# 國立交通大學

## 資訊科學與工程研究所

## 碩 士 論 文

為了工程變更時序最佳化的考慮拓樸的重新繞線

Topology-aware Rerouting for ECO Timing Optimization

研 究 生：童建勳

指導教授：李毅郎　教授

中 華 民 國　九 十 九　年 十 月

# 為了工程變更時序最佳化的考慮拓樸的重新繞線

# Topology-aware Rerouting for ECO Timing Optimization


研 究 生：童建勳　　Student：Jian-Syun Tong

指導教授：李毅郎　　Advisor：Yih-Lang Li


國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

October 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年十月

為了工程變更時序最佳化的考慮拓樸的重新繞線

研究生 : 童建勳　　指導教授: 李毅郎 博士

國立交通大學 資訊科學與工程研究所

# 摘要

由於增加的複雜度，現代設計產生越來越多的時序違反。時序工程變更有效地修正時序違反，而不需要重新設計整個晶片，並且可以有效地縮短再製造時間。慣例上插入緩衝器只專注於最小化最嚴重接點的延遲，卻忽略了障礙物和已繞線的區域。然而，忽略多接點網路的拓樸和障礙物，會使其他接點的延遲在插入緩衝器之後變得更糟。這篇論文導出兩個在插入緩衝器上的拓樸作用：打斷邊並且連接緩衝器和重新連結緩衝器。基於這些作用，這篇論文提出一個新穎的、有考慮已繞線區域的考慮拓樸的工程變更時序最佳化法， 來改進時序違反的接點，並且不會使其他接點的延遲變差。實驗結果顯示，提出的演算法可以平均地降低測資的最差負數延遲時間達 75.3%和總合負數延遲時間達 76.3%。相較於慣例上的以兩接點網路為基礎的插入緩衝器，這裡提出的演算法有更好的延遲改進，平均達到 35.2%，但需要花較長的執行時間。

**Topology-aware Rerouting for ECO Timing Optimization**

**Student: Jian-Syun Tong Advisor: Dr. Yih-Lang Li**

**Institute of Computer Science and Engineering**

**National Chiao Tung University**

# Abstract

Modern designs cause more and more timing violations due to the increased complexity. Timing ECO effectively fixes the timing violations incrementally without requiring redesigning the whole chips, and turnaround time can then be diminished significantly. Conventional buffer insertion only focuses on minimizing the delay of one critical sink and ignores the existing obstacles and routed wire segments when inserting buffers. However, ignoring the topology of one multi-pin net and obstacles may worsen the delays of other sinks after buffer insertion. This work derives two topology effects on buffer insertion: *edge breaking and buffer connection* and *buffer reconnection*. Based on the effects, this work presents a novel topology-aware ECO timing optimization considering routed wire segments to improve the delay of the violated sinks while preventing from worsening the delays of other sinks. Experimental results show that the proposed algorithm averagely improves the worst negative slack (WNS) and total negative slack (TNS) of benchmarks by 75.3% and 76.3%, respectively. Compared to the conventional 2-pin net-based buffer insertion, the proposed algorithm achieves better delay improvement by 35.2% on average at the cost of runtime.

# Acknowledgement

I am deeply grateful to my advisor, Dr. Yih-Lang Li for his continuous guidance, support, and ardent discussion throughout this research. His useful suggestions help me to complete the thesis. Also I express my sincere appreciation to all classmates in my laboratory for their encouragement and help. Especially I want to thank my senior classmate Yen-Hung Lin for his assistance and direction in this work.

This thesis is dedicated to my parents and my families for their patience, love, encouragement and long expectation.

# Contents

# List of Figures

# List of Tables

# Chapter 1
# Introduction

Interconnection delay has become the main factor to determine circuit delay in recent years because of the scaling of VLSI technology. However, precise timing information for critical paths/sinks with delay violations cannot be obtained before placement and routing (P&R). Timing violations can be fixed by redesign, but redesign is time-consuming and requires considerable efforts. Fortunately, engineering change order (ECO) provides an effective approach to fix timing violation or functional correctness after P&R. ECO utilizes the pre-placed spare cells to partially modify design. Spare cells are redundant cells. Different chip designs have different type and number of spare cells. For spare-cell placement, Chang *et al.* [1] proposed a comprehensive analysis on the configurations of spare-cell types and the strategies of spare-cell insertion. Jiang *et al.* [2] proposed a spare-cell-aware analytical placement framework which predicts the spare cell requirement and considers spare cell insertion during global placement.

ECO contains timing ECO [3]-[5] and functional ECO [6][7]. For timing ECO, Chou *et al.* [3] proposed a two-stage ECO algorithm Timing-Driven ECO Routing algorithm (TDER) by modifying detailed-routed nets to reduce delays of critical sinks. In the first stage, TDER selects a pair of suitable subtrees along the trunk, which is defined as a path from the source pin to the critical sink of a net, and then merges them. In the second stage, TDER determines the positions of Steiner points along the trunk to reduce the delay of critical sinks. Chen *et al.* [4] proposed a timing ECO framework considering buffer insertion and gate sizing simultaneously. They presented a dynamic programming algorithm considering the dynamic cost, called

dynamic cost programming (DCP). Chang *et al.* [5] proposed a framework, named MOESS, to solve the input-slew and output-loading violations by connecting spare cells onto the violated nets as buffers. MOESS provides two buffer insertion schemes performed sequentially to minimize the number of inserted buffers and then to solve timing violations. On the other hand, for functional ECO, Chang *et al.* [6] proposed a matching-based ECO synthesizer, ECOS. ECOS correctly implements the incremental design changes using the available spare cells, and also tries to reduce the prohibitive photomask cost at the same time. Tseng *et al.* [7] proposed an approach with two main steps: (1) technology remapping and (2) spare cell selection. In technology remapping step, their approach considers resource constraints. Traditional technology mapping might potentially exhaust resources or be unable to find suitable resources. In spare cell selection, they regard this problem as a question of resource allocation with the objective of simultaneously selecting the suitable spare cells to achieve functional changes and minimizing the increased wirelength.

