Model-Based I_{DDQ} Pass/Fail Limit Setting

T. Aruna Unni
Xilinx Corp.
2100 Logic Drive, San Jose CA 95050
Ph: (408) 879-5366, Fax: (408) 377-3259
Email: aruna@xilinx.com

D. M. H. Walker
Dept. of Computer Science
Texas A&M University, College Station TX 77843
Ph: (409) 862-4387, Fax: (409) 847-8578
Email: walker@cs.tamu.edu

Abstract

This paper describes several methods for setting IDDQ pass/fail limits using cell-based process, circuit and logic simulation. We demonstrate trade-offs in accuracy and model building effort on the ISCAS85 circuits.

1. Introduction

Quiescent current (I_{DDQ}) testing has been widely used for defect detection and diagnosis in CMOS integrated circuits [1,2,3,4,7]. However rising background currents in advanced technologies make discrimination between good and defect parts difficult [2,5,6,8]. High I_{DDQ} parts are more likely to fail [9], but any reasonable I_{DDQ} pass/fail limit results in test overkill [8]. Because of uncertainty in fault-free I_{DDQ} levels, engineers have usually chosen to minimize overkill, with pass/fail limits set 5-10x above nominal levels. Clearly many small I_{DDQ} defects will escape such limits.

If we knew the nominal I_{DDQ} for each circuit state and the normal variation, we could determine a faulty I_{DDQ} level. An empirically-derived pass/fail limit of 4 times the standard deviation (4σ) for each I_{DDQ} measurement has been proposed [10]. The disadvantage of this approach is that the large amount of production data needed can only be justified for high-volume products. An improvement is to use wafer medians or the I_{DDQ} levels of nearby die to estimate nominal values [11]. A further improvement is to use current signatures [12]. This approach avoids the need for empirical model building, and it factors out normal process, environmental, and vector-to-vector variations. The disadvantage of current signatures is that in general a pass/fail decision requires a jump in the I_{DDQ} or application of many or all vectors, resulting in increased average test time [13].

In this work we propose a model-based approach to the I_{DDQ} limit setting problem. For each circuit, we develop several models of circuit I_{DDQ} as a function of circuit state and process parameters. Deviations significantly above the predicted values can be classified as faults. The advantage of a model-based approach is that pass/fail limits can be determined with very little production experience, and can often be made quickly. This reduces test development and production test costs. Most model-based approaches to date have focused on the variation in I_{DDQ} from vector to vector [14,15,16] by evaluating the circuit state using a simulator and simple device equations or using a resistive switch model [16]. In the sections that follow we describe our I_{DDQ} models, limit-setting methodology, experimental results, and conclusions.

2. Model building approach

We use a hierarchical approach for computing I_{DDQ} assuming a cell-based circuit implementation. We first compute cell-level I_{DDQ} as a function of input values, circuit state and process parameters using *process, device and circuit simulation*. We then use this information in a logic-level simulation to compute circuit I_{DDQ} and I_{DDQ} process sensitivity for each input vector in a test set. In our work we assume all circuits are complementary CMOS, and so only subthreshold leakage is of interest [5]. Over limited device parameter intervals I_{OFF} (the leakage current at $V_{GS} = 0$) can be approximated by:

$$I_{off} \propto (WX_c / L) \cdot e^{-kV_T / T} \qquad (1)$$

where k is the Boltzmann constant, the product WX_c denotes the channel cross-sectional current flow area, W is the channel width, L_{eff} the effective channel length, V_T the device threshold voltage and T the device temperature. Since X_c is a function of a number of other process parameters which have low variance, W >> L_{eff} in most devices, and assuming constant temperature, the variation in I_{DDQ} is mostly be a function of variation in L_{eff} and V_T.

