[6] A. Chandra and K. Chakrabarty, "A unified approach to reduce SOC test data volume, scan power and testing time," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 3, pp. 352–363, Mar. 2003.

[7] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," *IEEE Trans. Comput.*, vol. 52, no. 8, pp. 1076–1088, Aug. 2003.

[8] R. Dorsch and H.-J. Wunderlich, "Tailoring ATPG for embedded testing," in *Proc. ITC*, 2001, pp. 530–537.

[9] P. Gonciari, B. Al-Hashimi, and N. Nicolici, "Variable-length input Huffman coding for system-on-a-chip test," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 783–796, Jun. 2003.

[10] P. Gonciari, B. Al-Hashimi, and N. Nicolici, "Synchronization overhead in SOC compressed test," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 1, pp. 140–152, Jan. 2005.

[11] I. Hamzaoglu and J. Patel, "Test set compaction algorithms for combinational circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 8, pp. 957–963, Aug. 2000.

[12] V. Iyengar, K. Chakrabarty, and B. Murray, "Deterministic built-in pattern generation for sequential circuits," *J. Electron. Test.—Theory and Applications*, vol. 15, no. 1/2, pp. 97–114, Aug./Oct. 1999.

[13] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs," in *Proc. ITC*, 1998, pp. 458–464.

[14] A. Jas, J. Ghosh-Dastidar, M.-E. Ng, and N. Touba, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 797–806, Jun. 2003.

[15] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel Huffman coding: An efficient test-data compression method for IP cores," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 6, pp. 1070–1083, Jun. 2007.

[16] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Optimal selective Huffman coding for test-data compression," *IEEE Trans. Comput.*, vol. 56, no. 8, pp. 1146–1152, Aug. 2007.

[17] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Test-data compression based on variable-to-variable Huffman encoding with codeword reusability," Dept. Comp. Sci., Univ. Ioannina, Greece, Tech. Rep. [Online]. Available: http://charon.cs.uoi.gr/~tech_report/hci/publications/tech_rep__01.pdf

[18] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel-Huffman test-data compression for IP cores with multiple scan chains," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to be published.

[19] C. Krishna and N. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. ITC*, 2001, pp. 321–330.

[20] C. Krishna and N. Touba, "Adjustable width linear combinational scan vector decompression," in *Proc. ICCAD*, 2003, pp. 863–866.

[21] J. Lee and N. Touba, "Low power test data compression based on LFSR reseeding," in *Proc. ICCD*, 2004, pp. 180–185.

[22] J. Lee and N. Touba, "Combining linear and nonlinear test vector compression using correlation-based rectangular encoding," in *Proc. 24th VTS*, 2006, pp. 252–257.

[23] L. Li and K. Chakrabarty, "Test set embedding for deterministic BIST using a reconfigurable interconnection network," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 9, pp. 1289–1305, Sep. 2004.

[24] L. Li, K. Chakrabarty, S. Kajihara, and S. Swaminathan, "Efficient space/time compression to reduce test data volume and testing time for IP cores," in *Proc. 18th Int. Conf. VLSI Des.*, 2005, pp. 53–58.

[25] S. Lin, C. Lee, and J. Chen, "Adaptive encoding scheme for test volume/time reduction in SOC scan testing," in *Proc. ATS*, 2005, pp. 324–329.

[26] A. El-Maleh and R. Al-Abaji, "Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression," in *Proc. ICECS*, 2002, vol. 2, pp. 449–452.

[27] M. Nourani and M. H. Tehranipour, "RL-Huffman encoding for test compression and power reduction in scan applications," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, no. 1, pp. 91–115, Jan. 2005.

[28] S. Reda and A. Orailoglu, "Reducing test application time through test data mutation encoding," in *Proc. DATE*, 2002, pp. 387–393.

[29] P. Rosinger, P. Gonciari, B. Al-Hashimi, and N. Nicolici, "Simultaneous reduction in volume of test data and power dissipation for systems-on-a-chip," *Electron. Lett.*, vol. 37, no. 24, pp. 1434–1436, Nov. 2001.

[30] M. Tehranipour, M. Nourani, and K. Chakrabarty, "Nine-coded compression technique for testing embedded cores in SoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 6, pp. 719–731, Jun. 2005.

[31] Z. Wang and K. Chakrabarty, "Test data compression for IP embedded cores using selective encoding of scan slices," in *Proc. ITC*, 2005, pp. 1–10.

[32] S. Ward, C. Schattauer, and N. Touba, "Using statistical transformations to improve compression for linear decompressors," in *Proc. IEEE Int. Symp. DFT*, 2005, pp. 42–50.

