Article

Not peer-reviewed version

A High Dimensional and High Sigma Statistical Model for SRAM Read Access Yield Estimation

Dong Xu Zhang [*] , Gang Lu Li , Si Rui Yan

Posted Date: 30 May 2024

doi: 10.20944/preprints202405.2006.v1

Keywords: SRAM; Yield Analysis; High Sigma; High Dimension

Article

A High Dimensional and High Sigma Statistical Model for SRAM Read Access Yield Estimation

Xudong Zhang [1],*, Lugang Li [2] and Ruisi Yan [2]

[1] North University of China
[2] Hubei Sanjiang Aerosapce Honglin Exploration and Control Co; 1210655995@qq.com
* Correspondence: almostday@163.com

Abstract: High dimensional statistical analysis for the yield of large-scale circuits is quite difficult due to expensive simulations, especially for the memory circuits with high sigma requirement (e.g., SRAM). In this paper, we developed an efficient sparse additive model to substitute simulations. To fit high sigma region accurately, the modeling center is moved to near failure boundary searched by scaling the shape of sampling function. To solve the model efficiently, the process variables are grouped by standard cells so that the model can be solved by our developed blockwise greedy algorithm. The experiments on the 28nm memory circuits validate that our method achieves high accuracy and efficiency compared with other state-of-art works.

Keywords: SRAM; yield analysis; high sigma; high dimension

1. Introduction

The reliability of integrated circuits is severely affected by process variation as semiconductor technology continues to advance, especially for memory such as the Static Random-Access Memory (SRAM). The SRAM consists of millions of replicated bitcells and is more susceptive to the variation due to the minimized cell size. The failure rate of each SRAM cell should be extremely low ($10^{-8}\sim10^{-5}$) to make sure the acceptable yield before taping out the chip.

SRAM yield can be measured by static stability metrics, such as static noise margin (SNM), and dynamic stability metrics, such as read access yield and write yield [1]. In this paper, we focus on read (access) yield because they can reflect the dynamical characteristics of SRAM under the assisted circuits and column leakage [1]. As a result, a large number of process variables should be considered for predicting yield, which forms a high dimensional variation space. For example, there are over ten thousand process variables to be considered for 256-bits column with sense amplifier. However, the methodology can be extended for other yield metrics.

To estimate such high-sigma "events" accurately, many statistical methods have been proposed. Among them, Standard Monte Carlo (MC) estimation is typically regard as the gold reference, which samples the variation space directly and simulates each sample to get the corresponding response. However, MC based on circuit simulator, such as Spice, is extremely slow because it needs hundred millions of simulations to capture a single failure point.

2. Preliminaries

2.1. Problem Formulation

We consider $\boldsymbol{X}=(x_1,x_2,\dots,x_m)$ as a vector consisted of m independent normalized Gaussian variables modeled from process parameters. And $H(\boldsymbol{X})$ is the probability density function (PDF) of $\boldsymbol{X}$. Let Y be the circuit performance which can be measured through expensive transistor-level simulation, such as inverter chain delay, SRAM read access time etc.

For the failure rate evaluation of SRAM, S denotes the failure region. Typically, the set S is extremely small. We define the circuit performance doesn't meet the specification when $Y \in S$. And we further introduce indicator function $I(X)$ to identify pass/fail of Y:

$$I(X) = \begin{cases} 0, & if \quad Y \notin S \\ 1, & if \quad Y \in S \end{cases} \tag{1}$$

Therefore, the probability can be calculated as:

$$P_{fail} = P(Y \in S) = \int I(\boldsymbol{X}) \cdot H(\boldsymbol{X}) d\boldsymbol{X} \tag{2}$$

Unfortunately, the formulation (2) is difficult to calculate analytically because we don't know what distribution $I(\boldsymbol{X})$ satisfies exactly. Traditionally, Monte Carlo is used to estimate the failure probability by sampling from $H(\boldsymbol{X})$ directly, and the unbiased estimate of P_{fail}:

$$\hat{P}_{fail} = \hat{P}(Y \in S) = \frac{1}{N}\sum_{i=1}^{N} I(x_i) \xrightarrow{N \to +\infty} P(Y \in S) \tag{3}$$

2.2. Importance Sampling

For estimating SRAM failure rate, $Y \in S$ is a high sigma "event". Standard MC needs hundred millions of expensive circuit simulations to capture such a rare "event". It is not realistic to apply MC to the practical design especially the circuit size is large because each sample needs a transistor-level simulation. To address this issue, IS methods have been introduced to sample near the failure region through a "distorted" sampling distribution $g(\boldsymbol{X})$.

