1991

A Metric Towards Efficient Exhaustive Test Pattern Generation

Dimitrios Kagaris
Dartmouth College

Fillia Makedon
Dartmouth College

Spyros Tragoudas
Southern Illinois University Carbondale

A METRIC TOWARDS EFFICIENT EXHAUSTIVE TEST PATTERN GENERATION

Fillia Makedon
Dimitrios Kagaris
Spyros Tragoudas

Technical Report PCS-TR91-162

Submitted to ISCAS '92

9/17/91
ISCAS'91

A METRIC TOWARDS EFFICIENT EXHAUSTIVE TEST PATTERN GENERATION

Dimitrios Kagaris, Fillia Makedon and Spyros Tragoudas
Dept.of Math. &Computer Science
Bradley Hall
Dartmouth College
Hanover, NH 03755

Computer Science Department
Faner Hall
Southern Illinois University
Carbondale, IL 62901

Abstract

A viable technique [7] in built-in self-test (BIST) [2] is to generate test patterns pseudo-exhaustively by using linear feedback shift registers (LFSR's). The goal is to find an appropriate primitive polynomial of degree d that will generate 2^d test patterns in order to exercise all circuit outputs simultaneously. In an attempt to reduce the degree d of the polynomial the following strategy was proposed in [6, 5]. In a first phase, partition the circuit into segments by inserting a small number of register cells, so that the input dependency of any circuit element in the segments is no more than d. Then, obtain an appropriate primitive polynomial of degree d by inserting additional register cells. In [12] we have proposed a heuristic for phase one that does not necessarily partition the circuit. Extensive experimentation has shown that this results in a considerably smaller cell overhead.

In this paper we extend our heuristic in [12], so that the minimization of the number of register cells is done in conjunction with a quantity that naturally reflects the difficulty of deriving an appropriate primitive polynomial of degree d. Experimentation shows that the proposed heuristic results again in an overall smaller number of register cells than a partition based approach and in an efficient framework for test pattern generation.

Key words: Computer Aided Design, Design for Testability, VLSI Design and applications.

1 Introduction

We are given a combinational circuit with p inputs and m outputs. The elements in the circuit may be simply boolean gates or more generally integrated modules. We want to test exhaustively

every output of the circuit by introducing some additional circuitry on the same chip/board. This approach, known as exhaustive built–in self–testing [2], is being given much attention as a promising technique for tackling the difficult problem of testing integrated circuits.

In accordance with the Level Sensitive Scan Design (LSSD) principle [3], we assume as an initial part of our testing circuitry, a number of register cells which provide the circuit inputs with desired test values. For each circuit output $i, 1 \leq i \leq m$, let A_i be the set of input cells on which i depends. These sets will be referred to as *Asets* (affector sets). If the size of the maximum *Aset* is a, then the exhaustive testing of the circuit requires at least 2^a time. Since a may be quite large, a general approach [1, 4, 6] is to partition the circuit into subcircuits so that each subcircuit output depends on no more than k subcircuit inputs for a given limit k. (In practice k is at most 20.) The partition and the reduction of the *Aset* sizes of the elements is done by inserting a number of additional register cells (referred to as *testers*) on some of the element interconnections. The testers are linked together to form linear feedback shift registers (LFSR's), whose feedback connections are determined by appropriate primitive polynomials [8, 9].

The goal is to minimize the overall number of inserted testers. All the existing heuristic approaches [6, 5, 12] are following a two-phase scheme, which initially minimizes the number of testers so that all *Aset* sizes are no more than k. Then, an appropriate primitive polynomial is obtained by inserting additional testers. Concerning the first phase, the difference between our approach in [12] and Jone and Papachristou's approach [5, 6] is that we are not necessarily partitioning the circuit under test, while the latter does. There is experimental evidence [12] that our technique, which is generally applicable to both the gate and the module level (the technique in [5, 6] applies to gate level only), results in a considerably smaller number of testers.

