Automated Testability Enhancements for Logic Brick Libraries*

Jason G. Brown, Brian Taylor, R. D. (Shawn) Blanton, and Larry Pileggi
Center for Silicon System Implementation
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh PA 15213
{jgbrown, briant, blanton, pileggi}@ece.cmu.edu

Abstract

Circuit fabrics composed of highly regular structures, called logic bricks, have been described recently for improving yield. An automated logic brick design flow based on a SAT formulation of the brick routing has been developed to minimize wire length and the number of vias while maintaining several design-for-manufacturability constraints. In this work, testability enhancements are imposed into a logic brick to reduce the likelihood of (i) feedback bridges to improve test and (ii) equivalent faults to improve diagnosis. This is accomplished by adding constraints to the SAT formulation of the logic brick routing that restricts certain wires from being routed in close proximity, thus making bridges between them unlikely. Application to several brick designs resulted in critical-area reductions for targeted bridges with little degradation in terms of additional wire length and via count.

1 Introduction

The testability of an IC is a measure of the ease or cost associated with determining if any defects adversely affect its correct functionality and performance. Design for testability (DFT) techniques are design methods that are employed specifically to improve the testability of a hardware product [1–3]. DFT techniques in the 1940s and 50s used switches and instruments to "scan" the voltage/current at internal nodes of a circuit (*i.e.*, analog scan). Modern techniques are similar in the sense that they make design modifications that improve access to the circuit for enhanced controllability and observability of internal circuit nodes. Other examples of DFT focus on electromechanical characteristics of the product-tester interface [4]. DFT can also consider other aspects of test however that include, for example, the size, shape, and spacing of the probe locations.

DFT can be employed to meet two objectives for test. First, as already alluded to above, the circuit can be designed so that *difficult-to-detect* defects are made easier to detect. A difficult-to-detect defect is one that affects a circuit node that is difficult to control or observe. Adding probe points to the circuit is a traditional approach for making defects affecting these locations easier to detect [1–3]. The scan DFT methodology [1] enables the flip-flops of an IC to be configured into one or more shift registers under a special test mode of operation. Direct control of the flip-flops through the shift operation reduces test development and application cost and enables high fault coverage. JTAG boundary scan [1] is a DFT standard that uses a shift register to provide direct access to chip inputs and outputs using only four I/O pins (clock, input, output, and control). This strategy lessens the need for probing I/O pins.

The second, less focused upon, objective of DFT is to design the circuit's layout so that difficult-to-detect defects are unlikely. Several works however have examined how the physical layout of a design can affect testability [5–7]. Levitt and Abraham [5] described layout-level DFT, where modifications to a standard cell library are performed to reduce the likelihood of stuck-open faults. The authors of [5] presented a study in which a 15% increase in chip area led to a 35% increase in fault coverage. Sudbrock *et al.* [6] proposed a "testability check" that occurs during test generation. If the check indicates that certain faults require high test cost (*e.g.*, test generation time), the layout generation is repeated until the cost is below a given threshold. Xu, Kundu, and Ferguson [7] proposed a strategy called Physical DFT that involves modifying the physical design of a circuit to shift the distribution of faults towards those that are most likely to be detected without modifying the logic-level structure. Specifically, Carafe [8], an inductive fault analysis tool developed at the University of California, Santa Cruz, is used to identify and simulate the most likely open and bridge defect sites and to determine which ones are not detected by a given test set. To make these defects less likely, redundant spacing is added between wires to prevent bridges, and redundant vias and increased wire thicknesses are added to prevent opens. Iizuka, Ikeda, and Asada [9] described a strategy for improving the yield of standard cells that do not lie on a critical path of a design. Based on the notion that standard cells not on the critical

*The authors acknowledge the support of the Focus Center for Circuit and System Solutions (C2S2), one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation Program.

978-3-9810801-3-1/DATE08 © 2008 EDAA

path can have degraded performance, these cells are potentially modified to improve overall yield. Yield improvements are made by simply widening the cell and spreading wires to reduce critical area. Therefore, based on a desired performance for a given cell, the cell with minimum critical area can be selected from the augmented standard cell library. Results show a 24% average reduction in critical area for six standard cells, but incurred an average cell area overhead of 26%.

