Ph.D. DISSERTATION

Timing Analysis on Interconnects and Prediction on Design Rule Violations for Synthesizing Deep Sub-micron Circuits

초미세 회로 설계를 위한 인터커넥트의 타이밍 분석 및 디자인 룰 위반 예측

BY

CHANGHO HAN

FEBRUARY 2021

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Timing Analysis on Interconnects and Prediction on Design Rule Violations for Synthesizing Deep Sub-micron Circuits

초미세 회로 설계를 위한 인터커넥트의 타이밍 분석 및 디자인 룰 위반 예측

BY

CHANGHO HAN

FEBRUARY 2021

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Timing Analysis on Interconnects and Prediction on Design Rule Violations for Synthesizing Deep Sub-micron Circuits

초미세 회로 설계를 위한 인터커넥트의 타이밍 분석 및 디자인 룰 위반 예측

지도교수 김 태 환

이 논문을 공학박사 학위논문으로 제출함

2020년 12월

서울대학교 대학원

전기컴퓨터공학부

한 창 호

한창호의 공학박사 학위 논문을 인준함

2021년 1월

위 원 장: 김 재 하

부위원장: 김 태 환

위 원: 서 광 석

위 원: 최 귀 형

위 원: 백 유

Abstract

Timing analysis and clearing design rule violations are the essential steps for taping out a chip. However, they keep getting harder in deep sub-micron circuits because the variations of transistors and interconnects have been increasing and design rules have become more complex. This dissertation addresses two problems on timing analysis and design rule violations for synthesizing deep sub-micron circuits.

Firstly, timing analysis in process corners can not capture post-Si performance accurately because the slowest path in the process corner is not always the slowest one in the post-Si instances. In addition, the proportion of interconnect delay in the critical path on a chip is increasing and becomes over 20% in sub-10nm technologies, which means in order to capture post-Si performance accurately, the representative critical path circuit should reflect not only FEOL (front-end-of-line) but also BEOL (back-end-of-line) variations. Since the number of BEOL metal layers exceeds ten and the layers have variation on resistance and capacitance intermixed with resistance variation on vias between them, a very high dimensional design space exploration is necessary to synthesize a representative critical path circuit which is able to provide an accurate performance prediction. To cope with this, I propose a BEOL-aware methodology of synthesizing a representative critical path circuit, which is able to incrementally explore, starting from an initial path circuit on the post-Si target circuit, routing patterns (i.e., BEOL reconfiguring) as well as gate resizing on the path circuit. Precisely, the synthesis framework of critical path circuit integrates a set of novel techniques: (1) extracting and classifying BEOL configurations for lightening design space complexity, (2) formulating BEOL random variables for fast and accurate timing analysis, and (3) exploring alternative (ring oscillator) circuit structures for extending the applicability of this work.

Secondly, the complexity of design rules has been increasing and results in more

design rule violations during routing. In addition, the size of standard cell keeps decreasing and it makes routing harder. In the conventional P&R flow, the routability of pre-routed layout is predicted by routing congestion obtained from global routing, and then placement is optimized not to cause design rule violations. But it turned out to be inaccurate in advanced technology nodes so that it is necessary to predict routability with more features. I propose a methodology of predicting the hotspots of design rule violations (DRVs) using machine learning with placement related features and the conventional routing congestion, and perturbating placed cells to reduce the number of DRVs. Precisely, the hotspots are predicted by a pre-trained binary classification model and placement perturbation is performed by global optimization methods to minimize the number of DRVs predicted by a pre-trained regression model. To do this, the framework is composed of three techniques: (1) dividing the circuit layout into multiple rectangular grids and extracting features such as pin density, cell density, global routing results (demand, capacity and overflow), and more in the placement phase, (2) predicting if each grid has DRVs using a binary classification model, and (3) perturbating the placed standard cells in the hotspots to minimize the number of DRVs predicted by a regression model.

keywords: Representative critical path, circuit delay prediction, process variation, BEOL, design rule violation, machine learning, placement perturbation, metaheuristic, Bayesian optimization

student number: 2001-21599

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Representative Critical Path Circuit

Due to the scaling down of transistors and interconnects, process variations are getting large and cause manufactured chips to expose a wide range of speed. Converting the fraction of too slow chips into good ones by adjusting their supply voltage, called voltage binning, is commonly applied in industry [1, 2]. In order to measure the maximum frequency of the chip, it takes a considerable time and it is not feasible in a limited time because we have to check whether the circuit is working with lots of test vectors at 2.50GHz, 2.51GHz, 2.52GHz and so on. Thus, the delay of representative circuit on a chip such as critical path replica or ring oscillator, which can be measured in a minute, is used to predict the delay of the post-Si target circuit. However, if the gap between the delays of the representative critical path circuit and the target post-Si chip is large, the chip shall be assigned to a wrong bin, resulting in a parametric yield loss. Consequently, it is highly important to install a representative critical path circuit on the target circuit such that its delay prediction error should be as minimal as possible.

Since the maximum frequency, F_{max}, of a target circuit is determined by the slowest path on the circuit, the critical path replica (CPR), which corresponds to the slowest path at nominal parameter values, is a widely acceptable candidate of representative

critical path circuit [3]. However, the CPR delay is not always the slowest one for every instance of target circuit due to process variation. As shown in Fig. 1.1, in which the blue and red dots indicate the delay of CPR and the delay of some near-critical path, respectively, the near-critical path is not the slowest one at nominal process condition while it becomes the slowest one at a certain process condition, as indicated by green rectangle.

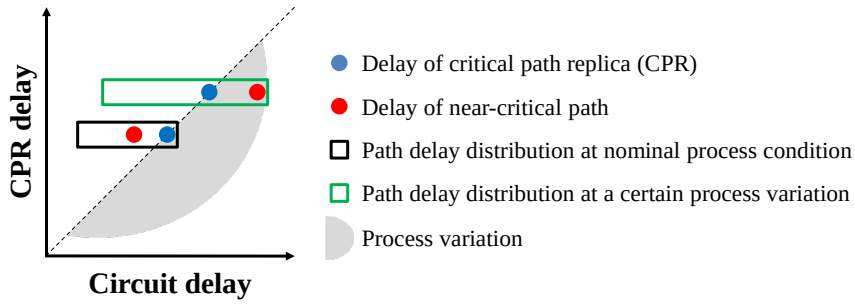


Figure 1.1: Changes of delay of critical path replica (CPR) and near-critical path at nominal and process variation conditions.

More importantly, the proportion of BEOL delay in the critical path occupies over 20% in sub-10nm technologies as shown in Fig. 1.2 and the number of metal layers increases, counting over ten layers [4, 5]. The fact that the metal layers are not so strongly correlated each other makes an accurate prediction of circuit performance hard, even causing circuit failure as Chiang *et al.* point out the potential impacting non-trivial BEOL contribution [6]. For instance, since the clock and data paths entail different BEOL configurations, their delays expose different characteristics by the BEOL variation, which may cause hold-time failure although there exists no hold-time violation under conventional BEOL corners based sign-off that assumes all BEOL layers move together to the same corner. Specifically, if the clock path is composed of upper metal layers in RC_{max} corner while the data path is composed of lower metal layers in RC_{min} corner, it may enable the data path to be faster than the clock path, causing hold-time failure.

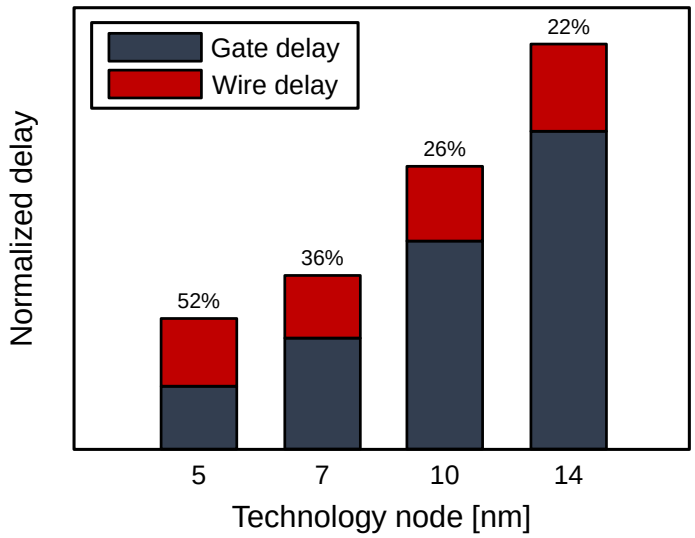


Figure 1.2: Transistor and interconnect delay of an inverter driving 150 units of contacted gate pitch of wire length and fan-out of 3. At 5-nm node technology, the contribution by interconnect becomes as significant as that by the devices [5].

If we can monitor FEOL process shift and BEOL resistance/capacitance (RC) on a chip, we can find the critical path delay through applying the variation to multiple candidates of critical path. FEOL process shift can be monitored by the pass-gate based process monitors [7]. Since special inverter structures with pass transistor are more sensitive to transistor variation than NAND and NOR gates, ring oscillators using the special inverter structures are suitable for monitoring FEOL process shift like TT, FF, SS, FS, and SF.

On the other hand, there is no easy way to accurately monitor the resistance and capacitance of metal and via layers on a chip. Even though resistance and capacitance of BEOL layers can be measured using TEGs in scribe line, the measured values are totally different from that of resistance and capacitance on the chip because the pattern densities are not identical and CMP (chemical mechanical polishing) process makes the metal height on the chip differ from that in scribe line [8]. There exists an approach

to monitor BEOL RC by using ring oscillators: Liu, Law, and Li proposed a test structure to extract resistance and capacitance of BEOL load from the frequency and I_{eff} values of ring oscillator at three modes (in-phase, out-of-phase, and quiet modes) [9]. It predicts RC of 1W1S metal reasonably well, but the prediction for 1W2S metal has large errors, i.e., 51% for resistance and 15% for capacitance. In addition, as the variation sources are diverse and some of them are hard to monitor, a new methodology to synthesize a representative critical path circuit for accurately capturing the chip performance is required.

Prior works on monitoring circuit performance have been done by either synthesizing a representative critical path circuit or a ring oscillator. Note that in case where the delay of critical path circuit is hundred picoseconds level, a signal generator and time-to-digital converter are needed [10] whereas in case where the delay of ring oscillator is a few GHz frequency, it can be lowered by frequency divider to KHz level [11]. Regarding synthesizing a representative critical path (RCP) circuit, Liu and Sapatnekar proposed an iterative gate sizing method to fine tune the nominal critical path circuit [3]. Since it is synthesized with 90nm technology, BEOL delay variation is expected to be much lower than FEOL delay variation. For synthesizing a ring oscillator, it provides advantages such as small area, easy to design, and easy to measure the characteristics like frequency and IDDQ (quiescent supply current). Chan *et al.* attempted to synthesize design-dependent ring oscillators [12], in which multiple design-dependent ring oscillators with BEOL load were designed while the BEOL load was taken from the wire-length distribution of nets in critical paths. Even though the method considers BEOL load in monitoring circuit performance, it does not take into account the configurations of multiple metal and via layers in the BEOL load formulation.

To the best of my knowledge, no conventional methods have synthesized representative critical path or ring oscillator circuit by taking into account the process variation on multiple BEOL layers. This work overcomes this limitation.

1.2 Prediction of Design Rule Violations and Placement Perturbation

Physical design is a process to generate a physical layout from circuit netlists. It is composed of several steps such as importing design, floorplan, powerplan, placement, clock tree synthesis, routing and chip finish. The objectives of physical design are to clear design rule violations (DRVs) and meet the constraints like timing and power. Because design rules are the geometric constraints not to cause open or short of patterns in transistors and interconnects, the design rule violations should be cleared for taping out a chip. However, there are tens of thousands of design rules in advanced technologies as shown in Fig. 1.3 and routing resources are not enough because the height of standard cell has been reduced and it becomes 6 tracks in 5nm technology.

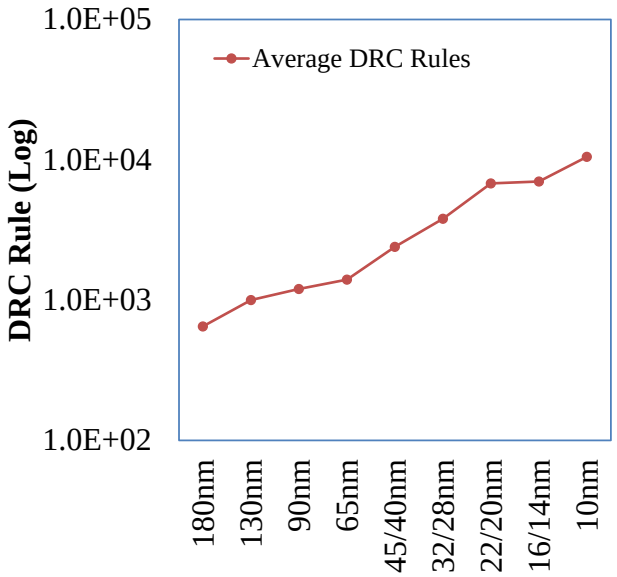


Figure 1.3: Growth in DRC rules [17].

The lack of pin access point in the standard cell makes the routing harder and causes lots of design rule violations. Prediction of routability in placement phase can help to reduce the number of DRVs in routed layout by avoiding the placement which

will cause DRVs. In order to predict the routability, the commercial P&R tools use routing congestion by calculating capacity, demand and overflow in each global routing cell but it turned out to be inaccurate and incomplete in advanced technologies [18]. The routing congestion map in pre-routed layout is totally different from the map of DRVs in routed layout. Thus, it is necessary to predict routability with more features.

Some noticeable works using machine learning have been reported. They introduced features determined by placed cells such as pin density, pin proximity, cell density and so on. The placement related features as well as routing congestion are used to train machine learning model and the hotspots including design rule violations are predicted by the model. They focused on improving the prediction accuracy with various machine learning algorithm and features. Chan *et al.* trained the machine learning model using SVM (Support Vector Machine) with RBF (Radial Basis Function) kernel with placement-derived parameters such as pin density, minimum proximity of any pair of pins, number of complex cells, sum of incoming and outgoing hyperedges, number of buried nets, arithmetic and geometric mean values of placement-base Rent parameter, the worst signal transition time of all pins at the worst corner and the smallest values of the worst negative slack of setup time of any pin within the grid [18]. Tabrizi *et al.* used RUSBoost for imbalanced data classification that combines data sample and boosting because the number of hotspots including DRVs is much less than that of non-hotspots and they are imbalanced data [19]. Islam *et al.* introduced random forest algorithm to predict design rule violations [20]. Yu *et al.* changed the pin patterns into image and CNN (Convolutional Neural Network) is used to train the model [21]. Liang *et al.* proposed customized CNN to improve the prediction accuracy [22].

The next step of routability prediction is to reduce the number of design rule violations using placement optimization. Chan *et al.* proposed machine learning predictor-guided routability optimization algorithm [23]. White spaces are calculated in local windows around hotspots and cells are moved incrementally to redistribute white

space. Yu *et al.* proposed machine learning model-guided placement [24]. The trained model is adopted to inference the DRV probabilities using the image of pins in placement cells and gives more space between cells until the probability is less than predefined threshold value. However, pin patterns are not the only one to result in design rule violations. Kahng *et al.* proposed mesh like placement and perturbations by two neighbor-swap moves to measure routing capacity [25]. Even though the pin shapes for all placements are exactly same, the number of design rule violations increases as the swapping number increases. It means that it is not enough to predict routability using only placement related features. The routing related features like capacity, demand and overflow should be considered to predict the routability. However, it takes time to extract the routing related features and dynamic programming based approach to explore all cases is not feasible. Thus, global optimization algorithms should be applied to this problem because the optimum value can be obtained from some samples.

This work presents machine learning guided placement perturbation for routing congested circuits. Placement related features and routing related features are combined to predict the routability, and the number of design rule violations is reduced by placement perturbation with global optimization algorithms in a limited time.

1.3 Contributions of This Dissertation

In this dissertation, synthesis of representative critical path circuits and prediction of design rule violations are studied, which brings better yield and reduced chip area.

