Minimum Leakage Vector Computation using Weighted Partial MaxSAT

Amrinder Singh[†] Kanupriya Gulati[*] Sunil P Khatri[†]
[†] Department of Electrical & Computer Engineering, Texas A&M University, College Station TX 77843.
[*] Strategic CAD Laboratories, Intel Corporation, Hillsboro OR 97124

Abstract— **Aggressive scaling of CMOS technology has enabled faster and smaller designs but has posed new challenges. In the deep-submicron era, leakage power has become a major contributor to the overall power dissipation of an IC. In this paper, we present a weighted partial Max-SAT (WPMax-SAT) based approach to find the minimum leakage vector (MLV) of a combinational design. In its exact form, this technique computes the input vector which gives the lowest leakage for a combinational design. For large designs, the exact WPMax-SAT based technique may require large runtimes. Therefore, for such designs, the exact technique is run for a fixed amount of time followed by a guided random search around the best leakage vector computed by the WPMax-SAT solver. We also present a variant of our approach in which the MLV is generated by including the effect of random variations in leakage due to variations in process, voltage and temperature (PVT). Experimental results on ISCAS85 and MCNC91 benchmark circuits show that for larger circuits on average, our method reports a 3.62% improvement in mean, 4.20% improvement in standard deviation and 3.67% improvement in $\mu + 3*\sigma$ leakage of the circuit under PVT variations, compared to a random vector based MLV determination approach (with the *same* runtime as the random vector based approach).**

I. Introduction

In deep-submicron designs, power dissipation is becoming a major concern for chip designers. Along with high performance, low power has emerged as a first order design goal. Static power consumption is caused by the leakage currents when there is no switching activity in the circuit, and constitutes a large fraction of the total power consumption of a design. Leakage currents can be further classified into subthreshold leakage, reverse-biased junction leakage, gate induced drain leakage and gate direct-tunneling leakage [1]. In this paper, we concentrate on reducing the subthreshold leakage of a combinational design in standby mode, although our approach can be extended to model the other forms of leakage as well. The expression for the subthreshold leakage current is given by Equation 1, where I_0 and V_{off} are constants, V_T is the threshold voltage, v_t is the thermal voltage (26mV at 300 K) and n is the subthreshold slope factor. Note that the subthreshold leakage current increases exponentially with a decrease in V_T.

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

Dynamic power is proportional to the square of supply voltage, therefore lower supply voltages have been used in recent process generations. However, in order to maintain the performance of the circuit, the threshold voltage (V_T) is also scaled proportionally. Due to this, in deep-submicron devices, the contribution of the leakage power to the total power is significant [2]. This is a matter of concern for portable devices which spend most of their time in the standby mode. Thus, it is crucial to reduce the leakage power in order to extend battery life. In this paper, we focus on subthreshold leakage reduction in combinational designs using input vector control.

A low leakage design technique which involves minimal design overhead or change in design methodology is *input vector control* (IVC). In this approach, the design is "parked" in its lowest leakage state during standby mode [3] [4] [5] [6] [7] [8]. If the design is scan enabled, the minimum leakage vector (MLV) can be scanned in using scan chains, else the inputs of the circuit can be forced to the MLV using MUXes (with the standby signal used as the MUX select line). The motivation behind this technique is that the subthreshold leakage current of a logic gate (combinational design) in the standby mode heavily depends upon the state of it's inputs (primary inputs). Although it may not be possible to have all the gates of a circuit in their lowest leakage state (due to the connectivity of these gates within the circuit), we can apply an input vector which minimizes the total leakage of the combinational design. However, the problem of finding the minimum leakage vector (MLV) is NP-complete [9]. In this paper, we propose a weighted partial Max-SAT based algorithm for the MLV computation for combinational circuits. Additionally, the effect of random variations in leakage due to PVT variations has been incorporated in our formulation.