In this paper, we extend our heuristic in [12] so that the minimization of the number of testers is done in conjunction with an introduced metric. This metric reflects the difficulty of finding an appropriate primitive polynomial, which is the task during the second phase. Of great help in the derivation of the heuristic was the generic for alternative heuristic implementations presented in [12].

2 Preliminaries

In this paper, we denote as a *tbt* element a circuit element that needs to be tested. Before any tester insertions, the set of the *tbt* elements consists of the set of elements feeding at least one circuit output net. After the insertions, the set of *tbt* elements consists of the latter elements plus any element for which at least one tester has been placed on some outgoing net. This is so because that element output value is overlayed by the tester value and thus its effect cannot be observed from any original circuit ouput.

Let the number of *tbt* elements be b. Each of the *tbt* elements requires at most 2^k test patterns to be exhaustively tested, so the total number of test patterns would seem to be at most $b2^k$. However, using a technique like the one proposed in [7], it may be possible to exhaustively test all *tbt*'s concurrently. Let $\mid \bigcup_{i=1}^{b} A_i \mid = n = p + T$, where p and T are the numbers of original inputs and testers, respectively. According to the results in [7], there exists a primitive polynomial $P(x)$ over GF(2), of degree d, $k \leq d \leq n$, so that an LFSR consisting of the n input and tester cells and having feedback connections at positions defined by the polynomial $P(x)$, produces all possible $2^{|A_i|}$ bit patterns for each *tbt* element i with *Aset* A_i. The selected polynomial must satisfy the following condition : For each i, all monomials $x^{e_j}, e_j \in A_i, 1 \leq j \leq \mid A_i \mid$, must be linearly independent mod $P(x)$. In this case, we say that $P(x)$ "satisfies" each *Aset* A_i.

The problem is how larger than k the degree d of $P(x)$ will be ($k \leq d \leq n$). The value of d is critical because the actual testing time will be 2^d, excluding the LFSR initialization time. Concerning the difficulty of finding an appropriate primitive polynomial, we observe the following :

Assume we have a number of *tbt* elements, for simplicity only two elements, with *Asets* A and B. Let us also assume $\mid A \mid \leq \mid B \mid$. We can distinguish between three cases: i) $A \subseteq B$ ii) $A \cap B \neq \emptyset \wedge A \not\subseteq B$ and iii) $A \cap B = \emptyset$.

In the first case, a primitive polynomial that satisfies B *will* also satisfy A. In the second case, a primitive polynomial that satisfies B *must* also satisfy A, whereas in the third case, we have the option of finding one primitive polynomial that satisfies A and one *different* primitive polynomial

that satisfies B. We will then have to form two LFSRs, but two applicable primitive polynomials are more likely to be found than one for a single LFSR and, since the LFSRs can work in parallel, the time bound is more likely to be achieved.

This desirable non-overlapping property is supported by the idea of partitioning the circuit into segments. We define a segment s to be the maximum subset of *tbt* elements of the circuit such that $Aset_i \cap Aset_j \neq \emptyset, \forall i,j \in s$ and $\{\bigcup_{i \in s} Aset_i\} \cap Aset_k = \emptyset, \forall k \notin s$. The advantage of segmentation is that each segment can be treated independently as a smaller circuit that has to be exhaustively tested. The disadvantage, however, is that, most of the time, segmentation is very expensive. We observe that in an extreme case, the segments may consist each of a single element, all of whose inputs have been blocked by testers.

The metric we introduce in this work, attempts to estimate this difficulty of finding an applicable primitive polynomial for a given collection of *Asets* (segmented or unsegmented). The nice property of this metric is that it also tends to minimize the number of inserted testers, although it is always to be used in conjunction with the actual value of the latter.

