Multi-Objective Statistical Yield Enhancement using Evolutionary Algorithm

Minoo Mirsaeedi, Morteza Saheb Zamani, Mehdi Saeedi
Amirkabir University of Technology
{mirsaeedi, szamani, msaeedi}@aut.ac.ir

Abstract

It has been shown that several process parameters encounter variation in the very deep submicron era. Due to the increased power and performance variability, a multi-objective variability-aware yield optimization method is crucial. However, most of current yield optimization methods use greedy single objective optimization approach. In this paper, a comprehensive and multi-objective yield optimization framework is proposed to consider the effects of both power and performance yield degradation. In other words, an evolutionary gate sizing optimization approach is introduced to be used in the proposed framework to enhance yield, significantly. Compared with recent yield optimization algorithms [7] and [12], the proposed framework leads to better results for the attempted circuits. In addition, the independency of proposed framework on selected power and delay analysis techniques makes it suitable for future investigations as using dual threshold voltage assignment approach.

Keywords—Process variation, yield optimization, evolutionary algorithm.

1. Introduction

Process variation is becoming more and more significant in deep sub-micron technology. The growth of variability can be attributed to multiple factors, including the difficulty of manufacturing control, the emergence of new systematic variation-generating mechanisms, and most importantly, the increase in fundamental atomic-scale randomness, such as variations in the number of dopants in transistor channels. As a consequence, yield loss due to variability has become a challenging issue in nanometer technology. The yield loss augments design costs because dies that do not meet requirements with respect to performance are rejected.

On the other hand, due to the exponential dependency between threshold voltage and leakage power, a significant growth in leakage power is expected which leads to power dominancy in most of the future designs [1]. Moreover, leakage variability becomes more critical in yield loss due to the increasing impact of variability in low-voltage designs. Thus, any yield improvement method should consider leakage variability improvement as a primary objective [2].

Because of the random nature of process variations, statistical timing optimization is the main approach to enhance timing yield. Since larger gates encounter less delay variability, many of the proposed methods, size up gates which are located in critical paths to statistically decrease the critical path delay and improve yield loss consequently [3]- [6]. However, most of the leakage variability optimization methods assign smaller sizes or higher threshold voltages to those gates which are located in non-critical paths to minimize leakage variability [7], [8] and [9].

Several statistical methods have been proposed to counter yield loss due to timing and power variability in fully combinational designs [3]- [8]. Some of mentioned methods [9] use energy as their primary objective to consider power-delay trade-off. Other proposed methods try to enhance timing yield subject to area or power as a constraint or vice versa. However, an appropriate constraint value, which is unknown before the optimization process, is determinant of final solution optimality as neither rigid nor floppy constraints may lead to feasible solution.

Many optimization methods employ a greedy strategy based on sensitivity [3]-[5]. In this set of methods, the smallest legal sizes are assigned to gates as an initial solution; afterward, the gates are sized up based on the performance or power sensitivities at each iteration. Since sensitivity-based optimizers search the design space in a restricted neighborhood, it is possible to fall in a local optimum. Therefore, their optimization quality depends on the convexity of the search space.

On the other hand, to mitigate yield loss due to unpredictable skew in sequential designs, clock skew has been applied as a manageable resource to design faster and more skew tolerant circuits, see for example [10] and [11]. Furthermore, simultaneous skew
scheduling and gate sizing can be applied to alleviate area overhead as the main gate sizing drawback [12]. However, since skew scheduled circuits are more sensitive to combinational path delay variations, to prevent yield loss, it is required to consider the statistical delay of combinational paths during scheduling.

Therefore, we confront a set of contradictions between operational frequency, power consumption and manufacturing yield. Although the least area is preferred with respect to power consumption, larger gates provide less delay variation. In such situation, multi-objective optimization methods are useful to consider trade-off between contradictory objectives, efficiently. In this paper, an evolutionary design space exploration framework is proposed. The proposed framework is more efficient than current yield optimization methods considering the following factors:

- First, compared with analytical approaches, our proposed method can decide in both discrete and non-convex search space which is more practically applicable.
- Secondly, in contrast to the constrained methods, the proposed framework is independent of initial constraints. Moreover, since the proposed framework tries to find Pareto-optimal solutions, a wide variety of choices and a better vision about the trade-off between different objectives is provided for designer.
- Finally, due to using ECEA (Epsilon Constraint Evolutionary Algorithm) [13] as our evolutionary engine, our proposed framework can work integrated with available single objective optimization methods to decrease optimization run time.

