LAYOUT OPTIMIZATION IN ULTRA DEEP SUBMICRON VLSI DESIGN

A Dissertation

by

DI WU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2006

Major Subject: Computer Science

LAYOUT OPTIMIZATION IN ULTRA DEEP SUBMICRON VLSI DESIGN

A Dissertation

by

DI WU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Co-Chairs of Committee,	Rabi N. Mahapatra
	Jiang Hu
Committee Members,	Duncan M. Walker
	Jianer Chen
Head of Department,	Valerie E. Taylor

May 2006

Major Subject: Computer Science

ABSTRACT

Layout Optimization in Ultra Deep Submicron VLSI Design. (May 2006)

Di Wu, B.E., Beijing University of Posts and Telecommunications;

M.S., East Carolina University

Co–Chairs of Advisory Committee: Dr. Rabi N. Mahapatra
Dr. Jiang Hu

As fabrication technology keeps advancing, many deep submicron (DSM) effects have become increasingly evident and can no longer be ignored in Very Large Scale Integration (VLSI) design. In this dissertation, we study several deep submicron problems (eg. coupling capacitance, antenna effect and delay variation) and propose optimization techniques to mitigate these DSM effects in the place-and-route stage of VLSI physical design.

The place-and-route stage of physical design can be further divided into several steps: (1) Placement, (2) Global routing, (3) Layer assignment, (4) Track assignment, and (5) Detailed routing. Among them, layer/track assignment assigns major trunks of wire segments to specific layers/tracks in order to guide the underlying detailed router. In this dissertation, we have proposed techniques to handle coupling capacitance at the layer/track assignment stage, antenna effect at the layer assignment, and delay variation at the ECO (Engineering Change Order) placement stage, respectively. More specifically, at layer assignment, we have proposed an improved probabilistic model to quickly estimate the amount of coupling capacitance for timing optimization. Antenna effects are also handled at layer assignment through a linear-time tree partitioning algorithm. At the track assignment stage, timing is further optimized using a graph based technique. In addition, we have proposed a novel gate splitting methodology to reduce delay variation in the ECO placement considering spatial correlations. Experimental results on benchmark circuits showed the effectiveness of our approaches.

To my parents

ACKNOWLEDGMENTS

First, I would like to thank my advisors Professor Rabi Mahapatra and Professor Jiang Hu for their continuous support, guidance and encouragement. They have led me into this exciting field of VLSI physical design and their insights on research directions and active mentoring have made this dissertation possible. I believe that what I have learned in my research will definitely benefit my career upon graduation.

I am also grateful to members of my advisory committee, Dr. Hank Walker and Dr. Jianer Chen for their valuable comments and precious time. My deep gratitude also goes to Dr. Ricardo Gutierrez-Osuna for helping with my dissertation proposal.

I would like to thank my colleagues and co-authors for sharing research ideas and for helping me on various research projects. Cliff Sze has helped me wrote a dual-driver timing analyzer and taught me how to use the C-tree program in the latest research project. Lu Xiang has provided me his ISCAS85 testcases and standard cell libraries and taught me how to extract timing information using HSPICE. Qiuyang Li helped me write a gate-sizing program for the lastest project. I would also like to thank Ke Cao, Ganesh Venkataraman and Dr Min Zhao for their help and discussions. It is my pleasure to work with them.

Finally, I dedicate this dissertation to my parents for their long-term encouragement, support and patience. They have given me so much inspiration and love throughout my graduate study at Texas A&M.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

A. Coupling Capacitance Mitigation and Timing Optimization at Layer and Track Assignment

Coupling capacitance (crosstalk, capacitive coupling or cross coupling) has become one of the most vital problems in DSM physical design because of (1) interconnect dominated circuit delay and (2) strong coupling effects between interconnect wires. According to ITRS roadmap [1], coupling capacitance starts to surpass wire self capacitance (including substrate capacitance and fringing capacitance) at $0.18\mu m$ technology.

Coupling capacitance can induce two unfavored problems: (1) glitches, which introduce unnecessary signal switching and power consumption. Glitches can also cause circuit malfunction, particularly for dynamic domino circuits; (2) delays, caused by the extra capacitive load, especially when two neighboring signal nets make transitions at the same time but at different direction. In this dissertation, we propose techniques to mitigate the crosstalk-induced-delays. These techniques can be used for high-performance microprocessor and ASIC design.

Most of the previous crosstalk-related research targets at the detailed routing stage which is, however, limited by its routing flexibility. Meanwhile, crosstalk avoidance at the global routing still remains as a challenging problem [2] since the capacitive coupling relies on the neighboring wires, which are difficult to determine at an early routing stage. Between the global routing and detailed routing is the layer and track assignment - an appealing stage to tackle crosstalk problem as neighboring information between long segments of wires can be decided at this stage. There are a number of existing works [3, 4]

The journal model is *IEEE Transactions on Computer-Aided Design.*

on layer/track assignment, however, they are mainly routability-driven and cannot directly address the delay problem caused by coupling capacitance.

Our strategy to handle crosstalk and its induced delay can be described as two steps: (1) At layer assignment, we propose a probabilistic model to quickly calculate the estimated capacitance coupling given number of tracks and segments in a routing region. The estimated coupling is then utilized to calculate delay cost. Layer assignment is performed such that minimum timing slack among all nets is maximized. (Antenna effect, which will be discussed later, is handled simultaneously at this stage). The entire problem can be formulated as a Mixed Integer Linear Programming (MILP) problem and we provide fast heuristic to solve it effectively. (2) Track assignment assigns wire segments to routing tracks once the layer assignment is done. Similar to the layer assignment, the objective of the track assignment is to maximize the minimum slack considering both coupling capacitance and wire detours. This difficult track-routing task can be converted to a well-defined Sequential Ordering Problem (SOP). Our SOP formulation can handle detour-induced-delay and coupling-induced-delay simultaneously. Empty tracks are utilized automatically in the SOP formulation to separate highly coupled signal nets.

Experiments on benchmark circuits showed the effectiveness of both the probabilistic coupling model and our layer/track routing approach. One of the major observations from the experiments is *coupling-induced-delay* must be optimized directly during layer/track assignment, instead of merely minimizing the total amount of coupling capacitance.

B. Antenna Avoidance at Layer Assignment

Antenna effect occurs during the manufacturing process when conductors are fabricated from the lowest layer to the highest layer. In the manufacturing process, conductors (such as gate poly and metal), which have not been covered by a shielding layer of oxide, act

like antenna that collect charges when exposed directly to the plasma [5]. If the conductors (antenna) are connected only to the transistor gates, the accumulated charge in manufacturing can damage the gate oxide through the Fowler-Nordheim (F-N) tunneling current. On the other hand, if the charges can be released through a low impedance path connected to a diffusion, gate damages can be avoided. The risk of the antenna damage to the gate oxide is proportional to the area and perimeter length of the antenna and inversely proportional to the area and perimeter length of the gate oxide.

Existing methods handling antenna effect include diode insertion [6] and jumper insertion [7]. However, both diode and jumper insertion degrade circuit performance. In particular, diodes introduces capacitive load and consumes routing resources. Jumper insertion adds vias and occupies extra space on the top metal layer.

We propose to solve the antenna effects at layer assignment, which are directly related to each other. A linear time optimal tree partitioning algorithm is adapted to solve the antenna problem during layer assignment. Compare to jumper insertion, our method results in significant via reductions. This linear time algorithm can also be applied to existing jumper insertion methods for better CAD tool performance.

C. Delay Variability Reduction at ECO Placement

The variability of circuit delay due to device and interconnect variations (eg. gate length, oxide thickness, threshold voltage and interconnect width variations) has become a great concern. Process variations can be further classified as inter-die and intra-die variations, where intra-die variation often exhibit spatial correlations - device variations have similar trends if placed in close proximity. A statistical timing analyzer is commonly used to find delay variations.

Our focus is to develop techniques to reduce delay variations. In recent literatures,

a number of approaches have been reported to reduce circuit delay variability through *redundancy*. In [8], cross links are inserted to a clock tree to reduce delay variations (clock skews). The work of [9] proposed a re-synthesis technique to trade extra circuitry for delay variation reduction. The use of redundancy can be also found in other works [10] for delay reduction.

Inspired by all these remarkable works, we have proposed a novel *gate splitting* methodology to reduce delay variation in the ECO placement considering spatial correlations. Our approach is integrated into the flow of an industrial place-and-route tool and tested on the ISCAS85 circuits. The experiemental results confirmed the effectiveness of our approach.

D. Organization of the Dissertation

The rest of the dissertation is organized as follows. In Chapter II, we introduce our work on mitigating the risk of crosstalk at layer assignment. Chapter III presents our work on timing-driven track routing considering coupling capacitance. Chapter IV discusses our layer assignment heuristic considering both antenna effects and coupling capacitance. In Chapter V, we present our gate splitting work for variation tolerance. Chapter VI concludes this dissertation and discusses future work.

CHAPTER II

LAYER ASSIGNMENT FOR CROSSTALK RISK MINIMIZATION

Under modern VLSI technology, crosstalk noise is so severe that effort merely in detaied routing stage is not adequate for solving the problem and it has to be considered in earlier design stages. In this work, we propose two heuristic algorithms for crosstalk mitigation in layer assignment, which is a stage between global routing and detailed routing, so that subsequent crosstalk avoidance in detailed routing can be more attainable. The pre-detailed-routing crosstalk is estimated through a probabilistic model. Constraint on the amount of vias is also considered. Experimental results on benchmark circuits confirm the effectiveness of the proposed heuristics.

A. Introduction

When VLSI technology feature size keeps shrinking, interconnect wire width scales faster than height and consequently wires look progressively thin and tall. Moreover, higher degree of integration makes wires to be placed much closer to each other. This technology trend leads to greater and greater coupling capacitance between neighboring wires. As a result, signal switching at one net may greatly affect its neighboring wires. Such crosstalk noise [11] may cause logical malfunction or at least extra signal propagation delay.

Since 1990s, crosstalk avoidance has been a focal point of research. Early works mostly solve the crosstalk problem through detailed routing [12–16] or wire spacing [17, 18], since crosstalk is directly determined by wire adjacency and spacing. Even though the contributions of these works are indispensable, the freedom and effect of change in detailed routing and wire spacing are limited. When the crosstalk problem is severely strong, a localized optimization is not adequate any more.

A much greater flexibility on crosstalk avoidance can be obtained at the global routing

stage. However, without wire adjacency information, it is very hard to estimate crosstalk with decent accuracy at this level. Perhaps the only crosstalk driven global routing work is reported by Zhou and Wong [2]. In this work, a simplified trial layer/track assignment is employed to estimate crosstalk and guide a Lagrangian relaxation based optimization. Post global routing adjustment techniques are proposed in [19, 20]. In [19], a graph-based technique is presented to check if certain crosstalk tolerance is satisfied. The work of [20] estimated crosstalk based on expected spacing in a gridless routing. As a compromise between diffculty and flexibility, crosstalk issue is considered in crosspoint assignment [21, 22] which is a stage between global routing and detailed routing.

Layer assignment is another stage between global routing and detailed routing which is suitable for solving crosstalk problem. There are greater flexibilities on changing wire route and thereby greater capability on mitigating crosstalk in the layer assignment stage. Furthermore, the problem size of layer assignment is usually smaller than that of global routing so that it is relatively easier to be handled. An optimal minimum crosstalk layer assignment algorithm is provided in [23]. It considered only VHV channel routing and assumed the horizontal tracks had been assigned. A combined crosstalk driven layer/track assignment technique is proposed in [24]. However, this work did not evaluate crosstalk quantitatively and only attempted to enforce the rule that wires of simultaneous switching nets should not be adjacent to each other. Such formulation neglects the difference between a short adjacency and a long adjacency. Moreover, it did not consider the option to break a long wire into segments and assign them to different layers [25].

We propose layer assignment heuristics that can reduce crosstalk *risk* so that crosstalk can be handled more effectively in subsequent detailed routing. The input to the layer assignment algorithms includes a global routing result and crosstalk tolerance for each net. The objective of this work is to maximize the minimum crosstalk slack among all nets. The crosstalk slack of a net is its crosstalk tolerance minus the estimated crosstalk of this net

in the resulting layer assignment solution. The estimated crosstalk is obtained by using a probability based model. We allow a long wire to be segmented in term of global routing cells, and these segments can be assigned to different layers so that more flexibilities can be exploited for crosstalk risk reduction. Since layer switching for wires of a same net implies vias, the constraint on the number of vias is also enforced in the proposed algorithms. The major contributions of this work are:

- This is the first work that optimizes crosstalk probability in layer assignment, to the best of our knowledge. Compared with previous works that optimize crosstalk according to trial track assignment, probability based approach is better at capturing overall crosstalk risk. Further, probability based crosstalk estimation is significantly faster than trial track assignment method.

- A crosstalk bound analysis is performed in this work and the analysis result reveals that many nets can be treated as "don't cares" in the process of layer assignment.

- A *soft net prefixing* technique is introduced to exploit the "don't cares" in the context of considering via constraints. This technique can improve solution quality significantly without increasing computation cost.

The proposed algorithms are tested on benchmark circuits with different levels of crosstalk tolerances and via constraints. The experiments show encouraging results, and particularly the soft net prefixing technique yields about 20% improvement on crosstalk slacks.

B. Preliminaries

In previous works, crosstalk estimation at the global routing level is usually conducted with a simplified trail track/layer assigment, of which the following limitations can be observed: (1) a simplified track/layer assignment may not reflect the actual track/layer assignment

later in the detailed routing; (2) even with a simplified track/layer assignment, it is still too time consuming in the inner loop of the optimization. In addition, the goal of the global level optimization is to reduce the risk of crosstalk by making the crosstalk-driven detailed routing easier, not to completely eliminate the crosstalk. Therefore, in this work, we employ a fast probabilistic model to estimate crosstalk noise based on the grid graph in global routing. Similar model has been reported in [26].

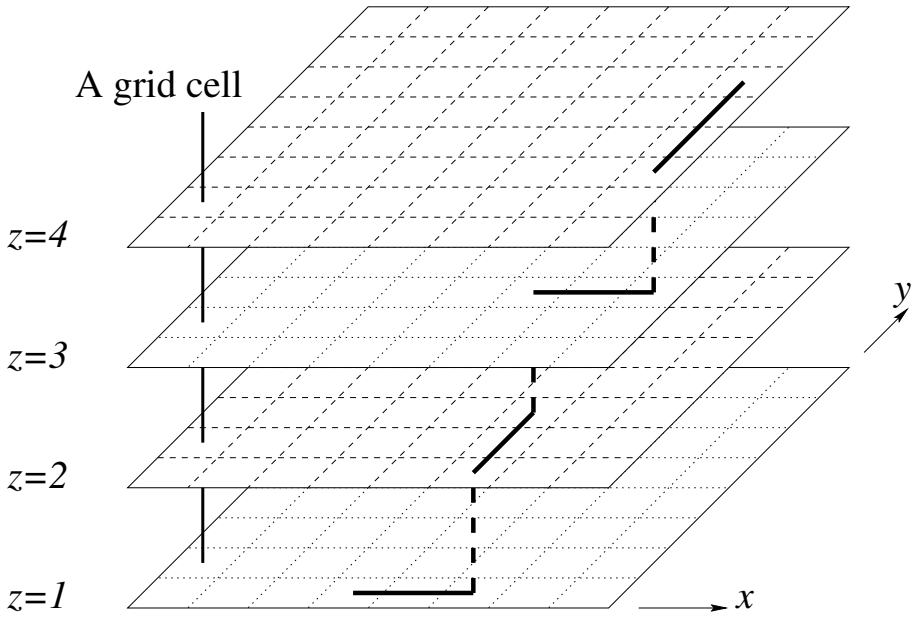


Fig. 1. Example: a net passes through a four-layer routing area.

Since our layer assignment heuristics take a global routing solution as input, we tessalate the whole routing region in a grid graph which is an array of grid cells $\{g_1, g_2, ...\}$. A routing solution for a net is expressed in terms of the grid cells its wiring route passes through, the detailed route within each grid cell will be specified in the detailed routing stage. When we consider multi-layer routing, each grid cell includes several layers. An example of a four layer routing is illustrated in Figure 1 where we use z as layer index. Please

note that each layer has a preferred direction for routing, either horizontally or vertically. A layer switching implies a via. Each grid cell g_j on layer z has a size λ_{jz} which is the cell width(height) if layer z is for horizontal(vertical) wires. A grid cell g_j can be divided into two *sub-grid-cells*. Each sub-grid-cell consists of layers with the same routing direction, i.e., $g_j, z = 1, 3, \ldots$ and $g_j, z = 2, 4, \ldots$.

Crosstalk or coupling capacitance between two nets is generally proportional to their adjacency length and inversely proportional to their spacing. We consider a gridded routing design where the routing track pitch is fixed. If there is at least one empty track between two wires, the crosstalk between them can be neglected for two reasons: (1) the spacing between them is relatively large and (2) a shield may be inserted in the empty track between them. If two wires occupy two adjacent tracks, then the crosstalk can be evaluated according to their adjacency length.

For a grid cell g_j on a routing layer z, the number of routing tracks and wires passing through are denoted as K_{jz} and u_{jz}, respectively. If we assume every wire segment has equal chance to occupy any of the routing track, then the probability that a wire has one other wire in its neighboring track is

$$P_{1,jz} = \frac{2(K_{jz} - u_{jz} + 1)(u_{jz} - 1)}{K_{jz}(K_{jz} - 1)} \tag{2.1}$$

then the probability that a wire has two other wires in its neighboring track is

$$P_{2,jz} = \frac{(u_{jz} - 1)(u_{jz} - 2)}{K_{jz}(K_{jz} - 1)} \tag{2.2}$$

Similar conclusion is reached in [26], thus we skip the derivation here. The expected crosstalk is

$$\chi_{jz} = (P_{1,jz} + 2P_{2,jz})\lambda_{jz} = 2\lambda_{jz}(u_{jz} - 1)/K_{jz} \tag{2.3}$$

Under this probabilistic model, we assume that a horizontal/vertical wire segment always

occupies an entire track of a grid cell in full length. If a horizontal/vertical wire segment is shorter than the grid cell width/height, the above crosstalk estimation introduces pessimism. However, this pessimism compensates well for the optimism due to the neglection of jogs that may be brought to a straight wire segment in detailed routing. Even though we consider gridded routing, our work can be extended to gridless routing easily.

C. Problem Formulation

For each net N_i, we assume there is a crosstalk tolerance τ_i given. Normally, a timing critical net has a low crosstalk tolerance and a non-critical net has a high tolerance. A crosstalk slack is $\phi_i = \tau_i - \chi_i$ where $\chi_i = \sum_{j:N_i \in g_j} \sum_{z:N_i \in z} \chi_{jz}$ is the total crosstalk for net N_i. For a long wire spanning multiple grid cells, we allow it to be broken into several segments in terms of grid cells. Each segment may be assigned to a different layer. If two segments in two neighboring cells are assigned to different layers, a via is incurred. The height of the via depends on the number of layer switching, i.e., if a net switches from layer z_1 to z_2, then the *via height* is proportional to $|z_1 - z_2|$. Sometimes we need such layer switching to reduce crosstalk risk, but we need to restrain such layer switching because of the vias. We define *total via height* via_{jk} as the summation of via heights of all nets between two neighboring grid cells g_j and g_k. via_{jk} is bounded by a user defined total via height constraint ψ_{jk}.

Besides their wire adjacency length, the crosstalk between two nets also depends on their switching activities. Two nets with simultaneous opposite signal switching need to be separated. A timing critical net should be placed away from a net that switches frequently. These switching activity related issues are not directly included in the formulation here for two reasons: (1) considering switching activities requires a great number of variables to specify constraint between every pair of nets in the same grid cell; (2) the primary goal in

layer assignment is to reduce the crosstalk risk, not to solve everything completely. Timing critical net will be placed into less congested layer so that there will be greater chance it is separated by a shield. The problem formulation is described as follows.

Layer Assignment for Minimum Crosstalk Risk(LA-MinCR)*: Given a grid graph composed by a set of grid cells* $\{g_1, g_2, ...\}$*, number of routing tracks* K_{jz} *for each grid cell* g_j *on layer* z*, total via height* ψ_{jk} *between two neighboring grid cells* g_j *and* g_k*, a set of nets* $\{N_1, N_2, ...\}$ *routed in terms of grid cells, and crosstalk tolerance* τ_i *for each net* N_i*, assign each net in each grid cell to a specific layer such that the minimum expected crosstalk slack among all nets is maximized while the number of wires in each grid cell on each layer does not exceed its corresponding number of tracks and total via heights between two grid cells is no greater than the given bound.*

The crosstalk estimation in this stage is based on probability estimation and the actual crosstalk after detailed routing may deviate from this estimation. Thus, we attempt to maximize the slack instead of just satisfying the tolerance so that there could be a maximum safety margin to cushion any deviations from the detailed routing solutions. Throughout this chapter, we use the term min_slack to represent the minimal slack among all nets in a circuit and our goal is to maximize min_slack.

If we let x_{ijz} be a decision variable to tell if net N_i in grid cell g_j is assigned to layer z, then the complete LA-MinCR problem can be formulated as a mixed 0-1 and non-linear programming problem as follows.

Maximize: ϕ

Subject to:

$$\phi + \sum_{j:N_i \in g_i} \sum_{\forall z} \frac{2\lambda_{jz}(\sum_{l:N_l \in g_j} x_{ljz} - 1)}{K_{jz}} x_{ijz} \leq \tau_i, \forall N_i \tag{2.4}$$

$$\sum_{i:N_i \in g_j} x_{ijz} \leq K_{jz}, \forall g_{j,z} \tag{2.5}$$

$$\sum_{i:N_i \in g_j, N_i \in g_k} |\sum_{\forall z} z x_{ijz} - \sum_{\forall z} z x_{ikz}| \leq \psi_{jk}$$
$$j,k : g_j \text{ adjacent to } g_k \tag{2.6}$$

$$\sum_{\forall z} x_{ijz} = 1, \forall N_i; g_j : N_i \in g_j \tag{2.7}$$

$$x_{ijz} \in \{0,1\}, \forall N_i, g_j, z \tag{2.8}$$

the sum of total estimated crosstalk and the slack variable ϕ should never exceed the crosstalk tolerance for each net. The next inequality (2.5) states that the number of wires on each layer of each grid cell is no greater than the number of tracks or wiring capacity. In constraint of (2.6), $\sum_{\forall z} z x_{ijz}$ tells the layer index number of net N_i in grid cell g_j in terms of decision variables. Inequality (2.7) is the exclusivity constraint ensuring that a wire is assigned only to one layer. The last constraint implies that this is an integer programming problem which is generally NP-hard. The computation cost for directly solving this non-linear integer programming problem is prohibitive since it would require tremendous CPU time and memory.

In this chapter, we propose two efficient heuristic algorithms to solve the LA-minCR problem. The first approach (LA-MinCR-$Solve$) is a greedy heuristic, i.e., layer assignment is performed on each of the sub-grid-cells one after another. Once a sub-grid-cell is processed, its layer asssignment is fixed. The second heuristic (LA-MinCR-$Solve^+$) extends LA-MinCR-$Solve$ by using a multi-candidate approach to better exploit the solution space. At the same time, a simple yet effective pruning technique is applied to make the multi-candidate approach more efficient. In this work, our heuristic algorithms are focused on a four-layer model, but they can be extended to handle models with more than four

layers.

D. Algorithm

In both of our LA-MinCR-$Solve$ and LA-MinCR-$Solve^{+}$ heuristics, we perform layer assignment on a panel-by-panel and cell-by-cell basis. We define a $panel$ as an entire row/column on the routing area. A row panel consists of horizontal layers and a column panel consists of vertical layers. Within a panel, sub-grid-cells are processed cell by cell. All sub-grid-cells in an entire panel need to be processed before we proceed to the next panel following the order of the panel/cell criticality we defined as follows.

