Parametric Yield Maximization using Gate Sizing based on Efficient Statistical Power and Delay Gradient Computation

Kaviraj Chopra, Saumil Shah, Ashish Srivastava, David Blaauw, Dennis Sylvester
University of Michigan, EECS Department, Ann Arbor, MI 48109
{kaviraj, saumil, ansrivas, Blaauw, dennis}@eecs.umich.edu

Abstract

With the increased significance of leakage power and performance variability, the yield of a design is becoming constrained both by power and performance limits, thereby significantly complicating circuit optimization. In this paper, we propose a new optimization method for yield optimization under simultaneous leakage power and performance limits. The optimization approach uses a novel leakage power and performance analysis that is statistical in nature and considers the correlation between leakage power and performance to enable accurate computation of circuit yield under power and delay limits. We then propose a new heuristic approach to incrementally compute the gradient of yield with respect to gate sizes in the circuit with high efficiency and accuracy. We then show how this gradient information can be effectively used by a non-linear optimizer to perform yield optimization. We consider both inter-die and intra-die variations with correlated and random components. The proposed approach is implemented and tested and we demonstrate up to 40% yield improvement compared to a deterministically optimized circuit.

1. Introduction and Overview of Approach

Continued process scaling has resulted in a large increase in process variability that leads to large fluctuations in process parameters from their nominal values. These variations have grown due in part to aggressive lithographic techniques that are used to pattern dimensions smaller than the wavelength of light. In addition, smaller device dimensions and a smaller number of doping atoms increase the influence of phenomena such as line edge roughness and random dopant effects. These process variations translate to wide ranges in performance metrics of current designs. In particular, leakage current which is extremely sensitive to a number of key process parameters has shown huge fluctuations with [1] showing a 20X variation in leakage power for a 30% variation in performance across 1000 samples of a design manufactured in a 180 nm technology. In addition, as the contribution of leakage power has grown the fluctuation in power dissipation is now dominated by leakage power. This results in a negative correlation between power dissipation and delay of a design. Thus high performance samples of a design are also expected to have higher power dissipation and vice-versa. This leads to a two-sided constraint on the feasible region of parametric yield defined by delay and power limits [2], and causes significant yield loss under variations in process parameters.

A number of analysis techniques to consider the impact of variability on timing [3-7] and power [8-9] have been developed. However, these approaches do not estimate the true parametric yield of a design considering both power and performance correlation. Reference [10] was the first to consider the impact of variability on circuit optimization. The authors described the formation of a timing wall due to deterministic power optimization which increases the susceptibility of the design to process variations, and proposed a heuristic approach to prevent the build-up of a large number of paths near the critical delay of the circuit. This was achieved by adding a penalty function which had a negative impact on the objective function value whenever a path had a delay which was near critical. However, the approach was deterministic in nature and did not use any statistical information during optimization.

Recently, several statistical timing [11-13] and power [14-16] optimization approaches have also been proposed. However, all these approaches neglect the correlation between power and performance. Therefore, timing yield improvements inevitably result in degradation in power yield and vice versa. Moreover, most of these approaches suffer from large computational complexity and runtimes. Thus, there is a critical need to develop approaches that perform true and efficient parametric yield optimization, where yield is defined using both power and timing constraints. Recently, [17] proposed an approach to perform gate-level yield analysis in a computationally efficient manner while considering the correlation between power and performance. The approach was based on a principal-component based process variation modeling technique to perform timing and power analysis using the same set of underlying random variables (RVs), allowing the correlation in power and performance to be captured. Additionally the approach considers all components of process variations: inter-die and intra-die (spatially correlated and random) variability and can therefore serve as a framework to enable true parametric yield optimization.

In this work, we propose a novel approach to perform yield optimization using gate sizing. The yield optimization is formulated as an unconstrained optimization problem, where the objective is to maximize the parametric yield of a design. The optimization is performed using a gradient-based non-linear optimizer. A brute-force gradient computation approach based on iterative yield analysis, however, leads to large computational overheads. Therefore, we propose an efficient heuristic technique to perform the computation of the yield gradient. This is achieved by perturbing the size of a gate in the circuit and heuristically recalculating the delay and power probability distribution functions (pdfs) of the perturbed circuit. The timing pdf of the perturbed circuit is calculated based on a novel cutset approach that analyzes only a subset of the nodes in the circuit to estimate the complete delay pdf.

The power pdf of the perturbed circuit is calculated with an incremental power analysis. This involves subtracting the power dissipation of the perturbed gate from the pdf of total power for the complete circuit and then adding the power dissipation of the perturbed gate while accounting for the gate size change. These perturbed pdfs are then used to compute the yield of the perturbed circuit by integrating the perturbed bivariate Gaussian distribution over the region defined by the leakage power and timing constraint. This gradient computation technique is then integrated with LANCELOT [18], a large-scale non-linear optimizer, to improve the parametric yield of the design, and is found to provide an 8X improvement in runtime with an average error of 0.1%.

