Physical Design of Testable VLSI: Techniques and Experiments

MARC E. LEVITT AND JACOB A. ABRAHAM, FELLOW, IEEE

Abstract—VLSI circuits can fail in ways that make the traditional stuck-at fault model and test generation techniques inadequate. It will be shown that the layout of these circuits can affect testability and in some cases reduce the number of likely faults in a design, easing test generation. A method is presented to analyze circuits at the symbolic layout level and to enhance testability using local transformations. To demonstrate the application of this technique a set of CMOS standard cells was redesigned. These standard cells are used in the MIS synthesis system [1], allowing the designer to interactively modify designs and perform trade-off analysis on testable designs. To show the usefulness of the technique, an experiment was performed: example circuits were synthesized and test vectors were generated and then used in a transistor-level fault simulator. It was found that the modified designs have significantly higher fault coverage than unmodified designs. A strategy is presented for the synthesis of easily testable combinational random logic circuits.

I. Introduction

THE classical fault model assumes that inputs or outputs of logic gates are stuck-at ONE or ZERO. Previous research has shown that the stuck-at fault model is inadequate for many types of VLSI technologies, such as nMOS [2], [3], CMOS [2], [4], and many types of bipolar logic [5]. Test generation for an arbitrary combinational logic circuit under the classical fault model has been shown to be NP-complete [6]. When the nonclassical faults, such as a line stuck open or two wires bridged together, are added to the fault set, test generation becomes even harder because classical faults change only the logic function while the nonclassical faults can change the whole circuit structure and operation. In order to cope with the complexity of testing large circuits, various design methods have been developed that reduce the problem to a more manageable one. They are collectively known as design for testability (DFT) [7]. Our approach in this paper is to use DFT techniques at the physical level to help mitigate the problem of the nonclassical faults. To demonstrate the technique we will use CMOS technology as an example.

Manuscript received August 2, 1989; revised December 7, 1989. This work was supported in part by the Semiconductor Research Corporation under University of Illinois-Urbana Contract 87-DP-109 and University of Texas-Austin Contract 88-DP-142, and in part by a grant from Tektronix Incorporated at the University of Illinois-Urbana.

M. E. Levitt is with the Computer Systems Group, Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801.

J. A. Abraham is with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712.

IEEE Log Number 8934061.

To model CMOS properly, one must not only look at the classical stuck-at faults but also nonclassical faults [2], [4], [8]. A combinational circuit can exhibit sequential behavior in the presence of stuck-open faults. Circuits can have intermediate output values if a transistor is stuck-on. Under a bridging fault, the logic function can completely change and remain combinational or turn sequential. In previous works it was shown that these faults do actually occur in production chips [9]–[12] and can result from the interaction of spot defects, as demonstrated by inductive fault analysis [13]–[15]. The stuck-open problem can be so great that in one study it was shown to lead to an escape rate of 1210 defective parts per million when the device was tested with a 100% stuck-at test set [9]. Since conventional stuck-at test coverage of these faults is inadequate [9], [13], [16], [17] but stuck-at test generation tools are good, we will apply layout-level DFT (LLDFT) in order to improve stuck-open coverage of stuck-at tests for CMOS VLSI.

II. Previous Work

DFT methods have been around since 1968. Only in the last few years have methods that improve testability at the circuit level been introduced. In this section, CMOS circuit level DFT techniques are reviewed. The techniques are divided into two groups: switch-level techniques and layout-level techniques.

A. Switch Level

The switch-level methods modify the circuit at the transistor level, where the transistor is treated as an ideal switch. The two classes of faults that can be handled using these methods are stuck-opens and stuck-ons in static CMOS logic. First, work involving stuck opens will be treated, then the methods covering both faults.

1) Stuck-Opens: Reddy *et al.* have proposed several methods to augment static CMOS gates so that they are testable for single stuck-open faults [18]. The methods require adding one or two control lines and two or four transistors. There are two techniques that apply to arbitrary complex gates. For the first method three patterns are needed to test a fault; the second method requires only two patterns. A method applicable only to AND–OR networks is also presented. It requires two patterns to detect a fault. No data were given for area overhead or performance penalties for circuits designed with these techniques. An advantage of this method is that the tests are not invalidated by circuit delays.

Jha and Abraham have introduced the hybrid-CMOS realization of AND–OR or OR–AND functions [19]. The extra transistors required for this method depend on the function being implemented. Its advantages are that no additional control line inputs are needed and the two-pattern tests generated are valid under delays.

Liu and McCluskey have presented a method for circuits in a scan path [20]. Two test patterns are needed per fault. A 4-b adder was designed using this technique, which used 8% more transistors with negligible change in the critical path delay. The overhead for other example circuits ranged from 7 to 19%. It should be noted that these circuits require a special shift-register latch (SRL) that would take up more area than a regular one. No data was provided concerning the SRL's area overhead or performance degradation.

