Generation of Optimized Single Distributions of Weights for Random Built-In Self-Test

Miguel A. Miranda, Carlos A. López-Barrio

Univ. Politécnica de Madrid, E.I.S.I.Telecom., Integrated Systems Laboratory
Ciudad Universitaria s/n, 28040 Madrid (SPAIN), email: miranda@die.upm.es

Abstract

This paper presents a probabilistic-based fault-model approach to the generation of optimized single distributions of weights for random Built-In Self-Test. Many techniques use multiple sets of weights to obtain an important reduction in the test length. However, these strategies consume large memory areas to store the different distributions. In order to obtain a highly optimized set of weights, which reduces area overhead, a global optimization procedure is used to minimize a testability cost function that projects random fault coverage. Important reduction in test length, using only a highly optimized single distribution of weights, is reported.

1. Introduction

The idea of incorporating test circuits on-chip (i.e., built-in self-test) is very attractive since tester independence suggests lower production costs than those offered by deterministic stored-pattern testing [1]. Unfortunately, experience with random pattern testing, has shown that some circuits require several million vectors to achieve an acceptable test coverage. Therefore, the generation of the random patterns and its associated signature analysis may consume an unreasonable amount of time.

An alternative to pseudorandom testing is the weighted random pattern testing (WRPT), which significantly reduces the test length requirements. WRPT is a technique in which the design of the pseudorandom pattern generator is changed to produce a variable distribution of logical 1s and 0s, for each primary input of the logic block under testing.

Many approaches are based on the use of multiple sets of weights to obtain an important reduction in the number of patterns. However, these strategies consume large amounts of silicon area to store the different sets of weights. Note that each primary input (PI) may require a different weight, thus a medium size combinational block with one hundred inputs and ten distributions would need a memory big enough to store one thousand weights. Therefore, these approaches are feasible only in boundary-scan environments. Our strategy is intended to obtain a significant performance using one distribution of weights optimized as much as possible, so that the memory area overhead is reduced considerably.

The idea of using weights to generate random patterns for VLSI circuits is somewhat recent. To generate distributions of weights, some authors [2,3] have proposed techniques based on the observation of internal switching activity obtained by simulation. These approaches have the drawback that not always functional test sets are a proper choice as the prime source of information for testing purposes.

Waicukauski *et al.* [4] developed a fast procedure based on the examination of all possible paths from inputs to outputs, with an initial weight set calculated depending on the number of inputs and type of gates, that feed the blocks along the traced paths. The faults that remain undetected are covered by patterns generated with a deterministic test-pattern generator (DTPG). However, differences in input gates fed by fanout points, could make the estimate of the initial weight set inaccurated.

Bardell, McAnney and Savir proposed in [5] a method to bound fault-detection probabilities. The core of their weight-optimization scheme relies on maximizing the cumulative detection probability for a given number of random test patterns (test length). Several optimal signal probability assignments for basic gates, are proposed to obtain an optimized set of weights. These assignments are backtraced from gate outputs to inputs until primary inputs are reached. This method is intended to produce highly optimized set of weights, an experimental evaluation [6] between this approach and the one proposed by Waicukauski, reveals that this strategy [5] performs better in terms of fault coverage.

Lisanke [7] and Wunderlich [8], presented approaches based on estimates of the detection probability for each

0-7803-1429-8/93 $3.00

stuck-at fault in the circuit. These estimates are used to build up a target function that projects the random fault-detection cost. This function depends on the signal probabilities associated to each primary input. So, the optimized set of weights is obtained minimizing the target function. However, the optimization procedures, implemented in these works, just provide solutions for local minimum. Thus, the resulting number of random patterns might not be optimal, requiring several sets of weights to obtain an important reduction in the test length.

Muradali *et al.* [9] proposed a novel method to obtain highly optimized sets of weights, where the circuit under test is considered as a black box, relying neither on signal probability calculations nor analysis of circuit structure. In this work, a first equiprobable random phase is performed until the fault coverage reaches a user defined threshold point. From the pool of patterns generated in this phase, only the patterns that detected difficult faults are considered. Each bit of these patterns is averaged to obtain a first distribution of weights. This first distribution is refined by means of a *bit flipping* algorithm until the final distribution is obtained. An important reduction in test length is achieved, combining a first equiprobable random sequence followed by a weighted random pattern generation.