In Chapter 2, I propose a BEOL-aware methodology of synthesizing a representative critical path circuit which is able to incrementally explore, starting from an initial path circuit on the post-Si target circuit, routing patterns (i.e., BEOL reconfiguring) as well as gate resizing on the path circuit, devising the following novel techniques: (1) extracting and classifying BEOL configurations for lightening design space complexity, (2) formulating BEOL random variables for fast and accurate timing analysis, and

(3) exploring alternative (ring oscillator) circuit structures for extending the applicability of my work. In short, through experiments with industry circuits, it is shown that the synthesis framework is able to reduce the prediction error by 54% and 19% on average over that using the conventional critical path replica and using the conventional method exploiting gate sizing only, respectively.

In Chapter 3, I propose a methodology to predict the hotspots which include design rule violations using machine learning and reduce the number of design rule violations using placement perturbation of standard cells in the hotspots. Conventional prediction method by using local routing congestion has its limitation of low prediction accuracy. Thus, recent machine learning based researches used more features to predict hotspots for the better accuracy. After predicting the hotspots, a method to reduce the number of design rule violations should be applied but there is no concrete method. I propose a method to predict and reduce the design rule violations by combining three techniques such as binary classification, regression and global optimization. The framework which is composed of hotspots prediction and placement perturbation reduces the number of design rule violations by 22% on average.

Chapter 2

Methodology for Synthesizing Representative Critical Path Circuits reflecting BEOL Timing Variation

2.1 Motivation

Because the maximum frequency or delay of post-Si circuit instances can't be measured in a limited time, a representative critical path circuit is necessary for the voltage binning.

Let $\mathcal{S} = \{C_1, \cdots, C_m\}$ be the set of all post-Si circuit instances of a target circuit C_{target} with a representative critical path circuit denoted as $rcp(C_{target})$, and $rcp(C_i)$ be the circuit instance corresponding to $rcp(C_{target})$ in $C_i \in \mathcal{S}$.

Then, the performance prediction error, ϵ_i, on $C_i \in \mathcal{S}$ is defined as

$$\epsilon(C_i) = |delay(rcp(C_i)) - delay(C_i)|/delay(C_i) \tag{2.1}$$

where $delay(rcp(C_i))$ represents the delay of the representative path in circuit instance C_i and $delay(C_i)$ indicates the longest path delay in C_i.

Then, since it is practically impossible to find the exact value of $delay(C_i)$ for every circuit instance in $\mathcal{S}$ in a limited time, I predict it by measuring $delay(rcp(C_i))$. Thus, it is very important to synthesize $rcp(C_{target})$ such that $\epsilon(C_i)$ for every $C_i \in \mathcal{S}$

should be as small as possible. Two metrics commonly used to evaluate the quality of circuit $rcp(C_{target})$ are the maximum prediction error (ϵ_{max}) and the correlation coefficient (ρ) between $delay(rcp(C_i))$ and $delay(C_i)$, as formally expressed as

$$\epsilon_{max} = max\{\epsilon(C_1), \epsilon(C_2), \cdots, \epsilon(C_m)\} \tag{2.2}$$

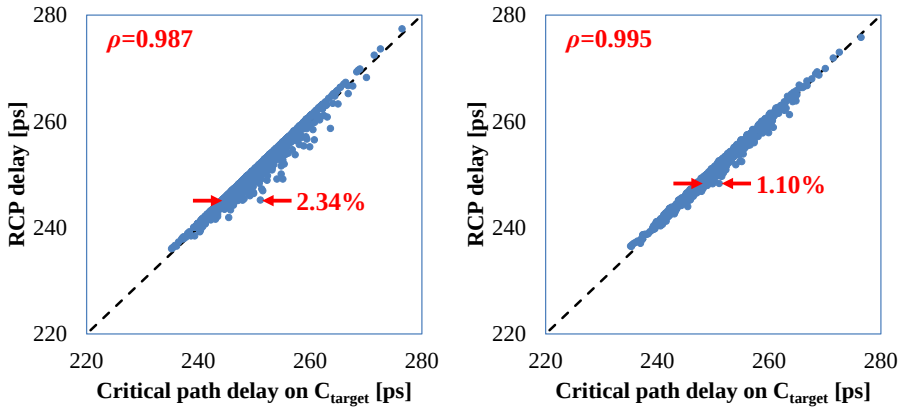
$$\rho = \frac{E[(\delta(rcp) - \mu_{\delta(rcp)})(\delta(c) - \mu_{\delta(c)})]}{\sigma_{\delta(rcp)} \cdot \sigma_{\delta(c)}} \tag{2.3}$$

where $\delta(rcp)$ and $\delta(c)$ are the set of the delay values on circuits $rcp(C_1), \cdots, rcp(C_m)$ and the set of the longest delay values on circuit instances $C_1, \cdots, C_m$ in $\mathcal{S}$, respectively. μ and σ indicate the average and standard deviation of each of the sets $\delta(rcp)$ and $\delta(c)$.

By performing Monte Carlo simulation on target circuit C_{target} and its representative critical path circuit rcp, reflecting process variation, it is possible to predict the values of ϵ_{max} and correlation coefficient ρ. Thus, the key challenge is to synthesize $rcp(C_{target})$ such that the value of ρ is as high as possible while the value of ϵ_{max} is as small as possible or is below a certain limit.

One noticeable work is that Liu and Sapatnekar [3] proposed to generate circuit $rcp(C_{target})$ by iteratively tuning the size of every gate namely considering FEOL variation on the critical path replica in C_{target}, in which they tried to maximize the value of correlation coefficient ρ. To see how much the method is effective in a deep submicron technology, I applied the method to circuit MEM_CTRL that is a design component in OpenCores [15], on which I performed synthesis, placement, and routing at 3GHz with 7nm technology of partner foundry. The critical path replica taken from MEM_CTRL is composed of 25 (gate-to-gate) stages and the interconnect delay occupies 8.9% of the critical path delay when all parameters are set to nominal values.

I ran Monte Carlo simulation with 1,000 samples (i.e., equivalently 1,000 post-Si circuit instances) for the representative circuit rcp(MEM_CTRL) synthesized by [3] by varying the values of FEOL and BEOL parameters. Fig. 2.1(a) shows the scatter plot



(a) (b)

Figure 2.1: Monte Carlo simulation results by varying the values of FEOL and BEOL parameters. (a) Scatter plot of performance prediction by using $rcp(mem\_ctrl)$ produced by gate sizing [3] (b) Scatter plot of performance prediction by using $rcp(mem\_ctrl)$ produced by randomly tuning BEOL configurations as well as gate sizing.

of the delay pairs of the 1000 circuit instances of target circuit MEM_CTRL and the rcp(MEM_CTRL), from which it is taken that ϵ_{max} = 2.34% and $\rho = 0.987$. On the other hand, Fig. 2.1(b) shows the scatter plot of delay pairs of another 1000 circuit instances of the target circuit MEM_CTRL and the rcp(MEM_CTRL synthesized by considering BEOL and FEOL variations, from which it is taken that ϵ_{max} = 1.10% and $\rho = 0.995$. Clearly, the improvement implies that tuning both of the gate size and interconnect on the critical path replica considering FEOL and BEOL variations is very necessary for accurate performance prediction of post-Si circuits in deep submicron technologies. In the following, I propose a systematic methodology of synthesizing a representative critical path circuit of C_{target} such that the value of correlation coefficient ρ is maximized while constraining the ϵ_{max} value to a certain bound.

2.2 Definitions and Overall Flow

This section describes the overall flow of generating a representative critical path circuit $rcp(C_{target})$ together with defining BEOL configuration, BEOL reconfiguring, and gate resizing, which are the essential ingredients for generating $rcp(C_{target})$.

Definition 1. BEOL configuration *from g_i (a driving logic cell) to g_{i+1} (one of its driven cells) is defined to the ordered sequence of metal/via layer numbers with metal length information corresponding to the detailed routing between g_i and g_{i+1}.*

Fig. 2.2 shows a simple critical path from flip-flop f_1 to flip-flop f_2 which consists of three BEOL configurations *BEOL1*, *BEOL2*, and *BEOL3*, showing routing details of *BEOL2*. I collect BEOL configurations on a set of critical and near-critical routing paths in a table form as shown in Table 2.1 where for example, $M2 : 5$ indicates metal-2 layer with length of 5um and $V2$ indicates via-2 layer.

Let $\mathcal{B}(C_{target})$ be the set of all BEOL configurations extracted from top K slowest (flip-flop to flop-flop) paths[1] in C_{target} to which the placement and routing have already been applied. I prepare $\mathcal{B}(C_{target})$, which I call *BEOL configuration library*, as a pre-processing step in the framework of synthesizing $rcp(C_{target})$.

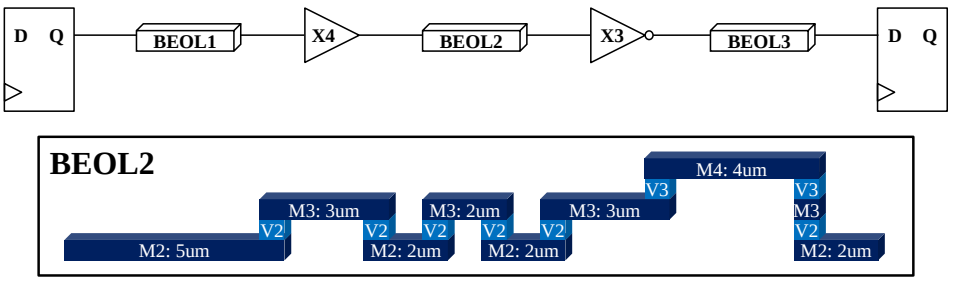


Figure 2.2: An example of critical path consisting of three BEOLs, showing the routing details of *BEOL2*.

Definition 2. BEOL reconfiguring *on $rcp(C_{target})$ in C_{target} refers to an incremen-*

[1] K is set to 1000 in this work.

Table 2.1: BEOL configuration in a table form corresponding to *BEOL2* in Fig. 2.2.

Driving cell : Driven cell	Routing details
BUF_X4 : INV_X3	M2:5 V2 M3:3 V2 M2:2 V2 M3:2 V2 M2:2 V2 M3:3 V3 M4:4 V3 M3:0 V2 M2:2

tally updated circuit of $rcp(C_{target})$ *in a way that one of BEOLs in* $rcp(C_{target})$ *is replaced with a BEOL with the same type and size of driving and driven cells in BEOL library* $\mathcal{B}$*. I use* $\mathcal{R}_{BEOL}$ *to represent the set of all possible BEOL reconfigurings on* $rcp(C_{target})$.

Fig. 2.3 shows a critical path on which *BEOL2* is tried to be reconfigured using n alternatives *BEOL2-1*, *BEOL2-2*, $\cdots$, *BEOL2-n*, one at a time.

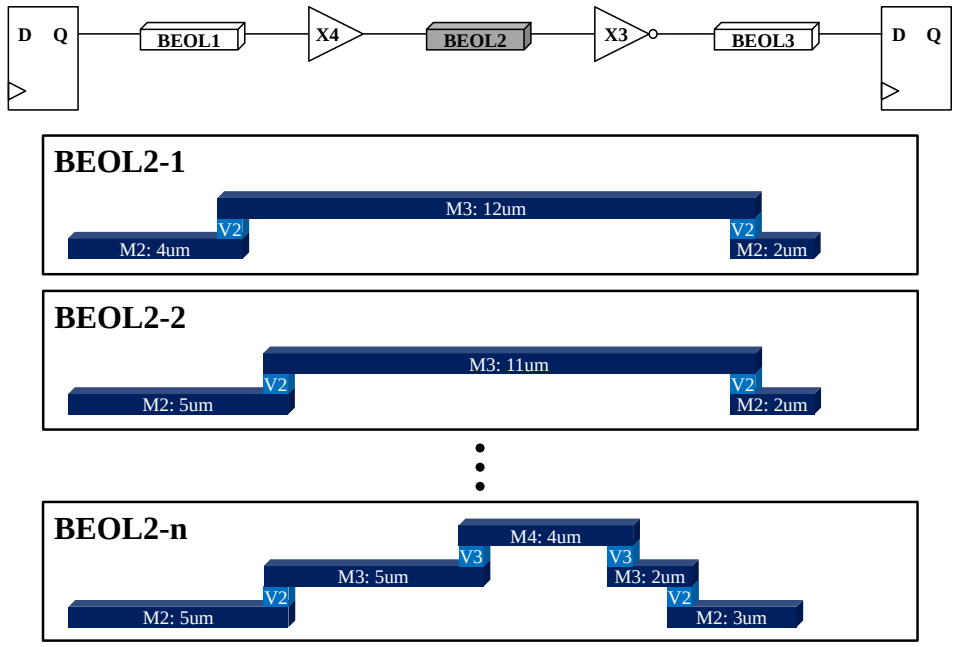


Figure 2.3: An example of BEOL reconfiguring for *BEOL2*. If there are n distinct BEOL configurations in $\mathcal{B}$ such that the left and right end cells are BUF_X4 and INV_X3, exactly n BEOL reconfigurations will be performed for *BEOL2*.

Definition 3. Gate resizing *on $rcp(C_{target})$ in C_{target} refers to an incrementally updated circuit of $rcp(C_{target})$ in a way that one of gate cells in $rcp(C_{target})$ is replaced with one-level upsized or downsized cell of the same type in the cell library. I use $\mathcal{R}_{FEOL}$ to indicate the set of all possible gate resizings on $rcp(C_{target})$.*

Fig. 2.4 shows a critical path where BUF_X4 is attempted to be upsized or downsized. Note that when the left end cell (BUF_X4) is resizing to produce an alternative $rcp(C_{target})$, the right end cell (INV) remains intact, and vice versa.

(a)

(b)

Figure 2.4: An example of gate resizing for BUF_X4 in Fig 2.2. (a) Upsizing. (b) Downsizing.

Given a layout of target circuit C_{target}, the proposed exploration algorithm, called RCP-exp, is to synthesize a representative critical path circuit $rcp(C_{target})$ which leads to the highest ρ value in *Eq.2.3* while ϵ_{max} in *Eq.2.2* does not exceed ϵ_{bound}. Precisely, as a pre-processing step, RCP-exp builds a BEOL configuration library from the layout of C_{target}, and tentatively assumes a critical path replica in C_{target} as $rcp(C_{target})$. Then, in the main step, RCP-exp improves the ρ value of $rcp(C_{target})$ while satisfying the ϵ_{max} constraint by iteratively tuning the gate cells and routing paths on $rcp(C_{target})$ through performing gate up/down sizing and BEOL reconfiguring. The pseudo-code in Algorithm 1 shows the algorithmic flow of my RCP-exp.

Algorithm 1 RCP-exp: The Proposed Algorithmic Flow of Synthesizing Representative Critical Path $rcp(C_{target})$

Input: C_{target} with layout information, K, T, ϵ_{bound}

Output: $rcp(C_{target})$

// Pre-processing step

Build BEOL configuration library $\mathcal{B}$ from top K slowest paths in C_{target};
Update BEOL configuration library $\mathcal{B}$; // Sec. 2.3.1
if (initial rcp from C_{target}) **then**
    Set $rcp(C_{target})$ = a critical path replica in C_{target};
else
    Set $rcp(C_{target})$ = a ring oscillator of NAND gates in T stages; // Sec. 2.3.4

// Main step: tuning FEOL and BEOL

while (1) **do**
    Apply Monte Carlo simulation to $rcp(C_{target})$; // Sec. 2.3.2
    Compute ρ of $rcp(C_{target})$;
    Set ρ_{best} = ρ;
    Extract $\mathcal{R}_{BEOL}$ from $rcp(C_{target})$ and $\mathcal{B}$;
    Extract $\mathcal{R}_{FEOL}$ from $rcp(C_{target})$ and cell library $\mathcal{L}$;
    Apply Monte Carlo simulation to every $c \in \mathcal{A} = \mathcal{R}_{BEOL} \cup \mathcal{R}_{FEOL}$;
    Compute ρ and ϵ_{max} values of every $c \in \mathcal{A}$;
    Select a circuit $c' \in \mathcal{A}$ with the largest ρ and $\epsilon_{max} \leq \epsilon_{bound}$;
    if ($c' \neq \phi$ and $\rho > \rho_{best}$) **then**
        Set $rcp(C_{target})$ = c';
    else
        Exit-loop
Return ($rcp(C_{target})$, ρ_{best})

For the pre-processing step in Algorithm 1, top K slowest paths are selected from C_{target} by using static timing analysis at nominal condition. I extract BEOL configurations of K paths by using a script in P&R tool to create a BEOL configuration library $\mathcal{B}$ composed of pairs of driving and driven cells with the routing details, as illustrated in Table 1. In order to reduce the design space complexity, I resized BEOL configuration library $\mathcal{B}$ by applying the designation-and-removal process discussed in Sec. 2.3.1. The library size is adjusted by Δ_{limit}. I use two options for initial setting of $rcp(C_{target})$: (1) the first option is to take an initial rcp from C_{target}. Precisely, I set the slowest path i.e. the critical path replica to the initial $rcp(C_{target})$; (2) the second option is to set a ring oscillator composed of NAND gates to the initial $rcp(C_{target})$.

