IEICE TRANS. INF. & SYST., VOL.E93–D, NO.10 OCTOBER 2010

PAPER

A Practical Threshold Test Generation for Error Tolerant Application

Hideyuki ICHIHARA[†a)], ***Member***, **Kenta SUTOH**[†*], ***Nonmember***, **Yuki YOSHIKAWA**[†], ***Member***, ***and*** **Tomoo INOUE**[†], ***Senior Member***

SUMMARY Threshold testing, which is an LSI testing method based on the acceptability of faults, is effective in yield enhancement of LSIs and selective hardening for LSI systems. In this paper, we propose test generation models for threshold test generation. Using the proposed models, we can efficiently identify acceptable faults and generate test patterns for unacceptable faults with a general test generation algorithm, i.e., without a test generation algorithm specialized for threshold testing. Experimental results show that our approach is, in practice, effective.

key words: *acceptable fault, test generation model, error significance, threshold testing and error tolerance*

1. Introduction

The progress in LSI process technology broadens the applications of LSIs. Some specific applications such as images, video, audio, graphics and games, do not require high reliability in LSIs, and thus some errors, which are of certain types and/or have severities within certain limits, are tolerable for these applications. A circuit is error tolerant with respect to an application or system if it contains any fault that causes external errors, and if the system including the circuit produces acceptable results [1]. Several useful techniques based on error tolerance have been proposed [1]–[9], and the domain in which error tolerance might be applicable further extends.

A fault that causes only such tolerable errors is called an acceptable fault [2]. For example, the motion estimation block of an MPEG encoder has many acceptable faults, which cause no or a small amount of distortion in the output [3]. A direct benefit of the consideration of acceptable faults is yield enhancement of LSI chips [2], [4], [5], i.e., we can treat the defective chips including only acceptable faults as acceptable. The authors of [2] proposed an efficient testing method based on the acceptability of faults. The testing method can distinguish good (perfect or acceptably faulty) chips from bad (unacceptably faulty) chips. In [5], the authors proposed an approach to re-design datapath modules to exploit acceptable faults in order to improve yield. Another benefit is low-cost hardening against soft errors [6], [7]. Since, according to the acceptability of faults, we can select parts that should be hardened in an LSI, the hardening cost of an LSI system is very low.

For error severity, two key metrics, error significance and error rate, are well known [8]. For a faulty circuit, error significance for a set of circuit outputs is defined as the maximum amount by which the response at the set of outputs can deviate from the corresponding error-free value. Error rate, on the other hand, is defined as the percentage of vectors applied during normal circuit operation for which the value at a set of outputs deviates from the corresponding error-free value. In this paper, we focus on error significance.

The authors of [2] proposed an LSI testing method, called threshold testing, based on error significance. Given a threshold, which is a numeric number, threshold testing distinguishes bad chips, whose error significance is greater than the threshold, from good chips, whose error significance is smaller than the threshold or which have no error. The authors of [2] also proposed a test generation method for threshold testing. The threshold test generation aims to identify acceptable faults and generate test patterns, called threshold test patterns, only for unacceptable faults, Since its algorithm is specially tailored to threshold testing with 16-valued logic, it can generate effective test patterns for threshold testing.

In this paper, we propose a practical method for threshold test generation by means of general test generation algorithms (or general commercial ATPGs). Namely, unlike [2], we do not require test generation algorithms specialized for threshold test generation. To utilize general test generation algorithms for threshold test generation, we introduce two test generation models: a difference model and an acceptable fault identification model. The difference model is complete for generating threshold test patterns so that, theoretically, we can identify all acceptable faults and generate test patterns for all unacceptable faults with the difference model. The acceptable fault identification model is introduced for identifying acceptable faults, and it is used for accelerating the proposed threshold test generation. Using two models, we propose an effective test generation flow with a general ATPG and fault simulator. Experimental results show the effectiveness of our threshold test generation with two models.

Manuscript received December 10, 2009.
Manuscript revised May 12, 2010.

[†]The authors are with Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
[*]Presently, with Renesas Technology Corp.
a) E-mail: ichihara@hiroshima-cu.ac.jp

DOI: 10.1587/transinf.E93.D.2776

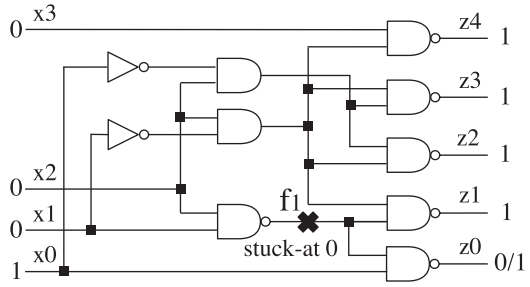


Fig. 1 Error of acceptable fault f_1.