In this paper, a strategy is proposed to improve the testability of a set of regularized standard cells called logic bricks. The rest of this paper is organized as follows. Section 2 describes the construction of logic bricks using SAT. Section 3 describes how testability enhancements are included in the logic brick routing to minimize critical area (Section 3.1), and reduce the likelihood of difficult-to-detect defects and equivalent faults (Section 3.2). Finally, Section 4 summarizes our strategy, improvements made, and overhead incurred for enhancing the testability of logic bricks.

2 Logic Bricks

Researchers have developed a circuit fabric composed of highly regular structures called *logic bricks* [10]. A logic brick consists of a set of interconnected logic primitives (NAND, NOR, etc.) that has six to ten inputs and one or two outputs. A brick is designed so that all wires on a given layer are unidirectional, have fixed pitch, and are wide enough to avoid notches and landing pads, characteristics which are all intended to improve manufacturability.

A brick layout is created in two phases consisting of (1) transistor placement and (2) routing. Transistor placement is performed using an extended branch-and-bound technique that minimizes wire length and cell area, both of which are estimated during brick creation. Routing for a logic brick is accomplished by transforming the problem to a SAT formulation [11]. A grid is used to represent the routing space, and each grid segment is a potential location for a wire to exist. A Boolean variable $x_{i,j}$ is associated with each grid segment, where x represents the layer of the grid segment (*e.g.*, metal1), i, j represent the x-y location of the segment within the grid, and the value $x_{i,j} = 1(0)$ indicates the segment is filled (empty). Also, each occupied grid segment (*i.e.*, $x_{i,j} = 1$) has an array of bits $\vec{x}_{i,j}$ to indicate the *net identifier* of the net passing through that segment, where $x_{i,j,k}$ represents the kth bit of $\vec{x}_{i,j}$. A set of *routing rules* are defined that constrain the values of the Boolean variables in order to create a valid routing. For example, for two grid segments $x_{a,b}$ and $x_{c,d}$ that share an endpoint (*i.e.*, adjacent segments), if both segments are filled ($x_{a,b} = 1, x_{c,d} = 1$), then they must have the same net identifier, that is, $\vec{x}_{a,b} = \vec{x}_{c,d}$. Routing rules, like the ones just described, are all mapped to a set of constraints that represent a SAT formulation of the brick routing. A traditional SAT solver is used to identify a routing that satisfies all the constraints.

The focus of our work here is to improve logic brick designs for test and diagnosis by adding new constraints to the SAT formulation that allow us to minimize critical area, and reduce the likelihood of difficult-to-detect feedback bridges and equivalent faults.

3 Testability Enhancements

Additional routing rules can be included in the SAT formulation to improve the testability of the logic brick. Section 3.1 describes how the critical area of a logic brick is minimized, and Section 3.2 describes how "forbidden pairs" of nets involved in difficult-to-detect defects and equivalent faults are identified and made improbable through design.

3.1 Critical Area Minimization

Critical area [12] (CA) is a metric that represents the defect sensitivity of a design and includes the regions of the layout where a spot defect of radius r is assumed to cause a circuit failure. Here, we describe a strategy that minimizes the total critical area of logic bricks in order to maximize yield. A set of *critical-area variables* are defined, each of which represents an interaction between two grid segments. For example, as shown in Figure 1, if grid segments i, j and k, l are both filled ($x_{i,j} = x_{k,l} = 1$) and the net identifiers are not the same ($\vec{x}_{i,j} \neq \vec{x}_{k,l}$), the corresponding critical-area variable v is asserted.

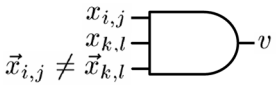


Figure 1: A critical-area variable v is equal to 1 if two grid segments are filled ($x_{i,j} = 1$ and $x_{k,l} = 1$) and their net identifiers are not equal ($\vec{x}_{i,j} \neq \vec{x}_{k,l}$).