Recently Brglez *et al.* [10,11] proposed a method and a hardware implementation to generate weighted test patterns for combinational logic. Their approach is based on a precomputed 3-valued deterministic set of vectors for random pattern-resistant faults. These deterministic patterns are used to generate optimized distributions of weights, that guarantee pattern coverage for the precomputed deterministic test set. Again this approach uses different distributions of weights to reduce the number of random patterns, although they offer the possibility of a tradeoff between the number of distributions (hardware memory overhead) and the length of the total patterns (overall testing time).

The approach proposed in this paper uses detection-probability estimates to build up a testability cost function. A probabilistic model for stuck-at faults for combinational logic is formulated using COP [12] testability measures. This model is employed to build a target function that projects the random fault-detection cost. Some properties are also presented to simplify the minimization procedure. The optimization problem is a member of the general class of multi-extremal problems. It will be shown that the solution is highly dependent on the starting point of the minimization procedure. So, to avoid this dependency we have implemented a global optimization procedure based on the fast simulated diffusion technique presented in [13] that is able to give an estimate of the global minimum.

This paper is divided into five sections. Section 1 (this section) summarizes several state-of-the-art approaches to WRPT, and presents an overall description of the proposed method. Section 2 introduces a probabilistic-based fault model for the target function that projects the random fault-detection cost. Section 3 describes the global minimization procedure used to optimize the target function. In section 4, we present experimental results for fault coverage and test length by simulating weighted patterns for a subset of ISCAS85 [14] and ISCAS89 [15] benchmark circuits. The paper concludes in section 5, where results from previous sections are discussed.

2. Probabilistic Model for Stuck-At Faults.

Several approaches [5,7,16] have been proposed for estimating fault detection probabilities for combinational circuits. Because of the complexity of exact computation of detection probabilities (NP-hard [8], and since they have to be evaluated very often for different weigth sets, heuristics for estimating them, are required. We choose COP [12] testability measures because of their simplicity and high efficiency in computational effort. However, COP makes extensive use of the signal independence assumption that can be incorrect in circuits with redundancies and reconvergent fanouts. Nevertheless, for most of circuits this testability procedure is enough accurate to establish a cost function model, as it will be shown by the experimental results of section 3.

2.1 Computation of Testability Measures

The notions of controllability and observability measures have been widely used in many testability applications. The controllability value of a signal, $c1_l$, is the probability that a line l takes, by convention, a logical value of "1". The observability of a signal, o_l, is the probability that the logical value of a line l can be observed at a primary output.

The COP program computes the controllability values for all the lines in the circuit by propagating the input signal probabilities (distribution of weights) from the inputs of the gates to their outputs. The observabilities are computed backtracing the gates outputs observabilities to their inputs, until the primary inputs are reached.

To implement the propagation scheme efficiently, an ordered graph representation of the circuit is generated. This graph is used for propagating the testability measures through gates and fanout points, tracing propagation paths from circuit inputs to outputs and viceversa. In this way, only the lines included in the propagation paths affected by input signal probabilities assignments, that could differ from the previous ones, are considered.

The process of propagating testability measures is based on a predefined library of controllability and observability

equations, for all basic gates. Furthermore, predefined hierarchical elements can be mapped into testability equations. This library has been created using the formulation of the COP definitions for controllability and observability measures:

Controllability: Let $c1_l$ be the 1-controllability of a line l. To compute $c1_l$ as a fuction of the input probabilities of a gate with inputs $\{x_1, x_2 ..., x_n\}$ and output z, all possible input assignments (A_z) for which line z takes on logic value "1", are considered, resulting:

$$c1_z = \sum_{A_z} \left\{ \prod_{i=1}^{n} p_{x_i} \right\} : \qquad p_{x_i} = \begin{cases} C1_{x_i}, if\ x_i = 1 \\ C0_{x_i}, if\ x_i = 0 \end{cases} \quad (1)$$

Observability: The input observabilities also can be determined in terms of p_{xi} and the output observability o_z as follows. Let S_z be the set of all possible input assignments that sensitize the output z to changes in input x_i. The observability of line i is then:

$$o_{x_i} = o_z \cdot \sum_{S_z} \left\{ \prod_{\substack{j=1 \\ i \neq j}}^{n} p_{x_j} \right\} \quad (2)$$