Definition 1: *The criticality α_{g_i} of a sub-grid-cell g_i is defined as the number of horizontal/vertical wire segments in g_i.*

Definition 2: *The criticality α_{s_i} of a horizontal/vertical panel s_i is equal to the value of the maximal α_{g_i} among all sub-grid-cells $\{g_1,g_2,..,g_i,...\}$ in panel s_i.*

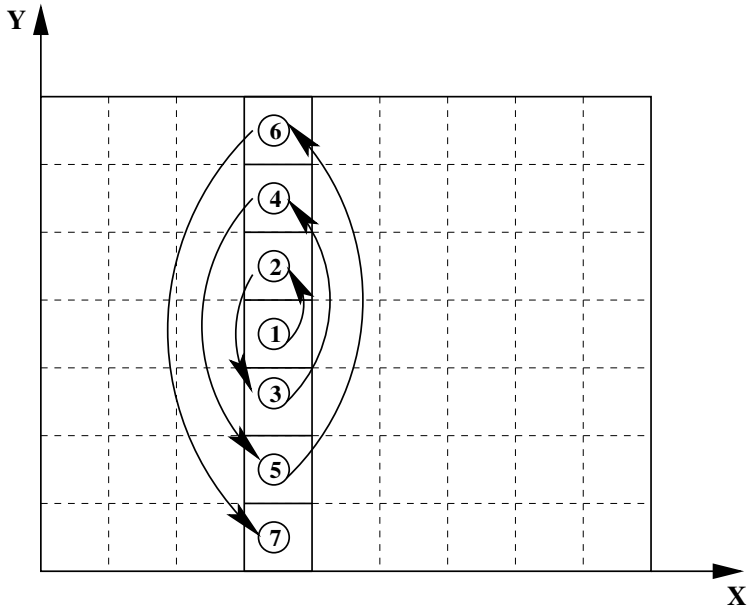


Fig. 2. A top view of the cell processing order of a panel.

For each panel, we first process the most critical sub-grid-cell in it. Then, we use this particular sub-grid-cell as a starting point and continue to process the remaining cells in an alternate direction towards both ends of the panel. For instance, in Figure 2, cell 1 is the most critical sub-grid-cell in this panel and the cell processing order is represented by the increasing number on each of the sub-grid-cells in this panel. This continuous cell-by-cell processing order prevents via height constraint deadlocks and facilitates our multi-candidate approach as described in section 3. Using congestion condition as the processing order alllows us to reach as many nets as possible in the early stage to reduce the chance of being trapped in local optima.

1. Greedy Heuristic (LA-MinCR-$Solve$)

Before we proceed to perform layer assignment on a single sub-grid-cell g_j, we need to the know the following information about g_j : track capacity K at each layer, nets in g_j and their current crosstalk slack ϕ_i. It is important to note that we use $\phi_i, \forall N_i$ here to represent a gradually reduced slack value during the progress of the layer assignment.

In order to maximize the minimum crosstalk slack for a sub-grid-cell g_j, nets with greater crosstalk slack need to be placed on a more congested layer and nets with less crosstalk slack can be placed on a less congested layer. For each sub-grid-cell g_j, we sort the nets into a sequence $\{N^1, N^2, ..., N^q\}$ in non-increasing order of their crosstalk slack. Then the layer assignment for this sub-grid-cell becomes to find an index p such that this net sequence is partitioned into two subsequence $S^1_{g_j} = \{N^1, N^2, ..., N^p\}$ and $S^2_{g_j} = \{N^{p+1}, N^{p+2}, ..., N^q\}$ with each subsequence corresponding to a layer. We select an index p such that the minimal slack among all nets in g_j is maximized. Furthermore, p must be selected in the range of $[max(1, q-K), min(K, q)](q \leq 2K)$. Otherwise, track capacity constraint K will be violated. After such a partitioner p is selected, we need to verify if this net partitioning solution satisfies the via height constraint between g_j and its

neighbore g_k, where a layer assignment is fixed previously. We represent the fixed layer assignment for g_k in a similar way by using subsequences $S^1_{g_k}$ and $S^2_{g_k}$. A via occurs only if a net passing through both g_j and g_k is placed on different layers. Given $S^1_{g_j}$ and $S^2_{g_j}$ obtained from the initial net partitioning, we first check if the current total via height via_{jk} between g_j and g_k is greater than the user defined total via height constraint ψ_{jk}. If yes, we feed $S^1_{g_j}$ and $S^2_{g_j}$ to the via enforcement procedure (as shown in Algorithm 1). Otherwise, we skip this procedure.

Algorithm 1: Via constraint enforcement

while $via_{jk} > \psi_{jk}$ **do**
  choose a net $N_i \in S^1_{g_j}$ and $S^2_{g_k}$, if such net doesn't exist, $N_i = null$;
  choose a net $N_j \in S^2_{g_j}$ and $S^1_{g_k}$, if such net doesn't exist, $N_j = null$;
  if ($N_i \neq null$ and $|S^2_{g_j}| + 1 \leq$ track capacity K) OR ($N_j \neq null$ and $|S^1_{g_j}| + 1 \leq$ track capacity K) **then**
    $S^2_{g_j} \leftarrow N_i$ or $S^1_{g_j} \leftarrow N_i$ depends on feasibility and which switching produces greater minimal slack;
    $via_{jk} \leftarrow via_{jk} - 2$;
  else
    (only swapping nets between $S^1_{g_j}$ and $S^2_{g_j}$ simultaneously can reduce vias)
    if ($N_i \neq null$ AND $N_j \neq null$) **then**
      $S^2_{g_j} \leftarrow N_i$ and $S^1_{g_j} \leftarrow N_j$;
      $via_{jk} \leftarrow via_{jk} - 4$;
    else if ($N_i \neq null$ AND $N_j = null$) **then**
      find a net $N_l \in S^2_{g_j}$ and $N_l \notin g_k$;
      $S^2_{g_j} \leftarrow N_i$ and $S^1_{g_j} \leftarrow N_l$;
      $via_{jk} \leftarrow via_{jk} - 2$;
    else if ($N_i = null$ AND $N_j \neq null$) **then**
      find a net $N_l \in S^1_{g_j}$ and $N_l \notin g_k$;
      $S^1_{g_j} \leftarrow N_j$ and $S^2_{g_j} \leftarrow N_l$;
      $via_{jk} \leftarrow via_{jk} - 2$;
    end if
  end if
end while

In Algorithm 1, whenever we have multiple candidate nets to choose from for a layer switching, we select one that can produce greater minimum slack. The via enforcement procedure ends whenever the via height constraint is satisfied. Then we fix the layer as-

signment for g_i and update the crosstalk slack ϕ_i for each net N_i in g_j. The same net partitioning and via enforcment procedure is repetitively utilized for layer assignment on the remaining sub-grid-cells. The via enforcement procedure only takes $O(K)$ time. The net partitioning at each sub-grid-cell takes $O(KlogK)$ time because its time complexity is bounded by the sorting procedure. The overall time complexity of LA-MinCR-$Solve$ is bounded by $O(GKlogK)$, where G is the total number of sub-grid-cells.

2. Crosstalk Bound Analysis and Net Pre-fixing

Since large number of nets are involved in a circuit, we would like to see if certain wire segments can be pre-fixed without affecting or even producing better min_slack solution. By carefully examing the problem, we have the following observations.

For an individual net N_i, its maximum and minimum estimated crosstalk can be represented by χ_i^- and the χ_i^+, respectively. If we define $G_i = \{g_{i1}, g_{i2}, ..., g_{ij}, ...\}$ as the set of sub-grid-cells on the route of net N_i, χ_i^- can be obtained by summing up the worse-case crosstalk that net N_i can experience at each g_{ij}. This is done by moving as many wires as possible to the same layer where N_i is placed under the limit of the track constraint. Similarly, χ_i^+ is obtained by summing up the best-case crosstalk that net N_i can experience at each g_{ij} and this is done by moving away as many wires as possbile from the layer where N_i is placed. Therefore, we can see that ϕ_i is bounded by [Γ_i^-,Γ_i^+], where $\Gamma_i^+ = \tau_i - \chi_i^+$ and $\Gamma_i^- = \tau_i - \chi_i^-$ Please note that we always put the most congested layer close to the bottom. For horizontal wires, the most congested layer is layer 1. For veritcal wires, it is layer 2.

Now let us look at a simple example, as shown in Figure 3. A net N_i with $\tau_i = 400$ routes through a 2×2 routing areas with each grid cell has 4 layers. Each of the grid cell has a size of 100×100 and each layer includes 3 tracks. We first calculate Γ_i^+ by placing all wire segments of N_i to the higher layer for each grid cell it passes through and push

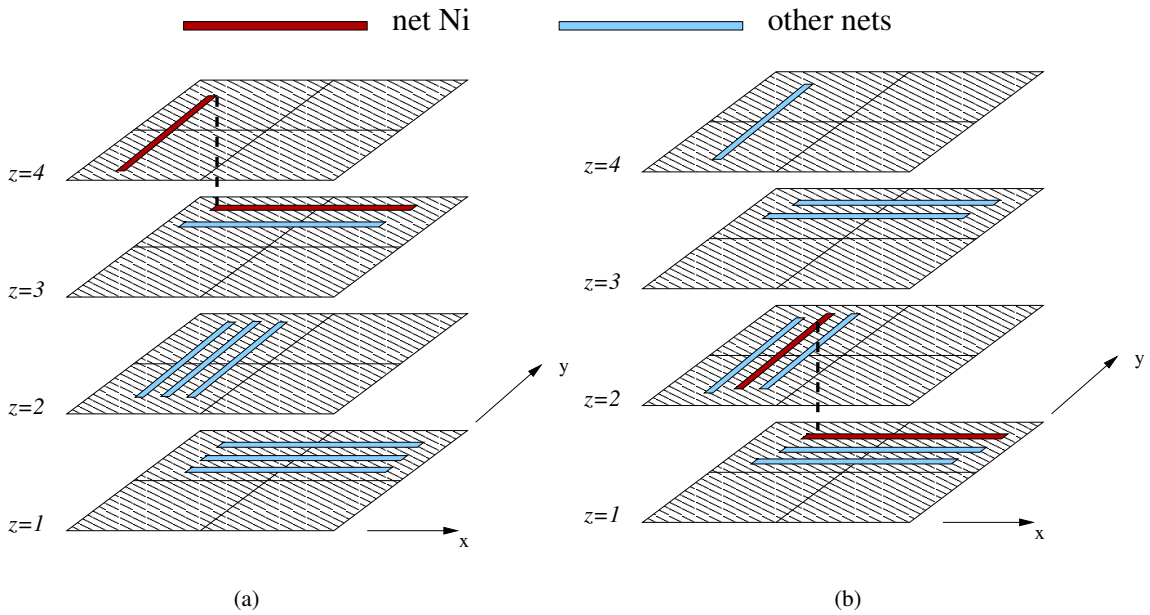


Fig. 3. (a) net N_i experiences minimum crosstalk noise. (b) net N_i experiences maximum crosstalk noise.

as many other wires as possible to lower layer, as illustrated in Figure 3(a). Based on our probability model (Equation 2.3), the estimated crosstalk noise χ_i^+ for N_i is $100 \times 2(1-1)/3 + 100 \times 2(1-1)/3 + 100 \times 2(2-1)/3 + 100 \times 2(2-1)/3 = 132$. Therefore, $\Gamma_i^+ = 400 - 132 = 268$. Similarly, the estimated crosstalk noise χ_i^- for net N_i in Figure 3(b) is $100 \times 2(3-1)/3 + 100 \times 2(3-1)/3 + 100 \times 2(3-1)/3 + 100 \times 2(3-1)/3 = 533$ and $\Gamma_i^- = 400 - 533 = -133$. As a result, ϕ_i will always be in $[-133, 268]$ under any layer assignment solutions.

If we denote Γ_{min}^+ as the minimal of $\Gamma_i^+, \forall N_i$. We can reach following conclusion:

Lemma 1: *Maximal* min_slack *is no greater than* Γ_{min}^+ *and a net* N_i *can be considered as a non-critical net if* $\Gamma_i^- \geq \Gamma_{min}^+$.

It is important to note that when calculating Γ_i^- and Γ_i^+, we focus our attention to an individual net N_i while treating other nets only as track occupiers. Therefore, Γ_{min}^+ only provides a upper bound for min_slack. We can claim that an optimal solution is achieved

if the maximal min_slack we find is equal to Γ^{+}_{min}. But it is not true vice versa.

It is our intention to pre-fix all non-critical nets to the lower layer so that crosstalk noise can be mitigated at the higher layer for critical nets. However, non-critical nets can account for a large portion of total nets in a circuit (As shown in TABLE VII), hence not all of them can be pre-fixed to the lower layer because of the track capacity constraint at each sub-grid-cell. Also, directly(hard) pre-fixing non-critical nets to certain layers can cause via constraint deadlocks if we use a cell-by-cell based layer assignment. To overcome these difficulties, we propose a technique called $soft$ pre-fixing by taking advantage of our net partitioning technique. The basic idea of the soft pre-fixing is to increase the initial crosstalk slack τ_i for non-critical nets to $\tau_i = \Phi + \Gamma^{-}_{i}$, where Φ is an extremely large value. With our net partitioning technique, this modification allows a non-critical net to be assigned towards the lower layer at each sub-grid-cell it passes through. We call this method a $soft$ net pre-fixing since we don't directly provide which layer a non-critical net is assigned to. Adding Γ^{-}_{i} as an offset to Φ provides two advantages: (1) it increases the minimal slack among non-critical nets. (2) it allows a non-critical net to keep the same relative ordering among other non-critical nets so that they have the tendency to remain in the same layer at neighboring sub-grid-cells. As a result, layer switchings between neighboring sub-grid-cells for non-critical nets are reduced and thereby vias can be utilized by critical nets to achieve greater min_slack.

Soft pre-fixing plays an important role when used in conjunction with our net paritioning technique. It incorporates global behavior of nets in terms of their crosstalk slack into the localized optimization, as a result, soft pre-fixing is guaranteed to achieve better global solution.

3. A Multi-candidate Approach (LA-MinCR-$Solve^{+}$)

To furthur reduce the chance of local optima, we extend our LA-MinCR-$Solve$ by perturbating extra candidate solutions at each sub-grid-cell. We call it a multi-candidate approach. Together with the original LA-MinCR-$Solve$ solution, the perturbated solutions are kept for each sub-grid-cell during the layer assignment process. This additional perturbation makes the solution space better exploited, in contrary to *LA-MinCR-Solve* where only a single solution is retained for each sub-grid-cell. A *solution tree* is generated and expanded during the progress of this muti-candidate approach, with the first sub-grid-cell being processed as its root. A $thread$ on the *solution tree* corresponds to a complete layer assignment solution for all sub-grid-cells having been processed so far. Upon completion of tree construction (that's when all sub-grid-cells are processed), the best thread with the maximal min_slack is picked and a backward traversal is utilized to perform the actual layer assignment.

Figure 4 shows an example of the multi-candidate approach. Each tuple in the graph corresponds to a layer assignment solution for a sub-grid-cell with each letter representing a net. The first subsequence in the tuple represents nets that are assigned to the lower layer and the second subsequence represents nets that are assigned to the higher layer. Arrows in the graph indicate the growing direction of the solution tree. At each sub-grid-cell, the perturbation process is accomplished by swapping nets between different layers based on the initial net partitioning result. Each of the perturbated tuples should perform no worse than the original tuple with respect to min_slack, and they are also subject to the via height constraint enforcement as described in Algorithm 1. At each sub-grid-cell, we allow a maximum of v candidate solutions.

Since explicit multi-candidate perturbation is involved, a pruning technique is needed to remove non-promising threads during the perturbation process to avoid number of tuples

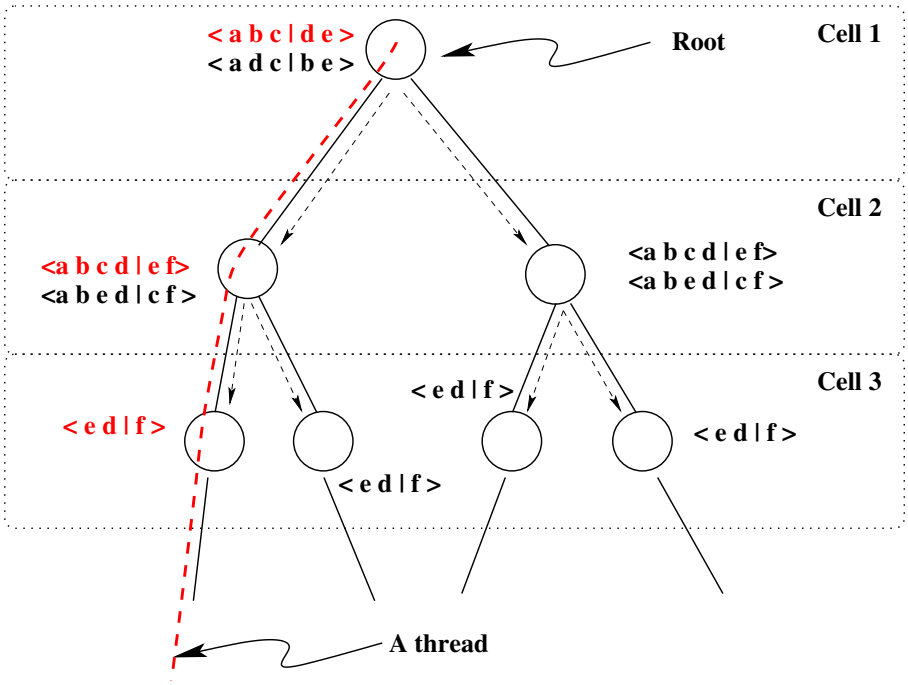


Fig. 4. Multi-candidate approach.

being exponentially increased. We define a upper limit γ as the maximal number of total threads we can retain during the expansion of the solution tree. If more than γ threads are encoutered, we must eliminate the exceeding threads by using the following pruning procedure.

1: sort the tuples with non-decreasing order according to its $min(\phi_i), \forall N_i$

2: remove tuples with lower crosstalk slack ϕ_i until the maximum thread number γ is satisfied.

3: if there are ties among tuples with the same $min(\phi_i), \forall N_i$, remove those that have lower $secondary\ min(\phi_i), \forall N_i$.

The perturbation process at each sub-grid-cell takes $O(\upsilon)$ time. The overall time complexity of the LA-MinCR-$Solve^+$ is bounded by $O(G\upsilon(KlogK)(\gamma log\gamma))$.

E. Experimental Results

We have implemented both LA-MinCR-$Solve$ and LA-MinCR-$Solve^+$ in GNU C on a Linux platform with a 2.4GHz Pentinum IV processor and 640MB memory. We use the following benchmark circuits to test our heurisitcs: $apte$, $a9c3$, $ac3$, $playout$ and $xc5$. Both benchmarks and their global routing solutions are obtained from authors of [27]. Crosstalk tolerances are generated randomly but we have verified our heuristics with different levels of crosstalk tolerances. Table I lists the specifications, slack upper bound Γ^+_{min} and the percentage of non-critical nets for each benchmark circuit based on Lemma 1.

Table I. Circuit specification.

circuit name	no. of grid cells	no. of nets	no. of non-critical net	non-critical net percentage	Γ^+_{min}
a9c3	35×33	1148	955	83%	508
ac3	27×28	200	65	33%	5905
apte	23×22	77	35	45%	5801
playout	42×36	1294	832	64%	777
xc5	42×39	975	674	69%	2611

Since no previous work is close to our formulation, we tested our heuristics with the following three different approaches, as shown in Table II. (1) LA-MinCR-$Solve$ with no soft pre-fixing. (2) LA-MinCR-$Solve$ with soft pre-fixing. (3) LA-MinCR-$Solve^+$ with soft pre-fixing and maximum threads $\gamma = 100$ and $\upsilon = 3$. We use a uniform via height constraint for each boundary between two neighboring sub-grid-cells and we vary the total via height constraint by 2,4,6 and infinity (no via constraint). As we expected, when via height constraint is relaxed, min_slack increases with approach (3) outperforming both (1) and (2). On the average, approach (2) increases min_slack by 20% compare to approach (1), and approach (3) increases min_slack by 6% compare to approach (2). Table III lists the CPU time averaged over different via height constraints for each benchmark circuit.

Table II. Experimental result: a comparison of min_slack between three different approaches as via height constraints increase.

		a9c3	ac3	apte	playout	xc5
(1) LA-MinCR-$Solve$ w/o prefix	via ht.=2	-2337	4817	5291	-175	1738
	via ht.=4	-1473	4789	5511	49	2031
	via ht.=6	-1185	4993	5731	385	2423
	via ht.=Inf	-1041	4993	5801	385	2611
(2) LA-MinCR-$Solve$ w/ soft prefix	via ht.=2	-500	4874	5291	329	2074
	via ht.=4	-68	5189	5511	329	2367
	via ht.=6	220	5189	5731	385	2535
	via ht.=Inf	508	5189	5801	777	2611
(3) LA-MinCR-$Solve^+$ w/ soft prefix (max. 100 threads)	via ht.=2	-212	4874	5511	385	2311
	via ht.=4	76	5189	5731	385	2367
	via ht.=6	364	5189	5801	441	2535
	via ht.=Inf	508	5189	5801	777	2611

Table III. Experimental result: CPU time (sec).

circuit name	avg. LA-MinCR-$Solve$	avg. LA-MinCR-$Solve^+$ (max. 100 threads)
a9c3	1.14	53.15
ac3	0.07	7.89
apte	0.02	1.68
playout	1.77	69.00
xc5	1.75	53.19

In particular, CPU time for LA-MinCR-$Solve$ is averaged over both approach (1) and (2) since little timing difference is observed between these two approaches.

F. Conclusion

In this chapter, we have proposed a new approach for crosstalk mitigation at the layer assignment stage between global routing and detailed routing in VLSI physical design. This approach aims to discover and reduce crosstalk risk at the pre-detailed-routing level

and the crosstalk noise is estimated based on a probabilistic model. We formulate our problem as an integer convex programming problem and provide two heurisitcs to solve it efficiently. Experimental results confirmed the effectiveness of our heuristics on benchmark circuits.

CHAPTER III

TIMING-DRIVEN TRACK ROUTING CONSIDERING COUPLING CAPACITANCE

Track routing is a step between layer assignment and detailed routing. In this chapter, we propose a coupling aware timing driven track routing heuristic. Given a global routing solution and timing constraint for each net, major trunks of wire segments are assigned to routing tracks such that the minimum timing slack among all nets is maximized. Delay penalties from both coupling capacitance and wire detour are considered in a unified graph model. The core problem is formulated and solved as a Sequential Ordering Problem (SOP). Routing blockages are handled in a post processing procedure. The experimental results on benchmark circuits show that the effect of coupling capacitance on timing is significant and the proposed heuristic results in greater improvement on coupling aware timing compared with other approaches.

A. Introduction

The sustained VLSI technology scaling leads to two trends: (1) interconnect dominated circuit delay and (2) strong coupling effects between interconnect wires. A tremendous amount of work has been reported on either timing driven interconnect routing [28–32] or coupling noise avoidance [2, 4, 13, 15–20, 22, 24, 33, 34]. However, very few works address the closely related timing and coupling issue at the same time. In most of the timing driven routing works, the coupling capacitance induced delay is neglected. In the coupling noise avoidance works, efforts are made to minimize either total coupling capacitance or violations on coupling constraints, but the impact on timing is not considered.

In ultra-deep submicron technology, the coupling capacitance starts to dominate self capacitance which includes substrate capacitance and fringing capacitance. Therefore, coupling capacitance greatly affects wire delay and can no longer be ignored in a timing driven

routing. Merely minimizing the total coupling capacitance is not adequate either, since the same amount of coupling capacitance may cause a different value of delay depending on its location. Therefore, the impact of coupling capacitance on delay needs to be handled directly in a timing driven routing. Considering coupling capacitance in timing driven routing presents a great challenge, since the delay estimation has to consider wire adjacencies in addition to individual wire routes. The simplest version of routing problem, which handles only routability and only 2-pin nets, is an NP-complete problem. Even when coupling capacitance is neglected and the delay of each net can be estimated independently, optimizing timing in routing is a notoriously difficult problem [30–32]. Including the coupling effect brings complex inter-wire dependencies to the complicated routing procedure.

Because of the high complexity, a routing problem is usually solved in two stages: global routing and detailed routing. In global routing, the entire routing area is tessellated into an array of rectangle global cells and each net is routed in term of the global cells. The detailed route within each cell is determined in subsequent detailed routing. In detailed routing [13, 16], the freedom to make route change is restricted to a small region and therefore the improvement on timing or coupling is limited. In contrast, global routing allows much greater freedom and flexibility on optimizing timing and coupling. However, the absence of wire adjacency information makes the coupling capacitance estimation hard to be obtained. A methodology level approach on coupling aware timing driven global routing is suggested in [35].

Recently, another routing stage - track routing - is proposed [3, 24, 34] to be performed between global routing and detailed routing. Taking a global routing result, a track router assigns major trunks of wire segments to routing tracks in a row(column) of global cells. Please note that the track assignment in a track routing is different from the track assignment in a detailed routing which only handles nets within one global cell. Track routing is an appealing stage for handling the coupling issue, as both the wire adjacency information

and decent freedom of route change are available. In addition, a track routing does not need to deal with complicated design rules as in detailed routing so that it can be focused on the timing and routability issue. In [3], Batterywala *et al.* modeled the track routing as a weighted bipartite matching problem, however their work is mainly routability driven and does not handle coupling and timing. In [4, 24, 34], coupling noise is avoided without considering timing. A similar stage between global routing and detailed routing is crosspoint assignment [21, 22]. When a wire route goes from one global cell across a boundary to another global cell, a crosspoint assignment determines the crossing location on the boundary. In [21], a coupling aware timing driven crosspoint assignment heuristic is proposed. This is a greedy approach that assigns wire segments sequentially in an order of timing criticality. After initial assignment, wire spacings are tuned to further reduce coupling capacitance for timing critical nets.