By constructing the distorted sampling function $G(\boldsymbol{X})$, we can pick more failure samples. And the failure probability can be expressed as (4):

$$P_{fail} = P(Y \in S) = \int I(\boldsymbol{X}) \cdot \frac{H(\boldsymbol{X})}{G(\boldsymbol{X})} \cdot G(\boldsymbol{X}) dX \tag{4}$$

$$= \int I(\boldsymbol{X}) \cdot w(\boldsymbol{X}) \cdot G(\boldsymbol{X}) dX \tag{5}$$

Here, the $w(\boldsymbol{X})$ denotes the likelihood ratio between original PDF $H(\boldsymbol{X})$ and the distort PDF $G(\boldsymbol{X})$ which compensates for the discrepancy between $H(\boldsymbol{X})$ and $G(\boldsymbol{X})$. And unbiased IS estimator $\hat{P}_{IS,fail}$ can be calculated as (6):

$$\hat{P}_{IS,fail} = \hat{P}_{IS}(Y \in S)$$

$$= \frac{1}{M}\sum_{j=1}^{M} w(x_j) I(x_j) \xrightarrow[M \to +\infty]{a.s.} P(Y \in S) \tag{6}$$

With a proper $g(\boldsymbol{X})$, $\hat{P}_{IS,fail}$ can be approximately equal to MC results.

However, the likelihood ratio $w(x_j)$ shows huge numerical instability in the high dimensional scenarios. Some $w(x_j)$ become dominant and even make the estimation result infinite so that the Equation (6) becomes unreliable.

2.3. Meta Modeling

Meta model is another solution to accelerate yield analysis by substituting the expensive simulations. It enables simulation evaluations from minutes to hours to be achieved in microseconds so that the large scale MC is feasible.

Due to high dimensional input variables, the effective meta models are typically written as the additive forms:

$$f(\boldsymbol{X}) = f_0 + \sum_{i=1}^{m} f_i(x_i) \tag{7}$$

where f_0 is the intercept used to measure the mean of response and f_i (x_i) represents the effect of a single variable x_i acting independently upon Y.

Equation (3) shows the $\hat{P}_{fail}$ heavily depended on the indicator function $I(\boldsymbol{X})$. And the circuit performance near the failure region predicted by meta model must be accurate enough to guarantee the correct judgement of $I(\boldsymbol{X})$. Otherwise, it will lead to a large bias due to the error accumulation. However, the existing meta-model methods suffers from two problems: first, the accuracy fails to be maintained in the nonlinear region far away the original points. Second, Equation (7) cannot reflect the correlated effects brought by two related variables, which also lead to larger bias of estimation.

3. Proposed Method

To formulate our sparse additive model, we first pre-process the input variables as follows:

$$\boldsymbol{X} = (\boldsymbol{X_L}, \boldsymbol{X_R}) = (x_1, x_2, \dots, x_m) \tag{8}$$

where $\boldsymbol{X_L} = (l_1, l_2, \dots, l_{nl})$ is the nl process variables related to devices, such as threshold parameter Vth_0, cutoff voltage V_{off}, and electron mobility u_0. And $\boldsymbol{X_R} = (r_1, r_2, \dots, r_{nr})$ is the critical parasitic parameters of long bitlines extracted from the post layout SRAM array.

Notice that the input signal only walks through the critical path to generate the output of a circuit, which means large number of standard cells have few effects on the output. Hence, we group the variables by the standard cells. Suppose there are P cells in total, then the grouped variables can be represented as:

$$\boldsymbol{X_L} = \{\boldsymbol{L_1}, \boldsymbol{L_2}, \dots, \boldsymbol{L_p}\} \tag{9}$$

where $\boldsymbol{L_i} = (l_1, l_2, \dots, l_{ci})$ $i = 1, \dots, P$. They are ci- dimensional vectors from the ith cell. $\boldsymbol{L_p}$ is the extracted post-layout parasitic parameters such as equivalent resistor-capacitance (RC) to ground of bitline and wordline.