For fanout points, the observability (o_s) of the fanout origin s is computed in terms of the observabilities of the fanout branches $\{b_1,b_2...,b_m\}$ as follows:

$$o_s = 1 - \prod_{i=1}^{m} (1 - o_{b_i}) \quad (3)$$

2.2 Modelling Fault-Detection Probabilities

After computing the controllability and observability measures for lines included in the propagation paths, the fault detection probabilities (s-a-1, and s-a-0) are modelled as the intersection of the controllability and observability values associated to these lines.

$$pf_i^{s-a-0} = c1_i \cdot o_i \qquad pf_i^{s-a-1} = c0_i \cdot o_i \quad (4)$$

The above expression represents the probability of detecting a certain fault f if a random pattern is applied to the circuit under test. If N patterns are generated, the expression for the *projected fault coverage* can be computed as an average over the whole target fault set $\mathcal{F}$ being K the number of faults contained in $\mathcal{F}$.

$$FC_{prj}(N) = \frac{1}{K} \cdot \sum_{f \in \mathcal{F}} [1 - (1 - p_f(X))^N] \quad (5)$$

FC_{prj} depends on N, the number of applied random patterns (test length), and on X, the input signal probability distribution. The estimated fault coverage approaches 100% as we apply more and more random test vectors to the circuit. This model of fault coverage provides a means to project random test generation cost, given a circuit topology, a fault set, and an input distribution of weights.

2.3 The Target Function

The next step is to find an expression to evaluate the fault detection cost of random test generation. Previous works, on probabilistic-based models for weights generation, have approached to this problem. Lisanke *et al.* [7] proposed the uncoverage area as a target cost function (eq. 6a), that is, the area above the estimate fault coverage curve. Wunderlich [8] proposed as cost function the probability that all faults are detected by N patterns with a distribution X (eq.6b). We have chosen the uncoverage area because its independence of the number of patterns (N) makes the optimization problem much easier. Otherwise, it would be necessary to bound the expression to make it not dependent on the number of trials N. Note that the condition $e^{-Np_f(X)} \ll 1$ is fulfiled since maximum fault coverage usually requires a number big enough of random patterns.

$$U(X,N) = \frac{1}{K} \cdot \sum_{f \in \mathcal{F}} \left. \frac{1 - e^{-Np_f(X)}}{p_f(X)} \right|_{e^{-Np_f(X)} \ll 1} \approx \frac{1}{K} \cdot \sum_{f \in \mathcal{F}} \frac{1}{p_f(X)} \quad (a)$$

$$U(X,N) \approx \sum_{f \in \mathcal{F}} e^{-Np_f(X)} \quad (b) \quad (6)$$

2.4 Properties of the Target Function

In this section, we present some mathematical properties of the target function that will be exploited in the minimization procedure (section 3).

It will be shown that the cost function is convex with respect to a direction defined by any single variable of the function domain. Therefore for each distribution of input signal probabilities $\{x_1...,x_{i-1},x_i,x_{i+1}...x_n\}$ there exists exactly one $x_i \in X$, along the direction defined by x_i with minimum $U(x_1...,x_i...,x_n)$.

Let $X=\{x_1...,x_n\}$, be an input signal probability distribution and y a scalar value that any component of this distribution could take. The conditional detection probability, of a certain component of X, to a given value y can be written as:

$$p_f(X/x_i=y) = p_f(x_1...x_{i-1},y,x_{i+1}...x_n)$$

and expanded using the Bayes theorem as,

$$p_f(X/x_i=y) = y \cdot p_f(X/x_i=1) + (1 - y) \cdot p_f(X/x_i=0)$$

thus, the partial derivative of the detection probability with respect to y is,

$$p'_f(X/x_i=y) = p_f(X/x_i=1) - p_f(X/x_i=0)$$

Therefore, the first and second partial derivatives of the target function can be computed easily, resulting:

$$\frac{\partial U(X/x_i=y)}{\partial y} = -\frac{1}{K}\cdot\sum_{f\in\mathcal{F}}\frac{p_f(X/x_i=1)-p_f(X/x_i=0)}{p_f^2(X/x_i=y)} \quad (7)$$

$$\frac{\partial^2 U(X/x_i=y)}{\partial y^2} = \frac{2}{K}\cdot\sum_{f\in\mathcal{F}}\frac{(p_f(X/x_i=1)-p_f(X/x_i=0))^2}{p_f^3(X/x_i=y)} > 0 \quad (8)$$

Note that the second partial derivative is positive for any input signal distribution X, since the detection probability for every fault considered, is also positive. Therefore, the objective function is convex with respect to a direction defined by any single variable of the function domain.

