ATPG is often used for high-volume manufacturing designs that cannot adopt the full-scan design methodology, e.g., microprocessors. Having more compacted test sequences will reduce the TAT and the overall testing cost.

V. Conclusion

In this paper, we have proposed several static compaction algorithms for sequential circuits based on efficient test relaxation and ROR schemes. The proposed work has the advantage of quickly restoring a test sequence for a set of faults compared with vector-by-vector fault-simulation-based restoration techniques. The restored subsequence is further compacted by ST algorithm, which allows the removal of redundant vectors without additional fault simulation. These restored subsequences can be either concatenated (having fully specified bits; making RX-LROR), or they can be subjected to increasing the fault coverage (SFC-LROR) and, finally, can also be merged (relaxed input assignments, MR). MR is found to be more effective after applying RX-LROR and SFC-LROR as demonstrated by ITE-Hybrid-II and ITE-Hybrid-III. Finally, we have also proposed an efficient way of taking any compaction algorithm out of saturation. This is achieved by using test relaxation and randomly filling the unspecified bits before reiterating the algorithm, demonstrated by ITE-Hybrid-I.

The proposed static compaction algorithms in this paper have clearly shown the tradeoffs between compaction quality and CPU time.

Acknowledgment

The authors would like to thank Dr. R. Guo for clarifying some of the concepts of vector restoration and K. Al-Utaibi for his help and support in the test relaxation algorithm.

References

[1] T. Marchok, A. El-Maleh, W. Maly, and J. Rajski, "A complexity analysis of sequential ATPG," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 15, no. 11, pp. 1409–1423, Nov. 1996.
[2] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Sequential circuit test generation using dynamic state traversal," in *Proc. Eur. Design Test Conf.*, Mar. 1997, pp. 22–28.
[3] I. Pomeranz and S. M. Reddy, "Vector restoration-based static compaction using random initial omission," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 11, pp. 1587–1592, Nov. 2004.
[4] ——, "A new approach to test generation and test compaction for scan circuits," in *Proc. Design Automation Test Eur.*, 2003, pp. 1000–1005.
[5] ——, "Procedures for static compaction of test sequences for synchronous sequential circuits," *IEEE Trans. Comput.*, vol. 49, no. 6, pp. 596–607, Jun. 2000.
[6] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Fast static compaction algorithms for sequential circuit test vectors," *IEEE Trans. Comput.*, vol. 48, no. 3, pp. 311–322, Mar. 1999.
[7] M. S. Hsiao and S. T. Chakradhar, "State relaxation based subsequence removal for fast static compaction in sequential circuits," in *Proc. DATE*, Feb. 1998, pp. 577–582.
[8] I. Pomeranz and S. M. Reddy, "Vector restoration based static compaction of test sequences for synchronous sequential circuits," in *Proc. Int. Conf. Computer Design*, Oct. 1997, pp. 360–365.
[9] R. Guo, I. Pomeranz, and S. M. Reddy, "On speeding-up vector restoration based static compaction of test sequences for sequential circuits," in *Proc. Asian Test Symp.*, 1998, pp. 467–471.
[10] X. Lin, W. T. Cheng, I. Pomeranz, and S. M. Reddy, "SIFAR: Static test compaction for synchronous sequential circuits based on single fault restoration," in *Proc. IEEE VLSI Test Symp.*, 2000, pp. 205–212.
[11] R. Guo, S. M. Reddy, and I. Pomeranz, "PROPTEST: A property based test pattern generator for sequential circuits using test compaction," in *Proc. Design Automation Conf.*, Jun. 1999, pp. 653–659.
[12] ——, "Reverse-order-restoration-based static test compaction for synchronous sequential circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 3, pp. 293–304, Mar. 2003.
[13] I. Pomeranz and S. M. Reddy, "Vector replacement to improve static-test compaction for synchronous sequential circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 2, pp. 336–342, Feb. 2001.
[14] ——, "Sequence reordering to improve the levels of compaction achievable by static compaction procedures," in *Proc. Conf. Design Automation Test Eur.*, Mar. 2001, pp. 214–218.
[15] A. El-Maleh and K. Al-Utaibi, "An efficient test relaxation technique for synchronous sequential circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 6, pp. 933–940, Jun. 2004.
[16] R. Roy, T. Niermann, J. H. Patel, J. Abraham, and R. Saleh, "Compaction of ATPG-generated test sequences for sequential circuits," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1988, pp. 382–385.
[17] I. Pomeranz and S. M. Reddy, "On static compaction of test sequences for synchronous sequential circuits," in *Proc. 33rd Conf. Design Automation*, Jun. 1996, pp. 215–220.
[18] S. M. Reddy, R. Guo, and I. Pomeranz, "PROPTEST: A property-based test generator for synchronous sequential circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 8, pp. 1080–1091, Aug. 2003.
[19] T. M. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits," in *Proc. EDAC*, 1991, pp. 214–218.
[20] H. K. Lee and D. S. Ha, "HOPE: An efficient parallel fault simulator for synchronous sequential circuits," in *Proc. Design Automation Conf.*, Jun. 1992, pp. 336–340.

Exact and Heuristic Approaches to Input Vector Control for Leakage Power Reduction

Feng Gao and John P. Hayes

***Abstract*—Leakage power consumption is an increasingly serious problem in very large-scale integration circuits, especially for portable applications. Two novel approaches to leakage power minimization in static complementary metal–oxide–semiconductor circuits that employ input vector control (IVC) are investigated. The authors model leakage effects by means of pseudo-Boolean functions. These functions are linearized and incorporated into an exact (optimal) integer linear programming (ILP) model, called virtual-gate ILP, which analyzes leakage variation with respect to a circuit's input vectors. A heuristic mixed-integer linear programming (MLP) method is also proposed, which has several advantages: it is faster, its accuracy can be quickly estimated, and tradeoffs between runtime and optimality can easily be made. Furthermore, the MLP model also provides a way to estimate a lower bound on circuit leakage current. The proposed methods are used to generate an extensive set of experimental results on leakage reduction. It is shown that average leakage currents are usually 1.25 times the minimum, confirming the effectiveness of IVC. The heuristic MLP approach is shown to be approximately 13.6 times faster than the exact ILP method, whereas finding input vectors whose power consumption is only a few percent above the optimum. In addition, the lower bound estimated by the MLP model is also within a few percent of the optimal value.**

***Index Terms*—Input vector control, integer linear programming, leakage current minimization, pseudo-Boolean functions.**

I. Introduction

As CMOS technology evolves, leakage currents are becoming responsible for more and more of a circuit's overall power consumption [10]. This problem is especially serious in portable and

Manuscript received January 25, 2005; revised June 29, 2005. This work was supported by the National Science Foundation under Grant CCR-0073406. This paper was presented at ICCAD 2004. This paper was recommended by Associate Editor M. Pedram.