Tseitin transformations [13] are used to create conjunctive normal form (CNF) constraints within the SAT formula for these variables. The critical-area variable v is defined by:

$$x_{i,j} \cap x_{k,l} \cap (\vec{x}_{i,j} \neq \vec{x}_{k,l}) \rightarrow v$$

$$\overline{x}_{i,j} \rightarrow \overline{v}, \overline{x}_{k,l} \rightarrow \overline{v}, (\vec{x}_{i,j} = \vec{x}_{k,l}) \rightarrow \overline{v}$$

Additional constraints are added to the SAT formula to ensure the number of active critical-area variables is bounded by some value b. However, since all critical-area variables do not represent equal amounts of critical area, critical area is more accurately calculated by using pseudo-Boolean constraints [14] to weight each critical-area variable. A pseudo-Boolean constraint has the form:

$$\sum_{i=1}^{n} a_i v_i \leq b$$

where a_i and b are integers, and v_i is a Boolean variable. Using pseudo-Boolean constraints, a critical-area variable v_i can be weighted by an integer a_i and a summation of critical-area variables can be bounded by b.

For a routing grid of r rows and c columns, there are rc grid locations and therefore $rc(rc-1)$ critical-area variables to represent all possible interactions between two grid segments. A set of constraints is added to the SAT formula to create the Tseitin transformation for each critical-area variable, and a weight is determined as a function of the Euclidean distance between the two grid segments. The sum of the weighted critical-area variables is bounded by a constant value b and included as a SAT constraint. The SAT solver minisat+ [15], which allows the use of pseudo-Boolean constraints, is used to determine a routing for a set of logic bricks with critical-area constraints. For the three logic brick designs shown in Figure 2, minimal-CA constraints are included in the SAT formula. Table 1 shows the reduction in total critical area that is achieved through this critical-area minimization approach. Over three designs, the average critical area reduction is 10.3%.

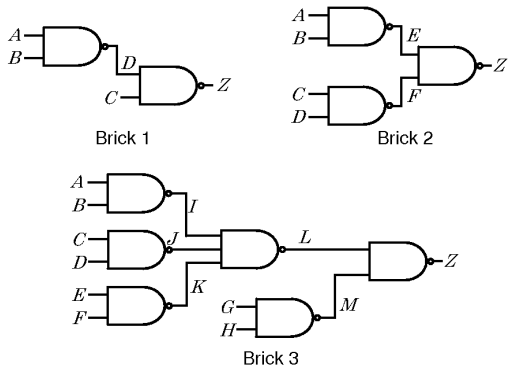


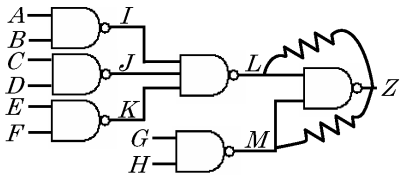
Figure 2: Logic-level schematics for three logic brick designs.

	Total CA (μm^2)	
	No minimal-CA constraints	**Minimal-CA constraints**
Brick 1	329.0	279.8
Brick 2	601.6	581.8
Brick 3	1531.8	1336.7

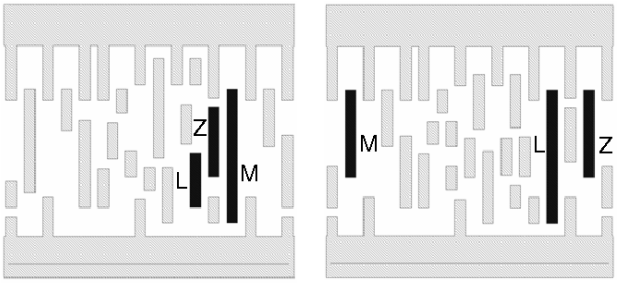
Table 1: CA reductions for three logic brick designs.

3.2 Forbidden pairs

The likelihood of a defect is a function of physical design. For example, the proximity of a pair of nets determines the likelihood of a bridge between them. A logic brick consisting of six logic gates is shown in Figure 3a. Assume a defect involving wires L and Z, and another involving M and Z, both of which cause difficulties for test. In the original physical design (shown in Figure 3b), all three wires exist in close proximity and therefore have some non-negligible probability to bridge. However, in the modified physical design (shown in Figure 3c), these three wires are not routed in close proximity and therefore are quite unlikely to short. The modified version of the brick design is presumably easier to test since these two difficult-to-detect defects are made, through design, unlikely.



(a)



(b) **(c)**

Figure 3: (a) Logic-level schematic of a six-gate logic brick, (b) the original physical design for the brick where L, M, and Z are routed in close proximity, and (c) the modified physical design where L, M, and Z are not routed in close proximity and therefore are unlikely to bridge.