In this chapter, we propose a new heuristic on timing driven track routing considering coupling capacitance. The objective is to find a feasible track assignment such that the minimum timing slack among all nets is maximized. In timing estimation, delay penalties due to both coupling capacitance and wire detour are counted. The heuristic proceeds in a panel (a row or a column of global cells) by panel manner. The track assignment problem within each panel is modeled by graphs. The problem of minimizing coupling induced timing penalty is equivalent to finding the minimum weight Hamiltonian path in a clique. Minimizing timing penalty due to wire detour can be solved via the minimum weight matching in a bipartite graph. The hard problem of minimizing both timing penalties simultaneously is formulated and solved as a Sequential Ordering Problem(SOP) [36]. A post processing step is performed to further improve timing and routability. Our heuristic is designed primarily for global wires and is more effective on upper layers where less pre-fixed local wires exist, even though it is capable of handling routing blockages. A major contribution of this work is that timing is optimized directly with consideration of both detour and

coupling induced delay. Experimental results on benchmark circuits demonstrate that our heuristic can remarkably improve coupling aware timing performance than other coupling aware approaches.

B. Preliminaries

1. Coupling Capacitance and Its Impact on Delay

Given two wire segments i and j, as shown in Figure 5, the coupling capacitance between them can be expressed as follows [2, 21]:

$$CC(i,j) = \alpha \cdot f_{ij} \cdot \frac{Len_{ij}}{Dist_{ij}^{\beta}} \tag{3.1}$$

where Len_{ij} is the coupling length, $Dist_{ij}$ is the wire spacing between i and j, α and β are technology dependent constants and f_{ij} is the *switching factor* for i and j. A switching factor is a real number between 0 and 1 according to [2] that indicates switching activity relations between two nets. The worst case coupling occurs when two adjacent signal nets make transitions at the same time but at opposite direction. For this worst case coupling, the switching factor between the two wires is 1. On the other hand, if two adjacent wire segments switch at the same time and at the same direction, the switching factor is 0. Any other switching activity relations should fall into the range between these two cases. Other definitions for the switching factor can be adopted easily to the works in this dissertation. As the spacing between two adjacent wire segments increases, the coupling capacitance between them decreases rapidly since emperical experiments in [20] showed β is approximately the constant of 2. Therefore, for track/detailed routing with fixed pitch, it is reasonable to assume that coupling capacitance occurs only between segments in adjacent tracks. This model is used to estimate the coupling capacitance throughout this dissertation.

Once the amount of coupling capacitance is calculated, the coupling induced delay

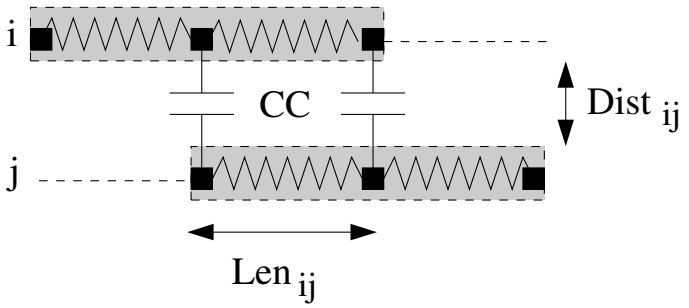


Fig. 5. Coupling capacitance between two wire segments i and j.

to each sink of a net can be found easily. Same as many other timing driven routing works [21, 31], the Elmore delay model [37] is employed for estimating wire delay in this dissertation. Even though the Elmore model is sometimes inaccurate, it has high fidelity [38] that can provide proper guidance for a combinatorial optimization. Moreover, layer/track assignment is a relatively early stage of wire timing optimization which allows and needs simple models. A more accurate model can be employed at later wire timing optimization stages to fine tune the coupling aware timing.

In Figure 6, we use a simple example to illustrate the coupling induced delay. Consider a net N and a wire segment $i \in N$. If segment i has an estimated coupling capacitance of CC, the extra delay due to CC for the critical sink of N is $R_{sa} \cdot CC + R_{ab} \cdot \frac{CC}{2}$ where R_{sa} is the path resistance from the source to point a and R_{ab} is the resistance between a and b. Please note that the same amount of coupling capacitance may cause different amount of delay depending on the coupling location.

2. Problem Formulation

Since our track routing takes a global routing solution as input, the whole routing region is tessellated into a grid of global cells (*GCs*) $\{g_{11}, g_{12}, ..g_{21}, g_{22}, ...\}$. A global routing solution for a net N_i is expressed in terms of the GCs that the wiring route of the net passes through. A complete row (column) of GCs is called a *panel*. In track routing, only wire

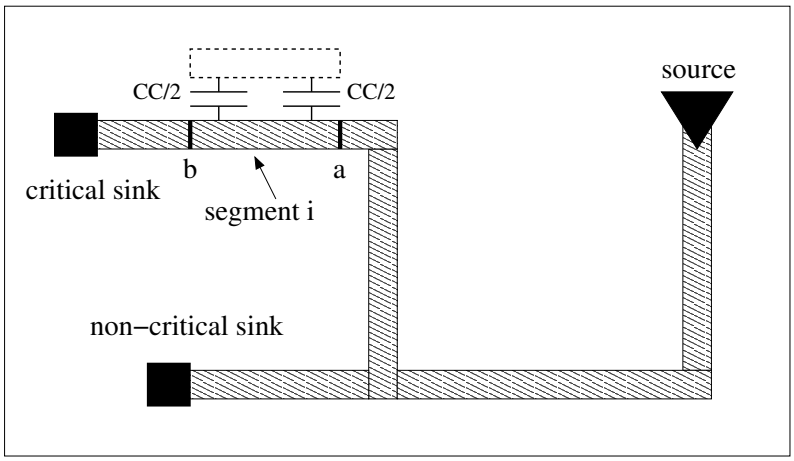


Fig. 6. Coupling capacitance between two wire segments.

segments whose lengths are at least one GC are considered. All the other small segments are handled in detailed routing. In gridded routings, each horizontal(vertical) panel consists of a set of K_p uniformly spaced horizontal(vertical) routing tracks. Some part of a track may be pre-occupied by certain blockages and cannot accommodate additional wire segments.

Given a net N_i with source node i, the timing slack s_{ik} for a sink k of the net is $s_{ik} = \tau_k - \chi_{ik}$ where τ_k is a timing constraint or required arrival time at sink k and χ_{ik} is the delay from source i to k. The sink with the minimal s_{ik} is called the critical sink of N_i and we use $S^i_{min} = min(s_{ik}), \forall k \in$ *sinks of* N_i to represent the minimum timing slack of net N_i. Throughout this chapter, we will use min_slack as an alternative to represent minimum timing slack. The problem formulation is described as follows.

Coupling Aware Timing-Driven Track Routing Problem: *Given an array of panels* $\{p_1, p_2, ...\}$ *each of which is composed of* K_p *routing tracks with certain blockages, a set of nets* $\{N_1, N_2, ..., N_i, ...\}$ *each of which is composed of segments* $\{N_{i1}, N_{i2}, ..., N_{ij}, ...\}$*, and timing constraints for each net, assign each segment* N_{ij} *to a track of its corresponding panel* p *in a non-overlapping manner such that the minimal slack among all nets is*

maximized.

C. Algorithm

In general, the number of wire segments in a routing procedure is huge and even a routability driven track routing problem is NP-complete [3], therefore a "divide and conquer" strategy is employed in this work. Since each panel consists of only horizontal/vertical wire segments, we solve the track routing problem panel by panel. Figure 7 shows an example of assigning wire segments *a, b, c, d, e* and f to the tracks of a horizontal panel of capacity $K_p = 4$.

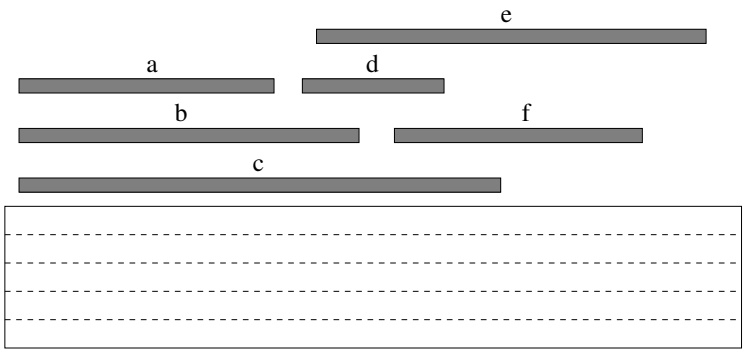


Fig. 7. An example of the track routing problem.

1. Constraint Graph

For each horizontal (vertical) panel p, we construct an undirected constraint graph $CG(V, E)$ with each vertex $v_i \in V$ corresponding to a segment i in panel p. There is an edge $e(v_i, v_j)$ between vertices v_i and v_j if there exists a span overlap between i and j. Figure 8 shows the constraint graph for the track routing example used in Figure 7. If segment i belongs to net N_m and segment j is a part of net N_n, the edge weight $wt(v_i, v_j)$ is defined as follows.

$$wt(v_i, v_j) = \frac{d_{ji}}{S_{min}^{N_m}} + \frac{d_{ij}}{S_{min}^{N_n}} \tag{3.2}$$

where d_{ji} and d_{ij} are delay costs defined in Section II-A. Note here d_{ji} and d_{ij} could be quite different depending on the coupling location of segment i, j of net N_m, N_n, respectively. The minimum timing slack $S_{min}^{N_n}$ and $S_{min}^{N_m}$ are used as weighting factors. In case of negative timing slacks, a large positive offset is temporarily added to the min_slack of *every* net in the constraint graph, only for calculating $wt(v_i, v_j)$. It is easy to observe that the edge weight between critical nets with strong coupling will be significantly higher. Our graph based algorithm, which will be discussed later, is able to avoid the adjacency of such pair of nets if their edge weight is large. Therefore, this edge weight definition enables us to optimize timing slacks and coupling-induced-delay simultaneously. Previous works [3, 4], however, are primarily focused on minimizing the total amount of coupling capacitance.

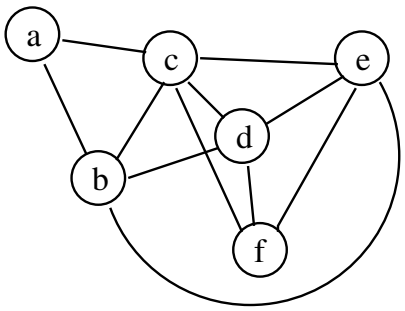


Fig. 8. A constraint graph.

Since the size of a constraint graph is generally very large, we consider it in a clique-by-clique manner [3, 4, 24]. A clique is a complete subgraph of the constraint graph. Each vertex (segment) in a clique has span overlap with every other vertex (segment) in the same clique. The size of a clique tells the minimal number of routing tracks needed to route all the segments in the clique. In the constraint graph, the largest clique in the graph is processed first. If there exist more than one largest cliques with the same size, we choose a clique with the maximal span. After performing track assignment for the largest clique, we remove this clique from the constraint graph and start to process the next largest clique.

Finding the largest clique can be done in polynomial time since our constraint graph is also an interval graph [3],

For each clique, a track assignment solution corresponds to a Hamiltonian path. If only coupling capacitance induced delays are considered, our objective becomes to find a Hamiltonian path that can provide a maximal min_slack among all the segments in the clique. Based on the edge weight definition that considers both coupling induced delay and timing criticality, our problem becomes a minimum weight Hamiltonian path problem.

2. Wire Detour Induced Delay and Bipartite Graph Model

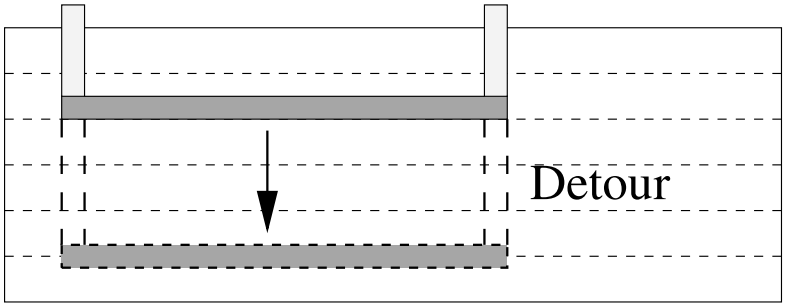


Fig. 9. An example of detour.

A track assignment result may affect delay through wire detour in addition to coupling capacitance. As illustrated in Figure 9, if a horizontal segment has both ends connected with two vertical segments upward, assigning this segment to a lower track will lead to longer detour and greater delay. Therefore, detour and detour induced delay need to be reduced. This problem can be modeled in a bipartite graph $BG = (V, E)$. The vertex set V is composed by V_w representing wire segments and V_r indicating routing tracks ($V = V_w \cup V_r$ and $V_w \cap Vr = \emptyset$). If a wire segment i is allowed to be assigned to track b, there is an edge $(v_i, v_b) \in E$. Vertex $v_i \in V_w$ and $v_b \in V_r$ represent segment i and

track b, respectively. The edge weight for (v_i, v_b) can be defined as $\frac{e_{i,b}}{S^i_{min}}$ where $e_{i,b}$ is the slack change when the segment i is moved from the minimum detour location to track b. If we only consider the detour induced delay, our problem can be formulated as a minimum weight matching problem for the bipartite graph. The edge weight can be defined to include other design concerns besides the detour induced delay penalty.

3. Sequential Ordering Problem Based Track Routing Algorithm

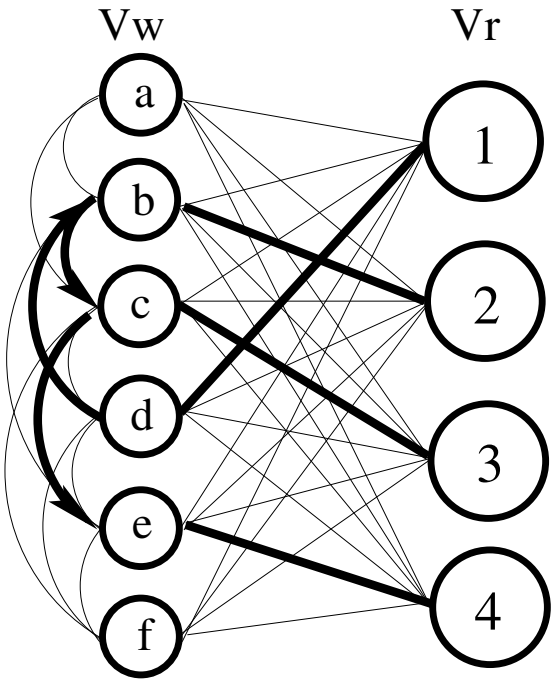


Fig. 10. The integration of a constraint and a bipartite graph.

When both the coupling capacitance induced delay and the detour induced delay are considered simultaneously, the clique model and the bipartite graph described in the previous subsection can be integrated into a hybrid graph as shown in Figure 10. In Figure 10, the highlighted edges on the constraint graph represent a Hamiltonian path for clique

(d, b, c, e) and the highlighted edges on the bipartite graph part indicate a matching solution. In this solution, wire segments b, c, d and e are assigned to track $2, 3, 1$ and 4, respectively. The integrated problem becomes to find a minimum weight Hamiltonian path on the clique part and a minimum weight matching on the bipartite part simultaneously. In addition, the min-path solution and the min-matching solution should be compatible with each other. The minimum weight Hamiltonian path problem alone is notoriously hard and the integrated problem is obviously more difficult.

The special difficulty of this problem is the partial correlation between two different kinds of cost (coupling induced delay cost and detour induced delay cost) in the objective. We solve this difficulty by transforming the detour induced delay cost into precedence constraints to the Hamiltonian path problem. A precedence constraint (i, j) tells that vertex i is required to precede j in the tour of a Hamiltonian path.

The cost-constraint transformation is carried out according to segment types defined as follows. (1) Type $\cup$ segments are those horizontal segments with two ends connected upward with vertical segments [1]. (2) Type $\cap$ segments are the horizontal segments with two ends connected downward with vertical segments. (3) Type H segments are those connected to hard pins in the panel. (4) "don't care" segments are the other horizontal segments in the panel. These segment types are illustrated in Figure 11.

Then, we add the following precedence constraints to the Hamiltonian path problem: *for any segment $i \in$ type $\cup$ set, there is a precedence constraint (i, j), $\forall j \in$ type $\cap$ group*. This set of precedence constraints can limit the vertical wire lengths for both type $\cup$ and $\cap$ wire segments. As a result, delays induced by redundant vertical wire lengths are alleviated. Other constraints can be added in a similar way to tighten the flexibility of a Hamiltonian path. For example, we can enforce a $\cup$ type segment to precede a H type segment in the

[1] Here we categorize these types in the scenario of a horizontal panel as in Figure 11. Segments in vertical panels can be categorized in the same way.

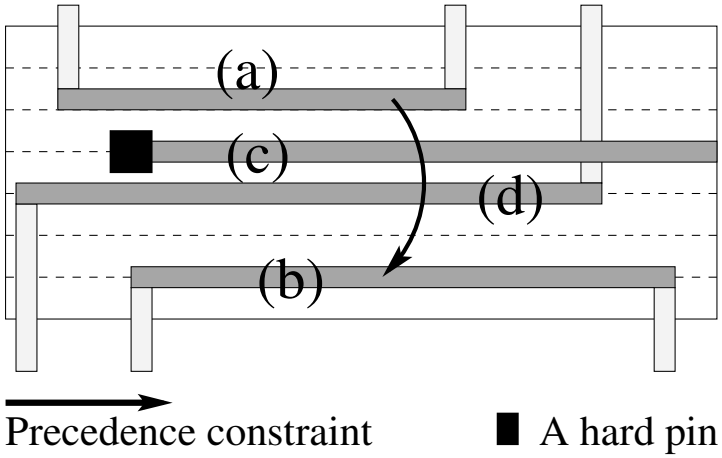


Fig. 11. (a) a type ∪ segment. (b) a type ∩ segment. (c) a type H segment. (d) a "don't care" segment.

lower half of the panel. The additional constraints may further reduce detour induced delay cost. However, they may degrade coupling induced delay cost as the solution space for the Hamiltonian path problem is reduced. Therefore, we believe our precedence constraint definition is generally sufficient to minimize detours while leaving enough solution space for the Hamiltonian path problem.

Finding a minimum weight Hamiltonian path with precedence constraints is known as a Sequential Ordering Problem (SOP) [36], which is an extension to the asymmetric traveling salesman problem (ATSP) with additional precedence constraints: given a complete graph G with directed weighted edges, a precedence constraint set R and designated start and terminal vertices, SOP finds a minimum weight Hamiltonian path from the start vertex to the terminal vertex which observes the precedence constraints. The precedence constraint set R can be represented by pairs of vertices (i, j) $(i \neq j)$. For each precedence constraint $(i, j) \in R$, vertex i is required to precede vertex j in the tour of Hamiltonian path. The SOP is also known to be an NP-complete problem. In our constraint graph, each undirected edge can be converted to two directed edges with the same edge weight.

Following the SOP formulation, we insert the start and terminal vertices as *dummy* vertices to the clique. They are connected to all the other vertices in the clique by zero weight edges. If the size of a clique is less than the panel capacity K_p, vertices denoting the empty tracks are inserted to the clique and we call them *empty* vertices. Any edge associated with an empty vertex has a weight of zero which indicates no delays are caused by an adjacent empty track.

Once the precedence constraints for our SOP formulation are decided, we finalize and send our formulation to an SOP solver and the SOP solver returns a minimum weight Hamiltonian path which follows the precedence constraints. We call this returned Hamiltonian path an SOP *tour*. Now let us look at a simple example of the SOP tour. As shown in Figure 12, suppose we have a clique with three segments $s1$, $s2$, $s3$, panel capacity $K_p = 4$ and a precedence constraint $(s1, s2)$. Start and terminal vertices are denoted as S and T in the graph, respectively. We also insert an *empty* vertex (E) to the clique to indicate there is an empty track. Then we feed this SOP formulation to the SOP solver. If the SOP tour is $S \rightarrow s1 \rightarrow E \rightarrow s3 \rightarrow s2 \rightarrow T$, then we know that s1, s2 and s3 are assigned to track 1, 4, and 3, respectively and no segment is assigned to track 2.

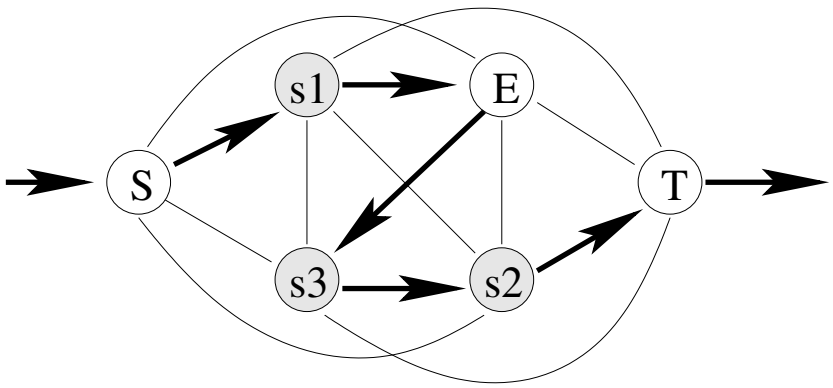


Fig. 12. An example SOP tour.

4. Post SOP Improvement

For each clique, its initial track routing solution from the SOP tour may be infeasible because conflicts can occur between a segment and existing blockages. Existing blockages are the pre-routes [3] and previously fixed wire segments in a panel. A track assignment for a segment i to a track t is called *failed* if i is entirely "embedded" in a blockage at t, as illustrated in Figure 13. For other types of conflicts, as the two examples illustrated in Figure 14, we allow segments to split because the non-conflicting portions of the segment i can still be assigned to track t while the conflicting portions of i become new (floating) segments, which will be represented as new vertices being inserted back to the constraint graph for later processing.

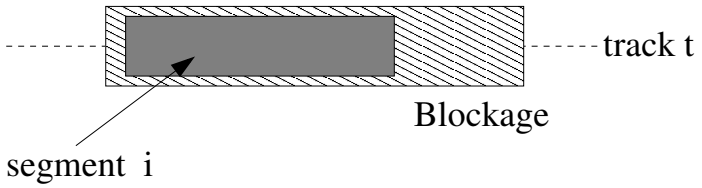


Fig. 13. Failed track assignment for a segment.

For an SOP solution, we further improve its timing and routability through an iterative procedure. We associate a $score$ with a track routing solution (a Hamiltonian path in the clique), as defined in Equation 3.3.

$$score = \alpha \cdot min\_slack + \beta \cdot N_f \tag{3.3}$$

where min_slack is the minimum slack for all segments in the clique under current track assignment considering both coupling capacitance and wire detours, N_f is the total length of overlap between the segments and blockages, α and β are weighting constants. For each iteration, we attempt to switch each pair of the segments in the clique and accept the switch

that can produce the best score. This process continues until no further improvement or a pre-defined iteration number is reached.

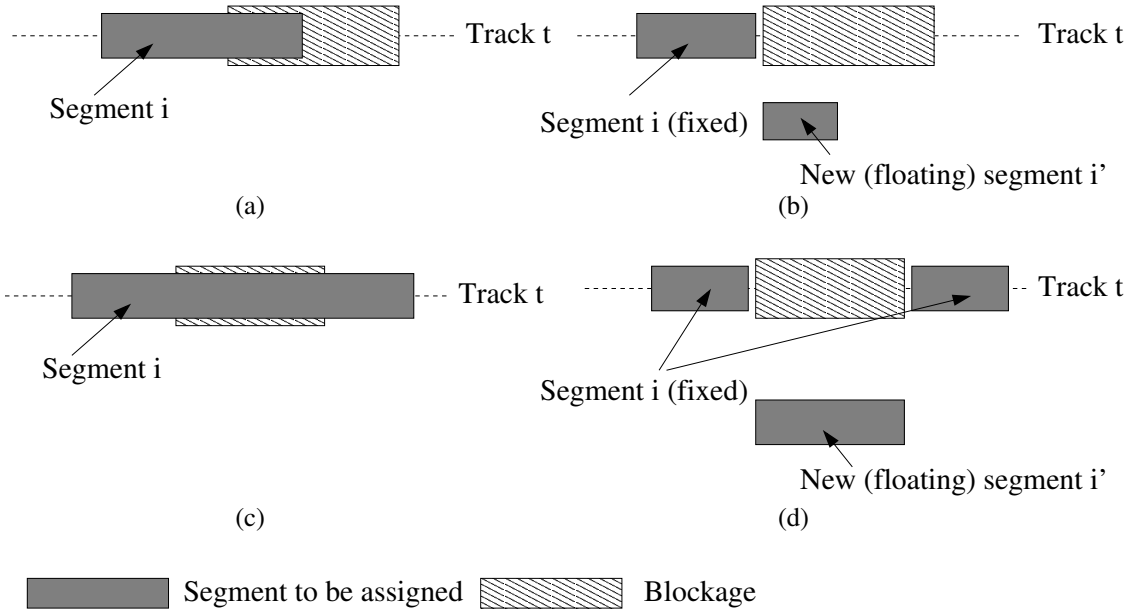


Fig. 14. Segments overlapping with blockages in (a) and (c) are split to fixed and floating segments in (b) and (d), respectively.

