Electronic Notes in Theoretical Computer Science 174 (2007) 83–93 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

On the Use of Functional Test Generation in Diagnostic Test Generation for Synchronous Sequential Circuits

Irith Pomeranz [1](#_bookmark0) *,*[2](#_bookmark0)

*School of ECE Purdue University*

*West Lafayette, IN 47907, U.S.A.*

# Sudhakar M. Reddy [3](#_bookmark0) *,*[4](#_bookmark0)

*ECE Department University of Iowa*

*Iowa City, IA 52242, U.S.A.*

Abstract

We study the relationship between diagnostic test generation for a gate-level fault model, which is used for generating diagnostic test sets for manufacturing defects, and functional test generation for a high-level fault model. In general, a functional fault may partially represent some of the effects of one gate-level fault but not another. Generating a test sequence for the functional fault is then likely to detect one gate- level fault but not the other, thus distinguishing the two faults. This relationship points to the ability to use a functional test generation procedure (that targets functional fault detection) as a way of generating diagnostic test sequences for gate-level faults. We use this observation in two ways. The more direct way is to define functional faults that correspond to the differences between pairs of gate-level faults. The second way is to use functional test sequences as diagnostic test sequences without explicitly considering gate-level faults. We support the use of the resulting procedures with experimental results.

*Keywords:* diagnostic test generation, functional test generation, state transition faults, synchronous sequential circuits.

# Introduction

Diagnostic test generation procedures [1]-[7] generate test sets or test sequences that distinguish pairs of faults out of a target fault model, typically stuck-at faults.

1 Research supported in part by SRC Grant No. 2005-TJ-1282

2 E-mail: [pomeranz@ecn.purdue.edu](mailto:pomeranz@ecn.purdue.edu)

3 Research supported in part by SRC Grant No. 2005-TJ-1243

4 E-mail: [reddy@engineering.uiowa.edu](mailto:reddy@engineering.uiowa.edu)

1571-0661 © 2007 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2006.12.031

Such test sets are useful for diagnosis of manufacturing defects even if the defects do not behave exactly as the target faults. Diagnostic test generation procedures perform test generation based on fault pairs (or larger subsets of faults) that are not yet distinguished by the test set. Their goal is to generate additional tests or test sequences (or extend an existing test sequence) so as to distinguish additional fault pairs, which are not yet distinguished. The need to consider fault pairs makes the diagnostic test generation process different and more complex than test gener- ation for fault detection. The test generation process can accommodate fault pairs either explicitly, or by modifying the circuit description to inject each one of the faults separately [6]. In the latter case, test generation for fault detection can be carried out on the modified circuits, and a test guarantees that the two faults will be distinguished. However, the circuit needs to be modified for every fault pair separately.

In this work we study the relationship between diagnostic test generation for gate-level faults and functional test generation for high-level faults. This relation- ship points to the ability to use a functional test generation procedure (that targets functional fault detection) as a way of generating diagnostic test sequences for gate- level faults in synchronous sequential circuits without modifying the circuit. Several functional test generation procedures and fault models exist [8]-[15]. The functional fault model we consider for this study is the state transition fault model [10]. A state transition fault is a fault affecting the next state or output vector of a state transition in the state table of the circuit. Other functional fault models can be used in a similar way. We consider single stuck-at faults as the targets of diagnostic test generation.

The advantage of using state transition faults (or high-level faults in general) for diagnostic test generation can be seen from the following example. Consider a state transition *si aj sk* from state *si* to state *sk* under primary input vector *aj*, producing primary output vector *zl*. Suppose that a stuck-at fault *fj*1 changes the next state of this state transition to *s*'*k*, while a stuck-at fault *fj*2 does not affect this state transition. Consider a test sequence *T* that detects the state transition fault where the next state is *s*'*k* instead of *sk*. Since *fj*1 causes the same change to the state table of the circuit as the state transition fault, *T* is likely to detect *fj*1. Since *fj*2 does not cause this change to the state table, it is not likely to be detected by *T* . If *T* detects *fj*1 but not *fj*2, then *T* distinguishes the two faults.

*zl*

In general, a functional fault may partially represent some of the effects of one stuck-at fault but not another. Generating a test sequence for the functional fault is then likely to detect one stuck-at fault but not the other, thus distinguishing the two faults. This observation can be used in one of two ways, both of them explored in this work.