The minimum of the cost function, along this direction, can be easily computed with the Newton iteration using formulas (7) and (8):

$$y_{next} = y - \frac{U_y'(X)}{U_y''(X)} \quad (9)$$

Formula (9) needs the values of $p_f(X/x_i=1)$ and $p_f(X/x_i=0)$. They can be computed before the iteration because these values are fixed, independently of the one that y could take. So, this method for estimating the derivatives of the detection probabilities, along a specific direction, is very efficient in terms of CPU time.

3. The Minimization Procedure

Known gradient-based minimization procedures, like the Newton iteration or the Steepest Descent method, can be easily trapped in a local minimum and, thus, cannot find a global minimizer. Recently a method named *Simulated Diffusion* [17,13] has been proposed to find the global minimum of a multiminimal function on continuous space. The main idea in Simulated Diffusion is based on the physical fact that a particle placed in a given potential, and with a Brownian motion, is diffused into the global minimum of the given potential.

The stochastic differential equation that describes the diffusion process of a particle with Brownian motion is given as:

$$dX = -\nabla U(X)\cdot dt + \sqrt{2T}\cdot dw \quad (10)$$

where t is time, X is the space coordinate that shows the location where the particle is, $U(X)$ is a potential function in which the particle is put, ∇ is a gradient operation, dw is Gaussian random noise, and T is temperature.

When the temperature is high, the Brownian term dominates and the movement of the particle is just stochastic. On the other hand when the temperature becomes low, the gradient potential term dominates and the process approaches pure hill descent. The Brownian term is essential to get out of the local minimum and the gradient operation gives the tendency to minimize the function.

The optimization procedure is based in the one proposed by Sakurai, Lin *et al.* [13], namely *Fast simulated Diffusion (FSD)*. In FSD, the computation of the next state (y_{next}) is carried out alternatively; that is, y_{next} is first calculated by a hill-descent (gradient driven) method and afterwards is generated by a random Gaussian distribution. If the cost is improved the state is saved otherwise, the accept/nonaccept function of a Boltzman distribution (commonly used in simulated annealing [18]) will decide whether the state is accepted or not. This method for finding global minimum on continuous multidimensional functions is similar to simulated annealing, however, since the simulated diffusion takes adavantage of the information from function derivatives, it is much more efficient in terms of computing time.

To implement the hill-descent procedure, several considerations make the method more efficient. First, if the state space has large dimensions, the projection of $\nabla U(X)$ in a randomly picked axis r approaches, in the long run, to $\langle \nabla U(X), r\rangle r$. We take advantage of this fact to reduce the number of gate evaluations during the propagation of the testability measures through the circuit graph. Second, as the objective function is convex with respect to a direction defined by any single variable of the function domain, the inexpensive down-hill procedure, described in section 2.4, can be applied.

The initialization scheme and the adaptive cooling schedule presented in [19] are adopted. That is, the initial temperature is determined by statistics collected over randomly selected starting points, and the cooling schedule is controlled to have a uniform decrease of the average cost versus the logarithm of temperature.

The equilibrium condition, the criteria to stop the generation of next-states and to proceed to update the temperature (inner-loop), is not adaptive. We have limited the number of times that the inner-loop is executed, to a number at least equal to the dimension of the problem (number of inputs to the circuit). Figure 1 shows a pseudocode description of the implemented Fast Simulation Diffusion algorithm.

Figure 2 displays the evolution of the average cost versus the temperature for the circuit c2670, showing a great dependency between the average cost of the accepted states and the cooling schedule.