The rest of the paper is organized as follows: Section 2 provides the relevant background information. Section 3 briefly introduces ECEA algorithm as the evolutionary core of the proposed framework. The statistical optimization framework is described in section 4. Section 5 presents experimental results and finally section 6 concludes the paper.

2. Basic concepts

In this section a brief overview on statistical gate sizing and skew scheduling methods is provided.

2.1 Statistical gate sizing

Conventional gate sizing problem attempts to resolve the conflicting requirements of decreasing both the circuit area and the circuit delay in a fully combinational circuit. This problem can be formulated as Equation (1):

\[
\text{Minimize Area} = \sum_{i \in \text{gates}} c_i s_i
\]

\[
\text{S.T.} \quad \begin{cases} 
  t_{a_i} \leq T_{\text{spec}} \\
  s_{\text{min}} \leq s_i \leq s_{\text{max}}
\end{cases}
\]

where \(s_i\) and \(t_{a_i}\) represent size of \(i\)th gate and signal arrival time to \(k\)th primary output respectively.

Similar to deterministic gate sizing, the objective of statistical gate sizing is to meet a timing constraint while minimizing total cell area or power. However, since the circuit delay is now a probability distribution, the timing constraint is probabilistic. Therefore, the statistical gate sizing problem can be formally defined as,

\[
\text{Minimize Area} = \sum_{i \in \text{gates}} c_i s_i
\]

\[
\text{S.T.} \quad Y(T_{\text{target}}) \geq Y_{\text{target}}
\]

where \(Y_{\text{target}}\) is a user-defined yield target and \(Y(\gamma)\) is the yield of a given circuit delay, \(\gamma\). Specifically, \(Y(\gamma) = 1-\text{CDF}(\gamma)\), since a larger slack has a lower yield. \(T_{\text{target}}\) is the same delay constraint as in the deterministic formulation.

As mentioned before, the final solution optimality depends on the prescribed constraints, \(T_{\text{target}}\) and \(Y_{\text{target}}\). Since setting constraints blindly can lead us to solutions far from global optimum, it is required to acquire vision about appropriate constraints.

2.2 Statistical skew scheduling

In general, given a combinational circuit that lays between two registers \(i\) and \(j\), \(\text{skew}_i\) and \(\text{skew}_j\) are the clock arrival times at the two subsequent registers, we have the following timing constraints:

\[
\text{skew}_i - \text{skew}_j + D_{ij}^{\text{max}} \leq T_{\text{clk}} - T_{\text{setup}}^j
\]

\[
\text{skew}_j - \text{skew}_i + T_{\text{hold}}^i \leq D_{ij}^{\text{min}}
\]

where \(D_{ij}^{\text{max}}\) and \(D_{ij}^{\text{min}}\) are the maximum and minimum combinational delays between registers \(i\) and \(j\).

In principle, since realizing dedicated skew for each register is impractical, multi-domain skew scheduling [14] is preferred. In this method, each clocking domain is derived from a higher frequency clock using some phase shifters. Therefore, using \(N\) frequency domains, the multi-domain clock skew scheduling problem can be defined as:
3. Epsilon constraint evolutionary algorithm

In this section, an overview of Epsilon Constraint Evolutionary Algorithm (ECEA) which is used as our evolutionary engine is provided. Similar to other multi-objective optimization methods, the final goal of this method is to find Pareto-optimal solutions.

Some of the proposed methods work by transforming the multi-objective problem into a sequence of parameterized single-objective problems such that the optimum of each single-objective problem corresponds to a Pareto-optimal solution. Compared with these methods, ECEA [13] determines the whole Pareto front by a number of single-objective sub-problems which depends only on the cardinality of the Pareto front and not on additional properties such as the location of Pareto-optimal solutions in objective space.