This work focuses on timing ECO. Chou *et al.* [3] only considered one critical sink of a routed net. Chen *et al.* [4] considered the two-pin net in the timing path and ignore the multi-pin net topology and the impact on the delays of other sinks of the same net when selecting inserted buffers. Chang *et al.* [5] considered multiple pins of a net but do not consider the topology of the net either. Moreover, Chen *et al.* [4] and Chang *et al.* [5] do not perform detailed routing to connect the selected spare cell with the net. Therefore, the accuracy of spare cell selection of previous researches is not enough because of disregarding the net topology and detailed routed wires.

To enhance the accuracy of buffer insertion, this work simultaneously considers the routed wires and the multi-pin net topology during buffer insertion. Moreover, this is the first work to improve the delay of critical sinks without degrading those of other sinks. This work presents a topology-aware ECO timing optimization algorithm to

modify a detailed-routed net such that the interconnection delays of all violated sinks in the net are minimized without creating additional violated sinks and modifying any other detailed-routed net.

The rest of this paper is organized as follows. Section II gives preliminaries and formulates the ECO timing optimization problem. Section III presents the topology effects on buffer insertion. Section IV presents our topology-aware ECO timing optimization algorithm. Section V reports the experimental results. Finally, we conclude our work in Section VI.

# Chapter 2
# Preliminaries

Timing path is a unit of timing ECO optimization and is defined as a path from fanin or register to register or fanout. To improve the delay of the timing path, ECO timing optimization inserts one buffer between two gates along the timing path by breaking the original interconnection and re-wiring the gates to the inserted buffers. Spare cells in the chip can be treated as the candidates of inserting buffers. Fig. 1 shows an example of timing ECO optimization in a timing path where $g_1$-$g_2$-$g_3$ is the timing path and $gs_1$ and $gs_2$ are spare cells. The timing path in Fig. 1(a) with a slack equal to -0.8 violates the timing constraint. After inserting $gs_1$ and $gs_2$ in the timing path, the timing violation is solved, as shown in Fig. 1(b).

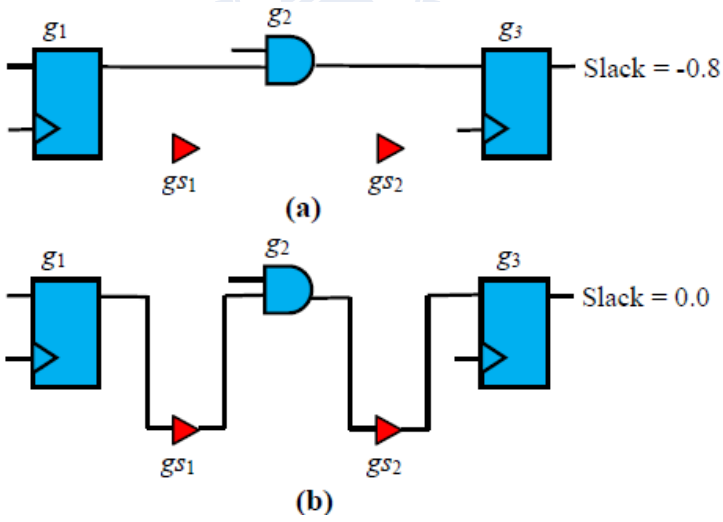


**Fig. 1. Example of timing path: (a) a timing path before buffer insertion; (b) a timing path after buffer insertion and re-wiring.**

## 2.1 Elmore Delay Model

The Elmore delay model [8] provides quick delay estimation. Although the accuracy of Elmore delay model is not sufficiently precise, the goal of this work is to

reduce the relative delays of the sinks. Therefore, this work adopts the Elmore delay model with π-model to estimate delay. The Elmore delay model is introduced for further descriptions as follows. $e_{ni}$ represents the wire segment from node $n_i$ to its parent in $T(n_0)$, where $T(n_0)$ is a routing tree rooted at the source $n_0$ and a node represents a Steiner point or a pin. $c(e_{ni})$ represents the capacitance of $e_{ni}$. $r(e_{ni})$ represents the resistance of $e_{ni}$ plus total pin or via resistance of the parent node of node $n_i$. $C(n_i)$ is the total capacitance of $T(n_i)$, which includes the capacitances of all wire segments and all pin load capacitances belonging to $T(n_i)$.

The Elmore delay of wire segment $e_{ni}$ with $\pi$-model equals $r(e_{ni})\times(c(e_{ni})/2+C(n_i))$. Let $R_d$ be the output driver resistance at source $n_0$. The Elmore delay $t_{ED}(n_k)$ at the node $n_k$ is then computed as:

$$t_{ED}(n_k) = R_d \times C(n_0) + \sum_{e_{n_i} \in path(n_o, n_k)} r(e_{n_i}) \times (\frac{c(e_{n_i})}{2} + C(n_i)) \qquad (1)$$

,where $path(n_0, n_k)$ is the path from $n_0$ to $n_k$ in $T(n_0)$.

## 2.2 Problem Formulation

*Topology-aware buffer insertion*: In a post-routing design $D$, a buffer set $B = \{ b_1 , b_2 , \dots , b_m \}$ is placed in $D$ as spare cells. A detailed-routed net $N = (P, E)$ where $P$ and $E$ represents the pin and the edge sets, respectively. $P = \{p_0, p_1, p_2, \dots , p_n\}$ where $p_0$ is the driver and others are sinks. Each one in $E$ is a path between two Steiner points. Timing-related information of $N$, driver arrival time $T_{arr}(p_0)$, sink required time $T_{req}(p_i)$, $i = 1\dots n$, and sink delay $T_d(p_i)$, $i=1\dots n$, are given. Given a set of timing violation sinks $VP = \{vp_1, vp_2, \dots , vp_k\} \in P\text{-}\{p_0\}$, where $T_{arr}(p_0)+T_d(vp_i)$ exceeds $T_{req}(vp_i)$, $i=1\dots k$. By inserting a buffer in $B$ into $N$, such that the topology of $N$ is changed and the interconnection delays of the set $VP$ are minimized without creating new violation sinks and causing any Design Rule Violations (DRVs) in $D$ and

modifying other detailed-routed nets belonging to *D*.