In the deep-submicron era, the effects of PVT variations have become significant. Since subthreshold leakage has an exponential dependence on temperature, threshold voltage and power supply, PVT variations heavily influence the circuit leakage and correspondingly the MLV of the circuit. In [10], the authors highlighted the importance of considering within-die variations of channel length and threshold voltage for accurate subthreshold leakage current estimation. They showed that subthreshold leakage power can be overestimated or underestimated by 1.5×-6.5× if these within-die variations are ignored. To account for the significant dependence of circuit leakage on PVT variations, our approach *incorporates the effect of random PVT variations while computing MLV of a circuit*.

It can be conjectured that considering PVT variations just leads to an increased mean value, but the best leakage state remains unaltered. According to this reasoning, finding the MLV without PVT variations would lead to the same result. In fact, this conjecture has been proven false [11]. Thus, finding the MLV with nominal leakage as the cost function might not be the best in the presence of PVT variations, since this might result in a higher worst case ($\mu + 3*\sigma$) leakage of the combinational circuit.

Boolean Satisfiability (SAT) is the problem of finding a satisfying assignment of the variables of a Boolean formula expressed in Conjunctive Normal Form (CNF), if such an assignment exists. There can be multiple satisfying assignments which satisfy a given Boolean formula, and SAT returns one such assignment. SAT based approaches are widely used in VLSI design for equivalence checking, software-verification, timing analysis and model checking [12]. A Boolean formula expressed in Conjunctive Normal Form is a conjunction (AND) of various clauses. Each clause consists of literals (a variable or its complement are both referred to as literals), which are ORed together.

A variant of SAT is Max-SAT [13], where the objective is to find an assignment which *maximally* satisfies the set of clauses of a given CNF. If the clauses are assigned individual *weights*, then finding the assignment which maximizes the sum of the weights of the clauses that are satisfied is called weighted Max-SAT [14]. Yet another variant of SAT is weighted *partial* Max-SAT (WPMax-SAT) [15] in which the set of clauses are divided into hard clauses and soft clauses. All the hard clauses must be satisfied. The soft clauses need to be satisfied such that the sum of the weights of the satisfied soft clauses is maximized. In this paper, we use WPMax-SAT to compute MLV for a combinational design.

The key contributions of this paper are:

- We propose a novel WPMax-SAT based algorithm for exact MLV computation.
- For larger circuits, we further propose a post processing technique which performs a guided random search around the best leakage vector computed by WPMax-SAT, in order to obtain the MLV.
- We also incorporate the effect of leakage variations due to PVT variations, while computing MLV.
- Our results show that in comparison with a random vector [3] based method (using 10K random vectors), for larger circuits, our approach provides an average improvement of 3.62% in the mean circuit leakage, 4.20% in the standard deviation of the circuit leakage and 3.67% in the $\mu + 3*\sigma$ leakage of the circuit with *identical runtimes* as the random vector based approach.

The remainder of the paper is organized as follows. Section II discusses previous work in the field of determining the minimum leakage vector. In Section III, we discuss the motivation behind computing PVT aware MLV for a circuit. In Section IV, we describe our approach of MLV computation. In Section V, we present our experimental results on ISCAS85 and MCNC91 benchmark circuits, while conclusions are drawn in Section VI.

II. Previous Work

In the past, various techniques have been proposed for MLV computation. It was reported [3] that a random search over 10,000 vectors gives 99% confidence that less than 0.5% of the vectors would have a leakage lower than the minimum leakage value obtained from random search. The major shortcoming of this technique is that for a higher degree of confidence, a large number of random vectors are required, which leads to higher runtime. In [5], the authors propose a greedy heuristic to guide the search. However, for small circuits (< 20 gates) the leakage for the computed MLV is worse than that of a random vector based technique [3].

The authors of [16] model the leakage of logic gates using pseudo Boolean functions, which are linearized and used to formulate an integer linear programming (ILP) based model for MLV computation. A faster heuristic using mixed integer linear programming is presented, which shows better runtime than the ILP based approach, but with an average degradation of 5.3% in the leakage value.

In [6], the authors make use of an incremental SAT solver to find the MLV, after a random vector generation step. The problem is formulated using pseudo Boolean constraints. A runtime of the order of a few hours has been reported for some circuits having more than 500 gates. In contrast, our approach is significantly faster.