F. Gao is with Advanced Micro Devices, Inc., Boxborough, MA 01719 USA (e-mail: feng.gao@amd.com).

J. P. Hayes is with the Advanced Computer Architecture Laboratory, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: jhayes@eecs.umich.edu).

Digital Object Identifier 10.1109/TCAD.2006.875711

battery-powered applications, which operate mostly in sleeping mode. Various techniques for reducing leakage power have been proposed, such as multiple V_t [14], variable V_t [11], and input vector control (IVC) [16].

The IVC problem is to find an input vector that minimizes a circuit's leakage current. It takes advantage of the transistor stacking effect within logic gates, where series-connected or stacked transistors tend to have lower leakage than the sum of each transistor's leakage in isolation. As shown in [15], IVC is an effective technique in terms of its ability to reduce leakage current and its granularity, i.e., the minimum sleeping time required. Furthermore, it scales well with CMOS process technology. Several techniques have been proposed to address this problem [5], [7]–[9], [12], [13].

Halter and Najm [5] derive a simple formula to calculate the number of trials necessary to be $\alpha\%$ confident that less than $\varepsilon\%$ of all input vectors produce leakage below a minimum level derived from random-vector simulation. For example, 10 000 random patterns give 99% confidence that less than 0.5% of all input vectors have leakage less than the minimal leakage searched. Unfortunately, a few percent of all 2^n input vectors of a medium-sized circuit with n inputs is still a large number, and some of these vectors may lead to much smaller leakage currents. A heuristic that assigns controllability and leakage penalties to cells to determine the dominating relations between them is proposed in [13]. This approach yields comparable or better results than 10 000 random simulated vectors, but uses much less runtime. This approach even yields an input vector with 10% less leakage current than that of the random search. This result shows the insufficiency of randomly searching even 99.5% of the input space and, at the same time, raises the question of how far the heuristic-based solutions are from the optimum.

Naidu and Jacobs [12] propose an integer linear programming (ILP) model for circuits composed of NAND or AND–OR–INVERT (AOI) gates, which guarantees to find a vector with minimum leakage. Because its runtime is excessive, they reduce the ILP model to a simpler, but nonoptimal, linear programming (LP) model by treating all integer variables as reals (variable-type relaxation). However, neither the nonoptimality of the LP relaxation results nor the tradeoffs between optimality and runtime are quantified.

In this paper, we first propose a novel technique for constructing an ILP model for exact IVC optimization, called the virtual-gate ILP (VG-ILP) model. Virtual gates are cells that we add to the given circuit to facilitate model formulation but have no impact to the function of the original circuit. We also define a faster heuristic technique for IVC optimization, which selectively relaxes variables of the ILP model, leading to a mixed-integer linear programming (MLP) model. Runtime–accuracy tradeoffs can readily be made with the MLP model, and its optimality can be predicted by a simple metric. In addition, the MLP model provides a way to estimate a lower bound on the leakage current for the given circuit.

We also present an extensive set of experimental results on leakage current minimization. We show that the average leakage current is around 25% more than the minimum, indicating the effectiveness of applying the input vectors with minimum leakage. A comparison with previous work indicates that the runtime to solve our VG-ILP model is 5.4 times less than an ILP model without the variable reduction technique [12]. The experiments also show that our heuristic MLP-based approach is able to obtain vectors with near-optimal leakage currents while running 13.6 times faster than the VG-ILP model. Furthermore, the lower bounds estimated using the MLP model are, on average, within a few percent from the optimal one, indicating a tight lower bound.

The paper is organized as follows. We describe the pseudo-Boolean leakage current function and its linearization in Section II. In Section III, we apply the linearized pseudo-Boolean function to IVC and present the VG-ILP model for exact leakage minimization. We describe the heuristic MLP model in Section IV. Experimental results are presented in Section V, followed by conclusions in Section VI.

II. Pseudo-Boolean Current Functions

We introduce pseudo-Boolean functions to represent the leakage current in different types of cells. Mappings of the form $F : B^n \to R$, where $B = \{0, 1\}$ and R is the set of real numbers, are called "pseudo-Boolean functions" [1]. Such functions can be given as algebraic expressions with the general sum-of-products form, i.e.,

$$F(i_0, i_1, \ldots, i_{n-1}) = \sum_{C \in R} C \prod_{0 \le j \le n-1} i_j. \tag{1}$$

Consider a two-input NAND gate G_1 implementing $o = \overline{i_0 i_1}$. Let $L_{ab} \in R$, where $a, b \in \{0, 1\}$ denote G_1's leakage current corresponding to input vector $i_0 i_1 = ab$. The leakage current is hence

$$F(i_0, i_1) = L_{00}\overline{i_0 i_1} + L_{10} i_0 \overline{i_1} + L_{01}\overline{i_0} i_1 + L_{11} i_0 i_1 \tag{2}$$

which is an example of a pseudo-Boolean function. It is easy to see that $L_{ab} = F(a, b)$. If we introduce four auxiliary variables to replace the product terms appearing in (2), F becomes linear in these variables. Such a linear form is needed in LP models. Because it is desirable to introduce as few extra variables as possible, we next derive a simpler form for F.

Consider again the leakage function $F(i_0, i_1)$ for NAND gate G_1. We can apply the well-known Boole–Shannon expansion theorem [4] to (2), i.e.,

$$\begin{aligned} F(i_0, i_1) &= F(i_0, 0)\overline{i_1} + F(i_0, 1) i_1 \\ &= F(i_0, 0) + (F(i_0, 1) - F(i_0, 0))\, i_1 \\ &= F(0, 0) + (F(1, 0) - F(0, 0))\, i_0 \\ &\quad + [F(0, 1) + (F(1, 1) - F(1, 0))\, i_0 \\ &\qquad - F(0, 0) - (F(1, 0) - F(0, 0))\, i_0]\, i_1 \\ &= F(0, 0) + C_0 i_0 + C_1 i_1 + C_2 i_0 i_1. \end{aligned} \tag{3}$$

Here, $C_0 = F(1, 0) - F(0, 0)$ and $C_1 = F(0, 1) - F(0, 0)$ are the coefficients of terms with one input variable. Similarly, the coefficient of the two-variable term is $C_2 = F(1, 1) - F(1, 0) - F(0, 1) + F(0, 0)$. If we define an extra variable X for term $i_0 i_1$, we obtain a modified expression for F that is linear in i_0, i_1 and X. Comparing (2) and (3), we see that we have reduced the number of extra variables from four (i.e., $\overline{i_0}\,\overline{i_1}$, $i_0\overline{i_1}$, $\overline{i_0} i_1$, and $i_0 i_1$) to one (i.e., $i_0 i_1$).