A *forbidden pair* is a pair of nets that causes difficulties for test when routed in close proximity. These forbidden pairs are prevented from being routed in close proximity by adding routing rules to the SAT formulation. Given two nets A and B whose net identifiers are represented with bit vectors $\{A_{m-1} \ldots A_1 A_0\}$ and $\{B_{m-1} \ldots B_1 B_0\}$, respectively, no neighboring grid segments (*i.e.*, two segments that are likely to bridge) are allowed to have these two net identifiers. For grid segment i, j, all neighboring segments are identified. The objective is to ensure that if segment $x_{i,j}$ has net identifier A, then each neighboring segment $x_{p,q}$ does not have net identifier B:

$$\vec{x}_{i,j} = \{A_{m-1} \ldots A_1 A_0\} \rightarrow \vec{x}_{p,q} \neq \{B_{m-1} \ldots B_1 B_0\}$$

$$\bigcap_{k=0}^{m-1} (x_{i,j,k} \,\overline{\oplus}\, A_k) \rightarrow \bigcup_{k=0}^{m-1} (x_{p,q,k} \,\overline{\oplus}\, \overline{B}_k)$$

Since $x \rightarrow y = \overline{x} \cup y$:

$$\bigcup_{k=0}^{m-1} (x_{i,j,k} \overline{\oplus} \overline{A}_k) \cup (x_{p,q,k} \overline{\oplus} \overline{B}_k)$$

For example, if $A = 100$ and $B = 001$, the routing rule instantiates to:

$$\overline{x}_{i,j,2} \cup x_{i,j,1} \cup x_{i,j,0} \cup x_{p,q,2} \cup x_{p,q,1} \cup \overline{x}_{p,q,0}$$

This routing rule is instantiated for every grid segment $x_{i,j}$ and every neighboring grid segment $x_{p,q}$. The set of neighboring grid segments is identified based on an assumed maximum defect radius of 0.25μm, a distance based on the critical dimension of the fabrication technology utilized.

The following sub-sections describe how routing rules are used to improve the testability of logic brick designs by reducing the likelihood of feedback bridges (Section 3.2.1) and equivalent faults (Section 3.2.2).

3.2.1 Feedback Bridges

Bridges that create structural feedback can cause circuit oscillation or intermediate voltage values, and therefore make testing difficult. However, at the transistor level, defects that make testing difficult can be easily identified. Routing rules can then be added to the logic brick SAT formulation to ensure that those defects are unlikely. In other words, if two nets in a logic brick cause feedback when they are bridged, they are identified as a forbidden pair. By examining the transistor-level representation of a given logic brick, the difficult-to-detect feedback bridges are those that have odd parity and have no fanout along the feedback path. If a feedback bridge has odd parity, circuit oscillation may occur and if there is no fanout along the feedback path, there is no way to propagate an error without sensitizing the feedback path. Pairs of lines that meet this criteria are identified as forbidden pairs.

The logic-level representations of three logic bricks analyzed are shown in Figure 2. Table 2 shows the number of potential bridges, the number of feedback bridges, and the number of forbidden pairs that are identified for each brick.

Brick design	No. of bridges	No. of feedback bridges	No. of forbidden pairs
1	12	6	4
2	42	10	6
3	182	30	19

Table 2: The total number of bridges, feedback bridges, and forbidden pairs for three logic-brick designs.

Table 3 compares the routing of three logic bricks using the original SAT-based router [11] (no DFT) and the modified router that includes DFT routing rules to ensure forbidden pairs are never routed in adjacent tracks. The results show an average increase in wire length (measured in number of grid segments) of 7.6% and an average increase in the number of vias by 13.0%. However, the average reduction of critical area (CA) associated with forbidden pairs is 61.5% across the three designs. Serendipitously, the total CA of logic brick 1 is reduced by 14.5%. The total CA of logic-brick designs 2 and 3 however increases by 3.9% and 11.4%, respectively.

Brick type	Total wire length	No. of vias	Total CA (μm^2)	Forbidden-pair CA (μm^2)
Brick 1 (no DFT)	35	7	279.8	91.0
Brick 1 (DFT)	38	8	239.1	36.6
Brick 2 (no DFT)	51	10	581.8	239.5
Brick 2 (DFT)	57	12	604.7	97.8
Brick 3 (no DFT)	169	21	1336.7	464.4
Brick 3 (DFT)	173	22	1489.1	160.5

Table 3: DFT routing rules reduce the CA associated with forbidden pairs but cause an increase in wire length and vias.

