Efficient Algorithms for Discrete Gate Sizing and Threshold Voltage Assignment based on an Accurate Analytical Statistical Yield-Gradient

Ramprasath S, Indian Institute of Technology, Madras
Vinita Vasudevan, Indian Institute of Technology, Madras

In this paper, we derive a simple and accurate expression for the change in timing yield due to a change in the gate delay distribution. It is based on analytical bounds that we have derived for the moments of the circuit and path delay. Based on this, we propose computationally efficient algorithms for (a) discrete gate sizing and (b) simultaneous gate sizing and threshold voltage (V_T) assignment so that the circuit meets a timing yield specification under parameter variations. The use of this analytical yield gradient within a gradient-based timing yield optimization algorithm results in a significant improvement in the run-time as compared to the numerical method, while achieving the same final yield. It also allows us to explore a larger search space in each iteration more efficiently, which is required in the case of simultaneous resizing and V_T assignment. We also propose heuristics for resizing/changing the V_T of multiple gates in each iteration. This makes it possible to optimize the timing yield for large circuits. Results on ITC'99 benchmarks show that the proposed multi-node resizing algorithm results in a significant improvement in the run-time with a marginal average area penalty and no cost to the final yield achieved.

CCS Concepts: •**Hardware** → **Electronic design automation; Statistical timing analysis;** *Yield and cost optimization;*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Timing Yield, Yield Optimization, Discrete gate sizing, V_T assignment

1. INTRODUCTION

With the entry into the nanometer regime of the process technologies, circuit optimization to meet timing specification with the deterministic worst case static timing analysis has become extremely pessimistic. Timing analysis and yield optimization require statistical models for the gate delay. Both the gate delays as well as the arrival and required time (AT and RT) at a node are random variables modelled using the canonical model [Visweswariah et al. 2004; Chang and Sapatnekar 2005]. Since the circuit delay is now a random variable, timing yield optimization requires statistical measures. A spectrum of approaches based on statistical static timing analysis (SSTA) have been proposed for yield optimization.

The approaches in the literature can broadly be classified into two types. The first class of methods are the ones that use nonlinear optimization techniques. These techniques pose the gate sizing problem as a continuous objective function. The methods proposed by [Jacobs and Berkelaar 2000; Choi et al. 2004; Mani and Orshansky 2004; Singh et al. 2005; Beece et al. 2010] and [Davoodi and Srivastava 2008] are examples of such an approach. In [Jacobs and Berkelaar 2000], LANCELOT [Conn et al. 1992] is used as the optimization engine, with the assumption that the arrival times (AT) at edges converging on a node are independent. In [Mani and Orshansky 2004], gate

 1084-4309/2015/-ARTXXXX $15.00
DOI: http://dx.doi.org/10.1145/0000000.0000000

sizing is framed as a linear programming (LP) problem with a linear model for the gate delay in terms of size. The gate delay variations are taken into account by adding to the gate delay a node margin function, which is essentially a multiple of the standard deviation of the gate delay. This multiple (node margin coefficient) is assumed to be the same as that in the target yield, making it overly pessimistic. Once again, the gate delays are assumed to be independent of each other. A Lagrangian relaxation is used in [Choi et al. 2004], to get a modified objective function, with gate sizes as the parameters. Only the variance of the circuit delay is used in the optimization and the gate sizes are determined with deterministic delays in each iteration. [Singh et al. 2005] use a geometric programming approach, with an uncertainty ellipsoid to model variations in transistor widths and lengths. The disadvantage of these techniques is that solution to the optimization may not be integer sizes for the transistors and thus require rounding to the nearest possible integer. This rounding operation could significantly alter the solution obtained.

The second type of techniques are the ones that try to solve the discrete problem directly. They define a statistical yield gradient (YG), which is the change in the timing yield for a unit change in the size. To evaluate the change in yield, an SSTA run is performed to obtain the change in the circuit delay due to a change in the size of the gate i.e., the YG is evaluated numerically. The optimization algorithms used are iterative in nature and in each iteration, the YG of a large number of gates needs to be evaluated in order to pick the best candidate for resizing. These two issues make the algorithm unsuitable for large circuits. To reduce the number of YG evaluations, [Guthaus et al. 2005] use criticality of the nodes to select a subset of gates whose YG is to be evaluated. In [Sinha et al. 2006], instead of using the circuit delay to evaluate the YG, they use an alternate figure of merit that involves performing SSTA on a subnetwork of each of the gates considered for resizing. This subnetwork involves two fan-in and fan-out levels of the gate under consideration. In [Srivastava et al. 2008], to evaluate the new circuit delay, the ATs of all the perturbed gates until the level of the resized gate are updated. The RTs till the level of the resized gate are assumed to remain the same. However, this could lead to errors since the slew change on the resized gate affect the downstream delays, which in turn affects the required time (RTs). [Davoodi and Srivastava 2008] use binning yield loss as the objective of optimization. In this approach, SSTA is used to obtain the change in binning yield loss that is required for gradient evaluation. This is computationally expensive. [Tang and Jha 2015] frame the yield optimization, leakage and dynamic power minimization as a multi-objective optimization problem and try to solve it using genetic algorithm. They use node criticalities as a heuristic to reduce the size of the search space. The criticality evaluation technique they use is from [Visweswariah et al. 2006], which assumes independence between ATs. The disadvantage with the use of genetic algorithm is that its applicability is limited to smaller size designs.

An alternative to the numerical approach is to evaluate YG using analytical expressions[Xiong et al. 2008; Beece et al. 2010]. Analytical expressions for the yield gradient have been derived in [Xiong et al. 2008], but no methodology/results on how these expressions are used for timing yield optimization have been included. Moreover, they do not include the effect of fan-in and fan-out gates on the yield gradient, which will lead to errors. [Beece et al. 2010] use a non-linear optimizer with the analytical YG expressions to perform sizing at transistor level assuming availability of continuous transistor widths. However, they write the analytical YG as a function of all edges in the graph, which makes the evaluation of analytical YG almost as expensive as the numerical YG.

Gradient-based algorithms have also been used for minimizing statistical leakage while meeting the timing yield specification. In [Srivastava et al. 2004], first the cir-

cuit is optimized for timing using TILOS with the lowest threshold voltage (V_T) cell library alone. For V_T assignment, they use a statistical sensitivity metric that is local to the gate and easy to compute. However, their algorithm involves multiple upsizings and potentially several SSTA runs every-time a timing violation is detected, which is expensive. [Hwang et al. 2013] use two metrics sequentially for the optimization. The first is local to the gate and easy to compute. But the second, the swap sensitivity, requires an incremental SSTA for its computation, making it expensive. In [Mani et al. 2007], once again in the first phase the circuit is sized using low V_T devices alone. This is followed by a statistical slack assignment phase, which is framed as second order conic program (SOCP). Node margin coefficients are used in the SOCP similar to [Mani and Orshansky 2004]. Node margin coefficients are evaluated using path criticalities found using the algorithm specified in [Zhan et al. 2005], which makes it run-time intensive. These coefficients are also pessimistic since the worst of all coefficients obtained from various paths through a node are used.

In this paper, we explore the use the YG as defined in [Xiong et al. 2008] for (a) statistical timing yield optimization using discrete gate sizing and (b) minimization of the nominal leakage power while meeting a timing yield constraint using both gate sizing and V_T assignment. The two main issues with the gradient based algorithms are the potentially large number of nodes for which the YG needs to be evaluated before the size/V_T of a gate is changed (candidate list) and the lack of an efficient method to evaluate the gradient itself. In this paper, we tackle both these issues. We use an analytical YG based approach to tackle the discrete gate sizing/V_T assignment problem directly. We improve the accuracy of the expressions derived in [Xiong et al. 2008] by taking into account the effect of resizing on the fan-in and fan-out gates. To efficiently evaluate the yield gradient of a large number of gates in each iteration, we derive a simple and accurate expression for the change in the timing yield due to a change in the gate delay distribution. This derivation is based on bounds that we have derived for μ, σ and σ/μ of the path and circuit delay. We have used this expression for the yield gradient to optimize the statistical timing yield with (a) area and (b) nominal leakage power as the cost, corresponding to the discrete gate sizing and simultaneous gate sizing and V_T assignment problem. Even with this simplified yield gradient, the cost of optimization becomes prohibitive for large circuits. To address this problem, we propose a heuristic for resizing multiple gates in an iteration, which improves the run-time considerably with a marginal area/leakage power penalty.

The paper is organised as follows. Section 2 describes a generic gradient based discrete gate sizing algorithm and the issues involved. In Section 3, we derive a simplified expression for the analytical yield gradient. Section 4 contains the algorithm for resizing multiple nodes in every iteration of the optimization. Section 5 describes the schemes used in performing simultaneous gate sizing and V_T assignment. Section 6 contains the results of various experiments performed on ITC'99 benchmarks. Section 7 concludes the paper.

2. GRADIENT BASED ALGORITHMS

Since the linear canonical model is used to represent the edge delays and AT, the circuit delay distribution can be written as a Gaussian random variable with a probability density function (PDF) $\mathcal{N}(\mu_c, \sigma_c)$. The timing yield ($Y$) of the circuit for a timing specification T_{spec} is given by:

$$Y = \Phi\left(\frac{T_{spec} - \mu_c}{\sigma_c}\right) \tag{1}$$

where Φ is the CDF of the standard normal distribution. It can be improved through change in either the size or V_T of the gates. Let λ be a generic cost that has to be minimized while improving the timing yield. For example, λ could be area or nominal value of the leakage power.

The yield gradient (YG) is defined as the expected change in yield for a unit change in cost λ. A typical discrete gate sizing algorithm that optimizes the circuit for timing yield using the YG is shown in Algorithm 1. In Algorithm 1, a candidate list is first created that contains the list of gates that could potentially improve the yield. For each of the gates in the candidate list, the change in yield (ΔY) due to a change in its size/V_T is computed. The yield gradient is evaluated as the ratio $\frac{\Delta Y}{\Delta \lambda}$, where $\Delta\lambda$ is the change in cost due to a change in the size/V_T of the gate. The gate with the largest YG is replaced and the new circuit delay is evaluated. This process is repeated until the target yield is met. For discrete gate sizing/V_T assignment, $\Delta\lambda$ is evaluated using the area/nominal leakage power available from the standard cell library data.

ALGORITHM 1: Optimize Circuit for Target Yield:

```
1: Perform forward SSTA to find AT of all nodes
2: repeat
3:    Create list of candidate gates(L)
4:    for all gates g ∈ L do
5:       Change the size/V_T of gate g
6:       Perform an incremental SSTA to obtain the change in timing yield ΔY
7:       Compute the change in cost Δλ
8:       Yield gradient of gate g, YG_g = ΔY/Δλ
9:       Restore size of g
10:   end for
11:   Sort L based on decreasing order of YG
12:   Change size/V_T of gate having the largest YG
13:   Perform incremental SSTA and update the circuit delay
14:   Find the new yield {δYield is the change in yield between subsequent iterations}
15: until ( Yield < Target Yield or δYield < ε)
```

From Algorithm 1, it is clear that an incremental SSTA is necessary to obtain the change in yield for every gate in the candidate list, making it computationally the most demanding step. To overcome this, we use an analytical expression for the YG, as mentioned in the introduction. In the next section, we derive simplified expressions for the analytical YG, which is both accurate and has run-time efficiency.

3. EFFECTIVE YIELD GRADIENT

We first define some terms that are required for the computation of the analytical yield gradient.

(1) Path delay (PD): Path delay associated with a node is defined as the maximum delay of all the paths that pass through the node. For a node n, PD is evaluated as the sum of its AT and RT i.e., $PD_n = AT_n + RT_n$.
(2) Complementary path delay (CPD): The CPD associated with a node is defined as the maximum delay of all the paths that do not pass through the node.