Based on these processed input variables, we formulate our meta as follows:

$$f(\boldsymbol{X}) = \beta_0 + \sum_{i=1}^{P} \mathbf{B}_{\boldsymbol{i}} \boldsymbol{\Phi_i}(\boldsymbol{L_i}) + \boldsymbol{C}\,\boldsymbol{\Psi}(\boldsymbol{X_R}) \tag{10}$$

$$s.t.\, \boldsymbol{B_i}\boldsymbol{\Phi_i}(\boldsymbol{L_i}) = \sum_{j=1}^{N} w_{i,j}\varphi_{i,j}(\boldsymbol{L_i}) + \sum_{j=1}^{ci} \beta_{i,j} s(l_{i,j}) \tag{11}$$

$$\boldsymbol{C}\,\boldsymbol{\Psi}(\boldsymbol{X_R}) = \sum_{j=1}^{N} w_j \varphi_j(\boldsymbol{X_R}) + \sum_{k=1}^{nr} c_k s(r_k) \tag{12}$$

where β_0 is the intercept to represent the modeling center. $\boldsymbol{\Phi_i}$ is the ith basic function set of the grouped $\boldsymbol{L_i}$ and $\mathbf{B}_{\boldsymbol{i}} = (w_{i,1}, \dots, w_{i,N}, \beta_{i,1}, \dots, \beta_{i,ci})$ is the ith grouped coefficient vector. $\boldsymbol{\Psi}$ is the basic function set of $\boldsymbol{X_R}$ and $\boldsymbol{C} = (w_1, \dots, w_N, c_1, \dots, c_{nr})$ is the coefficient vector. $\varphi_{i,j}(\boldsymbol{L_i})$ is the multivariable gaussian radial basic function (RBF) interpolation [10] of $\boldsymbol{L_i}$ to capture the nonlinear effects of process variables in the ith cell and $w_{i,j}$ is the corresponding weights. $s(l_{i,j})$ is the polynomial function of jth variable of $\boldsymbol{L_i}$. N is the number of training samples. φ_j is the RBF interpolation of parasitic parameter $\boldsymbol{X_R}$ and $s(r_k)$ is the polynomial function of kth variable of $\boldsymbol{X_R}$.

For Equation (10), there are large number of coefficients to be solved. And we notice that the sparsity is reflected not only between groups but also within the group. Because except that most cells that are not in the critical path do not affect circuit performance, most devices in a cell located in the critical path also have few effects on the circuit performance. To full exploit the sparsity from inter groups and intra group, we can formulate the objective function as follows.

$$Q(\boldsymbol{\Omega}) = \frac{1}{2N}\|Y - \sum_{i=1}^{P} \mathbf{B}_{\boldsymbol{li}}\boldsymbol{\Phi_{li}}(\boldsymbol{L_i}) - \boldsymbol{C}\,\boldsymbol{\Psi}(\boldsymbol{X_R})\|_2^2 + (1-\alpha)\lambda\sum_{i=1}^{P}\|\boldsymbol{B_i}\|_2 + \alpha\lambda\|\boldsymbol{\Omega}\|_1 \tag{13}$$

where $\boldsymbol{\Omega} = (\boldsymbol{B_1}, \dots, \boldsymbol{B_P}, \boldsymbol{C})$ is the entire parameter space; $\alpha \in [0\sim1]$ and λ are regularization constants; α decides the balances of the convex combination of L1 and L2 norm penalties and λ controls the degree of sparsity of the solution; N is the number of samples. Since the β_0 doesn't need to be penalized, we omit it from the objective function and calculate it when the other coefficients convergence.

The form of objective function is simplified to a sparse group lasso problem, which can be solved by the modified blockwise greedy algorithm efficiently.

The overview of the yield analysis method based on our model, named SAIM, is summed as Algorithm 1. The training samples generated by the distorted sampling function $\mathrm{g}(\boldsymbol{X})$ located on the failure boundary by our modeling center search method. After constructing our model, we generate enough number of MC samples and evaluate them by SAIM to obtain the yield prediction according to Equation (3).

Algorithm 1: SAIM Model Based Yield Analysis

```
Input: The sample set $(\boldsymbol{X_i}, \boldsymbol{y_i})$, $i = 1, \dots, N$ from the $\boldsymbol{H}(\boldsymbol{X})$
Output: The SRAM failure rate $\widehat{P}_{fail}$
1.  Modeling center search
      Finding the failure region globally by scaled the shape of $H(\boldsymbol{X})$ and initialize the distorted
      sampling function $\mathrm{G}(\boldsymbol{X})$ to generate training set.
2.  Model Construction
    2.1  Generate the training samples from $\mathrm{G}(\boldsymbol{X})$.
    2.2  Group the variables and construct the SAIM model:
           $f(\boldsymbol{X}) = \beta_0 + \sum_{i=1}^{P} \mathbf{B}_{li} \boldsymbol{\Phi}_{li} (\boldsymbol{L}_i) + \boldsymbol{C}\, \boldsymbol{\Psi}(\boldsymbol{X_R})$
    2.3  Using our blockwise greedy solver to solve the model coefficient.
3.  Failure Rate Calculation
      Calculate the failure rate as:
           $\hat{P}_{fail} = \frac{1}{N} \sum_{i=1}^{N} I(f(\boldsymbol{X}) \in S)$
```

4. Implement Details

4.1. High Dimensional Failure Boundary Search

In the deep submicron technology node below 65nm, the process variables have strong nonlinear effects on the circuit performance [11] due to the smaller device size and lower operation voltage.

