A New LP Based Incremental Timing Driven Placement for High Performance Designs *

Tao Luo
Department of ECE
University of Texas at Austin
Austin, TX 78712
tluo@ece.utexas.edu

David Newmark
Advanced Micro Devices
Austin, TX 78759
david.newmark@amd.com

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

ABSTRACT

In this paper, we propose a new linear programming based timing driven placement framework for high performance designs. Our LP framework is mainly net-based, but it takes advantage of the path-based delay sensitivity with limited-stage slew propagation, thus it enjoys certain hybrid feature of net and path-based timing driven placement. Our LP formulation considers not only cells on the critical paths, but also cells that are logically adjacent to the critical paths (i.e., the *criticality adjacency network*) in a unified manner. We further present a timing aware spreading method to preserve timing in legalization for high performance designs. Our algorithm has been tested on a set of 65nm industry circuits from a multi-GHz microprocessor, and shown to achieve much improved timing on hand-tuned circuits.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design

1. INTRODUCTION

In a typical custom design flow for high performance microprocessors, the chip is floorplaned into functional regions, then hierarchically partitioned into basic design units. The size of the basic design unit is usually small, ranging from a few hundred gates to tens of thousands of gates. Even for these relatively small circuits, timing driven placement is very important since the gate delay is very sensitive to wire capacitance load and input slew in deep submicron technology. Therefore, cell placement in high performance designs often involves extensive manual tuning iterations to meet stringent timing requirement.

*This work is supported in part by SRC under contract 2005-TJ-1321, IBM Faculty Award, Sun, and equipment donations from Intel.

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

It has been reported that significant performance gap exists between ASIC and custom design methodologies [1] because the custom designers understand the data flow of the circuit and take advantage of the inherent circuit regularity. On one hand, the ASIC methodology has fast turn-around time, but inferior performance for high-performance designs; on the other hand, the custom design methodology has much better performance, but very time-consuming.

To close such a gap, it is crucial to have powerful incremental timing driven placement which can iteratively improve the timing in custom designs. It helps to close not only the performance gap, but also the time-to-market gap.

Existing timing driven placement can be roughly classified into path-based and net-based approaches. The path-based algorithms try to minimize the critical paths of the circuit directly and have the advantage of holding an accurate timing view during the optimization. A common problem with them is their high computational complexity due to excessive number of paths. Path based timing driven placement includes [2] [3] [4] [5] [6] [7]. In [8], an accurate LP based differential timing analysis is proposed to improve the slack on critical paths that are identified by a static timer. However, one of the limitations of this approach is that if the static timer uses a sophisticated wireload model, such as a steiner routing tree based models, it is very difficult to formulate it into linear constraints. Therefore, any error arising from inaccurate wire models [8] in one stage of the path-based method will be propagated and accumulated in downstream stages on the timing path.

Net-based approaches usually transform timing to net budgets or weights, and perform constrained or weighted wire length optimization [9] [10] [11] [12] [13] [14] [15]. More recently, [16] proposed a sensitivity guided net weighting method that targets the net delay sensitivity. However, it did not consider slew propagation. The net-based approaches, especially the net weighting, have low computational complexity, high flexibility and is generally suitable for any wirelength minimization frameworks. Therefore, net-based approaches have more advantages as the circuit complexity continues to increase. However, net weighting often completely ignores slew propagation. Since timing is inherently path based, an effective net weighting algorithm should be based on path analysis and consider timing propagation. Furthermore, net-based approaches are often done in an ad-hoc manner and have problems with convergence [17] [7]. For instance, while the delay on critical paths decrease, other paths become critical, and this leads to a convergence problem. A systematic way of explicit perturbation control is important for net-based algorithms.

