Efficient Trace Signal Selection for Silicon Debug by Error Transmission Analysis

Joon-Sung Yang, *Member, IEEE,* and Nur A. Touba, *Fellow, IEEE*

***Abstract*—In this paper, a technique is presented for selecting signals to observe during silicon debug. Internal signals are used to analyze, understand, and debug circuit misbehavior. An automated procedure to select which signals to observe is proposed to facilitate early detection of circuit malfunction and to enhance the utilization of hardware resources for storage. Signals that are most often sensitized to possible errors are observed in sequential circuits. Given a functional input vector set, an error transmission matrix is generated by analyzing which flip-flops are sensitized to other flip-flops. Relatively independent flip-flops are identified and a set of signals that maximally cover the possible error sites with given constraints are identified through integer linear programming. Experimental results show that the proposed approach can rapidly and precisely identify the nonconforming chip behavior and thereby can speed up the post-silicon debug process.**

***Index Terms*—Error propagation matrix, integer linear programming, signal observability, silicon debug.**

I. Introduction and Related Work

THE ADVANCE of technology allows sophisticated designs with millions of transistors. Due to the complex designs and the inaccuracies in modeling integrated circuits (ICs) along with process variations during the manufacturing process, post-silicon validation (silicon debug) takes more than half of the chip development cycle [2], [17], [21]. Unlike during pre-silicon verification, the accessibility and visibility of internal signals are very limited in post-silicon debug and hence this is the major challenge in the validation and debug of first silicon. Hence, identifying and resolving problems in ICs after first silicon is a very time consuming and extremely complex task [6], [8], [9], [12]–[14], [17], [20], [22], [23].

The narrow observability of internal signals makes silicon debug costly and time consuming. For this reason, design-for-debug (DFD) techniques are introduced to enhance the silicon validation process. Two main hardware DFD solutions, scan chains and trace buffers, are proposed to add debug support for the more speedy and accurate process. These hardware features are used to enhance the data acquisition of internal signals. The captured data is used to investigate the temporal and spatial failure information by software-based debug tools [2], [10], [14], [24]. Scan chains are widely utilized to support manufacturing test as a design-for-testability feature. Scan-based debug techniques [9], [18], [19] significantly enhance the internal signal observability, however, the system needs to be halted to read out responses from the circuit-under-debug. A trace buffer is an on-chip memory that can store the continuous internal signal information for a limited time period.

Increased internal signal observability helps to discover erroneous behavior closer to the source of the problem, both in space and time. Previous research has been done to enhance internal observability in two ways.

1) Efficient use of limited hardware resources: data compression techniques have been applied to increase the volume of debug data [3], [4], [22].
2) Extra data extraction from captured trace data: some studies [1], [10], [24] use inductive reasoning to determine additional signal values from the captured debug data.

Reference [4] proposed a multiple input signature register based technique which implements an accelerated binary search. And [22] introduced a 2-D compaction that captures debug data in a trace buffer only during clock cycles in which errors are present. In [3], lossless compression methods based on dictionary coding were investigated for compressing the data stored in a trace buffer.

To extract more debug data than captured, [1] and [10] proposed data construction techniques via Boolean equations. And [13] introduced signal restorability for data reconstruction in sequential circuits and signals with high restorability are chosen. References [15] and [25] used scan dump values and find the failing registers by back-tracing and forward-tracing methods. Reference [16] showed an architectural level approach for post-silicon bug localization. It records the history of the program executed and identifies the bug location-time information at the system level.

In this paper, a systematic method for selecting the appropriate signals to observe is proposed to maximize the effectiveness of limited internal signal observability. The proposed method exploits the nature of error propagation in sequential circuits by observing signals which are most often sensitized to possible error sites. Integer linear programming (ILP) is used to determine the set of signals to observe which are easily sensitized by the possible errors with a given condition. Preliminary results were presented in [23].

II. Details of Signals to Observe Selection Process

The following subsections describe each of the steps in the proposed procedure for selecting the signals to observe.