1. The more direct way to use this observation is to define functional faults that correspond to the *dif f erences* between pairs of stuck-at faults (or subsets of stuck- at faults). By targeting a functional fault that represents the difference between two (or more) stuck-at faults, the test sequence is likely to detect one fault of a pair but not another, thus distinguishing the two faults. The difference between two

stuck-at faults is a subset of state transitions where the two stuck-at faults differ as explained next.

We note that each stuck-at fault *f* is equivalent to a single or multiple state transition fault (i.e., a fault that affects one or more state transitions in one or more ways). We denote the state transition fault equivalent to *f* by *ST* (*f* ). *ST* (*f* ) consists of one or more single state transition faults that are caused by the presence of *f* . The differences *ST* (*fj*1) − *ST* (*fj*2) and *ST* (*fj*2) − *ST* (*fj*1) between the state transition faults corresponding to two stuck-at faults *fj*1 and *fj*2 also constitute single or multiple state transition faults. The faults are considered during diagnostic test generation using a functional test generation procedure.

We first consider the (multiple) faults *ST* (*fj*1) − *ST* (*fj*2) and *ST* (*fj*2) − *ST* (*fj*1). Then, if |*ST* (*fj*1) − *ST* (*fj*2)| *>* 1, we also consider each single state transition fault included in *ST* (*fj*1) − *ST* (*fj*2) separately. Similarly, if |*ST* (*fj*2) − *ST* (*fj*1)| *>* 1, we consider each single state transition fault included in *ST* (*fj*2) − *ST* (*fj*1) separately.

1. By extension, a functional test sequence that detects all the single state transi- tion faults in a circuit, or all the single state transition faults that are components of stuck-at faults, is likely to be useful as a diagnostic test sequence. This view of the diagnostic test generation problem allows diagnostic test sequences to be gener- ated without considering a gate level implementation, for example, before such an implementation is available. It also supports the use of functional test sequences for defect diagnosis.

We describe the representation of stuck-at faults using single or multiple state transition faults in Section 2. In Section 3 we describe the derivation of state transition faults to be targeted during diagnostic test generation, and we describe a diagnostic test generation procedure based on functional test generation for state transition faults. Experimental results of diagnostic test generation are presented in Section 4. Results using a functional test sequence for single state transition faults are given in Section 5.

# Representation of single stuck-at faults using state transition faults

To demonstrate that stuck-at faults have equivalent state transition faults, we show in Table 1 the state table of finite-state machine benchmark *train*4. The state table is obtained from a gate-level implementation. The circuit has two state variables and four states with state vectors 00, 01, 10 and 11. It has two primary inputs with four input vectors *a* = 00, 01, 10 and 11, and one primary output with output vectors *z* = 0 and 1. *PS* stands for the present state and *NS* stands for the next state.

In Table 2 we show the effects of three stuck-at faults, *f*0, *f*2 and *f*12, in the gate- level implementation of *train*4 on the state table of the circuit. For every stuck-at fault *fj*, we simulate the circuit under every present state *s* ∈ {00, 01, 10, 11} and primary input vector *a* ∈ {00, 01, 10, 11}. If the faulty next state or output vector is different from the fault free next state or output vector, respectively, we include

Table 1

State table of *train*4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *NS, z* | | | |
| *PS* | *a* = 00 | *a* = 01 | *a* = 10 | *a* = 11 |
| 00 | 00,0 | 10,1 | 10,1 | 11,1 |
| 01 | 00,0 | 01,1 | 01,1 | 11,1 |
| 10 | 11,1 | 10,1 | 10,1 | 11,1 |
| 11 | 11,1 | 01,1 | 01,1 | 11,1 |

the faulty entry in the state table following a slash. For example, stuck-at fault *f*0

of *train*4 causes the next state of state 00 under input vector 10 to be 00 instead

of 10, and it causes the output to be 0 instead of 1. We represent this faulty state transition as 00  10 10*/*00. It consists of the single state transition faults 00 10 10*/*00

and 00  10 10.