We now generalize the variable reduction method illustrated above for a NAND gate. Let $F(S)$ where $S = \{j_1, j_2, \ldots, j_k\}$ denote F with variables $j_1, j_2, \ldots, j_k$ set to one and all the other variables set to zero. Furthermore, define $X(S)$ to be the logical AND of all variables in S, namely, $X(S) = j_1 j_2 \cdots j_k$. The cardinality of S is denoted by $|S|$. Assume $F(\varnothing) = F(0, 0, \ldots, 0)$, $X(\varnothing) = 1$, and $|\varnothing| = 0$, where $\varnothing$ denotes the empty set. We can now introduce a general expression $C(S)$ to replace C_0, C_1, and C_2 in (3), where S is the set of variables appearing in the product term, i.e.,

$$C(S) = \sum_{\text{All subsets } T \text{ of } S} (-1)^{|S|-|T|} F(S). \tag{4}$$

In fact, (4) holds for the coefficients of any $X(S)$ in a pseudo-Boolean function F, where S is a subset of F's input variables. This equation follows directly from the repeated application of the

Function	Logic operation	Linear form
INV	$f = \mathrm{NOT}(i_1)$	$i_1 + f = 1$
OR	$f = \mathrm{OR}(i_1, i_2, ..., i_n)$	$f \leq i_1 + i_2 + ... + i_n$ $f \geq (i_1 + i_2 + ... + i_n)/n$
NOR	$f = \mathrm{NOR}(i_1, i_2, ..., i_n)$	$f \leq 1 - (i_1 + i_2 + ... + i_n)/n$ $f \geq 1 - (i_1 + i_2 + ... + i_n)$
AND	$f = \mathrm{AND}(i_1, i_2, ..., i_n)$	$f \leq (i_1 + i_2 + ... + i_n)/n$ $f \geq (i_1 + i_2 + ... + i_n - n + 1)/n$
NAND	$f = \mathrm{NAND}(i_1, i_2, ..., i_n)$	$f \leq 2 - (i_1 + i_2 + ... + i_n + 1)/n$ $f \geq 1 - (i_1 + i_2 + ... + i_n)/n$

Fig. 1. Linearization of some basic logic functions.

Boole–Shannon expansion. Therefore, by introducing a set of auxiliary variables $X(S) = j_1, j_2, \ldots j_k$, we can change the leakage current function of an n-input gate to the form

$$F(i_0, i_1, \ldots, i_{n-1}) = \sum_S C(S)X(S) \qquad (5)$$

where S denotes all subsets of the input variables. We can hence linearize the leakage current functions using (5) and obtain efficient LP models for IVC.

III. VG-ILP Model

Because of stacking effects, the leakage current of a series connection of transistors varies significantly with the choice of input vectors [16]. We would like to know how much the leakage currents can vary from input to input and obtain input vectors that minimize or, for bounding purposes, maximize leakage currents.

We now define the problem of obtaining input vectors with minimum (maximum) leakage currents using an ILP formulation with the linearized leakage current expression derived in Section II. This requires us to specify an objective function to be optimized and a set of constraints to be satisfied, all of which must be linear. The variables required are all integers, in fact binary integers, in this particular case.

We define a set of "signal variables," one for each original signal of the circuit, and a set of "auxiliary variables" $X(S)$ for linearization. Suppose gate G has n inputs $i_0, i_1, \ldots, i_{n-1}$. If we view the leakage current of G as a function F of G's inputs, we can linearize $F(i_0, i_1, \ldots, i_{n-1})$ as in (5). Note that the constant coefficients $C(S)$, $S \subseteq \{i_0, i_1, \ldots, i_{n-1}\}$, of the auxiliary variables $X(S)$ are calculated using $F(i_0, i_1, \ldots, i_{n-1})$, which represents the leakage current of G. Consequently, assuming component cells are precharacterized, we can calculate these constants for our ILP model in advance. The overall leakage of a circuit is hence the sum of the leakage currents of all its gates in the form of (5), which serves as the objective function of the ILP model.

The ILP constraints fall into two categories, namely: 1) signal constraints and 2) auxiliary constraints. The signal constraints guarantee correct logic relations between each gate's inputs and output. These logic relations can readily be represented by linear functions [3]. Linear forms for some basic logic functions are shown in Fig. 1. The logic functions of complex gates can be linearized by combining these linear forms.

The auxiliary constraints are intended to determine the correct values of the auxiliary variables $X(S)$. Because $X(S)$ is formed by the logical AND function of all signal variables in S, we also use the inequalities in Fig. 1 to relate these variables. To reduce the size of the model, we reduce the number of auxiliary variables and, hence, the auxiliary constraints, as follows. First, consider single cells in the circuit. For example, if we have a three-input NAND gate G_2 realizing $o_2 = \overline{i_0 i_1 i_2}$ in the original circuit as in Fig. 2(a), we have

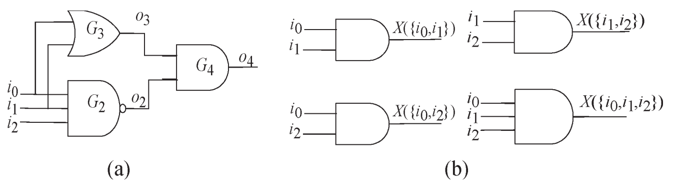

Fig. 2. (a) Small circuit and (b) virtual gate insertion for G_2 and G_3.

Maximize/Minimize

$$\sum_{\subseteq\{i_0,\, i_1,\, i_2\}} C_{G_2}(S)X(S) + \sum_{S\subseteq\{i_0,\, i_1\}} C_{G_3}(S)X(S) + \sum_{S\subseteq\{o_2,\, o_3\}} C_{G_4}(S)X(S)$$

Subject to:

$o_2 \geq 1 - (i_0 + i_1 + i_2)/3$ {Correctness constraints for G_2}

$o_2 \leq 2 - (i_0 + i_1 + i_2 - 2)/3$

$o_3 \geq (i_0 + i_1)/2$ {Correctness constraints for G_3}

$o_3 \leq i_0 + i_1$

$o_4 \geq (o_2 + o_3 - 1)/2$ {Correctness constraints for G_4}

$o_4 \leq (o_2 + o_3)/2$

{Linearization constraints for $X(S)$}

$(i_j + i_k)/2 \geq X(\{i_j, i_k\})$

$(i_j + i_k - 1)/2 \leq X(\{i_j, i_k\})\ \forall\ i_j \neq i_k \in \{i_0, i_1, i_2\}$

All variables are binary

Fig. 3. Example of an ILP model for IVC; the target circuit is that of Fig. 2(a).

eight auxiliary variables $X(S)$, where S is any subset of $\{i_0, i_1, i_2\}$. Because $X(\varnothing) = 1$, no auxiliary variable is necessary in this case. This is also true when S is $\{i_j\}$, a primary variable. In addition, noting that $X(i_0, i_1, i_2) + o_2 = i_0 i_1 i_2 + \overline{i_0 i_1 i_2} = 1$, we can ignore $X(i_0, i_1, i_2)$ while using $1 - o_2$ instead. Therefore, we only need three auxiliary variables, namely: 1) $X(\{i_0, i_1\})$; 2) $X(\{i_0, i_2\})$; and 3) $X(\{i_1, i_2\})$, instead of eight for the three-input NAND case. Generally, for a gate G with inputs $i_0, i_1, \ldots, i_{n-1}$, we can neglect $X(S)$ for $|S| = 0$ or 1, as well as for $|S| = n$ if G is an AND or NAND gate. We can further reduce the number of auxiliary variables by considering multiple cells in the circuit. For example, in the case of G_3 [Fig. 2(a)], which realizes $o_3 = i_0 + i_1$ and shares its inputs with G_2, both gates require an auxiliary variable $X(\{i_0, i_1\})$. We can hence keep one copy and drop any equivalent auxiliary variables.