The remainder of the paper is organized as follows. Section 2 briefly reviews the principal component based approach to perform yield analysis. Section 3 presents the incremental timing and power analysis techniques, which are then used to compute the gradient of yield. In Section 4 we provide details regarding the implementation of our yield optimization

0-7803-9254-X/05/$20.00 ©2005 IEEE.

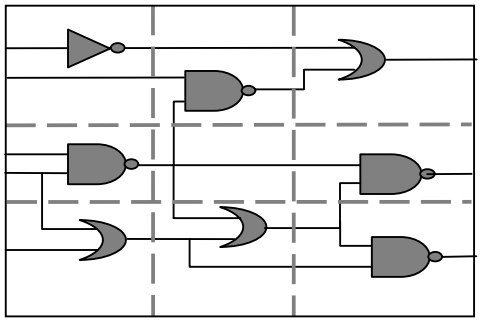

Figure 1. Example partitions of a circuit using a grid to model the correlated component of variations

approach and present results including a comparison of our approach to deterministic optimization. We provide conclusions in Section 5.

2. Yield analysis

In this section we briefly discuss our modeling assumptions and yield analysis approach. We also define the yield optimization problem and describe a brute-force approach to perform yield optimization. The computational complexity of this brute-force approach motivates the need for more efficient gradient computation techniques.

Our yield analysis framework is based on the approach in [17], and we express the delay and a leakage of a gate as

$$Delay = d_{nom} + \sum_{i=1}^{p} \alpha_p (\Delta P_p)$$
$$Leakage = \exp\left(V_{nom} + \sum_{i=1}^{p} \beta_p (\Delta P_p) \right) \quad (1)$$

where d_{nom} and $\exp(V_{nom})$ are the nominal gate delay and leakage, respectively, and α_p and β_p captures the dependence of gate delay and the log of gate leakage on the p process parameters of interest. The RVs ΔP_p in the above equation refers to the variations in these process parameters that are assumed to be Gaussian. The variation in process parameter is then partitioned into a correlated and random component. The correlated component is handled by partitioning the design as shown in Figure 1. For each parameter, each square in the grid is assigned to a Gaussian RV which captures the correlated variation in that process parameter, which is defined using a correlation matrix. This gives a total of N_g RVs for each parameter, which are assumed to have a joint multinormal distribution. Using principal components analysis [19] the correlated component is expressed as a linear combination of N_g independent Gaussian RVs (z_i), and the random variation in all the process parameters is lumped into a single RV - η_d for delay and η_l for leakage, and the coefficient of the random component is calculated by matching the variance of the random contribution. This approach gives us canonical expressions for gate delay and leakage power which are expressed as:

$$Delay = d_{nom} + \sum_{i=1}^{p} \left(\alpha_p \sum_{i=1}^{n} \gamma_i z_i \right) + \eta_d R$$
$$Leakage = \exp\left(V_{nom} + \sum_{i=1}^{p} \left(\beta_p \sum_{i=1}^{n} \gamma_i z_i \right) + \eta_l R \right) \quad (2)$$

Timing analysis is then performed in the spirit of [3][17], and the delay is propagated through the circuit, while maintaining the node delays in the same canonical form with different coefficients. The *sum* operation is performed by simply adding the coefficients for each of the RVs, other than the random component whose coefficient is obtained as the square root of the sum of the squared coefficients of the random components

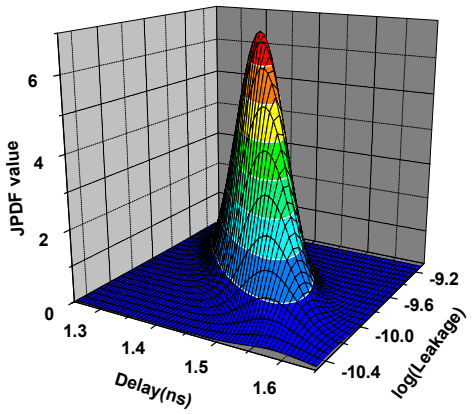

Figure 2. Joint probability distribution function for the bivariate Gaussian distribution for c3540.

of the summed pdfs. The *max* operation is performed by matching the mean, variance and the correlation of the max of two RV (which are obtained using [20]) and the canonical expression of the max.

Leakage power analysis is based on summing lognormal RVs using Wilkinson's method [21] as proposed in [8]. The leakage of each gate is iteratively added to the sum which is maintained in canonical form. In each addition the coefficients of the canonical expression for the sum are calculated by matching the mean, variance and correlation with the principal components of the sum (obtained using [8]) and the canonical expression for the sum.

