# A Probabilistic Method to Determine the Minimum Leakage Vector for Combinational Designs in the Presence of Random PVT Variations

Kanupriya Gulati [a], Nikhil Jayakumar [b], Sunil P. Khatri [a] and D. M. H. Walker [c]

[a] *Department of ECE, Texas A&M University, College Station TX 77843 USA*

[b] *Texas Instruments, Inc. Dallas TX 75243 USA*

[c] *Department of CS, Texas A&M University, College Station TX 77843 USA*

**Abstract**

The control of leakage power consumption is a growing design challenge for current and future CMOS circuits. Among existing techniques, 'parking' a circuit in a minimum leakage state during its standby mode of operation requires minimal circuit modification and results in significant leakage reduction. In this paper we present a heuristic approach (referred to as MLVC) to determine the input vector which minimizes leakage for a combinational design. This approach utilizes approximate signal probabilities of internal nodes to aid in finding the minimum leakage vector. We utilize a probabilistic heuristic to select the next gate to be processed as well as to select the best state of the selected gate. A fast SAT solver is employed to ensure the consistency of the assignments that are made in this process. A variant of MLVC, referred to as MLVC-VAR, is also presented. MLVC-VAR includes the effect of random variations in leakage values due to process, voltage and temperature (PVT) variations. Including the effect of PVT variations for determining minimum leakage vector is crucial because leakage currents have an exponential dependence on power supply, threshold voltage and temperature. To the best of the authors' knowledge, no other minimum leakage vector determination work has to date included the effect of PVT variations. Experimental results indicate that our MLVC method has very low runtimes, with excellent accuracy compared to existing approaches. Further, the comparison of the mean and standard deviation of the circuit leakage values for MLVC with MLVC-VAR and an existing random vector generating approach proves the need for considering these variations while determining the minimum leakage vector. MLVC-VAR reports, on average, about 9.69% improvement over MLVC with similar runtimes and 5.98% improvement over the random vector generation approach with significantly lower runtimes.

*Key words:* Subthreshold leakage, PVT variations.

## 1 Introduction

Traditionally, dynamic (switching) power has dominated the total power consumption of a VLSI IC. However, due to current scaling trends, leakage power has now become a major component of the total power consumption in VLSI circuits. The leakage current for a PMOS or NMOS device corresponds to the $I_{ds}$ of the device when the device is in the *cut-off* or *sub-threshold* region of operation. The expression for this current [1] is:

$$I_{ds} = \frac{W}{L} I_0 e^{(\frac{V_{gs}-V_T-V_{off}}{nv_t})}(1 - e^{(-\frac{V_{ds}}{v_t})}) \quad (1)$$

Here $I_0$ and $V_{off}$ [1] are constants, while $v_t$ is the thermal voltage (26mV at 300°K) and $n$ is the sub-threshold swing parameter. Note that $I_{ds}$ increases exponentially with a decrease in $V_T$. This is why a reduction in supply voltage (which is accompanied by a reduction in threshold voltage) results in exponential increase in leakage. This is expected to be a major concern for VLSI design in the nanometer realm [2].

Further, the increasing demand for portable/hand-held electronics has meant that leakage power consumption has received even greater attention. Since these portable devices spend most of their time in a *standby* state (also sometimes called the *sleep* state), reducing the leakage power consumption in this *standby* state is crucial to extending the battery life of these designs.

One of the natural techniques for reducing the leakage of a circuit is to gate the power supply using power-gating transistors (also called *sleep* transistors). Typically high-$V_T$ power-gating transistors are placed between the power supplies and the logic gates (MTCMOS [3,4]). In some cases these power-gating transistors are embedded in the logic gates [5]. In standby, these power-gating transistors are turned off, thus shutting off power to the circuit in question. Such power-gating techniques can reduce circuit leakages by 2 to 3 orders of magnitude. However, the addition of a power-gating transistor causes an increase in delay of the circuit. Further, the process of 'waking' the circuit involves a delay (and a power transient), since the supply rails need to reach their stable values before the circuit can operate again.

Increasing $V_T$ via body effect and bulk voltage modulation [6,7] is another way to reduce leakage power. The leakage current of a transistor decreases with greater applied Reverse Body Bias (RBB). RBB affects $V_T$ through the body

[1] Typically $V_{off} = -0.08V$

effect, and sub-threshold leakage has an exponential dependence on $V_T$, as seen in Equation 1. The body effect equation can be written as $V_T = V_T^0 + \gamma\sqrt{V_{sb}}$ where $V_T^0$ is the threshold voltage at zero $V_{sb}$.

All the techniques listed above require significant circuit modifications in order to reduce leakage. Another technique, which achieves significant leakage reduction, is the technique of *parking* a circuit in its minimum leakage state. This technique involves very little or no circuit modification and does not require any additional power supplies. A combinational circuit is *parked* in a particular state by driving the primary inputs of the circuit to a particular value. This value can be scanned in or forced using MUXes (with the standby/sleep signal used as a select signal for the MUX).

Table 1 shows the leakage of a NAND3 gate for all possible input vectors to the gate. The leakage values shown are from a SPICE simulation using the 100nm BPTM [8] models, with a $V_{DD}$ of 1.2V.

Table 1
Leakage of a NAND3 gate

| Input | Leakage(A) |
|---|---|
| 000 | 1.37e-10 |
| 001 | 2.70e-10 |
| 010 | 2.70e-10 |
| 011 | 4.96e-09 |
| 100 | 2.62e-10 |
| 101 | 2.68e-09 |
| 110 | 2.51e-09 |
| 111 | 1.01e-08 |

As can be seen from Table 1, setting a gate in its minimal leakage state (000 in the case of the NAND3 gate) can reduce leakage by about 2 orders of magnitude. Ideally, it is desirable to set *every* gate in the circuit to its minimal leakage state. However, this may not be possible due to the logical interdependencies of the inputs of the gates. Finding the minimum leakage input vector, at the circuit level, is an NP-hard problem. Several research efforts have addressed the problem of determining an input vector that minimizes leakage for a design. Our approach falls into this category. The problem of finding a minimal leakage vector, also termed Input Vector Control (IVC) determination, can be viewed as one of selecting the state of each gate in the circuit such that the total leakage over all gates is minimized, and the states of each gate in the circuit are logically feasible (i.e. is logically compatible with the states of all the other gates). The distinguishing feature of our approach is that it is guided by signal probabilities. In other words, the selection of the best candidate gate, as well as the input state to use for that gate, is performed probabilistically. The intuition behind such selections is that they have a high likelihood of resulting in a circuit state which is logically justifiable, while

minimizing leakage as well. Additionally, the effect of PVT variations can be elegantly incorporated into such a probabilistic formulation.

With the decrease in process feature sizes, the effect of PVT variations have become significant. Since sub-threshold leakage has a critical dependency on temperature, power supply, channel length and threshold voltage, the PVT variations heavily influence the leakage values and correspondingly the minimum leakage vector determination. In [9], the authors experimentally prove that a simple assumption of uniform temperature and power supply variation can underestimate the full chip leakage by 30%. In [10], the authors establish the importance of considering the variations of within-die threshold voltage and channel length for accurate sub-threshold leakage current prediction. They determine that the sub-threshold leakage power can be underestimated or overestimated by 1.5X to 6.5X by ignoring these within-die variations.

Keeping the significant dependence of leakage on PVT variations in mind, a variant of MLVC, called MLVC-VAR, is also presented. *MLVC-VAR includes the effect of random PVT variations while determining IVC.* Currently our approach does not account for correlation between the PVT variables. However, these correlations can be easily incorporated into the approach as we describe in the sequel. The effect of PVT variations are considered in the formulation of both the heuristics: for selecting the best candidate gate and the best leakage state for that gate. To the best of our knowledge, no other work on IVC determination to date has considered these important variations in its formulation.

In our experiments, we compare the accuracy and runtimes for MLVC with other existing techniques for determining IVC. Further, we compare the mean circuit leakage and the standard deviation of the circuit leakage of the input vectors determined by MLVC and MLVC-VAR. The remainder of this paper is organized as follows: Section 2 discusses some previous work in this area. In Section 3 we describe MLVC and MLVC-VAR. In Section 4 we present experimental results, while conclusions and future work are discussed in Section 5.

## 2 Previous Work

The problem of finding the minimum leakage sleep vector for a combinational CMOS gate-level circuit has received some attention recently. In [11], the authors find a minimal leakage vector using random search with the number of vectors used for the random search selected to achieve a specified statistical confidence and tolerance. In [12], the authors reported a genetic algorithm based approach to solve the problem. The authors of [13] introduce a concept called leakage observability, and based on this idea, describe a greedy ap-

proach as well as an exact branch and bound search to find the maximum and minimum leakage bounds. The work of [14] is based on an ILP formulation. It makes use of pseudo-Boolean functions which are incorporated into an optimal ILP model and a heuristic mixed integer linear programming method as well. In contrast to these approaches, our approaches (MLVC and MLVC-VAR) are heuristics that use signal probabilities and leakage values of the gates to help assign values to the nodes in a combinational circuit. In [15,16], the authors present an MDD [17] based algorithm to determine the lowest leakage state of a circuit. Unlike our method, [16] computes a leakage histogram for the design. The use of MDD based minimum leakage vector computations limits the applicability of [15] to small designs.