In this paper, we present an LP-based incremental timing-driven placement optimizer. Our key contributions include:

- Our LP framework is net-based, but it takes advantage of the path-based delay sensitivity with limited-stage slew propagation. Thus it combines the advantage of net-based approach (flexibility/lower computational complexity) and path-based approach (more accurate timing view).
- Our LP formulation considers not only cells on the timing-critical paths, but also cells that are logically adjacent to the critical paths in a unified manner, through weighted LP objective function and net stretching bound constraints. Therefore, our approach has precise control on timing perturbation during the optimization.
- We propose a timing aware spreading/legalization method to preserve timing for high performance custom designs. Our algorithm has been tested on a set of 65nm industry circuits from a multi-GHz microprocessor. It achieves much better timing even on carefully hand-tuned circuits (on average 20ps worst slack reduction, which is significant as the clock period is only a few hundred of pico-seconds)

The rest of the paper is organized as follows: The problem formulation is in section 2. We discuss how to generate the path-based delay sensitivity net weights in section 3. In section 4, we show a method to construct the criticality adjacency network. The overall LP program is presented in section 5. Section 6 presents the timing aware spreading algorithm. Experimental results are shown in section 7. We conclude in section 8.

2. PROBLEM FORMULATION

Table 1 lists the key notations used in the paper.

Table 1: The key notations in this paper.

c	The unit capacitance
r	The unit resistance
L_j	The wirelength of net j
Cap_j	Total output capacitive load on net e_j
$Cpin_j$	The sum of gate capacitance driven on net e_j
$Slew_i$	The input slew to cell i
Dg_i	The delay on cell i
Sg_i	The slew on cell i
K_D	Constant 0.69
K_S	Constant 2.2
a_i	The slew coefficient in cell i's delay formula in (6)
b_i	The delay coefficient in cell i's delay formula in (6)
u_i	The slew coefficient in cell i's slew formula in (7)
v_i	The delay coefficient in cell i's slew formula in (7)
De_j	The delay on net j
Se_j	The slew on net j
S_j	The delay propagation sensitivity of net j

2.1 LP formulation

In our algorithm, the timing optimizer selects a few critical paths from a timing report generated by an accurate static timer. Then it computes the delay propagation sensitivity on each net and inspects and classifies cells and nets into different categories based on their "criticality", which is logically how close they relate to critical paths. As the linear program has a system of well developed theories to solve, we formulate the timing optimization problem into an LP problem and solve it optimally.

The half parameter bounding box wirelength (HPWL) model can be formulated exactly into an LP framework. Our algorithm uses HPWL for wirelength estimation and the linear gate delay and transition/slew models for delay computation. Although HPWL may not be well correlated with the final routed wire, it still captures the fidelity of the problem with reasonable accuracy. A carefully designed algorithm can take advantage of the accurate timing information generated by a static timer to achieve the optimization objective for a certain level of accuracy.

The objective of our algorithm is to minimize the delay on timing critical paths. We formulate the linear program to minimize the weighted wirelength on selected critical timing paths,

$$minimize \quad \sum_p \sum_j L_{p,j} S_{p,j} \tag{1}$$

where $L_{p,j}$ is the wirelength of net e_j on timing path p, and $S_{p,j}$ is the delay propagation sensitivity of net e_j. In the following sections, we formulate the models and constraints of the LP problem.

2.2 The capacitive load and delay models

For cell n_i, center coordinates x_i, y_i are the variables of the LP program. For a net e_j, To model HPWL, four variables l_j, r_j, t_j and b_j are used to represent left, right, top, and bottom locations of the bounding box of net e_j. Assuming k cells are connected to net e_j, we have

$$\begin{aligned} l_j &\leq x_i + pin_x(i,j) \\ r_j &\geq x_i + pin_x(i,j) \\ t_j &\leq y_i + pin_y(i,j) \\ b_j &\geq y_i + pin_y(i,j), \qquad i = 1,2,..,k \end{aligned} \tag{2}$$

where $pin_x(i,j)$ and $pin_y(i,j)$ are the pin offset of cell i that connected to net e_j in horizontal and vertical directions respectively. The wirelength of net e_j is represented by L_j. We have

$$L_j = r_j - l_j + t_j - b_j \tag{3}$$

We use Cap_j to denote the total output capacitive load on net e_j. It is the sum of the wire capacitance of net e_j and the total pin capacitance driven on net e_j, which is denoted by $Cpin_j$, given by

$$Cap_j = c \cdot L_j + Cpin_j \tag{4}$$

where c is the unit capacitance constant. Assuming n_i is the driver of net e_j, the maximum capacitive load driven by n_i should not exceed the library specified maximum load $CMax_i$