In order to demonstrate the advantages of using a global optimization approach to the generation of single distributions of weights, we have implemented the local optimization scheme proposed by Lisanke *et al.* in [7].

```
Fast_Simulated_Diffusion() {
  Sample_Solution_Space();
  T_init = Initialize_Temperature();
  while ((T/T_init > T_Ratio_Min)
      AND (f(X_opt)/f(Xprev_opt)) > Cost_ratio_min {
    for (iINT = 0; iINT < iINT_max; iINT++) {
      X_new = Generate_X_new();
      if ((Δf = f(X_new) - f(X)) < 0) {
        X = X_new;
        if (f(X) < f(X_opt)) {
          X_opt = X; f(Xprev_opt) = f(X_opt); f(X_opt) = f(X);
        }
      } else {
        P = exp(-Δf/T),
        R = Random_Number_in_[0,1]();
        if (R < P)  X = X_new;
      }
    }
    T = Update_T();
    S = S_init * (T/T_init)^k;
  }
  Solution = Xopt;
}
Generate_X_new() {
  if (gradient_Flag == 1) {
    Random_Select_Single_Variable_x_i();
    Generate_X_new_with_Newton_Iteration;
    gradient_Flag = 0;
  } else {
    X_new = X + S * nth_Dimensional_Gaussian_Distribution;
    gradient_Flag = 1
  }
}
Update_T() {
  T_Factor = exp(-λT/σ);
  if (T_Factor < T_Factor_Min)  T_Factor = T_Factor_min;
  T *= T_Factor;
}
```

Figure 1. Description of the implemented Fast Simulated Diffusion Algorithm.

Their optimization procedure is based in a steepest descent minimization algorithm (the complementary Davidson-Fletcher-Powell), and their proposed equations for propagating the cost function derivatives, have been also implemented.

Table 1 shows the performances of using a global minimization procedure over a local one. The table presents the values of the target function (see section 2.3) for a subset of the ISCAS85 and for the combinational equivalent parts of an ISCAS89 benchmark subset. *Uinit* stands for the initial value of the target function corresponding to an equiprobable distribution of weights. *Ulocal* corresponds to the minimum cost function value, obtained using the local minimization algorithm. *Uglobal*

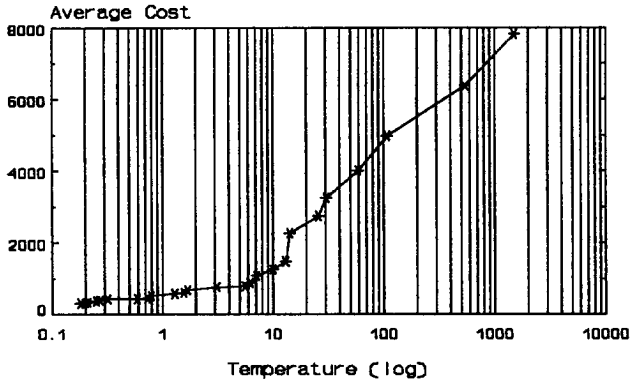

Figure 2. Average cost of accepted states versus temperature, for the circuit c2670.

shows the miniumum reached with the Fast Simulated Diffusion procedure. The last column shows the seconds consumed by the global optimization procedure running on a SUN-sparcStation 10/20.

Circuit	*Uinit*	*Ulocal*	*Uglobal*	*secs*
s344	23.13	14.40	10.03	217
s420	20,935	20,134	8,353	198
s526n	265.3	97.69	31.07	254
s641	31,039	31.12	9.54	566
s820	579.2	340.9	153.3	735
s838	669 10^6	668 10^6	253 10^6	803
c880	109.3	104.2	13.23	717
s953	589.9	102.9	59.02	570
s1488	134.7	84.39	67.41	1,167
c1908	341.2	228.9	66.86	2,335
c2670	19,922	10,139	277.6	3,797
c3540	2,190	1,541	415.9	4,562
s5378	1,246	305.7	77.05	7,081
c7552	12.34 10^9	1.11 10^9	3,356	12,081
s9234	3.74 10^6	2.56 10^6	1,036	20,159

Table 1. Minimum cost function values for local and global optimization procedures

Note that for many circuits, an important reduction in the cost function is achieved using the global optimization procedure. Furthermore, this reduction is much bigger than the one obtained with the local minimization strategy. So, it is cleary worthwhile to spend the CPU time reflected in Table 1, in order to perform the global minimization process.