The rest of the paper is organized as follows: In Section 3 we present our metric. In Section 4 we describe a heuristic that is expected to behave well according to this metric and give comparative experimental results with respect to a number of other alternatives. Finally, we conclude in Section 5 with an outline of our future work.

3 The metric

Consider a set A with size $\mid A \mid = n$ and a collection of m subsets of A whose union is A. In our case A is the set of circuit inputs and introduced testers (labelled with the numbers 1 to n) and the m subsets of A are the *Asets* of the m *tbt* elements . We assume the elements in each subset are in ascending order and we will refer to jth element of A_i as A_{ij}. Let $r_i = \mid A_i \mid$. We would like to find a primitive polynomial of degree $k = \max\ r_i$, that satisfies each *Aset*. The number of primitive polynomials of any degree d is $\phi(2^d - 1)/d$, where ϕ is the Euler's ϕ-function. (Some numbers (taken from [8]) are given in Table 1.) We could generate them one by one or randomly

and test for applicability. What we would like is a means of evaluating how good the prospects are that the search for the collection of *Asets* under consideration will be succesful.

Table 1

d	$\phi(2^d-1)/d$
15	1800
16	2048
17	7710
18	8064
19	27594
20	24000
21	84672

An upper bound on the total number of inapplicable primitive polynomials with degree d is given by the following theorem :

Theorem 3.1 *The number of inapplicable primitive polynomials of degree d for a collection of m subsets of a set A is no more than*

$$B = \sum_{i=1}^{m} \sum_{j=1}^{r_i} (2^{j-1}-1)\lfloor \frac{A_{ij}}{d} \rfloor$$

Proof:

A primitive polynomial is inapplicable for the given collection if it divides at least one linear combination of monomials with degrees from the same subset in the collection. For a set A_i, let $L(x) = \sum_{j=1}^{r_i} \lambda_j x^{A_{ij}}$ be such a linear combination, where the λ_j's are 1 or 0 but not all 0.

There can be r_i different degrees for the $L(x)$. Assuming the elements are in ascending order, there can be $2^{j-1}-1$ different $L(x)$'s of degree A_{ij}. The factorization of a polynomial of some degree e into irreducible polynomials can have no more than $\lfloor e/d \rfloor$ factors of degree d. In the worst case, all the factors of degree d can be primitive polynomials. So an upper bound on the number of inapplicable primitive polynomials of degree d for the subset A_i is $\sum_{j=1}^{r_i} (2^{j-1}-1)\lfloor \frac{A_{ij}}{d} \rfloor$ and an upper bound for the whole collection is $\sum_{i=1}^{m} \sum_{j=1}^{r_i} (2^{j-1}-1)\lfloor \frac{A_{ij}}{d} \rfloor$. □

This bound B is an estimate of the difficulty of finding the desirable primitive polynomial : the smaller it is for a given collection of *Asets*, the more the chances are for finding an applicable

polynomial. We observe that the metric generally favors small numbers of testers as well : each new tester will contribute some greater factor A_{ij} in the inner sum and may also increase the number m in the outer sum (in case it is placed on a an edge $(u\ v)$ and u is not a *tbt* node already). We also observe that if the computation of B has been done on a per–segment basis, the metric preserves the preference to segmented collections of *Asets* over unsegmented ones : the Aij values in each segment can be renumbered starting from 1, since they correspond to the positions of the particular LFSR for that segment. This makes B have lower values than those it would have in an unsegmented collection (and indeed in any segment with more affectors). Once we have computed the B values for all segments formed by each method under consideration, we can either choose the method resulting in a segment with the smaller maximum value for B (in case we have the possibilty of conducting the search in parallel), or (if no parallelism is available) the one with the smaller sum of B values. (In this work we assumed we have available a number of machines for searching for a polynomial for each segment.) This is the way we use the metric in giving preference to one choice over others.