Using the definition of PD and CPD of a node, the circuit delay d_c can be written as the MAX(PD,CPD). If x represents the PD of node n and y its CPD, then the circuit delay

d_c can be written as:

$$d_c = \max(x, y) = \mathcal{N}(\mu_c, \sigma_c) \tag{2}$$

$$\implies \mu_c = f(\mu_x, \sigma_x, \mu_y, \sigma_y, C_{xy}), \quad \sigma_c = g(\mu_x, \sigma_x, \mu_y, \sigma_y, C_{xy}) \tag{3}$$

where C_{xy} is the covariance between x and y and the functions f and g are given by Clark's formulas [Clark 1961].

The analytical YG for a node n is given by:

$$\text{YG}_n = \frac{\mathrm{d}Y}{\mathrm{d}\mu_c} \times \frac{\mathrm{d}\mu_c}{\mathrm{d}\lambda} + \frac{\mathrm{d}Y}{\mathrm{d}\sigma_c} \times \frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda} \tag{4}$$

$$= \phi\left(\frac{T_{spec} - \mu_c}{\sigma_c}\right) \times \frac{1}{\sigma_c} \times \left[-\left(\frac{\mathrm{d}\mu_c}{\mathrm{d}\lambda}\right)_n - \left(\frac{T_{spec} - \mu_c}{\sigma_c}\right)\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n\right] \tag{5}$$

where ϕ is the PDF of the standard normal distribution. $\dfrac{\Delta\mu_c}{\Delta\lambda}$ can be evaluated using the chain rule as:

$$\frac{\Delta\mu_c}{\Delta\lambda} = \frac{\mathrm{d}\mu_c}{\mathrm{d}\mu_x} \times \frac{\Delta\mu_x}{\Delta\lambda} + \frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_x} \times \frac{\Delta\sigma_x}{\Delta\lambda} + \frac{\mathrm{d}\mu_c}{\mathrm{d}C_{xy}} \times \frac{\Delta C_{xy}}{\Delta\lambda} \tag{6}$$

A similar expression can be written for $\dfrac{\Delta\sigma_c}{\Delta\lambda}$. Appendix A contains the expressions for the quantities $\frac{\mathrm{d}\mu_c}{\mathrm{d}\mu_x}$, $\frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_x}$ and $\frac{\mathrm{d}\mu_c}{\mathrm{d}C_{xy}}$ and the corresponding derivatives of σ_c. They are obtained using Clark's formula. The rest of the quantities are evaluated using data from the standard cell libraries. This is demonstrated using the example circuit shown in Fig. 1. Let x represent the PD associated with node n and x' represent the PD after

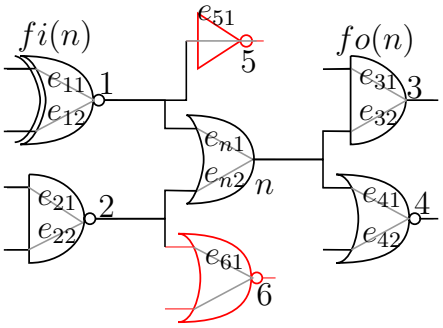


Fig. 1: Example circuit

the edge delays associated with e_{n1} and e_{n2} (d_{n1} and d_{n2}) are changed due to either a change in the size or the V_T of gate n. x and x' can be written as:

$$x = \max(AT_1 + d_{n1} + RT_n, AT_2 + d_{n2} + RT_n) \tag{7}$$

$$x' = \max(AT_1 + d'_{n1} + RT_n, AT_2 + d'_{n2} + RT_n) \tag{8}$$

$\Delta\mu_x$, $\Delta\sigma_x$ and ΔC_{xy} are given by:

$$\Delta\mu_x = (\mu_{x'} - \mu_x), \quad \Delta\sigma_x = (\sigma_{x'} - \sigma_x), \quad \Delta C_{xy} = (C_{x'y} - C_{xy}) \tag{9}$$

Evaluation of ΔC_{xy} requires the canonical form of the CPD. Using the reversible MAX operation described in [Sinha et al. 2012], CPD evaluation is a constant time operation for every node considered. Unlike this, the numerical YG techniques require at the

least as many MAX operations as the number of edges in the subnetwork [Sinha et al. 2006]. So analytical YG has a clear advantage in terms of run-time over the numerical YG. The values obtained from (9) can be substituted in (6) to obtain YG_n.

3.1. Fan-in/out effects

A change in the size/V_T of gate n has the following effects

(1) Increasing the size increases the load seen by the fan-in gates thereby increasing their delay and output slew.
(2) Siblings of n (fan-out gates of fan-in of n) now have an increased delay due to the change in their input slew.
(3) Fan-out gates of n now have a reduced delay due to the change in slew at their inputs.
(4) Changing only the V_T of a gate n changes its own delay and also its output slew. The delay of the gates in the fan-out cone will now potentially change due to the change in their input slew.

Neglecting the change in the delay of the fan-out gates will give a more pessimistic YG and it slows down convergence. Neglecting the fan-in effects results in an optimistic YG. In fact, in many cases the YG evaluated is positive but it turns out that it is actually negative. The reason is that the increased delays of the fan-in gates will propagate through their fan-out cones and cause a potential increase in the circuit delay. Unless both these effects are taken into account the change in yield will not be evaluated correctly. Therefore, an effective yield gradient (EYG) that considers these effects is obtained using chain rule. For example, in Fig. 1, the delays of the gates 1, 2, n, 3, 4, 5 and 6 are affected. This implies that the chain rule for evaluating $\frac{\Delta\mu_c}{\Delta\lambda}$ can be written as:

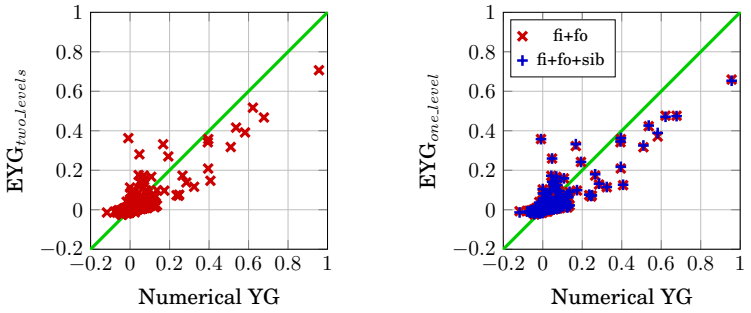
$$\frac{\Delta\mu_c}{\Delta\lambda} = \sum_{i\in\{1,2,n,3,4,5,6\}} \left[\frac{\mathrm{d}\mu_c}{\mathrm{d}\mu_i} \times \frac{\Delta\mu_i}{\Delta\lambda} + \frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_i} \times \frac{\Delta\sigma_i}{\Delta\lambda} + \frac{\mathrm{d}\mu_c}{\mathrm{d}C_{i\bar{i}}} \times \frac{\Delta C_{i\bar{i}}}{\Delta\lambda}\right] \tag{10}$$

where $C_{i\bar{i}}$ is the covariance between the PD and CPD associated with node i. A similar expression can be written for $\frac{\Delta\sigma_c}{\Delta\lambda}$. These two expressions can then be used to compute YG_n. YG_i for node i is evaluated in the same way as YG_n in equation (4) and (6) by considering the changes in their corresponding PDs. Therefore, the effective yield-gradient (EYG_n) for node n can be written as:

$$\text{EYG}_n = \sum_{i\in fi(n)} \text{YG}_i + \text{YG}_n + \sum_{j\in fo(n),fo(fi(n))} \text{YG}_j \tag{11}$$

Here, $fo(n)$ indicates all gates in the fanout cone of gate n and $fo(fi(n))$ indicates the fan-out cone of the predecessors of n. However, if all the gates in the fan-out cone are considered to evaluate the EYG, then there is not much advantage with respect to the numerical YG technique and one could use the numerical YG instead. The question is how many levels of fan-out are required to evaluate the EYG with reasonable accuracy. Fortunately, in practice, the change in delays are not significant after one or two levels of fan-out as seen in [Coudert 1997; Sinha et al. 2006]. Fig. 2 compares the EYG obtained using one and two levels of fan-out for b19 benchmark against the numerical YG. It is clear from the figure that there is no significant difference between one and two levels of fan-out and hence a single fan-out level can be considered to evaluate the EYG. Figure 2 also compares the effect of including the sibling gates of n (gates 5 and 6 in Figure 1) in evaluating the EYG. It is clear from the figure that the inclusion of

the sibling gates provides very little improvement in the accuracy of EYG. This is the case since the dominant components of the EYG arise from n and its fan-in/out. This effect is also seen by [Li et al. 2012].



(a)b19 (b)b19

Fig. 2: Comparison of numerical YG and EYG evaluated using one/two levels of fan-out gates for two of the large ITC'99 benchmarks

3.2. Accuracy of the EYG

The accuracy of EYG depends on the accuracy to which the coefficients given in Appendix A are computed. This in-turn depends on the accuracy of criticality computation and computation of the coefficients of the CPD. The accuracy of the criticality computation is discussed in [Ramprasath and Vasudevan 2012; 2014]. The computation of the CPD depends on the accuracy of the reversible MAX operation. This has an accuracy similar to criticality computation [Ramprasath and Vasudevan 2014]. Even with these errors, EYG follows a trend that is similar to the numerical YG as shown in Fig. 2.

Besides computational errors, there are still outliers where the numerical YG is close to zero but the EYG is a relatively large non-zero value. One such scenario is illustrated in Fig. 3. Consider the case where the EYG of node **c** is to be evaluated and (**s-a-b-c-e-f-g-t**) is the dominant path in the circuit, followed closely by the path (**s-a-b-d-e-f-g-t**). The PDs of these two paths are typically highly correlated since they share most of the nodes. In such a scenario, **d** gets close to zero criticality and **c** gets a larger criticality. But the numerical YG of node **c** will be close to zero, since upsizing **c** makes the path (**s-a-b-d-e-f-g-t**) dominant, so that the circuit delay remains almost the same. But the EYG could still be a large value as it only considers the changes in PD of **b**, **c** and **e**. Although in this case the analytical EYG differs from the numerical

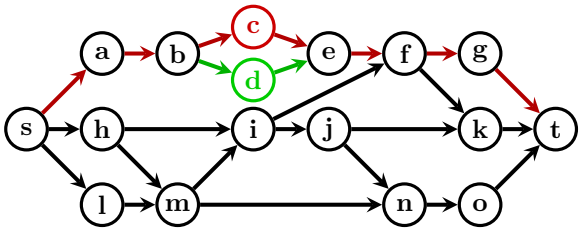


Fig. 3: Illustration demonstrating one of the sources of error in EYG when compared against Numerical YG

YG, it has an advantage. If node **c** were resized using the analytical EYG, node **d** which was overshadowed by **c** is now exposed and can be resized in the future iterations.

3.3. Using EYG for timing yield optimization

In order to test the effectiveness of the analytical EYG for discrete gate sizing, we used it for yield optimization of the ITC benchmarks. For this experiment, only the change in size of gates is considered and λ is chosen as the sum of area of all the cells. Evaluation of the EYG requires node criticalities, which were computed using the algorithm described in [Ramprasath and Vasudevan 2014]. It uses a pruning algorithm to improve accuracy and run-time efficiency. The pruned nodes have a marginal effect on the circuit delay and the corresponding gates are therefore removed from the candidate list. In many circuits, this is very effective in limiting the number of gates for which the EYG is evaluated in each iteration. The gate with the largest positive EYG is picked for upsizing and an incremental SSTA run is done to obtain the modified circuit delay and the corresponding yield. These steps are shown in Algorithm 2.