In general, the original circuit is traversed in topological order. For each gate G with output f and inputs $i_0, i_1, \ldots, i_{n-1}$, we first construct the candidate auxiliary variables $V = \{X(S) | S \subseteq \{i_0, i_1, \ldots, i_{n-1}\}, \text{for } 2 \leq |S| \leq n-1\}$. Then, we determine whether $X(i_0, i_1, \ldots, i_{n-1})$ should be included. In fact, if G is an AND or NAND gate, we can obtain $X(\{i_0, i_1, \ldots, i_{n-1}\})$ directly from f. Furthermore, if G is an OR or NOR gate, we can represent $X(\{i_0, i_1, \ldots, i_{n-1}\})$ as a linear function of f and $X(S)$ in V using De Morgan's law. For other gate types, we also add $X(\{i_0, i_1, \ldots, i_{n-1}\})$ to V. Before actually inserting a virtual gate representing an auxiliary variable in V, we first check whether the same gate exists already to avoid repeating virtual gates.

We define a binary variable for each signal of the augmented circuit. Variables for the signals of actual gates correspond to signal variables, whereas those for virtual gates correspond to auxiliary variables. The virtual gates provide a uniform framework for constructing signal and auxiliary constraints, because the relations are all defined as logic functions. We generate the objective function by adding the leakage function of each actual gate while constructing linearized constraints

Function	Logic operation	Simplified linear form
INV	$f = \mathrm{NOT}(i_1)$	$i_1 + f = 1$
OR	$f = \mathrm{OR}(i_1, i_2, ..., i_n)$	$f = (i_1 + i_2 + ... + i_n)/n$
NOR	$f = \mathrm{NOR}(i_1, i_2, ..., i_n)$	$f = 1 - (i_1 + i_2 + ... + i_n)/n$
AND	$f = \mathrm{AND}(i_1, i_2, ..., i_n)$	$f = (i_1 + i_2 + ... + i_n)/n$
NAND	$f = \mathrm{NAND}(i_1, i_2, ..., i_n)$	$f = 1 - (i_1 + i_2 + ... + i_n)/n$

(a)

i_1i_2	$l(o_3)$	$a(o_3)$
00	0	0
01	1	0.5
10	1	0.5
11	1	1

(b)

Fig. 4. (a) Simplified linearization of logic functions. (b) Logic and arithmetic values for OR gate G_3.

for all gates in the augmented circuits. As an example, Fig. 3 presents the VG-ILP model for the circuit in Fig. 2(a). Having obtained the ILP model, we can now resort to an off-the-shelf ILP solver such as "cplex" [7] to solve the optimization problem and obtain the input vectors with maximum or minimum leakage currents.

IV. MLP Model

We now propose a simplified MLP formulation [2] that employs selective variable-type relaxation to reduce the runtime. The variable set to be relaxed can be modified to obtain the ILP (all variables integer) and LP (all variables real) models, as well as various intermediate MLP models. We first describe the construction of the general MLP model, and then discuss how to choose the variables subject to type change and how to estimate the accuracy of an MLP model.

A. Model Formulation

To facilitate the usage of real signal and auxiliary variables, we linearize the logic functions using the simple equations of Fig. 4(a) instead of the inequalities of Fig. 1. We call the binary form of a variable f its "logic value" $l(f)$, whereas we call the result obtained from the equations in Fig. 4(a) its "arithmetic value" $a(f)$. Therefore, f is set to $l(f)$ if f is chosen to be an integer variable, whereas f is set to $a(f)$ if f is "relaxed" to become a real variable. The ILP model we derived in Section III can be viewed as a special case of the MLP model, where all variables take their logic values. Note that in the case of G_4 in Fig. 2(a), $X(\{o_2, o_3\}) = l(o_4)$.

We use the term "bound pair" for the real number pair $\langle zb, ob\rangle$ consisting of an upper bound zb on the arithmetic value corresponding to logic 0 and a lower bound ob on the arithmetic value corresponding to logic 1. Note that being less than zb or larger than ob is a necessary, but not sufficient, condition that the logic value is zero or one. In general, we calculate logic bounds for basic logic operations as follows:

$$\begin{aligned}&\textsc{not}\,(\langle zb, ob\rangle)\\ &\quad = \langle 1 - ob, 1 - zb\rangle \qquad (6)\\ &\textsc{and}\,(\langle zb_1, ob_1\rangle, \ldots, \langle zb_k, ob_k\rangle)\\ &\quad = \langle (k - 1 + \max(zb_i))/k, (ob_1 + \cdots + ob_k)/k\rangle \qquad (7)\\ &\textsc{or}\,(\langle zb_1, ob_1\rangle, \ldots, \langle zb_k, ob_k\rangle)\\ &\quad = \langle (zb_1 + \cdots + zb_k)/k, \min(ob_i)/k\rangle. \qquad (8)\end{aligned}$$

Logic bound pairs for other operations can be derived easily from these basic ones.

We illustrate the preceding equations using the gates in Fig. 2(a). For primary inputs, $zb = 0$ and $ob = 1$. Consider gate G_3. When both inputs are zero, $l(o_3)$ is zero. However, when any inputs are one, $l(o_3)$ is one, whereas $a(o_3)$ may be 0.5 or 1. Consequently, the logic bound pair for G_3 is $\langle 0, 0.5\rangle$. Similarly, we obtain the logic bound pair $\langle 0, 0.333\rangle$ for G_2. Now consider G_4. The maximum $a(o_4)$ for logic 0 corresponds to $a(o_3) = 0$ and $a(o_2) = 1$. Therefore, $zb = 0.5$. Similarly, $ob = 0.416$, which is obtained when $a(o_3) = 0.5$ and $a(o_2) = 0.333$.

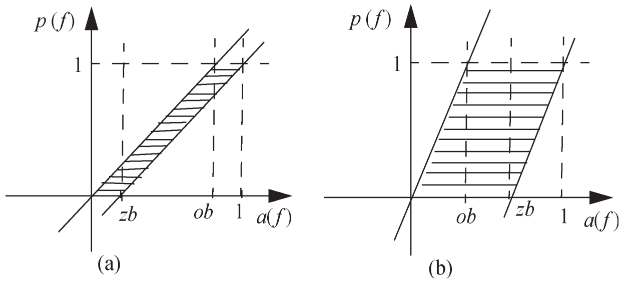

Fig. 5. Leakage parameter calculation.