2) Stuck-Ons and Stuck-Opens: Brzozowski has presented a method to design CMOS cells that allows the detection of both types of faults [21]. Two control lines are used and the inputs and their complements must be controllable as in the work by Liu and McCluskey [20]. The test takes two patterns for stuck-opens and two for stuck-ons. No data was available on the overhead required to actually use this technique.

Liu and McCluskey extended their previous work to come up with easy stuck-open/on fault testing for CMOS circuits in a scan path [22]. The method uses the same testable gate as Brzozowski but inserts an inverter after every nonprimary output logic gate. A three-pattern test will detect both a stuck-on and stuck-open fault. Some combinational cells were designed using these methods. The area overhead was around 25% and the critical path delay increased by 15%. Again, no overhead was given for the special scan latches needed for this technique.

The switch-level techniques mentioned have some major drawbacks. First, all add extra transistors and area to the design. Second, most need extra control lines routed to each gate. Last, they are not applicable to all CMOS logic styles. Due to the lack of numerous examples of circuits designed with these techniques, the actual costs cannot be quantified accurately for the wide range of circuits encountered in design.

B. Layout Level

Zasio has presented a method for gate arrays [23]. He configures the gate macro such that stuck-open faults are forced to appear as stuck-at faults. A problem with this method is that it cannot be used in any other design style and is *ad hoc* in nature, making the development of CAD tools very hard.

Koeppe presented a set of layout rules to cope with stuck-opens [24]. The stuck-opens are avoided or their effects are made easier to detect when these rules are applied. He proposes replacing parallel transistor structures by a ring-shaped LOCOS structure. Two methods were presented to handle gate line faults. The first method is redundant gate wiring, forming a loop between the pMOS and nMOS transistor inputs. The second is to use branchless wiring in fixed order. This ensures the direct coupling of both complementary transistors. The overhead of these rules results in an area increase of 6 to 20% and a delay increase of $\approx 12\%$.

The layout-level techniques presented do not add any control lines but require the designer to work at a low-level design representation. The techniques are not formalized and therefore require separate analysis and modification for each circuit. They are time consuming to perform, most likely account for the minimal number of example circuits presented.

III. Layout-Level Design for Testability

Over the last few years more attention has been paid to how a circuit's layout can influence testability [23]–[26]. Testability can be influenced at the layout level in the following three ways:

1) single faults or complete classes of faults can be eliminated,
2) "hard" to detect faults can be made "easier" to detect, and
3) test-pattern invalidation due to glitches or delays can be avoided.

In this section a method to enhance testability by modifying a circuit at the layout level will be presented. The method uses local transformations to design out some stuck-open faults and make the remaining stuck-open faults easier to detect.

In order to define what constitutes a easily testable design, the fault model and the design evaluation criteria must be examined. The fault model assumed in this work is that any given line can be stuck-at or stuck-open. We assume that the probability of an open in a metal line is much greater than that of a diffusion line of the same size, due to the manufacturing process. A major area where stuck-open faults will manifest themselves will be at the contacts. The reasoning behind this is that contacts are made up of a series of layers, each of which has to be defect free. Thus, the probability of a contact being defect free is the product of each step involved being defect free, which is less than or equal to any individual step making up the contact. This assumption that contacts have a high failure rate and cause opens is confirmed by published literature in quality control for integrated circuit manufacturing [27], in the area of reliability [28], [29], in published

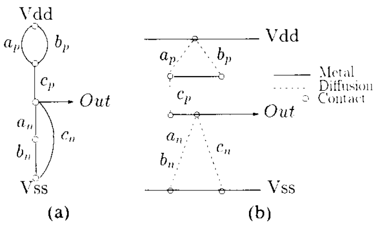

Fig. 1. (a) Standard switch graph. (b) Extended switch graph.

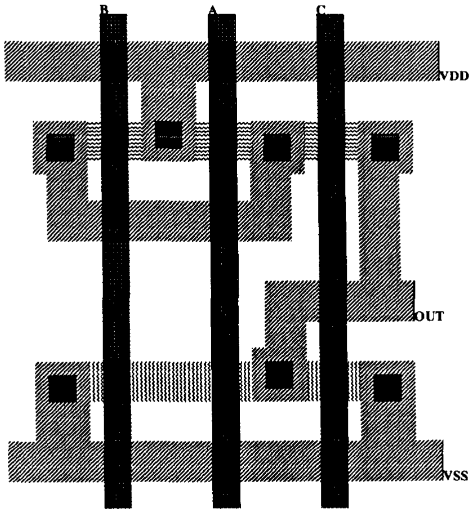

Fig. 2. Layout of Fig. 1.

failure analysis studies [11], [12], and in conference papers [10].