# Chapter 3
# Topology Effects on Buffer Insertion

Conventional timing ECO algorithms focus on improving the delay of one timing path at a time. Therefore, to trace the given timing path, previous works [4] [5] handle 2-pin net for each segment in the timing path although the 2-pin net is partial of one multi-pin net. Focusing on optimizing the delay of one 2-pin net of one multi-pin net

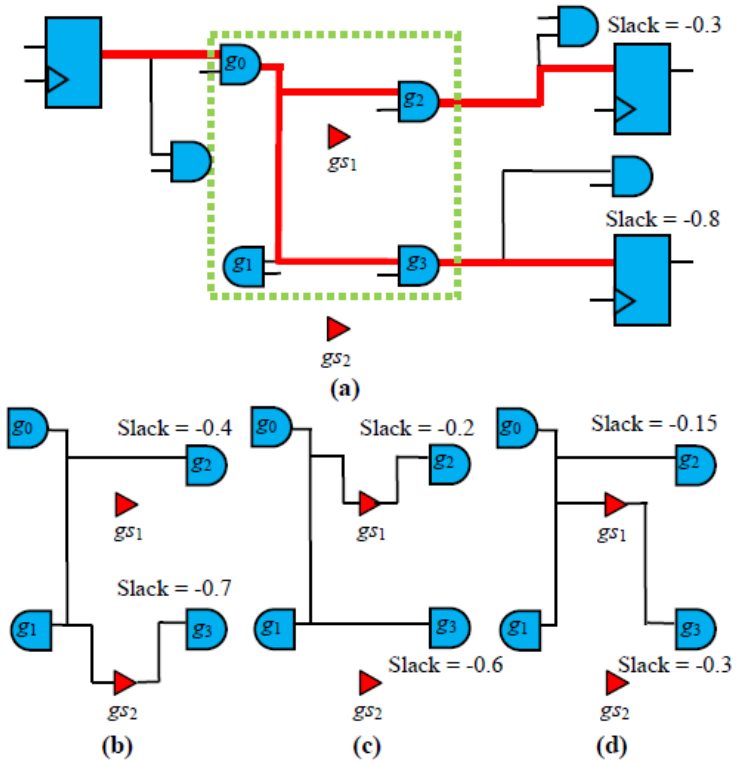


**Fig. 2. Example of buffer insertion on a multi-pin net: (a) two timing violation paths intersecting in a multi-pin net; (b) considering as 2-pin net with buffer insertion for the most critical timing path; (c) considering as multi-pin net with buffer insertion; (d) considering as multi-pin net with buffer insertion and buffer reconnection.**

for the most violation sink may degrades the delays of other sinks belonging to the same net, sequentially worsening other timing violation paths. Fig. 2(a) shows an example of two timing violation paths intersecting in one multi-pin net, where $g_0$-$g_3$ is

a part of the most critical path with slack equal to -0.8; $g_0$-$g_2$ is a part of the other violation path with slack equal to -0.3; and $gs_1$ and $gs_2$ are spare cells for buffering. In Fig. 2(b), considering the net as 2-pin net for buffer insertion increases the slack of the most critical path to -0.7 but decreases the slack of the other violation path to -0.4. In Fig. 2(c), considering the net as multi-pin net for buffer insertion increases the slacks of the most critical path and the other to -0.6 and -0.2, respectively. Moreover, reconnecting the buffer with shorter interconnections can further improve the delay of all sinks. In Fig. 2(d), reconnecting the buffer in Fig. 2(c) increases the slacks of the most critical path and the other to -0.3 and -0.15, respectively.

To insert one buffer in one multi-pin net, denoted as $N_m$, one edge, denoted as $E_{dis}$, in $N_m$ must be removed, and the inserted buffer, denoted as $B_{ins}$, connects to the two disconnected terminals due to removing $E_{dis}$. Fig. 3 depicts a net before/after buffer insertion. In Fig. 3(a), $e_{n(i+1)}$ is removed, and $E_{new1}$ and $E_{new2}$ are connected to the inserted buffer *buff*. In Fig. 3, $E_{dis}$ and $B_{ins}$ are $e_{n(i+1)}$ and *buff*, respectively. Choosing different pair of $E_{dis}$ and $B_{ins}$ leads to different delay effects on all sinks in $N_m$. The pair of $E_{dis}$ and $B_{ins}$ is defined as *buffering pair*, denoted as $pair_{buf}(E_{dis}, B_{ins})$. This work observes two topology effects on the buffer insertion in one multi-pin net from the Elmore delay: *edge breaking and buffer connection* and *buffer reconnection*. Edge breaking and buffer connection illustrates how to choose the most proper buffering pair for all sinks, even non-critical sinks, while buffer reconnection illustrates how to reconnect to the inserted buffer to further improve the delays of all sinks.

# 3.1 Edge Breaking and Buffer Connection Effect

The purpose of edge breaking and buffer connection is to find the proper buffering pair. Fig. 3 illustrates a multi-pin net, where $n_0$ is the source pin; $n_1$ to $n_j$ are Steiner

points; and $T_1$ to $T_j$ are subtrees of $T(n_0)$. In Fig. 3(b), $e_{n(j+1)}$ is removed and *buff* is inserted. Breaking the net by removing $e_{n(j+1)}$ separates the net into two sub-nets, one