In [18], the authors present a greedy search based heuristic, guided by node controllabilities and functional dependencies. The algorithm used in [18] involves finding the controllability and the controllability lists of all the nodes in a circuit and then using this information as a guide to choose gates to set to a low leakage state. The controllability of a node is defined as the minimum number of inputs that have to be assigned to specific states in order to force the node to a particular state (based on concepts used in automatic test pattern generation). Controllability lists are defined as the minimum constraints necessary on the input vector to force a node to particular state. The time complexity of their algorithm is reported to $O(n^2)$ where $n$ is the number of cells (gates) in the circuit. However in estimating the complexity of their algorithm, it is not clear if the authors include the time taken to generate the controllabilities and controllability lists of each node in the circuit. While finding the controllabilities can be done fairly easily [19], generating the controllability lists can be more involved. In both our approaches we do not compute node controllabilities or their controllability lists. We compute signal probabilities instead, which are computed in time that is linear in circuit size. The algorithms for both MLVC and MLVC-VAR are explained in detail in Section 3.

In [20], the authors describe several methods to set pass/fail limits for $I_{DDQ}$ testing, among which is a probabilistic method. For each cell in a design (each cell is assumed to have a single output, implemented in static CMOS), the authors compute the maximum $I_{DDQ}$ when the output is ON (OFF), assuming $4\sigma$ process variation limits. Additionally, the cell probabilities are determined for the input vectors that result in the maximum $I_{DDQ}$ of the cell for both the ON and OFF state. In contrast to [20], the approaches in this paper takes into account probabilities of all input vectors of a cell implicitly, and not just those of two outputs that result in a worst-case $I_{DDQ}$ value. Further, the signal probabilities, in the heuristics presented in this paper, are adjusted for reconvergence, unlike [20].

In [21], the authors express the problem of finding a minimum leakage vector

as a satisfiability problem and use an incremental SAT solver to find the minimum and maximum leakage current. While their approach works well for small circuits, the authors report very large runtimes for large circuits. The authors therefore suggest using their algorithm as a checker for the random search suggested in [11]. Our approaches can handle larger circuits with low runtimes and good accuracy, as shown in Section 4.

There are existing research efforts [22–26] in which circuit level modifications are performed in order to achieve lower leakage. In contrast to these approaches, our approach only involves parking the circuit in a particular state and requires no gate replacement or circuit modification. Some of these approaches [25,26] target simultaneous generation of minimum leakage vector and performing gate replacements, which makes the problem they address substantially different from ours.

All of the above cited IVC determination approaches ignore the within-die PVT variations. Some work which estimates leakage values considering these variations is discussed below.

The authors in [9] establish (by using iterative numerical methods) the dependency of leakage current on temperature and power supply, and prove that an assumption of uniform temperature and power supply variation can underestimate the full chip leakage by 30%. In [10], the authors find that the sub-threshold leakage power can be overestimated or underestimated by 1.5X - 6.5X if variations of within-die threshold voltage and channel length values are ignored. The authors of [27] present a probabilistic framework for full-chip estimation of leakage power distribution considering inter and intra-die process and temperature variations. A method for analyzing leakage current under process parameter variations including spatial correlations can be found in [28]. On the other hand, in [29], the authors develop an analytical expression to estimate the PDF of the leakage current and thence, estimate the variation in leakage current due to gate length and process variability.

The authors of [30] propose a projection-based algorithm to estimate the full-chip leakage power, considering both inter-die and intra-die variations, by extracting a low-rank quadratic model. In [31], the authors present a gate-sizing methodology to minimize the leakage power in the presence of process variations. They formulate a geometric programming problem by modeling leakage as a posynomial function.

Most of these papers provide a mathematical or probabilistic framework for estimation of circuit leakage current in the presence of PVT variations, and hence the leakage power consumption. Our heuristic, MLVC-VAR, in contrast, considers the PVT variations *while determining IVC*. No existing work on IVC determination considers PVT variations. An extended abstract of the MLVC

work presented in this paper can be found in [32]. It contains the details of MLVC but does not discuss MLVC-VAR, and its results. Further, the run times reported for MLVC in [32] have been improved by 10X by careful modifications in the algorithm, which is discussed in the next section.

**Motivating example for MLVC-VAR:** This paper addresses intra-die variations. Intra-die variations are an important contributor to the mean leakage and the standard deviation of leakage for a combinational circuit. Since the intra-die variations of a gate are dependent on the logic state of the gate [33,34], we propose the following objective for the MLVC-VAR approach: We aim at reducing the mean leakage plus 6 times the standard deviation (Cost function = $\mu + 6^*\sigma$) of the combinational circuit, by choosing the input vector that sets the logic states of all the gates in the most 'favorable' manner (conducive to lowering the cost function). It can be conjectured that considering intra-die variations just leads to an increased expectation value, but the best state remains the best state. By this reasoning, optimization without intra-die PVT will lead to nearly the same result. The following example explains why this conjecture is false:

The mean, nominal and standard deviation of the logic gates (Inverter, NOR2 and AND2) for different logic states are listed in Table 2. Note that the leakage values are from the library we used in our experiments.

Table 2
Mean, Nominal and Standard Deviation for the Logic Gates

| **Inverter** | | | | |
|---|---|---|---|---|
| Input | Output | $\mu$(nA) | Nominal (nA) | $\sigma$ (nA) |
| 0 | 1 | 1.8832 | 2.2904 | 1.5055 |
| 1 | 0 | 3.7881 | 6.5253 | 8.2548 |
| **NOR2** | | | | |
| Input | Output | $\mu$(nA) | Nominal (nA) | $\sigma$ (nA) |
| 00 | 1 | 3.7668 | 4.5818 | 3.0284 |
| 01 | 0 | 4.4738 | 7.0279 | 7.7904 |
| 10 | 0 | 7.5724 | 13.0926 | 16.7359 |
| 11 | 0 | 0.4468 | 0.5574 | 0.3854 |
| **AND2** | | | | |
| Input | Output | $\mu$(nA) | Nominal (nA) | $\sigma$ (nA) |
| 00 | 0 | 3.9742 | 6.7527 | 12.5603 |
| 01 | 0 | 7.0834 | 10.5649 | 11.3243 |
| 10 | 0 | 5.5780 | 8.6271 | 7.5410 |
| 11 | 1 | 9.4602 | 15.4030 | 10.3201 |

Consider the circuit in Figure 1 that is composed of three logic gates. The

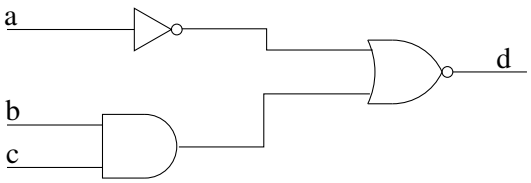


Fig. 1. Example Circuit for Motivating MLVC-VAR.

output $d$ of the circuit evaluates to $\overline{a} + b \cdot c$. In case of MLVC (or an IVC determination technique which only aims at reducing the nominal circuit leakage) the best input vector would be 000 (i.e. the assignment of 0 to all three inputs a, b and c). From Table 2, in this case the total nominal leakage of the circuit would be 16.0710 nA. Similarly, the metric $\mu + 6 * \sigma$ in this case would be 141.4684 nA.

On the other hand, MLVC-VAR aims at reducing the metric $\mu + 6 * \sigma$ as opposed to only reducing the nominal leakage for the combinational circuit. In this case the best input vector assignment would be a=0, b=1 and c=1. Again the values are computed using Table 2. In this case even though the nominal leakage of the circuit would be 18.2508 nA (which is 11.94% higher than that reported from MLVC) the metric $\mu + 6 * \sigma$ in this case would be 85.0562 nA (which is 66.32% lower than that reported using MLVC).

This example explains why MLVC in the presence of intra-die variations would not be adequate. MLVC alone might possibly yield a vector for which the worst case ($\mu + 6 * \sigma$) leakage of the combinational circuit is higher than what MLVC-VAR would compute.

## 3 Our Approach

The outlines of MLVC and MLVC-VAR are as follows:

- First, for both MLVC and MLVC-VAR, we compute signal probabilities for all nodes in the design, assuming that all inputs have a signal probability of 0.5. These probabilities are heuristically adjusted for inaccuracies arising from reconvergent fanouts.
- Next, we select the best candidate gate whose leakage we would like to set in a given iteration. For both MLVC and MLVC-VAR, this is performed by selecting the gate that is probabilistically most likely to result in the largest leakage reduction. For MLVC-VAR, we consider (in addition to the probabilistic signal values) the mean and standard deviation of the leakages at each state[2], of each gate, before choosing a gate which results in the

[2] A state of a gate stands for an assignment of the logical (1 or 0) value at each of its input, and hence a logical value assigned at its output. For example, a 3-input

lowering of the standard deviation of the circuit leakage.

