Variation-Aware Layer Assignment With Hierarchical Stochastic Optimization on a Multicore Platform

Xiaodao Chen, Dan Chen, *Member, IEEE,* Lizhe Wang, *Senior Member, IEEE,* Ze Deng, Rajiv Ranjan, Albert Zomaya *Fellow, IEEE,* Shiyan Hu, *Senior Member, IEEE*

Abstract—As the VLSI technology enters the nanoscale regime, VLSI design is increasingly sensitive to variations on process, voltage and temperature. Layer assignment technology plays a crucial role in industrial VLSI design flow. However, existing layer assignment approaches have largely ignored these variations, which can lead to significant timing violations. To address this issue, a variation-aware layer assignment approach for cost minimization is proposed in this work. The proposed layer assignment approach is a single-stage stochastic program that directly controls the timing yield via a single parameter; and it is solved using Monte Carlo simulations and the Latin Hypercube sampling technique. A hierarchical design is also adopted to enable the optimization process on a multi-core platform. Experiments have been performed on 5000 industrial nets, and the results demonstrate that the proposed approach (1) can significantly improve the timing yield by 64.0% in comparison with the nominal design and (2) can reduce the wire cost by 15.7% in comparison with the worst-case design.

Index Terms—Layer Assignment, Variation-aware Design, Stochastic Programming

1 Introduction

As technology enters the nanoscale regime, VLSI design is increasingly sensitive to process, voltage and temperature (PVT) variations, and circuit timing is significantly impacted by variations. In practice, design without considering variations can lead to significant impact. For instance, variation of the interconnect resistances along a global timing critical net can make the net violate its timing constraint, which could even result in the malfunction of the whole circuit. Therefore, the variation-aware design is highly desirable for the VLSI design.

A large multitude of research works have been focused on statistical variation-aware design. Existing statistical optimization techniques generally choose gate sizing as the target problem such as [1], [2]. In these methods, each circuit parameter is characterized by a probability density function (PDF) to account for the variational effects in contrast to a single deterministic value as in deterministic design. They often need the assumption on variation distribution such as Gaussian distribution which limits the applicability of the techniques. There are some existing variation-aware optimization techniques without probability distribution assumptions such as an interval arithmetic based technique proposed in [3]. The problem with this technique is that it is not always accurate to use interval arithmetic to approximate any probability distribution.

In VlSI design, layer assignment, which is to assign wires to different layers, is a powerful technique to effectively reduce the interconnect delay. It has been heavily used in the industrial physical synthesis flow [5]. Since wires on the thick metal layers have much smaller resistances compared to those on thin metal layers, assigning wires there can effectively reduce the delay. Layer assignment for efficient timing closure has been considered in [6] where the problem is formulated as using minimum amount of wire resources to meet a timing target for a given buffered routing tree. This formulation is particularly interesting when design proceeds to the late stage of a physical synthesis flow [7]. At such stage, layout tuning which leads to significant Engineering Change Orders (ECOs), needs to be avoided. Meanwhile, layer assignment is desired because it only needs to change wire layer but not gate locations [6]. With the above formulations, Zhuo *et al.* proposed an approach to perform concurrent buffer insertion and layer assignment and Shiyan *et al.* proposed a polynomial time approximation scheme for minimum cost layer assignment [5], [6]. However, they do not consider variations which are quite important and may significantly impact the design. There are some previous works on variation related layer assignment techniques, such as the redundant via insertion enhanced maze

- *X. Chen, D. Chen, L. Wang and Z. Deng are with the School of Computer Science, China University of Geosciences, Wuhan, 430074, P. R. China. (Corresponding author: Dan Chen, e-mail: dan.chen@ieee.org and Lizhe Wang, Lizhe.Wang@gmail.com).*
- *D. Chen is also with the Collaborative & Innovative Center for Educational Technology, P. R. China.*
- *L. Wang is also with the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences. Beijing, P. R. China.*
- *R. Ranjan is with the CSIRO Computational Informatics Division, Australia.*
- *A. Zomaya is with the School of Information Technologies, The University of Sydney, Australia.*
- *S. Hu is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI, 49931, USA.*

routing [9] and the lithography aware detailed routing [11]. However, they are focused on specific type of variations . The other issue with the polynomial time approximation scheme for minimum cost layer assignment [6] is their assumption that every wire between adjacent set of buffers is assigned to the same layer, i.e., no cross-layer assignment is allowed. This is appropriate for the designs in the early stage but not in the late stage. It is well-known that in the late stage, cross-layer assignment is very useful for the router to further tune the wire shapes/layer assignment to reduce coupling [8], optimize vias [9], and improve timing [10]. To address issues mentioned before, this work considers a variation-aware layer assignment approach which can deal with general variation models and allow cross-layer assignment. We propose a layer assignment approach based on a stochastic programming method to achieve above goals.

The proposed approach aims to minimize the wire cost with timing constraints using stochastic programming. The approach is focused on the general variation models with a systematic Monte Carlo simulation. In this way, the proposed approach can handle variations without any assumptions on variation models. More importantly, this work considers the variations in during the layer assignment the late design stage in which there is no significant ECOs occur. This work also modifies/improves the classic stochastic programming framework to a new single-stage stochastic programming. In general, stochastic programming is a powerful technique in handling the design with variations. The previous work [4] uses the stochastic programming for the continuous gate sizing problem but not for the discrete layer assignment problem. It follows the classic two-stage stochastic programming framework; and it only minimizes the expected delay and cannot directly control the timing yield. As a result, they usually achieve the yield 100% in their simulations (see [4]) indicating the waste of the resources. Observing the above critical limitation, the classic stochastic programming framework has been largely modified/improved in this work. Our stochastic programming is a single-stage stochastic programming technique, which is able to directly control the timing yield via a single parameter. It is able to handle the variation-aware optimization on layer assignment from a global point of view and does not need assumptions on the variation distributions. In addition, the hierarchical optimization allows the stochastic program to be parallelized in a multi-core computing environment. Furthermore, the proposed approach uses the Latin Hypercube sampling techniques for efficient Monte Carlo simulations. The main contributions of the paper is summarized as follows.

- A stochastic programming method is proposed for variation-aware layer assignment with timing constraints and cost consideration. It allows the usage of a parameter to control the yield of the obtained layer assignment solution.
- The stochastic programming formulation is able to handle the variation-aware optimization on layer assignment from a global point of view; and it does not need the assumptions on the variation distributions.
- The new algorithm can significantly improve the timing yield compared to the nominal design and significantly reduce the wire cost compared to worst-case design.

The rest of the paper is organized as follows: Section 2 introduces integer non-linear programming based layer assignment without considering variations, which is also the foundation for the proposed variation-aware design. Section 3 studies variations in layer assignment and proposes a parallel stochastic programming based variation-aware layer assignment. Section 4.2 presents the experimental results with analysis. A summary of work is given in Section 5.

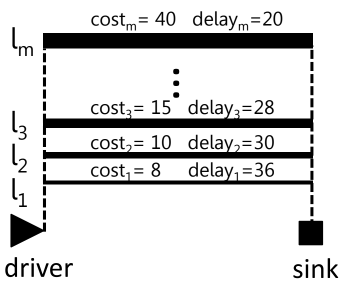
2 Integer Non-linear Programming Based Layer Assignment Approach

For the ease of presentation, this work first studies the layer assignment method without considering the variations which is based on the integer programming. This method is also the foundation for our proposed variation-aware design which will be introduced in the next section. The following subsections describe the problem formulation and integer programming formulation for layer assignment without considering variations.

2.1 Problem Formulation

Without considering the variations, our minimum cost layer assignment formulation with timing constraints follows the one in [6]. For completeness, they are included as follows. A buffered routing tree $\mathcal{T} = (V, E)$ is given as the input where V consists of driver, sinks, Steiner nodes and buffer locations. $E \subseteq V \times V$ and let $n = |V|$. Denote by $v_r, V_t(\mathcal{T}), V_b(\mathcal{T})$ the driver, the set of sinks and the set of buffers, respectively. The buffers in the buffered tree can be computed by various buffering techniques such as [14], [15]. Given a buffer b, denote by C_b, R_b, K_b the input capacitance, driving resistance and intrinsic delay, respectively.