ALGORITHM 2: Optimize Circuit for Target Yield:

```
Perform forward/reverse SSTA to find AT/RT of all nodes
Evaluate criticalities of all nodes
repeat
    Prune non-dominant nodes
    List of candidate gates(L): set of all unpruned nodes
    for all gates g ∈ L do
        Change the size of gate g
        Update the edge delays of gates in fan-in and required number of fan-out levels of g
        Evaluate EYG of g using the updated edge delays
        Restore size of g and the perturbed edge delays to their original values
    end for
    Sort the list of gates in L based on EYG
    if largest EYG > γ then
        Change size of gate having the largest EYG {γ is a small positive threshold}
    else
        STOP
    end if
    Perform incremental SSTA and update the AT/RT/PD of all nodes
    Find the new yield {δYield is the change in Yield between subsequent iterations}
until ( Yield < Target Yield OR δYield < ϵ)
```

Table I compares the area overhead and final yield after optimization using three different EYG using Algorithm 2:

(1) Without considering fan-in/fan-out effects
(2) Considering fan-in and one level of fan-out
(3) Considering fan-in and two levels of fan-out
(4) Considering fan-in, one level of fan-out and sibling gates

It is clear from the table that the inclusion of the fan-in/out gates generally results in a better final yield at a smaller area overhead. The difference is above 10% for b06, b08, b11 and b19, being as high as 41% in b11. In some of the benchmarks however (i.e. b09, b12, b20 and b21), the inclusion of fan-in/out effects results in a marginally lower final yield. This is due to the threshold γ in Algorithm 2. In our algorithm, positive EYGs below γ are treated as zero, to limit the number of iterations. The value used is

between 10^{-4} and 10^{-6}. Table II, shows that the yield can be improved by setting the threshold to zero, but the area overhead also increases as expected.

All the benchmarks have practically the same final yield and increase in area upon considering either one or two levels of fan-out. The results are also similar on inclusion of the sibling gates. Since the effect of sibling gates is marginal, they are not used for EYG evaluation in the rest of this paper.

Benchmark	Area overhead(%)[Final yield (%)]			
	No fan-in / fan-out	One-level fan-out	Two levels fan-out	one-level fan-out + siblings
b01	118.79[86.96]	23.76[96.62]	37.59[96.72]	22.34[96.66]
b02	72.78[88.72]	10.65[95.46]	10.65[95.46]	12.43[95.55]
b03	16.13[94.63]	7.93[98.35]	7.93[98.35]	8.20[98.35]
b04	13.20[96.81]	3.42[98.69]	3.32[98.66]	2.81[98.62]
b05	24.31[97.91]	4.75[99.02]	4.75[99.02]	4.25[99.03]
b06	60.06[83.52]	14.33[95.72]	14.33[95.72]	14.88[95.75]
b07	3.99[99.87]	3.01[99.80]	3.01[99.80]	2.97[99.80]
b08	36.62[81.65]	8.89[95.97]	10.05[95.90]	8.70[95.96]
b09	22.59[83.20]	6.73[82.40]	6.73[82.40]	6.53[82.34]
b10	45.75[94.62]	11.01[99.30]	11.01[99.30]	8.77[99.26]
b11	14.08[53.42]	4.45[94.68]	4.57[94.78]	4.02[94.71]
b12	7.30[99.65]	2.65[99.49]	2.78[99.54]	2.45[99.36]
b13	11.89[99.37]	2.02[99.86]	2.07[99.87]	1.15[99.69]
b14	8.09[93.05]	2.98[96.15]	3.06[96.19]	2.72[95.81]
b15	4.34[99.16]	1.25[98.68]	1.23[98.66]	1.12[98.55]
b17	1.21[99.22]	1.70[99.87]	1.65[99.87]	1.38[99.87]
b18	3.74[88.56]	2.83[97.01]	2.97[97.01]	2.03[97.47]
b19	2.93[84.40]	2.17[96.01]	2.26[95.83]	2.13[97.11]
b20	2.19[99.40]	1.20[98.53]	1.28[98.70]	1.00[98.36]
b21	1.09[99.71]	0.55[99.33]	0.57[99.34]	0.47[99.21]
b22	0.52[99.69]	0.59[99.87]	0.59[99.87]	0.50[99.87]
Average	22.46[91.60]	5.57[97.18]	6.30[97.19]	5.28[97.21]

Table I: Comparison of area penalty and final yield on using YG a) without considering fan-in/out effects b) considering fan-in and one level of fan-out, c) considering fan-in and two levels of fan-out

Benchmark	Area overhead (%)[Final yield (%)]
b09	9.73[85.33]
b12	4.46[99.87]
b20	1.47[99.87]
b21	0.85[99.87]

Table II: Final yield and area overhead upon lowering γ (in Algorithm 2) to zero.

3.4. Simplified expression for the EYG

Based on the following bounds, simplified expressions for the EYG are obtained:

(1) The upper bound for $\frac{\sigma}{\mu}$ of the PD is the maximum value of $\frac{\sigma}{\mu}$ over all edges in the circuit graph.
(2) $\sigma_c \leq \max(\sigma_x, \sigma_y)$.
(3) $\sigma_c \geq F \times \min(\sigma_x, \sigma_y)$ where $F = \sqrt{1 - \frac{1}{\pi}}$.

The proof for the first two bounds is given in [Ramprasath and Vasudevan 2012], while the proof for the third is included in Appendix B. In addition, we assume that the correlation coefficient between the PD and CPD does not change on replacing a gate. This is reasonable since the change in the correlation coefficient between paths due to change in the size of a single gate is marginal. Appendix C details the steps involved in obtaining simpler expressions for YG_n along with the results used and assumptions made to arrive at the result. From the appendix, $\frac{\Delta\mu_c}{\Delta\lambda}$ and $\frac{\Delta\sigma_c}{\Delta\lambda}$ can be written as:

$$\frac{\Delta\mu_c}{\Delta\lambda} \approx \sum_{i\in\{1,2,n,3,4\}} \frac{\mathrm{d}\mu_c}{\mathrm{d}\mu_i} \times \frac{\Delta\mu_i}{\Delta\lambda} \text{ with } \frac{\mathrm{d}\mu_c}{\mathrm{d}\mu_i} \approx \Phi(\alpha_i)$$
$$\frac{\Delta\sigma_c}{\Delta\lambda} \approx \sum_{i\in\{1,2,n,3,4\}} \frac{\mathrm{d}\sigma_c}{\mathrm{d}\mu_i} \times \frac{\Delta\mu_i}{\Delta\lambda} \text{ with } \frac{\mathrm{d}\sigma_c}{\mathrm{d}\mu_i} \approx f(\alpha_i, t)\frac{\theta_i}{2\sigma_c} \quad (12)$$

where $f(\alpha_i, t)$ is defined in equation (16) in Appendix A.

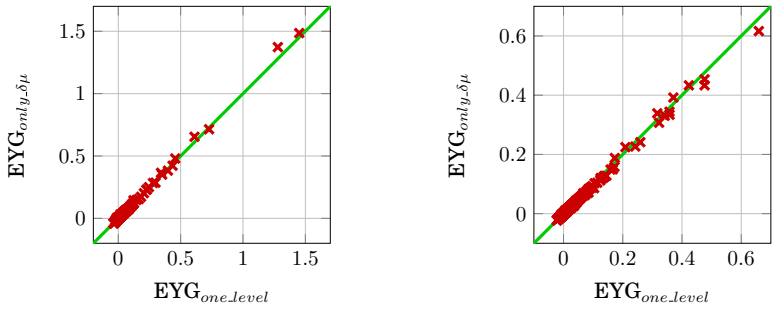
As discussed in the previous section, the EYG of all (unpruned) nodes have to be recomputed every iteration. This in turn requires computation of criticalities and the canonical form of the CPD. Computation of criticalities cannot be avoided and the techniques described in [Ramprasath and Vasudevan 2014] can be used to evaluate it efficiently. The canonical form of the CPD for all nodes is computed using the reversible MAX operation, which is potentially expensive if there are lots of principal components in the canonical form. With the simplified expression for the EYG, this step is eliminated leading to improvements in the run-time.

This approximation is validated by the results shown for two of the largest ITC'99 benchmarks in Fig. 4. It contains the EYG values of all gates in the circuit that were considered for resizing. The figure compares the EYG obtained using $\Delta\mu_x$, $\Delta\sigma_x$ and ΔC_{xy} terms against using $\Delta\mu_x$ terms alone. It can be see that the two match very closely. These trends were observed for the other benchmarks too. This simplified expression has practical value in terms of run-time since for each evaluation of the yield-gradient, there is only $\Delta\mu_x$ to be evaluated. This is illustrated with the improvement in run-time in Table III. The improvement is as high as 35% for the b05 benchmark. The final yield and the area penalty reported in the table also shows using only $\Delta\mu_x$ to evaluate EYG produces results similar to that on using the complete EYG.

4. MULTI-NODE RESIZE

Instead of resizing one node per iteration as described in section 3.3, multiple nodes can be upsized simultaneously in every iteration. For resizing multiple nodes in an iteration we need to look at the effect of changing the size of node n on the EYG of another node m. If EYG_m were to remain positive even after resizing the node n, then m and n can be resized simultaneously.

Changing the size of a node n changes the EYG of another node m due to a potential change in the PD and CPD associated with that node. This change in the PD and CPD is reflected as a change in α_m (see equation (12)). If the criticality of node m is already



(a)b18 (b)b19

Fig. 4: Comparison of EYG evaluated using $\Delta\mu_x$, $\Delta\sigma_x$ and ΔC_{xy} and using $\Delta\mu_x$ alone for some of the large ITC'99 benchmarks

Benchmark	Area overhead(%)[Final yield(%)]		Improvement in run-time(%)
	EYG$_{complete}$	EYG$_{only_\Delta\mu}$	
b01	23.76[96.62]	24.11[96.64]	18.49
b02	10.65[95.46]	10.65[95.46]	11.45
b03	7.93[98.35]	7.93[98.35]	10.36
b04	3.42[98.69]	3.10[98.65]	33.45
b05	4.75[99.02]	4.49[99.09]	35.46
b06	14.33[95.72]	14.33[95.72]	17.16
b07	3.01[99.80]	3.01[99.80]	19.39
b08	8.89[95.97]	8.89[95.97]	12.92
b09	6.73[82.40]	6.02[82.35]	20.75
b10	11.01[99.30]	8.86[99.20]	28.82
b11	4.45[94.68]	4.25[94.54]	33.72
b12	2.65[99.49]	2.36[99.24]	24.61
b13	2.02[99.86]	1.35[99.73]	22.88
b14	2.98[96.15]	2.81[95.98]	21.95
b15	1.25[98.68]	1.13[98.52]	22.95
b17	1.70[99.87]	1.82[99.87]	15.36
b18	2.83[97.01]	2.52[96.84]	10.78
b19	2.17[96.01]	2.16[95.41]	9.53
b20	1.20[98.53]	1.12[98.26]	18.88
b21	0.55[99.33]	0.50[99.15]	15.09
b22	0.59[99.87]	0.59[99.87]	13.12

Table III: Comparison of area penalty and the final yield achieved using EYG evaluated with $\Delta\mu_x$, $\Delta\sigma_x$ and ΔC_{xy} and with $\Delta\mu_x$ alone for ITC'99 benchmarks. Also shown is the percentage improvement in run-time on using $\Delta\mu_x$ alone.

very high (large α_m), the change in its EYG will be marginal as seen from Figure 5. The figure shows that for large α, the change in the coefficients due to a change in α is small. On the other hand, the figure shows that for medium and low criticality nodes, the coefficients can change significantly with a changing α. However, we only wish to eliminate nodes that would be falsely resized i.e. nodes for which EYG changes from positive to negative due to a perturbation in α_m. This typically occurs when EYG of node m is small to begin with. Therefore, we can use a threshold for the EYG and resize all the gates above the threshold simultaneously.