State-Sensitive X-Filling Scheme for Scan Capture Power Reduction

Jing-Ling Yang and Qiang Xu

***Abstract*—Based on the operation of a state machine, this paper elucidates a comprehensive frame for probability-based primary-input-dominated X-filling methods to minimize the total weighted switching activity (WSA) during the scan capture operation. Experimental results demonstrate that the proposed approach significantly reduces both average and peak WSAs.**

***Index Terms*—Scan test, sequential circuits, switching activity (SA), test generation.**

I. Introduction

Full scan is the most utilized test strategy in the semiconductor industry. Applying a scan test, however, results in the switching activity (SA) of a circuit under test (CUT) during test mode that is far beyond that during normal operational mode [1], [2]. Various techniques such as scan chain reordering, scan chain segmentation, clock gating, and low-power automatic test pattern generation (ATPG) have been developed to reduce scan shift power dissipation (e.g., [3]–[7]). Some techniques, including circuit modification [8], ATPG algorithm [9], and X-filling techniques [10]–[13], focused on scan capture power reduction. Among these scan capture power reduction methods, X-filling techniques do not require a modification in the CUT and do not need to rerun the time-consuming ATPG process and, hence, are widely accepted.

As well as having no effect on CUT and ATPG, X-filling techniques are compatible with those shift power reduction techniques that use or do not use X-bits. Procedures for generating X-bits for all the steps of the scan test (which are, namely, scan in, scan capture, and scan out) can be found in [10]. Examples of X-filling capture power reduction techniques that are compatible with non X-filling shift power reduction techniques can be found in [13].

Sankaralingam and Touba [10] introduced unspecified values (X-bits) in the scan vector and reassigned them to reduce scan peak power, which may be caused by scan-in, scan capture, and/or scan-out problems. To decrease scan peak power, first, X-bits are introduced in the scan vector and then reassigned to minimize the number of state changes in the scan flip-flops (SFFs) between two consecutive operation steps. For scan capture peak power reduction, incremental fault-free simulations are used in the procedure.

Manuscript received February 27, 2007; revised August 23, 2007, November 11, 2007, and March 1, 2008. This paper was recommended by Associate Editor S. Vrudhula.

The authors are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: jlyang@cse.cuhk.edu.hk; qxu@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TCAD.2008.923418

Wen *et al.* [11] developed an X-filling approach in which X-bits are set in the test cubes to decrease the number of transitions at the outputs of SFFs in capture operation. Repeated simulations of incrementally updated test cubes and the application of line justification and implications are utilized in the procedure.

Remersarol *et al.* [12] developed a probability-based X-filling procedure to fill X-bits in SFFs in one step instead of iterative incremental fill and simulation. This work is based on the two-pattern launch off test. Its object is to minimize the Hamming distance between the two consecutive test patterns.

Wen *et al.* [13] developed an incremental approach to fill the X-bits in a test cube in a one-by-one manner to minimize gate transitions.

In summary, existing lower capture power (LCP) X-filling techniques take three broad approaches: 1) reducing the Hamming distance between before- and after-capture SFFs [10], [11]; 2) reducing the Hamming distance between two consecutive test cubes in a two-pattern launch off test [12]; and 3) reducing the number of gate transitions using an incremental filling method [13].

The effectiveness of previous LCP X-filling techniques is still not very satisfactory. First, the target of LCP optimization is the total weighted switching activity (WSA) under the capture operation rather than the Hamming distance between the before- and after-capture SFFs or the Hamming distance between two consecutive test cubes in a two-pattern launch off test. Second, the operation of a state machine is based on the state transfer graph; therefore, the adopted X-filling approaches and their associated calculations must be based on state operations rather than state line operations. These state lines are correlated with each other. Such spatial relations are not well addressed. In [10]–[13], X-bits in SFFs are incrementally filled, and hence, the filling order significantly affects the effectiveness of the methods; whereas in [12], the X-bits in SFFs are considered to be independent during probability calculation. Third, the running of a sequential circuit, given an initial state, is controlled by primary inputs (PIs). Accordingly, setting X-bits in PIs is critical to the results of LCP X-filling methods. However, this issue has not been systematically considered in the cited works [10]–[13]. These observations motivate this work.

In this paper, by addressing the aforementioned problems, we first derive a probability-based WSA model for capture operation and then present an effective and efficient SSF scheme that targets the transitions of both SFFs and internal gates. Experimental results show a significant reduction in both average and peak capture power consumption.

The remainder of this paper is structured as follows. Section II introduces the terminologies and definitions used in this paper. WSA models for capture operation from a probability point of view are described in Section III. Next, Section IV presents our proposed SSF scheme. Section V presents the experimental results on ISCAS'89 benchmark circuits. Finally, Section VI concludes this paper.

II. Terminologies and Notations

In this section, we first briefly introduce some terminologies and definitions used in this paper.

A. Capture Operation

As can be observed in Fig. 1, a scan circuit has PIs and pseudoprimary inputs (PPIs), PIs are applied directly, and PPIs are applied through SFFs. The test response of the combinational circuit has primary outputs and pseudoprimary outputs (PPOs). The capture operation is to load PPOs into SFFs to replace PPIs.