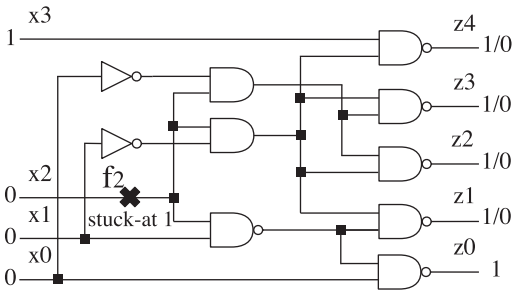


Fig. 2 Error of unacceptable fault f_2.

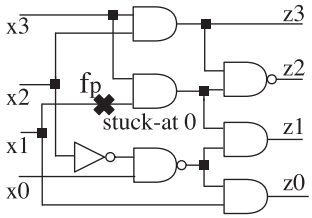


Fig. 3 Acceptable fault f_p.

2. Acceptable Fault and Threshold Testing

2.1 Acceptable Fault

Consider a combinational circuit C and its fault set F. The values of the input and its output for circuit C are denoted by x and $z(x)$, respectively. Here, the output value $z(x)$ expresses a non-negative numeric number and is defined as $z(x) = z_{n-1}(x)2^{n-1} + z_{n-2}(x)2^{n-2} + \ldots + z_0(x)2^0$, where n is the width of the output of circuit C and $z_i(x)$ is the binary value of the i-th output signal line. Similarly, the output value of faulty circuit C_f with fault f ($\in F$) is denoted by $z_f(x)$ when an input value x is given to C_f.

Error $e_f(x)$ of fault f for input value x is the difference between the output values of a fault-free circuit C and the faulty circuit C_f, i.e., $e_f(x) = |z_f(x) - z(x)|$. Figure 1 shows a circuit with stuck-at-0 fault f_1. The value, 0/1, of primary output z_0 denotes that its fault-free and faulty values are 0 and 1, respectively. Namely, the output value of this faulty circuit is $(z_4, z_3, z_2, z_1, z_0) = (1, 1, 1, 1, 1)$ and that of the fault-free circuit is $(1, 1, 1, 1, 0)$ when input value $(x_3, x_2, x_1, x_0) = (0, 0, 0, 1)$ is given. Thus, the error $e_{f_1}(0, 0, 0, 1)$ is $|31 - 30| = 1$.

Given a set X of input values, error significance $E(f)$ of fault f is defined by $E(f) = \max_{x \in X}\{e_f(x)\}$. For stuck-at-0 fault f_1 in the circuit of Fig. 1, the error $e_{f_1}(0, 1, 0, 1)$ is maximum of all the output errors and it is three; $E(f_1) = 3$.

Definition (Acceptable fault): If the error significance $E(f)$ of a fault f is smaller than a threshold T, i.e., $E(f) < T$, fault f is *acceptable* under threshold T.

If the error significance of a fault f is greater than or equal to a threshold T, fault f is *unacceptable* under threshold T. Consider the acceptability of faults f_1 and f_2 in Figs. 1 and 2 under threshold $T = 4$. Fault f_1 is acceptable under T since the error significance $E(f_1)$ of fault f_1 is 3, while fault f_2 is unacceptable because there exists a test pattern, $(1, 0, 0, 0)$, such that the error size for the pattern is over threshold T, $e_{f_2}(1, 0, 0, 0) = 30 > T$.

Note that threshold T does not uniquely specify the outputs where the effect of acceptable faults appears. Let us consider an acceptable fault f_p under $T = 4$ in another circuit of Fig. 3. The effect of fault f_p is propagated only by input patterns $(x_3, x_2, x_1, x_0) = (1, 0, 1, 0)$ and $(1, 1, 1, \mathrm{X})$, to output z_1 and to both of outputs z_2 and z_1, as $(z_3, z_2, z_1, z_0) = (0, 1, 1/0, 1)$ and $(1, 0/1, 1/0, 1)$, respectively. In either case, the error is two, which is smaller that T, such as $e_{f_p}(1, 0, 1, 0) = |7 - 5| = 2$ and $e_{f_p}(1, 1, 1, 1) = |11 - 13| = 2$.

Threshold T is given according to various circumstances of combinational circuit C. For example, when circuit C is a part of a system, threshold T should be determined with respect to the acceptability of the system's output. When circuit C is a combinational part of a sequential circuit, considering the effect of the feedbacks in sequential circuits, we can give a sufficient threshold T.

2.2 Threshold Test Generation

Threshold testing can be defined as a type of testing that distinguishes bad chips, including unacceptable faults, from good chips, including only acceptable faults or no faults, when a threshold is given [2]. It employs special test patterns, called *threshold test patterns*. The objective of threshold test patterns is different from that of general test patterns: a general test pattern aims to propagate the effect of faults to at least one primary output, while a threshold test pattern can not only propagate the effect of unacceptable faults to primary outputs but also the error $e_f(x)$ is greater than or equal to a given threshold T, $e_f(x) \geq T$. Having this property, threshold test patterns can distinguish bad chips from good chips, i.e., applying threshold test patterns to a chip derives only smaller errors than the threshold or no error, the chip is said to be good.