- We next select the best state for the chosen gate. In MLVC, for the gate thus selected we next assign its best state such that the leakage of the selected gate is probabilistically minimized. In MLVC-VAR, this state is chosen by considering not only the signal probabilities and leakage values, but also the standard deviation of the leakage values due to PVT variations. All other gates in the circuit which are newly implied by the state just selected, are accounted for while making this decision. Prior to computing the cost metric for this step, we first test if the candidate state is consistent with the assignments made in the previous runs.
- In both MLVC and MLVC-VAR, we next test if the logic values that were set to 1 or 0 during this iteration are satisfiable, by calling a Boolean Satisfiability (SAT) solver. The SAT [35] problem can be defined as follows. Given a set $V$ of variables, and a collection $C$ of Conjunctive Normal Form (CNF) clauses over $V$, the SAT problem consists of determining if there is a satisfying truth assignment for $C$. For any circuit, one can potentially generate a CNF to represent the circuit [36]. In our method, the SAT solver is called on the CNF of the circuit to test if the currently assigned logic values are consistent with the circuit. Further, the SAT solver is called every $p$ iterations to reduce the runtime. If the circuit is unsatisfiable, we undo the assignments of the last $p$ iterations, and find the iteration that caused the circuit to become unsatisfied. After making a different selection for that iteration, we proceed as before.
- After any iteration, for both MLVC and MLVC-VAR, gate probabilities are adjusted to account for the nodes that were newly assigned fixed logic values.
- A fixed number of passes are made for the circuit, with the above steps being applied successively. Each pass is more "lenient" in setting a node to a logic value $v$ when its signal probability is different from those of $v$. The last pass is most lenient, allowing any $v$ to be accepted. This feature is common to both MLVC and MLVC-VAR.

Algorithm 1 describes the pseudocode for MLVC, for a combinational network $\eta$. The algorithm for MLVC-VAR is identical to Algorithm 1, except for the functions $find\_best\_gate(\eta)$ and $find\_best\_leakage\_state(G, \eta)$. The differences are detailed in the following subsections.

### *3.1 Computing Signal Probabilities*

The algorithm $compute\_minimum\_leakage\_vector(\eta)$ for both MLVC and MLVC-VAR, begins by computing signal probabilities for all nodes in the network

gate will have 8 states.

**Algorithm 1** Pseudocode of MLVC

```
compute_minimum_leakage_vector(η, p){
compute_signal_probabilities(η)
finalvalues ← Φ
for i = 1; i ≤ k; i++ do
  temporaryvalues ← Φ
  iteration = 1
  (G = find_best_gate(η)
  if (G is not marked visited) then
    (S = find_best_leakage_state(G, η)
    if S satisfies m_i then
      temporaryvalues ← temporaryvalues ∪ S ∪ get_implications(S)
      propagate probabilities in TFO of temporaryvalues nodes
    end if
    if iteration is a multiple of p OR all inputs assigned/implied then
      if temporaryvalues are satisfiable then
        if all inputs assigned then
          exit
        end if
        finalvalues ← finalvalues ∪ temporaryvalues
      else
        temporaryvalues ← finalvalues
      end if
    end if
  end if
  iteration += 1
end for
}
```

$\eta$.

**Definition 1** *Signal probability of a node $X$ is the probability of $X$ being at logic level '1'.*

The inputs are assumed to have probabilities of 0.5, and these probabilities are propagated throughout the circuit. If the input $i$ of an $n$-input AND gate has probability $p_i$, then the output has probability $\Pi_i p_i$. Likewise, for an OR gate, the output has probability $1 - \Pi_i(1 - p_i)$. The probabilities of other gates can be found in a similar fashion. After the initial pass of propagation, we heuristically adjust for reconvergent fanouts. The heuristic for probability adjustment in the presence of reconvergence is explained with the help of Figure 2.

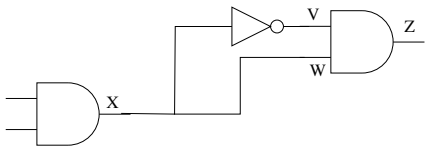


Fig. 2. Adjusting Probabilities for Reconverging Nodes

Suppose a node $X$, with a statically computed probability of $P_X$ reconverges at $Z$. Then we set the signal probability of $X$ to 1 and 0, and find the probabilities of the inputs to the reconvergent gate ($V$ and $W$). Suppose the probabilities of $V$ ($W$) are $V_1$ ($W_1$) and $V_0$ ($W_0$) respectively, when $X$ is set to 1(0). In this case, the new probability of $Z$ is $P_Z^{new} = \frac{V_0 \cdot W_0 + V_1 \cdot W_1}{2}$.

From this we compute the adjustment factor for the probability of $Z$, as follows. Note that the adjustment factor is computed exactly once, in the beginning of the procedure.

$Adjustment(Z) = \frac{(P_Z^{new} - P_Z)}{P_Z}$

The physical meaning of $P_Z^{new}$ is explained as follows. Assuming that 0 and 1 are equally likely at $X$, we can say that the signal probability of $Z$ is $P_Z^{new} = \frac{V_0 \cdot W_0 + V_1 \cdot W_1}{2}$. If there was no reconvergence, then $P_Z^{new}$ would be identical to $P_Z$. In the presence of reconvergence, however, $P_Z^{new}$ deviates from $P_Z$ by an amount equal to $Adjustment(Z)$.

In future updates of the probability of the node $Z$, suppose the statically computed probability of node $Z$ is $P_Z^{modified}$. In that case, the final adjusted value of the probability of node $Z$ is

$P_Z^{adj} = (P_Z^{modified}) \cdot (1 + Adjustment(Z))$.

In other words, $Adjustment(Z)$ is computed once, and utilized to adjust the statically computed values of the probability of node $Z$, each time it is modified due to other assignments in the circuit.

In the example of Figure 2, $Adjustment(Z) = -1$. Therefore, $P_Z^{adj} = 0$ each time the probability of $Z$ is modified. This is reasonable, given that the output $Z$ is logically 0.

If an adjustment of the probability of a node results in its probability becoming higher than $P_{threshold}$ (lower than $1 - P_{threshold}$), then the probability of the node is capped at $P_{threshold}$ ($1 - P_{threshold}$) respectively.

### *3.2 Finding the Best Leakage Candidate*

Once signal probabilities are computed, we next select the best candidate gate whose input state we would like to finalize. For MLVC, gates are ranked by the probabilistic criterion:

$C = \frac{\sum(p_i \cdot l_i)}{\sum(p_i)}(l_i^{max} - l_i^{min})$

Here, $p_i$ is the probability that the gate is in state $i$. By "state", we mean a complete assignment of the inputs of the gate. The quantity $l_i$ is the nominal leakage of the state $i$. The value $l_i^{max}$ ($l_i^{min}$) is the maximum (minimum) nominal leakage value of this gate. The gate with the maximum value of $C$ is selected. In other words, this criterion selects gates that have a high probability of being in a high-leakage state. The last term in the expression for $C$

ensures that gates with large leakage ranges are favored, since they offer potentially greater optimization flexibility. The gate that maximizes $C$ is selected preferentially over others.

Note that due to the 'snapping' of any signal probability higher (lower) than $P_{threshold}$ $(1 - P_{threshold})$ to $P_{threshold}$ $(1 - P_{threshold})$, no node can have signal probabilities identically equal to 1 or 0. Hence there are instances when the sum of probabilities of all states of a node does not sum to unity and therefore the denominator in the above expression is not replaced by unity.

For MLVC-VAR, the expression is further biased to select a gate for which the standard deviation in the leakage values due to PVT variations is maximum. This biasing favors the selection of a gate which has higher variations in the leakage values. This is reasonable because in the next step, this gate is set to a state (among the possible states) which minimizes leakage as well as the leakage variations. Hence it helps in avoiding a large standard deviation in the expected overall circuit leakage. Therefore the final expression for $C_{var}$ for MLVC-VAR is:
$C_{var} = \frac{\sum(p_i \cdot l_i \cdot r_i)}{\sum(p_i)}(l_i^{max} - l_i^{min})$

Here $r_i$ is the range of the leakage value of a gate in state $i$. This range accounts for the PVT variations. Note that the $l_i$ in this expression is the *mean* leakage of the state $i$, unlike for MLVC, where $l_i$ is the nominal leakage of state $i$. Again, the gate that maximizes $C_{var}$ is selected preferentially over others.