The bound pair $\langle zb, ob\rangle$ serves to differentiate the logic values of a gate. Consider gate G_3 as in Fig. 2(a). If its inputs are $i_0i_1 = 01$, the arithmetic value of its output $a(o_3) = 0.5$. Because we calculate the bound pair of this gate as $\langle 0, 0.5\rangle$, indicating that logic 0 corresponds to arithmetic values no greater than 0, we conclude that $a(o_3) = 0.5$ means $l(o_3) = 1$. On the other hand, if G_3 were to implement logic AND, its output arithmetic value corresponding to inputs $i_0i_1 = 01$ would still be 0.5. However, the corresponding bound pair would be $\langle 0.5, 1.0\rangle$, indicating that logic 1 corresponds to arithmetic values not less than 1.0. In that case, we would conclude that $a(o_3) = 0.5$ implies $l(o_3) = 0$. Consequently, the bound pair makes it possible to recover the logic value of a signal from its arithmetic value. We take advantage of this fact in our MLP model derivation.

Next, consider transforming the ILP model to an MLP one. First, we again add virtual gates as described in Section III. Then, we construct the signal and auxiliary constraints using the equations in Fig. 4(a) instead of those in Fig. 1. This therefore relates the variables for inputs and outputs of all gates, including the virtual ones. We next construct the objective function for the MLP model to approximate that of the ILP model and add corresponding constraints. Consider a term $C(S)X(S)$ of the ILP model. Inasmuch as there is always a variable f, signal or auxiliary, such that $l(f) = X(S)$ for any variable $X(S)$, we can rewrite $C(S)X(S)$ as $C(\{f\})l(f)$, where $C(\{f\})$ replaces $C(S)$ for simplicity.

In the MLP model, we approximate $C(\{f\})l(f)$ with $C(\{f\})p(f)$, where $p(f)$ is the "leakage parameter" of signal f. If we choose f to be a real variable, i.e., make $f = a(f)$, then $p(f)$ is defined as follows:

$$p(f) \geq (a(f) - zb)/(1 - zb) \qquad (9)$$
$$p(f) \leq a(f)/ob \qquad (10)$$
$$0 \leq p(f) \leq 1. \qquad (11)$$

The introduction of $p(f)$ for signal f prevents the errors of $a(f)$ from affecting the objective function. The possible values of $p(f)$ are shown in the shaded area in Fig. 5, which yields two cases, namely: 1) $ob \leq zb$ and 2) $ob \geq zb$. We introduce constraints (9)–(11) and add $C(\{f\})p(f)$ to the objective function if we choose to use $p(f)$. Note that the logic value $l(f)$ is contained in the defined range of $p(f)$. This property helps to significantly reduce the errors introduced by using arithmetic values. We will discuss how the errors are controlled in the next section.

On the other hand, if f is an integer, namely, $f = l(f)$, then $p(f)$ is defined to be $l(f)$, and $l(f)$ is restored using $a(f)$ and the logic bound pairs by adding the following two constraints:

$$a(f) - (ob + zb)/2 \geq l(f) - 1 \qquad (12)$$
$$a(f) - (ob + zb)/2 \leq l(f). \qquad (13)$$

It is straightforward to verify that $l(f) = 0$ when $a(f) \leq \min(zb, ob)$ and $l(f) = 1$ when $a(f) \geq \max(zb, ob)$. Therefore, we may not be able to restore $l(f)$ if $ob \leq zb$ and $zb \leq a(f) \leq ob$.

However, the error control and variable selection approach in the next section will take this inaccuracy into consideration and minimize the incurred error. Note that when $l(f)$ is used, the logic bound pairs of f should be reset to $\langle 0, 1\rangle$. The logic bound pairs of succeeding gates should be changed accordingly.

B. Variable Selection and Error Prediction

In general, with the goal of minimizing the objective function, the leakage parameter $p(f)$ of a signal f may be the exact logic value $l(f)$ or an approximation to it. Consider $l(f) = 0$ first. In this case, we have $a(f) \leq zb$. If $C(\{f\})$ is positive, the solution error is $p(f) - l(f) = 0$. If $C(\{f\})$ is negative, the error bound becomes

$$p(f) - l(f) = \begin{cases} \frac{a(f)}{ob}, & ob \geq zb \\ 1, & ob \leq zb. \end{cases} \tag{14}$$

Similarly, we can derive the errors for $l(f) = 1$, in which case $a(f) \geq ob$. If $C(\{f\})$ is negative, the error is $p(f) - l(f) = 0$. If $C(\{f\})$ is positive, the error bound is

$$p(f) - l(f) = \begin{cases} \frac{a(f)-1}{1-ob}, & ob \geq zb \\ 1, & ob \leq zb. \end{cases} \tag{15}$$

We hence define the "error bound" EB(f) for signal f as

$$\text{EB}(f) = \begin{cases} \min\left(\frac{zb}{ob}, 1\right), & C\{(f)\} < 0 \\ \min\left(\frac{1-ob}{1-zb}, 1\right), & C\{(f)\} > 0. \end{cases} \tag{16}$$

The errors reach their maxima when $C(\{f\}) < 0$ and $l(f) = 0$, or $C(f) > 0$ and $l(f) = 1$. Fortunately, these cases are rare. Intuitively, if f is an input to AND or NAND gates, then $C(\{f\})$ is positive; thus, we try to assign 0 to $l(f)$, which helps to reduce leakage current. Similarly, if f is an input to OR or NOR gates, $C(\{f\})$ is negative, and we try to assign 1 to $l(f)$. In both cases, the errors are avoided.

To reduce the runtime of the MLP model without significantly sacrificing optimality, we now consider selecting variables subject to type relaxation. Formally, we use

$$L' = C(\{x_1\})\,p(x_1) + C(\{x_2\})\,p(x_2) + \cdots + C(\{x_k\})\,p(x_k) \tag{17}$$

to approximate the leakage current of circuit U, which is represented by

$$L = C(\{x_1\})\,l(x_1) + C(\{x_2\})\,l(x_2) + \cdots + C(\{x_k\})\,l(x_k) \tag{18}$$

where $C(\{x_i\})$ are the coefficients and $p(x_i)$ are real variables whose divergences from the logic values $l(x_i)$ are bounded by (16). The total objective function error (TOFE) of (17) is defined as

$$\text{TOFE} = \max(L - L') = \sum_{x_j} |C\{(x_j)\}|\,\text{EB}(x_j). \tag{19}$$

Note that when we replace $p(x_i)$ with $l(x_i)$, the logic bound pairs of x_i and variables for its succeeding gates may also be changed, as is the resulting error. Therefore, to account for this effect, we define an objective function error (OFE) for each variable x_i as follows:

$$\text{OFE}(x_j) = \sum_{x_j} |C\{(x_j)\}|\,(\text{EB}'(x_j) - \text{EB}(x_j)) \tag{20}$$

where x_j is any real gate in the circuit, EB$'(x_j)$ is the EB of x_j on setting the logic bound pair of x_i to $\langle 0, 1\rangle$, and EB(x_j) is the original EB.