For the main step in Algorithm 1, the *while*-loop starts from the initial $rcp(C_{target})$ obtained from the pre-processing step. A statistical netlist of $rcp(C_{target})$ is generated and is run by MonteCarlo simulation with BEOL RC random variables and V_{th} variation of transistors described in Sec. 2.3.2. Since SPICE simulation is too slow, it is not affordable to run MonteCarlo simulation with SPICE, taking over 14 hours to run the simulation with SPICE on even a single path of 25 stages with 10,000 samples. Instead, by using statistical timing analysis, I generate a model of path delay and run MonteCarlo simulation on that model. (It runs about 1,000 times faster than that of using SPICE simulation.)

I compute the correlation coefficient ρ of $rcp(C_{target})$, which is then set to the value of ρ_{best}, followed by performing the five sub-tasks (line: 11 - 19). (line: 11) I extract BEOL configurations in $rcp(C_{target})$. (Example of BEOL configuration is illustrated in Figure 4.) I also extract, from BEOL library, BEOL configurations to be reconfigured, as illustrated in Figure 5; (line: 12) I take cells like BUF_X4 and INV_X3 from $rcp(C_{target})$ and cells to be resized like BUF_X3, BUF_X5 for BUF_X4 from cell library $\mathcal{L}$; (line: 13) I perform Monte Carlo simulation on every combination of $\mathcal{R}_{BEOL}$ and $\mathcal{R}_{FEOL}$; (line: 14-15) I compute the correlation coefficients and prediction errors, from which I choose the path circuit (c') with the largest correlation

coefficient under the prediction error constraint. (line: 16 - 19) if the correlation coefficient is larger than the current best one, $rcp(C_{target})$ is reset to c' and I repeat the loop. Otherwise, I terminate the loop and return $rcp(C_{target})$ and ρ_{best}.

The time complexity of the *while-loop* in RCP-exp is bounded by $O(|\mathcal{A}| \cdot M)$ where $\mathcal{A} = \mathcal{R}_{BEOL} \cup \mathcal{R}_{FEOL}$ and M is the number of Monte Carlo simulations for each circuit instance in $\mathcal{A}$. Thus, to reduce the time complexity, it is essential (*task 1*) *to reduce the size of* $\mathcal{R}_{BEOL}$ and (*task 2*) *to apply the simulation on corner cases as many as possible*. In addition, to extend the applicability of RCP-exp, it is necessary (*task 3*) *to explore more general representative circuit structures independently of* C_{target}. The following section provides the details of my solutions to the three tasks.

2.3 Techniques for BEOL-Aware RCP Generation

This section describes three techniques for BEOL-aware RCP generation. The first one is clustering BEOL configurations, the objective of which is to reduce time complexity by removing similarly behaviored configurations and maintaining a small-sized configuration for each cluster. The second one is formulating statistical BEOL random variables. I propose a method of generating random variables from the variation of physical parameters like width, height, and space on each metal layer. The random variables of resistance and capacitance are used to run Monte Carlo simulation. The last one proposes a method of exploring ring oscillator circuit structure for rcp. It is an alternative option to synthesize rcp by tuning gate size and BEOL configurations on a rather simple structure.

2.3.1 Clustering BEOL Configurations

For a pair of BEOL configurations B_i and B_j in library $\mathcal{B}$, I define a difference measure $\Delta(B_i, B_j)$:

$$\Delta(B_i, B_j) = \alpha \cdot \sum_{k \in L} |l(metal(k, B_i) - l(metal(k, B_j)| + (1 - \alpha) \cdot |n\_via(B_i) - n\_via(B_j)| \quad (2.4)$$

where $l(metal(k, \cdot)$ and $n\_via(\cdot)$ represent the total length of the metals in layer-k and the total number of vias in the corresponding BEOL configuration, respectively. L is the set of metal layers and α is a weighting factor in between 0 and 1.

For example, for the BEOLs in Fig 2.3, when $\alpha = 0.9$, Δ(*BEOL2-1*, *BEOL2-2*) = 0.9($|(4+2) - (5+2)|$ + $|12 - 11|$) + 0.1($|2 - 2|$) = 1.8 while Δ(*BEOL2-1*, *BEOL2-n*) = 0.9($|(4 + 2) - (5 + 3)|$ + $|12 - (5 + 2)|$ + $|0 - 4|$) + 0.1($|2 - 4|$) = 9.9 + 0.2 = 10.1. Thus, based on the $\Delta(\cdot, \cdot)$ value, *BEOL2-1* prefers clustering with *BEOL2-2* to clustering *BEOL2-n*.

I apply a greedy method to group BEOL configurations in $\mathcal{B}$: Randomly designate a BEOL configuration (say B_i) from $\mathcal{B}$ and then remove all BEOL configurations whose $\Delta(B_i, \cdot)$ values are less than a certain limit Δ_{limit}; Iteratively, process the designation-and-removal process until no removal can be found in $\mathcal{B}$.

2.3.2 Formulating Statistical BEOL Random Variables

I develop a method of generating statistical random variables for BEOL resistance (R) and capacitance (C) from variation of physical parameters. As shown in Fig. 2.5, a metal layer involves six physical parameters w, h, d_l, d_r, t_u, and t_d, from which R and C values on a metal of unit-length can be calculated by field solver, even it takes a long computation time.

BEOL typically has five corners $Nominal$, C_{max}, C_{min}, RC_{max}, and RC_{min}. (For simplicity, this work uses $Nominal$, C_{max}, and C_{min}.) I associate the corners with physical parameters by formulating them into second-order polynomials: for $Nominal$, I set all parameters to their nominal values; for C_{max}, C is assigned to the largest value and R is assigned to the smallest value, as a result, w and h each has

the statistical maximum values i.e., μ+3σ while d_l, d_r, t_u, and t_d each has the statistical minimum values i.e., μ-3σ; for C_{min}, its physical parameter values become the opposite values of that in C_{max}.

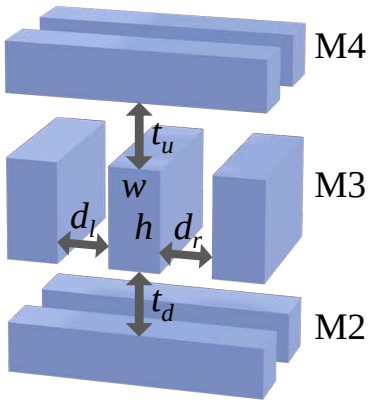


Figure 2.5: Physical parameters affecting resistance (R) and capacitance (C) of BEOL metal.

To be precise, I am required to formulate the capacitance on the left-hand, right-hand, upper, and lower sides of a metal. For example, the capacitance C_{left} can be expressed by γ_{left}, d_l, and h while the resistance R is simply by w and h as shown:

$$C_{left} = \eta_{left}\frac{h}{d_l} \tag{2.5}$$

$$R = \frac{\gamma}{w \cdot h} \tag{2.6}$$

in which η and γ indicate the dielectric permittivity and metal resistivity, respectively.

I introduce effective parameters $\eta_{e,left}$ and γ_e, and formulate them using the values of R, C, and physical parameters at RC corners. More precisely, if $\eta_{e,left}$ is a function, $f(\cdot)$, of d_l and h, and γ_e is a function, $g(\cdot)$, of w and h, C_{left} and R can be expressed

in terms of w, h, and d_l as

$$\eta_{e,left} = f\Big(\frac{d_l}{h}\Big) \tag{2.7}$$

$$C_{left} = \eta_{e,left}\frac{h}{d_l} = f\Big(\frac{d_l}{h}\Big)\frac{h}{d_l} \tag{2.8}$$

$$\gamma_e = g\left(w \cdot h\right) \tag{2.9}$$

$$R = \frac{\gamma_e}{w \cdot h} = \frac{g\left(w \cdot h\right)}{w \cdot h} \tag{2.10}$$

I formulate $\eta_{e,left}$ and γ_e into second-order polynomial regressions:

$$\eta_{e,left} = \beta_2\Big(\frac{d_l}{h}\Big)^2 + \beta_1\Big(\frac{d_l}{h}\Big) + \beta_0 \tag{2.11}$$

$$\gamma_e = \beta_2'\left(w \cdot h\right)^2 + \beta_1'\left(w \cdot h\right) + \beta_0' \tag{2.12}$$

Then, C_{left} and R can be expressed in terms of physical parameters:

$$C_{left} = \beta_2\Big(\frac{d_l}{h}\Big) + \beta_1 + \beta_0\Big(\frac{d_l}{h}\Big)^{-1} \tag{2.13}$$

$$R = \beta_2'\left(w \cdot h\right) + \beta_1' + \beta_0'\left(w \cdot h\right)^{-1} \tag{2.14}$$

C_{right}, C_{top}, and C_{bottom} can be similarly formulated in terms of physical parameters. Like this, I can produce all the five polynomials for R, C_{left}, C_{right}, C_{top}, and C_{bottom} on each metal layer and one polynomial for R on each via.

The flowchart for generating the six BEOL random variables for R and C is shown in Fig. 2.6. Fig. 2.7(a) and Fig. 2.7(b) show the relation between physical parameters and effective parameter in *Eq.2.11*, and the scatted plot of random variables R and C, respectively.

I use a statistical static timing analysis on BEOL configurations using the random variables of R and C for corner cases. For example, Table 2.2 shows an original netlist and the corresponding statistical netlist, in which resistance between nodes A and B is 100ohm and capacitance on A and B is 0.001F and 0.002F, respectively. I can see that A and B are on $M2$ since \$lvl=13 in SPEF. Thus, I can generate a statistical netlist by multiplying random variables $M2R$ and $M2C$ to the resistance and capacitance

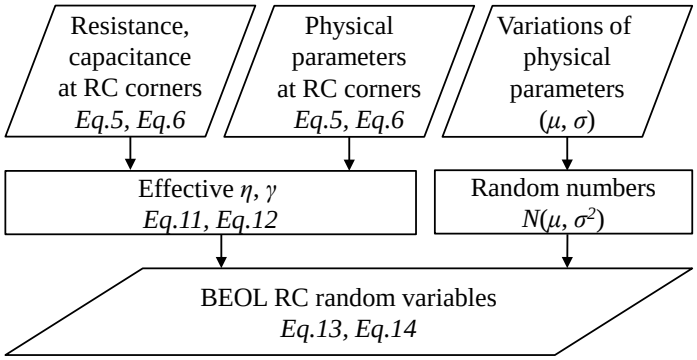


Figure 2.6: Flowchart for generating BEOL RC random variables.

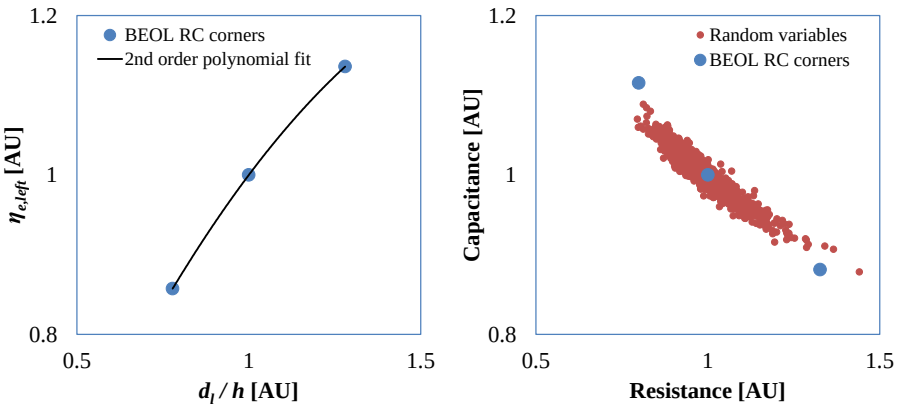


Figure 2.7: (a) Physical parameters vs. effective parameter η_e at three RC corners $Nominal$, C_{max} and C_{min}. (b) The scatted plot of random variables R and C.

Table 2.2: Statistical netlist.

Original	R1 A B 100 C1 A 0 0.001 C2 B 0 0.002
SPEF	// *LAYER_MAP // *13 M2 *CAP 1 *7626 0.001 2 *7661 0.002 *RES 1 *7626 *7661 100 // $lvl=13
Statistical	R1 A B 100*M2R C1 A 0 0.001*M2C C2 B 0 0.002*M2C

values in nominal condition. When the statistical netlist is ready, I can insert BEOL variation and V_{th} variation of transistors to the statistical netlist formulation.

2.3.3 Delay Modeling

In order to get the delay distribution within process variations, I developed delay modeling method. Elmore delay measure is an upper bound on the actual 50% delay of an RC tree response [13, 14]. As the input signal rise time increases, the actual delay approaches the Elmore delay. Therefore, the stage delay can be a function of Elmore delay as follows.

$$T_{stage} = \alpha T_{Elmore} \tag{2.15}$$

where α is varied by the input rise time and less than 1.0.

Assuming α is a constant within process variations, I can make T_{stage} is a function irrelevant to α.

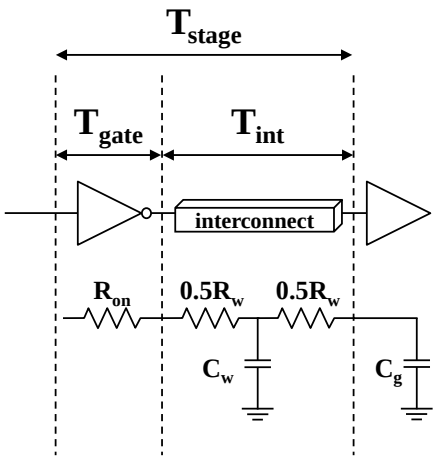


Figure 2.8: Stage delay is sum of gate delay and interconnect delay.

As you can see in the Fig 2.8, T_{stage} is sum of T_{gate} and T_{int}. If I model the gate as a resistance, T_{Stage} is as follows.

$$T_{Stage} = \alpha R_{on}(C_w + C_g) + \alpha R_w(0.5C_w + C_g) \tag{2.16}$$

$$= T_{gate} \qquad + T_{int} \tag{2.17}$$

Since I assume α is a constant within variations, T_{gate} and T_{int} with variations are obtained by their nominal values and the variation parameters such as V_{th}, R, C and so on.

$$\frac{T_{gate,var}}{T_{gate,norm}} = \Big(\frac{R_{on,var}}{R_{on,norm}}\Big)\Big(\frac{C_{w,var} + C_{g,norm}}{C_{w,norm} + C_{g,norm}}\Big) \tag{2.18}$$

$$\frac{T_{int,var}}{T_{int,norm}} = \Big(\frac{R_{w,var}}{R_{w,norm}}\Big)\Big(\frac{0.5C_{w,var} + C_{g,norm}}{0.5C_{w,norm} + C_{g,norm}}\Big) \tag{2.19}$$

Because the variation of gate input capacitance (C_g) is much less than that of wire capacitance, I use the nominal value of input capacitance. R_{on} is a linear function of $\Delta V_{th,n}$ and $\Delta V_{th,p}$, and the function is obtained by T_{gate} in five FEOL corners with BEOL nominal values. Therefore, $T_{gate,var}$ becomes a function of its nominal value,

$\Delta V_{th,n}$, $\Delta V_{th,p}$ and wire capacitance, and T_{int} becomes a function of wire resistance and capacitance. Now, I can get the delay distribution within process variations.