Figure 4 shows the effects that forbidden-pair CA reductions have on wire length, the number of vias, and overall CA of logic brick 3. Plot point "0" of each curve represents the logic brick design created with no DFT constraints, and each subsequent point includes an additional set of constraints. Point "1" includes a set of routing rules to ensure wires from a forbidden pair are never adjacent. Point "2" includes the same set of rules as point 1, but also restricts any diagonal adjacencies between forbidden pairs. Finally, point "3" includes the same set of rules as point 2, but also ensures there are no adjacencies within two routing tracks. In each graph, there is a steep decline in forbidden-pair CA for the first set of routing rules added without a significant increase in vias, wire length, or total CA. Although the forbidden-pair CA continues to decrease as more routing rules are added, the reduction is not quite as significant and the increase in wire length, vias, and total critical becomes substantial. Therefore, it seems the first set of routing rules is highly effective while the additional routing rules do not provide favorable improvements, at least for this particular logic-brick design.

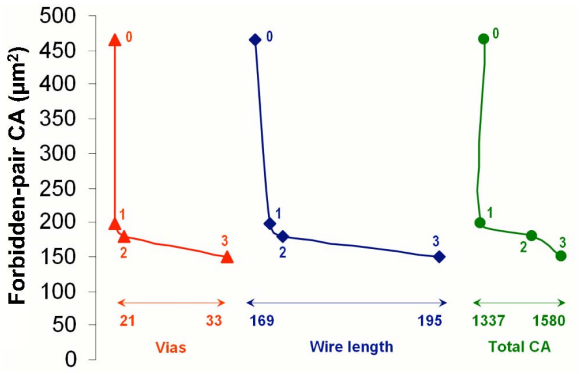


Figure 4: The effect that reducing forbidden-pair CA has on the number of vias, wire length and total CA for logic brick 3.

3.2.2 Equivalent Faults

A faulty logic brick produces output values that differ from the fault-free design. Each fault has an output response that deviates from the expected response. Equivalent faults have identical responses while non-equivalent faults have unique responses. A fault that generates a unique failure response to all possible input patterns to the brick, in theory, will be easier to diagnose since only one fault can cause that response. However, if several faults share the same response, they are virtually impossible to distinguish. Given a set of indistinguishable faults, if all but one of those faults is highly unlikely, diagnosis is improved.

For a given brick design, the failing response for each bridge is derived. Equivalent-fault partitions are then formed based on each fault's response to all possible input combinations of the brick. For each partition containing $n > 1$ faults, the forbidden pairs for each combination of $n - 1$ faults are identified. Any set of forbidden pairs that can be made highly unlikely would lead to a single likely fault with the given response. Since there may be several partitions of equivalent faults, a selection of forbidden pairs is made that distinguishes each partition while minimizing the number of forbidden pairs.

For example, as depicted in Table 4, faults F_1 and F_2 cause the response 111, so either fault F_1 or F_2 should be made unlikely in order to diagnose the cause of the erroneous response. In other words, if a circuit generates the failure response 111, the failure can be diagnoed accurately if either either F_1 or F_2 are made improbable. Moreover, since faults F_3, F_4, and F_5 cause the response 001, any two of these faults should be made unlikely in order to improve diagnosis.

Index	Fault	Response	Signals
-	None	110	n/a
F_1	And-bridge$\{A, C\}$	111	A, C
F_2	C stuck-at-1	111	C, Vdd
F_3	And-bridge$\{A, B\}$	001	A, B
F_4	Or-bridge$\{A, B\}$	001	A, B
F_5	A stuck-at-1	001	A, Vdd
F_6	Dom-bridge$\{A, B\}$	100	A, B
F_7	B stuck-at-0	100	B, Gnd

Table 4: Failure responses and signals involved for an example set of faults.