Consider an application of a test pattern $(x_3, x_2, x_1, x_0) = (1, 0, 1, 1)$ for fault f_2 when a given threshold is four. Although this test pattern can propagate the effect of fault f_2 to primary output z_0, it is not a threshold test pattern because $e_{f_2}(1, 0, 1, 1) = 1 < 4$. Note that the test pattern $(1, 0, 0, 0)$ shown in Fig. 2 is a threshold test pattern because

$e_{f_2}(1,0,0,0) = 30 > 4$.

Threshold test generation aims to identify acceptable faults from unacceptable faults and generate threshold test patterns for only unacceptable faults. In [2], a threshold test generation algorithm tailored to threshold testing has been proposed. The threshold test generation algorithm is an extension of PODEM [10] with 16-valued logic. During generating a test pattern, the algorithm always calculates the range of the error at the primary outputs; it backtracks when the error becomes smaller than a threshold, and it finishes when the error is equal to or greater than the threshold.

3. Threshold Test Generation with General Test Generation Algorithms

In this section, we propose a threshold test generation method with a general test generation algorithm. This means a general commercial test generation tool is used for threshold test generation, instead of test generation algorithms specialized for threshold test generation such as [2].

The proposed threshold test generation method employs two test generation models, which are converted from a circuit-under-test. We introduce the models, and then propose a test generation flow with two models.

3.1 Difference Model

A *DIFFerence model (DIFF model)* is a straightforward model for generating threshold test patterns. Using a DIFF model, we can distinguish between acceptable and unacceptable faults, and generate test patterns for unacceptable faults.

Figure 4 shows a DIFF model $M(C,T)$ for a circuit C under a threshold T. This model consists of two copies C_n and C_f of the original circuit C, and an absolutely differential circuit DIFF_ABS, and a comparator CMP. DIFF_ABS calculates the absolute difference, $|X-Y|$, between inputs X and Y, and CMP outputs zero when $|X-Y| < T$ and one when $|X-Y| \geq T$. Accordingly, this circuit indicates whether the absolute difference between the outputs of C_n and C_f is greater than a given threshold T.

A fault set of circuit C_f in a DIFF model is denoted by F_t, and fault f' ($\in F_t$) corresponds to fault f ($\in F$), where F is the fault set of the original circuit C. Here, a fault f_t in F_t corresponds to a fault f in F, and vice versa, if the location of fault f_t in C_f is the same as that of fault f in C and the faulty values of two faults f_t and f are identical.

Figure 5 shows the DIFF model for the combinational circuit shown in Fig. 1. This DIFF model has a fault f_1' ($\in F_t$) and an input pattern $(PI_3, PI_2, PI_1, PI_0) = (1,0,0,1)$ is applied to the model. A given threshold is four. For this pattern, the output values of C_n and C_f are 12 and 12/13, respectively. Here, value 12/13 means that the fault-free value is 12 and the faulty one is 13. Accordingly, since DIFF_ABS calculates the absolute difference between the two output values, the output value of DIFF_ABS is 0/1. Consequently, the output value of the primary output PO is always 0 because both the fault-free and faulty values are smaller than the threshold, 4.

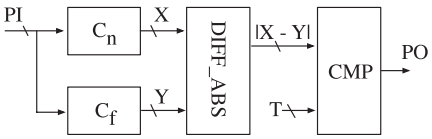


Fig. 4 DIFF model.

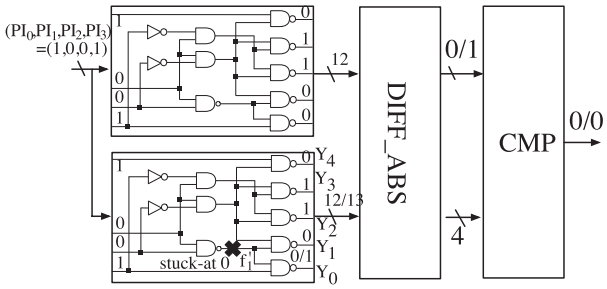


Fig. 5 Acceptable fault identification with DIFF model.

Theorem 1: For DIFF model $M(C,T)$, if a fault f' ($\in F_t$) is redundant, the corresponding fault f ($\in F$) in the original circuit is acceptable under threshold T. If not, the corresponding fault f is unacceptable, and a test pattern for fault f' is a threshold test pattern for fault f.

Proof: Let us suppose the fault-free case of $M(C,T)$. In this case, the output value of C_n is identical to that of C_f, i.e., $X = Y$, for any input pattern because C_n is identical to C_f, so that the output value of DIFF_ABS is always zero; the output value of CMP is always zero.