A set of m routing layers, denoted by $L = \{l_1, l_2, \ldots, l_m\}$, are also given. Given an edge $e = (u, v)$ on a layer l, let $D_{e,l}$ denote its delay. In this work, for the simplicity in illustrating out approach, the Elmore delay model is adopted which is widely used in physical synthesis [5]. Namely, $D_{e,l} = R_{e,l} \cdot (C_{e,l}/2 + C(v))$ where $R_{e,l}, C_{e,l}, C(v)$ refer to the edge resistance, edge capacitance and load capacitance viewing at the end-point of the edge, respectively. Note that the accuracy of the Elmore delay model can be further improved by the technique in [16]. In fact, since our problem is formulated as a general non-linear integer mathematical



$cost_m$= 40 $delay_m$=20
l_m
$cost_3$= 15 $delay_3$=28
l_3
$cost_2$= 10 $delay_2$=30
l_2
$cost_1$= 8 $delay_1$=36
l_1
driver
sink

Fig. 1. An example on layer assignment.

program, *more complicated and accurate delay model can be used.*

Each sink in $v \in V_t(\mathcal{T})$ is associated with a sink capacitance denoted by C_v and a required arrival time denoted by $RAT(v)$. $RAT(v)$ specifies the latest time a signal needs to arrive at the sink v. The driver v_r is associated with an arrival time, denoted by $AT(driver)$. A net is said to satisfy the timing constraint if the arrival time at each sink is no greater than its required arrival time. Equivalently, this means that the required arrival time at the driver is no earlier than its arrival time $AT(driver)$. Given an edge e on a layer l, denote by $w_{e,l}$ the cost of the edge. For example, wire cost could be defined using wire area or wire congestion estimation. In this paper, to illustrate the effectiveness of the approach, wire area is used as in [6]. Let *tree cost* denote the cost of a layer assignment for the whole tree. It is defined as the sum of the costs of all the edges in a layer assignment solution. Our problem is to meet a timing target by layer assignment with minimal tree cost. It can be formulated as follows [6].

Timing Constrained Minimum Cost Layer Assignment:
Given a buffered binary routing tree $\mathcal{T} = (V, E)$, a set of routing layers L, and cost of each wire on each layer, to compute a layer assignment solution such that the required arrival time at driver is no earlier than its arrival time and the tree cost is minimized.

Take Figure 1 as an example, there are m layers for the simple driver sink tree, where each layer costs differently. The larger cost layer has less timing delay. It is desirable to find the layer which satisfies the timing constraint with minimum cost. Suppose the time constraint for Figure 1 is 32, layer l_2 will be assigned.

2.2 Integer Programming Formulation For Layer Assignment Without Considering Variations

We first formulate timing driven layer assignment problem as an integer programming problem. This is in the same spirit as the dynamic programming formulation in [14], [6]. Note that the timing driven layer assignment problem without considering variations is already an NP-complete problem as shown in [6]. For the convenience of illustration, the routing tree will be modified by adding some nodes to form some zero-length wires. Precisely, it is modified such that there is one node immediately before and one node immediately after any buffer location. In addition, a node will be introduced to each branch at a branching node. Refer to Figure 2 for an example. The wire between any adjacent nodes are classified into three categories, namely, standard wire, branch wire and buffer wire. A branch wire refers to a zero-length wire formed by adding extra nodes at the end of each branch. A buffer wire refer to a zero-length wire formed by adding extra nodes before and after a buffer. All other wires are standard wires. In Figure 2, wires $(8, 4), (8, 7)$ are branch wires, wires $(10, 9), (3, 2)$ are buffer wires, and wires $(2, 1), (4, 3), (6, 5), (7, 6), (9, 8)$ are standard wires. The constraints associated with each type of wires are formulated as follows.

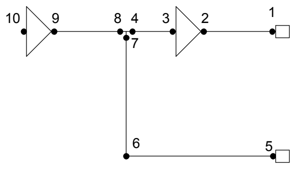


Fig. 2. A modified routing tree where more nodes are added without changing the topology of the tree.

Standard wire: Given a standard wire e, introduce binary variables $x_{e,l}$ for the layer assignment on e. That is, let $x_{e,l} = 1$ denote assigning e to layer l. For any edge e, it can only be assigned to one layer. Thus, $x_{e,l_1} + x_{e,l_2} + \ldots, +x_{e,l_m} = 1$. Given a node v, let $Q(v)$ denote the required arrival time at v and $C(v)$ the downstream capacitance when viewing at v. Recall that $R_{e,l}, C_{e,l}$ denote the wire resistance and capacitance of e when assigning e to layer l, respectively. Consider a standard wire $e = (u, v)$. $Q(u)$ can be computed as $Q(u) = Q(v) - \sum_{i=1}^{m} [R(e, l_i) \cdot [C(e, l_i)/2 + C(v)]] x_{e,l_i}$. Note that in the above, only one x_{e,l_i} will be 1 due to the constraint $x_{e,l_1} + x_{e,l_2} + \ldots, +x_{e,l_m} = 1$. $C(u)$ can be computed as $C(u) = C(v) + \sum_{i=1}^{m} C_{e,l_i} x_{e,l_i}$.

Buffer wire: When a buffer is reached, C will be set to the buffer capacitance since layer assignment will not change buffer. Otherwise, ECO placement needs to be invoked which is not desired in the late design stage. Q will be updated considering the buffer delay. For the buffer wire $(v_1, v_2) = (3, 2)$ in Figure 2, we have

$$Q(v_1) = Q(v_2) - D_b(v_2), C(v_1) = C_b, \tag{1}$$

where C_b is the buffer capacitance for buffer b and $D_b(v)$ is the buffer delay computed as $D_b(v) = R_b \cdot C(v) + K_b$ where R_b and K_b the buffer resistance and intrinsic delay, respectively. Note that the driver can be treated in the same way as a buffer (e.g., the buffer wire for the driver is $(10, 9)$ in Figure 2). The timing constraint says that $Q(driver) \geq AT(driver)$ at the input of the driver.

Branch wire: Suppose that two branches B_1 and B_2 meet at a branching node v and the newly introduced nodes are v_1 and v_2. For example, $(v, v_1) = (8, 4), (v, v_2) = (8, 7)$ in Figure 2. $Q(v), C(v)$ can computed as follows.

$$Q(v) = \min\{Q(v_1), Q(v_2)\}, C(v) = C(v_1) + C(v_2). \tag{2}$$

Note that min is taken since worst-case performance needs to be guaranteed. This is to follow the dynamic programming procedure in [14], [6].

Recall that C_v denotes the sink capacitance when v is a sink and $AT(driver)$ and $RAT(v)$ denote the arrival time and required arrival time at driver and at v, respectively. $w_{e,l}$ denotes the wire cost of a wire e at layer l. The timing constrained minimum cost layer assignment can be formulated in Eqn. (3).

$$\begin{array}{ll}
min & \sum_{\forall \text{ standard wire } e} \sum_{i=1}^{m} w_{e,l_i} x(e, l_i) \\
s.t. & Q(v_j) - \sum_{i=1}^{m} [R_{e,l_i} C_{e,l_i}/2 + R_{e,l_i} C(v_j)] x_{e,l_i} = \\
 & Q(v_i), \forall \text{ standard wire } e = (v_i, v_j) \\
 & C(v_j) + \sum_{i=1}^{m} C_{e,l_i} x_{e,l_i} = C(v_i), \\
 & \forall \text{ standard wire } e = (v_i, v_j) \\
 & Q(v_i) \le Q(v_j), \forall \text{ branch wire } e = (v_i, v_j) \\
 & C(v_i) = \sum_{\forall \text{ branch wire } e=(v_i,v_j)} C(v_j) \\
 & Q(v_i) = Q(v_j) - [R_{b,v_i} \cdot C(v_j) + K_{b,v_i}], \\
 & \forall \text{ buffer wire } e = (v_i, v_j) \text{ and buffer b is located} \\
 & C(v_i) = C_{b,v_i}, \\
 & \forall \text{ buffer wire } e = (v_i, v_j) \text{ and buffer b is located} \\
 & Q(driver) \ge AT(driver), \text{ at the input of the driver} \\
 & Q(v) = RAT(v), \forall v \text{ is a sink} \\
 & C(v_i) = C_{v_i}, v_i \text{ is a sink} \\
 & x_{e,l_i} = \{0, 1\}, \forall e, i.
\end{array} \tag{3}$$

To concentrate on illustrating our main idea, this work does not consider via delay. However, it can be easily handled by modifying the constraints since vias can be modeled as resistors and capacitors as well. For example, for two connecting wires (v_1, v_2) and (v_2, v_3), without handling via, the capacitance at v_2 is is $C(v_2) = C_{(v_2,v_3),l_1} x_{(v_2,v_3),l_1} + C_{(v_2,v_3),l_2} x_{(v_2,v_3),l_2} + C(v_3)$. To handle via, $C(v_2) = C_{(v_2,v_3),l_1} x_{(v_2,v_3)l_1} + C_{(v_2,v_3),l_2} x_{(v_2,v_3)l_2} + C(v_3) + x_{(v_1,v_2),l_1} \cdot x_{(v_2,v_3),l_2} \cdot C_{via,l_1,l_2} + x_{(v_1,v_2),l_2} \cdot x_{(v_2,v_3),l_1} \cdot C_{via,l_1,l_2}$ where C_{via,l_1,l_2} is the capacitance of the via connecting l_1 and l_2. It will just make the integer non-linear program more complicated but not impact our algorithmic framework.

