Timing ECO Optimization Via Bézier Curve Smoothing and Fixability Identification

Hua-Yu Chang, Iris Hui-Ru Jiang, *Member, IEEE,* and Yao-Wen Chang, *Senior Member, IEEE*

***Abstract*—Due to the rapidly increasing design complexity in modern integrated circuit design, more and more timing failures are detected at late stages. Without deferring time-to-market, metal-only engineering change order (ECO) is an economical technique to correct these late-found failures. Typically, a design might need to undergo many ECO runs in design houses; consequently, the usage of spare cells for ECO is of significant importance. In this paper, we aim at timing ECO by using as few spare cells as possible. We observe that a path with good timing is desired to be geometrically smooth. Unlike negative slack and gate delay used in most prior work, we propose a new metric of timing criticality, fixability, by considering the smoothness of timing violating paths. To measure the smoothness of a path, we use the Bézier curve as the golden path. Furthermore, in order to concurrently fix timing violations, we derive a propagation property to divide violating paths into independent segments. Based on Bézier curve smoothing, fixability identification, and the propagation property, we develop an efficient algorithm to fix timing violations. Experimental results show that we can effectively resolve all timing violations with significant speedups over the state-of-the-art works.**

***Index Terms*—Engineering change order, logic synthesis, physical design, spare cell, timing optimization.**

I. Introduction

THE FAST-GROWING design complexity in modern integrated circuit (IC) design has caused some hard-to-detect design failures to be found at late design stages, e.g., the post-layout stage, or even the post-silicon stage. Instead of backtracking to earlier stages, metal-only engineering change order (ECO) is widely adopted in IC design houses to rectify late-found failures. As revealed by [3] and [4], the transistor-layer masks typically cost much more than metal-layer masks. Hence, metal-only ECO, keeping the transistor-layer masks intact and changing only metal-layer masks, is an economical way to save the mask cost and meet the strict production deadline. To facilitate metal-only ECO, spare cells (redundant cells) are inserted by a placement tool [14]. Once a design failure is detected, adequate spare cells are selected and rewired to correct it. A design typically undergoes many ECO runs in design houses; as a result, it is of particular importance to economically utilize spare cells to complete ECO.

A. Prior Work

ECO can be classified into functional ECO and timing ECO. Functional ECO is used to correct functional errors and/or revise specification. Kuo *et al.* introduced constant insertion to enhance the functional capabilities of spare cells [15]. Modi and Marek-Sadowska rely on simulated annealing to refine the Boolean cover and wirelength in [17]. Jiang *et al.* presented a stable matching-based ECO engine to minimize wirelength in [13]. Huang *et al.* adopted cut-based Boolean matching and then reduced wirelength by greedy heuristics in [12]. These works focus on functional ECO alone, and timing is not their main concern.

On the other hand, timing ECO is used to remedy signal imperfection and fix timing violations by gate sizing and/or buffer insertion. Chen *et al.* observed the shielding effect and loading dominance, defined a bounding polygon to reduce the solution space for spare cell selection, and presented a sequential approach based on dynamic programming in [8]. Fang *et al.* concurrently sized gates and inserted buffers along timing violating paths. They also reused redundant wires to increase routability. They reduced this problem to a multicommodity network flow and solved it by integer linear programming in [9]. Lu *et al.* deliberately inserted buffers into a routing tree to improve signal transition and delay in [16]. Ho *et al.* used cut-based Boolean matching to iteratively remap subcircuits from the gate with the worst negative slack (WNS) in [11]. Very recently, Chang *et al.* viewed gate sizing and buffer insertion as special cases of functional ECO and further proposed a simultaneous functional and timing ECO engine in [6].

B. Our Contributions

Timing safety is an essential requirement not only for timing ECO but also for functional ECO. Consequently, we shall focus on timing ECO optimization in this paper. Conventionally,

Manuscript received October 27, 2011; revised March 7, 2012 and May 18, 2012; accepted June 26, 2012. Date of current version November 21, 2012. This work was supported in part by SpringSoft, Synopsys, Faraday, TSMC, and NSC of Taiwan, under Grants NSC 99-2221-E-002-210-MY3, NSC 99-2221-E-002-207-MY3, NSC 98-2221-E-002-119-MY3, NSC 97-2221-E-002-237-MY3, NSC 100-2220-E-009-047, NSC 101-2220-E-009-044, and NSC 101-2628-E-009-012-MY2. A preliminary version was presented at the IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, November 2011 [7]. This paper was recommended by Associate Editor C. J. Alpert.

H.-Y. Chang is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: huayu.chang@gmail.com).

I. H.-R. Jiang is with the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: huiru.jiang@gmail.com).

Y.-W. Chang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: ywchang@cc.ee.ntu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2012.2209117

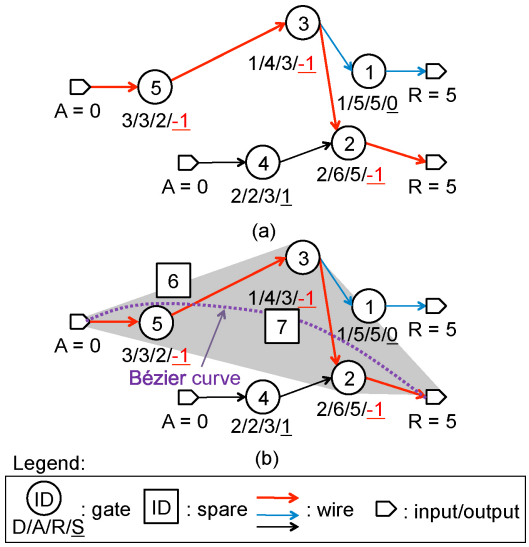

Fig. 1. Slack, delay, and timing criticality. (a) Example with a timing analysis result, where all gates along the most critical path, g_5, g_3, and g_2, have the same slack value, -1. (b) If g_5, the gate with the worst slack and the largest delay, is replaced with spare cell s_6, the timing improvement might be insufficient. However, gate g_3 is the most critical gate from the geometrical standpoint. If we fix gate g_3 using spare cell s_7 instead, the timing can be fixed.

the slack or delay of a gate is used to measure its timing criticality. However, we observe that neither slack nor delay can capture timing criticality well. Fig. 1 gives an example where each gate is associated with a delay (D), an arrival time (A), a required time (R), and a slack (S).[1] The critical path goes through gates g_5, g_3, and g_2, and each has the same slack value -1. The criticality among them cannot be differentiated based only on the slack value. If we select gate g_5 (since it has the WNS and largest delay) and then fix it by spare cell s_6, timing can be slightly improved. The timing improvement might be insufficient, and we need more spare cells to fix the timing violations. However, we observe that a path with good timing is desired to be geometrically smooth. From the geometrical standpoint, gate g_3 is the most critical cell (although it has the smallest delay). If we replace gate g_3 with spare cell s_7 instead, the timing can be optimized.

Based on the above observations, it is desirable to develop a new metric that can identify timing criticality more precisely. Timing criticality is related not only to slack or delay but also to geometrical information. With effective criticality identification, we could identify the truly critical gate(s) on a timing violating path, which can lead to fewer spare cells for timing fixing and thus preserve more resources for later ECO runs. Consequently, we address in this paper the cost-effective timing ECO problem: given a design with a set of preplaced spare cells, perform metal-only ECO timing optimization by gate sizing and buffer insertion such that all timing violations are resolved and the number of spare cells used is minimized.