A. Fault Simulation and Error Transmission Matrix Generation

A simple circuit example is shown in Fig. 1. There are six flip-flops represented by rectangles named A to F and combinational logic illustrated as a cloud. To show an example of a fault simulation, we assume that three functional vectors (v_1, v_2, and v_3) are applied to this logic. When the vectors are applied, if there is a bug, the erroneous response could be captured in some flip-flops at some time. That faulty response would likely keep propagating in a sequential circuit over

Manuscript received June 28, 2011; accepted September 6, 2011. Date of current version February 17, 2012. This research was supported in part by the National Science Foundation, under Grant CCF-0916837. Date of current version February 17, 2012. This paper was recommended by Associate Editor S. S. Sapatnekar.

J.-S. Yang is with Intel Corporation, Austin, TX 78746 USA (e-mail: joon.yang@intel.com).

N. A. Touba is with the Computer Engineering Research Center, Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712 USA (e-mail: touba@ece.utexas.edu).

Digital Object Identifier 10.1109/TCAD.2011.2171184

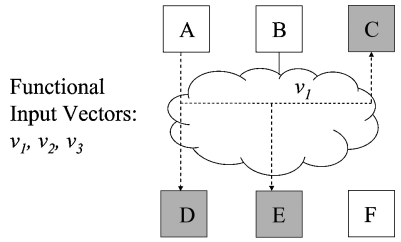

Fig. 1. Simple logic example.

multiple cycles. In this paper, however, we set the propagation depth as *1* and investigate the error propagation. Hence, a fault simulation is performed for one cycle to see where the error propagates and this can identify which flip-flops are corrupted by an error. As highlighted in gray in Fig. 1, the fault simulation shows the error propagation from *A* to flip-flops *C*, *D* and *E* for input vector v_1. In the same manner, the fault simulation can be performed for each test vector and errors.

Fault simulation identifies the error transmission and the results can be represented as a matrix. As shown in Fig. 2(a), each column represents a flip-flop *(A ∼F)* in a circuit, and each row shows the error information pair with an input vector and a flip-flop. The input vector and the erroneous flip-flop pair is represented as (A, v_1) in the first row of the matrix. For example, (A, v_1) tells an error in a flip-flop *A* with a vector v_1. The fault simulation shows that an error located in *A* with a vector v_1 propagates to *C*, *D* and *E*. Because *C*, *D* and *E* are influenced by (A, v_1), each have a "*1*" in the first row.

Once the error transmission matrix is generated, the flip-flops that are most often sensitized to possible errors can be identified assuming bugs in silicon are modeled as occurring evenly distributed in time and space because random errors can happen and cause issues in silicon anytime and anywhere. Note that an error will likely propagate for multiple clock cycles and need not necessarily be detected in the first cycle in which it occurs. Since the value "*1*" is assigned when the flip-flop value is corrupted by corresponding error pairs, the columns in the error transmission matrix with the most "*1*"s are probabilistically more likely to capture errors over time. This shows that errors are most frequently transmitted to them. Hence, we can find the columns with the most "*1*"s and they are candidates for signals to observe for better observability. Moreover, if a limited set of signals to observe is to be selected, then columns that are most nonoverlapping and cover as many rows as possible are also more likely to cover more errors. This will be discussed in more detail later.

B. Enhancement of Internal Signal Observability by Relatively Independent Flip-Flop Analysis

Here, we analyze the error transmission matrix to find independent flip-flops. Flip-flops are relatively independent if a single error in a circuit will not influence them simultaneously. If two flip-flops are relatively independent, the erroneous response for one error will not be simultaneously transmitted to both flip-flops. For the example in Fig. 2(a), since an error in *A* is transmitted to *C*, *D*, and *E* for vector v_1 and the values in flip-flops *A*, *B* and *F* are not influenced by an error. Hence,

	A	B	C	D	E	F
(A, v_1)	0	0	1	1	1	0
(B, v_1)	1	0	0	0	1	0
(C, v_1)	1	1	0	0	0	0
(D, v_1)	0	0	0	0	0	0
(E, v_1)	0	0	0	0	1	0
(F, v_1)	0	1	1	0	1	0
(A, v_2)	0	0	0	0	0	1
(B, v_2)	1	0	0	1	1	0
(C, v_2)	0	1	0	0	0	0
(D, v_2)	1	0	0	0	0	0
(E, v_2)	0	1	1	0	0	0
(F, v_2)	0	0	0	0	1	0
(A, v_3)	0	1	0	0	1	0
(B, v_3)	0	0	0	0	0	1
(C, v_3)	1	1	0	0	1	0
(D, v_3)	0	0	0	0	0	0
(E, v_3)	0	0	0	1	0	1
(F, v_3)	0	0	0	0	1	0