At the end of timing and power analysis (that provides the delay and leakage power canonical form) the correlation between delay and leakage power is estimated using:

$$Cov(Delay, Leakage) = \sum_{i=1}^{n} \alpha_i \beta_i \quad (3)$$

These five parameters (mean and variance of delay and leakage power and their correlation) are used to define a bivariate Gaussian distribution for delay and log of leakage power as shown in Figure 2. The parametric yield which is defined as:

$$Y = P(D \le D_0, P \le P_0) \quad (4)$$

where D is delay of the circuit constrained to be less than D_0 and P is the power of the design constrained to be less than P_0. In this work, we assume that variations in power are dominated by variations in leakage power and the dynamic power dissipation is assumed to be a fixed number and subtracted out of the total power budget of the design to define the leakage power constraint. Based on this assumption, we can rewrite (4) as

$$Y = P(D \le D_0, \log P_L \le \log(P_0 - P_D)) \quad (5)$$

where P_L and P_D are the leakage and dynamic power of the design. The above yield expression is now equivalent to the integral of a bivariate Gaussian RV over a rectangular region, and can be evaluated using expression developed in [22]. Both the timing and power computation steps require $O(nN_g)$ steps, where n is the number of gates in the design and N_g is the number of regions into which the design is partitioned to capture the correlation structure of correlated process variations. However, the final yield computation step (5) itself runs in constant time since the yield computation is always performed using the set of five parameters, independent of the size of the problem.

Based on this yield analysis engine, a brute-force approach to perform yield optimization using gate sizing can be developed. This involves computing the gradient of yield to the size of each gate, which can be estimated by resizing each gate and

performing yield analysis and setting the gate back to its original size. After computing the gradient, we use a large scale non-linear optimizer to improve the yield of the circuit. We now consider the computational complexity of one iteration of this approach. Each gradient computation requires $n+1$ yield analysis runs and thus has an overall complexity of $O(n^2N_g)$. Note that the since the size of the partitions is fixed, the number of partitions N_g can also be expected to increase with the size of the design. Thus, the overall computational requirements soon become untenable for large designs. Also, note that the brute-force approach spends most of the time recalculating the same information for most of the circuit and motivates the need for an efficient gradient computation approach.

3. Gradient Computation

In this section we will discuss our new gradient computation approach that calculates the updated timing and delay pdfs based on a change in gate size. Both the timing and power perturbation analysis techniques update the coefficients of the delay and leakage pdf expression based on the change in gate size. These updated delay and leakage power pdfs are then used to compute the yield of the perturbed design. The change in yield is used to estimate the gradient of yield to the size of each gate in the design.

3.1 Timing Perturbation Computation

We will explain our timing perturbation computation approach based on cutsets using the following graph representation for our circuits.

Definition 1: A *timing graph* is a directed acyclic graph having exactly one source and one sink: $G=\{N,E,ns,nf\}$, where $N=\{n_1,n_2,..,n_k\}$ is a set of nodes, $E=\{e1,e2,...,el\}$ is a set of edges, $ns \in N$ is the source node and $nf \in N$ is the sink node and each edge is an ordered pair of nodes $e=(n_i,n_j)$ and each node is associated with a delay for each fanin edge, which depends on the characteristics of the fanout nodes.

The nodes in the timing graph correspond to gates and the edges correspond to nets in a circuit. A *probabilistic timing graph* is defined as a timing graph where each node is associated with a RV for the delay for each fanin edge. Figure 3 shows an example timing graph with ten nodes, eight of which represent actual gates and nodes 1 and 10 represent the source and sink nodes, respectively. The latest arrival time (AT) and required arrival time (RAT) probability distribution functions (pdf) for each node in the timing graph are now defined as:

Definition 2: The latest *arrival time* (AT) at an edge e in the probabilistic timing graph is a RV whose CDF $A_e(t)$ gives the probability that a deterministic sample of this timing graph has an arrival time less than t.

Definition 3: The earliest *required arrival time* (RAT) at an edge e in the probabilistic timing graph is a RV whose CDF $R_e(t)$ gives the probability that the deterministic sample meets the timing constraint T_{crit} if the deterministic arrival time at the node is less than t.

Note that the sum of the AT and RAT at a node represents the partial pdf of delay since it does not take into account the influence of the edges that are not present in either the fanin or the fanout cone of the node on the pdf of circuit delay. To express the dependence of circuit delay on the delay of one of the nodes let us define the following terms.

Definition 4: A *linear topological ordering* (LTO) of the nodes in a timing graph is a total order based on the relationship that the order of any node x that lies in the fanout cone of a node n is strictly larger than the order of node n, and that no two nodes in a timing graph have the same order.