However, this guidance (if any) goes as far as the derivation of primitive polynomials is concerned. In deciding between two collections of *Asets*, of primary importance is the number T, which is the hardware overhead to obtain these collections. We have experimental evidence (from our work in [12] and here) that an approach based on segmentation of the circuit under test, results in small values for B in each segment, but large T. So the metric B is always to be consulted in conjunction with the total hardware overhead.

4 The heuristic

We work on the undirected acyclic graph corresponding to the combinational circuit under test. A node whose *Aset* exceeds the given limit k is said to be *divergent.* An outlined description of our basic heuristic scheme in [12] is given below : We proceed through the graph in a breadth-first search manner finding succesively sets ("fronts") of divergent nodes. For each front of divergent nodes met, we assign a cost to a local correction and a global correction routine. The local correction routine

places testers on some of the incoming edges of the divergent nodes so that their *Asets* change to new ones with sizes no greater than k. The global correction routine places testers on a set of edges before the front of the divergent nodes. This, in almost all versions of our scheme, is taken to be a cut set, cutting off the set of divergent nodes from some part of the graph above. This does not necessarily correct all the divergent nodes but gives a way to guard against local minima. The cost assignement is based strictly on the amount of testers needed to correct the whole graph (using local corrections only) if one of the tester placements is preferred.

Once we have obtained a feasible configuration of testers, we perturb it by a specified method in order to derive a new partial configuration, and run the heuristic all over again. This is repeated a number of times, each time taking into account the new imposed list of testers. The best among these configurations is the final solution.

The heuristic we propose in this work was derived in a number of stages, starting from this basic scheme. In the first stage, we recompared in terms of the metric two of the versions in [12] that were found closely competing in terms of hardware overhead only. Then, we used a variation on the "winner" in order to explicitly minimize the metric value. Finally in a third stage, we incorporated an additional mechanism to the heuristic of stage two in order to improve its performance further. The derivation stages are described in the following paragraphs. At each stage the versions were compared on the number of inserted testers (T), the metric value (B) for the "biggest" segment, and some other interesting quantities like the number of *tbt* nodes (TBT) (number m in the formula for B) and the average *Aset* size (SZ) of the *tbt* nodes. Small values of the last quantity are considered to be a sign of good behavior in the following sense : a smaller *Aset* is more likely to be satisfied by a candidate primitive polynomial, since fewer monomials have to be linearly independent.

The two competitive versions in [12] (named **E** and **R**) that we considered in the first stage, differ in their global correction routine : **E** uses the max-flow min-cut algorithm with unit edge capacities. **R** uses the max-flow min-cut algorithm with edge capacities of $c_{uv} = Aset_u$ for each edge going from a node u to a node v. The local correction routine for both versions is a greedy strategy that inspects all the incoming edges of a divergent node in descending order of *Aset* size

and places a tester on each edge until the node becomes non-divergent. The reason we considered version **R** is that it may possibly reduce the average *Aset* size of the *tbt* nodes finally derived : the more times the global correction routine has been preferred, the more likely a tester is to have been placed on an edge (u v)with "small" capacity, thus making node u with the small sized *Aset* be a *tbt* node instead of some other node with larger *Aset*. However, although this may be satisfied for the nodes in the cut, the nodes below may be affected in unpredictable ways.

We compared the two versions on a number of real and random circuit graphs. The real circuit graphs (rows 1-9 in the tables) were obtained from [10] and [11]. They were modified slightly in order to obey the LSSD principle and were examined on a number of different values of k. Some of the results are given in Table 2. (The values of B in the tables have been logarithmized.)