(a)

	S_0	S_1	S_2
R_0	1	1	1
R_1	1	0	1
R_2	1	1	0
R_3	0	0	0
R_4	0	0	1
R_5	1	1	1
R_6	1	0	0
R_7	1	1	1
R_8	0	1	0
R_9	1	0	0
R_{10}	1	1	0
R_{11}	0	0	1
R_{12}	0	1	1
R_{13}	1	0	0
R_{14}	1	1	1
R_{15}	0	0	0
R_{16}	1	1	0
R_{17}	0	0	1

(b)

Fig. 2. (a) Error transmission matrix. (b) Updated error transmission matrix by relatively independent flip-flop analysis.

A, *B*, and *F* are relatively independent of the possible error because they are influenced by an error in *A* with v_1.

Because the relatively independent flip-flops are not affected by the same error, they can be XORed together to increase the overall observability of the internal signals. The error transmission matrix can be updated by forming signal groups by combining (XORing) the relatively independent flip-flops in the matrix. For example, because the error (A, v_1) never propagates to *A*, *B* and *F*, XORing *A*, *B*, and *F* does not affect the error detection of (A, v_1). Relatively independent flip-flops in a circuit are identified and XORed to achieve better observation capability. The overall goal is to find misbehavior as early as possible, so observing more signals helps silicon debug by providing more internal signal information.

As explained, the relatively independent flip-flops can be XORed together without losing error observation for single flip-flop errors. Note, however, that flip-flops are relatively independent only with respect to single errors, so it is still possible for multiple errors to cancel. However, this serves as a good heuristic for increasing overall signal observability.

In Fig. 2(a), relative independence is analyzed among flip-flops (*A∼F*) and three relatively independent signal groups can be found. Therefore, the first signal group (S_0), the second group (S_1), and the last group (S_2) can be expressed respectively as follows.

Three Signal Groups Identified by Relatively Independent Flip-Flop Analysis:

$$S_0 = A \oplus C \oplus F, S_1 = B \oplus D, S_2 = E.$$

Once signal groups are identified by the relatively independent flip-flop analysis, the error transmission matrix can be updated. Fig. 2(b) shows the matrix update in Fig. 2(a). (A, v_1) to (F, v_3) error information pairs are indexed by R_0 to R_{17}. Instead of individual signal selection, signal groups are selected and this provides better signal observability.

$$max: R_0 + R_1 + R_2 + ... + R_{15} + R_{16} + R_{17} \quad (1)$$

$$s.t: S_0 + S_1 + S_2 = num.\ of\ signal\ groups\ to\ observe \quad (2)$$

$$\left.\begin{array}{r} S_{0\_0} + S_{0\_1} + S_{0\_2} \geq R_0 \\ S_{1\_0} + S_{1\_2} \geq R_1 \\ \vdots \\ S_{16\_0} + S_{16\_1} \geq R_{16} \\ S_{17\_2} \geq R_{17} \end{array}\right\} \quad (3)$$

$$R_0, R_1, R_2, \ ..., R_{15}, R_{16}, R_{17} \in \{0, 1\} \quad (4)$$

$$S_0, S_1, S_2 \in \{0, 1\} \quad (5)$$

Fig. 3. ILP formulation for updated error transmission matrix.

Since the relatively independent flip-flops are XORed, there may be concern about hardware implementation issues. Because a number of signals are XORed together, the signal groups need to be identified taking the number of signals in one signal group into consideration. If there are too many signals in one signal group, an XOR tree is needed and congested routing/wiring could be an issue. Hence, a limit can be placed on the number of signals which are XORed together when the error transmission matrix is updated to minimize delay and/or routing.