Given the OFE(x_i) where $i = 1, 2, \ldots, k$, we keep variables that satisfy OFE$(x_i) \geq \beta *$ TOFE as binary variables. We can hence adjust β for the number of variables to be fixed as binary and obtain MLP models with different runtime–accuracy tradeoffs.

We are also able to estimate the accuracy of the MLP model using the EBs. We approximate the error as follows. As discussed, the maximum error due to approximating $C(\{x_i\})l(x_i)$ with $C(\{x_i\})$ $p(x_i)$ is reached when $l(x_i) = 1$ and $p(x_i) = 0$ for $C(\{x_i\}) > 0$, or $l(x_i) = 0$ and $p(x_i) = 1$ for $C(\{x_i\}) < 0$. We hence calculate the error using the above variable assignments, in which case $L - L' =$ TOFE. Thus, we obtain an error factor (EF) for the MLP model as follows:

$$\text{EF} = \frac{\text{TOFE}}{\sum\limits_{C(\{x_j\})} C\{(x_j)\}}. \tag{21}$$

Our experimental results will show that EF is very good at predicting the errors of the MLP model.

C. Bounds on Minimum Leakage Current

For a circuit U, its leakage current when applying the input vector obtained using the MLP model will be no less than U's minimum leakage current and is hence an upper bound on the minimum leakage current. We would also like to develop a lower bound to further quantify U's leakage current.

Consider (17) and (18), which are the objective functions of the MLP and ILP models, respectively. Let $R(x)$ be the range of variable x. In fact, $R(p(f))$ in (17) is a superset of the corresponding $R(l(f))$ in (18) in most cases, i.e., $R(l(f)) \subset R(p(f))$. Hence, we use (17) to estimate the lower bound given by (18).

V. Experimental Results

We implemented a software tool to automatically generate the proposed models. The LP problems were solved using cplex. We applied the proposed methods to a set of representative ISCAS and MCNC benchmark circuits. The circuits were synthesized using the Synopsys Design Compiler and a TSMC 0.18-μm standard cell library.

A. VG-ILP Model and Leakage Variation

First, we studied leakage current variation. For each circuit, the VG-ILP model defined in Section III was generated. The input vectors with maximum and minimum leakage currents are obtained by maximizing or minimizing the leakage current objective functions. The leakage currents for given input vectors are then calculated. We also performed simulations to obtain the average and minimum leakage currents over 10 000 randomly chosen input vectors. The measured data are presented in Table I. For each circuit, we show its gate count, followed by maximum, average, minimum, and randomly selected minimum leakage currents in picoamperes. The ratios of the maximum, average, and randomly selected minimum leakage currents to the minimum one are also given. We can see that the leakage current variation is circuit dependent and ranges from 21% to 138%, with an average value of 59.8%. Average leakage currents fall between 6% and 70% of the minimum leakage, with an average of 26%. Therefore, we conclude that the use of IVC with the minimum leakage vector can reduce leakage power by 26% on average. In addition, the ratio D/C shows that randomly selected minima may vary widely from 2% to 24%, indicating the insufficiency of randomly searching even 99.5% of the input space.

We compared the VG-ILP model with the ILP model proposed in [12]. The ILP models used in [12] were regenerated using our tool without checking for the existence of equivalent auxiliary variables

TABLE I
LEAKAGE CURRENT VARIATION WITH INPUT VECTORS

Circuit	Number of gates	Max. leakage A	Ave. leakage B	Min. leakage C	Randomly selected min. leakage D	Ratio A/C	Ratio B/C	Ratio D/C
C432	121	544.1	417.8	344.3	361.9	1.580	1.213	1.051
C880	345	1557.5	1157.6	911.8	992.3	1.708	1.269	1.088
C1355	456	1541.1	1438	1360	1382.6	1.133	1.057	1.017
C1908	435	1534.4	1379.3	1274.5	1301.4	1.204	1.082	1.021
C2670	746	2882.4	2387.7	2152.9	2259.5	1.339	1.109	1.050
C3540	892	3327.9	2860.3	2442.6	2532.2	1.362	1.171	1.037
Dalu	787	3049.3	2678.1	2267.4	2342.7	1.345	1.181	1.033
Rot	536	2171.5	1728.6	1455.0	1594.9	1.492	1.188	1.096
X3	558	2515.5	1819.0	1208.0	1447.4	2.082	1.506	1.198
Vda	353	2017.2	1841.5	1589.5	1593.1	1.269	1.159	1.002
Alu4	565	2077.2	1903.6	1732.2	1735.7	1.199	1.099	1.002
Apex6	615	2436.2	1898.8	1630.0	1761.1	1.495	1.165	1.081
Frg2	587	2532.2	1836.7	1437.6	1632.3	1.762	1.278	1.130
I4	160	699.0	466.7	302.6	402.7	2.31	1.54	1.33
I5	134	1112.1	933.9	767.8	858.8	1.45	1.22	1.12
I6	376	1815.1	1314.2	764.1	949.9	2.375	1.720	1.243
I7	429	2334.3	1722.0	1135.0	1382.1	2.06	1.52	1.22
I10	1781	N/A	6356	5631	6019.7	N/A	1.129	1.069
Ave.						**1.598**	**1.256**	**1.100**

TABLE II
COMPARISON OF VG-ILP WITH THE ILP MODEL IN [11]

Circuit	Auxiliary variable number N_1 (VG-ILP)	Auxiliary variable number N_2 (ILP [11])	Auxiliary variable number ratio N_2/N_1	Runtime T_1 (VG-ILP)	Runtime T_2 (ILP [11])	Runtime ratio T_2/T_1
C432	174	206	1.18	0.81	0.29	0.36
C880	343	388	1.13	3.26	3.16	0.97
C1355	66	140	2.12	1.69	31.8	18.82
C1908	184	234	1.27	6.87	9.52	1.39
C2670	505	587	1.16	4.60	17.5	3.80
C3540	1020	1213	1.19	231.6	645.4	2.79
Dalu	838	924	1.10	62.58	152.8	2.44
Rot	367	473	1.29	2.43	6.87	2.83
X3	746	1105	1.48	0.76	13.85	18.22
Vda	797	856	1.07	38.1	32.6	0.85
Alu4	537	665	1.23	154.8	454.5	2.94
Apex6	392	435	1.11	7.79	6.68	0.85
Frg2	621	823	1.33	1.75	17.8	10.17
I4	84	84	1.00	0.07	0.07	1.00
I5	0	0	1.00	0.03	0.03	1.00
I6	422	426	1.01	0.32	0.31	0.97
I7	639	762	1.19	0.83	18.21	21.94
I10	1667	2061	1.24	10851	N/A	N/A
Ave.			**1.23**			**5.37**

in virtual gate insertion. The number of auxiliary variables required and the runtime to solve the models are presented in Table II. The auxiliary variable numbers and runtime ratios of these two models are also presented. On average, the VG-ILP model requires 23% fewer auxiliary variables, which leads to a 5.37× speedup in performance.