$$Cap_j < CMax_i \tag{5}$$

To formulate the optimization problem into an LP program, we use the linear delay models for gates and the Elmore delays for wires [18]. The gate delay and transition are linear functions of input slew, $Slew$, and total capacitive load, Cap. We compute the fitting coefficients of the linear models based on a SPICE circuit simulation generated library. Note that the delay models for each pin of a gate, and for the falling or rising transition are different. We use different models for different pins and different transitions in the implementation and show only one formula here for simplicity. The gate delay Dg_i is given by

$$Dg_i = d_I + a_i \cdot Slew_i + b_i \cdot Cap_i \tag{6}$$

The gate transition Sg is given by

$$Sg_i = s_I + u_i \cdot Slew_i + v_i \cdot Cap_i \tag{7}$$

where a_i, b_i, u_i, and v_i are the fitting coefficients. d_I and s_I denote the intrinsic delay and slew of the corresponding pin of the cell.

The Elmore delay is used to estimate wire delay and slew on net e_j, which are given by

$$De_j = K_D \cdot r \cdot L_j \cdot (\frac{c \cdot L_j}{2} + Cpin_j) \quad (8)$$

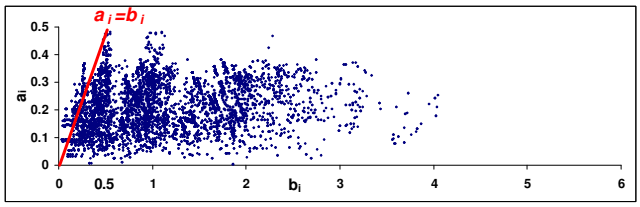
$$Se_j = K_S \cdot r \cdot L_j \cdot (\frac{c \cdot L_j}{2} + Cpin_j) \quad (9)$$

In recent publications, it has been shown that interconnect delay starts to dominate in deep submicron designs [19]. However, we should clarify that De_j in formula (8) is not the commonly referred interconnect delay, which is the part of the gate delay resulting from driving the interconnect/wire capacitance. Instead, De_j is the incremental RC delay on the wire, which is still relatively small for local nets under current technologies.

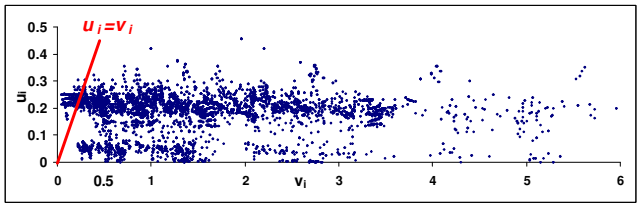
3. PATH BASED DELAY SENSITIVITY

Any change on a net will affect the delay and slew of not only the driver gate, but also all downstream receiver gates, because the change of slew on a net will propagate. The delay propagation sensitivity of a net is a measurement of the sensitivity of the path delay to a wire length change, i.e., to estimate the change in path delay due to wire adjustments. An effective net weighting method should not only consider the current stage, but it should also have a path or global timing view embedded in the formulation.

Our limited-stage delay propagation sensitivity computation considers only two stages. We have the following observation from extensive experiments.



(a) The coefficient a_i and b_i in gate delay formula (6)



(b) The coefficient u_i and v_i in gate slew formula (7)

Figure 1: The normalized coefficient for *Slew* is much smaller than that for *Cap* in both formulas

Observation 1 *The wire length change has much greater impact on delay of the current stage than its receiving gates for most cases, because the gate delay and slew are more sensitive to the output capacitive load than to the input slew for the majority types of gates.*

Figure 1 plots the normalized coefficients in formula (6) and (7) for all combinational gates in the library. In Figure 1(a), a point represents a pair of coefficients a_i and b_i in formula (6) for one gate . We see that the coefficient corresponding to Cap_i is much larger than that for $Slew_i$ in both gate delay and slew formulas in most of cases. In other words, the delay and slew are more sensitive to the output capacitive load than to the input slew. The impact of a wirelength change on delay for the down stream gates is much smaller, and decreases quickly. Meanwhile, the inaccuracy of HPWL wire model still dominates; adding more stages may not help but will increase the computation complexity. Therefore, we limit the delay propagation sensitivity computation two stages.