In [8], the authors use pseudo Boolean enumeration which makes use of integer valued decision diagrams. The authors report that their technique failed on 15 out of 75 (ISCAS85 and MCNC91) benchmark circuits. In [17], the problem of computing MLV is formulated as an integer linear program (ILP) and a linear relaxation of the formulated ILP is solved. The authors report very large runtimes for larger circuits. In such cases, their approach cannot generate a partial or suboptimal MLV result. Our approach, on the other hand, successfully computes the MLV for all the benchmark circuits, as described in the sequel.

The technique of [11] is the only other MLV computation technique which incorporates the effect of PVT variations on the leakage of the circuit. They report a 3.08% reduction in μ + 6*σ leakage of a circuit compared to the random vector based approach. On the other hand, for larger circuits, our technique obtains a 4.14% reduction in the μ + 6*σ leakage (over 10,000 Monte Carlo runs) over the random vector based approach.

Several WPMax-SAT solvers have been developed to date. In our approach, we use a WPMax-SAT solver called IncWMaxSatz [18]. It makes use of a dynamic variable selection heuristic, advanced inference rules and lower bound computation based on unit propagation and failed literals detection.

III. Approach

We cast the problem of determining the exact minimum leakage vector of a combinational design as a WPMax-SAT problem. A WPMax-SAT solver takes two sets of weighted clauses as inputs, namely *hard clauses* and *soft clauses*. The WPMax-SAT solver is required to satisfy *all* the hard clauses. The soft clauses are satisfied such that the sum of the weights of the satisfied soft clauses is maximized. The weight of a hard clause should be more than the sum of the weights of all the soft clauses.

The hard clauses of our WPMax-SAT instance are further classified into S clauses and T clauses. A detailed description of these clauses is given next.

A. Hard clauses

1) S clauses

S clauses are the circuit clauses which describe the functional behavior of the circuit (these must be satisfied by the solver). Let us consider a 2 input AND gate having a, b as input and y as output. The logical relation between y and a, b can be expressed in CNF form as $y \Leftrightarrow ab \equiv (y'+a)(y'+b)(y+a'+b')$. In a similar manner, we can write S clauses for a logic gate having an arbitrary number of inputs.

Consider the circuit having an INV and a NAND2 gate, described by the two equations $x = INV(a)$, and $c = NAND(x,b)$. It has 2 inputs a, b and a single output c. The S clauses for this circuit are $(x'+a')(x+a)(c+b)(c+x)(c'+b'+x')$.

2) T clauses

T clauses describe the leakage of the gates for their different input combinations. For example, consider a 2 input logic gate. The possible input combinations are 00, 01, 10 and 11. Let each combination be represented by the variables $y1$, $y2$, $y3$ and $y4$ respectively. They can be expressed in CNF form using following clauses:

$$y1 \Leftrightarrow a'b' \equiv (y1' + a')(y1' + b')(y1 + a + b)$$
$$y2 \Leftrightarrow a'b \equiv (y2' + a')(y2' + b)(y2 + a + b')$$
$$y3 \Leftrightarrow ab' \equiv (y3' + a)(y3' + b')(y3 + a' + b)$$
$$y4 \Leftrightarrow ab \equiv (y4' + a)(y4' + b)(y4 + a' + b')$$

The above idea can be generalized to any logic gate, with an arbitrary number of inputs. For our example circuit consisting of an INV and a NAND2 gate, the T clauses are shown below. For the INV, $x1$ and $x2$ represent the 2 possible input combinations (1 and 0). For the NAND2 gate, $c1$, $c2$, $c3$ and $c4$ represent the 4 possible input combinations (00, 01, 10 and 11).

$$(x1' + a)(x1 + a')$$
$$(x2' + a')(x2 + a)$$
$$(c1' + x')(c1' + b')(c1 + x + b)$$
$$(c2' + x')(c2' + b)(c2 + x + b')$$
$$(c3' + x)(c3' + b')(c3 + x' + b)$$
$$(c4' + x)(c4' + b)(c4 + x' + b')$$

In order to ensure that an assignment obtained from the WPMax-SAT solver selects *only one* input combination for any gate, additional T clauses are required. The additional T clauses for our example circuit are shown below:

$$(x1 + x2)(x1' + x2')$$
$$(c1 + c2 + c3 + c4)(c1' + c2')(c1' + c3')(c1' + c4')$$
$$(c2' + c3')(c2' + c4')(c3' + c4')$$

We refer to the S and T clauses generated above as hard clauses. We next model the leakage corresponding to each input combination for a given logic gate, as explained next.