The epsilon-constraint method works by choosing one objective function as the only objective and the remaining objectives as constraints. By a systematic iteration of the constraint bounds, different elements of the Pareto-front are obtained. Each iteration of the original epsilon constraint method could be modeled as below:

\[
\text{minimize } \Phi(f(x)) = \Phi(f_1(x),...,f_m(x)) \\
\text{subject to } f_i(x) < \epsilon_i \quad \forall i \in \{2,...,m\}, \\
x \in X,
\]

where \( f : X \rightarrow \mathbb{R}^m \) and \( \Phi : \mathbb{R}^m \rightarrow \mathbb{R} \).

In the above equation, bound of \( i^{th} \) objective, \( \epsilon_i \), is iteratively refined by a predefined constant value, \( \delta \). Since both large and fine values for \( \delta \) can discard some optimal solutions or generate many redundant solutions, the necessity to choose an appropriate value for \( \delta \) is the main drawback of this approach.

To circumvent the deficit of the original epsilon-constraint method, ECEA tries to make use of information about the objective space during the search.
Algorithm 1. Adaptive m-objective Epsilon-Constraint Method

1: $P := \emptyset$; $Q := \emptyset$; $e_j := (-\infty, +\infty) \ orall 2 \leq j \leq m$
2: $i := (|P| + 1)^{m-1}$ // {initialization subregion counter}
3: $i := i - 1$
4: if $i < 0$ then stop // {no new solution found}
5: $(e, \epsilon') := \text{getBounds}(i, P)$; //get lower & upper bounds for all objectives
6: if $[e, \epsilon'] \in Q$ then goto 3. // {if subregion already searched}
7: $x := \text{opt}(f, \epsilon, \epsilon')$ // {solve single-objective problem}
8: if $x = \text{null}$ then // {if subregion empty}
9: $Q := Q \cup [e, \epsilon']$; goto 3
10: if $\exists y \in P : y \text{ dominates } x$ then
11: $Q := Q \cup [e, \epsilon']$; goto 3 // {if solution dominated}
12: $P := P \cup \{x\}$; $Q := Q \cup [e, f(x)]$
13: $e := \text{updateConstraints}(f(x), e)$
14: goto 3 // {new optimal solution was found}

Output: set of Pareto-optimal solutions $P$

In each iteration of the outer loop one new Pareto-optimal point, $x$, is sought. Afterwards, this point is added to the set of already found solutions, $P$, and the grid is updated by recording its objective values. If no feasible solution can be found or the solution is dominated by any previously found solution, the searched objective co-domain is marked by storing the lower bound vector $\bar{e}$ in the set $Q$.

Further to being independent from prescribed constraints, some inherent specifications of ECEA make it appropriate for our application. First, partitioning search space into different regions makes the trade-off between different objectives more sensible. Moreover, final population in ordinary genetic algorithms contains many individuals which are located near each other. However, preference of each individual is more visible in a partitioned population, as what ECEA offers, which makes final decision making easier. Finally, when an individual has been selected as final solution, using bounds of its area, an appropriate set of constraints can be provided for a single-objective optimization to locate the locally optimal solution.

4. Proposed framework implementation

In this section different aspects of the proposed multi-objective framework are discussed in detail. First, the evolutionary-based framework is characterized for optimization in fully combinational designs. Next, the framework is generalized for sequential designs.

4.1. Yield optimization in combinational designs

Different factors such as gate delay and power modeling are involved in a multi-objective evolutionary-based optimization framework. Furthermore, fair objectives should be considered to assess each individual’s fitness.

4.1.1 Statistical gate delay and power modeling

Logical effort delay model is used in our statistical optimizer which is a simple and accurate model in deterministic timing optimization [17] and is shown in Eq.(7).

$$d_{\text{eff}} = \tau (g h + p)$$

(7)

where $\tau$ represents the reference inverter delay and $g$, $h$, and $p$ are logical effort, electrical effort, and parasitic delay factors respectively. Among different terms of logical effort formula (Eq. (7)), process variations are influential just on $\tau$ [18]. Since the delay of the gate is independent of the gates in its predecessor gate level, logical effort model provides more effective pruning during timing analysis and reduces the timing analysis run time overhead.