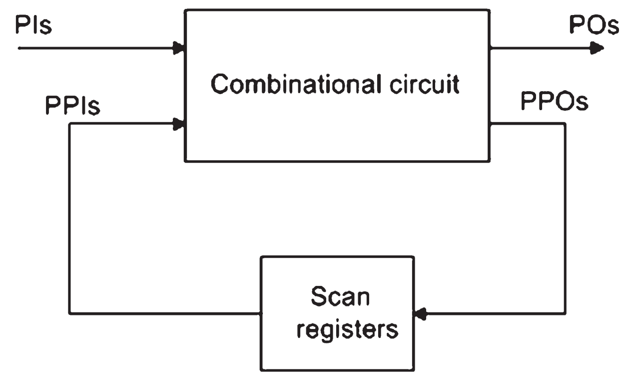

Fig. 1. Simple scan circuit.

B. Test Cube

A test cube is a partially specified input bit combination with at least one don't-care bit, whereas a test vector refers to a fully specified input bit combination without any don't-care bit. A don't-care bit is also called an X-bit. Test cubes can be generated during ATPG.

C. X-Filling

X-filling is the process of assigning logic values to the unspecified bits (X-bits) in a test cube so as to obtain a fully specified test vector with a certain characteristic.

D. WSA

The WSA of a signal line is the number of state changes at the line multiplied by its fanout [1]. The WSA of the entire circuit is obtained by summing the WSA of all the signal lines in the circuit. In this paper, WSA is used as a representation of power consumption.

E. Notations

The following notations will be used in this paper.

- n Number of PIs.
- m Number of SFFs.
- t Number of signal lines.
- I^k $(i_1^k, i_2^k, \ldots, i_n^k)$, PIs in the kth test cube, also called the before-capture PIs of the kth test cube.
- I^{k+1} $(i_1^{k+1}, i_2^{k+1}, \ldots, i_n^{k+1})$, the after-capture PIs of the kth test cube.
- S^k $(s_1^k, s_2^k, \ldots, s_m^k)$, PPIs in the kth test cube, also the before-capture state of the kth test cube.
- S^{k+1} $(s_1^{k+1}, s_2^{k+1}, \ldots, s_m^{k+1})$, PPOs, also called the after-capture state of the kth test cube.
- T^k (I^k, S^k), the kth test cube, composed of kth PIs and PPIs.
- G^k $(g_1^k, g_2^k, \ldots, g_t^k)$, signal lines under the capture operation defined by the kth test cube.
- fn $(\mathrm{fn}_1, \mathrm{fn}_2, \ldots, \mathrm{fn}_t)$, fanout of signal lines $1, 2, \ldots, t$.

III. WSA Models for Capture Operation

The problem in this section can be formulated as follows: Given a test cube and the logical structure of a scan circuit, determine the WSA models under capture operation.

The WSAs of these signal lines under the capture operation are important parameters that must be optimized because charging and discharging load capacitances are, by far, the most significant sources of energy consumption in digital circuits.

Because traditional WSA metric [1] can only be applied for test vectors, the probability-based WSA is used in this paper when calculating capture power for test cubes.

A. Definitions

Because state changes at signal lines depend only on the statistics for two consecutive time steps, using lag-one Markov chains [16] is adequate for the estimation of the SA.

The WSA of a signal line x is defined as follows:

$$\mathrm{WSA}(x) = \mathrm{fn}_x \times \left[p(x^t = 0) \times p(x^{t+1} = 1) + p(x^t = 1) \times p(x^{t+1} = 0)\right] \quad (1)$$

where $p(x^t = 0/1)$ is, at time t, the probability of signal line x having the 0/1 value, $p(x^{t+1} = 0/1)$ is, at time $t+1$, the probability of signal line x having the 0/1 value, and fn_x is the fanout of signal x.

1) CSA: For a given test cube T, the capture switching activity (CSA) of signal line g is defined as follows:

$$\mathrm{CSA}(g : T^k) = \mathrm{fn}_g \times \left[p(g^k = 0) \times p(g^{k+1} = 1) + p(g^k = 1) \times p(g^{k+1} = 0)\right] \quad (2)$$

where $p(g^k = 0/1)$ is the probability of signal lines g whose before-capture value is 0/1, $p(g^{k+1} = 0/1)$ is the probability of signal lines g whose after-capture value is 0/1, and fn_g is the fanout of signal g.

2) TCSA: For the kth test cube T, the total capture switching activity (TCSA) of all signal lines under capture operation is as follows:

$$\mathrm{TCSA}(T^k) = \sum_{j=1}^{t} \mathrm{CSA}(g_j : T^k). \quad (3)$$

Because WSA is used to represent power consumption, $\mathrm{TCSA}(T^k)$ represents the capture power consumption under the test cube T^k.