B. Soft clauses and weights of clauses

The weight of any input combination for a logic gate G is determined by the leakage of G for the corresponding input combination. For example, for a NAND2 gate, the weight corresponding to input combination 00 is determined by the leakage of the NAND2 gate with 00 as inputs. The input combination with a small leakage value is assigned a higher weight, and the input combination having a higher leakage value is assigned a lower weight. In our gate library (which was implemented using a BPTM 100nm model card [19]), the leakage values over all the gates for all the input combinations was less than 200nA. The weight for an input combination was therefore obtained by subtracting it's corresponding leakage number from a sufficiently large number (to avoid weights $\leq$ 0). The result was then multiplied by a large number to increase the resolution between the weights of any two input combinations having comparable leakage values. In particular, we used the formula

wt_G^j = (500nA - $leakage_G^j$)*1000 for weights. Where $leakage_G^j$ is the leakage (in nA) for gate G for input combination j. For our example circuit, the possible input combinations for the INV gate are specified by $x1$ and $x2$ (for input 1 and 0 respectively). Similarly, for the NAND2 gate, the four possible combinations are specified by $c1$, $c2$, $c3$ and $c4$ (for inputs 00, 01, 10 and 11 respectively). Thus, the soft clauses for the example circuit will be written using $x1$, $x2$, $c1$, $c2$, $c3$ and $c4$. The weight of a soft clause is computed using the above expression. The soft clauses for the example circuit are: $wt^{1}_{INV}\ x1$, $wt^{0}_{INV}\ x2$, $wt^{00}_{NAND2}\ c1$, $wt^{01}_{NAND2}\ c2$, $wt^{10}_{NAND2}\ c3$, $wt^{11}_{NAND2}\ c4$. Note that the weight of all hard clauses (clauses in S and T) should be more than the sum of the weights of all the soft clauses. The weight of each of the S and T clauses is same, and is given by the sum of the weights of all the soft clauses plus one.

The effect of PVT variations on circuit leakage can be elegantly incorporated in this approach by using μ + 3*σ as the leakage value for a given input combination, in the weight formula for soft clauses. If the objective is to optimize the nominal circuit leakage, the nominal leakage is used while computing the weights of soft clauses.

C. Post processing

For MLV computation, the WPMax-SAT instance of a circuit is created as the conjunction of the S, T and soft clauses as explained in Section III-A and III-B. This instance is provided as an input to the IncWMaxSatz solver. The solver is run for $\mathcal{L}$ seconds, which is a user defined input to the solver. For some large circuits, the solver is not able to find the optimal MLV assignment within $\mathcal{L}$ seconds, and it times out. In such cases, the vector reported at the end of timeout (which we refer to as vec_1) may not be the best leakage vector. Therefore we perform a post processing step, in which we search for a better leakage vector through a guided random search, by using the vector computed by the solver at timeout, as explained next.

Consider a circuit having k primary inputs, which times out after $\mathcal{L}$ seconds. Let us assume that the best vector computed by the IncWMaxSatz solver at timeout is $vec_1 = v_1 v_2 \cdots v_k$, where $v_i \in \{0,1\}$. Now, $10000/k$ random vectors are generated for each primary input x_i ($1 \leq$ i $\leq k$), such that the value of x_i is set to v_i, but all other primary inputs are randomly generated. Note that the total number of vectors generated in the post processing step is 10,000. The vector vec_2 which gives the lowest leakage value over the 10,000 vectors (in the post processing step) is retained. Finally, the algorithm returns vec_2 if its leakage is lower than vec_1, otherwise it returns vec_1.