Once a final track assignment solution is determined, the failed nets (if there is any) are labeled and the floating segments (if there is any) are inserted back to the constraint graph as unassigned segments. After these new floating segments are inserted back to the constraint graph, we update the constraint graph to reflect the changes. For the segments that are assigned, we fix them to the corresponding tracks and they become blockages as we proceed to the next clique.

This procedure is continued until no more vertices are left in the constraint graph, then we update the timing slacks for all segments in this panel through the delay computation. We iterate this procedure for each panel and the final track routing solution is reached once all panels are processed. The overall track routing algorithm is summarized in Algorithm

14.

Algorithm 2: Timing-driven track routing.

Input : Timing constraints for all sinks for each N_i and its wire segments in terms of the global cells (GCs).

Output : Track assignments for segments.

Objective: Maximizing the minimum timing slack among all nets.

```
1  while not all panels are processed do
2      Construct a constraint graph for the current panel;
3      while constraint graph ≠ empty do
4          Find the largest clique C_max in the constraint graph;
5          Remove C_max from the constraint graph;
6          Formulate the track assignment problem on C_max as an SOP;
7          Obtain initial track routing solution by calling SOP solver;
8          Iterative improvement on track routing solution;
9          Insert vertices for floating segments to constraint graph;
10         Update the constraint graph;
11     end
12     Update the timing slacks;
13     Proceed to the next panel in the circuit;
14 end
```

D. Experimental Results

We have implemented our timing-driven track routing algorithm in GNU C++ on a Linux platform with an 1.4GHz Intel Centrino processor and 256MB memory. We have tested our algorithm on the ISPD98 benchmark circuits [39] and the circuit specifications are listed

in Table IV. Since there is no timing information on these benchmark circuits, timing constraints are randomly generated. Switching activities f_{ij} between any pair of nets are also randomly selected from 0 to 1. The global routing solutions are obtained using a rip-up and re-route router similar to [40]. An SOP solver [36], which is based on Genetic Algorithm (GA), is used in our experiment. The only previous work close to our problem formulation is by Tseng [21], but in Tseng's work, tracks (crosspoints) are assigned using a sequential greedy approach, followed by a space relaxation algorithm as a post-processing procedure. Another crosstalk- and performance-driven greedy track routing algorithm is introduced in Ho's work [4]: for each clique, the longest segment l is first assigned to the uppermost available track. Then it chooses a segment h which has the min-coupling edge with l and assign h to the first available track starting from the top, then an unassigned segment having the min-coupling edge with h is considered for the next iteration. This procedure iterates itself until no more un-assigned segment is left in the clique. If there is no available track for a segment h, h is declared as a failed segment. We have implemented the following experiments for comparison:

1. **GreedyCC**: a greedy heuristic similar to the work of [4] except that we improved their method by utilizing empty tracks (if available) as shields to further reduce coupling capacitance.

2. **SOPCC**: SOP method except that it simply considers coupling capacitance, not its effect on timing - this is done by assigning the edge weight $wt(v_i, v_j)$ in the constraint graph as the coupling capacitance between i and j.

3. **SOPNPC**: SOP method with no considerations of wire detours - this is done by dropping the precedence constraints.

4. **SOP**: our purposed SOP method considering coupling and detour induced delays simultaneously.

Table IV. Circuit specification.

circuit name	no. of global cells	no. of global nets	no. of segments	no. of tracks h. panel	no. of tracks v. panel
ibm01	64 × 64	8.8K	39.8K	20	18
ibm02	80 × 64	15.7K	100.3K	35	43
ibm03	80 × 64	14.6K	79.4K	28	30
ibm04	96 × 64	17.9K	92.2K	34	37
ibm05	128 × 64	19.3K	247.5K	63	67
ibm06	128 × 64	21.9K	153.4K	30	35
ibm07	192 × 64	29.0K	225.7K	41	46
ibm08	192 × 64	36.3K	262.2K	35	43

Table V. Experimental results.

Circuit Name	min. slack (ps)				completion rate			
	GreedyCC	SOPCC	SOPNPC	SOP	GreedyCC	SOPCC	SOPNPC	SOP
ibm01	-31	-26	-24	-18	98.85%	99.96%	99.92%	99.74%
ibm02	-348	-246	-248	-225	98.47%	99.91%	99.78%	99.60%
ibm03	-380	-320	-308	-278	96.73%	99.94%	99.94%	99.99%
ibm04	-167	-164	-135	-133	98.44%	99.99%	99.98%	100.00%
ibm05	-647	-772	-725	-607	95.26%	99.98%	100.00%	99.98%
ibm06	-345	-306	-333	-275	96.63%	99.98%	100.00%	99.84%
ibm07	-181	-208	-206	-145	97.84%	99.99%	99.98%	99.93%
ibm08	-177	-166	-209	-150	97.64%	99.96%	99.93%	99.90%
Avg.	-285	-276	-274	-233	97.48%	99.96%	99.94%	99.87%

Table V and VI lists the experimental results of the GreedyCC, SOPCC, SOPNPC and SOP approach. Completion rate is defined as the number of non-failed segments versus the total number of segments. The experimental results show that the GreedyCC method, although faster, produces poor completion rate and min_slack. Compare to SOPCC and SOPNPC, the SOP method yields greater min_slack at the same level of completion rate. The CPU time for our SOP method is reasonable considering the size of the circuits and the GA based SOP solver which dominates our CPU time. This makes our method scalable

Table VI. CPU time.

Circuit name	GreedyCC (min:sec)	SOPCC (min:sec)	SOPNPC (min:sec)	SOP (min:sec)
ibm01	0:42	1:35	1:38	1:33
ibm02	1:58	5:50	5:55	5:50
ibm03	1:31	5:04	5:23	5:01
ibm04	2:10	5:35	5:43	5:36
ibm05	6:41	25:24	24:33	24:46
ibm06	3:27	8:06	7:52	7:49
ibm07	7:51	18:22	18:45	16:38
ibm08	8:03	17:47	18:05	17:44

for larger circuits. In summary, the experimental results confirm the effectiveness of our approach and lead to the following conclusions: (1) In timing-driven routing, merely considering coupling capacitance is not sufficient and its effect must be directly incorporated into the delay computation. (2) The effect of wire detours presents an important role, however, it must be considered with the coupling capacitance induced delay simultaneously.

E. Conclusion

In this chapter, we have proposed a new timing-driven track routing algorithm considering both coupling capacitance and wire detours. Unlike most of the previous work that considers only coupling capacitance, we optimize timing slack directly. We formulate our track routing problem as two graph problems (minimum weight Hamiltonian path and minimum weight bipartite matching) and further integrate them into a sequential ordering problem. Experiments showed significant timing improvement on benchmark circuits.

CHAPTER IV

COUPLING AWARE TIMING OPTIMIZATION AND ANTENNA AVOIDANCE IN LAYER ASSIGNMENT

In this chapter, we re-visit the layer assignment stage of physical design and propose techniques to handle the coupling aware timing and the antenna problem simultaneously during this stage. An improved probabilistic coupling capacitance model is suggested for coupling aware timing optimization. The antenna avoidance problem is modeled as a tree partitioning problem with a linear time optimal algorithm solution. This algorithm is customized to guide antenna avoidance in layer assignment. A linear time optimal jumper insertion algorithm is also derived. Experimental results on benchmark circuits show that the proposed techniques can lead to an average of $270ps$ timing slack improvement validated by track assignment, 76% antenna violation reduction and 99% via violation reduction.

A. Introduction

The sustained progress of VLSI technology has altered the landscape of routing which is a major physical design stage. In addition to traditional objectives such as congestion and self RC dominated wire delay, routing tools need to handle problems emerged from deep submicron era: coupling capacitance dominated wire delay and the antenna effect in manufacturing.

The coupling capacitance problem has been recognized for a long time. However, most of previous works [2,4,12,13,15,19,34] are focused only on controlling the amount of coupling capacitance. These techniques are useful for reducing coupling induced glitches but are inadequate for reducing coupling induced signal delays, as we have discussed in details at chapter III. Therefore, merely controlling the amount of coupling capacitance is not adequate and coupling induced timing must be optimized directly.

The technology scaling also gives rise to manufacturability problems among which the antenna effect is directly related to routing. In the manufacturing process, conductors (such as gate poly and metal), which have not been covered by a shielding layer of oxide, act like antenna that collect charges when exposed directly to the plasma [5]. If the charged conductors are connected to only a gate oxide, Fowler-Nordheim (F-N) tunneling current will discharge through the thin oxide and cause gate damage. The conductors connected only to the gate oxide, which are normally in lower routing layers, are called *antennas* (see Figure 15(a)). On the other hand, if the charges can be released through a low impedance path connected to a diffusion, gate damages can be avoided. The risk of the antenna damage to the gate oxide is proportional to the area and perimeter length of the antenna and inversely proportional to the area and perimeter length of the gate oxide.

There are two major existing approaches on antenna avoidance: (1) jumper insertion and (2) diode insertion. The jumper insertion approach [7] is based on the fact that wire segments on top routing layers are normally fabricated at the end and therefore always have low impedance path connected to diffusions. Therefore, a long antenna can be cut shorter by switching the wire to the top layer for a short length and then switching it back to its original layer as shown in Figure 15(b). The short segment on the top layer is called *jumper*. Evidently, jumpers introduce extra vias and therefore degrade both manufacturing yield and circuit timing performance. Diodes can be placed near the gates with antenna violations and protect the gates by restraining the charge voltage level [6, 41]. However, diode insertion depends on placement space and diodes present extra capacitive load to the signal nets they are attached to.

On handling the coupling aware timing and the antenna problem, each step of routing (global routing, layer assignment, track routing and detailed routing) has its own advantage and weakness. In general, detailed optimizations [12,13,15,18] have more definite relevant information but are normally restricted by global optimization results. In contrast, global

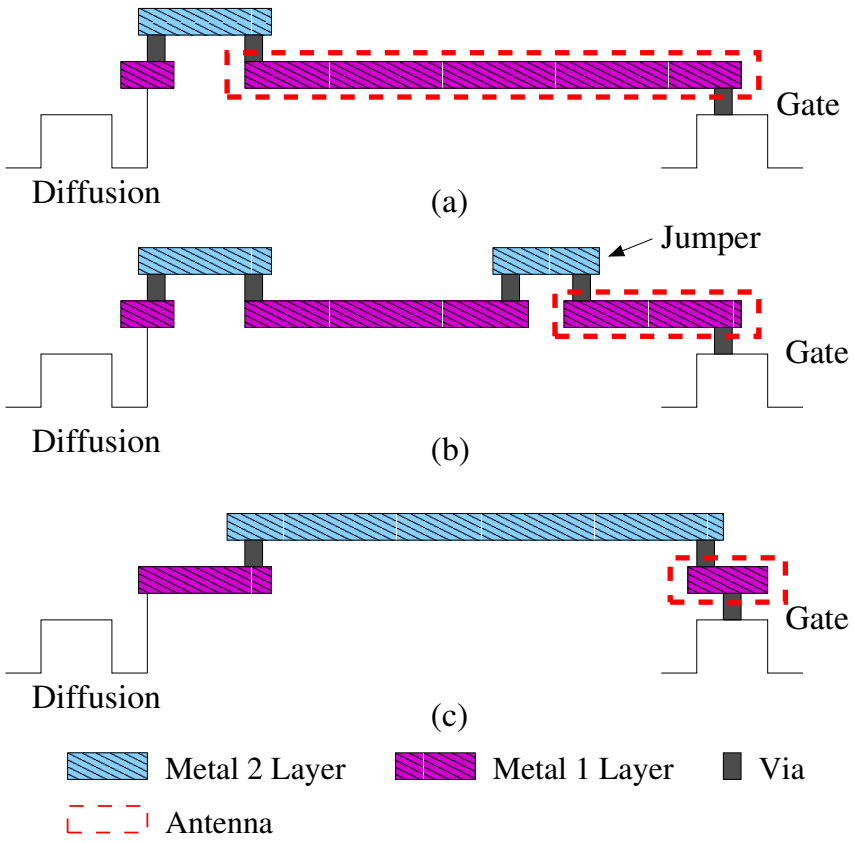


Fig. 15. (a)An antenna. (b) Reduce antenna length by inserting a jumper. (c) Reduce antenna length by layer assignment.

level optimizations [2, 19, 35] have much greater flexibility on making changes but lack precise information for evaluating solution quality. When the coupling aware timing and the antenna problem are considered together with traditional design objectives, the high complexity of the problem implies that efforts need to be made in *every* step to provide a complete solution. Among these routing steps, layer assignment[1] (similar as track routing [3,21,22,34]) is in the middle and has a good compromise between information availability and optimization flexibility. Coupling aware timing optimization and antenna avoidance in layer assignment can make subsequent optimizations in detailed routing much easier.

[1]There are other works on pre-global-routing layer assignment.

In [23], an optimal minimum crosstalk layer assignment algorithm is proposed with the assumption of VHV channel routing and pre-fixed horizontal wires. A combined coupling capacitance driven layer/track assignment technique is reported in [24]. However, this work did not evaluate coupling capacitance quantitatively and only attempted to disallow wires of simultaneous switching nets to be adjacent with each other. In [42], a layer assignment heuristic is proposed for crosstalk risk minimization without directly optimizing timing. Layer assignment is also a proper step for handling the antenna problem since they are directly related with each other. By assigning certain segments to top routing layers, antenna violations can be avoided naturally as shown in Figure 15(c). In [43], an antenna avoidance driven layer assignment algorithm is reported. However, it is restricted to only HVH channel routings.

In this work, we consider the coupling aware timing and the antenna avoidance simultaneously in layer assignment together with traditional objectives including congestion and via constraints. In order to improve the coupling aware timing performance, an improved probabilistic coupling capacitance model is proposed. The antenna avoidance problem is modeled as a tree partitioning problem with a linear time optimal algorithm solution. The tree partitioning algorithm is extended for jumper insertion and generating dynamic guidance for antenna avoidance during the layer assignment. The jumper insertion algorithm can handle general Steiner tree topology in contrast to the work of [7] which is limited to spanning trees. Experimental results on benchmark circuits show that the proposed techniques can lead to an average of $270ps$ timing slack improvement validated by track assignment, 76% antenna violation reduction and 99% via violation reduction.

B. Problem Formulation

Our layer assignment takes a global routing result as input in which the entire routing region is tessellated into an array of *Global Routing Cells* (**GRCs**) $\{g_{11}, g_{12}, ..g_{21}, g_{22}, ...\}$ [2, 40]. The global route for a net is expressed in term of the GRCs that this net passes through. Only global wires spanning at least one GRC are handled in the layer assignment. A complete row (column) of GRCs is called a **panel**. In gridded routings, each horizontal/vertical panel consists of a set of K_p uniformly spaced horizontal/vertical routing tracks per layer. If a net passes through a GRC g, we assume it occupies an entire track in g for simplification. Even though the simplification is pessimistic on congestion estimation, the pessimism may compensate the optimism from neglecting local wires. For a long wire spanning multiple cells, we allow it to be segmented at the boundary of GRCs and each chopped segment can be assigned to different layers [25]. Since layer switching of the same wire route causes vias, we consider the constraint that the number of vias between two adjacent GRCs cannot be greater than a certain upper limit. Given a net N_i with source node i, the timing slack $s_{i,v}$ at a sink v of N_i is $s_{i,v} = \tau_v - \chi_{i,v}$ where τ_v is a timing constraint or required arrival time for sink v and $\chi_{i,v}$ is the delay from source i to v. Our problem formulation is given as follows.

Layer assignment for coupling aware timing optimization and antenna avoidance: *Given an array of panels* $\{p_1, p_2, ...\}$ *each of which is composed of* L_p *horizontal/vertical layers and* K_p *routing tracks-per-layer, via constraints between adjacent GRCs, global routes for a set of nets* $\{N_1, N_2, ..., N_i, ...\}$ *each of which is composed of segments* $\{e_{i1}, e_{i2}, ..., e_{ij}, ...\}$*, and timing constraints for each sink, assign each segment* e_{ij} *to a layer within its corresponding panel such that (1) the minimum timing slack among all nets is maximized and (2) the total number of antenna violations is minimized subject to panel routing capacity and via constraints.*

C. Improved Probabilistic Coupling Capacitance Modeling

During layer assignment, the wire adjacency information is not available and therefore it is hard to estimate the coupling capacitance directly. Many previous works on global routings [2, 19] estimated coupling capacitance through trial track assignment which faces a dilemma: a sophisticated track assignment [19] is too slow for just an estimation while a simplified trial track assignment [2] may behave quite differently from the actual track assignment in detailed routings and the estimation may be unreliable.

In chapter II , a pre-track-assignment probabilistic coupling capacitance model is proposed. The computation of this probabilistic model is very fast. Even though it is sometimes inaccurate, it can capture the general trend of coupling capacitance risk. However, the model in chapter II assumes that each wire has an equal chance to occupy a track. This assumption neglects the fact that subsequent track assignment and detailed routing often have certain crosstalk avoidance capability and tend to assign wires to sparse regions. Therefore, the model in chapter II contains certain pessimism.

Assuming that coupling capacitance occurs only between segments in adjacent tracks [2], we improve this probability model with anticipation of prospective crosstalk avoidance in subsequent track assignment. Given a routing region with n uniformly spaced tracks and k wire segments with identical length, the probabilistic coupling capacitance for a wire segment in this region is determined by the probability that its adjacent tracks are occupied by other wires. We include the following two anticipations in the probability computation.

- If the number of wires is no greater than half of the number of tracks, we can let the probability of wire adjacency be zero since plenty of empty tracks can be inserted between wires to avoid any wire adjacency.

[2]The simplification in this estimation is necessary for obtaining practical computation speed. Accurate models are more suitable for subsequent detailed optimizations.

- If the number of wires is greater than half of the number of tracks, we disallow any two empty tracks to be adjacent with each other. If there are two empty tracks adjacent with each other, the empty tracks are not fully utilized to shield wires from coupling capacitance.

The above anticipations are in accordance with the behavior of a typical crosstalk avoidance driven track assignment which normally attempts to utilize empty tracks as shields between wires.

For a wire segment which is called **target wire**, the probability that it has adjacent wires when $k > \lceil n/2 \rceil$ can be categorized into two scenarios: (1) $P_{r,1}$: probability that there is an adjacent wire on one side and there is an empty track on the other side; (2) $P_{r,2}$: probability that there are adjacent wires on two sides. The probability $P_{r,1}$ and $P_{r,2}$ can be expressed as $P_{r,1} = \pi_{k,n,1}/\pi_{k,n}$ and $P_{r,2} = \pi_{k,n,2}/\pi_{k,n}$ where $\pi_{k,n,1}$ is the number of permutations that the target wire has adjacent wire on one side, $\pi_{k,n,2}$ is the number of permutations that the wire has adjacent wires on both sides and $\pi_{k,n}$ is the total number of permutations. These permutations are limited to the situations of no two empty tracks adjacent with each other.

Improved probabilistic coupling capacitance model: *Given a one-layer routing region r with n uniformly spaced routing tracks and k wire segments with identical length, any wire segment in this region has probabilistic coupling capacitance of:*

$$C_r = \begin{cases} 0 & \text{if } k \leq \lceil n/2 \rceil; \\ CC \cdot (\frac{\pi_{k,n,1}}{\pi_{k,n}} + 2 \cdot \frac{\pi_{k,n,2}}{\pi_{k,n}}) & \text{otherwise.} \end{cases} \tag{4.1}$$

where

$$\begin{aligned} \pi_{k,n,1} = {} & 2\binom{k-1}{1}\binom{k-1}{n-k-1}(k-2)!(k-1) \\ & +2\binom{k-1}{1}\binom{k-1}{n-k}(k-2)! \quad (k > \lceil n/2 \rceil) \end{aligned} \tag{4.2}$$

$$\pi_{k,n,2} = 2\binom{k-1}{2}\binom{k-1}{n-k}(k-3)!(k-2) \quad (k > \lceil n/2 \rceil) \tag{4.3}$$

$$\pi_{k,n} = \binom{k+1}{n-k} \cdot k! \qquad (k > \lceil n/2 \rceil) \tag{4.4}$$

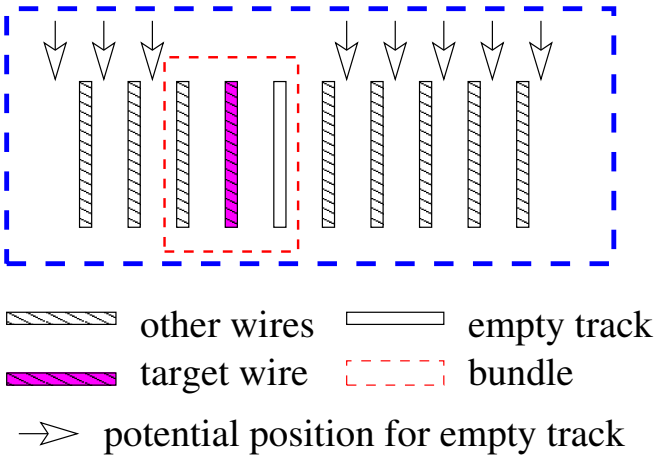


Fig. 16. $\pi_{k,n,1}$ case 1: the target wire has an adjacent wire on one side and an adjacent empty track on the other side.

1. Derivation of $\pi_{k,n,1}$ (Equation 4.2):

   - Case 1: The target wire i is not on any boundary track and it has one adjacent wire (Figure 16). First, another wire j is assigned to the neighbor of i and the adjacent track on the side is kept empty. Wire j can be at either side of i and this explains the first factor of 2 in the first line of Equation 4.2. Wire i, j and the empty track are tied together as a *bundle*. The wire j is selected among the remaining $k-1$ wires with equal chance and the second factor in the first line of Equation 4.2 indicates the number of the selections. Next, we "insert" the remaining $n-k-1$ empty tracks to the $k-1$ possible positions. The number of such insertions is represented by the third factor in line 1 of Equation 4.2.

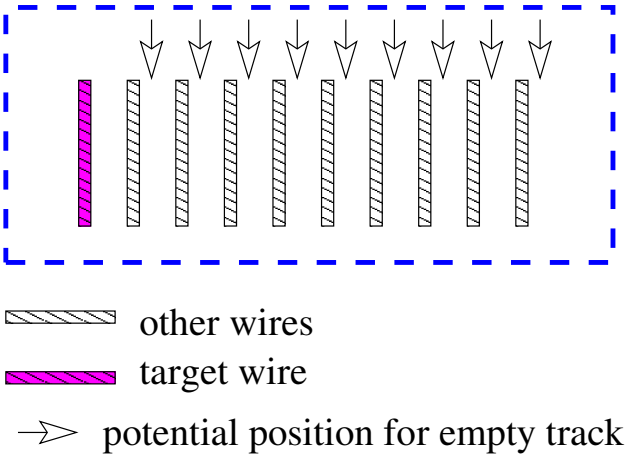


Fig. 17. $\pi_{k,n,1}$ case 2: the target wire is on a boundary track.

No empty track can be inserted inside or beside the empty track of the bundle. The bundle can be placed in $k-1$ positions relative to other wires. For each bundle position, there are $k-2$ permutations for the remaining wires.

- Case 2: The target net i is on a boundary track (Figure 17). In this case, there must be another wire j adjacent to i. First, we choose j among the rest $k-1$ wires. Then, we insert $n-k$ empty tracks to $k-1$ positions. For the remaining $k-2$ wires, there are $(k-2)!$ permutations in total. The second line of Equation 4.2 represents this case. The factor of 2 indicates that the wire i can be on either boundary of the region.