An LTO of a timing graph can be easily determined by performing a breadth-first traversal of the timing graph. Though a given timing graph can have many LTOs, finding the optimal LTO is not the focus of this paper. Figure 3 illustrates a timing graph with nodes labeled according to a LTO of the timing graph. Note that swapping nodes 8 and 9 will still maintain a valid LTO of the nodes.

Definition 5: A *cutset* of a timing graph with a given LTO of a node n is defined to be the set of edges (n_i,n_j) of the timing graph which satisfy $LTO(n_i) \leq LTO(n)$ and $LTO(n_j) > LTO(n)$.

Definition 6: A node x of the timing graph belongs to the *cutset-source* of node n if there exists an edge $(x,*)$ which belongs to the cutset of node n.

Definition 7: The *fanin-set* of a node n of a timing graph is the set of immediate predecessor nodes of node n.

Definition 8: The *arrival time set* or ATSet of a node n is the union of the fanin-set of node n and the nodes in the fanout cone of the fanin-set of node n that have order less than or equal to the order of node n.

Definition 9: The *convolution-set* or ConvSet of a node n is the intersection of the ATSet and cutset-source of node n.

Any cutset of the timing graph divides the timing graph into two disconnected components and the statistical maximum of the sum of the AT and RAT of all edges in the cutset gives the complete pdf of circuit delay. Now, if we perturb the delay characteristics of a node n (by gate sizing) we also change the capacitive loading of the fanin gates, affecting their delay characteristics as well. To compute the new circuit delay we note that the RAT of the edges in the cutset does not change, since all the gates in their fanout cone includes gates that have order strictly greater than the order of node n, and have unchanged delay characteristics[1].

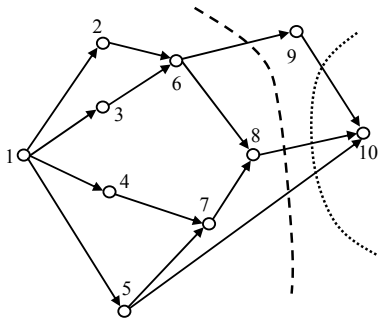

Figure 3. A timing graph showing a linear topological ordering for the nodes and cutsets for nodes 8 and 9

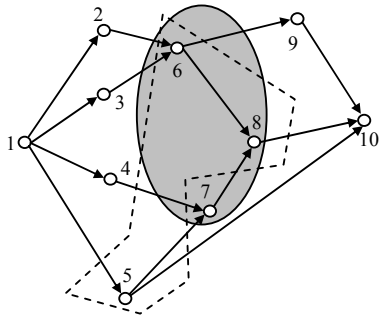

Figure 4. A timing graph showing the ATSet (nodes within the shaded ellipse) and cutset-source set (nodes within the dashed shape) for node 8

[1] This neglects the impact of the change in the slope of the circuit. However, the approach for timing analysis using backward propagation [24] can be used to consider their impact within the same framework.

However, the AT for all edges that are in the fanout cone of the fanin-set of node *n* changes. However we are only interested in AT changes for edges that are driven by nodes that have order less than the order of gate *n*, since we need to compute the AT for the edges in the cutset only. This is exactly the set of nodes defined by the ATSet of node *n*. If the AT of an edge in the cutset changes we need to recompute the convolution of the AT and RAT at that edge. These edges are driven by the nodes in the intersection of the ATSet and the cutset which is defined as the ConvSet of node *n*.

Let us revisit the example timing graph in Figure 3 and consider node 8. The cutset for this node is the set of edges (6,9), (8,10) and (5,10) as shown by the dashed line. The ATSet for the node can be identified as the set of nodes 6, 7 and 8 as shown in Figure 4. The intersection of the cutset-source and ATSet defines the ConvSet and is the set of nodes 6 and 8. Note that the ConvSet identifies that the AT and RAT has not changed on the edge (5,10) and we do not need to recompute the convolution of the AT and RAT for this edge. However, if we consider node 9 the cutset is defined by the edges from nodes 5, 8 and 9 to node 10, as shown as the dotted line in Figure 3. The pseudo-code to calculate the delay pdf of the perturbed circuit is shown below, where we refer to the edge by the name of the driving node. The pseudo-code involves the computation of the AT for all nodes in the ATSet, convolution of the AT and RAT for all nodes in the ConvSet and the statistical maximum of the convolution for all edges in the cutset.

Note that all the computations in *CutSetSsta* are performed using the same canonical expression for the delay pdf. Thus, the final delay pdf of the perturbed circuit is also expressed in the same form. Although, the approach as described seems exact, it is heuristic. This results from the fact that the computation of the *max* function of delay pdfs is not exact and forward and backward traversals of the graph result in timing delays that are not exactly same. However, this error is very small as will be shown later in the results section.