B. Calculating $CSA(g_j : T^k)$

The capture operation of a scan circuit can be presented as follows:

1) Before capture signal lines G

$$\begin{aligned} g_1^k &= f_{g_1}\left(i_1^k, i_2^k, \ldots, i_n^k, s_1^k, s_2^k, \ldots, s_m^k\right) \\ g_2^k &= f_{g_2}\left(i_1^k, i_2^k, \ldots, i_n^k, s_1^k, s_2^k, \ldots, s_m^k\right) \\ g_t^k &= f_{g_t}\left(i_1^k, i_2^k, \ldots, i_n^k, s_1^k, s_2^k, \ldots, s_m^k\right). \end{aligned} \quad (4)$$

2) After capture state S

$$\begin{aligned} s_1^{k+1} &= f_{s_1}\left(i_1^{k+1}, i_2^{k+1}, \ldots, i_n^{k+1}, s_1^{k+1}, s_2^{k+1}, \ldots, s_m^{k+1}\right) \\ s_2^{k+1} &= f_{s_2}\left(i_1^{k+1}, i_2^{k+1}, \ldots, i_n^{k+1}, s_1^{k+1}, s_2^{k+1}, \ldots, s_m^{k+1}\right) \\ s_m^{k+1} &= f_{s_t}\left(i_1^{k+1}, i_2^{k+1}, \ldots, i_n^{k+1}, s_1^{k+1}, s_2^{k+1}, \ldots, s_m^{k+1}\right). \end{aligned} \quad (5)$$

3) After capture signal lines G

$$\begin{aligned} g_1^{k+1} &= f_{g_1}\left(i_1^{k+1}, i_2^{k+1}, \ldots, i_n^{k+1}, s_1^{k+1}, s_2^{k+1}, \ldots, s_m^{k+1}\right) \\ g_2^{k+1} &= f_{g_2}\left(i_1^{k+1}, i_2^{k+1}, \ldots, i_n^{k+1}, s_1^{k+1}, s_2^{k+1}, \ldots, s_m^{k+1}\right) \\ g_t^{k+1} &= f_{g_t}\left(i_1^{k+1}, i_2^{k+1}, \ldots, i_n^{k+1}, s_1^{k+1}, s_2^{k+1}, \ldots, s_m^{k+1}\right). \end{aligned} \quad (6)$$

In (4) and (6), $f_{g_1}, f_{g_2}, \ldots, f_{g_t}$ are combinational logic functions of signal lines $g_1, g_2, \ldots, g_t$, respectively. In (5), $f_{s_1}, f_{s_2}, \ldots, f_{s_m}$ are sequential logic functions of SFFs $s_1, s_2, \ldots, s_m$, respectively.

The object of the LCP X-filling method is as follows. For a given test cube T^k, find the smallest possible $\mathrm{TCSA}(T^k)$.

TCSA is the sum of the CSA values of all signal lines under capture operation, as given by (3). CSA can be calculated from (2) if the before- and after-capture probabilities of every signal line are known. The before-capture probability of each signal line can be calculated from the values of I^k and S^k, which are provided by test cube T^k, based on the combinational logic functions captured in (4); the after-capture probability of each signal line can also be calculated, given the values of I^{k+1} and S^{k+1}, based on the combinational logic functions captured in (6).

Equations (4) and (6) represent combinational logic circuits. That is, in (4) and/or (6), when the probabilities of I^k, S^k and/or I^{k+1}, S^k/S^{k+1} are given, the probability of each signal line can be calculated directly using the probability calculation method for combinational signal lines [14], [15]. Once the before-capture probability (under I^k and S^k) and the after-capture probability (under I^{k+1} and S^{k+1}) have been determined, the CSA of each signal line is given by (2).

Equation (5), however, represents the state machine, in which the values of the SFFs are spatially related, as revealed in the following example.

If a state machine has states 00, 01, 10, and 11, which have state probabilities $p(00) = 1/5$, $p(01) = 1/3$, $p(10) = 2/15$, and $p(11) = 1/3$, respectively, and if $s1$ and $s2$ are the first and second state lines, respectively, in a state $(s1, s2)$, the state line probabilities can be calculated as follows:

$$\begin{aligned} p(s1 = 0) &= p(00) + p(01) = \frac{1}{5} + \frac{1}{3} = \frac{8}{15} \\ p(s1 = 1) &= p(10) + p(11) = \frac{2}{15} + \frac{1}{3} = \frac{7}{15} \\ p(s2 = 0) &= p(00) + p(10) = \frac{1}{5} + \frac{2}{15} = \frac{1}{3} \\ p(s2 = 1) &= p(01) + p(11) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}. \end{aligned}$$

Because $p(s1 = 0) \times p(s2 = 1) = 16/45$ is not equal to $p(01)$, $s1$ and $s2$ are not spatially independent.

Because state lines are spatially related, deriving the probability for state lines using the method that is the same as calculating combinational signal lines, as that in Remersaro *et al.* [12] and Wen *et al.* [13], without addressing the spatial relationships among the state lines will lead to inaccurate results.

The state line probability of the next capture state can be written as follows:

$$\begin{aligned} P\left(s_1^{k+1}\right) &= p\left(f_{s_1}\left(i_1^k, i_2^k, \ldots, i_n^k, s_1^k, s_2^k, \ldots, s_m^k\right)\right) \\ P\left(s_2^{k+1}\right) &= p\left(f_{s_2}\left(i_1^k, i_2^k, \ldots, i_n^k, s_1^k, s_2^k, \ldots, s_m^k\right)\right) \\ P\left(s_m^{k+1}\right) &= p\left(f_{s_m}\left(i_1^k, i_2^k, \ldots, i_n^k, s_1^k, s_2^k, \ldots, s_m^k\right)\right) \end{aligned} \quad (7)$$

where $p(s_i^{k+1})$ is the probability that s_i^{k+1} is one and $p(f_{s_i}(i_1^k, i_2^k, \ldots, i_n^k, s_1^k, s_2^k, \ldots, s_m^k))$ is the probability that $f_{s_i}(i_1^k, i_2^k, \ldots, i_n^k, s_1^k, s_2^k, \ldots, s_m^k)$ is one.