I compared the delay modeling with SPICE simulation. When a critical path is composed of 25 stages, delay modeling is about 1,200 times faster than SPICE simulation. I found that the error of delay model was less than 2% as indicated in Fig 2.9.

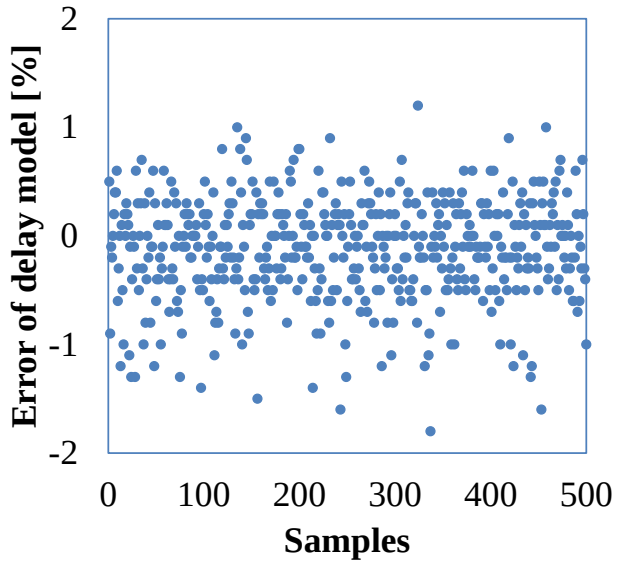


Figure 2.9: Scatter plot of delay error calculated by the model for 500 MC samples.

2.3.4 Exploring Ring Oscillator Circuit Structures

Rather than synthesizing $rcp(C_{target})$ starting from a critical path replica (CPR) in C_{target}, I propose another option to use a ring oscillator as an initial rcp. I use three simple ring oscillator structures as shown in Fig. 2.10 in which each ring oscillator is composed of gates of the same type and size, and BEOLs of an identical BEOL configuration.

The changes in performing RCP-exp using a ring oscillator (RO) are the followings:

1. In the pre-processing step, RCP-exp uses an RO with much fewer number of

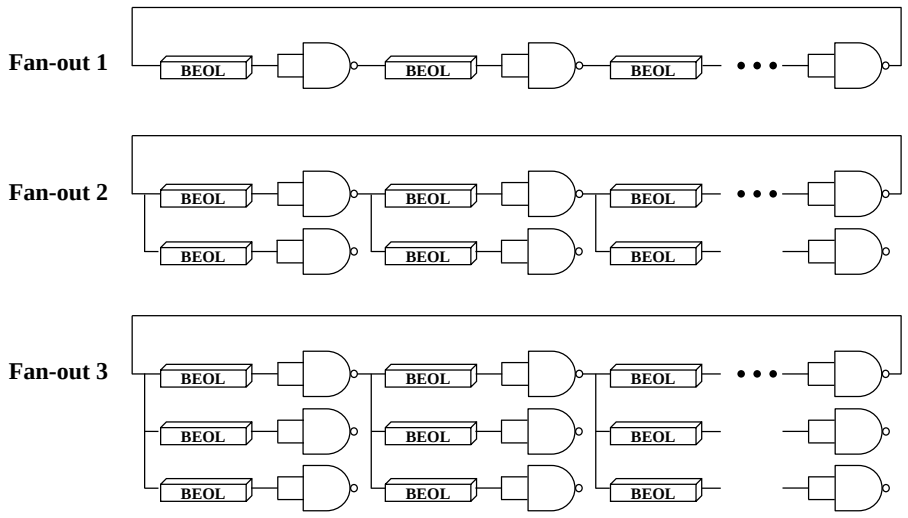


Figure 2.10: The proposed ring oscillator (RO) structures which can be used as an initial $rcp(C_{target})$, independently of target circuits.

stages (gate-to-gate), usually ~ 10 than that using CPR, usually ~ 25. Nevertheless. The ρ value is maintained while ϵ_{max} value can be proportionally scaled.

2. The *while-loop* in RCP-exp will be executed three times, one for each ring oscillator structure in Fig. 2.10.

3. RCP-exp produces $\mathcal{R}_{FEOL}$ by equally up/down sizing all gates in RO at once.

4. RCP-exp produces $\mathcal{R}_{BEOL}$ by uniformly replacing all BEOL configurations in RO at once using $\mathcal{B}$.

2.4 Experimental Results

I implemented my RCP-exp and the conventional method [3] in Python, and applied them to a set of design blocks taken from OPENCORES using 7nm technology of partner foundry. I performed a sequence of tasks namely synthesis, placement, routing, RC extraction, and static timing analysis (STA) by using Synopsys Design Compiler, IC

Compiler-II, StarRC, and Primetime. In all experiments, I used delay modeling with 10,000 samples, from which I delete the worst 100 samples to meet 99% of delay predictions pessimism. Table 2.3 summarizes the information of the target design blocks I tested. (I found that a considerable amount of gates in every target circuit drive just a single pin (i.e., Fan-out 1 gates) and Fan-outs 1, 2, and 3 gates occupy around 80% of all gates in circuits.)

Table 2.3: Target circuits.

Circuit	#Inst	#Net	*Clkp*	Fan-out			
				1	2	3	≥ 4
AC97_CTRL	6,150	6,178	300ps	66%	7%	6%	21%
DES_PERF	12,881	13,005	310ps	47%	16%	15%	22%
MEM_CTRL	4,703	4,766	380ps	56%	14%	9%	21%
PCI_BRIDGE32	10,280	10,452	340ps	64%	11%	5%	20%
USB_FUNCT	8,104	8,264	340ps	54%	20%	7%	19%

Table 2.4 summarizes, for each target circuit, the number of stages on the critical path (CP) at nominal condition and the portion of BEOL interconnect delay.

Table 2.4: Summary of critical paths of target circuits.

Circuit	Critical path	
	#stage	Portion of BEOL delay
AC97_CTRL	6	20.1%
DES_PERF	12	5.5%
MEM_CTRL	25	8.9%
PCI_BRIDGE32	26	14.3%
USB_FUNCT	22	9.4%

• **Synthesizing $rcp(C_{target})$ using critical path replica (CPR) in C_{target}**: Table 2.5 shows a comparison of the values of correlation coefficient (ρ) under maximum prediction error constraint ($\epsilon_{max} \leq \epsilon_{bound}$) among the initial critical path replica (CPR) and

rcp circuits produced by [3] considering FEOL variation only and my RCP-exp considering BEOL as well as FEOL variations. The constraints in Case1, Case2 and Case3 of RCP-exp are $0.8\epsilon_{bound}$, $1.0\epsilon_{bound}$ and $1.2\epsilon_{bound}$, respectively while for Case4 of RCP-exp, minimizing the maximum prediction error is the primary objective rather than constraint.

Note that for AC97_CTRL, the method in [3] failed to find an rcp better than CPR for Cases 1, 2, and 4 since its CPR consists of just 6 stages, which eventually leads to provide little room for improving rcp by performing gate resizing alone. To put it another way, if CPR of a target circuit has a small number of stages, tuning the routing paths on the CPR by remapping BEOL configurations is relatively more effective than gate resizing.

Table 2.6 shows a comparison of the values of maximum prediction error. My method is able to reduce the prediction error by 54% and 19% on average over that using the conventional critical path replica and using the conventional method exploiting gate sizing only, respectively.

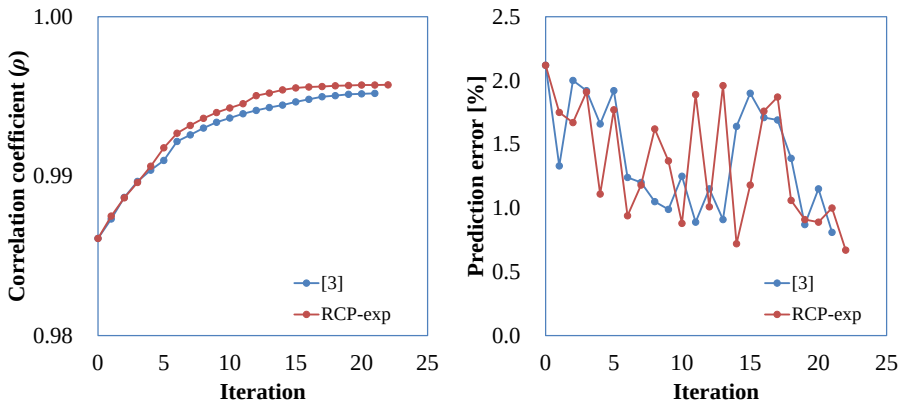
Fig. 2.11 shows the changes of correlation coefficient (ρ) and maximum prediction error (ϵ) for $rcp(mem\_ctrl)$, which is iteratively refined by the conventional method [3] and RCP-exp, corresponding to Case2 in Table 2.5. It is shown that at each iteration, the ρ value monotonically increases while the ϵ value fluctuates. At the last iteration, the ρ value arrives at the largest and the ϵ value at the least. The comparison of the two ρ curves in Fig. 2.11(a) clearly indicate that RCP-exp considering BEOL variation outperforms the conventional method which does not take into account BEOL variation at all. Fig. 2.12 shows the scatted plots of Monte Carlo simulation samples for the initial CPR in MEM_CTRL, $rcp(mem\_ctrl)$ produced by [3], and $rcp(mem\_ctrl)$ by RCP-exp.

Table 2.5: Comparison of the values of correlation coefficient (ρ) under maximum prediction error constraint ($\epsilon_{max} \leq \epsilon_{bound}$) among the initial critical path replica (CPR) and rcp circuits produced by [3] considering FEOL variation only and my RCP-exp considering BEOL as well as FEOL variations. The constraint for Case1, Case2 and Case3 are $0.8\epsilon_{bound}$, $1.0\epsilon_{bound}$ and $1.2\epsilon_{bound}$ respectively. Case 4 is to minimize the maximum prediction error.

Circuit	CPR (*Initial*)		[3] (*Considering FEOL variation only*)				RCP-exp (*Considering FEOL and BEOL variations*)			
			Case1	Case2	Case3	Case4	Case1	Case2	Case3	Case4
	ρ	ϵ_{bound}	ρ	ρ	ρ	ρ	ρ	ρ	ρ	ρ
AC97_CTRL	0.914	6.70%	0.914 (1.000x)	0.914 (1.000x)	0.929 (1.016x)	0.914 (1.000x)	0.921 (1.008x)	0.931 (1.019x)	0.931 (1.019x)	0.916 (1.002x)
DES_PERF	0.990	1.63%	0.991 (1.001x)	0.992 (1.002x)	0.992 (1.002x)	0.990 (1.000x)	0.992 (1.002x)	0.993 (1.003x)	0.993 (1.003x)	0.992 (1.002x)
MEM_CTRL	0.986	2.12%	0.995 (1.009x)	0.995 (1.009x)	0.995 (1.009x)	0.986 (1.000x)	0.996 (1.010x)	0.996 (1.010x)	0.996 (1.010x)	0.989 (1.003x)
PCI_BRIDGE32	0.982	2.40%	0.990 (1.008x)	0.991 (1.009x)	0.991 (1.009x)	0.983 (1.001x)	0.994 (1.012x)	0.994 (1.012x)	0.992 (1.010x)	0.983 (1.001x)
USB_FUNCT	0.991	1.58%	0.993 (1.002x)	0.993 (1.002x)	0.993 (1.002x)	0.991 (1.000x)	0.995 (1.004x)	0.995 (1.004x)	0.996 (1.005x)	0.993 (1.002x)
Impr.	-	-	1.004x	1.004x	1.008x	1.000x	1.007x	1.010x	1.009x	1.002x

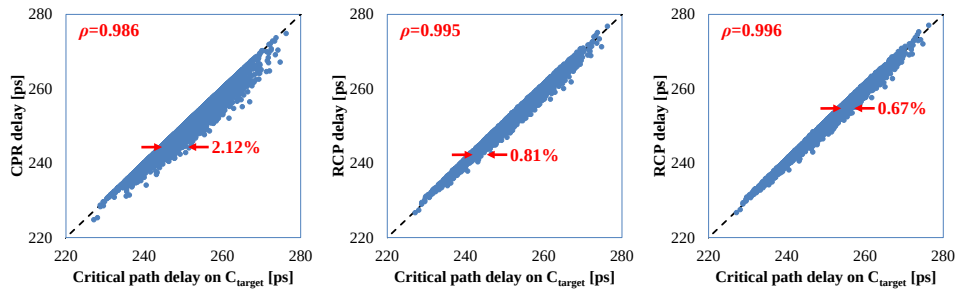
Table 2.6: Comparison of the values of maximum prediction error of Case 2.

Circuit	CPR (*Initial*)		[3] Case2	RCP-exp Case2		
	ρ	ϵ_{bound}	ϵ	ϵ	*Impr. over CPR*	*Impr. over [3]*
AC97_CTRL	0.914	6.70%	6.70%	3.60%	-46%	-46%
DES_PERF	0.990	1.63%	0.98%	0.88%	-46%	-10%
MEM_CTRL	0.986	2.12%	0.81%	0.67%	-68%	-17%
PCI_BRIDGE32	0.982	2.40%	1.11%	1.03%	-57%	-7%
USB_FUNCT	0.991	1.58%	0.86%	0.75%	-53%	-13%
Avg.	-	-	-	-	-54%	-19%



(a) (b)

Figure 2.11: The changes of correlation coefficient (ρ) and maximum prediction error (ϵ) for $rcp(mem\_ctrl)$ which is iteratively refined by the conventional method [3] and RCP-exp.



(a) (b) (c)

Figure 2.12: Monte Carlo simulation results. (a) Critical path replica (CPR) in MEM_CTRL. (b) $rcp(mem\_ctrl)$ produced by [3]. (c) $rcp(mem\_ctrl)$ produced by RCP-exp.

I analyzed the yield improvement by RCP-exp. Let us assume that the required delay for MEM_CTRL is 253.88ps which is the sum of mean and half of standard deviation (μ+0.5σ) of circuit delay, shown in Fig. 2.13. If voltage binning is not applied, the yields produced by the critical path replica, [3], and my method are 69.41%, 69.41%, and 69.41%, respectively. On the other hand, if I increase the supply voltage to shorten the delay by 8.1% when the RCP delay exceeds 253.88ps (post-Si instances in bin2), through analysis of the statistical distribution obtained by simulation, it is found that the corresponding yields become 95.97%, 99.17%, and 99.40%. As a result, my method improves yield by 3.43% over the critical path replica method and by 0.23% over the method in [3]. As the portion of BEOL delay increases, the yield improvement would be significant. For circuit AC97_CTRL, the portion of BEOL delay occupies about 20% and the correlation coefficient increases from 0.914 to 0.931. I found through distribution analysis that it leads to yield improvement of 3.38% over the method in [3]. The yield improvement for all the test circuits is summarized in Table 2.7.