It is helpful to illustrate Eqn. (3) using a simple example in Figure 2. Assuming that both buffers are of type b in Figure 2, the corresponding integer programming formulation is in Eqn. (4).

$$\begin{array}{l}
min \sum_{e \in \{(2,1),(4,3),(6,5),(7,6),(9,8)\}} \sum_{i=1}^{m} w_{e,l_i} x(e, l_i) \\
s.t. \\
Q(1) - \sum_{i=1}^{m} [R_{(2,1),l_i} C_{(2,1),l_i}/2 + R_{(2,1),l_i} C(1)] x_{(2,1),l_i} = Q(2), \\
Q(3) - \sum_{i=1}^{m} [R_{(4,3),l_i} C_{(4,3),l_i}/2 + R_{(4,3),l_i} C(3)] x_{(4,3),l_i} = Q(4), \\
Q(5) - \sum_{i=1}^{m} [R_{(6,5),l_i} C_{(6,5),l_i}/2 + R_{(6,5),l_i} C(5)] x_{(6,5),l_i} = Q(6), \\
Q(6) - \sum_{i=1}^{m} [R_{(7,6),l_i} C_{(7,6),l_i}/2 + R_{(7,6),l_i} C(6)] x_{(7,6),l_i} = Q(7), \\
Q(8) - \sum_{i=1}^{m} [R_{(9,8),l_i} C_{(9,8),l_i}/2 + R_{(9,8),l_i} C(8)] x_{(9,8),l_i} = Q(9), \\
C(1) + \sum_{i=1}^{m} C_{(2,1),l_i} x_{(2,1),l_i} = C(2), \\
C(3) + \sum_{i=1}^{m} C_{(4,3),l_i} x_{(4,3),l_i} = C(4), \\
C(5) + \sum_{i=1}^{m} C_{(6,5),l_i} x_{(6,5),l_i} = C(6), \\
C(6) + \sum_{i=1}^{m} C_{(7,6),l_i} x_{(7,6),l_i} = C(7), \\
C(8) + \sum_{i=1}^{m} C_{(9,8),l_i} x_{(9,8),l_i} = C(9), \\
Q(8) \le Q(4), \\
Q(8) \le Q(7), \\
C(8) = C(4) + C(7), \\
Q(3) = Q(2) - [R_{b,3} \cdot C(2) + K_{b,3}], \\
Q(10) = Q(9) - [R_{b,10} \cdot C(9) + K_{b,10}], \\
C(3) = C_{b,3}, \\
C(10) = C_{b,10}, \\
Q(driver) = Q(10) \ge AT(driver) = AT(10), \\
Q(1) = RAT(1), \\
Q(5) = RAT(5), \\
C(1) = C_1, \\
C(5) = C_5, \\
x_{e,l_i} = \{0, 1\}, \forall e, i.
\end{array} \tag{4}$$

Note that in Eqn. (3), $C(v), Q(v), x_{e,l}$ are variables and Eqn. (3) is not a linear program due to $C(v)x_{e,l}$. More importantly, when variations are considered, the constants in Eqn. (3) (which are coefficients and constant terms) such as wire capacitance $C_{e,l}$ are random variables as mentioned in Section 3.1. For example, each $C_{e,l}$ can be modeled as

$$C_{e,l} = C_{e,l}^0 + \frac{\partial C}{\partial W} \sum_{(a,b)} \alpha_{(a,b)} \Delta W_{(a,b)} + \frac{\partial C}{\partial T} \sum_{(a,b)} \beta_{(a,b)} \Delta T_{(a,b)} + \delta_g + \epsilon, \tag{5}$$

where ΔW and ΔT are random variables. In addition, buffer resistances (or capacitance and intrinsic delay) are shown as R_{b,v_i} to incorporate the fact that buffer resistance at different locations v_i may have different variations. The key problem is to solve an integer program considering the uncertainty on the coefficients and constant terms.

3 Variation-Aware Stochastic Layer Assignment

The focus of this paper is variation-aware design. To handle variations in layer assignment, a parallel stochastic optimization method based on the integer programming layer assignment method is proposed. The variation model and traditional stochastic programming are first studied. After that, a stochastic parallel yield-driven Stochastic Optimization Framework is proposed.

3.1 Variation Models

In this work, the variations on gates and wires are considered. To model the circuit variations, the widely accepted modeling techniques proposed in [17] is used. The model is a first order approximation of a general non-linear variation model and is able to capture the major components of most variations [17]. This model has been extensively used in statistical timing analysis and optimization works such as [18], [4]. In this paper, to illustrate the effectiveness of the proposed approach, for interconnects, variations on metal width W and metal thickness T are considered. However, our approach is not limited to these variations and other variations can be handled. For a wire e in layer l, the wire capacitance can be modeled as

$$C_{e,l} = C^0_{e,l} + \frac{\partial C}{\partial W}\Delta W + \frac{\partial C}{\partial T}\Delta T + \delta_{e,l} + \epsilon, \quad (6)$$

where $C^0_{e,l}$ refers to the nominal value, $\frac{\partial C}{\partial W}$ and $\frac{\partial C}{\partial T}$ refer to the sensitives of wire capacitance to metal width and metal thickness, respectively. ΔW and ΔT are random variables which refer to variations in meta width and metal thickness, respectively. $\delta_{e,l}$ and ϵ are random variables which refer to the local and global variations. We similarly model the wire resistance. Since the layer assignment is applied for a buffered routing tree, variations on driver, buffers and sinks also need to be considered. In this work, to illustrate the effectiveness of the proposed approach, variations on gate length L and threshold voltage Vt are considered and other variations can be handled as well. They can be similarly modeled as above. For example, gate capacitance can be modeled as

$$C_g = C_{g0} + \frac{\partial C}{\partial L}\Delta L + \frac{\partial C}{\partial V}\Delta V + \delta_g + \epsilon. \quad (7)$$

As in [18], to handle spatial correlation which is especially important for a global net, a mesh is layered on the circuit. Each segment of wire in a grid will be indexed by (a, b). Following [18], incorporating spatial correlation into consideration, we have

$$C_{e,l} = C^0_{e,l} + \frac{\partial C}{\partial W}\sum_{(a,b)} \alpha_{(a,b)}\Delta W_{(a,b)} + \frac{\partial C}{\partial T}\sum_{(a,b)} \beta_{(a,b)}\Delta T_{(a,b)} + \delta_{e,l} + \epsilon, \quad (8)$$

where α, β are the parameters, and $\Delta W_{(a,b)}$ and $\Delta T_{(a,b)}$ are independent components which can be obtained through performing principal component analysis to the correlated random variations. The models on gate capacitance and resistance considering spatial correlations can be similarly derived. Refer to [18] for more details.

3.2 Two-Stage Stochastic Programming

Stochastic programming is a popular technique to solve the mathematical program with uncertainty. It has been applied to various fields such as scheduling, telecommunications, computational finance, and production control [19]. One of the most well-known stochastic programming techniques is the *two-stage stochastic programming* [19].

In the first stage, one makes an optimization decision (i.e., compute a layer assignment solution) without considering the wire variations. In the second stage, when the variations are considered, the first-stage solution will be adjusted such that certain objective is optimized (delay of the circuit is minimized). The simple separate optimization as above would lead to the inferior solution since it solely relies on the second-stage adjustment to improve the solution quality. Thus, the *stochastic programming with recourse* is often used to link the two stages. It computes the solution minimizing the expected delay considering the variations which involves the second stage and it is also an iterative procedure.