3.2 Power Perturbation Computation

The statistical power computation is performed by summing the power dissipation of each gate in a circuit to compute the complete pdf of leakage power. To perform power analysis of a circuit with perturbations in the size of a gate, we first perform statistical power analysis of the unperturbed circuit as described in Section 2. Now the leakage power after the size of gate *i* has been perturbed is expressed as

$$\begin{aligned} P_{circ}^{pert} &= P_{circ}^{unpert} - P_{gate,i}^{unpert} + P_{gate,i}^{pert} \\ &= P_{circ\backslash i}^{unpert} + P_{gate,i}^{pert} \end{aligned} \quad (6)$$

where P^{pert} and P^{unpert} refer to the perturbed and unperturbed power, respectively and the subscript indicates whether the power refers to the circuit or to the gate. Since the leakage power is expressed a lognormal (exponential of a Gaussian) RV, we can approximate their sum using another lognormal. In general, if we sum P_{leak}^{b} and P_{leak}^{c} to obtain P_{leak}^{a}, which is

```
CUTSETSSTA (n)
  for each node (x ∈ ATSET (n))
    Compute AT(x);
  for each node (x ∈ CONVSET (n))
    CT(x) ← convolution (AT(x), RT(x))
  for each edge(x ∈ CUTSET (n))
    T_n ← maximum (T_n, CT(x))
  return T_n
```

Figure 5. Pseudo-code for the computation of perturbations in timing pdf of a circuit

mathematically expressed as,

$$\begin{aligned} P_{leak}^{a} &= \exp\left(a_0 + \sum_{i=1}^{n} a_i z_i + a_{n+1} \right) \\ &= \exp\left(b_0 + \sum_{i=1}^{n} b_i z_i + b_{n+1} \right) + \exp\left(c_0 + \sum_{i=1}^{n} c_i z_i + c_{n+1} \right) = P_{leak}^{b} + P_{leak}^{c} \end{aligned} \quad (7)$$

the coefficients in the expression for P_{leak}^{a} can be obtained by matching the mean, variance and the correlation coefficient with the exponential of the principal components (z_i's). This gives us a set of $n+2$ equations in $n+2$ variables which can be analytically solved to obtain the following expression for the coefficients associated with the principal components [17]

$$a_i = \log\left(\frac{E\left(P_{leak}^{a} e^{z_i}\right)}{E\left(P_{leak}^{a}\right) E\left(e^{z_i}\right)} \right) = \log\left(\frac{E\left(P_{leak}^{b} e^{z_i}\right) + E\left(P_{leak}^{c} e^{z_i}\right)}{\left(E\left(P_{leak}^{b}\right) + E\left(P_{leak}^{c}\right)\right) E\left(e^{z_i}\right)} \right) \quad (8)$$

Using the expressions developed in [13], the remaining two coefficients in the expression for P_{leak}^{a} can be expressed as

$$a_0 = \frac{1}{2} \log\left(\frac{\left(E\left(P_{leak}^{b}\right) + E\left(P_{leak}^{b}\right)\right)^4}{\left(E\left(P_{leak}^{b}\right) + E\left(P_{leak}^{b}\right)\right)^2 + Var\left(P_b\right) + Var\left(P_c\right) + 2Cov\left(P_b P_c\right)} \right) \quad (9)$$

$$a_{n+1} = \left[\log\left(1 + \frac{Var\left(P_b\right) + Var\left(P_c\right) + 2Cov\left(P_b P_c\right)}{\left(E\left(P_{leak}^{b}\right) + E\left(P_{leak}^{b}\right)\right)^2} \right) - \sum_{i=1}^{n} a_i^2 \right]^{0.5} \quad (10)$$

Note that to compute the expression in (6) we need to use the above expressions twice to calculate the final perturbed leakage. However, when one of the lognormals is subtracted the signs associated with its expected value and covariance terms in the above expressions are reversed.

3.3 Yield Gradient

To this point we have developed efficient approaches to perform statistical timing and power perturbation computation. Now, we will use these techniques to perform the computation of the gradient of yield in an efficient manner.