In the yield estimation scenario, only the accuracy near the failure boundary will be concerned. As illustrated in Equation (3), the error will be accumulated when indicator function I(X) has a wrong judgement. As a result, the closer training samples are to failure boundary, the more computational cost will be saved.

Directly fit the entire nonlinear region needs too large number of samples, especially for the samples far away the original points. To guarantee the accuracy in the entire region, we have to obtain the samples by its original distribution to avoid the “covariate shift” [13] that leads to large bias on the parameter estimation. Besides, the modeling efforts will increase drastically due to the needs of constructing high-order basic functions to fit the nonlinear region far away the original point.

The failure boundary is very difficult to search in the high dimension space for the too complicated shape. The Euclidean distance (or the L2-norm) of two high dimensional samples is almost same [12], which means the search method [4] based on minimizing the L2-norm fails in the high dimensional scenario.

We define $\boldsymbol{\Sigma}$ is the covariation matrix of $\boldsymbol{X}$, which has been normalized as identity matrix. The $\boldsymbol{\Sigma}$ determines the shape of sampling function $H(\boldsymbol{X})$. We enlarge the $\boldsymbol{\Sigma}$ by multiplying the scale parameter r to generate 1000 samples from $H(\boldsymbol{X})$ with $r\boldsymbol{\Sigma}$, $H(\boldsymbol{X}|r\boldsymbol{\Sigma})$.

As shown in Figure 1, with the increased r, the failure samples are more easily obtained. Suppose N failure samples have been collected, we can construct a new sample function as follows:

$$G(\boldsymbol{X}) = \frac{1}{N} \sum_{i=1}^{N} q_i(\boldsymbol{X}|\boldsymbol{u_i}, \boldsymbol{\Sigma}) \tag{12}$$

where the $\boldsymbol{u_i}$ is the i th failure sample represented by a vector. And $q_i(\boldsymbol{X}|\boldsymbol{u_i},\boldsymbol{\Sigma})$ is the multi-dimensional Gaussian distribution with mean $\boldsymbol{u_i}$ and covariation matrix $\boldsymbol{\Sigma}$. Then training set will be generated by $G(\boldsymbol{X})$.

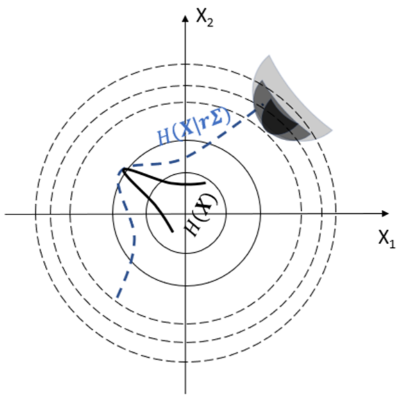

Figure 1. Scaled Shape Sampling.

4.2. Blockwise Model Solver

The derived objective function in Equation (11) is not a continuously differentiable function. The typically used Least Squares, Gradient Descent and Newton's method will not work. In this work, we developed the strategy of block coordinate descent (BCD) [14] to solve parameters efficiently. It selects the critical groups and iteratively minimizes the objective function in each group coordinate while the other coordinates are fixed. The solver is summarized in Algorithm 2.