2. Derivation of $\pi_{k,n,2}$ (Equation 4.3):

   The target wire i has two adjacent wires (Figure 18). First, we choose two wires j and h among $k-1$ wires to be assigned adjacent with i. A bundle is formed by wire i, j and h. The wire j and h can be on either side of i and this explains the coefficient 2 in Equation 4.3. There are $n-k$ empty tracks which can be inserted to

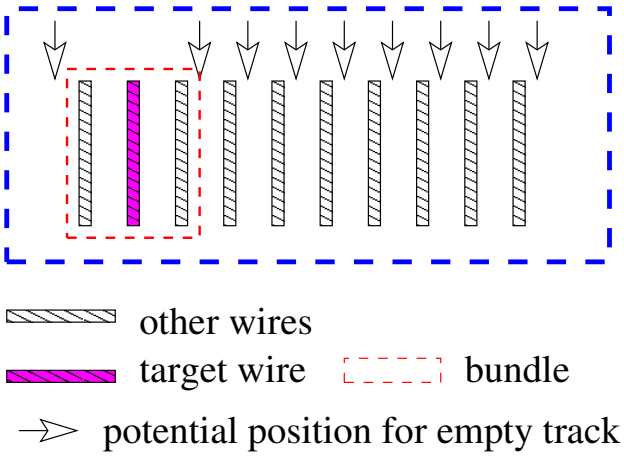


Fig. 18. $\pi_{k,n,2}$: the target wire has two adjacent wires.

$k-1$ positions among the wires except inside of the bundle. Relative to other nets, the bundle can be placed in $k-2$ positions. Last, we can obtain the permutations on the remaining $k-3$ nets.

3. Derivation of $\pi_{k,n}$ (Equation 4.4) :

   First, $n-k$ empty tracks are inserted among the k wires. Since empty tracks are not allowed to be adjacent with each other, there are $k+1$ positions for the n-k empty tracks. Thus, there are $\binom{k+1}{n-k}$ configurations. For each configuration, there are $k!$ permutations for all wires.

Figure 19 shows a comparison between the linear probabilistic coupling capacitance model [26] and the improved probabilistic model.

Once the estimated coupling capacitance is obtained, the coupling induced delay to each sink can be found easily. For details, please refer to chapter III.

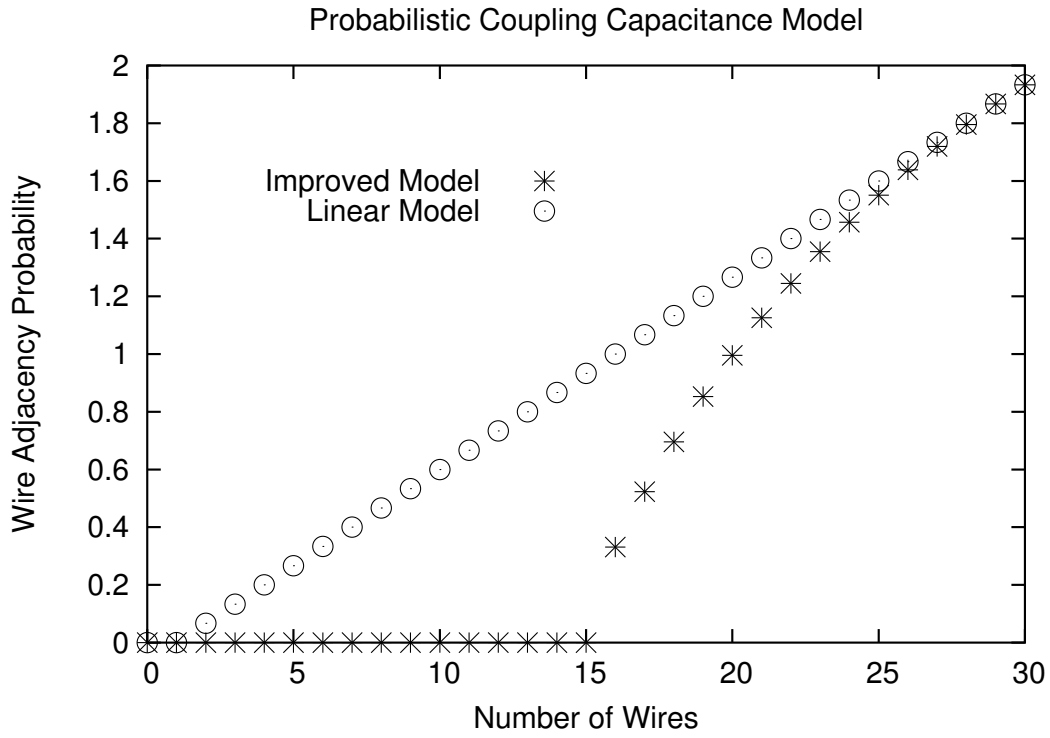


Fig. 19. Comparison of the improved probabilistic model with the linear model [26] when k = 0 to 30 and n = 30.

D. Antenna Avoidance through Tree Partitioning

In this section, we will model the antenna avoidance problem as a tree partitioning problem with linear time optimal algorithm. The tree partitioning algorithm framework can be extended for guiding antenna-avoidance-driven layer assignment and jumper insertion.

1. Tree Partitioning Problem Formulation

In routing or layer assignment, the term of *antenna* implies a sub-tree associated with a sink node in a Steiner tree for a specific net. Assume each edge e in a Steiner tree T has already been assigned to a layer $layer(e)$. For a sink node v in T, let L_{top} be the top-most layer containing an edge in the path from source to v. Then, the antenna for v is the maximal

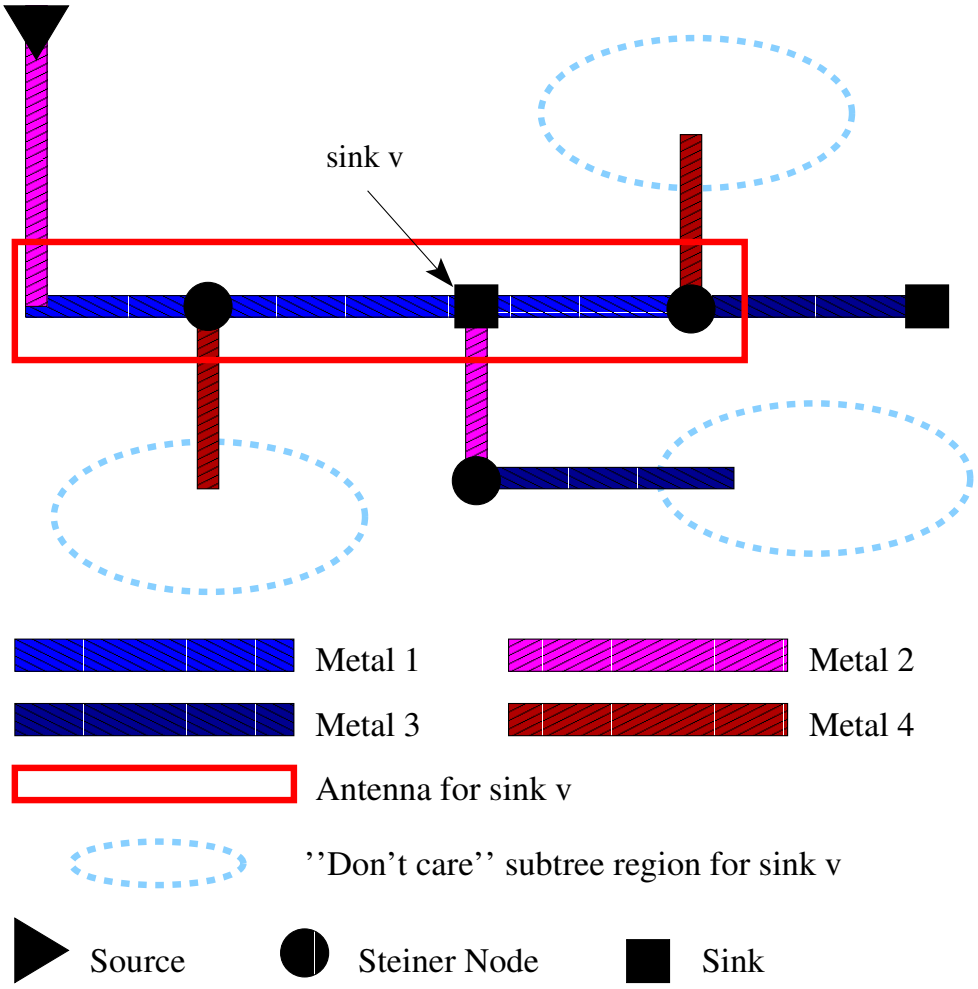


Fig. 20. Example: antenna for a sink v.

subtree that contains v and is surrounded by (but excluding) edges on L_{top} or layers higher than L_{top}. The edges on layer L_{top} or higher are called **separators**. If a breadth first search is performed starting from v, and the search is terminated whenever a separator is met, the visited edges form the antenna for v. In Figure 20, L_{top} for sink v is metal 2 and the antenna for v includes those edges in the highlighting box.

The *objective of antenna avoidance* is to ensure that the size (total wire length) for an antenna is no greater than certain bound A_{max} assuming uniform wire width. Non-uniform wire width can be handled easily by considering the total wire area of an antenna. The

charge sharing among multiple sinks can be considered by increasing A_{max} proportionally with the number of the sinks. Evidently, the size of an antenna can be reduced by properly placing separators around it. Since separators have to be assigned to the top layer, too many separators may intensify congestions on the top layer and may reduce the flexibility of coupling capacitance avoidance. Thus, the antenna avoidance problem can be formulated as:

Tree Partitioning for Antenna Avoidance(TPAA): *Given a Steiner tree, find the minimum number of separators such that the antenna size of each sink is no greater than* A_{max}.

This problem formulation is very similar to the traditional tree partitioning problem [44] [3] except the following differences: (1) The work of [44] considers to limit the total *node weight* of each subtree instead of total wire length; (2) The work of [44] applies the constraint of total node weight to subtrees *without* sinks as well.

2. Tree Partitioning Algorithm

In [44], a linear time optimal algorithm was proposed to solve the tree partitioning problem they formulated. With small modification, the algorithm of [44] can be applied to solve the TPAA problem. The pseudo code of the modified algorithm is given in Algorithm 3. This algorithm selects separators greedily in a tree traversal from the deepest level (leaf nodes) toward the source node. The subtree rooted at node u is denoted as T_u. The **weight** $W(T_u)$ indicates the total wire length in T_u. A **branch** of T_u is a subtree T_v plus its parent edge (u, v).

The critical step of Algorithm 3 is loop from line 4 to line 8 which is an iterative greedy branch removal. The set of child branches of node u is denoted as $B(u)$. If there are $m = |B(u)|$ branches in $B(u)$, a straightforward implementation of sorting or priority queue

[3] It is worth mentioning that an optimal tree buffering algorithm in [45] is also based on [44] .

Algorithm 3: Tree partitioning for antenna avoidance

Input : A Steiner tree.

Output: The minimum set C of separators.

```
C ← ∅;
for i ← maximum-level to 1 do
    while there is an unprocessed node u at level i do
        while W(T_u) > A_max and there is sink in T_u do
            remove the heaviest branch ((u, v) + T_v);
            C ← C ∪ (u, v);
            W(T_u) = W(T_u) − W((u, v) + T_v);
        end
    end
end
```

results in runtime of $O(m \log m)$. In [44], a linear time algorithm of SPLIT is suggested to implement this loop. For the completeness of the presentation, the SPLIT algorithm is described as follows.

The SPLIT algorithm attempts to partition $B(u)$ into two subsets $B_L(u)$ and $B_H(u)$ such that

$$q \in B_L(u) \quad \text{and} \quad r \in B_H(u) \Rightarrow W(q) \leq W(r) \tag{4.5}$$

$$\sum_{q \in B_L(u)} W(q) \leq A_{max} \tag{4.6}$$

$$\sum_{q \in B_L(u)} W(q) + W(r) > A_{max}, \forall r \in B_H(u) \tag{4.7}$$

After such partitioning is found, all branches in $B_H(u)$ are removed instead of going through the loop in Algorithm 3. The pseudo code of the SPLIT algorithm is given in Al-

gorithm 4. Initially, $w = A_{max}$ and $B(u)$ is partitioned into two subsets $B_l(u)$ and $B_h(u)$ satisfying condition 4.5 using a *Median-find-and-Halve* method and $|B_h(u)| \leq |B_l(u)| \leq |B_h(u)| + 1$. Then condition 4.6 and 4.7 are checked in $O(|B(u)|)$ time. If both conditions are satisfied (line 7 of Algorithm 4), the desired partition is found and returned. If 4.6 holds but 4.7 does not necessarily hold (line 8 of Algorithm 4), we continue to partition $B_h(u)$ into lower and higher subsets. If 4.7 holds but 4.6 does not hold (line 9 of Algorithm 4), then $B_l(u)$ is partitioned. The partitioning is repeated until all conditions are satisfied and the union of all generated $B_h(u)$ forms the $B_H(u)$ to be removed.

For example, the node u has five branches represented by the weight of 6, 2, 9, 7 and 4, respectively, and $A_{max} = 24$. We invoke the algorithm by calling *SPLIT({6,2,9,7,4},24)*. After the initial *Median-find-and-Halve*, the upper half partition is $\{9,7\}$ and the lower half is $\{6,2,4\}$ with $W_l = 12$. Because $W_l < w$, we proceed to call *SPLIT({9,7},12)*. This time the *Median-find-and-Halve* finds the upper half as $\{9\}$. Next, *SPLIT({9},5)* is called and the branch of weight 9 is returned. As a result, the branch with weight of 9 is removed from T_u.

Optimality. The optimality of Algorithm 3 can be directly derived from Lemma 1 and Lemma 2 below. Since Algorithm 3 is very similar to the tree partitioning algorithm in [44], the proof on optimality is also similar as [44].

LEMMA 1. Let p be a node in tree T such that $W(T_p) > A_{max}$ and $W(T_r) \leq A_{max}, \forall r \in S(p)$. $S(p)$ are the set of child nodes of p. Then there exists an optimal tree partitioning containing edge (p, r_0), where

$$W((p, r_0) + T_{r_0}) = \max\{W((p, r) + T_r), \forall r \in S(p)\}. \tag{4.8}$$

Proof: Since $W(T_p) > A_{max}$, any optimal partition C necessarily contains an edge from T_p. Let (u, v) be such an edge in C from T_p. Clearly, $C' = C - \{(u, v)\} + \{(p, r_0)\}$ is a feasible solution and also optimal.

Algorithm 4: SPLIT(B,w)

Input : A set B which contains branches to be partitioned.

Output: A set of B_h containing the minimum number of branches in B to be removed.

```
1  if |B| = 1 then
2      if W(B) ≤ w then return ∅ else return B;
3  else
4      Median-find-and-halve(B), B_h = higher half of B;
5      W_l = Σ_{q∉B_h} W(q);
6      switch the value of W_l do
7          case W_l = w return B_h;
8          case W_l < w return SPLIT(B_h, w − W_l);
9          case W_l > w return SPLIT(B − B_h, w) + B_h;
10     end
11 end
```

LEMMA 2. Let (p,r) be in some optimal partition of T. If C_1, C_2 are optimal partitions of $T - T_p$ and T_r respectively, then $C = C_1 + C_2 + \{(p, r)\}$ is an optimal partition for T.

Proof: Let C' be any optimal partition containing (p,r), and let C'_1, C'_2 denote the set of edges in C' from $T - T_p$ and T_r, respectively. Obviously, $|C_1| \leq |C'_1|$ and $|C_2| \leq |C'_2|$. Hence, $|C| \leq |C'|$.

Complexity. The *Median-find-and-Halve* method only takes linear time and the complexity for SPLIT algorithm is $O(|B(u)| + \frac{|B(u)|}{2} + \frac{|B(u)|}{4} + \cdots) = O(|B(u)|)$. Since every edge is processed only once in algorithm 3 with the SPLIT method, the overall complexity of algorithm 3 is $O(n)$ if n is the number of edges in the tree.

3. Constrained Tree Partitioning in Layer Assignment

In reality, not all edges can be assigned to the top layer L_{top} because L_{top} has a preferred routing direction and limited routing capacity. For example, if L_{top} is for vertical wires, a horizontal wire cannot be assigned to this layer. Therefore, we have to deal with a *constrained tree partitioning problem*. We define a **feasible branch** as a branch such that its root edge can be assigned to L_{top}. For a subtree T_u, a *maximal feasible branch* is a feasible branch in T_u but is not contained in another feasible branch. In Figure 21, the dashed edges cannot be assigned to L_{top} and there are 3 maximal feasible branches $(a, c) + T_c$, $(d, e) + T_e$ and $(u, b) + T_b$. The set of all maximal feasible branches for T_u is denoted as $B_{MF}(u)$. Let $A_{max,reduced} = A_{max} - (W(T_u) - \sum_{\forall b \in B_{MF}(u)} W(b))$. Then the constrained tree partitioning problem can be transformed to the TPAA problem by keeping only $B_{MF}(u)$ in T_u and replacing A_{max} with $A_{max,reduced}$. Of course, there is no feasible solution if $A_{max,reduced} < 0$. By this transformation, the constrained tree partitioning problem can be solved optimally in linear time through Algorithm 3. In implementation, $B_{MF}(u)$ for each subtree T_u is maintained in the bottom-up traversal so that no explicit transformation is necessary.

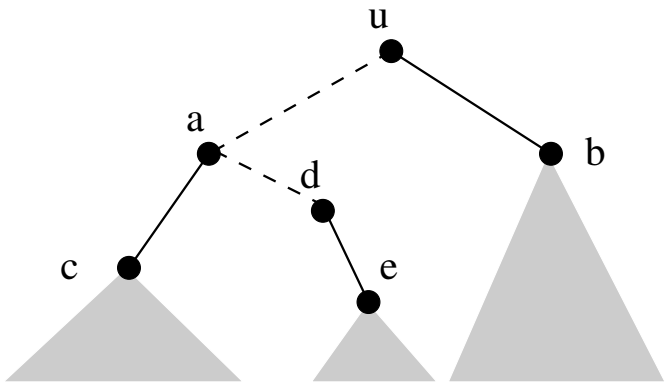


Fig. 21. Dashed edges cannot be assigned to L_{top} and there are 3 maximal feasible branches $(a, c) + T_c$, $(d, e) + T_e$ and $(u, b) + T_b$.

Before every edge along a source-sink path has been assigned to a specific layer, L_{top} is not well defined. In practice, we let L_{top} be the top-most routing layer available. Sometimes L_{top} can include a lower layer to allow greater flexibility in optimization. Consider an example where M1 and M3 are for horizontal wires, and M2 and M4 are for vertical wires. If a net is mostly composed by horizontal wires and there is no feasible tree partitioning using M4 as L_{top}, M3 can be utilized as L_{top} instead.

4. Tree Partitioning Based Jumper Insertion

Algorithm 3 can also be extended to Algorithm 5 for jumper insertion. The optimality proof of Algorithm 5 is shown as folows.

LEMMA 3. Let p be a node in T such that $W(T_p) > A_{max}$ and $W(T_r) \leq A_{max}$, $\forall r \in S(p)$. $S(p)$ are the set of children of node p. If there exists a branch $(p, \hat{r}) + T_{\hat{r}}$ and $W(\hat{r}) + w(p, \hat{r}) > A_{max}, \hat{r} \in S(p)$ Then there exists an optimal tree partitioning containing jumper j, where

$$W(T_{\hat{r}}) + w(j, \hat{r}) = A_{max} \tag{4.9}$$

Proof. Since $W(T_j) = A_{max}$, any optimal parition J necessarily contains a jumper in T_j. Let h be such a jumper in T_j. Clearly, $J' = J - h + j$ is a feasible solution and also optimal.

LEMMA 4. Let p be a node in T such that $W(T_p) > A_{max}$ and $W(T_r) + w(p, r) \leq A_{max}$, $\forall r \in S(p)$. $S(p)$ are the set of children of node p. Then there exists an optimal tree partitioning containing jumper j right below p on edge (p, r_0), where

$$W(T_{r_0}) + w(p, r_0) = max\{W(T_r) + w(p, r)\} \tag{4.10}$$

Proof. Since $W(T_p) > A_{max}$, any optimal parition J necessarily contains a jumper from T_p. Let h be such a jumper in J from T_p. Clearly, $J' = J - h + j$ is a feasible solution and also optimal.

LEMMA 5. Let j be in some optimal partition of T. If J_1, J_2 are optimal partitions of $T - T_j$ and T_j respectively, then $J = J_1 + J_2 + j$ is an optimal partition for T.

Proof. J is a feasible partition. Let J' be any optimal partition containing j, and let J'_1, J'_2 denote the set of jumpers in J' from $T - T_j$ and T_j respectively. Obviously, $|J_1| \leq |J'_1|$ and $|J_2| \leq |J'_2|$. Hence $|J| \leq |J'|$.

Lemma 3, 4 and 5 altogether lead to optimality of algorithm 5.

The loop between line 4 and line 13 in Algorithm 5 can be implemented with the SPLIT algorithm. The jumper insertion algorithm in [7] is similar to Algorithm 5 except two major differences: (1)The algorithm in [7] is designed for antenna avoidance *planning* and is limited to spanning trees; (2) It does not use SPLIT technique and its complexity is $O(n \log n)$ instead of $O(n)$. Actually, the practical advantage of linear time SPLIT is not obvious for Steiner trees since the node degree in a Steiner routing tree is normally small. However, the node degree in a spanning tree can be up to $O(n)$ and the SPLIT algorithm may make significant difference in practice. Therefore, the SPLIT based Algorithm 5 is more useful in the scenario of [7].

E. Layer Assignment Heuristic

In this section, we will describe how to apply the probabilistic coupling capacitance model for timing optimization and how to apply the tree partitioning techniques for antenna avoidance in a layer assignment heuristic for a general multi-layer routing structure.

The layer assignment proceeds from one panel to another with the most congested panel being processed first. For each panel, layer assignment is first performed in the most congested routing region. The congestion is the ratio of the number wires vs. the number of tracks in a region. A **routing region** is a set of consecutive horizontal (or vertical) GRCs. Since congested regions are relatively difficult to deal with, they need to be processed early

with large flexibility. After the most congested region is processed, layer assignment is repeatedly conducted on routing regions adjacent to processed regions in the same panel till the entire panel is completed. This continuous expansion style is for preventing via constraint violations.

For the layer assignment in each routing region, we first decide if any wire segment in this region is critical for antenna avoidance. If a wire segment is antenna-critical, it has to be assigned to the top layer L_{top} to avoid antenna effect. Whether a wire is antenna-critical or not may depend on the wires which have already been assigned to certain layers. For the example in Figure 22, each edge in $\{(a,b),(b,c),(b,d),(d,e)\}$ has a weight of 1 with $A_{max} = 2.5$ and (a,b) has already been assigned to L_{top}. If $\{(b,c),(d,e)\}$ have been assigned to layers lower than L_{top}, (b,d) is a critical segment in T_a. However, if $\{(b,d),(d,e)\}$ are assigned to layers lower than L_{top}, (b,c) is the critical segment instead. The antenna-criticality of a wire e can be detected by a probing tree partitioning in which the wire e is forbidden to be assigned to L_{top}. If no feasible solution can be found, the wire e is antenna-critical.

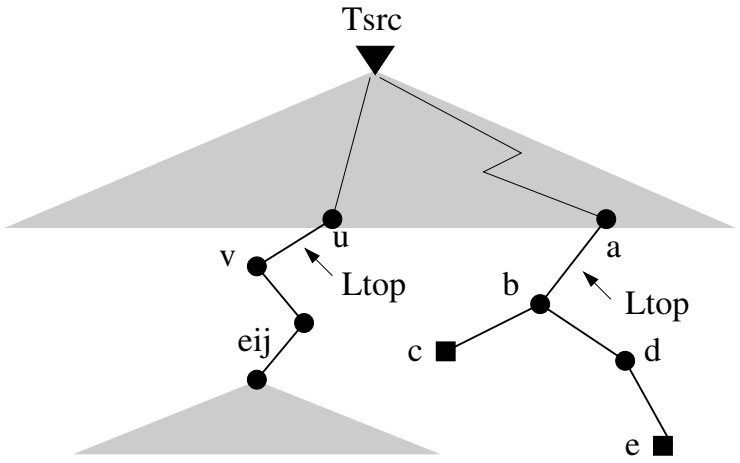


Fig. 22. Example for checking antenna-critical segments.

An initial tree partitioning is performed before the beginning of layer assignment

to identify candidate separators. Since only candidate separators may become antenna-critical, the antenna-criticality detection procedure is necessary only for candidate separators. If a none-candidate-separator edge of a Steiner tree is assigned to L_{top} or a candidate separator edge is assigned to a non-top layer, the tree partitioning needs to be run again to update the candidate separators. During the layer assignment, some edges of a Steiner tree may have already been assigned to certain layers. Thus, the probing tree partitioning or the candidate separator updating is sometimes applied on a subtree isolated by separators which have been assigned. For example, edge (u, v) has already been assigned to L_{top} in Figure 22 and the probing tree partitioning for edge e_{ij} is performed only on subtree T_v.

