TESTCHIP: A Chip for Weighted Random Pattern Generation, Evaluation, and Test Control

Albrecht P. Ströle and Hans-Joachim Wunderlich, *Associate Member, IEEE*

Abstract—**In self-testable circuits additional hardware is incorporated for generating test patterns and evaluating test responses. In this paper a built-off test strategy is presented which moves the additional hardware to a programmable extra chip. This is a low-cost test strategy in three ways: 1) the use of random patterns eliminates the expensive test pattern computation; 2) a microcomputer and an ASIC replace the expensive automatic test equipment; and 3) the design for testability overheads are minimized. The presented ASIC generates random patterns, applies them to a circuit under test, and evaluates the test responses by signature analysis. It contains a hardware structure that can produce weighted random patterns corresponding to multiple programmable distributions. These patterns give a high fault coverage and allow short test lengths. A wide range of circuits can be tested as the only requirement is a scan path and no other test structures have to be built in.**

Index Terms—**Built-off test, low-cost test, multiple weights, random test, test equipment.**

I. Introduction

CONVENTIONAL test strategies using deterministic test pattern generation and automatic test equipment cause high costs and lead to severe problems, particularly for ASIC's with moderate production volumes. As an alternative, random pattern testing is attractive, since the computationally intensive automatic test pattern generation is eliminated and pseudorandom pattern generators can be built with a small amount of hardware.

The basic test configuration is shown in Fig. 1. A pattern generator (PG) produces bit patterns that are applied to the primary inputs of the circuit under test (CUT). The responses at the primary outputs of the CUT are fed to a test response compressor (TRC), e.g., a signature register. A control unit clocks the PG and the TRC and controls the CUT (reset, clock, etc.).

When sequential circuits are tested using this basic configuration, the test length may become very large, since a random pattern sequence to take the CUT to a given state can increase at a greater than exponential rate with the number of flip-flops of the CUT [19]. In order to avoid this, sequential circuits are usually provided with a scan path. In the test mode the circuit is partitioned into a combinational logic part and a set of storage elements configured a shift register chain. The test configuration must be modified (Fig. 2) since additional patterns are loaded serially into the scan path, and the output of the scan path is evaluated, too.

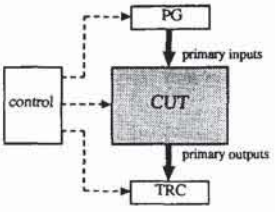

Fig. 1. Basic test configuration.

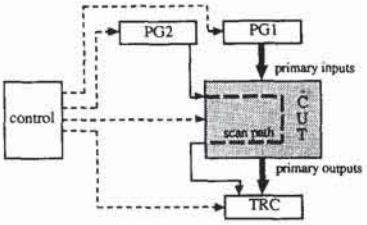

Fig. 2. Test configuration for a sequential CUT.

The tasks of pattern generation, response compression, and test control can be performed by automatic test equipment, which must be able to apply a large set of test patterns at high speed. High-speed testing is required for economical reasons in order to shorten the test time, and for technical reasons in order to obtain a certain product quality as some faults are not detectable otherwise. Usually the test patterns are precomputed and stored in a fast buffer.

Empirical studies have shown that the size of deterministic test sets increases in the order of the circuit complexity, but since a pattern is shifted into the scan path serially the testing time grows quadratically [7]. Moreover,

Manuscript received December 3, 1990; revised March 8, 1991.

A. P. Ströle is with the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, D-7500 Karlsruhe 1, Germany.

H.-J. Wunderlich is with the University of Duisburg, Duisburg, Germany, on leave from the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, D-7500 Karlsruhe 1, Germany.

IEEE Log Number 9100042.

these large test sets cannot entirely be stored in the buffer; they must be divided into several blocks and downloaded from some backup storage. Downloading requires a time which exceeds the duration of applying the patterns to the CUT by orders of magnitude. If randomly chosen bit patterns are applied to the inputs of the circuit under test, they can easily be generated on-line during the test execution, and only a small amount of data is required to define the entire test.

In [15] deterministic tests and random pattern tests are compared. Empirical results on a large number of LSSD logic chips of varying size give a range of about 10 to 50 times more weighted random patterns than deterministic patterns. But the larger number of random patterns can be applied by external test equipment in less time than a deterministic set, as no downloading is required. In addition, the increase in the number of patterns (*test length*) improves the ability to detect nonmodeled faults (e.g., shorts, delay and transition faults).