There are two criteria for evaluating a design. The first is to compare two designs based on an easily computed measure of the number of nonequivalent faults. An implementation of a design is considered more testable than another implementation if it has fewer nonequivalent faults. The reason behind this is that if one design's fault set is a subset of the other, in the worst case it will require the same test set. The second evaluation criterion is to check the actual fault coverage by using fault simulation and a set of test vectors.

A. Local Transformation for LLDFT

Local transformations to improve circuit design have been used by the CAD community for some time [30]. The main idea behind local transformations is that a local block of a design is replaced by a "better" functionally equivalent block, as measured by a defined criterion. This is repeated until no improvements can be made and no transformations are left to try.

To design out stuck-open faults, the local transformations work on an extended switch graph (*esg*) representation of the circuit. The extended switch graph differs from a standard switch graph in that it keeps track of some implementation details, notably the routing layers between

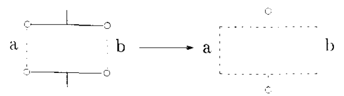

Fig. 3. Transform 1.

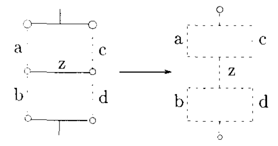

Fig. 4. Transform 2.

the transistors (metal or diffusion) and contacts used. Thus it is a cross between a switch graph and a symbolic (sticks) layout. The example function, $\overline{Out} = (a \wedge b) \vee c$, is shown in the following set of figures. Fig. 1(a) shows the standard switch graph, Fig. 1(b) is the extended switch graph, and Fig. 2 is the actual layout.

B. The Local Transformations

A library of local transformations has been developed that enhances testability for CMOS circuits. This was done by hand analysis and computer simulation of basic CMOS logic building blocks. Some example transformations are as follows.

Transform 1 (Fig. 3): This transform is similar to the LOCOS structure rule described by Koeppe [24]. For the unmodified layout an open at a contact can happen at either branch of a set of parallel transistors or in the metal line after the fan-out point. If this happens, a simple test may not detect an open fault. Both branches must be tested separately. The modified layout does not have a metal fan-out point and only one equivalent contact fault is possible, given our assumption of high yielding diffusion wires. To detect the open fault only one branch needs to be tested. The number of nonequivalent faults has been reduced from two to one, and it is easier to find a test for the modified block due to the freedom in assigning inputs during test generation. This type of fault has been shown to manifest itself in production chips [10]. An initial functional test missed the fault and only an extended test caught the fault. If the modified layout had been used, the failure would have disappeared or the first functional test would have caught it, reducing test time.

Transform 2 (Fig 4): In this transform an OR–AND block is made more testable. The first modification is similar to that of Transform 1, removing the fan-out nodes in the parallel branches. When this is done for the internal wire, labeled z, a hard-to-detect function-altering fault is removed. The original pass function of this block is $(a \vee c) \wedge (b \vee d)$. If an open occurs in the metal line z, the function is altered to $(a \wedge b) \vee (c \wedge d)$. In the redesigned block an open in line z will render e block inoperative. This condition is easier to test for. The transform's overall effect is to reduce the fault list and remove a hard-to-detect function-altering fault.

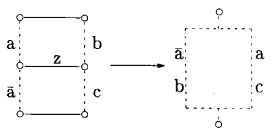

Fig. 5. Transform 3.

Transform 3 (Fig. 5): This transform rearranges the input assignment to allow the removal of a conduction path, line z. The logic function is exactly the same as before but a failure at a contact in the modified gate renders it inoperative, unlike the original. By removing the metal line and rearranging the inputs, the hard to detect faults due to line z are eliminated. Also, in this transform, the fan-out from the metal lines has been reduced.

The local transformations need a control strategy to guide their application. The one chosen was a bottom-up approach. This was due to the low-level nature of the input to the system. The transforms first act at the individual switch level, and then they are applied at a group level. The groups, which act as the switches, can be the previously transformed set of individual switches (transistors). By using this control strategy, any number of parallel and series-parallel networks can be transformed to a more easily testable network.

IV. Design Examples

This section presents a standard cell library that was designed with techniques described in the previous section. Standard cells were chosen as the demonstration vehicle because of their popularity in application-specific IC designs and because they are used in the random logic portions of silicon compiled designs [31]. Another factor in this choice is that many CAD tools work with standard cell libraries; thus, it is possible to integrate testability considerations and improvements into these tools with minimal effort.

This library is a modified subset of the CMOS3 library by Heinbuch [32]. A subset of the cells was chosen for modification. They are the inverter cells, NAND cells, and NOR cells. The basic conversion procedure employed was:

1) extract the *esg* from the layout,
2) apply local transforms to the *esg*, and
3) lay out a new version of the cell that is compatible with the new *esg*.