We used the technology CAD framework *pdFab* [17] to compute the impact of variations in process parameters, on L_{eff} and V_T. Nominal process simulation was done, and then nominal BSIM3v3 device parameters extracted for use in the HSPICE [18] circuit simulator. We used a two-fold process to determine the sensitivity V_T and L_{eff} to process parameters, by first determining the sensitivity of I_{DDQ} in typical circuits to V_T and L_{eff}, and then determining the sensitivity of V_T and L_{eff} to process parameters that have significant variance. The result of the analysis is that for a typical deep submicron process, V_T is most sensitive to changes in the V_T adjust implant dose and L_{eff} is most

sensitive to changes in polysilicon linewidth. We compute 4σ values for V_T and L_{eff} based on the estimated process parameter variance. We then use HSPICE to compute I_{DDQ} as a function of input values, circuit state and process parameters for each cell in the cell library. The simulation is done for all input vectors and state combinations using nominal, high and low V_T, and L_{eff} devices. A table is built for each cell with each entry containing the input values, state, nominal I_{DDQ} and sensitivities. Note that we are assuming that the increase/decrease in the I_{DDQ} with respect to process variation is linear. Since Equation 1 is concave up, the linear approximation forms an upper bound. We are also approximating that all devices within a cell have the same process parameters, which is reasonable for small cells.

The Cadence Verilog-XL logic simulator [19] is used to simulate a set of test vectors and compute the input values and state of each cell for each vector. For each vector *j*, the mean I_{DDQ} μ_j and sensitivity S_{ij} to each process parameter *i* for the entire circuit is computed by summing the values for each cell.

3. Limit setting methods

In the following subsections we propose several methods to set I_{DDQ} limits with differing simulation requirements and accuracy.

3.1. Worst case method

Using the μ_j and S_{ij} values, we compute worst-case circuit I_{DDQ} for each vector *j*. We make the reasonable assumptions that process parameters are independent and normally-distributed with standard deviation σ_i, and that for each vector the sign of S_{ij} is the same for all cells. This allows us to assume the same parameter value for all cells.

By approximating I_{DDQ} as linear in the process parameters, we can define σ_j the standard deviation for circuit I_{DDQ} for vector *j* as:

$$\sigma_j = \sqrt{\sum_i S_{ij}^2 \cdot \sigma_i^2} \tag{2}$$

and set a 4σ pass/fail limit as:

$$max\ I_{DDQj} = \mu_j + 4\sigma_j . \tag{3}$$

3.2 Dynamic method

The problem with the worst case method is that a fault could be present in a low-I_{DDQ} chip (e.g. a slow process corner) and escape detection. Rather than a single worst-case limit for each vector, we can correct for process variation by setting dynamic limits for each chip by measuring the I_{DDQ} values for the first few vectors. If we see that the relationship among vectors fits the expected pattern and is within the worst-case limits, then we can assume that these vectors do not sensitize a defect. We can then use these measurements to estimate the process parameters for this chip, and then compute pass/fail limits for all remaining vectors.

The difference ΔI_{DDQj} between the measured I_{DDQ} with process variations $I_{DDQ\ measj}$ and the nominal value is:

$$\Delta I_{DDQj} = I_{DDQ\ measj} - \mu_j \tag{4}$$

The process parameters we consider the poly linewidth ΔP and the V_T adjust implant dose ΔD. Then for the first k vectors:

$$\begin{aligned} \Delta I_{DDQ1} &= S_{11}\Delta P + S_{21}\Delta D \\ \Delta I_{DDQ2} &= S_{12}\Delta P + S_{22}\Delta D \\ &\cdots \\ \Delta I_{DDQk} &= S_{1k}\Delta P + S_{2k}\Delta D \end{aligned} \tag{5}$$

We then use singular value decomposition to compute ΔP and ΔD and compute the I_{DDQ} limit for vector *j* as:

$$max\ I_{DDQj} = \mu_j + S_{1j} \cdot \Delta P + S_{2j} \cdot \Delta P \tag{6}$$

We can estimate ΔP and ΔD from wafer test structures and scan chain flush delays to detect early vectors that are faulty, but do not exceed the worst-case limits.

