PARTITIONING FOR PSEUDO-EXHAUSTJVE TESTING IS A/P-COMPLETE

indexing terms: Logic und logic design, Integrated circuits, LSI, Computers

In the letter two schemes of circuit partitioning for pseudo-exhaustive testing are described. The complexities of both are proved to be NP-complete.

Introduction: Test generation is known to be an NP-complete problem. To cope with testing for VLSI circuits, a variety of approaches have been presented. Among these, the pseudo-exhaustive testing methodology is seen to be quite promising. The method, suggested by McCluskey at Standford University,',' is expected to eliminate the difficult test-generation process while maintaining a comparable quality of test coverage.

Traditionally, test generation is a deterministic process in which a test set is generated and subsequently applied to the primary inputs (PIs) of a combinational circuit to reveal any discrepancy between the actual primary output (PO) response and the expected one. This is to be contrasted with random testing, in which no test generation is required, but a set of randomly generated input vectors is applied to the circuit under test whose output response can be analysed by signature or symptom or similar technique. Pseudo-exhaustive testing is a method aimed to overcome the shortcomings of each of these two extremes.

In principle, for pseudo-exhaustive testing, a circuit can be divided into blocks such that each block is bounded by a manageable number of inputs. By identifying exhaustive test vectors for each of the blocks, the test coverage is guaranteed perfect. In his recent textbook' and a previous paper,[2]

McCluskey has suggested two approaches to partitioning a circuit, one using hardware multiplexers and the other adopting a software technique to identify the partitions.

In this letter we first present two schemes of partitioning. One, termed 'gate partitioning*, divides a circuit into macro-blocks. The other, coined 'PI partitioning', groups the primary inputs of a circuit into clusters. To predict the time complexity of both schemes, a series of problems are raised and their consequences evaluated. According to the analysis presented, all turn out to be in the class of N&complete.

*Partitioning **schemes:***

(a) Gate partitioning : Gate partitioning divides a circuit into subcircuits, called here blocks, each of which may entail several gates. To make exhaustive testing practical, the number of inputs to each block has to be limited. Furthermore, the partition should enable a polynomial algorithm to find, for each block, a set of PI vectors which exhaustively produce, under the topological and functional constraints of the circuit, all feasible yet distinctive input vectors of the block. Also, for each of the vectors there should be at least one output of the block with a sensitised path to a PO, a primary output of C, such that its status will change as the status of the particular output of the block changes while the PI vector is held constant.

(b) PI partitioning PI partitioning divides the PI set of a circuit into subsets, each of which, here called a cluster, has a limited number of PI lines. The partition should enable a polynomial algorithm to find, for the complementary PI set of each cluster, a set of vectors, called bias vectors. Testing is executed for each cluster by cycling all vectors of the cluster under testing for each of the corresponding bias vectors. It is obvious that the number of vectors in a bias set should also be bounded to make the scheme practical.

Complexity analysis: Five problems are listed here to provide a broad basis on which to discuss the complexity issue of test generation. Of these, two, IC and FD, were proved in Reference 3 to be NP-complete. The former (IC) is used for the proofs of the main results of the paper, while the latter (FD) is included because it is presented in a form different to that in Reference 3, as is the proof. The form developed here reveals the optimisation nature of the problem.

The first step in proving a problem to be NP-complete is to show that it is in *NP*. This is easy to see for all the problems presented since, given an answer for a problem, its verification can be accomplished in polynomial steps in terms of the number of lines in the circuit. Thus in each proof to follow, it suffices to show that a polynomial algorithm for the problem concerned can be utilised to develop another polynomial algorithm for a known NP-complete problem.

(1) *Problem I (PI, irredundancy check or IC):* Is a combinational circuit irredundant (i.e. can all single stuck faults be detected)?

Lemma I: IC is NP-complete.

Proof: This is proved in theorem 3.1 in Reference 3.

(2) *Problem 2* (P2, *fault detection or FD):* Given an irredundant combinational circuit C and given an integer B, find a test set T such that $|T| < B$ and T detects all single stuck faults in C.

Theorem I : FD is NP-complete.

Proof: We show that IC is transformable to FD, i.e. IC a FD.

Assuming that A2 is a polynomial algorithm for FD, we show how to use it to obtain a polynomial algorithm Al for problem IC. Let N denote the number of lines in an arbitrary circuit C, and p(.) the polynomial bound of A2. We let B = 2N, and apply A2 to C. If A2 does not halt within p(.)

steps, then the circuit C must be redundant. If A2 halts within p(.) steps, with a test set T, then there are two cases:

(a) T is empty, or T is not empty but $|T| > 2N$, which means C cannot be irredundant.

(b) 7 is not empty and $|T| \leq 2N$. Then use fault simulation to see if all single stuck faults are detected by tests in T. If it is found that not all single stuck faults are detected by the set, then C must be redundant, as the maximum volume of a test set is 2N. If the result of fault simulation tells us that all faults are detected by T, then C is irredundant.