### *3.3 Finding Best Leakage State for Selected Gate*

Suppose a gate $G$ was selected by the previous step. For MLVC, we now want to assign it a state such that its leakage is minimized. This is done by applying the probabilistic criterion $L$ below. Note that all gates other than $G$ whose states become fully assigned [3] on account of implying the current state of $G$, are also included in the computation of $L$. Let the number of such states be $n$. The value of probabilistic leakage in the numerator of $L$ is normalized with respect to the number of such states, and is computed as follows:

$$L = \frac{\sum_j (d_j \cdot l_j)}{n}$$

$L$ is computed for only those states whose assignments are consistent with the assignments made in the earlier passes. This test is done by invoking the BerkMin [37] satisfiability solver. If the assignment of a state $s$ fails the test, we proceed to the next state for $G$, else (i.e. $s$ is 'legal') we compute $L$

[3] A gate is said to be *fully assigned* if all its inputs are assigned to specific logic values

for $s$. Among all the legal states of the gate $G$, the state that minimizes $L$ is preferentially selected over others. Here $d_j$ is the *distance* of the values assigned to the gate inputs from their probabilistic values. For example, consider an AND gate with inputs $a$ and $b$ with probabilities 0.1 and 0.7 respectively. If inputs $a$ and $b$ in state $j$ are logic 1 and logic 0 respectively, then the distance $d_j$ is $(|1-0.1|)(|0.7-0|)$. $l_j$ is the nominal leakage of state $j$. Note that the probability of $a$ or $b$ can never be exactly 1 or 0, because probability values higher (lower) than $P_{threshold}$ $(1-P_{threshold})$ are snapped to $P_{threshold}$ (or 1-$P_{threshold}$). Hence $d_j$ can never be exactly 0.

By minimizing $L$, we choose a state which has the lowest distance from its current probabilities and because these probabilities are updated to account for the logic and for structure of the circuit, this state would reduce the chances of assigning logically conflicting states. In order to bias the state selection towards assignments with lower leakage the distance is incremented by a value $\beta$. Likewise, in order to bias the state selection towards those with lower distance, we increment $l_j$ by a fixed value $\gamma$. The relative values of $\beta$ and $\gamma$ are selected based on the relative scale of $d_j$ and $l_j$ values. In practice these values are determined experimentally.

Therefore, the modified value of $L$ that is used is

$$L = \frac{\sum_j (d_j+\beta)\cdot(l_j+\gamma)}{n}.$$

For MLVC-VAR, analogous to $L$ above, we utilize the selection criterion $L_{var}$. It is computed as follows:

$$L_{var} = \frac{\sum_j (d_j+\beta)\cdot(l_j+\gamma)\cdot(r_j+\rho)}{n}$$

All variables in this expression denote the same values as the ones explained earlier in this sub-section, except $l_j$ is the mean (instead of nominal) leakage at state $j$. $r_j$ is the range due to variations in the leakage values due to PVT variations. If the leakage distribution of a gate is $N(\mu_g, \sigma_g)$, then $r_j = 6 \cdot \sigma_g$. Here, by minimizing $L_{var}$, we choose a state which in addition to minimizing leakage and the chances of a conflict, minimizes the leakage range for that gate as well. The idea for opting for such a state is to reduce the circuit leakage and also minimize any variations in the expected overall circuit leakage.

Similar to the above biasing approach, in order to bias our selection towards assignments with lower leakage and lower distance, we increment the range by a fixed value $\rho$, in the computation of $L_{var}$.

### *3.4 Accepting Leakage States and Final IVC Determination*

The state selected from the previous step is now implied throughout the transitive fanout (TFO) of the chosen gate. The resulting values are referred to as *temporary* values. The distance of the resulting implications are now checked against a margin value $m_i$. If any distance is greater than $m_i$, then the assignment to gate $G$ is discarded. Initially, $m_i$ is set to a small value, and with increasing iteration $i$, it is relaxed. This is in an attempt to get closer to a global minima, by a more careful selection of states in early iterations. We perform $k = 3$ iterations in our experiments.

Once the new implications are computed, the implied nodes' probabilities are adjusted to reflect the freshly computed implications. If a node is set to a logic 1, then its probability is set to (1-$\alpha$), while a node which is set to logic 0 has its probability updated to $\alpha$.

For every $p$ iterations (or if all primary inputs have been assigned or implied), we test if the *temporary* values are satisfiable (this test is done by invoking the BerkMin [37] satisfiability solver). If so, then all *temporary* values are designated as new *final* values, never to be modified in the future. If the *temporary* values are satisfiable, and all inputs are assigned, then the algorithm exits. If the *temporary* values could not be satisfied, then we roll back the *temporary* values, by copying the last set of *final* values into the set of *temporary* values. For up to the next $p$ iterations, we call the satisfiability solver after each new state assignment. This is in an attempt to locate which of the last $p$ assignments caused the unsatisfiability condition to occur. Once this state is identified, we again revert to calling the satisfiability solver after every $p$ state assignments. If the satisfiability solver returns an unsatisfiable condition for a certain state $s$ assigned at a particular gate $g$, then we never try assigning $s$ to $g$ again. An example explaining the invocation to a Boolean satisfiability solver is explained next.

**Invoking a Boolean Satisfiability Solver:** A combinational circuit can be represented in a Conjunctive Normal Form (CNF) which is the input format for most SAT solvers including BerkMin [37]. In our work, we invoke the SAT solvers in every few iteration to check the compatibility of intermediate assignments. This is done by augmenting the existing CNF for the combinational circuit with the clauses which represent the intermediate assignments.

For instance, a two input AND gate with inputs A and B and output C such that C = A.B, in CNF consists of the following three clauses:
$(C + A^{'} + B')$
$(C^{'} + A)$

$(C' + B)$
Suppose our intermediate assignments on the different variables are:
A=1, B=0 and C=0.
Now, to check the consistency of these assignments with the circuit we add the following clauses to the original CNF formula:
$(A)$
$(B')$
$(C')$
The resulting six clauses together are passed as the input to a SAT solver. Since the result is 'Satisfiable' we know that our intermediate assignments are logically consistent with the circuit. On the other hand if our intermediate assignments were:
A=1, B=1 and C=0,
then we add the following clauses to the original CNF formula for the AND gate circuit:
$(A)$
$(B)$
$(C')$
The resulting six clauses would return an 'Unsatisfiable' result from the SAT solver and hence we would know that the intermediate assignment is inconsistent with the original circuit.

We use a SAT solver instead of other possible options for detecting logical consistencies of the intermediate assignments due to the fact that generating the original CNF formula for the circuit needs to be done only once, we need only augment the golden (consistent) values in future calls. Also, Boolean satisfiability is a well studied problem with highly efficient solvers such as BerkMin [37] easily available in the public domain.

## 4 Experimental Results

This section discusses two different sets of experiments. One set compares MLVC with other existing IVC determination techniques in terms of accuracy and run times. The second set compares and discusses the mean and standard deviation of circuit leakage values computed by applying MLVC, MLVC-VAR and a random vector based approach explained in the following subsection. All leakage values reported are in nA.

### *4.1 Selecting Parameter Values for MLVC and MLVC-VAR*

For the results presented in this paper, we experimented with numerous combinations of the many parameters listed in Table 3. Against each parameter, the set of values considered during these experiments has also been listed.

Table 3
Parameters' values considered in Experiments for MLVC and MLVC-VAR

| Parameter | Values | | | | |
|---|---|---|---|---|---|
| $m_1$ | 0.4 | 0.5 | 0.6 | 0.7 | |
| $m_2$ | 0.92 | 0.94 | 0.96 | 0.98 | |
| $m_3$ | 1 | | | | |
| $\beta$ | 0.1 | 0 | 2 | 5 | 10 |
| $\gamma$ | 10 | 20 | 50 | 70 | 100 |
| $\rho$ | 0.2 | 0.5 | 1 | 2 | |
| $P_{threshold}$ | 0.9 | 0.95 | | | |
| $\alpha$ | 0.9 | 0.95 | | | |

We define a *method* as an assignment of values to each parameter within a set of parameters. The details of our experimentation for determining the parameter values chosen (or methods used) are explained next.

We choose the values of $m_1$ to be lower than $m_2$, so that we are more selective about states in the early iterations. $m_3$ is 1 since we accept all states in the final iteration. Values of $\beta$, $\gamma$ and $\rho$ are selected based on the scale of $d_j$, $l_j$ and $r_j$. They are chosen such that a large value of $\beta$ erases the effect of $d_j$ on $L$ or $L_{var}$, and a large value of $\gamma$ erases the effect of $l_j$ on $L$ or $L_{var}$. A large value of $\rho$ erases the effect of $r_j$ on $L_{var}$. The various values of $\beta$, $\gamma$ and $\rho$ used are chosen such that our experiments explore the continuum along these three dimensions. $P_{threshold}$ and $\alpha$ need to be values close to 1 but not exactly 1, so we chose them to be 0.9 and 0.95.