The automatic test equipment can be avoided if the PG's, the TRC, and the test control unit are implemented on the chip itself (built-in self-test) or moved to an extra chip (built-off test). An extra chip is often more advantageous, as the design effort and the silicon area for a built-in self-test are saved. Except for a scan path, no other test features need to be integrated into the CUT.

Such an approach for external random pattern generation was reported in [1] and [6]. A higher fault coverage is attainable if weighted patterns are used [18]. In [15] a complex system for weighted random pattern testing is described for the production test of LSSD logic chips at IBM. In this paper a single chip for a built-off test is presented that generates weighted random patterns corresponding to multiple distributions, performs signature analysis, and controls the whole process. The central parameters are programmable in order to adapt them to a wide range of CUT's. This makes the designed chip a key element in building low-cost test equipment. Currently the presented test system is intended to be used for testing chips of multichip projects designed by students.

This paper is an extension of [13]. Section II describes the characteristic features of a random pattern test using multiple distributions and its advantages for a built-off test. In Section III a weighted random pattern generator is presented that can be programmed for multiple distributions. The test configuration and the chip design are described in Section IV. Section V demonstrates the application of the chip. Finally, Section VI concludes with a short summary.

II. Random Pattern Test Corresponding to Multiple Distributions

Deterministic test patterns are computed in order to detect the faults of a certain fault model. A test pattern for a fault must result in a response at the primary outputs that differs from the fault-free case. This way of test generation requires a large amount of computation, the deterministic test set must be stored, and complex equipment is needed to execute the test at a sufficiently high speed. A random test dispenses with the time-consuming test pattern computation.

In the following only combinational circuits and sequential circuits with a complete scan path are considered. The extension to more general sequential circuits can be found in [19]. Let $I := \{i_1, \cdots, i_n\}$ be the primary inputs of a circuit S, and let F be the set of faults of S. A tuple $B := (b_1, \ldots, b_n)$ of Boolean random variables is assigned to the primary inputs I. A Boolean random variable b_i gets the value ONE with probability $x_i := P(b_i) \in [0, 1]$ and the value ZERO with probability $1 - x_i$. The Boolean random variables $b_1, \cdots, b_n$ are independent, if $\forall J \subset \{1, \cdots, n\}\ P(\prod_{j \in J} b_j) = \prod_{j \in J} x_j$ holds. The tuple $X := (x_1, \cdots, x_n) \in [0, 1]^n$ of real numbers (*input probabilities*) uniquely determines the tuple $B := (b_1, \cdots, b_n)$ of independent Boolean random variables and defines a distribution of the random patterns at the primary inputs.

The value of each node v of the circuit is a Boolean function of the values at the primary inputs. So the probability that the node v takes the value ONE (*signal probability*) is completely determined by the tuple X. Let $T(f)$ be the complete set of patterns that detects the fault f. Then $p_f(X) := P(B \in T(f))$ is the probability that a randomly chosen pattern B detects the fault f (fault detection probability). For an arbitrary tuple X where $0 < x_i < 1$ for $i = 1, \cdots, n$, each nonredundant fault $f \in F$ can be detected, hence all the detection probabilities $p_f(X)$ are positive. For $X = (0.5, \cdots, 0.5)$ all input patterns are equally likely and appear with probability $1/2^n$. In this case the fault detection probability is $P_f(0.5, \cdots, 0.5) = |T(f)|/2^n$. If a fault has only few test patterns, it results in a very small detection probability.

The probability that a fault f is detected by at least one pattern is $1 - (1 - p_f(X))^N$; the expected detection of a fault in a faulty circuit by a set of N input patterns (*fault coverage*) is (see [10])

$$FC(N, X) := 1 - \frac{1}{|F|} \cdot \sum_{f \in F} \left(1 - p_f(X)\right)^N. \qquad (1)$$

The probability $P(N, X)$ that each fault $f \in F$ is detected by N random patterns is estimated by the formula

$$P(N, X) \approx J_N(X) := \prod_{f \in F} \left[1 - \left(1 - p_f(X)\right)^N\right]. \quad (2)$$

The formula holds exactly if it is assumed that the detection of some faults by N patterns forms completely independent events. But even without this assumption it gives a very precise estimation [18].

Equations (1) and (2) show the interdependence between the test length N on the one side and the fault coverage and the probability of detecting all faults on the other side. Both $FC(N, X)$ and $P(N, X)$ increase strictly monotonically with the test length N. In order to obtain a predetermined test quality in terms of certain values of $FC(N, X)$ or $P(N, X)$, the test length chosen must be

sufficiently large. Reference [16] gives an efficient procedure to determine the test length for a given value $C < 1$ (*confidence*) of the probability $P(N, X)$.