If the cost function to be minimized is μ + 3*σ leakage then the leakage values computed in the post processing step use the μ + 3*σ leakage of the individual gate's input combinations.

IV. Experimental Results

We tested our proposed algorithm on ISCAS85 and MCNC91 benchmark circuits. A 100nm [19] technology library consisting of INV, NAND2, NAND3, NOR2, NOR3, AND2, AND3, OR2, OR3 gates was used for mapping the circuits. After running technology independent logic optimizations, the circuits were mapped for minimum area using SIS [20]. Each input combination of every cell in the cell library was precharacterized for subthreshold leakage (nominal, mean and standard deviation) at a supply voltage of 1.2V. To generate these statistics, Monte Carlo (MC) simulations were run in SPICE using random PVT variations for 30,000 samples. These simulations were run with a 3σ variation of 15% in the channel length, 10% in the power supply and 12.7% in the threshold voltages of devices. The mean and standard deviation of the PVT variables used in MC simulations are listed in Table I. The software code for generating the WPMax-SAT instance of a circuit was written using the Tcl scripting language. IncWMaxSatz [18] was used as the WPMax-SAT solver. The source code of the solver was modified to timeout after $\mathcal{L}$ seconds (which is a user defined input). All the experiments were conducted on a Linux computer with a 2.6 GHz Intel processor and 4GB RAM.

Parameter	μ	σ
Channel length	0.1μm	0.05μm
Power supply	1.2V	0.04V
V_T PMOS	0.3030V	0.0127V
V_T NMOS	0.2607V	0.0110V
Temperature	30°C	1°C

TABLE I

PVT PARAMETER VARIATIONS

We conducted two set of experiments. In the first set, we compare the performance of our approach against the random vector based approach with nominal leakage and μ + 3*σ leakage, as our cost functions respectively. In the second set, we compare the performance of our approach against the random vector based technique using μ + 3*σ as our cost function with total runtime of our approach constrained to be the same as that of random vector based approach.

In all the experiments reported in this paper, for the random vector based approach, the MLV was obtained after 10K random simulations. According to [3], this statistically yields a higher than 99% confidence that we will obtain a leakage vector which is 0.5% from the minimum leakage value.

Also, in all the experiments reported in this paper, after we obtain the MLV for any circuit using any of the competing methods, we perform 10K Monte Carlo simulations and compute the μ, σ and μ + 3*σ leakage from these MC simulations.

A. Comparing against random vector for different cost functions

In this set of experiments, we enforced a timeout of 600 seconds. Note that no post processing step was required for these circuits, since our MLV approach found an exact minimum solution within the timeout duration. Detailed results are not provided due to lack of space. We used both cost functions (nominal leakage as well as μ + 3*σ leakage) for our approach. Over 22 examples (with between 25 and 477 gates, and between 14 and 199 inputs), our approach with nominal leakage as cost function, gives a 7.32% (7.33%) improvement in mean leakage (μ + 3*σ leakage). With μ + 3*σ as the cost function, our approach gives a 7.17% (7.74%) improvement in mean leakage (μ + 3*σ leakage). With μ + 3*σ as the cost function (as opposed to nominal leakage), the average improvement in standard deviation over the random vector based approach increases from 7.08% to 8.61%. A higher improvement in the μ + 3*σ value ensures a lower worst case circuit leakage variation under PVT variations. The remaining results in this paper use μ + 3*σ leakage as the cost function.

Table II contains the circuits for which our approach timed out, and post processing was used for further improvement. Column 1 lists the circuit name, number of logic gates in the circuit, and the number of primary inputs, Columns 2, 3 and 4 show the mean, standard deviation and μ + 3*σ value of the leakage obtained by the random vector based approach, after running Monte Carlo simulations on the MLV that was returned. %Imp_t refers to the percentage improvements at timeout and %Imp_p refers to the percentage improvements after post processing. Columns 5, 6 and 7 show the percentage improvement in the mean, standard deviation and μ + 3*σ value of the leakage of our technique over the random vector based approach (at timeout). Columns 8, 9 and 10 list the percentage improvement in the mean, standard deviation and μ + 3*σ value of the leakage of our approach (after post-processing) over the random vector based approach. Columns 11 and 12 report the runtimes for the random vector based approach and our algorithm respectively. From the results shown in Table II, the average reduction in mean, standard deviation and μ + 3*σ leakage is 4.03%, 4.85% and 4.08% respectively.