After the cells had been laid out, they were extracted to produce a simulation file for SPICE. The cell characterization circuit is shown in Fig. 6. Two values were extracted from the output waveforms t_r and t_f: the 10% to 90% rise time and the 90% to 10% fall time. The average delay of the gate is then given by $\tau_{\text{ave}} = (t_r + t_f)/4$. The regular cell library characteristics are given in Table I. The testable cell characteristics are given in Table II. The overheads in area, delay, and fault list are given in Table III.

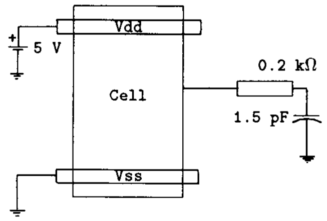

Fig. 6. Delay characterization circuit.

TABLE I
Regular Standard Cell Characteristics

Cell	Width(λ)	τ_{ave}(ns)	Faults
INV	16	1	6
NOR2	24	3.75	9
NOR3	32	4.25	15
NOR4	40	5	19
NAND2	24	3	10
NAND3	32	3.75	15
NAND4	40	4.25	19

TABLE II
Testable Standard Cell Characteristics

Cell	Width(λ)	τ_{ave}(ns)	Faults
INV	18	1.125	4
NOR2	24	4	7
NOR3	32	5.25	11
NOR4	40	5.5	15
NAND2	24	3.25	7
NAND3	32	4.75	11
NAND4	40	5.5	13

TABLE III
Comparison of the Two Cell Libraries

Cell	Area Overhead%	Delay Overhead%	Fault Reduction%
INV	12.5	12.5	33.3
NOR2	12.5	6.67	22.2
NOR3	12.5	23.53	26.67
NOR4	0	10	21.05
NAND2	16.67	8.33	30
NAND3	12.5	26.67	26.67
NAND4	20	29.4	31.6

In order to show the usefulness of LLDFT, we coupled multilevel logic synthesis with the standard cell library and ran numerous design examples through the synthesis system shown in Fig. 7. The example circuits fell into three groups. The first group was the combinational logic portion of a RISC microprocessor that consisted of 57 logic blocks with a total gate count of 6730. The complexity of the blocks ranged from four gates to around 550 gates. The second group was the ISCAS combinational test generation benchmarks [33]. The last group was a collection of miscellaneous industry and university circuits that included the logic synthesis benchmarks from the 1986 Design Automation Conference [34]. The gate complexity ranged from 100 to over 3000 for the designs in this group.

The designs were first globally minimized using a MIS script. Next the designs were mapped into the two cell libraries [35], regular and easily testable. The mappings were optimized for each cost function available. The area, critical path delay, and number of faults present were collected as figures of merit for each mapping of a circuit.

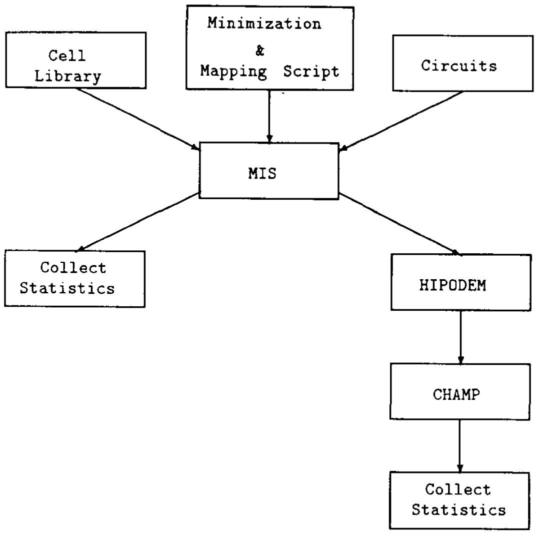

Fig. 7. Synthesis system setup.

TABLE IV
Overhead of Testable Design: Area Minimization Mapping Criteria

Group	Δarea%	Δdelay%	Δfaults%
RISC	12.91±5.41	17.8±11.3	-29.2±3.17
ISCAS	14.78±0.9	12.9±5.6	-29±0.7
MISC	13.96±2.28	12.56±4.9	-28.11±1.7

TABLE V
Overhead of Testable Design: Delay Minimization Mapping Criteria

Group	Δarea%	Δdelay%	Δfaults%
RISC	17.14±8.9	10.64±5.46	-27.9±5.57
ISCAS	15.63±1.4	8.25±1.85	-29.9±1
MISC	15.52±1.68	8.03±3.33	-29.21±1.37

TABLE VI
Overhead of Testable Design: Fault Minimization Mapping Criteria

Group	Δarea%	Δdelay%	Δfaults%
RISC	15.15±5.25	12.08±10.85	-27.9±4.33
ISCAS	14.93±0.5	5.5±1.37	-28.77±0.73
MISC	14.67±1.15	6.96±7.12	-28.06±1.17

The results of the synthesis examples are given in terms of the percentage overhead of the testable version versus the regular implementation. The overhead was calculated for area, critical path delay, and number of faults in a circuit. The results consist of the average and standard deviation of the overhead for each group of circuits.