For the addressed problem, we propose a new metric, fixability, to enhance the modeling of timing criticality based on the four criteria: flexibility, path sharing, spare-cell availability, and smoothness. Flexibility means the room for timing improvement. For each gate on a timing violating path, flexibility is determined not only by the impact of wire loading on its delay but also by the slack difference between the most and the second most critical paths through its fan-out. Path sharing indicates how many violating paths pass through a gate. Spare-cell availability considers the distribution of available spare cells. Smoothness models the closeness of gates to an ideally smooth path, captured by a Bézier curve that has the following properties: 1) the curve is completely contained in the convex hull[2] of its control points (i.e., gates on a timing violating path); 2) the curve does not wiggle any more than the control polyline formed by the control points and their corresponding lines; and 3) the curve is predictable. The more the timing violating path approaches to its Bézier curve, the better the resulting timing could be optimized. A special case with optimal timing is when all gates are aligned with a straight line connecting the beginning and ending points of a path. In this special case, the corresponding Bézier curve is a straight line. As shown in Fig. 1(b), gate g_3 is the most critical one, and it can be detected by the Bézier curve. Moreover, replacing gate g_5 with spare cell s_6 makes the timing violating path deviate from the Bézier curve, thus increasing the variation along the path. In contrast, if we repeatedly adjust a timing violating path to its corresponding Bézier curve, its variation along this path would be reduced.

In order to fix multiple timing violations concurrently, we explore the essence of timing slacks and the independency of timing violating paths. It can be shown that each timing violating path can be decomposed into segments, each having the same timing slack value. Moreover, we observe the propagation property: if we improve the delay of a gate, then the slack of any other gate on the same violating path segment is also improved by the same amount. With the propagation property, we can simultaneously optimize the timing for all violating path segments.

Based on Bézier curve smoothing, fixability identification, and the propagation property, we develop a timing ECO engine as follows. First of all, we decompose all timing violating paths into segments. Second, we extract the most critical gates for each segment. Third, we use minimum weight perfect matching to select adequate spare cells for these critical gates [10]. We repeat the process until all violations are resolved.

We summarize the features of our proposed approach as follows.

1) A new metric of timing criticality, fixability, is developed. Different from slack and/or delay, we incorporate the criteria of flexibility, path sharing, spare-cell availability, and path smoothness into the metric. Hence, we can further differentiate the criticalities along timing violating paths.
2) We adopt Bézier curves to measure the smoothness of timing violating paths and guide spare cell selection. As a result, the timing violating paths gradually converge to

[1]The arrival time (A) and required time (R) of a gate are defined with respect to its output pin. The slack (S) of a gate is defined as its required time minus its arrival time (S = R − A).

[2]The convex hull for a given set of points is the smallest convex polygon containing all points.

their corresponding Bézier curves, thus optimizing their timing.

3) Our approach is efficient. With the propagation property, we could fix the timing for all violating path segments concurrently. Furthermore, our minimum weight perfect matching method is very efficient, compared with [9], which is based on the NP-hard multicommodity flow method.
4) Our approach is cost effective. We fix timing violations from the most critical gates, and the cost setting in the bipartite graph favors the spare cells within the convex hull [18] of the corresponding path and the one that just meets the required amount of timing improvement. In particular, our results also lead to smaller spare cell consumption.

Compared with the state-of-the-art timing ECO engines [8], [9], our approach can fix all timing violations with significant speedups.

The remainder of this paper is organized as follows. Section II formulates the cost-effective timing ECO problem and briefly describes the timing model. Section III derives our slack properties that are the foundation of our algorithm. Section IV gives the formula of fixability and details our timing ECO engine. Section V shows the experimental results. Finally, Section VI concludes this paper.

II. Preliminaries and Problem Formulation

In this section, we introduce the timing model used in this paper and give the problem formulation.

A. Timing Library and Timing Model

A timing path is a circuit path that begins with a primary input or the data output pin of a flip-flop and ends with a primary output or the data input pin of a flip-flop. Timing constraints describe the clock period, the input or output delay, and timing exceptions.[3] Considering rise or fall delays, timing analysis reports the timing violating paths with respect to the timing constraints [19].

The cell timing and wire delay models used in this paper are the same as [8], [9], and [11]. In Synopsys' Liberty library [2], the calibrated values of rise or fall propagation delay and output transition of each library cell are stored in tables (four tables); all values in these tables are indexed by its input transition and output loading. For a sequential element, four additional tables are used to describe setup or hold time values.

Chen *et al.* [8] pointed out two observations. The shielding effect implies that sizing a gate or inserting a buffer influences only the delays of its fan-in and fan-out gates. Loading dominance means that the impact of changing the output loading on the gate delay is much larger than that of changing the input transition. The wire delay is lumped into the propagation delay of the gate driving this wire. The output loading of a gate contains wire loading, the input capacitance of fan-out gates, and its output pin capacitance. The wire loading of a gate is proportional to the summation of pairwise wirelength between this gate and each of its fan-out gates. The simplified wire delay model is used here for fair comparison with prior works, but a more sophisticated one can also be used.

[3]The input (output, respectively) delay means the delay consumed by the external environment for each primary input (output). Timing exceptions describe false paths and multicycle paths.

TABLE I
Notations

Notation	Description
g_i	gate (node) g_i
$e(i, j)$	wire (edge) between g_i and g_j
$D(i)$	gate delay of g_i
$A(i)$	arrival time of g_i
$R(i)$	required time of g_i
$R(i, j)$	edge required time of $e(i, j)$
$S(i)$	node slack of g_i
$S(i, j)$	edge slack of $e(i, j)$
P	timing violating path
$P(i, j)$	violating path segment from g_i to g_j
$T_x(i)$	timing fixability of g_i
$T_f(i)$	flexibility of g_i
$T_s(i)$	smoothness of g_i
$T_h(i)$	path sharing of g_i
$T_a(i)$	spare-cell availability of g_i
$T_l(i)$	wire loading impact of g_i
$T_d(i)$	slack difference of g_i
$D_0(i)$	gate delay with zero wire loading
$D_B(i)$	gate delay with deviation from Bézier curve
$w(g_{c_i}, g_{s_j})$	cost of matching gate g_{c_i} with spare g_{s_j}
$S(c_i \mid s_j)$	updated slack of matching g_{c_i} with g_{s_j}
α	user specified parameter used for $T_a(i)$
$n_s(i)$	number of spare cells inside g_i's bounding box

B. Problem Formulation

Since a design typically undergoes multiple ECO runs in design houses, it is of particular importance to economically use spare cells to complete ECO. Hence, we shall aim at timing ECO using the least number of spare cells. The problem formulation is thus described as follows.

Cost-Effective Timing ECO Problem: Given a placed design with a set of preplaced spare cells and timing violating paths, perform metal-only timing ECO using gate sizing and buffer insertion such that all timing violations are fixed and the number of spare cells used is minimized.

III. Slack Properties

In this section, we derive slack properties that are the foundation to our algorithm. Our derivation is based on manipulating the slack of an edge. We first summarize the notations used in this paper in Table I.

A circuit graph $H = (G, E)$ is used to represent a design, where each node $g_i \in G$ represents a gate that is associated with its delay $D(i)$, and each edge $e(i, j) \in E$ represents the wire between two gates $g_i, g_j \in G$. $D(i)$ can be the rise or fall delay of g_i. A primary input or output can be viewed as a gate whose delay is equal to its associated input or output delay. Similarly, the data output (input, respectively) pin of flip-flop is defined as a pseudo primary input (output). A (pseudo) primary output is the ending point of a circuit path.