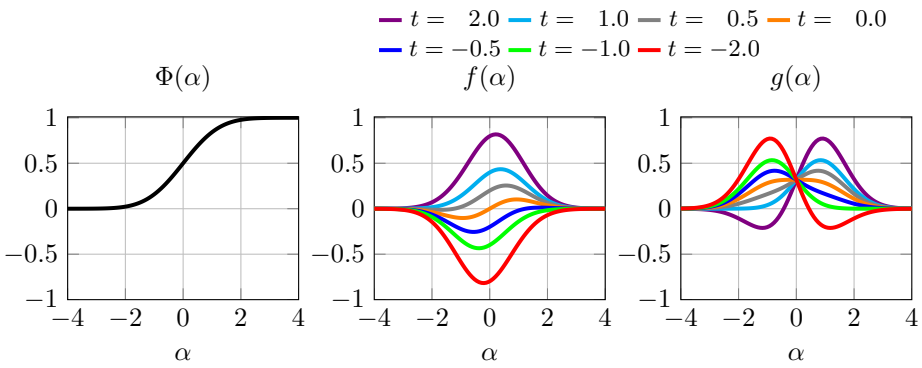


Fig. 5: Plots of coefficients used in evaluating the derivatives required for computation of the YG

We use a dynamic threshold that is a fraction (k) of the maximum EYG (EYG$_{max}$) in every iteration. Dynamic threshold is used because the values of EYG change in every iteration and can have a different range of values for different circuits. The trade-off here is the run-time versus area penalty. With multi-node resize, there will always be some gates that are resized unnecessarily resulting in an area penalty. However, this increase is very marginal as will be seen in the results.

5. SIMULTANEOUS GATE SIZING AND V_T ASSIGNMENT

Standard cell libraries with multiple threshold voltages (V_T) are typically used to reduce the leakage power. In order to optimize timing yield while minimizing the nominal value of the leakage power, the cost λ in the EYG formulation is changed to leakage power. With this, EYG$_n$ for node n can be written as:

$$\frac{\Delta Y}{\Delta P} = \phi\left(\frac{T_{spec}-\mu_c}{\sigma_c}\right) \times \frac{1}{\sigma_c} \times \left[-\left(\frac{\mathrm{d}\mu_c}{\mathrm{d}P}\right)_n - \left(\frac{T_{spec}-\mu_c}{\sigma_c}\right)\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}P}\right)_n\right] \tag{13}$$

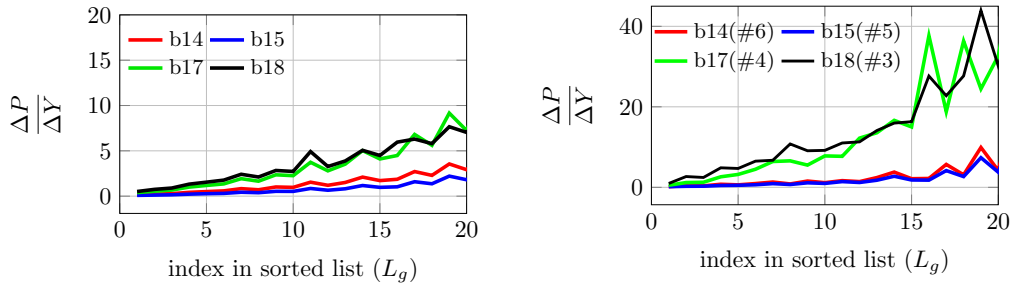
where ΔP represents the change in total leakage power of the circuit upon changing the size or V_T of a gate. Depending on the state of the system, the leakage power can increase or decrease every-time a gate is changed, so that ΔP can be positive or negative. Since our objective is to minimise power, we redefine EYG as EYG=$\frac{\Delta P}{\Delta Y} = \left(\frac{\Delta Y}{\Delta P}\right)^{-1}$ and choose the gate with the minimum EYG as the candidate for change. Once again, the simplified expression for the EYG can be used for the optimization.

In the case of gate sizing, there exists a monotonic relationship between the size of a gate and its edge delays. Upsizing a gate guaranteed a reduced nominal delay value of the gate at the cost of increased power/area. Decreasing V_T also guarantees this monotonic relationship. However, there are also scenarios like simultaneously increasing size and increasing V_T which may or may not lead to a decreased nominal delay. As a result, the search space is much larger and unless the metric (ΔY) can be evaluated efficiently, the optimization becomes impractical for large circuits. Since our expression for the analytical EYG can be evaluated efficiently, we can afford to have a larger search space. Nevertheless, we also propose some algorithms for pruning the search space and replacing multiple gates in each iteration, making it possible to optimize large circuits. The other issue is the initial state for the optimization algorithm. Although the final solution is relatively insensitive to the initial state, the run time is significantly affected. In this paper we have explored two possibilities. The first uses

the lowest leakage power state as the initial state for the algorithm. This corresponds to assigning the highest V_T and lowest size to all gates in the circuit. This state is denoted as I1. In the second case, we first do a deterministic gate-sizing and V_T assignment so that the circuit meets the nominal timing specification while minimizing the nominal leakage power. This is obtained using Design Compiler from Synopsys. Following this, gates corresponding to all the low criticality nodes that are pruned before the optimization procedure, are replaced by the lowest leakage equivalent. This state, denoted I2, is then used as the initial state for the statistical timing yield optimization. For each of the initial states, a major issue is the search space or equivalently, the number of EYG evaluations in each iteration. This is discussed in more detail in the following subsections.

5.1. Optimization starting with I1

With a view to prune the search space, corresponding to each gate type, we create a list L_g of (size,V_T) combinations. The list is sorted in increasing order of the leakage power. Figure 6 shows $\frac{\Delta P}{\Delta Y}$ for a resized gate in the i^{th} iteration as a function of the index in the list. It is clear that the EYG increases as the index increases so that the combination with lowest EYG is most likely to be the immediate neighbour in the list. We also have the additional constraint that ΔY should be positive. With this in mind, we propose the steps outlined in Algorithm 3(a) to pick the candidate gate. Corresponding to the initial state, the index of each list points to the first element in the list L_g. The EYG of size/V_T combinations that have an index higher than the current index is evaluated until a combination that has a positive ΔY is obtained. The EYG of this combination is assumed to be the EYG of the gate and the index is advanced to this element. This is reasonable since the EYG is only likely to increase further with an increasing index. For the optimization, as before, the candidate list L in Algorithm 2 contains the list of all unpruned gates. Steps 7 to 10 in Algorithm 2 are now replaced by Algorithm 3(a) and the gate with the smallest EYG is chosen for replacement.



(a)First iteration

(b) i^{th} iteration shown in the legend as #i

Fig. 6: $\frac{\Delta P}{\Delta Y}$ evaluated in Algorithm 3(a) for various indices in the sorted list L_g for the gate whose (size,V_T) was changed in the i^{th} iteration of optimization

5.2. Optimization using I2

The state I2 contains gates with (size,V_T) combinations that have been optimized so that the circuit meets the nominal timing specification. Therefore corresponding to the initial state the index in L_g is in an arbitrary position for each gate. To optimize the timing yield, both V_T and size can be either reduced or increased. All candidates with a higher V_T and size could potentially improve the yield with reduced leakage power.

ALGORITHM 3: Select next gate(Gate g, List $L_g(g)$, Index i)

g: gate type;
$L_g(g)$: list of gates of type g sorted based on leakage power;
i: index of current (size,V_T) of gate in list L(g)

(a)I1–Explore (size,V_T) till ΔY>0

1: $j \leftarrow i+1$
2: $EYG \leftarrow -\infty$
3: $\Delta Y \leftarrow -\infty$
4: **while** j <size(L(g)) **and** $\Delta Y < 0$ **do**
5: Evaluate ΔY using the j^{th} element of $L_g(g)$
6: $\Delta P \leftarrow$ Leakage_Power(j) - Leakage_Power(i)
7: **if** $\Delta Y > 0$ **then**
8: EYG $\leftarrow \frac{\Delta Y}{\Delta P}$
9: **end if**
10: Increment j
11: **end while**
12: **return** EYG, (size,V_T) of $(j-1)^{th}$ index of L_g

(b) I2(i) – Explore higher V_T and size first then explore using L_g

1: $(l, V_{T_m}) \leftarrow$(size,V_T) of index i in $L_g(g)$
2: $(p, q) \leftarrow (l+1, m-1)$ $\{V_{T_{m-1}} > V_{T_m}\}$
3: EYG $\leftarrow -\infty$
4: $\Delta Y \leftarrow -\infty$
5: **while** Leakage_Power(p, V_{T_q}) < Leakage_Power($l+1$, V_{T_m}) **and** $\Delta Y < 0$ **do**
6: Evaluate ΔY using the (size p, V_{T_q})
7: $\Delta P \leftarrow$ Leakage_Power(p, V_{T_q})− Leakage_Power(l, V_{T_m})
8: **if** $\Delta Y > 0$ **then**
9: EYG $\leftarrow \frac{\Delta P}{\Delta Y}$
10: **return** EYG, (size p,V_{T_q})
11: **end if**
12: Increment p
13: **end while**
14: $j \leftarrow i+1$
15: **while** j <size(L(g)) **and** $\Delta Y < 0$ **do**
16: Evaluate ΔY using the j^{th} element of $L_g(g)$
17: $\Delta P \leftarrow$ Leakage_Power(j) - Leakage_Power(i)
18: **if** $\Delta Y > 0$ **then**
19: EYG $\leftarrow \frac{\Delta Y}{\Delta P}$
20: **end if**
21: Increment j
22: **end while**
23: **return** EYG, (size,V_T) of $(j-1)^{th}$ index of L_g

However, if the V_T is much higher, this probability drops rapidly. Therefore, we only search among candidates with the next higher V_T. If a positive ΔY is not obtained, the search continues in the sorted list L_g starting with the current variant of the gate. This method of searching is depicted pictorially in Figure 7(a). This is also shown in Algorithm 3(b). The first while loop in Algorithm 3(b) searches through the higher V_T gates and the second while loop searches the list L_g. We also tried to prune the search space further by ignoring the sorted list and searching for a suitable combination as shown in figure 7(b). Although it led to a significant improvement in the runtime for the same final yield, the final leakage power was much higher in some of the larger benchmarks. This is illustrated in Table IV, which shows the increase in leakage power for some of the large ITC benchmarks upon using the two search schemes in Fig. 7. Therefore, for all results, the method in Figure 7(a) was adopted.

6. RESULTS

This section comprises of two subsections. The first part contains results obtained on using discrete gate sizing to achieve target yield with the area as the cost. The second part contains the results for the simultaneous gate sizing and V_T assignment experi-

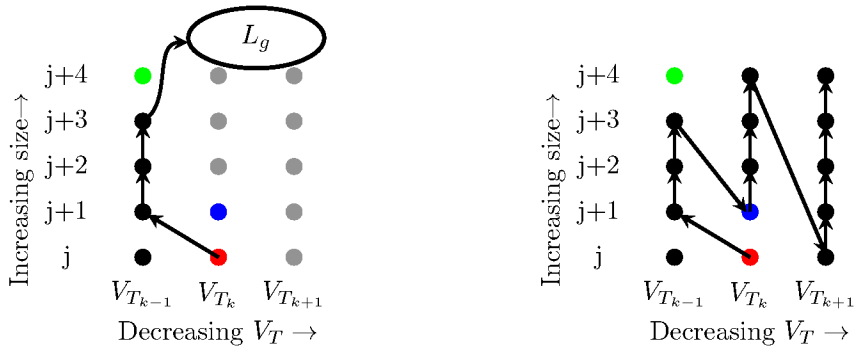


Fig. 7: Pictorial representation of the search of (size,V_T) using two different approaches. Red node corresponds to the current (size,V_T) of the gate. Green is the first node with $V_{T_{k-1}}$ having leakage power greater than the blue node.