Algorithm 2: Blockwise Model Solver

```
Input: The sample set $(X_i, y_i),\ i = 1, \dots, N$ generated by the constructed $G(X)$
Output: The model parameter $\Omega = (B_1, \dots, B_P, C)$
    1.    Initialization
        1.1. $B_1, \dots, B_P, c_1, \dots, c_{nr} \leftarrow 0$
        1.2 Set iteration $t = 0$
        1.3 The residual is:
            $R^* = y - \sum_{i=1}^{P} \mathrm{B}_i \Phi_i (L_i) - C\,\Psi(X_R)$
    2.    Repeat
    2.1 for $k = 1, ?? P$
    # Update $\mathrm{B}_k$
    2.2    $R \leftarrow R^* + \mathrm{B}_k^{(t)} \Phi_k (L_k)$
    2.3 $\mathrm{B}_k^{(t+1)} \leftarrow arg \min_{B_k} \frac{1}{2N} \|R - \mathrm{B}_k \Phi_k(L_k)\|_2^2 + (1-\alpha)\lambda \sum_{i=1}^{P} \|B_i\|_2 + \alpha\lambda \|\Omega\|_1$
    2.4    $\Delta = \mathrm{B}_k^{(t+1)} \Phi_k(L_k) - \mathrm{B}_k^{(t)} \Phi_k (L_k)$
    2.5    $R^* \leftarrow R^* + \Delta$
    2.6 endfor
    # Update $C$
    2.8 $R < -R^* + C\, ?\, (X_R)$
```

```
  2.9   $C \leftarrow \arg\min_{\beta_r} \frac{1}{2N}\|R - C\,\Psi(X_R)\|_2^2 + \alpha\lambda\|\Omega\|_1$
  2.10  $\Delta = C^{(t+1)}\,\Psi(X_R) - C^{(t)}\,\Psi(X_R)$
  2.11  $R^* \leftarrow R^* + \Delta$
  2.12  $t = t + 1$
Until satisfy the convergence:
                    $\frac{|Q^{(t)}(\Omega) - Q^{(t-1)}(\Omega)|}{Q^{(t-1)}(\Omega)} \le \varepsilon$
 3.     Compute the intercept $\beta_0$
  $\beta_0 = average(y - \sum_{i=1}^{P} B_i\Phi_i(L_i) - C\,\Psi(X_R))$
```

The ε is the specific convergence threshold. α and λ are regularization constants, which can be optimized by cross-validation [15].

In the step 2.3 and step 2.9, we optimize the coefficients of our basic functions. Our basic function consists of the RBF interpolation and polynomial to capture the local features and the global features of strong nonlinear curve, respectively. And the RBF interpolation has the following form:

$$\sum_{j=1}^{N} w_{i,j}\varphi_{i,j}(\boldsymbol{L_i}) = \sum_{j=1}^{N} w_{i,j} e^{(-\epsilon^2\|L_i - L_{i,j}\|^2)} \tag{12}$$

where $\|\boldsymbol{L_i} - \boldsymbol{L_{i,j}}\|$ is the L2-norm of $(\boldsymbol{L_i} - \boldsymbol{L_{i,j}})$ which can be viewed the radius of RBF. The RBF interpolation consists of N weighted RBF centered N samples which can accurately fit the local feature of an arbitrary curve. However, the RBF interpolation has two drawbacks: the convergence speed is slow when N is large because the weights w are obtained by calculating N^2 number of $e^{(-\epsilon^2\|L_i - L_{i,j}\|^2)}$, and the evaluation is instability when the radius is large because the L2-norm of two vectors will be much similar with increased dimension, which is also known as "curse of dimension" in measuring distance.

To address the first issue, we notice that it needs large number of samples to use RBF interpolation to fit the entire curve directly. We combine low-order polynomial to fit the curve globally and the RBF interpolation to fix locally to decrease N. In addition, we vectorize the RBF calculation so that we can utilize the special math kernel library (MKL) [16] embedded in MATLAB to calculate very fast with several milliseconds (In our experiment, we obtained over 1000x speedup compared with calculating weights by $N \times N$ nested loop). To address the second issue, we separate the parameter optimization of RBF interpolation and polynomial. We first solve the polynomial with L1 penalty and then we filter out the variable with zero coefficient to decrease the dimension of $\boldsymbol{L}$. If all the variables in a cell have no effects on SRAM performance, the entire grouped coefficients can be set "0" directly.

5. Experiment Result

The proposed SAIM based yield analysis method will be verified on the logical circuit path and a SRAM column, which both of them are frequently used in circuit design. And we extract the three process variables from PDK: Vth0_mis, u0_mis, Voff_mis, which represent the threshold mismatch, the electron mobility mismatch and cut-off voltage mismatch respectively. The parasitic parameters are directly extracted from post-layout circuits. Besides, we also implement OMP [7], LRTA [8] for model comparison and SSAIS [2], HDIS [3] for yield estimation comparison. All experiments are performed with MATLAB and HSPICE with 28nm TSMC model on the Server with Intel Xeon Gold 5118 CPU @ 2.30 GHz.

5.1. SRAM Path

A simplified SRAM path with sense amplifier (SA) is shown in Figure 2. The read operation begins by activating the first word-line (WL1) and the pre-charged bit-lines. One bit-line BL will discharge through the first accessed cell and enlarges the voltage difference between BL and BLB. The read (access) delay is defined as the time required to generate the voltage difference between two bit-lines that can be sensed by the sense amplifier. Notice that to generate the worst case for the read operation, the accessed Bit cell 1 stores “0” and other idle cells store “1”, which maximum the leakage current through idle bits to increase the read access delay and impede the successful read. We consider SRAM read delay as the performance metric, which the failure event happens when the read delay is beyond the specified threshold (i.e., the enable time of WL1).