Following the conventions in the stochastic programming literature [20], the above can be formulated as follows. For the convenience of illustration, only the stochastic linear program is shown.

$$\begin{array}{ll} \min & c^T x + E[Q(x, \vartheta(\omega))] \\ s.t. & \\ & Ax \geq b \\ & x \geq 0, \end{array} \quad (9)$$

where $c^T x$ denotes the optimization objective without variations (e.g., nominal design), $E[\cdot]$ denotes the expected value, and $Q(x, \vartheta(\omega))$ denotes the optimal objective value for the second-stage problem. $E[Q(x, \vartheta(\omega))]$ serves as a penalty function (e.g., additional delay due to variations). The second stage problem is defined as follows.

$$\begin{array}{ll} \min & = p(\omega)^T y \\ s.t. & \\ & T(\omega)x + W(\omega)y = h(\omega) \\ & y \geq 0, \end{array} \quad (10)$$

where ω are random variables, $p(\omega)$, $T(\omega)$, $W(\omega)$ are random coefficients and $h(\omega)$ are random constants. Note that in the first-stage problem, x is the variable while in the second stage problem, y is the variable and x is treated as a constant. The above $\vartheta(\omega)$ refers to the tuples consisting of $p(\omega), T(\omega), h(\omega)$. $E[\cdot]$ is taken with the probability distribution of ω. To solve the two-stage stochastic linear program, a commonly-used technique is the sample average approximation technique [12][13][20]. In this method, roughly speaking, each time some random samples are generated according to the probability density function of ω and the expected value $E[Q(x, \vartheta(\omega))]$ can be approximated by averaging the Q values among samples. This procedure is iterated until some stopping criterion is met.

The main weakness of the above framework is that it tries to minimize the expected delay that is only weakly related to the timing yield, which is our main target.

3.3 Single-Stage Stochastic Optimization Framework

In variation-aware optimization, our target is actually to compute the best solution satisfying a yield target. In the layer assignment context, this means that we want to compute the minimum cost layer assignment such that 99% (this is the target value in [13], it can be set to other values) of circuits satisfy a timing constraint T. The above classic stochastic programming framework does not capture this. The previous stochastic programming based gate sizing technique in [4] follows the above framework and thus they cannot directly control the yield as well.

Our idea is to formulate the stochastic layer assignment problem into into a single integer program instead of a two-stage integer program (with two separate integer programs). This allows us to introduce a parameter called *robust parameter*, denoted by γ, to directly control the yield. Further, such a new framework allows us to easily parallelize the algorithm.

3.3.1 The Single-Stage Stochastic Program

The traditional two-stage stochastic program iteratively processes the yield factor tuning and yield factor evaluation until it satisfies the stop criteria. In contrast, the proposed single-stage stochastic program for this layer assignment case, has been designed to explicitly control the yield value by a one step integer program. For this layer assignment problem, without loss of solution quality, the proposed single-stage method has better performance efficiency than the traditional two-stage one. For this layer assignment problem, without loss of solution quality, the proposed single-stage method has better runtime than the traditional two-stage one. We will transform the integer program with random coefficients in Eqn. (3) to an integer program without random variables. This is accomplished by instantiating random coefficients to deterministic constants through Monte Carlo samples. Let us denote the constraints in Eqn. (3) by $\mathcal{C}(\Omega)$, where Ω is an instance of the set of all coefficients/constants in Eqn. (3). That is, a Ω corresponds to a Monte Carlo sample. Since in a Monte Carlo sample, the wire width and wire thickness for all wires in all layers are known, the wire capacitances and wire resistances for all wires in all layers can be obtained. More importantly, they are constants in each Monte Carlo sample. Note that they are the coefficients/constants in Eqn. (3). The above also applies to the gate capacitances and resistances. Suppose that we have randomly generated k Monte Carlo samples for the variation-aware layer assignment problem. Let Ω_i denote the i-th Monte Carlo sample in the totally k Monte Carlo samples. The new integer programming formulation is as follows.

$$\begin{array}{ll} min & \sum_{\forall \text{ standard wire } e} \sum_{i=1}^{m} w_{e,l_i} x(e,l_i) \\ s.t. & \mathcal{C}(\Omega_1) \\ & \mathcal{C}(\Omega_2) \\ & \dots \\ & \mathcal{C}(\Omega_k) \\ & x_{e,l_i} = \{0,1\}, \forall e, i. \end{array} \tag{11}$$

For example, Eqn. (4) can be instantiated as follows. For simplicity, suppose that there are only two layers, namely, $m = 2$. Further suppose that there are two Monte Carlo samples. In the first Monte Carlo sample, $C_{e,l_1} = 2$ and $C_{e,l_2} = 4$ for all wire e, and $R_{e,l_1} = 20$ and $R_{e,l_2} = 10$ for all wire e. The capacitance of each buffer/sink/driver is 5 and resistance is 5. The buffer/driver intrinsic delay is 1. In the second Monte Carlo sample, $C_{e,l_1} = 4$ and $C_{e,l_2} = 8$ for all wire e, and $R_{e,l_1} = 40$ and $R_{e,l_2} = 20$ for all wire e. The capacitance of each buffer/sink/driver is 10 and resistance is 10. The buffer/driver intrinsic delay is 2.

Let Q_i, C_i denote the variables corresponding to the sample Ω_i. The Monte Carlo simulation based integer program corresponding to Figure 2 is shown in Eqn. (16). Note that this is a deterministic integer program with no random variables. In the above formulation, the target variables are x_{e,l_i} and they are shared in all $\mathcal{C}(\Omega_1), \mathcal{C}(\Omega_2), \dots, \mathcal{C}(\Omega_k)$. However, each $\mathcal{C}(\Omega_i)$ has different set of variables of capacitance $C_i(v)$ and RAT $Q_i(v)$. In particular, we are interested in $Q_i(driver)$ which is the RAT at driver (e.g., $Q_i(10)$ in Eqn. (16)) in a $\mathcal{C}(\Omega_i)$. Since we have k samples, we have $Q_1(driver), Q_2(driver), \dots, Q_k(driver)$.

Note that by default, we have set all of the $Q_i(driver) \geq AT(driver)$ as in Eqn. (16). This means that we are to compute the worst-case design since all of the k samples need to satisfy the timing constraint. We are to investigate setting $\frac{1}{k}\sum_{i=1}^{k} Q_i(driver) \geq AT(driver)$ as a constraint. Roughly speaking, this means that we compute a design satisfying the timing constraint in an average-case sense.

Let $\mathcal{C}(\Omega_i) \backslash Q_i(driver)$ denote the set of constraints except the constraint of "$Q(v) \geq AT(driver)$" in Eqn. (3). Roughly speaking, the average-case design can be obtained from

$$\begin{array}{ll} min & \sum_{\forall \text{ standard wire } e} \sum_{i=1}^{m} w_{e,l_i} x(e,l_i) \\ s.t. & \mathcal{C}(\Omega_1) \backslash Q_1(driver) \\ & \mathcal{C}(\Omega_2) \backslash Q_2(driver) \\ & \dots \\ & \mathcal{C}(\Omega_k) \backslash Q_k(driver) \\ & \frac{1}{k}\sum_{i=1}^{k} Q_i(driver) \geq AT(driver) \\ & x_{e,l_i} = \{0,1\}, \forall e, i. \end{array} \tag{12}$$

Consider a yield target Y, e.g., $Y = 99\%$. We actually want to compute a design such that 99% of $Q_i(driver)$ satisfy the timing constraint, i.e., greater than $AT(driver)$. Thus, if we would be able to sort all k $Q_i(driver)$ and set the $0.01k$ smallest one to be greater than $AT(driver)$, we would compute a design exactly matching the yield requirement with no overdesign. However, one cannot sort the constraints in a mathematical programming formulation. Thus, we propose to use the following approximate method.

Two additional variables are introduced. The first variable Q_w computes the worst-case RAT among all k layer assignment solutions, and the second variable T_a to compute the average-case RAT value among all k layer assignment solutions. Precisely, as mentioned above, $Q_w = \max\{Q_1(driver), Q_2(driver), \ldots, Q_k(driver)\}$ and $Q_a = \frac{1}{k}\sum_{i=1}^{k} Q_i(driver)$. These can be certainly formulated in mathematical program.