Next, we suppose the faulty case of $M(C,T)$ with fault f'. If f' is redundant, the value of the primary output, PO, is zero for any test pattern. This means that there are no patterns making the difference $|X-Y|$ greater than T, and therefore the corresponding fault f is acceptable. On the other hand, if f' is not redundant, there exists at least one test pattern by which the value of the primary output of the faulty circuit is set to one, i.e., the propagated fault-free/faulty value is D (or 0/1) to the primary output. Accordingly, such a test pattern makes the difference $|X-Y|$ greater than T, so that the test pattern is a threshold test pattern for fault f'. Consequently, fault f' is unacceptable. □

Fault f_1' shown in Fig. 5 is redundant because there is no pattern to propagate the effect to the primary output PO. For example, the input $(1,0,0,1)$ can propagate the effect of fault f_1' to the output of DIFF_ABS. However, both the fault-free and faulty values of primary output PO are zero, as explained above. Like this pattern, any pattern cannot propagate the effect of fault f_1' to PO. According to Theorem 1, the fault corresponding to f_1', i.e., fault f_1 shown in Fig. 1, is acceptable.

According to Theorem 1, a DIFF model is complete in terms of threshold test generation, that is, applying a general test generation algorithm to a DIFF model, we can obtain threshold test patterns for all unacceptable faults and iden-

tify all acceptable faults.

3.2 Acceptable Fault Identification Model

An *Acceptable fault IDentification model (AID model)* aims to identify a part of acceptable faults. The DIFF model proposed in the previous section may require considerably time if it is simply applied for test generation because it is almost twice as large as the original circuit and its structure is complex. In contrast, since an AID model is almost the same as the original circuit and its structure is simple, it can contribute for accelerating the threshold test generation with the DIFF model. Note that the identification of unacceptable faults with AID models is as helpful for threshold test generation as redundant fault identification techniques, such as [11], are helpful for general test generation.

An AID model is a circuit whose outputs are masked with an appropriate set of masks. For example, Fig. 6 shows three AID models for the circuit shown in Fig. 1. Output z_0 of the first circuit, named C_1, is masked, i.e., it is connected to an AND gate whose off-input is fixed to zero.

The masked outputs of an AID model are specified by a vector $m = (a_{n-1}, a_{n-2}, \ldots, a_0)$. When $a_i = 1$ ($a_i = 0$), the output z_i is masked (not masked). The weight $w(m)$ for a vector m is defined as $w(m) = \sum_{i=0}^{n-1} a_i \cdot 2^i$. An AID model specified by vector m is denoted by $C_{w(m)}$ or C_m. For example, three AID models in Fig. 6 have vectors $m = (0,0,0,0,1)$, $(0,0,0,1,1)$ and $(0,0,1,0,0)$, and denoted by C_1, C_3 and C_4 (or $C_{(0,0,0,0,1)}$, $C_{(0,0,0,1,1)}$ and $C_{(0,0,1,0,0)}$), respectively. Moreover, a fault set for model C_m is denoted by F_m and a fault f' ($\in F_m$) corresponds to the fault f. Note that the relationship between a fault and the corresponding fault is the same as described above.

Lemma 1: If a fault f' ($\in F_m$) in an AID model C_m is redundant, the fault f ($\in F$) corresponding to f' in the original circuit C is acceptable under a threshold T such that $T > w(m)$.

Proof: When fault f' is redundant, we can consider two cases where the corresponding fault f is also redundant or not. In the case that fault f is redundant, no effect is propagated to any primary outputs. Therefore fault f is acceptable under any threshold. Next, consider the case that fault f is not redundant in circuit C and its effect is propagated to a primary output z_p of C. Nevertheless, the corresponding fault f' in C_m is redundant, i.e., its effect cannot be propagated to even primary output z'_p, which is the output of the AND gate masking z_p, of C_m; this means that the mask of z_p prevents the effect of fault f' from propagating to z'_p. Here, the error significance of fault f is at most $w(m)$, when the effect of fault f' is propagated to all the masked primary outputs. Accordingly, fault f is acceptable under threshold $T > w(m)$. □

Let us denote the set of all AID models for circuit C by $\boldsymbol{C} = \{C_m | \ m = \{(0,0,\ldots,0),(0,0,\ldots,1),\ldots,(1,1,\ldots,1)\}\}$. The *AID model set* $\boldsymbol{C_T}$ *for threshold* T, a subset of $\boldsymbol{C}$, is defined as