C. Integer Linear Programming to Determine the Set of Signals to Observe

As stated in Section I, since there are limitations on storage, bandwidth and overhead for observing signals, it is very important to choose the best set of signals to observe for the most efficient debugging. Linear programming is a well-known technique used for a wide range of optimization problems in many fields. Linear programming with integer unknown variables is called an ILP. Here ILP is employed to select an optimal set of signals based on the updated error transmission matrix. The matrix in Fig. 2(b) is formulated as a set of ILP equations in Fig. 3.

In Fig. 3, R_0 denotes the 0th row in the updated error transmission matrix and S_0 represents the 0th column. Since we want to enhance the debugging capability, the objective function of this ILP problem is to maximize the number of errors covered by a set of signal groups. Hence, the objective equation ("maximize the covered errors") is expressed as the summation of the entire error sets *(1)*. In the ILP solution, if an error is covered by any signal group, then the value *"1"* is assigned to a corresponding error variable (R_k). And if an error is not covered, *"0"* will be assigned. Therefore, the solution space for R_k is *"0"* or *"1" (4)*. Depending on the bandwidth and storage requirements of hardware resources, the number of signal groups can be decided. The solution space for signal groups is also *"0"* or *"1" (5)*. When a signal group is selected for observation, S_i has *"1"*, otherwise, *"0"* is assigned. Because the number of signal groups to observe is determined by the amount of hardware available, the summation of S_k equals to the number of signal groups to observe *(2)* and they are always larger than *1* (i.e. $S_0 + S_1 + S_2 \geq 1$). More ILP constraints are derived from each row of Fig. 2(b) and they are represented as equations *(3)*. S_{k-i} is defined as a signal group in kth row and ith column to differentiate the variables between rows. For example, three signal groups in the first row are expressed as S_{0-0}, S_{0-1} and S_{0-2} and the second row signal groups are denoted as S_{1-0}, S_{1-1} and S_{1-2}. In the first row, R_0 is always detected by S_{0-0}, S_{0-1} or S_{0-2}, hence, the ILP formulation from the first row can be expressed as $S_{0-0} + S_{0-1} + S_{0-2} \geq R_0$. In the same manner, from the second row, the constraint can be driven as $S_{1-0} + S_{1-1} + S_{1-2} \geq R_1$. Because the signals S_1 cannot detect R_1 in the second row, the equation can be written as $S_{1-0} + S_{1-2} \geq R_1$. In this same manner, 18 ILP formulations are generated in *(3)*.

TABLE I
NUMBER OF SIGNALS OBSERVED BY PROPOSED METHOD FOR NOC DESIGN

Pattern Set	Threshold Values	Number of Signal Groups			
		8	12	16	32
Using Selective Patterns	8	64	93	126	256
	12	92	134	184	379
	∞	367	921	1169	1543
Using Entire Patterns	8	64	96	121	256
	12	96	144	189	380
	∞	507	829	1296	1637

III. TECHNIQUES TO GENERATE ERROR TRANSMISSION MATRIX

The performance of the ILP solver depends on the number of variables in the constraints. To avoid excessive run time overhead by an ILP solver, we show two ways to deal with an error transmission matrix and these are used for experimental setups in Section IV.

One way to reduce the number of variables or equations is to use less number of functional vectors. As shown in Fig. 4(a), some important vectors or representing vectors can be used for fault simulation and error transmission matrix generation. Commonly, since random instructions are run to exercise various part of a chip [6], [18], [19], some random instructions could be a good choice for sample vector based approach.

The other method uses the entire functional vectors, however, the error transmission matrix is partitioned and set smaller matrices are used for debug signal selection. As illustrated in Fig. 4(b), fault simulation is performed with a full suite of test vectors and the error transmission matrix is generated. Since the matrix size is big, it is split into multiple pieces of smaller matrices to reduce the overhead of solving a big matrix by ILP. A set of signal groups are determined for each matrix. The results from each matrix are analyzed and the signal groups which are mostly found are determined as final signal groups. Experimental results for both techniques are shown in Section IV.