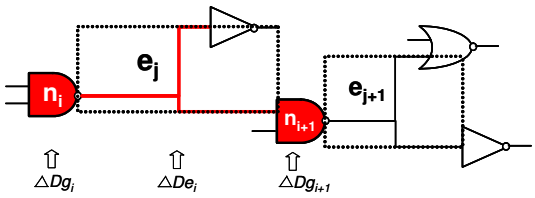


Figure 2: A circuit example for delay propagation sensitivity computation

We use the example in Figure 2 to show how to compute the delay propagation sensitivity. In Figure 2, cell n_i drives net e_j on a timing path, and cell n_{i+1} is the receiver gate connected to net e_j. Let S_j denotes the delay propagation sensitivity for net e_j. We have

$$S_j = \frac{\partial D_j}{\partial L_j} \quad (10)$$

where D_j is the portion of delay associated with net e_j. If the wirelength of net e_j changed by $\triangle L_j$, $\triangle D_j$ changes due to three components, the delta delay on the driving gate i, $\triangle Dg_i$, on net j, $\triangle De_j$, and on the receiving gate $i+1$, $\triangle Dg_{i+1}$.

$$\triangle D_j = \triangle Dg_i + \triangle De_j + \triangle Dg_{i+1} \quad (11)$$

From equation (4), (6), (7), (8), and (9), we have

$$\begin{aligned} \triangle Dg_i &= b_i \cdot c \cdot \triangle L_j \\ \triangle De_i &= K_D \cdot r \cdot \triangle L_j \cdot (\frac{c \cdot \triangle L_j}{2} + Cpin_j) \\ \triangle Dg_{i+1} &= a_{i+1} \cdot (\triangle Sg_i + \triangle Se_j) \end{aligned}$$

where $\triangle Sg_i$ denotes the slew change on cell n_i, and $\triangle Se_j$ is the slew change on net e_j. We have

$$\begin{aligned} \triangle Sg_i &= v_i \cdot c \cdot \triangle L_j \\ \triangle Se_j &= K_S \cdot r \cdot \triangle L_j \cdot (\frac{c \cdot \triangle L_j}{2} + Cpin_{j+1}) \end{aligned}$$

The formula (10) becomes

$$\begin{aligned} S_j &= b_i \cdot c + K_D \cdot r \frac{c \cdot \triangle L_j}{2} + K_D \cdot r \cdot Cpin_j \\ &+ a_{i+1} \cdot (v_i \cdot c + K_S \cdot r \frac{c \cdot \triangle L_j}{2} + K_S \cdot r \cdot Cpin_{j+1}) \end{aligned} \quad (12)$$

$\triangle L_j \to 0$ gives

$$S_j = c \cdot (b_i + a_{i+1} \cdot v_i) + r \cdot (K_D \cdot Cpin_j + K_S \cdot a_{i+1} \cdot Cpin_{j+1}) \quad (13)$$

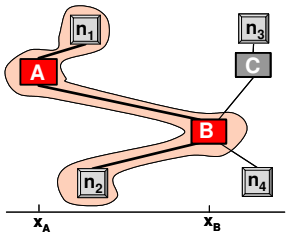
In above formula, the value of the unit resistance r is in order of magnitude smaller than that of the unit capacitance c, and the dominant term in formula (13) is $c \cdot (b_i + a_{i+1} \cdot v_i)$. Formula (13) is used to compute the delay propagation sensitivity of the net and also helps to guide the timing aware spreading/legalization.

4. CRITICALITY ADJACENCY NETWORK

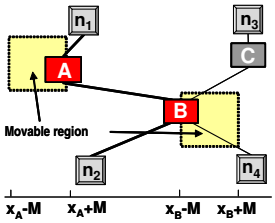
To optimize the delay on critical paths, we adjust the coordinates of all cells associated with critical paths. If we do not control the timing perturbation on non-critical paths during the optimization,

non-critical paths may become critical. In Figure 3(a), the path $n_1 \rightarrow A \rightarrow B \rightarrow n_2$ is critical, and the path $n_3 \rightarrow C \rightarrow B \rightarrow n_4$ may become critical after the optimization, as shown in Figure 3(b). Previous LP based approaches such as [13] [15] [8] set a fixed range to restrict every movable cell, as shown in Figure 3(b).