Only the few faults $F' \subset F$ with lowest detection probability have impact on the required test length. In the worst case all faults of F' have the same minimal detection probability $p(X)$. Then (2) leads to

$$J_N(X) = \prod_{f \in F'} \left[1-(1-p(X))^N\right] = \left[1-(1-p(X))^N\right]^{|F'|}, \quad (3)$$

and the necessary number N of random patterns to reach a given probability $P(N, X) = C$, $C \approx 1$, can be estimated by

$$N \approx \frac{\ln(|F'|) - \ln(1-C)}{p(X)}. \quad (4)$$

The test length increases linearly with the reciprocal of the minimal fault detection probability. As a consequence, the test length for equally distributed input probabilities ($X = (0.5, \cdots, 0.5)$) may grow exponentially with the number of the primary outputs.

To reduce the test length, two approaches are known. One approach modifies the structure of the CUT and inserts additional gates and multiplexers in order to improve the controllability and observability of critical parts of the CUT (e.g., [5], [12]). But this does not conform to the intention of a built-off test, since it leads to additional hardware costs for the CUT. The other approach is to modify the random patterns. By changing the input probabilities x_i, the detection probabilities of the faults with lowest detection probabilities can be increased. This reduces the test length for a given confidence corresponding to (4). The input probabilities x_i, also called *weights*, are optimized such that the fault coverage or the probability of detecting all faults reaches a specified value with a minimum number of weighted random patterns. This can reduce test lengths by orders of magnitude. To test the ISCAS'85 benchmark circuit c880, for example, only 660 optimized random patterns are required rather than 37 000 patterns without optimization [4], [8].

Some circuits are resistant to this kind of optimization, when only one distribution (one tuple of weights) is used. Improving the fault detection probabilities in some parts of these circuits requires weights that change the fault detection probabilties in other parts for the worse and vice versa. Fig. 3 gives an example. When an attempt is made to optimize the weights for this circuit, the AND32 favors weights near 1, and the OR32 favors weights near 0. For the combination of an AND32 and an OR32, no better single tuple of weights exists than $X = (0.5, \cdots, 0.5)$. This would require a test length of $N \approx 4.8 \cdot 10^{10}$ to get a confidence of $C = 0.999$. The problem is solved by first applying 600 patterns with input probabilities $x_1 = \cdots = x_{32} = \sqrt[32]{0.5}$ and then 600 patterns with input probabilities $x_1 = \cdots = x_{32} = 1 - \sqrt[32]{0.5}$. This way the confidence $C = 0.999$ is obtained with $N = 1200$ random patterns.

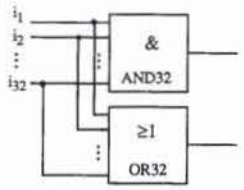

Fig. 3. Circuit that is not random testable using one distribution.

Multiple tuples of weights $X^1, \cdots, X^k$ can shorten the test length in all cases. Then the probability $P(N, X)$ to detect all faults of F is given by

$$P(N, X) \approx J_N(X) := \prod_{f \in F} \left[1 - \prod_{i=1}^{k} \left(1 - p_f(X^i)\right)^{N_i}\right], \quad N := \sum_{i=1}^{k} N_i. \quad (5)$$

Here a number k, k tuples of weights $X^i := (x_1^i, \cdots, x_n^i)$, and k numbers N_i, $i = 1, \cdots, k$, must be found such that $P(N, X)$ exceeds a given confidence value C and N is minimal. The test configuration presented in the following sections uses weighted random patterns corresponding to multiple distributions.

Methods to optimize the weights for a given CUT are reported in [10], [11], [15], [16], and [18]. Probabilistic testability analysis provides means to develop an objective function for the optimization procedure. A fault-coverage estimate based on controllability and observability measures or the probability that all faults are detected can be used. Formula (2) is transformed into

$$\ln(J_N(X)) \approx -\sum_{f \in F} \left(1 - p_f(X)\right)^N \approx -\sum_{f \in F} e^{-Np_f(X)} \quad (6)$$

using some well-known approximations. A tuple $X \in [0, 1]^n$ is called optimal, if the objective function

$$\delta_N^F(X) := \sum_{f \in F} e^{-Np_f(X)} \quad (7)$$

is minimum. Obviously this corresponds to the fact that the probability of detecting all faults by N patterns is maximum. The objective function (7) is strictly convex with respect to a single variable. The first partial derivative of (7) can be computed explicitly, and the optimal value for x_i, resulting in $\partial \delta_N^F(X)/\partial x_i = 0$, can be found by the bisection method. Applying this procedure to the variables $x_1, x_2, \cdots, x_n$ iteratively, an optimal single tuple X is determined.