Because the steady-state probabilities of SFFs are constant [16], the solution to (7), which is a nonlinear system, yields the required probabilities of the state lines.

Equation (7) can be solved by using the Picard–Peno iteration method [16]. The detailed steps are the following: Begin from the initial probability of $(p(S^{k+1}))^0 = p(S^k)$ and recursively compute (7) until $(p(s_i^{k+1}))^{r+1} - (p(s_i^{k+1}))^r$ is sufficiently small. In practice, it is observed that with three or four recursions ($r = 2$ or 3) or so, good results can be acquired. After the recursions are finished, the probabilities of the state lines are obtained.

In summary, CSA($g : T^k$), including both SFFs and combinational signal lines, can be calculated using the following steps:

1) Assign probabilities to I^k and S^k according to Table I. Simply, we can use $p(g^k)$ to represent $p(g^k = 1)$ and $1 - p(g^k)$ to represent $p(g^k = 0)$.
2) Calculate after-capture state probability $p(S^{k+1})$ by (5) using the Picard–Peno iteration method with $p(I^k)$ and $p(S^k)$ as inputs.
3) Calculate the before-capture signal line probabilities $p(g_1^k), p(g_2^k), \ldots, p(g_t^k)$ by (4) using the probability calculation method for combinational signal lines introduced in [14] and [15], given $p(I^k)$ and $p(S^k)$ as inputs.
4) Calculate the after-capture signal line probabilities $p(g_1^{k+1}), p(g_2^{k+1}), \ldots, p(g_t^{k+1})$ by (6) using the probability calculation method for combinational signal lines introduced in [14] and [15], given $p(I^{k+1})$ and $p(S^{k+1})$ as inputs.
5) Calculate CSA($g_j : T^k$), $1 \leq j \leq t$, of all signal lines using (2) based on the before- and after-capture signal probabilities acquired in steps 3) and 4).

C. Lowest TCSA(T^k)

The LCP X-filling problem is as follows. For a given T^k, find values (0 or 1) for X-bits in I^k, S^k, and I^{k+1} that minimize the TCSA(T^k).

Clearly, the smallest TCSA(T^k) can be obtained when each signal line has its smallest WSA

$$\text{Min}\left(\text{TCSA}(T^k)\right) = \sum_{j=1}^{t} \text{Min}\left(\text{CSA}(g_j : T^k)\right). \quad (8)$$

The CSA($g_j : T^k$) of each signal line is the smallest if the Hamming distance between its before- and after-capture values is the shortest.

To minimize the TCSA(T^k), the following relations must be maintained:

$$I^k = I^{k+1}, \quad \text{for all bits in } I^k$$

$$S^k = S^{k+1}, \quad \text{for all X-bits in } S^k.$$

The LCP X-filling problem can be rewritten as follows. For a given T^k, which includes I^k and S^k, find values (0 or 1) for X-bits in I^k and S^k that minimize the TCSA(T^k).

Because X-bits in S^k can be filled after S^{k+1} values are acquired, the LCP X-filling method, thus far, can be rewritten as follows.

1) Find a set of X-bits in PIs that yields the smallest TCSA(T^k); fill the X-bits in I^k.
2) Calculate S^{k+1} using (7) with the filled I^k and given S^k.
3) Fill the X-bits in S^k by using the probabilities of S^{k+1}.

IV. Proposed LCP X-Filling Methods

The LCP X-filling problem can be summarized as follows. For a given T^k, determine the filling I^k and S^k that yield the smallest TCSA(T^k). Because X-bits in S^k can be filled based on the steady probability of S^{k+1} once I^k is given, setting I^k is the only controllable way to obtain the smallest TCSA(T^k).

A. Enumeration Selection Method

Obviously, choosing the smallest TCSA(T^k) equals to choosing the I^k that leads to it; therefore, all the possible combinations of X-bits in I^k can be exhaustively explored to obtain the smallest TCSA(T^k).

TABLE I
Probability Assignments for I^k and S^k

The value of signal g	$P(g^k = 1)$	$P(g^k = 0)$
X	0.5	0.5
0	0	1
1	1	0

This method works well for small circuits, but it is not efficient for circuits with a large number of X-bits in I^k. Thus, an approximate method that can fill X-bits in I^k both effectively and efficiently needs to be explored.

B. SSF Method

In filling the X-bits in PIs, the following principles should be applied. First, X-bits in PIs are state dependent; filling individually would compromise the filling effectiveness. An effective method should be state based. Second, an effective filling method should guarantee that TCSA(T^k) is as small as possible.