For the MLVC approach, the parameters that can be varied are $m_1$, $m_2$, $m_3$, $\beta$, $\gamma$, $P_{threshold}$ and $\alpha$. Therefore, the total number of methods can be 1600. We ran $\sim$19 benchmark circuits using these methods. The three methods which, among them, provided the best results for the maximum number of benchmark circuits (85%), were chosen for the rest of the MLVC experiments. These methods are called M1, M2 and M3, and their parameters assignments are listed in Table 4. Similarly, for the MLVC-VAR approach, the parameters that can be varied are $m_1$, $m_2$, $m_3$, $\beta$, $\gamma$, $\rho$, $P_{threshold}$ and $\alpha$. Therefore, the total number of methods we have are 6400. Again, we ran $\sim$19 benchmark circuits using these methods. The three methods which, among them, provided the best results for the maximum number of benchmark circuits, were chosen for the rest of the MLVC-VAR experiments. These methods are called M1-Var, M2-Var and M3-Var, and their parameters assignments are listed in Table 8.

### *4.2 Comparing MLVC with Existing Techniques*

We performed extensive experiments to validate MLVC and compare its results to the exact or near-exact minimum circuit leakage values. We created the leakage table for all gates in our library, i.e. computed the nominal leakage value for all input vectors, for all gates, using SPICE [38] with a 100 nm BPTM model card, at 30 degree Celsius temperature. All our experiments were run on a 3.0 GHz Pentium 4 Linux machine with 1.0 GB RAM.

In all our experiments, we utilized a value of $k = 3$ iterations. The 3 methods (M1, M2 and M3) that we utilized for our experiments are described in Table 4. The value of $p$ used was 1, but it can be increased for less accurate but faster invocations of the algorithm. The values reported in Table 4 were determined after extensive experimentation with many circuits as described in Section 4.1.

Table 4
Parameters used in our Experiments for MLVC

| Method | $m_1$ | $m_2$ | $m_3$ | $\beta$ | $\gamma$ | $P_{threshold}$ | $\alpha$ |
|---|---|---|---|---|---|---|---|
| M1 | 0.6 | 0.96 | 1 | 1 | 10 | 0.95 | 0.95 |
| M2 | 0.6 | 0.96 | 1 | 0.1 | 100 | 0.95 | 0.95 |
| M3 | 0.4 | 0.96 | 1 | 5 | 10 | 0.9 | 0.9 |

Methods M1 and M2 utilize a value of 0.6 for $m_1$. As a consequence, we expect to set more gates to platinum values in the first iteration. These methods are designed to reduce the number of gates discarded due to margin violations. Among these methods, M1 has a higher $\gamma$ value, and therefore biases the state selection towards states which have smaller distance. On the other hand, M2 has a higher $\beta$ value, and as a consequence, state selection favors states with lower leakage. Method M3 has a smaller $m_1$ value, and therefore tends to reject gates due to margin violations. It is biased towards state selections which have smaller distances. Our method exhibits very low runtimes. Given that the run-times are very small, we can afford to apply all three methods (M1, M2 and M3), and choose the best result among the three. In general, we may try several methods and select one that yields the vector with the smallest leakage. In all experiments in this paper, we run each example using all available methods and then choose the best result.

In general the parameter sets need to be computed if the process technology is changed. This is done exactly once, hence it is a tractable task.

Using these three methods, we first compared the results of MLVC with the exact minimum circuit leakages. This was performed for small examples, and results are reported in Table 5. The minimum leakage value returned by MLVC (Column 4), along with the exact maximum (Column 3) and minimum (Column 2) leakages are shown in this table. Further, we report a figure of merit

$R$ in Column 5.

$$R = \frac{MLVC\,min\,leakage - Exact\,min\,leakage}{Exact\,max\,leakage - Exact\,min\,leakage}$$

The values of the maximum and minimum leakages are computed based on an exhaustive simulation of the circuit. Ideally, $R$ should be 0. Runtimes for MLVC are reported in Column 8, while the method utilized is reported in Column 7.

Note that the figure of merit $R$ is a more rigorous metric for comparing the effectiveness of any IVC determination technique. In the prior approaches to the IVC determination problem, the figure of merit utilized was

$$R_{std} = \frac{Heuristic\,min\,leakage - Exact\,min\,leakage}{Exact\,min\,leakage}$$

Based on Table 5, the average value of $R$ for MLVC was 0.13. For MLVC, the average value of the previously utilized figure of merit is 0.06.

Table 5 shows that the runtimes for MLVC are very small, with a good figure of merit for the method. The runtimes reported here are on average 10X faster then those reported in [32]. The reason for this improvement is the modification in the approach while choosing the best state for a selected gate. In the current form of MLVC, we use a SAT-solver to test if a particular state $s$ can be applied to a gate $G$ without requiring to unroll the assignments from previous passes. Only if the state $s$ clears this test do we consider it as a candidate state and then proceed to find the implications of assigning $s$ on the other gates in the circuit, and compute $L$. In the published approach [32], implications on the other gates in the circuit (due to assigning $s$) were generated without first testing for satisfiability on $s$. For certain test cases, the approach in [32] generated its implications and then detected unsatisfiability. This led to the additional runtime.

We also tested MLVC on larger circuits. The results of this experiment are shown in Table 6. Columns 2,3 and 4 list the number of gates, number of inputs and number of outputs respectively, for each circuit in Column 1. Columns 5 through 11 in this table are similar to Columns 2 through 8 in Table 5, with the exception that exact leakage values are not computed in this table. Instead, the minimum and maximum leakage found over 10,000 random vectors is shown in Table 6. According to [11], this statistically yields a greater than 99% confidence that we will obtain a leakage vector which is 0.5% from the minimum. *This is referred to as the Random Vectors Approach (RVA).*

Table 6 shows that MLVC produces minimum leakage vectors with very low errors, with extremely small runtimes. From [14], for the previously reported methods of [14], [39] and [18], the average errors were respectively 5.3%, 3.7%

and 10.4% (using the $R_{std}$[4] metric, for which MLVC results in an error of 3.5%). Further, the runtimes for MLVC are significantly smaller than those of [39], which is the most accurate known method for IVC determination.

Table 5
Exhaustive and Estimated Leakages for Small Circuits

| Circuit | Low | High | MLVC Low | $R$ | $R_{std}$ | Meth. | Time (s) |
|---|---|---|---|---|---|---|---|
| decod | 78.29 | 122.67 | 78.29 | 0.00 | 0.00 | M1 | 0 |
| cm82a | 115.20 | 133.00 | 115.20 | 0.00 | 0.00 | M3 | 0.02 |
| cm42a | 106.64 | 141.87 | 115.38 | 0.25 | 0.08 | M1 | 0.04 |
| cm152a | 80.10 | 124.64 | 84.35 | 0.10 | 0.05 | M1 | 0.02 |
| cm151a | 93.48 | 141.83 | 103.08 | 0.20 | 0.10 | M2 | 0.01 |
| cm138a | 98.19 | 136.22 | 98.19 | 0.00 | 0.00 | M1 | 0.01 |
| C17 | 19.76 | 37.99 | 20.07 | 0.02 | 0.02 | M1 | 0.01 |
| majority | 36.69 | 57.40 | 40.51 | 0.18 | 0.10 | M1 | 0 |
| cm85a | 183.23 | 271.51 | 221.16 | 0.43 | 0.21 | M1 | 0.05 |
| AVG | | | | 0.131 | 0.062 | | |

### *4.3 Comparing MLVC-VAR with MLVC and RVA*

Since, to the best of the authors' knowledge, there is no other work to date which considers PVT variations in the determination of IVC, we compare the performance of MLVC-VAR with MLVC and RVA. We compare the mean, $\mu$, and standard deviation, $\sigma$, of the circuit leakage values computed by applying the input vectors determined by MLVC-VAR, MLVC and RVA.

For use in the MLVC-VAR approach, we created an extended leakage table (for each gate) that contains the variations in leakage values due to PVT variations. For generating this table, we ran Monte Carlo (MC) simulations in SPICE, using the random PVT variations reported in [40], for 30000 samples. These variations were assumed to be random (uncorrelated). Hence the $r_i$ (leakage range) values of a gate $g$ were fixed. If detailed spatial information of gates was available, the correlation of these variables could be determined, resulting in different $r_i$ values for different instances of any gate $g$. This can be done as follows:

Under intra-die variations, the value of any parameter $p$ located at ($x$,$y$) can be modeled as [28,41]:
$p = p_n + x \cdot (S_x) + y \cdot (S_y) + e$

where $p_n$ is the nominal design parameter value at die location (0,0), and $S_x$ and $S_y$ are gradients of the parameter indicating the spatial variations of pa-

[4] Although the $R$ metric is more rigorous, our comparisons to existing approaches utilize the $R_{std}$ metric since these approaches utilize the $R_{std}$ metric.