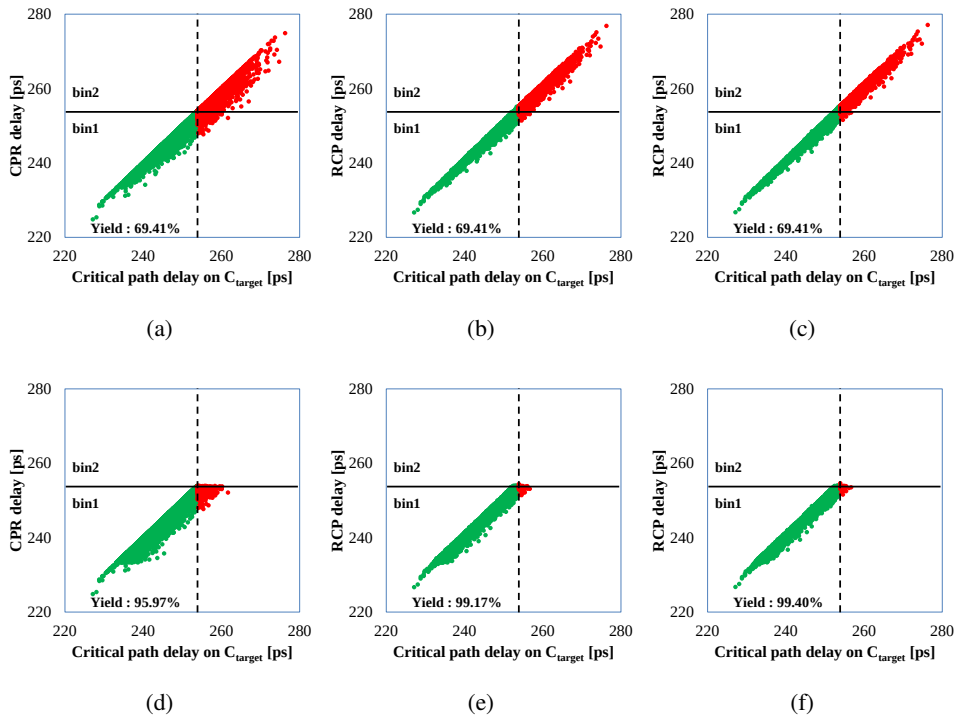


Figure 2.13: Green dots indicate the post-Si instances whose RCP delay is lower than the required delay while red dots indicate the post-Si instances whose RCP delay exceeds the required delay. (a) Critical path replica (CPR) in MEM_CTRL when voltage binning is not applied. (b) $rcp(mem\_ctrl)$ by [3] when voltage binning is not applied. (c) $rcp(mem\_ctrl)$ by RCP-exp when voltage binning is not applied. (d) Critical path replica (CPR) in MEM_CTRL when voltage binning is applied. (e) $rcp(mem\_ctrl)$ by [3] when voltage binning is applied. (f) $rcp(mem\_ctrl)$ by RCP-exp when voltage binning is applied.

Table 2.7: Yield improvement by critical path replica (CPR), rcp circuits produced by [3] and my RCP-exp when the required delay is μ+0.5σ of each circuit delay and voltage binning is applied.

Circuit	CPR $(Initial)$	[3]	RCP-exp		
		Case2	Case2		
	Yield	Yield	Yield	*Impr. over CPR*	*Impr. over [3]*
AC97_CTRL	95.31%	95.31%	98.69%	3.38%	3.38%
DES_PERF	97.86%	98.97%	99.16%	1.30%	0.19%
MEM_CTRL	95.97%	99.17%	99.40%	3.43%	0.23%
PCI_BRIDGE32	95.21%	98.74%	99.11%	3.90%	0.37%
USB_FUNCT	96.96%	99.43%	99.59%	2.66%	0.16%
Avg.	-	-	-	2.93%	0.87%

As explained in Sec. 2.3.1, I reduce the size of library $\mathcal{B}$ by clustering the BEOL configurations. The number of configurations is controlled by using Δ_{limit}. When α = 0.9 and Δ_{limit} is enumerated from 0 to 8, the runtime is reduced up to 17% for MEM_CTRL and 53% for AC97_CTRL without worsening the correlation coefficient, as shown in Fig. 2.14. Since for the maximum value Δ_{limit} of 4, the corresponding correlation coefficients are not degraded for the two testcases, I applied the Δ_{limit} value uniformly to all testcases.

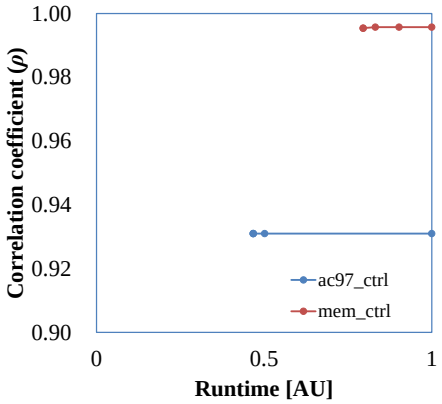


Figure 2.14: Runtime reduction by the technique in Sec. 2.3.1, in which BEOL configurations are reduced by controlling Δ_{limit} value.

I also evaluated the results by changing the value of K (i.e., the number of critical paths selected from C_{target}). Initially, I set the value of K to 1000 and gradually decrease the value as long as the correlation coefficient is not worsen. As revealed in Fig. 2.15, the value can be down to 200.

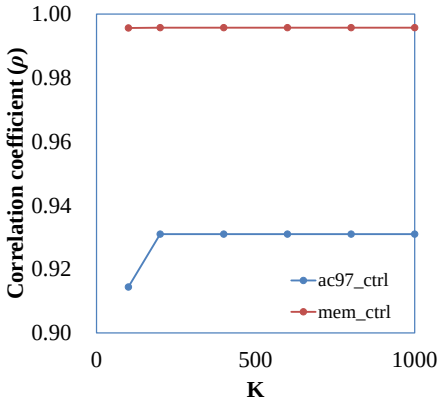


Figure 2.15: The changes of correlation coefficients as the value of K varies. The curves show that at most top 200 slowest paths suffice to get the best correlation coefficient.

• **Synthesizing $rcp(C_{target})$ using ring oscillator (RO)**: Table 2.8 shows a compar-

ison of the values of correlation coefficient (ρ) under maximum prediction error constraint ($\epsilon_{max} \leq \epsilon_{bound}$) among the initial critical path replica (CPR) and rcp circuit produced by my RCP-exp. (The constraints in Case1, Case2 and Case3 of RCP-exp are $0.8\epsilon_{bound}$, $1.0\epsilon_{bound}$ and $1.2\epsilon_{bound}$, respectively while for Case 4 of RCP-exp, minimizing the maximum prediction error is the primary objective rather than constraint.) I can see that RCP-exp improves over CPRs for all target circuits except AC97_CTRL, for which its CPR has a small number of stages and its sensitivity looks far away from resizing NAND gates in RO.

Table 2.8: Comparison of the values of correlation coefficient (ρ) under maximum prediction error constraint ($\epsilon_{max} \leq \epsilon_{bound}$) among the initial critical path replica (CPR) and ring oscillators produced by my RCP-exp considering BEOL as well as FEOL variations. The constraint for Case1, Case2 and Case3 are $0.8\epsilon_{bound}$, $1.0\epsilon_{bound}$ and $1.2\epsilon_{bound}$ respectively. Case 4 is to minimize the maximum prediction error.

Circuit	CPR (*Initial*)		RCP-exp (*Considering FEOL and BEOL variations*)			
			Case1	Case2	Case3	Case4
	ρ	ϵ_{bound}	ρ	ρ	ρ	ρ
AC97_CTRL	0.914	6.70%	0.866 (0.947x)	0.868 (0.950x)	0.868 (0.950x)	0.862 (0.943x)
DES_PERF	0.990	1.63%	0.994 (1.004x)	0.994 (1.004x)	0.994 (1.004x)	0.990 (1.000x)
MEM_CTRL	0.986	2.12%	0.992 (1.002x)	0.992 (1.002x)	0.992 (1.002x)	0.991 (1.001x)
PCI_BRIDGE32	0.982	2.40%	0.985 (0.995x)	0.985 (0.995x)	0.985 (0.995x)	0.981 (0.991x)
USB_FUNCT	0.991	1.58%	0.995 (1.004x)	0.995 (1.004x)	0.995 (1.004x)	0.994 (1.003x)
Impr.	-	-	0.990x	0.991x	0.991x	0.988x

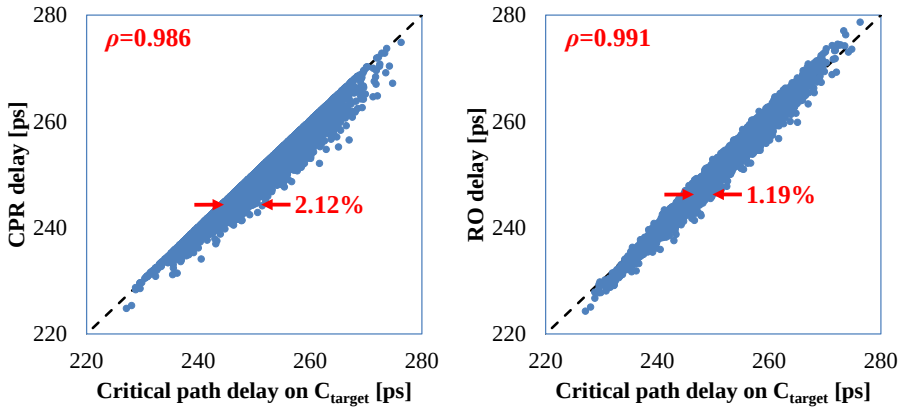


Figure 2.16: Monte Carlo simulation results. (a) Critical path replica (CPR) in MEM_CTRL. (b) $rcp(mem\_ctrl)$ produced by Case4 in RCP-exp.

Fig. 2.16 compares the scatter plots of Monte Carlo simulations for CPR and rcp produced by Case4 in RCP-exp for target circuit MEM_CTRL. Specifically, for circuit MEM_CTRL I explored 26,196 candidates of RO based rcp where I considered 4 sizes of NAND gate, 3 types of Fan-out, and 2,183 BEOL configurations extracted from MEM_CTRL. Table 2.9 shows the largest value of correlation coefficient under maximum error constraint produced by Case2 in RCP-exp for every RO corresponding to the combinations of 4 gate sizes and 3 Fan-out types for target circuit MEM_CTRL. ('-' indicates that RCP-exp could not find rcp that meets the maximum error constraint.) I can see that the best rcp is the RO which uses the BEOL configuration labeled $\#1542$, NAND of size X2, and Fan-out of 1. Fig 2.17 shows the details of BEOL configuration $\#1542$ of the best representative RO obtained in Table 2.9. Finally, Table 2.10 summarizes the best rcp (RO) structures for all target circuits, from which I can see that the BEOL configurations in the layout of all target circuits are highly sensitive to $M2$, $M3$, and up to $M4$. Furthermore, though Fan-out 1 is the majority in all target circuits, the most appropriate Fan-outs for representative RO circuits are $2\sim3$.

Table 2.9: The BEOL configuration and correlation coefficient value of the best rcp circuits produced by Case2 in RCP-exp for various combinations of gate sizes and Fan-out types for target circuit MEM_CTRL.

	Fan-out	Gate	BEOL	ρ	ϵ_{max}
CPR	-	-	-	0.986	2.12%
RCP-exp	1	NAND2_X1	-	-	-
		NAND2_X2	#1542	**0.992**	2.01%
		NAND2_X3	#1282	0.988	1.46%
		NAND2_X4	-	-	-
	2	NAND2_X1	-	-	-
		NAND2_X2	#31	0.988	1.95%
		NAND2_X3	#2094	0.991	2.08%
		NAND2_X4	#1103	0.991	1.74%
	3	NAND2_X1	-	-	-
		NAND2_X2	-	-	-
		NAND2_X3	#31	0.990	1.55%
		NAND2_X4	-	-	-

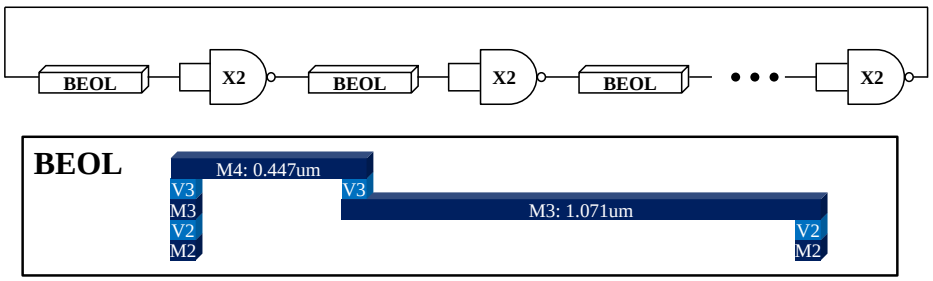


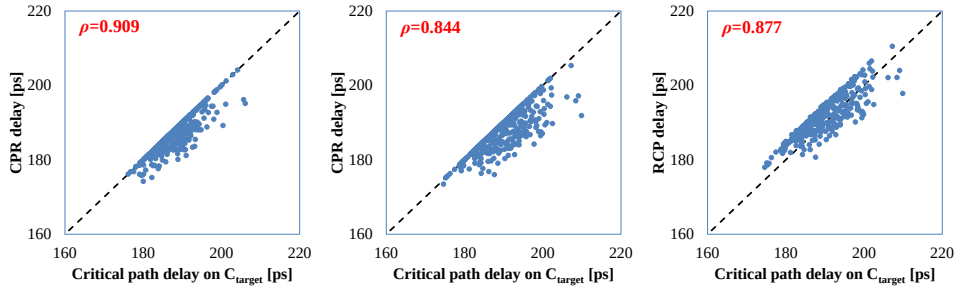
Figure 2.17: The representative RO structure for MEM_CTRL produced by Case2 in RCP-exp.

Table 2.10: The best rcp ring oscillator structures synthesized by my RCP-exp.

Circuit	Fan-out	Gate	Top metal
AC97_CTRL	3	NAND2_X4	M4
DES_PERF	2	NAND2_X1	M4
MEM_CTRL	1	NAND2_X2	M4
PCI_BRIDGE32	3	NAND2_X4	M4
USB_FUNCT	3	NAND2_X2	M4

2.5 Further Study on Variations

In this work, only process variations are considered. When the portion of BEOL variations in the critical paths is increasing, methodologies for synthesizing RCP are proposed. Because voltage is also one of variation sources in the manufactured chips, the extended availability of this work under process and voltage variations is presented in this section. I assume that the variation of supply voltage is 5% of target and it follows uniform distribution. The experiment with AC97_CTRL is performed using Monte-Carlo simulation with 500 samples. As shown in Fig 2.18, the correlation coefficient is 0.909 under process variations and it becomes 0.844 when the voltage variation is also considered. RCP under process and voltage variations are produced by RCP-exp and the correlation coefficient becomes 0.877. From the experiment result, I can see that the methodologies in this work can be extended to RCP generation with more variation sources.



(a) (b) (c)

Figure 2.18: Monte Carlo simulation results under process and voltage variations. (a) Critical path replica (CPR) in AC97_CTRL under process variations. (b) Critical path replica (CPR) in AC97_CTRL under process and voltage variations. (c) $rcp(ac97\_ctrl)$ by RCP-exp under process and voltage variations.

Chapter 3

Methodology for Reducing Routing Failures through Enhanced Prediction on Design Rule Violations in Placement

3.1 Motivation

In advanced technology nodes, it takes more than a week to complete physical design of a circuit including millions of instances. Routing occupies about 50% of total runtime of physical design when there is no design rule violation but the routing runtime increases exponentially as the number of design rule violations increases. The runtime of physical design with a sub-block of CPU, which has about 360,000 gates, is shown in Fig. 3.1. The portion of routing runtime is 56% when the number of design rule violations (DRVs) is less than 100. It increases rapidly as the number of DRVs exceeds 1,000 while the sum of runtime of floorplan, powerplan, placement and clock tree synthesis is slightly changed. The portion becomes 88% when the number of DRVs is more than 150,000. If it is predicted before routing is performed, designs whose predicted DRVs are more than a certain number can be filtered out and the computing resources can be saved.

When the P&R tools perform placement, the objective function is to minimize

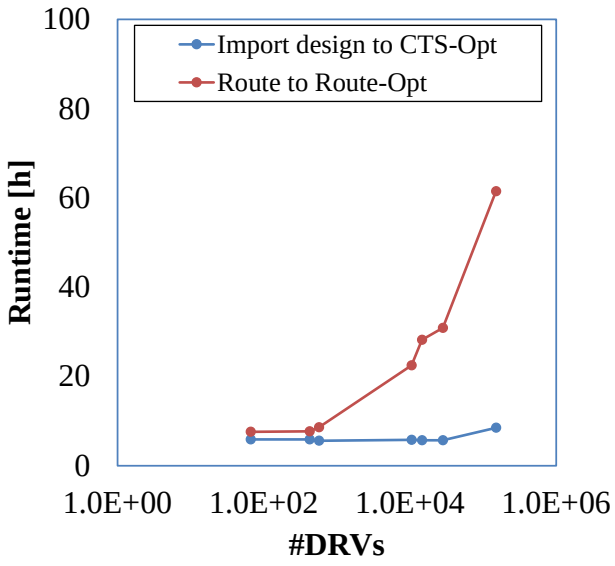


Figure 3.1: Runtime of physical design using a sub-block of CPU which is composed of 360,000 gates.

wirelength and maximize routability. In order to predict wirelength and routability for the pre-routed layout, global routing has been used. However, the congestion obtained by global routing turned out to inaccurate [23]. Actual hotspots including design rule violations are totally different from the predicted hotspots using routing congestion. Thus, it is necessary to enhance the prediction accuracy. Various methods to predict the hotspots using machine learning have been reported [18][19][20][21][22][23][24]. Some of them used only placement related features like pin density, and the others used placement related features as well as routing related features. They improved the prediction accuracy by using various machine learning algorithms such as support vector machine (SVM), RUSBoost, random forest, convolutional neural network (CNN) and customized CNN. All works predict whether the local window has DRVs using binary classification. The limitation is that they can't predict the number of DRVs.