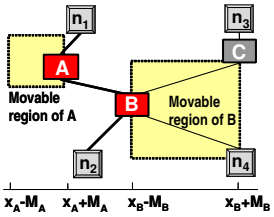
However, the delay of some cells may be extremely sensitive to wirelength changes, and other cells can be moved farther without significantly affecting the timing on non-critical paths. Such a potential to move is determined by not only the delay sensitivity of the net it drives, but also the "criticality" of the net and the cell itself. In other words, how sensitive the timing is subject to the net change and how logically "close" a cell is to the critical paths. In the following, we present the **criticality adjacency network** to classify cells and nets into different categories depending on their "criticality". As shown in Figure 3(c), we set different maximum movable ranges for cell A and B depending on the sensitivities of all nets they connected to. Furthermore, cell C in Figure 3 is not on critical path, but is also movable for it is logically adjacent to critical path, i.e., cell C is critical adjacent, as show in Figure 3(d).



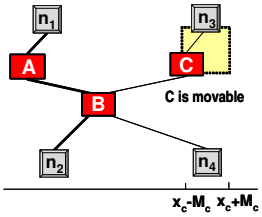
(a) Path $n_1 \rightarrow A \rightarrow B \rightarrow n_2$ is critical



(b) Fixed movable ranges to avoid path $n_3 \rightarrow C \rightarrow B \rightarrow n_4$ becoming critical



(c) Variable movable ranges depend on the sensitivity of the connected nets



(d) Cell C becomes movable in the criticality adjacency network

Figure 3: The advantages of the criticality adjacency network

4.1 Criticality adjacency network

Figure 4 is an example of a combinational netlist with one critical path. Let G= (N, E) represents a netlist that has n cells, N = $\{n_1, n_2, ...n_n\}$, and m nets, E = $\{e_1, e_2, ...e_m\}$. The criteria to construct the criticality adjacency network is essentially based on how close the non-critical branches relate to critical paths and if they are helpful for reducing the delay on critical paths.

Let symbol $\rightarrow$ denotes the connection relationship. The construction of the criticality adjacency network is through the following definitions.

Definition 1 *$N(0)$ represents the set of cells on critical paths, and $E(0)$ is the set of nets on critical paths.*

$N(0)$= $\{n_1, n_2, n_3, n_4\}$ and $E(0)$ = $\{e_4, e_5, e_6\}$ in Figure 4. All non-critical cells and nets in the circuit are classified by following definitions.

Definition 2 *$N(1)$ is the set of cells connected to nets in $E(0)$, excluding all critical nodes in $N(0)$. Set $N(2)$ contains all cells connected to cells in $N(0)$, excluding cells in $N(0)$ and $N(1)$. $N(3)$ is the set contains all other cells in N.*

Therefore,

$$\begin{aligned} N(1) &= \{c : c \rightarrow E(0), c \in N \setminus N(0)\} \\ N(2) &= \{c : c \rightarrow N(0), c \in N \setminus (N(0) \cup N(1))\} \\ N(3) &= \{c : c \in N \setminus (N(0) \cup N(1) \cup N(2))\} \end{aligned}$$

Hence, in Figure 4, $N(1)$ = $\{n_5, n_6\}$, $N(2)$ = $\{n_7, n_8, n_9\}$, and $N(3) = \{n_{10}\}$.

Definition 3 *$E(1)$ is the set of nets connected to cells in $N(0)$, excluding nets in $E(0)$. $E(2)$ is the set for nets connected to cells in $N(1)$ or $N(2)$, excluding nets in $E(0)$ and $E(1)$. All other nets are in set $E(3)$.*

Similarly,

$$\begin{aligned} E(1) &= \{e : e \rightarrow N(0), e \in E \setminus E(0)\} \\ E(2) &= \{e : e \rightarrow (N(1) \cup N(2)), e \in E \setminus (E(0) \cup E(1))\} \\ E(3) &= \{e : e \in E \setminus (E(0) \cup E(1) \cup E(2))\} \end{aligned}$$

In the example in Figure 4, $E(1)$ = $\{e_8, e_{11}\}$, $E(2)$ = $\{e_1, e_2, e_3, e_7, e_9, e_{10}, e_{12}, e_{13}\}$, and $E(3)$ = $\{e_{14}, e_{15}\}$.