The selection of $n - 1$ signal pairs from each partition can be described as a traditional minimum vertex cover problem, a well-known NP-hard problem. A graph $G(V, E)$ is created and for each bridge fault involving a unique pair of signal lines, a vertex is added to the graph to represent the signal pair. Several faults can involve the same set of signal lines, so one vertex may represent several bridge faults. If two faults share the same output response (*i.e.*, are equivalent faults), an edge is created between the two vertices. The solution to the minimum vertex cover problem is a minimal subset of vertices V' that removes all edges from G when V' is eliminated, thus ensuring that the remaining faults produce unique failure responses. In other words, the set of vertices V' is a minimal set of forbidden pairs.

For example, Figure 5 is a graph that represents the set of faults from Table 4. Since three faults involve signal lines A and B, there is one vertex $F_{3,4,6}$ that represents these three faults. An edge connects vertices F_1 and F_2 because those faults both create the response 111. Likewise, an edge connects $F_{3,4,6}$ to F_7 due to the response 100 and F_5 due to the response 001. A minimum vertex cover includes vertices F_2 and $F_{3,4,6}$. Therefore, the forbidden pairs (C, Vdd) and (A, B) will enable accurate diagnosis of the responses from Table 4. Since the identification of an optimal set of forbidden pairs is an NP-hard problem, this is included as part of the SAT formulation used for brick routing.

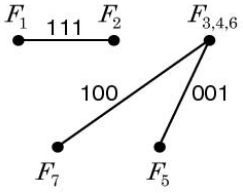


Figure 5: The minimum vertex cover of graph $G(V, E)$ represents the minimum set of forbidden pairs.

The minimal set of forbidden pairs to distinguish faulty responses of a logic brick is not necessarily the *best* set, however. The minimal set of forbidden pairs may be an unsatisfiable problem, even though there may exist a set of forbidden pairs that can distinguish the equivalent fault universe. Therefore, the set of equivalent faults are included as part of the SAT formulation used for brick routing. To simplify this discussion, a forbidden pair i is represented using a variable P_i that corresponds to fault F_i. The variable P_i represents all the routing rules required to make fault F_i unlikely. For each failing response, a disjunction of all combinations of $n - 1$ faults is created. For the example shown in Table 4, the additional constraint is $(P_1 \cup P_2) \cap (P_{3,4,6} \cup P_5) \cap (P_{3,4,6} \cup P_7)$. Although the solution to this SAT formulation will not necessarily be the minimum number of forbidden pairs (*i.e.*, the solution to the minimum vertex cover problem), it will produce a routing for the brick that completely distinguishes the given fault universe.

Table 5 shows the critical area (assuming a defect size of 0.25μm) of each equivalent fault in logic brick 1 from Figure 2. For one set of equivalent faults (response 10111011), there is no critical area and therefore is an unlikely response altogether. For response 10101010, only fault F_7 has critical area and therefore can be confidently diagnosed to be uniquely F_7. Although routing rules are included to reduce the likelihood of equivalent faults, faults F_{10} and F_{11} both have critical area and have the response 11111111. However, since F_{11} has nearly 25 times as much critical area as fault F_{10}, the response 11111111 can most

likely be diagnosed to be F_{11}.

Index	Fault	Response	CA (μm^2)
F_1	A stuck-at-1	10111011	0
F_2	Bdom-bridge$\{A, B\}$	10111011	0
F_3	Cdom-bridge$\{A, C\}$	10111011	0
F_4	OR-bridge$\{B, Z\}$	10111011	0
F_5	A stuck-at-0	10101010	0
F_6	B stuck-at-0	10101010	0
F_7	D stuck-at-1	10101010	47.5
F_8	Cdom-bridge$\{C, D\}$	10101010	0
F_9	C stuck-at-0	11111111	0
F_{10}	D stuck-at-0	11111111	2.6
F_{11}	Z stuck-at-1	11111111	63.5

Table 5: Failure responses and CA (defect radius = 0.25μm) for equivalent faults of logic brick 1 of Figure 2.

3.3 Routing Time Overhead

As shown in Table 6, for the three logic bricks designs examined, the additional routing rules increased the number of clauses by 10% on average. The run-time increased by 88% but never exceeded more than 38 seconds. Likewise, the memory usage increased by 18% but never exceeded 30 MB. Since the creation of a logic brick does not require a significant amount of run-time or memory, this overhead is concluded to be acceptable.