As a next step of hotspot prediction, the placement optimization using space adjustment in the hotspots was proposed [23][24]. They also used binary classification model

to determine if the local window is hotspot or not, and give more spaces between cells in each hotspot to reduce the probability that there are DRVs in the hotspot. As I mentioned before, the prediction using binary classification model lets us know the probability but we don't know how many DRVs are existed in the local window. In some cases, binary classification based placement optimization can't reduce the number of DRVs. For example, if moving cell in the left direction can't make the probability less than 50% and moving cell right can't do as well, cell is not moved in either directions because the probability after cell movement indicates that there are still DRVs in the hotspot. However, the number of DRVs can be reduced by moving cell. This is the first limitation of placement optimization through binary classification. The other limitation of the prior works is the size of prediction window. The height of prediction window was equal to the cell height so that the placement perturbation is fast. However, interaction between rows is not considered and it can make the routability worse when the vertical connection is changed and it causes routing congestion. Because the minimum window that commercial P&R tool uses for the routing is 10x10 of standard cell height, the routability should be considered in the larger window.

I propose a methodology to predict the hotspots using binary classification through machine learning, and perform placement perturbation in the hotspots until the predicted number of design rule violation is minimized. The number of DRVs is predicted by regression through machine learning and the global optimization algorithms are used to find the optimal placement. Placement related features like pin density, pin proximity and more, are extracted and combined with global routing parameters such as capacity, demand and overflow. The machine learning models are improved by hyperparameter tuning with Bayesian optimization.

3.2 Overall Flow

This section describes the overall flow of reducing routing failures through enhanced prediction on design rule violations in placement phase. The flow is composed of feature extraction, hotspot prediction and placement optimization to minimize the predicted number of DRVs. I assume that a design is routable when the number of DRVs is less than 200. Because the number of DRVs is predicted, it is possible to filter out a design which is doomed to be routing failure. Decision of filtering out is also included in the overall flow.

The conventional P&R flow is composed of importing design, floorplan, powerplan, placement, post-placement optimization, clock tree synthesis (CTS), post-CTS optimization, route and post-route timing optimization. In the conventional flow, the routing failure can't be determined in placement phase and it is not possible to filter out a design doomed to be routing failure because the number of DRVs can't be predicted. In the proposed P&R flow, the features like pin density, pin proximity, capacity, demand, overflow and more, are extracted when post-CTS timing optimization is finished. And then the number of DRVs is predicted using pre-trained machine learning model to decide filtering out. I set the number doomed to be routing failure to 1000. When the number of DRVs are more than 1000, it is filtered out because it can't be routable even if placement optimization is applied. The filtered design should be run P&R flow again from importing design by modifying the P&R variables like frequency, initial utilization and tool options. If the predicted number is less than 150, it proceeds routing as conventional P&R flow does. When the predicted number is between 150 and 1000, placement perturbation is applied. The comparison of conventional P&R flow and proposed P&R flow is shown in Fig. 3.2.

When we focus on the placement perturbation, the flow is composed of two steps as shown in Fig. 3.3. Firstly, the hotspots which are predicted to include DRVs, are determined by pre-trained binary classification model. And then placement perturbation

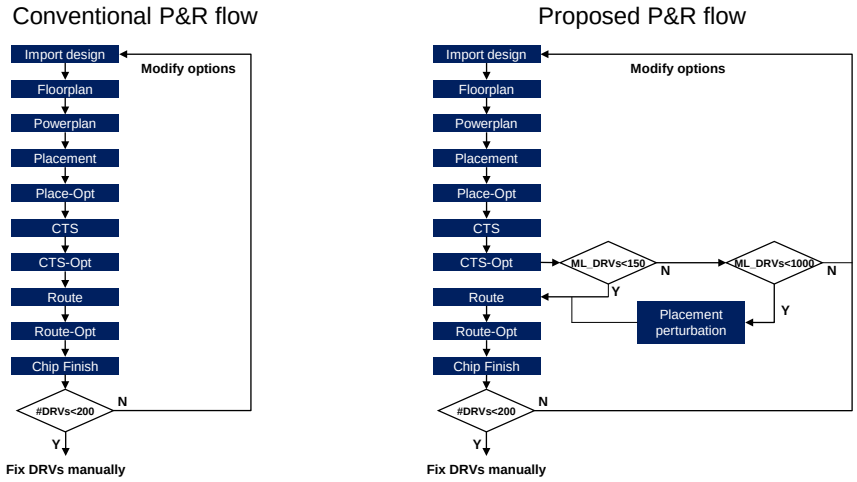


Figure 3.2: Comparison of conventional P&R flow and proposed P&R flow.

of cells in all hotspots of entire layout is started, in which the objective is to minimize the number of DRVs in entire layout. Because the global routing for feature extraction takes time, it is not possible to explore every placement perturbation. Thus, it is necessary to use global optimization algorithm to find the placement to minimize the number of DRVs in a limited time. Once the optimized placement is found, routing is performed to check if the actual DRVs are reduced or not. If the error of regression model is 0%, the predicted DRVs are exactly same as the actual DRVs. However, machine learning model is not perfect and there must be a gap between predicted and actual DRVs. The gap can be minimized by improving accuracy of the regression model.

3.3 Techniques for Reducing Routing Failures

3.3.1 Binary Classification

In order to predict hotspots which include DRVs, it is necessary to generate binary classification model using machine learning. There are lots of machine learning al-

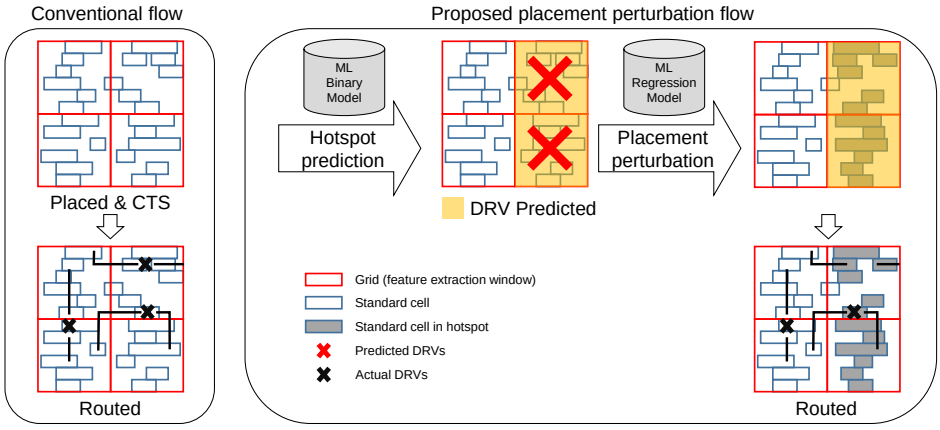


Figure 3.3: Comparison of conventional flow and proposed placement perturbation flow. The proposed flow is composed of two steps such as hotspot prediction and placement perturbation of cells in the hotspots.

gorithms such as linear regression, logistic regression, SVM, random forest, boosting, multi-layer perceptron and so on. I selected two algorithms of boosting and multi-layer perceptron (MLP) because they outperform the others in some test cases. Because the machine learning model should be used to predicted hotspots in unseen design, model training including itself is not prohibited. For example, if there are 7 circuits like A, B, C, D, E, F and G, a model to predict DRVs of A is trained by B, C, D, E, F and G. For the training and validation, 80% of data are used for the training and the rest are used for the validation. To avoid overfitting, early stopping is applied. That is to say, the model is saved when the validation loss is minimized. Logloss is used as a loss function.

To select the optimal model, the exploration of model architecture and hyperparameter tuning are performed. In multi-layer perceptron, the network structure is very important to optimize the model. The structure is defined by the number of layers, the number of nodes in each layer and the regularization. In this work, 5 number of layers and 6 nodes are explored. Layers can be 1, 2, 3, 4 and 5, and nodes can be 16, 32, 64, 128, 256 and 512. For the regularization, three methods such as original (no regular-

ization is applied), batch normalization and dropout, are compared. Boosting depends on the hyperparameters. The optimal hyperparameters to minimize the loss function are found by Bayesian optimization. Each machine learning algorithm generates 90 models and the best one is chosen. The training procedure is shown in Fig. 3.4.

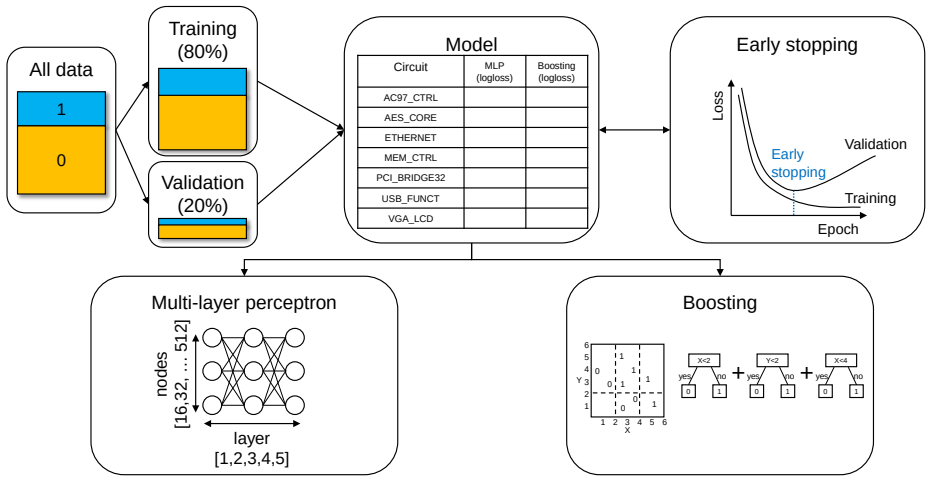


Figure 3.4: The training procedure for the binary classification.

3.3.2 Regression

Boosting and multi-layer perceptron are also used to predict the number of design rule violation. The procedure of training is similar to that of binary classification. The only difference is that root mean squared error (RMSE) is used as a loss function because the values are integer, not binary. The training procedure is shown in Fig. 3.5. Because DRVs are the non-negative values and the pre-defined loss function of boosting can't handle non-negative value, a customized loss function which changes negative value into zero is used. On the other hands, multi-layer perceptron is able to handle non-negative output easily by using ReLU (Rectified Linear Unit).

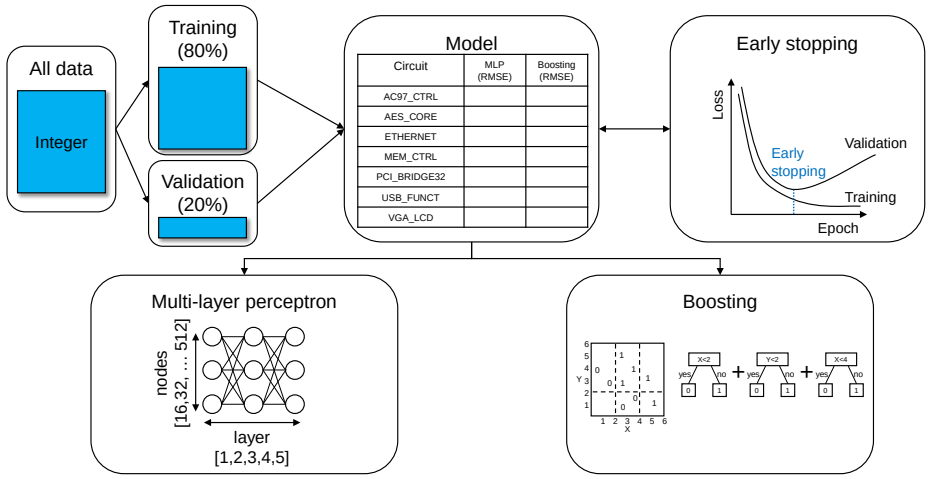


Figure 3.5: The training procedure for the regression of the number of DRVs.

3.3.3 Optimization

In this work, I compared two global optimization algorithms. One is Bayesian optimization which is a machine learning based optimization and the other is particle swarm optimization (PSO) which is one of metaheuristic optimization algorithms.

Bayesian optimization generates a surrogate model and obtain acquisition function by calculating the probability of improvement using the average and standard deviation of predicted value in each point of the entire space. In general, gaussian process regression is used as a surrogate model. It is started from a prior distribution that the physical distance between two points is related to the covariance of the points [26]. Once the samples are observed, the covariance can be obtained. And then the average and standard deviation of predicted value in each point of entire space are calculated. From the average and standard deviation, the probability that each point is better than the current best value can be calculated and the best one becomes the next candidate. There are several acquisition functions such as expected improvement, lower confidence bound and probability improvement [27][28][29]. In this work, expected improvement is used as an acquisition function. Once the next candidate is observed, the

whole process from the surrogate model to acquisition function, is repeated.

PSO generates lots of particles in random locations with random velocities and explores the optimum by updating the position of particles. Particle best (pbest) and global best (gbest) can be changed in every iteration. The velocity of each particle in the next iteration is determined by the velocity of the particle in current iteration, pbest location and gbest location. The new positions of particles are updated by the velocity. It doesn't need complicated computation and it can be performed by parallel computing because the movements of particles in each iteration are independent. The procedure is shown in Fig. 3.6.

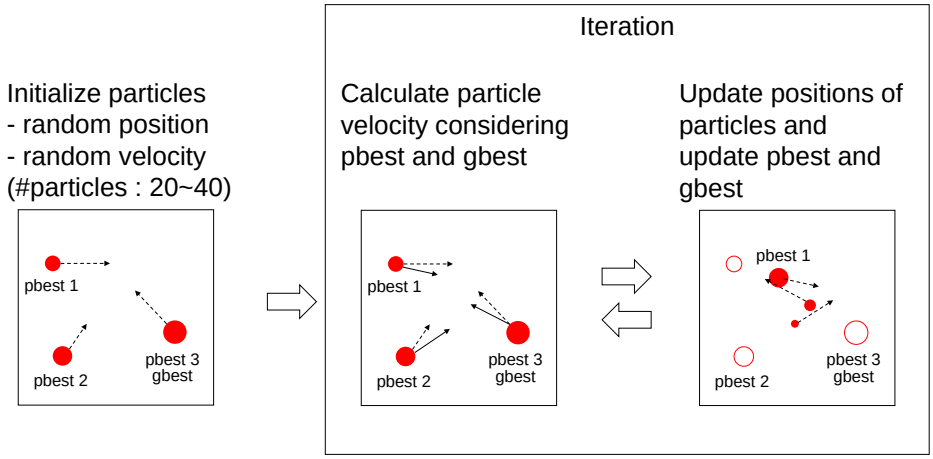


Figure 3.6: The procedure of particle swarm optimization.

Bayesian optimization is originally sequential optimization because covariance should be calculated and the next candidate is suggested after observations are done. I will compare parallel PSO and sequential Bayesian optimization in the experimental results.

3.3.4 Placement Perturbation

In the previous literatures, the cells in a row were spreaded to minimize the probability of DRVs in the window [23][24]. Because the interaction between rows is not considered, it is very fast but it can make routability worse in some cases. I propose placement perturbation of all cells in the hotspots of entire layout. The advantage of this methodology is that the cells in the hotspots move simultaneously and the interaction between rows is considered. The objective function of optimization is to minimize the number of DRVs in entire layout.