The results for area minimization during the technology mapping phase are given in Table IV. Table V has the results for when critical path delay was minimized. In Table VI the number of faults were used as the mapping criteria.

The three tables show that the change in delay is the least predictable of all the figures of merit, i.e., it has high

TABLE VII
Overhead of Testable Design: Direct Mapping

Group	Area Optimization		Delay Optimization	
	Δarea%	Δfaults%	Δarea%	Δfaults%
RISC	14.33±1.87	-28.77±3.41	14.84±0.71	-29.41±1.03
ISCAS	15.31±0.23	29.62±0.25	15.28±0.14	-29.86±0.25
MISC	14.17±1.45	28.52±1.17	14.99±0.42	29.2±0.67

standard deviation. This seems to result from the wide range of delay overhead in the individual easily testable standard cells themselves. The changes in the other figures of merit are near constant regardless of the mapping criteria. The average area overhead is approximately 15% while fault list reduction is 28–29%. The largest standard deviations in the tables are for the RISC microprocessor group. When the RISC group was examined closely, it was observed that most of the variation comes from a few large values in the data tables. These large values are mostly due to the various multiplexors in the RISC design, which MIS maps inefficiently to a tree of NOR gates due to the limited cell library.

To see the effect of a "dumb" translation of a design, the circuits were converted from the regular to easily testable standard cells without any technology mapping being performed. This will be referred to as direct mapping. The results are given in Table VII. It can be seen that the area and fault overhead is approximately the same as with technology mapping. The standard deviation is smaller because there is no structural change happening during the translation from a regular to an easily testable design. Unfortunately, the critical path delays could not be calculated when direct mapping was performed. To get a feel for the critical-path-delay overhead, circuits that were directly mapped by MIS were examined. The results for area are $14.78 \pm 2.12\%$, for the number of faults $-28.95 \pm 1.64\%$, and for the delay $13.35 \pm 9.79\%$. The area and fault overhead fall into the same range as Table VII but the delay is still very unpredictable. If the RISC MUX problem is taken into account the new Δ delay value would be $9.13 \pm 4.67\%$. This value is a little better, but of no use in making generalizations about LLDFT overhead in the critical path delay of a circuit.

V. Testing

To investigate the true effectiveness of the design methodology, switch-level fault simulation was performed on some circuits from each group. The simulations were done using CHAMP [36] on both the regular and easily testable designs. A gate-level stuck-at test pattern set was generated for each circuit using HIPODEM [37]. Table VIII gives the fault coverage when the stuck-at test pattern set is used. The designs were also evaluated for random pattern test sets. Random patterns were applied to each version of the circuit and the results are given in Table IX

The tables show that by using layout-level design for testability, the coverage of stuck-open faults by the stuck-at test vector set can be greatly improved. This will allow the

TABLE VIII
FAULT COVERAGE OF SELECTED CIRCUITS, DETERMINISTIC PATTERNS

Circuit	Regular		Testable	
	stuck-at (%)	stuck-open (%)	stuck-at (%)	stuck-open (%)
RISC1	100	76.7	100	100
RISC2	100	73	100	100
ISCAS17	100	87	100	100
ISCAS499	100	82.6	100	99.6
MISC1	100	76.5	100	98.7
MISC2	100	67.65	100	94.6

TABLE IX
FAULT COVERAGE OF SELECTED CIRCUITS, RANDOM PATTERNS

Circuit	Regular		Testable	
	stuck-at (%)	stuck-open (%)	stuck-at (%)	stuck-open (%)
RISC1	100.0	74.0	100.0	100.0
RISC2	100.0	83.0	100.0	100.0
ISCAS17	100.0	80.0	100.0	100.0
ISCAS499	100.0	70.0	100.0	98.0
MISC1	100.0	77.0	100.0	100.0
MISC2	100.0	79.7	100.0	95.6

TABLE X
COMPARISON OF AREAS UNDER DIFFERENT MAPPING CRITERIA

Circuit Group	Area Optimization	Delay Optimization	Fault Optimization
RISC	0.79	0.18	0.40
ISCAS	0.60	0.00	0.60
MISC	0.73	0.13	0.37

designer to use traditional test generation tools or to use built-in self-test (BIST) techniques [7] and know that both classical stuck-at and nonclassical stuck-open fault coverage is high.