B. Comparing against random vector based technique for μ + 3*σ cost function with same total runtime

In this experiment, we compare the performance of our proposed algorithm with the random vector based approach, when the total runtime of our approach is kept the same as that of the random vector based approach using 10K vectors, for larger circuits only.

Let the runtime to generate 10K random vectors for a circuit be τ. The WPMax-SAT solver is run for ατ seconds where $0 \leq \alpha \leq 1$ and post processing is done for (1-α)*τ seconds. Post processing for (1-α)*τ seconds is equivalent to generating (1-α)*10000 post processing vectors instead of 10000 vectors. We experimented with different values of α (0.5 to 0.9) and found that α=0.7 gave best results. Table III shows results with α=0.7, for μ + 3*σ leakage as the cost function. The average improvement in mean is 3.62%, improvement in standard deviation is 4.20%, and in μ + 3*σ leakage, the average improvement is 3.67% (over the random vector

Circuit	(#Gates / #Inputs)	Random μ	Random σ	Random μ + 3*σ	%Imp_t μ	%Imp_t σ	%Imp_t μ + 3*σ	%Imp_p μ	%Imp_p σ	%Imp_p μ + 3*σ	Random T(s)	Our T(s)
i3	(106 / 132)	941.0	105.2	1256.6	22.97	24.85	23.44	22.97	24.85	23.44	20.27	620.61
i5	(132 / 133)	1332.3	140.0	1752.5	-0.10	3.91	0.86	-0.10	3.91	0.86	24.17	624.41
i4	(158 / 192)	892.8	121.0	1255.9	1.78	1.86	1.80	1.78	1.86	1.80	30.88	631.38
i2	(182 / 201)	1038.8	109.3	1366.8	39.69	23.13	35.72	39.69	23.13	35.72	34.26	634.93
x1	(238 / 51)	1436.2	144.9	1870.9	-2.36	4.86	-0.68	-1.21	1.39	-0.61	37.98	637.20
example2	(245 / 85)	1380.1	146.0	1818.1	6.94	0.99	5.51	6.94	0.99	5.51	40.45	639.87
x4	(307 / 94)	1975.5	177.4	2507.9	-2.64	0.86	-1.9	0.32	-0.20	0.21	48.71	648.52
c880	(330 / 60)	1996.3	168.7	2502.5	-6.19	-0.74	-5.09	0.92	4.78	1.70	52.35	652.10
c1908	(408 / 33)	2567.5	193.5	3148.2	2.14	1.94	2.10	2.14	1.94	2.10	62.32	661.74
c499	(447 / 41)	2691.6	191.7	3266.9	-0.99	1.78	-0.50	-0.04	2.20	0.35	68.26	666.56
c1355	(447 / 41)	2663.3	187.5	3225.9	-1.84	-0.65	-1.63	-1.84	-0.65	-1.63	68.1	667.39
rot	(539 / 135)	3318.7	212.3	3955.9	-10.82	-9.46	-10.60	-1.87	-1.34	-1.78	85.10	683.87
frg2	(640 / 143)	4234.0	249.6	4983.0	-6.15	-2.98	-5.67	-0.49	-1.44	-0.64	99.60	698.82
apex6	(646 / 135)	4407.1	270.6	5219.0	4.92	14.97	6.49	4.92	14.97	6.49	100.07	700.06
x3	(662 / 135)	4690.9	270.3	5502.1	4.10	7.50	4.60	4.10	7.50	4.60	101.68	701.66
c3540	(1014 / 50)	7134.4	338.5	8150.1	-1.37	-1.72	-1.41	-0.38	1.82	-0.10	155.33	752.75
c5315	(1496 / 178)	9526.1	383.6	10676.9	-0.52	6.16	0.20	-0.52	6.16	0.20	229.54	827.61
c7552	(2002 / 207)	12712.5	427.9	13996.5	-3.12	-2.43	-3.05	0.10	1.11	0.19	306.39	901.98
c6288	(2942 / 32)	17126.6	482.5	18574.4	-15.92	-11.61	-15.58	-0.81	-0.77	-0.80	443.25	1038.53
AVG					1.60	3.32	1.82	4.03	4.85	4.08		