Table 6
Leakages for Large Circuits

| Circuit | N. Gts | N. Inps. | N. Outs. | Low | High | MLVC Low | $R$ | $R_{std}$ | Meth. | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| tcon | 41 | 17 | 16 | 174.82 | 211.84 | 173.10 | -0.05 | -0.01 | M1 | 0.05 |
| cm163a | 50 | 16 | 5 | 154.30 | 245.48 | 167.95 | 0.15 | 0.09 | M1 | 0.04 |
| pm1 | 52 | 16 | 13 | 191.90 | 269.98 | 208.20 | 0.21 | 0.08 | M1 | 0.04 |
| cm162a | 56 | 14 | 5 | 186.46 | 264.69 | 204.01 | 0.22 | 0.09 | M3 | 0.07 |
| cm150a | 58 | 21 | 1 | 203.63 | 340.77 | 245.81 | 0.31 | 0.21 | M1 | 0.08 |
| cu | 62 | 14 | 11 | 205.01 | 306.62 | 214.63 | 0.09 | 0.05 | M3 | 0.07 |
| cc | 74 | 21 | 20 | 269.61 | 354.98 | 295.70 | 0.31 | 0.10 | M1 | 0.05 |
| parity | 75 | 16 | 1 | 276.78 | 363.04 | 278.50 | 0.02 | 0.01 | M3 | 0.09 |
| pcle | 78 | 19 | 9 | 261.60 | 376.85 | 269.75 | 0.07 | 0.03 | M1 | 0.05 |
| pcler8 | 102 | 27 | 17 | 385.74 | 507.22 | 401.69 | 0.13 | 0.04 | M3 | 0.09 |
| lal | 109 | 26 | 19 | 399.16 | 534.47 | 416.90 | 0.13 | 0.04 | M2 | 0.22 |
| b9 | 119 | 41 | 21 | 398.49 | 600.45 | 403.94 | 0.03 | 0.01 | M3 | 0.2 |
| unreg | 120 | 36 | 16 | 440.20 | 538.22 | 452.32 | 0.12 | 0.03 | M1 | 0.17 |
| comp | 131 | 32 | 3 | 454.01 | 613.59 | 486.08 | 0.20 | 0.07 | M1 | 0.24 |
| count | 132 | 35 | 16 | 491.94 | 655.46 | 530.06 | 0.23 | 0.08 | M1 | 0.27 |
| c8 | 138 | 28 | 18 | 532.14 | 652.65 | 535.09 | 0.02 | 0.01 | M2 | 0.2 |
| cht | 198 | 47 | 36 | 772.27 | 965.94 | 753.75 | -0.10 | -0.02 | M3 | 0.57 |
| ttt2 | 213 | 24 | 21 | 809.02 | 983.85 | 821.68 | 0.07 | 0.02 | M3 | 0.71 |
| C432 | 237 | 36 | 7 | 874.29 | 1110.63 | 929.66 | 0.23 | 0.06 | M3 | 0.95 |
| i5 | 198 | 133 | 66 | 858.95 | 945.20 | 875.81 | 0.20 | 0.02 | M1 | 0.61 |
| i3 | 258 | 132 | 6 | 1135.17 | 1327.06 | 1127.74 | -0.04 | -0.01 | M1 | 1.16 |
| x1 | 305 | 51 | 35 | 1148.05 | 1384.54 | 1180.63 | 0.14 | 0.03 | M3 | 1.47 |
| example2 | 330 | 85 | 66 | 1193.91 | 1472.34 | 1170.95 | -0.08 | -0.02 | M1 | 1.95 |
| x4 | 455 | 94 | 71 | 1705.54 | 2096.10 | 1724.77 | 0.05 | 0.01 | M3 | 3.72 |
| C1908 | 565 | 33 | 25 | 2115.12 | 2345.72 | 2181.09 | 0.29 | 0.03 | M1 | 4.28 |
| C499 | 582 | 41 | 32 | 2092.65 | 2249.22 | 2106.21 | 0.09 | 0.01 | M1 | 4.2 |
| rot | 711 | 135 | 107 | 2805.18 | 3145.62 | 2890.93 | 0.25 | 0.03 | M3 | 9.74 |
| apex6 | 794 | 135 | 99 | 2930.22 | 3407.96 | 2977.63 | 0.10 | 0.02 | M1 | 13.51 |
| x3 | 908 | 135 | 99 | 3377.40 | 3744.96 | 3370.34 | -0.02 | 0.00 | M3 | 16.53 |
| C3540 | 1354 | 50 | 22 | 5179.71 | 5757.48 | 5236.28 | 0.10 | 0.01 | M3 | 41.03 |
| C5315 | 1963 | 178 | 123 | 7982.04 | 8569.38 | 8027.37 | 0.08 | 0.01 | M3 | 131.9 |
| C6288 | 3734 | 32 | 32 | 14416.17 | 16000.10 | 14733.79 | 0.20 | 0.02 | M3 | 540.07 |
| C7552 | 2729 | 207 | 108 | 10989.43 | 11586.96 | 11087.33 | 0.16 | 0.01 | M2 | 254.74 |
| AVG | | | | | | | 0.119 | 0.035 | | |

rameter along the $x$ and $y$ directions, respectively. The term $e$ stands for the random intra-chip variation, and the vector of all random components across the chip has a correlated multivariate normal distribution due to spatial correlations in the intra-chip variation. This vector depends on the correlation matrix of the spatially correlated parameters. Effectively, for each type of parameter, a correlation matrix of size $n \times n$, where $n$ is the number of grid regions, represents the spatial correlation. This matrix could be determined

from data extracted from manufactured wafers or derived from the spatial correlation models such as the one presented in [42]. Further, the correlation between different types of parameters can be added in this correlation matrix. This can be done by decomposing the correlated parameters into an uncorrelated set using an orthogonal transformation such as the principal component analysis (PCA) technique, or by constructing a covariance matrix for all correlated parameters.

By using this model in the generation of the extended leakage table (the table which tabulates the nominal, mean and standard deviation for every input, for each instance of any gate $g$), our approach can account for spatial correlation of every parameter. The steps of our approach, namely the selection of the best candidate gate and the selection of the best state for that gate, are decided using the data in the new extended leakage table. Note that our current implementation does not account for spatial correlations or correlation between different types of parameters. The above discussion, however, explains the methodology in order to account for these correlations. Implementing this methodology is a possible future work.

The mean and standard deviation of the PVT variables are listed in Table 7. We generated the $\mu$ and $\sigma$ for the leakage values for all states for all gates in the library using the variations shown in Table 7.

Table 7
Parameter Variations

| Parameter | $\mu$ | $\sigma$ |
|---|---|---|
| Channel length | 0.1$\mu$m | 0.05$\mu$m |
| Power Supply | 1.2 V | 0.04 V |
| Threshold Voltage PMOS | 0.3030 V | 0.0127 V |
| Threshold Voltage NMOS | 0.2607 V | 0.0110 V |
| Temperature | 30° C | 1° C |

In the experiments in this subsection, the parameter values used for MLVC were identical to those in Table 4. For MLVC-VAR, the parameters used are listed in Table 8.

Table 8
Parameters used in our Experiments for MLVC-VAR

| Method | $m_1$ | $m_2$ | $m_3$ | $\beta$ | $\gamma$ | $\rho$ | $P_{threshold}$ | $\alpha$ |
|---|---|---|---|---|---|---|---|---|
| M1-Var | 0.6 | 0.96 | 1 | 5 | 10 | 2 | 0.95 | 0.95 |
| M2-Var | 0.6 | 0.96 | 1 | 5 | 10 | 0.2 | 0.95 | 0.95 |
| M3-Var | 0.4 | 0.96 | 1 | 1 | 70 | 1 | 0.90 | 0.9 |

Again, these parameters were chosen after extensive experimentation on several circuits. The margins $m_1$, $m_2$ and $m_3$ and parameters $P_{threshold}$ and $\alpha$ chosen are identical to those described in Table 4. Among the three methods M1-Var, M2-Var and M3-Var, M2-Var has the lowest value of $\rho$ and therefore

biases the state selection towards states with lower range. M3-Var favors states with lower leakage (since it has the lowest value of $\gamma$) whereas M1-Var favors states with lower distance, in comparison with M2-Var and M3-Var.

Table 9 compares the MLVC-VAR with MLVC and RVA. The $\mu$ and $\sigma$ of the circuit leakage values are computed with similar Monte Carlo experiments as described previously for generating the extended leakage table. We use the same set of circuits as those used in Table 6. These are listed in Column 1. Column 2 reports the method used for MLVC-VAR. Note that the method used for MLVC is as reported in Table 6 for these circuits.

Columns 3 and 4 report $\mu$ and $\sigma$ of the circuit leakage values computed by applying MLVC-VAR. The time taken for generating the input vector using MLVC-VAR is reported in Column 5. These run times, as expected, are about equal to those reported for MLVC in Table 6. Note that MC simulations are performed one time, upfront for each gate. Hence runtimes for MC simulations are not added in Column 5. Column 6, titled *diff* shows a '$\surd$' for circuits in which MLVC-VAR yielded a different input vector with respect to MLVC. A '$\times$' on the other hand, represents that both MLVC-VAR and MLVC returned the same vector. On average, for about 85% of benchmarks, MLVC-VAR returns a different best case input vector as compared to MLVC. This reiterates the notion that MLVC alone might possibly yield a vector for which the worst case ($\mu$+6$\sigma$) leakage of the combinational circuit is higher than what MLVC-VAR would compute. The leakage values for MLVC-VAR and MLVC are slightly different for the '$\times$' circuits, since unassigned inputs are randomly set before Monte Carlo simulations. The next three columns report the percentage improvement of MLVC-VAR over MLVC. The percentage improvement (decrease) in $\mu$ is shown in Column 7, the percentage improvement (decrease) in $\mu \cdot \sigma$ is shown in Column 8 and the percentage improvement (decrease) in $\mu$+6$\sigma$ is shown in Column 9. On average these improvements are 5.98%, 9.69% and 5.37% respectively.