The computation of yield gradient involves the computation of the perturbation in yield for small changes in the size of gates in the design. The pseudo-code for the computation of yield is included in Figure 6. After each resizing move, the non-linear

```
FASTYIELDGRADIENT (CIRCUIT, SIZE )
  for each gate (g ∈ CIRCUIT)
    update load cap and size of (g) using SIZE;
  for each gate (g ∈ CIRCUIT)
    compute new gate delay & power of (g)
  T ← FORWARDSSTA (CIRCUIT)
  P ← STATPOWERANALYSIS (CIRCUIT)
  Y ← YIELD (P, T)
  do REVERSESSTA (CIRCUIT)
  for each gate(g ∈ CIRCUIT)
    save current state of CIRCUIT
    s+ ← SIZE(g)+ ΔSIZE(g)
    compute new gate delay & power of g
    for each gate(i ∈ FANIN(g))
      update load cap and delay of i
    T+ ← CUTSETSSTA (CIRCUIT)
    P+ ← INCREMSPA (CIRCUIT, P)
    Y+ ← YIELD (P+, T+)
    ∇Y(g) ← (Y+ -Y)/ ΔSIZE(g)
    restore the original state of the CIRCUIT
  return ∇Y
```

Figure 6. Pseudo-code for the computation of the yield gradient

optimizer calls the yield computation routine "FastYieldGradient". The first step is to initialize the circuit so that all nodes are assigned the correct load capacitance based on the sizes of the gates in its immediate fanout, and the correct leakage power based on its own size. Based on the load capacitance of the node each input of a node is assigned to a delay pdf, which represents the delay of the timing arc from that particular input to the output of the gate.

After the initialization step, the next step involves the propagation of the AT from the source node to the sink node in the timing graph. This is represented as "ForwardSSTA" in the pseudo-code. The next step is to perform statistical power analysis and generate the leakage current pdf using "StatPowerAnalysis". The "Yield" function is then used to compute the yield based on the timing and leakage power pdfs given a leakage power constraint P and a delay constraint D, as outlined in Section 2.

To perform the computation of yield gradient, we first propagate the RAT from the sink node to the source node using "ReverseSSTA". Then we go through each node in the circuit iteratively and perturb the size of each gate by a small amount. The load capacitance of the nodes in the fanin-set of the node and the delay pdf assigned to each timing arc of this node and the nodes in the fanin-set are updated. Then using the statistical timing and power perturbation computation techniques discussed in Sections 3.1-3.2 we compute the delay and leakage power pdfs of this perturbed circuit. The yield corresponding to the perturbed circuit is then calculated and the change in yield is used to define the particular component of the yield gradient.

Let us consider the computational complexity of our proposed approach and compare it to the brute-force approach, where each iteration had a complexity of $O(n^2N_g)$. Each iteration, in our proposed approach, involves a single run of the complete yield analysis approach, as discussed above, which has a complexity of $O(nN_g)$. The timing and power perturbation computation is repeated $O(n)$ times. The complexity of the incremental power analysis is $O(N_g)$ since we require two sum operations of the lognormal RVs. For the statistical timing perturbation computation most of the *max* computations in the cutset can be reused by storing the information as a heap. Thus, the timing perturbation computation has a complexity of $O(N_g \log(n))$. Thus the overall complexity of the approach is $O(nN_g\log(n))$, which is a large improvement compared to the brute-force approach.

4. Results and Implementation Details

We implemented the proposed approach in C and compared our yield improvements to a deterministic circuit optimization technique. Our proposed approach for the computation of the yield gradient is written as a subroutine which the optimizer uses to calculate the gradient of the objective function. The yield analysis engine serves as the subroutine to calculate the objective function itself. Following, we present the accuracy and runtime results for the proposed approach.

If we assume that reverse and forward SSTA give exact timing distributions at each node, then the procedure would be exact as well. However, as noted before, due to the Gaussian approximation considered while computing the maximum introduces a small inaccuracy while performing forward and reverse SSTA . Now, since this inaccuracy is a function of circuit topology and reconvergence structure the sensitivity of yield computed using only FORWARDSSTA based brute-force is negligible. Table 1 shows the runtime comparison and accuracy results of the proposed gradient computation procedure as compared to the naïve brute force approach. The circuit size in terms of the number of gates and the average cut-width over all nodes in the circuit is also reported in the second and the third columns, respectively. Runtime per gradient vector computation

Table 1.Comparison of Yield gradient computation using FASTYIELDGRADIENT and brute-force approach.

Bench. Circuit	#gates	Average Cut-Size	Brute-Force	Fast Gradient	Speed-up	Max. Error (%)	Avg. Error (%)
c432	257	46.8	0.6	0.1	7.0	1.2	6.7E-03
c499	545	96.0	5.2	0.7	7.1	0.3	3.4E-03
c880	501	103.8	4.8	0.6	7.9	0.2	1.2E-02
c1908	604	84.3	6.5	0.7	9.8	2.6	1.7E-02
c2670	781	248.2	5.7	1.1	5.3	5.2	1.0E-03
c3540	1164	140.6	41.9	3.5	12.1	1.0	1.5E-03
c5315	1693	295.1	130.3	13.1	9.9	7.4	2.4E-03
c6288	3835	297.9	991.3	51.0	19.5	1.8	1.4E-03
c7552	2153	317.8	979.4	51.1	19.2	1.8	1.4E-03
i2	193	105.0	214.8	20.3	10.6	2.0	1.5E-03
i4	265	105.8	0.4	0.1	2.9	1.3	1.8E-02
i5	424	137.5	0.6	0.2	2.8	8.0	9.4E-02
i6	462	177.8	3.2	0.7	4.5	0.1	2.3E-02
i7	770	252.3	2.0	0.7	3.0	1.8	6.0E-02
i8	1014	243.7	10.4	2.2	4.7	1.0	4.0E-02
i10	2483	413.0	19.6	3.8	5.2	2.3	2.3E-03