3.3 Probability method

The probability method uses a cell-level statistical analysis to determine I_{DDQ} limits. For each cell we can determine the maximum I_{DDQ} when the cell is "ON" (output at logic 1) or "OFF" (output at logic 0). We assume single-output cells. Let a circuit have n cell instances G_1 to G_n. Let the probability for cell G_i to be ON be P_{ONi} and to be OFF be P_{OFFi}. Let the maximum I_{DDQ} drawn by G_i be I_{ONi} when it is ON and be I_{OFFi} when it is OFF, assuming 4-sigma process limits. We can estimate the worst case circuit I_{DDQ} for all vectors as:

$$\max I_{DDQ} = \sum_{i=1}^{n} (P_{ONi} \cdot I_{ONi} + P_{OFFi} \cdot I_{OFFi}) \tag{7}$$

The advantage of the probability method is that we do not have to simulate the entire test set, but only have to simulate enough vectors to estimate P_{ONi} and P_{OFFi}.

A lower-cost approach that we will use is to assume a Universal gate circuit, i.e. composed of NAND, NOR and NOT gates, with XOR gates decomposed. For simplicity, we model all gates as two-input gates. By assuming equal probabilities for the input values to produce a given output value, we compute 4-sigma maximum I_{ON} and I_{OFF} values for each gate type using the tables of Section 2. Given G_{NAND} NAND, G_{NOR} NOR and G_{NOT} NOT gates, the circuit I_{DDQ} is computed as:

$$\begin{aligned} I_{DDQ} = {} & G_{NAND} (P_{ON,NAND} I_{ON,NAND} + P_{OFF,NAND} I_{OFF,NAND}) + \\ & G_{NOR}(P_{ON,NOR} I_{ON,NOR} + P_{OFF,NAND} I_{OFF,NOR}) + \\ & G_{NOT}(P_{ON,NOT} I_{ON,NOT} + P_{OFF,NOT} I_{OFF,NOT}) \end{aligned} \tag{8}$$

For complementary CMOS, NAND and NOT gates have maximum I_{DDQ} when OFF, while NOR gates have maximum I_{DDQ} when ON. So overall I_{DDQ} is maximized when $P_{OFF,NAND} = P_{OFF,NOT} = 1$ and $P_{ON,NOR} = 1$. Since there is no logic simulation, model-building need only be done once per cell library.

4. Experimental results

We performed simulation experiments using the ISCAS85 circuits [19]. We assume the gates are implemented as complementary cells in a typical 350 nm CMOS process. We computed I_{DDQ} and sensitivity values for each cell at 25C as described in Section 2, using typical process variations. We used ATALANTA-generated stuck-at test sets [20]. We then calculated I_{DDQ} using the worst case (WCI_{DDQ}), dynamic ($DynI_{DDQ}$) and probability ($ProbI_{DDQ}$) methods. For the dynamic method, the first few vectors were used (enough to get sufficiently different S_{ij} values), with their I_{DDQ} values generated using the worst case method. Thus the WCI_{DDQ} and $DynI_{DDQ}$ predictions should be similar.

The nominal I_{DDQ} and the three computed maximum I_{DDQ} limits for each vector for the ISCAS85 circuits are shown in Figures 1 to 11. The WCI_{DDQ} values can be considered the most accurate. The $DynI_{DDQ}$ predictions are slightly lower due to the linear approximation of Equation 1. Computing the process sensitivities around the worst-case parameter values would guarantee that the $DynI_{DDQ}$ prediction exceeds the WCI_{DDQ} prediction. The $ProbI_{DDQ}$ limits are the most conservative for most vectors of most circuits, except for c6288. This can be attributed to the fact that the assumption that all gates are in their maximum leakage state (output value) is conservative. The reason the prediction for c6288 is low is that the assumption of equally-probable inputs for each gate output value is incorrect. A better estimate could be obtained with a small amount of random logic simulation. Table 1 shows a comparison of the maximum and minimum 4σ I_{DDQ} limits over all vectors for the ISCAS85 circuits using the three methods.

5. Conclusions

In this paper we have described several approaches for computing I_{DDQ} pass/fail limits that trade off model-building cost, accuracy, and quality. The worst-case method minimizes test overkill, but it requires logic simulation of the test set, and can let faults in a slow process escape. The dynamic method avoids this problem, but requires solution of a matrix problem during testing. The no-simulation probability is most practical for large circuits, but results in a single pass/fail limit, and causes overkill if its assumptions are invalid (as in c6288). The results show that for logic circuits, I_{DDQ} varies mostly due to the process. This suggests using the dynamic method for good coverage without excessive overkill.