Table 2

	T		B		TBT		SZ	
No	**E**	**R**	**E**	**R**	**E**	**R**	**E**	**R**
1	109	125	14.42	14.89	49	40	10.24	12.47
2	113	101	14.97	14.77	54	52	10.22	10.19
3	92	91	15.37	15.38	55	55	10.58	10.65
4	90	89	16.44	16.71	59	40	12.18	12.75
5	68	66	16.69	16.71	41	40	12.26	12.75
6	60	62	17.12	16.62	44	40	12.70	12.75
7	59	58	18.24	18.24	66	66	15.95	15.96
8	12	12	9.05	9.05	119	119	4.02	4.02
9	8	8	8.92	8.92	116	116	4.05	4.05
10	142	142	12.33	12.33	93	93	8.33	8.33
11	81	75	17.64	17.63	69	71	15.20	15.18
12	152	152	12.83	12.83	96	96	7.79	7.79
13	131	131	18.36	18.36	100	100	16.66	16.66
14	294	292	14.00	14.12	181	179	7.80	7.94
15	271	271	19.49	19.49	174	174	16.08	16.08
16	278	260	13.69	13.59	164	160	7.55	7.50
17	456	456	14.58	14.58	248	248	7.69	7.69
18	363	374	19.96	19.88	225	227	15.61	15.91
19	334	330	17.53	17.61	215	209	12.35	12.18
20	63	63	16.42	16.42	33	33	11.93	11.93

Heuristic **R** results in greater B values than **E**, although the competition in the values of T is close. The average *Aset* size also turned out to be worse than that of **E**. So we prefer **E**.

In the second stage of our derivation process, we replaced the cost assignment routine of heuristic

E with a new one in an attempt to minimize the metric quantity instead of the hardware cost. In the new heuristic (called **G**), for each set of divergent nodes met, we compute the number of segments that would be formed under the local and global placements (using again only local corrections for the rest of the graph) and choose the placement resulting in a segment with the smaller maximum value for B. As seen in Table 3, the **G** results in smaller values for B. The number of testers T is sometimes much higher than the one for **E** but is still much less than the one for a a partition based technique (see Table 5).

Table 3

	T		B		TBT		SZ	
No	**E**	**G**	**E**	**G**	**E**	**G**	**E**	**G**
1	109	124	14.42	14.36	49	55	10.24	9.72
2	113	113	14.97	14.96	54	55	10.22	10.21
3	92	91	15.37	15.08	55	31	10.58	14.09
4	90	88	16.44	15.22	59	32	12.18	14.40
5	68	86	16.69	17.36	41	86	12.26	13.10
6	60	78	17.12	16.41	44	45	12.70	12.33
7	59	58	18.24	17.67	66	59	15.95	13.01
8	12	12	9.05	9.03	119	118	4.02	4.01
9	8	8	8.92	8.87	116	115	4.05	4.03
10	142	152	12.33	11.97	93	93	8.33	8.03
11	81	88	17.64	17.25	69	75	15.20	14.52
12	152	160	12.83	12.52	96	95	7.79	7.32
13	131	147	18.36	18.25	100	103	16.66	15.89
14	294	334	14.00	14.06	181	196	7.80	7.76
15	271	306	19.49	19.12	174	193	16.08	15.33
16	278	299	13.69	13.53	164	161	7.55	7.58
17	456	490	14.58	14.45	248	250	7.69	7.61
18	363	419	19.96	19.80	225	255	15.61	14.83
19	334	376	17.53	17.20	215	229	12.35	11.31
20	63	62	16.42	16.44	33	32	11.93	16.21

In a next (and last) stage, we tried to minimize the metric from another direction. From the formula for B, we see that it is reasonable to expect that by decreasing the number m of *tbt* nodes whose *Asets* have to be satisfied by a primitive polynomial, the number of applicable primitive polyniomials will increase. We observe that each tester placed on an edge $(u\ v)$ in order to reduce the size of the *Aset* of v (or the sizes of the *Asets* of nodes below v), makes u a *tbt* node. If a tester that is initially to be placed on some other edge on the part of the graph below u can still be useful

if placed on an outgoing edge of u with no tester on, the number of *tbt* nodes is surely not increased, i.e. the final B value will be thus smaller. The way we incorporated this idea in the heuristic is as follows : At each step, after a placement has been decided upon, we inspect every edge $(u\ v)$ on which a new tester has been placed. Let w be any other child of u for which there is no tester on $(u\ w)$. If the size of the *Aset* of w is larger than a large fraction f of the allowable limit k and moreover, u is responsible for a large part ($\geq r\%$) of the *Aset* size of w, then a placement of a tester on $(u\ v)$ is expected to be proved useful in the rest of the graph. The specific value we chose in the implementation for f was 0.9 and for r 50(%). The results of this heuristic (called **Q**) are given in Table 4.