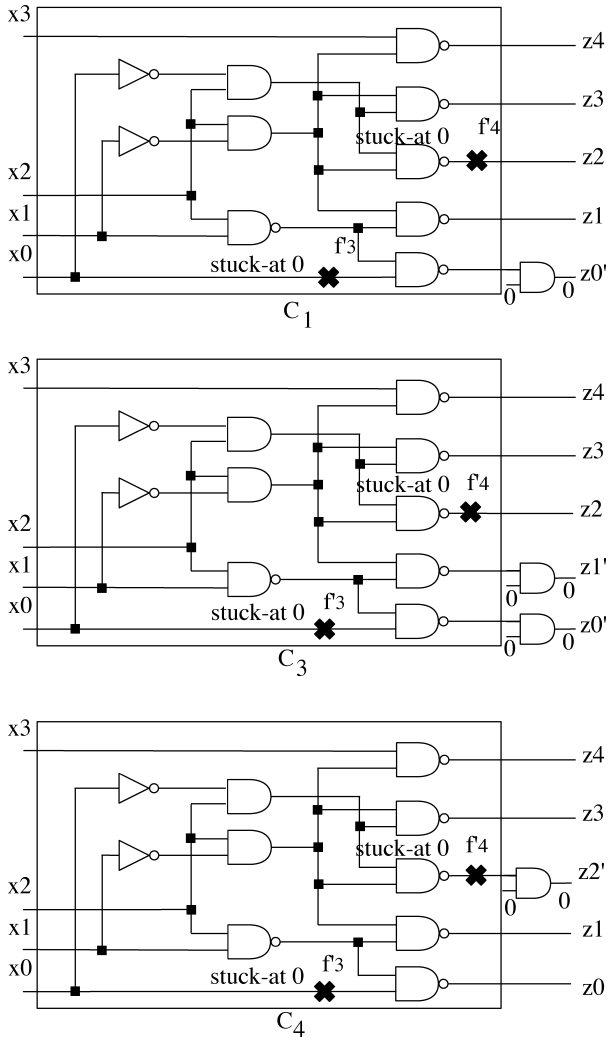


Fig. 6 AID models.

$$\boldsymbol{C_T} = \{C_m \in \boldsymbol{C} | w(m) < T\}.$$

For example, when threshold T is five, $\boldsymbol{C_5} = \{C_1, C_2, C_3, C_4\}$.

According to Lemma 1, if a fault f' in any AID model C_m ($\in \boldsymbol{C_T}$) is redundant, the corresponding fault f in the original circuit C is acceptable under threshold T. Such a fault f is called an acceptable fault identified with the AID model set $\boldsymbol{C_T}$ for threshold T. For example, the AID models C_1, C_3 and C_4 shown in Fig. 6 are included in $\boldsymbol{C_5}$, and fault f'_3 is redundant in models C_1 and C_3, and fault f'_4 is redundant in model C_4. Accordingly, the corresponding faults f_3 and f_4 are acceptable faults identified with $\boldsymbol{C_5}$. Note that Lemma 1 gives a sufficient condition for the acceptability of faults, and therefore all acceptable faults under threshold T are not identified with the AID model set $\boldsymbol{C_T}$ for threshold T.

In general, an acceptable fault identified with an AID model C_p is not necessarily identified with an AID model C_q such that $p < q$. For instance, acceptable fault f_4 corresponding to fault f'_4 in Fig. 6 is identified with model C_4 but not with models C_1 and C_3. To discuss this relationship, we

define the following relation between AID models.

Definition (Relation between AID models): A relation $\preceq$ between two AID models C_{m_1} and C_{m_2}, where $m_1 = (a_{n-1}, a_{n-2}, \ldots, a_0)$ and $m_2 = (b_{n-1}, b_{n-2}, \ldots, b_0)$, are defined as follows.

$$C_{m_1} \preceq C_{m_2} \stackrel{\text{def}}{\Leftrightarrow} \forall i, 0 \leq i < n[a_i \leq b_i].$$

Note that relation $\preceq$ is partially ordered according to this definition.

Lemma 2: Consider two AID models C_{m_1} and C_{m_2} ($\in \boldsymbol{C_T}$). Let f'_{m1} and f'_{m2} be faults in C_{m_1} and C_{m_2}, respectively, and correspond to a fault f in the original circuit C. When $C_{m_1} \preceq C_{m_2}$, if f'_{m1} is redundant, f'_{m2} is also redundant.

Proof: Assume that $m_1 = (a_{n-1}, a_{n-2}, \ldots, a_0)$ and $m_2 = (b_{n-1}, b_{n-2}, \ldots, b_0)$. Since $C_{m_1} \preceq C_{m_2}$, if $a_i = 1$, then $b_i = 1$. Namely, if the i-th output of AID model C_{m_1} is masked, then the i-th output of AID model C_{m_2} is also masked. This means that, if the effect of fault f'_{m1} in C_{m_1} is propagated to a primary output but masked, then the effect of fault f'_{m2} in C_{m_2} is also necessarily masked. Accordingly, if f'_{m1} is redundant, f'_{m2} is also redundan. □

For example, since $C_1 \preceq C_3$, f'_3 of C_1 in Fig. 6 is redundant, while f'_3 of C_3 is also redundant. In other words, fault f_3 is identified as acceptable with model C_1, while it is also identified as acceptable with model C_3.