1*/*0

1*/*0 1

Table 2

Stuck-at faults of *train*4

|  |  |  |  |
| --- | --- | --- | --- |
| *f*0  *PS* | *a* = 00 | *NS, z*  *a* = 01 *a* = 10 | *a* = 11 |
| 00 | 00,0 | 10,1 10/00,1/0 | 11/10,1 |
| 01 | 00,0 | 01,1 01/00,1/0 | 11/01,1 |
| 10 | 11,1 | 10,1 10/11,1 | 11/10,1 |
| 11 | 11,1 | 01,1 01/11,1 | 11/01,1 |
| *f*2 |  | *NS, z* |  |
| *PS* | *a* = 00 | *a* = 01 *a* = 10 | *a* = 11 |
| 00 | 00,0 | 10/00,1/0 10,1 | 11/10,1 |
| 01 | 00,0 | 01/00,1/0 01,1 | 11/01,1 |
| 10 | 11,1 | 10/11,1 10,1 | 11/10,1 |
| 11 | 11,1 | 01/11,1 01,1 | 11/01,1 |

Fault *f*0 in Table 2 results in the single state transition faults 00 10 10*/*00, 00 11 11*/*10,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *f*12 | *NS, z* | | | |
| *PS* | *a* = 00 | *a* = 01 | *a* = 10 | *a* = 11 |
| 00 | 00,0 | 10,1 | 10,1 | 11/10,1 |
| 01 | 00,0 | 01,1 | 01,1 | 11/01,1 |
| 10 | 11,1 | 10,1 | 10,1 | 11/10,1 |
| 11 | 11,1 | 01,1 | 01,1 | 11/01,1 |

1 1

01 10 01*/*00, 01 11 11*/*01, 10 10 10*/*11, 10 11 11*/*10, 11 10 01*/*11, 11 11 11*/*01, 00  10 10 and

1 1 1 1 1 1 1*/*0

01  10 01. Thus, *f*0 is equivalent to the multiple state transition fault that consists of the single state transition faults listed above. State transition faults that are equivalent to the other stuck-at faults can be obtained in a similar way.

1*/*0

The state transition fault equivalent to a stuck-at fault *f* can be found by com-

binational test generation for *f* . Combinational test generation considers only the combinational logic of the circuit. It assigns values to the primary inputs and present state variables, and observes fault effects on the primary outputs and next state variables. A combinational test *t* for a fault *f* defines a single state transition

fault *si aj sk/s*'*k* and/or a single state transition fault *si aj* ' *sk*, where *si* is the state

*zl zl/z l*

vector included in *t*, *aj* is the primary input vector included in *t*, *zl* is the fault free

primary output vector under *t*, *z*'*l* is the faulty primary output vector under *t*, *sk* is the fault free next state under *t*, and *s*'*k* is the faulty next state under *t*. We denote by *ST* (*f* ) the set of single state transition faults that define the (single or multiple) state transition fault which is equivalent to *f* . For our purposes it is sufficient to use a subset of all the single state transition faults for every fault *f* , since detection of a single state transition fault out of the difference between two faults is typically sufficient for distingiushing the faults. Thus, it is possible to use combinational *n*-detection test generation in order to compute the sets *ST* (*f* ) for target faults, instead of deriving the complete sets *ST* (*f* ) (*n*-detection test generation attempts to generate *n* tests for each target fault, resulting in sets *ST* (*f* ) of size *n* or more). This observation makes the proposed test generation procedure applicable to large circuits for which state tables are not available or the complete sets *ST* (*f* ) cannot be derived efficiently.

For the purpose of diagnostic test generation we will define a set of state tran- sition faults *STT ARG*. An element *STi* of *STT ARG* will be a single or multiple state transition fault. Diagnostic test generation will consist of generating tests for the faults defined by *STT ARG*. It is possible to use a functional test generation pro- cedure such as the one described in [10] or [11]. Alternatively, test generation for state transition faults can be done at the gate level as follows.

Initially, the test sequence *T* is empty and the circuit is in its reset state. For every fault *STi* ∈ *STT ARG*, a sequential test generation procedure is used for gener- ating a test subsequence *T*ˆ such that when *T*ˆ is added to *T* the resulting sequence *T T*ˆ detects *STi*. If *T*ˆ can be found, it is concatenated to *T* . The sequential test generation procedure is similar to one that targets stuck-at faults, with the follow- ing differences. (1) Fault activation conditions are extracted from the initial states and primary input vectors of the state transitions in *STi*. (2) The fault is activated

every time a state transition included in *STi* is traversed. The final state and output

values are obtained from *STi*.