Subsequently, a parameter, called *robust parameter* denoted by γ, is introduced to control the tradeoff between the worst-case delay Q_w and the average delay Q_a. Precisely, the constraint is set as $(1-\gamma)Q_a+\gamma Q_w \geq AT(driver)$. Clearly, setting $\gamma = 0$ leads to the average case design while setting $\gamma = 1$ leads to the worst case design. γ is directly related to the timing yield and varying γ, different tradeoff can be obtained. Our formulation is as follows.

$$\begin{array}{ll} min & \sum_{\forall \text{ standard wire } e} \sum_{i=1}^{m} w_{e,l_i} x(e,l_i) \\ s.t. & \mathcal{C}(\Omega_1)\backslash Q_1(driver) \\ & \mathcal{C}(\Omega_2)\backslash Q_2(driver) \\ & \ldots \\ & \mathcal{C}(\Omega_k)\backslash Q_k(driver) \\ & Q_w \leq Q_i(driver), \forall i \\ & Q_a = \frac{1}{k}\sum_{i=1}^{k} Q_i(driver) \\ & (1-\gamma)Q_a + \gamma Q_w \geq AT(driver) \\ & x_{e,l_i} = \{0,1\}, \forall e, i. \end{array} \tag{13}$$

In the example of Eqn. (16), this means that we replace

$$\begin{array}{l} Q_1(10) \geq AT(driver) = AT(10) \\ Q_2(10) \geq AT(driver) = AT(10) \end{array} \tag{14}$$

with

$$\begin{array}{l} Q_w \leq Q_1(10) \\ Q_w \leq Q_2(10) \\ Q_a = [Q_1(10) + Q_2(10)]/2 \\ (1-\gamma)Q_a + \gamma Q_w \geq AT(driver) = AT(10). \end{array} \tag{15}$$

Note again that the above formulation is a deterministic integer program without any random coefficients and γ is a constant. The benefit of the whole technique is that the variation is handled in a global way since each time the mathematical program is solved in a global fashion by the mathematical program solver. There are two issues with the above formulation, namely, how large the value of k can be and how to set γ.

3.3.2 Hierarchical Optimization

Suppose that $k = 5000$ Monte Carlo samples are used. This means that the obtained integer program has the size 5000× of the original layer assignment integer program in Eqn. (3). Solving this large system is computationally prohibitive. We will first reduce the number of Monte Carlo samples used in the stochastic programming formulation. For this, the Latin Hypercube sampling based Monte Carlo simulation will be used. This will enable us to reduce the sample sizes from commonly used large number samples to a small size sample set. In [12], it reduces 5000 samples to just $k = 200$ Latin Hypercube samples. Refer to Section 3.3.4 for details.

The problem is that even for $k = 200$ Latin Hypercube samples, the integer program in Eqn. (13) would still be too large to be efficiently solved for large signal nets. We use the following hierarchical optimization technique to tackle this difficulty. First pick r samples out of k samples and remove them from k samples. We formulate the integer program as in Eqn. (13) except that there are only r samples in contrast to k samples. r is chosen such that the resulting integer program can be efficiently solved using the mathematical programming solver.

After computing the solution, we find the $Q_i(driver)$ closest to $AT(driver)$. This Q and the corresponding sample roughly determines the yield on these r samples since one just needs to calculate the number of $Q(driver)$ smaller than the picked $Q_i(driver)$ and it gives the number of samples satisfying the timing constraint. Due to this, the sample corresponding to the picked $Q_i(driver)$ is called a *critical sample* for those r samples since it determines the yield on them. We then proceed to the other r samples out of $k-r$ samples and perform the same as above to find the second critical sample. This process is iterated until all k samples have been considered.

In a total, we will have $\lceil k/r \rceil$ integer programs each of which contain r samples (perhaps except the last integer program). $\lceil k/r \rceil$ critical samples will be computed. They will be used to formulate a new integer program which contains $\lceil k/r \rceil$ samples. If $\lceil k/r \rceil$ samples are still too large to be efficiently solved, we can recursively apply the same decomposition procedure as above on them. As indicated in our experiments, a two-level decomposition is sufficient. Precisely, we set $k = 200$ and $r = 15$. Each of the 14 first-level integer programs contains $r = 15$ samples (except the last one). After solving all 14 such integer programs, we have 14 critical samples which will be formulated into the second-level integer program. When solving this second-level integer program, we need to guarantee that the worst-case design still satisfies the timing constraint since they are the representative critical samples in each set. When all of them satisfy the timing constraint, there should be enough samples (which are the ones less critical than them) satisfying the timing constraint and therefore the yield target is satisfied. It is also clear that $r = 15$ is chosen to balance the first-level integer program size and second-level integer program size.

Note that γ and the yield target are correlated in weak sense. The first-level solution to Eqn. (13) (consisting of $r = 15$ samples) depends on γ. If the yield target is set to 99%, it does not mean that we need the Q of all of $r = 15$ samples to be $\geq AT(driver)$. The number of such samples depends on γ. In fact, typically γ is much smaller than 1 for 99% yield target according to our experiments. On the other hand, due to the change in sample space between two levels, the criticality estimation might not be accurate. Therefore, the above decomposition technique is not guaranteed to compute the optimal solution. However, it works well as indicated by our experiments.

Note that each integer program is actually hard to solve due to integer constraints. Thus, we relax each integer constraints to real constraint, solve it using a mathematical programming solver (IPOPT), and then applies the well-known sequential rounding technique to round real-valued solution to integer solution. Such a technique has been widely used in VLSI CAD (see., e.g., [21]). For completeness, we briefly describe it as follows. First, the integer constraints are relaxed as $0 \leq x_{e,l_i} \leq 1, \forall e, i$. The new mathematical program is then solved where the solution may contain fractional numbers. All variables in the solution that are equal to 0 or 1 will be fixed accordingly. This is accomplished by adding new constraints such as $x_{e,l_i} = 1$ to Eqn. (13) if x_{e,l_i} is equal to 1 in the solution of the relaxed mathematical program. In addition, In the solution, those variables close to 1 or 0 will be rounded and fixed. Precisely, the variables $< \delta$ will be rounded to 0 and the variables $> 1 - \delta$ will be rounded to 1. If there is no such variable, the one with the smallest rounding error will be rounded. The above can be accomplished by adding new constraints such as $x_{e,l_i} = 1$ or $x_{e,l_i} = 0$ to Eqn. (13) for some x_{e,l_i}. The new mathematical program will be solved again (with some fixed variables) to obtain the new solution. This process is iterated until all variables are integers.