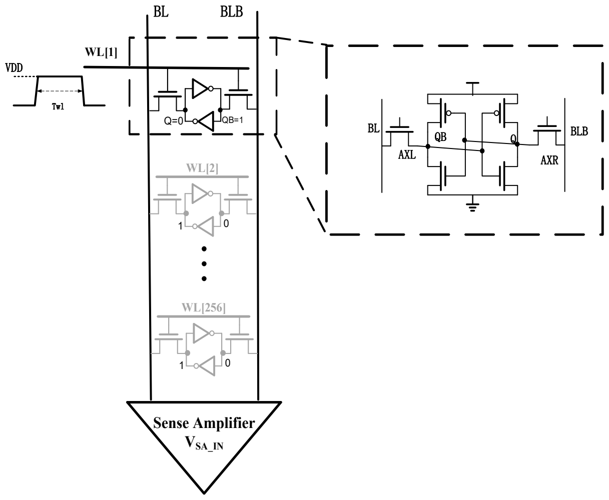

Figure 2. The simplified SRAM read path with sense amplifier.

5.2. Model Comparison

Our blockwise solver efficiency was first validated by comparing the convergence of our model with different configureures: the model with our blockwise solver and the model with generalized coordinate descent (GCD)[17]. The training set and test set were generated by $G(\boldsymbol{X})$. The simulations run parallel on the 10 cores Server. As shown in Figure 3, the model with blockwise solver quickly converges to around 3% relative error within 600 samples. While the model with GCD converges to around 8% relative error with 1200 samples.

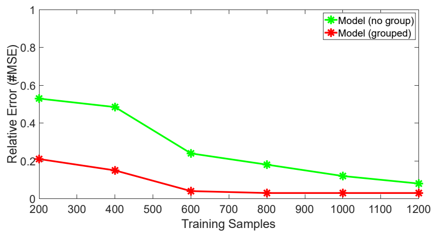

Figure 3. The training cost of model (grouped) v.s model (no group).

We also compared our model with the OMP and LRTA on different voltages with 1000 training samples. Both of them are high dimensional models. As shown the right original bars in Figure 4, our model converges to the relative error of around 3~5% on different voltages. While the accuracy of OMP and LRTA show high difference on different voltages. As the left bule bar and red bar shown in Figure 4, the relative errors of OMP and LRTA on 0.55V are up to 78% and 83%, respectively. The OMP and LRTA have a relative large bias compared with our model. It is because OMP and LRTA just are polynomial-based meta model. The basic functions of are polynomial which failed to capture the local features of the strong nonlinear curve. To illustrate it, we consider a univariate polynomial $F(x)$. The basis of $F(x)$ is just $(x, x^1, \dots, x^n)$ where n is the order of $F(x)$ and the maximum number of coefficients of $F(x)$ is n, which means n samples are enough to solve $F(x)$ and extra samples are useless. As a result, $F(x)$ can only fit the curve globally due to its limited components. Figure 5 shows the sorted read delay and predicted results of different basic functions when considering a single variable V_{th0}. As the black dot shown in Figure 5a), the read delay shows strong nonlinearity on 0.55V and 2~5 order polynomial failed to fit this curve. While in Figure 5b), the 4~5 order polynomial can fit the slightly nonlinear curve accurately under 0.7V. The accuracy of OMP and LRTA has improved greatly on high voltage.

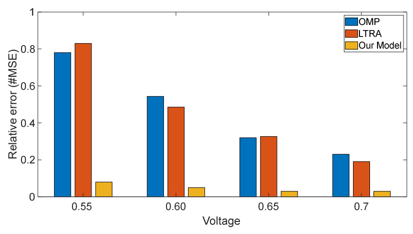

Figure 4. Accuracy comparison on different voltages.

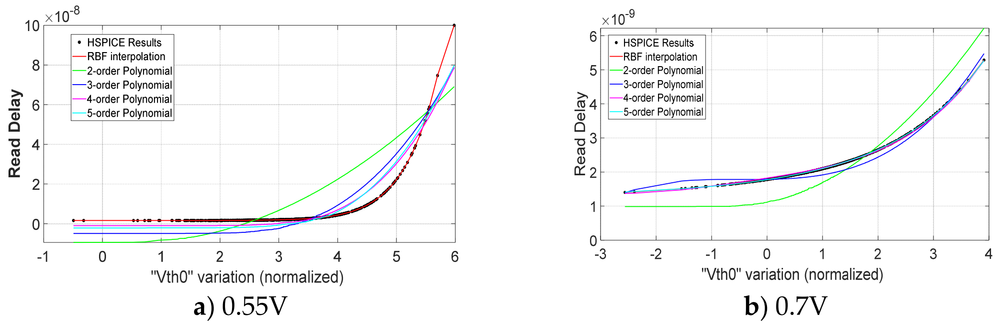

Figure 5. The fitting effects of different basic functions on different voltages.