Placement in a row can be defined as a problem to make an integer by adding some numbers. The space between cells is called white space. Moving cell is equal to white space redistribution. As shown in Fig. 3.7, there are 4 movable cells and 1 fixed cell in the window. The fixed cell in the window indicates that it is used for the clock tree synthesis. When the cells are used for the clock tree synthesis, I fixed the cells to maintain the timing and skew of clock tree. The cells out of hotspot are also fixed. The placement of movable cells can be expressed by white spaces like [1,2,2,3]. Thus, the placement perturbation in the row is a problem to make 8 by adding 4 numbers.

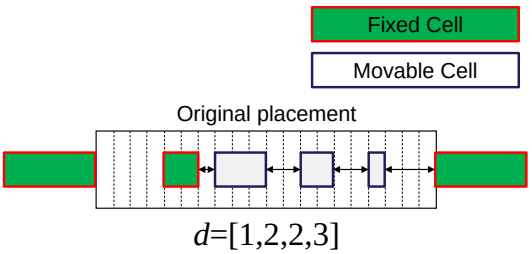


Figure 3.7: Placement in local window is expressed by white spaces between cells.

As shown in Fig. 3.8, there are 165 cases to make 8 by adding 4 numbers such as [0,0,0,8], [0,0,1,7] and so on. It explodes when the number is bigger. For example, a problem to make 40 by adding 8 numbers has 62,891,499 cases. If we perturb all cells in the hotspots of the entire layout, the solution space is too large to find the optimum value. Thus, it is necessary to reduce the solution space.

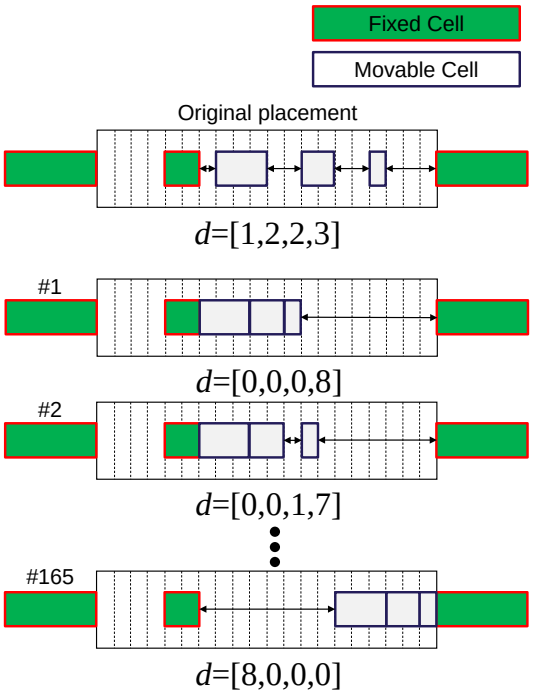


Figure 3.8: There are 165 cases to make 8 by adding 4 numbers.

I propose a reduced solution space by limiting the moving range of each cell. For the reduced solution space, I set a limit of moving range of each cell and the half of white space is used as a maximum movement. For example, if the white space is 2 in the left direction and 3 in the right direction, the maximum movement is 1 in the left direction and 1 in the right. Because only integer is allowed in the placement problem, 1 is selected instead of 1.5 in the right direction. As a result, the cell can select one of three movements like -1, 0 and +1, in which positive values mean movement in the right direction, negative ones do in the left direction. It can reduce 165 cases to 18 cases as shown in Fig. 3.9.

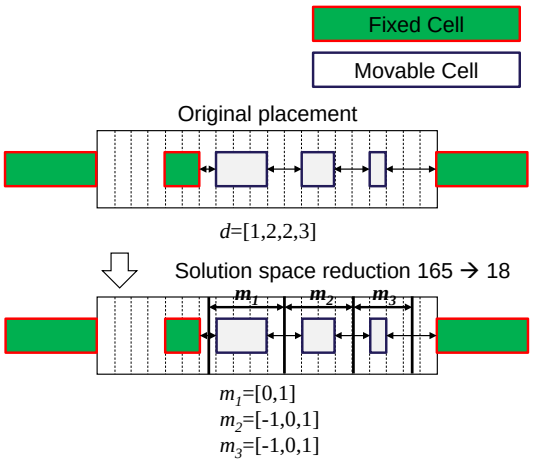


Figure 3.9: The solution space is reduced from 165 to 18.

The method limiting the moving range is easily extended to the entire layout. Assuming that the circuit layout is divided into 2x2 grids as shown in Fig. 3.10, 2 hotspots out of 4 grids are predicted by binary classification. And then the moving range of each cell is defined. The placement perturbation in the entire layout is also expressed by the movement of each cell. The next step is to find the optimal values among all combinations to minimize DRVs. Because original solution space includes the reduced solution space, the optimization result of original one should be better than that of reduced one if there is no limit of iteration. However, the original solution space may not be better

than the reduced one because the optimum solution is explored in a limited time. It will be compared in the experimental results.

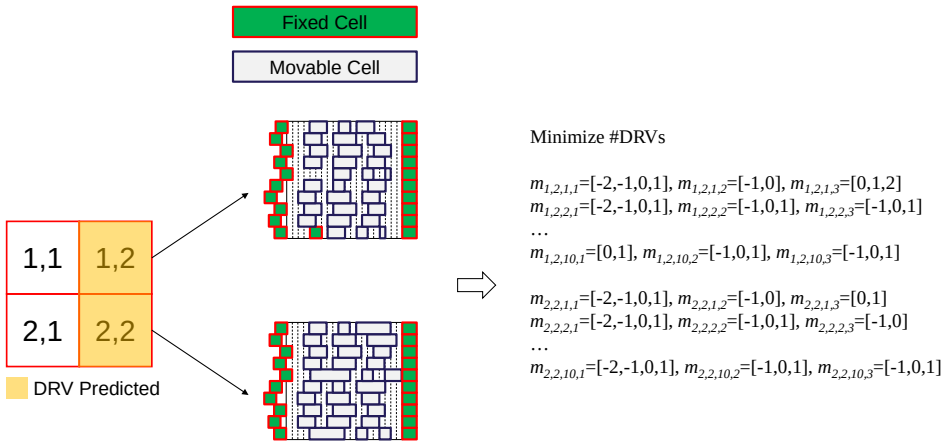


Figure 3.10: Placement perturbation in entire layout. Once the hotspots are determined, the cells in the hotspots can be moved in the reduced solution space.

3.4 Experiments

3.4.1 Experiments Setup

Seven circuits taken from OPENCORES used for the experiments as shown in Table 3.1. I performed synthesis, placement and routing using Synopsys Design Compiler and IC Compiler-II respectively with 28nm technology of foundry. Because all standard cells are composed of M1, the cells are routed from M2. In order to generate DRVs intentionally, M2 and M3 are used for the clock and signal routing. Model training with a circuit itself is not prohibited. For example, when I generate model for AC97_CTRL, 6 circuits excluding AC97_CTRL are used for the training and validation.

3.4.2 Hotspot Prediction

I implemented the binary classification in Python. TensorFlow is used to implement multi-layer perceptron and XGBoost is used for the boosting [30][31]. Hyperparameter

Table 3.1: Target circuits and the training data. The numbers in the training data indicate the initial utilization.

Circuit	#Inst	Initial utilization	Size (cell height x cell height)	Training data						
				AC97_CTRL	AES_CORE	EHERNET	MEM_CTRL	PCI_BRIDGE32	USB_FUNCT	VGA_LCD
AC97_CTRL	4,948	70	130x130	-	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70
AES_CORE	10,428	50	160x160	40, 45, ... , 70	-	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70
ETHERNET	24,073	50	350x350	40, 45, ... , 70	40, 45, ... , 70	-	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70
MEM_CTRL	4,123	60	110x110	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	-	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70
PCI_BRIDGE32	8,579	55	190x190	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	-	40, 45, ... , 70	40, 45, ... , 70
USB_FUNCT	7,239	45	170x170	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	-	40, 45, ... , 70
VGA_LCD	34,872	40	500x500	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	40, 45, ... , 70	-

tuning is performed by Bayesian optimization. The comparison results between multi-layer perceptron and boosting for 7 test circuits are shown in Table 3.2. In all test cases, XGBoost shows better performance than multi-layer perceptron. I generated 90 models in each machine learning algorithm by changing the model architecture for multi-layer perceptron and hyperparameter tuning for XGBoost. The value of logloss in the table is the best one among 90 models.

Table 3.2: Comparison results of machine learning models in binary classification tasks. In all circuits, XGBoost outperforms multi-layer perceptron.

Circuit	Initial utilization	Grid size	logloss	
			Multi-layer perceptron	XGBoost
AC97_CTRL	70	13x13	0.12	0.11
AES_CORE	50	16x16	0.11	0.10
ETHERNET	50	35x35	0.12	0.10
MEM_CTRL	60	11x11	0.11	0.10
PCI_BRIDGE32	55	19x19	0.12	0.11
USB_FUNCT	45	17x17	0.11	0.10
VGA_LCD	40	50x50	0.11	0.10

The prediction results are shown in Table 3.3. Because DRVs data are quite imbalanced, negative labels are much more than positive labels. The accuracy shows 95% on average. The number of predicted hotspots is equal to the sum of true positive and false positive which means cells in false negative are not moved even if they are actual hotspots.

Table 3.3: Summary of hotspot prediction results.

Circuit	Initial utilization	#Grids	True positive (TP)	True negative (TN)	False positive (FP)	False negative (FN)	Accuracy (TP+TN)/(TP+TN+FP+FN)
AC97_CTRL	70	13x13	16	145	1	7	0.95
AES_CORE	50	16x16	4	221	2	29	0.88
ETHERNET	50	35x35	26	1171	10	18	0.98
MEM_CTRL	60	11x11	6	105	3	7	0.92
PCI_BRIDGE32	55	19x19	18	334	1	8	0.98
USB_FUNCT	45	17x17	16	256	5	12	0.94
VGA_LCD	40	50x50	9	2457	3	31	0.99

3.4.3 Regression

The regression is also implemented in Python. The comparison results between multi-layer perceptron and boosting for 7 test circuits are shown in Table 3.4. In all circuits, XGBoost outperforms multi-layer perceptron. I generated 90 models in each machine learning algorithm by changing the model architecture for multi-layer perceptron and hyperparameter tuning for XGBoost. The value of root mean squared error in the table is the best one among 90 models.

Table 3.4: Comparison results of machine learning models in regression tasks. In all circuits, XGBoost outperforms multi-layer perceptron.

Circuit	Initial utilization	#Grids	Root mean squared error	
			Multi-layer perceptron	XGBoost
AC97_CTRL	70	13x13	7.64	7.34
AES_CORE	50	16x16	7.05	6.94
ETHERNET	50	35x35	7.72	7.31
MEM_CTRL	60	11x11	7.51	7.25
PCI_BRIDGE32	55	19x19	7.58	7.34
USB_FUNCT	45	17x17	7.33	7.02
VGA_LCD	40	50x50	7.34	7.16

The predicted results using regression model with XGBoost is shown in Fig. 3.11. The number of DRVs are well predicted along the line of y=x even if unseen layout is predicted by pre-trained model. As shown in Fig. 3.12, designs doomed to be routing failure are filtered out when the predicted DRVs are more than 1000.

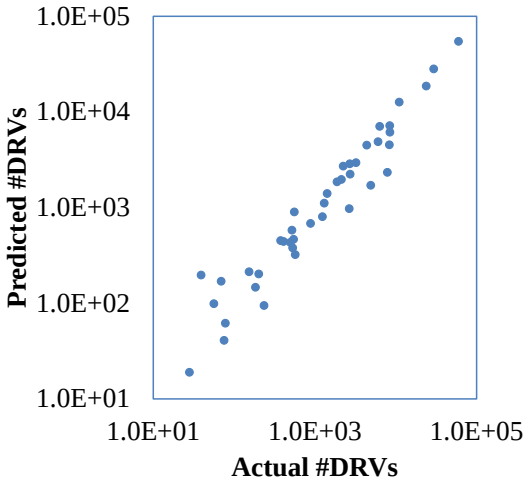


Figure 3.11: Regression results of 49 circuit layouts. There are 7 circuits and each circuit has 7 different utilization values.

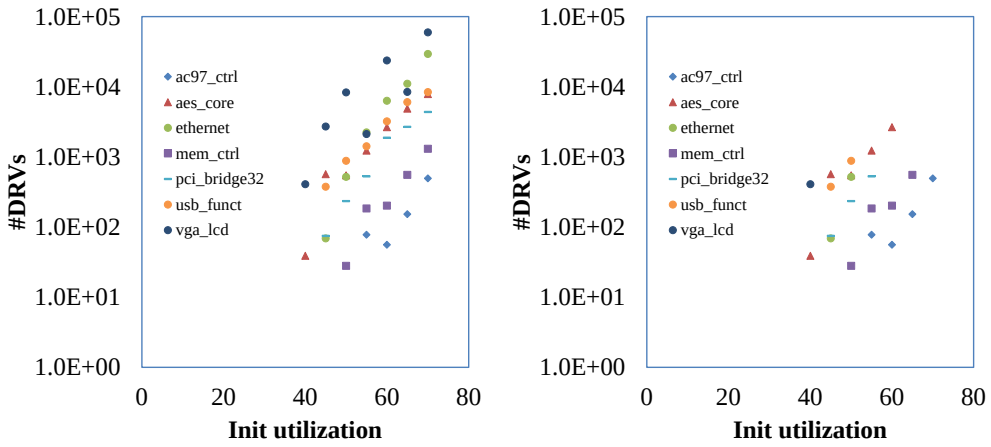


Figure 3.12: Initial utilization vs. DRVs. (a) scatter plot of DRVs of 7 circuits when filtering is not applied (b) scatter plot of DRVs of 7 circuits when filtering is applied

3.4.4 Placement Perturbation

Placement perturbation is implemented in Python. It invokes commercial P&R tool to move cells in the hotspots and extract features in entire layout. Precisely, it generates a tcl script which can be readable in commercial P&R tool and the script includes which cells are moved and how many sites each cell moves. Because the movement is calculated from the bounding boxes, overlaps of cells are not found after cell movement. If the placement has complicated rules, legalization should be performed after cell movement. Once the features are gathered, predicted number of DRVs is obtained using pre-trained regression model. Since the optimization algorithms propose better values in each iteration, the number of DRVs keep decreasing. I use the GPyOpt module for the Bayesian optimization [32]. The comparison results of optimization using particle swarm optimization and Bayesian optimization are shown in Fig. 3.13. They are obtained from the reduced solution spaces. Original solution spaces have different results. They reduced the predicted number of DRVs. One of optimization algorithm is not always better than the others.

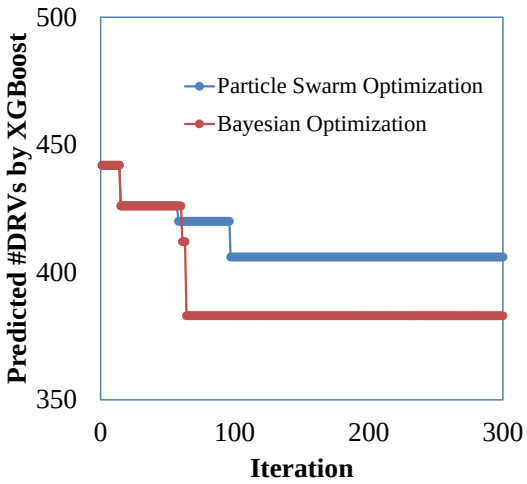


Figure 3.13: Placement perturbation result of VGA_LCD. The predicted DRVs are reduced by particle swarm optimization and Bayesian optimization.

I compared the predicted number of DRVs between original solution spaces and reduced solution spaces as shown in Table. 3.5. In all circuits, the reduced solution spaces have better results than the original solution spaces. Because the number of evaluations is 300, the reduced solution spaces can have better results. If the number becomes infinity, the original solution spaces will have better results because they include the reduced solution spaces.

Table 3.5: Comparison results of solution spaces. In all circuits, reduced solution spaces have better results than original solution spaces.