In the following, a novel SSF method is proposed according to the two aforementioned principles. Two new terms must be introduced before the details of this new SSF method can be explained.

1) PP: PP($i_j^X : T^k$) is a newly defined probability for an X-bit in $I^k(i_j^X, 1 \leq j \leq n)$. It utilizes the relation between a particular filling value and its resulting TCSA($i_j^X : T^k$) to generate the probability of i_j^X being filled with a particular value of 0 or 1.

PP($i_j^X = 1 : T^k$), which is simplified as PP($i_j^X : T^k$), is derived as follows:

1) Calculating TCSA($i_j^X = 0/1 : T^k$): Assign the I^k and S^k as follows: set $i_j^X = 0$ or 1; set other bits in I^k and S^k according to Table I; compute TCSA($i_j^X = 0 : T^k$) and TCSA($i_j^X = 1 : T^k$) according to (3).
2) Calculating PP(i_j^X): For each X-bit in I^k, define

$$\text{PP}\left(i_j^X : T^k\right) = \frac{\text{TCSA}\left(i_j^X = 0 : T^k\right)}{\text{TCSA}\left(i_j^X = 0 : T^k\right) + \text{TCSA}\left(i_j^X = 1 : T^k\right)}. \quad (9)$$

A similar formula could be derived for PP($i_j^X = 0 : T^k$), which is simplified as PP($\overline{i_j^X} : T^k$) or $1 -$ PP($i_j^X = 0 : T^k$).

2) Potential PI: I^k with X-bits initialized with their related potential probability (PP) values. The general procedure of the SSF method is described as follows:

1) Calculating PP for all X-bits in I^k.
2) Calculating TCSA($i_j^X = 0/1 : T^k$(PP)): Assign the I^k as follows: set $i_j^X = 0$ or 1; set other bits in I^k with their PP values, and set S^k according to Table I; compute TCSA($i_j^X = 0 : T^k$(PP)) and TCSA($i_j^X = 1 : T^k$(PP)) according to (3).

   The difference between TCSA($i_j^X = 0/1 : T^k$) and TCSA($i_j^X = 0/1 : T^k$(PP)) is that for all these X-bits except the jth X-bit, TCSA($i_j^X = 0/1 : T^k$) uses the probability values given in Table I, whereas TCSA($i_j^X = 0/1 : T^k$(PP)) uses the PP values. TCSA($i_j^X = 0/1 : T^k$) is the TCSA under original test cube T^k with its jth X-bit with a value of 0/1; TCSA($i_j^X = 0/1 : T^k$(PP)) is the TCSA under potential test cube T (PP) with its jth X-bit with a value of 0/1.
3) Selecting filling values: The final filling value for i_j^X is selected by the following:

$$X = \begin{cases} 1, & \text{if TCSA}(1) < \text{TCSA}(0) \\ 0, & \text{if TCSA}(0) \leq \text{TCSA}(1). \end{cases} \quad (10)$$

TABLE II
TEST CUBE INFORMATION

Circuits	Num. of test vec.	Num. of PIs(n)	Num. of PPIs(m)	Fault cov.(%)	X-ratio(%)
$s1196$	115	14	18	100	23.07
$s1238$	125	14	18	95	34.00
$s1423$	24	17	74	99	36.45
$s5378$	101	35	179	99	74.01
$s9234$	108	36	211	94	68.95
$s13207$	236	62	638	99	90.41
$s15850$	94	77	534	97	74.91
$s35932$	13	35	1728	90	45.53
$s38417$	87	28	1636	100	76.84
$s38584$	118	38	1426	96	82.29

TABLE III
CPU TIME OF VARIOUS LCP X-FILLING METHODS (SECONDS)

Circuits	Santiago[12]	Wen[13]	SSF	Circuits	Santiago[12]	Wen[13]	SSF
$s1196$	0	1.6	0.2	$s13207$	0	9590.3	840.3
$s1238$	0	1.9	0.3	$s15850$	0	2964.4	250.1
$s1423$	0	2.0	0.4	$s35932$	0.1	1109.3	33.3
$s5378$	0	341.4	59.6	$s38417$	0.1	18901.3	242.1
$s9234$	0	744.6	111.3	$s38584$	0.1	26801.4	631.6

In the aforementioned equation

$$\begin{cases} \text{TCSA}(1) = \text{TCSA}\left(i_j^X = 1 : T^k(\text{PP})\right) \\ \text{TCSA}(0) = \text{TCSA}\left(j_j^X = 0 : T^k(\text{PP})\right). \end{cases}$$

The advantage of using the SSF method is that, when choosing the value of a particular X-bit in I^k, the remaining X-bits are assigned with state-based tight probabilities, thus yielding an effective filling value.

C. Filling X-Bits in S^k

X-bits in S can be filled by using the steady probability of S^{k+1}.

The probabilities of S^{k+1} can be calculated after the X-bits in I^k are filled. With X-bits in S^k that are set as that in Table I, the probabilities of S^{k+1} can be computed by solving (7) using the Picard–Peno iteration method.