Our model shows high accuracy across all voltages. It is because we use RBF interpolation to fit the nonlinear curve. The one-dimensional RBF interpolation has the form of $\sum_{i=1}^{N} w_i \varphi_i(\|\boldsymbol{x} - \boldsymbol{x_i}\|)$ which consists of N local functions centered $x_i, i = 1, \dots, N$. It can quickly learn the local features of arbitrary nonlinear function through several weighted RBFs close to the local sample. Figure 6 shows the fitting process of RBF interpolation.

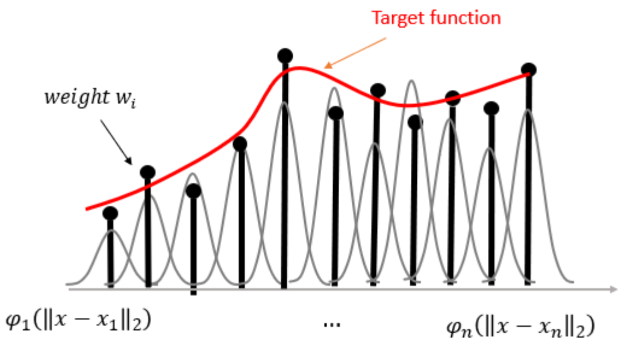

Figure 6. The fitting process of RBF interpolation.

5.3. Yield Estimation

The Figure of Merit (FOM) ρ defined in [4] is used to measure the convergence of different algorithm:

$$\rho = \frac{\sqrt{VAR_{\hat{P}_{fail}}}}{\hat{P}_{fail}} \tag{13}$$

where the $VAR_{\hat{P}_{fail}}$ is the variance of $\hat{P}_{fail}$. And $\rho < \varepsilon\sqrt{log(1/\delta)}$ means one estimation has reached $(1-\varepsilon)100\%$ accuracy with $(1-\delta)100\%$ confidence. In this work, we set $\rho = 0.1$ which means 90% accuracy with 90% confidence level.

For generalization, the failure rate is expressed as equivalent sigma points of a standard normal distribution,

$$Y_\sigma = \phi^{-1}(1 - P(y > T)) \tag{14}$$

where ϕ is the standard normal cumulative distribution function (CDF) and T is the design specification which is our specified read access time threshold. $Y_\sigma = 3$ implies a cumulative probability of 0.99865 and a failure probability of 0.00135.

We compare the accuracy and efficiency of our model with other methods. Figure 7 illustrates the converged failure rate estimated by different methods. The golden reference is 4.12 sigma obtained by NanoYield, a state-of-art commercial yield analysis tool, which cost 4.6 hours. And the SSAIS shows 8.0% relative error with 3.79 sigma due to the weight degeneration in high dimension. HDIS shows 6.3% relative error with 4.38 sigma, which is better than SSAIS for its bounded conditional failure rate calculation to relieve the weight degeneration. Our model shows the relative of 0.7% with 4.15 sigma.

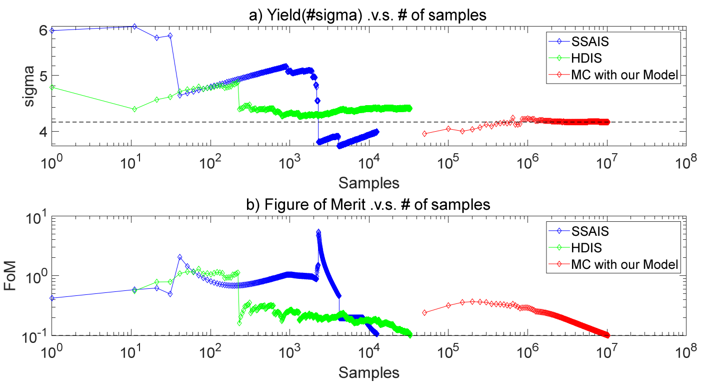

Figure 7. Yield convergence of different methods on 0.6V.