$$
\begin{aligned}
&min \sum\nolimits_{e\in\{(2,1),(4,3),(6,5),(7,6),(9,8)\}} \sum\nolimits_{i=1}^{m} w_{e,l_i} x(e,l_i)\\
&s.t.\\
&Q_1(1) - [20 + 20C_1(1)]x_{(2,1),l_1} - [20 + 10C_1(1)]x_{(2,1),l_2} = Q_1(2),\\
&Q_1(3) - [20 + 20C_1(3)]x_{(4,3),l_1} - [20 + 10C_1(3)]x_{(4,3),l_2} = Q_1(4),\\
&Q_1(5) - [20 + 20C_1(5)]x_{(6,5),l_1} - [20 + 10C_1(5)]x_{(6,5),l_2} = Q_1(6),\\
&Q_1(6) - [20 + 20C_1(6)]x_{(7,6),l_1} - [20 + 10C_1(6)]x_{(7,6),l_2} = Q_1(7),\\
&Q_1(8) - [20 + 20C_1(7)]x_{(9,8),l_1} - [20 + 10C_1(8)]x_{(9,8),l_2} = Q_1(9),\\
&C_1(1) + 2x_{(2,1),l_1} + 4x_{(2,1),l_1} = C_1(2),\\
&C_1(3) + 2x_{(4,3),l_1} + 4x_{(4,3),l_1} = C_1(4),\\
&C_1(5) + 2x_{(6,5),l_1} + 4x_{(6,5),l_1} = C_1(6),\\
&C_1(6) + 2x_{(7,6),l_1} + 4x_{(7,6),l_1} = C_1(7),\\
&C_1(8) + 2x_{(9,8),l_1} + 4x_{(9,8),l_1} = C_1(8),\\
&Q_1(4) \geq Q_1(8),\\
&Q_1(7) \geq Q_1(8),\\
&C_1(8) = C_1(4) + C_1(7),\\
&Q_1(3) = Q_1(2) - [5 \cdot C_1(2) + 1],\\
&Q_1(10) = Q_1(9) - [5 \cdot C_1(9) + 1],\\
&C_1(3) = 5,\\
&C_1(10) = 5,\\
&Q_1(1) = RAT(1),\\
&Q_1(5) = RAT(5),\\
&C_1(1) = 5,\\
&C_1(5) = 5,\\
&Q_1(driver) = Q_1(10) \geq AT(driver) = AT(10), \qquad\qquad (16)\\
&Q_2(1) - [80 + 40C_2(1)]x_{(2,1),l_1} - [80 + 20C_2(1)]x_{(2,1),l_2} = Q_2(2),\\
&Q_2(3) - [80 + 40C_2(3)]x_{(4,3),l_1} - [80 + 20C_2(3)]x_{(4,3),l_2} = Q_2(4),\\
&Q_2(5) - [80 + 40C_2(5)]x_{(6,5),l_1} - [80 + 20C_2(5)]x_{(6,5),l_2} = Q_2(6),\\
&Q_2(6) - [80 + 40C_2(6)]x_{(7,6),l_1} - [80 + 20C_2(6)]x_{(7,6),l_2} = Q_2(7),\\
&Q_2(8) - [80 + 40C_2(7)]x_{(9,8),l_1} - [80 + 20C_2(8)]x_{(9,8),l_2} = Q_2(9),\\
&C_2(1) + 4x_{(2,1),l_1} + 8x_{(2,1),l_1} = C_2(2),\\
&C_2(3) + 4x_{(4,3),l_1} + 8x_{(4,3),l_1} = C_2(4),\\
&C_2(5) + 4x_{(6,5),l_1} + 8x_{(6,5),l_1} = C_2(6),\\
&C_2(6) + 4x_{(7,6),l_1} + 8x_{(7,6),l_1} = C_2(7),\\
&C_2(8) + 4x_{(9,8),l_1} + 8x_{(9,8),l_1} = C_2(8),\\
&Q_2(4) \geq Q_2(8),\\
&Q_2(7) \geq Q_2(8),\\
&C_2(8) = C_2(4) + C_2(7),\\
&Q_2(3) = Q_2(2) - [10 \cdot C_2(2) + 2],\\
&Q_2(10) = Q_2(9) - [10 \cdot C_2(9) + 2],\\
&C_2(3) = 10,\\
&C_2(10) = 10,\\
&Q_2(1) = RAT(1),\\
&Q_2(5) = RAT(5),\\
&C_2(1) = 10,\\
&C_2(5) = 10,\\
&Q_2(driver) = Q_2(10) \geq AT(driver) = AT(10),\\
&x_{e,l_i} = \{0,1\}, \forall e, i.
\end{aligned}
$$

3.3.3 *Parallel Optimizing Procedures*

The hierarchical optimization on layer assignment can make the large system efficiently solvable. More importantly, it can be easily parallelized in the multi-core programming environment. This is due to the fact that the integer programs formed by each set of (first-level) r samples can be solved totally independently. The synchronization is only needed when we come to solve the second-level integer program (consisting of $\lceil k/r \rceil$ critical samples). This process is illustrated in Figure 3.

It remains to show how to determine γ. In fact, this can be accomplished by searching for various γ with certain step size. For example, if the step size is 0.1, one could search for γ from 0.1 to 0.2 then to 0.3 by noting that $\gamma = 0$ refers to the average-case design and $\gamma = 1$ refers to the worst-case design. For each solution, we will evaluate its yield and return the minimum cost solution satisfying the yield target. It is clear that one can parallelize this procedure since the mathematical program corresponding to different γ will not interact with each other. This can also be easily implemented in multi-core programming environment.

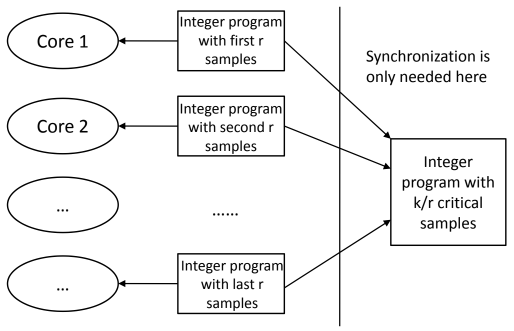

Core 1
Integer program with first r samples
Core 2
Integer program with second r samples
...
......
...
Integer program with last r samples
Synchronization is only needed here
Integer program with k/r critical samples

Fig. 3. Illustration of the hierarchical optimization. Each integer program can be assigned to different cores (subject to availability) and synchronization is needed only when solving the final (second-level) $\lceil k/r \rceil$ critical samples.

3.3.4 *Latin Hypercube Sampling Based Monte Carlo Simulation*

In our formulation, $k = 200$ Latin Hypercube samples are used. These are sufficient to obtain a close approximation to the full-fledged Monte Carlo simulation technique. The latter uses simple sampling, often needs over 5000 samples and optimization with these samples would be very time consuming. Latin Hypercube

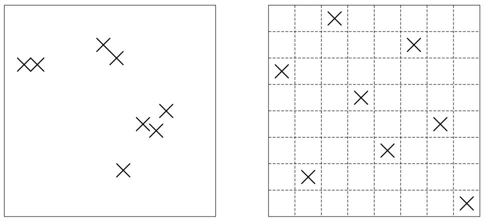


Fig. 4. (a) 8 samples generated by standard sampling (b) 8 samples generated by Latin Hypercube sampling.

sampling allows us to sample the variation space more evenly, which avoids generating many samples in a small local region [22]. Consequently, one can cover most simulation space with small number of samples. Such a technique has been used in statistical timing analysis and optimization in VLSI CAD literatures [23], [24]. They have demonstrated excellent yield estimation accuracy compared to the full-fledged 5000 samples based Monte Carlo samples. For example, the work [22] shows that compared to 5000 samples based Monte Carlo simulation, 200 Latin Hypercube samples based Monte Carlo simulation can have error of $< 1\%$. Note that our approach is not restricted to Latin Hypercube sampling technique. Other sample reduction technique such as Quasi-Monte carlo techniques can be certainly used.

For completeness, the details of generating Latin Hypercube samples are included as follows. Let us use a simple two-dimensional simulation example to illustrate the Latin Hypercube sampling technique. Given two random variables x, y, we are to generate 8 simulation samples. Suppose that they are as shown in Figure 4(a) by the standard sample generation technique. These samples do not evenly distributed in the simulation space and more importantly, they do not cover the simulation space well. The simulations with these samples would not lead to good yield estimation accuracy. To generate samples well distributed in the simulation space, one needs to memorize the locations of the previously generated samples. Latin Hypercube sampling technique accomplishes this by dividing simulation space into rows and columns and requiring that only one sample can be generated in each row and column. For example, in Figure 4(b), we first divide the simulation space into 8×8 equal-probability bins. When a sample is generated from a bin, its corresponding row and column will be removed to avoid generating more samples in the row and column. It can be seen that the generated samples could be well distributed in the simulation space. Therefore, one can use a much smaller number of samples (200 samples as in [22], [23], [25]) to well approximate the full-fledged 5000 samples based Monte Carlo simulation. Refer to [22] for further details. Latin Hypercube sampling has been successfully used in VLSI CAD (see, e.g., [23], [24], [25]).

4 Experiments and Results

Extensive experiments were performed to validate the proposed layer assignment approach, which targets to reduce the wire cost with timing yield consideration. The following subsections describe the experimental setups and analysis the experimental results.

4.1 Experimental Setups

The proposed stochastic layer assignment approach is implemented in C++. Sequential rounding is used to compute the integer solutions and in each step the nonlinear optimization tool IPOPT [26] is used to solve the relaxed mathematical programming problem. The approach is tested on a Pentium IV machine with 2.5GHz Quad-Core CPU and 8G memory. The experiments are performed to a set of 5000 industrial buffered nets and there are eight layers. Most buffered nets have ≤ 20 wires. The wire cost is measured by scaled wire area in this paper. However, other metric can be easily incorporated in our stochastic programming approach. In the experiments, variations are assumed to follow Gaussian distribution. Due to the usage of Monte Carlo simulation, our technique is not limited to Gaussian distribution and can handle other distribution as well. In the experiments, metal width, metal thickness, gate length and gate threshold voltage are assumed to follow normal distributions and the 3σ value for each variation source is set to 15% of the mean value. In our experiments, yield target is set to 99%. As mentioned above, Latin Hypercube sampling based Monte Carlo simulations with 200 Latin Hypercube samples are used for yield estimation. Note that after optimization, full-fledged Monte Carlo simulations with 5000 samples are used to accurately evaluate the timing yield for the layer assignment solution.