# Diagnostic test generation

Given a set of stuck-at faults *F* and an equivalent single or multiple state transition fault *ST* (*f* ) for every *f* ∈ *F* , the sets *ST* (*f* ) for *f* ∈ *F* provide functional faults that are targets for diagnostic test generation as described in this section.

Consider two faults *fj*1*, fj*2 ∈ *F* with corresponding state transition faults *ST* (*fj*1) and *ST* (*fj*2), respectively. The set *ST* (*fj*1) ∩ *ST* (*fj*2) consists of sin- gle state transition faults that are common to *fj*1 and *fj*2. Consider the subset *ST* (*fj*1) − *ST* (*fj*2). This subset consists of single state transition faults that result

from *fj*1, but not from *fj*2. Suppose that *ST* (*fj*1) − *ST* (*fj*2) /= ∅. A test sequence that detects *ST* (*fj*1) − *ST* (*fj*2) at time unit *u* is likely to detect *fj*1 at time unit *u* (unless some of the remaining single state transition faults in *ST* (*fj*1) cancel the fault effects due to *ST* (*fj*1) − *ST* (*fj*2) and prevent *fj*1 from being detected at time unit *u*). In addition, a test sequence that detects *ST* (*fj*1) − *ST* (*fj*2) at time unit *u* is not likely to detect *fj*2 at time unit *u*, since none of the single state transition faults that result from *fj*2 is targeted. We conclude that a test sequence that detects *ST* (*fj*1) − *ST* (*fj*2) at time unit *u* is likely to distinguish *fj*1 and *fj*2 at time unit *u*. In a similar way it is possible to argue that if *ST* (*fj*2) − *ST* (*fj*1) /= ∅, a test sequence that detects *ST* (*fj*2) − *ST* (*fj*1) at time unit *u* is likely to distinguish *fj*1

and *fj*2 at time unit *u*.

For illustration, consider *f*0 and *f*2 of *train*4 shown in Table 2. *ST* (*f*0) −

*ST* (*f*2) = {00 10 10*/*00, 01 10 01*/*00, 10 10 10*/*11, 11 10 01*/*11, 00  10 10, 01  10 01}. The

1 1 1

1 1*/*0

1*/*0

input sequence 10 detects the fault defined by *ST* (*f*0) − *ST* (*f*2). The same input

sequence detects *f*0 but not *f*2. Thus, it distinguishes the two faults.

The faults *f*0 and *f*12 demonstrate a case where *ST* (*f*12) − *ST* (*f*0)= ∅. In this case, only *ST* (*f*0) − *ST* (*f*12) can be used during diagnostic test generation.

By using *ST* (*fj*1) − *ST* (*fj*2) and *ST* (*fj*2) − *ST* (*fj*1), we perform functional test generation for at most two state transition faults for every pair of stuck-at faults that need to be distinguished. Both *ST* (*fj*1) − *ST* (*fj*2) and *ST* (*fj*2) − *ST* (*fj*1) may be multiple faults. It is also possible to consider single state transition faults during functional test generation by considering each *st* ∈ *ST* (*fj*1) − *ST* (*fj*2) and each *st* ∈ *ST* (*fj*2) − *ST* (*fj*1) separately. The number of state transition faults that need to be considered in this case is higher; however, each functional fault consists of a single state transition fault.

In the proposed diagnostic test generation procedure, we first consider the two functional faults *ST* (*fj*1) − *ST* (*fj*2) and *ST* (*fj*2) − *ST* (*fj*1) for every pair of stuck- at faults *fj*1*, fj*2 that needs to be distinguished. We then consider single state transition faults included in *ST* (*fj*1) − *ST* (*fj*2) and *ST* (*fj*2) − *ST* (*fj*1), but only if

|*ST* (*fj*1)| ≤ *N* and |*ST* (*fj*2)| ≤ *N* for a constant *N* . This restriction is imposed in order to limit the number of functional faults considered. It is justified by the expectation that if the difference between *ST* (*fj*1) and *ST* (*fj*2) is large, the faults are likely to be distinguished by test sequences generated based on other fault pairs. The overall diagnostic test generation process is given as Procedure 1 next.