A similar comparison can be done in terms of test length sizes for some of the circuits used in Table 1. Table 2 shows the *predicted* (from fault coverage projections) and *experimental* values for test length sizes for these circuits. *TLeqp* is the test length corresponding to an equiprobable distribution, *TLlcl* stands for the test length for a distribution obtained using the local minimization

Circuit	Predicted			Experimental			Fault Coverage (detectbl)
	TLeqp	TLlcl	TLglb	TLeqp	TLlcl	TLglb	
s344	1,009	369	187	436	142	135	100.00 %
s420	872,728	726,644	279,151	342,320	327,827	126,926	99.78 %
s526n	21,262	5,588	3,680	21,231	2,646	1,033	100.00 %
s641	2,619,559	13,480	182	771,829	7,535	404	100.00 %
s820	24,628	14,384	6,815	34,992	13,930	4,303	100.00 %
s838	489.877	291,160	242,932	301,237	297,040	219,171	74.87 %
c880	10,853	10,176	223	19,142	9,435	272	100.00 %
s953	32,366	4,751	2,041	43,541	12,255	1,859	100.00 %
s1488	4,818	2,391	1,880	7,871	2,509	1,528	100.00 %
c1908	12,341	9,039	2,872	25,195	14,951	6,629	100.00 %
c3540	46,496	37,423	6,492	76,093	18,336	8,553	100.00 %
s5378	88,854	26,586	5,588	73,731	65,452	10,420	100.00 %

Table 2. Predicted and experimental test length for local and global optimization procedures.

procedure, and *TLglb* is the test length that corresponds to a distribution resulting from the global optimization procedure. Finally, in the last column the target fault coverages, over detectable faults, are displayed. In order to generate the random patterns in the same conditions as they would be generated on chip, we have implemented the hardware generator for weighted random patterns proposed by Brglez, Gloster *et al.* [10]. This hardware uses a Cellular Automata register (CAR) (with a configuration based on the rules 90 and 150) as the source of equiprobable patterns and one register to store the weights, both driving an iterative array of 2-input multiplexer cells. In our implementation, the length of the CAR has been fixed to 22 bits and it uses an 8 bit register to quantize the weights.

From results of Table 2, it can be concluded that there exists a good agreement between the predicted and experimental test length values. For most circuits a significant reduction in test length, is obtained when using the equiprobable or the global optimized distributions. Additionally, note that for all the circuits, the global optimization-driven distribution of weights, performs better than the local driven one, and for some of them (s641, c880, and s953), this difference is about one order of magnitude.

Two special cases are the circuits s420, and s838. For these circuits the benefit of using single distributions of weights is not useful compared to the equiprobable distribution. These circuits are controllers that have been implemented using PLA-based structures [15]. It is well known that such structures are very hard to be tested using random patterns. Moreover, if a weighted random-pattern testing approach is selected, the use of multiple distributions of weights would be necessary [8], since the solution for a single distribution could be very close to the equiprobable case. However, in order to increase their random-pattern testability, these structures can be modified by adding special hardware, without making use of multiple distributions of weights.

4. Test Length Experiments

This section is inteded to show the performances of the proposed approach in terms of test length. For that reason we have fixed the fault coverage to some certain value (normally, because of the limited number of equiprobable random patterns that could handle the fault simulator), and we have compared the required test length to achieve these fault coverages between an equiprobable and weighted distribution.

Several small and large scale experiments have been prepared to check the test length performances of the single-distribution driven approach, for any circuit size. The small experiments correspond to the circuits presented in the previous section and for the large experiments we have selected the two most random resistant circuits from the ISCAS85 benchmark set (c2670, c7552), three medium size (s9234, s13207, s15859) ISCAS89 circuits, and two of the largest circuits (s38417, s38584), also from the ISCAS89 benchmark set.

Table 3 shows some of the circuit characteristics for the chosen subset to give an idea of their size and complexity.

Both pseudorandom pattern generators, equiprobable and weighted, have been implemented using a cellular automata formed with rules 90 and 150. The reason for choosing a CA register instead of a LFSR is that the CAR generator has a better chaotic behaviour than the LFSR pseudoandom generator [20]. To generate the weightedpatterns we have used the same weighting circuitry pesented in the previous section. For all the experiments the length of the CAR was fixed to 22 bit, and the register for weights was sized to 8 bit.

Circuit	collpsd. faults	redun. faults	inputs	outputs	gates	lines
s344	342	0	24	26	160	335
s526n	553	0	24	27	194	526
s641	463	0	54	42	379	637
s820	850	0	23	24	289	820
c880	942	0	60	26	383	880
s953	1079	0	45	52	395	953
s1488	1486	0	14	25	653	1488
c1908	1879	9	33	25	880	1908
c2670	2747	117	233	140	1193	2670
c3540	3428	137	50	22	1669	3540
s5378	4551	40	214	213	2779	5269
c7552	7559	131	207	108	3512	7552
s9234	6927	452	247	250	5597	9234
s13207	9815	151	700	790	7951	13179
s15850	11725	389	611	684	9972	15847
s38417	31186	171	1664	1742	22179	38339
s38584	36306	1509	1464	1730	19253	38432

Table 3. Characteristics of the circuits used for benchmarking.