By classifying cells and nets based on "criticality", our algorithm optimally moves cells not only in $N(0)$ and $N(1)$, but also in $N(2)$. All cells in $N(3)$ and all nets in $E(3)$ are fixed. Therefore, the criticality adjacency network helps to obtain more room for optimization, while explicitly controls the timing perturbation.

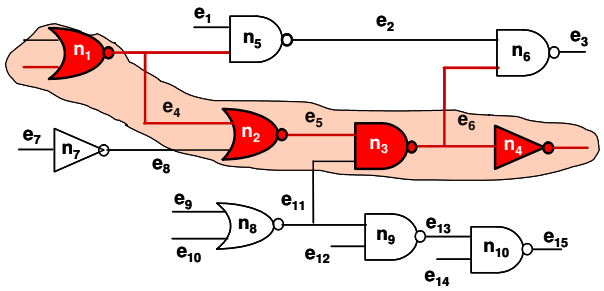


Figure 4: The criticality adjacency network

4.2 The timing perturbation constraints

With the help of the criticality adjacency network, instead of setting a fixed maximum movement range for all cells, we set a sensitivity based stretching bound for each net. The tightness of the stretching bound of a net is based on how sensitive the path delay is subject to the net change.

For a net e in $E(1)$ or $E(2)$, n_d is the driver cell and n_i is one of the receiver cells. From each pin of cell n_d to the receiver cell n_i, we compute a delay propagation sensitivity score. We compute the inverse of the sensitivity score and use it as the weight for net e. The sensitivity weights for all nets are scaled between zero and Max. Max is an experimental setting parameter. Let W_e denotes the weight for nets in $E(2)$ and $W_e^{'}$ denotes for nets in $E(1)$. The cell movement is restricted by the following net stretching bound constraints,

$$\begin{aligned} L_e - W_e \cdot L_e \leq L_e \leq L_e + W_e \cdot L_e, \forall e \in E(2) \\ L_e - W_e^{'} \cdot L_e \leq L_e \leq L_e + W_e^{'} \cdot L_e, \forall e \in E(1) \end{aligned} \quad (14)$$

The criticality adjacency network may be expanded to have more criticality levels to include more movable cells. Our experience shows that current level of critical adjacency network is sufficient.

5. THE OVERALL LINEAR PROGRAM ALGORITHM

We formulate the optimization problem into an LP program and perform weighted critical nets optimization. Powered with the criticality adjacency network, the algorithm unifies the objective of timing optimization and perturbation control into one LP framework.

Furthermore, the slack is also considered for weight adjustment. We adjust the weight of nets on critical paths according to their slacks or original delays. Assuming there are top P critical paths in a circuit, and the delay in the timing report for a path p is t_p. For a net e_j, it is possible that the net e_j belongs to P_j critical paths. Let S'_{k,e_j} denotes the delay propagation sensitivity of net e_j corresponding to the critical path p. Then, the adjusted delay propagation sensitivity weight of net e_j, which is denoted by S'_{e_j}, becomes

$$S'_{e_j} \geq S'_{p,e_j}, \quad p = 1,2,...,P_j \tag{15}$$

where,

$$\begin{aligned} S'_{p,e_j} &= \frac{t_p}{T_{min}} \cdot S_{p,e_j} \\ T_{min} &= min(t_p), \quad p = 1,2,..,P \end{aligned} \tag{16}$$

T_{min} is the shortest path delay among all P critical paths, which is used to normalize the original delay of all paths. Formula (16) adds additional weight for paths with larger negative slacks. Using the adjusted delay propagation sensitivity as net weight, the objective of the LP program is to minimize the sum of the weighted wirelength on critical paths. The following formulation is equivalent to formula (1).

$$\begin{aligned} &min \ \ \sum S'_{e_j} \cdot L_{e_j} \\ &\forall e_j \rightarrow E(0) \end{aligned} \tag{17}$$

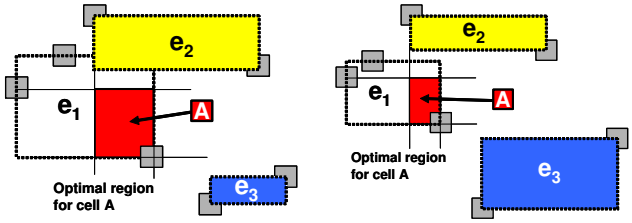
The LP is formed under a set of constraints defined in previous sections. Such as the wire length and capacitive load constraints (2), (3), and (4), the maximum load constraint (5), and the timing perturbation constraints (14),etc.