Procedure 1: Diagnostic test generation based on state transition faults

1. Let *F* be the set of single stuck-at faults. Let *ST* (*f* ) be the state transition fault equivalent to *f* for every *f* ∈ *F* . Let *P* be the set of fault pairs defined over *F* . Set *T* = ∅.
2. *Multiple state transition faults* Set *PT ARG* = *P* .
3. Select *fj*1*, fj*2 ∈ *PT ARG*. Remove *fj*1*, fj*2 from *PT ARG*.
4. If *ST* (*fj*1) − *ST* (*fj*2) /= ∅:
   1. Generate a test subsequence *T*ˆ for *ST* (*fj*1) − *ST* (*fj*2). If *T*ˆ is generated:
      1. Concatenate *T*ˆ to *T* .
      2. Perform diagnostic fault simulation of *T* and drop distinguished fault pairs

from *P* and from *PT ARG*.

* + 1. If no fault pair is distinguished, remove *T*ˆ from *T* .

1. If *fj*1*, fj*2 ∈ *P* and *ST* (*fj*2) − *ST* (*fj*1) /= ∅:
   1. Generate a test subsequence *T*ˆ for *ST* (*fj*2) − *ST* (*fj*1). If *T*ˆ is generated:
      1. Concatenate *T*ˆ to *T* .
      2. Perform diagnostic fault simulation of *T* and drop distinguished fault pairs from *P* and from *PT ARG*.
      3. If no fault pair is distinguished, remove *T*ˆ from *T* .
2. If *PT ARG* /= ∅, go to Step 3.
3. *Single state transition faults* Set *PT ARG* = *P* .
4. Select *fj*1*, fj*2 ∈ *PT ARG*. Remove *fj*1*, fj*2 from *PT ARG*.
5. If |*ST* (*fj*1)| ≤ *N* and |*ST* (*fj*2)| ≤ *N* , for every *st* ∈ *ST* (*fj*1) − *ST* (*fj*2):
   1. Generate a test subsequence *T*ˆ for *st*. If *T*ˆ is generated:
      1. Concatenate *T*ˆ to *T* .
      2. Perform diagnostic fault simulation of *T* and drop distinguished fault pairs from *P* and from *PT ARG*.
      3. If no fault pair is distinguished, remove *T*ˆ from *T* .
6. If |*ST* (*fj*1)| ≤ *N* , |*ST* (*fj*2)| ≤ *N* and *fj*1*, fj*2 ∈ *P* , for every *st* ∈ *ST* (*fj*2) −

*ST* (*fj*1):

* 1. Generate a test subsequence *T*ˆ for *st*. If *T*ˆ is generated:
     1. Concatenate *T*ˆ to *T* .
     2. Perform diagnostic fault simulation of *T* and drop distinguished fault pairs from *P* and from *PT ARG*.
     3. If no fault pair is distinguished, remove *T*ˆ from *T* .

1. If *PT ARG* /= ∅, go to Step 8.

# Experimental results

The results of Procedure 1 for finite-state machine benchmarks are shown in Tables 3 and 4. We use *N* = 100 in Procedure 1. We assume the existence of fault free hardware reset to the all-zero state.

In Table 3, after the circuit name we show the number of primary inputs, the number of primary outputs, and the number of state variables. Under column *flts* we show the number of stuck-at faults *f* with *ST* (*f* ) /= ∅. These faults constitute the set of target faults *F* . We then show the number of fault pairs defined over *F* . In Table 4, under column *mult st flts* we show the results of Procedure 1 after considering faults of the form *ST* (*fj*1) − *ST* (*fj*2) and faults of the form *ST* (*fj*2) − *ST* (*fj*1). Under column *single st flts* we show the results of Procedure 1 after considering single state transition faults out of *ST* (*fj*1) − *ST* (*fj*2) and *ST* (*fj*2) − *ST* (*fj*1). In each case, we show the number of calls to the functional test generation process that produces a test subsequence for a state transition fault, the length of the test sequence *T* generated by Procedure 1, the number of stuck-at faults detected by *T* , the number of stuck-at fault pairs distinguished by *T* , and the percentage of

stuck-at fault pairs distinguished by *T* .