4.2 Experimental Results

Refer to Table 1 for the results. It shows the average result on these 5000 nets as well as the results on 10 selected nets at various scales. Note that the large nets among these 10 nets do not represent the typical nets in 5000 nets since most buffered signal nets are often small, i.e., have $\leq$ 20 wires in our industrial signal nets. Nominal design is obtained by solving Eqn. (3) using the nominal value of each circuit parameter and thus no variations are considered. Worst-case design is obtained using the worst-case corner, i.e., setting each capacitance and resistance to the worst case $\mu + 3\sigma$ corner (μ is the mean and σ is the standard deviation). Stochastic design is obtained from the proposed our stochastic layer assignment approach. Note that all of the above are integer programs and thus sequential rounding is involved. Note that some nets are global nets and some are local nets. Although they have the similar number of wires, they can have quite different total length and thus the cost could be very different. We make the following observations.

TABLE 1
Comparison of the layer assignment solutions by nominal design (without considering variations), worst-case design (worst-corner) and the proposed stochastic programming based design. The results on 10 selected global or local nets and the average result on 5000 nets are shown. Note that most nets in 5000 nets have size < 20.

Net		Nominal Design			Worst-Case Design			Stochastic Design					
	# Wires	Cost	Yield	CPU(s)	Cost	Yield	CPU(s)	Cost	Yield	CPU(s)	CPU w/ multi-core	Cost Red.	Speedup
Net 1	5	170.5	30.6%	0.3	227.4	100.0%	0.3	192.5	99.2%	5.6	2.3	18.1%	2.4×
Net 2	10	229.0	27.5%	0.5	289.7	100.0%	0.5	263.2	99.4%	12.9	5.1	10.1%	2.5×
Net 3	15	215.3	32.1%	1.0	280.8	100.0%	1.0	247.1	99.0%	35.7	14.5	13.6%	2.5×
Net 4	20	282.5	42.9%	1.4	382.5	100.0%	1.4	335.4	99.1%	58.4	25.7	14.0%	2.3×
Net 5	30	147.2	35.7%	2.9	189.6	100.0%	3.0	165.1	99.5%	102.2	37.6	14.8%	2.7×
Net 6	38	232.1	29.0%	4.6	294.5	100.0%	4.7	255.2	99.0%	162.0	67.3	15.4%	2.4×
Net 7	51	318.9	34.1%	5.5	419.2	100.0%	5.5	360.4	99.2%	322.8	109.4	16.3%	3.0×
Net 8	63	237.4	39.4%	5.9	302.3	100.0%	6.0	262.8	99.3%	354.2	135.2	15.0%	2.6×
Net 9	77	213.2	46.8%	6.2	281.3	100.0%	6.4	243.1	99.1%	415.3	164.7	15.7%	2.5×
Net 10	85	256.4	30.2%	7.9	337.9	100.0%	8.0	287.6	99.2%	572.4	206.3	17.5%	2.8×
Avg. on 5000 nets	18.2	228.5	35.2%	1.70	297.6	100.0%	1.70	257.2	99.2%	69.1	27.2	15.7%	2.54×

- The nominal design does not consider variations and thus its yield is always too small (35.2% on average).
- The worst-case design is too conservative about the variations and its yield is always 100% which leads significantly waste of the resources. This is clear by comparing cost results between our proposed stochastic programming approach to the worst case design.
- Our stochastic programming approach computes a good tradeoff between the nominal design and worst-case design. We are always able to closely match the yield target. Compared to nominal design, our stochastic design largely improves its yield. Compared to worst-case design, our stochastic design achieves 15.7% reduction in cost.
- The stochastic optimization is slower than computing the nominal design and worse-case design. However, one can tackle this through parallelizing our stochastic programming approach since the proposed stochastic technique is parallelization friendly as indicated in Section 3.3.3. The parallelized version of our stochastic programming technique is implemented in a quad-core computing environment. It can be seen that on average we reduce the runtime by 2.54×. It would be expected that with more cores, the speedup would be more significant.
- The average running time of the proposed algorithm in parallel is 27.2 seconds, but for some cases, the running time can be greater than 200 seconds. The reasons are: (1) the integer program taking longer time with complex nets, and (2) a large amount of γ searching iterations.

Varying the robust parameter γ, we are able to obtain different yield-cost tradeoff. A sample yield-cost tradeoff curve is shown in Figure 5 for Net5. The curve shows a tradeoff between the area and the yield. To satisfy the large yield value, more area cost are required. For this net, to achieve 98% yield, the area cost has to be greater than 165.

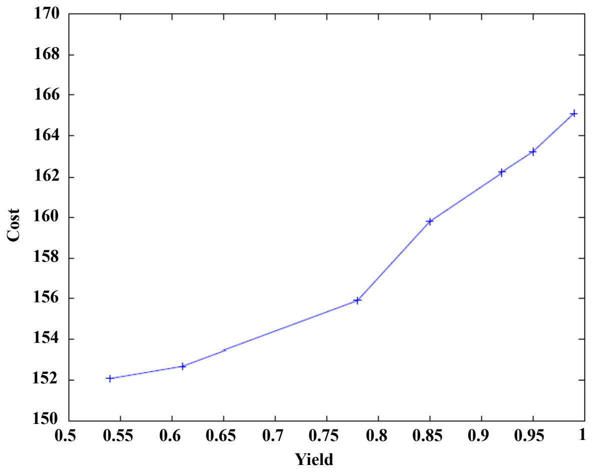


Fig. 5. Yield-Cost tradeoff curve for Net5.

5 Conclusion

This study was targeted on handling variations on process, voltage and temperature to avoid potential risks (e.g., timing violations) brought by these variations in VLSI design. A variation-aware layer assignment approach has been successfully developed with focuses on the general variation models. The proposed approach explored a systematic stochastic programming method for timing yield driven layer assignment, such that, the timing yield can be well maintained and wire cost can also be minimized. The single-stage stochastic program has been designed to explicitly control the yield, which exhibits advantages in timing yield control over its traditional two-stage counterpart. In addition, the hierarchical optimization framework allows parallel execution of the optimization process on a multi-core platform to gain significant improvement in runtime performance.

The results obtained from experiments on 5000 industrial nets demonstrate that the the proposed approach (1) can significantly improve the timing yield by 64.0% in comparison with the nominal design and (2) can reduce the wire cost by 15.7% in comparison with the worst-case design. We also observed that a quad-core implementation of our stochastic optimization approach can reduce the runtime by 2.54× in comparison with its sequential

Acknowledgement

This study was supported in part by the National Natural Science Foundation of China (Nos. 61272314, 61361120098), the Program for New Century Excellent Talents in University (NCET-11-0722), the Excellent Youth Foundation of Hubei Scientific Committee (No. 2012FFA025), the Specialized Research Fund for the Doctoral Program of Higher Education (20110145110010), the Fundamental Research Funds for Central Universities (China University of Geosciences,Wuhan, Nos. CUG120114, CUG130617, CUG140612), and Beijing Microelectronics Technology Institute under the University Research Programme (No. BM-KJ-FK-WX-20130731-0013).

References

[1] D. Sinha, N. Shenoy, and H. Zhou, "Statistical gate sizing for timing yield optimization," *In Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design (ICCAD '05)*, pp. 1037–1041, 2005.

[2] A. Davoodi and A. Srivastava, "Variability driven gate sizing for binning yield optimization," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 6, pp. 683 – 692, 2008.

[3] A. Singhee, C. Fang, J. Ma, and R. Rutenbar, "Probabilistic interval-valued computation: toward a practical surrogate for statistics inside cad tools," *In Proceedings of the 43rd annual Design Automation Conference (DAC '06)*, pp. 167–172, 2006.

[4] V. Khandelwal and A. Srivastava, "Monte-carlo driven stochastic optimization framework for handling fabrication variability," *In Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design (ICCAD '07)*, pp. 105–110, 2007.

[5] Z. Li, C. Alpert, S. Hu, T. Muhmud, S. Quay, and P. Villarrubia, "Fast interconnect synthesis with layer assignment," *In Proceedings of the 2008 international symposium on Physical design (ISPD '08)*, pp. 71 – 77, 2008.

[6] S. Hu, Z. Li, and C. Alpert, "A polynomial time approximation scheme for timing constrained minimum cost layer assignment," *In Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design (ICCAD '08)*, pp. 112–115, 2008.

[7] C. Alpert, S. Karandikar, Z. Li, G.-J. Nam, S. Quay, H. Ren, C. Sze, P. Villarrubia, and M. Yildiz, "Techniques for fast physical synthesis," *Proceedings of IEEE*, vol. 95, no. 3, pp. 573–599, 2007.