Circuit	TLeqp.	TLwght.	Und. Faults	FC (%) (detctbl)	Reduc-tion
s344	436	142	0	100.00	3.07
s526n	21,231	1,033	0	100.00	20.55
s641	771,731	404	0	100.00	1910.46
s820	34,992	4,303	0	100.00	8.13
c880	19,142	272	0	100.00	70.37
s953	43,541	1,859	0	100.00	23.42
s1488	7,871	1,528	0	100.00	5.15
c1908	25,195	6,629	0	100.00	3.80
c2670	820,426	7,554	31	98.82	106.14
c3540	76,093	8,553	0	100.00	8.89
s5378	73,731	10,460	0	100.00	7.37
c7552	683,800	19,002	91	98.78	35.98
s9234	989,532	11,190	107	98.35	88.43
s13207	303,630	56,927	1	99.99	5.33
s15850	999,707	97,958	88	99.23	10.20
s38417	202,068	66,396	448	98.56	3.04
s38584	347,986	42,426	34	99.90	8.20

Table 4. Test length experiments.

Table 4 summarizes the test length results obtained after fault simulations, for both equiprobable and weighted pattern generation. *TLeqp* and *TLwght* are the test length sizes corresponding to an equiprobable and weighted distributions, respectively. *UndF* is the number of detectable faults that remain undetected after fault simulations, and *FC* is the associated fault coverage. Finally *Reduction* shows the reduction level achieved (*Reduction=TLeqp/TLwght*) using the set of weights obtained with the proposed approach.

It is important to remark that the previous results have been obtained using only one distribution of weights for each circuit, so these results are not comparable with solutions based on multiple distributions.

To have a graphical representation of the improvement on the fault coverage using highly optimized single distributions of weights, we have monitored in Figure 3, the fault coverage curves for the c2670, for both equiprobable and weighted distributions. Note that the predicted fault coverages are also displayed, showing the relative accuracy of the probabilistic fault model. The upper solid lines show the improvement, on the fault coverage, using weighted random patterns. Note that in the steeper sloped region in the begining of the curves, faults are easy-to-detect with respect to the equiprobable distribution, since the performance of the random patterns is already acceptable in this region. Note, that in the long flattened tail region at the end of the curves, an improvement of about 10% on the fault coverage is reached for the same test length. Furthermore, for the same fault coverage an improvement in orders of magnitude of the test length is observed.

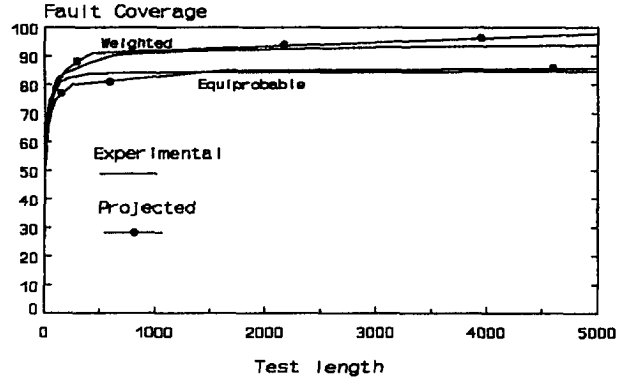

Figure 3. Fault coverage curves (experimental and projected) for the c2670, using equiprobable and weighted distributions.

5. Conclusions

In this paper a probabilistic-based fault-model approach to the generation of optimized single distributions of weights is presented. The proposed approach is oriented to reduce the memory area overhead introduced by the use of multiple distributions of weights, and therefore the proposed strategy is well suited for Built-In Self-Test.

Experiments over a subset of the ISCAS85 and ISCAS89 benchmark circuits show that important reduction levels in test length, compared to the equiprobable solution, can be obtained. It has been also demonstrated the importance of using a global optimization strategy in order to obtain highly optimized set of weights, independently of the fault model used. Of course, the price to pay is the

computing time necessary to perform a global optimization strategy. Nevertheless, the benefits of using the proposed approach can be very high also, since an important reduction in test application time can be obtained.