References

1. R. Rajsuman, *I_{DDQ} Testing for CMOS VLSI*, Artech, 1995.
2. A. Ferre and J. Figueras, "I_{DDQ} Characterization in Submicron CMOS", *ITC,* Wash. DC, 1997, pp. 136.
3. R. Gulati and C. Hawkins, *I_{DDQ} Testing of VLSI Circuits*, Kluwer Academic Publishers, Boston, 1992.
4. H. Balachandran, Y. J. Kwon and D. M. H. Walker, "I_{DDQ} Current Calibration with Process Variations", *IEEE Int'l Workshop on I_{DDQ} Testing,* Wash. DC, 1995, pp. 61-65.
5. T. W. Williams, R. H. Dennard, R. Kapur, M. R. Mercer and W. Maly, "I_{DDQ} Test: Sensitivity Analysis of Scaling", *Int'l Test Conf.,* Washington DC, Oct. 1996, pp. 786-792.
6. J. Acken and S. Millman, "Current Measurement Data To Demonstrate Setting Current Limits in a Complete I_{DDQ} Testing Methodology", *IEEE Int'l Workshop on I_{DDQ} Testing,* Washington DC, 1995, pp. 69-72.
7. J. Acken, "Testing for Bridging Faults (Shorts) in CMOS Circuits," *Design Automation Conf.*, 1983, pp. 717-718.
8. SEMATECH Study, "An Experimental Study Comparing the Relative Effectiveness of Functional Scan, I_{DDQ} and Delay-Fault Testing", *IEEE VLSI Test Symp.*, Apr. 1997, pp. 459.
9. T. Barrette, "Evaluation of Early Failure Screening Methods", *IEEE International Workshop on I_{DDQ} Testing*, Washington DC, October 1996 , pp. 14-27.
10. T. Henry and T. Soo, "Burn-In Elimination of a High Volume Microprocessor Using I_{DDQ}", *IEEE Int'l Test Conf.,* Washington DC, October 1996, pp. 242-249.
11. A. D. Singh, "A Comprehensive Wafer Oriented Test Evaluation (WOTE) Scheme for the I_{DDQ} Testing of Deep Sub-Micron Technologies", *IEEE Int'l Workshop on I_{DDQ} Testing*, Washington DC, Nov. 1997, pp. 40-43.
12. A. Gattiker and W. Maly, "Current Signatures: Application", *Int'l Test Conf.*, Wash. DC, 1997, pp. 156.
13. A. Gattiker and W. Maly, "Current Signatures", *VLSI Test Symp.*, Princeton NJ, April 1996, pp. 112-117.
14. A. Ferre and J. Figueras, "On Estimating Bounds of the Quiescent Current for I_{DDQ} Testing", *VLSI Test Symposium*, Princeton, April 1996, pp. 106.
15. P. Nigh, W. Needham, K. Butler, P. Maxwell, R. Aitken and W. Maly, "So What is an Optimal Test Mix? A Discussion of the Sematech Methods Experiment", *Int'l Test Conf.*, Washington DC, Nov. 1997, pp. 1037-1038.
16. P. C. Maxwell and J. R. Rearick, "A Simulation-Based Method for Estimating Defect-Free IDDQ", *IEEE Int'l Workshop on I_{DDQ} Testing*, Wash. DC, 1997, pp. 80-84.
17. PDF Solutions, Inc., *pdFab* Version 3.1, User Manual.
18. Meta-Software, Inc., *HSPICE Manual.*
19. F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", *ISCAS,* 1985, pp. 663-698.
20. H. Lee and D. Ha, "On the Generation of Test Patterns for Combinational Circuits", Tech. Report No. 12_93, EE Dept, Virginia Polytechnic Institute and State University.