Table 3 Circuit parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| circuit | inp | out | sv | flts | pairs |
| bbara | 4 | 2 | 4 | 138 | 9453 |
| bbsse | 7 | 7 | 4 | 238 | 28203 |
| bbtas | 2 | 2 | 3 | 63 | 1953 |
| beecount | 3 | 4 | 3 | 110 | 5995 |
| cse | 7 | 7 | 4 | 355 | 62835 |
| dk14 | 3 | 5 | 3 | 207 | 21321 |
| dk15 | 3 | 5 | 2 | 151 | 11325 |
| dk16 | 2 | 3 | 5 | 530 | 140185 |
| dk17 | 2 | 3 | 3 | 128 | 8128 |
| dk27 | 1 | 2 | 3 | 67 | 2211 |
| dk512 | 1 | 3 | 4 | 124 | 7626 |
| ex2 | 2 | 2 | 5 | 312 | 48516 |
| ex3 | 2 | 2 | 4 | 153 | 11628 |
| ex4 | 5 | 9 | 4 | 176 | 15400 |
| ex5 | 2 | 2 | 3 | 138 | 9453 |
| ex6 | 5 | 8 | 3 | 229 | 26106 |
| ex7 | 2 | 2 | 4 | 159 | 12561 |
| keyb | 7 | 2 | 5 | 470 | 110215 |
| lion | 2 | 1 | 2 | 40 | 780 |
| lion9 | 2 | 1 | 3 | 59 | 1711 |
| mark1 | 4 | 14 | 4 | 203 | 20503 |
| mc | 3 | 5 | 2 | 73 | 2628 |
| opus | 5 | 6 | 4 | 181 | 16290 |
| shiftreg | 1 | 1 | 3 | 28 | 378 |
| tav | 4 | 4 | 2 | 64 | 2016 |
| train11 | 2 | 1 | 4 | 104 | 5356 |
| train4 | 2 | 1 | 2 | 34 | 561 |

From Table 4 it can be seen that diagnostic test generation based on multiple state transition faults distinguishes most of the distinguishable fault pairs. Consid- eration of single state transition faults increases the number of distinguished fault pairs at the cost of increasing the number of calls to the functional test generation procedure.

# Test sequences for single state transition faults

The results of Section 4 indicate that functional test generation for single state transition faults of the form *si aj sk/s*'*k* and *si aj* ' *sk* will result in test sequences

*zl zl/z l*

that are useful for fault diagnosis. This is a result of the fact that any single

state transition fault can potentially be included in the difference between the state transition faults equivalent to two stuck-at faults. Targeting it may thus result in a sequence that distinguishes the stuck-at faults.

Table 4

Results of diagnostic test generation

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| circuit | calls | len | mult det | st flts  dist | %dist | calls | len | single det | st flts  dist | %dist |
| bbara | 107 | 86 | 130 | 9211 | 97.440 | 2345 | 114 | 130 | 9249 | 97.842 |
| bbsse | 50 | 156 | 235 | 28167 | 99.872 | 1089 | 180 | 235 | 28176 | 99.904 |
| bbtas | 12 | 52 | 62 | 1638 | 83.871 | 4471 | 57 | 62 | 1640 | 83.973 |
| beecount | 11 | 39 | 109 | 5969 | 99.566 | 258 | 52 | 110 | 5978 | 99.716 |
| cse | 51 | 217 | 354 | 62778 | 99.909 | 329 | 240 | 354 | 62783 | 99.917 |
| dk14 | 26 | 56 | 206 | 21314 | 99.967 | 205 | 56 | 206 | 21314 | 99.967 |
| dk15 | 15 | 37 | 150 | 11312 | 99.885 | 66 | 42 | 150 | 11316 | 99.921 |
| dk16 | 44 | 166 | 529 | 140170 | 99.989 | 242 | 175 | 529 | 140174 | 99.992 |
| dk17 | 13 | 45 | 127 | 8108 | 99.754 | 53 | 50 | 127 | 8110 | 99.779 |
| dk27 | 5 | 21 | 67 | 2198 | 99.412 | 25 | 21 | 67 | 2198 | 99.412 |
| dk512 | 16 | 57 | 122 | 7609 | 99.777 | 76 | 69 | 122 | 7612 | 99.816 |
| ex2 | 32 | 164 | 311 | 48487 | 99.940 | 624 | 217 | 311 | 48500 | 99.967 |
| ex3 | 15 | 57 | 152 | 11621 | 99.940 | 182 | 57 | 152 | 11621 | 99.940 |
| ex4 | 52 | 95 | 171 | 15368 | 99.792 | 948 | 101 | 171 | 15369 | 99.799 |
| ex5 | 15 | 55 | 138 | 9408 | 99.524 | 108 | 67 | 138 | 9419 | 99.640 |
| ex6 | 25 | 59 | 228 | 26095 | 99.958 | 158 | 64 | 228 | 26096 | 99.962 |
| ex7 | 131 | 70 | 149 | 12493 | 99.459 | 699 | 73 | 149 | 12494 | 99.467 |
| keyb | 85 | 366 | 468 | 110151 | 99.942 | 233 | 392 | 468 | 110165 | 99.955 |
| lion | 30 | 14 | 40 | 693 | 88.846 | 470 | 17 | 40 | 701 | 89.872 |
| lion9 | 42 | 28 | 53 | 1593 | 93.103 | 1105 | 33 | 53 | 1595 | 93.220 |
| mark1 | 46 | 61 | 197 | 20466 | 99.820 | 934 | 82 | 197 | 20472 | 99.849 |
| mc | 6 | 20 | 73 | 2616 | 99.543 | 125 | 25 | 73 | 2617 | 99.581 |
| opus | 38 | 65 | 180 | 16267 | 99.859 | 63 | 73 | 180 | 16271 | 99.883 |
| shiftreg | 3 | 14 | 28 | 323 | 85.450 | 443 | 14 | 28 | 323 | 85.450 |
| tav | 12 | 19 | 64 | 1967 | 97.569 | 1978 | 19 | 64 | 1967 | 97.569 |
| train11 | 35 | 36 | 100 | 5285 | 98.674 | 1309 | 58 | 100 | 5294 | 98.842 |
| train4 | 12 | 14 | 33 | 479 | 85.383 | 436 | 20 | 33 | 481 | 85.740 |
| average | 34 | 77 | 166 | 21548 | 97.268 | 703 | 88 | 166 | 21553 | 97.369 |