Benchmark	δP(%)	
	I2(i)	I2(ii)
b17	183.98	182.86
b18	69.15	72.85
b19	42.94	96.36
b21	86.98	145.07
b22	82.93	158.29

Table IV: Comparison of percentage increase in final leakage power between the two methods to choose (size,V_T) shown in Fig. 7. $\delta P\% = \frac{\text{Final leakage - Nominal leakage}}{\text{Nominal leakage}} \times 100\%$, where nominal leakage is the leakage power of the circuit state which meets the timing at nominal corner as provided by Synopsys Design Compiler.

ments. For both cases, we used the ITC'99 benchmarks which were synthesized using the UMC 65nm libraries. For experiments that used area as the cost, the gates used for synthesis had the same threshold voltage. For the simultaneous V_T assignment and gate sizing experiments, the benchmarks were synthesized using the full library which has gates with three different V_Ts. For process variations, gates were assumed to have a 10% $(\frac{\sigma}{\mu})$ for V_T, L and W of transistors. The independent random component was assumed to have a sensitivity that is 5% of the nominal delay. To model the spatial correlation, we used the quad-tree structure with the number of layers in the quad-tree varying from two to seven depending on the size of the benchmark.

6.1. Discrete gate sizing

Table V compares the area overhead, final yield and improvement in run-time between the numerical YG algorithm and the simplified expression for the EYG(EYG$_{only_\delta\mu}$) for some of ITC'99 benchmarks. The larger benchmarks b18 and b19 did not achieve convergence on using numerical YG even after 7+ days of run-time and hence are not included in the table. It is clear from the table that the analytical YG achieves results similar to the numerical YG at a fraction of the run-time. Analytical YG is on an average $\sim$500$\times$ faster than the numerical YG method.

Figures 8, 9 and Table VI contain the results obtained on resizing multiple nodes in each iteration. The final yield achieved using different values of threshold factor k is

Bench-mark	Circuit size # edges[# gates]	Final yield(%)[Area overhead(%)]		Run-time (s)	
		Numerical YG	EYG$_{only_\delta\mu}$	Numerical	EYG$_{only_\delta\mu}$
b10	324[121]	99.28[9.40]	99.20[8.86]	18.65	0.71
b11	776[282]	92.69[4.41]	94.54[4.25]	168.71	2.69
b12	1823[688]	99.87[4.46]	99.24[2.36]	720.12	5.36
b13	520[211]	99.85[7.65]	99.73[1.35]	21.36	0.27
b14	7946[2750]	97.00[4.21]	95.98[2.81]	82586.01	138.89
b15	11990[3985]	99.51[4.09]	98.52[1.13]	196229.73	211.64
b17	38829[12829]	99.87[0.76]	99.87[1.82]	752031.43	1332.47
b20	16736[5696]	99.87[2.58]	98.26[1.12]	424247.07	269.86
b21	17166[5901]	99.87[0.75]	99.15[0.50]	123906.29	111.64
b22	24807[8472]	99.87[0.43]	99.87[0.59]	145922.30	210.26
Average		98.77[3.87]	98.44[2.48]	172585.17	228.38

Table V: Comparison of final yields, area overhead and ratio of run-times between the analytical and numerical yield gradients for some of the ITC'99 benchmarks

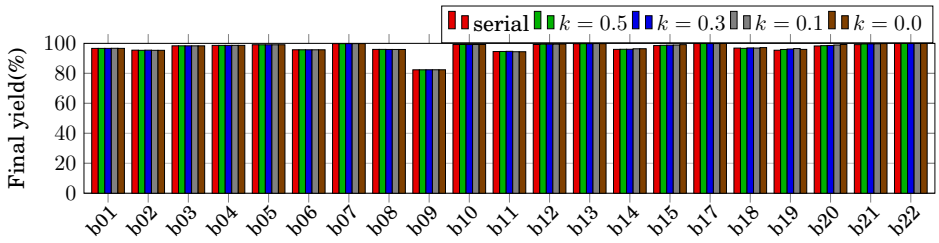


Fig. 8: Comparison of final yield achieved at the end of optimization for different values of k

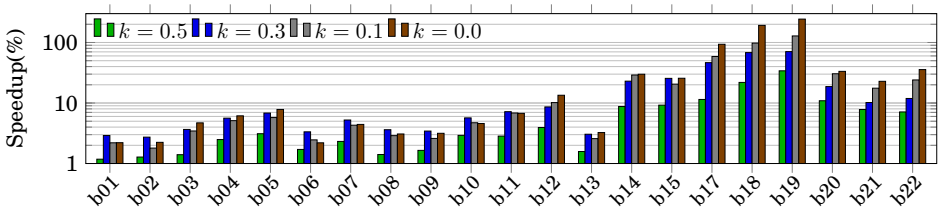


Fig. 9: Speedup with respect to the serial resizing for different values of k

plotted. $k = 0$ corresponds to resizing all nodes with positive EYG in every iteration. For all the cases the simplified expression for the EYG is used. Fig. 8 shows that the final yields for the different thresholds are comparable to the serial case. Choosing all nodes with positive EYG for resizing in each iteration has the maximum benefit in terms of run-time($\sim$35$\times$) at an average $\sim$1.5% increase in area overhead over the serial case.

6.2. Simultaneous gate sizing and V_T assignment

Figure 10 compares the final yield achieved on using the two different initial conditions I1 and I2 and using Algorithms 3(a) and (b) to choose the (size,V_T) of gates. It is clear from the figure that the final yield achieved using the two initial conditions is practically the same for all the benchmarks. Table VII compares the final leakage power and the number of iterations for convergence between using the two different initial states I1 and I2. For the larger benchmarks, convergence was not achieved even after 2+ days of run-time. So the multi-node resize described in the section 4 was used

Benchmark	Serial	$k = 0.5$	$k = 0.3$	$k = 0.1$	$k = 0.0$
b01	24.11	25.18	22.70	25.89	28.37
b02	10.65	10.65	10.65	10.65	10.65
b03	7.93	7.93	8.20	8.20	8.20
b04	3.10	3.10	3.10	3.10	3.32
b05	4.49	4.82	4.82	4.58	4.68
b06	14.33	14.33	14.60	14.33	14.60
b07	3.01	3.01	3.01	3.01	3.34
b08	8.89	8.89	8.89	8.70	8.70
b09	6.02	6.02	6.33	6.33	6.63
b10	8.86	9.49	8.86	9.49	9.49
b11	4.25	4.41	4.57	4.73	4.69
b12	2.36	2.67	2.79	2.95	3.89
b13	1.35	1.83	1.83	1.78	1.97
b14	2.81	2.90	2.83	3.21	3.40
b15	1.13	1.27	1.31	1.47	1.74
b17	1.82	1.74	1.47	1.44	2.09
b18	2.52	2.89	2.78	3.02	4.27
b19	2.16	2.00	1.90	1.84	2.46
b20	1.12	1.19	1.24	1.42	1.79
b21	0.50	0.66	0.71	0.66	1.14
b22	0.59	0.62	0.63	0.56	1.11
Average	16.36	16.72	15.98	16.96	17.87

Table VI: Comparison of increase in area for various values of $\text{EYG}_{thresh} = k \times \text{EYG}_{max}$ for ITC'99 benchmarks

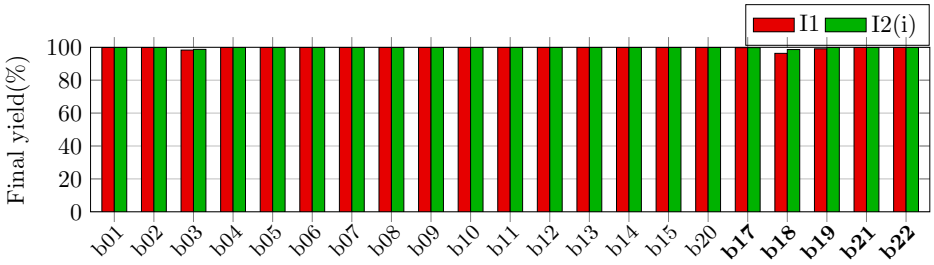


Fig. 10: Comparison of final yield on using two different initial states.

to improve the number of iterations for convergence. It is clear from the table that the final leakage powers attained in the two cases are almost the same in most cases, indicating that the final state is more or less independent of the initial state. The number of iterations for convergence is also significantly smaller when initial state I2 is used, which leads to the decreased runtime as seen in the speedup column of the table.

Fig. 11(a) and (c) compare the PDFs obtained PDF obtained by performing SSTA on the following states of the circuit:

— DC PDF: circuit state provided by the Synopsys Design Compiler
— I1 init. PDF: All gates are assigned highest V_T and lowest size
— I2 init. PDF: All gates which were pruned in the first iteration are assigned highest V_T and lowest size

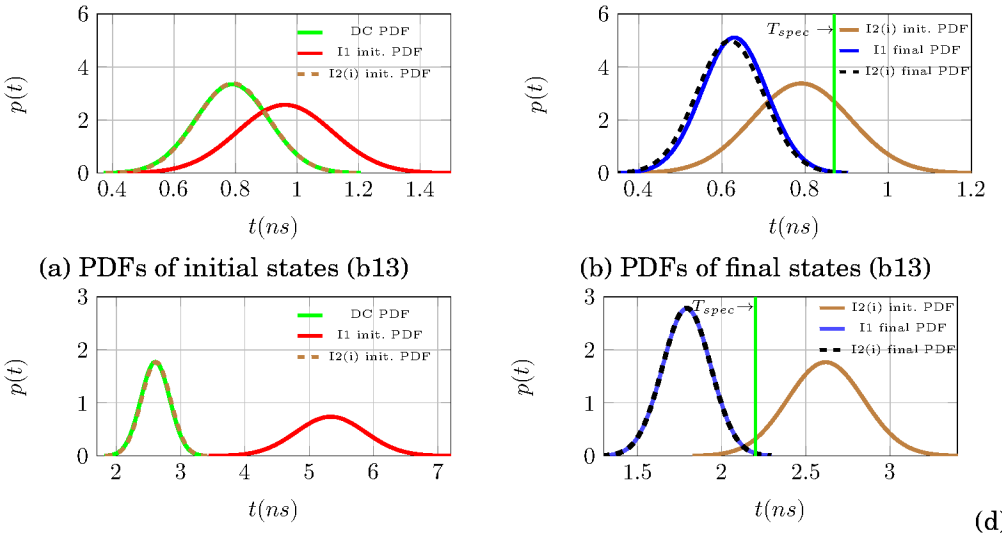
Bench-mark	Final power(μW)		# iterations		Speedup
	I1	I2(i)	I1	I2(i)	
Serial resizing					
b01	0.26	0.27	60	52	1.19
b02	0.15	0.16	42	31	1.66
b03	4.84	4.85	621	454	1.69
b04	1.63	1.73	85	38	3.66
b05	1.71	1.83	161	105	1.92
b06	0.24	0.26	49	27	2.36
b07	1.11	1.19	57	37	2.10
b08	0.59	0.61	75	39	2.46
b09	0.62	0.62	78	29	3.85
b10	0.61	0.65	67	29	3.75
b11	1.32	1.40	94	40	4.16
b12	2.96	3.07	91	26	5.17
b13	1.05	1.10	59	30	2.67
b14	23.99	24.85	4980	2667	3.27
b15	41.17	40.21	8392	4468	2.80
b20	70.23	73.34	13821	8595	1.37
Multi-node resize: Threshold = $2\times$EYG$_{min}$					
b17	232.53	226.42	2193	828	2.76
b18	466.13	474.38	2540	381	8.99
b19	852.81	590.95	3834	175	12.78
b21	64.39	67.51	521	137	2.34
b22	95.97	97.89	649	162	4.73
Ave	88.78	76.82	1832	874	3.60

Table VII: Comparison of final power, number of iterations for convergence and speedup between using the two initial states I1 and I2. Speedup = $\frac{\text{Run-time(I1)}}{\text{Run-time(I2(i))}}$.

It is clear from the Figures 11(a) and (c) that assigning highest V_T and lowest size to the pruned gates has practically no effect on the circuit delay. Figures 11(b) and(d) compare the final PDFs obtained on using the two different initial states. From the figures, it is clear that the final PDFs match very closely independent of the initial states.