Total power consumption can be expressed as the sum of dynamic power and leakage power. As it is shown in Eq. (8), dynamic power depends on its frequency ($f$), supply voltage ($V_{dd}$) and load capacitance ($C$). Since each gate capacitance is related to its size, total capacitance can be reasonably estimated by the design area.

$$P_{\text{dyn}} = f.C.V_{dd}^2$$

(8)

However, area cannot reflect the variability of power especially leakage power. In this work, we assume that variations in power are dominated by variations in subthreshold leakage power. Transistor
subthreshold leakage power dissipation is calculated as:

\[ P_{\text{sub}} = I_{\text{sub}} V_{dd} = e^{-c E_{\text{sub}}} \frac{e^{f(N_{\text{eff}})}}{V_{dd}} \]  

(9)

where \( L_{\text{eff}} \) and \( V_t \) represent nominal values of effective channel length and threshold voltage, respectively.

Suppose all process parameters are renamed \( z_i \) where all \( z_i \)'s are normally distributed. Now, based on Eq. (10) \( I_{\text{sub}} \) follow lognormal distribution and its mean and variance can be calculated using mean and variance of process parameters. In Eq. (10), \( \mu \) and \( \sigma \) correspond to mean and variance of linear summation of process parameters, \( z_i \).

\[ I_{\text{sub}} = \exp\left( a_0 + \sum_{i=1}^{n} a_i z_i \right) \]

\[ E[I_{\text{sub}}] = e^{a_0 + \sum_{i=1}^{n} a_i \mu_i / 2} \]  

\[ \sigma^2(I_{\text{sub}}) = e^{2a_0 + \sum_{i=1}^{n} a_i \sigma_i^2} \]  

(10)

Total leakage current is calculated through summation of leakage current of each transistor. However sum of two lognormal variables does not follow lognormal distribution, it can be approximated by a lognormal variable as following [19]:

\[ \sum_{i} \exp(z_i) \approx \exp(\sum_i z_i) \]

4.1.2 Statistical timing analysis and yield representation

To increase the robustness of a combinational design against process variation, it is required to reduce the delay variability in statistically critical paths. Accordingly, designs with less critical variability, \( \sigma_{\text{critical-delay}} / \mu_{\text{critical-delay}} \), are more appropriate respected to their timing yield. However, since less critical variability may be acquired by increase in \( \mu_{\text{critical-delay}} \), the nominal path delay should be considered as an independent objective. In order to increase the yield assessment accuracy, critical variability is defined as:

\[ \text{Critical variability tot} = \sum_{i=1}^{m} \left( \frac{\sigma_d}{\mu_d} \right) \]  

(11)

where a user defined parameter, \( m \), controls the number of paths which contribute in during yield evaluation and may vary between 1 and the number of designs’ primary outputs.

Nominal value and variance of path delays can be calculated using statistical timing analysis. Since timing graph pruning is more applicable in block-based methods [5], such methods are more appropriate for evolutionary frameworks where each individual may be very similar to its parents and its pruned timing graph analysis wastes less computation. In our framework, timing analysis is performed as [20].

Similar to timing analysis, power consumption can statistically calculated using the statistical gate’s power model provided in Eq. (10). After each generation, it is only required to compute the leakage current overhead of perturbed gates as follows:

\[ I_{\text{pert, total}} = I_{\text{unpert, total}} + \sum_{\text{in pert}} (g_i^{\text{unpert}} - g_i^{\text{unpert}}) \]  

(12)

where \( I_{\text{pert}} \) and \( I_{\text{unpert}} \) refer to perturbed and unperturbed leakage current of critical paths. As mentioned before, leakage current of each gate follows lognormal distribution.

Finally, the proposed ECEA-based optimization framework can formally be described as:

\[ \text{find } S \text{ minimize } \text{critical variability tot} \]

\[ \text{S.T.} \]

\[ \mu_{\text{max-delay}} < \epsilon_{\text{max-delay}} \]

\[ \text{Area} < \epsilon_{\text{Area}} \]

\[ \mu_{\text{pert, var}} < \epsilon_{\text{pert, var}} \]

\[ \sigma_{\text{pert, var}} / \mu_{\text{pert, var}} < \epsilon_{\text{var, var}} \]

\[ s_j^L \leq s_j \leq s_j^H \quad \forall j \in [1..N] \]  

4.2 General yield optimization framework

To generalize our yield optimization framework to deal with sequential designs, clock skew scheduling is appended to our framework to augment design’s robustness to both clock and combinational logic delay variations.