For multiple distributions the set of faults F is partitioned into k subsets $F_1 \cup F_2 \cup \cdots \cup F_k = F$, and for each subset F_i an optimal tuple X^i of weights is computed. First, the set F is split into two subsets $F_1 \cup F_2 = F$,

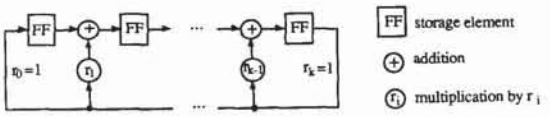

Fig. 4. Linear feedback shift register in modular form.

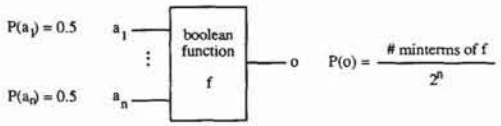

Fig. 5. Generation of a weighted random bit sequence.

such that

$$\delta_N^{F_1}(X^1) + \delta_N^{F_2}(X^2) := \sum_{f \in F_1} e^{-Np_f(X_1)} + \sum_{f \in F_2} e^{-Np_f(X_2)} < \delta_N^F(X) \quad (8)$$

is minimum for optimal tuples X, X^1, and X^2. The subsets F_1 and F_2 are constructed by enumerating the faults; a gradient optimization technique is used to decide whether to take a fault to F_1 or to F_2. This procedure is iteratively applied to the subsets that require the largest test sets, until the test length is sufficiently small.

III. Random Pattern Generation

A. Generation of Weighted Pseudorandom Patterns

The often used term "random" pattern is somewhat misleading as the patterns are not truly random patterns but algorithmically generated *pseudorandom* patterns. Pseudorandom patterns are reproducible. So the correct response of the circuit, the correct signature, and the fault coverage can be computed by simulation. The formulas of Section II, although developed for truly random patterns, give precise estimations for pseudorandom patterns, too, and the same optimization procedures can be applied [18]. Pseudorandom patterns never increase the test length.

Equally distributed pseudorandom patterns can be generated using a linear feedback shift register (LFSR) over the field $GF(2)$ as shown in Fig. 4. If the characteristic polynomial $r(x) = x^r + r_{k-1}x^{k-1} + \cdots + r_1 x + 1$, $r_i \in \{0,1\}$ for $i = 1, \cdots, k-1$, is primitive, the state transition diagram of the LFSR comprises a cycle of length $2^k - 1$ and a cycle of length 1, i.e., the all-ZERO state. When the LFSR is started in a state different from the all-ZERO state, it produces bit sequences of period $2^k - 1$ at all its stages. These bit sequences are pseudorandom sequences as they have features very similar to truly random sequences [8], [20]. ZERO and ONE have almost the same frequency. Within one period of length $2^k - 1$ there are $2^{k-1} - 1$ ZERO's and 2^{k-1} ONE's. A sequence of i times a ZERO (ONE) immediately followed by another is called a ZERO run (ONE run) of length i. In the produced bit sequence, $1/2^i$ of the ZERO runs and $1/2^i$ of the ONE runs have the length i. Furthermore, the autocorrelation is as small as possible.

For a weighted random test, random bit sequences with weights different from 0.5 are required. To this purpose the bits of several independent random bit sequences of weight 0.5 are combined by a Boolean function f (Fig. 5). The number of minterms of the function f determines the probability of a ONE in the output bit sequence. The output sequence is also a random sequence.

For generating weights with a quantization of 1/8, the following seven Boolean functions can be used:

$$f_1 = a_1 a_2 a_3$$
$$f_2 = a_2 a_3$$
$$f_3 = \overline{a_1 + a_2 a_3}$$
$$f_4 = a_3$$
$$f_5 = \overline{f_3} = a_1 + a_2 a_3$$
$$f_6 = \overline{f_2} = \overline{a_2 a_3}$$
$$f_7 = \overline{f_1} = \overline{a_1 a_2 a_3}.$$

As each function f_i, $i = 1, \cdots, 7$, has i minterms, it produces a pseudorandom bit sequence of weight $i/8$.