Table VIII compares the increase in leakage power with respect to the circuit state provided by Design Compiler. The values are typically negative when the initial yield was $> 50\%$ i.e. $\mu_c < T_{spec}$ like the example in Figure 11(a). When the T_{spec} is aggressive like the case in Figure 11(c), the increase in leakage power is positive. This behaviour is expected, since an aggressive T_{spce} requires larger number of gates to be in lower V_T or higher size state leading to an increased leakage power.

Table IX compares the area and power overhead obtained using (a) Gate resizing alone and (b) Gate resizing and V_T assignment. In both cases, the lowest V_T variant was used for the initial state. For (a) EYG was evaluated with area as the cost and for (b), it was the leakage power. As expected, the area overhead is smaller for case (a) and the leakage power is lower in case (b). However, the final area achieved in case (b) is only marginally higher than for case (a), while a significant reduction in the leakage power is achieved. The actual values will depend on how aggressive the timing specification is and the structure of the circuit (fraction of gates that have low criticality).



(a) PDFs of initial states (b13)

(b) PDFs of final states (b13)

(c) PDFs of initial states (b17)

PDFs of final states (b17) (d)

Fig. 11: Initial and final PDFs of the circuit delay upon using the two different initial conditions I1 and I2 for optimization. The timing specification that they are to meet is indicated using the green line. DC PDF corresponds the PDF obtained on performing SSTA on the circuit state provided by Synopsys Design Compiler.

Benchmark	$\delta P(\%)$		Benchmark	$\delta P(\%)$	
	I1	I2(i)		I1	I2(i)
Serial resizing			Serial resizing		
b01	53.52	61.49	b12	-14.55	-11.51
b02	47.73	51.11	b13	-16.33	-12.27
b03	374.64	376.26	b14	44.99	50.14
b04	-13.82	-8.55	b15	66.10	62.22
b05	-5.46	0.96	b20	98.93	107.73
b06	10.14	15.39	Multi-node resize		
b07	-13.47	-7.18	b17	191.65	183.98
b08	-5.17	-1.26	b18	66.21	69.15
b09	-0.58	-0.38	b19	106.29	42.94
b10	-9.87	-3.89	b21	78.33	86.98
b11	-12.49	-6.75	b22	79.33	82.93

Table VIII: Comparison of increase in leakage power with respect to circuit optimized using STA with nominal corner. $\delta P\% = \frac{\text{Final leakage - Nominal leakage}}{\text{Nominal leakage}} \times 100\%$, where nominal leakage is the leakage power of the circuit optimized using STA to meet timing at the nominal corner as provided by Synopsys Design Compiler.

7. CONCLUSION

The main contribution of this paper is a simple and accurate expression for the change in timing yield due to a change in the gate delay distribution. This is based on bounds that we have derived for the moments of the path and circuit delay. As a result, yield gradients that are used in sensitivity based optimization algorithms can be evaluated very efficiently. We have demonstrated its effectiveness by using it for timing yield

Bench mark	Resizing alone			Size/V_T assignment		
	Final yield(%)	Area overhead (%)	δP(%)	Final yield(%)	Area overhead (%)	δP(%)
Serial resizing						
b01	96.64	24.11	74.75	96.21	29.96	42.31
b02	95.46	10.65	47.59	95.40	13.02	42.98
b03	98.35	7.93	27.14	86.80	11.80	-6.16
b04	98.65	3.10	10.20	98.81	7.96	-23.31
b05	99.09	4.49	16.99	98.85	13.39	8.80
b06	95.72	14.33	50.12	94.09	15.23	38.82
b07	99.80	3.01	10.28	99.87	4.22	-35.71
b08	95.97	8.89	34.15	94.66	9.70	-0.88
b09	82.35	6.02	21.19	82.90	10.04	13.95
b10	99.20	8.86	33.78	98.91	13.19	26.74
b11	94.54	4.25	13.52	96.18	14.43	-3.77
b12	99.24	2.36	8.54	99.87	16.17	-5.25
b13	99.73	1.35	5.14	99.87	2.98	-41.35
b14	95.98	2.81	9.04	96.84	8.53	-27.39
b15	98.52	1.13	4.65	95.34	5.14	-44.03
b20	98.26	1.12	4.12	99.75	11.86	-15.90
Multi-node resize: Threshold = $2\times \text{EYG}_{min}$						
b17	99.87	1.74	5.67	99.85	4.56	-45.37
b18	96.67	2.89	9.05	95.34	5.14	-44.03
b19	95.92	2.00	6.49	94.67	5.37	-45.18
b21	99.60	0.66	2.59	99.96	3.57	-36.20
b22	99.89	0.62	2.38	99.90	1.35	-53.43
Average	97.12	5.34		96.38	9.89	

Table IX: Comparison of increase in area and leakage power between resizing and simultaneous V_T assignment and sizing.

optimization with (a) area and (b) nominal leakage power as the cost. The first corresponds to the discrete gate sizing problem and the second to simultaneous size and V_T assignment.

With the use of this analytical yield gradient, it is possible to optimize the timing yield for relatively large circuits. However as the circuit size grows (100,000 gates), the runtime is still very large. To address this, we propose a heuristic for resizing/changing V_T of multiple gates in each iteration of the optimization algorithm. This proves to be very effective, giving an average of 35$\times$ speedup at the cost of a marginal area overhead. Another advantage of our simplified yield gradient expression is that it makes it possible for us to explore a larger search space in each iteration. We have demonstrated this for the case of simultaneous resizing and V_T assignment.

A. EVALUATION OF DERIVATIVES FOR THE ANALYTICAL YG

$\frac{\mathrm{d}\mu_c}{\mathrm{d}\mu_x}$, $\frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_x}$ and $\frac{\mathrm{d}\mu_c}{\mathrm{d}C_{xy}}$ required for evaluation of the analytical YG are evaluated using Clark's formulas. ATs/RTs and PDs are evaluated using Block based SSTA[Visweswariah et al. 2004; Chang and Sapatnekar 2005] and are thus expressed

in the canonical form. Therefore, the PD x of node n in canonical form is given by:

$$x = \mu_x + \sum_{i=1}^{M} x_i p_i + x_{M+1} r \tag{14}$$

where p_i are the principal components, r is the independent random component and x_i are the sensitivities to each of the components. With $\theta = \sqrt{\sigma_x^2 + \sigma_y^2 - 2C_{xy}}$, $\alpha_n = \left(\frac{\mu_x - \mu_y}{\theta}\right)$ and $t = \frac{(\sigma_x^2 - \sigma_y^2)}{\theta^2}$ the derivatives $\frac{\mathrm{d}\mu_c}{\mathrm{d}\mu_x}$, $\frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_x}$, $\frac{\mathrm{d}\mu_c}{\mathrm{d}C_{xy}}$, $\frac{\mathrm{d}\sigma_c}{\mathrm{d}\mu_x}$, $\frac{\mathrm{d}\sigma_c}{\mathrm{d}\sigma_x}$ and $\frac{\mathrm{d}\sigma_c}{\mathrm{d}C_{xy}}$ can be evaluated using Clark's formula[Clark 1961] as:

$$\frac{\mathrm{d}\mu_c}{\mathrm{d}\mu_x} = \Phi(\alpha_n), \quad \frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_x} = \frac{\sigma_x \phi(\alpha_n)}{\theta}, \quad \frac{\mathrm{d}\mu_c}{\mathrm{d}C_{xy}} = -\frac{\phi(\alpha_n)}{\theta} \tag{15}$$

$$\frac{\mathrm{d}\sigma_c}{\mathrm{d}\mu_x} = \frac{\theta}{2\sigma_c}\left[2\alpha_n \Phi(\alpha_n)\Phi(-\alpha_n) - \phi(\alpha_n)\left[2\Phi(\alpha_n) - 1\right] + t\phi(\alpha_n)\right] = f(\alpha_n, t) \times \frac{\theta}{2\sigma_c} \tag{16}$$

$$\frac{\mathrm{d}\sigma_c}{\mathrm{d}C_{xy}} = \frac{1}{2\sigma_c}\left[2\phi^2(\alpha_n) + \alpha_n \phi(\alpha_n)\left[2\Phi(\alpha_n) - 1\right] + t\alpha_n \phi(\alpha_n)\right] = g(\alpha_n, t) \times \frac{1}{2\sigma_c} \tag{17}$$

$$\frac{\mathrm{d}\sigma_c}{\mathrm{d}\sigma_x} = \Phi(\alpha_n)\frac{\sigma_x}{\sigma_c} - \sigma_x \frac{\mathrm{d}\sigma_c}{\mathrm{d}C_{xy}} \tag{18}$$

where ϕ and Φ are the PDF and CDF of the standard normal distribution respectively. $\Phi(\alpha_n)$ represents the criticality of the node n.

B. LOWER BOUND OF σ_C

Before we find the lower bound for σ_c, we prove some properties of the following function

$$h(\alpha) = \frac{(\mu_c - \mu_x)(\mu_c - \mu_y)}{\theta^2} \tag{19}$$

$$= \phi^2(\alpha) + \phi(\alpha)\alpha\left(\Phi(\alpha) - \Phi(-\alpha)\right) - \alpha^2 \Phi(\alpha)\Phi(-\alpha) \tag{20}$$

Here μ_c is the mean of the circuit delay and α is as defined in Appendix A. $h(\alpha)$ has the following properties:

(1) $h(\alpha) > 0$ and $\lim_{\alpha \to -\infty} h(\alpha) = \lim_{\alpha \to +\infty} h(\alpha) = 0$
This property follows from the fact that $\mu_c > \max(\mu_x, \mu_y)$ [Ramprasath and Vasudevan 2012] and tends to μ_x (μ_y) as α tends to ∞ ($-\infty$).

(2) $h(\alpha) \leq \Phi(\alpha)$.
Let $f(\alpha) = \Phi(\alpha) - h(\alpha)$.

$$f(\alpha) = \Phi(\alpha) - \phi^2(\alpha) - \phi(\alpha)\alpha\left(\Phi(\alpha) - \Phi(-\alpha)\right) + \alpha^2 \Phi(\alpha)\Phi(-\alpha) \tag{21}$$

$$\frac{\mathrm{d}f}{\mathrm{d}\alpha} = 2\Phi(-\alpha)\left[\phi(\alpha) + \alpha\Phi(\alpha)\right] \tag{22}$$

The proof for $[\phi(\alpha) + \alpha\Phi(\alpha)] \geq 0$ can be found in [Ramprasath and Vasudevan 2012]. This implies $\frac{\mathrm{d}f}{\mathrm{d}\alpha} \geq 0$. Therefore $f(\alpha)$ is monotonically increasing function of α. Also, as $\alpha \to -\infty$, $f(\alpha) \to 0$ since both $\Phi(\alpha)$ and $h(\alpha)$ tend to 0.
The two facts:
(a) $f(\alpha)$ is a monotonically increasing function of α
(b) $f(\alpha) \to 0$ as $\alpha \to -\infty$

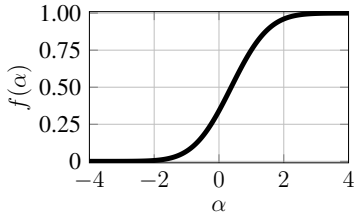


Fig. 12: Plot of $f(\alpha)$ showing that it is an increasing function of α

together imply that $f(\alpha) \geq 0$. This is also seen in Figure 12.