6. TIMING AWARE SPREADING FOR LEGALIZATION

The timing optimizer will generate a solution with cell overlaps. To legalize the placement, timing improvements may decrease. It is important that the legalization algorithm should avoid too much timing degeneration. According to [20] and [21], to maintain the relative orders among cells during the legalization helps preserve timing. We use the bin-stretching and Delaunay-triangulation based spreading algorithms similar to [21] for cell spreading to reduce overlaps.

We modify the spreading process in [21] to become timing aware to help cells with higher delay sensitivities move closer toward their "optimal" regions. The BoxPlace heuristic [22] is an effective method to reduce the wirelength in detailed placement. In brief, the BoxPlace moves a cell to the mean of its connected net bounding boxes to reduce the wire length. We propose a weighted BoxPlace to improve the cell spreading timing aware.

In Figure 5, e_1, e_2 and e_3 are nets connected to cell A. Figure 5(a) shows the optimal region for cell A, which is the medium of all cells connected to cell A. Figure 5(b) shows that nets are shrunk or expanded depending on their delay sensitivity weights. Conceptually, a net with higher delay propagation sensitivity will be shrunk and a net with lower weight will be expanded. The new optimal region for cell A shown in Figure 5(b) is better than that in Figure 5(a) from the timing perspective. To pull cell A to its optimal region, a timing optimization force is generated on cell A. The timing force is scaled and vector combined with the spreading force to generate a timing aware cell spreading force to guide the movement of a cell. The cell spreading stops once the cells density distribution satisfies certain criteria, and we legalize the placement.



(a) The optimal region for cell A

(b) The weighted optimal region for cell

Figure 5: Under the weighted nets e_1, e_2 and e_3, the cell's optimal region changed

7. EXPERIMENTAL RESULTS

We implement the algorithm in C++ and use the commercial tool MOSEK [23] as the LP solver. Seven circuits from a multi-GHz processor in 65nm process technology are used for experiments. The circuits are manually placed and have been optimized by designers to obtain the desired performance. The circuit sizes range from 6k standard cells to 28k. We take the critical paths above a certain threshold to optimize, then call the timing aware legalizer to remove overlaps. For each circuit, different thresholds are tested, and the best result is kept. Although those circuits have been manually optimized, our algorithm still achieved significant improvements; the result is shown in Table 2.

In table 2, column *Gates* and *Nets* are the number of cells and nets of the testcases. Column *MCells* is the number of movable cells in the LP formulation. Column *Initial* is the initial worst slack of the circuit. Column *Final* is the worst slack after the optimization. $\triangle slack$ summarizes the worst slack improvement on each circuit. Column *TNSReduced* is the total negative slack improved. Column *Base* is the timing baseline we used to measure how much slack improvement is obtained. If *Base* equals 0, *TNSReduced* is the total negative slack reduced. Because the final worst slack in a few circuits is positive, we use a *base* larger than zero to measure the total slack improvement for those circuits. Note that *ckt*1 has a positive initial worst slack. We should point out that it is meaningful to improve a positive worst slack design unit, because any slack improvement can be traded for power reduction later, where designer downsizes gates on timing paths with large positive slack to reduce the power consumption.

Column *OrignWL* and *FinalWL* are the initial and final HPWL wirelength. $\triangle WL$ is the wirelength change. We can see that most of wirelength changes are within 0.1%, which implies that the disturbance to the circuits is very small. We did not show the computation timing data because the computation is very fast for all testcases. The algorithm only handles a small number of cells and because of the efficient linear formulations, the algorithm runs