Definition 1: In [19], the arrival time $A(i)$ of the output signal of node $g_i \in G$ is computed as

$$A(i) = \max_j\{A(j)|e(j,i) \in E\} + D(i) \quad (1)$$

while the required time $R(i)$ of node $g_i \in G$ is computed as

$$R(i) = \min_k\{R(i,k)|R(i,k) = R(k) - D(k), e(i,k) \in E\} \quad (2)$$

where $R(i,k)$ is the edge required time of $e(i,k)$. If g_i is a (pseudo) primary output, $R(i)$ is given according to the input design.

Definition 2: In [19], the edge slack $S(i,j)$ is the slack of edge $e(i,j)$ that is contributed from node g_j back to node g_i

$$S(i,j) = R(i,j) - A(i). \quad (3)$$

The node slack $S(i)$ of node g_i is the slack of node g_i

$$S(i) = \min_j\{S(i,j)|e(i,j) \in E\}. \quad (4)$$

The path slack is the node slack of its ending point.

Alternatively, the node slack can be computed as $S(i) = R(i) - A(i)$, which is equivalent to that in Definition 2. For example, as shown in Fig. 1(a), $S(3,1) = 4 - 4 = 0$, $S(3,2) = 3 - 4 = -1$, and $S(3) = -1$.

Definition 3: A timing violating path P is a simple path with $S(i,j) < 0$, $\forall e(i,j) \in P$.

For example, the path through gates g_5, g_3, and g_2 in Fig. 1(a) is a timing violating path. Based on the above definitions, we derive our slack definitions and properties as follows.

Lemma 1: There exists a most timing violating path on which every node slack and edge slack are identical.

Proof: To prove Lemma 1, it is equivalent to prove that we can find a path on which each gate g_{j_i} has a fan-in gate $g_{j_{i-1}}$ (i.e., $e(j_{i-1}, j_i) \in E$) such that $S(j_{i-1}) = S(j_{i-1}, j_i) = S(j_i)$. Lemma 1 can be proved by mathematical induction. Let P be the most timing violating path ending at g_{j_n}, as shown in Fig. 2. We shall find such a timing violating path as follows. By Definitions 2 and 3, P has the WNS $S(j_n)$. The ending point g_{j_n} has a unique fan-in gate, namely, $g_{j_{n-1}}$

$$\begin{aligned} S(j_n) &= R(j_n) - A(j_n) = R(j_n) - A(j_{n-1}) - D(j_n) \\ &= R(j_{n-1}, j_n) - A(j_{n-1}) \\ &= S(j_{n-1}, j_n) \\ &= \min_{i,j}\{S(i,j)|e(i,j) \in E\} \\ &= S(j_{n-1}). \end{aligned}$$

Hence, we can find such a gate $g_{j_{n-1}}$ for g_{j_n}. Assume that we can successfully find such fan-in gates backtracking from g_{j_n} to g_{j_i}

$$S(j_i) = S(j_i, j_{i+1}) = S(j_{n-1}, j_n) = S(j_n).$$

We shall find $g_{j_{i-1}}$ as follows:

$$\begin{aligned} S(j_i) &= S(j_i, j_{i+1}) = R(j_i, j_{i+1}) - A(j_i) \\ &= R(j_i, j_{i+1}) - (\max_k\{A(k)|e(k, j_i) \in E\} + D(j_i)) \\ &= R(j_i) - D(j_i) - \max_k\{A(k)|e(k, j_i) \in E\}. \end{aligned}$$

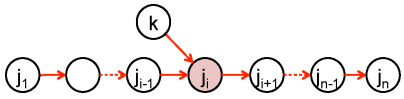

Fig. 2. Most timing violating path.

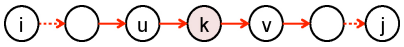

Fig. 3. Propagation property.

Let $g_{j_{i-1}} = \arg\max_k\{A(k)|e(k, j_i) \in E\}$.

$$\begin{aligned} S(j_i) &= R(j_{i-1}, j_i) - A(j_{i-1}) \\ &= S(j_{i-1}, j_i) \\ &= S(j_{i-1}). \end{aligned}$$

Hence, we can find such a path backtracking from the ending point to some starting point. ■

As shown in Fig. 1(a), the node slack and the edge slack along the timing violating path (through gates g_5, g_3, and g_2) are identical, i.e., -1.

Definition 4: A violating path segment $P(i,j)$ is a maximal subset of a timing violating path P, where every edge slack associated with this segment has the same value, and these edges are consecutive.

As a result, all timing violating paths can be decomposed into edge-disjoint violating path segments. By Lemma 1, the most timing violating path has only one violating path segment.

Definition 5: The slack difference $T_d(i)$ of node g_i is the difference between the worst and the second worst edge slacks of node g_i's critical fan-out

$$T_d(i) = \min_k\{S(k,j) - S(i)|S(i) = S(i,j), e(k,j) \in E\}. \quad (5)$$

The following theorem provides the feasibility of concurrently optimizing violating path segments.

Theorem 1: Propagation Property: If the delay of a node on a violating path segment $P(i,j)$ is improved (decreased) by t, $t \leq \min\{T_d(w)|g_w \in P(i,j)\}$, then each edge slack on the segment will also be improved (increased) by t.

Proof: Consider a violating path segment $P(i,j)$ as shown in Fig. 3, where nodes g_u, g_k, and g_v are consecutive, and $S(u) = S(k) = S(v)$. If the delay of node g_k on $P(i,j)$ is improved by t (t can be positive or negative), then we have the new delay, arrival time, and slack as follows:

$$\begin{aligned} D'(k) &= D(k) - t, \\ A'(k) &= A(u) + D(k) - t = A(k) - t \\ S'(k) &= R(k) - A'(k) = S(k) + t. \end{aligned}$$

Due to the choice of t, node g_k is still the most critical fan-out of g_u and the most critical fan-in of g_v. For g_u and g_v, the new slacks are

$$\begin{aligned} S'(u) &= S'(u,k) = S(k) + t \\ S'(v) &= R(v) - A'(v) = S(k) + t. \end{aligned}$$

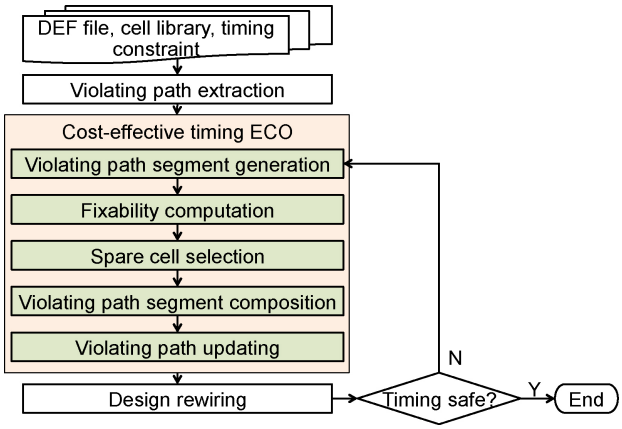

Fig. 4. Our timing ECO framework.

Similarly, we can claim that the slack of any node in between nodes g_i and g_u or in between nodes g_v and g_j is also improved by t. This property thus follows. ■

Because of the shielding effect and the propagation property, we can choose one or more nodes from each segment and concurrently optimize all segments. We will show in Section IV-D that if a node g_k is passed by multiple timing violating paths, all violating path segments passing through g_k should be improved by the amount of its negative slack to remove timing violations.