Both SSAIS and HDIS are sequence algorithms and the time cost is still large with 6 hours and 12 hours, respectively. Our model is trained with 600 samples. In additional, 3000 samples are used to search failure boundary. However, these samples are simulated parallel on the 10-cores server. And MC samples predicted by our model can be negligible compared simulation cost. Hence, the total cost of yield estimation based on our model is just 3.3 hours, which gains 1.4X speedup over NanoYield and 1.8X speedup over AIS and 3.6X speedup over HDIS. We also validate the yield prediction on different voltages. As shown in Figure 8, the maximum error of our model is 0.03 sigma and the average error is 0.01 sigma, which is very close to the golden reference.

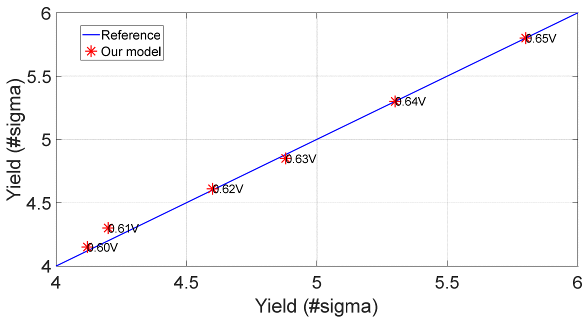

Figure 8. Yield estimation on different voltages.

6. Conclusions

In this paper, we proposed a sparse interactive meta model to accelerate the post-layout yield analysis. The model center is moved to the failure boundary by scaled shape sampling. And the model is derived in the group form, which can be solved very efficiently by our blockwise greedy algorithm. The experimental results show that our method is 1.8X-3.6X faster than other state-of-art methods with little accuracy loss.

Author Contributions: Conceptualization, Xudong Zhang.; methodology, Xudong Zhang; validation, Lugang Li and Ruisi Yan.; formal analysis, Ruisi Yan.; writing—original draft preparation, Xudong Zhang.; writing—review and editing, Lugang Li.; supervision, Ruisi Yan.

Funding: "This research received no external funding"

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Dong, Wei, Peng Li, and Garng M. Huang. "SRAM dynamic stability: Theory, variability and analysis." 2008 IEEE/ACM International Conference on Computer-Aided Design. IEEE, 2008.
2. Pang, Liang, Mengyun Yao, and Yifan Chai. "An efficient SRAM yield analysis using scaled-sigma adaptive importance sampling." 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2020.
3. Wu, Wei, et al. "A fast and provably bounded failure analysis of memory circuits in high dimensions." 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2014.
4. Dolecek, Lara, et al. "Breaking the simulation barrier: SRAM evaluation through norm minimization." Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design. IEEE Press, 2008.
5. Wang, Mengshuo, et al. "Efficient yield optimization for analog and sram circuits via gaussian process regression and adaptive yield estimation." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37.10 (2018): 1929-1942.
6. Yao, Jian, Zuochang Ye, and Yan Wang. "An efficient SRAM yield analysis and optimization method with adaptive online surrogate modeling." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 23.7 (2014): 1245-1253.
7. Li, Xin. "Finding deterministic solution from underdetermined equation: large-scale performance variability modeling of analog/RF circuits." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 29.11 (2010): 1661-1668.
8. Shi, Xiao, et al. "Meta-model based high-dimensional yield analysis using low-rank tensor approximation." Proceedings of the 56th Annual Design Automation Conference 2019. 2019.
9. T. B. B. Li and P. Bickel, "Curse-of-dimensionality revisited: Collapse of importance sampling in very high-dimensional systems," Technical Report No.696,Department of Statistics, UC-Berkeley, 2005.
10. Wright, Grady Barrett. Radial basis function interpolation: numerical and analytical developments. University of Colorado at Boulder, 2003.
11. Rithe, Rahul, et al. "Non-linear operating point statistical analysis for local variations in logic timing at low voltage." 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010). IEEE, 2010.
12. Aggarwal, Charu C., Alexander Hinneburg, and Daniel A. Keim. "On the surprising behavior of distance metrics in high dimensional space." International conference on database theory. Springer, Berlin, Heidelberg, 2001.
13. McGaughey, Georgia, W. Patrick Walters, and Brian Goldman. "Understanding covariate shift in model performance." F1000 Research 5 (2016).
14. Liu, Han, Mark Palatucci, and Jian Zhang. "Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery." Proceedings of the 26th annual international conference on machine learning. 2009.
15. Stone, M. "Cross-validation: A review." Statistics: A Journal of Theoretical and Applied Statistics 9.1 (1978): 127-139.
16. Wang, Endong, et al. "Intel math kernel library." High-Performance Computing on the Intel® Xeon Phi™. Springer, Cham, 2014. 167-188.
17. Wright, Stephen J. "Coordinate descent algorithms." Mathematical Programming 151.1 (2015): 3-34.