using the brute approach and the proposed procedure are given in Columns 4 and 5, respectively. The speed up of the proposed method over the brute-force approach is given in Column 6, and ranges between 3X to 20X and is found to be larger for bigger circuits. The maximum error, over all gates, found using gradient computation normalized with respect to the brute-force method is given in Column 7, and is found to be small in most cases with an average of 2.4%. The error averaged error over all gates in the circuit is given in the last columns of Table 1 and is found to be extremely small.

4.1 Yield Optimization

The gates in our standard cell library are characterized for a set of sizes in the range from minimum size to maximum size and the delay and leakage power for intermediate gate sizes is obtained using linear interpolation. All designs are then deterministically optimized for power under delay constraints using either design compiler or LANCELOT [18]. We use our statistical yield maximization approach to improve the yield of this optimized design for a set of different power and timing constraints. Our results indicate that performing statistical optimization can significantly improve the timing yield of the design. We compare our results based on the ISCAS85 [23] benchmarks which were synthesized in a 130 nm technology.

The yield optimization results are given in Table 2. The first sub-section including columns 2, 3, 4 and 5 report the initial timing and power statistics resulting from a deterministically optimized circuit. The deterministic optimization was performed using nominal delay and power models. We present yield optimization results for two sets of constraints. The first set includes yield optimization for aggressive nominal value constraints. As a deterministic optimizer is unaware of the variation in power and timing and their correlation, the initial yield at nominal constraints is extremely small. However, the proposed variability aware yield optimization significantly improves the yield. For example, the yield for benchmark circuits c432, c499 and c880 dramatically improves from close to 0% to up to 40%. Column 6, 7, 8 and 9 report the post optimization timing and power statistics of the circuit. The initial yield subject to nominal value constraints and the yield after performing optimization are given in Columns 10 and 11, respectively. The second set of results report the performance of yield optimization under pessimistic constraints. For this case we use the nominal value offset by one standard deviation as the constraint for both power and timing while defining the objective function for optimization. Again columns 12, 13, 14 and 15 list the post optimization statistics of the circuit whereas columns 16 and 17 report the results achieved after performing the proposed yield optimization. As the constraints are relaxed in this case the initial yield of the circuit improves and for the same reason the improvements achieved are relatively smaller as compared to the previous case. The maximum improvement in this case is found to be greater than 20% for the benchmark circuit c1908.

Table 2: Yield Optimization results for different power and timing constraints

Bench. Circuits	Initial solution				D < Dm, P < Pm						D < Dm + Ds, P < Pm +Ps					
	Delay(ps)		Power (μW)		Delay(ps)		Power (μW)		Yield(%)		Delay(ps)		Power (μW)		Yield(%)	
	Dm	Ds	Pm	Ps	Dm	Ds	Pm	Ps	Init.	Opt.	Dm	Ds	Pm	Ps	Init.	Opt.
c432	669.75	23.194	10.35	3.78	627	22	10.2	3.72	~0	48.7	650.7	22	9.98	3.62	42.07	48.83
c499	754.89	25.622	30.66	10.56	716	23	31.01	10.1	~0	46.87	739	25	28.81	9.87	41.87	46.87
c880	701.86	23.005	25.48	8.83	669	21	23.1	7.99	~0	45.26	695	22	23.2	8	42	45.2
c1908	924.83	27.311	16.09	5.47	928.1	27.48	15.4	5.2	1.25	19.25	938	31	4.8	1.6	42.6	65.7
c2670	684.82	23.241	6.3	2.16	690	23.5	6.1	2.09	0.98	19.28	657	22	6.15	2.15	42.1	49.22
c3540	1134.5	34.641	48.73	16.14	1126	34.6	46.1	15.2	0.05	29.3	1126	34.6	46.1	15.25	42	44.7
c5315	981.82	30.862	74.97	24.4	995	31	69.7	22.6	~0	16.41	985	31	72.6	23.6	42.03	42.05
c6288	2638.7	71.818	98.37	30.66	2651	72	91.2	28	1.2	19.2	2591	70.95	96.5	30	42.1	47.62
c7552	1180.5	33.774	72.39	23.54	1186	34.5	67.4	21.8	~0	19.1	1150	30.3	67.8	21	42.03	48.4