IV. Our Algorithm

In this section, we first present the new metric, named fixability, to model timing criticality, and then detail our algorithm based on the slack properties derived in Section III.

A. Overview

Fig. 4 depicts the flow for our timing ECO framework. Initially, the timing analysis engine reports all timing violating paths. Then, a six-step process is performed: 1) each timing violating path is decomposed into violating path segments (see Definition 4 in Section III); 2) for each segment, one or more timing critical nodes (in terms of fixability) are then extracted; 3) an appropriate spare cell is then selected for each extracted critical node based on minimum weight perfect matching [10]; 4) the refined violating path segments are combined, and redundantly selected spare cells are released; 5) timing violating paths are then re-extracted according to incremental timing analysis; and 6) finally, spare-cell rewiring is applied. This process is repeated until all violations are resolved.

B. Fixability Computation

As mentioned in Section I, neither slack nor delay can capture timing criticality well. In this section, therefore, we introduce a new metric of timing criticality, fixability, to identify the most critical gate on a violating path segment. The most critical gate means a gate that can induce the most delay improvement for this segment. Fixability (denoted by $T_x(i)$ for gate g_i) contains four terms: flexibility ($T_f(i)$), smoothness ($T_s(i)$), path sharing ($T_h(i)$), and spare-cell availability ($T_a(i)$). Flexibility and smoothness are modeled as delay penalties,

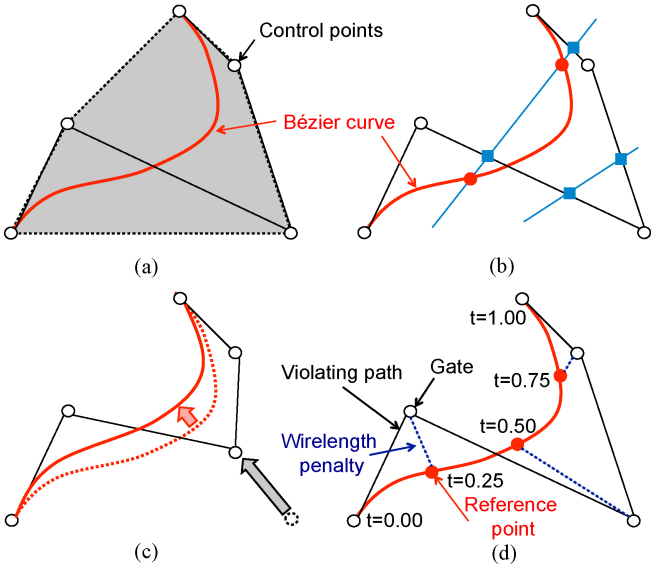

Fig. 5. Bézier curve. (a) Convex hull: the Bézier curve is completely contained inside the convex hull of given control points. (b) Variation diminishing property: the Bézier curve does not wiggle any more than the control polyline. (c) High predictability: if one control point is moved toward the Bézier curve, the updated curve moves accordingly. (d) Timing violating path with five gates, where the reference point of each gate is obtained by setting t by (9) as 0.0, 0.25, 0.5, 0.75, and 1.0.

while path sharing is a weighting function for the timing criticality, and spare-cell availability is a scale factor to represent the impact of available spare cells

$$T_x(i) = (T_f(i) + T_s(i))T_h(i)T_a(i), \forall g_i \in G. \tag{6}$$

1) *Flexibility:* Flexibility captures the room for timing improvement, and considers two factors: one is the impact of wire loading (denoted by $T_l(i)$), and the the other is the slack difference between the most and the second most critical paths through this gate's fan-out ($T_d(i)$). $T_l(i)$ for a gate is the difference of its current gate delay $D(i)$ and that without the wire loading $D_0(i)$

$$T_l(i) = D(i) - D_0(i), \forall g_i \in G. \tag{7}$$

This factor implies that the maximum improvement can be obtained by wire loading reduction. As defined in Section III, $T_d(i)$ gives the upper bound of the impact on slack when the delay of this gate is improved. We have

$$T_f(i) = \min(T_l(i), T_d(i)). \tag{8}$$

2) *Smoothness:* Conceptually, a path with good timing is desired to be geometrically smooth. We need an ideally smooth path as the baseline to measure the smoothness of path. In this paper, we use Bézier curve as the ideal path. Bézier curves are widely used in computer graphics to model smooth curves.

Definition 6: From [5], given a set of $n+1$ control points $c_0, c_1, \ldots, c_n$, the corresponding Bézier curve is given by the weighted sum of control points

$$C(t) = \sum_{i=0}^{n} c_i B_{\text{in}}(t) \tag{9}$$

where $B_{\text{in}}(t)$ is a Bernstein polynomial and $t \in [0, 1]$.

Bézier curves have the following nice properties.

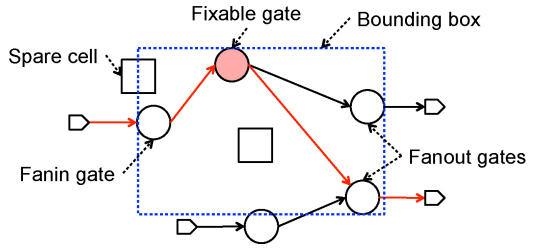

Fig. 6. Spare-cell availability. The spare cells inside the bounding box are typically better than those outside the box.

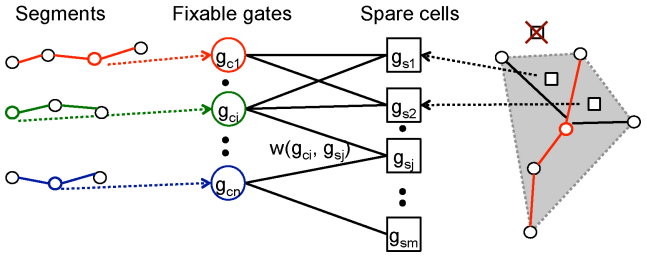

Fig. 7. Spare cell selection by minimum weight perfect matching. $w(g_{ci}, g_{sj})$ is the cost of selecting spare cell g_{sj} for fixable gate g_{ci}.

1) *Convex hull*: A Bézier curve must be completely contained inside the convex hull [18] of the control points. The convex hull is the smallest convex polygon containing these points [see Fig. 5(a)].
2) *Variation diminishing*: As shown in Fig. 5(b), the number of intersection points of any straight line with a Bézier curve (dots) is at most the number of intersection points of the same straight line with the control polyline formed by the control points and their corresponding lines (boxes).
3) *High predictability*: The Bézier curve follows the corresponding control points. As shown in Fig. 5(c), therefore, adjusting the position of some control point changes the shape of the Bézier curve in a predictable manner.

Because of the three properties, Bézier curves are suitable to model geometrically smooth paths. Moreover, if we move control points toward a Bézier curve, the curve gradually converges to a straight line [see Fig. 5(c)]. Here, we set the nodes on a timing violating path as the control points of a Bézier curve. These control points are expected to be evenly distributed along the path for timing optimization. Hence, we evenly sample the curve to find the reference point for each control point. For example, as shown in Fig. 5(d), we have a timing violating path with five gates. We then obtain the reference point for each gate by setting t by (9) as 0.0, 0.25, 0.5, 0.75, and 1.0.