VI. A SYNTHESIS STRATEGY

In this section some general guidelines for the synthesis of testable random logic are presented. It should be noted that the results presented here are applicable only to logic designed using LLDFT standard cells.

To derive the guidelines for how a designer should proceed when synthesizing testable logic, all the easily testable solutions for a given circuit were compared to each other. For a given circuit the values for a metric (either area, delay, or number of faults) are compared and the mapping criterion for the optimal solution was recorded. A ratio was then computed that consisted of the number of optimal solutions a mapping criterion produced for a given metric divided by the total number of circuits in the group. This was done for each metric and each circuit group. It should be noted that the ratio computed is not a probability because more than one mapping criterion can produce the best solution. The results are shown in Tables X–XII for the three figures of merit: area, critical path delay, and number of faults, respectively. As expected, the highest ratios are in the columns that correspond to the metric being examined.

During analysis it was noted that the difference in some values was negligible. This led to the computation of a new set of tables. The new tables count a solution as optimal if it is within 2.5% of the true optimal solution. Thus, if the

TABLE XI
COMPARISON OF DELAY UNDER DIFFERENT MAPPING CRITERIA

Circuit Group	Area Optimization	Delay Optimization	Fault Optimization
RISC	0.26	0.93	0.25
ISCAS	0.20	1.00	0.20
MISC	0.10	0.97	0.30

TABLE XII
COMPARISON OF FAULTS UNDER DIFFERENT MAPPING CRITERIA

Circuit Group	Area Optimization	Delay Optimization	Fault Optimization
RISC	0.51	0.21	0.72
ISCAS	0.2	0	1
MISC	0.47	0.13	0.80

TABLE XIII
COMPARISON OF AREA UNDER DIFFERENT MAPPING CRITERIA WITH ERROR

Circuit Group	Area Optimization	Delay Optimization	Fault Optimization
RISC	0.86	0.21	0.61
ISCAS	0.80	0.00	0.60
MISC	0.90	0.17	0.70

TABLE XIV
COMPARISON OF DELAY UNDER DIFFERENT MAPPING CRITERIA WITH ERROR

Circuit Group	Area Optimization	Delay Optimization	Fault Optimization
RISC	0.28	0.93	0.25
ISCAS	0.20	1.00	0.20
MISC	0.10	0.97	0.30

TABLE XV
COMPARISON OF FAULTS UNDER DIFFERENT MAPPING CRITERIA WITH ERROR

Circuit Group	Area Optimization	Delay Optimization	Fault Optimization
RISC	0.93	0.32	0.93
ISCAS	1.00	0.20	1.00
MISC	0.80	0.30	0.97

optimal mapping produced a design with 100 faults and another mapping produced a design with 101 faults, the fault ratio counter for both mapping criteria would be incremented. The results of this recomputation are given in Tables XIII–XV. When these numbers are analyzed, it can be seen that the area optimization mapping produces very good results not only for area but also for the number of faults. Therefore the number of faults in a design can be eliminated as a technology mapping criterion.

Using the results presented, the following synthesis strategy is proposed for multilevel logic. If a designer can afford to have $\approx 15\%$ area overhead for a $\approx 28\%$ reduction in the number of faults in a circuit, the following steps should be taken:

1) Use area optimization when mapping a design to testable standard cells.
2) Check the critical path delay. If it meets the design specifications then the design is done.

3) If the delay is too large, remap the circuit using delay as the mapping criterion.
4) Check the delay and area against the system specifications. If they meet the specifications then the design is done.
5) If the delay and area are too high then no solution currently exists; the circuit may need to be reoptimized or a "mixed" cell library can be used for technology mapping.

VII. Conclusion

Layout-level design for testability, LLDFT, has been shown to yield more easily testable logic. Rules were developed for basic building blocks and then represented as a set of local transformations. The local transformations allow for the formal analysis and application of LLDFT. As an example, a set of design rules has been derived that allows for the design of more easily testable CMOS logic. The local transformations can easily be changed and augmented and are applicable to all forms of CMOS logic.

It was then demonstrated that random logic blocks implemented in the standard cell methodology can be made more easily testable by using a standard cell library that was designed with layout-level design for testability rules. For an increase in the area of 15%, the fault list can be reduced by 28% allowing the stuck-at vector set to increase its coverage of stuck-open faults by up to 35% and many times achieve at or near 100% coverage. It should be remembered that the area and delay overhead is only incurred for the random logic portion of a design and the total chip overhead for many designs will be less. The method also achieves high stuck-open coverage using pseudorandom tests; thus, it can be used to improve fault coverage in circuits designed using certain BIST techniques. When synthesizing the testable logic, the criterion for technology mapping should be either area or delay. The experimental results suggest that area mapping should be tried first.