From the probability point of view, the best X-filling happens if the filled value of X can make

$$s_j^k = \begin{cases} 1, & \text{if } p\left(s_j^{k+1}\right) > 0.5 \\ 0, & \text{if } p\left(s_j^{k+1}\right) \leq 05 \end{cases}, \qquad 1 \leq j \leq t.$$

The complete procedure of the SSF method is summarized as follows.

1) Given test cube T^k, which is composed of I^k and S^k.
2) Calculate PP for each X-bit in I^k.
3) Calculate TCSA$(i_j^X = 0 : T^k(\text{PP}))$ and TCSA$(i_j^X = 1 : T^k(\text{PP}))$ with i_j^X with 0 and 1, other X-bits in I^k with their PP values, and X-bits in S^k with the given values. Choose i_j^X as the one that results in a smaller TCSA.
4) Calculate S^{k+1} using the filled I^k and given S^k.
5) Fill X-bits in S^k with S^{k+1} values.

D. Computation Complexity of SSF Method

In the SSF method, X-bits in I^k are filled one by one based on the potential I^k, X-bits in S^k are filled using steady state probabilities of S^{k+1}.

If t_s is the time required to calculate the steady-state probability in a scan circuit, n_X is the number of X-bits in the PIs, m_X is the number of X-bits in PPIs, and T_{SSF} is the time required to complete the SSF process, then

$$T_{\text{SSF}} = (2n_x + 1) \times t_S \tag{11}$$

which means that filling each X-bit in I^k needs time $2t_S$ and filling all X-bits in S^k needs time t_S.

For comparison, the computation times in [12] and [13], which are given by $T_{[12]}$ and $T_{[13]}$, respectively, are as follows:

$$T_{[12]} = t_S \tag{12}$$

$$T_{[13]} = 2(n_X + m_X)t_S. \tag{13}$$

In (12), almost no time is needed to fill I^k because the solution proposed by [12] concerns a two-pattern launch off test. The Hamming distance between PI_1 and PI_2 is minimized by first filling the unspecified values in $\text{PI}_1(\text{PI}_2)$ to match the specified values in $\text{PI}_2(\text{PI}_1)$. After the first step, all of the remaining unspecified values in PI_1 and PI_2 are at the same positions. Then, in the second step, randomly fill these values to have the same specified value.

In (13), filling each X-bit in both I^k and S^k requires time $2t_S$.

The efficiency of the SSF method is quite high because for a large circuit, the number of PIs is often negligible in comparison with the number of PPIs: $n_X \ll m_X$.

V. EXPERIMENT RESULTS

The proposed algorithm was implemented in C language and run on ten ISCAS'89 benchmark circuits using a Sun Ultra 5/440 machine with 512-MB memory. Information about test cubes provided by the authors in [11] and [13] is listed in Table II.

Table III presents the CPU time for the proposed SSF technique and other LCP X-filling methods. It can be seen that the computational time of the SSF method is quite small.

Table IV compares the results of the proposed SSF method and the state-of-art LCP X-filling methods introduced in [12] and [13]. Among these three methods, SSF has the best performance both in average and peak power reduction.

In Table IV, TCSA is measured on per test vector basis; the average TCSA represents the average TCSA over the test vectors, and the peak CSA represents the highest TCSA among the test vectors. In Table IV, the results of [13] are provided by their authors; the result of [12] is implemented again based on the scan capture test using the WSA defined herein.

TABLE IV
CSA REDUCTION

	Average CSA			Peak CSA		
Circuits	Santiago[12]	Wen[13]	SSF	Santiago[12]	Wen[13]	SSF
$s1196$	38	41	18	89	94	65
$s1238$	39	45	18	90	105	54
$s1423$	377	384	259	527	561	425
$s5378$	1296	1524	705	1853	1879	1345
$s9234$	2841	2749	1278	3381	3368	2306
$s13207$	2553	3835	1659	3716	4908	3210
$s15850$	2001	3129	1220	3656	4111	2604
$s35932$	9042	9239	7763	14853	15456	13167
$s38417$	9707	10637	7486	12411	12685	10525
$s38584$	6461	7624	4631	11862	13527	8820

VI. CONCLUSION

This paper presents a novel X-filling approach, which is the SSF scheme. Unlike previous works that do not address the spatial relationship among state lines, the proposed SSF method first obtains the potential vector sets and then selects the filling values using a selecting method that minimizes the number of gate transitions. The benefit of the proposed approach is that it retains the spatial relation of state lines, thus guaranteeing the quality of the filling results. Another benefit is that all the states are filled in parallel, yielding a short execution time. Experimental results indicate that both average and peak capture power consumptions were significantly reduced and computational cost was small.

ACKNOWLEDGMENT

The authors would like to thank Dr. K. Miyase and Dr. X. Wen from Kyushu Institute of Technology for providing the test cubes used in the experiment and the editors and reviewers of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS for their opinions and suggestions.