In recent years, several hardware structures have been proposed that generate weighted pseudorandom patterns [3], [11], [15], [17]. All of them are based on an LFSR with a primitive feedback polynomial. The LFSR is tapped at some stages and the pseudorandom bit streams at these stages are combined using Boolean functions and multiplexers to get the desired weights. The approaches differ in the selection of the tap positions and in the way the bit streams are combined. But none of them allows programmable distributions integrated on a single chip.

B. The Effect of Quantization

The computed optimized weights x_j^i for a given CUT are real numbers out of $[0,1]$. As only the subset of weights is used that can easily be implemented by Boolean functions as shown in Fig. 5, the realized input probabilities differ from the computed values and the test length is influenced.

Let X be a tuple of weights and $p_f(X|x_i = \alpha)$ the probability of detecting the fault f under the condition of $x_i = \alpha$ for a fixed value $\alpha \in \{0,1\}$. Applying the Shannon expansion to the fault detection probability $p_f(X)$ gives

$$p_f(X) = p_f(X|x_i = 0) - x_i \cdot \left[p_f(X|x_i = 0) - p_f(X|x_i = 1) \right]. \quad (9)$$

$p_f(X)$ depends linearly on the input probability x_i. For a tuple of weights $\tilde{X} := (x_1, \cdots, x_{i-1}, x_i + \Delta x_i, x_{i+1}, \cdots, x_n)$ with a quantization error Δx_i for the weight x_i, the fault detection probability is

$$p_f(\tilde{X}) = p_f(X) - \Delta x_i \cdot \left[p_f(X|x_i = 0) - p_f(X|x_i = 1) \right] \quad (10)$$

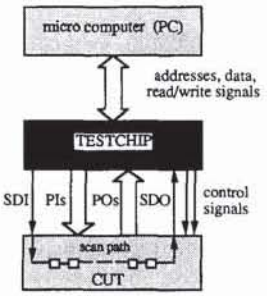

Fig. 6. Test configuration using TESTCHIP.

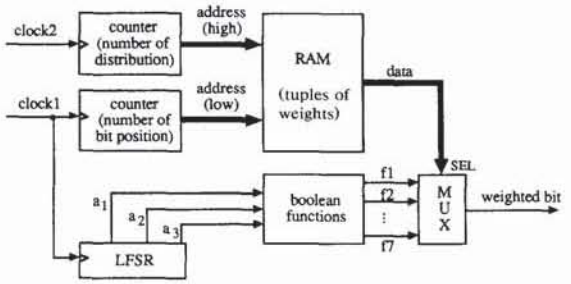

Fig. 7. Block diagram of the weighted pseudorandom pattern generator.

and using (4) the test length necessary to get the same probability of detecting all the faults of F is

$$N(\bar{X}) \approx N(X) \cdot \frac{p_f(X)}{p_f(\bar{X})} = N(X) \cdot \frac{1}{1 - \Delta x_i \cdot \dfrac{p_f(X|x_i=0) - p_f(X|x_i=1)}{p_f(X)}}. \tag{11}$$

A small quantization error Δx_i does not significantly change the test length as the factor $(p_f(X|x_i=0) - p_f(X|x_i=1))/p_f(X)$ is small. This allows the weights to be restricted to a small set of values, e.g., $\{1/8, 2/8, \cdots, 7/8\}$. A procedure to optimize the input probabilities using only a limited set of different weights is described in [20].

IV. Architecture of TESTCHIP

For a complete built-off test, patterns must be generated and applied to the CUT, the test responses must be collected and compressed, and the CUT must be controlled during the whole test execution. All these functions are integrated in an ASIC called TESTCHIP. Fig. 6 shows the test configuration. Instead of expensive test equipment, only low-cost hardware is used. At one end TESTCHIP is coupled with a personal computer (PC) or another microcomputer that supplies the user interface, initializes the test execution, and evaluates the results. On the other end it is connected to the CUT that is provided with a scan path (SDI: serial data in, SDO: serial data out). TESTCHIP generates patterns for the primary inputs (PI's) and the scan path of the CUT. It controls the test execution by means of a clock signal and a mode control signal (test mode or normal mode). Finally, TESTCHIP compresses the test responses from the primary outputs (PO's) and the scan path of the CUT.

A. Programming Weights for Pattern Generation

As a built-off test using weighted random patterns is not intended for just one specific CUT, a hardware pattern generator is required where the weights x_j^i for all the inputs i_j, $j = 1, \cdots, n$, and for all the tuples of weights X^i, $i = 1, \cdots, k$, can be programmed independently.