Circuit	Predicted number of DRVs				
	Placement by P&R tool	Placement perturbation			
		Particle swarm optimization		Bayesian optimization	
		Original	Reduced	Original	Reduced
AC97_CTRL	434	401 (-7.6%)	387 (-10.8%)	406 (-6.5%)	403 (-7.1%)
AES_CORE	466	425 (-8.8%)	408 (-12.4%)	416 (-10.7%)	408 (-12.4%)
ETHERNET	581	477 (-17.9%)	417 (-28.2%)	547 (-5.0%)	453 (-22.0%)
MEM_CTRL	203	160 (-21.2%)	141 (-30.5%)	170 (-16.3%)	140 (-31.0%)
PCI_BRIDGE32	379	379 (0.0%)	283 (-25.3%)	379 (0.0%)	334 (-11.9%)
USB_FUNCT	453	421 (-7.1%)	385 (-15.0%)	421 (-7.1%)	383 (-15.5%)
VGA_LCD	442	429 (-2.9%)	406 (-8.1%)	422 (-4.5%)	383 (-13.3%)

After the optimization is finished, routing is performed using the best result. Routed #DRVs are also reduced as shown in Table. 3.6 and Table. 3.7. Because Bayesian optimization was performed sequentially, the runtime is much longer than that of PSO. Though the reduction of predicted #DRVs of PSO is more than that of Bayesian optimization, Bayesian optimization shows better results for the routed #DRVs due to the error of machine learning. In the routed layouts, the number of design rule violations is reduced by 22% on average over that using the conventional method to predict routability with routing congestion. Since routed wirelength is correlated with timing slack, it is extracted when routing is done. The maximum degradation of routed wirelength is about 1.4%.

Table 3.6: Placement perturbation results using particle swarm optimization. Predicted #DRVs are reduced by -18.6% on average and the routed #DRVs are reduced by -18.9% on average

Circuit	Placement perturbation is not applied			Placement perturbation is applied			
	Predicted #DRVs	Routed #DRVs	Routed wirelength	Predicted #DRVs	Routed #DRVs	Routed wirelength	Runtime [h]
AC97_CTRL	434	495	90525.44	387 (-10.8%)	454 (-8.3%)	89430.61 (-1.2%)	0.28
AES_CORE	466	545	162596.31	408 (-12.4%)	446 (-18.2%)	162263.06 (-0.2%)	0.27
ETHERNET	581	520	768266.62	417 (-28.2%)	363 (-30.2%)	768017.71 (0.0%)	0.63
MEM_CTRL	203	202	72230.24	141 (-30.5%)	137 (-32.2%)	73039.51 (1.1%)	0.20
PCI_BRIDGE32	379	529	177192.28	283 (-25.3%)	475 (-10.2%)	177936.93 (0.4%)	0.26
USB_FUNCT	453	376	140619.49	385 (-15.0%)	348 (-7.4%)	140532.29 (-0.1%)	0.23
VGA_LCD	442	405	1242841.67	406 (-8.1%)	299 (-26.2%)	1241066.05 (-0.1%)	0.77
Improvement				-18.6%	-18.9%	0.0%	-

Table 3.7: Placement perturbation results using Bayesian optimization. Predicted #DRVs are reduced by -16.2% on average and the routed #DRVs are reduced by -22.2% on average

Circuit	Placement perturbation is not applied			Placement perturbation is applied			
	Predicted #DRVs	Routed #DRVs	Routed wirelength	Predicted #DRVs	Routed #DRVs	Routed wirelength	Runtime [h]
AC97_CTRL	434	495	90525.44	406 (-7.1%)	436 (-11.9%)	90056.73 (-0.5%)	7.22
AES_CORE	466	545	162596.31	408 (-12.4%)	468 (-14.1%)	162112.63 (-0.3%)	7.80
ETHERNET	581	520	768266.62	453 (-22.0%)	415 (-20.2%)	769163.49 (0.1%)	20.08
MEM_CTRL	203	202	72230.24	140 (-31.0%)	140 (-30.7%)	73241.28 (1.4%)	6.02
PCI_BRIDGE32	379	529	177192.28	334 (-11.9%)	426 (-19.5%)	177659.83 (0.3%)	7.98
USB_FUNCT	453	376	140619.49	383 (-15.5%)	346 (-8.0%)	140990.87 (0.3%)	7.02
VGA_LCD	442	405	1242841.67	383 (-13.3%)	199 (-50.9%)	1244099.5 (0.1%)	22.03
Improvement				-16.2%	-22.2%	0.2%	-

Chapter 4

Conclusions

4.1 Synthesis of Representative Critical Path Circuits reflecting BEOL Timing Variation

The section proposed a BEOL-aware methodology for synthesizing a representative critical path circuit which can provide an accurate performance prediction on post-Si target circuit in deep sub-micron technologies. Precisely, I proposed a methodology which was able to incrementally explore routing patterns (i.e., BEOL reconfiguring) as well as the conventional gate resizing. My synthesis framework integrated a set of novel techniques: (1) extracting and classifying BEOL configurations for lightening design space complexity, (2) formulating BEOL random variables for fast and accurate timing analysis, and (3) exploring alternative (ring oscillator) circuit structures for extending the applicability of this work. Through experiments with industry circuits, it was shown that the synthesis framework was able to reduce the prediction error by 54% and 19% on average over that using the conventional critical path replica and using the conventional method exploiting gate sizing only, respectively.

4.2 Reduction of Routing Failures through Enhanced Prediction on Design Rule Violations in Placement

The section proposed a prediction methodology of design rule violations and a perturbation methodology of placement to reduce design rule violations in deep sub-micron technologies. Precisely, I combined machine learning and global optimization to minimize the predicted number of design rule violations. The hotspots are predicted by binary classification and then placement perturbation in hotspots are performed using particle swarm optimization or Bayesian optimization until the number of design rule violations is minimized, in which the number of DRVs is predicted by regression. Since the optimization should be done in a limited time, the reduced solution space for placement perturbation is also proposed. The solution space is reduced by limiting the range of cell movement. Through experiments with industry circuits, it was shown that the framework was able to reduce the number of design rule violations by 22% on average over that using the conventional method to predict routability with routing congestion.

Bibliography

[1] M. W. Kuemerle, S. K. Lichtensteiger, D. W. Douglas and I. L. Wemple, "Integrated circuit design closure method for selective voltage binning," U.S. Patent 7,475,366, 2009.

[2] V. Zolotov, C. Visweswariah and J. Xiong, "Voltage binning under process variation," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 425–432, 2009.

[3] Q. Liu and S. S. Sapatnekar, "Capturing post-silicon variations using a representative critical path," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 2, pp. 211–222, 2010.

[4] L. Lu, "Physical design challenges and innovations to meet power, speed, and area scaling trend," *Keynote in ACM International Symposium on Physical Design*, 2017.

[5] T. Huynh-Bao, J. Ryckaert, Z. Tokey, A Mercha, D. Verkest, A Thean and P. Wambacq, "Statistical timing analysis considering device and interconnect variability for beol requirements in the 5-nm node and beyond," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 25, no. 5, pp. 1669-1680, 2017.

[6] K. Chiang, J. Huang, T. Cheng, C. Hsiao, J. Sun, C. Cheng, K. Lu, K. Su, C. Lin, K. Chen, K. Tam, T. Liu, K. Su and M. Jeng, "A comprehensive solution

for beol variation characterization and modeling," *Simulation of Semiconductor Processes and Devices*, pp. 307–310, 2016.

[7] A.K.M. M. Islam and H. Onodera, "On-chip detection of process shift and process spread for post-silicon diagnosis and model-hardware correlation," *IEICE Transactions on Information and Systems*, vol. E96D, no. 9, pp. 1971–1979, 2013.

[8] X. Qi, A. Gyure, Y. Luo, S. C. Lo, M. Shahram, and K. Singhal, "Measurement and characterization of pattern dependent process variations of interconnect resistance, capacitance and inductance in nanometer technologies," *ACM Great Lakes Symposium on VLSI*, pp. 14–18, 2006.

[9] C. Liu, O. Law and F. Li, "An accurate interconnect test structure for parasitic validation in on-chip machine learning accelerators," *arXiv:1701.03181 [cs]*, 2017.

[10] A. Drake, R. Senger, H. Singh, G. Carpenter and N. James, "Dynamic measurement of critical-path timing," *IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, pp. 249–252, 2008.

[11] M. Bhushan, A. Gattiker, M. Ketchen and K. Das, "Ring oscillators for cmos process tuning and variability control," *IEEE Transaction on Semiconductor Manufacturing*, vol. 19, no. 1, pp. 10–18, 2006.

[12] T. Chan, P. Gupta, A. B. Kahng and L. Lai, "Synthesis and analysis of design-dependent ring oscillator (ddro) performance monitors," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 22, no. 10, pp. 2117–2130, 2014.

[13] W. C. Elmore, "The transient analysis of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, 1948.

[14] R. Gupta, B. Tutuianu and L. T. Pileggi, "The elmore delay as a bound for rc trees with generalized input signals," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 1, pp. 95–104, 1997.

[15] OpenCores: Open Source IP-Cores, http://www.opencores.org

[16] TSMC and Samsung 5nm comparison, https://semiwiki.com/semiconductor-manufacturers/samsung-foundry/8157-tsmc-and-samsung-5nm-comparison/

[17] M. Hogan, "Leveraging baseline checks for robust reliability verification," White paper of Mentor, A Siemens business.

[18] W-T. J. Chan, Y. Du, A. B. Kahng, S. Nath, and K. Samadi, "BEOL stack-aware routability prediction from placement using data mining techniques," *International Conference on Computer Design*, pp. 41–48, 2016.

[19] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings, and L. Behjat, "Detailed routing violation prediction during placement using machine learning," *International Symposium on VLSI Design, Automation and Test*, pp. 1–4, 2017.

[20] R. Islam, and A Shahjalal, "Predicting drv violations using ensemble random forest algorithm," *ACM/IEEE Design Automation Conference*, pp. 1–2, 2019.

[21] T-C. Yu, S-Y. Fang, H-S. Chiu, K-S. Hu, P. H-Y. Tai, C. C-F. Shen, and H. Sheng, "Pin accessibility prediction and optimization with deep learning-based pin partern recognition," *ACM/IEEE Design Automation Conference*, pp. 1–6, 2019.

[22] R. Liang, H. Xiang, D. Pandey, L. Reddy, S. Ramji, G-J. Nam, and J. Hu, "DRC hotspot prediction at sub-10nm process nodes using customized convolutional network," *International Symposium on Physical Design*, pp. 135–142, 2020.

[23] W-T. J. Chan, P-H. Ho, A. B. Kahng, S. Nath, and P. Saxena, "Routability optimization for industrial designs at sub-14nm process nodes using machine learning," *International Symposium on Physical Design*, pp. 15–21, 2017.

[24] T-C. Yu, S-Y. Fang, H-S. Chiu, K-S. Hu, P. H-Y. Tai, C. C-F. Shen, and H. Sheng, "Lookahead placement optimization with cell library-based pin accessibility prediction via active learning," *International Symposium on Physical Design*, pp. 65–72, 2020.

[25] A. Kahng, A. B. Kahng, H. Lee, and J. Li, "Probe: a placement, routing, back-end-of-line measurement utility," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 7, pp. 1459–1472, 2018.

[26] C. E. Rasmussen, and C. Williams, "Gaussian processes for machine learning," The MIT Press, 2006.

[27] D. R. Jones, M. Schonlau and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, pp 455—492, 1998.

[28] D. D. Cox and S. John, "Sdo: a statistical method for global optimization," *Multidisciplinary Design Optimization: State-of-the-Art*, pp 315—329, 1997.

[29] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 37, no. 1, pp 97—106, 1964.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: large-scale machine learning on heterogeneous systems," *arXiv:1603.04467 [cs]*, 2015.

[31] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," *arXiv:1603.02754 [cs]*, 2016.

[32] GPyOpt, http://github.com/SheffieldML/GPyOpt

초 록

타이밍 분석 및 디자인 룰 위반 제거는 반도체 칩 제조를 위한 마스크 제작 전에 완료되어야 할 필수 과정이다. 그러나 트랜지스터와 인터커넥트의 변이가 증가하고 있고 디자인 룰 역시 복잡해지고 있기 때문에 타이밍 분석 및 디자인 룰 위반 제거는 초미세 회로에서 더 어려워지고 있다. 본 논문에서는 초미세 설계를 위한 두가지 문제인 타이밍 분석과 디자인 룰 위반에 대해 다룬다.

첫번째로 공정 코너에서 타이밍 분석은 실리콘으로 제작된 회로의 성능을 정확히 예측하지 못한다. 그 이유는 공정 코너에서 가장 느린 타이밍 경로가 모든 공정 조건에서도 가장 느린 것은 아니기 때문이다. 게다가 칩 내의 임계 경로에서 인터커넥트에 의한 지연 시간이 전체 지연 시간에서의 영향이 증가하고 있고, 10나노 이하 공정에서는 20%를 초과하고 있다. 즉, 실리콘으로 제작된 회로의 성능을 정확히 예측하기 위해서는 대표 회로가 트랜지스터의 변이 뿐만아니라 인터커넥트의 변이도 반영해야한다. 인터커넥트를 구성하는 금속이 10층 이상 사용되고 있고, 각 층을 구성하는 금속의 저항과 캐패시턴스와 비아 저항이 모두 회로 지연 시간에 영향을 주기 때문에 대표 회로를 찾는 문제는 차원이 매우 높은 영역에서 최적의 해를 찾는 방법이 필요하다. 이를 위해 인터커넥트를 제작하는 공정(백 엔드 오브 라인)의 변이를 반영한 대표 회로를 생성하는 방법을 제안하였다. 공정 변이가 없을때 가장 느린 타이밍 경로에 사용된 게이트와 라우팅 패턴을 변경하면서 점진적으로 탐색하는 방법이다. 구체적으로, 본 논문에서 제안하는 합성 프레임워크는 다음의 새로운 기술들을 통합하였다: (1) 라우팅을 구성하는 여러 금속 층과 비아를 추출하고 탐색 시간 감소를 위해 유사한 구성들을 같은 범주로 분류하였다. (2) 빠르고 정확한

타이밍 분석을 위하여 여러 금속 층과 비아들의 변이를 수식화하였다. (3) 확장성을 고려하여 일반적인 링 오실레이터로 대표회로를 탐색하였다.

두번째로 디자인 룰의 복잡도가 증가하고 있고, 이로 인해 표준 셀들의 인터커넥트를 통한 연결을 진행하는 동안 디자인 룰 위반이 증가하고 있다. 게다가 표준 셀의 크기가 계속 작아지면서 셀들의 연결은 점점 어려워지고 있다. 기존에는 회로내 모든 표준 셀을 연결하는데 필요한 트랙 수, 가능한 트랙 수, 이들 간의 차이를 이용하여 연결 가능성을 판단하고, 디자인 룰 위반이 발생하지 않도록 셀 배치를 최적화하였다. 그러나 기존 방법은 최신 공정에서는 정확하지 않기 때문에 더 많은 정보를 이용한 회로내 모든 표준 셀 사이의 연결 가능성을 예측하는 방법이 필요하다. 본 논문에서는 기계 학습을 통해 디자인 룰 위반이 발생하는 영역 및 개수를 예측하고 이를 줄이기 위해 표준 셀의 배치를 바꾸는 방법을 제안하였다. 디자인 룰 위반 영역은 이진 분류로 예측하였고 표준 셀의 배치는 디자인 룰 위반 개수를 최소화하는 방향으로 최적화를 수행하였다. 제안하는 프레임워크는 다음의 세가지 기술로 구성되었다: (1) 회로 레이아웃을 여러 개의 정사각형 격자로 나누고 각 격자에서 라우팅을 예측할 수 있는 요소들을 추출한다. (2) 각 격자에서 디자인 룰 위반이 있는지 여부를 판단하는 이진 분류를 수행한다. (3) 메타휴리스틱 최적화 또는 베이지안 최적화를 이용하여 전체 디자인 룰 위반 개수가 감소하도록 각 격자에 있는 표준 셀을 움직인다.

주요어: 대표 임계 경로 회로, 회로 지연 시간 예측, 공정 변이, 백 엔드 오브 라인, 디자인 룰 위반, 기계 학습, 스탠다드 셀 배치 변경, 메타휴리스틱, 베이지안 최적화
student number: 2001-21599