5. Conclusions

To the best of our knowledge, we have presented the first approach to perform gate-level parametric yield optimization considering constraints on power and performance, along with their correlation. The approach for yield computation is shown to be computationally efficient and is shown to provide an 8X improvement in runtime, as compared to a brute-force gradient computation approach. The yield gradient is used to guide a large-scale non linear optimizer to improve the yield of a design that has been optimized deterministically, under varying power and delay constraints. The results show that we can achieve improvements in yield which are as large as 40%.

Acknowledgements

This work was supported in part by NSF, SRC and MARCO/DARPA.

References

[1] T. Karnik, S. Borkar, and V. De, "Sub-90 nm technologies challenges and opportunities for CAD," *ACM/IEEE ICCAD*, pp. 203-206, 2002.

[2] R. Rao *et al*., "Parametric yield analysis and constrained-based supply voltage optimization," *ACM/IEEE ISQED*, pp. 284-290, 2005.

[3] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," *ACM/IEEE ICCAD*, pp. 621-625, 2003.

[4] C. Visweswariah *et al.*, "First-order incremental block-based statistical timing analysis," *ACM/IEEE DAC*, pp. 331-336, 2004.

[5] A. Agarwal *et al*., "Statistical timing analysis using bounds and selective enumeration," *IEEE Trans. on CAD*, pp. 1243-1260, pp. 1243-1260, Sept. 2003.

[6] A. Devgan and C. Kashyap, "Block-Based statistical timing analysis with uncertainty," *ACM/IEEE ICCAD*, pp. 607-614, 2003.

[7] M. Orshansky and A. Bandopadhyay, "Fast statistical timing analysis handling arbitrary delay correlations," *ACM/IEEE DAC*, pp. 337-342, 2004.

[8] R.R. Rao, *et al*., "Statistical analysis of subthreshold leakage current for VLSI circuits," *IEEE Trans. VLSI Systems*, pp.131-139, Feb. 2004.

[9] S. Narendra *et al*., "Full-chip sub-threshold leakage power prediction model for sub-0.18μm CMOS," *ACM/IEEE ISLPED, pp. 19-23,* 2002.

[10] X. Bai, *et al*., "Uncertainty aware circuit optimization," *ACM/IEEE DAC,* pp.58-63, 2002.

[11] S. Raj, S. Vrudhula, and J. Wang, "A methodology to improve timing yield in the presence of process variations," *ACM/IEEE DAC*, pp. 448-453, 2004.

[12] S. Choi, B. Paul and K. Roy, "Novel sizing algorithm for yield improvement under process variation in nanometer technology," *IEEE/ACM DAC*, pp. 454-459, 2004.

[13] A. Agarwal *et al*., "Statistical timing based optimization using gate sizing," *ACM/IEEE DATE*, pp. 400-405, 2005.

[14] A. Srivastava, D. Sylvester, and D. Blaauw, "Statistical optimization of leakage power considering process variations using dual-Vth and sizing," *ACM/IEEE DAC*, pp. 773-778, 2004.

[15] A. Davoodi, V. Khandelwal, and A. Srivastava, "Variability inspired implementation selection problem," *ACM/IEEE ICCAD*, pp. 423-427, 2004.

[16] M. Mani and M. Orshansky, "A new statistical algorithm for gate sizing," *ACM/IEEE ICCD*, pp. 272-277, 2004.

[17] A. Srivastava *et al.*, "Accurate and efficient gate level parametric yield estimation considering correlated variations in leakage power and performance," *ACM/IEEE DAC*, pp. 535-540, 2005..

[18] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *LANCELOT: A Fortran package for large-scale non-linear optimizer (Release A)*. Springer, Verlag, 1992.

[19] D. F. Morrison, *Multivariate statistical methods*, McGraw-Hill Book Company, 1967.

[20] C. Clark, "The greatest of a finite set of random variables," *Operations Research,* vol. 9, pp. 85-91, 1961.

[21] S.C. Schwartz and Y.S. Yeh, "On the distribution function and moments of power sums with lognormal components," *Bell Systems Technical Journal*, vol.61, pp.1441-1462, Sep. 1982.

[22] J. H. Cadwell, "The bivariate normal integral," *Biometrika*, pp. 31-35, Dec. 1951.

[23] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," *Proc. ISCAS*, pp. 695-698, May 1989.

[24] D. Lee, V. Zolotov and D. Blaauw, "Static timing analysis using backward propagation," *ACM/IEEE DAC*, pp. 664-669, 2004.