IV. EXPERIMENTAL RESULTS

In this section, experimental results are presented for ISCAS-89 benchmark circuits [5] and an network-on-chip (NoC) design [11]. Stuck-at faults were randomly injected in

TABLE II
AVERAGE ERRONEOUS RESPONSE DETECTION LATENCY RESULTS

(a) Random/Structural

Circuit	Random/Structural	Average Detection Latency with Different Number of Signal Groups						
		8	12	16	20	24	28	32
s9234	Random	320.81	212.18	186.82	180.13	175.10	174.9	173.73
	Structural	244.28	197.71	176.82	175.91	173.01	173.01	173.01
s38584	Random	197.56	184.46	131.85	124.68	118.80	115.31	109.87
	Structural	178.08	146.73	127.32	121.77	119.31	118.29	115.86
NoC Design	Random	594.42	571.87	574.82	562.69	581.36	532.13	504.65
	Structural	643.26	551.66	541.23	532.78	519.95	506.11	492.24

(b) Using Selective Vector Patterns

Circuit	Threshold Value	Average Detection Latency with Different Number of Signal Groups						
		8	12	16	20	24	28	32
s9234	8	49.36	41.84	41.06	33.84	20.52	20.52	20.52
	12	49.36	41.84	41.06	33.84	20.52	20.52	20.52
	∞	49.36	41.84	41.06	33.84	20.52	20.52	20.52
s38584	8	67.35	61.44	51.87	46.97	41.78	35.25	31.41
	12	59.42	54.13	49.04	31.59	28.98	25.91	19.67
	∞	10.24	6.62	4.58	2.46	1.39	0.35	0.34
NoC Design	8	201.56	153.68	124.08	117.42	110.28	97.36	68.37
	12	148.97	117.79	109.80	94.36	62.71	51.96	45.21
	∞	47.63	21.79	16.37	14.08	12.83	10.52	9.12

(c) Using Entire Vector Patterns

Circuit	Threshold Value	Average Detection Latency with Different Number of Signal Groups						
		8	12	16	20	24	28	32
s9234	8	45.47	41.26	41.06	33.84	20.52	20.52	20.52
	12	45.47	41.26	41.06	33.84	20.52	20.52	20.52
	∞	45.47	41.26	41.06	33.84	20.52	20.52	20.52
s38584	8	63.68	55.17	50.20	42.43	39.19	32.25	27.86
	12	50.75	42.26	39.44	29.80	23.13	20.66	16.62
	∞	8.39	5.67	2.83	1.98	0.84	0.31	0.26
NoC Design	8	187.32	141.69	114.20	103.36	98.42	76.34	61.44
	12	139.36	105.97	87.26	67.24.	48.90	39.76	33.96
	∞	42.55	19.36	13.25	11.93	9.98	8.54	7.34

circuits to produce misbehavior of the systems and to generate erroneous data. Fault simulation for one depth is performed to investigate the error transmission using random vectors for ISCAS-89 benchmarks and deterministic verification vectors for NoC design.

As discussed in Section II-B, while the error matrix is updated by finding the independent flip-flops, the number of signals which can be merged for one signal groups can be limited to void issues related to the physical design such as timing and wiring by signal groups. Three threshold values (8, 12, and ∞) are set to see how fast the circuit misbehavior is detected with different signal numbers. To find the final set of signals to observe, GNU Linear Programming Kit (GLPK) 4.32 [7] was used as the ILP solver.

Table I shows how many signals are observed with each of three threshold values for a single signal group in updating the error transmission matrix by the relatively independent flip-flop analysis. For the NoC design, there are 1991 flip-flops in the design. Table I shows the debug signal selection results with the partial functional vector set and the signal selection results with the entire vectors set. The first column shows the pattern set. The second column shows the maximum number of signals that can be merged into one signal group when updating the error transmission matrix which is referred to as *threshold values*. In the third column, the number of flip-flops observed is shown when 8, 12, 16, and 32 signal groups are chosen by ILP in the updated error transmission matrix. As can be seen from the results, more flip-flops can be observed as the maximum value in the second column is increased.

As can be seen from Table I, the number of signals to observe is not determined by the number of functional vectors used to generate the error transmission matrix. Hence, it is more important to observe right signals for the efficient silicon debug than to have larger number of signals. This is clearer in Table II. Since the ILP size is determined by the total number of signals groups [e.g., 3 in Fig 2(b)] and the total number of error and vector pairs, it may vary.