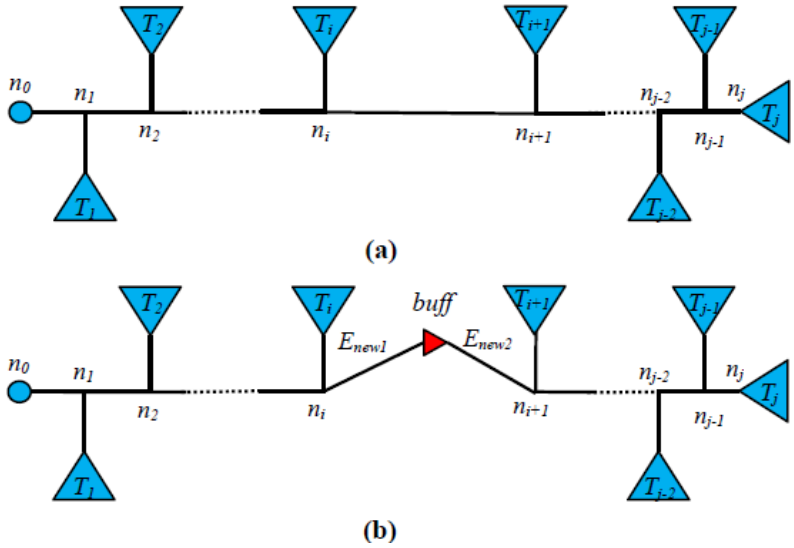


**Fig. 3. Illustration of edge breaking and buffer connection effect: (a) a net before buffer insertion; (b) one inserted buffer *buff* and two reconnections.**

including $n_0$ and the other excluding $n_0$, and $E_{new1}$ and $E_{new2}$ reconnect the two sub-nets by connecting to *buff*. Based on the Elmore delay model, the delay of each node in Fig. 3(a) is:

$$t_{ED}(n_k) = R_d \times C\_a(n_0) + \sum_{e_{n_y} \in path(n_0, n_k)} r(e_{n_y}) \times \left(\frac{c(e_{n_y})}{2} + C\_a(n_y)\right) \quad \text{, where } k = 1, \cdots, j. \qquad (2)$$

Because the topology of the net in Fig. (a) and in Fig. (b) are different, $C(n)$ in Fig. (a) and Fig. (b) are also different. Thus, we define $C(n)$ in Fig. (a) as $C\_a(n)$ and $C(n)$ in Fig. (b) as $C\_b(n)$.

The delay of each node in Fig. 3(b) is:

$$t_{ED}(n_k) = R_d \times C\_b(n_0) + \sum_{e_{n_y} \in path(n_0, n_k)} r(e_{n_y}) \times \left(\frac{c(e_{n_y})}{2} + C\_b(n_y)\right) \quad \text{, where } k = 1, \cdots, i. \qquad (3)$$

$$t_{ED}(n_k) = R_d \times C\_b(n_0) + \sum_{e_{n_y} \in path(n_0, n_i)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_b(n_y))$$
$$+ r(E_{new1}) \times (\frac{c(E_{new1})}{2} + c(buff)) + r(buff)$$
$$\times \left(c(E_{new2}) + C\_b(n_{i+1})\right) + r(E_{new2}) \times (\frac{c(E_{new2})}{2} + C\_b(n_{i+1}))$$
$$+ \sum_{e_{n_y} \in path(n_{i+1}, n_k)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_b(n_y))$$
$$, \text{ where } k = i+1, \cdots, j \qquad (4)$$

Thus, the delay difference of each node between Fig. 3(b) and Fig. 3(a) is:

$$\Delta t_{ED}(n_k) = (R_d + \sum_{e_{n_y} \in path(n_0, n_k)} r(e_{n_y})) \times (c(E_{new1}) + c(buff)$$
$$- c(e_{n_{i+1}}) - C\_a(n_{i+1})) \quad , \text{where } k = 1, \cdots, i. \qquad (5)$$

$$\Delta t_{ED}(n_k) = (R_d + \sum_{e_{n_y} \in path(n_0, n_i)} r(e_{n_y})) \times (c(E_{new1}) + c(buff) - c(e_{n_{i+1}})$$
$$- C\_a(n_{i+1})) + r(E_{new1}) \times (\frac{c(E_{new1})}{2} + c(buff)) + r(buff)$$
$$\times (c(E_{new2}) + C\_b(n_{i+1})) + r(E_{new2}) \times (\frac{c(E_{new2})}{2} + C\_b(n_{i+1}))$$
$$- r(e_{n_{i+1}}) \times (\frac{c(e_{n_{i+1}})}{2} + C\_a(n_{i+1})), \text{ where } k = i+1, \cdots, j \qquad (6)$$

Notably, the delay difference of each pin in $T_i$ is equal to the delay difference of $n_i$. Thus, when $\Delta t_{ED}(n_k)$ where $k$ = 1, …, $i$ is non-positive, the delay difference of each pin in the sub-net including $n_0$ is also not positive. Notably, the delay difference in the sub-net including n0 has the common term $c(E_{new1}) + c(buff) - c(e_{n(i+1)}) - C\_a(n_{i+1})$, and the delay of the sub-net including $n_0$ can be improved by satisfying the following inequality.

$$c(E_{new1}) + c(buff) - c(e_{n_{i+1}}) - C\_a(n_{i+1}) \le 0 \qquad (7)$$

In other words, the delay of each pin in the sub-net including $n_0$ can be improved by inserting the buffer.

The negative delay difference in (6) implies that the buffer insertion can reduce the

delay of the sub-net excluding $n_0$. Therefore, we derive the following inequality based on (6).

$$(R_d + \sum_{e_{n_y} \in path(n_0, n_i)} r(e_{n_y})) \times (c(E_{new1}) + c(buff) - c(e_{n_{i+1}})$$
$$- C\_a(n_{i+1})) + r(E_{new1}) \times (\frac{c(E_{new1})}{2} + c(buff)) + r(buff)$$
$$\times (c(E_{new2}) + C\_b(n_{i+1})) + r(E_{new2}) \times (\frac{c(E_{new2})}{2} + C\_b(n_{i+1}))$$
$$- r(e_{n_{i+1}}) \times (\frac{c(e_{n_{i+1}})}{2} + C\_a(n_{i+1})) \le 0 \qquad (8)$$

On other words, satisfying (8) guarantees that the delays of all pins in the sub-net excluding $n_0$ can be improved by inserting the buffer.

In conclusion, choosing a proper buffering pair that satisfies both (7) and (8) guarantees that the delay improvement of all sinks after buffer insertion.