Acknowledgment

The authors thank Dr. Bill Linn for his many interesting discussions about the optimization procedure.

References

[1] R. Basset, B. Butkus, L. Dingle, M. Faucher, P. Gillis, J. Panner, J. Petrovick, and D. Wheater. Low-Cost Testing of High-Density Logic Components. *IEEE Design & Test of Computers*, pages 15-28, April 1990.

[2] H. Schnurmann, E. Lindbloom, and R. Carpenter. The Weighted Random Test-Pattern Generator. *IEEE Trans. on Computers*, c-24(7):695-700, July 1975.

[3] F. Siavoshi. WTPGA: A Novel Weighted Test_pattern Generation approach for VLSI Built-In Self-Test. In *Proc. IEEE Intl. Test Conference*, pages 256-262, 1988.

[4] J. Waicukauski, E. Lindbloom, E. Eichelberger, and O. Forlenza. A Method for Generating Weighted Random Test Patterns. *IBM Journal of Research and Development*, 33(2):149-161, March 1989.

[5] P.H. Bardell, W.H.McAnney, and J. Savir. Built-In Test for VLSI: Pseudorandom Techniques. *John Wiley & Sons*, New York, 1987, pages 217-225.

[6] W.H. Debany Jr., C.R. Hartmann, P.K. Varshney, and K.G. Mehrotra. Comparison of Random Test Vector Gneration Strategies. In *Proc. IEEE Intl. Conf. on Computer-Aided Design*, pages 244-247, 1991.

[7] R. Lisanke, F. Brglez, A. Degeus, and D. Gregory. Testability-Driven Random Test-Pattern Generation. *IEEE Trans. on Computer-Aided Design*, CAD-6(6):1082-1087, November 1987.

[8] H.J. Wunderlich. Multiple Distributions for Biased Random Test Patterns. *IEEE Trans. on Computer-Aided Design*, 9(6):584-593, June 1990.

[9] F. Muradali, V. Agarwal, and B. Nadeau-Dostie. A New Procedure for Weighted Random Built-In Self-Test. In *Proc. IEEE Intl. Test Conference*, pages 660-669, 1990.

[10] F. Brglez, C. Gloster, and G. Kedem. Hardware-Based Weighted Random Pattern Generation for Boundary Scan. In *Proc. IEEE Intl. Test Conference*, pages 264-274, 1989.

[11] F. Brglez, C. Gloster, and G. Kedem. Built-In Self Test with Weighted Random Pattern Hardware. In *Proc. Intl. Conf. Computer Design: VLSI in Computers and Processors*, pages 161-166, 1990.

[12] F. Brglez. On Testability of Combinational Networks. IN *Proc. Intl. Symp. on Circuits and Systems*, pages 221-225, 1984.

[13] T. Sakurai, B. Lin, and R. Newton. Fast Simulated Diffusion: An Optimization Algorithm for Multiminimum Problems and Its Application to MOSFET Model Parameter Extraction. *IEEE Trans. on Computer-Aided Design*, 11(2):228-234, February 1992.

[14] F. Brglez, and H. Fujiwara. A Neutral Netlist of Ten Combinational Benchmark Circuits and a Target Translator in FORTRAN. In *Proc. Intl. Symp. on Circuits and Systems* , June 1985.

[15] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits. In *Proc. Intl. Symp. on Circuits and Systems*, pages 1929-1934, 1989.

[16] H.J. Wunderlich. PROTEST: A Tool for Probabilistic Testability Analysis. In *Proc. IEEE 22nd Design Automation Conference*, pages 204-211, 1985.

[17] F. Aluffi-Pentini, V. Parisi, and F. Zirilli. A Global Optimization Algorithm Using Stochastic Differential Equations. *ACM Trans. Mathematical Software*, 14(4):345-365, December 1988.

[18] S. Kirkpatrick, C. Gellat, and P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671-690, May 1983.

[19] M. Huang, F. Romeo, and A. Sangiovanni-Vincentelli. An Efficient General Cooling Schedule for Simulated Annealing. In *Proc. IEEE Intl. Conf. on Computer-Aided Design*, pages 381-384, 1986.

[20] P. Hortensius, R. McLeod, W. Pries, D. Miller, and H. Card. Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test. *IEEE Trans. Computer-Aided Design*, 8(8):842-859, August 1989.