Once all antenna-critical segments in a routing region are identified, they are assigned to the top layer L_{top}. Next, a coupling aware timing driven layer assignment, as described in chapter II, is performed for the other wire segments in the region. These non-critical wires are sorted in non-increasing order of their timing slack. Then, these wires are partitioned into a few subsets each of which corresponds to a routing layer. The subset of wires in nets with relatively large(small) timing slacks are assigned to lower(upper) layer. Obviously, the partitioning may affect the congestion of each layer and the delay due to the probabilistic coupling capacitance. Hence, the partitioning is performed to maximize the minimum timing slack among all nets in the routing region subject to routing capacity constraint. The coupling capacitance is estimated by using our improved probabilistic model.

Vias may be induced when a wire is assigned to a layer different from its neighboring region. If there are too many vias induced and the via constraint is violated, wires in the routing region under current processing is heuristically swapped to remove the violations. After the layer assignment is completed for a routing region, the timing slacks for nets in this region are updated.

Please note that layer assignment alone cannot guarantee that the antenna problem is completely solved because of the constraints of preferred routing directions and routing

capacity. However, the antenna avoidance in layer assignment will at least reduce the workload for subsequent jumper insertion and reduce the overall via overhead. The pseudo code of our layer assignment heuristic is shown in Algorithm 6.

F. Experimental Results

Our simultaneous timing-driven and antenna-avoidance-driven layer assignment heuristic is implemented in C++. The experiment is performed on a Linux machine with a 1.4 GHz CPU and 512MB memory. The benchmark circuits are obtained from the ISPD98/IBM suites [39]. The circuit specifications are shown in Table VII. The cell placement of the circuits is generated by Dragon [46] and the global routing solutions are obtained from a rip-up and re-route router [40]. Since there is no timing information provided with the benchmark circuits, we set the timing constraint for each net as the initial delay plus a random perturbation term. The maximum antenna length A_{max} is chosen similar to the work of [7].

Experiments are performed to test our layer assignment method on both coupling aware timing performance and the effect on antenna avoidance. Since there is no published works with formulation similar to ours, the following methods are implemented and compared with our method, to check the effect on coupling aware timing performance.

- **Method 1**: A layer assignment heuristic in which only the total coupling capacitance is minimized and coupling capacitance estimation is based on a trial track assignment similar as that in [2].
- **Method 2**: Almost same as our method except that the coupling capacitance is estimated by a trial track/layer assignment method [2].
- **Method 3**: Almost same as our method except that the coupling capacitance is estimated by using the linear probability model [26].

All timing results in Table VIII are based on our coupling aware timing driven track routing over the layer assignment results. Therefore, the values in Table VIII are not probabilistic. In most cases, the timing result from method 2 is significantly better than the result from method 1. This indicates that merely optimizing total coupling capacitance is not sufficient and optimizing coupling aware timing is better in general. However, the trial track assignment employed in method 2 may lead to unreliable estimation on coupling capacitance and therefore may result in very poor solution. This can be seen for the case of ibm10. By using a probability model as in method 3, the overall timing becomes much better. The improved probability model utilized in our method can make further improvement compared with method 3.

Please note that both coupling aware timing optimization and antenna avoidance are performed simultaneously in our layer assignment. For antenna avoidance in layer assignment, there is no previous work for general multi-layer routings. Therefore, we compare our method (**LAAA**) with a similar baseline layer assignment (**LA**) without antenna avoidance. The total number of antenna violations from these two methods are shown in column 2 and 3 of Table IX. On average, our method can reduce the number of antenna violations by 76%. In order to see the overall picture, we also demonstrate the result of Jumper Insertion (**JI**) with Algorithm 5 after the layer assignment. Since jumpers bring extra vias, the number of via violations due to jumpers are shown in the right two columns of Table IX. Via violation indicates the number of vias exceeding the allowed upper limit between two adjacent GRCs. We can see that our method results in 99% less number of via violations. In other words, the antenna avoidance in layer assignment can greatly improve the feasibility of jumper insertions.

G. Conclusions

In this chapter, we propose to perform coupling aware timing optimization and antenna avoidance simultaneously in layer assignment. An improved probabilistic coupling capacitance model is suggested to facilitate the coupling aware timing optimization. The experimental result on the timing is validated with a coupling aware timing driven track router. The antenna avoidance in layer assignment is modeled and solved with tree partitioning algorithm. Experimental results show that (1) directly optimizing coupling aware timing is indeed necessary; (2) the proposed probability model correlates well with track routing results; (3) antenna avoidance in layer assignment can greatly improve the feasibility of jumper insertion.

Algorithm 5: Jumper insertion

Input : A Steiner tree.

Output: A minimum set J of jumpers which partitions the tree such that the weight of each resulting subtree is no greater than A_{max}.

```
1  J ← ∅;
2  for i ← maximum-level to 1 do
3      while there is an unprocessed node u in level i do
4          while W(T_u) > A_max and there is sink in W(T_u) do
5              if there exists a branch W((u, v) + T_v) > A_max then
6                  insert a jumper j at an exact position in (u, v) such that (u, v) is
                   partitioned into (u, j) and (j, v) and W(T_j) = A_max;
7
8              else
9                  insert a jumper j right below u on the heaviest branch (u, v) + T_v
                   of T_u.
10             end
11             J ← J ∪ j;
12             update W(T_u);
13         end
14     end
15 end
```

Algorithm 6: Layer Assignment

```
Determine L_top for each net;
Perform algorithm 1 on each net and find candidate separators;
Sort routing regions in non-increasing order of congestion;
foreach unprocessed routing region r do
    foreach candidate separator e_ij ∈ r do
        Detect antenna criticality for e_ij;
        if a segment e_ij is critical then
            assign e_ij to its L_top layer;
        end
    end
    Sort non-critical segments in non-increasing order of their minimum timing
    slacks;
    Partition non-critical segments for layer assignment such that the minimum
    timing slack among all nets in r is maximized;
    Swap segments to remove any via violation;
    if a non-candidate-separator is assigned to L_top then
        Run algorithm 1 on subtrees to update candidate separators;
    end
    if a candidate separator is assigned to layer < L_top then
        Run algorithm 1 on subtrees to update candidate separators;
    end
end
```

Table VII. Benchmark circuit specification.

Circuit	#GRC	#nets	#tracks verti panel	#tracks hori panel
ibm01	64 × 64	8.8K	10	10
ibm02	80 × 64	15.7K	22	18
ibm03	80 × 64	14.6K	15	14
ibm04	96 × 64	17.9K	19	17
ibm05	128 × 64	19.3K	34	32
ibm06	128 × 64	21.9K	18	15
ibm07	192 × 64	29.0K	23	21
ibm08	192 × 64	36.3K	22	18
ibm09	256 × 64	41.6K	18	14
ibm10	256 × 128	43.7K	23	20
ibm11	256 × 128	50.0K	13	12
ibm12	256 × 128	51.6K	18	15
ibm13	256 × 128	59.4K	13	12

Table VIII. Experimental results on coupling aware timing validated through track assignment.

	Minimum slack (ps)				CPU(sec)
Circuit	Method 1	Method 2	Method 3	Our method	Our method
ibm01	-55	-52	-41	-27	10
ibm02	-112	-144	-73	-62	50
ibm03	-66	-26	-25	-24	36
ibm04	-175	-101	-45	-39	43
ibm05	-245	-78	-57	-56	415
ibm06	-265	-76	-55	-41	81
ibm07	-843	-381	-180	-215	144
ibm08	-356	-226	-147	-147	228
ibm09	-474	-307	-109	-69	162
ibm10	-1044	-2185	-456	-173	562
ibm11	-255	-166	-118	-107	221
ibm12	-535	-355	-166	-95	331
ibm13	-265	-218	-105	-90	213
Average	-361	-332	-121	-88	192

Table IX. Experimental results on antenna violations and via violations.

	# Antenna violations		# Via violations	
Circuit	LA	LAAA	LA+JI	LAAA+JI
ibm01	1002	208	923	1
ibm02	2471	294	2169	2
ibm03	1919	420	2016	32
ibm04	1490	274	2267	3
ibm05	5645	301	6332	0
ibm06	1920	124	2662	0
ibm07	4092	144	6488	5
ibm08	7425	2089	7049	113
ibm09	7487	1647	8405	0
ibm10	11679	2818	20144	79
ibm11	7998	2452	4130	99
ibm12	11006	3444	13668	121
ibm13	7438	2913	10440	163
Average reduction		76%		99%

CHAPTER V

DICER: DISTRIBUTED AND COST-EFFECTIVE REDUNDANCY FOR VARIATION TOLERANCE

Increasingly prominent variational effects impose imminent threat to the progress of VLSI technology. This work explores redundancy, which is a well-known fault tolerance technique, for variation tolerance. It is observed that delay variability can be reduced by making redundant paths distributed or less correlated. Based on this observation, a gate splitting methodology is proposed for achieving distributed redundancy. We show how to avoid short circuit and estimate delay in dual-driver nets which are caused by gate splitting. A spin-off gate placement heuristic is developed to minimize redundancy cost. Monte Carlo simulation results on benchmark circuits show that our method can improve timing yield from 59% to 72% with only 0.3% increase on cell area and 2.2% increase on wirelength on average.

A. Introduction

When VLSI technology approaches nanoscale regime, circuit performance is increasingly affected by variational effects such as process variations [47], power supply noise [48], coupling noise [49], soft delay error [50] and temperature changes [51]. In order to have sufficient safety margins for the variations, circuit designers have to set unnecessarily aggressive timing targets which may waste both design effort and power [52]. On the other hand, current circuit designs are progressively limited by power budget [53] and unnecessary waste on power is no longer tolerable. Therefore, timing variations need to be minimized together with critical path delays during timing optimization. Recently, several statistical gate sizing methods [8, 54, 55] are proposed for process variation aware timing optimization.

In this work, we explore another direction - hardware redundancy - for improving timing tolerance to variations. Hardware redundancy[1] is a well known technology for fault tolerant systems [56]. In circuit designs, redundant transistors can help to reduce the chance of stuck-open faults [57]. Redundant vias [58] and redundant connections [59] are employed to improve manufacturing yield. Mesh [60] and redundant connections [8] have already been utilized to reduce clock skew variability. In contrast, redundancy has rarely been mentioned for signal path timing variation tolerance until the recent work of [9]. In [9], Triple Modular Redundancy(TMR), which is a classic fault tolerance technique, is applied in re-synthesis for variation tolerance [9]. However, this work is limited to pre-layout designs and does not consider interconnect delay which is a widely-recognized dominating factor for circuit timing.

In general, redundancy is a relatively expensive technique. For example, TMR requires two replicates of the original module plus a voting circuit and therefore at least triples the hardware cost and power consumption. Although redundancy is suitable for fault tolerance, it is often an unaffordable overkill for variation tolerance. In [9], substantial effort is made to modify the TMR technique so that the hardware replication is minimized. Nevertheless, the re-synthesis method in [9] still causes 20% increase on cell area for 10% improvement on delay variation tolerance.

In fault tolerance driven redundancy designs, common-mode failure (CMF) [56, 61] is an important issue. In a redundant system, a common-mode failure occurs when a single failure affects more than one identical modules at the same time [61]. Obviously, a redundant system is more likely to fail when a CMF occurs. In order to protect redundant systems against CMFs, people advocate design diversity [61] which implies different implementa-

[1]Other typical redundancy techniques include information redundancy, timing redundancy and software redundancy. We refer hardware redundancy simply as redundancy in this work.

tions of the replicated modules. Another example is differential signaling [62] which is employed to cope with common-mode noise. In the world of variations, the corresponding issue is correlation among variations. An occurrence of multiple perfectly correlated variations is an analogue of a CMF. Partially correlated variations, which are typical cases of intra-die variations [63, 64], can be treated as partial CMF. Corresponding to design diversity, less correlation among the redundant parts of a circuit implies that the redundancy is more robust to variations. This observation inspires us to explore distributed redundancy for delay variability reduction.

In this work, we propose a new variation tolerance driven redundancy methodology through gate splitting. Our major contributions are listed as follows.

- We reveal the relationship between spatial correlation and delay variability through an Elmore delay based analysis and SPICE based Monte Carlo simulations. The result shows that less correlation in redundancy may lead to less delay variability.

- According to the above observation, we propose a distributed redundancy technique based on gate splitting.

- We show how to avoid short circuit in dual-driver nets which are caused by gate splitting. We also developed an Elmore-like delay estimation method for dual driver nets.

- A spin-off gate placement heuristic is developed to minimize the cost of the distributed redundancy. In general, the cost of our method is significantly less than that of [9].

- The proposed redundancy methodology is designed for post-placement optimization, therefore, the corresponding timing results are more meaningful than that of [9].

Please note that our approach is radically different from traditional gate duplication works [65–67] even though they have some superficial resemblance. The spin-off gate and the original gate drive the same set of nets in our approach while in traditional gate duplication works [65–67] the duplicated gate and the original gate drive separated nets and no signal redundancy exists. The proposed technique can be applied together with statistical gate sizing [8, 54, 55] to further improve timing yield which is the probability of satisfying timing constraints considering variations.

Monte Carlo simulations are performed on benchmark circuits after placement legalization by commercial tool. Spatially correlated process variations and power supply variations are considered in the simulations. The results show that our method can improve timing yield from 59% to 72% with only 0.3% increase on cell area and 2.2% increase on wirelength on average.

B. Motivation Examples

In this section, we will use a few examples to demonstrate that delay variability depends on the correlation between redundant paths. These examples, which include both Elmore delay based analysis and SPICE based Monte Carlo simulations, motivate us to pursue distributed redundancy for delay variability reduction.

1. Elmore Delay Based Analysis

The first example is simply a 1-sink net as shown in Figure 23(a). The wire is split into two halves which are routed separately as in Figure 23(b). Please note that the wire area in (b) is the same as in (a) as the wire width in (b) is a half of that in (a). We will show that the variability of the delay from the driver to the sink is less in (b) than in (a) through an Elmore delay based analysis.

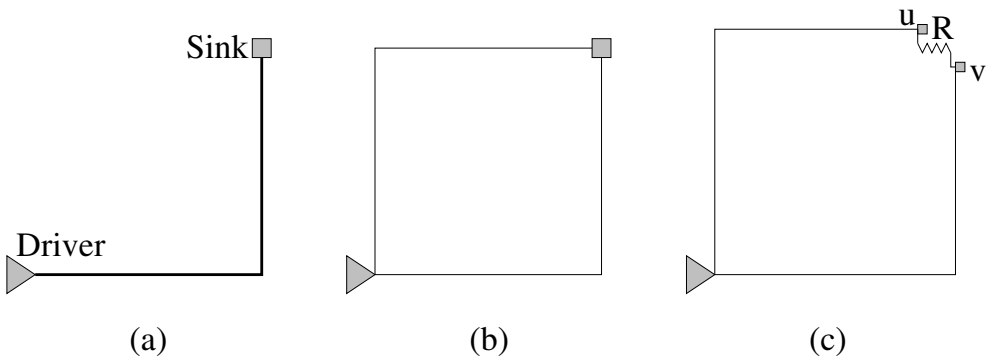


Fig. 23. A simple example of redundancy.

Obviously, Figure 23(b) is a non-tree routing and the delay can be obtained by the tree partitioning and link insertion method [68]. More specifically, a non-tree can be partitioned into a tree plus a set of link edges. The Elmore delay in the tree can be calculated easily. Then, the non-tree delay can be obtained by successively adding the link edges and updating the delay. The non-tree in Figure 23(b) can be partitioned into two parts connected by a virtual link resistor $R = 0$ as in Figure 23(c). In this partitioning, the sink capacitance is split into two equal parts u and v which are at the same location even though they are drawn separately in Figure 23(c). If the Elmore delays without link R are t_u and t_v, respectively, the delays after inserting link $R = 0$ become [68]:

$$\hat{t}_u = t_u - \frac{t_u - t_v}{r_u - r_v} r_u \tag{5.1}$$

$$\hat{t}_v = t_v - \frac{t_u - t_v}{r_u - r_v} r_v \tag{5.2}$$

where r_u and r_v are equal to the Elmore delay at u and v, respectively, when node capacitance $C_u = 1, C_v = -1$ and the other node capacitances are zero [68]. By defining $\alpha = \frac{r_u}{r_u - r_v}$, the above equations can be rewritten as:

$$\hat{t} = \hat{t}_u = \hat{t}_v = (1 - \alpha) t_u + \alpha t_v \tag{5.3}$$

It is not difficult to see that $r_v < 0$ and therefore $0 < \alpha < 1$.

Observation 1: ***In an RC circuit, a short (link resistor of*** $R = 0$***) between two redundant paths averages the delays of the two paths.***

Since Equation (5.1) and (5.2) hold for any arbitrary non-tree, *observation 1* is true for general RC circuits although it is derived based on the example of Figure 23 for the simplicity of illustration.

If variations are considered, delay t_u and delay t_v become two random variables with standard deviation σ_u and σ_v, respectively. Let the covariance between t_u and t_v be $\sigma_{u,v}$. Since the delay $\hat{t}$ is a linear combination of t_u and t_v, the variance of delay $\hat{t}$ is given by [69]:

$$\hat{\sigma}^2 = (1-\alpha)^2\sigma_u^2 + \alpha^2\sigma_v^2 + 2\alpha(1-\alpha)\sigma_{u,v} \tag{5.4}$$

If α is approximated as a constant, we can reach the following conclusion which is an important basis of our work.

Observation 2: ***The delay variance of shorted redundant paths in an RC circuit increases(decreases) as the covariance among the redundant paths increases(decreases).***

The rationale behind *observation 2* or Equation (5.4) is that less correlated delay variations along redundant paths have more chances to cancel out each other in the delay averaging implied by *observation 1*. Please note that *Observation 2* does not depend on any specific variation model and is applicable for many different kinds of variations such as process variation, power supply noise and temperature change.

Since the correlation between two redundant path delays normally increases when they are moved close to each other [64], the layout in Figure 23(a) can be treated as aggregated redundant paths with perfect correlation. Thus, $\hat{\sigma}^2$ has the maximal value in Figure 23(a) when covariance $\sigma_{u,v}$ reaches its maximal value of $\sigma_u\sigma_v$. On the other hand, separating redundant paths apart as in Figure 23(b) can reduce spatial correlation between the two path delays and thereby reduce the variability of the averaged delay $\hat{t}$. This observation implies

that gate/wire sizing [8, 55] improves timing yield more through reducing nominal delays than through reducing delay variability. Therefore, making redundancy to be distributed may reduce delay variability further than performing gate/wire sizing alone. This is very similar as the diversified redundancy design in fault tolerant systems [61].

2. SPICE Based Monte Carlo Simulations

Since the Elmore delay model is sometimes inaccurate despite its high fidelity [38], we perform SPICE based Monte Carlo simulations to verify *observation 2*. To be more general, buffers are included in the interconnect. Buffer gate length variation and wire width variation are considered and assumed to follow Gaussian distribution [64]. A layout area is tessellated into an array of tiles indicated by the dashed grid in Figure 24. The spatial correlations among the variations are handled by applying the Principle Component Analysis (PCA) method on this grid as in [64]. The variations in the same tile are approximately treated with perfect spatial correlation. Two variations at two tiles far apart have relatively small spatial correlation.

Two cases are tested. In the first one, a single wire path with one buffer in Figure 24(a) is split into two wire paths and two buffers as in Figure 24(b). The buffer and wire size in Figure 24(b) is a half of that in Figure 24(a). Therefore, there is no buffer or wire area change due to the splitting. The *distance* between the two redundant paths in Figure 24(b) is specified in term of the number of tiles between the two buffers in rectilinear space. For example, the distance between the two paths in Figure 24(b) is 2. We vary the distance and observe the impact to delay variations at the sink. The other case is very similar except that two buffers are split into four buffers as shown in Figure 24(c) and (d). The distance in Figure 24(d) is 6.

The Monte Carlo simulation results are depicted in Figure 25. The horizontal axis indicates the distance between two redundant paths. The zero distance results are obtained

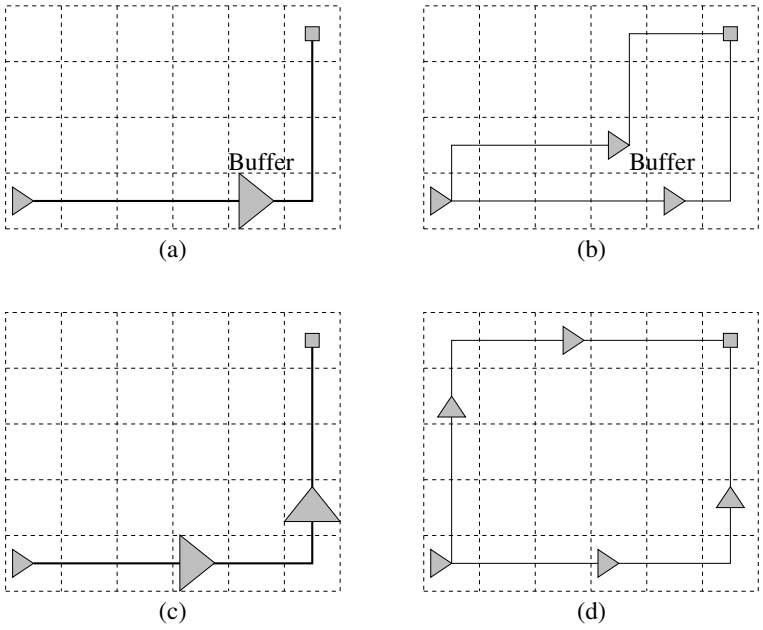
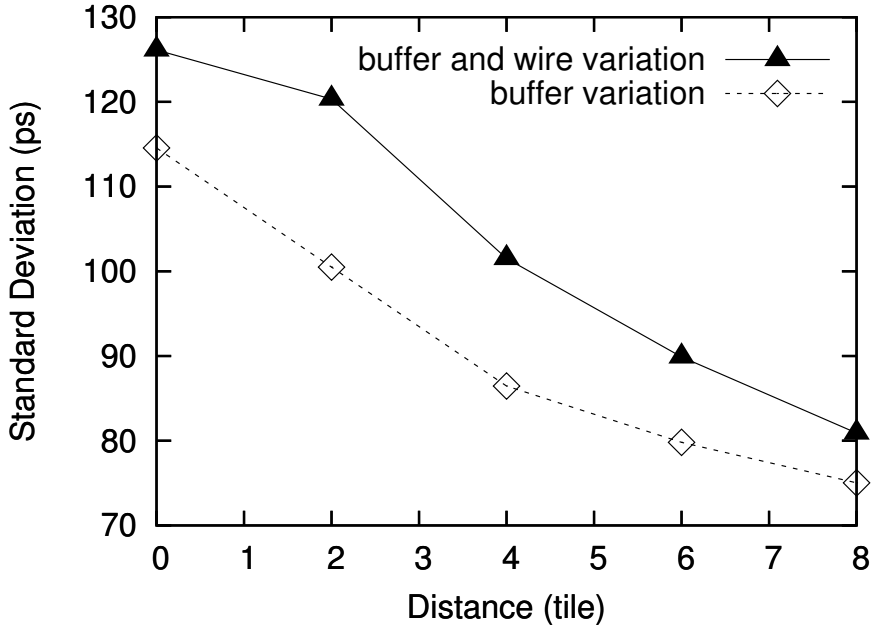
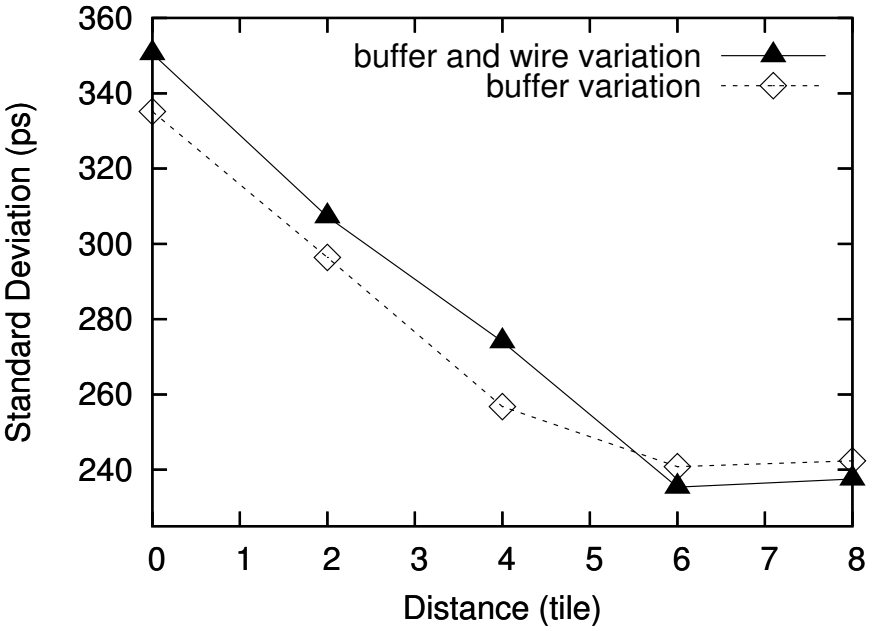


Fig. 24. Layout configurations for Monte Carlo simulations.

before the splitting. As the distance varies, the change on the mean value of delays is negligible. In other words, the splitting does not affect the nominal delay. Therefore, only the delay variations in term of standard deviation are shown in Figure 25. The results of Figure 25(a) are from the case of two split buffers in Figure 24(b). The lower curve is from the result considering only buffer gate length variations while both the gate length variation and the wire width variation are included for the upper curve. The results of the four buffer case in Figure 24(d) are shown in Figure 25(b). All these curves show a common trend that the delay variation reduces when the redundant paths are far apart and the spatial correlation is reduced. These are supporting evidence for *Observation 2* based on accurate gate and wire models.