We define *ST* = ∪{*ST* (*f* ): *f* ∈ *F* }. *ST* is the set of all the single state transition faults that correspond to all the single stuck-at faults (when information about stuck-at faults is not available, it is possible to use all the single state transition faults). We perform functional test generation targeting every single state transition fault *st* ∈ *ST* . We perform fault simulation and diagnostic fault simulation for stuck-at faults only after the test generation process is complete. The results of this process are shown in Table 5 in a format similar to the one used in Table 4.

From Table 5 it can be seen that targeting single state transition faults typically increases the number of calls to the functional test generation process and the test length. However, it also increases the number of distinguished fault pairs. This points to the usefulness of functional test sequences as diagnostic test sequences for stuck-at faults. More generally, such sequences are expected to be effective for

Table 5

Results of test generation for all single state transition faults

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| circuit | calls | all len | single det | st flts  dist | %dist |
| bbara | 1179 | 1274 | 130 | 9243 | 97.778 |
| bbsse | 7912 | 12517 | 235 | 28182 | 99.926 |
| bbtas | 93 | 150 | 62 | 1748 | 89.503 |
| beecount | 156 | 293 | 110 | 5979 | 99.733 |
| cse | 3749 | 17212 | 355 | 62781 | 99.914 |
| dk14 | 206 | 270 | 207 | 21314 | 99.967 |
| dk15 | 50 | 146 | 151 | 11316 | 99.921 |
| dk16 | 362 | 314 | 529 | 140175 | 99.993 |
| dk17 | 39 | 159 | 128 | 8110 | 99.779 |
| dk27 | 20 | 23 | 67 | 2197 | 99.367 |
| dk512 | 61 | 125 | 122 | 7614 | 99.843 |
| ex2 | 764 | 623 | 312 | 48497 | 99.961 |
| ex3 | 230 | 180 | 153 | 11616 | 99.897 |
| ex4 | 2176 | 2861 | 171 | 15369 | 99.799 |
| ex5 | 39 | 162 | 138 | 9424 | 99.693 |
| ex6 | 1012 | 1114 | 229 | 26098 | 99.969 |
| ex7 | 372 | 166 | 149 | 12495 | 99.475 |
| keyb | 26648 | 20206 | 468 | 110178 | 99.966 |
| lion | 18 | 65 | 40 | 707 | 90.641 |
| lion9 | 52 | 137 | 53 | 1625 | 94.974 |
| mark1 | 1944 | 844 | 197 | 20470 | 99.839 |
| mc | 26 | 86 | 73 | 2616 | 99.543 |
| opus | 2850 | 2016 | 180 | 16249 | 99.748 |
| shiftreg | 8 | 48 | 28 | 320 | 84.656 |
| tav | 72 | 254 | 64 | 1964 | 97.421 |
| train11 | 178 | 441 | 100 | 5261 | 98.226 |
| train4 | 21 | 78 | 34 | 487 | 86.809 |
| average | 1861 | 2288 | 166 | 21557 | 97.642 |

defect diagnosis.