Fig. 7 shows the structure of the pattern generator. The modular LFSR with a primitive feedback polynomial forms the base. Its length is chosen such that the state sequence does not repeat during the whole test execution. It is tapped at three maximally spaced stages. There are several feedback connections between the tap positions. So the three produced pseudorandom bit sequences at the outputs are practically independent. To produce bit sequences with weights different from 0.5, these equally distributed sequences are combined using the Boolean functions $f_1, \cdots, f_7$.

The weights are coded by 3-b words and stored in the RAM. When patterns of width n are generated for k distributions, the RAM must contain $n \cdot k$ 3-b words. The RAM is divided into separate sections, one for each distribution. The n weights of a distribution are stored in subsequent cells of the same section. The counter for the lower part of the address cycles through the addresses of a section. Each of these addresses corresponds to a bit position within the generated pattern and selects the weight programmed for this bit position. By means of the multiplexer this weight chooses one bit from the appropriate weighted bit stream. The LFSR and the counter are clocked simultaneously. With every clock pulse a weighted random bit is produced, and a new pattern is composed serially bit by bit. This requires much less hardware than generating parallel patterns simultaneously. After sufficient patterns corresponding to the currently used distribution have been generated, the counter for the upper part of the address is incremented and switches to another section of the RAM and thus to another tuple of weights.

B. The Testing Procedure

A more detailed view of TESTCHIP is shown in Fig. 8. It contains two completely separated pattern generators to guarantee that the pseudorandom test patterns produced for the primary inputs and the scan path of the CUT are statistically independent. The pattern produced serially by pattern generator 2 is immediately shifted into the scan path. The pattern from pattern generator 1 is

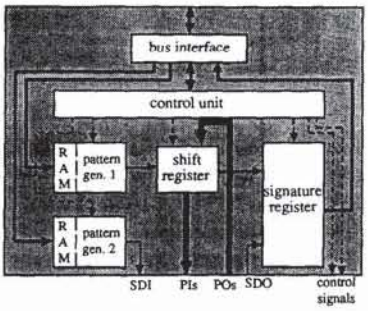

Fig. 8. Internal structure of TESTCHIP.

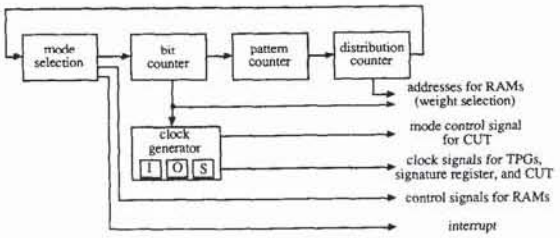

Fig. 9. Control unit.

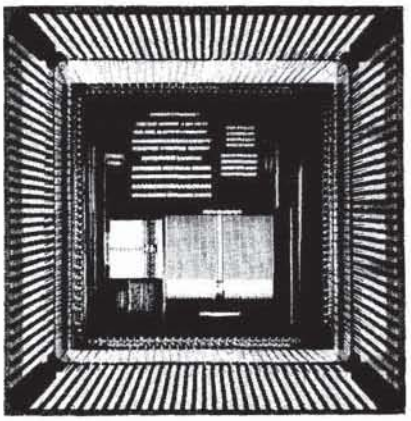

Fig. 10. TESTCHIP.

first shifted into an internal shift register of TESTCHIP (externally extensible) and then applied in parallel to the primary inputs. Usually the serial pattern generation does not increase the test execution time, since the scan path has to be loaded serially anyway, and in most cases the number of scan path elements is larger than the number of the primary inputs.

When the two patterns have been applied, the CUT is switched to normal mode and clocked once. This gives a test response in the storage elements of the scan path. At the same time the results of the primary outputs are loaded into the internal shift register of TESTCHIP (the data for the primary inputs are not required any more at that time). Afterwards the CUT is switched back to test mode. The contents of the shift register and the scan path are transferred to the two-input signature register, which compresses the test responses bit by bit. The implemented signature register is based on an LFSR with a primitive feedback polynomial of degree 32. The probability of aliasing (i.e., a faulty circuit leads to the same signature as the faultless circuit and thus the fault cannot be detected) is 2^{-32} for long test lengths [14]. Signature analysis and test pattern generation are performed simultaneously.