There are a number of possible areas for future work in layout-level design for testability. First, the current techniques could be applied to sequential logic blocks, such as the flip-flops and latches in the standard cell library. Deriving more local transformation rules and generalizing the present ones can help in adding testability to more basic building blocks. Technology-specific transformation rules could be developed for specific design styles and processes, accounting for such things as defect densities and the likelihood of interlayer interactions. Layout-level DFT can be used for designing out bridging faults, which are a growing problem in testing. We would also like to investigate how the transforms improve yield and reliability. It has been known that metal migration at contacts is a reliability problem [28], [29]. Our reduction of this problem can possibly improve yield after burn-in [12], [25] and reduce field failure rate. This can help to pay for some or all of the area cost imposed by our methods, although verifying this is presently not possibly by the authors.

In the field of random logic synthesis the standard cell library should be expanded and optimized more carefully. This will lead to a reduction in the area and delay overhead that is currently incurred for a design.

Acknowledgment

The authors wish to thank Prof. R. Saleh of the University of Illinois for his help with the Berkeley Synthesis System. The authors also wish to thank S. Gibbs and J. Tobin of Hewlett-Packard's Northwest Integrated Circuit Division for their helpful comments and criticisms.

References

[1] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, no. 6, pp. 1062–1081, Nov. 1987.
[2] P. Banerjee and J. A. Abraham, "Fault characterization of VLSI MOS circuits," in *Proc. IEEE Conf. Circuits and Computers*, 1982, pp. 564–568.
[3] W. Maly, "Realistic fault modeling for VLSI testing," in *Proc. 24th Design Automation Conf.*, 1987, pp. 173–180.
[4] R. Wadsack, "Fault modeling and logic simulation of CMOS and MOS integrated circuits," *Bell Syst. Tech. J.*, pp. 1449–1473, May–June 1978.
[5] C. Beh, K. Arya, C. Radke, and K. Torku, "Do stuck fault models reflect manufacturing defects," in *Proc. Int. Test Conf.*, 1982, pp. 35–42.
[6] H. Fujiwara, *Logic Testing and Design for Testability*. Cambridge, MA: M.I.T. Press, 1985.
[7] T. Williams and K. Parker, "Design for testability—A survey," *IEEE Trans. Comput.*, vol. C-31, pp. 2–15, 1982.
[8] D. Baschiera and B. Courtois, "Advances in fault modelling and test pattern generation for CMOS," in *ICCAD Dig. Tech. Papers*, 1986, pp. 82–85.
[9] B. W. Woodhall, B. D. Newman, and A. G. Sammuli, "Empirical results on undetected CMOS stuck-open failures," in *Proc. Int. Test Conf.*, 1987, pp. 166–170.
[10] J. M. Soden, R. K. Treece, M. R. Tatlor, and C. F. Hawkins, "CMOS IC stuck-open fault electrical effects and design considerations," in *Proc. Int. Test Conf.*, 1989, pp. 423–430.
[11] J. A. Cunningham, *CMOS Technology: Overcoming Yield Problems*, Technology Associates, 1987.
[12] H. K. Dicken, *Physics of Semiconductor Failures*, D. M. Data Inc., 1988.
[13] F. J. Ferguson and J. P. Shen, "Extraction and simulation of realistic CMOS faults using inductive fault analysis," in *Proc. Int. Test Conf.*, 1988, pp. 475–484.
[14] M. Jacomet, "FANTESTIC: Towards a powerful fault analysis and test pattern generator for integrated circuits," in *Proc. Int. Test Conf.*, 1989, pp. 633–642.
[15] J. Grácio, P. Bicudo, N. Rua, A. Olivedira, and C. Almeida, "Test preparation and fault analysis using a bottom-up methodology," in *Proc. European Test Conf.*, 1989, pp. 168–174.
[16] C. Timoc, W. Scott, K. Wickman, and L. Hess, "Adaptive self-test for a microprocessor," in *Proc. Int. Test Conf.*, 1983, pp. 701–703.
[17] C. Timoc and W. Scott, "Simulation of stuck-open faults in CMOS integrated circuits," in *Proc. Int. Symp. Test and Failure Analysis*, 1981, pp. 53–56.
[18] S. M. Reddy, M. K. Reddy, and J. G. Kuhl, "On testable design for CMOS logic circuits," in *Proc. Int. Test Conf.*, 1983, pp. 435–445.
[19] K. Jha and J. Abraham, "Design of testable CMOS logic circuits under arbitrary delays," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, no. 3, pp. 264–268, 1985.
[20] D. L. Liu and E. J. McCluskey, "CMOS scan-path IC design for stuck-open fault testability," *IEEE J. Solid-State Circuits*, vol. SC-22, no. 5, pp. 880–885, Oct. 1987.
[21] J. A. Brzozowski, "Testability of combinational networks of CMOS cells," in *Developments in Integrated Circuit Testing*, D. M. Miller, Ed. San Diego, CA: Academic, 1987, pp. 315–357.