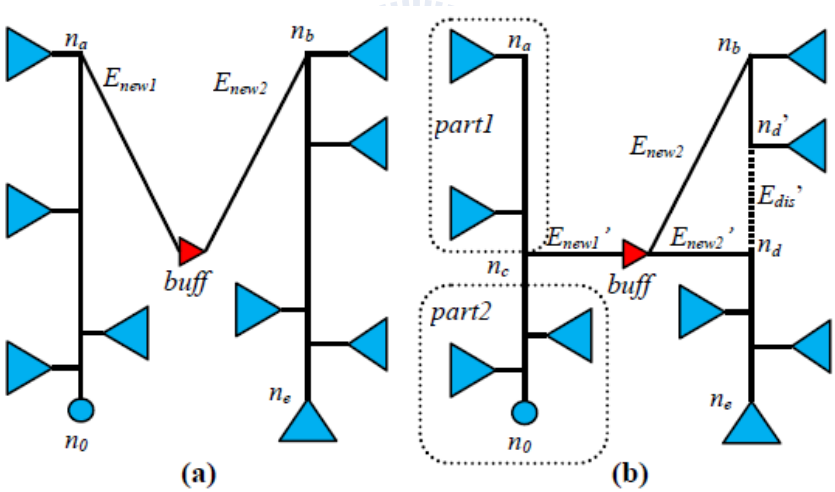


**Fig. 4. Illustration of buffer reconnection effect: (a) a net after buffer insertion; (b) a net after buffer reconnection.**

## 3.2 Buffer Reconnection Effect

After buffer insertion, one net can be partitioned into two parts: one includes the driver and the other excludes the driver. The delays of all pins in these two parts could be further improved by reconnecting them to the inserted buffer. Fig. 4(a) shows a net after buffer insertion where $n_0$ is the driver, and $n_a$, $n_b$, and $n_e$ are Steiner points. The

part including $n_0$ reconnects to the buffer by finding the nearest path from *buff* to the *path*($n_0$,$n_a$), as shown in Fig. 4(b) where $E_{new1'}$ is the nearest path from *buff* to the *path*($n_0$,$n_a$) and $n_c$ is the intersection point. After reconnection, the delay differences of the sub-net including $n_0$ can be partitioned into three parts: *part1*, *part2*, and *buff* as shown in Fig. 4(b). The delay of each node in the sub-net including $n_0$ in Fig. 4(a) is:

$$t_{ED}(n_i) = R_d \times C\_a(n_0) + \sum_{e_{n_y} \in path(n_0, n_i)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_a(n_y)) \quad (9)$$

where $n_i$ is not including *buff*.

$$t_{ED}(buff) = R_d \times C\_a(n_0) + \sum_{e_{n_y} \in path(n_0, n_a)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_a(n_y)) + r(E_{new1}) \times (\frac{c(E_{new1})}{2} + c(buff)) \quad (10)$$

The delay of each node in the sub-net including $n_0$ in Fig. 4(b) is:

$$t_{ED}(n_i) = R_d \times C\_b(n_0) + \sum_{e_{n_y} \in path(n_0, n_i)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_b(n_y)) \quad (11),$$

where $n_i$ is not including *buff*.

$$t_{ED}(buff) = R_d \times C\_b(n_0) + \sum_{e_{n_y} \in path(n_0, n_c)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_b(n_y)) + r(E_{new1}') \times (\frac{c(E_{new1}')}{2} + c(buff)) \quad (12)$$

Thus, the delay difference of *part1* between Fig. 4(b) and Fig. 4(a) is in the following:

$$\Delta t_{ED}(n_i) = (\sum_{e_{n_y} \in path(n_0, n_c)} r(e_{n_y})) \times c(E_{new1}') - (\sum_{e_{n_y} \in path(n_0, n_i)} r(e_{n_y})) \times c(E_{new1}) \quad (13),$$

where $n_i$ is the pins in *part1*. $\Delta t_{ED}(n_i)$ has a higher possibility to become negative when $E_{new1}$' is shorter than $E_{new1}$. The delay difference of *part2* between Fig. 4(b) and Fig. 4(a) is in the following.

$$\Delta t_{ED}(n_i) = (\sum_{e_{n_y} \in path(n_0, n_i)} r(e_{n_y})) \times (c(E_{new1}') - c(E_{new1})) \quad (14),$$

where $n_i$ is the pins in *part2*. Similarly, $\Delta t_{ED}(n_i)$ has a higher possibility to become negative when $E_{new1}$' is shorter than $E_{new1}$. The delay difference of buff between Fig. 4(b) and Fig. 4(a) is in the following.

$$\Delta t_{ED}(buff) = (\sum_{e_{n_y} \in path(n_0, n_c)} r(e_{n_y})) \times (c(E_{new1}') + c(buff))$$
$$+ r(E_{new1}') \times (\frac{c(E_{new1}')}{2} + c(buff)) - (\sum_{e_{n_y} \in path(n_0, n_a)} r(e_{n_y})) \times (c(E_{new1}) + c(buff))$$
$$- r(E_{new1}) \times (\frac{c(E_{new1})}{2} + c(buff))$$
$$- \sum_{e_{n_y} \in path(n_c, n_a)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_a(n_y) - c(E_{new1}) - c(buff)) \qquad (15)$$

$\Delta t_{ED}(buff)$ has a higher possibility to become negative when $E_{new1}$' is shorter than $E_{new1}$. Therefore, according to (13), (14), and (15), we can derive the delay of each pin in the sub-net including $n_0$ could be improved when reconnecting to *buff* through the nearest path.

In the sub-net excluding $n_0$, separating it into more than one part can effectively decrease the downstream capacitance of each part based on the Elomre delay model. Thus, the delay of the pin in the sub-net excluding $n_0$ can be further improved. However, separating one sub-net into more than one part requires reconnecting these parts to *buff*, sequentially resulting in more wire length and downstream capacitance. The delay improvement diminishes when the number of parts is too much. Therefore, this work only partitions the sub-net excluding $n_0$ into two parts at most. The nearest path from *buff* to *path*($n_b$, $n_e$) is found. In Fig. 4(b), $E_{new2}$' is the nearest path from *buff* to *path*($n_b$, $n_e$), and $n_d$ is the additional Steiner point. The edge $E_{dis}$' between $n_d$ and the upper stream Steiner point of $n_d$ is disconnected to separate the sub-net into two parts. Then, the delays of all pins in the parts including $n_b$ and $n_d$, respectively, in Fig. 4(a) are in the following.