A set $\boldsymbol{C_T}$ of AID models under threshold T is partially ordered with relation $\preceq$. A set $\widehat{\boldsymbol{C}}_{\boldsymbol{T}}$ ($\subseteq \boldsymbol{C_T}$) is said to be maximal if the following condition is satisfied.

$$\widehat{\boldsymbol{C}}_{\boldsymbol{T}} = \{\widehat{C}_m | \forall C_m \in \boldsymbol{C_T}[\widehat{C}_m \preceq C_m \Rightarrow C_m = \widehat{C}_m]\}.$$

When the threshold is five, $\widehat{\boldsymbol{C}}_{\boldsymbol{5}} = \{C_3, C_4\}$ because $C_1 \preceq C_3$ and $C_2 \preceq C_3$.

Theorem 2: The faults that are identified with an AID model set $\boldsymbol{C_T}$ under threshold T are also identified with the maximal AID model set $\widehat{\boldsymbol{C}}_{\boldsymbol{T}}$.

Proof: According to the definition of maximal AID model sets, $\forall C_{m2} \in \boldsymbol{C_T}, \exists C_{m1} \in \widehat{\boldsymbol{C}}_{\boldsymbol{T}}[C_{m2} \preceq C_{m1}]$. Moreover, from Lemma 2, any redundant fault in any $C_{m2} \in \boldsymbol{C_T}$ is also redundant in some $C_{m1} \in \widehat{\boldsymbol{C}}_{\boldsymbol{T}}$. Thus, according to Lemma 1, this proposition is true. □

Theorem 2 shows that the acceptable fault set identified with $\boldsymbol{C_T}$ is identical to the fault set identified with $\widehat{\boldsymbol{C}}_{\boldsymbol{T}}$. Consequently, we utilize only the models in the maximal AID model set for identifying acceptable faults in the proposed test generation method. In the case where threshold $T = 5$, we utilize only two AID models C_3 and C_4 out of C_1, C_2, C_3 and C_4.

3.3 Test Generation Flow

The proposed test generation flow with the two kinds of models is shown in Fig. 7. Given a circuit-under-test C with fault set F and a threshold T, we construct the DIFF model $M(C, T)$ and the maximal AID model set $\widehat{\boldsymbol{C}}_{\boldsymbol{T}}$, explained above. First, by means of $\widehat{\boldsymbol{C}}_{\boldsymbol{T}}$, we identify an acceptable fault subset F'_a of F, and store generated test patterns as a set T'. Note that the faults detected in this step are possibly unacceptable because they are activated and their effects are propagated to at least one primary output of a maximal AID model by test patterns in T'.

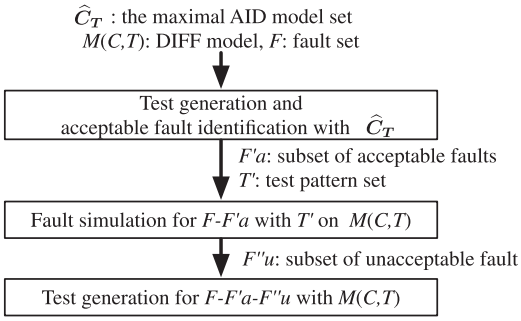


Fig. 7 Test generation flow.

Next, to clarify the acceptability of such possibly unacceptable faults in the remaining fault set $F - F'_a$, we perform fault simulation on the DIFF model $M(C, T)$. In this simulation, for faults in $F - F'_a$, test patterns in T' are applied to $M(C, T)$ and then the set F''_u ($\subseteq F - F'_a$) of faults detected by T' is obtained. Because the DIFF model is complete in terms of threshold test generation as described in Sect. 3.1, any fault in subset F''_u is guaranteed to be unacceptable. We also obtain a test pattern set T'', which is a subset of T', comprising test patterns that detect faults in this simulation.

Finally, we generate test patterns for faults in $F - F'_a - F''_u$ with $M(C, T)$. In this step, we can completely identify acceptable faults and generate test patterns T''' for the remaining unacceptable faults. The resultant test pattern set is $T'' \cup T'''$.

4. Experimental Results

We implemented the proposed threshold test generation flow and applied it to ISCAS'85 benchmark circuits on Dell PowerEdge III (OS: Redhat Linux, CPU: Xeon 2.33 GHz Quad, Memory: 4 GB). As a general ATPG and fault simulator, we employed TetraMAX (Synopsys Inc.). The proposed models are automatically generated from a given benchmark circuit, which is specified with VerilogHDL. This transformation requires negligibly small computational time. Stuck-at fault model is used in this experiment.