TABLE II

LEAKAGE RESULTS WITH $\mu + 3*\sigma$ LEAKAGE AS COST FUNCTION (USING POST PROCESSING)

Circuit	(#Gates / #Inputs)	Random μ	Random σ	Random μ + 3*σ	%Imp_t μ	%Imp_t σ	%Imp_t μ + 3*σ	%Imp_p μ	%Imp_p σ	%Imp_p μ + 3*σ
i3	(106 / 132)	941.0	105.2	1256.6	23.12	23.98	23.34	23.12	23.98	23.34
i5	(132 / 133)	1332.3	140.0	1752.5	-1.40	1.63	-0.67	-1.57	4.04	-0.22
c432	(144 / 36)	924.1	124.3	1297.2	-5.38	1.10	-3.52	4.75	7.64	5.58
i4	(158 / 192)	892.8	121.0	1255.9	1.42	2.81	1.82	1.42	2.81	1.82
i2	(182 / 201)	1038.8	109.3	1366.8	39.47	23.32	35.59	39.47	23.32	35.59
x1	(238 / 51)	1436.2	144.9	1870.9	-13.32	-4.19	-11.20	0.01	-1.02	-0.23
example2	(245 / 85)	1380.1	146.0	1818.1	7.19	2.90	6.16	7.19	2.90	6.16
x4	(307 / 94)	1975.5	177.4	2507.9	-2.71	0.27	-2.08	-2.71	0.27	-2.08
c880	(330 / 60)	1996.3	168.7	2502.5	-10.24	-5.94	-9.37	0.06	0.43	0.14
c1908	(408 / 33)	2567.5	193.5	3148.2	0.42	-0.68	0.22	0.42	-0.68	0.22
c499	(447 / 41)	2691.6	191.7	3266.9	-2.11	1.11	-1.54	-2.11	1.11	-1.54
c1355	(447 / 41)	2663.3	187.5	3225.9	-3.24	-0.49	-2.76	-0.67	-0.56	-0.65
rot	(539 / 135)	3318.7	212.3	3955.9	-12.02	-10.79	-11.82	-1.64	-4.56	-2.11
frg2	(640 / 143)	4234.0	249.6	4983.0	-6.59	-3.72	-6.16	0.30	0.59	0.35
apex6	(646 / 135)	4407.1	270.6	5219.0	3.97	12.02	5.22	3.97	12.02	5.22
x3	(662 / 135)	4690.9	270.3	5502.1	3.43	7.22	3.99	3.43	7.22	3.99
c3540	(1014 / 50)	7134.4	338.5	8150.1	-1.89	0.31	-1.62	-0.57	0.91	-0.39
c5315	(1496 / 178)	9526.1	383.6	10676.9	-0.49	4.91	0.09	-0.49	4.91	0.09
c7552	(2002 / 207)	12712.5	427.9	13996.5	-3.07	-1.58	-2.93	-0.54	0.55	-0.44
c6288	(2942 / 32)	17126.6	482.5	18574.4	-19.03	-14.36	-18.67	-1.34	-1.71	-1.37
AVG					-0.12	1.99	0.20	3.62	4.20	3.67

TABLE III

OUR APPROACH WITH $\mu + 3*\sigma$ COST FUNCTION WITH TOTAL RUNTIME SAME AS THAT OF RANDOM VECTOR BASED APPROACH USING 10K VECTORS AND $\alpha = 0.7$

based approach). Thus, given the same runtime as the random vector based technique, our algorithm ensures lower worst case leakage under the effect of process variations.