Similarly, Columns 10, 11 and 12 report the percentage improvement of MLVC-VAR over RVA. The percentage improvement in $\mu$, $\mu \cdot \sigma$ and $\mu$+6$\sigma$ over RVA is 2.07%, 5.98% and 3.08% respectively.

It is important to note that PVT variations can in general result in large leakage variations. However, since we are considering leakage variations during the sleep mode of operation, our temperature variation is considered to have a $\sigma$ of 1°C. This results in lowered leakage variations as can be observed in Table 9, than one would intuitively guess.

Table 9
Comparing MLVC-VAR, MLVC and RVA

| Circuit | MLVC-VAR | | | | | % Improv. w.r.t. MLVC | | | % Improv w.r.t. RVA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Method | $\mu$ | $\sigma$ | Time (s) | *diff* | $\mu$ | $\mu \cdot \sigma$ | $\mu + 6\sigma$ | $\mu$ | $\mu \cdot \sigma$ | $\mu + 6\sigma$ |
| tcon | M2-Var | 270.77 | 67.33 | 0.09 | × | 0.72 | 2.87 | 1.59 | 1.38 | 3.19 | 1.65 |
| cm163a | M2-Var | 243.36 | 55.01 | 0.04 | √ | -1.63 | -16.76 | -8.86 | -3.50 | -4.89 | -2.25 |
| pm1 | M2-Var | 296.41 | 65.09 | 0.08 | √ | 7.24 | 13.13 | 6.74 | 0.82 | 3.46 | 1.88 |
| cm162a | M1-Var | 290.61 | 59.13 | 0.12 | √ | 9.50 | 22.42 | 12.19 | -1.38 | 4.00 | 2.41 |
| cm150a | M1-Var | 268.29 | 50.92 | 0.05 | √ | 27.99 | 50.68 | 29.91 | 6.06 | 12.47 | 6.47 |
| cu | M2-Var | 306.70 | 58.74 | 0.12 | √ | 18.43 | 33.79 | 18.64 | 0.68 | 7.56 | 4.12 |
| cc | M1-Var | 423.63 | 79.37 | 0.17 | √ | 11.55 | 19.69 | 10.32 | 1.05 | 3.56 | 1.84 |
| parity | M3-Var | 403.71 | 67.47 | 0.18 | √ | 3.67 | 13.60 | 7.11 | 3.19 | 13.61 | 7.13 |
| pcle | M1-Var | 397.66 | 75.78 | 0.21 | √ | 11.94 | 18.01 | 9.32 | 2.62 | 4.53 | 2.27 |
| pcler8 | M3-Var | 606.54 | 105.54 | 0.31 | × | 3.92 | 4.89 | 2.46 | 2.92 | 6.01 | 3.05 |
| lal | M1-Var | 648.27 | 98.66 | 0.17 | √ | -5.41 | -17.43 | -8.19 | -5.95 | -12.80 | -6.19 |
| b9 | M3-Var | 598.68 | 91.92 | 0.4 | × | 2.62 | 5.37 | 2.72 | -1.38 | -5.65 | -2.72 |
| unreg | M2-Var | 645.86 | 82.79 | 0.35 | × | 4.22 | 8.50 | 4.33 | 6.97 | 28.40 | 14.71 |
| comp | M2-Var | 759.37 | 97.35 | 0.26 | √ | -2.09 | -3.63 | -1.83 | -10.88 | -10.87 | -5.87 |
| count | M3-Var | 751.68 | 97.04 | 0.45 | √ | 5.95 | 5.79 | 3.37 | -1.17 | 6.01 | 2.61 |
| c8 | M3-Var | 812.82 | 105.88 | 0.41 | √ | 3.03 | 1.86 | 1.21 | -2.05 | -2.69 | -1.42 |
| cht | M1-Var | 1165.31 | 134.85 | 0.98 | × | 1.45 | 4.28 | 2.04 | 2.00 | 0.40 | 0.54 |
| ttt2 | M1-Var | 1224.16 | 128.36 | 1.03 | √ | 8.97 | 19.44 | 9.96 | 0.99 | 7.93 | 3.40 |
| C432 | M1-Var | 1325.58 | 129.86 | 1.51 | √ | 2.81 | 4.53 | 2.43 | 1.82 | 10.31 | 4.46 |
| i5 | M2-Var | 1181.13 | 136.61 | 1.57 | √ | 10.14 | 10.98 | 6.59 | 10.10 | 14.35 | 7.97 |
| i3 | M3-Var | 1489.03 | 139.11 | 1.42 | √ | 8.50 | 8.67 | 5.68 | 13.76 | 25.37 | 13.66 |
| x1 | M3-Var | 1875.22 | 176.48 | 1.97 | √ | 3.44 | -2.47 | 0.20 | -5.59 | -10.60 | -5.28 |
| example2 | M2-Var | 1693.13 | 159.78 | 2.27 | √ | 7.22 | 13.27 | 6.97 | 8.06 | 12.22 | 6.81 |
| x4 | M1-Var | 2414.85 | 169.22 | 5.27 | √ | 7.67 | 15.69 | 7.97 | 8.38 | 21.22 | 10.12 |
| C1908 | M3-Var | 3257.70 | 216.46 | 6.16 | √ | 4.14 | 9.46 | 4.55 | 0.47 | 2.78 | 1.00 |
| C499 | M3-Var | 3225.41 | 206.44 | 6.07 | √ | 3.58 | 9.39 | 4.27 | 1.18 | 5.65 | 2.13 |
| rot | M1-Var | 4119.71 | 242.07 | 11.49 | √ | 6.11 | 8.39 | 5.18 | 4.97 | 9.39 | 4.89 |
| apex6 | M3-Var | 4126.95 | 236.48 | 15.26 | √ | 9.51 | 18.18 | 9.53 | 8.84 | 17.21 | 8.93 |
| x3 | M3-Var | 5002.58 | 255.71 | 22.05 | √ | 2.26 | 3.41 | 2.01 | 3.86 | 7.07 | 3.74 |
| C3540 | M3-Var | 7925.77 | 358.34 | 46.69 | √ | 7.51 | 9.47 | 6.41 | 0.54 | -1.26 | 0.05 |
| C5315 | M3-Var | 11598.26 | 415.87 | 134.95 | √ | 6.13 | 11.78 | 6.11 | 6.33 | 12.02 | 6.29 |
| C6288 | M3-Var | 22012.21 | 577.94 | 339.72 | √ | 3.99 | 6.49 | 3.81 | 1.69 | 2.79 | 1.61 |
| C7552 | M3-Var | 16544.98 | 484.45 | 272.22 | √ | 2.29 | 5.94 | 2.51 | 1.53 | 4.68 | 1.78 |
| Average of ratio | | | | | | 5.98 | 9.69 | 5.37 | 2.07 | 5.98 | 3.08 |

## 5 Conclusions

We have developed a probabilistic method, MLVC, to perform input vector assignment for leakage minimization in a combinational circuit. We start by computing signal probabilities throughout the circuit. These probabilities are

used to guide the selection of the next gate to assign. The selected gate is the one with the probabilistic highest leakage value and the largest leakage range due to process variations. Once this gate is selected, it is assigned a state, again in a manner which probabilistically minimizes its leakage. The implications induced by such a state selection are computed. A satisfiability solver is invoked, to validate the state selection before our algorithm commits to this assignment. The algorithm terminates when all inputs have been assigned or are implied.

The MLVC technique is fast, flexible and provides accurate results. On average, for small examples, MLVC found minimum leakage values which were 6.2% from the minimum circuit leakage. For larger examples, it was impractical to compute the minimum circuit leakage exactly. We computed our statistics on the basis of running 10,000 samples of circuit leakage computation. For these examples, MLVC produces leakage vectors with leakage within 3.5% from the minimum. The runtimes of MLVC are much lower than existing techniques which produce results of similar quality. Additionally, the effect of PVT variations can be easily incorporated into such a probabilistic formulation.

A variant of MLVC, termed MLVC-VAR, was also presented. MLVC-VAR takes into account the effect of variations in leakage values due to PVT variations. Including the effect of PVT variations for determining minimum leakage vector is important because of the strong dependence of leakage currents on power supply, threshold voltage and temperature. Further, MLVC-VAR can be modified to account for spatial correlation and correlation between different parameter types as described in Section 4.3. This modification is a possible future work. The comparison of the mean and standard deviations of the circuit leakages induced by the input vectors generated by MLVC-VAR, MLVC and RVA further proves the relevance of taking into account the PVT variations while determining IVC. On average, MLVC-VAR reports a 9.69% (5.37%) improvement in final circuit leakage over MLVC with respect to $\mu \cdot \sigma$ ($\mu$+6$\sigma$) with similar runtimes. The improvement over RVA is 5.98% (3.08%) with much lower runtimes.