The wirelength between a gate and its reference point is considered as a wirelength penalty [see Fig. 5(d))]. The gate delay with zero wire loading is considered as the baseline. Smoothness is defined as the added wire delay of the wirelength penalty with respect to zero wire loading

$$T_s(i) = D_B(i) - D_0(i), \forall g_i \in G \tag{10}$$

where $D_B(i)$ is the delay of gate g_i whose wire loading is the wirelength between g_i and its reference point. It should be noted that Bézier curves are used not only to compute smoothness but also to guide spare cell selection in our algorithm.

3) *Path Sharing:* The more timing violating paths pass through a gate, the more timing critical the gate is. Hence, $T_h(i)$ is defined as the number of timing violating paths passing through gate g_i. $T_h(i)$ can be obtained by parsing the timing report.

4) *Spare-Cell Availability:* ECO timing fixing cannot be done without spare cells. $T_a(i)$ is a scale factor corresponding to the impact of available spare cells. As shown in Fig. 6, we consider the number of spare cells inside the bounding box of the fixable gate and its fan-out or fan-in gates. The guidance is based on two motivations: one is for fast computation, and the other is that the spare cells inside the bounding box are typically better than those outside from the geometrical standpoint. Hence, $T_a(i)$ is defined as follows:

$$T_a(i) = \begin{cases} 1, n_s(i) > 0 \\ \alpha, n_s(i) = 0 \end{cases} \tag{11}$$

where α is a user-specified parameter $\in (0, 1]$, and $n_s(i)$ is the number of spare cells inside the bounding box. If $n_s(i) = 0$, α is used to reduce the fixability accordingly.

C. Spare Cell Selection

Based on the new metric of timing criticality given by (6), we extract one or more gates with the highest fixability from each violating path segment. By the shielding effect and the propagation property, we can fix timing by concurrently selecting an appropriate spare cell for each extracted fixable gate. For easier understanding, we explain the case of extracting one gate per segment first, and then extend to the case with multiple gates per segment later. Similar to [6], we also perform buffer insertion, and thus the buffers and inverter pairs on timing violating paths are initially removed before spare cell selection.

We reduce spare cell selection to minimum weight perfect matching [10]. The bipartite graph $B = (G_C, G_S, E_B)$ contains a set G_C of fixable gates, a set G_S of spare cell candidates, and a set E_B of edges connecting fixable gates and spare cell candidates. Let node $g_{c_i} \in G_C$ denote the most fixable gate on the ith violating path segment and spare cell $g_{s_j} \in G_S$ denote a spare cell candidate. A spare cell candidate may be used for gate sizing or buffer insertion. An edge $e(c_i, s_j) \in E_B$ is constructed if g_{s_j} is g_{c_i}'s spare cell candidate; the weight $w(g_{c_i}, g_{s_j})$ expresses the slack if g_{c_i} is matched to g_{s_j}. Fig. 7 shows a bipartite graph constructed for minimum weight perfect matching.

For cost-effective timing ECO, we prefer the spare cell that can just clean the violation (i.e., zero weight). $w(g_{c_i}, g_{s_j})$ is within the range [0, 2]

$$S'(c_i|s_j) = R(g_{ci}) - A(g_{sj}).$$
$$w(g_{ci}, g_{sj}) = \begin{cases} 0, & S'(c_i|s_j) = 0 \\ \dfrac{S'(c_i|s_j)}{\max_k\{S'(c_i|s_k)|S'(c_i|s_k) > 0\}}, & S'(c_i|s_j) > 0 \\ 1 + \dfrac{S'(c_i|s_j)}{\min_k\{S'(c_i|s_k)|S'(c_i|s_k) < 0\}}, & S'(c_i|s_j) < 0. \end{cases} \tag{12}$$

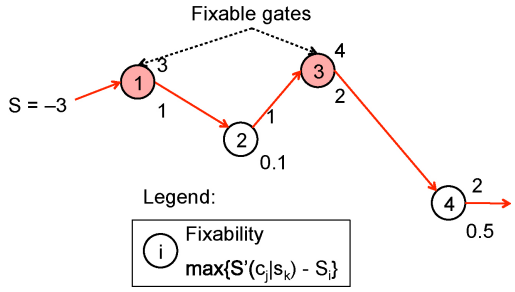

Fig. 8. Example of multiple fixable gate extraction. Two gates are extracted from this violating path segment.

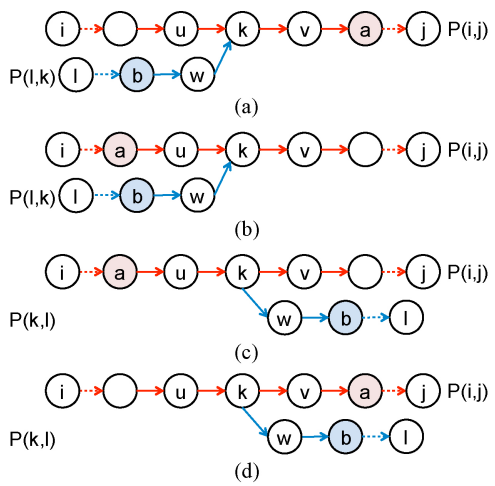

Fig. 9. Violating path segment composition.

To reduce the search space of spare cell candidates, for each extracted fixable gate, we first consider spare cells located within the convex hull of its fan-out gates (based on the loading dominance mentioned in Section II) and the gates on its corresponding timing violating path. If there exist no available spare cells that can improve timing inside the convex hull, we then enlarge the search region to the bounding polygon proposed by [8]. In addition, we omit the spare cells that cause more timing violations. If it fails to find any spare cell candidates, we switch to the second highest fixable gate. Based on the requirement of perfect matching on a bipartite graph, if fixable gates and included spare cells are of unequal cardinality, dummy gates or cells are added, and each related weight is set to a large constant value.

Once the selected fixable gates on two violating path segments are directly connected, the gate with higher fixability is handled first because of the shielding effect. Since we identify the highest fixable gate on each violating path segment, we can fix timing more cost effectively. Note that the selected spare cells are applied after violating path segment composition.

To further effectively utilize the higher fixability gates, we extend the extraction process from one gate per segment to multiple gates per segment. We try to estimate the number of extracted gates for each segment as follows. First of all, we sort and extract the gates for each segment in the nonincreasing order of fixabilities. Then, the number of gates to be extracted for each segment is determined as follows. Assume that some

Algorithm 1 Cost Effective Timing ECO

```
while there exist timing violating paths do
  Remove buffers and inverter pairs with large delay
  foreach timing violating path P do
    Decompose P into violating path segments
    Generate Bezier curve C for P
    Generate convex hull for P
  foreach P(i, j) in P do
    foreach gate g on P(i, j) do
      Compute fixability
  foreach P(i, j) do
    Extract critical gates and their related spare cells
  Apply minimum weight perfect matching
  Compose violating path segments
  Update violating paths
  Rewire design
```

violating path segment has slack $S_i < 0$ and gates $g_{c_1}, g_{c_2}, \ldots$ in the nonincreasing order of fixabilities. We have

$$S_i \geq -\sum_{j=1}^{n} \max_k \{S'(c_j|s_k) - S_i\} \tag{13}$$

where n is the number of extracted gates from this segment and $\max_k\{S'(c_j|s_k)-S_i\}$ is the upper bound of improvement on g_{c_j}'s slack. n is the minimum number of extracted gates to fix the timing violation of a segment. For example, as shown in Fig. 8, we have one violating path segment with slack -3. The fixability for each gate and its $\max_k\{S'(c_j|s_k) - S_i\}$ value are indicated beside each node. We extract gates from the highest fixability gate g_3 until the segment's slack is larger than or equal to the sum of estimated fixable slacks. In this example, we extract gates g_3 and g_1 $(-3 \geq -(2+1))$ as highly fixable gates for this segment. Similar to the extraction process in one gate per segment, once the selected fixable gates are directly connected, the gate with lower fixability will be ignored at this iteration.