In order to satisfy setup and hold time constraints during skew scheduling, it is essential to schedule each register’s skew within its permissible range [21]. Considering setup and hold time constraints, slack of each subsequent register pairs can be defined as Eq.(14).

\[ \Delta(i, j) = \text{skew}_i + w(i, j) - \text{skew}_j \]  

(14)

\[ \Delta(j, i) = \text{skew}_j + w(i, j) - \text{skew}_i \]

Since positive slacks demonstrate a timing constraint violation, permissible range of each register can be defined such that no timing constraint violation occurs:

\[ r(i) = \left[ \frac{1}{2} \min \Delta(i, j), \text{skew}_j, \frac{1}{2} \min \Delta(j, i) \right] \]  

(15)

Although skew scheduling tries to assure timing constraints through skew assignment in permissible range, delay variations may violate constraints and degrade manufacturing yield.

In a nutshell, ECEA-based generic yield optimization framework can be formally expressed as...
Eq. (16). Please note that combinational paths’ delay variability is folded in path slacks. Also, $P_{sub}$ represents worst case subthreshold leakage power.

$$\text{minimize} \quad T_{\text{clock}}$$

Subject to:

$$\sum_{(k,j) \in E} \Delta(k, l) < \varepsilon_{\text{slack}} \quad \text{if} \quad \Delta(k, l) > 0$$

$$\text{Area} < \epsilon_{\text{Area}}$$

$$P_{sub} < \epsilon_{P_{sub}}$$

$$\text{size}_j^L \leq \text{size}_j \leq \text{size}_j^H, \quad \forall j \in \text{gates}$$

$$\text{skew}_k \in r(k) \cap \text{skew} \_ \text{domains}, \quad \forall k \in \text{FFs}$$

The fully evolutionary approach where both skews and gate sizes are perturbed in each generation implies permissible range computation for all registers after each generation. In a sequential design with $N$ registers, since each register can be connected to up to $N - 1$ registers, $\min \Delta(i, j)$ in Eq. (15) can be computed in $T_{\text{STA}} \cdot O(N)$ where $T_{\text{STA}}$ represents time complexity of statistical timing analysis. Consequently, the overall complexity of updating permissible ranges on perturbed skews is $T_{\text{STA}} \cdot O(N^2)$ which is really time-consuming.

On the other hand, as formulated in Eq. (4), given combinational path delays, clock skew scheduling can reasonably be performed using an ILP solver. Due to capability of ECEA to work integrated with a single objective optimizer, time complexity overhead of the framework can be alleviated in expense of negligible restriction search space exploration.

5. Implementation and experimental results

Implementation details are discussed in this section. Based on experimental results provided in [14], 6 discrete skew domains which are uniformly distributed along clock period are considered. Moreover, we assumed that gates have discrete sizes, ranging from 1x to 8x of minimum size. To extract statistical reference inverter delay from Spice Monte Carlo simulation, 70 nm Berkeley Predictive model was used [22] while parameters $L_{eff}$ and $V_{th}$ were considered as normal random variables.

Moreover, based on Pelgrom’s model [23], it is assumed that $\sigma_{Vth}$ of each transistor is inversely proportional to its size. As statistical gate delay, Monte Carlo simulation was utilized to model statistical subthreshold leakage for each size of nMOS/pMOS transistors.

Epsilon constraint evolutionary algorithm was implemented based on the standard implementation offered in [24]. The priority of optimization objectives was chosen similar to Eq. (16). To provide a demonstrative view about the capability of the proposed framework in considering power-performance tradeoffs, the quality of final population is considered in Figure 1 for design c432 from ISCAS’89 benchmark circuits.

![Figure 1. Power-performance tradeoff among final solutions for c432](image)

To generate initial population, first, a uniform set of power-performance constraints were generated. Next, gate sizes were determined using a greedy statistical sizing algorithm similar to [7]. It can be seen that final solutions have appropriate diversity over the search space. While some of the solutions can operate in higher frequencies with more variability and power consumption, less variable designs have lower operational frequency.