# Concluding remarks

We studied the possibility of using functional test generation for state transition faults as part of a diagnostic test generation process for stuck-at faults. A stuck-at fault *f* has an equivalent single or multiple state transition fault *ST* (*f* ). It is thus possible to represent the difference between two stuck-at faults *fj*1 and *fj*2 by state transition faults that correspond to the difference between *ST* (*fj*1) and *ST* (*fj*2). Generating a test sequence for such a functional fault is likely to detect one stuck-at fault but not the other, thus distinguishing the two faults. We used this observation directly to define functional faults that correspond to the differences between pairs of

stuck-at faults. Functional test generation for the differences resulted in a diagnostic test sequence. We also used a functional test sequence for single state transition faults as a diagnostic test sequence without considering stuck-at faults explicitly. We supported the use of these procedures as diagnostic test generation procedures with experimental results.

# References

1. P. Camurati, D. Medina, P. Prinetto and M. Sonza Reorda, ”A Diagnostic Test Pattern Generation Algorithm”, in Proc. Intl. Test Conf., 1990, pp. 52-58.
2. T. Gruning, U. Mahlstedt and H. Koopmeiners, ”DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits”, in Proc. Intl. Conf. on Computer-Aided Design, Nov. 1991, pp. 194-197.
3. F. Corno, P. Prinetto, M. Rebaudengo and M. Sonza Reorda, ”GARDA: A Diagnostic ATPG for Large Synchronous Sequential Circuits”, in Proc. Europ. Design & Test Conf., March 1995, pp. 267-271.
4. I. Hartanto, V. Boppana, J. H. Patel and W. K. Fuchs, ”Diagnostic Test Pattern Generation for Sequential Circuits”, in Proc. VLSI Test Symp., April 1997, pp. 196-202.
5. I. Pomeranz and S. M. Reddy, ”A Diagnostic Test Generation Procedure for Synchronous Sequential Circuits Based on Test Elimination”, in Proc. Intl. Test Conf., Oct. 1998, pp. 1074-1083.
6. V. Boppana and W. K. Fuchs, ”Dynamic Fault Collapsing and Diagnostic Test Pattern Generation for Sequential Circuits”, in Proc. Intl. Conf. on Computer-Aided Design, Nov. 1998, pp. 147-154.
7. Y. Xiaoming, J. Wu and E. M. Rudnick, ”Diagnostic Test Generation for Sequential Circuits”, in Proc. Intl. Test Conf., Oct. 2000, pp. 225-234.
8. K. Sabnani and A. T. Dahbura, ”A Protocol Test Generation Procedure”, Computer Networks, 1988,

pp. 285-297.

1. H.-K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli, ”Test Generation for Sequential Circuits”, IEEE Trans. on Computer-Aided Design, Oct. 1988, pp. 1081-1093.
2. K.-T. Cheng and J. Y. Jou, ”Functional Test Generation for Finite State Machines”, in Proc. Intl. Test Conf., 1990, pp. 162-168.
3. I. Pomeranz and S. M. Reddy, ”On Achieving Complete Fault Coverage for Sequential Machines”, IEEE Trans. on Computer-Aided Design, March 1994, pp. 378-386.
4. A. Fin and F. Fummi, ”A VHDL Error Simulator for Functional Test Generation”, in Proc. Design, Automation and Test in Europe Conf., March 2000, pp. 390-395.
5. M. B. Santos, F. M. Goncalves, I. C. Teixeira and J. P. Teixeira, ”RTL-Based Functional Test Generation for High Defects Coverage in Digital SOCs”, in Proc. European Test Workshop, May 2000, pp. 99-104.
6. F. Ferrandi, G. Ferrara, D. Sciuto, A. Fin and F. Fummi, ”Functional Test Generation for Behaviorally Sequential Models”, in Proc. Design, Automation and Test in Europe Conf., March 2001, pp. 403-410.
7. V. M. Vedula and J. A. Abraham, ”FACTOR: A Hierarchical Methodology for Functional Test Generation and Testability Analysis”, in Proc. Design, Automation and Test in Europe Conf., March 2002, pp. 730-734.