(3) $h(\alpha)$ is an even function. This property directly follows from the definition of $h(\alpha)$.

$$h(-\alpha) = \phi^2(\alpha) - \phi(\alpha)\alpha\left(\Phi(-\alpha) - \Phi(\alpha)\right) - \alpha^2\Phi(\alpha)\Phi(-\alpha) = h(\alpha) \tag{23}$$

(4) $h(\alpha) \leq \Phi(-|\alpha|)$.
As $h(\alpha)$ is an even function, from property (1) we have $h(-\alpha) = h(\alpha) \leq \Phi(-\alpha)$. Since $h(\alpha) \leq \Phi(\alpha)$ and $h(\alpha) \leq \Phi(-\alpha)$, we have $h(\alpha) \leq \Phi(-|\alpha|)$.

(5) $h(\alpha) \leq \phi^2(0)$.

$$\frac{\mathrm{d}h}{\mathrm{d}\alpha} = \phi(\alpha)\left[\Phi(\alpha) - \Phi(-\alpha)\right] - 2\alpha\Phi(\alpha)\Phi(-\alpha) \tag{24}$$

$$\frac{\mathrm{d}^2h}{\mathrm{d}\alpha^2} = \alpha\phi(\alpha)\left[(\Phi(\alpha) - \Phi(-\alpha))\right] + 2\phi^2(\alpha) - 2\Phi(\alpha)\Phi(-\alpha) \tag{25}$$

When $\alpha = 0$, $\frac{\mathrm{d}h}{\mathrm{d}\alpha} = 0$ and $\frac{\mathrm{d}^2h}{\mathrm{d}\alpha^2} = -2\left(\Phi^2(0) - \phi^2(0)\right) < 0$, indicating $h(\alpha)$ has a local maxima at $\alpha = 0$. $h(\alpha)$ is upper bounded by $\Phi(-|\alpha|)$ and $\Phi(-|\alpha|)$ is a decreasing function for $\alpha > 0$. $\Phi(-|\alpha|)$ attains the value of $\phi^2(0)$, when $\alpha \approx 0.9979$. This implies $h(\alpha) < \phi^2(0)$ when $\alpha > 0.9979$. The same argument holds when $\alpha < -0.9979$. The plot of $h(\alpha)$ in the range $\alpha \in [-0.9979, 0.9979]$ is shown in Fig. 13, where it is clearly seen to have only one maxima. Therefore, $\phi^2(0)$ is also a global maxima.

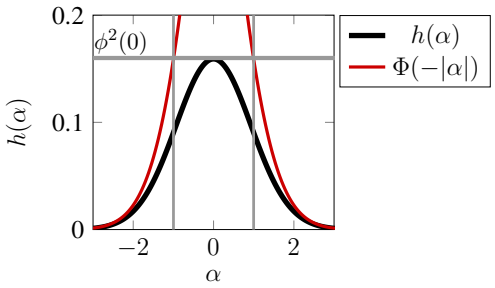


Fig. 13: Plot of $h(\alpha)$ showing its maxima of $\phi^2(0)$

Using these properties, we show that $\sigma_c \geq F \times \min(\sigma_x, \sigma_y)$ if $\rho_{xy} \geq 0$.

Proof:
Using $(\mu_c - \mu_y) = \theta\left[\alpha\Phi(\alpha) + \phi(\alpha)\right]$ and $(\mu_c - \mu_x) = \theta\left[-\alpha\Phi(-\alpha) + \phi(\alpha)\right]$, σ_c can be written as:

$$\sigma_c^2 = \sigma_x^2\Phi(\alpha) + \sigma_y^2\Phi(-\alpha) - (\mu_c - \mu_x)(\mu_c - \mu_y) \tag{26}$$

$$= \sigma_x^2\Phi(\alpha) + \sigma_y^2\Phi(-\alpha) - \theta^2 h(\alpha) \tag{27}$$

Let F be some constant and $\sigma_y \leq \sigma_x$. The largest F so that σ_z is always greater than $F \times \sigma_y = F \times \min(\sigma_x, \sigma_y)$ is required to get a tight bound.

$$F^2\sigma_y^2 - \sigma_c^2 = \sigma_y^2(F^2 - \Phi(-\alpha)) - \sigma_x^2\Phi(\alpha) + \theta^2 h(\alpha) \tag{28}$$

$$= \sigma_y^2(F^2 - \Phi(-\alpha) + h(\alpha)) - \sigma_x^2(\Phi(\alpha) - h(\alpha)) - 2\rho_{xy}\sigma_x\sigma_y h(\alpha) \tag{29}$$

Under the assumption $\rho_{xy} \geq 0$,

$$F^2\sigma_y^2 - \sigma_c^2 \leq \sigma_y^2(F^2 - \Phi(-\alpha) + h(\alpha)) - \sigma_x^2(\Phi(\alpha) - h(\alpha)) - 2\rho_{xy}\sigma_y^2 h(\alpha) \tag{30}$$

$$\leq \sigma_y^2(F^2 - 1 + 2h(\alpha)) + (\Phi(\alpha) - h(\alpha))(\sigma_y^2 - \sigma_x^2) \tag{31}$$

From the properties of $h(\alpha)$, $\Phi(\alpha) \geq h(\alpha)$ and under the assumption $\sigma_y \leq \sigma_x$, the second term in equation (31) is always negative. Using the fact that $h(\alpha) \leq \phi^2(0)$, the value of F for which the right hand side remains negative independent of α is:

$$F^2 - 1 + 2h(\alpha) \leq 0 \implies F \leq \sqrt{1 - 2\phi^2(0)} = \sqrt{1 - \frac{1}{\pi}} \tag{32}$$

The same steps can be repeated for $\sigma_x \leq \sigma_y$ to achieve the same value of F. This implies $\sigma_c \geq F \times \min(\sigma_x, \sigma_y)$. Fig. 14 shows the plot of σ_c vs its lower bound $F \times \min(\sigma_x, \sigma_y)$. MAX operation follows the shifting and scaling properties. So without loss of generality it can be assumed that $\sigma_y \leq \sigma_x$, $\sigma_x = 1.0$ and $\mu_y = 0.0$. The plot is generated using the following range of values: $\rho_{xy} = [0.0 : 0.1 : 1.0]$, $\sigma_y = [0.02 : 0.02 : 1.0]$ and $\alpha = [-5.0 : 0.1 : 5.0]$.

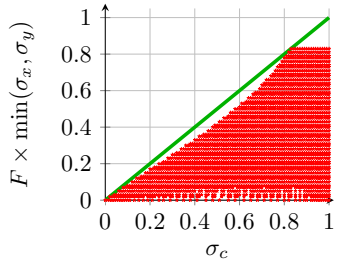


Fig. 14: σ_c vs $(F \times \min(\sigma_x, \sigma_y))$

C. SIMPLIFICATION OF THE EXPRESSION FOR YG

The simplification of EYG expression is done in two steps. The first step involves simplification of the $\frac{\Delta\mu_c}{\Delta\lambda}$ term as shown below:

C.1. Simplification of $\left(\frac{\mathrm{d}\mu_c}{\mathrm{d}\lambda}\right)_n$

To approximate $\left(\frac{\mathrm{d}\mu_c}{\mathrm{d}\lambda}\right)_n$ we use a bound derived in [Ramprasath and Vasudevan 2012]:

— The upper bound for $\frac{\sigma}{\mu}$ of the PD is the maximum value of $\frac{\sigma}{\mu}$ over all edges in the circuit graph.

For the current day technologies, $\frac{\sigma}{\mu}$ of edge delay lies between 10% and 20%. This implies:

$$\frac{\Delta\sigma_x}{\Delta\lambda} \ll \frac{\Delta\mu_x}{\Delta\lambda} \tag{33}$$

Now $C_{xy} = \rho_{xy}\sigma_x\sigma_y$. Assuming that the correlation coefficient does not change much on replacing the gate, we have

$$\frac{\Delta C_{xy}}{\Delta\lambda} \approx \rho_{xy}\sigma_y\frac{\Delta\sigma_x}{\Delta\lambda} \tag{34}$$

This is reasonable since the change in the correlation coefficient between paths due to change in the size of a single gate is marginal. Using equation (34) the term $\left[\frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_x} \times \frac{\Delta\sigma_x}{\Delta\lambda} + \frac{\mathrm{d}\mu_c}{\mathrm{d}C_{xy}} \times \frac{\Delta C_{xy}}{\Delta s_n}\right]$ can be simplified as :

$$\frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_x} \times \frac{\Delta\sigma_x}{\Delta\lambda} + \frac{\mathrm{d}\mu_c}{\mathrm{d}C_{xy}} \times \frac{\Delta C_{xy}}{\Delta s_n} \approx \frac{\phi(\alpha)}{\theta} \times \frac{\Delta\sigma_x}{\Delta\lambda} \times [\sigma_x - \rho_{xy}\sigma_y] \tag{35}$$

Now

$$0 \leq \frac{|\sigma_x - \rho_{xy}\sigma_y|}{\theta} \leq 1 \tag{36}$$

$$\left\{ \because\ 1 - \left(\frac{\sigma_x - \rho_{xy}\sigma_y}{\theta}\right)^2 = \frac{(1-\rho_{xy}^2)\sigma_y^2}{\theta^2} \geq 0 \right\}$$

Using (36) in (35), we get

$$\left|\frac{\mathrm{d}\mu_c}{\mathrm{d}\sigma_x} \times \frac{\Delta\sigma_x}{\Delta\lambda} + \frac{\mathrm{d}\mu_c}{\mathrm{d}C_{xy}} \times \frac{\Delta C_{xy}}{\Delta s_n}\right| \leq \phi(\alpha) \times \frac{\Delta\sigma_x}{\Delta\lambda} \tag{37}$$

Equation (37) indicates we are left with the two terms: $\Phi(\alpha) \times \frac{\Delta\mu_x}{\Delta\lambda}$ and $\phi(\alpha) \times \frac{\Delta\sigma_x}{\Delta\lambda}$. Since typically $\Phi(\alpha) > \phi(\alpha)$ or $\Phi(\alpha)$ has magnitude close to $\phi(\alpha)$, using (33), $\phi(\alpha) \times \frac{\Delta\sigma_x}{\Delta\lambda} \ll \Phi(\alpha) \times \frac{\Delta\mu_x}{\Delta\lambda}$. Therefore

$$\left(\frac{\mathrm{d}\mu_c}{\mathrm{d}\lambda}\right)_n \approx \Phi(\alpha) \times \frac{\Delta\mu_x}{\Delta\lambda} \tag{38}$$

The second step involves simplification of the $\frac{\Delta\sigma_c}{\Delta\lambda}$ term as shown below:

C.2. Simplification of $\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n$

To approximate $\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n$, we use the bounds derived for σ_c.

— $\sigma_c \leq \max(\sigma_x, \sigma_y)$. This result is proved in [Ramprasath and Vasudevan 2012]

— $\sigma_c \geq F \times \min(\sigma_x, \sigma_y)$ where $F = \sqrt{1 - \frac{1}{\pi}}$. Proved in Appendix B.