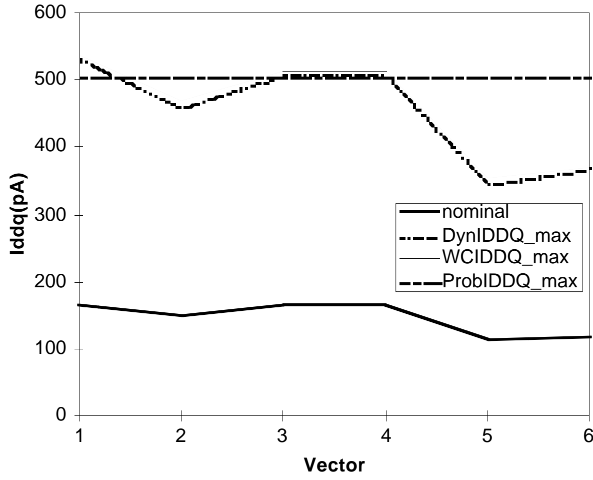

Figure 1. I_{DDQ} current variations and limits for c17

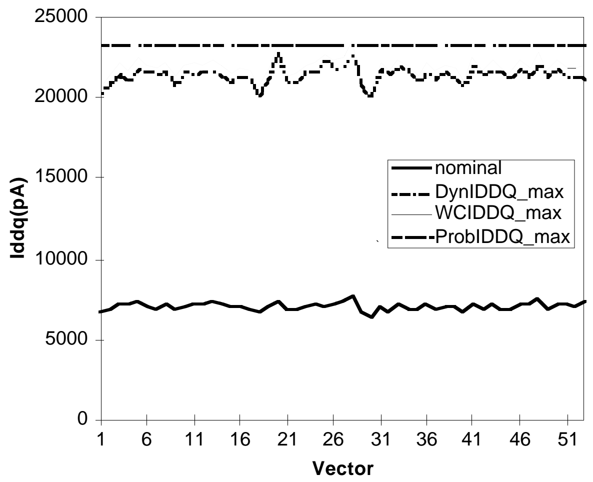

Figure 2. I_{DDQ} current variations and limits for c432

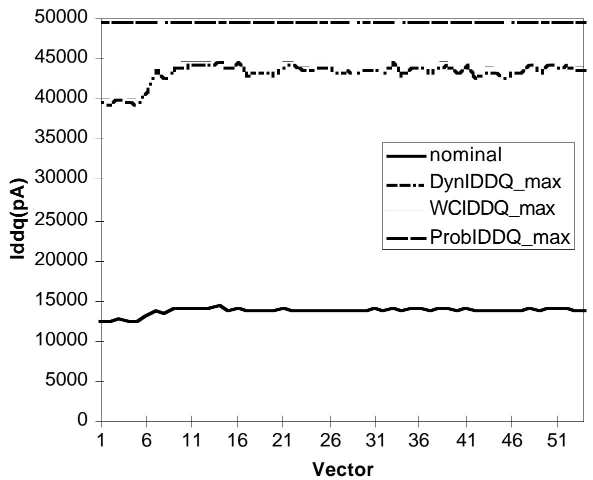

Figure 3. I_{DDQ} current variations and limits for c499

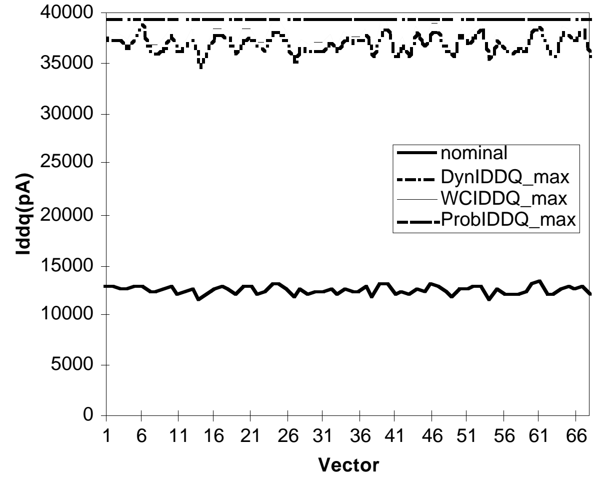

Figure 4. I_{DDQ} current variations and limits for c880

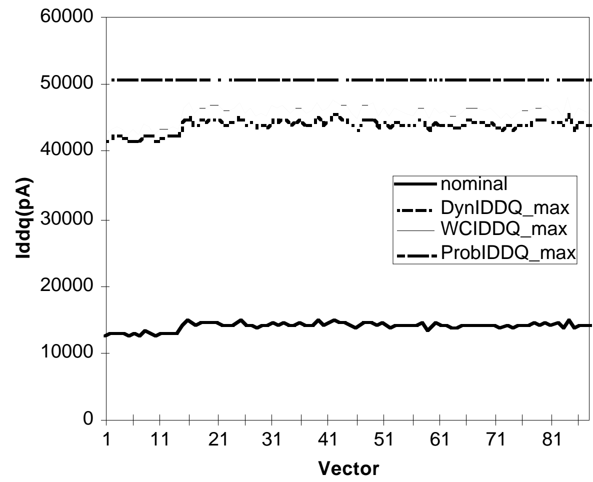

Figure 5. I_{DDQ} current variations and limits for c1355

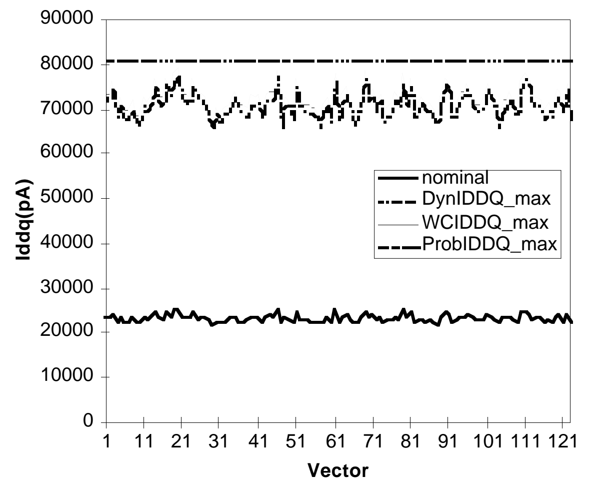