## References


[1] BSIM3 Homepage, `http://www-device.eecs.berkeley.edu/~bsim3`.

[2] The International Technology Roadmap for Semiconductors, `http://public.itrs.net/` (2003).

[3] J. T. Kao, A. P. Chandrakasan, Dual-threshold voltage techniques for low-power digital circuits, IEEE Journal of Solid-State Circuits 35 (7) (2000) 1009–1018.

[4] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, J. Yamada, 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS, IEEE Journal of Solid-State Circuits 30 (8) (1995) 847–854.

[5] N. Jayakumar, S. Khatri, An ASIC design methodology with predictably low leakage, using leakage-immune standard cells, in: Proceedings of the International Symposium on Low Power Electronics and Design, 2003, pp. 128–133.

[6] H. Kawaguchi, K. Nose, T. Sakurai, A super cut-off CMOS (SCCMOS) scheme for 0.5-v supply voltage with picoampere stand-by current, IEEE Journal of Solid-State Circuits 35 (10) (2000) 1498–1501.

[7] F. Assaderaghi, D. Sinitsky, S. A. Parke, J. Bokor, P. K. Ko, C. Hu, Dynamic threshold-voltage MOSFET (DTMOS) for ultra-low voltage VLSI, IEEE Transactions on Electron Devices 44 (3) (1997) 414–422.

[8] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, C. Hu, New paradigm of predictive MOSFET and interconnect modeling for early circuit design, in: Proc. of IEEE Custom Integrated Circuit Conference, 2000, pp. 201–204, http://www-device.eecs.berkeley.edu/ ptm.

[9] H. Su, F. Liu, A. Devgan, E. Acar, S. Nassif, Full-chip leakage estimation considering power supply and temperature variations., in: Proceedings, International Symposim on Low Power Electronics and Design, 2003, pp. 78–83.

[10] S. Narendra, V. De, S. Borkar, D. Antoniadis, A. Chandrakasan, Full-chip sub-threshold leakage power prediction for sub-0.18$\mu$m cmos, in: Proceedings, International Symposim on Low Power Electronics and Design, 2002.

[11] J. Halter, F. Najm, A gate-level leakage power reduction method for ultra low power CMOS circuits, in: Proceedings of CICC, 1997, pp. 475–478.

[12] Z. Chen, M. Johnson, L. Wei, W. Roy, Estimation of standby leakage power in CMOS circuit considering accurate modeling of transistor stacks, in: International Symposium on Low Power Electronics and Design, 1998, pp. 239–244.

[13] M. Johnson, D. Somasekhar, K. Roy, Models and algorithms for bounds on leakage in CMOS circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 18 (6) (1999) 714–725.

[14] F. Gao, J. Hayes, Exact and heuristic approaches to input vector control for leakage power reduction, in: Proceedings, International Conference on Computer-aided Design, 2004, pp. 527–532.

[15] K. Chopra, S. Vrudhula, Implicit pseudo Boolean enumeration algorithms for input vector control, in: Proceedings, Design Automation Conference, San Diego, 2004, pp. 767–772.

[16] K. Gulati, N. Jayakumar, S. Khatri, An Algebraic Decision Diagram (ADD) based technique to find leakage histograms of combinational designs, in:

Proceedings, International Symposium on Low Power Electronic Design (ISLPED), San Diego, CA, 2005.

[17] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi, Algebraic decision diagrams and their applications, Formal Methods in Systems Design 10 (2/3) (1997) 171–206.

[18] R. Rao, F. Liu, J. Burns, R. Brown, A heuristic to determine low leakage sleep state vectors for CMOS combinational circuits, in: Proceedings, International Conference on Computer-aided Design, 2003, pp. 689–692.

[19] M. Abramovici, M. A. Breuer, A. D. Friedman, Digital Systems Testing and Testable Design, Computer Science Press, 1990.

[20] T. A. Unni, D. M. H. Walker, Model-based $i_{DDQ}$ pass/fail limit setting, in: IEEE International Workshop on $I_{DQQ}$ Testing, 1998, pp. 43–47.

[21] F. Aloul, S. Hassoun, K. Sakallah, D. Blauuw, Robust SAT-based search algorithm for leakage power reduction, in: Proceedings, Power and Timing Models and Simulation (PATMOS), 2002.

[22] A. Abdollahi, F. Fallah, P. Massoud, Runtime mechanisms for leakage current reduction in CMOS VLSI circuits, in: Proceedings of the 2002 International Symposium on Low Power Electronics and Design, 2002, pp. 213–218.

[23] A. Abdollahi, F. Fallah, P. Massoud, An effective power mode transition technique in MTCMOS circuits, in: Proceedings, IEEE Design Automation Conference, 2005, pp. 13–17.

[24] L. Yuan, G. Qu, Enhanced leakage reduction technique by gate replacement, in: Proceedings, IEEE Design Automation Conference, 2005, pp. 47–50.

[25] L. Cheng, L. Deng, D. Chen, M. D. F. Wong, A fast simultaneous input vector generation and gate replacement algorithm for leakage power reduction, in: Proceedings, IEEE Design Automation Conference, 2006, pp. 117–120.

[26] N. Jayakumar, S. P. Khatri, An algortihm to minimize leakage through simultaneous input vector control and circuit modification, in: Proceedings, Design Automation and Test in Europe Conference, 2007.

[27] S. Zhang, V. Wason, K. Banerjee, A probabilistic framework to estimate full-chips subthrehold leakage power distribution considering within-die and die-to-die P-T-V variations, in: Proceedings, International Symposim on Low Power Electronics and Design, 2004, pp. 156–161.

[28] H. Chang, S. S. Sapatnekar, Full-chip analysis of leakage power under process variations, including spatial correlations, in: DAC '05: Proceedings of the 42nd annual conference on Design automation, ACM Press, New York, NY, USA, 2005, pp. 523–528.

[29] R. Rao, A. Srivastava, D. Blaauw, D. Sylvester, Statistical estimation leakage currents considering inter-and intra-die process variations, in: Proceedings, International Symposim on Low Power Electronics and Design, 2003, pp. 84–89.

[30] X. Li, J. Le, L. T. Pileggi, Projection-based statistical analysis of full-chip leakage power with non-log-normal distributions, in: Proceedings, 43rd Design Automation Conference, 2006, pp. 103–108.

[31] S. Bhardwaj, S. B. K. Vrudhula, Leakage minimization of nano-scale circuits in the presence os systematic and random variations, in: Proceedings, 42nd Design Automation Conference, 2005, pp. 541–546.

[32] K. Gulati, N. Jayakumar, S. P. Khatri, A probabilistic method to determine the minimum leakage vector for combinational designs, in: Proceedings, IEEE International Symposium on Circuits and Systems (ISCAS), Kos, Greece, 2006.

[33] A. Agarwal, K. Kang, K. Roy, Accurate estimation and modeling of total chip leakage considering inter- & intra-die process variations, in: ICCAD '05: Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design, IEEE Computer Society, Washington, DC, USA, 2005, pp. 736–741.

[34] R. Rao, A. Srivastava, D. Blaauw, D. Sylvester, Statistical estimation of leakage current considering inter- and intra-die process variation, in: ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design, ACM Press, New York, NY, USA, 2003, pp. 84–89.

[35] S. Cook, The complexity of theorem-proving procedures, in: Proceedings, Third ACM Symp. Theory of Computing, 1971, pp. 151–158.

[36] N. S. Saluja, S. P. Khatri, Efficient SAT-based combinational ATPG using multi-level don't-cares, in: Proceedings, IEEE International Test Conference, 2005.

[37] E. Goldberg, Y. Novikov, BerkMin: A fast and robust SAT-solver, in: Proceedings, Design Automation and Test in Europe (DATE) Conference, 2002, pp. 142–149.

[38] L. Nagel, Spice: A computer program to simulate computer circuits, in: University of California, Berkeley UCB/ERL Memo M520, 1995.

[39] S. Naidu, E. Jacobs, Minimizing stand-by leakage power in static CMOS circuits, in: Proceedings, Design Automation and Test in Europe (DATE) Conference, 2001, pp. 370–376.

[40] Y. Cao, C. Hu, A. B. Kahng, D. Sylvester, Improved estimates of process variation impact on deep submicron circuit performance, in: Unpublished, 2006.

[41] H. Chang, S. S. Sapatnekar, Statistical timing analysis under spatial correlations, in: Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, Vol. 24, 2005, pp. 1467– 1482.

[42] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, R. Panda, Statistical delay computation considering spatial correlations, in: ASPDAC: Proceedings of the 2003 conference on Asia South Pacific design automation, ACM Press, New York, NY, USA, 2003, pp. 271–276.