Table II shows simulation results for the average latency to detect the circuit misbehavior. The detection latency is measured by the number of clock cycles after the error is injected until it is observed. The measured latency is averaged over 300 different random error injections. To compare the effectiveness of the proposed method, in addition to the proposed method, two other methods are used (random and structural selection method). The size of each logic cone is sorted using the structural information. Since the largest logic cones have higher probability of detects, the flip-flops that are fed by the largest logic cones are selected to monitor the nonconforming circuit behavior. Since the random and structural based signal selection does not require threshold

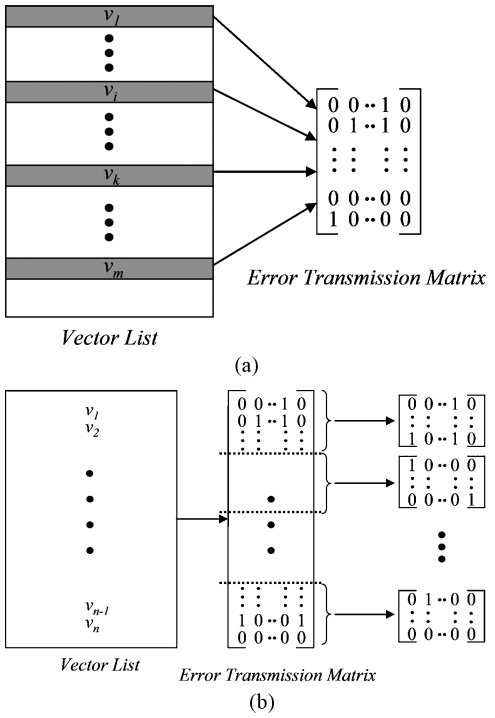

Fig. 4. Techniques to generate error transmission matrix. (a) Generating error transmission matrix with sample vectors. (b) Generating error transmission matrix with entire vectors.

values, they are not applied. The third column shows the circuit misbehavior detection latency with different number of signals. The results by the proposed methods are shown in Table II(b) and (c). Table II(b) shows the results with partial vectors to perform the debugging and results with the entire vector sets are illustrated in Table II(c). Because the proposed method selects a list of signals that are most often sensitized to possible errors, it detects the erroneous data more rapidly in all cases than other methods shown in Table II(a). Table II(c) generally shows shorter detection latency with more functional vectors used. For s9234 and s38584, because random functional vectors are used, we can say that selective random vector selection may be one of good ways of choosing representing entire vectors. Experimental results show moderately close detection latency from s9234 and s38584.

These results show that careful signal selection can be used to increase the efficiency and speed of silicon debug.

V. Conclusion

An automated procedure for selecting which signals to observe was proposed for more efficient silicon debug. The set of signals selected by the proposed method are most often sensitized to possible errors and maximally cover the errors within given constraints. The result shows that the proposed method can detect the faulty response rapidly. Multicycle fault simulation can find different set of signals, however, it exponentially increases the problem space in ILP. Fault simulation with a depth of one helps to avoid the scalability issue and serves as a good heuristic. It should also be noted that the proposed technique could be universally applied to any designs including those which do not have scan chains with nondestructive scan out capability. And the proposed method also can be applied to a selected part of a design such as newly implemented and unverified modules that require more debugging effort.