Figure 1 shows the relation between worst case power and operational frequency improvement. As illustrated in this figure, while power-performance trade off is obvious, several low-power high-speed designs could be found in the final solutions. Therefore, compared with a population generated by a greedy algorithm, our resulted Pareto front members have better characteristics because our method has more comprehensive insight about power-performance tradeoffs.

In a similar way, the proposed generic framework can produce more appropriate set of solutions compared with an analytical approach to simultaneous statistical gate sizing and skew scheduling [12]. Figure 2 shows the comparison between first and final populations for s838 from ISCAS’89 benchmark circuits. In this figure first population has been produced by analytical approach with different constraints and final population has been proposed by our evolutionary framework.

In Figure 2, while the population labeled “Final generation 1” has been produced by fully evolutionary approach, “Final generation 2” population has been resulted using the proposed evolutionary framework integrated with a linear programming solver. As it is
demonstrated, both populations' qualities are comparable. However, based on run time comparison provided in Figure 3, for some test cases from IWLS'05 benchmark sets, integrated evolutionary-analytical approach is considerable faster than fully evolutionary method.

To verify the results, a selected set of optimization results for IWLS'05 benchmark circuits are reported in Table 2. Among different designs in this benchmark set, designs with more registers and more complicated combinational blocks were chosen. Accordingly, the optimization framework has been verified in the highest level of complexity. The results were chosen such that the extreme corners of search space can be observed in the final population. It can be seen that for each design, a variety of choices are offered. In run time column, total run time of statistical timing and power consumption analysis and the perturbation time are separately reported. As can be seen, with increase in number of gates, due to natural specification of ECEA and effective pruning, the increase in total run time is reasonable. It should be noted that for each design the number of perturbations are determined according to its size and at least 100 perturbations were generated.

6. Conclusion

In this paper, a general multi-objective evolutionary framework was proposed for multi-objective gate sizing to optimize statistical yield. Proposed framework offers solutions in different domain of design space with acceptable operational frequency and leakage power consumption. To evaluate results, we compared the results with those from both the recent proposed evolutionary method and a greedy-generated one. More techniques like gate duplication and dual-Vth assignment could be used simultaneously in order to obtain better results.

7. References

statistical yield improvement,” accepted for ISVLSI 2008.


### Table 1 Results of our approach on IWLS benchmark

<table>
<thead>
<tr>
<th>Design Name</th>
<th>Gate #</th>
<th>C.P. delay (ns)</th>
<th>Delay variability</th>
<th>Power (μW)</th>
<th>Power variability</th>
<th>Run time (sec)</th>
<th>Σ STA</th>
<th>Σ Pert.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Regs</td>
<td>Total</td>
<td>$\sigma_{max_delay}$</td>
<td>$\mu_{max_delay}$</td>
<td>$\sigma_{P_sub_total}$</td>
<td>$\mu_{P_sub_total}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b21</td>
<td>490</td>
<td>18718</td>
<td>0.3283</td>
<td>4463.75</td>
<td>0.0137</td>
<td>15.06</td>
<td>58.34</td>
<td></td>
</tr>
<tr>
<td>b20</td>
<td>490</td>
<td>19107</td>
<td>0.3983</td>
<td>4513.2</td>
<td>0.0136</td>
<td>12.65</td>
<td>57.65</td>
<td></td>
</tr>
<tr>
<td>aes-core</td>
<td>530</td>
<td>20795</td>
<td>0.1359</td>
<td>5076.04</td>
<td>0.0122</td>
<td>13.48</td>
<td>109.12</td>
<td></td>
</tr>
<tr>
<td>b22</td>
<td>703</td>
<td>28317</td>
<td>0.1382</td>
<td>5052.49</td>
<td>0.0124</td>
<td>35.04</td>
<td>183.96</td>
<td></td>
</tr>
<tr>
<td>wb-conmax</td>
<td>770</td>
<td>29034</td>
<td>0.1522</td>
<td>5623.86</td>
<td>0.0146</td>
<td>26.88</td>
<td>165.62</td>
<td></td>
</tr>
<tr>
<td>ethernet</td>
<td>10544</td>
<td>46771</td>
<td>0.1109</td>
<td>300.27</td>
<td>0.036</td>
<td>23.36</td>
<td>210</td>
<td></td>
</tr>
</tbody>
</table>