C. Comparison with previous work

To the best of our knowledge, [11] is the only existing work on MLV computation which includes the effect of PVT variations. All other previous approaches ignore within-die variations and only focus on MLV with nominal leakage as the cost function. As we noted in Section I, this MLV may not be optimal under PVT variations. In [11], the authors report an average improvement of 2.07% in mean circuit leakage and 3.08% improvement in μ + 6*σ leakage compared to a random vector based approach. In contrast, for larger circuits, our approach yields an average improvement of 4.03% in mean circuit leakage and 4.14% improvement in μ + 6*σ leakage.

References

[1] F. Fallah and M. Pedram, "Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits," *IEICE Trans. on Electronics, Special Section on Low-Power LSI and Low-Power IP*, pp. 509–519, 2005.

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

[3] J. Halter and F. Najm, "A Gate Level Leakage Power Reduction Method for Ultra Low Power CMOS Circuits," *Proceedings of CICC*, pp. 475–478, 1997.

[4] M. Johnson, D. Somasekhar, and K. Roy, "Models and algorithms for bounds on leakage in CMOS circuits," *IEEE Transactions on Computer aided design of Integrated circuits and systems*, pp. 714–725, 1999.

[5] R. M. Rao, F. Liu, J. L. Burns, and R. B. Brown, "A Heuristic to Determine Low Leakage Sleep State Vectors for CMOS Combinational Circuits," *Proceedings of ICCAD'03*, p. 689, 2003.

[6] F. Aloul, S. Hassoun, K. Sakallah, and D. Blaauw, "Robust SAT-Based Search Algorithm for Leakage Power Reduction," *Proceedings, Power and Timing Models and Simulations (PATMOS)*, pp. 167–177, 2002.

[7] N. Jayakumar and S. P. Khatri, "An algorithm to minimize leakage through simultaneous input vector control and circuit modification," *Design Automation and Test in Europe Conference*, pp. 618–623, 2007.

[8] K. Chopra and S. Vrudhula, "Implicit Pseudo Boolean Enumeration Algorithms for Input Vector Control," *Proceedings, Design automation conference*, pp. 767–772, 2004.

[9] S. B. et al., "Maximum Leakage Power Estimation for CMOS Circuits," *In proceedings IEEE, AVWLPD*, p. 116, 1999.

[10] S. Narendra, V. De, S. Borkar, D. Antoniadis, and A. Chandrakasan, "Full-chip Sub-threshold Leakage Power Prediction for sub-0.18um CMOS," *Proceedings, ISLPED, 2002*, pp. 501–510, 2002.

[11] K. Gulati, N. Jayakumar, S. P. Khatri, and D. Walker, "A Probabilistic Method to Determine the Minimum Leakage Vector for Combinational Designs in the Presence of Random PVT Variations," *Integration, The VLSI Journal*, vol. 41, pp. 399–412, 2007.

[12] J. Marques-Silva and K. A. Sakallah, "Boolean Satisfiability in Electronic Design Automation," *Proceedings of the 37th conference on design automation*, pp. 675–680, 2000.

[13] H. Zhang, H. Shen, and F. Manya, "Exact Algorithms for MAX-SAT," *In Electronic Notes on Theoretical Computer Science, 86(1)*, 2003.

[14] Z. Xing and W. Zhang, "Efficient Strategies for (weighted) Maximum Satisfiability," *In Proceedings of CP-2004*, pp. 690–705, 2004.

[15] Fu, Z., Malik, and S., "On Solving the Partial Max-SAT Problem," *In Proceedings of SAT'06*, 2006.

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

[17] S. R. Naidu and E. Jacobs, "Minimizing stand-by leakage power in static CMOS circuits," *Design, Automation and Test in Europe*, 2001.

[18] H. Lin, K. Su, C. M. Li, and J. Argelich, "IncWMaxSatz," *Max-SAT Evaluation*, 2008.

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

[20] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. S. Vincentelli, "SIS: A System for Sequential Circuit Synthesis," *University of California-Berkeley, UCB/ERL M92/41*, 1992.