References

[1] M. Abramovici and Y.-C. Hsu, "A new approach to silicon debug," in *Proc. IEEE SDD Workshop*, Nov. 2005.
[2] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for SoCs," in *Proc. Des. Automation Conf.*, 2006, pp. 7–12.
[3] E. Anis and N. Nicolici, "On using lossless compression of debug data in embedded logic analysis," in *Proc. IEEE Int. Test Conf.*, Oct. 2007, pp. 1–10.
[4] E. Anis and N. Nicolici, "Low cost debug architecture using lossy compression for silicon debug," in *Proc. DATE*, 2007, pp. 1–6.
[5] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. Int. Symp. Circuits Syst.*, 1989, pp. 1929–1934.
[6] D. Van Campenhout, H. Al-Asaad, J. P. Hayes, T. Mudge, and R. B. Brown, "High-level design verification of microprocessors via error modeling," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 3, no. 4, pp. 581–599, 1998.
[7] GLPK [Online]. Available: http://www.gnu.org/software/glpk/glpk.html
[8] G. Parthasarathy, M. K. Iyer, T. Feng, L.-C. Wang, K.-T. Cheng, and M. S. Abadir, "Combining ATPG and symbolic simulation for efficient validation of embedded array systems," in *Proc. IEEE Int. Test Conf.*, Dec. 2002, pp. 203–212.
[9] A. Hopkins and K. McDonald-Maier, "Debug support for complex systems on-chip: A review," *IEEE Proc. Comput. Digital Tech.*, vol. 153, no. 4, pp. 197–207, Jul. 2006.
[10] Y.-C. Hsu, F. Tsai, W. Jong, and Y.-T. Chang, "Visibility enhancement for silicon debug," in *Proc. Des. Autom. Conf.*, Jul. 2006, pp. 13–18.
[11] W. Jang, D. Ding, and D. Pan, "A voltage-frequency island aware energy optimization framework for networks-on-chip," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2008, pp. 264–269.
[12] D. Josephson and B. Gottlieb, "The crazy mixed up world of silicon debug," in *Proc. Custom Integr. Circuits Conf.*, Oct. 2004, pp. 665–670.
[13] H. F. Ko and N. Nicolici, "Automated trace signals identification and state restoration for improving observability in post-silicon validation," in *Proc. DATE*, 2008, pp. 1298–1303.
[14] M. N. Velev, "Collection of high-level microprocessor bugs from formal verification of pipelined and superscalar designs," in *Proc. IEEE Int. Test Conf.*, Oct. 2003, pp. 138–147.
[15] O. Caty, "Microprocessor silicon debug based on failure propagation tracing," in *Proc. IEEE Int. Test Conf.*, Nov. 2005, p. 10.
[16] S.-B. Park and S. Mitra, "IFRA: Instruction footprint recording and analysis for post-silicon bug localization in processors," in *Proc. Des. Autom. Conf.*, 2008, pp. 373–378.
[17] *The International Technology Roadmap for Semiconductors*, Semiconductor Industry Association, 2005.
[18] B. Vermeulen, S. Oostdijk, and F. Bouwman, "Test and debug strategy of the PNX8525 nexperiaTM digital video platform system chip," in *Proc. IEEE Int. Test Conf.*, Aug. 2001, pp. 121–130.
[19] B. Vermeulen, T. Waayers, and S. K. Goel, "Core-based scan architecture for silicon debug," in *Proc. IEEE Int. Test Conf.*, Oct. 2002, pp. 638–647.
[20] L.-T. Wang, C. E. Stroud, and N. A. Touba, *System-on-Chip Test Architectures: Nanometer Design for Testability (Systems on Silicon)*. San Mateo, CA: Morgan Kaufmann, ISBN: 012373973X.
[21] Q. Xu and X. Liu, "On signal tracing in post-silicon validation," in *Proc. DATE*, Jan. 2009, pp. 262–267.
[22] J.-S. Yang and N. A. Touba, "Expanding trace buffer observation window for in-system silicon debug through selective capture," in *Proc. IEEE VLSI Test Symp.*, May 2008, pp. 345–351.
[23] J.-S. Yang and N. A. Touba, "Automated selection of signals to observe for efficient silicon debug," in *Proc. IEEE VLSI Test Symp.*, May 2009, pp. 79–84.
[24] Y.-S. Yang, B. Keng, A. Veneris, N. Nicolici, and H. Mangassarian, "Software solutions to automating data analysis and acquisition setup in silicon debug," in *Proc. IEEE SDD Workshop*, Mar. 2010.
[25] C. C. Yen, T. Lin, H. Lin, K. Yang, Y. Liu, and Y. C. Hsu, "Diagnosing silicon failures based on functional test patterns," in *Proc. Int. Workshop Microprocessor Test Verification*, Dec. 2006, pp. 94–97.