Table 1 shows the experimental results. Following circuit names and given thresholds, four columns show results of test generation with only DIFF models, i.e., without using AID models. The number under circuit names is the total number of faults. Columns *acc*, *un*, and *abort* correspond to the numbers of acceptable faults, unacceptable faults and aborted faults, respectively. The backtrack limits are set to 100 for c880 and c3540, 10,000 for c432, c6288 and c7552, and 15,000 for c5315. Column *time* is the test generation time in second. The following nine columns show results of the proposed test generation flow. Columns $|F'_a|$ and $|F''_u|$ mean the size of fault sets F'_a and F''_u introduced in Sect. 3.3, respectively. The computational time for each step and the

Table 1 Experimental results.

		with only DIFF model				Proposed flow									comparison	
						1st		2nd		3rd				total	time	abort
circ.	T	acc	un	abort	time	$\|F'_a\|$	time	$\|F''_u\|$	time	acc	un	abort	time	time	ratio	diff.
c432	2	40	824	0	2.31	40	0.36	821	0	0	3	0	0.2	0.56	4.1	0
(864)	4	87	777	0	7	87	0.74	777	-	-	-	-	-	0.74	9.5	0
	5	36	758	70	9.06	97	0.8	758	0	0	0	9	1.06	1.86	4.9	−61
	6	107	742	15	12.29	104	0.85	742	0	18	0	0	2.54	3.39	3.6	−15
c880	2	34	1684	42	0.13	74	0.01	1678	0	0	8	0	0	0.01	13	−42
(1760)	8	103	1504	153	0.36	256	0.02	1477	0.01	0	27	0	0.01	0.04	9	−153
	32	165	1368	227	0.46	392	0.01	1368	-	-	-	-	-	0.01	46	−227
c3540	2	260	6818	2	0.24	261	0.07	6819	-	-	-	-	-	0.08	3	−2
(7080)	8	268	6805	7	0.32	271	0.09	6809	-	-	-	-	-	0.1	3.2	−7
	32	276	6794	10	0.34	281	0.11	6799	-	-	-	-	-	0.13	2.6	−10
c5315	2	88	10528	14	6.21	100	0.59	10530	-	-	-	-	-	0.6	10.4	−14
(10630)	8	140	10452	38	19.14	176	0.59	10454	-	-	-	-	-	0.6	31.9	−38
	32	217	10376	37	31.92	252	0.59	10378	-	-	-	-	-	0.6	53.2	−37
c6288	2	71	12486	19	65.63	74	0.09	12502	-	-	-	-	-	0.17	386.1	−19
(12576)	8	85	12352	139	150.56	110	3.52	12363	0.08	0	0	103	81.65	85.25	1.8	−36
	32	99	12012	465	386.97	142	2.65	12020	0.08	0	0	414	305.28	308.01	1.3	−51
c7552	2	228	14842	34	25.55	249	2.85	14842	0.02	0	0	13	5.62	13.01	2	−21
(15104)	4	237	14789	78	40.53	285	3.14	14788	0.03	0	1	30	13.17	18.01	2.3	−48
	8	246	14728	130	56.08	329	3.17	14728	0.03	0	0	47	20.98	19.01	3	−83
	32	264	14574	266	100.35	449	3.58	14581	0.03	0	0	74	29.5	29.01	3.5	−192

total time (or the sum of three computational times) are also shown. Value '0' in column *time* is smaller than 0.01 sec. Symbol '-' in column *3rd* means the third step of the flow is not required because the acceptability of every fault is identified at the second step. The last two columns under column *comparison* are the (acceleration) ratio of the test generation time with only a DIFF model to that for the proposed test generation flow, and the difference of the number of aborted faults between the two test generations.

As we can see from Table 1, compared with the test generation with only DIFF models, the proposed test generation flow can reduce the test generation time and the number of aborted faults for all circuits. Especially, for c432 (except for the case when $T = 5$), c880, c3540, c5315 and c6288 under $T = 2$, we can achieve complete fault efficiency (or no aborted faults) with smaller computational time. This is because the proposed test generation flow can quickly identify the acceptable faults, which are not identified with only the DIFF model, with the maximal AID model set.

We also compared the proposed test generation flow with [2]. The authors of [2] reported a comparison of their threshold test generation method with a general test generation (or a threshold test generation when threshold $T = 1$) in terms of test generation time and fault efficiency. Table 2 shows the ratios of test generation time and fault efficiency. These results are originally depicted as graphs in [2] and several typical points are picked up. This table also shows the time and efficiency ratios by our method in the same manner. From this table, we can see that, compared with the test generation method of [2], our test generation method can obtain higher or comparable fault efficiency and smaller computational time, especially for larger circuits. Note that the method [2] employs a well-tailored test generation algorithm, while our method employs an off-the-shelf algorithm. Hence, our method is practical and effective.

Table 2 Comparison with [2].

		[2]		Proposed	
circ.	T	time ratio	eff. ratio	time ratio	eff. ratio
c432	4	2.5	0.98	3.89	1
	5	3.4	0.97	9.79	0.99
	6	3	0.98	17.84	1
	7	3.7	0.97	21.53	0.97
c880	18	3.4	1	3	1
	32	6	1	1	1
c5315	18	9.7	0.99	2.03	1
	32	11	0.99	1.03	1

5. Conclusion

In this paper, we proposed two test generation models for threshold test generation with general test generation algorithms, and introduced the efficient test generation flow. Experimental results show the test generation flow can achieve high fault efficiency with small computational effort.