D. Violating Path Segment Composition

After spare cell selection, all violating path segments are combined for timing check and/or the next iteration.

Without loss of generality, we consider two violating path segments that join or fork at some node g_k. Assume the original edge slack of $P(i, j)$ ($P(l, k)$, respectively) is S_1 (S_2). Their highest fixable gates are g_a and g_b, improved by t_1 and t_2. As shown in Fig. 9, there are four cases for merging these two violating path segments as follows.

Case 1) See Fig. 9(a). There are still two segments: $P(i, j)$ is of edge slack $S_1 + t_1$, and $P(l, k)$ is of edge slack $S_2 + t_1 + t_2$.

Case 2) See Fig. 9(b). There are three segments: $P(i, k)$ is of edge slack $S_1 + t_1$, $P(l, k)$ is of edge slack $S_2 + t_2$, and $P(k, j)$ is of edge slack $S_1 + \max\{A(u) + D(k) - t_1, A(w) + D(k) - t_2\}$.

Case 3) See Fig. 9(c). There are still two segments: $P(i, j)$ is of edge slack $S_1 + t_1$, and $P(k, l)$ is of edge slack $S_2 + t_1 + t_2$.

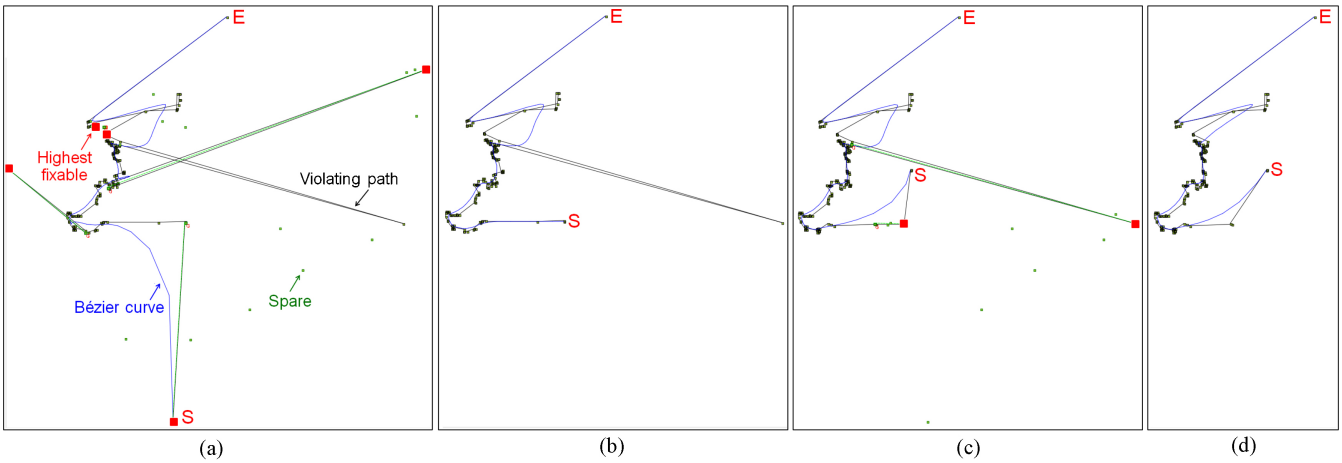

Fig. 10. Layout of Industry3. (a) At iteration 1, on the timing violating path from S to E, the corresponding Bézier curve and five selected fixable gates are highlighted. (b) Result after iteration 1. (c) At iteration 2, two fixable gates are selected in the re-extracted timing violating path. (d) Result after iteration 2.

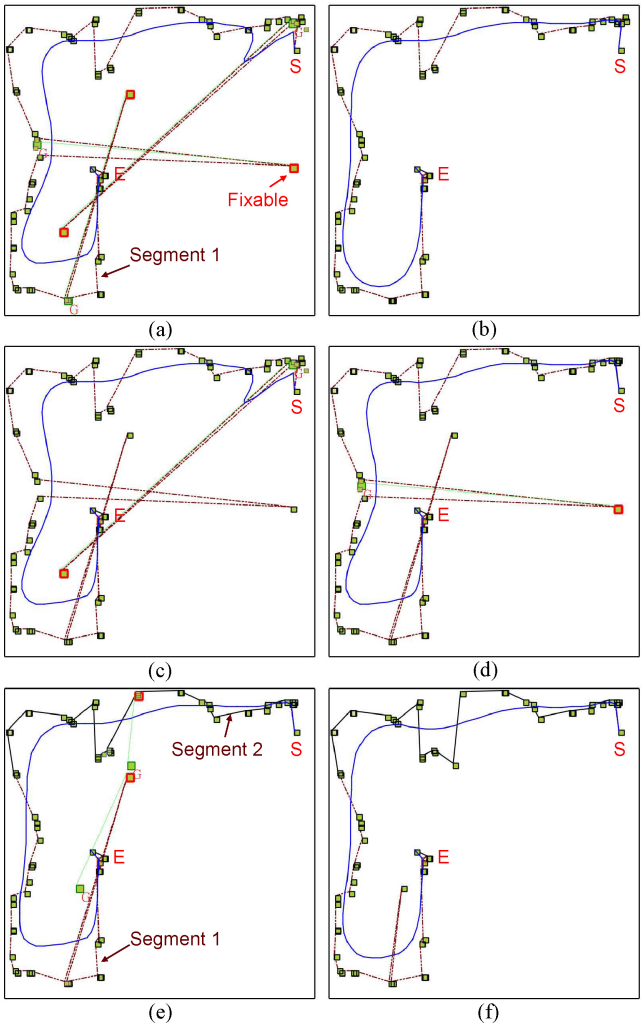

Fig. 11. Comparison between our basic framework and the final one. Here, we show one of the timing violating paths in Industry4. (a) and (b) From our final framework. (c)–(f) From our basic framework. (a) At iteration 1, three fixable gates are extracted from segment 1. (b) Result after iteration 1. (c) and (d) Iterations 1 and 2 for the basic framework. (e) At iteration 3, two fixable gates are extracted from segments 1 and 2. (f) Result after iteration 3.

TABLE II
STATISTICS OF THE BENCHMARK CIRCUITS

Circuit Name	Gate Count	#Spare Cells	Cycle (ns)	#Violating Paths	Max #Gate	#Gate Passed	WNS (ns)	TNS (ns)
Industry1	28 927	860	38	16	164	2604	1.1	9.8
Industry2	200 504	860	40	80	178	13 627	10.8	312.0
Industry3	91 107	860	37	27	173	4059	19.3	319.0
Industry4	18 932	860	18	22	85	1278	6.8	70.0
Industry5	38 011	8600	18	137	72	9160	2.8	161.0

Case 4) See Fig. 9(d). There are three segments: $P(i,k)$ is of edge slack $S_1+\min\{R(v)-D(v)+t_1, R(w)-D(w)+t_2\}$, $P(k,l)$ is of edge slack S_2+t_2, and $P(k,j)$ is of edge slack S_1+t_1.

