard cell library, and by using state of the art RC estimation and timing analysis engines. For our experiments, we used a set of six circuits from a 3.0 GHz microprocessor designed on 0.13 micron process. Circuits range from a few thousand cells to 40,000 cells, as listed in Table 1. Initial placements of these circuits were generated using a leading industrial physical synthesis tool. Our results will show that placements generated from a leading timing-driven placement tool leave a lot of room for timing improvement, which could be recovered by an incremental placer that models timing more accurately. We used an internally-developed state of the art static timing analysis engine to generate timing report for the initial placement of circuits. Timing report contains slacks and slopes at the pins of all cells in the circuit.

Cells are selected as movable based on slack at their output pins. We define a slack cutoff, such that all cells with slack worse than the cutoff are selected. We also select cells in the transitive fanout of critical cells (cells with slack worse than the cutoff), because these cells directly affect the load on critical cells. We also have an upper bound on the number of cells selected, because selecting too many cells might lead to a lot of overlaps, resulting in a timing degradation during legalization (we have mostly used an

upper bound of 500 cells or 5% of total cells, whichever is smaller). Runtime of our engine is within 2-3 minutes. Biggest LP problem has 30,000 variables and 60,000 constraints [2].

Table 1 shows timing improvements by placing a small set of cells using our incremental placer. The number of cells moved by our placer is within 5%, yet we were able to improve WNS and TNS on average by 30% and 33%, respectively. (Note that the initial and final timing numbers were generated by the static timing analysis engine.) These results show that our incremental placer can substantially improve timing of the initial placement, just by moving a small set of cells. The amount of timing improvement depends on the timing quality of the initial placement. In case of the largest circuit ckt6 with 40K cells, we found that moving a small set of 50 cells gave huge improvement in TNS, which was due to the optimization of a few timing critical nets with high fanout that were not correctly optimized during global placement. It should be noted that these timing improvements are obtained only by our incremental placement, without doing any buffer insertion or circuit sizing. Based on our experience, further improvement could be obtained using placement coupled with buffer insertion and circuit sizing.

We found that other placement characteristics like routability and wirelength are largely unaffected by this optimization since these are global parameters of the design and don't get perturbed much by moving a few cells. Total wirelength of the final placement was within +/-1% of the wirelength of initial placement. We ran global routing on these placements and found similar congestion maps in the initial and final placements. We also found that the changes in slacks and slopes reported by our placement engine correlate well with these changes reported by static timing engine.

Circuit	Total cells	Cells moved	Initial		Final	
			WNS (ps)	TNS (ns)	WNS (ps)	TNS (ns)
ckt1	2,544	50	-88	-2.139	-77	-1.877
ckt2	2,683	54	-22	-0.248	-9	-0.085
ckt3	3,995	199	-59	-3.79	-51	-2.39
ckt4	4,022	169	-53	-2.72	-50	-2.31
ckt5	15,361	253	-174	-26.84	-72	-18.96
ckt6	40,011	50	-74	-3.60	-68	-0.90

Table 1: Timing improvement from using our incremental placer.

We also ran multiple runs of incremental placement on the same circuit to illustrate the full extent of optimization that can be done using our incremental placer. Chart 1 shows the results of multiple placer iterations. In every iteration, we select cells in different slack ranges to allow placement optimization of different subcircuits of the same circuit. The number of cells moved per iteration of incremental placer is bounded by 5% of the total number of cells. Chart 1 shows that TNS improved by 24-87% for these four circuits – an average improvement of 51%. Each iteration of incremental placer was able to further reduce TNS significantly, because it worked on different subcircuits. The first iteration for ckt6 improved TNS a lot, because the starting placement had some very high fanout nets which were not optimized for timing, resulting in a huge improvement in TNS by moving just 50 cells. We found that results on these six benchmark circuits are representative of results on other circuits.

7. Conclusions and Future Directions

We have developed a novel differential timing analysis model that uses reference timing from a state of the art static timer and models timing changes as a result of changes in placement with a high degree of accuracy. We have designed a powerful algorithm for incremental placement optimization based on our differential timing model. We formulated placement optimization problem as an LP problem which is then solved optimally and quickly by an LP solver. The main strength of our algorithm is that the timing model is based closely on a signoff timing analysis. We achieved improvements in WNS and TNS on average of 30% and 33%, respectively, for a set of six circuits by incrementally placing only 5% of total cells, even when the starting placement was generated by a leading tool for timing-driven synthesis and placement. We ran several iterations of our incremental placer on some circuits and got huge improvements in TNS by selecting different subcircuits in each iteration. Our incremental placer can be even more beneficial when coupled with sizing and buffer insertion.

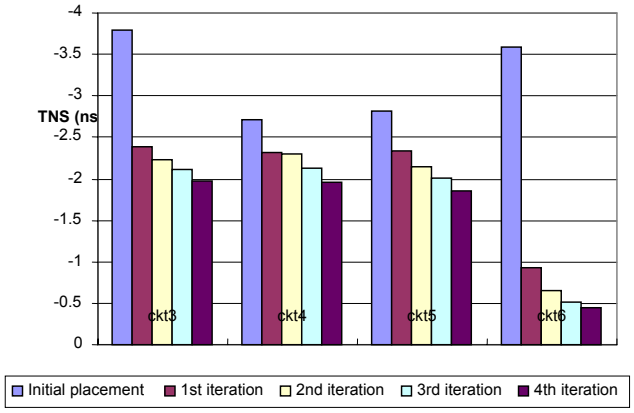

Chart 1: Iterative run of our placer: Improvements in TNS.

8. REFERENCES

[1] W. C. Elmore, "The transient response of Damped Linear network with particular regard to wideband amplifier", *Journal of Applied Physics*, pp.55-63, 1948.

[2] ILOG, *ILOG CPLEX 8.0 User's Manual*. ILOG, 2002.

[3] B. Halpin, C. Y. R. Chen, N. Sehgal, "Timing driven placement using physical net constraints", *Proc. Design Automation Conf.*, pp. 780-783, 2001.

[4] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary, B. Halpin, "Force directed timing driven placement with physical net constraints", *Proc. Intl Symp. on Physical Design*, pp. 147-152, 2003.

[5] A.H. Ajami, M. Pedram, "Post-layout timing driven cell placement using an accurate net length model", *Proc. Design Automation Conf.*, pp. 595-600, 2001.

[6] A. B. Kahng, S. Mantik, I. L. Markov, "Min-max placement for large-scale timing optimization", *Proc. Intl. Symp. of Physical Design*, pp. 143-148, 2002.

[7] W.Choi, K.Bazargan, "Incremental Placement for Timing Optimization", *Proc. Intl Conf. on CAD*, 2003.

[8] C.-C. Chang, J.Cong, M. Xie, "Optimality and scalability study of existing placement algorithms", *Proc. of the ASP-DAC*, Jan. 2003.