$$t_{ED}(n_b) = R_d \times (c(E_{new2}) + C\_a(n_b)) + r(E_{new2}) \times (\frac{c(E_{new2})}{2} + C\_a(n_b)) \qquad (16)$$

$$t_{ED}(n_d) = R_d \times (c(E_{new2}) + C\_a(n_b)) + r(E_{new2}) \times (\frac{c(E_{new2})}{2} + C\_a(n_b)) + \sum_{e_{n_y} \in path(n_b, n_d)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_a(n_y)) \quad (17)$$

The delays of all pins in the parts including $n_b$ and $n_d$, respectively, in Fig. 4(b) are in the following.

$$t_{ED}(n_b) = R_d \times (c(E_{new2}) + C\_b(n_b) + c(E_{new2}') + C\_b(n_d)) + r(E_{new2}) \times (\frac{c(E_{new2})}{2} + C\_b(n_b)) \quad (18)$$

$$t_{ED}(n_d) = R_d \times (c(E_{new2}) + C\_b(n_b) + c(E_{new2}') + C\_b(n_d)) + r(E_{new2}') \times (\frac{c(E_{new2}')}{2} + C\_b(n_d)) \quad (19)$$

Thus, the delay differences of all pins in the parts including $n_b$ and $n_d$, respectively, between Fig. 4(b) and Fig. 4(a) are in the following.

$$\Delta t_{ED}(n_b) = R_d \times (c(E_{new2}') - c(E_{dis}')) - r(E_{new2}) \times (c(E_{dis}') + C\_a(n_d)) \quad (20)$$

$$\Delta t_{ED}(n_d) = R_d \times (c(E_{new2}') - c(E_{dis}')) + r(E_{new2}') \times (\frac{c(E_{new2}')}{2} + C\_b(n_d)) - r(E_{new2}) \times (\frac{c(E_{new2})}{2} + C\_a(n_b)) - \sum_{e_{n_y} \in path(n_b, n_d)} r(e_{n_y}) \times (\frac{c(e_{n_y})}{2} + C\_a(n_y)) \quad (21)$$

If $\Delta t_{ED}(n_b)$ is negative, the delay of each pin passing through $n_b$ is improved. If $\Delta t_{ED}(n_d)$ is negative, the delay of each pin passing through $n_d$ is improved. By (20) and (21), we can find that if $E_{new2}'$ is shorter, $\Delta t_{ED}(n_b)$ and $\Delta t_{ED}(n_d)$ are more possible to be negative.

# Chapter 4
# Topology-aware ECO Timing Optimization

Fig. 5 depicts the proposed topology-aware ECO timing optimization flow. Given a DEF file, LEF file, .lib file, and violation sink information, the *score* of each buffering pair is calculated based on the two topology effects on buffer insertion. The buffering pair with the highest score is selected for buffer insertion. After edge breaking and buffer connection, if the inserted buffer can satisfy the constraints which will be introduced in the following subsection, *buffer reconnection* modifies the two sub-nets to further improve the timing. Otherwise, the buffering pair is re-selected until the selected buffer satisfies the constraints. Finally, the modified DEF file is output. *Edge breaking and buffer connection* and *buffer reconnection* are introduced prior to the *buffering pair score computation* which is based on them.

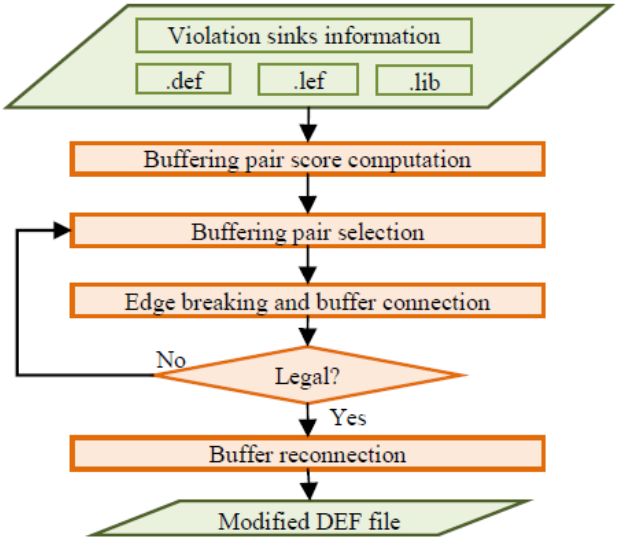


**Fig. 5. The flow of the topology-aware ECO timing optimization.**

## 4.1 Edge Breaking and Buffer Connection

This operation is based on the first derived effect: edge breaking and buffer connection effect. Given a detailed-routed net $N_m$ and a buffering pair, the edge $E_{dis}$ is broken, and the generated sub-nets are connected to the buffer *buff* through two additional connections $E_{new1}$ and $E_{new2}$, as illustrated in Fig. 3. Noticeable, connecting sub-nets to the buffer adopts detailed routing aware routed wire segments and obstacles. If $E_{dis}$, $E_{new1}$, and $E_{new2}$ can satisfy (7) and (8), the buffering pair is legal. Otherwise, the buffering pair is illegal.

## 4.2 Buffer Reconnection

This operation is based on the second derived effect: buffer reconnection effect. In Fig. 4(b), buffer reconnection routes the additional interconnections, such as $E_{new1'}$ and $E_{new2'}$, aware routed wire segments and obstacles. Therefore, the timing impact can be obtained accurately. For the sub-net including $n_0$, if (13), (14) and (15) are negative when disconnecting $E_{new1}$ and connecting $E_{new1'}$, $E_{new1}$ is removed. Then the sub-net reconnects to *buff* through $E_{new1'}$. Then the sub-net reconnects to *buff* through $E_{new1'}$. For the sub-net excluding n0, if (20) and (21) are negative when disconnecting $E_{dist'}$ and connecting $E_{new2'}$, $E_{dist'}$ is disconnected and $E_{new2'}$ is connected to $n_d$.