B. Evaluation of MLP Model

Next, we applied our simplified MLP model to the same set of circuits as [12] and also to some larger ones such as c6288 and c7552. For these experiments, β is set to 0.01. Intuitively, this value fixes to integers the variables that reduce the TOFE by more than 1%. The results are presented in Table III. Note that I7 is too large for the ILP model; thus, the corresponding entries are marked N/A.

The results in Table III show that MLP is an efficient way to obtain a low-leakage input vector. On average, the leakage currents only differ by 4.6% from the optimal ones, whereas the runtime is on the order of seconds. In fact, we obtain a 13.6× speedup over the VG-ILP model, excluding the extreme case for I10 where the speedup is more than 700×. Furthermore, MLP models with small EFs, for example, $\text{EF} < 0.1$, lead to input vectors with very small errors in the minimum leakage current. For all circuits with $\text{EF} < 0.1$, the vectors obtained

TABLE III
COMPARISON OF THE MLP AND VG-ILP MODELS

Circuit	MLP leakage E	Time (s)	Speedup factor	EF	Leakage ratio E/C
C432	353	3.01	0.27	0.015	1.025
C880	1045	0.22	14.82	0.186	1.147
C1355	1379.6	0.06	28.17	0.154	1.018
C1908	1285	0.76	9.04	0.001	1.008
C2670	2255	0.2	23.0	0.007	1.048
C3540	2443	19.52	11.86	0.011	1.000
C6288	10848.5	43.82	N/A	0.350	1.000
C7552	5920	74.65	N/A	0.050	0.960*
Dalu	2402	3.48	17.98	0.001	1.059
Rot	1484.3	0.14	17.36	0.004	1.020
X3	1239.8	0.18	4.22	0.028	1.026
Vda	1590.1	3.1	12.29	0.016	1.000
Alu4	1732.2	5.12	32.23	0.096	1.002
Apex6	1673.5	0.17	45.82	0.001	1.026
Frg2	1437.6	0.4	4.38	0.028	1.000
I4	309.3	0.02	3.5	0.074	1.022
I5	769.4	0.02	1.5	0.423	1.002
I6	846.6	0.07	4.57	0.254	1.108
I7	1141.2	0.29	2.86	0.08	1.005
I10	5731	15.24	712.0	0.012	1.018
Ave.			**13.64***		**1.028**

using the MLP approach exhibit leakage currents that are only 1.9% from the optimum.

The small EF of c7552 suggests that the solution of the MLP model for c7552 should be very close to the optimum, whereas in the case of c6288, we can draw no conclusions about optimality from the EF alone. However, we know that c6288 is a multiplier composed of (modified) full and half adders [6]; thus, is it easy to analyze manually. It is readily seen that the all-0s input vector results in the lowest leakage in each adder. Because zeroes propagate through all the adders, we conclude that the all-zeroes input vector leads to the minimum leakage current in the c6288. In fact, our MLP model gives exactly the same input vector!

In addition, our heuristic MLP approach provides a way to control the tradeoff between runtime and optimality: We can simply adjust β to obtain a faster or more accurate model. For example, if we set $\beta = 0.005$ for c880, the MLP model gives a vector with leakage current 4% higher than the optimal in 3 s. In fact, we can iteratively reduce β until the EF is below the desired value, for example, 0.1.

We also examined the accuracy of our lower bound estimates with the MLP model. The results appear in Table IV, which shows estimated leakage current lower bounds and the corresponding ratios with the minimum leakage currents. Note that we do not have the minimum leakage current for c7552. However, because the estimated lower bound for c7552 is close to the result in E, we use that data for E instead. The errors of the lower bounds are within 13% of the optimum, with an average of 6%. Hence, we conclude that our MLP model also provides a relatively efficient way to estimate a lower bound on leakage current for a given circuit.

C. Comparison With Previous Work

Finally, we compare our results with previously reported results in [12] and [13] (see Table V). We present the results obtained by random search (referred to as result D), the MLP-based approach (referred to as result E), and the cited earlier methods. The results are given as ratios to the minimum leakage current, namely, result C. On average, the error of our MLP-based approach is 5.3%, whereas the errors of the methods of [12] and [13] are 3.7% and 10.4%, respectively. Considering those circuits with EFs larger than 0.1 (marked with asterisks beside the circuit names in Table V), the accuracy of the MLP-based approach is even greater.

TABLE IV
LOWER BOUNDS ESTIMATED BY THE MLP MODEL

Circuit	Leakage lower bound F	Leakage ratio F/C
C432	321.0	0.93
C880	904.6	0.99
C1355	1193.0	0.88
C1908	1269.6	1.00
C2670	2087.3	0.97
C3540	2193.7	0.90
C6288	9398.6	0.87
C7552	5890.4	0.99
Dalu	2155.1	0.95
Rot	1391.8	0.96
X3	1073.1	0.89
Vda	1486.9	0.94
Alu4	1563.3	0.90
Apex6	1587.6	0.97
Frg2	1305.6	0.91
I4	299.4	0.99
I5	726.1	0.95
I6	711.7	0.93
I7	1100.2	0.97
I10	5305.9	0.94
Ave.		**0.94**

TABLE V
COMPARISON WITH PREVIOUS WORK

Circuit	Ratio D/C	Ratio E/C	Results in [13]	Results in [12]
C432*	1.051	1.025	1.067	N/A
C880	1.088	1.147	1.081	1.04
C1355	1.017	1.018	1.004	1.04
C1908*	1.021	1.008	1.000	1.08
C2670*	1.050	1.048	1.047	1.07
C6288*	N/A	1.000	1.011	N/A
C7552*	N/A	0.960*	0.993*	N/A
I4*	1.33	1.022	1.20	1.00
I5	1.12	1.002	1.04	1.03
I6	1.243	1.107	1.300	1.00
I7*	1.22	1.005	1.23	1.00
I10*	1.069	1.018	1.074	1.07
Ave.	**1.113**	**1.036**	**1.104**	**1.037**

VI. CONCLUSION

We have presented an efficient method to model a circuit's leakage currents by means of linearized pseudo-Boolean functions. Exact and approximate LP models are systematically constructed by applying the linearization approach to IVC. Selective variable-type relaxation is proposed to enable a tradeoff between the runtime of the approximate MLP model and its optimality.