(a) 2 buffers



(b) 4 buffers

Fig. 25. Standard deviations of delay vs. distance between split buffers.

C. Gate Splitting Methodology

Based on *observation 2*, we propose a gate splitting technique for reducing signal delay variability. The basic idea of gate splitting is illustrated in Figure 26. A gate G_0 on the timing critical path is split into two halves and the **spin-off gate** G_f is placed some distance away from G_0 with separated wire connections indicated by the dashed lines.

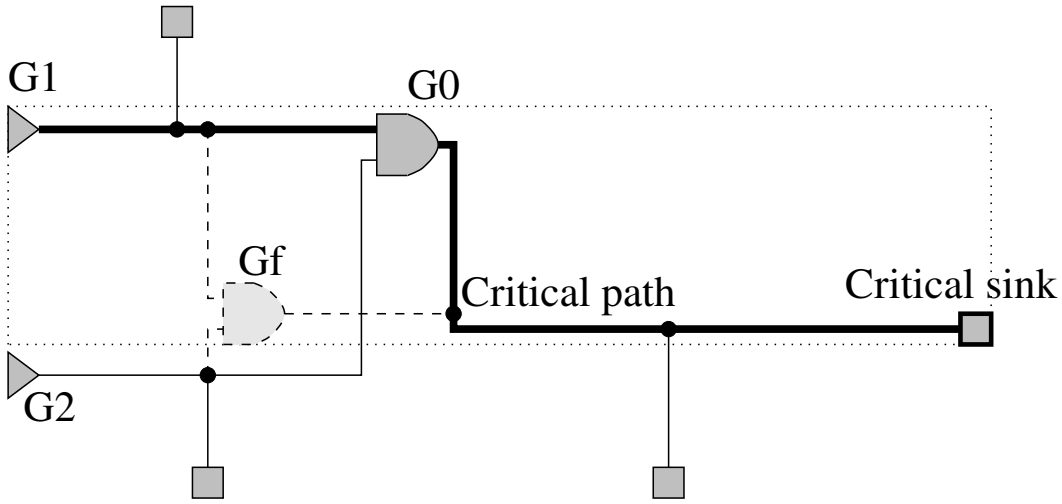


Fig. 26. An example of gate splitting. G_f is the spin-off gate and the dotted rectangle is the feasible region.

If gate G_0 originally has size w_0, its size after splitting and the size of spin-off gate G_f are $w_0/2$. If there is no gate of size $w_0/2$ in cell library, they are rounded to the closest available size in library. Alternatively, we can perform *gate duplication* where G_0 retains its original size and the size of gate G_f also equals w_0. However, the gate duplication here is quite different from traditional gate duplication works [65–67] in which the duplicated gates drive separated nets and no signal redundancy exists. Therefore, we use the term of *gate splitting* to avoid confusion. In fact, the sizes of G_0 and G_f can be tuned in a post processing for either gate splitting or gate duplication.

Another issue worth discussion is wiring cost. Even though it seems that layout in Figure 24(b) has the same wire area as that in Figure 24(a), the former consumes more wire pitch than the later. In modern IC designs, wire resource is more scarce than gate resource. In addition, separated wires may encounter more crosstalk noise. Therefore, we emphasize more on gate splitting than wire splitting and try to minimize extra wires incurred by gate splitting.

The gate splitting can be performed after either cell placement or global routing. Without loss of generality, we assume that the input for gate splitting includes a cell placement solution and Steiner trees for each signal net. We propose a gate splitting methodology which is composed by the following four phases.

1. **Gate selection**. In this phase, the gates to be split are identified. There are two options for the selection. One approach is to select gates based on their timing criticality and variability. For example, gates along paths with large disutility function value [8] can be selected. The other approach is to select gates according to a gate sizing solution [8, 55, 70, 71] as gate sizing algorithms are mostly decided by timing criticality [8,55,70,71] and/or timing variability [8,55]. Usually the gates along timing critical paths are sized up. The gates which are sized up in sizing can be regarded as aggregated redundancy and can be split to obtain distributed redundancy.

2. **Spin-off gate placement**. The spin-off gates are placed such that they are far apart from the original gates, the monotonicity of each critical path is not degraded, short circuit in dual-driver nets is avoided(Section 1) and the wiring cost is minimized. This phase will be discussed in details in Section E.

3. **ECO placement and placement legalization**. The spin-off gate placement may cause cell overlaps which need to be removed through ECO(Engineering Change Or-

der) placement or placement legalization. There are existing ECO placer and placement legalizer tools which can be adopted directly.

4. **Spin-off gate connection**. The spin-off gates need to be connected to their fanin and fanout nets through wires. For each fanin net, an extra sink due to the spin-off gate is added and a new Steiner tree [72] can be constructed to accommodate this change. For the fanout net, the spin-off gate make it become a dual-driver net. Since there is no Steiner tree algorithm for dual-driver nets, to the best of our knowledge, we just connect the spin-off gate with the original fanout Steiner tree through the shortest feasible routes considering wiring blockages and congestions. If the gate splitting is performed after global routing, then ECO routing needs to be conducted.

D. Handling Dual-driver Nets

Due to the gate splitting, a net may be driven by two drivers as in Figure 26. This phenomenon raises two issues which do not exist in conventional single driver routings. One is the risk of short circuit between the two drivers. The other is the fast estimation of signal delays in dual-driver nets.

1. Short Circuit Avoidance

If the signal arrival times at the two drivers are different, there is a risk of short circuit between power supply and ground through the two drivers. For the example in Figure 26, gate G_0 and G_f drive the same net. If the output of G_0 switches to high while the output of G_f is still at low, there is a direct short circuit path from power supply to ground through the output of G_0 and G_f. However, there is time delay for a signal to be propagated from one driver to the other driver. If this delay is greater than the difference of signal arrival time at the drivers, there is no sufficient time to establish the short circuit current. According

to this observation, we apply the following design criterion for short circuit avoidance in dual-driver nets.

Let the upper bound of difference between signal arrival time at gate G_0 and G_f be $\Delta_{0f,max}$ considering variations. If the signal delay from a fanin gate G_i to G_0 and G_f are $t_{i,0}$ and $t_{i,f}$, respectively, this upper bound is:

$$\Delta_{0f,max} = \max_{\forall G_i \in \mathcal{G}_{in}} |t_{i,0} - t_{i,f}| \tag{5.5}$$

where $\mathcal{G}_{in}$ is the set of fanin gates for G_0 and G_f.

In the fanout Steiner tree of G_0 and G_f, the lower bound $\tau_{0\rightsquigarrow f}(\tau_{f\rightsquigarrow 0})$ of delay from $G_0(G_f)$ to $G_f(G_0)$ can be obtained through the method of [73]:

$$\tau_{0\rightsquigarrow f} = \frac{\sum_{e_{uv}\in path(G_0\rightsquigarrow G_f)} R_{uv}^2 C_v}{\sum_{e_{uv}\in path(G_0\rightsquigarrow G_f)} R_{uv}} \tag{5.6}$$

where e_{uv} indicates an edge between node u and v, R_{uv} is the edge resistance and C_v is the total capacitance downstream of node v. Then the criterion for avoiding short circuit between G_0 and G_f is:

$$\min(\tau_{0\rightsquigarrow f}, \tau_{f\rightsquigarrow 0}) > \beta\Delta_{0f,max} \tag{5.7}$$

where $\beta > 1$ is a constant parameter for extra safety margin.

2. Dual-driver Delay Estimation

Even though signal delay in a dual-driver net can be computed by SPICE or model order reduction methods such as AWE [74], an Elmore-like method is necessary for fast delay estimation during optimizations. We solve this problem by transforming a dual-driver net to an equivalent single driver net.

If there are two drivers for a net as in Figure 27(a), the term of signal delay is not well defined as the signal departure time from the two drivers may be different. More

precisely speaking, we need to find the signal arrival time t_i to a node i in the net given the signal departure time t_1 and t_2 from node 1 and node 2 in Figure 27(a). Without loss of generality, it can be assumed that $t_1 \geq t_2$, i.e., $t_1 = t_2 + \Delta$ and $\Delta \geq 0$. Now let us consider inserting a virtual resistance R_v between the signal source s_1 and node 1. If the signal delay across the virtual resistance equals Δ and the signal departure time from s_1 is t_2, the signal departure time t_1 at node 1 is not changed after this virtual resistance insertion. If the signal departure time at both s_1 and s_2 are t_2 after this insertion, we can merge s_1 with s_2 into a single source as shown in Figure 27(b). Please note that this merging does not affect the gate driving capability as the driving capability is decided only by R_{d1} and R_{d2} in this model. After this merging, the dual-driver net in Figure 27(a) is converted to a single driver net.

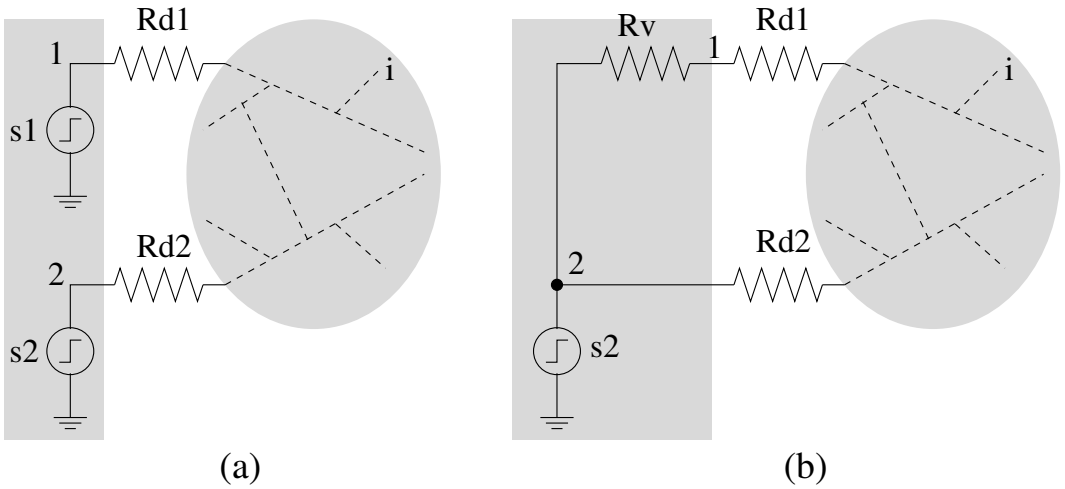


Fig. 27. The dual driver net in (a) can be converted to the single driver net in (b) when signal departure time t_1 at node 1 is no less than the signal departure time t_2 at node 2.

In the above transformation, we also need to find the value of the virtual resistance R_v such that the delay across it is equal to Δ. Since the net in Figure 27(b) has a single driver, the signal delay t_1 at node 1 is well defined by letting the signal departure time $t_2 = 0$.

Evidently, t_1 is a function $t_1(R_v)$ depending on R_v and the value of R_v can be obtained by solving the equation $t_1(R_v) = \Delta$.

Since the non-tree here is special case which is $2-$edge connected [59], an analytical expression for function $t_1(R_v)$ can be obtained by the tree partitioning and link insertion method [68]. A graph is $k-$edge connected if it cannot be separated by removing less than k edges. Hence, for a $2-$edge connected non-tree, we can always find a single node, which is called joint node, such that the non-tree can be transformed into a tree by tearing the joint node into two separated nodes u and w. In fact, Figure 23(c) is an example of such node tearing if the link resistor R is removed. We can first obtain the delay expression $t_1(R_v)$ in the resulting tree and then update it by merging u and w or inserting a zero resistance between them.

The node tearing separates the non-tree in Figure 27(b) into two subtrees T_1, which is driven through R_v and R_{d1}, and T_2 which is driven through R_{d2}. If node $u \in T_1$ and node $w \in T_2$, the delay at node 1 equals $R_v C_1$ where C_1 is the total downstream capacitance at node 1 in subtree T_1. After node u and node w are merged back, the delay at node 1 has to satisfy [68]:

$$t_1(R_v) = R_v C_1 - \frac{t_u - t_w}{r_u - r_w} R_v = \Delta \tag{5.8}$$

where t_u and t_w are delays at node u and w before the merging. The values of r_u and r_w are equal to the Elmore delay at u and w, respectively, when node capacitance $C_u = 1, C_w = -1$ and the other node capacitance are zero [68].

Since $u \in T_1$, t_u can be decomposed as $t_u = R_v C_1 + t_{1,u}$ where $t_{1,u}$ is the delay from node 1 to u before merging. Similarly, r_u can be decomposed as $r_u = R_v + R_{1,u}$ where $R_{1,u}$ is the total path resistance from node 1 to node u. The value of $-r_w$ is equal to the total path resistance $R_{2,w}$ from node 2 to node w before the merging. Then the value of R_v

can be derived from Equation (5.8) as:

$$R_v = \frac{(R_{1,u} + R_{2,w})\Delta}{(R_{1,u} + R_{2,w})C_1 + t_w - t_{1,u} - \Delta} \tag{5.9}$$

E. Spin-off Gate Placement

1. Problem Formulation

The spin-off gates (like G_f in Figure 26) need to be placed such that they are far apart from the original gates, the monotonicity of each critical path is not degraded, short circuit in dual-driver nets is avoided and the wiring cost is minimized. The short circuit avoidance constraint has been introduced in Section 1. The other objectives and constraints will be described as follows.

Monotonicity is one of the most desired properties for timing critical paths. For a path $a \rightsquigarrow b \rightsquigarrow c$ which starts from node a and then reaches node c through node b, it is monotone if $length(a \rightsquigarrow b) + length(b \rightsquigarrow c) = length(a \rightsquigarrow c)$. If node b is outside of the minimum bounding box containing a and c, path $a \rightsquigarrow b \rightsquigarrow c$ is not monotone. Obviously, a monotone path can achieve better timing than a non-monotone path. This fact motivates gate duplication works [66, 67] to straighten non-monotone critical paths for timing improvement.

In our work, we wish to reduce delay variability without hurting the nominal delay. Therefore, we need to ensure that the spin-off gate placement does not degrade monotonicity of each critical path. This can be achieved by restricting spin-off gates within the feasible region [67] of the critical paths it is associated with. The concept of **feasible region** is suggested in [67] and we restate it here for the completeness of the description. Consider splitting a gate G_0 with fanin gates $\mathcal{G}_{in} = \{G_1, G_2, ..., G_m\}$. Let $\mathcal{G}_c \subseteq \mathcal{G}_{in}$ be the set of fanin gates on timing critical paths and $v_{0,c}$ be the most critical sink in the fanout net

of G_0. The feasible region $R_f(G_j, G_0, v_{0,c})$ for a fanin gate $G_j \in \mathcal{G}_c$ is simply the minimum bounding box containing G_j, G_0 and $v_{0,c}$. The feasible region $R_f(\mathcal{G}_c, G_0, v_{0,c})$ for the entire set of critical fanin gates is the overlap of the feasible regions for all gates in $\mathcal{G}_c$. As an example shown in Figure 26, the dotted bounding box is the feasible region and G_1 is the only fanin gate of G_f on timing critical paths.

A major objective of spin-off gate placement is to separate it far apart from the original gate so that the spatial correlation between them as well as the delay variability can be reduced. Thus, the distance between the original gate G_0 and the spin-off gate G_f is $d(G_0, G_f)$ needs to be maximized in the spin-off gate placement.

Last but not the least, the wiring cost needs to be minimized. Since expensive cost is a major hurdle that prevents wide application of redundancy for variation tolerance, cost minimization is a major goal in our method. As gate and wire size can be tuned in a post processing, we focus on wirelength minimization in the spin-off gate placement.

The wiring cost is determined by the wire connection between the spin-off gate G_f and T_0, which is the fanout Steiner tree driven by G_0, and the set of fanin Steiner trees $\mathcal{T} = \{T_1, T_2, ..., T_m\}$ for gate G_0. In order to quantify the wiring cost, we need to define the distance $d(G_f, T_i)$ between gate G_f and a Steiner tree $T_i \in \{T_0, T_1, ..., T_m\}$. This is based on defining the distance $d(G_f, e_{uv})$ between G_f and an edge $e_{uv} \in T_i$ with two end nodes u and v. The distance $d(G_f, e_{uv})$ is equal to the distance between the location (x_f, y_f) of G_f and the closest connection point (x_c, y_c) which is given by:

$$\begin{aligned} x_c &= \text{median}(x_f, x_u, x_v) \\ y_c &= \text{median}(y_f, y_u, y_v) \end{aligned}$$

where (x_u, y_u) and (x_v, y_v) are the coordinates of node $u \in T_i$ and node $v \in T_i$, respec-

tively. Thus, the distance between G_f and edge e_{uv} is:

$$d(G_f, e_{uv}) = \sqrt{(x_f - x_c)^2 + (y_f - y_c)^2} \tag{5.10}$$

Then, the distance between gate G_f and Steiner tree T_i is defined as

$$d(G_f, T_i) = \min_{\forall e_{uv} \in T_i} d(G_f, e_{uv}) \tag{5.11}$$

The above objectives and constraints can be summarized as a formulation of the spin-off gate placement problem as:

$$\mathcal{SGP}:$$

$$\text{Minimize} \quad \sum_{i=0}^{m} \omega_i d^2(G_f, T_i) - \lambda d^2(G_f, G_0) \tag{5.12}$$

$$\text{Subject to} \quad G_f \in R_f(\mathcal{G}_c, G_0, v_{0,c}) \tag{5.13}$$

$$\min(\tau_{0 \rightsquigarrow f}, \tau_{f \rightsquigarrow 0}) > \beta \Delta_{0f,max} \tag{5.14}$$

where ω_i and λ are non-negative weighting factors. The first term in the objective function (5.12) is for wiring cost minimization and the second term is to separate the spin-off gate G_f apart from the original gate G_0. The reason that we employ the quadratic objective function in (5.12) is the same as conventional quadratic placement [75]. The weighting factor ω_i is decided based on the timing criticality. We adopt the timing criticality definition in [76]. The maximum combinational logic path delay from primary input (or flip-flop) to primary output (or flip-flop) is denoted as D_{max} and the timing slack at gate G_i is represented as $slack(G_i)$. Then, the timing criticality or the net weight can be obtained as [76]:

$$\omega_i = (1 - \frac{slack(G_i)}{D_{max}})^{\gamma} \tag{5.15}$$

where γ is a constant.

The constraint (5.13) is to restrict the spin-off gate within the feasible region so that

there is degradation on the monotonicity of critical paths. The constraint (5.14) is for short circuit avoidance in dual-driver nets.

2. Algorithm

The spin-off gate placement problem $\mathcal{SGP}$ formulated above is difficult to be solved directly because of two reasons. The constraint (5.14) is troublesome. Depending on the location (x_f, y_f) of the spin-off gate, the connection point on the fanout Steiner tree T_0 may change from one edge to another edge. Thus, $\tau_{0 \rightsquigarrow f}$ and $\tau_{f \rightsquigarrow 0}$ are not continuous functions with respect to the location (x_f, y_f). Similarly, the distance function $d(G_f, T_i)$ in the objective function (5.12) is not well-behaved as the connection point $v_{i,c}$ varies depending on location (x_f, y_f).

We employ two common and effective techniques - *relaxation* and *restriction* - to handle the difficulties on solving $\mathcal{SGP}$. The relaxation is applied to constraint (5.14), i.e., constraint (5.14) is temporarily dropped and any violation on it will be fixed later. The restriction technique is applied to $d(G_f, T_i)$ in the objective function (5.12). More specifically, we temporarily restrict the connection point for tree T_i at a node $v_{i,c} \in T_i$. Then, we can first solve the following relaxed and restricted problem. The method of fixing violations on (5.14) and selection of connection nodes $v_{i,c}$ will be described later.

$\mathcal{SGP}_{\mathcal{RR}}$:

$$\text{Minimize} \quad \sum_{i=0}^{m} \omega_i d^2(G_f, v_{i,c}) - \lambda d^2(G_f, G_0) \tag{5.16}$$

$$\text{Subject to} \quad G_f \in R_f(\mathcal{G}_c, G_0, v_{0,c}) \tag{5.17}$$

Problem $\mathcal{SGP}_{\mathcal{RR}}$ is a constrained quadratic programming problem and can be solved along x and y directions separately as in quadratic placement [75]. The feasible region

$R_f(\mathcal{G}_c, G_0, v_{0,c})$ is normally a rectangle and can be represented by its corner coordinates $(x_{min}, y_{min}) - (x_{max}, y_{max})$. The location of gate G_0 is at (x_0, y_0). The location of a connection node $v_{i,c}$ is represented by $(x_{i,c}, y_{i,c})$. Then, the subproblem along x direction becomes:

$$\begin{array}{ll} \text{Minimize} & \phi(x_f) = \sum_{i=0}^{m} \omega_i (x_f - x_{i,c})^2 - \lambda (x_f - x_0)^2 \\ \text{Subject to} & x_{min} \leq x_f \leq x_{max} \end{array}$$

If $\tilde{x}_f$ satisfies $\frac{d\phi(x_f)}{dx_f}|_{x_f=\tilde{x}_f} = 0$ and $\hat{x}_f = \min(x_{max}, \max(x_{min}, \tilde{x}_f))$, the above problem has an optimal solution at:

$$x_f^* = \begin{cases} x_{min} & : \quad \frac{d^2\phi(x_f)}{dx_f^2} \leq 0, \phi(x_{min}) \leq \phi(x_{max}) \\ x_{max} & : \quad \frac{d^2\phi(x_f)}{dx_f^2} \leq 0, \phi(x_{min}) > \phi(x_{max}) \\ \hat{x}_f & : \quad \text{otherwise} \end{cases} \tag{5.18}$$

The optimal solution along the y direction can be obtained similarly.

The overall algorithm for solving the spin-off gate placement problem $\mathcal{SGP}$ is summarized in Figure 28. Initially, we approximate the connection points $v_{i,c}$ by the centroid of each tree T_i as indicated by step 3 of Figure 28. If a Steiner tree T_i has nodes $V_i = \{v_{i,0}, v_{i,1}, ...\}$ including the source, sinks and Steiner nodes, then the x coordinate of its centroid is simply the average value of x coordinates of all nodes in V_i. The y coordinate of the centroid can be obtained similarly. After locations of the connection points are found, an optimal solution of $\mathcal{SGP}_{\mathcal{RR}}$ can be obtained in step 4 as described above. After the initial solution is found in step 4, the solution is refined iteratively starting from step 5. At the beginning of each iteration which is step 6, the connection points $v_{i,c}$ are updated according to the (x_f, y_f) obtained previously. Then, the solution can be refined by running $\mathcal{SGP}_{\mathcal{RR}}$ at step 7. The constraint (5.14) is checked at step 8. If the solution satisfies constraint (5.14), the algorithm is finished. Otherwise, the value of λ is increased

Procedure: $SpinoffGatePlacement(G_0, T_0, \mathcal{T})$

Input: Gate G_0 to be split

Fanout Steiner tree T_0, critical sink $v_{0,c} \in T_0$

A set of fanin Steiner trees $\mathcal{T} = \{T_1, T_2, ...T_m\}$

Output: Location (x_f, y_f) of spin-off gate G_f

1. Find feasible region $R_f(\mathcal{G}_c, G_0, v_{0,c})$
2. Initialize λ
3. $v_{i,c} \leftarrow$ centroid of T_i, $i = 0, 1, ..., m$
4. $(x_f, y_f) \leftarrow$ solve $\mathcal{SGP}_{\mathcal{RR}}$
5. While (true)
6. $v_{i,c} \leftarrow$ closest connection on T_i to (x_f, y_f)

   $i = 0, 1, ..., m$
7. $(x_f, y_f) \leftarrow$ solve $\mathcal{SGP}_{\mathcal{RR}}$
8. If inequality (5.14) is satisfied

   Return (x_f, y_f)
9. Else if (x_f, y_f) is at corner of $R_f(\mathcal{G}_c, G_0, v_{0,c})$

   Return failure
10. Else $\lambda = \lambda + \sigma$

Fig. 28. Main algorithm of spin-off gate placement.

by a small positive value σ at step 10 and next iteration is started. Increasing the value of λ may push the spin-off gate G_f farther away from gate G_0 and may improve the chance that constraint (5.14) is satisfied. If gate G_f is pushed to a corner of the feasible region and constraint (5.14) is not satisfied yet, it is quite likely that there is no feasible solution and the iterations terminate at step 9. It is not hard to see that this iterative procedure is similar as Lagrangian relaxation and λ plays a role of Lagrangian multiplier.