## 4.3 Buffering Pair Score Computation

Different buffering pair results in different timing impacts on all pins of one net. Therefore, the buffering pair score computation estimates the timing impacts of all buffering pairs based on the two derived effects. To accurately assign the scores of buffering pairs, *pseudo edge breaking and buffer connection* and *pseudo buffer reconnection* are applied by estimating the wire length of interconnections by Manhattan distance instead of detailed routed wire length. With a buffering pair, the delays of all pins after buffer insertion can be improved when (7) and (8) are satisfied.

To differentiate buffering pairs, the delay differences of the timing violation sinks *VP* are only concerned. $\Delta T1_{VP}$ = {$\Delta t_{ED1}(vp_1)$, $\Delta t_{ED1}(vp_2)$, …, $\Delta t_{ED1}(vp_k)$}is the delay differences of *VP* after pseudo edge breaking and buffer connection while $\Delta T2_{VP}$ = {$\Delta t_{ED2}(vp_1)$, $\Delta t_{ED2}(vp_2)$, …, $\Delta t_{ED2}(vp_k)$}is the delay differences of *VP* after pseudo buffer reconnection. The slack of each pin indicates its importance of timing optimization. A pin with more negative slack represents that it violates the timing constraint more seriously. Each violation sink is assigned a weight, denoted as $w(vp_i)$, in the following according to its original slack.

$$w(vp_i) = \frac{slack(vp_i)}{\sum_{j=1}^{k} slack(vp_j)} \qquad (22)$$

(22) represents that the violation sink with a more negative slack has a greater weight. Therefore, the score of each buffering pair is computed in the following.

$$\text{score}(pair_{buf}(E_{dis}, B_{ins})) = -\sum_{i=1}^{k} w(vpi) \times (\Delta t_{ED_1}(vp_i) + \Delta t_{ED_2}(vp_i)) \qquad (23)$$

The score of the buffering pair is greater means that the delays of the violation sinks can be improved more by selecting the buffering pair.

# Chapter 5
# Experimental Results

The proposed algorithm is implemented in C++ language on an AMD Opteron 3.0GHz 8-core processor with 48GB memory. Two circuits in IWLS 2005 benchmarks [9] *s35932* and *s38417* with DEF, LEF, and .lib files are used. The original benchmarks contain no spare cells. Therefore, this work inserts 300 spare cells in each circuit, and the ratio of the number of spare cells to the total cells number is between 3% to 5%, which is reasonable for modern designs [5]. SOC Encounter 8.1 runs P&R on the modified circuits. The statistics of the circuits are shown in Table I, where "Circuit Name" denotes the names of circuits; "#Cell" denotes the number of cells in the circuit; "#Buffer" denotes the number of buffers in the circuit; "#Net" denotes the number of nets in the circuit; and "CU" denotes the core utilizations of circuits. To demonstrate the efficiency of the proposed algorithm, this work selects five nets, *N1-N5*, and three nets, *N6-N8*, in *s35942* and *s38417*, respectively. The statistics of the selected nets are shown in Table II, where "Name" gives the name of selected net; "Degree" denotes the number of pins connected by the net; and "Original" denotes the worst negative slack (WNS) and total negative slack (TNS) calculated by the Elmore delay (EL) before ECO timing optimization.

TABLE I
STATISTICS OF THE BENCHMARK CIRCUITS

| Circuit Name | #Cell | #Buffer | #Net | CU (%) |
|---|---|---|---|---|
| *s35932* | 7402 | 300 | 7430 | 80% |
| *s38417* | 8459 | 300 | 8192 | 80% |

Table II shows the experimental results of the proposed topology-aware ECO timing optimization (TOPO). As shown in Table II, edge breaking and buffer

connection (EB) and buffer reconnection (BR) of TOPO averagely improve the WNS by 36.5% and 38.8%, respectively, resulting in 75.3% total WNS improvement. This demonstrates the efficiency of buffering pair score computation of considering both EB and BR. The buffering pair with small delay improvement after EB can be selected because its delay improvement after BR is larger than those of other buffering pairs. Moreover, TOPO improves the TNS by 76.3% on average. Besides adopting the Elmore delay, Table II also shows the WNS improvements computed by SOC Encounter 8.1 (SE). The WNS improvements computed by Elmore delay and SE are similar. The overall runtime, that of detailed routing (DR), and that except for DR are shown in Table II. A grid-based DR is adopted herein. Almost runtime expending on DR demonstrates the efficiency of TOPO.

TABLE II
STATISTICS OF BENCHMARK NETS AND EXPERIMENTAL RESULTS OF TOPOLOGY-AWARE TIMING ECO OPTIMIZATION (TOPO).

| Net | Degree | Original | | TOPO | | | | | Runtime | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | WNS (*us.*) | TNS (*us.*) | WNS imp. by EB (%) | WNS imp. by BR (%) | WNS imp. (%) | TNS imp. (%) | SE WNS imp. (%) | Estimate (*s.*) | DR (*s.*) | Total (*s.*) |
| *N1* | 32 | 40 | 90 | 25.3 | 71.2 | 96.5 | 87.3 | 100 | 0.1 | 62.95 | 63.05 |
| *N2* | 128 | 150 | 370 | 4.1 | 42.9 | 47.0 | 52.8 | 42.0 | 0.92 | 11.43 | 12.35 |
| *N3* | 260 | 250 | 500 | 28.2 | 71.8 | 100 | 98.6 | 100 | 3.79 | 127.51 | 131.3 |
| *N4* | 154 | 200 | 340 | 64.6 | 0.6 | 65.2 | 79.0 | 63.1 | 1.63 | 127.3 | 128.94 |
| *N5* | 82 | 200 | 300 | 21.2 | 78.8 | 100 | 100 | 84.6 | 0.81 | 14.99 | 15.8 |
| *N6* | 23 | 5 | 7 | 37.4 | 19.4 | 56.8 | 51.4 | 100 | 0.03 | 0.43 | 0.46 |
| *N7* | 17 | 9 | 18 | 100 | 0 | 100 | 83.9 | 71.1 | 0.02 | 7.94 | 7.96 |
| *N8* | 26 | 4 | 7 | 11.3 | 25.3 | 36.6 | 57.1 | 42.8 | 0.04 | 143.41 | 143.45 |
| **Ave.** | | **107.25** | **204** | **36.5** | **38.8** | **75.3** | **76.3** | **75.5** | **0.91** | **61.99** | **62.91** |