$\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n$ is given by:

$$\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n = \frac{\mathrm{d}\sigma_c}{\mathrm{d}\mu_x} \times \frac{\Delta\mu_x}{\Delta\lambda} + \frac{\mathrm{d}\sigma_c}{\mathrm{d}\sigma_x} \times \frac{\Delta\sigma_x}{\Delta\lambda} + \frac{\mathrm{d}\sigma_c}{\mathrm{d}C_{xy}} \times \frac{\Delta C_{xy}}{\Delta\lambda} \tag{39}$$

where $\frac{\mathrm{d}\sigma_c}{\mathrm{d}\mu_x}$, $\frac{\mathrm{d}\sigma_c}{\mathrm{d}C_{xy}}$ and $\frac{\mathrm{d}\sigma_c}{\mathrm{d}\sigma_x}$ are given by equations (16), (17) and (18). Using (34), $\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n$ can be written as:

$$\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n \approx f(\alpha,t)\frac{\theta}{2\sigma_c}\frac{\Delta\mu_x}{\Delta\lambda} + \Phi(\alpha)\frac{\sigma_x}{\sigma_c}\frac{\Delta\sigma_x}{\Delta\lambda} - g(\alpha,t)\left[\frac{(\sigma_x-\rho_{xy}\sigma_y)}{\theta}\right]\frac{\theta}{2\sigma_c}\frac{\Delta\sigma_x}{\Delta\lambda} \tag{40}$$

where $f(\alpha,t)$ and $g(\alpha,t)$ are defined in equations (16) and (17). Fig. 5 shows the plots of $f(\alpha,t)$ and $g(\alpha,t)$ for various values of t. It is clear from the figure that $f(\alpha,t)$ and $g(\alpha,t)$ have similar magnitudes. Using the argument $\frac{\Delta\mu_x}{\Delta\lambda} \gg \frac{\Delta\sigma_x}{\Delta\lambda}$ and $\frac{|\sigma_x-\rho_{xy}\sigma_y|}{\theta} \leq 1$, we can neglect the third term in equation (40) in comparison to the first. So $\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n$ can be written as:

$$\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n \approx f(\alpha,t)\frac{\theta}{2\sigma_c}\frac{\Delta\mu_x}{\Delta\lambda} + \Phi(\alpha)\frac{\sigma_x}{\sigma_c}\frac{\Delta\sigma_x}{\Delta\lambda} \tag{41}$$

Two different cases are considered depending on the criticality of the node:

(1) Low and medium criticality nodes:
In this case, $\Phi(\alpha)$ and $f(\alpha,t)$ have similar magnitudes as shown in Fig. 5. Recall that $\frac{\Delta\mu_x}{\Delta\lambda} \gg \frac{\Delta\sigma_x}{\Delta\lambda}$. Therefore if θ and σ_x have comparable values or $\theta > \sigma_x$, we can neglect the second term in comparison to the first. It is easily seen to be true when $\rho_{xy} < 0$ since $\theta \geq \sigma_x$. If $\rho_{xy} > 0$, it is also seen to be true for the extreme cases when (a) $\sigma_x \gg \sigma_y$, then $\theta \approx \sigma_x$ and (b) $\sigma_y \gg \sigma_x$, then $\theta > \sigma_x$. In other cases, the lower and upper bounds of σ_c gives us an upper bound for $\frac{\sigma_x}{\sigma_c}$ and a lower bound for $\frac{\theta}{\sigma_c}$ as follows. The upper bound of $\frac{\sigma_x}{\sigma_c}$ can be written as:

$$\frac{\sigma_x}{\sigma_c} \leq \begin{cases} h \times \dfrac{1}{F}, & \text{if } h > 1 \\ \dfrac{1}{F} & \text{if } h \leq 1 \end{cases} \tag{42}$$

where $h = \frac{\sigma_x}{\sigma_y}$. The lower bound of $\frac{\theta}{\sigma_c}$ can be written as

$$\frac{\theta}{\sigma_c} \geq \frac{\sqrt{\sigma_x^2+\sigma_y^2-2\rho_{xy}\sigma_x\sigma_y}}{\max(\sigma_x,\sigma_y)} = \left(\frac{\theta}{\sigma_c}\right)_{LB} \tag{43}$$

$$\left(\frac{\theta}{\sigma_c}\right)_{LB} = \begin{cases} \sqrt{1+\dfrac{1}{h^2}-2\dfrac{\rho_{xy}}{h}}, & \text{if } h \geq 1 \\ \sqrt{1+h^2-2\rho_{xy}h}, & \text{if } h < 1 \end{cases} \tag{44}$$

If σ_x and σ_y have comparable values, $h \approx 1$, the upper bound for $\frac{\sigma_x}{\sigma_c}$ gives $\frac{\sigma_x}{\sigma_c} \leq \frac{1}{F} \approx 1.21$. Fig. 15 shows the plot of lower bound of $\frac{\theta}{\sigma_c}$ for various values of ρ_{xy} and h. The figure indicates that $\frac{\theta}{\sigma_c}$ to a large extent is of the order of 1.0. Since $\frac{\Delta\mu_x}{\Delta\lambda} \gg \frac{\Delta\sigma_x}{\Delta\lambda}$, the second term in equation (41) can be neglected in comparison with the first. Thus $\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n$ can be approximated as:

$$\left(\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}\right)_n = f(\alpha,t)\frac{\theta}{2\sigma_c}\frac{\Delta\mu_x}{\Delta\lambda} \tag{45}$$

(2) For the high criticality nodes and nodes for which both ρ_{xy} and h are $\sim$1, $\left(\Phi(\alpha)\frac{\sigma_x}{\sigma_c}\frac{\Delta\sigma_x}{\Delta\lambda}\right)$ dominates the $\left(f(\alpha,t)\frac{\theta}{2\sigma_c}\frac{\Delta\mu_x}{\Delta\lambda}\right)$. So ignoring the $\delta\sigma_x$ terms in this scenario leads to an erroneous $\frac{\mathrm{d}\sigma_c}{\mathrm{d}\lambda}$. Although there is an error, it will not make

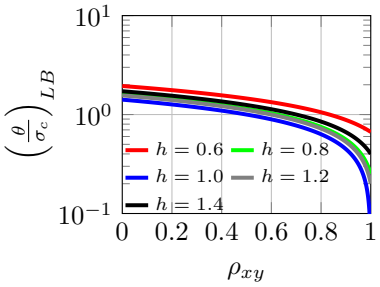


Fig. 15: Lower bound of $\frac{\theta}{\sigma_c}$

much of a difference in this case since the YG will be dominated by the first term, namely, $\frac{\Delta\mu_c}{\Delta\lambda}$.

References

D.K. Beece, Jinjun Xiong, C. Visweswariah, V. Zolotov, and Yifang Liu. 2010. Transistor sizing of custom high-performance digital circuits with parametric yield considerations. In *Proceedings of 47th Design Automation Conference*. IEEE, 781 –786.

Hongliang Chang and S.S. Sapatnekar. 2005. Statistical timing analysis under spatial correlations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 24, 9 (Sept. 2005), 1467 – 1482. DOI:http://dx.doi.org/10.1109/TCAD.2005.850834

Seung Hoon Choi, B.C. Paul, and K. Roy. 2004. Novel sizing algorithm for yield improvement under process variation in nanometer technology. In *Proceedings of 41st Design Automation Conference*. IEEE, 454 –459.

Charles E. Clark. 1961. The Greatest of a Finite Set of Random Variables. *Operations Research* 9, 2 (1961), 145–162. DOI:http://dx.doi.org/10.1287/opre.9.2.145

A. R. Conn, N. I. M. Gould, and P. L. Toint. 1992. *Lancelot: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)* (1st ed.). Springer Verlag.

O. Coudert. 1997. Gate sizing for constrained delay/power/area optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 5, 4 (1997), 465–472. DOI:http://dx.doi.org/10.1109/92.645073

A. Davoodi and A. Srivastava. 2008. Variability Driven Gate Sizing for Binning Yield Optimization. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 16, 6 (june 2008), 683 –692. DOI:http://dx.doi.org/10.1109/TVLSI.2008.2000252

M.R. Guthaus, N. Venkateswaran, C. Visweswariah, and V. Zolotov. 2005. Gate sizing using incremental parameterized statistical timing analysis. In *IEEE/ACM International Conference on Computer-Aided Design, 2005.* IEEE, 1029 – 1036. DOI:http://dx.doi.org/10.1109/ICCAD.2005.1560213

Eun Ju Hwang, Wook Kim, and Young Hwan Kim. 2013. Timing Yield Slack for Timing Yield-Constrained Optimization and Its Application to Statistical Leakage Minimization. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 21, 10 (Oct 2013), 1783–1796. DOI:http://dx.doi.org/10.1109/TVLSI.2012.2220792

E.T.A.F. Jacobs and M.R.C.M. Berkelaar. 2000. Gate sizing using a statistical delay model. In *Proceedings of Design, Automation and Test in Europe*. IEEE, 283 –290. DOI:http://dx.doi.org/10.1109/DATE.2000.840285

Li Li, Peng Kang, Yinghai Lu, and Hai Zhou. 2012. An efficient algorithm for library-based cell-type selection in high-performance low-power designs. In *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*. 226–232.

M. Mani, A. Devgan, M. Orshansky, and Yaping Zhan. 2007. A Statistical Algorithm for Power- and Timing-Limited Parametric Yield Optimization of Large Integrated Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26, 10 (Oct. 2007), 1790 –1802. DOI:http://dx.doi.org/10.1109/TCAD.2007.895797

M. Mani and M. Orshansky. 2004. A new statistical optimization algorithm for gate sizing. In *Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings. IEEE International Conference on*. 272–277. DOI:http://dx.doi.org/10.1109/ICCD.2004.1347933

S. Ramprasath and V. Vasudevan. 2012. On the computation of criticality in statistical timing analysis. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 172 –179.

S. Ramprasath and V. Vasudevan. 2014. Statistical Criticality Computation Using the Circuit Delay. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 33, 5 (May 2014), 717–727. DOI:http://dx.doi.org/10.1109/TCAD.2013.2296436

J. Singh, V. Nookala, Zhi-Quan Luo, and S. Sapatnekar. 2005. Robust gate sizing by geometric programming. In *Proceedings of 42nd Design Automation Conference*. IEEE, 315 – 320. DOI:http://dx.doi.org/10.1109/DAC.2005.193824

D. Sinha, N.V. Shenoy, and Hai Zhou. 2006. Statistical Timing Yield Optimization by Gate Sizing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 10 (Oct. 2006), 1140 –1146. DOI:http://dx.doi.org/10.1109/TVLSI.2006.884166

Debjit Sinha, Chandu Visweswariah, Natesan Venkateswaran, Jinjun Xiong, and Vladimir Zolotov. 2012. Reversible statistical max/min operation: Concept and applications to timing. In *Proceedings of 49th Design Automation Conference*. IEEE, 1067 –1073.

A. Srivastava, K. Chopra, S. Shah, D. Sylvester, and D. Blaauw. 2008. A Novel Approach to Perform Gate-Level Yield Analysis and Optimization Considering Correlated Variations in Power and Performance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 27, 2 (Feb. 2008), 272 –285. DOI:http://dx.doi.org/10.1109/TCAD.2007.907227

A. Srivastava, D. Sylvester, and D. Blaauw. 2004. Statistical optimization of leakage power considering process variations using dual-Vth and sizing. In *Design Automation Conference, 2004. Proceedings. 41st*. 773 –778.

A. Tang and N.K. Jha. 2015. GenFin: Genetic Algorithm-Based Multiobjective Statistical Logic Circuit Optimization Using Incremental Statistical Analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* PP, 99 (2015), 1–1. DOI:http://dx.doi.org/10.1109/TVLSI.2015.2442260

C. Visweswariah, K. Ravindran, K. Kalafala, S.G. Walker, S. Narayan, D.K. Beece, J. Piaget, N. Venkateswaran, and J.G. Hemmett. 2006. First-Order Incremental Block-Based Statistical Timing Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 10 (oct. 2006), 2170 –2180. DOI:http://dx.doi.org/10.1109/TCAD.2005.862751

C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. 2004. First-order Incremental Block-based Statistical Timing Analysis. In *Proceedings of the 41st Annual Design Automation Conference (DAC '04)*. ACM, New York, NY, USA, 331–336. DOI:http://dx.doi.org/10.1145/996566.996663

Jinjun Xiong, V. Zolotov, and C. Visweswariah. 2008. Incremental Criticality and Yield Gradients. In *Proceedings of Design, Automation and Test in Europe*. IEEE, 1130 –1135. DOI:http://dx.doi.org/10.1109/DATE.2008.4484830

Yaping Zhan, A.J. Strojwas, Xin Li, L.T. Pileggi, D. Newmark, and M. Sharma. 2005. Correlation-aware statistical timing analysis with non-Gaussian delay distributions. In *Proceedings of the 42nd Design Automation Conference*. 77–82. DOI:http://dx.doi.org/10.1109/DAC.2005.193777