C. Control Unit

The control unit initializes all parts of TESTCHIP, supervises the loading of the weights into the RAM's, and controls the test execution. All the parameters, instructions, and status information are kept in registers, accessible to the microcomputer via an asynchronous bus interface. The control unit contains three cascaded counters (Fig. 9). The bit counter gives the bit position in the generated patterns and the test responses. Its contents are compared to the number of primary inputs (register I) and primary outputs (register O), and the number of scan path elements (register S). The results of these comparisons are used to mask the clock for shifting the scan path of the CUT and the clocks for the pattern generators and the signature register. When the bit counter has reached the maximum of the values in the registers I, O, and S, it is reset, the pattern counter is incremented, the mode control signal for the CUT is switched to normal mode for one clock period, and then the bit counter is started again.

For each distribution the number of patterns can be programmed independently. When the pattern counter reaches that value, the distribution counter is incremented and the other two counters are restarted. Eventually, when the distribution counter indicates an overflow, the whole process is stopped, and an interrupt signal for the connected microcomputer is activated. Also a flag in the status information register is set that can be polled by the microcomputer. The test execution can be stopped at any time by sending an instruction to TESTCHIP.

D. Implementation

TESTCHIP can be used for test circuits with up to 127 primary inputs, 127 primary outputs, and 511 scan path elements. Weights for up to four distributions and test lengths from 1 to 10^6 are programmable. If the requirements of a CUT exceed these features, several TESTCHIP's can be combined. The pattern generators and the signature analyzer can operate at a speed of $2 \cdot 10^7$ b/s.

TESTCHIP has been implemented using the standard cell design system VENUS [9]. Samples of the CMOS chip, which contains about 64 000 transistors, completely satisfy the specifications. The chip photo (Fig. 10) shows the RAM's containing the weights for the scan path

TABLE I
WEIGHTS FOR THE CIRCUIT $s1196$ (UNITS OF 1/8)

	distribution 1	distribution 2	distribution 3	distribution 4
test length (#patterns)	30145	49073	49073	25313
weights for the primary inputs	4,7,6,6,2,3,5, 4,6,5,4,7,4,2	7,5,6,7,4,1,7, 7,7,1,2,2,1,1	3,7,7,7,1,4,7, 7,7,7,2,6,6,2	3,6,7,7,3,3,6, 4,2,5,3,7,4,2
weights for the scan path elements	7,1,3,1,7,1,7, 7,7,7,1,7,1,1, 7,7,1,4	1,1,1,7,7,1,7, 7,7,7,1,1,1,1, 7,7,1,4	1,1,1,1,7,1,7, 1,1,1,1,1,1,1, 7,1,1,7	1,7,7,1,7,1,7, 7,7,4,1,1,1,1, 7,1,1,6

elements and the primary inputs (the block in the middle of the lower half and the smaller block to the left of it). The large standard cell block above them is the control unit. The three smaller standard cell blocks contain the LFSR's for pattern generation and signature analysis.

V. APPLICATION

The low-cost test configuration of Fig. 6 can be used for the test of all circuits with a scan path. As an example we take the ISCAS'89 benchmark circuit $s1196$ of [2] with 14 primary inputs, 14 primary outputs, and a scan path of 18 storage elements. The first step is to calculate the tuples of weights for the random patterns generated to test the circuit $s1196$. The resulting four tuples (using only weights $1/8, 2/8, \cdots, 7/8$) are listed in Table I. Then the test lengths to get a desired fault coverage of 99.9% are estimated and the fault coverage is validated by fault simulation. In the last step the signature for the fault-free circuit is determined.

The parameter registers and the RAM's of TESTCHIP are loaded with the characteristic data of the circuit $s1196$: number of primary inputs, primary outputs, and scan path elements, and the test lengths and weights of Table I. Whereas all these preparations have been done on the PC, now TESTCHIP carries out the test. The test execution takes 0.3 s. Afterwards the signature register of TESTCHIP is read, and its contents are compared with the expected signature in order to decide whether the circuit under test is faulty or not.

VI. CONCLUSIONS

A built-off test strategy has been described that uses weighted random patterns corresponding to multiple distributions. The presented chip generates random patterns, compresses the test responses, and controls the circuit under test. It contains pattern generators that allow multiple distributions with programmable weights. Thus it can be adapted to test a wide range of circuits with a scan path, and the test execution time is short. As no other modifications of the circuits are required, the design effort and silicon area for a built-in self-test are saved.

Samples of this CMOS chip have been produced and tested. In a built-off test configuration the chip can easily be connected to a microcomputer and thus constitutes the key element of low-cost test equipment.

REFERENCES

[1] P. H. Bardell and W. H. McAnney, "Self-testing of multichip logic modules," in *Proc. Int. Test Conf.*, 1982, pp. 200–204.

[2] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. Int. Symp. Circuits Syst.* (Portland, OR), 1989, pp. 1929–1934.