As fault simulation is possible with polynomial algorithms,[4] we prove that FD is indeed NP-complete.

(3) Problem 3 (P3, gate *exhaustive testing or GET):* Given C and B as above, find an input vector set V for a gate G in C, such that by applying vectors in V to the primary inputs of C, the following results hold:

(a) Each feasible yet distinct input vector of G appears once as part of the circuit status of C under stimulus of a vector in *V.*

(b) For each of the vectors in *V* applied to PI, there is a sensitised path from the output of G to the PO, such that the status of the PO will change as the output of G changes with no changes in the PI vector.

Theorem 2: P3 is NP-complete.

Proof: We show that IC is transformable to P3, i.e. IC a P3.

Given an arbitrary combinational circuit C, assume A3 is a polynomial algorithm for P3. We construct, based on P3, a polynomial algorithm Al for IC.

Applying A3 to one gate G, which has N inputs, in C, there are two possibilities regarding the computing time. If A3 does not halt within p(.) steps, then C must be redundant. If A3 halts within p(.) steps with a set *V,* two cases are possible:

(a) *V* is empty or $|V| > 2^N$, which means C must be redundant.

(b) *V* is not empty and $|V| \leq 2^N$. Then use fault simulation to verify whether all single stuck faults of the I/O lines of G are detected by vectors in *V.* If it is found that at least one fault is not detected by *V,* it follows then that the corresponding line is redundant, which in turn means that C is redundant. If the result of fault simulation shows that all faults of G are detected by *V,* then Al invokes A3 to check another gate in C.

As the above procedure is proportional to the number of gates in C, and fault simulation is a polynomial procedure, hence Al is polynomial as long as A3 is. Thus P3 is proved to be NP-complete.

(4) *Problem 4 (P4, gate partitioning or GP):* Given C and B as above and another integer *E,* find a partition, assuming its existence, of C into blocks such that the maximum input number of these blocks (Ml) satisfies $M1 < E$, and for each of the blocks there is a PI vector set *V* which satisfies the two conditions in P3 if we define the block as a generalised gate.

Theorem 3: P4 is NP-complete.

Proof: By defining a block as a generalised gate, the proving procedure can be applied here for P4, i.e. IC is transformable to P4. Therefore P4 is NP-complete

(5) *Problem 5 (P5, PI partitioning or PIP):* Given C, B and *E* as above, find a partition, assuming its existence, of the primary input set PI into clusters, such that the following conditions hold :

(a) The maximum input number in a cluster MI satisfies $Ml < E$.

(b) For the complementary set of each cluster *CL* relative to *Pl, PI – CL,* there exists a binary vector set P, and the maximum number of vectors in *P* for a cluster, *MP,* satisfies $MP < B$.

(c) All single stuck faults in C can be detected by applying input vectors formed in the following way: each cluster *CL cycles* **all** its input combinations $|P|$ times, and for each of the cycles the complementary cluster of *CL* is supplied with a binary vector in the corresponding P.

Theorem 4: P5 is NP-complete

Proof: By a procedure similar to that in the proof of theorem 2, we can show that IC is transformable to P5; thus P5 is NP-complete

Conclusions: It is important to recognise that the purpose of this letter is not to discourage the use of pseudo-exhaustive testing, but rather to provide a complete theoretical perspective indicating that the underlying problem of test generation remains, as long as the complete test coverage of the single stuck fault model is sought after. Albeit the general problem of partitioning a circuit for pseudo-exhaustive testing **is** shown in this letter to be as hard as test generation, two reasonable directions for progress are suggested: the first possibility is that in design for testability, the partition should be considered in advance. The second possibility is to seek efficient heuristic partition algorithms with probabilistically complete test coverage.

W. SONG
K. C. SMITH
W. M. SNELGROVE
Department of Electrical Engineering
University of Toronto
Toronto, Ontario, Canada M5S IA4

30th June I987

References

1 MCCLUSKY, E. J.: 'Logic design principles* (Prentice-Hall, Englewood Cliffs, NJ, 1986)
2 MCCLUSKY, E. J.: 'Built-in self-test techniques', *IEEE Des. & Test Comput.,* April 1985, 2, pp. 21-28
3 IBARRA, 0. H., and SAHNI, S. K.: 'Polynomially complete fault detection problems', *IEEE Trans., 1975, C-24,* pp. *242-249*
4 GOEL, P.: 'Test generation costs analysis and projections'. Proc. 17th design automation conf., Minneapolis, MN, 23-25 June 1980, pp. 77-84
5 GAREY, M. R., and JOHNSON, D. S.: 'Computers and intractability: a guide to the theory of NP-completeness' (W. H. Freeman, New York, 1979)