Table 2: Experimental results

Testcases	*Gates*	*Nets*	*MCells*	*Initial*	*Final*	$\triangle slack$	*TNSReduced*	*Base*	*OrignWL*	*FinalWL*	$\triangle WL$
ckt1	6671	7261	340	22	32	10	297	40	193904	192962	-0.50%
ckt2	8249	9640	89	-15	5	20	412	10	240671	239693	-0.41%
ckt3	9541	12161	92	-33	-15	18	937	0	267661	268240	0.22%
ckt4	13220	14479	164	-54	-30	24	559	0	483479	483037	-0.09%
ckt5	15486	19515	587	-37	-12	25	331	0	432319	436170	0.89%
ckt6	27014	28961	60	-3	12	15	136	15	659269	659171	-0.01%
ckt7	28535	31893	62	-36	-22	14	1211	0	921118	921518	0.04%

very fast, within a few minutes for the largest circuit. By moving a small number of critical cells, slack can be improved considerately. We observe a slack improvement of 20 picosecond on average, which is significant considering the circuits have already been hand-optimized.

8. CONCLUSION

We proposed a new LP-based incremental timing optimizer for timing optimization in placement for high performance custom designs and ASICs. Our LP framework uses an accurate delay sensitivity based net-weighting method that combines the advantage of the path-based approach. We further presented a novel criticality adjacency network concept to formulate cells both on and adjacent to critical paths into the optimization framework, which helps to precisely control the timing perturbation during the optimization. In addition, we developed a timing aware spreading method to preserve timing during the legalization. Our experimental results showed that the proposed algorithm significantly improved timing on a set of manually-optimized industry circuits from a 65nm multi-GHz high performance custom processor.

9. REFERENCES

[1] D. G. Chinnery and K. Keutzer, "Closing the gap between asic and custom: an asic perspective," in *Proc. Design Automation Conf.*, pp. 637–642, 2000.

[2] M. Burstein and M. N. Youssef, "Timing influenced layout design," in *Proc. Design Automation Conf.*, pp. 124–130, 1985.

[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 placement 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] Y.-C. Ju and R. A. Saleh, "Incremental techniques for the identification of statically sensitizable critical paths," in *Proc. Design Automation Conf.*, pp. 541–546, 1991.

[6] W. Swartz and C. Sechen, "Timing driven placement for large standard cell circuits," in *Proc. Design Automation Conf.*, pp. 211–215, 1995.

[7] A. B. Kahng, S. Mantik, and I. L. Markov, "Min-max placement for large-scale timing optimization," in *Proc. Int. Symp. on Physical Design*, pp. 143–148, 2002.

[8] A. Chowdhary, K. Rajagopal, S. Venkatesan, T. Cao, V. Tiourin, Y. Parasuram, and B. Halpin, "How accurately can we model timing in a placement engine?," in *Proc. Design Automation Conf.*, pp. 801–806, 2005.

[9] M. Marek-Sadowska and S. Lin, "Timing driven placement.," pp. 94–97, 1989.

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

[11] B. Halpin, C. Y. R. Chen, and N. Sehgal, "A sensitivity based placer for standard cells," in *Proc. of Great Lakes symp. on VLSI*, pp. 193–196, 2000.

[12] C. Chen, X. Yang, and M. Sarrafzadeh, "Potential slack: an effective metric of combinational circuit performance," in *Proc. Int. Conf. on Computer Aided Design*, pp. 198–201, 2000.

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

[14] T. 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. Choi and K. Bazargan, "Incremental placement for timing optimization," in *Proc. Int. Conf. on Computer Aided Design*, p. 463, 2003.

[16] 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, 2004.

[17] M. Sarrafzadeh, D. Knol, and G. Tellez, "Unification of budgeting and placement," in *Proc. Design Automation Conf.*, pp. 758–761, 1997.

[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] P. Gopalakrishnan, A. Odabasioglu, L. Pileggi, and S. Raje, "Overcoming wireload model uncertainty during physical design," in *Proc. Int. Symp. on Physical Design*, pp. 182–189, 2001.

[20] H. Ren, D. Z. Pan, C. J. Alpert, and P. Villarrubia, "Diffusion-based placement migration," in *Proc. Design Automation Conf.*, June, 2005.

[21] T. Luo, H. Ren, C. J. Alpert, and D. Z. Pan, "Computational geometry based placement migration," in *Proc. Int. Conf. on Computer Aided Design*, 2005.

[22] A. A. Kennings and I. L. Markov, "Analytical minimization of half-perimeter wirelength," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 179–184, 2000.

[23] MOSEK, "http://www.mosek.com," 2005.