To be cost effective, we can define redundant fixable gates based on the above classifications.

Definition 7: A fixable gate is redundant if its corresponding violating path segment is completely fixed by another fixable gate.

For example, in Fig. 9(a), if $S_2+t_1 \geq 0$, the fixable gate g_b is redundant. Hence, we apply only the selected spare cells of irredundant gates.

E. Complexity Analysis

Our overall procedure is summarized in Algorithm 1. The timing ECO is performed when there exist timing violating paths reported by the STA engine. Line 2 removes the buffers and inverter pairs with large delay. Then, in lines 3–6, for each timing violating path, we decompose it into path segments and generate the corresponding Bézier curve and convex hull. For each violating path segment, lines 7–9 compute fixability for each gate along the segment. Next, lines 10 and 11 extract critical gates and their related spare cells for each path segment. After that we apply minimum weight perfect matching in line 12 to select appropriate spare cells, compose violating path segments in line 13, and update violating paths in line 14. Finally, spare cells are rewired in line 15.

Assume there are n timing violating paths and k spare cells; each path has m gates. ($nm << |G|$ and $k << |G|$.) Bézier

TABLE III
COMPARISON BETWEEN [8], [9], AND OUR METHOD

Circuit Name	Initial			DCP [8]		ILP [9]		Ours							
	TNS (ns)	#Violating Paths	#Spare Cells	TNS (ns)	Runtime (s)	TNS (ns)	Runtime (s)	TNS (ns)	Runtime (s)	#Resulting Spares	#Used Buffers	#Released Buf./Inv.	#Ite.	Max #Path Shared	#Path Segments
Industry1	9.8	16	860	0.00	6.12	0.00	42.42	0.00	0.28	880	0	8/12	1	16	19
Industry2	312.0	80	860	0.00	25.71	0.00	109.48	0.00	3.90	893	0	7/26	2	30	138
Industry3	319.0	27	860	14.93	12.33	0.00	26.29	0.00	0.99	949	0	19/70	2	27	73
Industry4	70.0	22	860	6.27	24.31	0.00	55.41	0.00	0.05	898	0	32/6	1	9	22
Industry5	161.0	137	8600	0.00	1761.16	0.00	3182.87	0.00	9.69	8600	0	0/0	31	5	137
Ratio			1.00		122.71		229.14		1.00	1.01					

TABLE IV
COMPARISON BETWEEN OUR BASIC FRAMEWORK, RESOURCE-AWARE FIXABILITY, AND OUR FINAL FRAMEWORK

Circuit Name	Ours Final (+Multiple Extracted Gates Per Seg.)					+Resource-Aware Fixability					Basic [7]				
	Runtime (s)	#Resulting Spares	#Used Buffers	#Released Buf./Inv.	#Ite.	Runtime (s)	#Resulting Spares	#Used Buffers	#Released Buf./Inv.	#Ite.	Runtime (s)	#Resulting Spares	#Used Buffers	#Released Buf./Inv.	#Ite.
Industry1	0.28	880	0	8/12	1	0.39	879	1	8/12	1	0.38	879	1	8/12	1
Industry2	3.90	893	0	7/26	2	6.13	895	0	7/28	3	6.14	895	0	7/28	3
Industry3	0.99	949	0	19/70	2	1.14	944	1	17/68	2	1.11	944	1	17/68	2
Industry4	0.05	898	0	32/6	1	0.31	898	0	32/6	3	0.29	898	0	32/6	3
Industry5	9.69	8600	0	0/0	31	41.56	8553	47	0/0	74	53.01	8553	67	0/0	73
Ratio	1.00		0.00	1.00	1.00	3.32		0.71	1.00	2.24	4.09		1.00	1.00	2.22

curve generation for one timing violating path can be done in $O(m^2)$ time. The running time for generating the convex hull for one timing violating path is $O(m^2)$ in the worst case. Extracting critical gates and their related spare cells can be done in $O(nmk)$ time. The general approach of minimum weight perfect matching takes $O(X^2Y)$ running time, where $X = nm+k$ and $Y = nmk$. All other steps could be performed in $O(nm)$ time. Hence, the overall running time of our framework for one iteration is $O(nm + nm^2 + nmk + X^2Y) = O(X^2Y) = O(n^3m^3k + nmk^3 + n^2m^2k^2)$.

V. EXPERIMENTAL RESULTS

Our algorithm was implemented in the C++ programming language with a LEDA package [1] on a Linux workstation with a 2.33 GHz Intel Xeon CPU and 12 GB memory.

The experiments were conducted with five industrial benchmark circuits adopted in [8] and [9]. The statistics of these circuits are summarized in Table II, including the benchmark name (Circuit Name), the number of gates in each design (Gate Count), the number of available spare cells (#Spare Cells), the clock period (Cycle), the number of timing violating paths (#Violating Paths), the maximum number of gates on one timing violating path (#Max Gate), the total number of gates passed by the timing violating paths (#Gate Passed), the WNS, and the total negative slack (TNS). Same as the prior works in [8] and [9], the slack of a timing violating path is defined by the node slack of its ending point. Hence, WNS is the amount of the negative slack of the most violating path, while TNS is the sum of the negative slacks of all reported violating paths. For fair comparison, the cell timing and wire delay models used in our experiments are the same as [8] and [9].

Table III gives the comparison of dynamic cost programming (DCP) method in [8], integer linear programming (ILP) method in [9], and our algorithm on TNS and runtime. We also list the number of resulting spare cells (#Resulting Spares) and the number of released buffers or inverters (#Released Buf./Inv.). Generally, the number of released spare cells is significantly larger than that of used. Hence, our algorithm is very cost effective. In addition, our algorithm can improve smoothness by the guidance of Bézier curves. Interestingly, it can be seen that our algorithm does not need any buffers to complete the timing ECO in these five cases.

We fixed all timing violations in the shortest runtimes and achieve 122.71X and 229.14X speedups over [8] and [9], respectively. Compared to the number of gates passed by all timing violating paths (#Gate Passed), the number of path segments (#Path Segments) is very small. Consequently, the problem size is significantly reduced. The maximum number of timing violating paths shared (Max #Path Shared) reveals that some gates are shared by multiple paths; once the timing violating paths are fixed, timing can be effectively improved. In particular, the number of iterations (#Ite.) is small, implying that our algorithm converges very fast. Fig. 10 shows the layout of Industry3, where timing violating paths are well fixed by the guidance of Bézier curves after only two iterations.

We have also performed two additional experiments to examine the effectiveness of our two key ideas. One is the resource-aware fixability, spare-cell availability consideration in the fixability, and the other is the multiple gate extraction scheme. Table IV summarizes the comparison of these two key ideas. "Basic" means our basic framework that does not consider spare-cell availability and extracts only one highest fixable gate at a time [7]. "+Resource-Aware Fixability" means our basic framework with spare-cell availability consideration. "Ours Final" means our final framework presented in this paper with resource-aware fixability and multiple gate extraction. If some fixable gates have no spare cells inside their bounding boxes, we should reduce their fixabilities, and α was set to 0.5 in these experiments accordingly. It can be

seen that the resource-aware fixability can guide fixability more accurately, and consume fewer buffers. Furthermore, the multiple gate extraction scheme reduces the number of iterations and buffer usage (even to zero). Compared with our basic framework, the running time for the final framework can be further reduced by 4.09X due to the two key ideas. Fig. 11 shows the effectiveness of our final framework. It can be seen that we can fix Industry4 in one iteration, while the basic framework takes three iterations and needs one more sized gate to fix all timing violations.