[3] F. Brglez, C. Gloster, and G. Kedem, "Hardware-based weighted random pattern generation for boundary scan," in *Proc. Int. Test Conf.* (Washington), 1989, pp. 264–274.

[4] F. Brglez, P. Pownall, and R. Hum, "Accelerated ATPG and fault grading via testability analysis," in *Proc. Int. Symp. Circuits Syst.*, 1985, pp. 695–698.

[5] A. J. Briers and K. A. E. Totton, "Random pattern testability by fast fault simulation," in *Proc. Int. Test Conf.* (Washington), 1986, pp. 274–281.

[6] E. B. Eichelberger, and E. Lindbloom, "Random-pattern coverage enhancement and diagnosis for LSSD logic self-test," *IBM J. Res. Develop.*, vol. 27, no. 3, May 1983.

[7] E. B. Eichelberger and E. Lindbloom, "Trends in VLSI-testing," in *Proc. VLSI'83, IFIP*.

[8] S. W. Golomb, *Shift Register Sequences*. Laguna Hills, CA: Aegan Park, 1982.

[9] E. Hörbst, M. Nett, and H. Schwärtzel, *VENUS — Entwurf von VLSI-Schaltungen*. Berlin: Springer, 1986 (in German).

[10] R. Lisanke, F. Brglez, A. de Geus, and D. Gregory, "Testability-driven random pattern generation," in *Proc. Int. Conf. Computer-Aided Design*, 1986, pp. 144–147.

[11] F. Muradali, V. D. Agarwal, and B. Nadeau-Dostie, "A new procedure for weighted random built-in self-test," in *Proc. Int. Test Conf.* (Washington), 1990, pp. 660–669.

[12] B. H. Seiss, P. M. Trouborts, and M. H. Schulz, "Test point insertion for scan-based BIST," in *Proc. European Test Conf. ETC'91* (Munich, Germany), 1991.

[13] A. P. Ströle, H.-J. Wunderlich, and O. F. Haberl, "TESTCHIP: A chip for weighted random pattern generation, evaluation, and test control," in *Proc. European Solid-State Circuits Conf. ESSCIRC'90* (Grenoble, France), pp. 101–104.

[14] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, "Aliasing errors in signature analysis registers," *IEEE Design & Test*, pp. 39–45, Apr. 1987.

[15] J. A. Waicukauski, E. Lindbloom, E. B. Eichelberger, and O. P. Forlenza, "A method for generating weighted random test patterns," *IBM J. Res. Develop.*, vol. 33, no. 2, pp. 149–161, Mar. 1989.

[16] H.-J. Wunderlich, "PROTEST: A tool for probabilistic testability analysis," in *Proc. 22nd Design Automation Conf.* (Las Vegas, NV), 1985, pp. 204–211.

[17] H.-J. Wunderlich, "Self-test using unequiprobable random patterns," in *Proc. Int. Symp. Fault-Tolerant Computing* (Pittsburgh, PA), 1987, FTCS-17, pp. 258–263.

[18] H.-J. Wunderlich, "Multiple distributions for biased random test patterns," in *Proc. Int. Test Conf.* (Washington), 1988, pp. 236–244.

[19] H.-J. Wunderlich, "The design of random-testable sequential circuits," in *Proc. Int. Symp. Fault-Tolerant Computing* (Chicago), 1989, FTCS-19, pp. 110–117.

[20] H.-J. Wunderlich, *Hochintegrierte Schaltungen: Prüfgerechter Entwurf und Test*. Berlin: Springer, 1991 (in German).

Albrecht P. Ströle received the diploma degree in electrical engineering from the University of Darmstadt, Germany, in 1980. In 1988 he joined the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, Germany, where he is currently working towards the Ph.D. degree.

From 1981 to 1987 he was with Siemens AG where he was involved in image processing and computer architecture. His research interests include built-in self-test techniques, signature analysis, and cellular automata.

Hans-Joachim Wunderlich (A'86) received the diploma degree in mathematics from the University of Freiburg, Germany, in 1981, and the Dr. rer. nat. (Ph.D.) degree from the University of Karlsruhe, Germany, in 1986.

In 1982 he was a consultant at the Fraunhofer-Institute of Industrial Engineering, Stuttgart, Germany, where he worked in the field of operations research. In 1983, he joined the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, where he has been the head of a research group on automation of circuit design and testing since 1986. His research interests include computer-aided design for testability, test generation, and digital simulation. Currently he is a Visiting Full Professor at the University of Duisburg, Germany.