Brick type	No. of clauses	Run-time (sec)	Memory (MB)
Brick 1 (no DFT)	17,591	0.07	4.91
Brick 1 (DFT)	18,891	0.08	5.04
Brick 2 (no DFT)	33,821	1.03	6.14
Brick 2 (DFT)	36,875	2.93	6.79
Brick 3 (no DFT)	92,995	22.87	21.21
Brick 3 (DFT)	105,334	37.97	29.71

Table 6: DFT routing rules incur a small overhead in run-time and memory.

4 Summary

In past years, many researchers have proposed circuit fabrics that exhibit extreme regularity to improve manufacturability and predictability. Recently, a heterogeneous fabric composed of highly regular structures called logic bricks has been proposed. The regularity of these structures is imposed by converting layout restrictions to a SAT formulation, thus enabling a SAT-based design flow.

In this work, an automatic method has been described that uses a SAT-based design flow to improve the testability of logic bricks by modifying the physical design. By preventing certain wires from being routed in close proximity, bridges involving those wires are made unlikely. Based on this idea, difficult-to-detect feedback bridges are identified from the logic brick netlist and constraints are added to the SAT formulation to ensure these defects are unlikely. Also, a strategy to reduce the likelihood of equivalent faults is described. Given a set of equivalent faults, all but one is made unlikely, thus allowing one to confidently attribute a diagnosed failure response to the most probable fault. For a few example logic-brick designs, the critical area associated with difficult-to-detect feedback bridge faults can be reduced by 61.5% with a 7.6% increase in wire length and a 13.0% increase in vias for several brick designs. A 10% reduction in total critical area over the same logic brick designs is also achieved. Applying these DFT strategies to a logic brick library ensures that an entire design is more testable. This approach is cost-effective for improving front-end testability since a brick library typically consists of only 20-30 bricks.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. IEEE Press, 1990.

[2] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-signal VLSI Circuits*. Kluwer Academic Press, 2000.

[3] L. T. Wang, C. E. Stroud, and N. A. Touba, *System-on-chip Architectures: Nanometer Design for Testability*. Morgan Kaufmann, Nov. 2007.

[4] L. Scheffer, L. Lavagno, and G. Martin, *EDA for IC System Design, Verification, and Testing*. CRC Press, 2006.

[5] M. E. Levitt and J. A. Abraham, "Physical Design of Testable VLSI: Techniques and Experiments," *IEEE Journal of Solid-state Circuits*, vol. 25, pp. 474–481, April 1990.

[6] J. Sudbrock *et al.*, "Defect-oriented Test and Layout Generation for Standard-cell ASIC Designs," in *Euromicro Conference on Digital System Design*, pp. 79–82, Sept. 2005.

[7] J. Xu, R. Kundu, and E. J. Ferguson, "A Systematic DFT Procedure for Library Cells," in *IEEE VLSI Test Symposium*, pp. 460–466, April 1999.

[8] A. L. Jee and F. J. Ferguson, "CARAFE: An Inductive Fault Analysis Tool for CMOS VLSI Circuits," in *IEEE VLSI Test Symposium*, pp. 92–98, April 1993.

[9] T. Iizuka, M. Ikeda, K. Asada, "Timing-driven Cell Layout Decompaction for Yield Optimization by Critical Area Minimization," in *Design, Automation and Test in Europe*, pp. 1–6, March 2006.

[10] V. Kheterpal, *et al.*, "Design Methodology for IC Manufacturability Based on Regular Logic-Bricks," in *Design Automation Conference*, pp. 353–358, June 2005.

[11] B. Taylor and L. Pileggi, "Exact Combinatorial Optimization Methods for Physical Design of Regular Logic Bricks," in *Design Automation Conference*, pp. 344–349, June 2007.

[12] W. Maly and J. Deszcka, "Yield Estimation Model for VLSI Artwork Evaluation," *Electronics Letters*, pp. 226–227, March 1983.

[13] G. S. Tseitin, "On the Complexity of Derivation in Propositional Calculus," *Studies in Constructive Mathematics and Mathematical Logic*, pp. 115–125, 1970.

[14] F. A. Aloul *et al.*, "Generic ILP versus specialized 0-1 ILP: An Update," in *IEEE International Conference on Computer-aided Design*, pp. 450–457, Nov. 2002.

[15] N. Een and N. Sorensson, "MiniSat - a SAT Solver with Conflict-clause Minimization," in *International Conference on Theory and Applications of Satisfiability Testing*, 2005.