Figure 6. I_{DDQ} current variations and limits for c1908

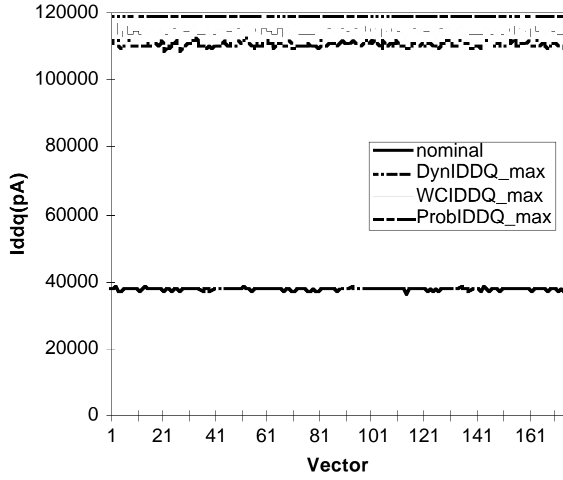

Figure 7. I_{DDQ} current variations and limits for c2670

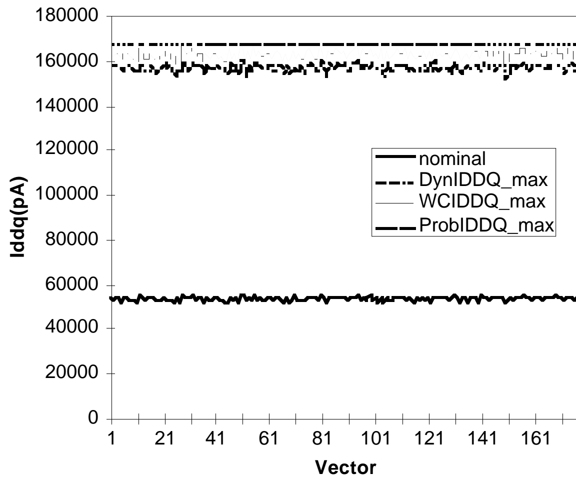

Figure 8: I_{DDQ} current variations and limits for c3540

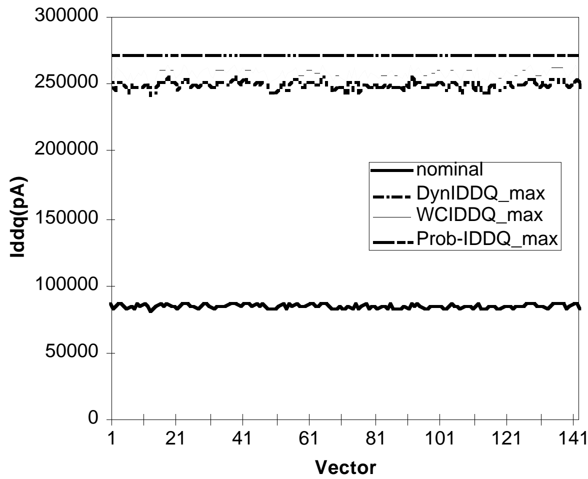

Figure 9. I_{DDQ} current variations and limits for c5315

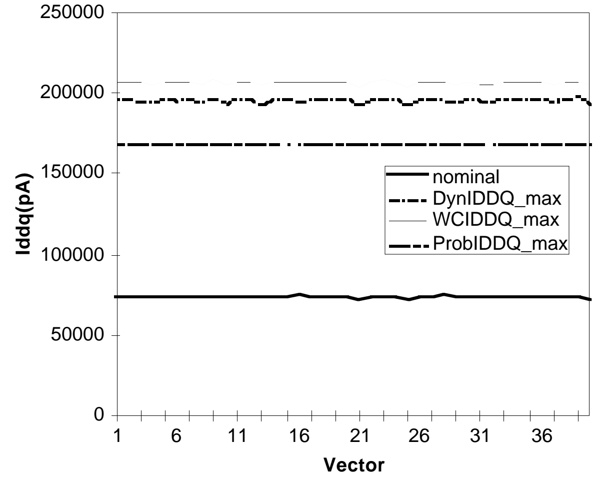

Figure 10. I_{DDQ} current variations and limits for c6288

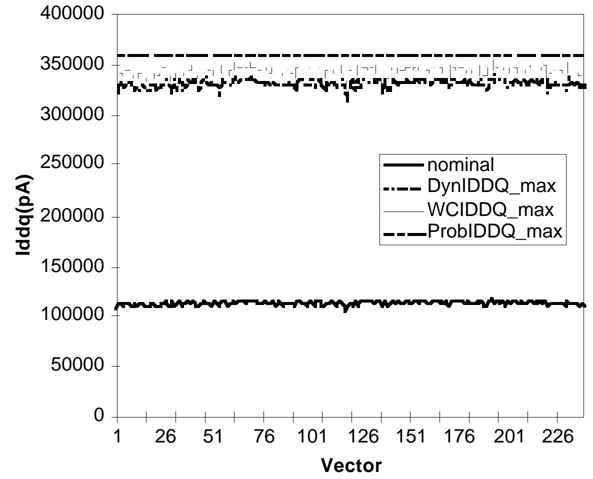

Figure 11. I_{DDQ} current variations and limits for c7552

Table 1. ISCAS85 I_{DDQ} 4-sigma values (pA)

Circuit	Worst Case		Dynamic		Probability	
	Min	Max	Min	Max	Min	Max
c17	4	351	7	343	8	506
c432	373	23515	423	22905	492	23240
c499	738	44700	788	45185	872	49598
c880	992	39479	1062	38567	1130	39443
c1355	872	47831	902	45368	910	50949
c1908	1207	78904	1407	77310	1628	80812
c2670	2696	117208	2696	113351	3039	119464
c3540	3591	166819	3691	163112	4865	168344
c5315	7492	265117	7642	256818	7367	271361
c6288	8171	207443	8363	195628	8635	168955
c7552	9046	353959	9246	341329	9355	360111