VI. Conclusion

Different from negative slack and gate delay, we presented a new metric of timing criticality. Based on the metric, we developed an efficient and cost-effective timing ECO engine. The engine adopted Bézier curves to smooth the timing violating paths, applied the propagation property to concurrently fix timing violations with all violating path segments, and used minimum weight perfect matching to select better spare cells. The experimental results showed the effectiveness and efficiency of our algorithm.

References

[1] *Library of Efficient Data Types and Algorithms Package*. (2012) [Online]. Available: http://www.algorithmic-solutions.com

[2] *Liberty: The EDA Library Modeling Standard*. (2012) [Online]. Available: http://www.opensourceliberty.org

[3] *International Technology Roadmap for Semiconductors. (ITRS)* (2009) [Online]. Available: http://www.itrs.net

[4] A. Balasinski, "Optimization of sub-100-nm designs for mask cost reduction," *J. Microlithography, Microfabricat., Microsyst.*, vol. 3, no. 2, pp. 322–331, Apr. 2004.

[5] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. San Mateo, CA: Morgan Kaufmann, 1998.

[6] H.-Y. Chang, I. H.-R. Jiang, and Y.-W. Chang, "Simultaneous functional and timing ECO," in *Proc. ACM/IEEE DAC*, Jun. 2011, pp. 140–145.

[7] H.-Y. Chang, I. H.-R. Jiang, and Y.-W. Chang, "Timing ECO optimization via Bézier curve smoothing and fixability identification," in *Proc. IEEE/ACM ICCAD*, Nov. 2011, pp. 742–746.

[8] Y.-P. Chen, J.-W. Fang, and Y.-W. Chang, "ECO timing optimization using spare cells and technology remapping," in *Proc. IEEE/ACM ICCAD*, Nov. 2007, pp. 530–535.

[9] S.-Y. Fang, T.-F. Jian, and Y.-W. Chang, "Redundant-wires-aware ECO timing and mask-cost optimization," in *Proc. IEEE/ACM ICCAD*, Nov. 2010, pp. 381–386.

[10] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logist. Quar.*, vol. 2, nos. 1–2, pp. 83–97, Mar. 1995.

[11] K.-H. Ho, J.-H. R. Jiang, and Y.-W. Chang, "TRECO: Dynamic technology remapping for timing engineering change orders," in *Proc. ACM/IEEE ASP-DAC*, Jan. 2010, pp. 331–336.

[12] S.-L. Huang, C.-A. Wu, K.-F. Tang, C.-H. Hsu, and C.-Y. R. Huang, "A robust ECO engine by resource-constraint-aware technology mapping and incremental routing optimization," in *Proc. ACM/IEEE ASP-DAC*, Jan. 2011, pp. 382–387.

[13] I. H.-R. Jiang and H.-Y. Chang, "ECOS: Stable matching based metal-only ECO synthesis," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 3, pp. 485–497, Mar. 2012. Also see in *Proc. ACM/IEEE DAC*, Jul. 2009, pp. 408–411.

[14] Z.-W. Jiang, M.-K. Hsu, Y.-W. Chang, and K.-Y. Chao, "Spare-cell aware multilevel analytical placement," in *Proc. ACM/IEEE DAC*, Jul. 2009, pp. 430–435.

[15] Y.-M. Kuo, Y.-T. Chang, S.-C. Chang, and M. Marek-Sadowska, "Engineering change using spare cells with constant insertion," in *Proc. IEEE/ACM ICCAD*, Nov. 2007, pp. 544–547.

[16] C.-P. Lu, M. C.-T. Chao, C.-H. Lo, and C.-W. Chang, "A metal only-ECO solver for input slew and output loading violations," in *Proc. ACM ISPD*, Mar. 2009, pp. 191–198.

[17] N. Modi and M. Marek-Sadowska, "ECO-map: Technology remapping for post-mask ECO using simulated annealing," in *Proc. IEEE ICCD*, Oct. 2008, pp. 652–657.

[18] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer, 1985.

[19] L.-T. Wang, Y.-W. Chang, and K.-T. Cheng, Eds., *Electronic Design Automation: Synthesis, Verification, and Testing*. New York: Elsevier, 2009.

Hua-Yu Chang received the B.S. degree from National Chengchi University, Taipei, Taiwan, in 1998, and the M.S. degree from National Chiao Tung University, Hsinchu, Taiwan, in 2001, both in computer science. He is currently pursuing the Ph.D. degree in electronic design automation with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei.

He has 11 years of experience in networking and computer graphics. His current research interests include physical design and logic synthesis.

Iris Hui-Ru Jiang (M'07) received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1995 and 2002, respectively.

From 2002 to 2005, she was with VIA Technologies, Inc., New Taipei, Taiwan. She is currently an Associate Professor with the Department of Electronics Engineering and the Institute of Electronics, NCTU. Her current research interests include physical design optimization and interaction between logic design and physical synthesis.

Dr. Jiang is a member of the ACM and the Phi Tau Phi Scholastic Honor Society.

Yao-Wen Chang (S'94–A'96–M'96–SM'12) received the B.S. degree from the National Taiwan University (NTU), Taipei, Taiwan, in 1988, and the M.S. and Ph.D. degrees from the University of Texas at Austin, Austin, in 1993 and 1996, respectively, all in computer science.

He is currently the Director of the Graduate Institute of Electronics Engineering and a Distinguished Professor with the Department of Electrical Engineering, NTU. His current research interests include very large-scale integration physical design and design for manufacturability. He has been working closely with the industry in these areas. He has co-edited one textbook on electronic design automation (EDA) and co-authored one book on routing. He has published over 200 ACM/IEEE conference or journal papers in these areas.

Dr. Chang was a recipient of six Best Paper Awards (ICCD, etc.) and 20 Best Paper Award nominations from DAC (five times), ICCAD (four times), and others, in the past 10 years. He has received many research and teaching awards, such as the Distinguished Research Award from the National Science Council of Taiwan (twice), the IBM Faculty Award, the CIEE Distinguished EE Professorship, the MXIC Young Chair Professorship, and the Excellent Teaching Award from NTU (seven times). He was the first-place winner of the 2012 DAC Placement Contest, the first-place winner of the 2011 PATMOS Timing Analysis Contest, and a five-time winner of the ACM ISPD Contests on placement, global routing, clock network synthesis, and discrete gate sizing. He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and the IEEE DESIGN AND TEST OF COMPUTERS. He is an editor of the *Journal of Information Science and Engineering*. He was the General and Steering Committee Chair of ISPD and the Program Chair of ASP-DAC, FPT, and ISPD. He is an IEEE CEDA Executive Committee Member, an ICCAD Executive Committee Member (the Technical Program Committee Vice Chair), an ASP-DAC Steering Committee Member, and an ACM/SIGDA Physical Design Technical Committee Member. He was a Technical Program Committee Member of major EDA conferences, including ASP-DAC, DAC, DATE, FPL, FPT, GLSVLSI, ICCAD, ICCD, ISPD, SLIP, SOCC, and VLSI-DAT. He was an Independent Board Director of Genesys Logic, Taipei, a Technical Consultant of Faraday, MediaTek, and RealTek, the Chair of the EDA Consortium of the Ministry of Education, Taiwan, and a member of the Board of Governors of the Taiwan IC Design Society.