F. Experimental Results

Our methodology and algorithm are tested on the ISCAS85 benchmark circuits with 180nm technology. The circuit specifications are listed in Table X. All timing related parameters of the standard cells are extracted from HSPICE. The nominal wire resistance and capacitance is $0.076\Omega/\mu m$ and $0.118fF/\mu m$, respectively. The experiments are performed on a Linux platform with a 1.3 GHz Intel processor.

The initial placement is obtained from Cadence Silicon Ensemble. Initial Steiner trees are constructed by using the C-Tree [72] software downloaded from GSRC Bookshelf(http://dropzone.tamu.edu/~cnsze/GSRC/ctree.html). Phase 1 gate selection and phase 2 spin-off gate placement algorithms are implemented in C++ by ourselves. The ECO placement of phase 3 is also obtained from Cadence Silicon Ensemble. In phase 4, the Steiner trees for the fanin nets of spin-off gates are constructed by C-Tree. Spin-off gates are connected to their existing fanout Steiner tree through the shortest feasible route by our own implementation in C++.

The final timing results are estimated through Monte Carlo simulations. Variations on gate length, power supply level and wire width are considered. These variations are assumed to follow Gaussian distribution with standard deviations equal to 10% of their nominal values. Spatial correlations among the variations are handled by the PCA(Principle

Table X. Benchmark circuit specification.

Circuit	# cells	# nets
c432	160	196
c880	383	443
c1355	546	587
c1908	880	913
c2670	1269	1502
c3540	1669	1719
c5315	2307	2485
c6288	2416	2448
c7552	3513	3720

Component Analysis) method as in [64]. In applying the PCA method, the die area of each circuit is tessellated into an 64×64 array of tiles.

Table XI. Experimental results.

	Nominal delay (ps)		Std deviation (ps)		Timing yield			Cell area (μm^2)		Wirelength (μm)		$d(G_0, G_f)$	
Circuit	Initial	Split	Initial	Split	Constraint	Initial	Split	Initial	Split	Initial	Split	ave(tile)	#split
c432	6094	5633	327	239	6100	28.8%	94.8%	7154	7222	5897	6570	8.4	16
c880	4250	4255	214	154	4440	66.8%	77.0%	16203	16291	15458	15872	3.6	23
c1355	4044	4086	335	288	4550	60.4%	62.8%	21831	21870	18197	18634	3.6	24
c1908	5412	5354	280	295	5800	82.6%	90.3%	40280	40415	29250	29829	3.0	40
c2670	5135	5131	297	249	5350	48.0%	60.6%	52215	52391	76117	76659	2.0	32
c3540	7491	7450	597	560	8100	57.2%	65.0%	68915	69109	64800	65441	1.4	41
c5315	6789	6751	412	392	7000	35.0%	50.6%	99601	99835	133848	134640	1.4	48
c6288	18912	18913	1528	1486	20500	69.6%	70.8%	96113	96132	60319	60710	0.7	62
c7552	6595	6519	339	318	7000	67.8%	74.4%	143136	143311	161401	162182	2.0	42
Average	Decrease 1%		Decrease 11%			59%	72%	Increase 0.3%		Increase 2.2%			

To the best of our knowledge, there is no similar approach in previous works. Hence, comparisons are made between the initial results and the results after our 4-phase gate splitting method. The comparison results are shown in Table XI. The data of the maximal nominal path delay are in column 2 and 3. Same as the observation in Section 2 and our expectation, the influence from our method to the nominal delay is usually small and is only an average of 1% reduction.

The data in column 4 and 5 of Table XI are the standard deviations of the maximal path delay variation. Except c1908, our method always reduces the standard deviations.

The magnitude of the reduction is often large when the distance $d(G_0, G_f)$ between the original gate G_0 and the spin-off gate G_f is large. The average distances in term of the number of tiles are listed in column 13. For c432, the standard deviation is reduced by 27% from the gate splitting as the average distance is greater than 8 tiles. In contrast, the standard deviation reduction for c6288 is only 3% since its average distance is less than 1. Normally, the distance $d(G_0, G_f)$ is constrained by the size and aspect ratio of feasible regions. Overall, our method can reduce the standard deviation by 11% on average.

The timing yield results are shown in column 7 and 8 of Table XI. These data are obtained based on the timing constraints in column 6. Occasionally, the improvement from our method is insignificant due to tight feasible region constraints like in circuit c6288. Otherwise, our method usually results in significantly better timing yield than the initial results. Our method can increase the timing yield from an average of 59% to an average of 72%.

The cost overhead of our method is very small in general. The increase on cell area is at a negligible level of 0.3% and the increase on total wirelength is 2.2% on average. Therefore, our approach is much more cost-effective than the work of [9] which increases cell area by 20%. The numbers of split gates are in the rightmost column of Table XI. The CPU time of each phase of our method and the total runtime are displayed in Table XII. Usually, our method takes only a few seconds to complete.

G. Conclusion and Future Work

In order to cope with the threat of variation problems, we explore redundancy technique which is a relatively new direction for variation tolerance. We show that distributed redundancy can effectively reduce delay variability. The redundancy cost and short circuit risk can be handled through a careful algorithm design for spin-off gate placement.

Table XII. CPU time (sec).

Circuit	Phase 1	Phase 2	Phase 3	Phase 4	Total
c432	0.01	0.013	1.0	0.15	1.18
c880	0.02	0.015	1.0	0.22	1.26
c1355	0.16	0.019	1.0	0.47	1.65
c1908	1.02	0.028	1.0	0.69	2.74
c2670	0.16	0.030	2.0	0.77	2.96
c3540	2.73	0.033	2.0	1.03	5.80
c5315	1.10	0.043	3.0	1.98	6.12
c6288	8.62	0.077	4.0	2.96	15.65
c7552	1.57	0.050	4.0	2.50	8.12

We believe our method is a complementary solution instead of a competing solution to the statistical gate sizing approaches [8,54,55]. The proposed redundancy technique can be enhanced if gate/wire sizing and Steiner tree construction algorithms for dual-driver nets are available. A dual-driver net gate/wire sizing method can improve the performance/cost ratio of the redundancy in a post processing. A dual-driver Steiner tree algorithm allows more flexibility on reducing wire cost and short circuit avoidance. These two problems will be tackled in our future works.

CHAPTER VI

CONCLUSION AND FUTURE WORK

In this disseration, we have explored several aspects of the exciting field of VLSI physical design. Although we have studied only a few of the many chanllenging problems in modern VLSI design, these problem (eg. coupling capacitance, antenna effect and process variation) are real problems and urgently need to be resolved. In this chapter, we conclude this disseration and summarize its contributions and future works.

In chapter II, a probabilistic crosstalk model is proposed to guide the coupling aware layer assignment before track routing. The objective is to minimize the crosstalk risk, especially on critical nets. We have introduced a crosstalk bound analysis at layer assignment which can quickly determine the criticality of each net, given its global route and crosstalk tolerance. The work presented in chapter IV is also focused on layer assignment. There are two major contributions from the work in chapter IV: (1) an improved probabilistic model is suggested for early coupling estimation and prevention without performing track/detailed routing. This improved model is tailored for SI (Signal Integrity) driven routing; (2) a hybrid layer assignment heuristic is proposed to handle both coupling aware timing and antenna effect. The handling on antenna effect at layer assignment is based on a linear-time optimal tree partitioning algorithm from [44]. A jumper insertion algorithm is also presented with proof of its optimality. Future work will include the support for current design rule modeling of antenna effect, for instance, Partial Antenna Ratio (PAR) and Cumulative Antenna Ratio (CAR). In addition, diode insertions can be considered simultaneously with jumper insertions.

Moving forward from layer assignment, chapter III presented an algorithmic timing-driven track router. Track routing can be considered as a high-level planning of the subsequent detailed routing and it can address the issue of coupling capacitance without dealing

with the complicated design rules at detailed routing. We formulated the timing-driven track routing as a sequential ordering problem (SOP), which is a well known graph problem. Our SOP formulation of the track routing considers both coupling induced delay and wire detours simultaneously, where previous works merely considers to reduce the total amount of coupling capacitance.

The last chapter explores a slight different direction - process variation, which is one of the current buzzwords in VLSI design. A lot of works have been made to either analyze or mitigate the effects of process variations. Our work aims to reduce the delay variation and increase the timing yield. We have proposed techniques to use dual-driver and distributed wire interconnects to mitigate the variational effects. A novel gate splitting and placement methodology is also proposed. Both experimental results and HSPICE simulation showed the effectiveness of our approaches. In future work, we would like to combine our gate splitting and placment technique with gate sizing to furthur reduce variation and improve timing yield.

REFERENCES

[1] "International technology roadmap for semiconductors," Report of Semiconductor Industry Association, 2003.

[2] H. Zhou and D. F. Wong, "Global routing with crosstalk constraints," *IEEE Transactions on Computer-Aided Design*, vol. 18, no. 11, pp. 1683–1688, 1999.

[3] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, "Track assignment: a desirable intermediate step between global routing and detailed routing," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2002, pp. 59–66.

[4] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D. T. Lee, "A fast crosstalk- and performance-driven multilevel routing system," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2003, pp. 382–387.

[5] H. Shirota, T. Sadakane, M. Terai, and K. Okazaki, "A new router for reducing antenna effect in asic design," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, Santa Clara, CA, 1998, pp. 601–604.

[6] L.-D. Huang, X. Tang, H. Xiang, D. F. Wong, and I.-M. Liu, "A polynomial time-optimal diode insertion/routing algorithm for fixing antenna problem," *IEEE Transactions on Computer-Aided Design*, vol. 23, no. 1, pp. 141–147, Jan. 2004.

[7] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, "Multilevel routing with antenna avoidance," in *Proceedings of the ACM International Symposium on Physical Design*, Phoenix, AZ, 2004, pp. 34–40.

[8] J. Hu, A. Rajaram, and R. Mahapatra, "Reducing clock skew variability via cross links," in *Proceedings of the ACM/IEEE Design Automation Conference*, San Diego, CA, 2004, pp. 18–23.

[9] C.-T. Hsieh, S.-C. Chang, and K.-C. Wu, "Re-synthesis for delay variation tolerance," in *Proceedings of the ACM/IEEE Design Automation Conference*, San Diego, CA, 2004, pp. 814–819.

[10] J. Lilis, M. Krkic, and G. Beraudo, "An approach to placement-coupled logic replication," in *Proceedings of the ACM/IEEE Design Automation Conference*, San Diego, CA, 2004, pp. 711–716.

[11] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Reading, MA, Addison-Wesley, 1990.

[12] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli, "Techniques for crosstalk avoidance in the physical design of high-performance digital systems," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 1994, pp. 616–619.

[13] T. Gao and C. L. Liu, "Minimum crosstalk channel routing," *IEEE Transactions on Computer-Aided Design*, vol. 15, no. 5, pp. 465–474, May 1996.

[14] K.-S. Jhang, S. Ha, and C. S. Jhon, "COP: a Crosstalk OPtimizer for gridded channel routing," *IEEE Transactions on Computer-Aided Design*, vol. 15, no. 4, pp. 424–429, Apr. 1996.

[15] A. Vittal and M. Marek-Sadowska, "Crosstalk reduction for VLSI," *IEEE Transactions on Computer-Aided Design*, vol. 16, no. 3, pp. 290–298, Mar. 1997.

[16] P. Saxena and C. L. Liu, "Crosstalk minimization using wire perturbations," in *Proceedings of the ACM/IEEE Design Automation Conference*, New Orleans, LA, 1999, pp. 100–103.

[17] A. Onozawa, K. Chaudhary, and E. S. Kuh, "Performance driven spacing algorithms using attractive and repulsive constraints for submicron LSI's," *IEEE Transactions on Computer-Aided Design*, vol. 14, no. 6, pp. 707–718, June 1995.

[18] J. Cong, L. He, C.-K. Koh, and Z. Pan, "Interconnect sizing and spacing with consideration of coupling capacitance," *IEEE Transactions on Computer-Aided Design*, vol. 20, no. 9, pp. 1164–1169, Sept. 2001.

[19] T. Xue, E. S. Kuh, and D. Wang, "Post global routing crosstalk synthesis," *IEEE Transactions on Computer-Aided Design*, vol. 16, no. 12, pp. 1418–1430, Dec. 1997.

[20] P. N. Parakh and R. B. Brown, "Crosstalk constrained global route embedding," in *Proceedings of the ACM International Symposium on Physical Design*, Monterey, CA , 1999, pp. 201–206.

[21] H.-P. Tseng, L. Sheffer, and C. Sechen, "Timing- and crosstalk- driven area routing," *IEEE Transactions on Computer-Aided Design*, vol. 20, no. 4, pp. 528–544, Apr. 2001.

[22] C.-C. Chang and J. Cong, "Pseudopin assignment with crosstalk noise control," *IEEE Transactions on Computer-Aided Design*, vol. 20, no. 5, pp. 598–611, May 2001.

[23] S. Thakur, K.-Y. Chao, and D. F. Wong, "An optimal layer assignment algorithm for minimizing crosstalk for three layer VHV channel routing," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Seattle, WA, 1995, pp. 207–210.

[24] R. Kay and R. A. Rutenbar, "Wire packing: a strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution," in *Proceedings of the ACM International Symposium on Physical Design*, San Diego, CA, 2000, pp. 61–68.

[25] C.-C. Chang and J. Cong, "An efficient approach to multilayer layer assignment with an application to via minimization," *IEEE Transactions on Computer-Aided Design*, vol. 18, no. 5, pp. 608–620, May 1999.

[26] M. R. Becer, D. Blaauw, I. N. Hajj, and R. Panda, "Early probabilistic noise estimation for capacitively coupled interconnects," in *IEEE International Workshop on System Level Interconnect Prediction*, San Diego, CA, 2002, pp. 77–83.

[27] C. J. Alpert, J. Hu, S. S. Sapatnekar, and P. G. Villarrubia, "A practical methodology for early buffer and wire resource allocation," in *Proceedings of the ACM/IEEE Design Automation Conference*, New Orleans, LA, 2001, pp. 189–194.

[28] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Boston, MA, Kluwer Academic Publishers, 1995.

[29] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration: the VLSI Journal*, vol. 21, pp. 1–94, 1996.

[30] X. Hong, T. Xue, J. Huang, C.-K. Cheng, and E. S. Kuh, "TIGER: an efficient timing-driven global router for gate array and standard cell layout design," *IEEE Transactions on Computer-Aided Design*, vol. 16, no. 11, pp. 1323–1331, Nov. 1997.

[31] J. Hu and S. S. Sapatnekar, "A timing-constrained simultaneous global routing algorithm," *IEEE Transactions on Computer-Aided Design*, vol. 21, no. 9, pp. 1025–1036, Sept. 2002.

[32] T. Jing, X. Hong, H. Bao, Y. Cai, J. Xu, C.-K. Cheng, and J. Gu, "UTACO: a unified timing and congestion optimizing algorithm for standard cell global routing," in *Proceedings of Asia and South Pacific Design Automation Conference*, Yokohama, Japan, 2003, pp. 834–839.

[33] J. Ma and L. He, "Towards global routing with rlc crosstalk constraints," in *Proceedings of the ACM/IEEE Design Automation Conference*, New Orleans, LA, 2002.

[34] P. Saxena and S. Gupta, "On integrating power and signal routing for shield count minimization in congested regions," *IEEE Transactions on Computer-Aided Design*, vol. 22, no. 4, pp. 437–445, Apr. 2003.

[35] J. Xu, X. Hong, T. Jing, Y. Cai, and J. Gu, "A novel timing driven global routing algorithm considering coupling effects for high performance circuit design," in *Proceedings of Asia and South Pacific Design Automation Conference*, Yokohama, Japan, 2003, pp. 847–850.

[36] S. Chen and S. Smith, "Commonality and genetic algorithm," Carnegie Mellon University, Pittsburgh, PA, Technical Report CMU-RI-TR-96-27, 1996.

[37] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, pp. 55–63, Jan. 1948.

[38] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Near-optimal critical sink routing tree constructions," *IEEE Transactions on Computer-Aided Design*, vol. 14, no. 12, pp. 1417–36, Dec. 1995.

[39] C. J. Alpert, "The ispd98 circuit benchmark suite," in *Proceedings of the ACM International Symposium on Physical Design*, Monterey, CA, 1998, pp. 80–85.

[40] R. Nair, "A simple yet effective technique for global wiring," *IEEE Transactions on Computer-Aided Design*, vol. CAD-6, no. 2, pp. 165–172, Oct. 1987.

[41] C.-M. Peng, P. H. Chen, S. Malkani, and J. Lin, "Fixing antenna problem by dynamic diode dropping and jumper insertion," in *IEEE International Symposium on Quality Electronic Design*, San Jose, CA, 2000, pp. 275–282.

[42] D. Wu, J. Hu, R. Mahapatra, and M. Zhao, "Layer assignment for crosstalk minimization," in *Proceedings of Asia and South Pacific Design Automation Conference*, Yokohama, Japan, 2004, pp. 159–162.

[43] Z. Chen and I. Koren, "Layer reassignment for antenna effect minimization in 3-layer channel routing," in *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance*, Sendai, Japan, 1996, pp. 77–85.

[44] S. Kundu and J. Misra, "A linear tree partitioning algorithm," *SIAM Journal on Computing*, vol. 6, no. 1, pp. 151–154, Mar. 1977.

[45] B. Liu, I. Mandoiu, C. J. Alpert, A. B. Kahng, and A. Zelikovsky, "Minimum-buffered routing of non-critical nets for slew rate and reliability control," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2001, pp. 408–415.

[46] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: standard-cell placement tool for large industry circuits," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2000, pp. 260–263.

[47] S. R. Nassif, "Modeling and analysis of manufacturing variations," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, San Diego, CA, 2001, pp. 223–228.

[48] S. Zhao, K. Roy, and C.-K. Koh, "Estimation of inductive and resistive switching noise on power supply network in deep sub-micron CMOS circuits," in *Proceedings of the IEEE International Conference on Computer Design*, Austin, Texas, 2000, pp. 65–72.

[49] S. Muddu, A. B. Kahng, and D. Vidhani, "Noise and delay uncertainty studies for coupled RC interconnects," in *Proceedings of the IEEE International ASIC/SOC Conference*, Washington, DC, 1999, pp. 3–8.

[50] C. Papachristou, B. S. Gill, and F. G. Wolff, "Soft delay error effects in CMOS combinational circuits," in *Proceedings of the IEEE VLSI Test Symposium*, Napa, CA, 2004, pp. 325–331.

[51] A. H. Ajami, K. Banerjee, M. Pedram, and L. P.P.P. van Ginneken, "Analysis of non-uniform temperature-dependent interconnect performance in high performance ICs," in *Proceedings of the ACM/IEEE Design Automation Conference*, Las Vegas, NV, 2001, pp. 567–572.

[52] S. E. Rich, M. J. Parker, and J. Schwartz, "Reducing the frequency gap between ASIC and custom designs: a custom perspective," in *Proceedings of the ACM/IEEE Design Automation Conference*, Los Angeles, CA, 2000, pp. 432–437.

[53] T. Sakurai, "Perspectives on power-aware electronics," in *Proceedings of the IEEE International Solid-State Circuits Conference*, San Francisco, CA, 2003, pp. 26–29.

[54] E.T.A.F. Jacobs and M.R.C.M. Berkelaar, "Gate sizing using a statistical delay model," in *Proceedings of Design, Automation and Test in Europe Conference*, Paris, France, 2000, pp. 283–290.

[55] S. H. Choi, B. C. Paul, and K. Roy, "Novel sizing algorithm for yield improvement

under process variation in nanometer technology," in *Proceedings of the ACM/IEEE Design Automation Conference*, San Diego, CA, 2004, pp. 454–459.

[56] B. W. Johnson, *Design and Analysis of Fault-tolerant Digital Systems*, Reading, MA, Addison-Wesley Publishing Company, 1989.

[57] B. C. Paul N. Sirisantana and K. Roy, "Enhancing yield at the end of the technology roadmap," *IEEE Design and Test of Computers*, vol. 21, no. 6, pp. 563–571, 2004.

[58] D. Z. Pan G. Xu, L.-D. Huang and M. D.F. Wong, "Redundant-via enhanced maze routing for yield improvement," in *Proceedings of Asia and South Pacific Design Automation Conference*, Shanghai, China, 2005, pp. 1148–1151.

[59] A. B. Kahng, B. Liu, and I. I. Mandoiu, "Non-tree routing for reliability and yield improvement," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2002, pp. 260–266.

[60] P. J. Restle, T. G. McNamara, D. A. Webber, P. J. Camporese, K. F. Eng, K. A. Jenkins, D. H. Allen, M. J. Rohn, M. P. Quaranta, D. W. Boerstler, C. J. Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovick, B. L. Krauter, and B. D. McCredie, "A clock distribution network for microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 5, pp. 792–799, May 2001.

[61] N. R. Saxena S. Mitra and E. J. McCluskey, "A design diversity metric and analysis of redundant systems," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 498–510, May 2002.

[62] H. Johnson and M. Graham, *High-speed Signal Propagation: Advanced Black Magic*, Upper Saddle River, NJ, Prentice Hall, 2003.

[63] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical clock skew analysis considering intra-die process variations," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2003, pp. 914–921.

[64] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2003, pp. 621–625.

[65] C. Chen A. Srivastava, R. Kastner and M. Sarrafzadeh, "Timing driven gate duplication," vol. 12, no. 1, pp. 42–51, Jan. 2004.

[66] G. Beraudo and J. Lillis, "Timing optimization of FPGA placement by logic replication," in *Proceedings of the ACM/IEEE Design Automation Conference*, New Orleans, LA, 2003, pp. 196–201.

[67] G. Chen and J. Cong, "Simultaneous timing-driven placement and duplication," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays - FPGA*, Monterey, CA, 2005, pp. 51–59.

[68] P. K. Chan and K. Karplus, "Computing signal delay in general RC networks by tree/link partitioning," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 8, pp. 898–902, Aug. 1990.

[69] S. C. Port P. G. Hoel and C. J. Stone, *Introduction to Probability Theory*, Boston, MA, Houghton Mifflin Company, 1971.

[70] C. C.-N. Chu C.-P. Chen and D. F. Wong, "Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation," *IEEE Transactions on Computer-Aided Design*, vol. 18, no. 7, pp. 1014—1025, 1999.

[71] W. W. Wolzen, Jr, P. R. O'Brien, P. N. Strenski, C. Visweswariah, A. R. Conn, I. M. Elfadel, and C. B. Whan, "Gradient-based optimization of custom circuits using a static-timing formulation," in *Proceedings of the ACM/IEEE Design Automation Conference*, New Orleans, LA, 1999, pp. 452–459.

[72] C.J. Alpert, G. Gandham, M. Hrkic, J. Hu, A.B. Kahng, J. Lillis, B. Liu, S.T. Quay, S.S. Sapatnekar, and A.J. Sullivan, "Buffered Steiner trees for difficult instances," *IEEE Transactions on Computer-Aided Design*, vol. 21, no. 1, pp. 3–14, Jan. 2002.

[73] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, no. 3, pp. 202–211, July 1983.

[74] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 4, pp. 352–366, Apr. 1990.

[75] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Transactions on Computer-Aided Design*, vol. 10, no. 3, pp. 356–365, Mar. 1991.

[76] V. Betz A. Marquardt and J. Rose, "Timing-driven placement for FPGAs," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays - FPGA*, Monterey, CA, 2000.

VITA

Di Wu was born in Beijing, China in 1976. He graduated from Beijing No. 4 High School in 1994. After that, he received his B.E. in Computer Engineering from Bejing University of Posts and Telecommunications, Beijing, China and M.S. in Applied Physics at East Carolina University, Greenville, North Carolina, USA, in the year of 1998 and 2000, respectively. He began pursuing a Ph.D degree in Computer Science at Texas A&M University in 2000. Since then, he has worked as a teaching assistant, research assistant and instructor for the Department of Computer Science and the Department of Electrical Engineering at Texas A&M University. He recently joined the Cadence Design Systems, Inc. as a senior member of technical staff. His company address is 2655 Seely Avenue, San Jose, CA 95134.