Our experiments show that the average leakage current is around 25% larger than the minimum ones. Therefore, by applying the minimum leakage input vector, leakage current can be significantly decreased. The introduction of virtual gates helps to reduce the number of auxiliary variables by about 23%, which in turn results in 5.4× runtime improvement.

An efficient heuristic approach was proposed to selectively relax some binary constraints of the ILP model leading to an MLP model. As shown by the experiments, the resulting input vectors have leakage currents that are only slightly higher than the best ones. Furthermore, we can solve the MLP model about 13.6 times faster than the exact ILP model and can also make runtime–accuracy tradeoffs quite easily. The MLP model provides a relatively tight lower bound estimation to the minimum leakage current as well. Comparison with previous work shows that the MLP-based approach also achieves better leakage reduction.

References

[1] E. Boros and P. L. Hammer. (2001). Pseudo-Boolean optimization. Rutgers University, New Brunswick, NJ, Rutcor Res. Rep. 2001-33. [Online]. Available: FTP://dimacs.rutgers.edu/pub/dimacs/TechnicalReports/TechReports/2001/2001-33.ps.gz

[2] F. Gao and J. P. Hayes, "Gate sizing and V_t assignment for active-mode leakage power reduction," in *Proc. ICCAD*, 2004, pp. 258–264.

[3] A. Gupta and J. P. Hayes, "CLIP: Integer-programming-based optimal layout synthesis of 2D CMOS cells," *ACM TODAES*, vol. 5, no. 3, pp. 510–547, Jul. 2000.

[4] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Optimization*. Boston, MA: Kluwer, 2000.

[5] J. Halter and F. Najm, "A gate-level leakage power reduction method for ultra low power CMOS circuit," in *Proc. CICC*, 1997, pp. 475–478.

[6] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Des. Test Comput.*, vol. 16, no. 3, pp. 72–80, Jul.–Sep. 1999.

[7] *High-performance Software for Mathematical Programming and Optimization*, Mountain View, CA: ILOG Technology, ILOG cplex webpage. [Online]. Available: http://www.ilog.com/products/cplex/

[8] M. C. Johnson, D. Somasekhar, and K. Roy, "Models and algorithms for bounds on leakage in CMOS circuits," *IEEE Trans. Comput.-Aided Des.*, vol. 18, no. 6, pp. 714–725, Jun. 1999.

[9] M. C. Johnson, D. Somasekhar, L. Y. Chiou, and K. Roy, "Leakage control with efficient use of transistor stacks in single threshold CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 1, pp. 1–5, Feb. 2002.

[10] Semiconductor Industry Association. (2003). *International Technology Roadmap for Semiconductors*. [Online]. Available: http://public.itrs.net/

[11] T. Kuroda *et al.*, "A 0.9-V, 150-MHz, 10-mW, 4 mm^2, 2-D discrete cosine transform core processor with variable threshold-voltage (VT) scheme," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1770–1779, Nov. 1996.

[12] S. Naidu and E. Jacobs, "Minimizing stand-by leakage power in static CMOS circuits," in *Proc. DATE*, 2001, pp. 370–376.

[13] R. M. Rao *et al.*, "A heuristic to determine low leakage sleep state vectors for CMOS combinational circuits," in *Proc. ICCAD*, 2003, pp. 689–692.

[14] S. Shigematsu *et al.*, "A 1-V high-speed MTCMOS circuit scheme for power-down applications," in *Proc. Dig. Symp. Tech. VLSI Circuits*, 1995, pp. 125–126.

[15] Y.-F. Tsai *et al.*, "Implications of technology scaling on leakage reduction techniques," in *Proc. DAC*, 2003, pp. 187–190.

[16] Y. Ye, S. Borkar, and V. De, "A new technique for standby leakage reduction in high-performance circuits," in *Proc. Symp. VLSI Circuits*, 1998, pp. 40–41.

Comparison of Algorithms for Frequency Domain Coupled Device and Circuit Simulation

Yutao Hu and Kartikeya Mayaram

***Abstract*—This paper examines three algorithmic approaches for the frequency-domain harmonic-balance method in a coupled device and circuit simulator. These are the nonquasi-static, quasi-static, and modified Volterra series approaches. A detailed qualitative and quantitative comparison between these approaches is provided in terms of accuracy and the computational resources required.**

***Index Terms*—Coupled device and circuit simulation, coupled simulation, harmonic-balance method, mixed-level simulation, steady-state analysis.**

I. Introduction

With the continued increasing demand for RF ICs there is a critical need for an accurate and efficient simulation of circuits in the periodic steady state. Certain aspects of system performance are easier to characterize and verify in the periodic or quasi-periodic steady state. Examples of these are large-signal distortion, power, frequency, noise, and transfer characteristics such as gain and impedance. Steady-state analyses are available in several commercial circuit simulators.

For RF applications, distributed device effects are important and must be included in simulations with accurate device models. Otherwise, the simulated circuit performance will deviate significantly from the actual circuit. Since model accuracy is critical for high-frequency simulations, it has been receiving attention recently. In the absence of accurate compact models, coupled or mixed-level circuit and device simulators [2]–[6], [13] can be used. These simulators provide a framework in which physical (numerical) models for critical devices can be simulated in conjunction with standard circuit-level compact models. To enable an accurate simulation of RF circuits, steady-state analyses are required in coupled device and circuit simulators.

The steady-state analyses can be classified as frequency-domain or time-domain analyses [1]. The frequency-domain harmonic-balance method can solve quasi-periodic steady-state problems, such as mixers, which are usually impractical to solve by the traditional time-domain methods [7]. In the context of coupled device and circuit simulation, a time-domain shooting method [20] has been implemented in [8] and the frequency-domain harmonic-balance method in [5], [9]–[11]. The harmonic-balance method has previously been applied for the simulation of semiconductor devices in [12], [13], [24], and [25], and coupled to linear circuit elements in [12] and [13]. However, there have been no extensions to general coupled device and circuit simulation.

This paper compares three algorithmic approaches for harmonic-balance analysis in the coupled device and circuit simulator (CODECS) [3]. These include quasi-static (QS) [5], nonquasi-static (NQS) [12], and modified Volterra series (MVS) [14]–[16] methods. From the circuit-level point of view, the coupled simulation can handle and interpret device-level models in different ways. The NQS approach

Manuscript received April 30, 2004; revised December 7, 2004 and June 2, 2005. This work was supported by the National Science Foundation under Grant CCR-0120275, by the Semiconductor Research Corporation under Contract 2001-TJ-922, and by the Defense Advanced Research Projects Agency NeoCAD program. This paper was recommended by Associate Editor Z. Yu.

Y. Hu is with Agilent Technologies, Santa Rosa, CA 95403 USA.

K. Mayaram is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331 USA.

Digital Object Identifier 10.1109/TCAD.2006.871621