Threshold test patterns generated by the proposed method may accidentally detect some acceptable faults, provided that the test is practically applied, i.e., the output responses are just checked against the expected ones, without respect to their error significance. To reduce the number of such acceptable faults, the authors of [4] have proposed a method for generating test patterns that detect as few acceptable faults as possible for threshold testing based on error rate. Such a method for threshold testing based on error significance still remains as a challenging problem. Another future work is to propose a threshold test generation method for sequential circuits as an extension of the threshold test generation method proposed in this paper.

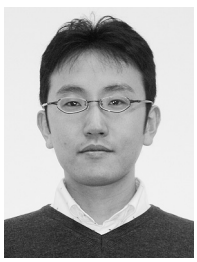
Acknowledgment

This research was partially supported by Japan Society for the Promotion of Science (Grant-in-Aid for Young Scientists (Start-up) (19800035) and Grant-in-Aid for Scientific Research (C) (19500048)), and by Hiroshima City University under the HCU Grant for Special Academic Research

(General Studies).

References

[1] M.A. Breuer and H. Zhu, "An illustrated methodology for analysis of error tolerance," IEEE Design and Test Magazine, pp.168–177, 2008.
[2] Z. Jiang and S.K. Gupta, "An ATPG for threshold testing: Obtaining acceptable yield in future processes," Proc. ITC, pp.824–833, 2002.
[3] H. Chung and A. Ortega, "Analysis and testing for error tolerant motion estimation," Proc. DFT, pp.514–522, 2005.
[4] T.-Y. Hsieh, K.-J. Lee, and M.A. Breuer, "Reduction of detected acceptable faults for yield improvement via error-tolerance," Proc. DATE, pp.1599–1604, 2007.
[5] D. Shin and S.K. Gupta, "A re-design for datapath modules in error tolerant applications," Proc. ATS, pp.431–437, 2008.
[6] I. Polian, B. Becker, M. Nakasato, S. Ohtake, and H. Fujiwara, "Low-cost hardening of image processing applications against soft errors," Proc. DFT, pp.274–279, 2006.
[7] I. Polian, D. Nowroth, and B. Becker, "Identification of critical errors in imaging applications," Proc. IOLTS, pp.201–202, 2007.
[8] S. Shahidi and S.K. Gupta, "A theory of error-rate testing," Proc. ICCD, pp.438–445, 2006.
[9] K.-J. Lee, T.-Y. Hsieh, and M.A. Breuer, "A novel test methodology based on error-rate to support error-tolerance," Proc. ITC, pp.1136–1144, 2005.
[10] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," IEEE Trans. Comput., vol.C-30, no.3, pp.215–222, 1981.
[11] M.A. Iyer and M. Abramovici, "Low-cost redundancy identification for combinational circuits," Proc. VLSID, pp.315–318, 1994.



Hideyuki Ichihara received his M.E. and Ph.D. degrees from Osaka University in 1997, 1999, respectively. He was a research scholar of University of Iowa, U. S. A. from February to July in 1999. Since December 1999, he had been an assistant professor of Hiroshima City University, and he is currently an associate professor of the university. He received IEICE Best Paper Award 2004 and Workshop on RTL and High Level Testing 2004 Best Paper Award. His research interests are VLSI testing and design for testability. He is a member of the IEEE.

Kenta Sutoh received his Bachelor and M. E. degrees of Information Engineering from Hiroshima City University in 2007 and 2009, respectively. He is currently with Renesas Technology Corp.

Yuki Yoshikawa received the B.E. degree in computer science from Kyoto Institute of Technology, Kyoto, Japan in 2003 and the M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Nara, Japan in 2005 and 2007, respectively. Presently he is an Assistant Professor at the Graduate School of Information Sciences, Hiroshima City University. He received Workshop on RTL and High Level Testing 2007 Best paper award. His research interests are delay test, design for testability and fault tolerance. He is a member of IEEE Computer Society and IPSJ.

Tomoo Inoue received B.E., M.E. and Ph.D. degrees from Meiji University, Kawasaki, Japan, in 1988, 1990 and 1997, respectively. He was with Matsushita Electric Industrial Co. Ltd. from 1990 to 1992. He was an Assistant Professor at the Graduate School of Information Sciences, Nara Institute of Science and Technology from 1993 to 1999. In 1999, he joined Faculty of Information Sciences, Hiroshima City University as an Associate Professor, and presently he is a Professor at Graduate School of Information Sciences, Hiroshima City University. His research interests include VLSI CAD, test generation algorithm and design and synthesis for testability and dependability. He received WRTLT (Workshop on RTL and High Level Testing) 2004 Best Paper Award. He is a member of the IEEE Computer Society and IPSJ.