Table 4

	T		B		TBT		SZ	
No	**E**	**Q**	**E**	**Q**	**E**	**Q**	**E**	**P**
1	109	120	14.42	14.16	49	63	10.24	7.60
2	113	114	14.97	15.01	54	60	10.22	9.5
3	92	110	15.37	14.78	55	58	10.58	8.41
4	90	98	16.44	16.40	59	64	12.18	11.32
5	68	69	16.69	16.71	41	53	12.26	10.60
6	60	64	17.12	16.35	44	48	12.70	11.54
7	59	57	18.24	17.28	66	61	15.95	12.14
8	12	12	9.05	9.05	119	119	4.02	4.02
9	8	8	8.92	8.92	116	116	4.05	4.05
10	142	161	12.33	11.91	93	85	8.33	7.11
11	81	74	17.64	17.13	69	71	15.20	12.87
12	152	171	12.83	12.34	96	98	7.79	6.42
13	131	134	18.36	17.97	100	93	16.66	14.51
14	294	307	14.00	13.51	181	168	7.80	6.90
15	271	274	19.49	18.91	174	161	16.08	13.24
16	278	291	13.69	13.41	164	156	7.55	6.94
17	456	489	14.58	14.22	248	242	7.69	6.49
18	363	361	19.96	19.47	225	216	15.61	13.28
19	334	346	17.53	16.90	215	203	12.35	10.27
20	63	64	16.42	16.85	33	61	11.93	10.40

The increase in the number of testers is again slight but the drop in the values of B and in addition on SZ are signs of very good behavior. The overall number of *tbt* nodes is also decreased. The reason that in some instances **Q** has larger TBT values than **E** is that the TBT entries were computed for the segment with the maximum value in B in each case, and **Q** happens to result

in such segments with a larger number of *tbt*'s than **E**. For comparison reasons, Table 5 contains the results of a partition based heuristic (called **P** in [12]), which is the same as **G** but explicitly enforces partition with each placement at each step. The large hardware overhead does not justify the low B values.

Table 5

	T		B		TBT		SZ	
No	**Q**	**P**	**Q**	**P**	**Q**	**P**	**Q**	**P**
1	120	175	14.16	8.27	63	10	7.60	12.20
2	114	161	15.01	9.03	60	33	9.50	8.93
3	110	139	14.78	12.03	58	12	8.41	13.08
4	98	125	16.40	13.49	64	30	11.32	13.36
5	69	105	16.71	14.79	53	28	10.60	13.10
6	64	110	16.35	14.01	48	13	11.54	10.38
7	57	105	17.28	13.10	61	10	12.14	15.90
8	12	37	9.05	8.01	119	133	4.02	3.82
9	8	27	8.92	7.17	116	129	4.05	3.87
10	161	252	11.91	9.54	85	16	7.11	4.18
11	74	174	17.13	15.12	71	26	12.87	6.26
12	171	380	12.34	9.72	98	7	6.42	6.71
13	134	344	17.97	15.60	93	14	14.51	5.35
14	307	678	13.51	9.75	168	14	6.90	6.28
15	274	586	18.91	16.30	161	23	13.24	5.86
16	291	637	13.41	9.19	156	6	6.94	5.00
17	489	913	14.22	10.20	242	33	6.49	3.87
18	361	753	19.47	17.34	216	28	13.28	8.96
19	346	796	16.90	14.89	203	34	10.27	6.73
20	64	135	16.85	12.82	61	25	10.40	8.56