Conventional buffer insertion of timing ECO only concerns one critical sink in one multi-pin net. To compare with conventional buffer insertion, a simulated 2-pin net-based buffer insertion algorithm (SIM) is implemented herein. SIM ignores the topology of the net and treats one multiple-pin net as a two-pin net including driver and the critical sink. SIM searches all buffers and inserts the one into the path from the driver to the critical sink which achieve the best delay improvement of the critical sink. The original benchmark nets are modified as containing only one violation sink, which is the WNS sink in the original benchmark, by relaxing the required time of other sinks. The comparisons of delay improvement and runtime between SIM and TOPO are shown in Table III. The delay improvement is computed by considering the

whole multi-pin net. The delay improvements of TOPO are better than those of SIM by 35.2% and 30.1% in term of Elmore delay and SE, respectively. TOPO searches all buffering pairs near the whole net in EB while SIM only searches buffering pairs near the path between the driver to the critical sink. Moreover, SIM does no BR. Therefore, compared to SIM, TOPO significantly improves the delay at the cost of runtime.

TABLE III
COMPARISON OF THE DELAY IMPROVEMENT OF THE CRITICAL SINK BASED ON THE ELMORE DELAY (EL) AND SOC ENCOUNTER (SE) AND RUNTIME ANALYSIS BETWEEN THE SIMULATED 2-PIN NET-BASED BUFFER INSERTION (SIM) AND TOPOLOGY-AWARE ECO TIMING OPTIMIZATION (TOPO).

| Net | Critical Sink Delay | | Delay Improvement | | | | Runtime | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SIM | | TOPO | | SIM | | | TOPO | | |
| | EL (*us.*) | SE (*us.*) | EL (%) | SE (%) | EL (%) | SE (%) | Est. (*s.*) | Maze (*s.*) | Total (*s.*) | Est. (*s.*) | DR (*s.*) | Total (*s.*) |
| $N1_{one}$ | 108.2 | 122.1 | 8.9 | 3.5 | 46.2 | 50.6 | 0.02 | 4.85 | 4.87 | 0.09 | 29.24 | 29.33 |
| $N2_{one}$ | 324.2 | 282.7 | 6.8 | 8.3 | 51.8 | 44.4 | 0.08 | 4.76 | 4.84 | 0.69 | 211.19 | 211.88 |
| $N3_{one}$ | 783.6 | 809.7 | 14.1 | 14.5 | 56.0 | 59.0 | 0.51 | 67.81 | 68.32 | 3.23 | 128.71 | 131.94 |
| $N4_{one}$ | 371.6 | 351.1 | 13.5 | 26.9 | 43.8 | 35.1 | 0.19 | 5.87 | 6.06 | 1.17 | 261.34 | 262.51 |
| $N5_{one}$ | 290.7 | 251.7 | 22.1 | 19.1 | 69.6 | 53.0 | 0.28 | 2.87 | 3.15 | 0.8 | 13.02 | 13.82 |
| $N6_{one}$ | 30.8 | 23.4 | 2.3 | 9.7 | 9.2 | 21.4 | 0.02 | 1.95 | 1.97 | 0.03 | 0.43 | 0.46 |
| $N7_{one}$ | 13.4 | 8.9 | 0 | 0 | 67.3 | 71.9 | 0.01 | 0 | 0.01 | 0.02 | 1.95 | 1.97 |
| $N8_{one}$ | 28.9 | 19.4 | 1.6 | 4.1 | 7.4 | 7.9 | 0.02 | 0.28 | 0.3 | 0.03 | 0.65 | 0.68 |
| **Ave.** | | | **8.7** | **10.8** | **43.9** | **42.9** | **0.14** | **11.05** | **11.19** | **0.76** | **80.82** | **81.58** |

# Chapter 6
# Conclusions

This work derives two topology effects on buffer insertion: *edge breaking and buffer insertion* and *buffer reconnection*. Based on the topology effects, a novel topology-aware ECO timing optimization algorithm is proposed. Experimental results show that the proposed algorithm averagely improves the worst negative slack (WNS) and total negative slack (TNS) effectively. Compared to the conventional 2-pin net-based buffer insertion, the proposed algorithm achieves better delay improvement at the cost of runtime.

# Chapter 7
# Bibliography

[1] K.-H. Chang, I. L. Markov, and V. Bertacco, “Reap what you sow: Spare cells for post-silicon metal fix,” *ISPD*, pp. 103–110, 2008.

[2] Z-W. Jiang, M.-K. Hsu, Y.-W. Chang, and K.-Y. Chao, “Spare-Cell-Aware Multilevel Analytical Placement,” *DAC*, pp. 430–435, 2009.

[3] L.-C. Chou and Y.-L. Li, “A Timing-Driven ECO Router with Elmore Delay Model,” *A Thesis Submitted to Institute of Computer Science and Engineering College of Computer Science National Chiao Tung University*, 2008.

[4] Y.-P. Chen, J.-W. Fang and Y.-W. Chang, “ECO Timing Optimization Using Spare Cells,” *ICCAD*, pp. 530-535, 2007.

[5] C. -W. Chang, C.-T. Chao, C. -P. Lu, C. -H. Lo, “A Metal-Only-ECO Solver for Input-Slew and Output-Loading Violations,” *ISPD*, pp. 191-198, 2009.

[6] L.-G. Chang, H.-B. Hung, H.-Y. Chang and I. H.-R. Jiang, “Matching-Based Minimum-Cost Spare Cell Selection for Design Changes,” *DAC*, pp. 408–411, 2009.

[7] Y.-S. Tseng, Y.-K. Yeh, M.-C. Chi and T.-M. Hsieh, “Technology Remapping for Engineering Change with Wirelength Consideration,” *ISCAS*, pp. 2602–2605, 2010.

[8] J. Rubinstein, P. Penfield, and M. Horowitz, “Signal Delay in RC Tree Networks,” *IEEE Trans*. *Computer-Aided Design*, Vol. 2, pp. 202-211, July 1983.

[9] http://www.iwls.org/iwls2005/benchmarks.html