[22] D. L. Liu and E. J. McCluskey, "Designing CMOS circuits for switch level testability," *IEEE Design and Test*, pp. 42–49, Aug. 1987.
[23] J. J. Zasio, "Non stuck fault testing of CMOS VLSI," in *Proc. CompCon*, 1985, pp. 388–391.
[24] S. Koeppe, "Optimal layout to avoid CMOS stuck-open faults," in *Proc. 24th Design Automation Conf.*, 1987, pp. 829–835.
[25] D. Wu, C. Beh, and C. Radke, "Improve yield and quality through testability analysis of VLSI circuits," in *Proc. Int. Test Conf.*, 1984, pp. 713–717.
[26] T. H. Spencer and J. Savir, "Layout influences testability," *IEEE Trans. Comput.*, vol. C-34, no. 3, pp. 287–290, Mar. 1985.
[27] M. Phadke, R. Kackar, D. Speeney, and M. Grieco, "Off-line quality control in integrated circuit fabrication using experimental design," *Bell Syst. Tech. J.*, vol. 62, no. 5, pp. 1273–1309, May–June 1983.
[28] H. Tomioka, S. Tanabe, and K. Mizukami, "A new reliability problem associated with Ar ion sputtered cleaning of interconnect vias," in *Proc. Int. Reliability Phys. Symp.*, 1989, pp. 53–58.
[29] P. Gargini, C. Tseng, and M. Woods, "Elimination of silicon electromigration in contacts by the use of an interposed barrier metal," in *Proc. Int. Reliability Phys. Symp.*, 1982, pp. 66–76.
[30] J. A. Darringer, W. H. Joyner, C. L. Berman, and L. Trevillyan, "Logic synthesis through local transformations," *IBM J. Res. Develop.*, vol. 25, no. 4, pp. 272–280, July 1981.
[31] D. D. Gajski, Ed., *Silicon Compilation*. Reading, MA: Addison-Wesley, 1988.
[32] D. V. Heinbuch, *CMOS3 Cell Library*. Reading, MA: Addison-Wesley, 1988.
[33] F. Brglez and H. Fujiwara, "Neutral netlist of ten combinational benchmark circuits and a target translator in fortran," presented at the IEEE Int. Symp. Circuits Syst., 1985.
[34] A. J. de Geus, "Logic synthesis and optimization benchmarks for the 1986 design automation conference," in *Proc. 23rd Design Automation Conf.*, 1986, p. 78.
[35] E. Detjens, G. Gannot, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, "Technology mapping in MIS," in *ICCAD Dig. Tech. Papers*, 1987, pp. 116–119.
[36] D. Saab, R. Mueller-Thuns, D. Blaauw, and J. Abraham, "CHAMP: Concurrent hierarchial and multi-level program for the simulation of VLSI circuits," in *ICCAD Dig. Tech. Papers*, 1988, pp. 246–249.
[37] S. Chandra and J. Patel, "A hierarchial approach to test vector generation," in *Proc. 24th Design Automation Conf.*, 1987, pp. 495–501.

Marc E. Levitt received the B.S. degree in computer engineering from Lehigh University, Bethlehem, PA, in 1986, and the M.S. degree in electrical engineering from the University of Illinois, Urbana-Champaign, in 1989, where he is currently working towards the Ph.D. degree in electrical engineering.

Since 1986 he has been a Research Assistant with the Computer Systems Group at the Coordinated Science Laboratory, University of Illinois. In addition, he has worked at Hewlett-Packard and Digital Equipment Corporation. His research interests include testing, CAD, VLSI system design, manufacturing, and complex systems.

Jacob A. Abraham (S'71–M'74–SM'84–F'85) received the Bachelor's degree in electrical engineering from the University of Kerala, India, in 1970. He received the M.S. degree in electrical engineering and the Ph.D. degree in electrical engineering and computer science from Stanford University, Stanford, CA, in 1971 and 1974, respectively.

From 1975 to 1988 he was on the faculty of the University of Illinois, Urbana. He is now a Professor in the Department of Electrical and Computer Engineering and the Director of the Computer Engineering Research Center at the University of Texas at Austin, where he also holds the eighth Cockrell Family Regents Chair in Engineering. His research interests include fault-tolerant computing, VLSI design and test, computer-aided design, and computer architecture. He is the principal investigator of several contracts and grants in these areas and the director of a major research program in the design of testable systems, funded by the Semiconductor Research Corporation. He is a consultant to industry and government in the areas of testing and fault-tolerant computing. He has published over 100 papers in refereed journals and conferences, and has supervised 22 Ph.D. dissertations.