REFERENCES

[1] P. Girad, "Survey of low-power testing of VLSI circuits," *IEEE Des. Test Comput.*, vol. 19, no. 3, pp. 82–92, May/Jun. 2002.
[2] T. Yoshida and M. Watari, "A new approach for low power scan testing," in *Proc. Int. Test Conf.*, 2003, pp. 480–487.
[3] R. Chou, K. Saluja, and V. Agrawal, "Scheduling tests for VLSI systems under power constraints," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 5, no. 2, pp. 175–185, Jun. 1997.
[4] S. Wang and S. Gupta, "ATPG for heat dissipation minimization during test application," *IEEE Trans. Comput.*, vol. 47, no. 2, pp. 256–262, Feb. 1998.
[5] F. Corno, P. Prinetto, M. Redaudengo, and M. Reorda, "A test pattern generation methodology for low power consumption," in *Proc. VLSI Test Symp.*, 2000, pp. 35–40.
[6] S. Kajihara, K. Ishida, and K. Miyase, "Test vector modification for power reduction during scan testing," in *Proc. VLSI Test Symp.*, 2002, pp. 160–165.
[7] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S. Reddy, "Techniques for minimizing power dissipation in scan and combinational circuits during test application," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 12, pp. 1325–1333, Dec. 1998.
[8] K.-J. Lee, T.-C. Huang, and J.-J. Chen, "Peak-power reduction for multiple-scan circuits during test application," in *Proc. Asian Test Symp.*, 2000, pp. 453–458.
[9] R. Sankaralingam, R. Oruganti, and N. Touba, "Reducing power dissipation during test using scan chain disable," in *Proc. VLSI Test Symp.*, 2001, pp. 319–324.
[10] R. Sankaralingam and N. A. Touba, "Controlling peak power scan testing," in *Proc. VLSI Test Symp.*, 2002, pp. 153–159.
[11] X. Wen, Y. Yamashita, S. Kajihara, L.-T. Wang, K. K. Saluja, and K. Kinoshita, "On low-capture-power test generation for scan testing," in *Proc. VLSI Test Symp.*, 2005, pp. 265–270.
[12] S. Remersaro, X. Lin, Z. Zhang, and S. M. Reddy, "Preferred fill: A scalable method to reduce capture power for scan based designs," in *Proc. Int. Test Conf.*, 2006, pp. 1–10.
[13] X. Wen, K. Miyase, T. Suzuki, Y. Yamato, S. Kajihara, L.-T. Wang, and K. K. Saluja, "A highly-guided x-filling method for effective low-capture-power scan test generation," in *Proc. ICCD*, 2006, pp. 251–258.
[14] K. P. Parker and E. J. McCluskey, "Probability treatment of general combinational networks," *IEEE Trans. Comput.*, vol. C-24, no. 6, pp. 668–670, Jun. 1975.
[15] A. Papoulis, *Probability, Random Variables and Stochastic Processes*, 3rd ed. New York: McGraw-Hill, 1991.
[16] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. Boston, MA: Academic, 1970.

A Compositional Method With Failure-Preserving Abstraction for Asynchronous Design Verification

Hao Zheng, Jared Ahrens, and Tian Xia

***Abstract*—This paper presents a compositional method with failure-preserving abstraction for scalable asynchronous design verification. It combines efficient state-space reductions and novel interface refinement and can dramatically reduce the complexity of state space while decreasing the introduction of false failures. This allows much larger designs to be verified as demonstrated in the experimental results.**

***Index Terms*—Abstraction, asynchronous, compositional, formal verification, model checking, refine.**

I. INTRODUCTION

Compositional methods are essential to address state explosion in model checking. A compositional minimization method is described in [7], where the global minimized state transition system is built by iteratively minimizing and composing the processes in a finite-state system. To contain the size of the intermediate results, user-provided context constraints are required. This may be a problem in that the state space may be large in the first place. The requirement of user-provided context constraints may also be a problem in that the constraints may be overly restrictive, thus resulting in the escape of real design errors. Similar work is described in [2].

In general, compositional approaches need an approximate environment for each module of a design under consideration. This approximate environment should be simple and relatively accurate. However, coming up with such an environment is a daunting task and is traditionally done by hand. Lately, some automated approaches [1], [3], [4], [6] based on machine learning are proposed to generate environment assumptions for compositional reasoning. Basically, assumptions are generated for a module of a design to eliminate the counterexamples of that module. Next, assumptions are validated by checking the rest of the design.

Manuscript received July 17, 2007; revised January 25, 2008. This work was supported in part by a grant from the National Institute for Systems Test and Productivity, University of South Florida, and in part by the National Science Foundation under CAREER Award CCF 0546492 and Award CNS 0551621. This paper was recommended by Associate Editor W. Kunz.

H. Zheng is with the Department of Computer Science and Engineering, College of Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: zheng@csee.usf.edu).

J. Ahrens was with the Department of Computer Science and Engineering, College of Engineering, University of South Florida, Tampa, FL 33620 USA.

T. Xia is with the Department of Electrical and Computer Engineering, University of Vermont, Burlington, VT 05405 USA.

Digital Object Identifier 10.1109/TCAD.2008.923104