5 Conclusions-Future Work

We proposed a metric that attempts to capture the difficulty of finding applicable primitive polynomials that will exercise concurrently and exhaustively the *tbt* nodes of the circuit. We have proposed a heuristic that behaves well on this metric. Our future work [13] will continue on the non-necessarily partitioning approach we have described in [12] and here. We already have indication that a primitive polynomial (picked at random after only a couple of trials) satisfies a large percentage of the *tbt* nodes. In Table 6 we give the ratios of satisfied (LI) and unsatisfied (LD) *tbt*'s for heuristic **Q** for primitive polynomials picked with one trial from the table given in [9]. The

unsatisfied *tbt*'s could, in a first approach, be cut from the rest of the graph by placing testers on all their incoming edges. We have shown[13] that the already satisfied *tbt*'s are not affected by the new insertions. We suspect that even this brute force solution incures less overhead than the one of partitioning the circuit.

Table 6

No	LI	LD
1	58	5
2	51	9
3	54	4
4	42	22
5	38	15
6	46	2
7	55	6
8	113	6
9	111	5
10	59	26
11	61	10
12	86	12
13	87	6
14	137	31
15	143	18
16	129	27
17	210	32
18	199	17
19	180	23
20	51	10

References

[1] E. J. McClauskey, S. Bozorgui–Nesbat, "Design for Autonomous Test", *IEEE Transactions on Computers*, vol. C-30, pp. 866–875, 1981.

[2] E. J. McClauskey, "Built–in Self–testing Techniques", *IEEE Design and Test*, pp. 21–28, April 1985.

[3] E. B. Eichelberger, T. W.Williams, "A Logic Design Structure for LSI Testability", *Proc. 14th DAC*, pp. 462–468, June 1977.

[4] S. N. Bhatt, F. R. K. Chung, A. L.Rosenberg, "Partitioning Circuits for Improved Testability", *MIT Conference on Advanced Research in VLSI* , pp. 91–106, 1986.

[5] W. Jone and C. A. Papachristou, "A Coordinated Approach to Partitioning and Test Pattern Generation for Pseudoexhaustive Testing", *26th ACM/IEEE Design Automation Conference* , pp. 525–530, 1989.

[6] W. Jone and C. A. Papachristou, "On Partitioning for Pseudoexhaustive Testing of VLSI Circuits", *IEEE ISCAS'88* , pp. 1843–1846, 1988.

[7] Z. Barzilai, D. Coppersmith, A. L. Rosenberg, "Exhaustive Bit Generation with Application to VLSI Self–testing", *IEEE Transactions on Computers*, vol. C-32, pp. 190–194, 1983.

[8] S. W. Golomb, *Shift Register Sequences*, Aegean Park Press, 1982.

[9] W. W. Peterson, E. J. Weldon, *Error–Correcting Codes*, 2nd ed., MIT Press, Cambridge, MA, 1972.

[10] J. J. Hallenbeck, J. R. Cybrynski, N. Kanopoulos, T. Markas, N. Vasanthavada, "The Test Engineer's Assistant : A Support Environment for Hardware Design for Testability", *Computer* , vol. 22, pp. 59–68, 1989.

[11] A. Mennone, R. L. Russo, "An Example Computer Graph and Its Partitions and Mappings", *IEEE Transactions on Computers*, vol. C-23, pp. 1198–1204, 1974.

[12] D. Kagaris, F. Makedon, S. Tragoudas, "On Minimizing Hardware Overhead for Circuit Testability" , paper submitted to the European Conference on Design Automation. Also Technical Report, Dept. of Math. & Computer Science, Dartmouth College, August 1991.

[13] D. Kagaris, F. Makedon, S. Tragoudas, "A Framework for Testability", paper in preparation.