[8] R. M. D. Wu, J. Hu and M. Zhao, "Layer assignment for crosstalk risk minimization," *In Proceedings of the 2004 Asia and South Pacific Design Automation Conference (ASP-DAC '04)*, pp. 159–162, 2004.

[9] G. Xu, L.-D. Huang, D. Z. Pan, and M. D. F. Wong, "Redundant-via enhanced maze routing for yield improvement," *In Proceedings of the 2005 Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, pp. 1148–1151, 2005.

[10] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "Boxrouter 2.0: Architecture and implementation of a hybrid and robust global router," *In Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design (ICCAD '07)*, pp. 503–508, 2007.

[11] M. Cho, K. Yuan, Y. Ban, and D. Z. Pan, "Eliad: Efficient lithography aware detailed router with compact printability prediction," *In Proceedings of the 45th annual Design Automation Conference (DAC '08)*, pp. 504–509, 2008.

[12] X. Chen, T. Wei and S. Hu, "Uncertainty-Aware Household Appliance Scheduling Considering Dynamic Electricity Pricing In Smart Home," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 932-940, 2013.

[13] T. Wei, X. Chen and S. Hu, "Reliability-Driven Energy Efficient Task Scheduling for Multiprocessor Real-Time Systems," *IEEE Transactions on Computer-Aided Design (TCAD)*, vol. 30, no. 10, pp. 1569-1573, 2011.

[14] J. Lillis and C.-K. Cheng and T.-T.Y. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model," *IEEE Journal of Solid State Circuits*, vol. 31, no. 3, pp. 437–447, 1996.

[15] S. Hu, C. J. Alpert, J. Hu, S. Karandikar, Z. Li, W. Shi, and C. N. Sze, "Fast algorithms for slew constrained minimum cost buffering," *IEEE Transactions on Computer-Aided Design (TCAD)*, vol. 26, no. 11, pp. 2009–2022, 2007.

[16] A. Abou-Seido, B. Nowak, and C. Chu, "Fitted elmore delay: a simple and accurate interconnect delay model," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 7, pp. 691 – 696, 2004.

[17] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," *In Proceedings of the 41st annual Design Automation Conference (DAC '04)*, pp. 331–336, 2004.

[18] H. Chang and S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Transactions on Computer-Aided Design (TCAD)*, vol. 24, no. 9, pp. 1467–1482, 2005.

[19] J. Birge and F. Louveaux, "Introduction to stochastic programming," Springer, 1997.

[20] A.J. Kleywegt and A. Shapiro, "Stochastic Optimization, Handbook of Industrial Engineering, 3rd Edition," G. Salvendy (editor), John Wiley and Sons, 2001.

[21] S. Shah, A. Srivastava, D. Sharma, D. Sylvester, D. Blaauw, and V. Zolotov, "Discrete vt assignment adn gate sizing using a self-snapping continuous formulation," *In Proceedings of the 2005 IEEE/ACM international conference on Computer-aided design (ICCAD '05)*, 2005.

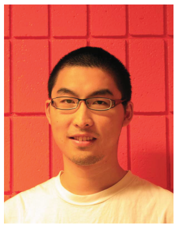
[22] K. Fang, and L. Runze, "Design and modelling for computer experiments," *CRC Press*, 2005.

[23] S. Hu and J. Hu, "Unified adaptivity optimization of clock and logic signals," *In Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design (ICCAD '07)*, pp. 125–130, 2007.

[24] A. Singhee, S. Singhal, and R. A. Rutenbar, "Practical, fast monte carlo statistical static timing analysis: why and how," *In Proceedings of the 2008 IEEE/ACM international conference on Computer-aided design (ICCAD '08)*, 2008.

[25] S.K. Tiwary and P.K. Tiwary and R.A. Rutenbar, "Generation of yield-aware Pareto surfaces for hierarchical circuit design space exploration," *In Proceedings of the 43rd annual Design Automation Conference (DAC '06)*, pp. 31–36, 2006.

[26] "https://projects.coin-or.org/ipopt."

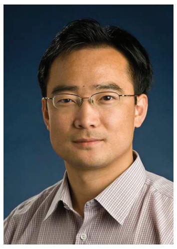


Xiaodao Chen received the B.Eng. degree in telecommunication from the Wuhan University of Technology, Wuhan, China, in 2006, the he M.Sc. degree in electrical engineering from Michigan Technological University, Houghton, USA, in 2009, and the Ph.D. in computer engineering from Michigan Technological University, Houghton, USA, in 2012.

He is currently an Assistant Professor with School of Computer Science, China University of Geosciences, Wuhan, China.

Dan Chen (M' 2002) received the B.Sc. degree in applied physics from Wuhan University, Wuhan, China, and the M.Eng. degree in computer science from Huazhong University of Science and Technology, Wuhan, China. He received the M.Eng. and the Ph.D. in computer engineering fromNanyang Technological University, Singapore.

He is currently a Professor, Head of the Department of Network Engineering, and the Director of the Scientific Computing lab with School of Computer Science, China University of Geosciences, Wuhan, China. He was a HEFCE Research Fellow with the University of Birmingham, U.K. His research interests include modelling and simulation of complex systems, neuroinformatics, and high performance computing.



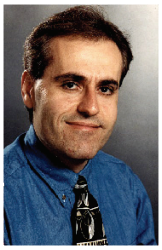
Lizhen Wang (SM' 2009) received the B.Eng. degree (with honors) and the M.Eng. degree both from Tsinghua University, Beijing, China, and the Doctor of Engineering in applied computer science (magna cum laude) from University Karlsruhe (now Karlsruhe Institute of Technology), Karlsruhe, Germany.

He is a "100-Talent Program" Professor at Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences (CAS), Beijing, China and a "ChuTian" Chair Professor at School of Computer Science, China University of Geosciences, Wuhan, China.

Prof. Wang is a Fellow of IET and Fellow of BCS.

Ze Deng received the B.Sc. degree from China University of Geosciences, the M.Eng. degree from Yunnan University, and the Ph.D. degree from Huazhong University of Science and Technology, China. He is currently an Assistant Professor with the School of Computer Science, China University of Geosciences, Wuhan, China. He is currently also a Postdoctor with the Faculty of Resources , China University of Geosciences, Wuhan, China.

Rajiv Ranjan Rajiv Ranjan is a Research Scientist and a Julius Fellow in CSIRO Computational Informatics Division (formerly known as CSIRO ICT Centre). His expertise is in datacenter cloud computing, application provisioning, and performance optimization. He has a PhD (2009) in Engineering from the University of Melbourne. He has published 62 scientific, peer-reviewed papers (7 books, 25 journals, 25 conferences, and 5 book chapters). His hindex is 20, with a lifetime citation count of 1660+ (Google Scholar). His papers have also received 140+ ISI citations. 70% of his journal papers and 60% of conference papers have been A*/A ranked ERA publication.



Albert Zomaya (F 2004)

Albert Y. Zomaya is currently the Chair Professor of High Performance Computing & Networking and Australian Research Council Professorial Fellow in the School of Information Technologies, The University of Sydney. He is also the Director of the Centre for Distributed and High Performance Computing which was established in late 2009. Professor Zomaya held the CISCO Systems Chair Professor of Internetworking during the period 2002-2007 and also was Head of school for 2006-2007 in the same school. Prior to his current appointment he was a Full Professor in the School of Electrical, Electronic and Computer Engineering at the University of Western Australia, where he also led the Parallel Computing Research Laboratory during the period 1990-2002. He served as Associate-, Deputy-, and Acting-Head in the same department, and held numerous visiting positions and has extensive industry involvement. Professor Zomaya received his PhD from the Department of Automatic Control and Systems Engineering, Sheffield University in the United Kingdom.

Shiyan Hu (SM' 2010) received the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2008.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, where he serves as the Director of the Michigan Tech VLSI CAD Research Laboratory. He was a Visiting Professor with the IBM Austin Research Laboratory, Austin, TX, in 2010. He has over 50 journal and conference publications. His current research interests include computer-aided design for very large-scale integrated circuits on nanoscale interconnect optimization, low power optimization, and design for manufacturability.

Dr. Hu has served as a technical program committee member for a few conferences such as ICCAD, ISPD, ISQED, ISVLSI, and ISCAS. He received the Best Paper Award Nomination from ICCAD 2009.